UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

INTERCONEXÃO DE PROCESSADORES E MEMÓRIAS PARA MULTIMICROPROCESSADORES

por

JAIRO ALBERTO PREZZI

Dissertação submetida como requisito parcial para obtenção do grau de Mestre em Ciência da Computação

Philippe Navaux

Prof. Philippe O. A. Navaux

Orientador

Porto Alegre, janeiro de 1981

AGRADECIMENTOS

Ao meu orientador, prof. Philippe O.A. Navaux, pela dedicação e empenho.

Ao prof. Francisco Bernardo Moser Filho, pela orientação inicial.

Ao Curso de Pós-Graduação em Ciência da Computação, à CAPES e ao CNPq, por propiciarem as condições para que este trabalho fosse realizado.

Aos colegas Antônio Carlos da Rocha Costa, Maria Helena Jaeger e Carla Maria Dal Sasso Freitas, pela revisão de trechos do texto. Ao Antônio Carlos agradeço a idéia do barramento multiplexado.

À Margarida e à Zita, pela revisão das referências bibliográficas.

À Maria Helena, pelo carinho e estímulo em todas as horas.

Para Maria Helena

e meus pais

PREFÁCIO

O assunto deste trabalho tem sua origem na disciplina Processadores oferecida pelo Curso de Pós-Graduação em Ciência da Computação da Universidade Federal do Rio Grande do Sul em 1979. Mais especificamente, de estudos realizados acerca de processamento paralelo e de um trabalho de simulação da interconexão de múltiplos processadores e módulos de memória.

A escolha do assunto, bem como a estratégia adotada na sua abordagem, resultou do propósito de se expandir a bagagem de conhecimento do CPGCC-UFRGS. Reconhecendo-se que o Curso procurava novos caminhos de pesquisa, optou-se por um trabalho que, possivelmente, viesse a se constituir em subsídio para uma nova frente de estudo. Neste sentido, e motivados pelo estudo principiado na disciplina Processadores, propusemo-nos a elaborar um trabalho de iniciação na área de multiprocessamento.

A idéia de se penetrar numa área de pesquisa relativamente nova, pelo menos a nível nacional, era reforçada pelo princípio que a Universidade deve assumir um papel de vanguarda em relação à indústria. Para isto, ela deve transcender a função de criação de recursos humanos, passando também a apontar caminhos para o desenvolvimento científico e tecnológico do país. Neste aspecto o assunto escolhido encontra confirmação nas palavras textuais do prof. Wolfgang Giloi em suas palestras no CPGCC-UFRGS:

"Multi-minicomputers (or microcomputers) will offer countries that do not have a large national computer industry

a chance to become less dependent of foreign mainframe manufacturers".

Assim sendo, o trabalho adquiriu um caráter de estudo das formas de interconexão entre processadores e módulos de memória. Esta dissertação não tem como objetivo um produto final mas, isto sim, investigar os aspectos construtivos de três destas estruturas de interconexão: um barramento multiplexado, múltiplos barramentos dedicados/memórias multiporta e uma matriz de barramentos cruzados.

Por outro lado, reconhecendo-se que este é apenas o passo inicial para um trabalho mais amplo, procurou-se trazer à tona outros aspectos relevantes de multiprocessadores. Com isto pretende-se obter uma melhor visão do multiprocessamento.

Além dos dois objetivos primordiais acima citados, este trabalho também visa situar os multiprocessadores dentro do contexto de processamento paralelo. Para tanto, diferentes arquiteturas capazes de prover paralelismo são identificadas.O aspecto de interconexão entre as múltiplas unidades de processamento é abordado.

O encaminhamento dado ao trabalho permite que ele seja subdividido em três áreas:

Uma das áreas (constituída pelos capítulos 2 e 4) aborda o problema de interconexão entre múltiplas unidades de processamento. São identificadas no capítulo 2 várias maneiras de realização de uma estrutura de interconexão. Como os multiprocessadores utilizam via de regra a memória como meio de interconexão, e considerando-se que o aspecto relevante

neste tipo de estrutura é a via de acesso à memória, são introduzidos no capítulo 4 os elementos a serem levados em conta no projeto de barramentos digitais. Este capítulo aborda os barramentos segundo o enfoque hierárquico dado no capítulo 2. São examinados os principais protocolos de arbitração e protocolos de comunicação empregados em barramentos digitais.

Uma segunda área (capítulo 3) aborda exclusivamente os aspectos operacionais de multiprocessadores. São examinadas as diferentes organizações de hardware e software possíveis de serem implementadas, além de funções de controle necessárias neste tipo de sistema. Particular atenção é dada às primitivas de sincronização que permitem o compartilhamento de recursos.

A terceira parte do trabalho constitui-se no projeto lógico de três alternativas de interconexão entre processadores e módulos de memória para um multimicroprocessador. Utilizando como subsídio as questões levantadas nos capítulos 2, 3 e 4, são descritos: um barramento multiplexado explicitamente no tempo (capítulo 5); múltiplos barramentos dedicados/memórias multiporta (capítulo 6); uma matriz de barramentos cruzados (capítulo 7).

O capítulo 8 resume as conclusões a que se chegou ao final do trabalho.

Porto Alegre, janeiro de 1981.

"Era uma vez em que os computadores eram muito simples."

Andrew S. Tanenbaum

SINOPSE

Este trabalho descreve o projeto lógico de três meios alternativos de interconexão entre processadores e módulos de memória para um sistema multimicroprocessador: barramento multiplexado, múltiplos barramentos dedicados/memórias multiporta e matriz de barramentos cruzados.

Com vistas ao projeto, são analisadas as características operacionais de multiprocessadores e identificadas algumas de suas funções de controle. O problema de interconexão em sistemas compostos de múltiplas unidades de processamento é abordado hierarquicamente. São mostradas as formas de se realizar a estrutura de interconexão, dando-se maior atenção aos barramentos digitais. São apresentados os protocolos de arbitração e protocolos de comunicação mais utilizados neste tipo de estrutura.

ABSTRACT

This work describes the logical project of three alternative ways of interconnecting processors and memory modules in a multimicroprocessor system: multiplexed bus, multiple dedicated buses/multi-port memories, and cross-bar matrix.

Aiming the project, the operational features of multiprocessors are analysed and some control functions identified. The interconnection problem in multiple processing units systems is hierarchichally approached, emphasizing digital buses. The arbitration protocols and communication protocols mostly used in this kind of structure are shown.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1 INTRODUÇÃO: DA MÁQUINA DE VON NEUMANN AO PROCESSAMENTO PARALELO

1 INTRODUÇÃO: DA MÁQUINA DE VON NEUMANN AO PROCESSAMENTO PARALELO

1.1 Organização de um computador de Von Neumann

A grande maioria dos computadores comerciais existentes na atualidade baseia-se nos princípios definidos por John Von Neumann em 1946[39]. Embora a máquina tenha recebido inúmeros aperfeiçoamentos desde então, sua organização básica não se alterou.

O computador de Von Neumann (figura 1), como ficou conhecido, é formado por uma unidade de armazenamento onde são mantidos os dados e as instruções, uma unidade aritmética e lógica para manipulação e transformação de dados, uma unidade de controle para interpretar as instruções e supervisio-

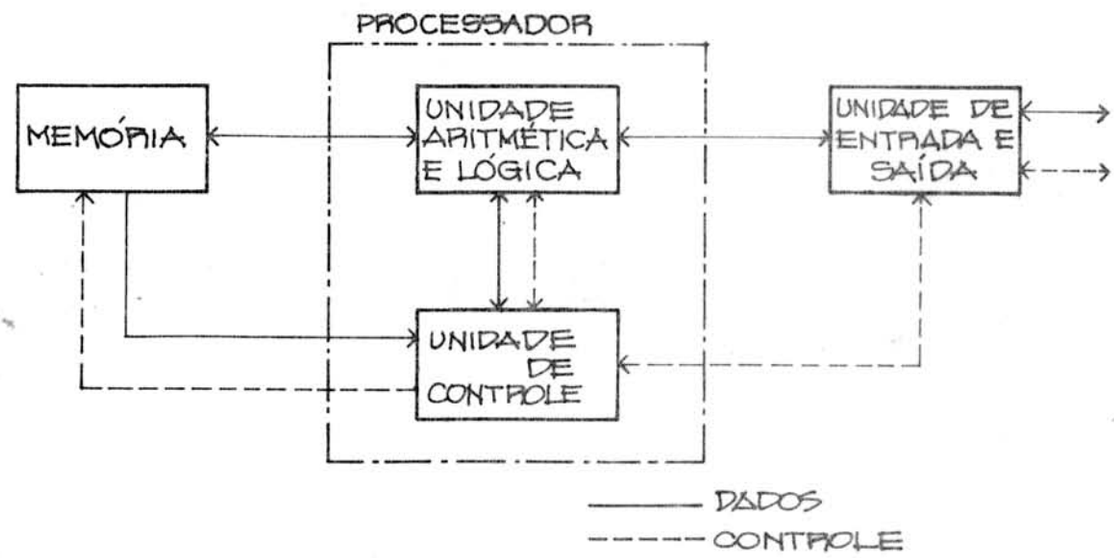


Figura 1 - Organização de um computador de Von Neumann

nar o sistema, e uma unidade de entrada e saída para comunicação com o meio exterior.

As modificações introduzidas no sentido de melhorar sua performance aconteceram na implementação das unidades, suas capacidades funcionais e meios de interconexão. Duas forças propulsoras patrocinaram estas modificações: o avanço tecnológico e a criação de algoritmos e técnicas mais eficientes[39].

Na unidade aritmética e lógica, o desenvolvimento deveu-se em grande parte aos avanços tecnológicos, e ao surgimento de técnicas mais velozes. Na unidade de controle a micro programação, além de tornar o computador flexível, simplificou o seu projeto e depuração. Na unidade de entrada e saída, os conceitos de interrupção e canal possibilitaram a utilização mais eficiente dos recursos do sistema. Na memória, o conceito de hierarquia fez com que o tamanho da mesma passasse a ser praticamente ilimitado. A memória virtual desvinculou os espaços de endereçamento lógico e físico. A necessidade de uma memória rápida e ao mesmo tempo de baixo custo foi solucionada pelo uso de memórias cache. A técnica de entrelaçamento amenizou a discrepância entre a velocidade da UCP e da memória primária.

Note-se que todas as melhorias mencionadas acima, exceto o conceito de canal de entrada e saída, não modificaram a organização da máquina nem a interrelação entre as unidades.

1.2 Efeito da integração em larga escala na arquitetura

O rápido avanço tecnológico na fabricação de circuitos integrados tem aumentado a performance dos circuitos lógicos em termos de velocidade, densidade, confiabilidade e consumo. As técnicas de integração em larga escala possibilitaram a fabricação em massa de pastilhas altamente complexas e memórias extensas, a um custo cada vez menor[21].

O progresso alcançado pela indústria de componentes teve um impacto direto sobre a organização tradicional dos sistema de computação[52]. Os aperfeiçoamentos citados na seção anterior surgiram em torno de um preceito básico: a unidade central de processamento e a memória deviam ser mantidas permanentemente ocupadas devido ao seu alto custo. O advento dos circuitos integrados em larga escala teve a força de mudar esta orientação[15]. O processador e a memória passaram a representar uma parcela menor do custo total enquanto o equipamento periférico e o software ascendiam. O conceito tradicional de se manter uma UCP poderosa constantemente ocupada gradualmente passou a dar lugar a várias unidades de processamento mais simples, e portanto mais baratas, operando em paralelo.

Uma determinada arquitetura somente é factível quando adaptada às restrições tecnológicas do momento. Com o custo não mais se constituindo numa barreira intransponível a idéia de paralelismo explícito encontrou condições para seu desenvolvimento. A integração em larga escala tornou o processamento paralelo uma hipótese viável e atrativa economicamente.

1.3 Formas de processamento paralelo

O termo processamento paralelo tem sido usado de uma maneira ampla para se referir à execução simultânea de várias operações em um computador digital. O paralelismo manifesta-se através de diferentes arquiteturas e filosofias de operação[19]. Por isto uma série de tentativas de classificação dos sistemas foram feitas. Uma das mais conhecidas, a classificação de Flynn[61], divide os sistemas em quatro categorias, baseando-se no número de fluxos de instruções e no número de fluxos de dados (figura 2):

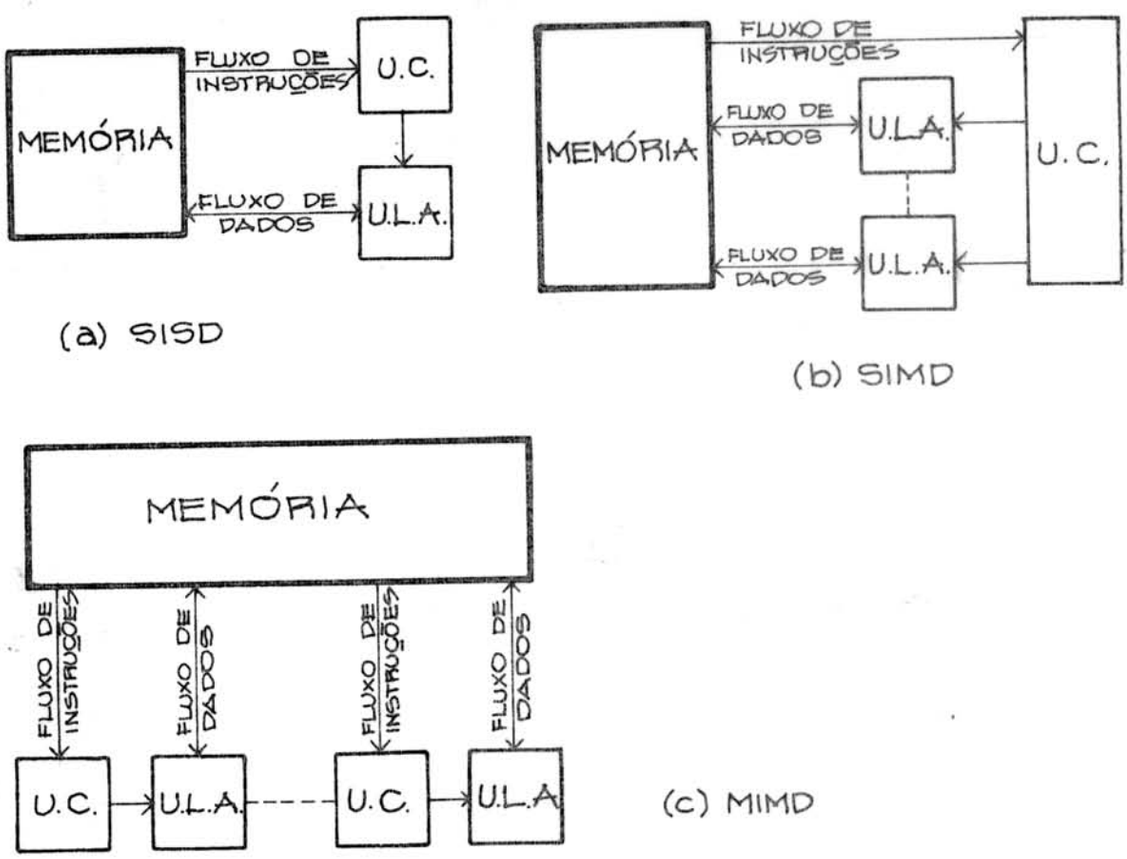


Figura 2 - Tipos de sistemas de computação

SISD - um fluxo de instruções e um fluxo de dados. Este é o monoprocessador tradicional.

SIMD - um fluxo de instruções e múltiplos fluxos de dados.

As unidades de processamento executam a mesma operação sobre dados distintos. Nesta categoria encontram-se os processadores de arranjos, associativos e "pipeline".

MISD - múltiplos fluxos de instruções e um fluxo de dados.

Este conceito de sistema não é realizável fisicamente. Alguns autores enquadram os processadores "pipeline" nesta categoria.

MIMD - múltiplos fluxos de instruções e múltiplos fluxos de dados.

Cada unidade de processamento executa operações distintas sobre dados distintos. Nesta categoria encontram-se os multiprocessadores e sistemas de múltiplos computadores independentes.

Searle e Freberg[50] observam que os sistemas multicomputadores (sistemas que contêm mais de um processador interligados fisicamente) são organizados em função de diferentes conceitos e objetivos:

Sistemas redundantes - utilizam múltiplas unidades de processamento em paralelo objetivando a confiabilidade.

Processadores "pipeline" - permitem a sobreposição de operações seqüenciais a fim de tornar o processamento mais veloz.

Processadores paralelos - compostos por múltiplas unidades de processamento as quais operam simultaneamente so-

bre múltiplos fluxos de dados (processadores de arranjos e processadores de vetores).

Redes de computadores - contêm computadores completos, independentes, descentralizados e ligados entre si através de linhas de comunicação. Objetivam o compartilhamento dos computadores existentes entre todos os usuários da rede.

Sistemas de múltiplos processadores - as várias unidades de processamento operam cooperativamente formando um único sistema integrado. Os sistemas com múltiplos processadores podem ser subdivididos em sistemas distribuídos e multiprocessadores.

Sistemas distribuídos - cada uma das unidades de processamento é dotada da capacidade de executar um número limitado de funções. Os sistemas distribuídos são úteis onde um conjunto de processadores agindo cooperativamente pode resolver um problema através da sua decomposição em problemas menores aos quais cada processador é especialmente adaptado.

Multiprocessadores - contêm múltiplos processadores com capacidades mais ou menos idênticas, sendo todos eles controlados por um único sistema operacional, o qual é capaz de alocar dinamicamente as tarefas.

1.4 Multiprocessadores

As novas arquiteturas introduzidas como alternativa ao processador tradicional visam, primordialmente, aumentar a confiabilidade, a performance e a modularidade. Dentre todas, apenas duas são potencialmente capazes de satisfazer os

três objetivos simultaneamente: os sistemas distribuídos e multiprocessadores simétricos[22].

Os multiprocessadores provêem uma alta confiabilidade pela existência de múltiplas unidades de processamento capazes de executar as mesmas tarefas e pela capacidade de reconfiguração do sistema.

A performance é aumentada em relação a um monoprocessador por vários motivos: (1) capacidade de execução de tarefas independentes em paralelo; (2) equilíbrio da carga de trabalho entre os processadores (em sistemas distribuídos isto nem sempre é possível); (3) flexibilidade para suportar demandas variadas; (4) utilização eficiente dos recursos do sistema.

A modularidade de sistemas multiprocessados, assim como a performance, está intimamente ligada à forma como são interconectadas suas unidades funcionais.

2 ESTRUTURAS DE INTERCONEXÃO

2 ESTRUTURAS DE INTERCONEXÃO

2.1 Introdução

Uma estrutura de interconexão digital pode ser definida como sendo qualquer subsistema usado para interligar dispositivos digitais. Uma classificação assim tão ampla abrange os mais variados tipos de estruturas, desde uma linha dedicada ligando um dispositivo periférico e uma interface até um computador dedicado a direcionar mensagens em uma rede. A abrangência da definição é aqui limitada às estruturas destinadas à conexão entre as unidades de um sistema multicomputador. Embora o âmbito da definição diminua consideravalmente, ainda assim ela engloba pelo menos três tipos de sistemas[50]: redes de computadores geograficamente distribuídos, sistemas localmente distribuídos e multiprocessadores.

Todos estes sistemas apresentam dois pontos em comum: (1) são constituídos por mais de um elemento processador interligados fisicamente, e (2) distribuem o processamento entre estas unidades criando a necessidade de intercomunicação entre elas.

A figura 3 mostra o modelo geral de uma arquitetura com múltiplos processadores[59]. Tal sistema é visto como um conjunto de processos que se comunicam através de uma hierarquia de protocolos de software e hardware.

O protocolo de comunicação a nível de usuário refere-se aos serviços por este requisitados, os quais podem necessitar a utilização de mais de um processador. O estabelecimento, manutenção e término da comunicação entre processos cria

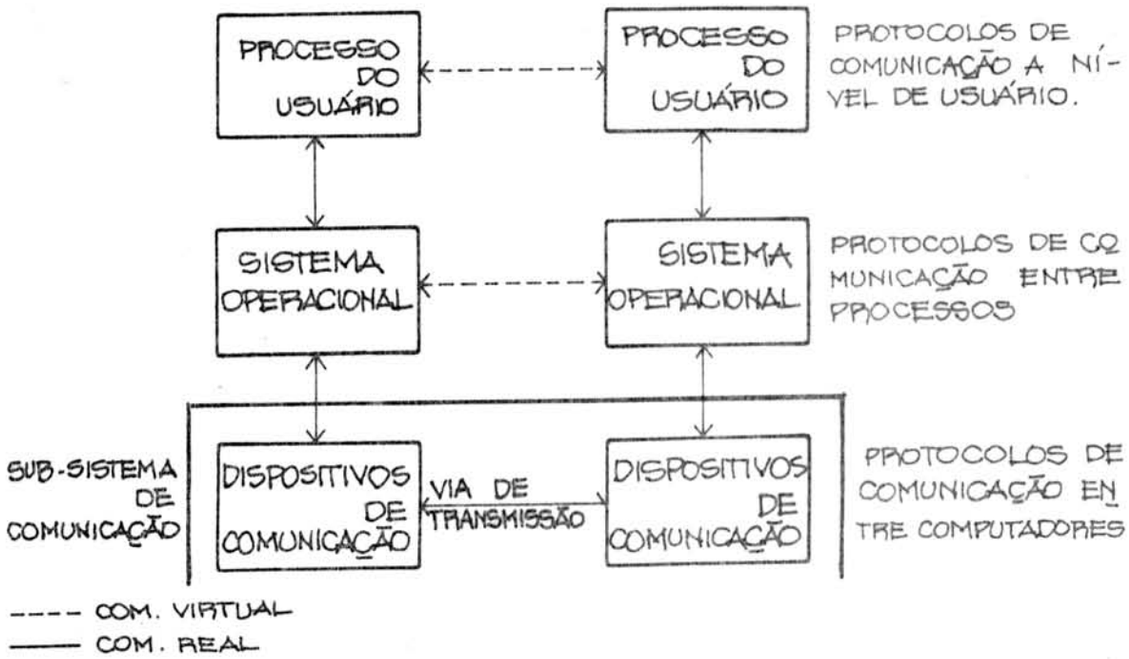


Figura 3 - Modelo de comunicação em um sistema multicomputador

dos no atendimento de diretivas do usuário é geralmente tarefa do sistema operacional, responsável pela definição do protocolo de comunicação entre processos. A comunicação entre processos rodando em processadores diferentes é virtual no sentido que as informações são passadas entre eles indiretamente através de um meio de comunicação. Isto equivale a dizer que toda comunicação entre processos executados em processadores diferentes é realizada através de alguma espécie de ligação física entre estes últimos. Essa interconexão física é o nível mais baixo da hierarquia, e constitui o subsistema de comunicação.

2.2 Projeto do subsistema de comunicação

O projeto do subsistema de comunicação envolve uma série de considerações prévias pois é natural que as decisões tomadas em níveis superiores influenciem os caminhos tomados nos níveis mais baixos.

Visando possibilitar o desenvolvimento metódico de arquiteturas com múltiplos processadores propõe-se neste capítulo uma abordagem sistemática e hierárquica ao projeto de estrutura de interconexão. Esta abordagem baseia-se nas áreas de decisão de projeto apontadas por Thurber[59] e na classificação de Anderson e Jensen[2] com respeito às estruturas de interconexão.

A abordagem proposta divide-se em dois níveis: nível do sistema e nível do subsistema de comunicação. As decisões tomadas no nível do sistema estão vinculadas à arquitetura do mesmo. A partir dos objetivos e características pretendidas definem-se a topologia de interconexão, o modo de transferência da informação, a localização do controle e o uso de vias partilhadas ou dedicadas. É necessária uma decisão quanto à forma e envio de mensagens desde a fonte até o destino: indiretamente através de elementos direcionadores (chaves), ou diretamente através de vias dedicadas ou partilhadas (caminhos). A escolha de uma destas alternativas determinará certas características fundamentais da arquitetura.

As considerações feitas no nível do sistema abstraem detalhes específicos de implementação pertinentes ao próximo nível. Ao final das decisões tomadas no nível do sistema tem-se definida qual a estrutura de interconexão a ser imple-

mentada mas não se encontram ainda definidas questões tais como: mecanismos de arbitração de vias partilhadas, comunicação síncrona ou assíncrona, tamanho de mensagens, etc..

A partir do momento em que está estabelecida a arquitetura, em função das necessidades e requisitos impostos na fase inicial, pode-se iniciar o projeto da estrutura de interconexão propriamente dita. Neste ponto são definidas as técnicas de comunicação, ou seja, a maneira como ocorre a transferência de informação entre as unidades. As questões neste nível estão vinculadas à via física por onde transitam os dados. O caminho a ser seguido é ditado pela escolha feita anteriormente quanto à comunicação indireta ou direta. No primeiro caso a estrutura é realizada através do chaveamento de mensagens/pacotes ou chaveamento de circuitos. No segundo caso encontram-se barramentos, linhas dedicadas e memórias compartilhadas.

2.3 Nível do sistema

A definição de um sistema multicomputador é feita, no nível do sistema, seguindo-se o modelo de Anderson e Jensen[2]. Este baseia-se única e exclusivamente no aspecto de comunicação entre as unidades de processamento, o que corresponde ao nível mais baixo da figura 3.

2.3.1 Definição das entidades

São definidas três entidades, as quais combinadas caracterizam topologicamente a estrutura de interconexão: mensagens, caminhos e chaves.

Mensagem é a unidade de informação transferida entre processos que são executados em unidades processadoras diferentes. Uma mensagem pode conter uma variedade de tipos de informção como, por exemplo, blocos de dados, informação de controle ou estado. Nenhuma distinção é feita quanto a este aspecto no presente nível. Igualmente, não se distingue nenhum formato específico de mensagem, embora ele difira em função do tipo de via utilizada. Este tipo de abstração pode ser feito tendo-se em mente que estas questões são pertinentes ao nível inferior (nível do subsistema) e lá serão definidas.

Caminho é o meio por onde transitam as mensagens. Este não tem um sentido físico mas sim virtual. Um caminho pode tomar a forma de um barramento, ou uma linha telefônica, ou uma memória partilhada entre os elementos processadores.

Chave é o elemento capaz de tomar uma decisão (se for necessário) quanto ao direcionamento de uma mensagem através de rotas (caminhos) alternativas desde a fonte até o destino.

Da mesma forma que os aspectos particulares de implementação de mensagens são abstraídos do nível do sistema, o mesmo procedimento é adotado com relação às chaves e caminhos. A forma real que estas entidades assumirão é uma questão resolvida ao nível do subsistema.

2.3.2 Tipos de sistemas

Definidos os elementos constituintes de um sistema, este pode ser enquadrado dentro de uma das classes resultantes de quatro níveis de decisão (figura 4).

O algoritmo de decisão é representado por uma árvore onde a raiz é uma interconexão genérica e as folhas são as implementações topológicas. Entre a raiz e as folhas estão localizadas as decisões quanto ao trânsito de mensagens entre as unidades.

Efetuando-se a escolha entre as alternativas possíveis, na seqüência especificada, chega-se até as folhas da árvore. São identificadas assim dez estruturas de interconexão diferentes.

O primeiro nível de decisão refere-se à estratégia de transferência. Uma mensagem pode ser enviada de um processador a outro diretamente através de um caminho, ou indiretamente através de uma ou mais chaves. Esta última opção implica a passagem das mensagens por elementos que alteram seu conteúdo ou escolhem um dentre vários caminhos alternativos.

O segundo nível de decisão refere-se exclusivamente a mensagens enviadas de forma indireta. A pergunta formulada é: onde está localizada a chave? Se ela pode ser identificada em apenas um ponto do sistema, então o controle é do tipo centralizado. Se o chaveamento é executado por mais de uma chave localizadas em pontos diferentes, então o controle de transferência é descentralizado.

No terceiro nível é feita a escolha entre vias dedicadas ou partilhadas, englobando qualquer decisão tomada an-

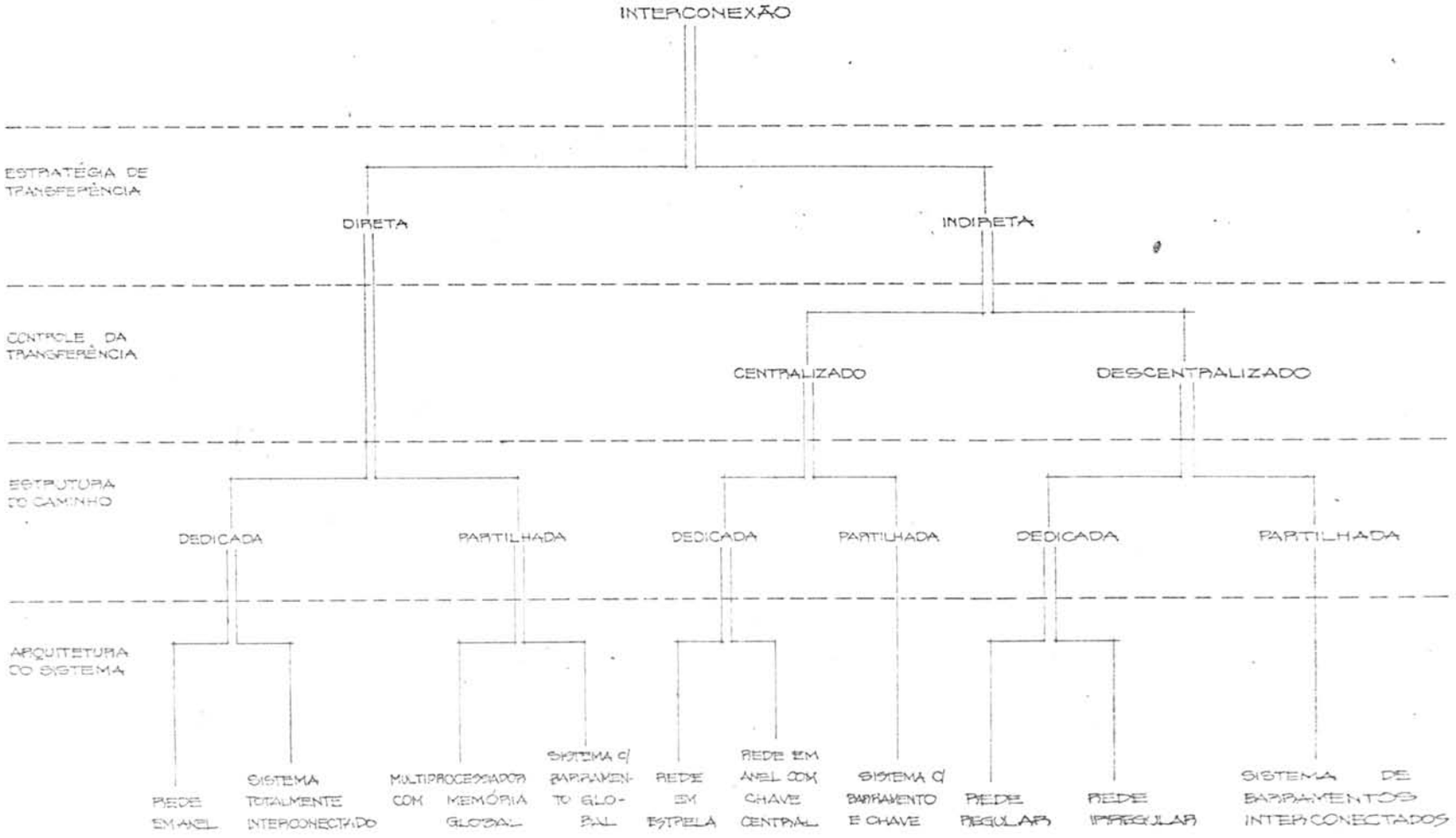


Figura 4 - Tipos de sistemas multicomputadores

teriormente. Uma via é partilhada quando mais de duas unidades têm acesso a ela. Isto exclui a possibilidade de uma via bidirecional entre dois processadores ser partilhada.

O quarto e último nível caracteriza topologicamente o sistema. Neste ponto é estabelecida a forma de realização da estrutura.

Os dois últimos níveis de decisão estão relacionados à implementação do subsistema de comunicação. Porém, a simplicidade dos elementos em jogo (mensagens, caminhos, chaves) não permite que se avance muito no terreno da implementação. Na realidade, as decisões tomadas nestes dois níveis esclarecem detalhes e características no âmbito do sistema, ao mesmo tempo que estabelecem apenas a direção a ser tomada na implementação do subsistema de comunicação.

2.3.3 Características dos sistemas

Os sistemas identificados da forma descrita na seção anterior, tendo por base a estrutura de interconexão, possuem características diferenciadas. Ao nível do sistema, pode-se de imediato inferir certas propriedades que lhes são atribuídas pela própria estrutura. Estas características relevantes são a modularidade, a tolerância a falhas, a performance, a complexidade lógica e a flexibilidade.

A modularidade pode ser medida levando-se em conta dois aspectos[2]: custo e localização. Um sistema é modular quanto ao custo se a inclusão de mais um elemento implicar o acréscimo ao custo total de uma parcela igual (ou aproximadamen

te igual) ao custo do elemento isolado. No aspecto localização, o termo modularidade diz respeito ao grau de liberdade possível ao se adicionar um novo elemento em qualquer ponto do sistema a fim de se conseguir uma melhor performance.

A maior ou menor tolerância a falhas é medida pelo efeito de uma falha sobre o sistema e o custo do isolamento da falha e da reconfiguração do sistema.

No segundo e terceiro níveis de decisão determina-se a tolerância a falhas do sistema. Aqueles que dependem de um caminho partilhado ou de chaves centralizadas são altamente suscetíveis a defeitos nesses elementos.

A performance não pode ser estimada quantitativamente. Este tipo de medida está associado à implementação e à tecnologia usada, portanto fora do nível do sistema. As observações restringem-se à identificação de possíveis pontos de saturação e gargalos. Da mesma forma que a tolerância a falhas, a performance é em parte determinada no segundo e terceiro níveis de decisão. Ela é limitada por caminhos partilhados e chaves centralizadas.

A complexidade lógica refere-se ao número e natureza das decisões a serem feitas no estabelecimento de um caminho de comunicação, tanto nos pontos extremos como nos elos intermediários (chaves). Quanto maior o número de possibilidades mais complexa a decisão. Assim, após a primeira decisão estratégica (transmissão direta ou indireta), já se tem uma noção de complexidade. Os sistemas que utilizam estratégia direta apresentam uma baixa complexidade lógica em virtude da ausência de elementos especializados em dirigir mensagens. Todos os sistemas que adotam a estratégia indireta apresentam um

certo grau de complexidade lógica. Em geral, aqueles sistemas que descentralizam as chaves direcionadoras de mensagens são mais complexos pois, todos os elementos processadores devem ter capacidade de chaveamento. Portanto, os elementos processadores necessitam conhecer a topologia do sistema.

A flexibilidade diz respeito aos meios alternativos de se interconectar mais um elemento à rede. Ela se distingue da modularidade no sentido de ser uma medida do número de alternativas possíveis, enquanto esta última é uma medida de custo. Portanto, só faz sentido analisar a flexibilidade de sistemas que utilizam estratégia indireta de comunicação pois os diretos não apresentam alternativas. Nos sistemas que empregam estratégia direta de comunicação só há um meio de se estabelecer uma nova interconexão e esta é definida pelo tipo de caminho implementado.

2.4 Nível do subsistema de comunicação

2.4.1 Alternativas de implementação

A próxima etapa no projeto de uma estrutura de interconexão compreende a decisão de como implementar os caminhos e chaves considerados ao nível do sistema. Os caminhos são vias de comunicação direta entre unidades. Já as chaves agem sobre as mensagens escolhendo caminhos alternativos para as mesmas. As opções estão mostradas na figura 5.

Uma chave é realizada fisicamente através da comutação de pacotes e mensagens ou através da comutação de circui

tos. No primeiro caso os responsáveis pela comutação são computadores. No segundo caso as chaves de circuitos são implementadas através de matrizes de chaveamento (cross-bar) ou multiplexadores. Um caminho é implementado através de linhas dedicadas, barramentos partilhados, ou memórias acessíveis por todos os elementos processadores.

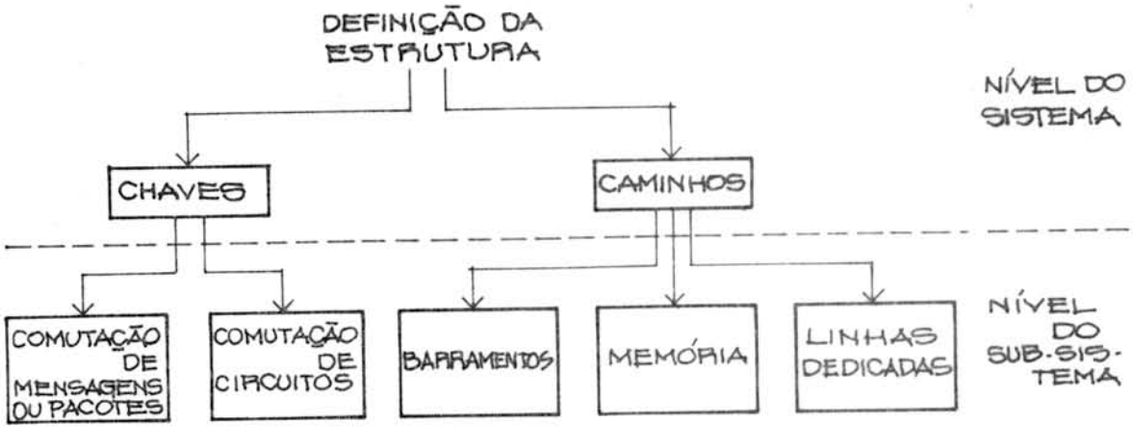


Figura 5 - Alternativas de implementação de uma estrutura de interconexão

2.4.2 Influência da distância

Topologicamente falando, foram apresentadas nas seções anteriores várias maneiras de se estabelecer uma conexão entre elementos processadores de um sistema multicomputador. A escolha de uma entre várias estruturas resulta de uma série de decisões tomadas ao nível do sistema, em função das características desejadas.

Um fato ainda não mencionado até aqui, o qual irá influenciar as decisões a serem tomadas com vistas à implementação é a distância entre os elementos processadores.

Distinguem-se três casos na comunicação entre unidades de um sistema[24]:

1º) Comunicação interna. Efetuada dentro de uma mesma unidade, onde as distâncias são muito pequenas (tipicamente menos de um metro).

2º) Comunicação local. Efetuada entre unidades a pequena e média distância, porém eletricamente distantes (dezenas ou centenas de metros).

3º) Comunicação a longa distância. Efetuada entre unidades afastadas de uma distância da ordem de quilômetros.

O primeiro caso não apresenta problemas para a transmissão de sinais elétricos, o que ameniza os requisitos de comunicação. Uma única unidade controla a operação dos subsistemas e um único relógio pode ser usado para sincronizar a transferência de informação entre eles. O projeto da estrutura de interconexão fica restrito ao nível de transferência entre registradores.

No segundo caso a distância envolvida geralmente impede o uso de um relógio sincronizador devido a problemas de escorregamento e atraso dos sinais. O projeto da estrutura de interconexão exige mecanismos de sincronização mais complexos, os quais serão abordados no capítulo 4.

No terceiro caso a separação física é tão grande, que a sinalização digital (usada nos casos anteriores) é impraticável, devido à distorção introduzida. A quantidade de distorção aumenta com a distância e a freqüência de transmis-

são. Além disso, as linhas telefônicas, normalmente utilizadas, não se prestam a transmissões digitais. De maneira que é necessário recorrer a técnicas de modulação. Estas técnicas fazem parte de uma área específica de estudo, a transmissão de dados, a qual não será investigada neste trabalho.

Feitas estas observações conclui-se que todos os sistemas que utilizam um barramento ou memória como meio de interconexão restringem-se a aplicações nas quais os elementos encontram-se próximos uns dos outros. Neste grupo incluem-se os sistemas ligados por um barramento global com ou sem chave centralizada, os sistemas de barramentos interconectados e os multiprocessadores. Os sistemas completamente interconectados também são enquadrados neste caso pois o alto custo das linhas inviabiliza-os em redes geograficamente dispersas. Se o sistema é geograficamente disperso e as mensagens são enviadas diretamente, então deve-se usar linhas dedicadas.

2.4.3 Comutação de mensagens e pacotes

Na comutação de mensagens, o subsistema de comunicação pode ser visto como um conjunto de linhas físicas interconectadas por chaves. Para enviar uma mensagem, o subsistema estabelece um canal, transmite-a e desliga-se do mesmo. O canal é o caminho lógico através do qual se estabelece a conexão. Um canal é criado pela conexão física de linhas através de chaves. Estas, por sua vez, são processadores dedicados exclusivamente à tarefa de comunicação.

Na comutação de pacotes, as mensagens são fracio-

nadas e cada pacote resultante enviado ao mesmo destino individualmente, podendo cada um seguir por caminhos diferentes. As duas técnicas requerem estruturas de interconexão equivalentes, pelo menos no que diz respeito ao hardware.

O subsistema de comunicação para comutação de mensagens ou pacotes pode ser subdividido em dois níveis (figura 6). O nível inferior (Protocolos de Linha) relaciona-se com o

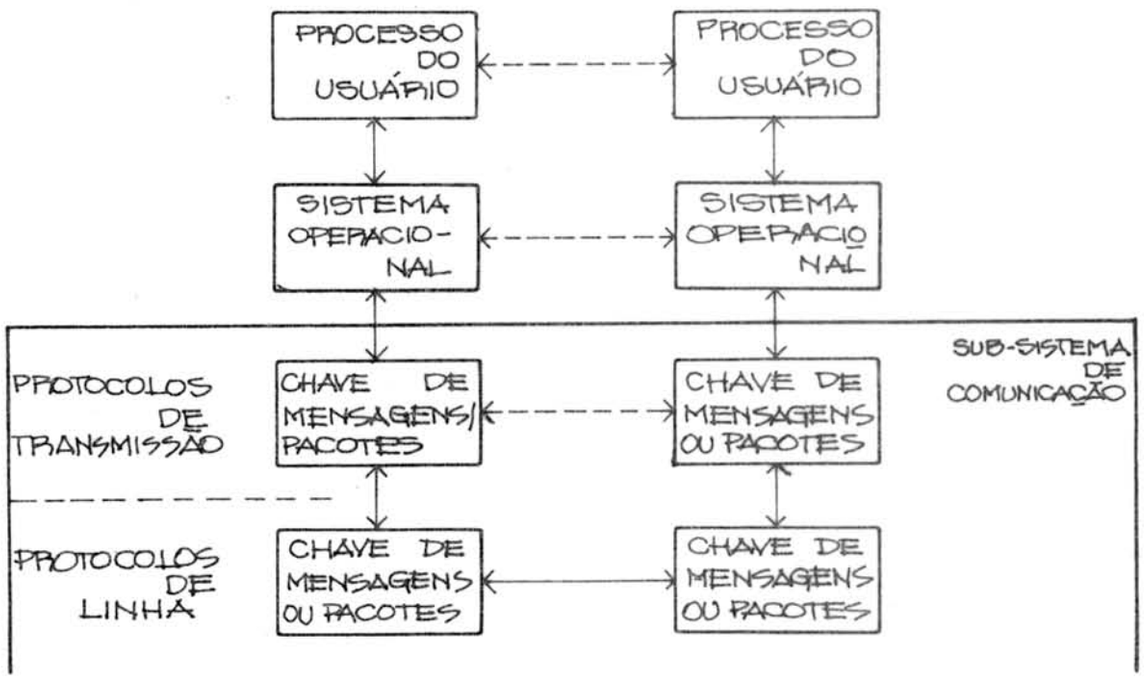


Figura 6 - Níveis de protocolo em comutação de mensagens ou pacotes

estabelecimento e o controle do circuito físico entre as chaves. Estão ligadas a este nível questões tais como: protocolos de transferência(handshaking), sincronização, detecção e correção de erros, modulação.

O nível superior (Protocolos de Transmissão) rela-

ciona-se com o estabelecimento de um canal virtual entre a fonte e o destino. Estão ligadas a este nível questões tais como: escolha de rotas (direcionamento), montagem e desmontagem de mensagens, troca de informações de reconhecimento ou de erro nas mensagens e resolução de impasses de transmissão.

As questões relacionadas à implementação de redes de comutação de mensagens e pacotes não serão estudadas em maior detalhe por se desviarem do objetivo principal deste trabalho. Foram consideradas até o presente momento no intuito de inserir as estruturas de interconexão para multiprocessadores no contexto do problema de comunicação entre processadores.

2.4.4 Comutação de circuitos

Na comutação de circuitos o subsistema de comunicação recebe uma mensagem a ser enviada, estabelece um circuito e o mantém até o final da transmissão. As chaves de comutação podem ser multiplexadores ou matrizes de chaveamento. Uma terceira opção são as redes de interconexão[3,6,35,58,60] as quais oferecem a vantagem de necessitar um menor número de chaves que uma matriz de chaveamento.

Analogamente à comutação de mensagens e pacotes, o subsistema de comunicação para comutação de circuitos subdivide-se em dois níveis de protocolo (figura 7).

As questões relacionadas ao nível inferior (Protocolos de Comunicação da Linha) são as mesmas do nível correspondente na comutação de mensagens e pacotes.

Os algoritmos de controle da chave constituem o

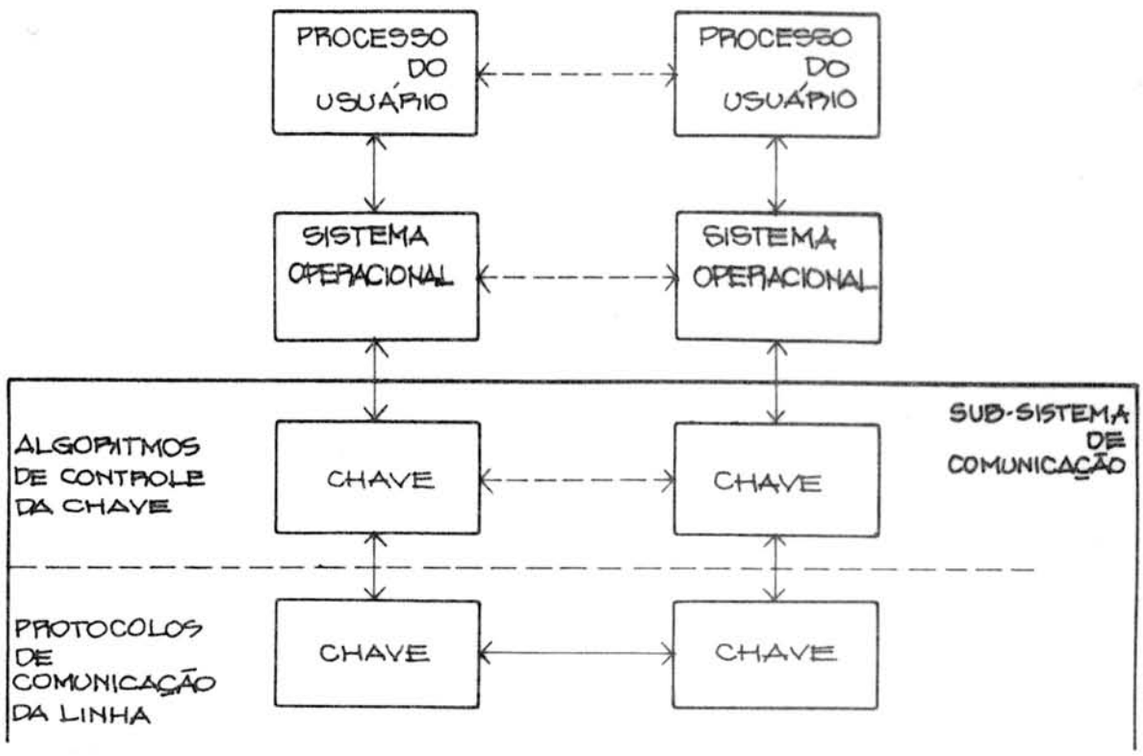


Figura 7 - Níveis de protocolo em comutação de circuitos

nível superior. A diferença fundamental entre este nível e o seu correspondente em comutação de mensagens e pacotes é a forma como são controladas as chaves. No caso em questão todo o controle é implementado em hardware, enquanto que na comutação de mensagens e pacotes são usados processadores para este fim.

A comutação de circuitos, por ser mais veloz, é usada quando a velocidade de chaveamento é importante. Por outro lado, a comutação de mensagens e pacotes possibilita algoritmos de chaveamento mais complexos e uma utilização mais eficiente das linhas.

A comutação de circuitos não será considerada em maior profundidade pelas mesmas razões mencionadas na seção an

terior. A matriz de chaveamento, encarada como um caso particular de barramento, é analisada nos capítulos 4 e 7.

2.4.5 Linhas dedicadas, barramentos e memórias

A alternativa às chaves é a transmissão direta da informação entre o remetente e o destinatário, sem a intervenção de nenhum elemento inteligente capaz de direcionar as mensagens. A via física por onde transitam os dados pode ser uma linha dedicada, um barramento ou uma memória global.

Neste tipo de implementação, uma das questões fundamentais é a alocação do caminho para transmissão. Com linhas dedicadas entre dois dispositivos este problema não existe, pois não ocorre disputa pelo mesmo. A comunicação pode ser efetuada a qualquer instante, uma vez que o caminho está sempre disponível. Uma única situação de interferência pode ocorrer quando a linha é bidirecional e ambos os extremos iniciam uma transmissão simultaneamente. Neste caso é necessário dotar os dispositivos de mecanismos capazes de detectar a colisão[64].

Quando o caminho é implementado com uma memória global são necessários mecanismos de sincronização de alto nível (como semáforos) e de baixo nível (operações de leitura-modificação-escrita com acessos indivisíveis) para efetuar a arbitração do uso da memória. Este problema é analisado no capítulo 3, na seção referente à sincronização de multiprocessadores.

Quando o caminho é implementado sob a forma de um barramento partilhado, um árbitro ou controlador de barramen-

to é responsável pela exclusão mútua. A figura 8 ilustra a hierarquia do subsistema de comunicação implementado com barramentos. O nível inferior (Protocolos de Comunicação) está relacionado com a transferência de informação entre as unidades. Fazem parte deste nível questões tais como: técnica de comunicação (síncrona ou assíncrona), técnicas de transferência (entrelaçamento de sinais assíncronos), largura de barramento, conversão série-paralelo. O nível imediatamente superior (Protocolos de Arbitração) relaciona-se aos mecanismos de alocação de barramento.

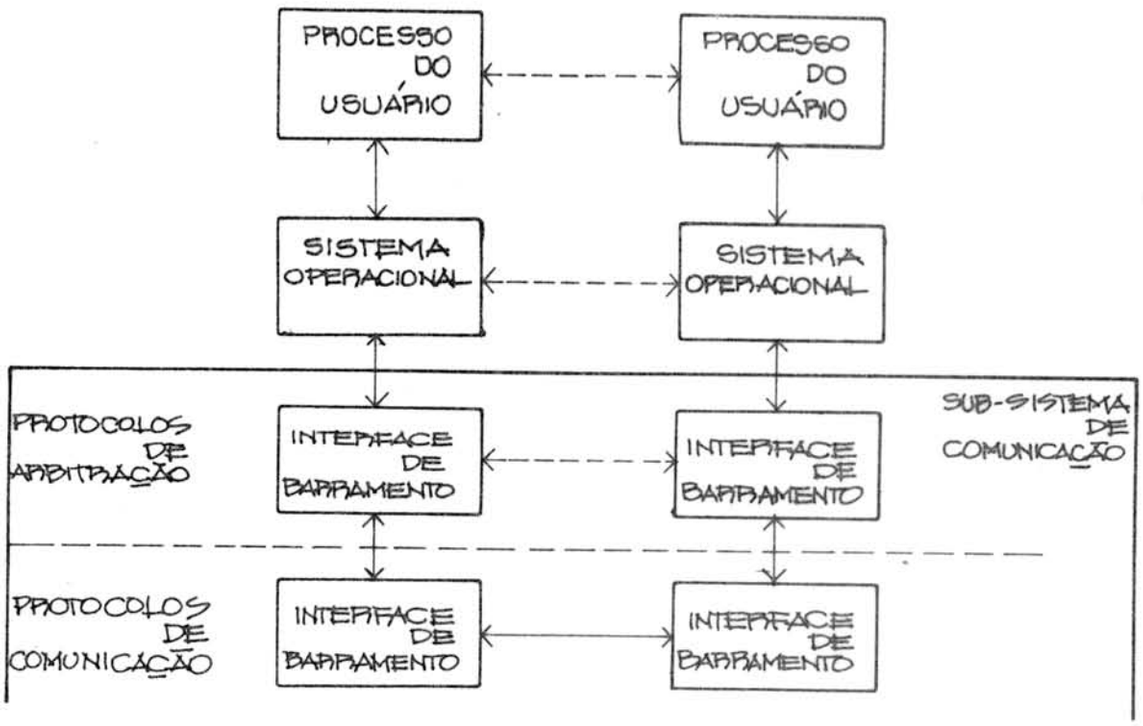


Figura 8 - Níveis de protocolo em barramentos

Barramentos e memórias globais são estruturas de interconexão largamente utilizadas em multiprocessadores[20,44,56]. Existem também sistemas híbridos[8,14] nos quais a comunica-

ção é efetuada através de barramentos e linhas dedicadas. Por esta razão os níveis de protocolo dos barramentos serão aprofundados no capítulo 4. A comunicação através de uma memória global é considerada em maior detalhe no capítulo 3. A implementação de mecanismos de acesso indivisível é abordada nos capítulos 5,6 e 7.

3 MULTIPROCESSADORES

3 MULTIPROCESSADORES

3.1 Caracterização de um multiprocessador

O termo multiprocessador refere-se a um sistema formado por múltiplas unidades de processamento atuando cooperativamente e supervisionadas por um único sistema operacional[18,24]. A atuação cooperativa subentende a interação entre as várias unidades de processamento, objetivando a resolução de um problema computacional.

A definição acima não exprime completamente as características operacionais de um multiprocessador. O que realmente o caracteriza entre outros sistemas é a filosofia de compartilhamento dos recursos disponíveis tanto de hardware como de software.

Os recursos de hardware partilhados pelos processadores são a memória e os periféricos (figura 9). As definições mais pragmáticas afirmam que a memória de um multiprocessador é uma só, global ao sistema. Isto significa que ela deve ser acessível por todos os processadores. Da mesma forma, os dispositivos de entrada e saída não são privativos de nenhum processador em especial. Qualquer processador pode assumir a função de controle de um periférico. Na prática isto nem sempre ocorre. Na verdade é mais freqüente acontecer de os processadores possuírem áreas de memória privativas onde são armazenadas as estruturas de dados e rotinas utilizadas mais freqüentemente, ou memórias cache individuais[46]. Analogamente, um processador pode ter periféricos para seu uso exclusivo. Entretanto, o conceito de partilhamento continua

válido pois apenas uma parte da memória é privativa. Os dispositivos de entrada e saída, apesar de não serem controlados diretamente, são partilhados no sentido que eles estão disponíveis para prestar serviços requisitados por qualquer processador do sistema.

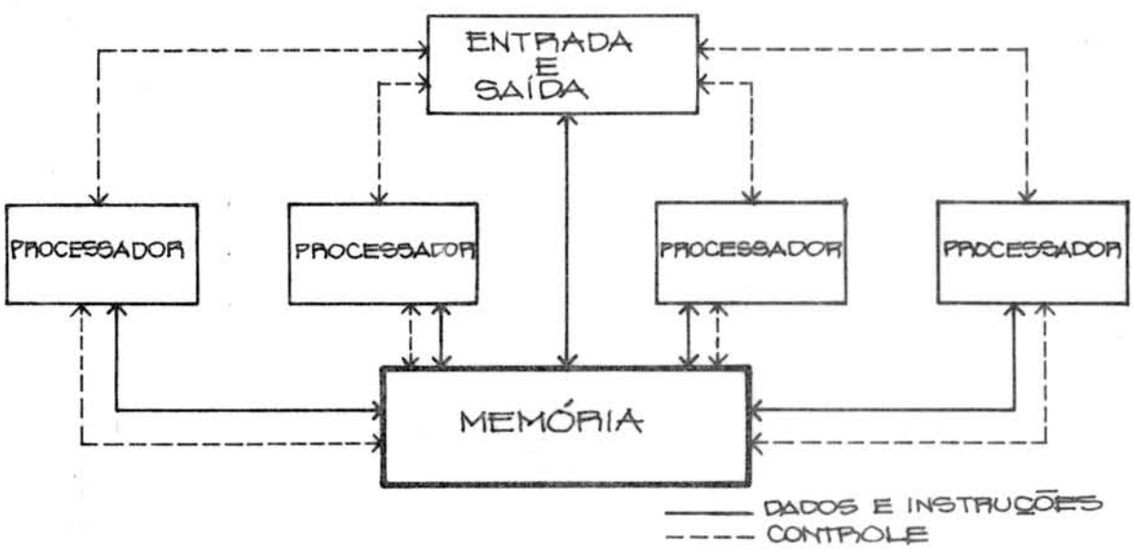


Figura 9 - Organização básica de um multiprocessador

O compartilhamento direto da memória central e dos recursos de entrada e saída caracteriza os multiprocessadores como sistemas "fortemente acoplados".

As adaptações visam a melhoria da performance, em função das características específicas desejadas em cada caso particular. No que diz respeito ao software, o sistema permite o partilhamento real de programas e estruturas de dados na memória global. Uma única cópia de um programa na memória global é executada em paralelo por mais de uma unidade de processa-

mento, desde que cada uma delas possua uma cópia da estrutura de dados manipulada. A própria estrutura de dados pode ser partilhada de uma maneira análoga à multiprogramação, porém mais sofisticada, devido ao paralelismo real[29] proporcionado por múltiplas unidades de processamento. Entretanto, o compartilhamento traz consigo um problema: devem ser implementados mecanismos capazes de garantir a utilização ordenada tanto dos recursos de hardware como de software.

Uma outra particularidade associada aos multiprocessadores é o conceito de interação em todos os níveis entre as unidades processadoras que constituem o sistema. Em redes de computadores geograficamente distribuídos e em sistemas acoplados indiretamente a interação acontece num nível alto. A unidade básica de interação é o arquivo. Em sistemas distribuídos a interação é um pouco maior, ao nível de tarefas[38]. Já em multiprocessadores a interação entre unidades acontece também ao nível de hardware (interrupção) e software. Nestes sistemas a interação deve ser possível com arquivos, estruturas de dados, e até mesmo dados elementares. Do ponto de vista do controle, partes de um mesmo programa podem ser executadas em unidades diferentes, o que exige a interação ao nível de programas, tarefas e instruções individuais.

Portanto, três são os requisitos básicos que caracterizam um sistema multiprocessador:

1º) a existência de mais de uma unidade central de processamento;

2º) o compartilhamento dos recursos de hardware (memória e periféricos);

3º) a supervisão do sistema por um único sistema

operacional capaz de permitir a interação entre as unidades de processamento até o nível mais elementar de dados e instruções.

3.2 Motivações para o emprego de multiprocessadores

Além da já citada inversão do custo relativo dos componentes (processadores, memórias, equipamento de entrada e saída) existem outras razões a fomentar o interesse por multiprocessadores. Como principais motivações cita-se: o aumento da performance, a flexibilidade, a modularidade e a disponibilidade.

3.2.1 Performance

A performance é definida em termos de velocidade de execução das tarefas (throughput) e do tempo de resposta.

Existe uma série de fatores que contribuem para a limitação da performance dos monoprocessadores. Um deles é o fator tecnológico. Os processadores convencionais não conseguirão aumentar sua velocidade ao esbarrar no limite imposto pelo tempo de propagação dos sinais. Embora este limite ainda não tenha sido atingido é possível prever-se que ele fatalmente o será. A demanda por uma velocidade maior terá então de ser suprida por modificações organizacionais. Os candidatos com maior probabilidade de virem a prover este aumento de velocidade são a execução simultânea ou concorrente das operações.

Um segundo fator de degradação em sistemas monoprocessados é o tempo desperdiçado com a multiplexação dos processos. O tempo de resposta é aumentado pela preempção na mudança de contexto. O tempo de resposta é um aspecto muito importante para aplicações em tempo real como, por exemplo, controle de processos. Neste tipo de aplicação é exigida uma resposta rápida aos eventos externos. Neste aspecto, um multiprocessador é mais efetivo pois um evento pode ativar um processador ocioso sem suspender o processamento dos demais.

Um terceiro aspecto negativo a ser levado em conta nos sistemas monoprocessados é o crescimento exponencial do custo de software a medida que o monoprocessador aproxima-se dos limites de performance[34]. O aumento de velocidade por si só não dá uma medida real dos benefícios alcançados por uma determinada arquitetura multiprocessada em relação a outra ou a uma monoprocessada: é preciso levar em consideração o custo envolvido. Por isto, uma medida mais real de comparação é a relação velocidade/custo (cost-effectiveness). Há indicações que a performance é consideravelmente melhorada pelos multiprocessadores. Um estudo comparativo entre o C.mmp (um multiprocessador) e o PDP-10 (um monoprocessador) revelou que o primeiro é mais efetivo que o segundo num fator de 3,59 [38].

A performance de sistemas multiprocessados está intimamente ligada à sua organização interna. O número de processadores, módulos de memória, estrutura de interconexão são alguns dos aspectos determinantes. Existem muitos estudos teóricos visando determinar a performance de multiprocessadores[4,5,7,46,55]. Estes estudos são muito difíceis de serem realizados devido ao grande número de variáveis em jogo. Existem

discordâncias entre os resultados em virtude das hipóteses feitas em cada caso. Uma questão irrefutável é o aumento não linear de performance a medida que cresce o número de processadores. A adição de um segundo processador não proporciona uma duplicação de performance como se poderia esperar. Isto se deve a dois fatores primordiais: a maior intensidade de contenção devido aos conflitos de acesso aos recursos compartilhados e o uso ineficiente das unidades funcionais por parte do sistema operacional.

3.2.2 Confiabilidade e disponibilidade

A confiabilidade de um sistema é uma medida da sua capacidade de operar sem falhas. A disponibilidade é uma medida da capacidade do sistema tolerar eventos que violem suas especificações e, mesmo assim, continuar operacional. A necessidade de um alto grau de disponibilidade em equipamentos computadorizados de naves espaciais foi uma das principais motivações para o desenvolvimento de multiprocessadores.

Em sistemas monoprocessados, uma das técnicas utilizadas para se garantir a confiabilidade é a redundância. Duas ou mais cópias sincronizadas de uma mesma unidade executam as mesmas tarefas paralelamente. Qualquer discordância nas respostas obtidas por cada uma das unidades é a primeira indicação de uma falha. O sistema é então testado e a origem da falha isolada. Uma outra maneira de se garantir a integridade consiste em manter equipamento reserva em prontidão. Quando uma unidade falha, uma unidade reserva toma o seu lugar através de

chaveamento automático ou manual. Qualquer uma destas técnicas é dispendiosa pois o equipamento adicional não contribui para a produtividade do sistema.

Multiprocessadores são potencialmente confiáveis pois a falha de um processador não implica a falha do sistema todo. Quando isto acontece há uma degradação da performance (graceful degradation) causada pela perda do processador, mas o sistema continua operando. Isto é possível graças à capacidade inerente de reconfiguração. Se existem múltiplas unidades de processamento com capacidades idênticas, a carga de trabalho pode ser redistribuída e uma outra unidade tomar o lugar da que falhou. A alocação dinâmica de tarefas entre os processadores é um requisito essencial para a disponibilidade.

A existência de múltiplas unidades de processamento com capacidades mais ou menos idênticas não é suficiente para assegurar a confiabilidade. Além da alocação dinâmica de tarefas é imprescindível a existência de mecanismos capazes de resolver o impasse criado por uma situação de erro. As questões básicas relacionadas à confiabilidade são: detecção e correção de erro, isolamento das falhas, recuperação de dados manipulados pelo processador onde ocorreu a falha e reconfiguração do sistema[54].

3.2.3 Flexibilidade e modularidade

Várias têm sido as formas de interpretação dos termos flexibilidade e modularidade[34,38,50]. Pode-se pensar na flexibilidade como sendo a habilidade do sistema reconfigurar-

se[50], mas isto é, na verdade, uma capacidade funcional que garante a disponibilidade.

A flexibilidade de um multiprocessador refere-se à capacidade do sistema responder eficientemente a uma gama ampla de cargas de trabalho[9]. A operação do sistema é melhorada em relação a um monoprocessador pela utilização mais eficiente dos recursos do sistema e do tempo dos processadores. A flexibilidade é demonstrada através de um exemplo[49]. Supondo-se que dez programas devam ser executados e um único monoprocessador não tenha condições de fazê-lo, a solução é dividir o trabalho em dois grupos de cinco programas executados por dois monoprocessadores diferentes. Isto pode resultar num grupo de programas com alta carga de entrada e saída num computador e outro grupo com alta carga de processamento aritmético e lógico no outro. Isto faz com que a primeira UCP tenha uma taxa de utilização baixa enquanto a segunda fica sobrecarregada. A execução dos dez programas em um multiprocessador é mais eficiente pois a carga de trabalho é equilibrada dinamicamente.

A flexibilidade também se manifesta em situações onde não há possibilidade de paralelismo na execução de um programa. Neste caso, ao invés de as unidades de processamento executarem processos independentes, cada uma delas é desginada para programas independentes.

Evidentemente, a flexibilidade de um multiprocessador está condicionada ao comportamento do sistema operacional diante das diferentes situações.

A modularidade ou extensibilidade é uma medida da capacidade de modificação da performance e funcionalidade do sistema. Um sistema modular pode ser configurado de acordo com

as necessidades de processamento assegurando-se sempre a viabilidade econômica. Assim, para ambientes com demanda reduzida,o número de processadores seria pequeno. A medida que a demanda por capacidade de processamento cresce, o número de processadores também cresce. A adição de um processador (juntamente com memória e equipamento periférico necessário) em um sistema modular provê um aumento discreto de performance acompanhado de um aumento também discreto de custo.

A modularidade de sistemas monoprocessadores limita-se à memória e ao subsistema de entrada e saída. A adição de mais um processador, salvo exceções[49], envolve modificações bastante grandes na arquitetura do sistema. Os sistemas multiprocessados, por sua vez, são potencialmente modulares, permitindo configurações variadas para uma ampla gama de ambientes. Eles são "potencialmente" modulares porque este atributo está intimamente ligado à estrutura de interconexão entre as unidades do sistema. Dependendo da forma como ela é implementada o sistema será mais ou menos modular. Além disso, uma expansão de hardware tem implicação direta sobre o sistema operacional. A modificação do sistema operacional necessária para suportar um sistema maior pode mostrar-se difícil se ele não foi projetado para tal.

Os integrados LSI, principalmente os microprocessadores, prestam-se ao desenvolvimento de sistemas modulares pelo seu baixo custo. A incorporação de microprocessadores como UCP's de multiprocessadores objetiva alcançar uma alta relação velocidade/custo.

3.3 Organização de hardware

3.3.1 Requisitos básicos

Um dos requisitos que caracterizam um multiprocessador é o partilhamento dos recursos do sistema entre as unidades de processamento que o compõem. Esta característica tem implicações tanto sobre o hardware como o software.

Com respeito ao hardware este requisito, aliado à interação entre unidades, reflete-se sobre a estrutura de interconexão do sistema. Esta estrutura não é encarada sob o ponto de vista comunicação (como no capítulo 2) mas, isto sim, sob o enfoque da organização do sistema. Neste aspecto,ela destina-se a interligar as subunidades do multiprocessador (memórias, UCP's e dispositivos de entrada e saída). É claro que, no caso da comunicação entre processadores ser feita através da memória, a estrutura de interconexão fará parte do subsistema de comunicação, por se constituir na via de acesso à memória.

Portanto, sob o enfoque da organização do sistema, a estrutura de interconexão deve:

1º) Permitir que todos os processadores tenham acesso à memória global e aos dispositivos de entrada e saída.

2º) Permitir que a interação entre os processadores aconteça até o nível de operação mais elementar. Por exemplo, dois processadores devem ser capazes de ter acesso a uma mesma palavra de memória concorrentemente.

A maneira como essa estrutura é implementada é um fator chave na determinação das características do sistema. Os objetivos de um multiprocessador estão intimamente ligados à

forma como as suas subunidades são interligadas. Existem três organizações fundamentais: um único barramento partilhado, matriz de barramentos cruzados, e múltiplos barramentos/memórias multiporta.

As estruturas de interconexão mencionadas acima possibilitam o compartilhamento dos recursos. Além da estrutura de interconexão, outras funções (estas relacionadas ao controle) devem existir para que o sistema operacional possa operar efetiva e eficientemente.

Para assegurar a integridade de estruturas de dados manipulados pelos vários processadores, o sistema operacional utiliza mecanismos chamados de primitivas de sincronização. A implementação destes mecanismos requer construções em hardware as quais atuam como semáforos, a fim de impedir que um processador tenha acesso a uma estrutura de dados enquanto um outro a está utilizando.

Um mecanismo de interrupção entre processadores permite que um processador requisite a atenção de um outro a fim de que este execute uma determinada tarefa, ou receba uma mensagem.

3.3.2 Barramento único

A maneira mais simples de se interconectar as subunidades de um multiprocessador é através de um único barramento comum a todas elas (figura 10).

Toda comunicação entre subunidades efetua-se através do barramento, não havendo conexões contínuas entre elas.

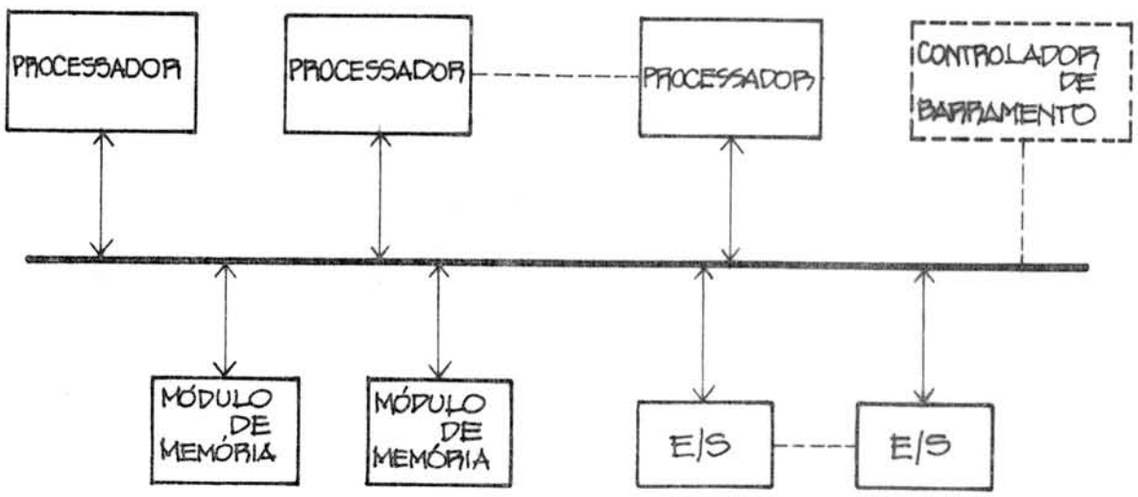


Figura 10 - Interconexão das unidades funcionais através de um barramento compartilhado

O barramento é um elemento totalmente passivo que não possui chaves direcionadoras de mensagens. O controle da utilização é feito através do partilhamento no tempo.

Existem, ligados ao barramento, elementos ativos e também passivos. Elementos ativos ou mestres são aqueles capazes de iniciar uma transferência. Elementos passivos ou escravos não iniciam uma transferência, apenas participam dela sob iniciativa de um mestre. Os processadores e canais de entrada e saída são elementos ativos. Memórias e dispositivos de entrada e saída (sem ADM) são elementos passivos.

O barramento, como recurso partilhado entre os elementos ativos do sistema, está sujeito ao problema de interferência. A interferência acontece quando dois ou mais dispositivos tentam utilizar o recurso ao mesmo tempo. Por esta razão qualquer dispositivo deve obter o controle do barramento antes de efetuar uma transferência. O protocolo de arbitração é res-

ponsabilidade do árbitro ou controlador de barramento. Ele aparece na figura 10 como um bloco pontilhado pois não é necessariamente um elemento isolado no sistema. O árbitro pode estar fragmentado nas interfaces dos elementos ativos.

As vantagens da interconexão através de um único barramento são:

- Simplicidade das interfaces devido ao fato de não serem necessárias chaves. Os dispositivos devem apenas obedecer os protocolos de arbitração e comunicação estabelecidos pelo barramento.

- Modularidade. A adição ou remoção de um dispositivo em qualquer ponto do sistema exige pequenas ou nenhuma modificação de hardware.

- Baixo custo de hardware. Cada dispositivo ou unidade funcional necessita de apenas uma interface. O barramento é formado por um conjunto de linhas totalmente passivas. Além disso, a simplicidade das interfaces exige poucos componentes.

A par das vantagens oferecidas, o barramento único tem algumas limitações importantes:

- O barramento é um componente crítico no qual repousa toda a operação do sistema. Uma falha em qualquer uma das interfaces pode resultar em falha total do sistema.

- O sistema não alcança o nível mais baixo de interação já que a cada instante só pode acontecer uma transferência. Em conseqüência, a performance fica amarrada à velocidade do barramento.

- O barramento é o gargalo do sistema. A expansão é limitada pela degradação de performance resultante do congestionamento da via.

A fim de suplantar os problemas de congestionamento e suscetibilidade a falhas pode-se expandir o número de barramentos (figura 11). Todas as unidades são conectadas a qual-

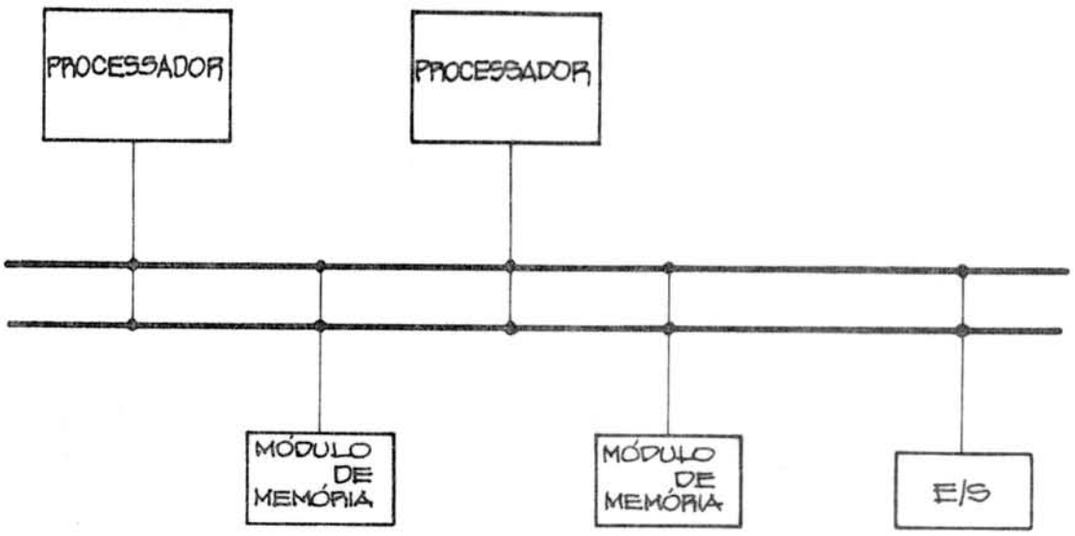


Figura 11 - Interconexão das unidades funcionais através de múltiplos barramentos compartilhados

quer um dos barramentos, de forma que pode acontecer mais de uma transferência a cada instante. A falha de um dos barramentos não implica parada total do sistema pois existe um segundo caminho alternativo. Em compensação, o barramento deixa de ser uma estrutura passiva para tornar-se um elemento chaveador de circuitos. Cada ponto de conexão é uma chave que controla o direcionamento de informação. A maior confiabilidade e performance é obtida em troca do aumento de complexidade e custo.

3.3.3 Matriz de barramentos cruzados

A matriz de barramentos cruzados (figura 12) é um exemplo de chaveamento de circuitos (seção 2.4.4). Ela é capaz de interconectar elementos de um grupo (processadores ou canais de entrada e saída) a elementos de um outro grupo(módulos

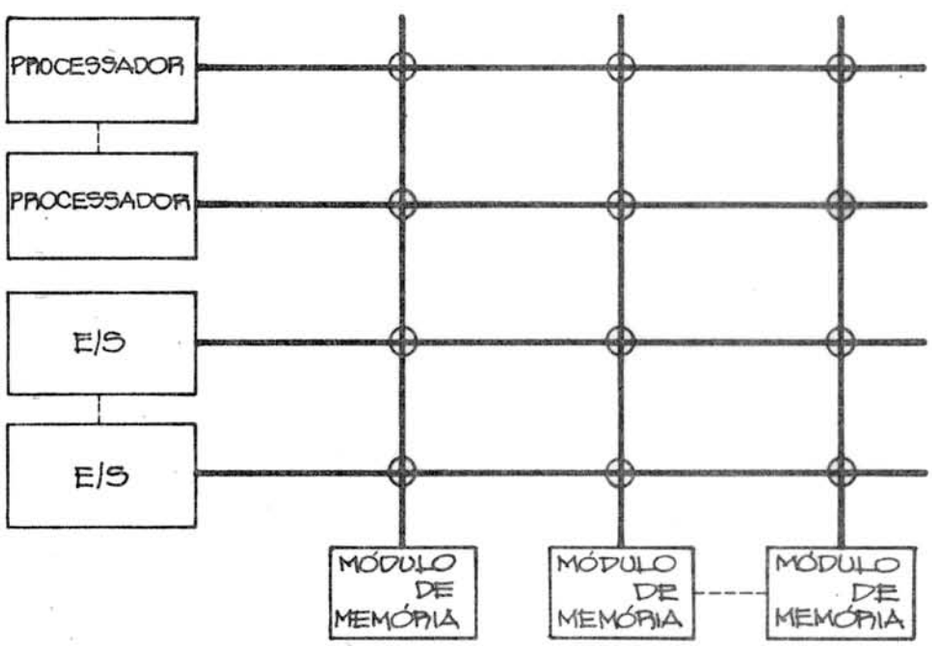


Figura 12 - Interconexão das unidades funcionais através de uma matriz de barramentos cruzados

de memória) aos pares. Múltiplas conexões simultâneas podem ocorrer desde que os elementos de cada par sejam mutuamente exclusivos. A conexão pode ser efetuada com base em duas filosofias. A primeira é função dos endereços gerados pelos processadores os quais são mapeados nos módulos de memória. Neste caso a conexão é mantida durante um ciclo de memória apenas.

A segunda filosofia consiste em estabelecer a conexão com base nos processadores, módulos de memória e equipamentos de entrada e saída necessários à execução de um processo[40]. Neste caso a conexão deve ser feita pelo sistema operacional com o conhecimento prévio das necessidades de cada processo. A conexão não mais se limita a pares de dispositivos, mas a partições do sistema. Além disso, ela deve ser mantida durante todo o tempo em que o processo está ativo.

A matriz é um meio conceitualmente simples de interligar os processadores e módulos de memória, porém seu controle é bastante complexo. O controle deve permitir o estabelecimento de várias conexões simultâneas e resolver os conflitos de acesso a um mesmo módulo de memória. Além deste, um outro fator negativo é o crescimento geométrico da matriz de chaves a medida que novos dispositivos são acrescentados. Isto torna o seu custo proibitivo em sistemas com muitos processadores ou módulos de memória[23].

A maior vantagem da matriz de chaveamento é a possibilidade de uma alta taxa de transferência entre os processadores e a memória, o que contribui para a melhoria da performance do sistema.

Uma característica da matriz é a possibilidade de particionar e reconfigurar o sistema[20]. Por exemplo, um módulo de memória pode ser retirado de operação desabilitando-se a sua coluna de chaves; pode-se também obter dois sistemas independentes desabilitando-se todas as chaves de dois quadrantes diametralmente opostos da matriz. A reconfiguração dinâmica pode ser usada pelo sistema operacional para isolar elementos defeituosos. É possível, também, através de reconfiguração ma-

nual, executar reparos sem interromper a operação do sistema. É claro que alterações desta ordem somente são possíveis se o software do sistema assim o permite.

A expansão da capacidade do sistema não é limitada como no caso do barramento único pois a matriz não é um ponto de estrangulamento. Ela pode ser projetada de forma a admitir expansões apenas pela adição de mais chaves de uma forma modular, ou então pode-se implementar a matriz com a configuração máxima de chaves e equipar o sistema com uma configuração menor. Isto implica um alto custo inicial[20], porém a expansão é mais simples. Em ambos os casos não são necessárias alterações nas interfaces de dispositivos, pois toda a lógica de controle e chaveamento está localizada na matriz. As interfaces individuais de memória são muito simples já que elas não precisam reconhecer tentativas de acesso nem resolver conflitos.

3.3.4 Múltiplos barramentos/memórias multiporta

Múltiplos barramentos e memórias multiporta constituem-se numa terceira alternativa para a estrutura de interconexão de multiprocessadores.

Neste tipo de organização cada processador possui um barramento privativo para comunicar-se com a memória (figura 13). Como os barramentos são dedicados, não há contenção no acesso aos mesmos. Cada um dos barramentos deve ser ligado aos módulos de memória independentemente dos demais. A conexão em separado é feita através de portas nas interfaces das memórias. As portas provêm o necessário isolamento elétrico entre

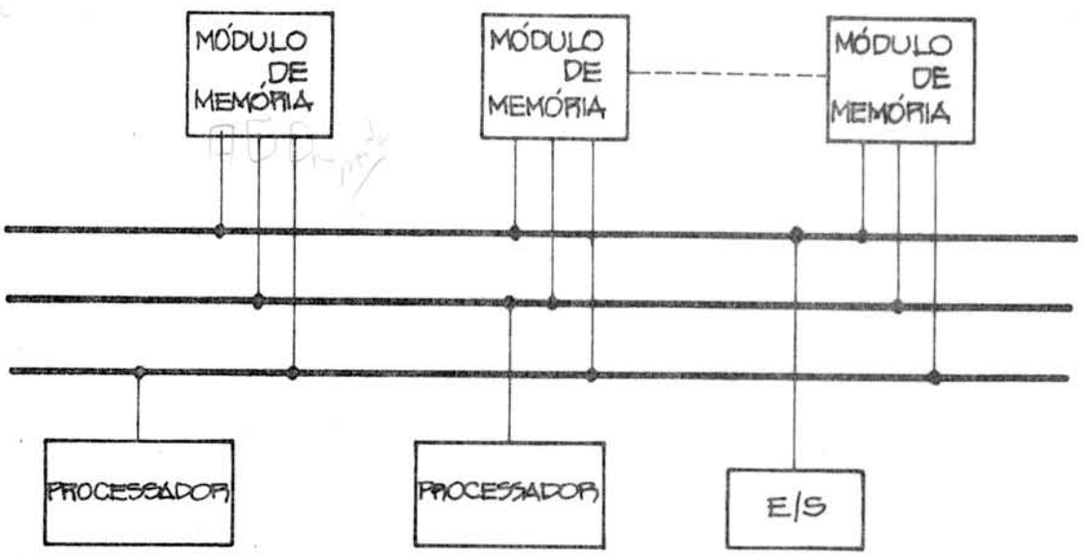


Figura 13 - Interconexão das unidades funcionais através de múltiplos barramentos e memórias multiporta

os barramentos e o arranjo de memória abrigado pelo módulo. O problema resume-se à implementação de um algoritmo de controle que permita a passagem dos sinais de um único barramento a cada instante. Um método freqüentemente utilizado é a designação de prioridade fixa para cada porta. As portas de um mesmo barramento podem ter prioridades diferentes de um módulo para outro.

A estrutura pode ser vista como uma versão descentralizada da matriz de chaveamento[13,18]. De fato, o problema a ser resolvido é o mesmo, com a diferença que o controle é independente para cada módulo pois migrou da matriz para as interfaces de memória. Uma memória multiporta exige a multiplicação da lógica de arbitração em cada módulo do sistema. Isto, aliado a necessidade de tantas portas quantos forem os processadores, faz com que os módulos de memória tornem-se caros.

Não há diferença na taxa de transferência proporcionada pela organização através de memórias multiporta ou matriz de barramentos cruzados. A diferença maior está na grande quantidade de conectores necessários para a primeira. Se existem m processadores e n módulos de memória são necessários m x n conectores para se obter interconectividade total.

Uma outra desvantagem é a inaptidão ao crescimento modular. O tamanho máximo do sistema é limitado pelo número de portas disponíveis. A configuração máxima é definida desde a configuração inicial e qualquer modificação posterior torna-se difícil.

As memórias multiporta facilitam a designação de áreas de memória privativas a certos processadores. Isto possibilita a proteção de rotinas e dados contra acessos não autorizados. Por outro lado, em caso de falha do processador, torna-se impossível a reconfiguração do sistema se nenhum outro tem acesso à informação armazenada na memória associada ao que falhou.

3.4 Subsistema de entrada e saída

Segundo a definição mais estrita de multiprocessamento, todos os processadores partilham os recursos do sistema. Entre estes se incluem os dispositivos de entrada e saída. A forma de permitir o partilhamento destes recursos tem implicações decisivas nas características do sistema. O projeto do subsistema de entrada e saída não deve ser relegado a segun-

do plano. Pelo contrário, ele deve ser pensado em conjunto com a memória no projeto da estrutura de interconexão.

No caso do barramento único, uma opção arquitetônica consiste em permitir que qualquer processador controle diretamente qualquer periférico[1,50]. Os dispositivos de entrada e saída ativos (ADM) e passivos (sem ADM) são ligados diretamente ao barramento (figura 14).

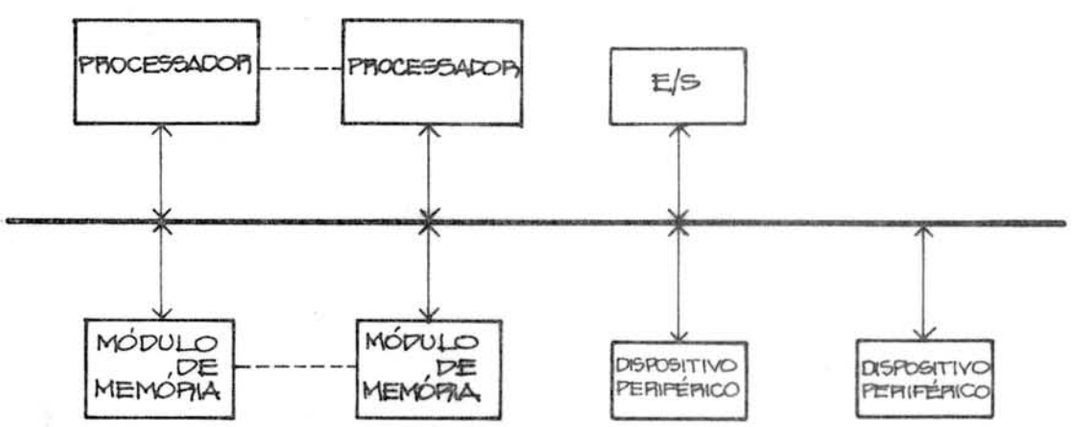


Figura 14 - Entrada e saída no barramento do sistema

Neste esquema todos os mestres utilizam o barramento na busca de instruções, leitura e escrita de dados na memória ou dispositivos de entrada e saída. O resultado é a rápida saturação do barramento, principalmente se existem dispositivos rápidos operando em ADM[1]. A interferência resultante pode causar esperas muito longas que fazem baixar a performance do sistema. Além disso, os processadores ficam responsáveis pela monitoração da entrada e saída além das tarefas normais de processamento.

O barramento do sistema é desengarrafado com a im-

plementação de uma estrutura que prevê a utilização de processadores dedicados à entrada e saída. Ao barramento local são ligados os dispositivos periféricos e uma memória, ambos privativos do processador ao qual pertencem (figura 15). Estes processadores ficam responsáveis por todos os processos de entra-

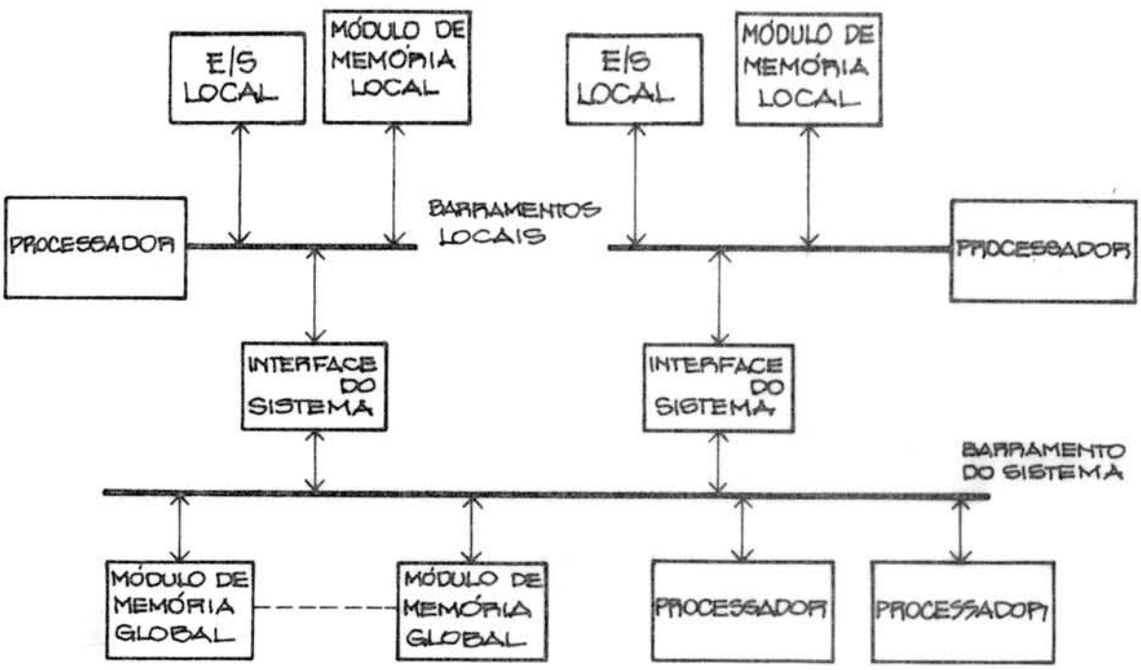


Figura 15 - Barramento único com processadores de entrada e saída

da e saída. A separação em processadores orientados à entrada e saída e processadores de uso geral melhora a performance do sistema por três motivos:

1º) Os processadores de uso geral são aliviados da tarefa de monitorar operações de entrada e saída; desta forma eles ficam liberados para a execução de outros processos.

2º) O barramento do sistema fica livre das operações de entrada e saída. Em conseqüência, a memória global po

de ser acessada por qualquer processador durante as transferências de e para periféricos.

3º) Possibilidade de operação em paralelo dos dois tipos de processadores.

O terceiro item, acima citado, origina-se do fato de as operações sobre o barramento local não interferirem no barramento do sistema, a não ser que o acesso refira-se a um endereço da memória global. O acesso ao barramento global é feito através da interface do sistema. A recíproca não é verdadeira. Ou seja, um processador de uso geral não pode referenciar endereços de memória local ou periféricos de outro. As referências à memória global estão sujeitas ao protocolo de arbitração do barramento. Se há um outro mestre utilizando o barramento, o pedido de acesso é postergado.

A arbitração do barramento local só é necessária se existem canais de ADM a ele ligados. Estes, como potenciais mestres de barramento, concorrem com o processador na utilização do mesmo.

A arbitração do barramento local é um exemplo de problema que pode ser resolvido pela utilização de um processador voltado especificamente à entrada e saída[17]. O Processador de Entrada e Saída é projetado especificamente para esta função e como tal é capaz de melhorar ainda mais a performance. Sua arquitetura interna, conjunto de instruções e interface com o meio são adaptados para o controle de operações de E/S.

Da mesma forma que o processador de entrada e saída é usado no barramento único, ele também se aplica à matriz de barramentos cruzados (figura 16) e à memória multiporta (figura 17).

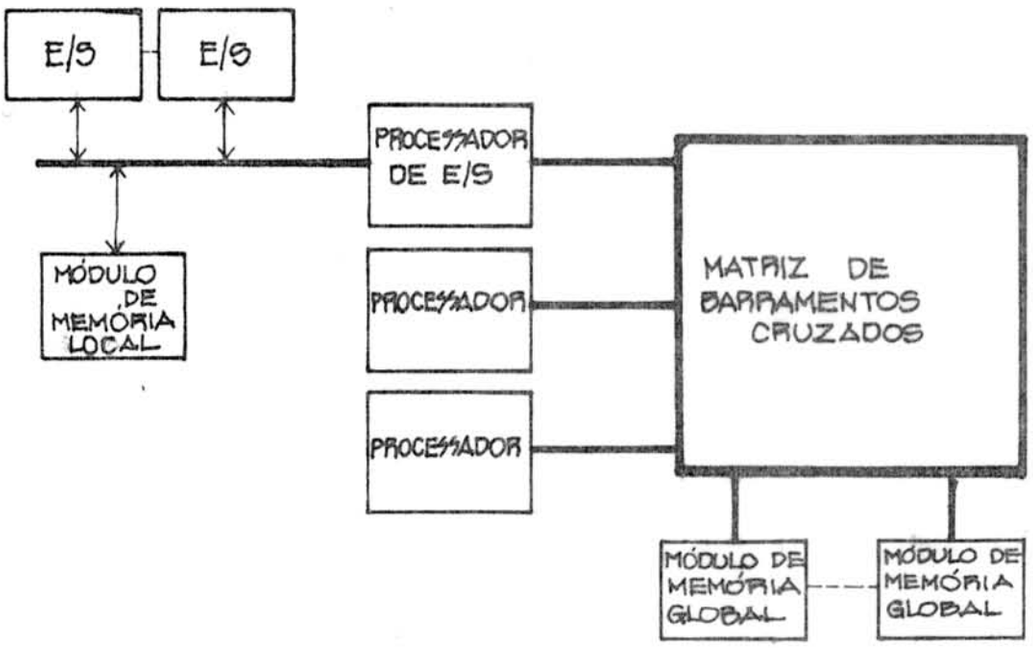


Figura 16 - Matriz de barramentos cruzados com processadores de entrada e saída

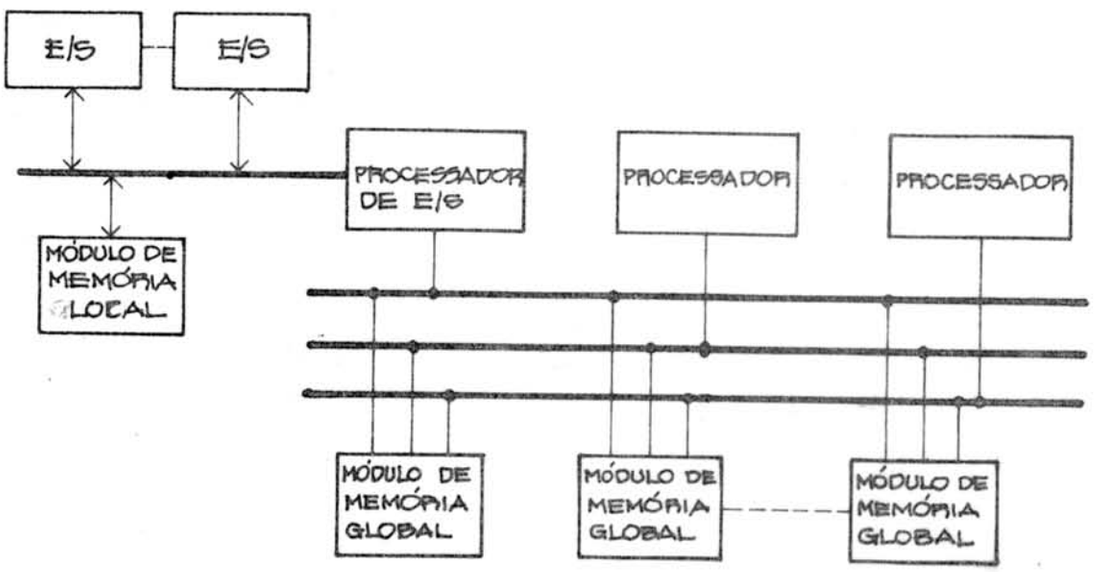


Figura 17 - Múltiplos barramentos/memórias multiporta com processadores de entrada e saída

As figuras acima mostram apenas um processador de entrada e saída em cada caso. Todavia, nada impede, a priori, que mais de um seja usado, ou mesmo que todo processador do sistema possua um certo número de periféricos sob seu controle.

O único problema de se dedicar periféricos a um só processador é a perda dos mesmos em caso de falha do processador. Uma opção interessante, capaz de anular esta desvantagem e dar maior flexibilidade ao subsistema de entrada e saída é a interligação de periféricos e processadores de entrada e saída através de uma matriz de barramentos cruzados (figura 18).

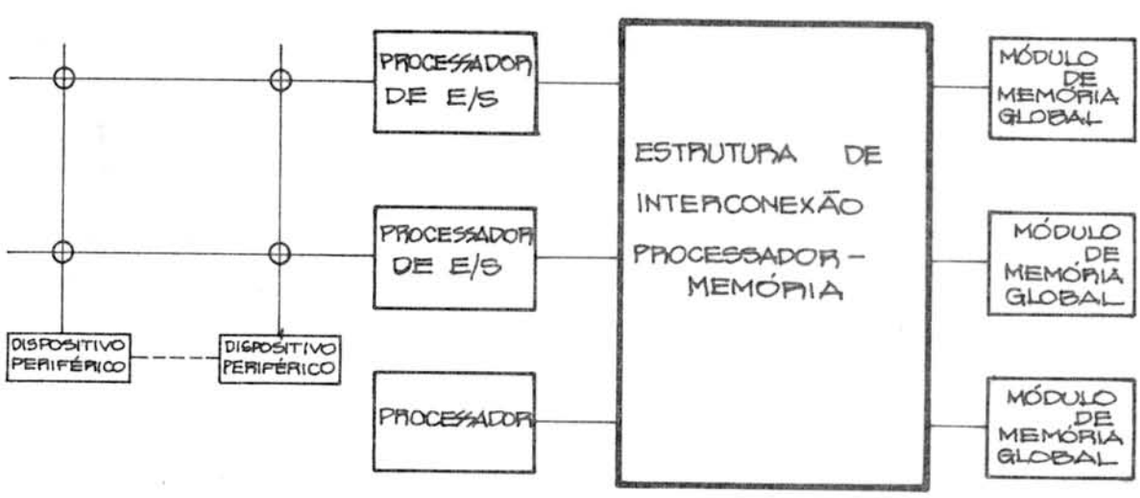


Figura 18 - Interligação de periféricos e processadores de entrada e saída via matriz de barramentos cruzados

Conceitualmente, a matriz é idêntica àquela que une a memória e os processadores. Porém ela é de implementação mais simples em virtude de os dispositivos de entrada e

saída reconhecerem apenas um pequeno número de endereços (registradores internos), ao contrário de um módulo de memória o qual compreende um grande número de endereços.

3.5 Sistema operacional

3.5.1 Controle do sistema

Os mecanismos de controle são o ponto vital de qualquer sistema de computação. Os sistemas multiprocessados em particular exigem mecanismos de controle mais sofisticados que um monoprocessador multiprogramado. Embora os sistemas operacionais de ambos possuam capacidades funcionais muito parecidas, há algumas importantes diferenças geradas pelo paralelismo real introduzido. Citam-se, entre outras: a sincronização, a comunicação entre processadores, a recuperação do sistema em caso de falhas, a detecção de paralelismo, o equilíbrio de processamento e entrada/saída entre os processadores, o gerenciamento de memória, a prevenção de impasses.

As funções de controle consideradas neste trabalho são a sincronização, a comunicação entre processadores e o gerenciamento de memória. Estas são analisadas em virtude de estarem relacionadas à memória do sistema.

3.5.1.1 Sincronização

Em um multiprocessador existem recursos de hardwa-

re compartilhados tais como memórias, barramentos, dispositivos de entrada e saída, e até mesmo processadores. Ao nível de software existem os arquivos, tabelas, áreas de armazenamento temporário de dados, e variáveis. Para que estes recursos sejam utilizados ordenadamente é necessário implementar mecanismos de exclusão mútua também chamados de primitivas de sincronização. Estes mecanismos garantem que não aconteça a situação de dois ou mais processadores alterarem o estado de determinado recurso ao mesmo tempo. Os resultados de tal situação são imprevisíveis.

A forma como é implementado um mecanismo de sincronização depende fundamentalmente do tipo de recurso ao qual ele é aplicado. A coordenação acontece tanto em alto nível entre processos executados em diferentes processadores como em baixo nível entre processadores competindo por um barramento ou pelo acesso à memória[53]. Tanto a memória como o barramento são recursos que sofrem uma demanda muito grande por parte dos processadores. A alta freqüência de acesso exige mecanismos de sincronização muito rápidos de forma a não manter o recurso bloqueado por um período de tempo muito longo enquanto se processa a sincronização. Por este motivo, as primitivas de sincronização de baixo nível são implementadas em hardware. Os protocolos de arbitração de barramentos (introduzidos no capítulo 2) são exemplos de primitivas de sincronização desse tipo. Os protocolos de arbitração mais comuns são detalhados no capítulo 4.

A sincronização acontece num nível superior quando os processos competem pela utilização de recursos mais sofisticados, tais como, dispositivos de entrada e saída, ou pela en-

trada em seções críticas de código. Uma seção crítica é um segmento de programa o qual, uma vez iniciado, deve ser executado até o fim antes que outro processador tenha acesso ao mesmo. Todos os segmentos de programa que manipulem estruturas de dados compartilhadas constituem seções críticas. A sincronização em alto nível é obtida pela construção MUTEXBEGIN/MUTEXEND[29] da forma ilustrada abaixo:

```
PROCESSO P1: BEGIN                    PROCESSO P2: BEGIN

        -                                     -
        -                                     -
    MUTEXBEGIN                            MUTEXBEGIN
    acessa variáveis      SEÇÃO           acessa variáveis
    compartilhadas        CRÍTICA         compartilhadas
    MUTEXEND                              MUTEXEND
        -                                     -
        -                                     -
        -                                     -
    END                                   END
```

A construção acima garante que os processos P1 e P2 terão acesso mutuamente exclusivo às seções de programa entre MUTEXBEGIN e MUTEXEND. Num determinado instante P1 ou P2 podem executar a seção crítica, mas não ambos.

As primitivas de sincronização de alto nível foram introduzidas no intuito de permitir a cooperação entre processos em um monoprocessador multiprogramado. No entanto, elas mostram-se aplicáveis a multiprocessadores igualmente.

A questão básica a ser resolvida, com respeito às primitivas de sincronização é a forma como são implementadas as

construções MUTEXBEGIN e MUTEXEND. MUTEXBEGIN deve determinar se já existe um processo executando a seção crítica. Em caso afirmativo o processo que deseja utilizá-la deve esperar. Se, ao contrário, ela está livre o processo passa a executá-la. Ao penetrar na seção crítica, o processo deve ligar um indicador que barra a entrada a outros processos. MUTEXEND desliga o indicador no instante em que o processo deixa a seção crítica, habilitando a entrada de um outro (se houver) em estado de espera.

Uma maneira de se implementar estas construções é através de uma variável booleana associada à seção crítica a qual indica se a seção está ou não ocupada. Quando um processador encontra a seção crítica ocupada ele permanece testando repetidamente a variável até encontrá-la desocupada. Nesse instante o processador escreve um valor que bloqueia novamente a seção e passa a executá-la. As operações de leitura e escrita da variável constituem uma seção crítica. Ambas devem ser feitas de forma indivisível, senão dois processadores podem testá-la e ligá-la entrelaçadamente. A seqüência de eventos abaixo (figura 19) ilustra a situação.

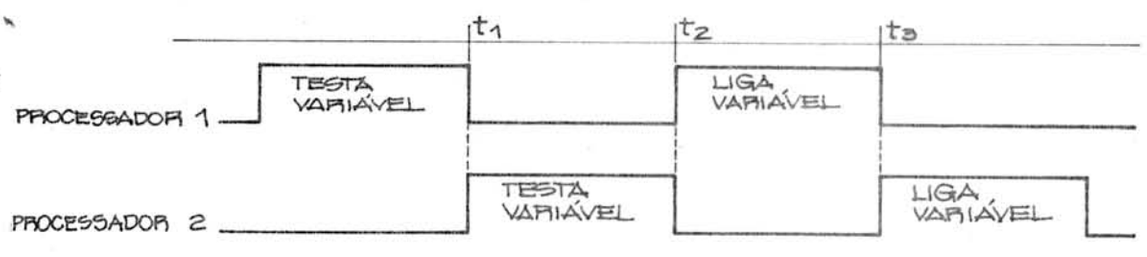


Figura 19 - Entrada simultânea de dois processadores na mesma seção crítica

No instante t_1 o processador 1 acabou de testar a variável e, encontrando a seção crítica desocupada, prepara-se para bloqueá-la ligando a variável em t_2. Porém, nesse mesmo instante, um segundo processador testa a variável e também encontra-a desocupada. O resultado é a entrada simultânea dos dois processadores na mesma seção crítica, e a conseqüente destruição da integridade dos dados, ou a dupla inicialização de um dispositivo de entrada e saída.

A indivisibilidade do teste e ligação da variável pode ser implementada de várias maneiras. Muitos computadores e alguns microprocessadores possuem esta facilidade na forma de uma instrução (TEST AND SET, READLOCK) indivisível[42,67]. As instruções de leitura-modificação-escrita, como são chamadas, mantêm o controle da memória entre os dois ciclos. Uma outra forma de se obter a indivisibilidade consiste em bloquear a via de acesso à memória aos demais processadores durante os dois ciclos[32]. Se nenhum dos mecanismos citados está disponível pode-se implementar a indivisibilidade por um módulo especial de memória dotado de circuitos de controle que causam uma escrita automática toda vez que um ciclo de leitura é executado.

3.5.1.2 Comunicação entre processadores

A comunicação entre processadores é um mecanismo essencial ao funcionamento de um multiprocessador. Através de la um processo sendo executado num processador recebe ou transmite mensagens a um segundo processo sendo executado em outro processador. Além desta função, o mecanismo de comunicação é

usado quando um processador requisita serviços a outro.

A existência de uma memória partilhada sugere que a comunicação entre processadores seja efetuada através dela. Deve-se salientar que a implementação de uma memória global como via de comunicação não acontece em virtude de decisões tomadas ao nível do sistema (ver capítulo 2), mas sim pela própria filosofia de compartilhamento do multiprocessador. Seu uso como via de comunicação é um efeito colateral.

Na comunicação através da memória cada processador possui uma área de memória chamada de caixa-postal onde são colocadas as mensagens provenientes de outros processadores, e a ele endereçadas. A caixa postal é composta por uma área de armazenamento de mensagens e uma área de controle onde é mantido o estado da caixa postal (figura 20).

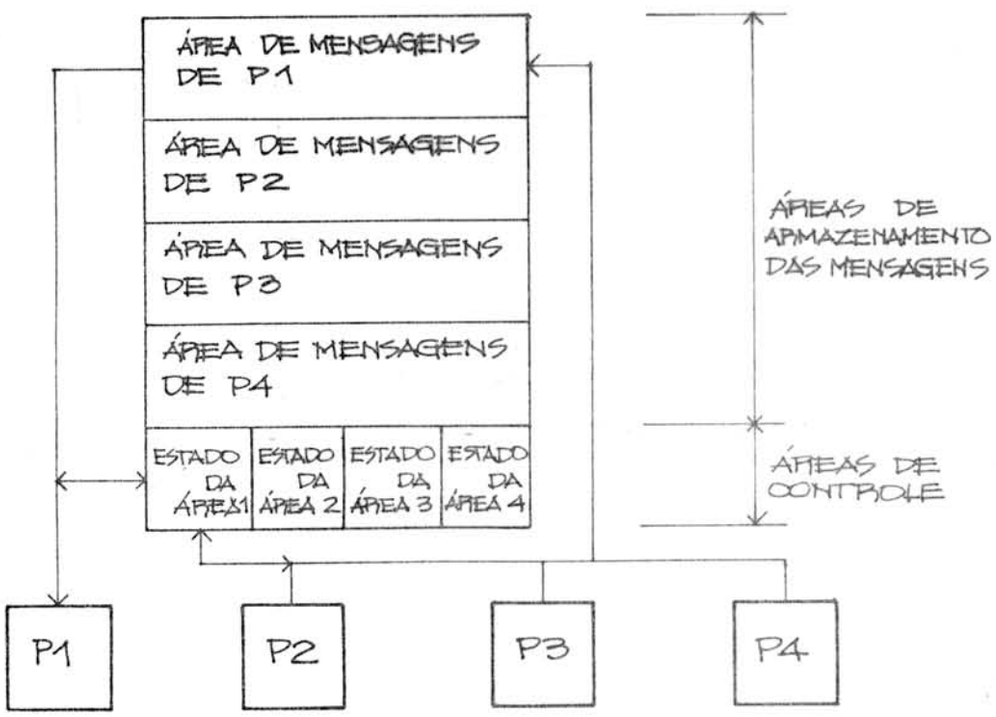


Figura 20 - Mapa de uma caixa postal

Quando um processador qualquer deseja estabelecer comunicação com P1 ele deposita na área de mensagens deste a própria mensagem ou então o endereço de memória onde ela se encontra e o seu comprimento. Quando a mensagem está pronta para ser transferida o remetente deve habilitar P1 a reconhecer a presença de uma mensagem na sua caixa-postal, e também descobrir sua procedência. Estas informações são depositadas nos campos de controle da caixa postal de P1. Uma vez executado este procedimento P1 deve ser informado da existência de uma mensagem em sua caixa-postal. A monitoração da caixa postal pode ficar a cargo de cada processador. Nesse caso cada processador deve, periodicamente, verificar o estado de sua caixa postal. Entretanto, uma resposta mais rápida é obtida através de um mecanismo de interrupção entre os processadores[33].

A manipulação da caixa postal é uma seção crítica, e como tal deve ser protegida contra acessos simultâneos por mais de um processador.

A comunicação via memória não é a única opção existente para um multiprocessador. Outros mecanismos de comunicação, como barramentos[36] e linhas de comunicação seriais[14] também são utilizados.

3.5.1.3 Gerenciamento de memória

As técnicas de gerenciamento de memória de multiprocessadores são essencialmente as mesmas empregadas em monoprocessadores: a paginação e a segmentação.

A questão fundamental a ser resolvida com relação

à conversão dos endereços lógicos em endereços físicos é a localização do hardware de transcodificação dinâmica de endereços. O mapeamento pode ser feito na memória, nos processadores ou em uma unidade especial que armazena a tabela de páginas em registradores rápidos[37].

Se os registradores que armazenam o mapa de páginas estão na memória isto implica um único conjunto de registradores para todos os processadores. Conseqüentemente a memória só poderia contar com uma porta de comunicação com os processadores. Isto, por sua vez, limitaria a organização do sistema ao barramento único ou à matriz de barramentos cruzados.

Se os registradores estão localizados nos processadores ou em unidades especiais associadas a cada processador[67], então o efeito se fará sentir sobre o sistema operacional. Quando acontece uma chamada do sistema operacional ou uma interrupção, o processador que executa o atendimento pode necessitar do mapa de páginas do programa que requisitou atendimento. Este mapa, por sua vez, não está necessariamente localizado no processador que atende o pedido. A solução para este problema consiste em manter uma tabela de páginas global na memória. De posse desta tabela, o sistema operacional tem uma visão de todas as páginas presentes na memória e a qual processador elas pertencem. Toda modificação da tabela de páginas individual de um processador requer a atualização da tabela global. Estas modificações devem ser feitas de forma mutuamente exclusiva.

3.5.2 Tipos de sistema operacional

O sistema operacional de um multiprocessador assemelha-se ao de um monoprocessador multiprogramado[24,39,47]. Ele é um pouco mais complexo pelo fato de os processos poderem ser executados em mais de uma unidade de processamento física. Entretanto, o paralelismo real introduzido pelo multiprocessador não é necessariamente causador de maiores dificuldades. Um cuidado adicional é assegurar que apenas um processador assuma a tarefa de supervisão do sistema a cada instante. Por exemplo, a concorrência pela utilização de um recurso partilhado deve ser resolvida por apenas um processador que executa a rotina de sincronização. Dependendo da forma como ele é implementado, nem sempre a mesma UCP supervisiona o sistema porém num determinado instante apenas uma delas incumbe-se desta tarefa.

Os sistemas operacionais de multiprocessadores são organizados em torno de três filosofias[9]: mestre-escravo, cópias distintas por processador e flutuante.

3.5.2.1 Sistema mestre-escravo

Este é o sistema mais simples e também o menos eficiente na utilização dos recursos do sistema.

Uma das unidades de processamento é designada como Mestre. A execução das rotinas do sistema operacional ficam a ele restritas. Todas as outras são designadas como escravas e dependem da unidade mestre para obter auxílio do sistema opera

cional. A centralização do sistema em um único processador facilita sua implementação pois as rotinas não precisam ser reentrantes. Além disso, somente o mestre tem acesso ao sistema. Portanto, este pode ser armazenado na sua memória local. Em compensação, a distribuição de tarefas torna-se mais lenta obrigando os escravos a esperas ociosas, o que diminui a eficiência.

O sistema mestre-escravo é eficiente em aplicações dedicadas onde as tarefas são conhecidas de antemão e a sua alocação pré-determinada[50].

3.5.2.2 Sistema privativo por processador

Neste tipo de organização cada unidade de processamento tem sob seu controle um grupo de dispositivos os quais não são partilhados com os demais. Portanto, uma unidade de processamento possui apenas as rotinas necessárias para controlar os seus recursos e para comunicar-se com as demais. Como elas são privativas de um processador elas podem ser armazenadas na memória local, diminuindo a demanda pela memória global. As rotinas comuns a mais de um processador devem ser replicadas ou então feitas reentrantes. As estruturas de dados compartilhadas por cada um dos sistemas devem ser armazenadas na memória global.

3.5.2.3 Sistema flutuante

O sistema operacional flutuante é o mais flexível dos três e também o mais complexo.

Todas as unidades de processamento têm acesso ao sistema operacional, localizado na memória global. As tarefas do sistema são executadas ora por um processador ora por outro. Por isto diz-se que o sistema flutua de um processador para outro. As rotinas partilhadas devem ser feitas reentrantes já que mais de um processador pode acessá-las ao mesmo tempo.

Através desta organização obtém-se uma utilização mais eficiente dos recursos devido à maior rapidez de atendimento do sistema operacional. Também, a carga de trabalho é melhor distribuída.

4 BARRAMENTOS DIGITAIS

4 BARRAMENTOS DIGITAIS

4.1 Introdução

Os barramentos digitais são largamente utilizados nos vários níveis de organização de um computador. Assim, por exemplo, eles interligam registradores e unidades funcionais (ao nível da microprogramação ou transferência entre registradores), subunidades (ao nível do sistema)[30,43], e até mesmo unidades de processamento completas (em sistemas distribuídos)[14].

A complexidade crescente dos sistemas digitais faz crescer o interesse no problema de interconexão. O encapsulamento de um grande número de funções em um pequeno espaço traz consigo um problema fundamental: como interconectar de forma eficiente sistemas cada vez mais complexos e densos. Em particular a resposta a esta pergunta é importante para um multiprocessador onde uma grande quantidade de dados é trocada entre os processos. O tipo de interconexão escolhida irá ter uma influência direta nas características físicas e funcionais do sistema.

Este capítulo visa classificar e descrever os barramentos digitais através de um conjunto de parâmetros que os caracterizam. Seu conteúdo será usado como subsídio no projeto de uma estrutura de interconexão entre processadores e memórias de um sistema multiprocessador.

É importante salientar que a abordagem sistemática ao projeto de barramentos não se aplica somente aos sistemas que utilizam este tipo de estrutura como via de comunicação. Os multiprocessadores fortemente acoplados efetuam a comu

nicação através da memória; a via de acesso entre esta e os processadores é, geralmente, um ou mais barramentos. Neste aspecto, o(s) barramento(s) que interliga(m) processadores e memórias e um barramento usado apenas para comunicação desempenham tarefas distintas dentro do sistema. Entretanto, sob o ponto de vista funcional as considerações a serem feitas no projeto de um ou outro são idênticas. Por esta razão a abordagem feita neste capítulo é aplicável a ambos.

4.2 Tipos de barramentos

O conceito mais difundido de barramento baseia-se na filosofia de partilhamento de um caminho, de forma a acomodar as necessidades de comunicação entre múltiplos elementos. Entretanto, existem situações nas quais a filosofia de compartilhamento não se aplica.

Qualquer barramento pode ser enquadrado como pertencente a um dos dois tipos: dedicados ou não dedicados (partilhados). Um barramento dedicado é designado permanentemente a uma função ou a um par de dispositivos[62]. De acordo com esta definição o barramento de instruções de um computador da classe Harvard (figura 21) é dedicado fisicamente pois ele se destina exclusivamente a ligar o processador a uma memória. De acordo com a mesma definição ele também é dedicado funcionalmente na medida que somente instruções por ele transitam.O barramento de operandos, por sua vez, é dedicado funcionalmente aos operandos mas não é dedicado fisicamente pois existem múltiplos módulos de memória. Do ponto de vista do protocolo de

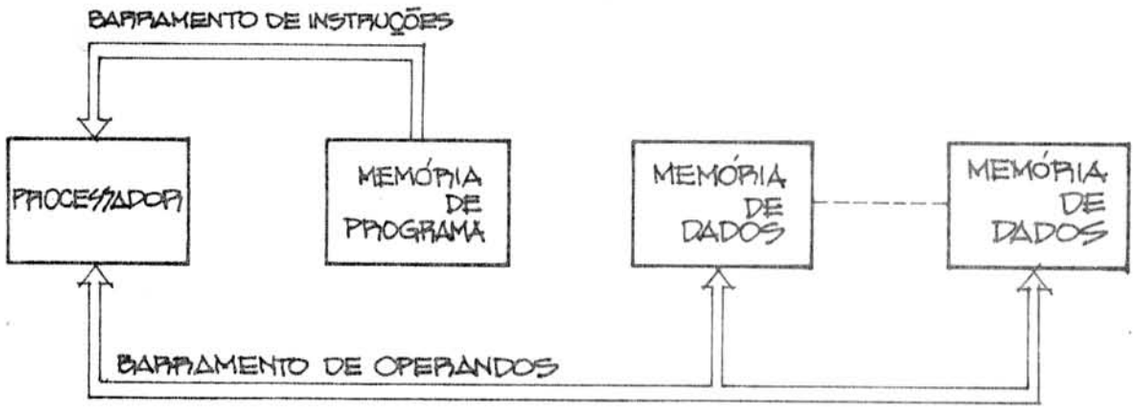


Figura 21 - Barramentos de um computador da classe Harvard

arbitração (vide seção 4.3), o número de módulos presentes não faz diferença uma vez que são elementos passivos, isto é, não iniciam um acesso ao barramento. A definição de Thurber et alii[62] desconsidera este fato, que, ao nosso ver, é relevante pois somente a existência de dois ou mais elementos ativos determina a necessidade de um árbitro[26,45], característica de barramentos não dedicados. Portanto, pode-se considerar um barramento fisicamente dedicado se existe apenas um elemento ativo, ou seja, um elemento capaz de requerê-lo. A diferença entre os dois conceitos reflete-se também ao nível do protocolo de comunicação. A primeira definição permite que a comunicação se efetue sem endereçamento explícito uma vez que só existem dois elementos conectados. Nossa definição exige endereçamento explícito em virtude da possível existência de múltiplas unidades passivas.

O conceito de dedicação funcional pode ser encarado de mais de uma maneira. Por exemplo, ambos os barramentos da figura 21 são não-dedicados se considerarmos que por eles tran

sitam endereços e dados. A dedicação funcional não afeta o protocolo de arbitração uma vez que não interessa a este o uso que será feito do barramento. Ao nível do protocolo de comunicação um barramento funcionalmente não-dedicado deve possuir informação de controle para indicar o tipo de informação contida a cada instante.

Para efeitos de alocação do barramento somente a decisão entre dedicados ou partilhados fisicamente tem influência.

4.2.1 Barramentos dedicados

Os barramentos dedicados oferecem vantagens e desvantagens. Entre as vantagens cita-se a disponibilidade constante. Não há problemas de contenção o que possibilita uma alta taxa de transferência entre os dispositivos. Uma outra vantagem é a desnecessidade de um árbitro uma vez que não há interferência.

A desvantagem dos barramentos dedicados reside no grande número de conectores e cabos necessários, principalmente se é exigida conectividade total entre os elementos. A modularidade é reduzida. Tomando-se como exemplo um sistema multiprocessado com múltiplos barramentos/memórias multiporta (figura 13), a adição de um processador requer n novos conectores e portas nos módulos, além de mais um barramento dedicado.

4.2.2 Barramentos partilhados

Quando existe mais de um elemento ativo ligado, o barramento é do tipo partilhado. Em cada instante só pode acontecer uma transferência. A priori, não há restrição quanto ao envio simultâneo a mais de um destino.

As vantagens oferecidas pelo partilhamento são as seguintes:

- a estrutura é modular tanto no aspecto custo como localização. Um novo elemento (processador, memória ou periférico) pode ser ligado em qualquer ponto do barramento bastando para isto mais um conector.

- baixo custo total pois todos os elementos possuem apenas uma interface.

Como desvantagens cita-se:

- o aumento da contenção, a medida que novos elementos ativos são adicionados. O barramento é o gargalo do sistema.

- a suscetibilidade a falhas. Um defeito no barramento pode bloquear todo o sistema. Esta dificuldade pode ser contornada duplicando-se o barramento. O preço pago pela redundância é o aumento do custo.

4.3 Protocolos de arbitração de barramento

4.3.1 Modelo

Um subsistema de comunicação pode ser dividido em

dois subníveis (seção 2.4.5). Um deles é responsável pela alocação da via e o outro pelo controle das transferências. Este conceito aplicado ao caso de barramentos resulta no Protocolo de Arbitração de Barramento e no Protocolo de Comunicação respectivamente. Esta seção descreve os tipos de arbitração comumente utilizados. As técnicas de comunicação empregadas após a alocação do barramento são assunto da seção 4.4.

Se o barramento é dedicado, o problema de alocação inexiste já que não ocorrem disputas pela sua utilização. Um único dispositivo é mestre e tem direito exclusivo sobre o mesmo. Porém, se o barramento é partilhado por dois ou mais dispositivos capazes de requisitá-lo assincronamente, podem ocorrer conflitos. O elemento responsável pela solução dos conflitos é o árbitro ou controlador de barramento. Antes de atuar sobre o barramento um dispositivo deve solicitar a autorização ao arbitro e aguardar a resposta afirmativa. A figura 22 mostra a concepção de um árbitro de conflito para dispositivos que geram requisições não críticas.

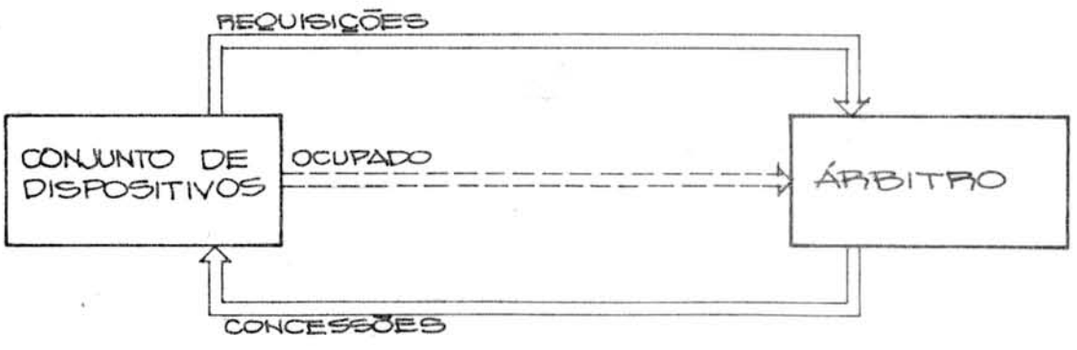


Figura 22 - Modelo de um árbitro de conflitos

Uma requisição não-crítica é aquela que não sofre restrições no intervalo de tempo entre a sua geração e a concessão[25]. Um dispositivo que necessita do barramento informa ao árbitro através de uma REQUISIÇÃO. O árbitro defere o pedido enviando uma CONCESSÃO. Se existe mais de uma requisição o árbitro escolhe apenas uma por meio de um mecanismo de prioridades nele implementado. As demais requisições permanecem aguardando sua vez. As requisições que chegam durante o intervalo de atendimento são registradas e também ficam aguardando.

A liberação do barramento é feita pelo próprio dispositivo que detém o controle (informando o árbitro que o barramento não está mais OCUPADO) ao final da transferência. Alternativamente, o próprio árbitro pode fazê-lo designando um tempo fixo de utilização para cada dispositivo, ao fim do qual uma nova seleção é executada. O barramento constitui-se num recurso de natureza elementar para o sistema. Ele tem que ser usado muitas vezes para se obter um resultado significativo. Por exemplo, uma instrução de alteração de dados na memória exige três acessos ao barramento: busca de instrução, busca de operandos, armazenamento do resultado. O fato de a utilização do barramento ser tão elementar exige que o mecanismo de arbitração seja extremamente veloz. Em geral, o processo de arbitração acontece em paralelo com a transferência corrente, ou então representa uma fração pequena do tempo de utilização[53].

4.3.2 Classificação dos árbitros

A localização do árbitro dentro do sistema permite

que se classifique-o como centralizado ou descentralizado. Ele é centralizado se os seus circuitos estão concentrados em um só ponto do sistema. Se, entretanto, cada elemento ativo contém uma parte dele, então o árbitro é do tipo descentralizado.

Existem, basicamente, três métodos de arbitração: em cadeia, por interrogação, ou por requisições independentes. Os três podem ser implementados de forma centralizada ou descentralizada. Um quarto método descentralizado é a auto-seleção[41].

4.3.3 Árbitros centralizados

4.3.3.1 Árbitro em cadeia centralizado

O sistema de arbitração centralizado em cadeia é um dos mais utilizados devido a sua simplicidade, reduzido número de linhas (quatro) e boa modularidade.

O diagrama em blocos do referido árbitro é visto na figura 23. O algoritmo de controle, tanto do árbitro como dos dispositivos está na figura 24.

Um dispositivo pede o barramento acionando a linha REQUISIÇÃO. Múltiplos pedidos podem coexistir pois ela é implementada com a técnica "wired-or". Ao receber um pedido, estando o barramento ocioso, o árbitro gera o sinal CONCESSÃO para o primeiro dispositivo na cadeia. Se ele não está requisitando o barramento, o sinal é passado ao seguinte, e assim sucessivamente até chegar a um dispositivo que necessite do barramento. Este, ao invés de continuar propagando-o, bloqueia SCONC, acio

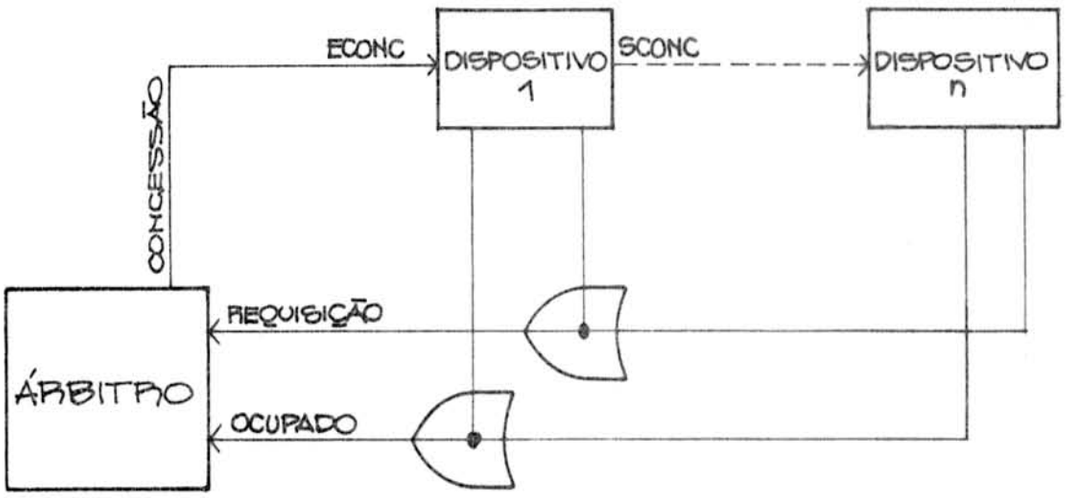


Figura 23 - Diagrama em blocos de um árbitro em cadeia centralizado

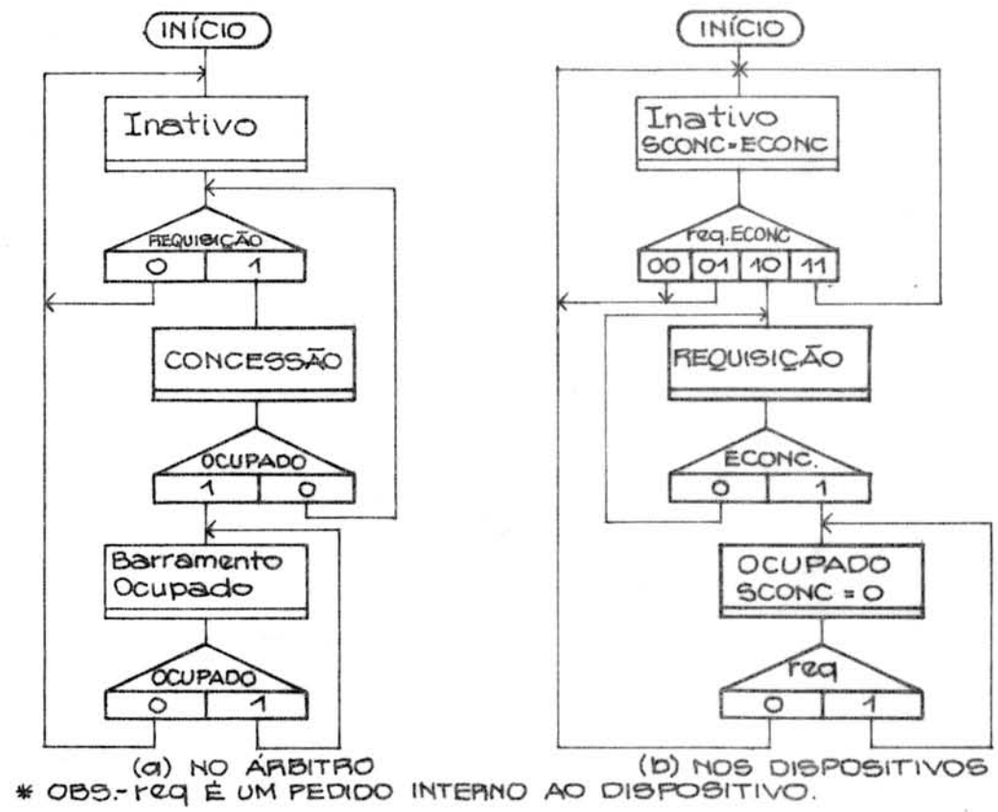


Figura 24 - Algoritmos da arbitração centralizada em cadeia

na a linha OCUPADO e desaciona REQUISIÇÃO. Desta forma o barramento lhe é assegurado e o controlador informado que a partir deste momento o barramento não está mais disponível. Ao encerrar a transmissão, o dispositivo desaciona a linha OCUPADO e o controlador volta ao estado inicial. Se a linha REQUISIÇÃO continua acionada o ciclo se repete.

Note-se que o dispositivo que dispara o processo não é necessariamente o selecionado. A prioridade de cada um na disputa pelo barramento é caracterizada pela posição física ocupada na cadeia. Quanto mais próximo do árbitro maior a prioridade. O esquema de prioridade fixa está entre os de mais fácil implementação, porém os dispositivos de menor prioridade podem ficar totalmente bloqueados devido à demanda dos de mais alta prioridade.

O árbitro em cadeia é altamente suscetível a falhas. Um defeito em qualquer uma das linhas paralisa o barramento. O mesmo ocorre se a cadeia é interrompida pela desativação de um dos dispositivos ou pane na fonte de alimentação de um deles. A retirada de um dispositivo implica o deslocamento de todos os outros abaixo deste para que a cadeia não se interrompa.

O número de dispositivos é teoricamente ilimitado, porém, na prática, ele é restrito devido ao atraso introduzido no sinal CONCESSÃO por cada interface. A medida que novos dispositivos são adicionados ele se torna mais lento.

A inclusão de mais uma linha permite que o processo de seleção seja feito em paralelo com a transmissão atual[53] (figura 25). A linha ACEITO desempenha a função que antes era

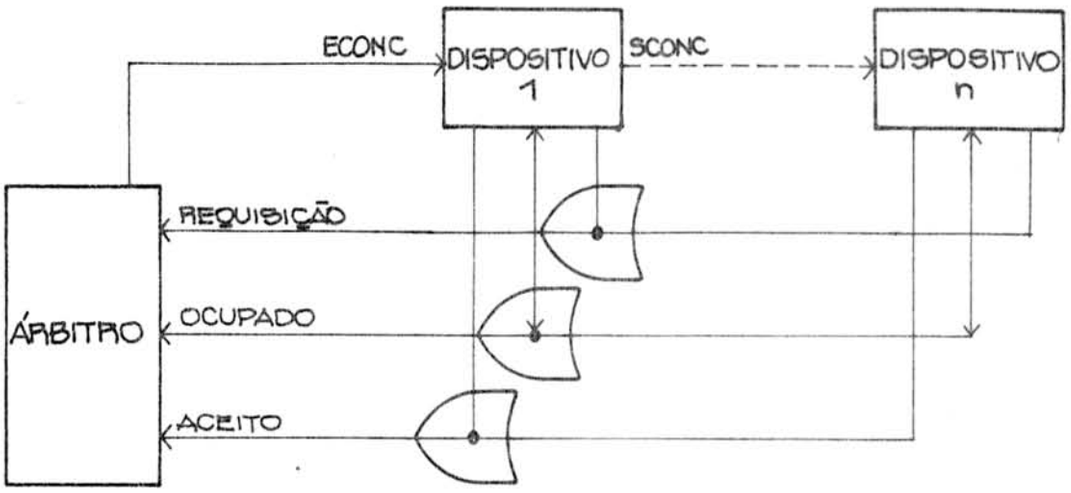


Figura 25 - Árbitro em cadeia centralizado

da linha OCUPADO. Esta indica apenas o fim do ciclo atual (figura 26).

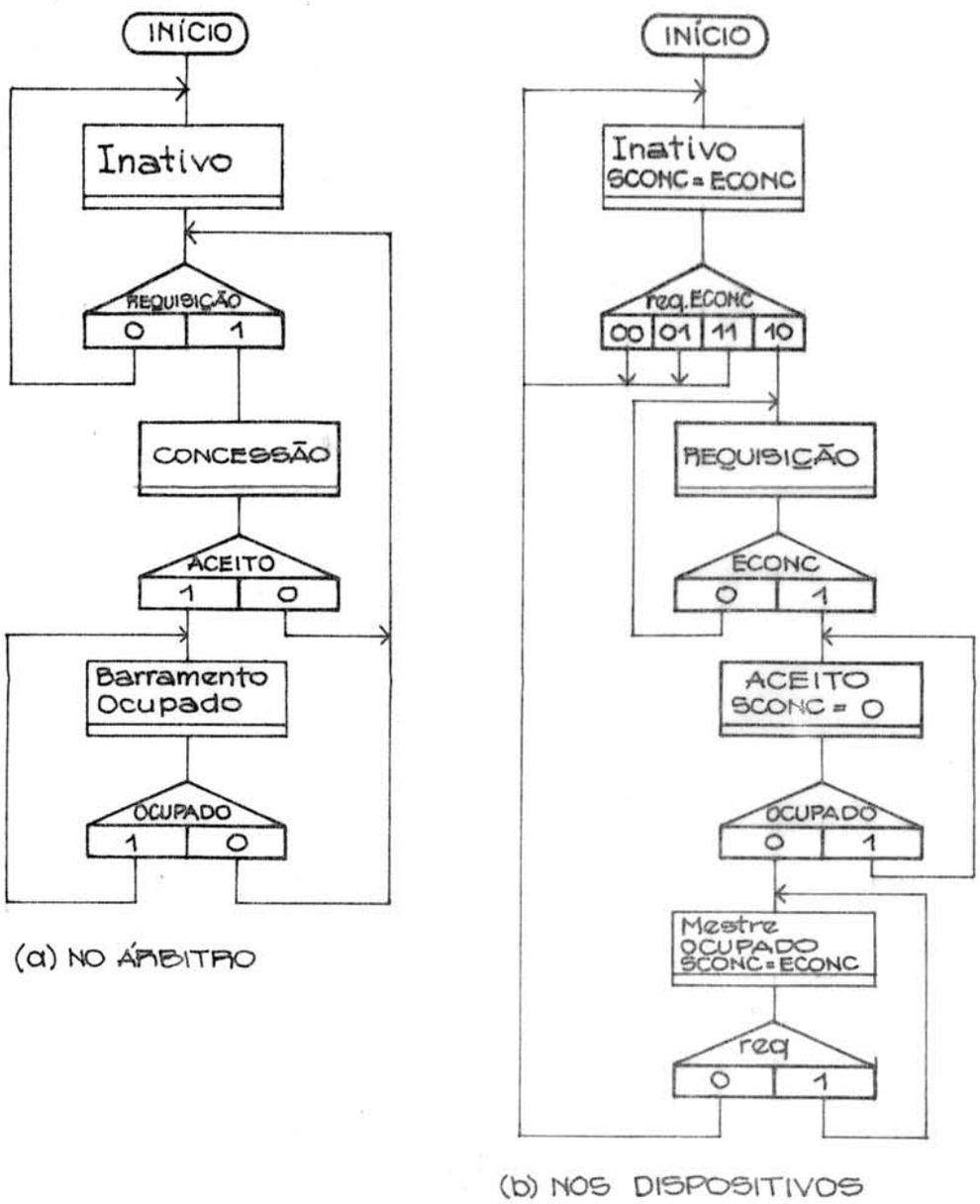


Figura 26 - Algoritmos da arbitração em cadeia centralizada

4.3.3.2 Árbitro por interrogação centralizado

Este tipo de árbitro, ao invés de enviar um sinal de concessão indistintamente ao grupo de dispositivos, interroga-os um a um procurando por quem fez o pedido. O diagrama em blocos deste esquema é mostrado na figura 27.

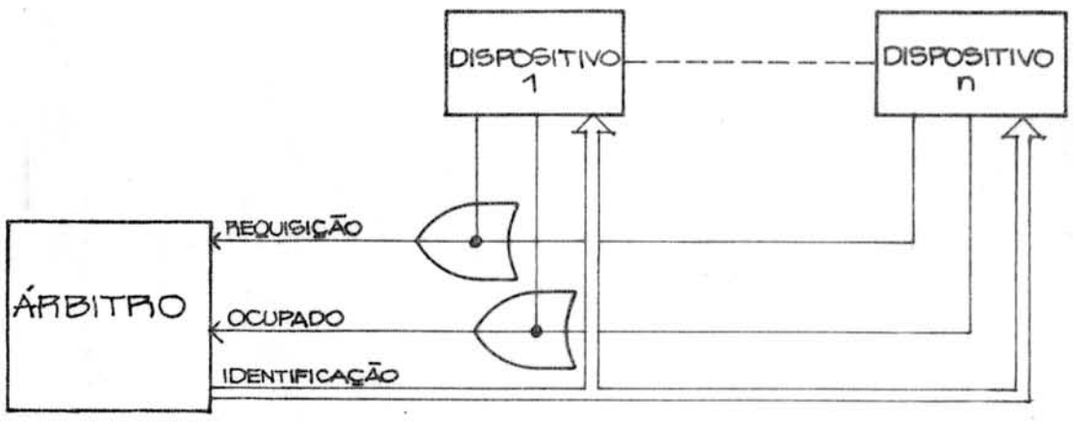


Figura 27 - Árbitro por interrogação centralizado

Ao sentir a linha REQUISIÇÃO ativada por um ou mais dispositivos, o controlador passa a interrogar um a um os dispositivos, enviando um código binário nas linhas IDENTIFICAÇÃO. Cada usuário é associado biunivocamente a um código, de maneira que a interrogação só é reconhecida por um deles a cada instante (ver figura 28).

O dispositivo que necessita do barramento aciona a linha OCUPADO ao reconhecer seu código de identificação. A ativação de OCUPADO força o árbitro a suspender a interrogação e ficar aguardando a desativação deste sinal a qual indica que a transferência foi concluída. Após, se ainda há uma requi

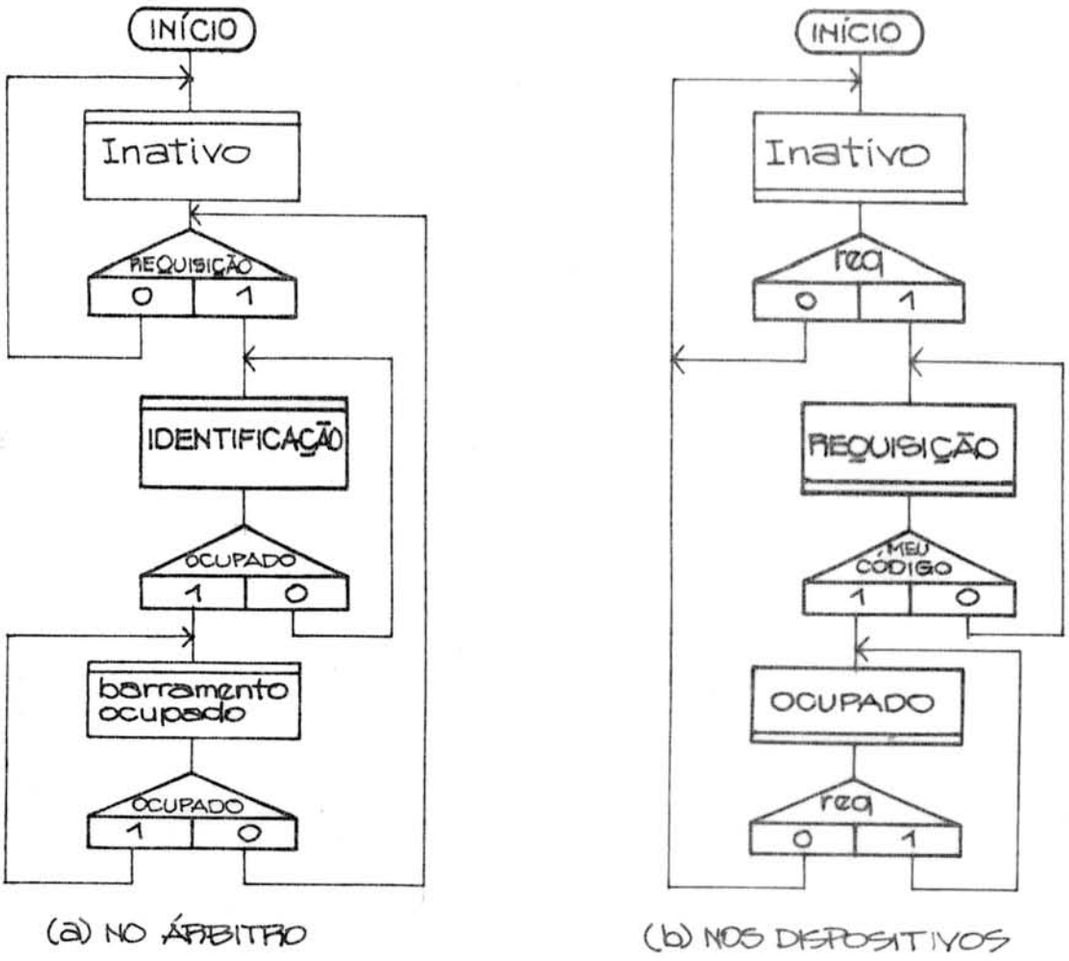


Figura 28 - Algoritmos de arbitração por interrogação centralizada

sição pendente, o árbitro retoma a interrogação, senão volta ao estado inativo.

O sistema de identificação é facilmente implementado através de um circuito contador. Deste modo é possível adotar um esquema de prioridade circular obrigando o contador a retomar a contagem do ponto onde havia parado. O problema do bloqueio dos dispositivos de menor prioridade, que ocorre em árbitros em cadeia, é resolvido com este sistema. A seqüência de interrogação pode ser feita variável em função do último dispositivo atendido usando-se uma memória de leitura apenas (ROM) ou um seqüenciador para implementá-la[16].

A suscetibilidade a falhas deste tipo de árbitro é menor que a do árbitro em cadeia pois não há sinais propagados em série entre os dispositivos. Pelo mesmo motivo torna-se mais fácil remover uma unidade sem alterar a posição das demais. Em compensação, o processo é mais lento e só pode ser implementado sincronamente. A modularidade fica de certa forma prejudicada pois o número máximo de dispositivos é limitado pelas linhas IDENTIFICAÇÃO.

As linhas de identificação podem ser dispensadas dotando-se os dispositivos com contadores. Um relógio único, centralizado no árbitro, atua sobre todos os contadores (figura 29). O sinal de relógio só é bloqueado quando um dispositivo aciona a linha OCUPADO. O ponto crítico passa a ser o relógio. A necessidade absoluta de manter os contadores sincronizados torna o sistema suscetível a ruído. Se os contadores perdem

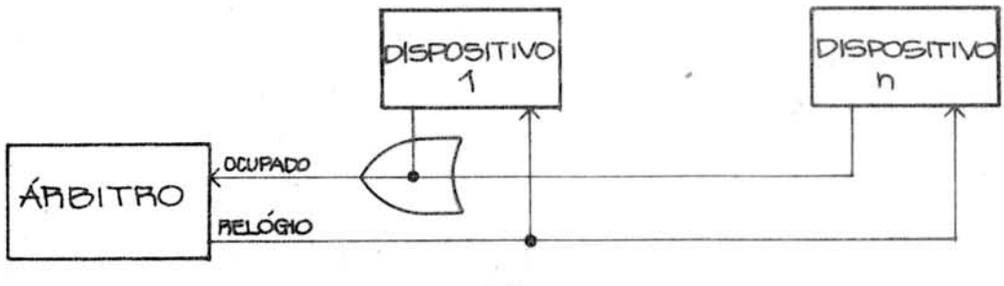


Figura 29 - Árbitro interrogador centralizado com contadores independentes

o sincronismo de contagem ocorre interferência no barramento. Para que isto não aconteça o escorregamento do relógio deve ser mínimo.

O sistema fica limitado a aplicações de curta distância. Em compensação, sua modularidade é melhorada em comparação ao sistema em cadeia pois o número de dispositivos só depende do valor máximo de contagem.

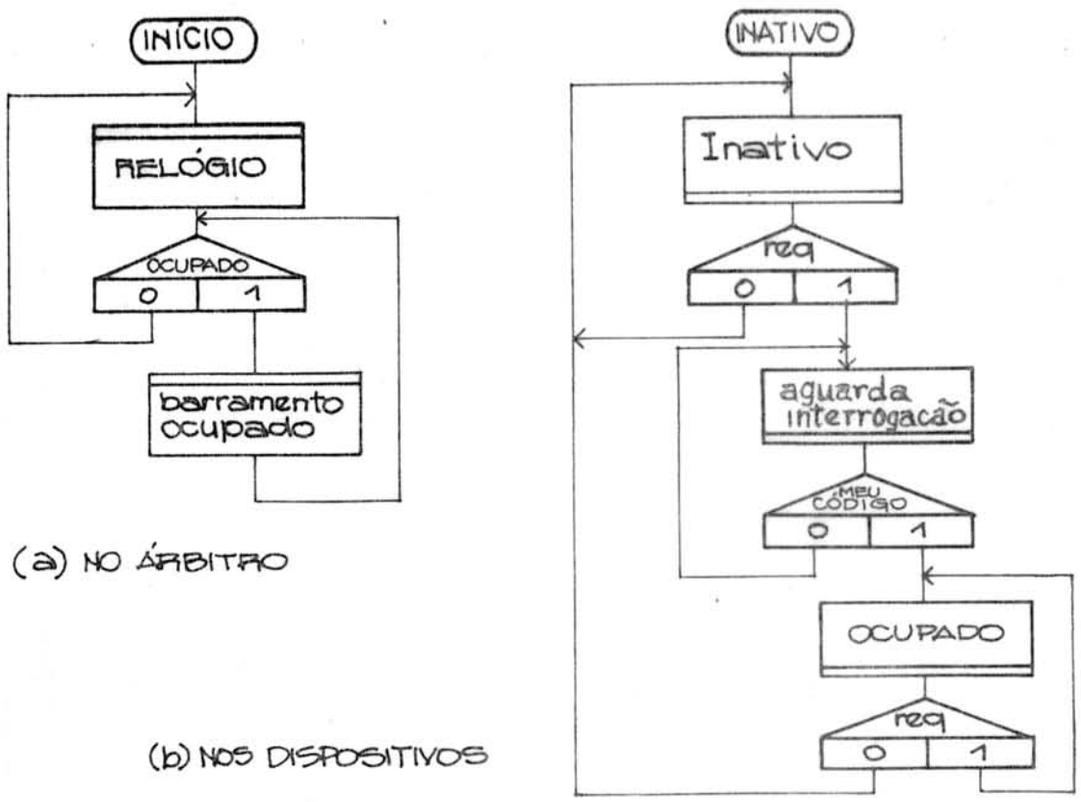


Figura 30 - Algoritmos de arbitração por interrogação centralizada

4.3.3.3 Árbitro de requisições independentes centralizado

O árbitro de requisições independentes (figura 31) é também chamado de árbitro paralelo porque, ao contrário do árbitro em cadeia (série), possui múltiplas linhas de requisição e concessão.

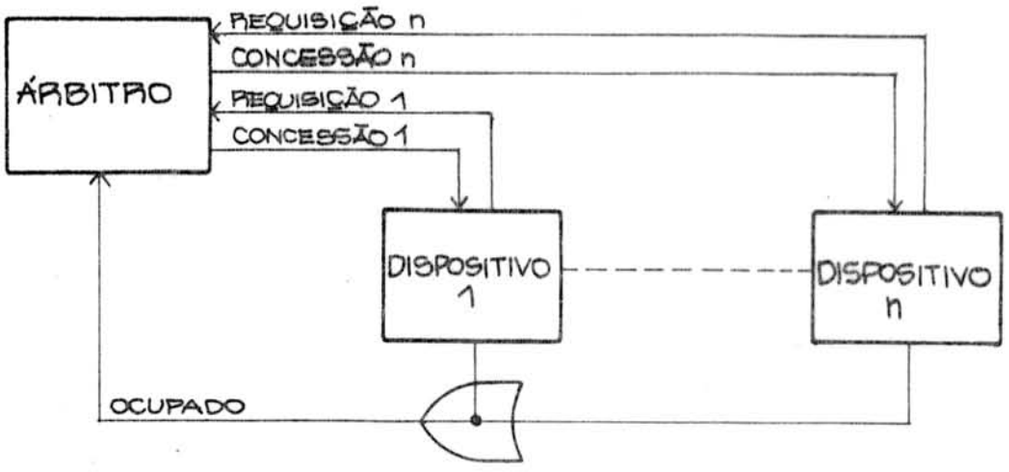


Figura 31 - Árbitro de requisições independentes centralizado

Cada dispositivo tem uma linha privativa de requisição e outra de concessão dirigidas ao árbitro. Uma terceira linha, de uso comum, informa o árbitro se o barramento está ou não disponível. Quando a linha OCUPADO é desacionada o controlador compara todas as requisições presentes e seleciona a de mais alta prioridade para atendimento; somente esta recebe o sinal CONCESSÃO. O recebimento do sinal CONCESSÃO causa o desacionamento da REQUISIÇÃO e o acionamento de OCUPADO. O processo repetir-se-á quando o dispositivo ao qual foi concedido o barramento desacionar a linha OCUPADO. O algoritmo (figura 32) permite que as requisições sejam feitas a qualquer instante e em paralelo.

Uma das vantagens deste tipo de árbitro é a rapidez. Ele é o mais veloz dos apresentados até aqui. Uma outra vantagem oferecida é a possibilidade de escolher qualquer tipo de prioridade[26].

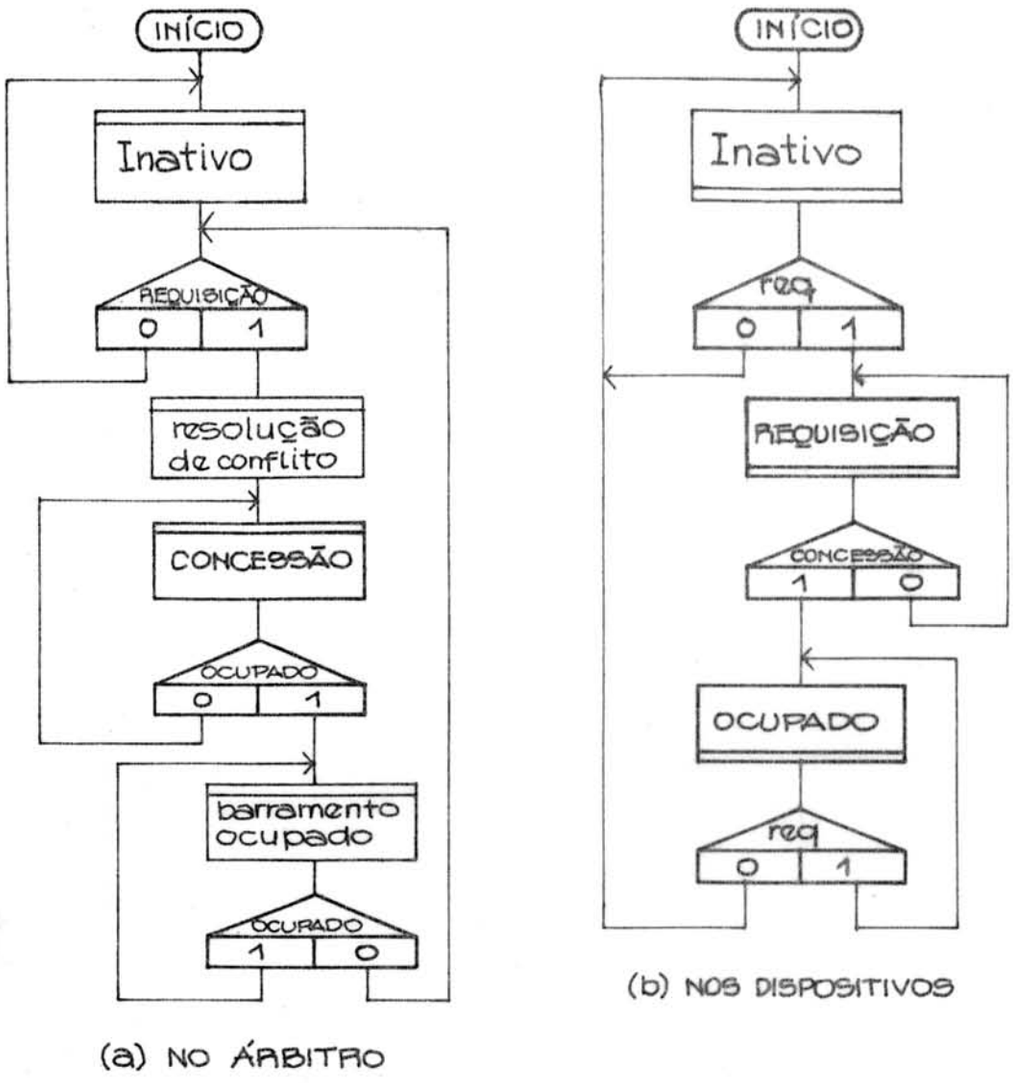


Figura 32 - Algoritmos de arbitração centralizada de requisições independentes

O preço pago por estas vantagens é o custo das linhas dedicadas. A modularidade é limitada pelo número de linhas de requisição e concessão disponíveis.

Exemplos de implementação de árbitros deste tipo são dados nas referências 16, 26, 27, 28, 45 e 51.

4.3.4 Árbitros descentralizados

Os árbitros descentralizados não são localizados num ponto específico do sistema, mas sim nas interfaces dos dispositivos. Portanto, o que caracteriza os árbitros descentralizados é a interação íntima entre as unidades na disputa pelo barramento.

4.3.4.1 Árbitro em cadeia descentralizado

O diagrama em blocos de um árbitro em cadeia descentralizado (figura 33) ilustra a diferença deste para a versão centralizada (figura 23). Neste, não existe a linha OCUPADO. A linha requisição continua a existir, porém ela retorna ao primeiro dispositivo da cadeia desempenhando o papel da li-

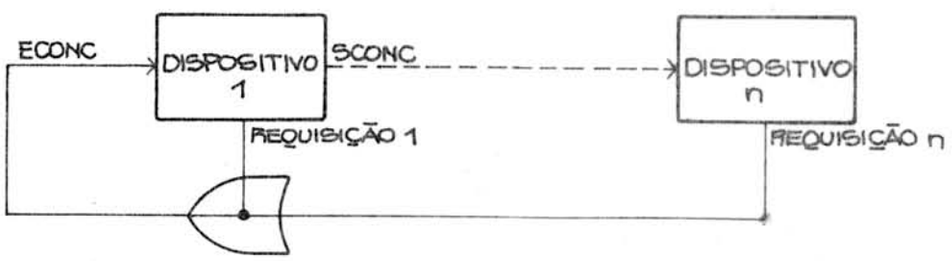


Figura 33 - Árbitro em cadeia descentralizado

nha de concessão (ECONC da figura). Este é o ponto de referência que marca o início da cadeia. Um dispositivo só deve acionar a linha REQUISIÇÃO se a sua entrada ECONC está desativada.

Uma ou mais requisições sentidas pelo primeiro dispositivo fazem-no propagar o sinal de concessão (SCONC) ao seguinte, caso ele próprio não esteja requerendo o barramento. Ao receber o sinal de concessão, um dispositivo que está requerendo o barramento bloqueia a propagação do sinal SCONC e torna-se o novo mestre de barramento. A posse do barramento lhe é assegurada enquanto a concessão não é propagada adiante (ver figura 34). Ao fim da utilização, o sinal REQUISIÇÃO é desacionado e a concessão propagada para os próximos dispositivos. Neste instante, somente os dispositivos colocados adiante na ca-

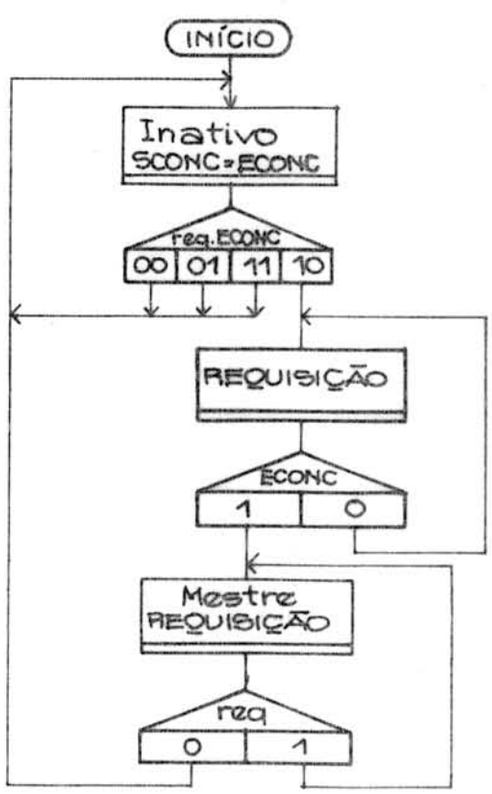


Figura 34 - Algoritmo do árbitro em cadeia descentralizado

deia têm autorização para requerer o barramento. Desta maneira, um esquema de prioridade circular é obtido. Os elementos posteriores ao último atendido detêm sempre a mais alta prioridade no instante em que o barramento se torna disponível. Entretanto, se nenhum deles deseja utilizá-lo, a prioridade máxima volta ao primeiro dispositivo.

O árbitro em cadeia descentralizado sofre dos mesmos problemas que o centralizado, porém, com duas exceções:ele economiza uma linha (OCUPADO) e não é limitado a prioridade fixa. Sua lentidão é agravada pelo problema de corrida("race"). No instante (vide figura 35) em que o dispositivo i do inte-

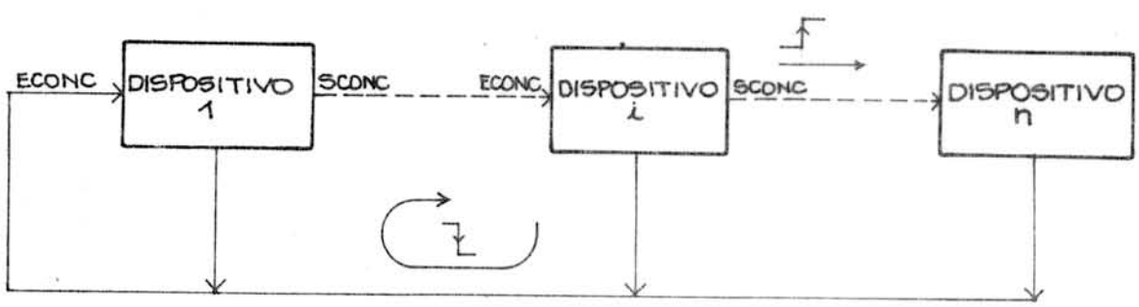


Figura 35 - Situação de "race" no árbitro em cadeia descentralizado

rior da cadeia libera o barramento e nenhum outro o está requisitando, a concessão é desativada na entrada do primeiro dispositivo e é propagada para a frente; ao mesmo tempo o dispositivo i propaga a concessão ativa. Se, neste instante, acontecerem duas requisições simultâneas, uma à esquerda e outra à direita de i, ambas ganharão acesso ao barramento. Para evitar este problema, a captura do barramento deve ser atrasada o tem

po suficiente para que os sinais estabilizem. Isto implica aumentar a lentidão do sistema, além do acréscimo de complexidade.

Um árbitro em cadeia descentralizado pode ser construído de uma forma um pouco diferente do primeiro (figura 36).

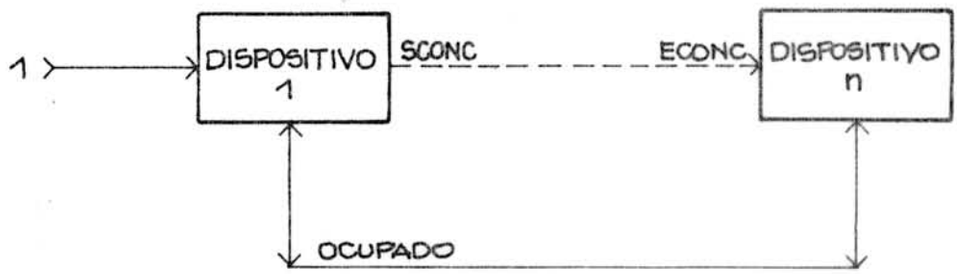


Figura 36 - Árbitro em cadeia descentralizado (Alternativa 1)

Neste, o mesmo número de linhas se faz necessário, porém, a informação contida na linha de uso comum é o estado do barramento (ocupado ou não). Ela é uma linha bidirecional. Os usuários em potencial monitoram o estado desta linha para se apossarem do barramento e ativam-na para tornarem-se mestres.

A linha SCONC informa se algum dispositivo de mais alta prioridade está requisitando o barramento. O algoritmo de controle (figura 37), descrito a seguir, é executado por todos os dispositivos coordenados por um mesmo relógio. Enquanto o controlador está inativo (não há requisição interna) o sinal de concessão é simplesmente passado adiante. Ao entrar no esta

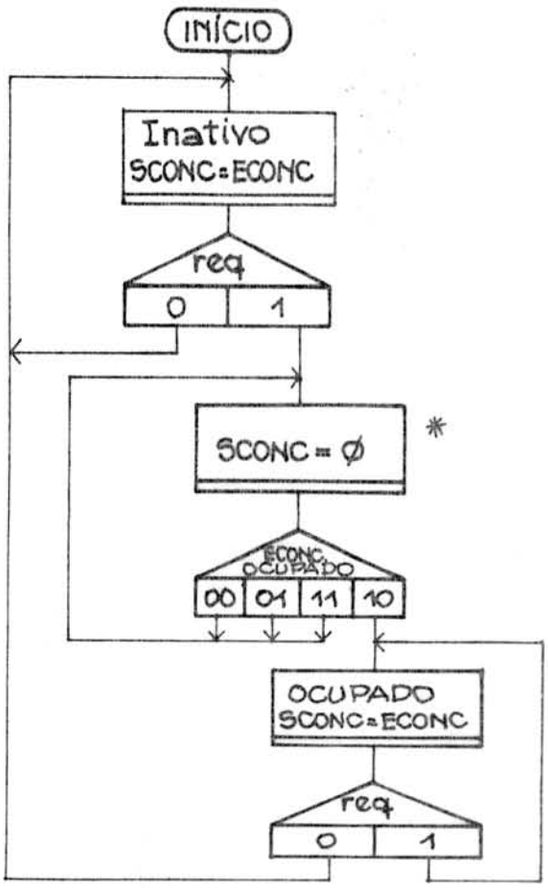


Figura 37 - Algoritmo do árbitro em cadeia descentralizado (Alternativa 1)

do ativo por força de uma requisição interna a concessão é desativada na saída o que impede um dispositivo de menor prioridade apossar-se do barramento. Para passar à condição de mestre o controlador espera que a linha OCUPADO seja desativada e que sua entrada ECONC esteja ativa. Neste estado o controlador não participa do processo de seleção do próximo usuário o qual ocorre em paralelo com a transferência atual.

Um terceiro tipo de árbitro em cadeia descentralizado é mostrado na figura 38.

A concessão de barramento é passada através de uma transição na linha entre dois vizinhos. Um dispositivo que

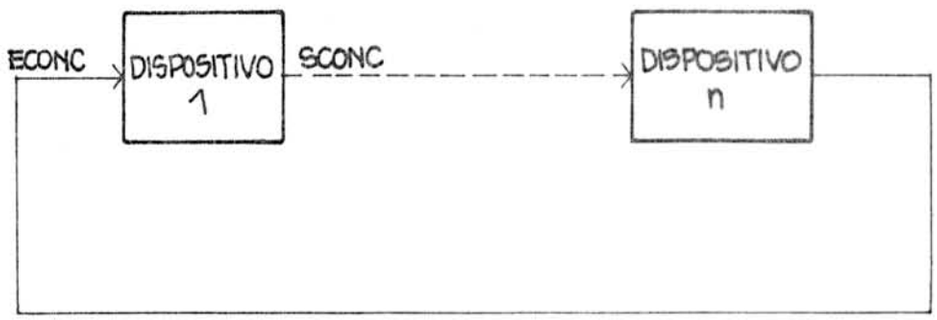


Figura 38 - Árbitro em cadeia descentralizado (Alternativa 2)

não necessita do barramento, ao sentir sua entrada ECONC mudar de estado, força sua saída SCONC a também mudar de estado. Se, entretanto, ele necessita do barramento a transição na saída é bloqueada e ele passa a ser o mestre. Ao concluir a transferência, a concessão é passada adiante normalmente. Este é um esquema de prioridade circular.

A vantagem oferecida em relação ao anterior é a economia de uma linha. A desvantagem é a possibilidade de um dispositivo receber uma falsa concessão provocada por sinais espúrios na linha.

O algoritmo (figura 39) é idêntico para todos os controladores exceto para o último da cadeia no qual a saída SCONC é invertida. Se ninguém deseja utilizar o barramento a concessão fica oscilando no laço.

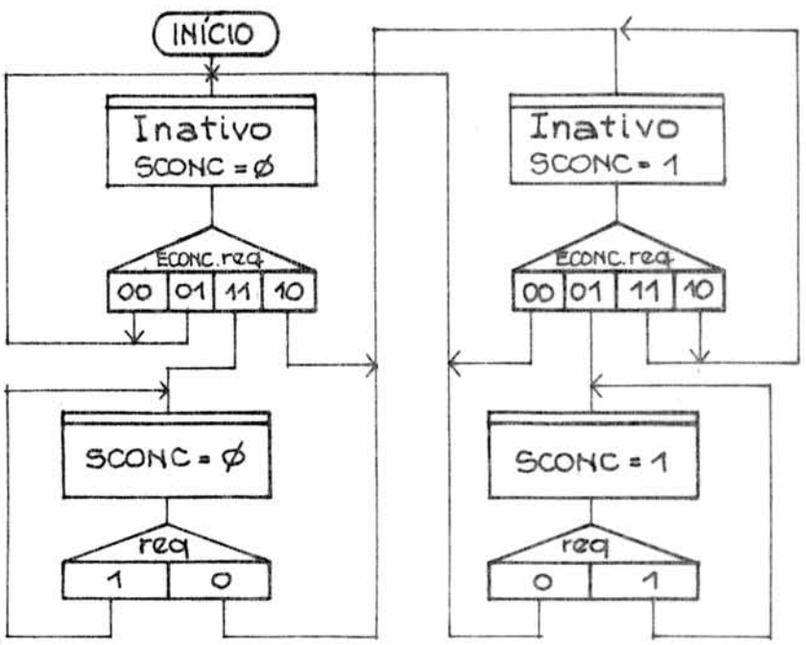


Figura 39 - Algoritmo do árbitro em cadeia descentralizado (Alternativa 2)

4.3.4.2 Árbitro por interrogação descentralizado

O diagrama em blocos de um árbitro interrogador descentralizado é visto na figura 40.

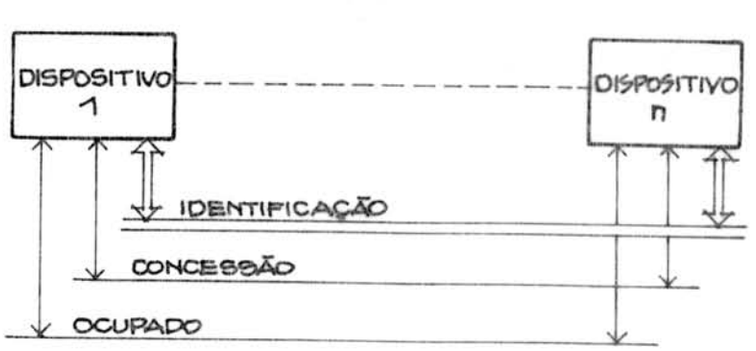


Figura 40 - Árbitro interrogador descentralizado

Analogamente à versão centralizada, todo dispositivo possui um código identificador que o distingue dos demais. Ao encerrar a transmissão um dispositivo interroga um a um os demais procurando por alguém que necessita do barramento. Isto é feito colocando-se um código nas linhas IDENTIFICAÇÃO e ativando a linha CONCESSÃO. O dispositivo endereçado ativa ou não a linha OCUPADO dependendo de sua necessidade. O interrogador, ao perceber a ativação do sinal OCUPADO, retira-se do processo de seleção pois já há um novo mestre. Se, entretanto, não é recebida resposta (OCUPADO não ativo) o interrogador repete o procedimento endereçando um após o outro todos os dispositivos até encontrar um que queira utilizá-lo. Quando o sistema é ligado apenas um dispositivo deve receber o controle do barramento.

Este árbitro aceita qualquer tipo de prioridade. A maior desvantagem é a necessidade de dotar cada dispositivo com um circuito de arbitração completo. Com prioridade fixa pode-se usar um circuito contador para gerar os códigos de identificação, o qual é reinicializado em zero ("reset") antes de iniciar-se o processo de interrogação. Usando-se prioridade circular, cada controlador deve carregar o seu contador com um valor diferente dos demais. Esquemas de prioridade mais flexíveis exigem circuitos mais complexos ou uma memória, fazendo com que o seu custo aumente.

Uma outra desvantagem do árbitro interrogador é o fato de ser lento pois uma série de interrogações inócuas podem acontecer antes de ser cedido o barramento. Entretanto, sua confiabilidade é maior que a dos anteriores, pois uma falha em um dispositivo não bloqueia necessariamente o sistema.

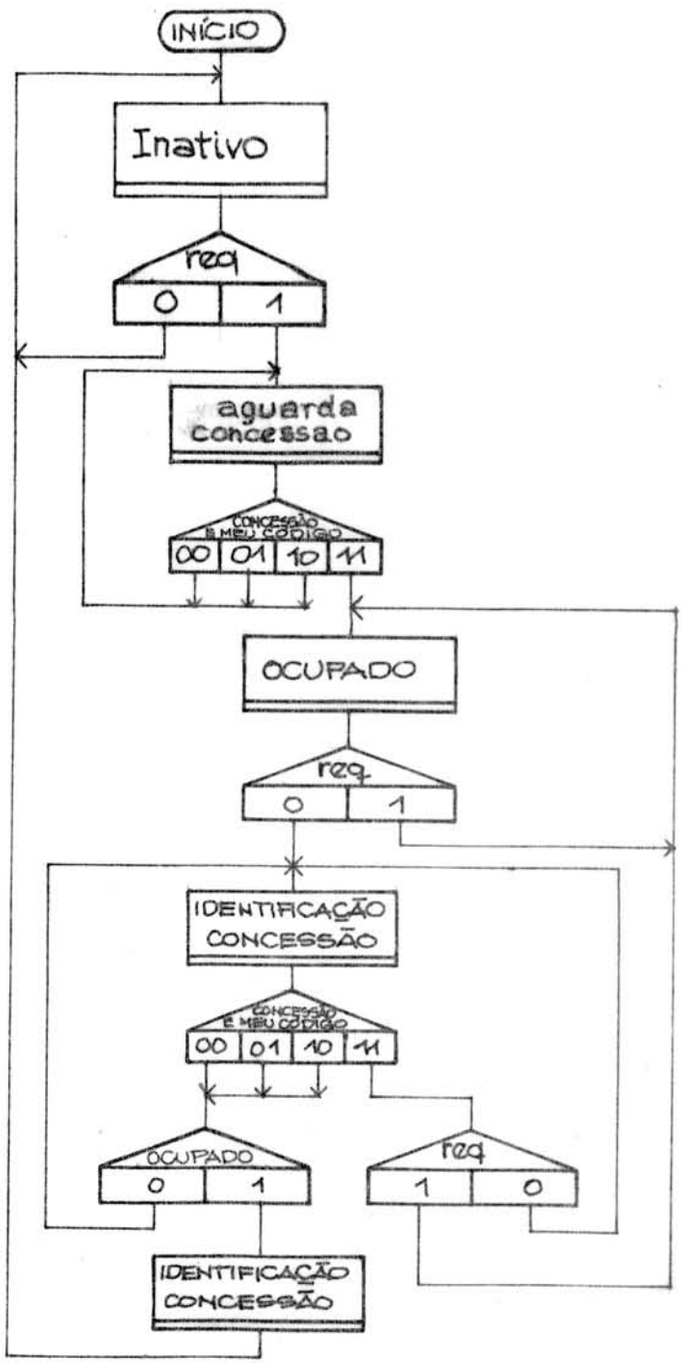


Figura 41 - Algoritmo do árbitro interrogador descentralizado

4.3.4.3 Árbitro de requisições independentes descentralizado

Em um árbitro de requisições independentes descentralizado cada elemento possui uma linha de requisição privada que contém implicitamente a sua prioridade (figura 42).

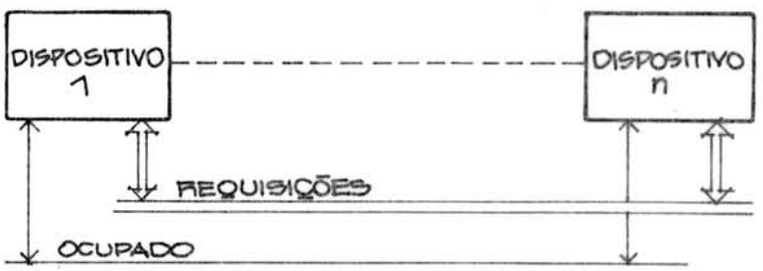


Figura 42 - Árbitro de requisições independentes descentralizado

A desativação da linha OCUPADO indica a todos os usuários que o mestre atual está liberando o barramento. Isto faz com que todos examinem as linhas de requisição verificando se a sua é a de mais alta prioridade. Aquele que efetivamente possui a maior prioridade aciona a linha OCUPADO e passa a ser o mestre. Os demais retiram suas requisições e aguardam a próxima liberação para uma nova tentativa (figura 43).

A concepção funcional é bastante simples o que significa que ele pode ser implementado com uma pequena quantidade de hardware. Além desta, uma outra vantagem é a possibilidade de se escolher entre prioridade fixa ou circular. O ponto desfavorável deste tipo de árbitro é o grande número de linhas

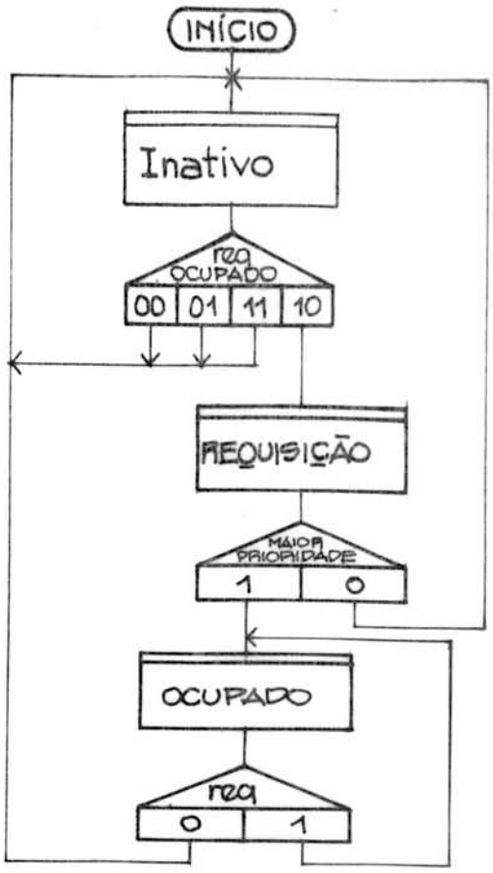


Figura 43 - Algoritmo do árbitro de requisições independentes descentralizado

necessário. Também há a necessidade de sincronização absoluta o que restringe sua aplicação a pequenas distâncias.

4.3.4.4 Árbitro por autoseleção descentralizado

Um quarto método descentralizado de alocação[41] através de linhas codificadas (análogo à interrogação) é a autoseleção (figura 44). Neste sistema, cada dispositivo possui um código binário designado para si o qual caracteriza sua

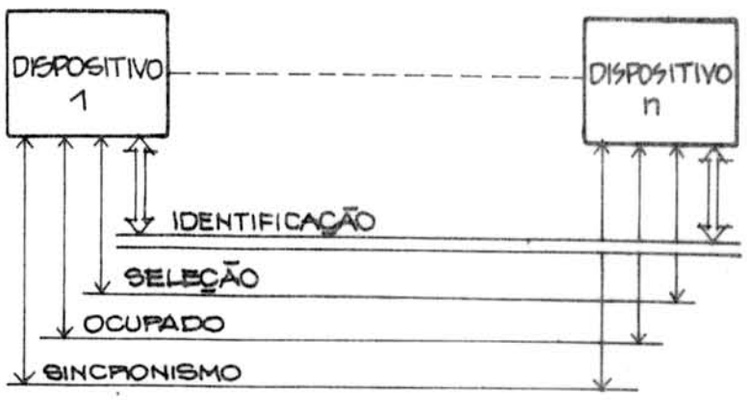


Figura 44 - Árbitro auto-selecionador

prioridade em relação aos demais. Quanto mais alto o código, maior a prioridade. O processo de arbitração (figura 45) inicia com o sinal SELEÇÃO ativado por um ou mais dispositivos que queiram utilizar o barramento. O acionamento deste sinal faz com que estes dispositivos coloquem os seus códigos de prioridade no barramento de identificação. A cada pulso de relógio subseqüente os árbitros examinam as linhas IDENTIFICAÇÃO uma a uma iniciando pela mais significativa. Este exame consiste na comparação de um bit do código de identificação com o respectivo bit de seu código de prioridade. Se o bit no barramento é maior que o de seu código isto significa que há alguém com prioridade mais alta participando da arbitração. Em resposta o dispositivo se retira do processo, retirando o seu código do barramento. Caso o bit do barramento seja igual ou menor que o de seu código, o teste continua até o bit menos significativo. Apenas o dispositivo de mais alta prioridade sobrevive a este procedimento e se torna o novo mestre.

A comparação deve ser efetivada em todos os árbitros no mesmo instante. Por esta razão, um único relógio central deve manter os controladores sincronizados. O sinal SINCRONISMO indica aos requisitantes o instante exato de iniciar o processo. A partir daí todos estão sincronizados. A lentidão do processo é absorvida executando-o em paralelo com a transferência corrente. Neste caso o mestre atual do barramento é responsável por disparar o processo de arbitração num determinado momento de seu ciclo. O sinal SINCRONISMO pode ser um dos sinais de controle da transferência, como, por exemplo, o que sinaliza dados válidos.

Uma vez adquirido o barramento, o dispositivo aguarda o fim do ciclo atual para executar sua transferência. Enquanto o processo de arbitração não é iniciado pode-se tirar partido das linhas de identificação para impedir o acesso a determinadas áreas de memória ou periféricos por parte do usuário corrente.

O esquema de prioridade é fixo porém impede que o barramento seja concedido a um mesmo dispositivo duas vezes consecutivas, fazendo com que o usuário corrente não participe do processo de arbitração. O número de linhas é o mesmo que o de um árbitro interrogador descentralizado (o sinal SINCRONISMO não é de uso exclusivo do árbitro).

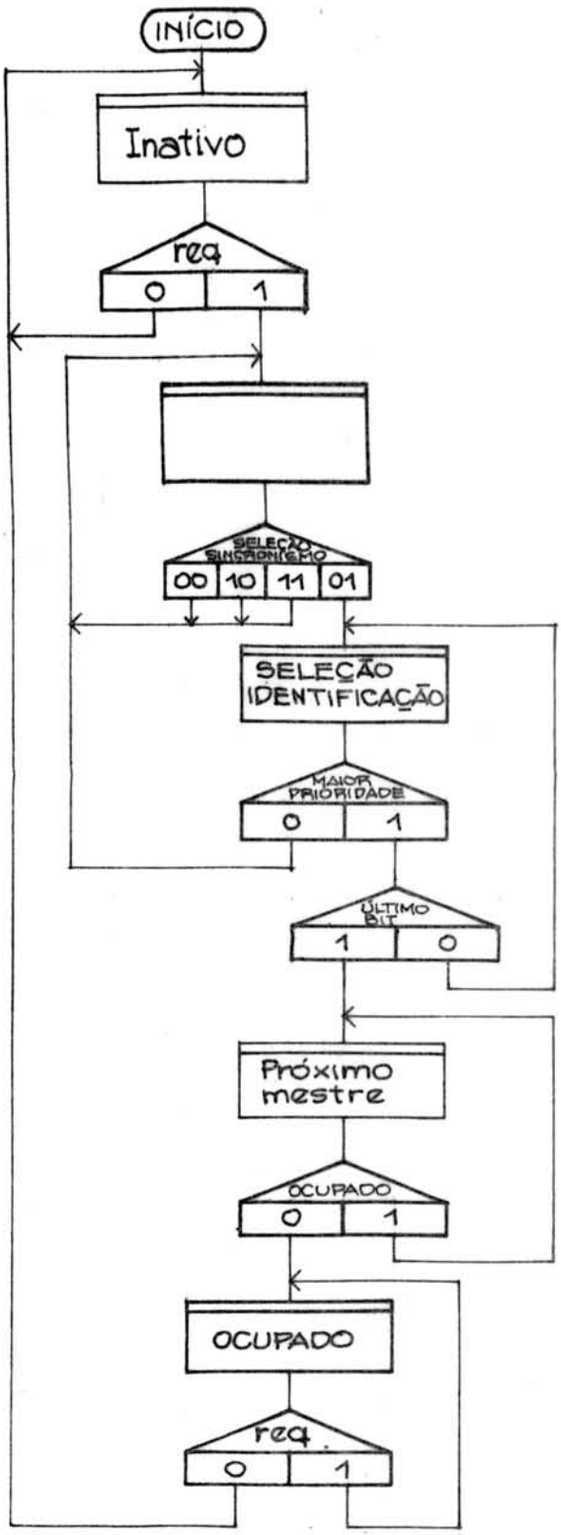


Figura 45 - Algoritmo do árbitro auto-selecionador

4.4 Protocolos de comunicação

O primeiro passo no sentido de estabelecer uma conexão entre dispositivos consiste na disputa pela via (protocolo de arbitração). Uma vez obtida a concessão, têm início os processos relacionados com a transferência da informação. Os protocolos de comunicação constituem o nível mais baixo do modelo da figura 8.

4.4.1 Técnicas de transferência

A técnica ou método pelo qual a informação é transferida através da via é um fator muito importante pelas implicações sobre a performance do sistema. Distinguem-se dois casos: comunicação síncrona ou assíncrona[66].

4.4.1.1 Comunicação síncrona

Na técnica de comunicação síncrona cada item de informação é transferido durante um período de tempo fixo conhecido tanto pelo dispositivo fonte como pelo destino. Isto exige que ambos estejam sincronizados. A sincronização pode ser obtida através de um único relógio central comum a todos os dispositivos. Este método é plausível quando o barramento alcança pequenas distâncias. Barramentos longos impedem o uso deste método devido aos problemas de atraso e de escorregamento entre os sinais. Por exemplo, um sinal de relógio com perío

do de 100 ns transmitido a um dispositivo distante 30 m chegará ao destino 100 ns depois, ou seja, com um período de atraso. O escorregamento, por sua vez, é provocado por pequenas diferenças de tempo de propagação entre as linhas. Por exemplo, se duas linhas têm um atraso que difere em 10%, dois sinais transmitidos ao mesmo tempo chegarão ao destino defasados de 10 ns (d = 30 m). Uma forma alternativa de sincronização consiste em dotar cada unidade com um relógio próprio, todos eles de mesma freqüência. A sincronização é obtida através de sinais periódicos transmitidos juntamente com os dados.

Em um barramento encontra-se, via de regra, dispositivos de diferentes velocidades. A desvantagem maior dos sistemas síncronos advém do fato de a janela de tempo usada para transmissão ter sua largura determinada pelo dispositivo mais lento[63]. Conseqüentemente, os dispositivos mais rápidos não podem operar à sua velocidade máxima. Se a janela é determinada em função do dispositivo mais rápido, todos os outros devem ser providos de "buffers" de armazenamento temporário. A escolha entre um ou outro método é um compromisso entre velocidade e custo.

4.4.1.2 Comunicação assíncrona

A técnica de comunicação assíncrona é largamente utilizada pela sua flexibilidade. Ao invés de ser transferido dentro de um tempo fixo conhecido pelo fonte e destinatário, cada item de informação é acompanhado por sinais de controle que indicam sua presença no barramento. Existem duas filoso-

fias de comunicação assíncrona, em função de quem toma parte no controle da transferência. Se apenas um dos dispositivos é encarregado desta tarefa o controle é dito unidirecional. Se tanto o dispositivo fonte como o destino tomam parte no controle, então ele é do tipo bidirecional. Os termos unidirecional e bidirecional referem-se à existência de sinais de controle em um sentido ou ambos.

No controle unidirecional distinguem-se dois casos: comandado pelo dispositivo fonte ou comandado pelo destinatário. No primeiro caso (figura 46) o dispositivo fonte coloca os dados no barramento e sinaliza DADOS PRONTOS ao destinatários. Este, ao receber o sinal DADOS PRONTOS, copia os da-

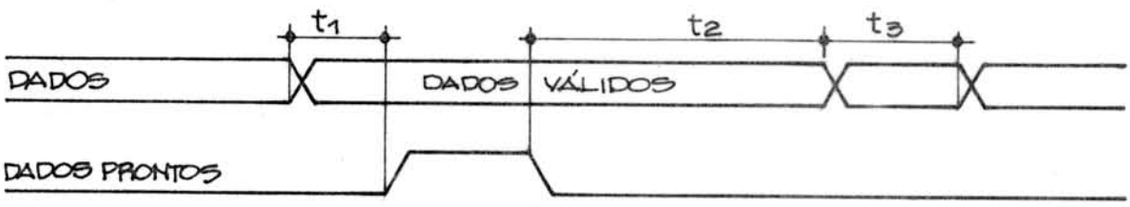


Figura 46 - Protocolo unidirecional controlado pelo dispositivo fonte

dos do barramento. Se a cópia é feita diretamente pelo sinal proveniente do fonte deve-se prover um atraso (t_1) para garantir que os dados estejam estáveis no instante da cópia.

Uma outra possibilidade é manter o sinal DADOS PRONTOS durante todo o tempo em que os dados permanecem estáveis no barramento ($t_1 = t_2 = 0$) e deixar por conta do destinatário a decisão quanto ao momento da cópia. Este protocolo é

inerentemente simples pois só utiliza um sinal de controle. É também veloz pois só envolve um tempo de propagação pelo barramento. Todavia, ele não se presta a comunicação entre dispositivos de diferentes velocidades pela falta de um sinal que indique que o destinatário recebeu os dados. Para evitar que uma nova transmissão seja iniciada antes de ser concluída a primeira deve-se prover um tempo (t_3) o qual garante a integridade da comunicação. Este tempo é determinado pelos dispositivos mais lentos. Portanto, a deficiência é a mesma dos barramentos síncronos. Não há possibilidade de retransmissão em caso de erro pela ausência de um sinal apropriado. A confiabilidade é prejudicada pelo problema de sinais espúrios na linha DADOS PRONTOS serem confundidos com o sinal verdadeiro.

Quando o controle da transferência é assumido totalmente pelo destinatário (figura 47), este requisita um dado ativando a linha REQUISIÇÃO. O dispositivo fonte responde colocando os dados no barramento. Este protocolo também não se presta à comunicação entre dispositivos de diferentes velocidades pois o destinatário não tem meios de saber se o fonte está apto a enviar os dados. O tempo entre duas requisições (t_1) deve

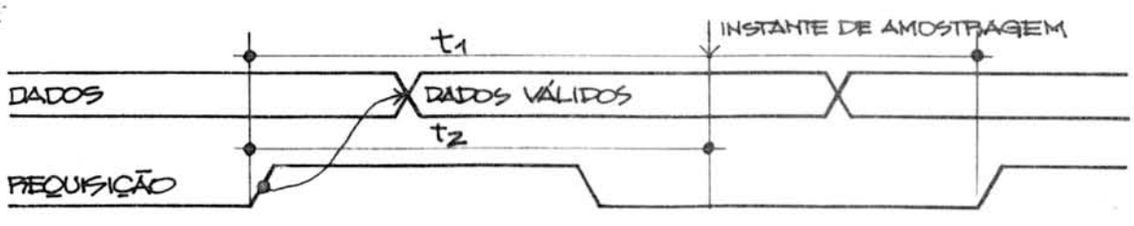


Figura 47 - Protocolo unidirecional controlado pelo destinatário

ser maior que o tempo de resposta do dispositivo mais lento. Além disso, o destinatário deve saber quando amostrar os dados (t_2) o que depende da distância entre os dispositivos.

O controle através do destinatário é mais lento que o controlado pelo fonte pois envolve dois tempos de propagação pelo barramento.

A comunicação entre dispositivos de diferentes velocidades é executada mais eficientemente por um protocolo bidirecional. Distinguem-se três casos distintos de protocolos bidirecionais: não-entrelaçado, semi-entrelaçado e entrelaçado.

A figura 48 mostra o diagrama de tempos do método não-entrelaçado.

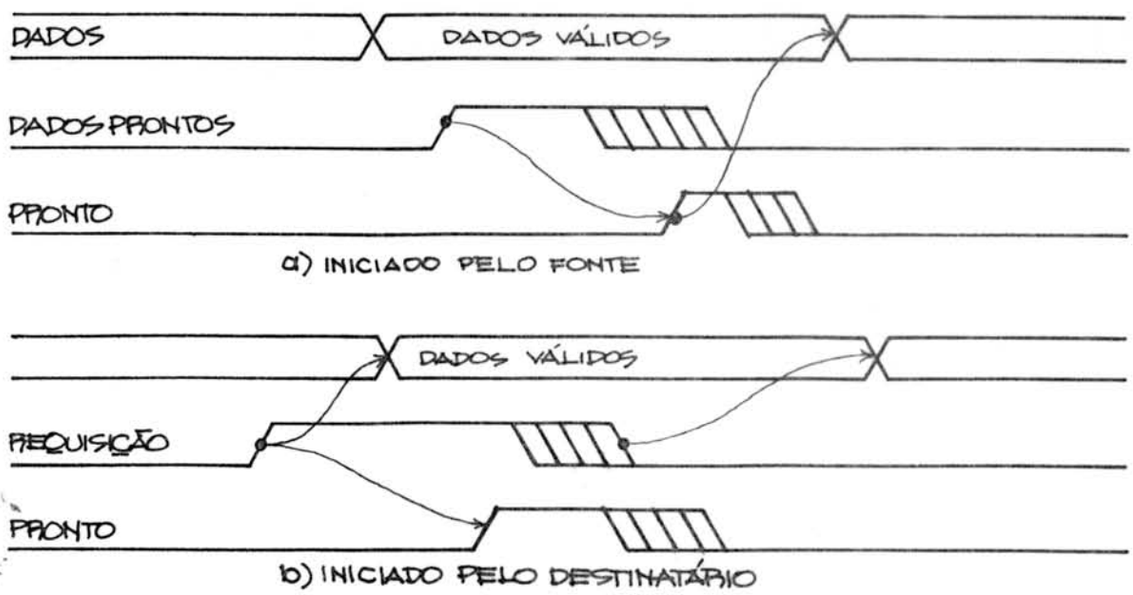


Figura 48 - Protocolo bidirecional não-entrelaçado

Quando iniciado pelo fonte, este coloca os dados no barramento e sinaliza DADOS PRONTOS. O destinatário copia

os dados e em resposta envia o sinal PRONTO o que causa a retirada dos dados por parte do primeiro. Quando iniciado pelo destinatário este aciona a linha REQUISIÇÃO o que causa a colocação dos dados no barramento e o envio de PRONTO por parte do fonte. A chegada de PRONTO faz com que o destinatário copie os dados.

O protocolo bidirecional entre os dois participantes facilita em muito a comunicação entre dispositivos de diferentes velocidades. Cada um opera na sua velocidade potencial. O sinal PRONTO impede que o dispositivo que iniciou a operação tente uma nova transferência antes de concluída a primeira. A sinalização de erro também é facilitada se for incluída uma terceira linha com a finalidade única de indicar uma condição de erro nos dados recebidos. Quem aciona esta linha é sempre o destinatário. No caso iniciado pelo destinatário ela não tem função pois este não precisa informar a condição de erro para se repetir a transferência.

O não-entrelaçamento faz com que a largura dos sinais de controle seja especificado pelo projetista. Isto pode acarretar problemas se a relação entre estes tempos e o de propagação do barramento não forem bem calculados. A subida do sinal PRONTO faz com que uma nova transmissão seja iniciada ou o barramento seja liberado. Se o dispositivo controlador da transferência é suficientemente rápido ou o árbitro de barramento já escolheu o próximo usuário, um novo pulso DADOS PRONTOS ou REQUISIÇÃO pode subir e descer enquanto o PRONTO anterior ainda não caiu. O resultado de tal situação é o bloqueio do barramento.

O protocolo bidirecional semi-entrelaçado visto na

figura 49 evita o bloqueio fazendo com que DADOS PRONTOS/REQUISIÇÃO tenham sua largura determinada pelo sinal PRONTO. Mais precisamente, a subida de PRONTO causa a descida de DADOS PRONTOS/REQUISIÇÃO. Assim, se DADOS PRONTOS/REQUISIÇÃO sobem antes da desativação de PRONTO o que ocorre é o retardo da transferência. Esta é uma solução paliativa pois não evita o problema, apenas minimiza seus efeitos.

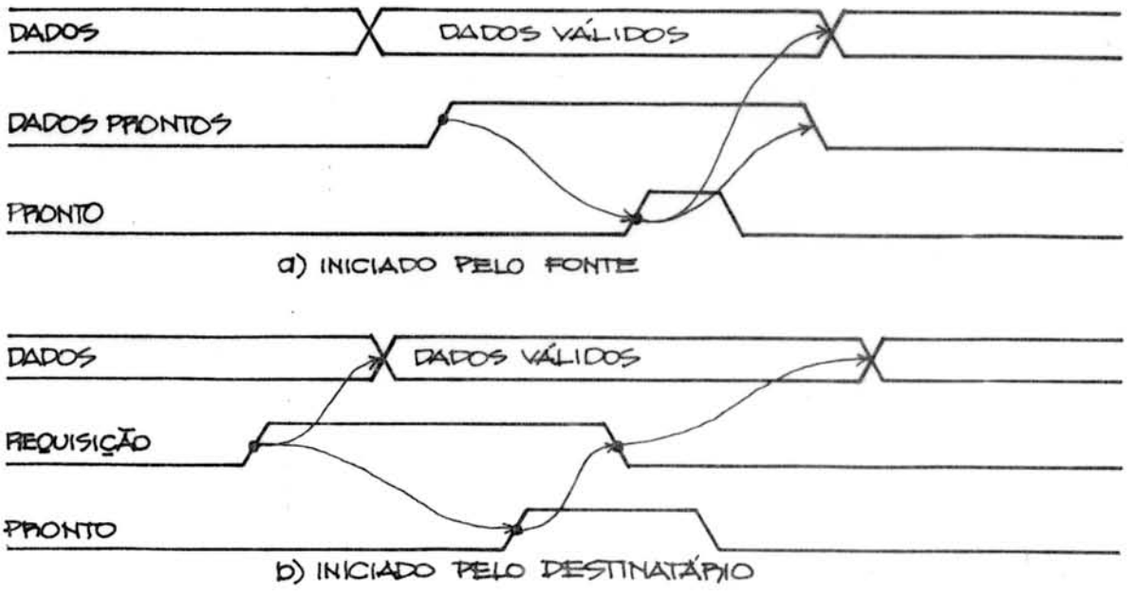


Figura 49 - Protocolo bidirecional semientrelaçado

O protocolo bidirecional entrelaçado (figura 50) evita completamente o problema condicionando as transições de REQUISIÇÃO/DADOS PRONTOS às transições de PRONTO e vice-versa. Desta maneira garante-se que DADOS PRONTOS/REQUISIÇÃO não ocorra antes da queda do PRONTO. Em compensação a transferência fica amarrada ao tempo de quatro propagações pelo barramento, o que diminui a taxa de transferência.

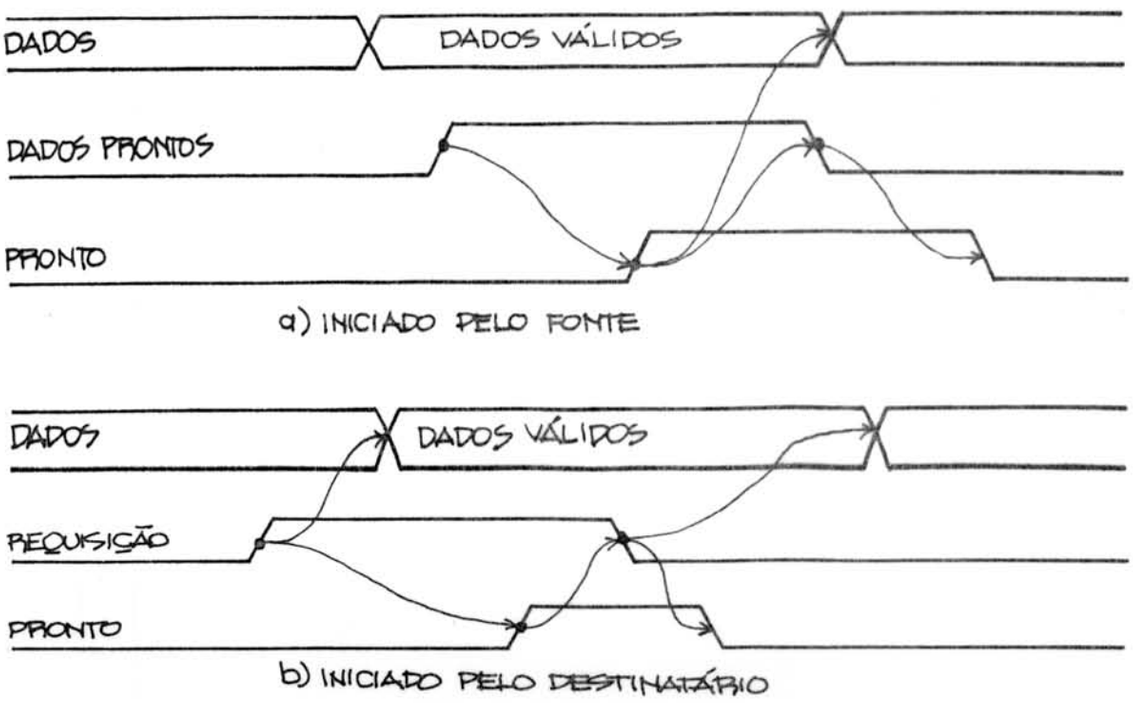


Figura 50 - Protocolo bidirecional entrelaçado

4.4.2 Filosofia de transferência

A expressão filosofia de transferência se refere ao número de itens de informação transferidos em cada concessão de barramento. Existem, basicamente, três filosofias: uma palavra apenas, blocos de tamanho fixo e blocos de tamanho variável. A escolha entre um tipo ou outro depende muito das características de acesso e tempos de resposta dos dispositivos ligados ao barramento.

Um barramento que permite transferir apenas uma palavra em cada acesso impede a conexão direta de dispositivos que operam com blocos de dados (discos, por exemplo). A solu-

ção neste caso vem na forma de canais de acesso direto à memória operando por roubo de ciclo ou processadores de entrada e saída interpostos entre o barramento e os dispositivos. O árbitro de barramento deve ser o mais veloz possível uma vez que a cada palavra transferida é feita uma nova alocação. O tempo de arbitração é mascarado se a alocação é feita em paralelo com a transferência atual.

A transferência de blocos de tamanho fixo pode aumentar a taxa de transferência em relação à filosofia de uma palavra por acesso. A melhoria é decorrente do menor tempo gasto alterando conexões estabelecidas. Uma vez que a conexão só é alterada depois de transferido um certo número fixo de palavras, o árbitro pode ser mais lento, ou seja, mais simples. O problema desta filosofia surge quando os dispositivos trabalham com blocos de diferentes tamanhos. Se o bloco de barramento é menor que o do dispositivo, há a necessidade de mais de um acesso. Se, ao contrário, o bloco de barramento é maior, ocorre uma perda de tempo transferindo-se dados inválidos enquanto os dispositivos ficam alocados inutilmente. A filosofia de blocos de tamanho fixo só é realmente eficiente quando todos os dispositivos operam com o mesmo tamanho de bloco. Um exemplo onde é viável aplicar-se esta filosofia é o barramento que une a memória principal a uma memória cache.

Com blocos de tamanho variável programado a cada transferência consegue-se uma utilização mais eficiente do barramento e dos dispositivos. Porém, isto implica uma maior carga de trabalho na inicialização da transferência. A sobrecarga reflete-se negativamente sobre transferências de uma palavra apenas. Estas têm que ser encaradas como blocos de uma

unidade de informação. Como tal, o processo de inicialização representa uma sobrecarga considerável em relação a quantidade de informação transferida, contribuindo para a deterioração da performance.

A combinação da filosofia de uma palavra com a filosofia de blocos de tamanho fixo em um mesmo barramento mantém as vantagens de ambos os métodos isolados. Além disso, o problema de discrepância entre o tamanho dos blocos dos dispositivos e do barramento é parcialmente solucionado. A opção de transferir um bloco menor que o bloco padrão como uma série de palavras individuais aumenta a disponibilidade do barramento e reduz o número de casos em que os dispositivos ficam presos a uma transferência inútil.

A conjugação das filosofias de transferência de uma palavra e blocos de tamanho variável em um mesmo barramento proporciona a maior eficiência da utilização. Assim, a necessidade de transmitir um único item não envolve a sobrecarga característica da filosofia de blocos de tamanho variável. Ao mesmo tempo, qualquer dispositivo pode operar adaptando o tamanho do bloco às suas necessidades.

4.4.3 Largura do barramento

O número total de linhas que compõe um barramento é um fator importante no custo de um sistema. Uma linha física subentende a existência de acionadores, receptores e conectores em cada interface. Estes são componentes caros. Por esta razão e também por razões de confiabilidade é desejável mantê-

los a um mínimo necessário. A comunicação através de cabos traz problemas mecânicos e de ruído externo, que diminuem a confiabilidade.

A redução do número de linhas é alcançada de várias maneiras. Por exemplo, ao invés de dois barramentos unidirecionais, um de entrada e outro de saída, a opção de juntar ambos em um único bidirecional reduz à metade o número de linhas físicas necessárias. O aumento de complexidade lógica nas interfaces é pequeno em comparação com o benefício alcançado.

A multiplexação de funções mutuamente exclusivas em uma mesma linha é usada extensivamente em microprocessadores. Embora isto seja feito em função do número limitado de pinos, nada impede, em princípio, que a multiplexação se estenda até o barramento. Novamente, há um compromisso entre complexidade lógica nas interfaces e número de linhas, pois em algum ponto do sistema deve ocorrer a demultiplexação.

A transmissão serial é um método que permite reduzir a via de transmissão a uma única linha física. Ela é usada principalmente quando os dispositivos se encontram muito distantes entre si. A distância encarece não só as linhas mas também os receptores e transmissores nas interfaces, de modo que a transmissão em paralelo pode representar um alto custo. O compromisso, nesse caso, é entre velocidade e custo.

Em geral, a diminuição do número de linhas físicas, seja qual for o método empregado, adiciona uma dose de complexidade às interfaces[9] e, às vezes, uma diminuição de velocidade. O compromisso entre esses fatores e o custo das linhas deve ser estabelecido em função das especificações feitas a nível do sistema.

5 BARRAMENTO MULTIPLEXADO

5 BARRAMENTO MULTIPLEXADO

5.1 Introdução

Este capítulo descreve uma estrutura de interconexão destinada a ligar os processadores e a memória em um sistema multiprocessador. A estrutura consiste num barramento multiplexado entre os elementos ativos do sistema. Além desta, são apresentadas duas outras alternativas de interconexão: barramentos dedicados/memórias multiporta (capítulo 6) e uma matriz de barramentos cruzados (capítulo 7).

Para se levar a efeito o projeto lógico são estabelecidas algumas hipóteses como, por exemplo, o tipo de processador previsto. Isto decorre do fato de as decisões tomadas ao longo do trabalho estarem condicionadas às características dos componentes. Procurou-se, entretanto, definir as características dos processadores e memórias da forma mais ampla possível de modo a não ficar o sistema atrelado a um único componente em particular. A seção 5.2 contém as hipóteses assumidas com vistas à definição das três estruturas propostas (cap. 5, cap. 6, cap. 7).

5.2 Características e hipóteses assumidas

5.2.1 Organização do sistema

O sistema ao qual se aplica a estrutura de interconexão é composto de m microprocessadores e n módulos de memó-

ria global (figura 51).

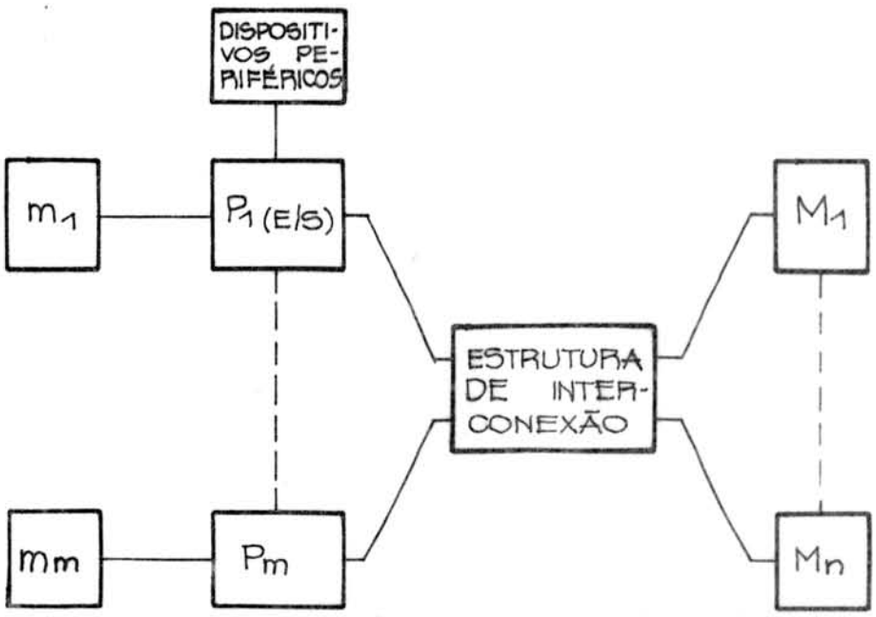


Figura 51 - Arquitetura do sistema

O ponto central que desperta maior interesse é a estrutura de interconexão. A forma como os microprocessadores têm acesso à memória é parcialmente responsável pelas características funcionais do sistema[14]. Ela deve ser construída de forma que todos os processadores tenham acesso a todos os módulos de memória M_1 a M_n. Além destes, cada processador possui um módulo privativo (m_1 a m_m) ligado ao seu barramento local. Um módulo de memória privativo só pode ser acessado pelo processador ao qual pertence. Portanto, o espaço de endereçamento de cada microprocessador é dividido em duas partes (figura 52).

A memória total do sistema é a soma de todas as memórias locais mais a memória global.

Os dispositivos periféricos interligam-se à memória global através de processadores de entrada e saída. O acesso àqueles recursos é garantido a todos os processadores do

Figura 52 - Espaço de endereçamento de um processador

sistema desde que exista um mecanismo de comunicação entre estes últimos. A memória global é a via física que possibilita a implementação de tal mecanismo.

No que tange ao tipo de sistema operacional capaz de suportar a organização, nenhuma restrição é imposta. O sistema operacional terá influência no mecanismo de comunicação e no mecanismo de interrupção entre processadores.

5.2.2 Requisitos impostos ao microprocessador

O projeto não prevê um microprocessador específico. Pelo contrário, os sinais da interface de processador recebem um mínimo de restrições, de forma a acomodar mais de um tipo de microprocessador. Conseqüentemente é permitida a assimetria entre os processadores. Na realidade não é obrigatório o uso de um microprocessador; qualquer UCP compatível com os sinais de interface pode ser usada. Entretanto, os micros constituem-

se, pelo seu baixo custo, em uma das maiores motivações para o multiprocessamento. A UCP escolhida deve atender certos requisitos.

Um requisito fundamental é a possibilidade de adiar indefinidamente a efetivação de um acesso à memória após iniciado[50]. Esta imposição justifica-se em virtude dos acessos conflitantes a barramentos ou módulos de memória. Acessos simultâneos por parte de dois ou mais processadores só podem ser identificados depois de iniciados. Como apenas um é deferido em cada instante, os demais devem ser informados da situação e entrar num estado de espera até serem atendidos. Em aplicações convencionais[42],os microprocessadores não esperam por um sinal de resposta do dispositivo endereçado indicando a efetivação da transferência. Uma vez iniciado um ciclo de acesso ao meio, eles esperam que o dispositivo endereçado complete a operação dentro de um tempo bem especificado. Em uma operação de entrada de dados estes devem estar disponíveis, e numa escrita esta deve ter sido efetivada ao fim deste tempo. Entretanto,a maioria dos micros possui a característica desejada na forma de um sinal de entrada o qual é capaz de adiar indefinidamente a leitura ou manter os dados e endereço estáveis para escrita enquanto ativado. Este sinal, normalmente utilizado para periféricos e memórias lentas, adapta-se perfeitamente ao propósito de manter uma UCP esperando pela deferição de um pedido de acesso à memória. Conseqüentemente, um microprocessador que não possui esta facilidade não se adapta a nenhuma das estruturas propostas.

Um segundo requisito é a disponibilidade de informação de estado no início de cada ciclo de máquina. Os micro-

processadores, em geral, liberam sinais que identificam o tipo de acesso juntamente com o endereço ou até mesmo antes dele.As informações básicas necessárias devem especificar se o ciclo que se inicia é de interrupção, de acesso a memória ou periféricos, leitura ou escrita. Estes sinais em conjunto com a decodificação de endereço são usados para direcionar o pedido à interface correta. Assim distingue-se um acesso à memória local sem modificar o estado da interface de memória global. As informações de estado e endereço são freqüentemente multiplexadas com outras funções. Porém, isto não se constitui em problema desde que os microprocessadores liberem primeiramente as informações de estado.

A existência de sinais individuais para indicar a validade do conteúdo barramentos de endereço e dados, embora não seja estritamente necessária, é uma característica desejável. Definindo-se as interfaces em função de sinais individuais torna-as mais flexíveis no sentido de acomodar diferentes microprocessadores sem restrições. Aqueles micros que não possuem sinal para indicar a validade dos dados (como o 6800) devem providenciá-lo através de lógica externa na interface.

Além destes requisitos existem outras funções úteis para multiprocessamento[50], como acesso direto à memória, informação de acesso à pilha, dados ou código, etc. No entanto para efeito de interconexão com a memória estes são suficientes.

5.2.3 Sinais externos do microprocessador

Todo microprocessador possui um conjunto de sinais para efetuar a comunicação com o meio exterior. Basicamente, um acesso ao meio pode ser classificado como sendo dirigido à memória ou ao subsistema de entrada e saída. A interação com o meio é feita através de sinais de endereços, dados e controle. Para efeitos do protocolo de comunicação é admitida a existência dos seguintes sinais do microprocessador hipotético.

ENDP (saída) - sinais de endereço. Identifica tanto endereços de memória como de entrada e saída. A distinção entre um e outro é feita por um sinal de controle ou por informação de estado. O barramento de endereços pode ser multiplexado com outras funções.

STBEND (saída) - enquanto ativo sinaliza a presença de um endereço válido no barramento.

DADOSP (entrada/saída) - sinais de dados bidirecionais. É a via de transferência de dados de e para o microprocessador. Num ciclo de interrupção vetorada contém o vetor de interrupção. Pode ser multiplexado com outras funções.

STBDADO (saída) - sinaliza o instante em que existe um dado válido no barramento em um ciclo de escrita, ou o instante em que os dados de entrada devem ser colocados no barramento em operações de leitura.

LEIT/$\overline{\text{ESC}}$ (saída) - informação de estado. Especifica se o ciclo em andamento é de leitura ou escrita na memória.

PRONTO (entrada) - sinal de controle assíncrono que indica o término de uma operação de acesso ao meio. En-

quanto desativado, o microprocessador mantém os sinais STBDADO e LEIT/$\overline{ESC}$ ativados. Se a operação é de saída os dados também permanecem estáveis. Em operações de entrada o barramento de dados permanece disponível para os dados provenientes do exterior até a ativação do sinal PRONTO. Nesse instante o micro retira todos os sinais, copia os dados (se for o caso) e encerra o ciclo.

INTACK (saída) - informação de estado. Especifica o reconhecimento de um pedido de interrupção e a execução do ciclo de recebimento do vetor de interrupção. Este sinal em conjunto com STBDADO indica o instante em que o dispositivo interruptor deve colocar o vetor no barramento.

HLTACK (saída) - informação de estado. Indica um ciclo de parada do microprocessador em resposta a uma instrução HALT.

Um diagrama de tempos típico dos sinais é visto na figura 53. Nela estão assinalados os parâmetros de tempo que devem ser levados em consideração no projeto da estrutura de interconexão. O significado dos tempos está na tabela 1.

Os sinais pertinentes a ciclos de interrupção e de parada do microprocessador não são usados na comunicação com a memória. Eles são incluídos pelo fato de os sinais de controle (STBEND, STBDADO) serem ativados por alguns microprocessadores durante estes ciclos. As interfaces de processador não devem confundi-los com tentativas de acesso à memória.

Na figura citada são mostrados dois ciclos de máquina. O primeiro é o ciclo normal do microprocessador. O segundo é atrasado pelo sinal PRONTO. A contenção na disputa por um barramento ou memória provoca um ciclo do segundo tipo.

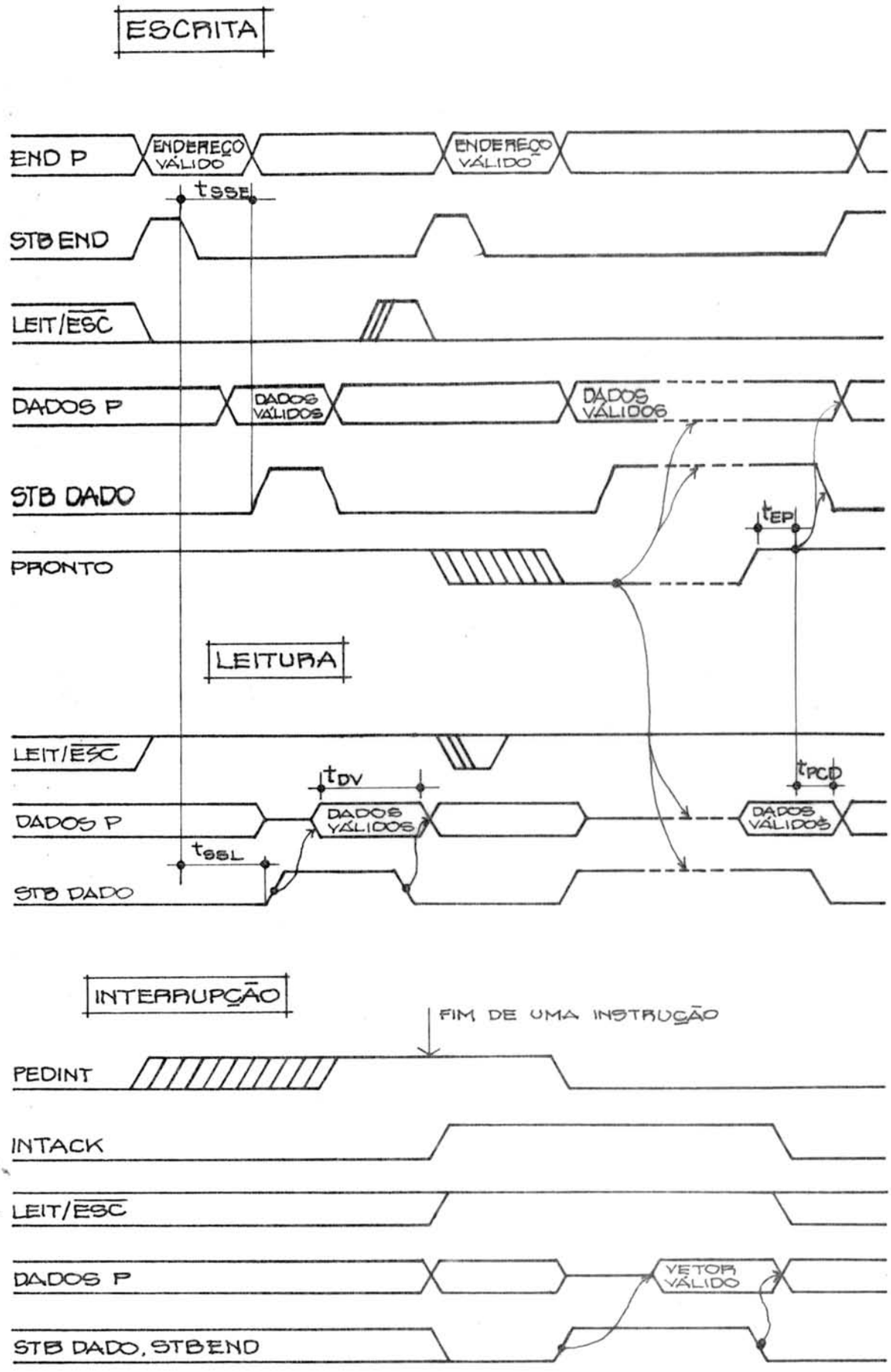


Figura 53 - Diagrama de tempos dos sinais do micro-processador

t_{SSE}	atraso entre os sinais STBEND e STBDADO em um ciclo de escrita
t_{SSL}	atraso entre os sinais STBEND e STBDADO em um ciclo de leitura
t_{DV}	tempo de pré-estabilização dos dados antes da cópia pelo microprocessador
t_{PCD}	tempo de atraso entre o instante de amostragem do PRONTO e a cópia dos dados
t_{EP}	tempo de pré-estabilização do sinal PRONTO

Tabela 1 - Significado dos tempos do microprocessador

O microprocessador utilizado não precisa possuir todos estes sinais da forma especificada. A adição ou modificação da lógica da interface com o barramento permite acomodar diferentes tipos de processadores.

5.2.4 Memória

O espaço de endereçamento de cada microprocessador é dividido em duas partes. Uma acessível somente ao próprio microprocessador (local) e outra de uso comum (global). A distinção entre elas é feita pela decodificação de endereço com base no mapeamento pré-estabelecido. A relação de tamanho entre as duas depende das características de partilhamento desejadas. Se a memória global é usada apenas como via de comunicação entre os processadores, então não há necessidade de uma memória extensa. Se, por outro lado, o objetivo é partilhar programas e estruturas de dados, uma memória maior se faz necessária.

Nesse caso cresce a atividade na memória global, crescendo junto a probabilidade de interferência entre os processadores. Este fator é de extrema importância na escolha da estrutura de interconexão.

São feitas as seguintes hipóteses acerca da memória global:

1º) seu espaço de endereçamento é subdividido em n módulos;

2º) os módulos são implementados com memórias monolíticas estáticas;

3º) não entrelaçamento;

4º) a unidade básica de organização é a palavra.

Um endereço de barramento refere-se a uma palavra. Um endereço em conjunto com dois sinais de controle (ver tabela 2) especificam se uma palavra inteira ou apenas um byte deve ser transferido.

BYTSUP	BYTINF	TRANSFERÊNCIA
0	0	BYTE INFERIOR
1	1	BYTE SUPERIOR
1	0	NENHUMA
0	1	PALAVRA (16 BITS)

Tabela 2 - Sinais de controle para transferência de byte ou palavra

O byte superior é sempre transferido nos 8 bits superiores do barramento de dados e o byte inferior nos 8 bits inferiores. Para permitir que um microprocessador de 8 bits tenha acesso a todos os bytes deve-se fazer BYTSUP=BYTINF=A_0/ (bit

zero de endereço) e usar os bits restantes como endereço. Desta maneira o micro endereça palavras seqüencialmente fazendo 2 acessos em cada uma: primeiro o byte superior (endereço mais baixo) e depois o byte inferior (endereço mais alto). Como os dados aparecem nas duas metades do barramento a interface de um micro de 8 bits deve ser dotada de dois transceptores (figura 54). A seleção entre os dois é feita pelo bit zero de endereço.

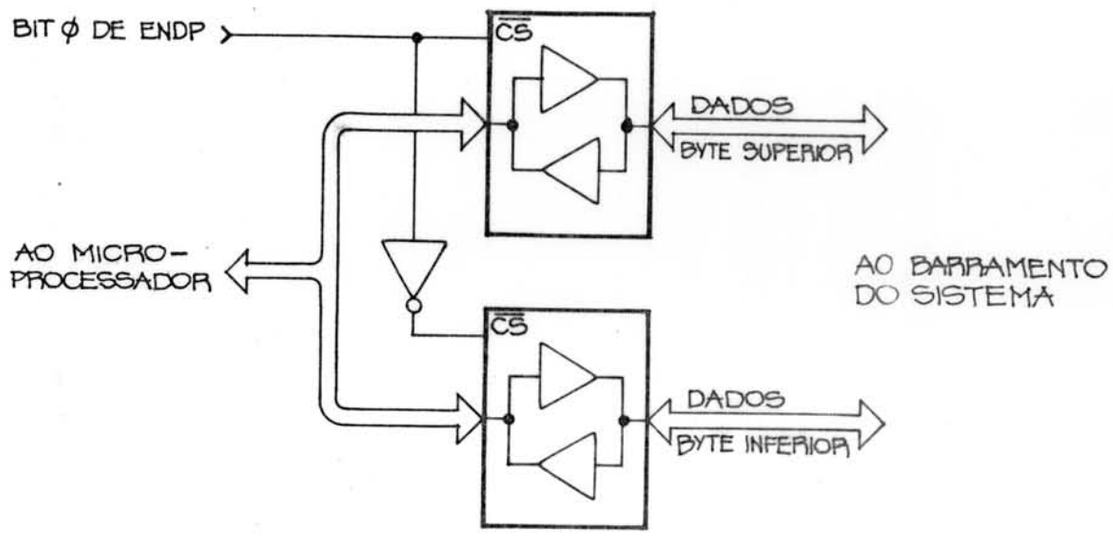


Figura 54 - Ligação de um microprocessador de 8 bits a um barramento de 16 bits

O não-entrelaçamento é escolhido pelo fato de gerar um menor grau de interferência em relação a uma memória entrelaçada[9,55]. Num sistema multiprocessador com memória entrelaçada as requisições distribuem-se uniformemente sobre os módulos. A probabilidade de um processador acessar um determinado módulo é idêntica para qualquer combinação processador-módu

lo. Porém, deve-se observar que a efetividade de uma memória entrelaçada provém da operação simultânea de vários módulos em resposta a um acesso. Estendendo-se o raciocínio ao conjunto de processadores operando assincronamente conclui-se que esta estratégia de mapeamento dos endereços físicos leva a um alto grau de interferência.

Ao invés de um padrão de referência uniforme pode-se, através do não-entrelaçamento e de um mecanismo de alocação de memória adequado, fazer com que cada processador referencie um ou mais módulos preferencialmente. Assim, os dados e o código de um processo ficariam, sempre que possível, armazenados no módulo preferencial do processador no qual ele está sendo executado[13]. Isto, em conjunto com a memória privativa, tende a reduzir o número de conflitos de acesso à memória.

O uso de memórias estáticas é assumido a fim de abstrair os detalhes pertinentes a refrescamento. Se é uma imposição o uso de memória dinâmica, então os circuitos de refrescamento devem ser localizados nos módulos[11]. Assim, o único problema a ser resolvido é a concorrência do refrescamento com as tentativas de acesso externas. A lógica de interface dos módulos teria de sofrer alterações mas o barramento não.

5.2.5 Comunicação entre processadores

Num sistema multiprocessado é imprescindível a existência de um mecanismo de comunicação entre processadores. Ao nível de hardware a memória global é suficiente para a implementação deste mecanismo. A comunicação através da memória

(via caixa postal) exige que todos os processadores tenham acesso a esta memória, além de primitivas de sincronização para efetuar a exclusão mútua na manipulação de informações nela contida.

No sistema em questão assume-se este tipo de mecanismo de comunicação. Sob o ponto de vista taxonômico(capítulo 2) estamos diante de um sistema DPM (Direto-Partilhado-Memória). A comunicação é feita diretamente através de uma via partilhada, a memória.

A alocação de memória para as caixas postais é função do sistema operacional. Este deve estar ciente dos endereços reservados a cada caixa-postal. A maneira como um processador informa outro da existência de uma mensagem (por interrupção, ou verificação periódica do estado da caixa-postal) não é relevante para a definição da estrutura. Por isto, as duas funções são abstraídas deste trabalho.

5.2.6 Entrada e saída

Assume-se que as unidades periféricas serão ligadas ao sistema através de um ou mais processadores de entrada e saída. Esta escolha é feita por força de uma série de considerações.

A primeira consideração diz respeito a filosofia de transferência (seção 4.4.2) imposta à estrutura de interconexão do sistema. Um dispositivo periférico com capacidade de acesso direto à memória pode requerer transferência em bloco ao invés de roubo de ciclo. Um microprocessador, por sua vez,

requisita um barramento para transferir uma palavra apenas. Se os dispositivos de entrada e saída fossem ligados diretamente ao(s) barramento(s) do sistema, estes deveriam suportar tanto transferências de uma palavra como blocos de tamanho variável. Com a inclusão de processadores de entrada e saída os dados são transferidos entre periféricos e a memória local. A transferência entre a memória local e a global é feita por software pelo processador de entrada e saída[49]. Portanto, a via de acesso deve suportar a filosofia de transferência de uma palavra apenas.

O confinamento de periféricos ao barramento do processador de entrada e saída propicia a simplificação do mecanismo de interrupção. Os dispositivos não precisam escolher qual processador será interrompido ao concluírem uma operação. A interrupção é dirigida sempre ao processador de entrada e saída ao qual o periférico pertence.

Uma outra questão a ser considerada é a forma como é assegurada a integridade da transferência durante uma operação de acesso direto à memória. Se há um só barramento e a ele são ligados os periféricos então todos os processadores devem ser bloqueados o que ocasiona uma diminuição de eficiência. Se a estrutura de interconexão é uma matriz de barramentos cruzados ou memórias multiporta, então não é necessário parar os processadores; os conflitos são resolvidos na chave ou nos módulos respectivamente. Porém, periféricos que geram requisições de acesso,críticas (como um disco, por exemplo) não podem estar sujeitos à interferência muito severa, a não ser que ele tenha uma alta prioridade na disputa pela memória. Mesmo assim o número de dispositivos de alta velocidade é limitado pela

interferência resultante. A interposição de um armazenamento temporário ("buffer") entre estes dispositivos e a memória é uma forma de resolver o problema. A memória local do processador de entrada e saída provê esta facilidade. Assim os dados podem ser transferidos velozmente entre periféricos e memória local. Dali para a memória global é tarefa do processador de E/S.

5.3 O barramento

A primeira solução apresentada para a estrutura de interconexão da figura 51 é um único barramento interligando todos os processadores e módulos de memória do sistema. A figura 55 ilustra a organização sugerida.

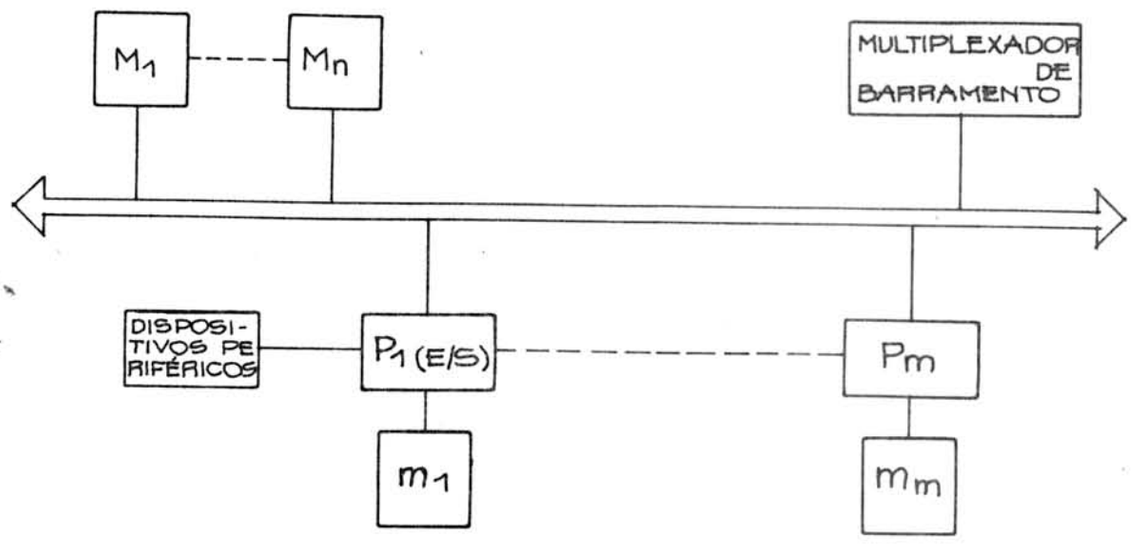


Figura 55 - Organização de um multimicroprocessador em torno de um barramento multiplexado

O barramento da figura compreende linhas de endereços, dados e controle. Ele é do tipo não dedicado pois existem m elementos capazes de requisitá-lo e tornarem-se mestres. A cada instante só pode ocorrer uma transferência. É necessário, portanto, incluir um mecanismo de exclusão mútua. O mecanismo responsável por esta função difere dos apresentados no capítulo 4. As alternativas de exclusão baseadas em árbitros (cadeia, interrogação, requisições independentes ou auto-seleção) foram descartadas em favor da multiplexação explícita do barramento. Este é o aspecto que o caracteriza, donde o nome Barramento Multiplexado.

A idéia é dispensar o árbitro. Mais importante que dispensar o árbitro é evitar o tempo gasto no processo de alocação, o qual não contribui com nenhuma forma de processamento útil. Paralelamente, abre-se a possibilidade de operação de mais de um módulo de memória ao mesmo tempo, uma característica incomum em multiprocessadores com um único barramento.

A exclusão mútua é obtida designando-se uma janela de tempo a cada processador (figura 56).

Figura 56 - Janelas de barramento

As janelas são geradas ciclicamente, quer os processadores necessitem delas ou não. O multiplexador é o órgão encarregado da geração, cabendo a cada interface de processador reconhecer sua janela. Na verdade, as interfaces de memória também devem ser capazes de reconhecê-las para permitir o-

peração simultânea de mais de um módulo, conforme será explicado. Como se vê, o esquema dispensa o árbitro convencional, porém exige um multiplexador. Com um único barramento é imprescindível alguma forma de exclusão mútua.

Mesmo que todos os módulos sejam implementados com memórias de igual velocidade os conflitos de acesso fazem com que eles se comportem como dispositivos de velocidade diferentes.

A técnica de comunicação assíncrona é necessária devido à impossibilidade de se saber o instante exato em que um acesso à memória é completado a partir do seu início. Os microprocessadores, normalmente, fazem uso da técnica de comunicação síncrona para comunicar-se com a memória. Porém, pode-se, facilmente, torná-la assíncrona na interface entre o micro e o barramento. Isto é possível graças à capacidade inerente de esperar por sinais externos para efetivar uma transferência quando isto se faz necessário (ver seção 5.2.2). O protocolo bidirecional é iniciado pelo multiplexador com um sinal global de sincronismo e terminado pelo módulo de memória endereçado com um sinal de aceitação do pedido de acesso. O assincronismo entre as interfaces permite a operação com microprocessadores diversos e memórias com tempo de acesso diferentes.

A largura do barramento de dados é fixada em 16 bits. Entretanto, pode-se manipular apenas um dos dois bytes que compõem a palavra. Tipicamente, a comunicação entre processador e memória envolve uma palavra ou um byte. No intervalo de duração de uma janela pode ser transferida uma palavra no máximo.

A largura do barramento de endereço é função do ta

manho da memória global. Por esta razão ele não é fixado a priori. A capacidade de endereçamento limitada dos microprocessadores de 8 bits (64 Kbyte) se constitui em mais um problema se é exigida uma memória global extensa[65]. Uma solução para esse problema consiste em ampliar o espaço de endereçamento do microprocessador pela inclusão de mais bits do barramento de endereço. Isto é feito pela interposição de um mecanismo de transcodificação de endereço entre o microprocessador e o barramento do sistema (figura 57)[37].

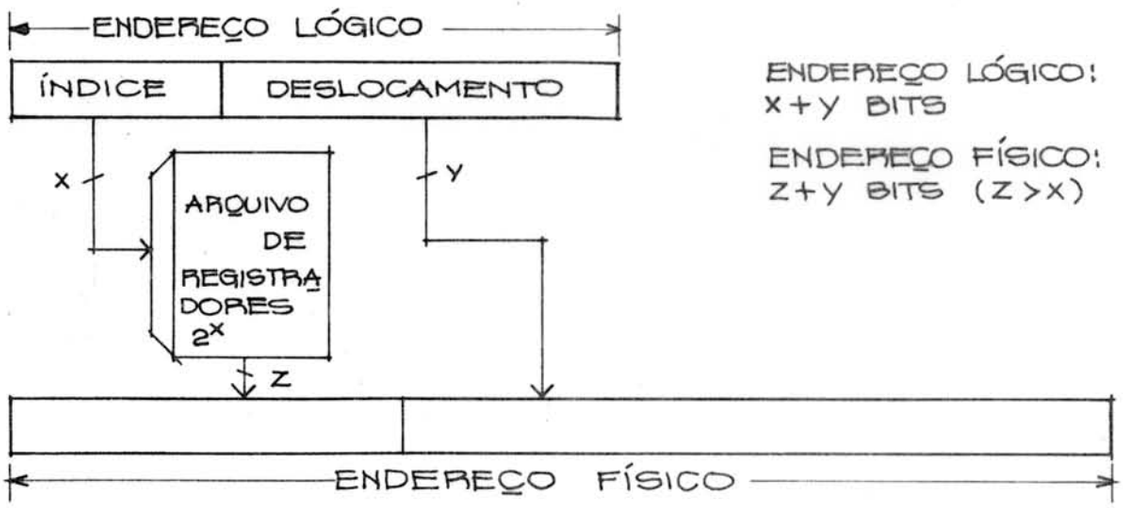


Figura 57 - Ampliação do espaço de endereçamento de um microprocessador

A adição de bits de endereço externamente força o microprocessador a suportar um tipo de dado não compatível com sua arquitetura e conjunto de instruções. Por exemplo a adição de 2 bits a um endereço lógico de 16 bits significa que o micro pode endereçar 256 Kbytes. Todavia, ele também é obrigado a operar com dados de 18 bits, assim como armazenar e carregar

estes dados da memória.

5.3.1 Janelas de tempo

Uma janela de tempo nada mais é que um código binário associado biunivocamente com um dos processadores do sistema. O código é transmitido ao barramento de identificação pelo multiplexador e decodificado em todas as interfaces de processador (figura 58).

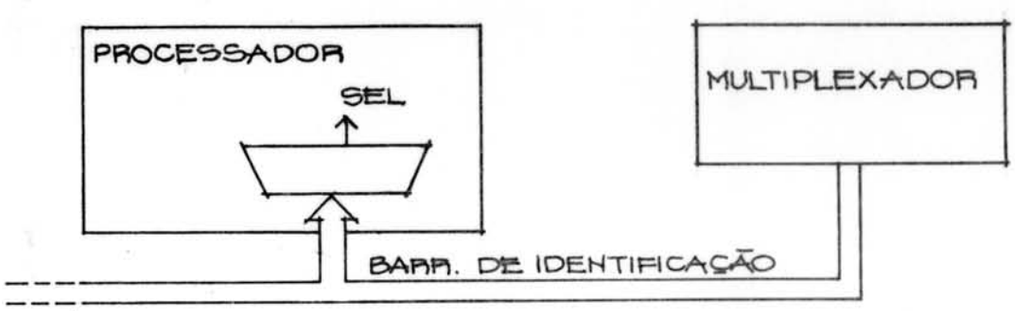


Figura 58 - Identificação de janela de tempo

Ao verificar a presença de seu código de identificação, o processador pode iniciar um acesso externo. Durante o tempo em que o seu código permanece ativo nas linhas, nenhum outro dispositivo pode atuar nas demais linhas que compõem o barramento.

As janelas são geradas a intervalos de tempo regulares independentemente do seu uso ou não. De forma que podem acontecer janelas "vazias" nas quais não acontece nenhuma atividade. Isto tende a diminuir a taxa de utilização do barramen

to. A velocidade de operação individual e coletiva também é prejudicada na medida que uma janela é gerada em vão enquanto um ou mais processadores estão trancados aguardando suas respectivas janelas para tentar um acesso à memória.

Uma questão fundamental é a determinação da largura das janelas, ou seja, a fração de tempo dedicada a cada processador. Existem basicamente dois caminhos a seguir. O primeiro é permitir que uma leitura ou escrita na memória se complete dentro do tempo de janela, mantendo o barramento durante todo o acesso. Nesse caso a largura é determinada pelo tempo de ciclo da memória acrescido dos tempos de propagação e estabilização dos sinais de barramento. A segunda hipótese consiste em utilizar a janela apenas para tranferir o endereço (e dados no caso de escrita) desde o processador até a memória. A conclusão de um ciclo de leitura é feita na janela subseqüente do mesmo processador. Nesse caso a largura da janela é determinada apenas pelo tempo de propagação e estabilização dos sinais trocados entre as interfaces. Partindo-se da premissa que, inevitavelmente, serão geradas janelas vazias, é altamente desejável que o seu tempo seja o mais curto possível. Assim sendo, optou-se pela segunda hipótese como forma de minimizar o prejuízo causado pelas janelas ociosas.

A escolha feita implica abandonar os sinais temporizadores da leitura e escrita dos microprocessadores, uma vez que a conexão não se mantém durante todo o ciclo. Conseqüentemente, os módulos de memória devem ser dotados de circuitos para realizar esta função. A desvantagem é compensada pela possibilidade de operação simultânea de mais de um módulo de memória. Uma outra possibilidade interessante é o uso de memórias

mais lentas, e portanto mais baratas, sem prejuízo da performance desde que o número de janelas assim o permita. Isto porque o tempo de acesso é transparente ao microprocessador. O fator limitante é a freqüência das janelas. O requisito básico de uma leitura estar concluída no espaço de tempo entre duas janelas consecutivas do mesmo processador fornece a relação entre o tempo de ciclo máximo da memória e o número mínimo de janelas (n), por ciclo do multiplexador.

$$tc < n \times tj$$

onde tj é a largura das janelas e tc o tempo do ciclo da memória. Portanto, se o número mínimo de janelas geradas é fixado pelo projeto, também fica determinado o tempo de ciclo máximo admissível para a memória. Uma memória mais rápida que esta só traz benefício em operações de escrita, nas quais o módulo fica bloqueado somente até o fim do ciclo de escrita. Em operações de leitura o módulo fica bloqueado durante uma seqüência completa de janelas independentemente do tempo de acesso da memória.

As janelas são geradas assincronamente em relação aos processadores por um relógio privativo do multiplexador. O número máximo de processadores é limitado teoricamente pelo número de linhas do barramento de identificação. Se existem i linhas então podem ser ligados 2^i processadores.

A figura 59 ilustra as janelas da forma como são vistas no barramento. O significado dos tempos do diagrama estão na tabela 3.

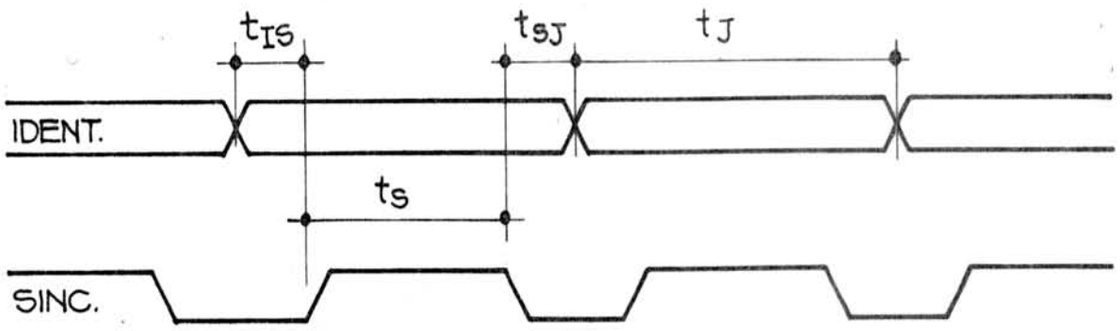


Figura 59 - Diagrama de tempos das janelas

SIGLA	SIGNIFICADO	REQUERIMENTOS
t_{IS}	Atraso entre identificação VÁLIDA e SINC	> $t_{PB} + t_{ID}$
t_S	Largura do pulso de sincronismo	> $t_{PB} + t_{EV}$
t_J	Tempo de janela	< $t_{c/n}$
t_{PB}	Tempo de propagação e estabilização do barramento	-
t_{SJ}	Tempo de desativação de SINC antes do fim da janela	> t_{PB}

Tabela 3 - Significado dos tempos referentes às janelas

O sinal de sincronismo SINC coordena a atividade nas interfaces de processador e de memória. A sincronização das interfaces em relação às janelas é feita em ambas as bordas do sinal SINC, de maneira a evitar uma falsa seleção causada por sinais espúrios ou na transição das linhas de identificação.

O atraso do sinal de sincronismo SINC em relação ao início da janela é determinado em função do tempo que a in-

terface de processador leva para reconhecer sua janela. Devem ser levados em conta o tempo de propagação e estabilização das linhas. Identificação desde o multiplexador até a interface e também o atraso interno a interface (t_{ID}). Este último medido de acordo com o número de portas pelas quais o sinal passa.

A largura do pulso de sincronismo é determinada em função do protocolo estabelecido. Admita-se que na subida de SINC o processador abra suas portas de saída para o barramento e na descida de SINC o módulo de memória responda afirmativamente ou negativamente ao pedido de acesso. Isto leva a estabelecer um tempo de propagação de barramento e mais o atraso interno da interface (t_{EV}) como o suficiente para o módulo identificar uma requisição (endereço válido).

A desativação de SINC deve ocorrer pelo menos um tempo de propagação de barramento antes do fim da janela para permitir que o processador receba o sinal de aceitação proveniente do módulo endereçado.

5.3.2 Multiplexador de barramento

A função primordial do multiplexador de barramento é gerar os códigos de identificação correspondentes aos processadores. O diagrama em bloco do multiplexador está na figura 60.

O número de processadores ativos varia dinamicamente em um sistema multiprocessado. Na hipótese de um processador ficar inativo durante um certo tempo, sua janela estará

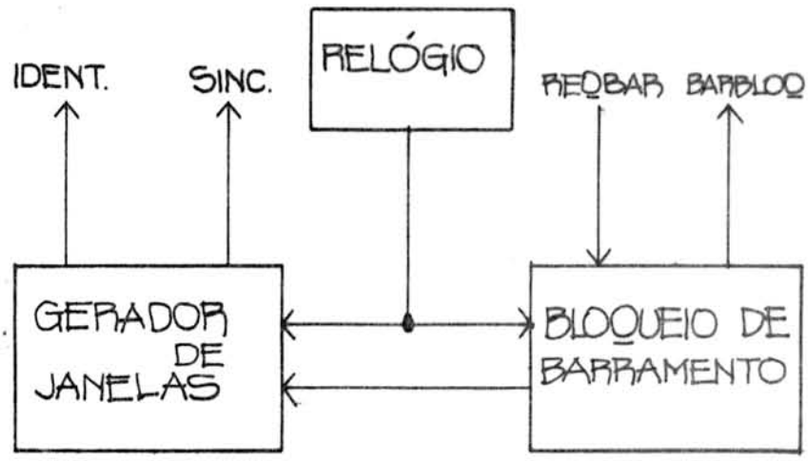


Figura 60 - Diagrama em blocos do multiplexador de barramento

sendo gerada em vão. Em conseqüência o sistema como um todo não estará operando a sua velocidade máxima. Uma solução para este problema é fazer com que o multiplexador também possa alterar dinamicamente o número de janelas, baseado no conhecimento de quem está e quem não está ativo. O candidato mais provável a esta tarefa é o sistema operacional pois ele deve ter uma visão global das atividades em curso.

A maneira mais simples de se implementar o circuito gerador de janelas é através de um contador síncrono de módulo variável conforme mostrado na figura 61.

Porém, esta implementação permite alterar apenas o módulo de contagem e não a seqüência de geração de janelas. Assim, se uma janela do meio da seqüência deve ser retirada, todas as outras subseqüentes devem ser deslocadas de uma posição para preencher o espaço deixado por aquela. A situação é mostrada na figura 62, tomando como exemplo a retirada da janela nº 6 numa seqüência de nove janelas.

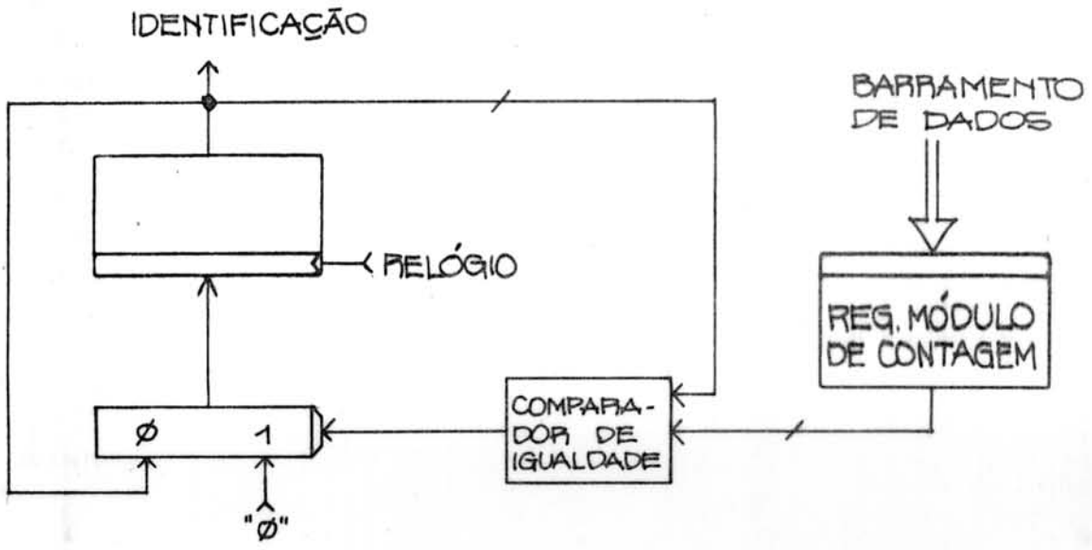


Figura 61 - Gerador de janelas implementado com contador

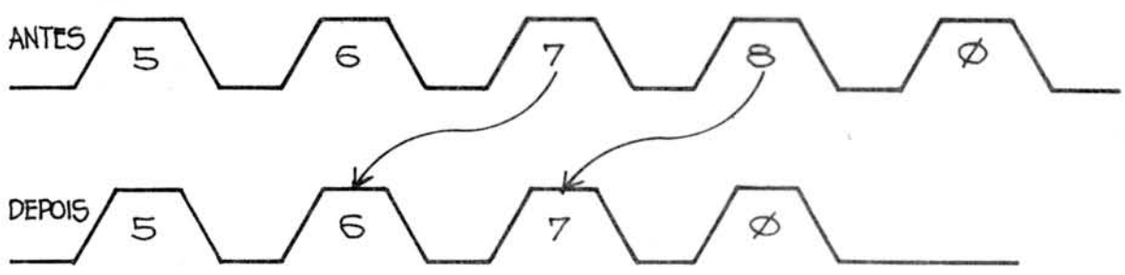


Figura 62 - Retirada de uma janela gerada por contador

Os processadores devem ter seus códigos de identificação alterados toda vez que se modifica o módulo de contagem. Isto implica a capacidade de modificar um registrador de identificação dentro de cada interface de processador.

Uma outra alternativa que facilita a geração de se

qüências aleatórias é obtida pelo uso de uma lista encadeada de janelas. A implementação deste gerador emprega uma memória para armazenar a lista (figura 63).

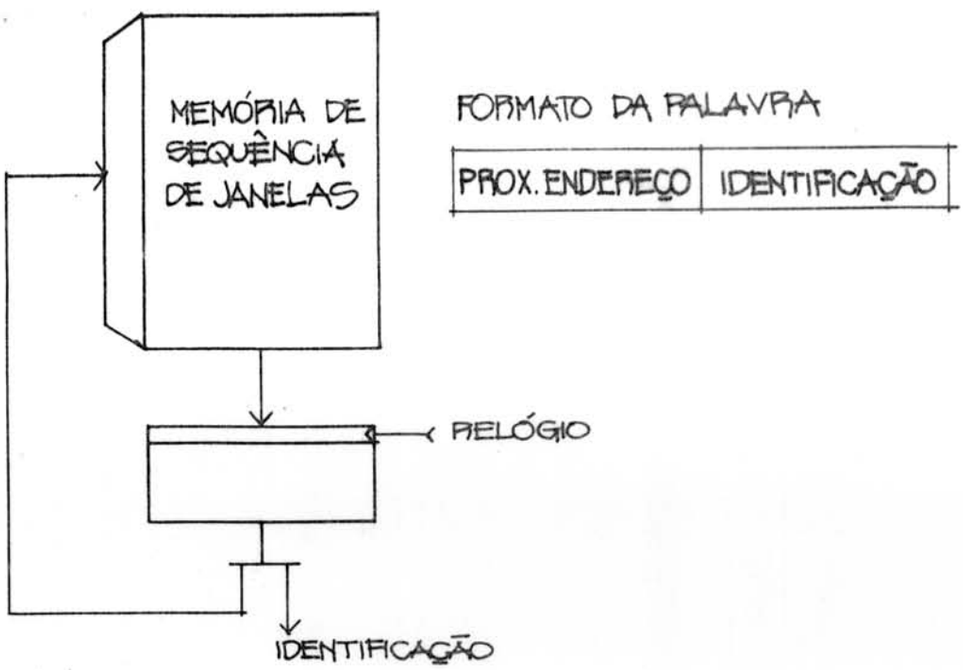


Figura 63 - Gerador de janelas implementado com memória

As exigências impostas à memória de janela são:

1) tempo de ciclo menor que t_J. Portanto ela deve ser uma memória rápida.

2) ser do tipo leitura/escrita de modo que diferentes seqüências possam ser armazenadas sob o controle do sistema.

Para promover uma alteração na memória a geração de janelas deve ser interrompida a fim de impedir que outros processadores tenham acesso ao barramento durante a modificação. O circuito de bloqueio do barramento é responsável por esta função. O sinal REQBAR é acionado por qualquer um dos processadores habilitados a executar uma alteração na memória de

janelas. Em resposta, a geração de janelas é suspensa, permanecendo presente o sinal SINC. O sinal BARBLOQ gerado pelo multiplexador é a resposta sincronizada a um pedido. Note-se que a existência de apenas uma linha (REQBAR) poderia gerar conflitos se mais de um processador tentasse modificar a memória de janelas ao mesmo tempo. Portanto, é necessário implementar uma primitiva de sincronização de alto nível. O acionamento da linha REQBAR (por software) deve ser uma seção crítica do sistema operacional.

Internamente ao multiplexador, a geração de janelas é feita por um relógio de duas fases (figura 64). Um ciclo de bloqueio de barramento é mostrado na figura 65.

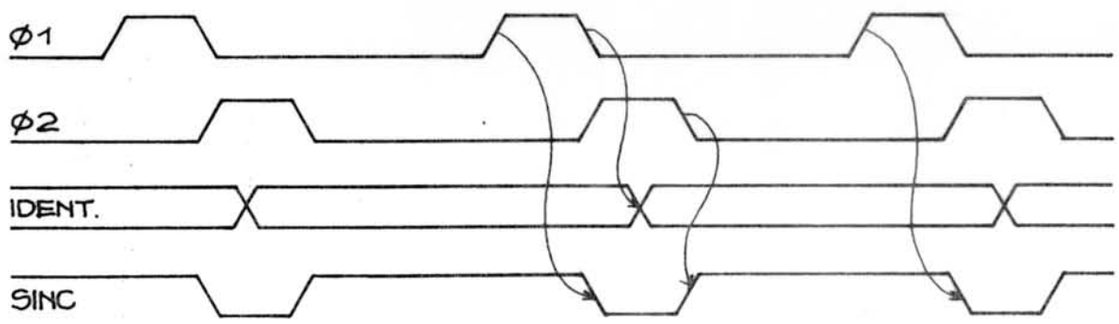


Figura 64 - Diagrama de tempos dos sinais do gerador de janelas

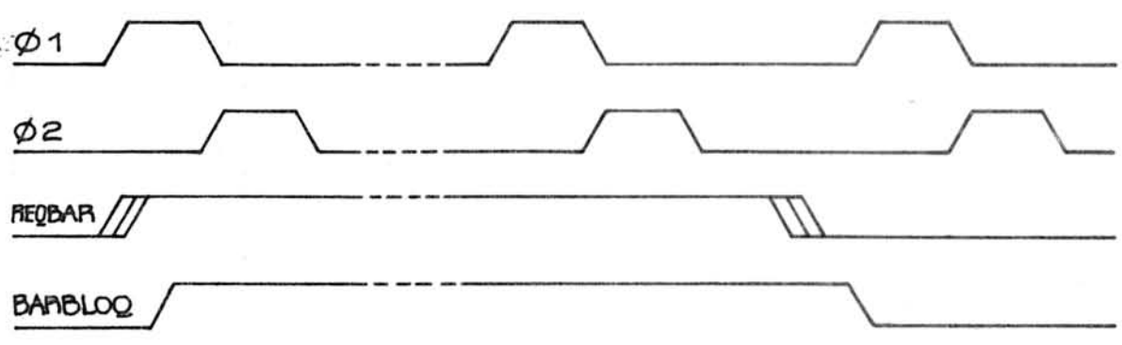


Figura 65 - Ciclo de bloqueio de barramento

O diagrama lógico do multiplexador é visto na figura 66.

Resta ainda citar uma possível alteração no sentido de desvincular o tempo de ciclo da memória e o número mínimo de janelas. Conforme salientado anteriormente (ver página 130), o número mínimo de janelas está condicionado ao tempo de ciclo da memória empregada. Esta relação não mais precisa ser observada desde que um novo sinal seja incluído no barramento. Este deve indicar a validade dos dados numa operação de leitura. Assim sendo, o tempo de ciclo da memória pode estender-se por um período maior que um ciclo de janelas. Em conseqüência disso torna-se desencessário fixar um número mínimo de janelas em função da memória. Evita-se, deste modo, a geração de janelas ociosas (não utilizadas por nenhum processador) destinadas a garantir a relação $t_c < n.t_j$, situação esta que pode ocorrer em configurações onde n é pequeno.

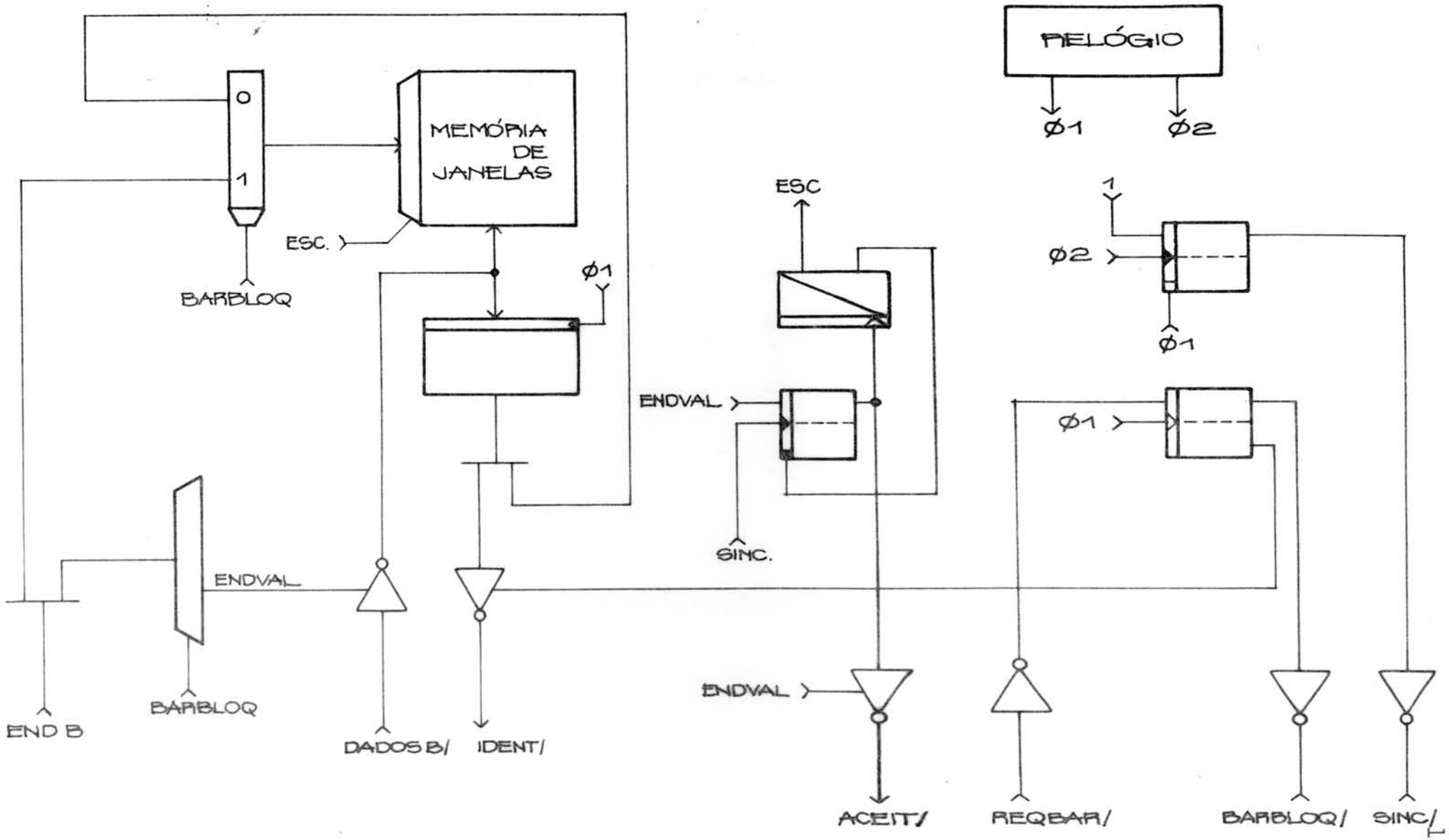


Figura 66 - Diagrama lógico do multiplexador

5.3.3 Interface de processador

A interface de processador converte os sinais de acesso à memória provenientes do microprocessador adaptando-os ao protocolo de barramento.

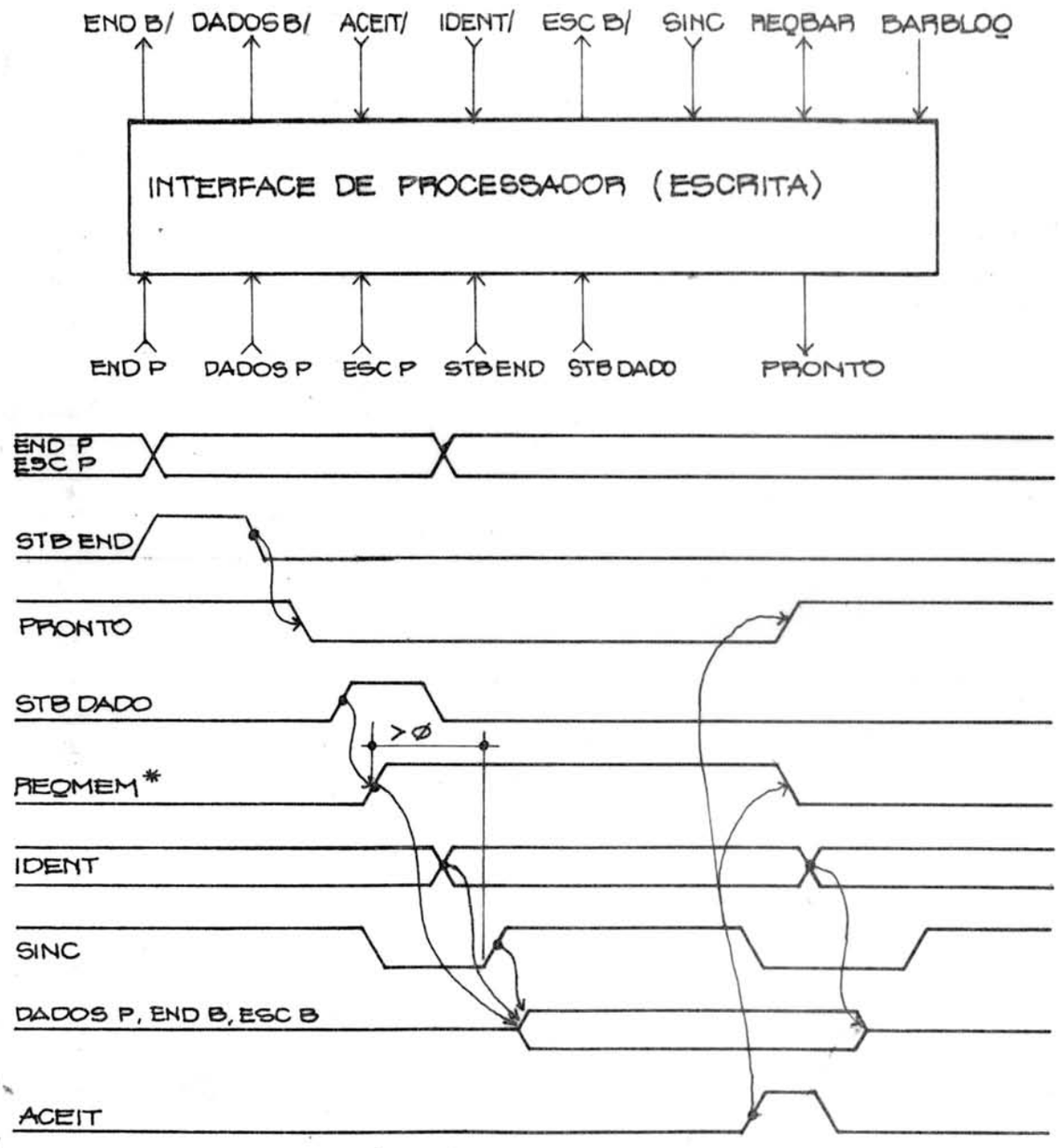
Para uma melhor compreensão de seu funcionamento, são definidos separadamente os protocolos de escrita e leitura, juntamente com o diagrama lógico que executa cada um deles. Posteriormente, as duas interfaces assim obtidas são fundidas em uma só para formar o interface de processador.

A interface de escrita em termos de sinais de entrada e saída e o protocolo a ser observado (figura 67) são descritos a seguir.

Alguns mnemônicos dos sinais finalizam com as letras B e P para indicar se o sinal é de barramento ou do processador.

Um ciclo de escrita se inicia com o microprocessador fornecendo os sinais de endereço e estado. Neste ponto já pode-se distinguir entre um acesso a memória local ou global, com a decodificação de endereço de acordo com o mapeamento pré estabelecido.

O sinal STBEND armazena o endereço e imediatamente baixa o sinal PRONTO. Este informa ao microprocessador para prolongar o seu ciclo. Em uma operação de escrita, o pedido de acesso a memória só pode ser registrado na interface quando os dados estão disponíveis. O sinal STBDADO armazena os dados e registra o pedido. Isto se faz necessário tendo em vista que os microprocessadores amostram o sinal PRONTO antes de acionar STBDADO. Desta maneira impede-se que dados inválidos sejam es-



* REQMEM É UM ESTADO INTERNO A INTERFACE. INDICA UMA TENTATIVA DE ACESSO A MEMÓRIA GLOBAL, AINDA NÃO CONCLUÍDA.

Figura 67 - Diagrama de tempos da interface de processador (escrita)

critos se a janela chega entre a subida de STBEND e STBDADO.

A transferência de dados e endereço é feita assincronamente por um protocolo bidirecional assíncrono não-entrelaçado. O sinal SINC informa a presença de endereços e dados válidos no barramento. O sinal ACEIT, embora gerado sempre na descida de SINC, é assíncrono pois ele surge somente quando o módulo está livre. Embora a velocidade da memória seja transparente ao microprocessador, o problema de conflitos não o é. Daí a necessidade de um protocolo assíncrono.

A identificação de janela em conjunto com uma requisição de acesso à memória causa a abertura das portas de endereço e dados na subida do sinal SINC. Endereço, dados e sinal de escrita são propagados ao barramento.

O sinal ACEIT informa a aceitação do pedido de escrita pela memória. Sua chegada causa o acionamento do sinal PRONTO permitindo ao microprocessador prosseguir. Isto é possível já que o micro não atua diretamente sobre a memória. Ele também causa a retirada dos dados e endereço do barramento, uma vez que já devem ter sido copiados na interface de memória. Se o sinal ACEIT não é recebido em virtude do módulo endereçado estar ocupado, nada acontece. A interface aguardará a próxima janela para resubmeter o pedido. A figura 68 resume a lógica de escrita na interface de processador.

O protocolo de leitura difere de uma escrita por serem necessárias duas janelas para se completar a operação (figura 69). Na primeira é transferido o endereço a ser lido e o sinal de leitura (fase 1). Na segunda os dados contidos no referido endereço são passados ao processador (fase 2).

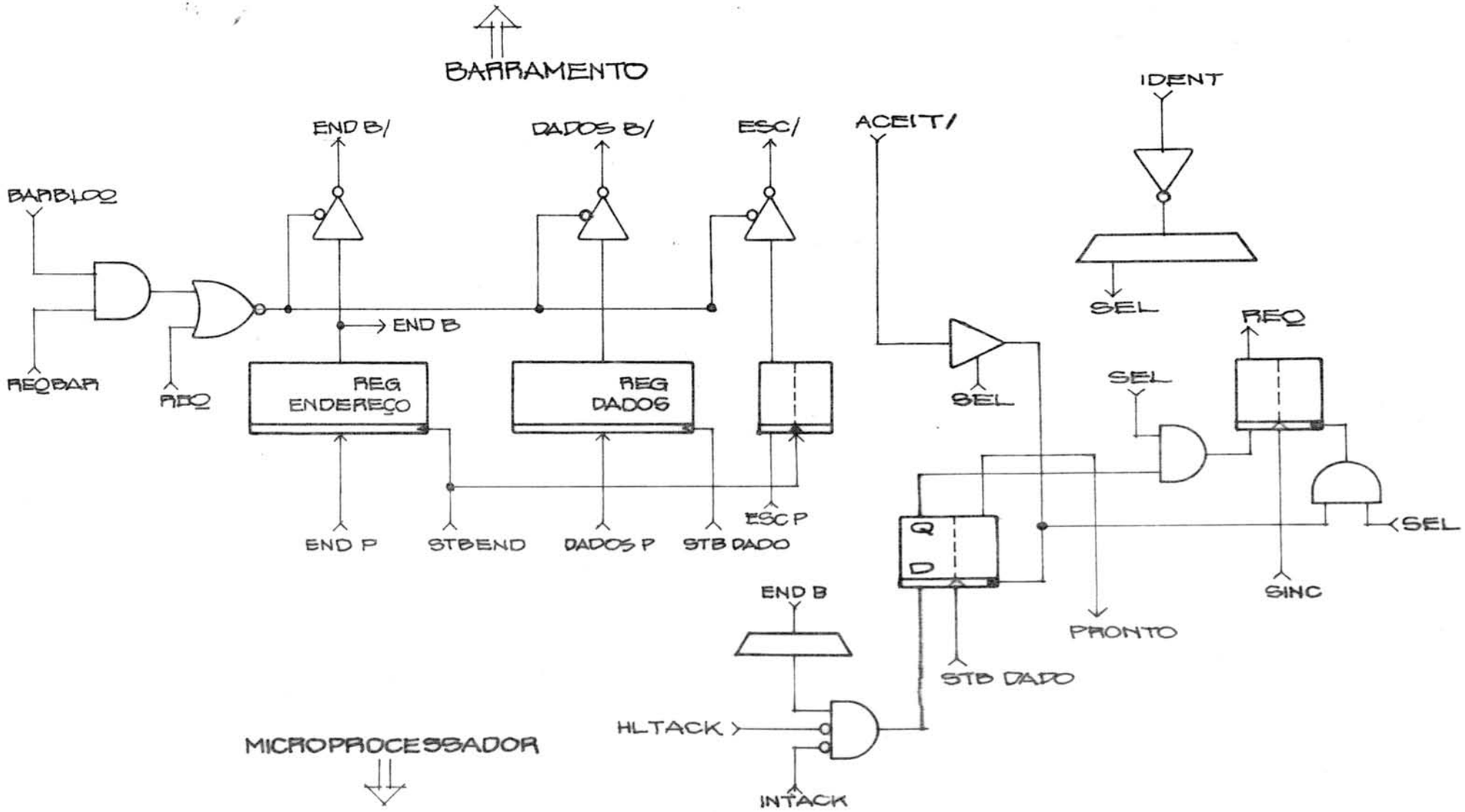


Figura 68 - Barramento multiplexado: interface de escrita do processador

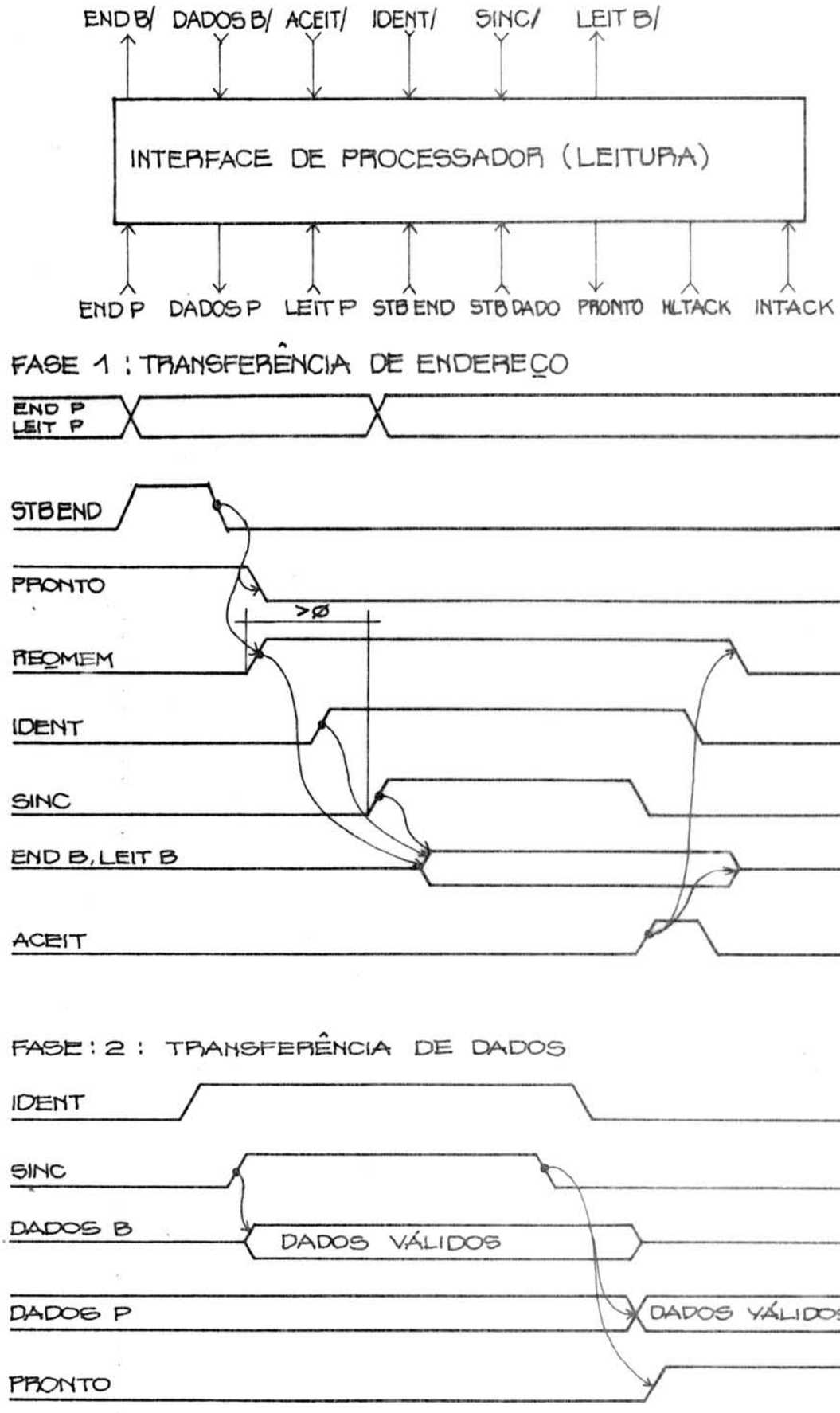


Figura 69 - Diagrama de tempos da interface de processador (leitura)

O ciclo de leitura se inicia, da mesma forma que a escrita, com o microprocessador fornecendo um endereço da memória global e o sinal de leitura LEITP. Agora, porém, o pedido pode ser registrado imediatamente sem necessidade de esperar pelo sinal STBDADO. Basta a identificação de janela para se iniciar a fase 1 (transferência de endereço). O sinal ACEIT marca o fim da fase 1. O endereço é retirado do barramento mas o PRONTO permanece desativado. A interface aguarda a sua próxima janela na qual os dados são copiados e o PRONTO ativado. Isto ocorre na descida do sinal SINC para dar tempo ao módulo de abrir suas portas de saída para o barramento. O diagrama lógico da interface de leitura é mostrado na figura 70.

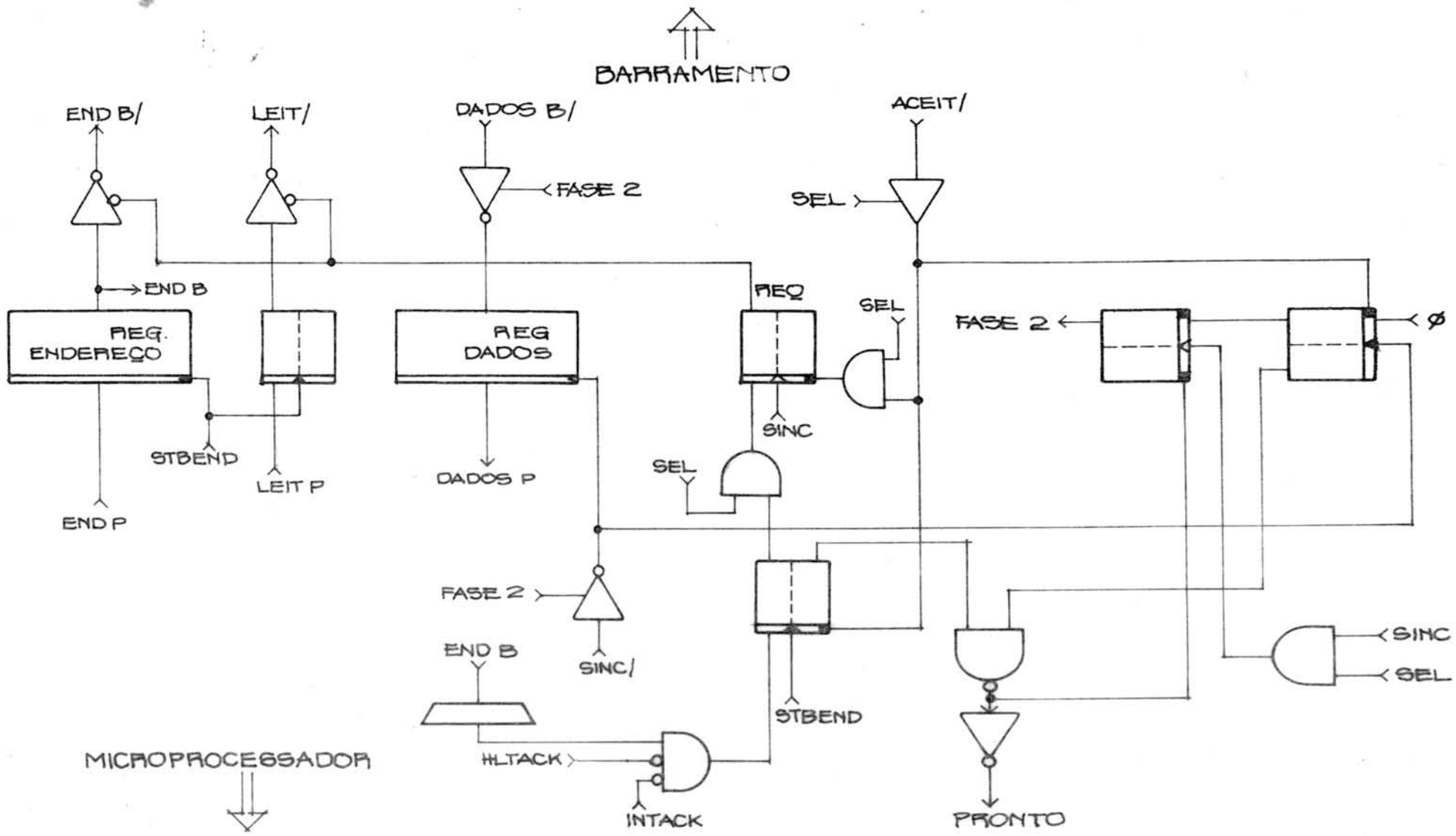


Figura 70 - Barramento multiplexado: interface de leitura do processador

A fusão dos dois diagramas fornece a interface de processador completa. Ela é formada por uma parte operacional (figura 71) e uma parte de controle (figura 72).

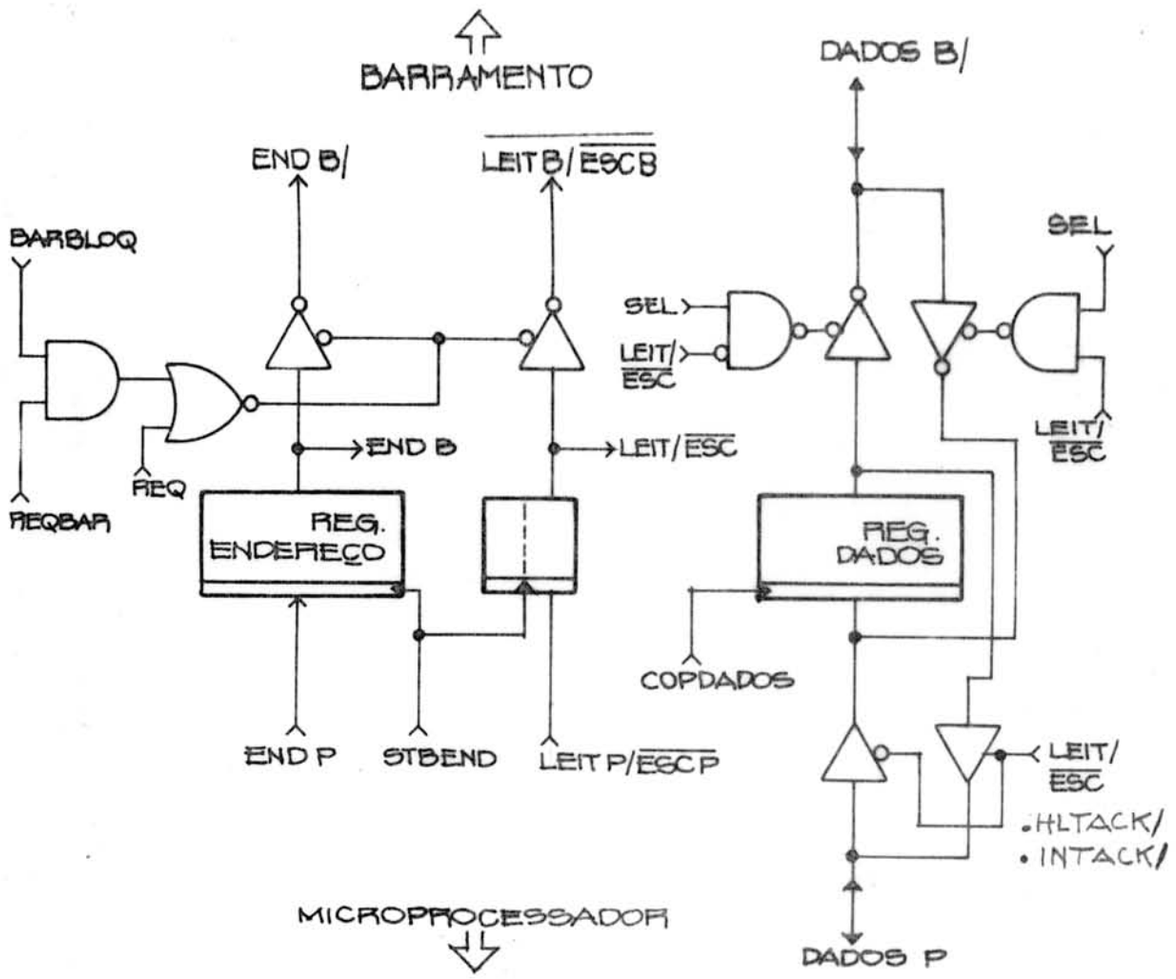


Figura 71 - Barramento multiplexado: interface de processador (parte operacional)

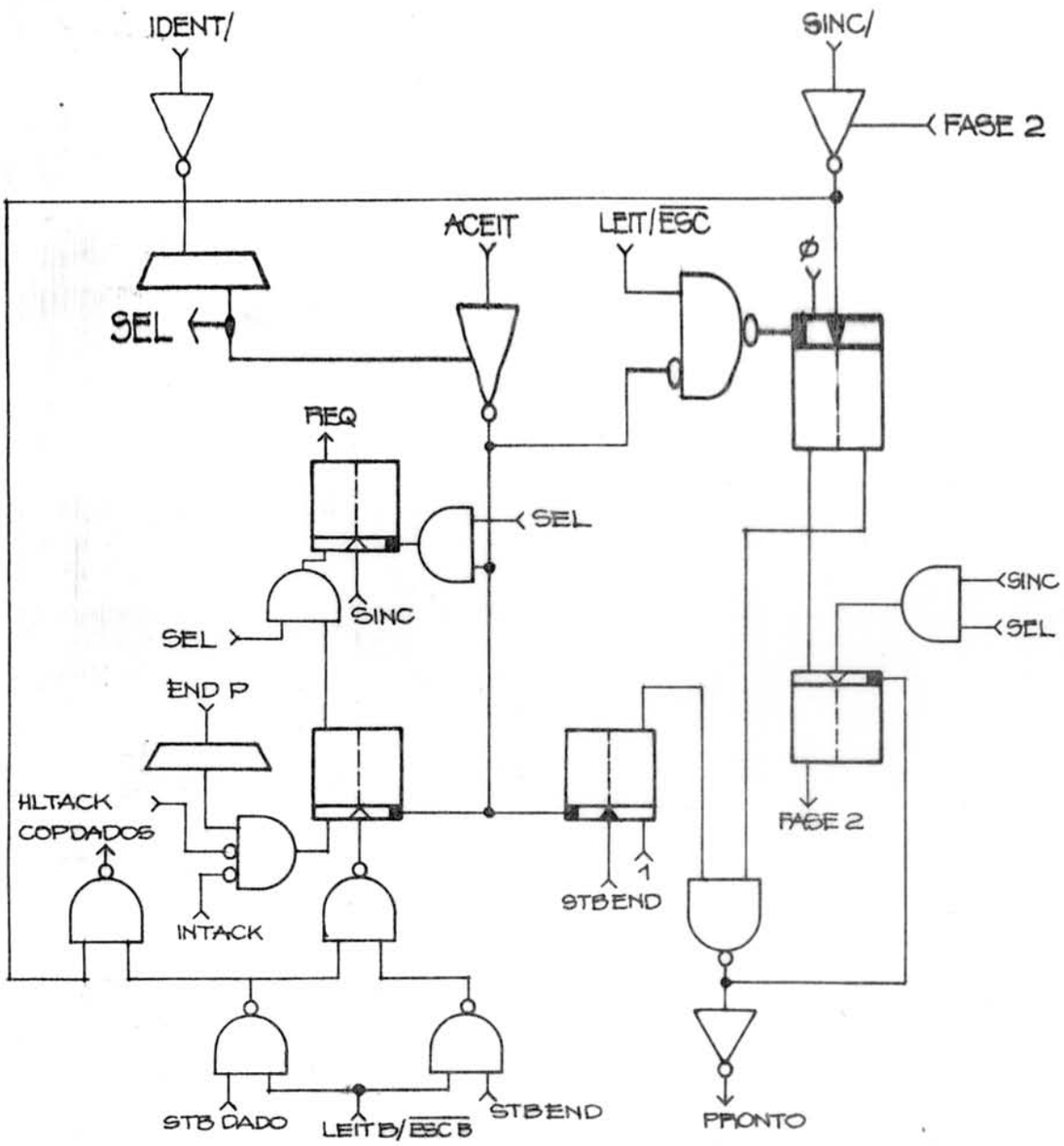


Figura 72 - Barramento multiplexado: interface de processador (controle)

5.3.4 Interface de memória

A interface de memória é responsável pelas seguintes tarefas:

- identificar uma tentativa de acesso ao módulo ao qual pertence.

- responder ou não às tentativas de acesso com base em seu estado (ocupada).

- armazenar as informações necessárias (endereço, dados, controle, identificação) para executar a operação requisitada.

- gerar sinais de leitura e escrita compatíveis com o tipo particular de memória utilizado.

Analogamente à interface de processador, os protocolos de leitura e escrita e suas interfaces são definidos em separado. Posteriormente ambos são integrados formando a interface de memória.

A figura 73 ilustra os sinais de entrada e saída e o diagrama de tempos em uma operação de escrita.

O sinal ACEIT é gerado na descida do sinal SINC ao ser reconhecido um endereço pertencente ao módulo caso este não esteja ocupado. A aceitação do pedido de acesso causa a passagem de endereço e dados desde a entrada da interface até a memória propriamente dita. Os tempos t_{EE} e t_{PE} são, respectivamente, o tempo em que o endereço deve permanecer estável antes do pulso de escrita e a largura deste pulso. Ambos são definidos em função das especificações da memória usada para implementar o módulo.

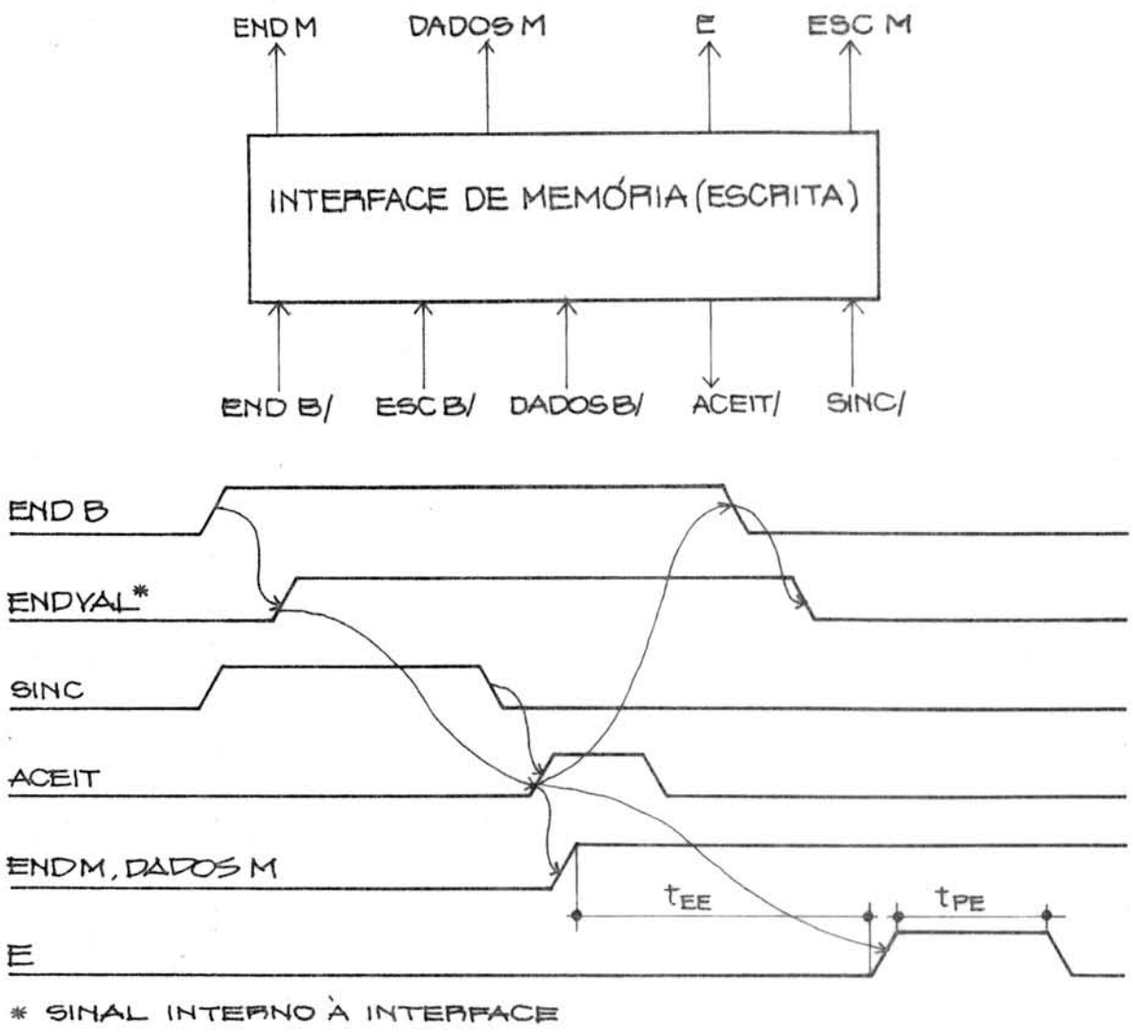


Figura 73 - Diagrama de tempos da interface de memória (escrita)

A figura 74 resume a lógica necessária à interface de memória para realizar uma operação de escrita.

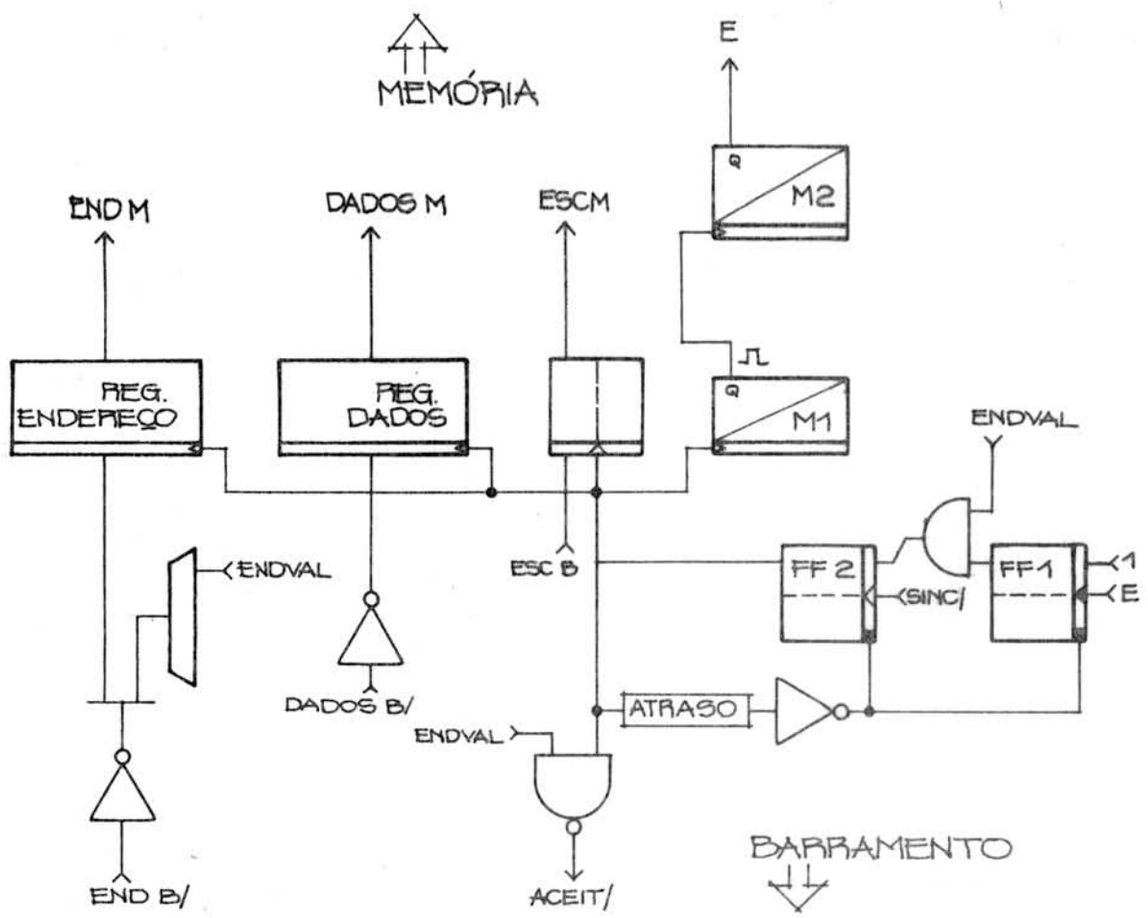


Figura 74 - Barramento multiplexado: interface de memória (escrita)

O flip-flop FF1 indica o estado do módulo(1-livre, 0-ocupado). Ele é desligado na descida de SINC se há um endereço válido na entrada da interface e ligado na descida do pulso de escrita (fim de operação). A amostragem de FF1 na descida de SINC exige um segundo flip-flop (FF2). Isto é necessário pois a memória opera assincronamente em relação aos processadores. O estado do módulo pode mudar de ocupado para livre a qualquer instante. O flip-flop FF2 isola o pulso ACEIT da mu-

dança de estado, impedindo que o sinal ACEIT seja gerado muito tarde na janela em um instante em que o processador não pode mais reconhecê-lo. A largura do sinal ACEIT é determinada pela quantidade de atraso introduzida pelo circuito de retardo.

Os monoestáveis M1 e M2 asseguram os tempos t_{EE} e t_{PE}.

A figura 75 ilustra os sinais de entrada e saída e o diagrama de tempos de uma operação de leitura.

A fase 1 se processa de forma idêntica a uma operação de escrita exceto que o módulo não é liberado ao final do pulso de leitura. Isto só pode ser feito na próxima janela do processador que iniciou a leitura (fase 2). O diagrama lógico da interface de leitura é mostrado na figura 76.

A diferença em relação à interface de escrita é a presença de um monoestável (M3) para copiar os dados lidos. A largura do pulso é determinada em função do tempo de acesso da memória. Os monoestáveis M1 e M2 são responsáveis pelos tempos t_{EL} (endereço estável antes do pulso de leitura) e t_{PL} (tempo mínimo de leitura), respectivamente.

É necessário, também, um circuito para detectar a janela na qual as portas de saída para o barramento serão abertas (fase 2). O circuito consta de um registrador para armazenar o código da janela, um comparador e um flip-flop para amostrar cada janela subseqüente à fase 1. O flip-flop é necessário para impedir que a comparação ative o sinal HABSAIDA ainda durante a fase 1. A ativação de HABSAIDA é irrelevante quanto ao barramento de dados na fase 1. Porém, ao fim da janela, FF1 seria ligado pela queda de HABSAIDA, liberando desta maneira o módulo. Qualquer tentativa de acesso subseqüente seria aceita

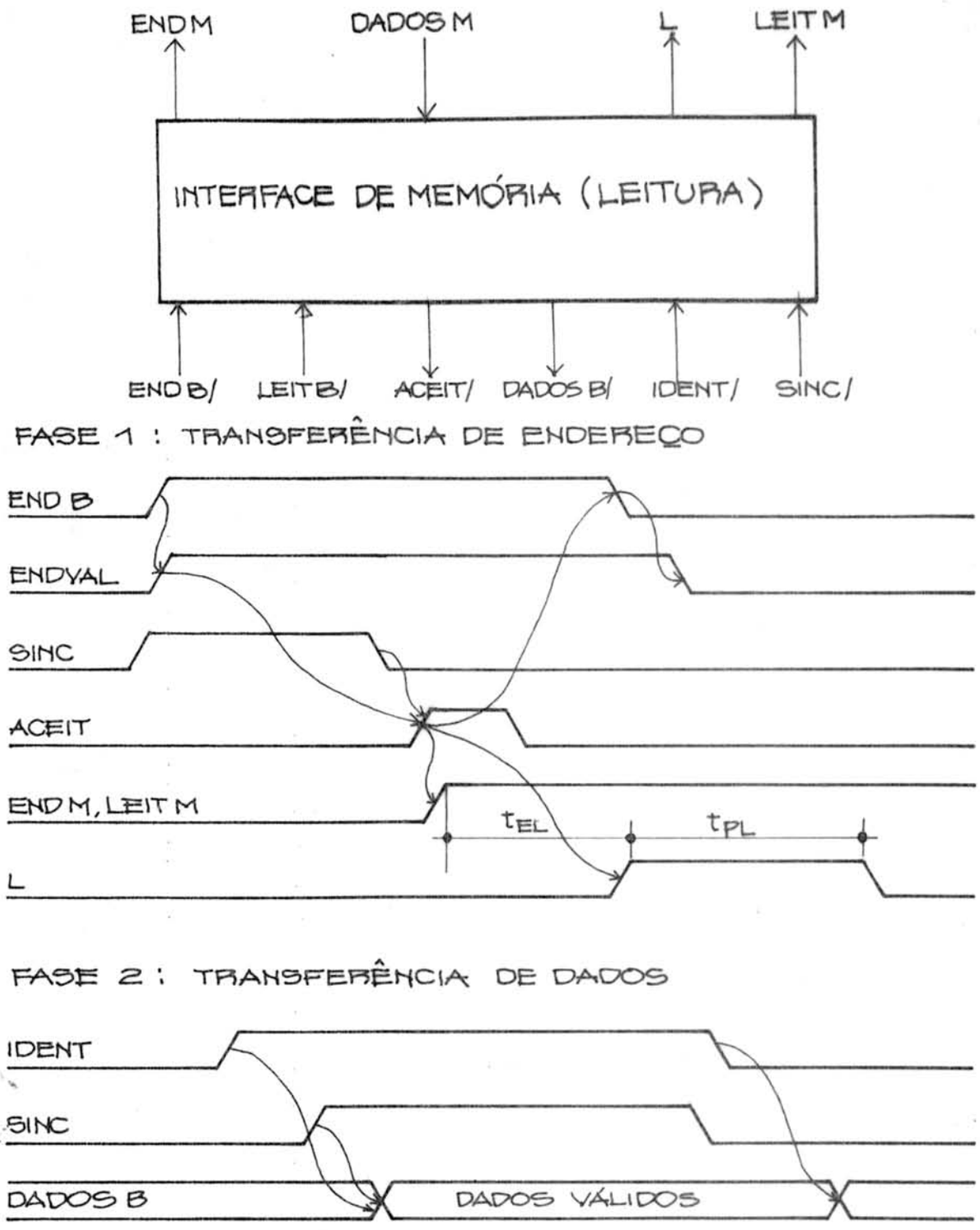


Figura 75 - Diagrama de tempos da interface de memória (leitura)

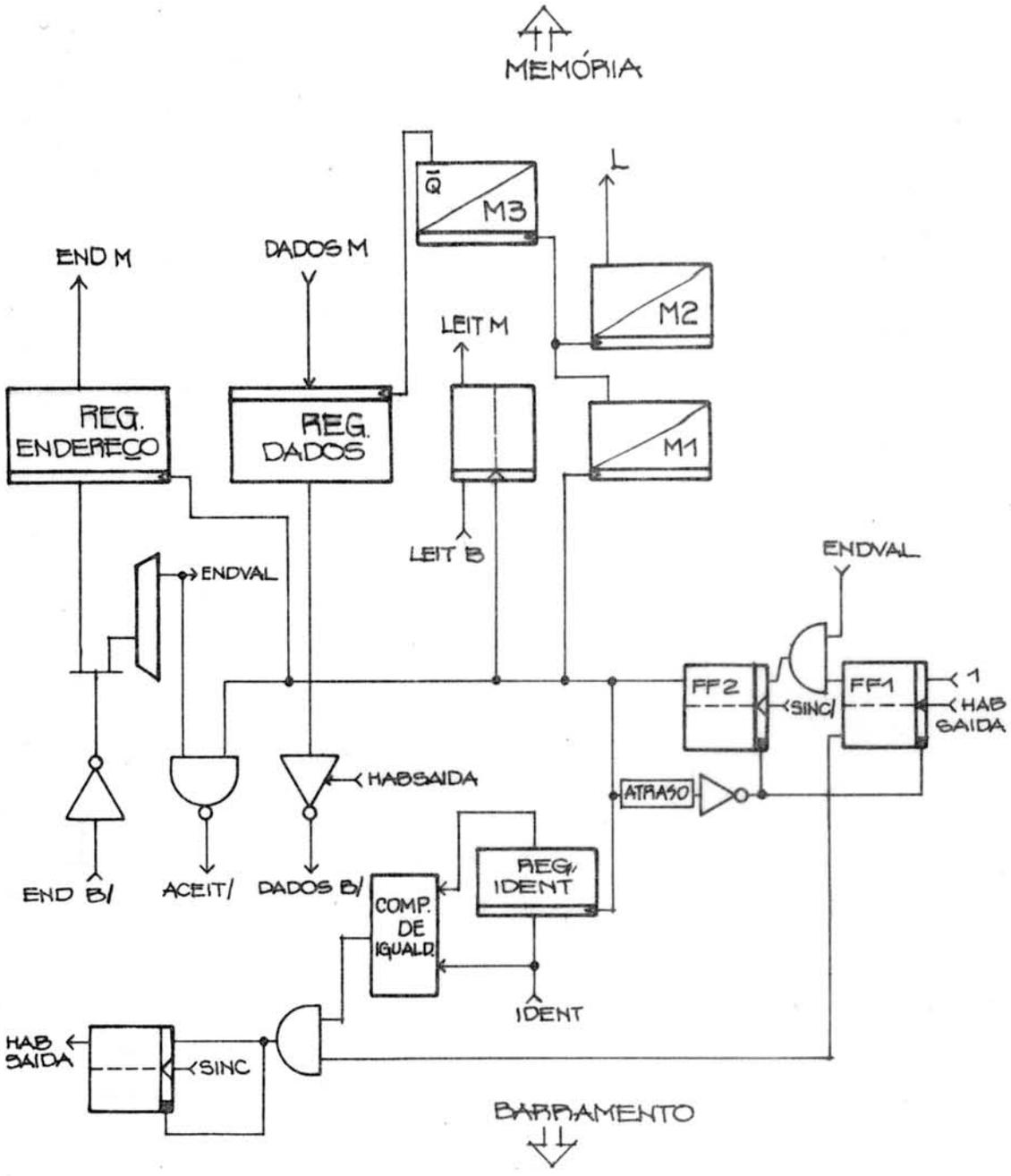


Figura 76 - Barramento multiplexado: interface de memória (leitura)

e, em conseqüência, os dados da leitura destruídos.

Analogamente à interface de processador, a interface de memória é composta de uma parte operacional (figura 77) e uma parte de controle (figura 78).

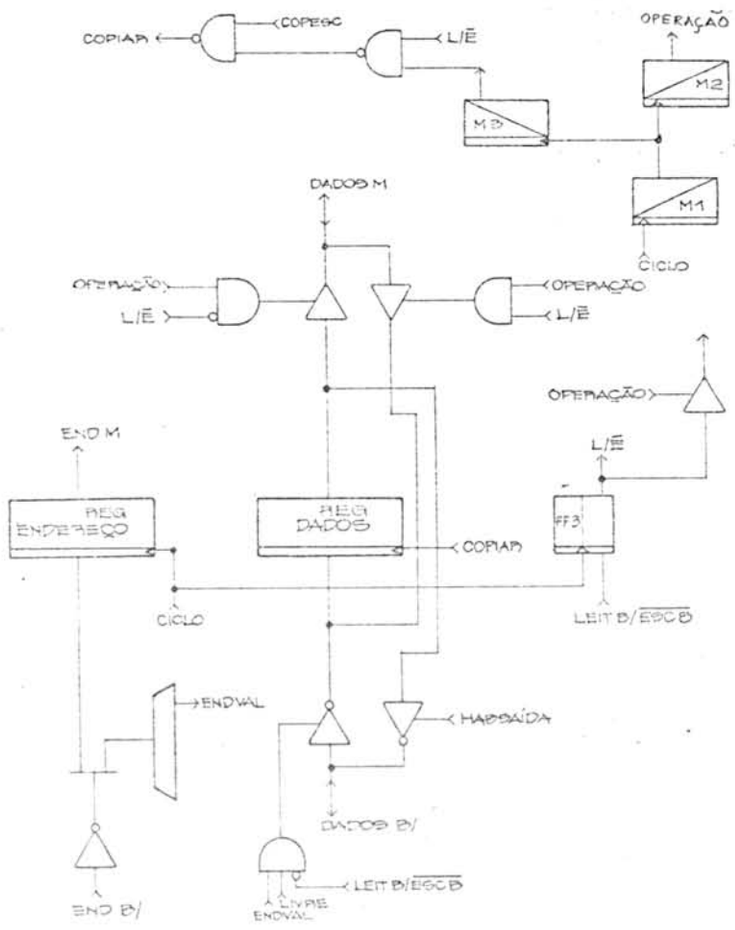


Figura 77 - Barramento multiplexado: interface de memória (parte operacional)

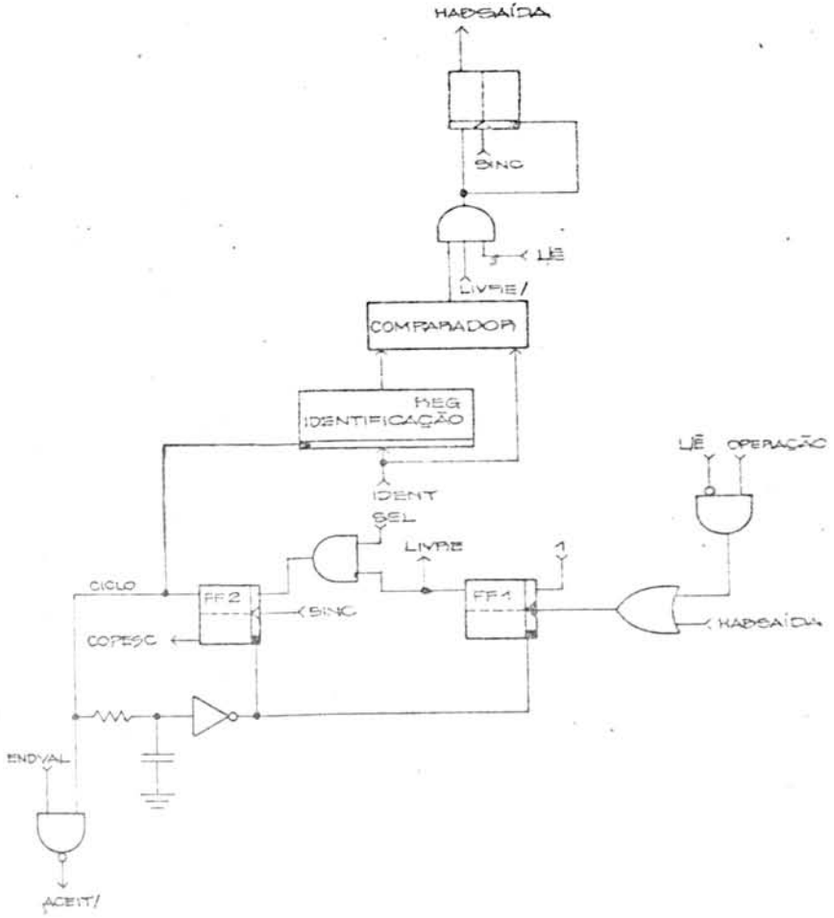


Figura 78 - Barramento multiplexado: interface de memória (controle)

O sinal OPERAÇÃO que não aparecia nas interfaces anteriormente é o sinal que habilita a escrita ou leitura de acordo com o flip-flop FF3. É o "OU" lógico dos sinais L e E usados anteriormente. Os diagramas de tempo dos sinais vistos pela memória estão na figura 79. Seu significado e os requisitos estão na tabela 4.

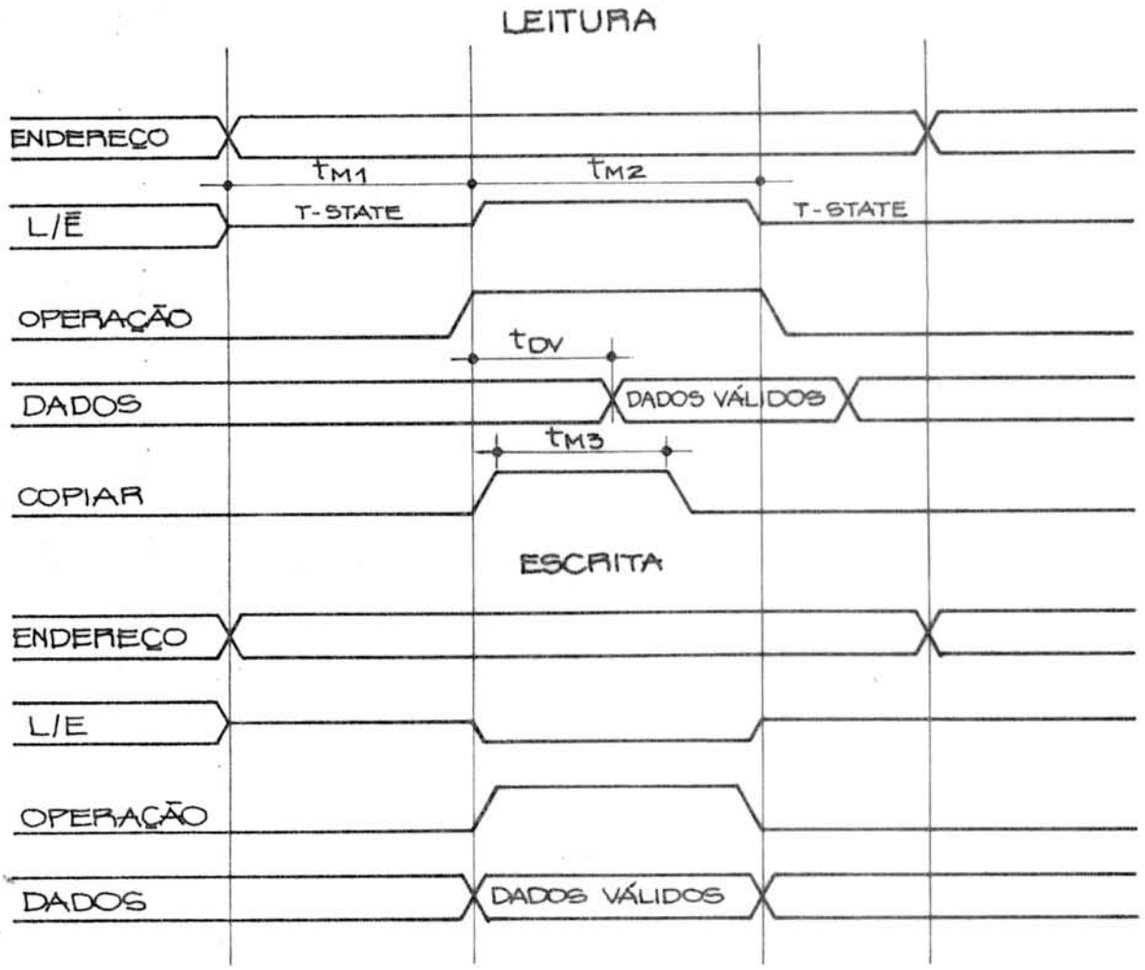


Figura 79 - Diagrama de tempos dos sinais vistos pela memória

SINAL	SIGNIFICADO	REQUISITO
t_{M1}	Largura de pulso do monoestável 1	$\geq t_c - t_{M2}$
t_{M2}	Largura de pulso do monoestável 2	$\geq \max (t_E, t_L)$
t_{M3}	Largura de pulso do monoestável 3	$\geq t_{DV}, < t_{M2}$
t_c	Tempo de ciclo da memória	$\leq N x t_J$ (N=nº de janelas)
t_E	Tempo de escrita da memória	-
t_L	Tempo de leitura	-
t_{DV}	Dados válidos a partir do início da leitura	-

Tabela 4 - Sinais de acionamento da memória

5.4 Mecanismo para acessos indivisíveis

Duas hipóteses foram consideradas com vistas a implementação de um mecanismo que assegurasse acessos indivisíveis. Uma solução para este problema é bloquear o barramento antes de iniciar o acesso, da mesma forma que é feito para alterar a memória de seqüência de janelas. Esta hipótese foi descartada pelos seguintes motivos:

1) diminuição da performance do sistema;

2) necessidade de alteração nas interfaces de memória e processador;

3) necessidade de alterar o circuito de bloqueio de barramento;

4) "não-transparência" da operação indivisível.

O primeiro motivo é decorrente do bloqueio do barramento durante as operações indivisíveis. Apenas um processador e um módulo de memória ficariam ativos durante este tempo. Apesar de o número de operações indivisíveis representarem uma parcela pequena da atividade na memória o atraso seria agravado pela necessidade de ativar o sinal BARBLOQ somente após um ciclo completo de janelas a partir do instante de ativação do sinal REQBAR. Esta espera é necessária para a complementação de possíveis ciclos de leitura em andamento. Além disso, nenhuma tentativa de leitura poderia ser aceita após a ativação de REQBAR. Ambas as exigências tornariam as interfaces mais complexas.

O segundo motivo (necessidade de alteração nas interfaces) é decorrente da dualidade de filosofias de transferência; as interfaces teriam de suportar operações na memória sem a presença de janelas. Isto acarretaria uma complexidade excessiva nas interfaces.

O circuito de bloqueio de barramento teria de ser substituído por um árbitro centralizado tendo em vista que o acesso à linha REQBAR não poderia depender de primitivas de sincronização de alto nível.

O quarto motivo decorre da exigência de toda operação indivisível ser precedida de um pedido de bloqueio de barramento. Isto implica onerar o sistema operacional com mais esta tarefa. Observa-se, entretanto, que a migração vertical de funções[10] do sistema operacional para o hardware ou microprograma é uma tendência.

Devido as desvantagens constatadas e seguindo-se es

ta última orientação, optou-se pela implementação do mecanismo através de um módulo de memória especial. Nele, um certo número de endereços é designado para as operações de acesso indivisível. Uma operação de leitura em uma destas posições causa a escrita automática de 1's em todos os bits do byte lido. A leitura e posterior escrita são feitas de forma indivisível sob o controle da lógica de interface. Uma operação de escrita não causa a inserção de 1's, de maneira que as variáveis podem ser resetadas através da escrita simplesmente. A operação indivisível é transparente ao programa, podendo ser executada a qualquer instante através de uma operação de leitura na memória.

O módulo de memória que contém as variáveis deve ter sua interface alterada. As modificações necessárias em relação a um módulo comum são poucas (figura 80).

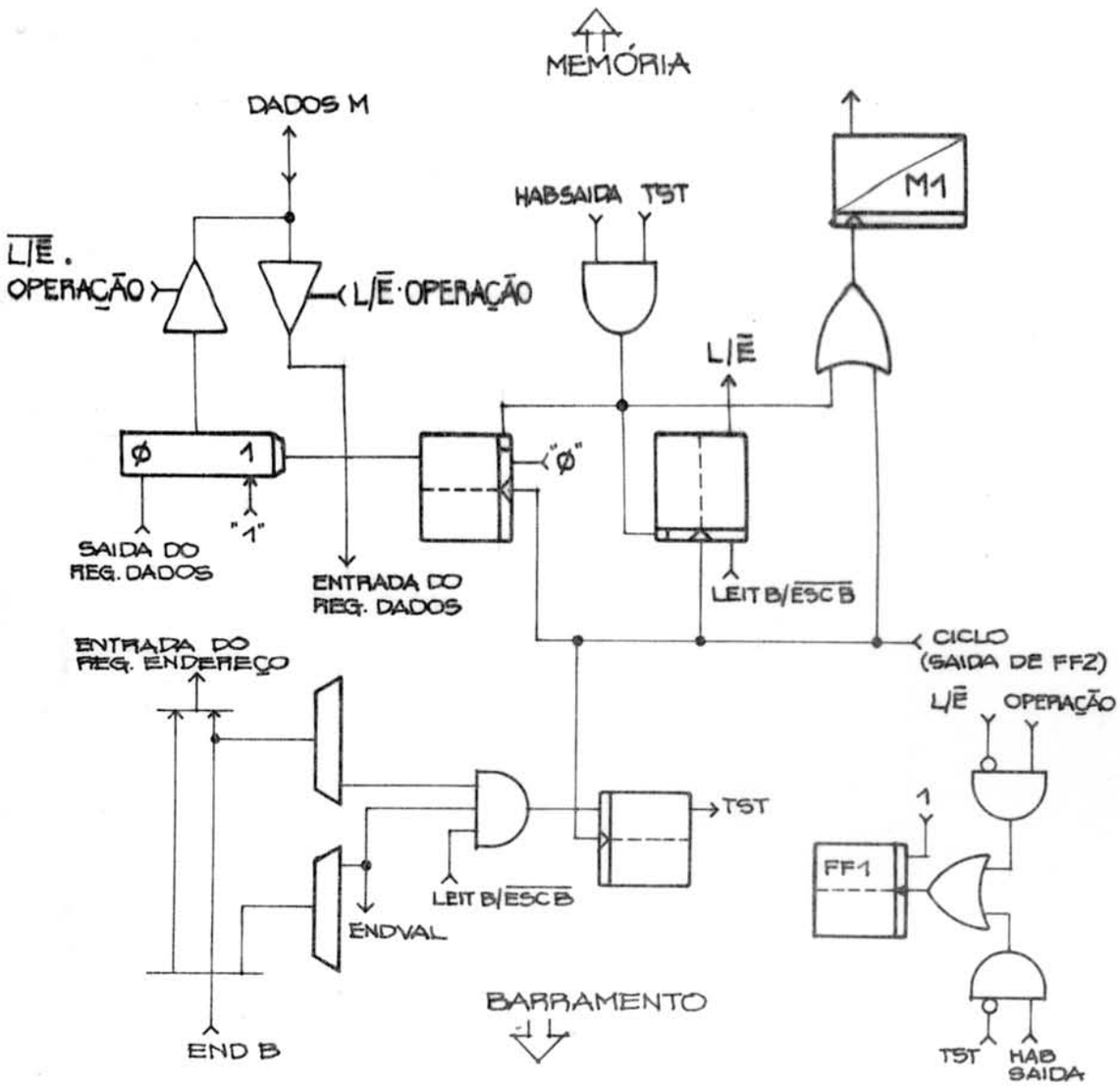


Figura 80 - Interface de memória para operações indivisíveis

6 BARRAMENTOS DEDICADOS/MEMÓRIAS MULTIPORTA

6 BARRAMENTOS DEDICADOS/MEMÓRIAS MULTIPORTA

6.1 Introdução

A segunda alternativa apresentada para a estrutura de interconexão da figura 51 consiste em dedicar um barramento a cada processador (figura 81).

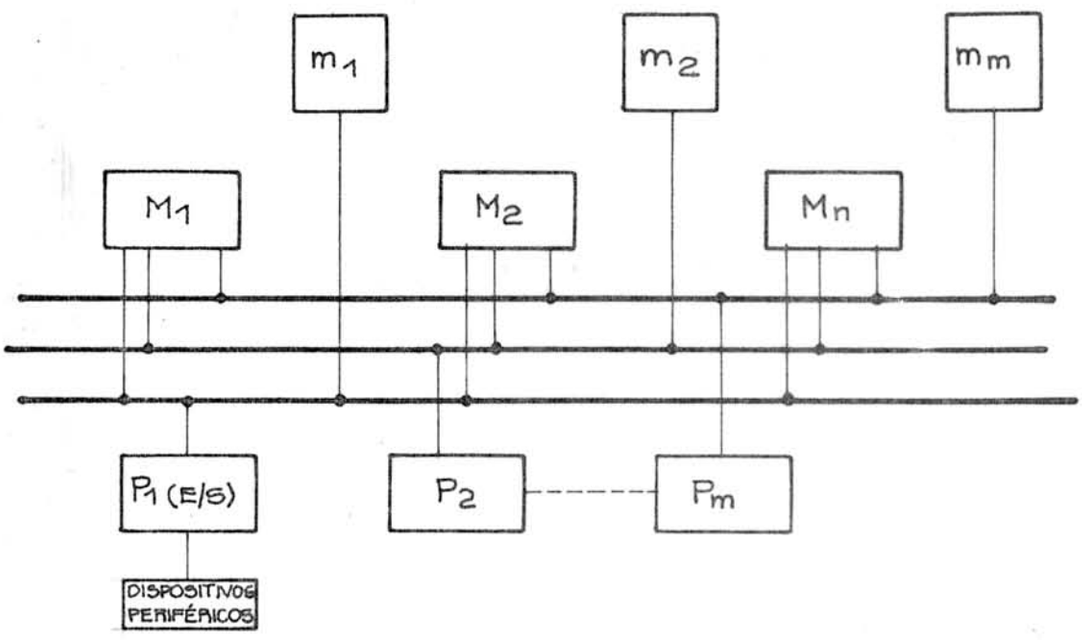


Figura 81 - Organização do multimicroprocessador com barramentos dedicados/memórias multiporta

A dedicação dos barramentos permite dispensar o árbitro. Entretanto, há um aumento considerável de interconectividade física como se pode ver na figura.

Com esta configuração uma maior ênfase é dada à performance do sistema em detrimento da facilidade de interconexão do barramento único. Cada processador comunica-se com a

memória através de seu barramento. O acesso a este barramento é vedado aos demais processadores. A independência entre os barramentos possibilita o paralelismo real no acesso à memória; enquanto um processador executa uma operação sobre um módulo, um outro acesso pode acontecer entre outro par processador-módulo. O número máximo de transferências em paralelo é limitado pelo número de barramentos e módulos existentes. Se existem menos módulos de memória que barramentos, então em nenhum momento estarão todos os barramentos ocupados. Se, entretanto, o número de módulos excede o de barramentos nunca irá acontecer a situação de ocupação total dos módulos. A situação ideal é a igualdade entre processadores, barramentos e módulos. Assim, é possível atingir o paralelismo máximo sem introduzir uma capacidade ociosa nos barramentos ou na memória.

Entretanto, a situação de paralelismo máximo (todos os barramentos e módulos em operação) não é comum. Isto só acontece quando os processadores geram endereços referentes a módulos distintos, pois um módulo só pode atender uma requisição a cada instante. Duas ou mais requisições simultâneas a um mesmo módulo são conflitantes, obrigando o módulo a atender apenas uma delas e adiar o atendimento da(s) outra(s). Estas funções são executadas por circuitos localizados na interface de cada módulo.

6.2 Interface de processador

A interface de processador para um sistema com múltiplos barramentos torna-se muito simples pois não há concor-

rência pelo barramento. Cada processador pode atuar com seus sinais a qualquer instante sem se preocupar com os demais. A existência de outros processadores compartilhando a mesma memória é transparente. A concorrência irá aparecer somente nos módulos de memória.

Como resultado, os sinais de processador irão atuar diretamente nas memórias. Este é um ponto muito importante: permitir que os sinais do micro participem ativamente no ciclo de memória significa um melhor aproveitamento das capacidades integradas. Como conseqüência diminui a quantidade de componentes necessários nas interfaces.

As funções primordiais desempenhadas pela interface são o registro de uma requisição de acesso à memória e a manutenção do microprocessador em estado de espera caso ela não seja atendida. O diagrama de tempo da interface (figura 82) ilustra a maneira como isto é realizado. O protocolo bidirecional não-entrelaçado entre os sinais STBEND e PRTO garante a manutenção do micro em estado de espera.

Os processadores operam assincronamente um em relação aos outros. A operação assíncrona pressupõe que os pedidos de acesso são apresentados à memória a qualquer instante. O ciclo é iniciado com o endereço e o sinal LEIT/$\overline{\text{ESC}}$ sendo transmitidos ao barramento. Nesse instante a requisição de acesso é reconhecida. O sinal PRONTO deve ser desativado dentro de um tempo bem definido a partir do início do ciclo, na hipótese de a memória não poder atender o pedido. Entretanto, este tempo difere de um microprocessador para outro e, além disso, ele é função da freqüência de relógio do microprocessador. Ao invés de impor ao árbitro de memória a resposta ao pedido dentro de

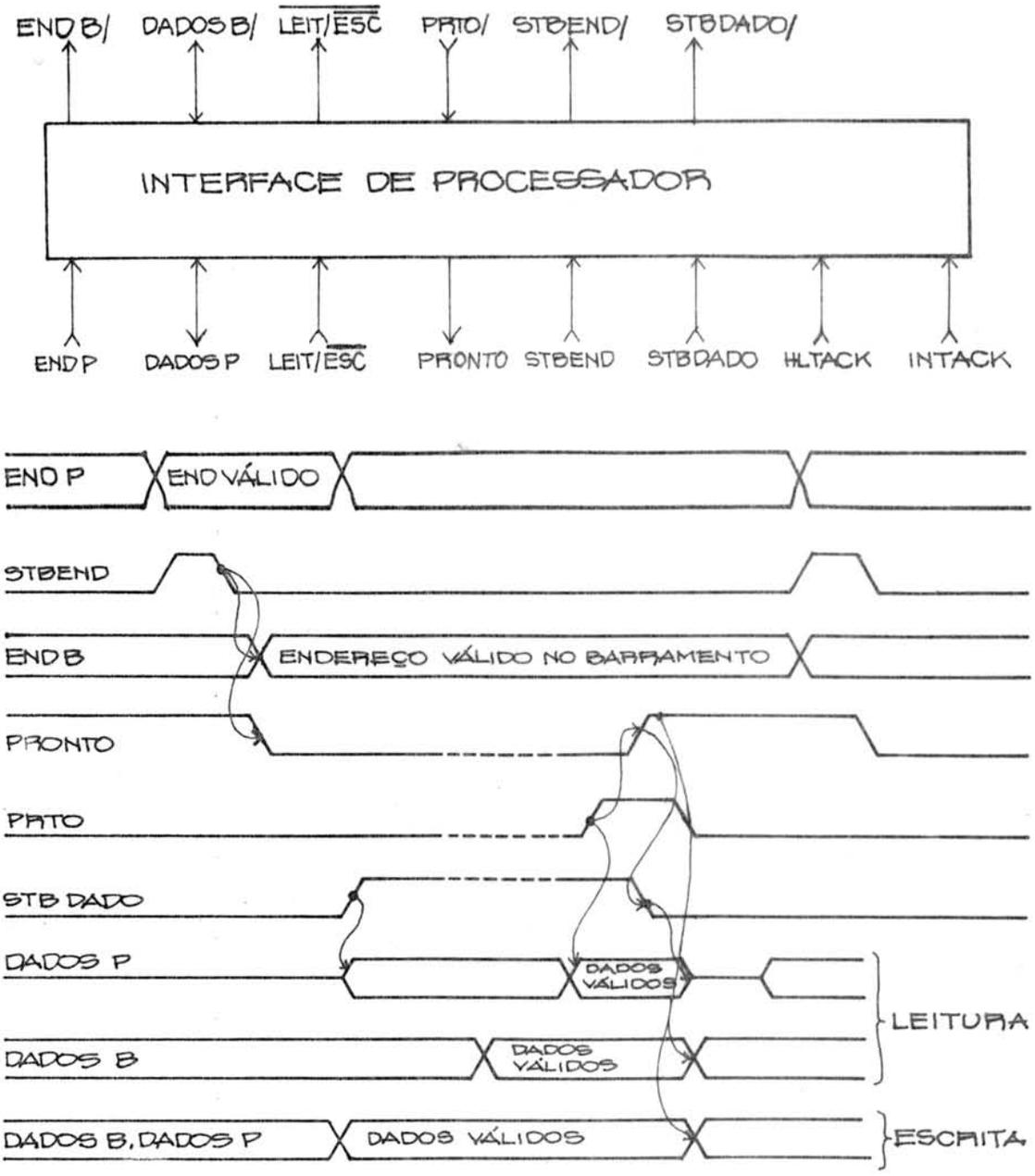


Figura 82 - Diagrama de tempo da interface de processador

um tempo máximo, o procedimento contrário é adotado. A ativação do sinal STBEND causa a descida imediata do sinal PRONTO, mesmo desconhecendo se o acesso será indeferido, ou seja, assume-se que o acesso não será completado no tempo normal. O PRONTO só é acionado quando a memória garantidamente teve tem-

po para concluir o acesso. O sinal PRTO proveniente do controlador do módulo endereçado causa a ativação do PRONTO e a retirada dos sinais do barramento.

A interface de processador (figura 83) provê um registrador de endereço para microprocessadores que multiplexam as linhas de endereço com outras funções. Ele não é necessário se não ocorre a multiplexação e o endereço permanece válido durante todo o ciclo. O registrador deve ser do tipo transparente ("transparent latch")[31] para que o endereço se propague até a memória e lá seja decodificado antes da chegada do sinal STBEND. Isto dá tempo aos módulos para reconhecerem uma tentativa de acesso e registrarem-na na descida do sinal STBEND.

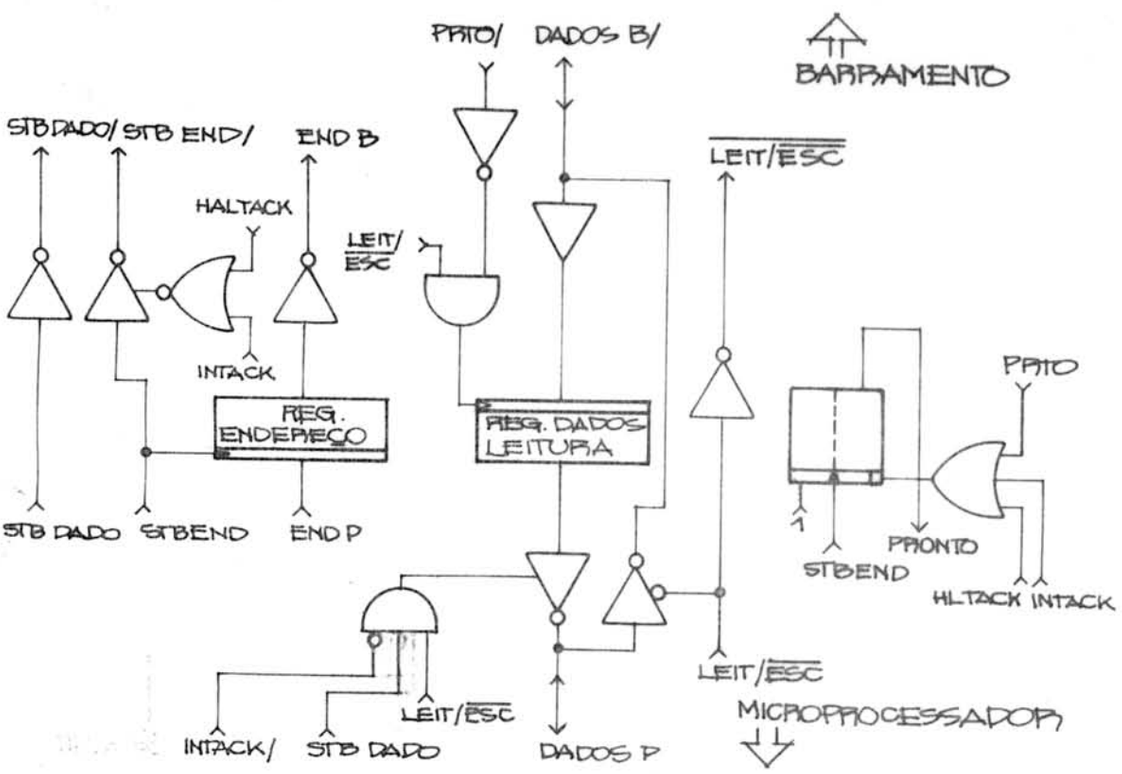


Figura 83 - Diagrama lógico da interface de processador

O registrador de dados armazena os dados provenientes da memória em uma operação de leitura. A carga é feita na borda ascendente do sinal PRTO. Desta maneira o tempo que o microprocessador leva para copiar os dados a partir da amostragem do sinal PRONTO (t_{PCD} da figura 53) torna-se transparente ao controlador de memória. Assim que é acionado PRTO o controlador de módulo pode iniciar o atendimento de uma requisição pendente. Portanto, ocorre uma sobreposição de operações no microprocessador e na memória desde o instante em que é gerado PRTO até o fim do ciclo de máquina. A inclusão do registrador de dados permite que a memória seja liberada o mais rapidamente possível, para atender uma nova requisição.

6.3 Interface de memória

A interface entre um módulo e os barramentos é composta de m portas e um controlador (figura 84).

6.3.1 Portas de barramento

As portas são o ponto de conexão entre a memória e os vários barramentos do sistema. A cada processador irá corresponder exatamente uma porta. O conjunto de portas de uma interface representa sua parte operacional. Elas são encarregadas da recepção e transmissão de sinais de e para os processadores, reconhecer tentativas de acesso e submetê-las ao controlador.

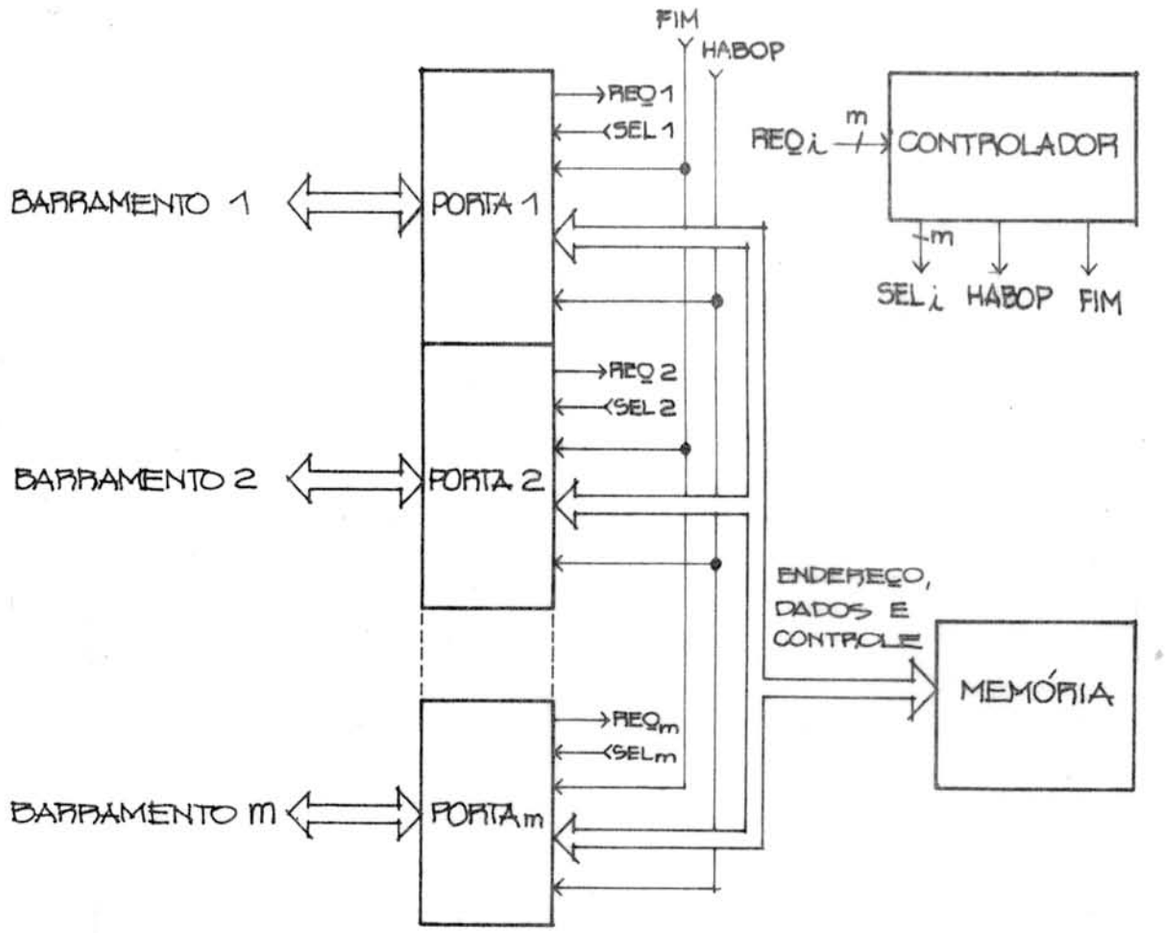


Figura 84 - Diagrama em blocos da interface de memória

Um módulo de memória pode executar somente uma operação de leitura ou escrita a cada instante. Portanto, apenas uma porta pode estar ativa a cada instante. Entende-se por "ativa" aquela que está habilitada pelo sinal SELi a fechar a conexão entre o seu barramento e a memória. As portas não-selecionadas devem manter o arranjo de memória isolado dos demais barramentos. O diagrama de tempos de uma porta é mostrado na figura 85.

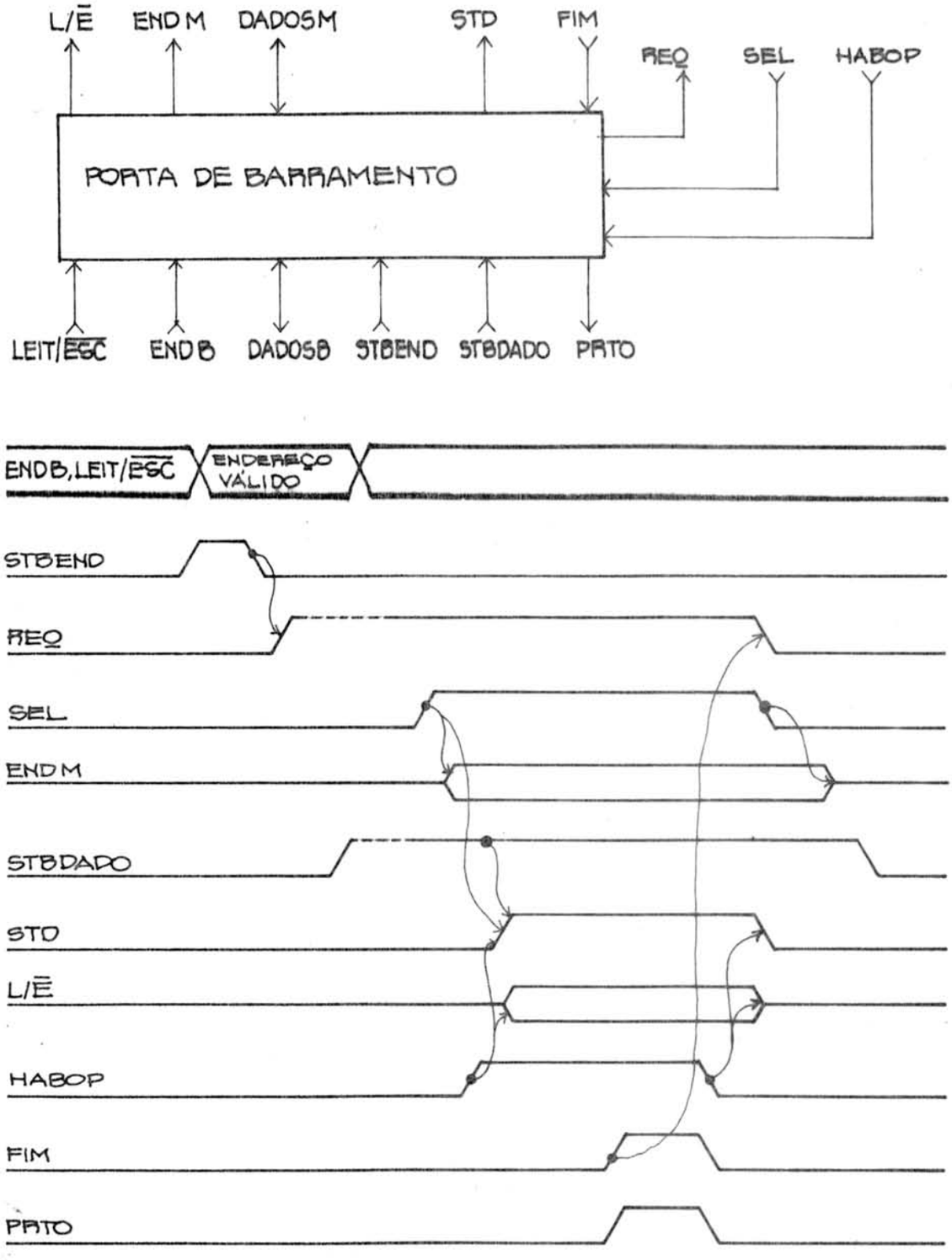


Figura 85 - Diagrama de tempos de uma porta

Uma porta deve reconhecer uma tentativa de acesso a qualquer instante. O assincronismo entre elas é uma exigên-

cia do próprio assincronismo entre os processadores.

O diagrama lógico de uma porta é mostrado na figura 86.

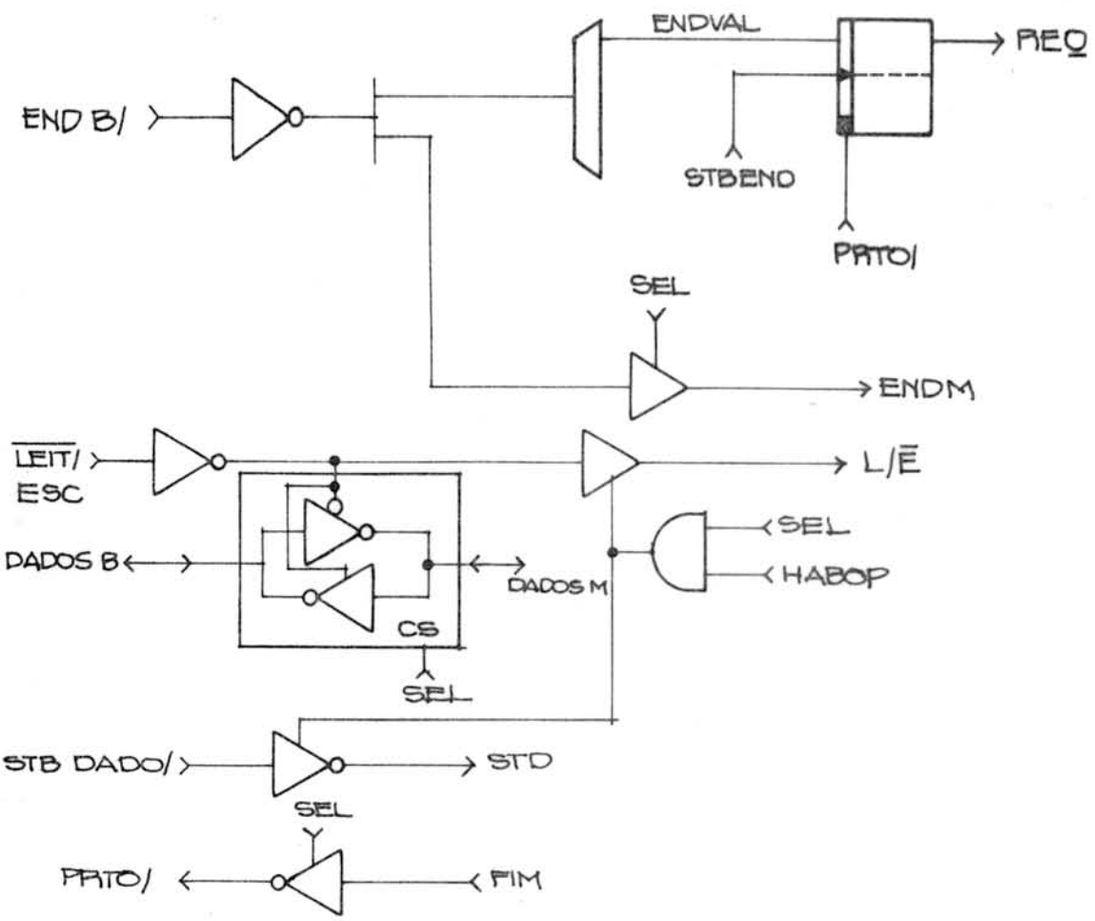


Figura 86 - Diagrama lógico de uma porta

O sinal STBEND propagado a todos os módulos amostra o endereço presente no barramento. A decodificação de um endereço válido só acontece em um dos módulos de acordo com a distribuição de endereço nos módulos. Este terá o sinal de requisição de acesso (REQ) ativado na porta em questão. Enquanto isso todos os demais sinais permanecem bloqueados na porta aguardando a resposta do controlador (SEL) indicando a aceitação da requisição. A chegada do sinal SEL marca o início do

ciclo de memória. A partir deste instante o controlador passa a marcar o tempo do ciclo e supervisionar a atividade no módulo sincronamente. A chegada do sinal SEL faz com que os sinais de endereço (e dados na escrita) passem livremente. Os sinais de controle do microprocessador (L/E e STD) ficam bloqueados até a chegada do sinal HABOP (habilita operação) no próximo pulso de relógio. Esta defasagem entre SEL e HABOP garante que o sinal de escrita não esteja ativo durante as transições de endereço, uma exigência de alguns tipos de memórias.

Quando a memória já teve o tempo necessário para ler ou escrever, o controlador aciona o sinal FIM, dirigido a todas as portas. Porém, apenas aquela que está selecionada propaga-o até o barramento (PRTO).

6.3.2 Controlador de módulo

O circuito controlador de módulo é responsável pela geração dos sinais de chaveamento das portas (SELi). Além disso, ele deve também marcar o tempo de um ciclo de memória com base nas especificações do tipo particular de memória implementada no módulo.

O chaveamento das portas está subordinado a um árbitro síncrono de requisições independentes (visto na parte superior da figura 87). As múltiplas requisições assíncronas são submetidas a uma rede de prioridade que seleciona uma delas para atendimento. O processo de seleção acontece em paralelo com a operação corrente, o que lhe confere velocidade. Assim que a operação é completada já existe uma outra selecionada pa

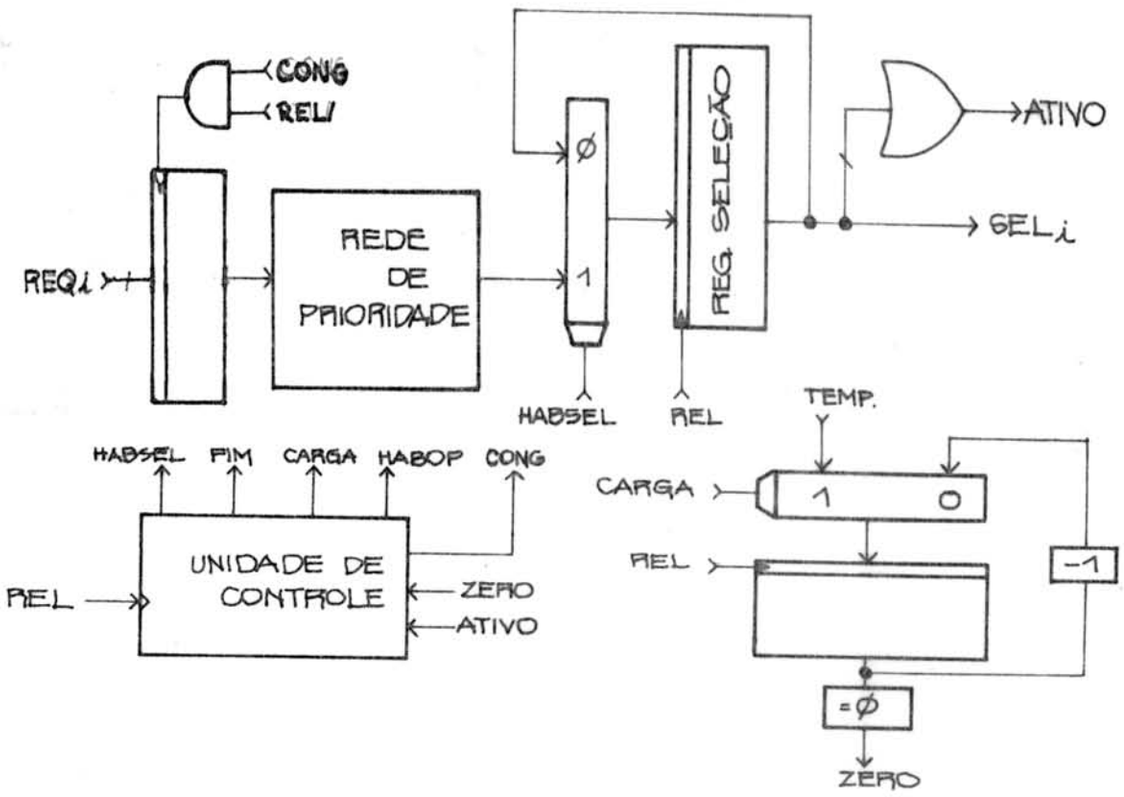


Figura 87 - Diagrama em blocos do controlador

ra atendimento. O registrador na saída da rede armazena o resultado da arbitração e mantém uma das portas ativa durante todo o ciclo de memória. Novas requisições que chegam em meio ao ciclo provenientes de outras portas participam da concorrência pelo próximo ciclo sem perturbar o atual.

Um ciclo de atendimento de uma requisição é composto de um número inteiro de períodos de relógio. A marcação do tempo é feita através de um contador descendente (figura 87) o qual é carregado com um certo valor TEMP (em função do tempo de ciclo da memória) e decrementado em cada pulso de relógio subseqüente. Ao atingir a contagem zero o controlador encerra o ciclo de memória.

A unidade de controle gerencia todo o processo. Es

tá sob sua responsabilidade a geração dos sinais de controle do árbitro e das portas, e também a monitoração do estado do contador.

6.3.3 Algoritmo de controle

As funções da unidade de controle são executadas por um algoritmo implementado sob a forma de uma máquina de estados síncrona (figura 88). Elas são definidas informalmente com o auxílio da figura 87.

Uma das funções é a carga do Registrador de Seleção a cada fim de ciclo. A habilitação da carga é feita pelo sinal HABSEL.

Simultaneamente à carga do Registrador de Seleção, o contador de tempo é carregado com um valor que depende exclusivamente do tempo de ciclo da memória (ver seção 6.4). A partir do momento em que uma porta é selecionada ela deve permanecer ativa o tempo suficiente para que uma leitura ou escrita se efetive.

Ao habilitar a carga a unidade de controle passa a monitorar a linha ATIVO que informa se alguma requisição foi registrada e selecionada para atendimento. Se não há requisições presentes a entrada do Registrador de Seleção é mantida aberta até que uma tentativa de acesso seja registrada. Assim que isto acontece o controlador gera o sinal HABOP no próximo pulso de relógio habilitando os sinais de controle do microprocessador a atuarem sobre a memória. Neste estado a unidade de controle passa a monitorar a linha ZERO, esperando pelo

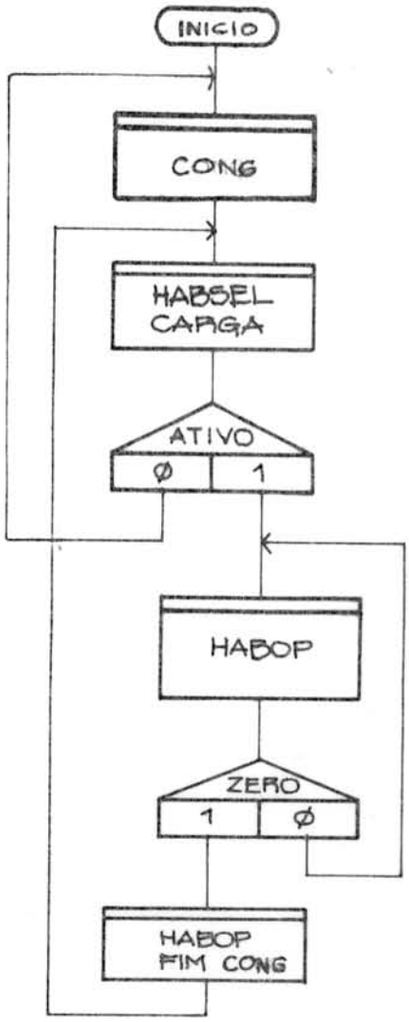


Figura 88 - Algoritmo de controle de uma memória multiporta

fim do ciclo. Ao sentir esta linha ativada é gerado o sinal FIM o qual informa às portas que o ciclo se encerrou. Este sinal, apesar de ser propagado a todas as portas só é relevante para aquela que está ativa. As demais desconsideram-no.

Após gerar o sinal FIM o controlador pode voltar ao estado inicial e tentar uma nova seleção sem se preocupar com o tempo que o microprocessador leva para encerrar o ciclo.

6.3.4 Prioridade relativa entre requisições

A rede de prioridade (figura 66) é quem determina a prioridade relativa entre as requisições conflitantes. Ela pode ser implementada sob a forma de um circuito combinacional pois não sofre restrições quanto a transições nas suas saídas. O Registrador de Requisições amostra-as num instante preciso de modo a permitir o assincronismo entre elas.

Se prioridade fixa é suficiente, então o diagrama lógico da figura 87 adapta-se ao esquema proposto. Entretanto, se um outro tipo de prioridade mais sofisticado (circular, por exemplo) é desejável, então não basta usar as linhas de requisição como entradas da rede. É preciso realimentar mais uma informação: qual a requisição atendida no momento (figura 89).

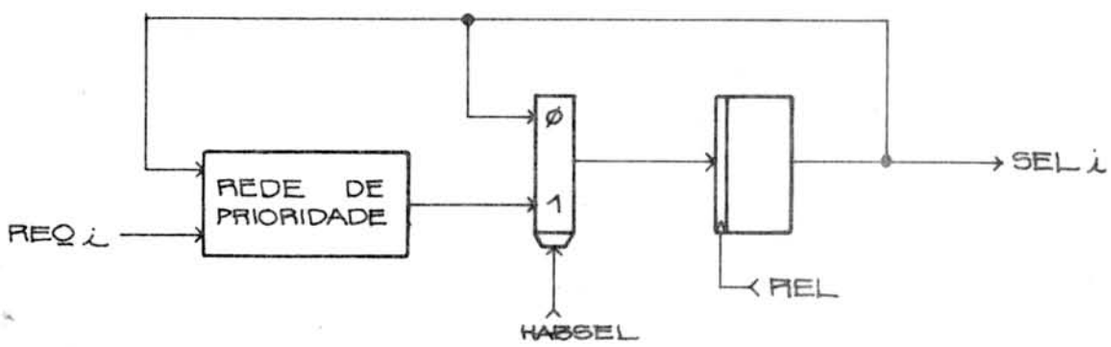


Figura 89 - Arbitração paralela utilizando prioridade cirular

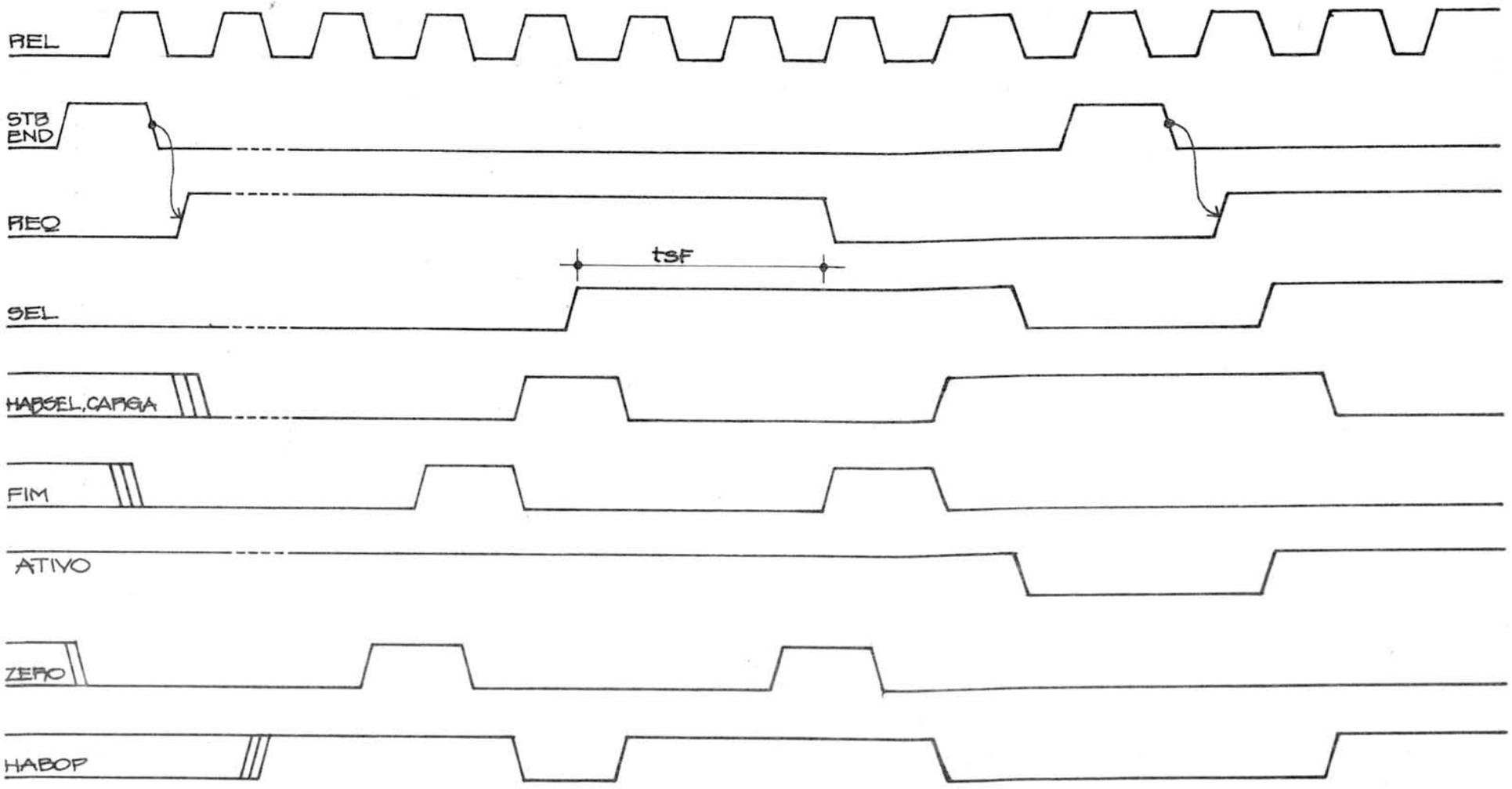


Figura 90 - Diagrama de tempos do controlador

6.4 Considerações de tempo

O diagrama de tempo do controlador (figura 90) ilustra duas seqüências diferentes de atendimento de uma requisição. No primeiro caso ela não é atendida de imediato. Isto acontece em duas situações: quando a requisição é feita em meio a um ciclo de memória ou quando ela é preterida em favor de uma outra de mais alta prioridade. O sinal REQ é mantido ativo indefinidamente até que ocorra o atendimento. No segundo caso a requisição é atendida de imediato. Isto acontece quando a requisição é iniciada com o módulo livre. As duas situações devem ser levadas em conta na determinação das relações de tempo. A diferença fundamental entre elas é que uma requisição atendida de imediato pode não ter ainda o sinal STBDADO acionado quando sobe o sinal HABOP.

A situação é mostrada na figura 91.

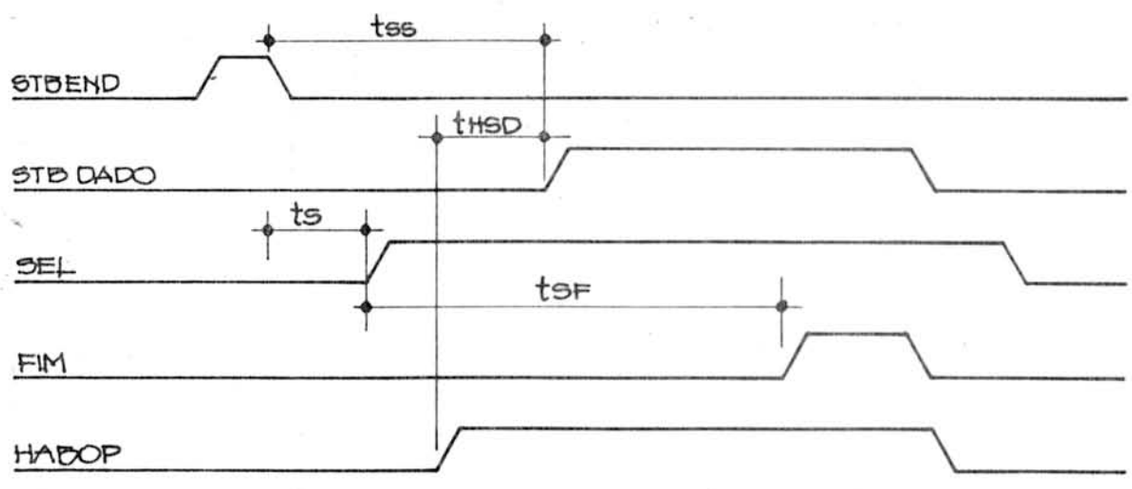


Figura 91 - Deferimento de uma requisição antes do acionamento de STBDADO

O pior atraso entre a ativação de HABOP e a chegada de STBDADO (t_{HSD}) acontece quando a requisição é aceita no primeiro pulso de relógio subseqüente à chegada de STBEND. Para que isto aconteça basta que t_S seja igual ao atraso da rede de prioridade e menor que t_{REL}. Nestas condições deve-se garantir que a duração de STBDADO até a geração do sinal FIM seja maior que o tempo de leitura ou escrita da memória. Ou seja,

$$t_{SF} \geq t_{HSD} + t_E + \frac{1}{2} t_{REL} \quad e \quad t_{SF} \geq t_{HSD} + t_L + \frac{1}{2} t_{REL}$$

ou

$$t_{SF} \geq t_{SSE} + t_E - t_S \quad e \quad t_{SF} \geq t_{SSL} + t_L - t_S$$

onde t_{SSE} e t_{SSL} representam o tempo t_{SS} numa operação de escrita e leitura respectivamente.

O tempo decorrido desde o instante da seleção até o acionamento do sinal FIM (t_{SF}) deve ser menor que o tempo de acesso (t_a) da memória. O tempo t_{SF} é determinado pelo valor TEMP carregado no contador decrescente. Portanto, pode-se calcular este valor em função de t_a e t_{REL}.

$$t_a \leq (TEMP + 1/2)\, t_{REL}$$

ou

$$TEMP \geq \frac{t_a}{t_{REL}} - 1/2$$

Com isto garante-se o acesso antes da geração do sinal FIM.

As três condições impostas ao tempo t_{SF} permitem tirar importantes conclusões a respeito dos microprocessadores e das memórias utilizadas:

1º) A velocidade da memória aumenta a velocidade do sistema até um limite. Nominalmente, uma memória com tempo de acesso menor que 3/2 t_{REL} não traz benefícios pois fica limitada pela velocidade do controlador.

2º) O tempo de ativação de uma porta está relacionado não só às especificações de tempo da memória, mas também às dos microprocessadores. Se o atraso entre os sinais STBEND e STBDADO (t_{SS}) não satisfaz as condições impostas, então o microprocessador não pode ser utilizado. Visto de um outro ângulo pode-se afirmar que um microprocessador lento pode ser utilizado se o projeto inicial assim o prevê. Porém, ele irá nivelar por baixo a performance dos micros mais rápidos ao acessar a memória.

6.5 Memória privativa

A interconexão de processadores e memória através de múltiplos barramentos dedicados facilita a implementação de áreas de memória privativas. Todo processador pode ter sua memória privativa ligada diretamente ao seu barramento de memória global. Conseqüentemente, a interface de processador não precisa decodificar endereços de uma ou outra e direcionar a requisição. Qualquer acesso a memória é propagado ao barramento sem distinguir se ele pertence à memória global ou privativa. Se ele pertence a esta última então é aceito de imediato pois não existe possibilidade de interferência. A estratégia descrita é possível graças à dedicação do barramento. Os acessos à memória local não causam nenhuma contenção.

O fato de um módulo ser ou não ser global é apenas uma questão de conectar suas portas aos barramentos. Assim, um módulo de memória privativa pode ser obtido de um global simplesmente deixando de realizar a conexão física de suas portas, exceto uma, aos barramentos do sistema.

Porém, com apenas uma porta em atividade, a interface de memória local pode ser simplificada (figura 92).

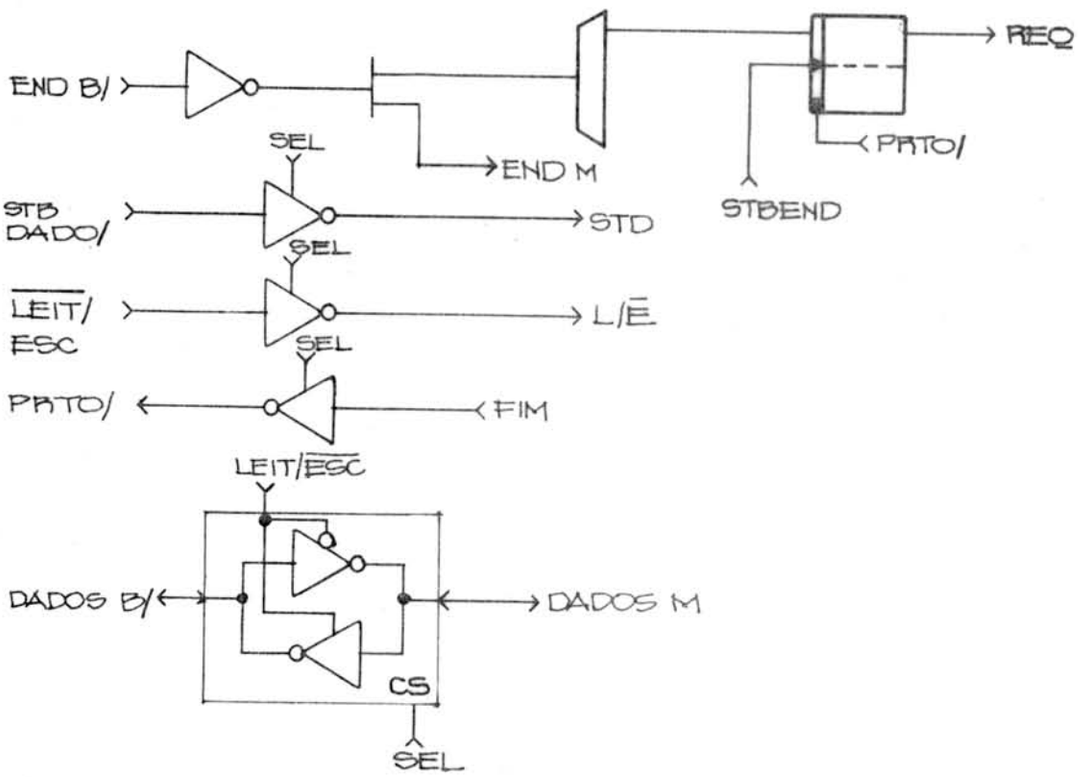


Figura 92 - Diagrama lógico da porta de memória local

O endereço de barramento chega até a memória livremente. Apenas os sinais de controle LEIT/$\overline{\text{ESC}}$, STBDADO e PRTO são bloqueados pela ausência do sinal SEL até que o controlador sincronize a requisição.

Por outro lado, o protocolo de barramento deve ser observado também pela memória privativa. O sinal PRTO deve ser gerado para permitir que o micro prossiga. Por esta razão, e para que o tempo de acesso da memória continue independente da duração dos sinais dos microprocessadores, o controlador é mantido com algumas modificações (figura 93) descritas a seguir.

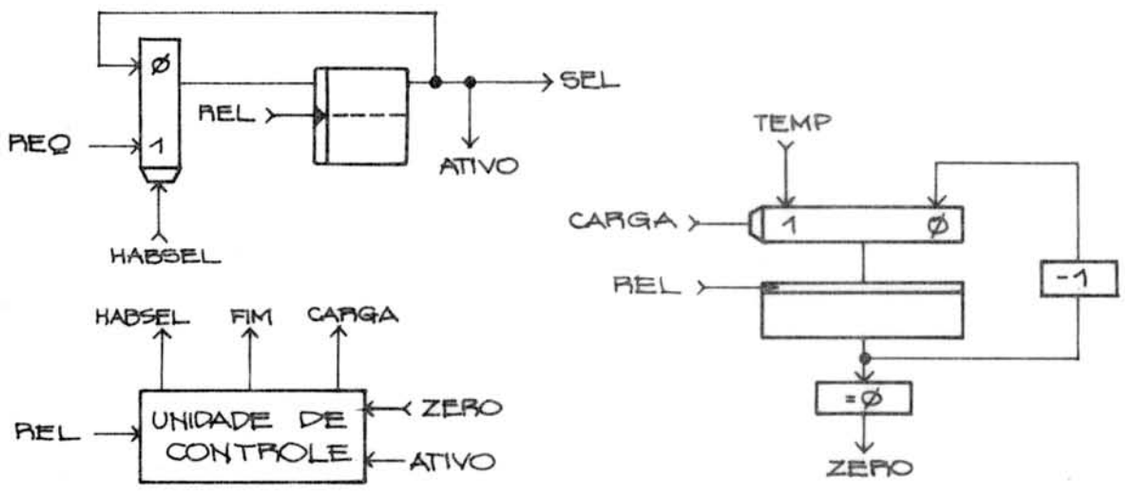


Figura 93 - Diagrama lógico do controlador de memória local

A rede de prioridade é eliminada por não ser mais necessária. O registrador de seleção é substituído por um único flip-flop. Este, além de habilitar a passagem dos sinais de controle e dados, também fornece uma indicação direta de início de ciclo (ATIVO) para a unidade de controle.

O algoritmo de controle (figura 94) é simplificado pela inexistência de pedidos concorrentes.

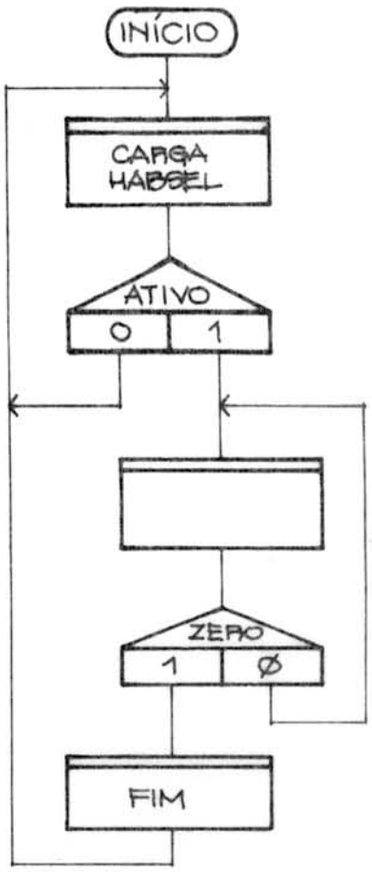


Figura 94 - Algoritmo de controle da memória local

6.6 Mecanismo para acessos indivisíveis

A implementação de um mecanismo de acessos indivisíveis através de um módulo especial de memória é descrita a seguir.

A parte operacional do módulo em questão (o conjunto de portas) deve sofrer algumas modificações em relação a um módulo comum (figura 95). As alterações refletem a lógica necessária para criar um ciclo de escrita artificial sem a intervenção do processador.

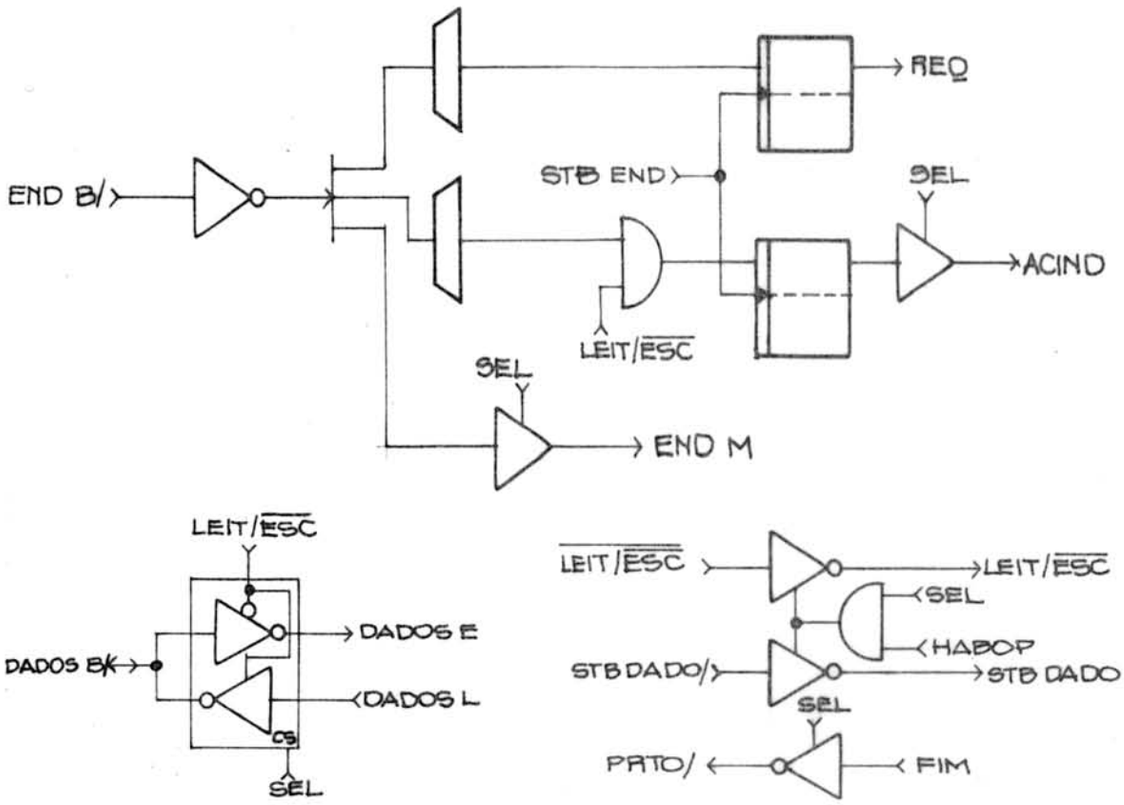


Figura 95 - Diagrama lógico da porta do módulo de variáveis semáforo

Na hipótese de um módulo conter um grande número de posições de memória, e não ser necessário um grande número de variáveis semáforo pode-se designar apenas uma faixa de endereços para esta função. Conseqüentemente, a interface deve ser capaz de suportar operações de leitura e escrita em posições comuns de memória além de operações indivisíveis sobre as variáveis semáforo. A distinção entre as duas é feita, da mesma forma como é reconhecido um acesso ao módulo, através de um decodificador e um flip-flop.

Toda operação de leitura de uma variável semáforo provoca uma escrita automática de 1's em todos os bits. A escrita não deve causar o mesmo efeito para que o processador

possa "resetar" o semáforo. Por isto o sinal ACIND somente é acionado em operações de leitura.

Em suma, um acesso ao módulo é constituído por um ou dois ciclos de memória na seguinte ordem:

1º) um ciclo normal de leitura ou escrita, especificado pelo processador;

2º) um ciclo de escrita se o ciclo anterior foi uma leitura de uma variável semáforo.

O controlador de memória é responsável pela execução do segundo, uma vez que o microprocessador não toma parte nele. Os ciclos de leitura de uma posição comum de memória e de uma variável semáforo diferem apenas quanto a geração do sinal FIM. Neste último ele não é gerado ao fim da leitura para impedir que o processador prossiga (se isto acontecesse o barramento de endereço alterar-se-ia ainda durante a escrita indivisível). O sinal FIM que completa a leitura só é gerado após a conclusão do ciclo de escrita. O resultado prático desta imposição é visto nas figuras 96 e 97. O caminho bidirecional de dados é bifurcado em dois caminhos unidirecionais. O circuito de escrita forçada (figura 96) é interposto entre as portas e a memória.

O registrador de dados armazena o conteúdo lido de um semáforo, para posterior envio ao processador.

A proveniência dos dados de escrita é escolhida pelo seletor: do barramento em qualquer operação de escrita proveniente do microprocessador ou um valor constante (no caso 1's) ao ciclo de escrita de uma operação indivisível.

As duas portas lógicas (E e OU) forçam os sinais $L/\overline{E}$ e STD a uma escrita sob o comando da unidade de controle.

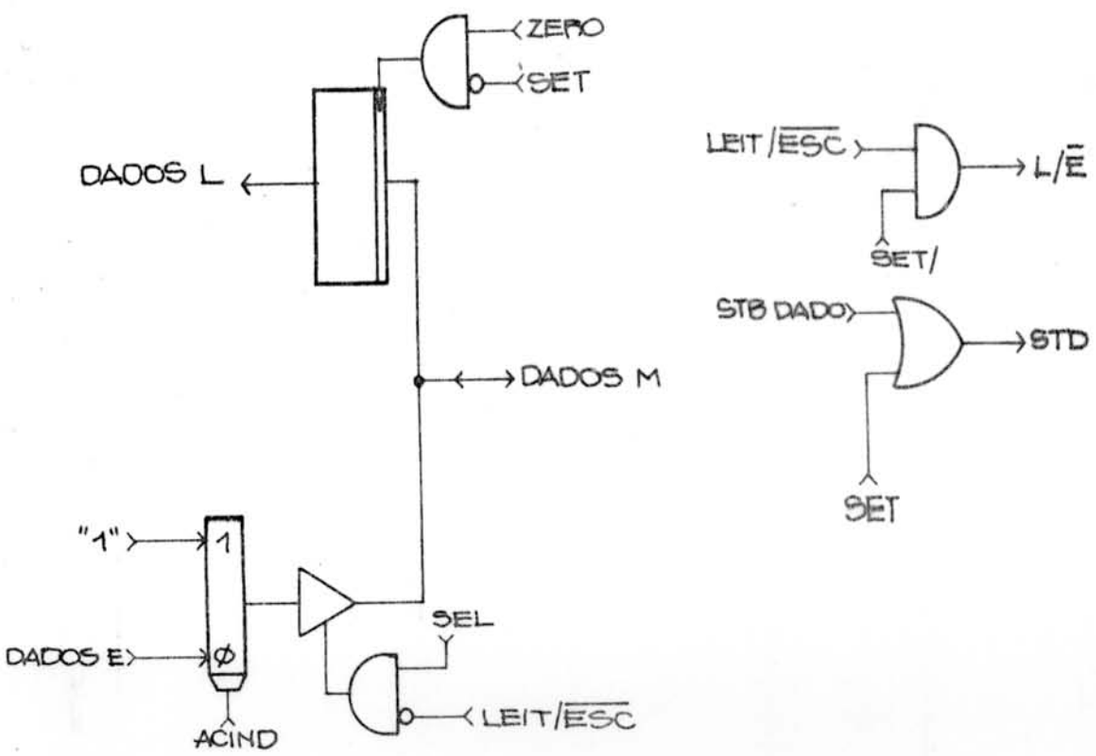


Figura 96 - Circuito de escrita forçada

A figura 97 mostra o algoritmo de controle.

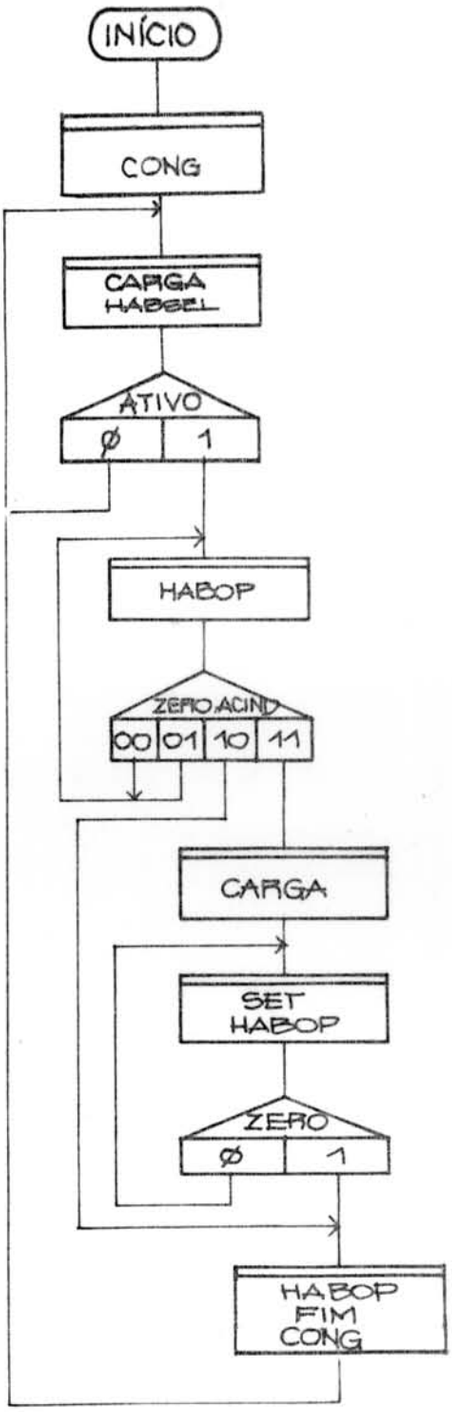


Figura 97 - Algoritmo de controle do módulo de semáforos

7 MATRIZ DE BARRAMENTOS CRUZADOS

7 MATRIZ DE BARRAMENTOS CRUZADOS

7.1 Introdução

A terceira alternativa apresentada para a estrutura de interconexão da figura 51 é uma matriz de barramentos cruzados. A organização do sistema é apresentada na figura 98.

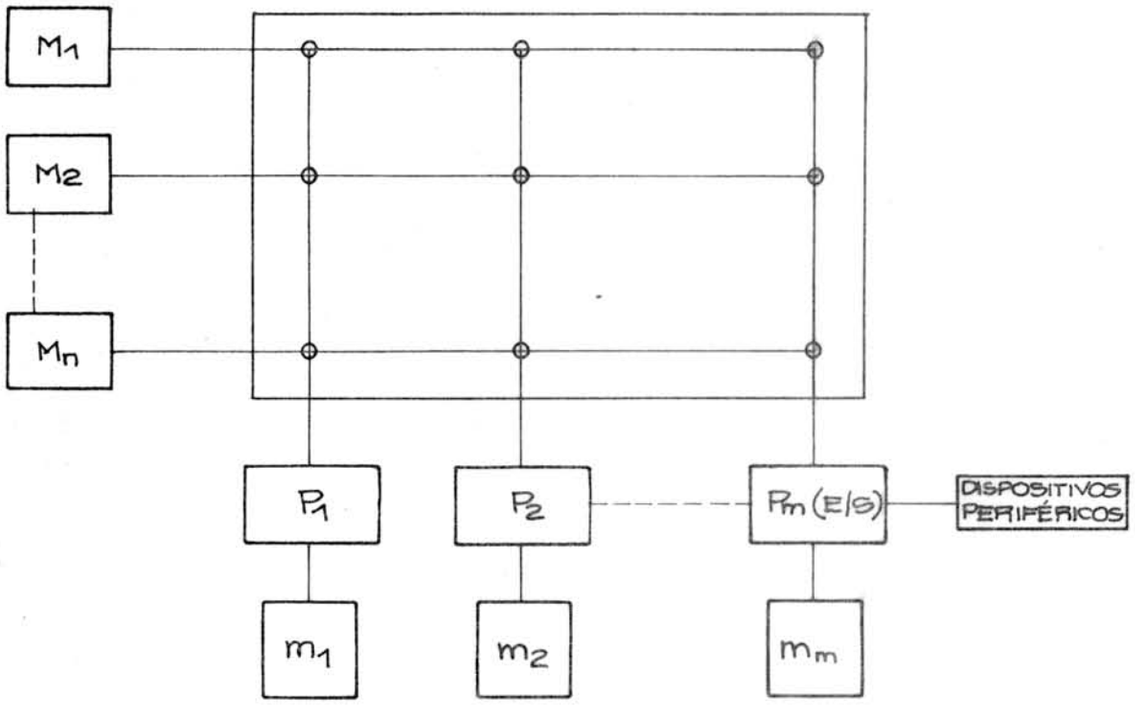


Figura 98 - Organização do multimicroprocessador com uma matriz de barramentos cruzados

A matriz permite que todos os processadores tenham acesso a todos os módulos de memória numa relação 1 para 1, ou seja, não mais de um processador pode ser ligado a um módulo de memória em cada instante. Igualmente, um processador pode referenciar somente um módulo de cada vez.

Um segundo atributo da matriz é a característica não-bloqueante[18,60]. Uma estrutura de interconexão é bloqueante se um determinado par de elementos livres não pode ser conectado por não existir um caminho disponível. Uma estrutura pode ser classificada como não-bloqueante em dois sentidos: (1) se o rearranjo das ligações correntes provê um caminho à requisição bloqueada, ou (2) se a estrutura não possui estados bloqueantes. A matriz de barramento cruzados se enquadra no segundo caso pois uma requisição de acesso à memória sempre encontra um caminho disponível. A disponibilidade é uma decorrência da dedicação dos barramentos. Os barramentos verticais são privativos dos processadores aos quais eles estão ligados. O mesmo ocorre com os barramentos horizontais em relação aos módulos de memória.

A matriz possibilita a transferência de uma palavra entre um processador e um módulo global pela conexão de seus respectivos barramentos nos pontos de cruzamento (vide figura 98).

A interconexão de processadores e memória através da matriz de barramentos cruzados pode ser vista como uma extensão do caso de memórias multiporta onde o controle migrou dos módulos para a matriz. Funcionalmente elas são semelhantes. Podem acontecer até max(m,n) acessos simultâneos desde que eles se refiram a módulos distintos. A diferença entre elas reside na centralização do controle e das chaves. A centralização pode trazer algum benefício à modularidade do sistema, dependendo da forma como é implementada[18,20].

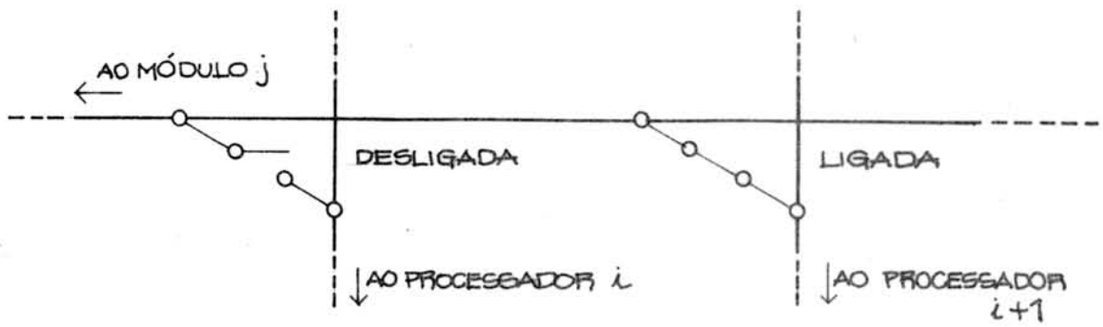
A matriz oferece a vantagem de necessitar menos cabos e conectores, componentes de alto custo e suscetíveis

a problemas mecânicos e elétricos. A organização através da matriz de barramentos cruzados requer m+n cabos enquanto a organização com múltiplos barramentos/memórias multiporta requer mxn.

7.2 A matriz de chaveamento

7.2.1 Funcionamento

A conexão de um processador a um módulo de memória é efetuada no ponto de cruzamento de seus barramentos. Em cada ponto existe uma chave que assume um de dois estados: ligada ou desligada. Funcionalmente, ela pode ser vista como uma chave de um pólo e uma posição (figura 99).



OBS.: OS ÍNDICES i E j USADOS NESTE CAPÍTULO SE REFEREM SEMPRE A UM PROCESSADOR E A UM MÓDULO DE MEMÓRIA GENÉRICOS, RESPECTIVAMENTE.

Figura 99 - Representação funcional de um ponto de cruzamento

Na figura acima são mostradas as duas chaves que interligam o módulo j aos processadores i e i+1. O processador i+1 está ligado ao módulo j. Enquanto isso, todas as demais chaves na linha do módulo j devem estar desligadas para que os outros processadores não interfiram na transferência. Da mesma forma, todas as chaves da coluna i+1 devem estar desligadas para garantir que apenas um módulo participe na operação. Em suma, a matriz permite que apenas um módulo seja conectado a um processador num dado instante e vice-versa.

Os barramentos que se cruzam na chave compreendem dados, endereço e controle de leitura e escrita. O barramento de endereço contém apenas os bits que identificam a palavra dentro do módulo. Os demais bits de endereço (que identificam o módulo requisitado) são dirigidos ao árbitro do módulo.

7.2.2 Tipo de chave

Foram consideradas três hipóteses na implementação da matriz de chaveamento no que diz respeito ao componente básico a ser utilizado: arranjos de chaves integradas[48], transceptores "tri-state"[31], ou seletores.

O arranjo de chaves consiste de dezesseis portas de transmissão bidirecionais (ver figura 100) as quais podem ser selecionadas individualmente para realizar a conexão de um grupo de quatro linhas (X) sobre um segundo grupo também de 4 linhas (Y).

Qualquer uma das portas é selecionada aplicando-se um endereço apropriado nas linhas ADDRESS. A porta seleciona-

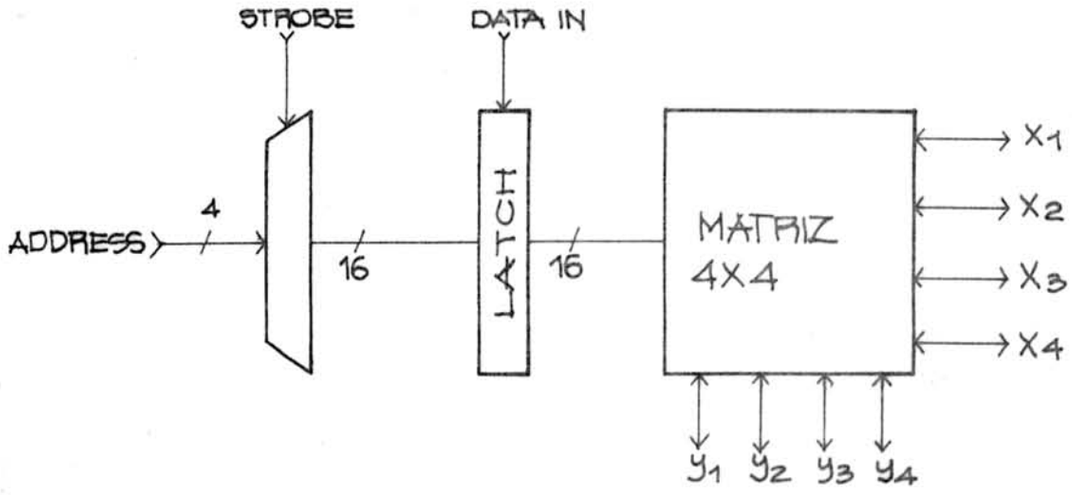


Figura 100 - Diagrama funcional do arranjo de chaves integradas

da é ligada ou desligada, aplicando-se 1 ou 0 lógico à entrada DATA IN e ativando-se a linha STROBE. Qualquer número de portas pode estar ativa a cada instante. Isto significa que as conexões não se restringem a pares de dispositivos.

O uso deste componente favorece a construção da matriz no que tange à redução do número de pastilhas. Este é um aspecto importante pois a matriz de barramentos cruzados apresenta um padrão de crescimento não linear. Entretanto, o arranjo possui o inconveniente de não permitir o controle simultâneo de todas as chaves. Apenas uma delas pode ser ligada a cada instante. A conseqüência imediata é o atraso no chaveamento de requisições que acontecem assincronamente e em paralelo. Este tipo de circuito se presta a interconexões que se mantêm por um tempo bastante mais longo que o processo de conexão e desconexão. Exemplificando, se a ligação fosse feita com base nos processos e não no ciclo de memória, este componente cer

tamente seria mais indicado.

O estudo comparativo das duas opções restantes é feito tomando-se como parâmetros o número de chaves e sinais de controle necessários a cada uma delas. A tabela 5 resume os dois parâmetros para a interconexão de um bit (bidirecional).

	NÚMERO DE ELEMENTOS	SINAIS DE CONTROLE
TRANSCEPTORES TRI-STATE	m x n	m x n
SELETORES	m + n	$m\log_2 n + n\log_2 m$

Tabela 5 - Parâmetros comparativos entre portas tri-state e seletores

A primeira os seletores parecem ser a melhor opção. Entretanto, algumas considerações de ordem tecnológica devem ser levadas em conta. O número de transceptores "tri-state" integrados em uma pastilha é maior que o de seletores. Isto tende a equilibrar o número de pastilhas nos dois casos. Examinando-se várias configurações possíveis verifica-se que os transceptores exigem menos pastilhas, principalmente se a configuração é de pequeno porte. Como o problema econômico e de espaço é equacionado pelo número de integrados[13], a opção por transceptores "tri-state" é mais indicada.

Um outro problema relacionado aos seletores é o fan-out dos mesmos. A figura 101 ilustra o tipo de ligação a ser feita entre os seletores para interconectar um bit. A saída de um seletor de módulo é ligada a n seletores, além do processador. A saída de um seletor de processador é ligada a m outros seletores além do módulo. A menos que seja colocado

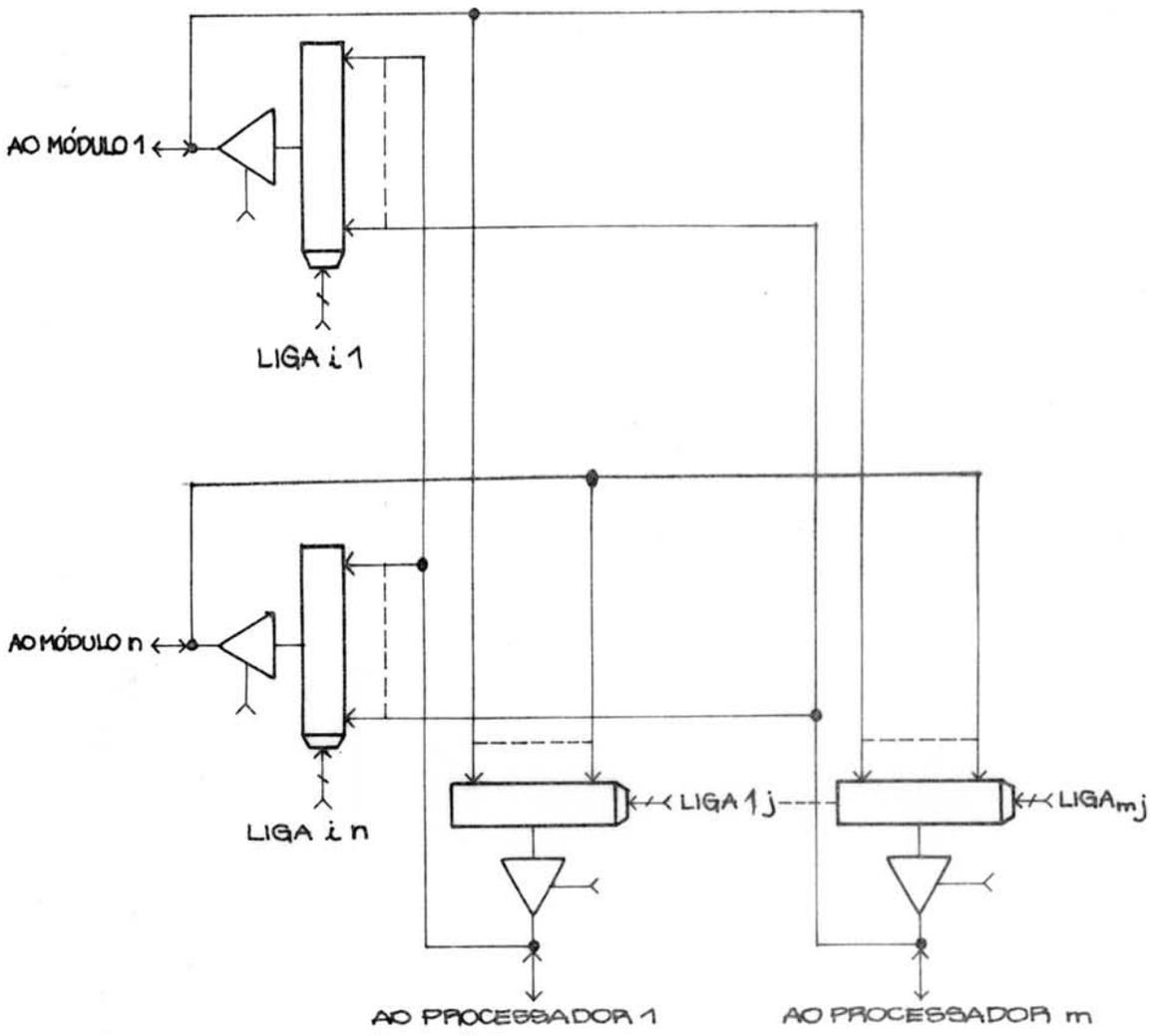


Figura 101 - Matriz de chaveamento implementada com seletores

um amplificador em cada saída de seletor o "fan-out" é logo excedido.

Uma última consideração, esta de ordem tática, favorece os transceptores. Com seletores a configuração máxima é definida no projeto inicial pelo número de entradas dos mesmos. Isto implica um alto custo inicial se as expansões futuras devem permitir configurações com muitos processadores e/ou módulos de memória. Usando-se transceptores o sistema pode crescer de forma incremental desde que a unidade de controle o per

mita.

7.2.3 Chaveamento de linhas unidirecionais

O barramento que une um processador à memória compreende linhas unidirecionais (endereço e controle) e linhas bidirecionais (dados). A parte unidirecional da matriz de chaveamento é implementada (figura 102) com transmissores "tri-state". Em cada ponto de cruzamento existe um transmissor por bit de informação transferida.

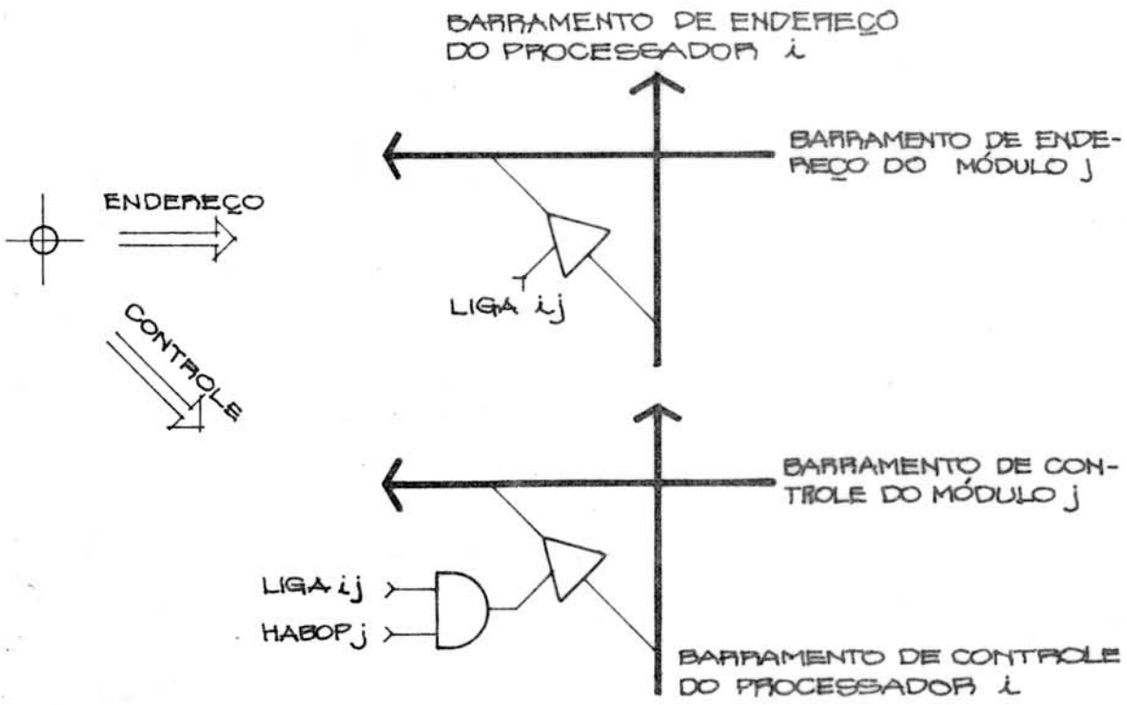


Figura 102 - Chaveamento de linhas unidirecionais

A informação flui do barramento do processador i

para o barramento do módulo j sob o comando do sinal LIGAij (e do sinal HABOPj no barramento de controle) enviado pela unidade de controle do módulo. Enquanto este sinal não está ativado o transmissor se desliga isolando eletricamente os barramentos. Conseqüentemente os sinais do processador i não interferem sobre o módulo j, de modo a permitir que ambos executem duas transferências disjuntas simultaneamente.

O sinal HABOPj é necessário pelo mesmo motivo que foi usado na memória multiporta: impedir que o sinal de escrita esteja ativo durante as transições das linhas de endereço.

Os índices i e j indicam que em cada ponto de cruzamento há um sinal LIGAij específico para controlar aquele ponto. São necessários, portanto, m x n sinais de controle.

7.2.4 Chaveamento de linhas bidirecionais

A construção da porta bidirecional da matriz de chaveamento é feita de uma forma idêntica à porta unidirecional, exceto pelo controle de sentido do fluxo de dados (figura 103).

Em cada ponto de cruzamento existe um transceptor por bit de informação bidirecional. O sentido de fluxo da informação é dado pelo sinal LEIT/$\overline{\text{ESC}}$i sob o comando do processador i. Portanto cada coluna de chaves possui um sinal LEIT/$\overline{\text{ESC}}$ privativo. São necessários m destes sinais.

O sinal LEIT/$\overline{\text{ESC}}$ isolado não é suficiente para fechar a conexão entre dois barramentos. Enquanto o sinal LIGAij não é acionado nenhuma informação flui pela chave. O sinal

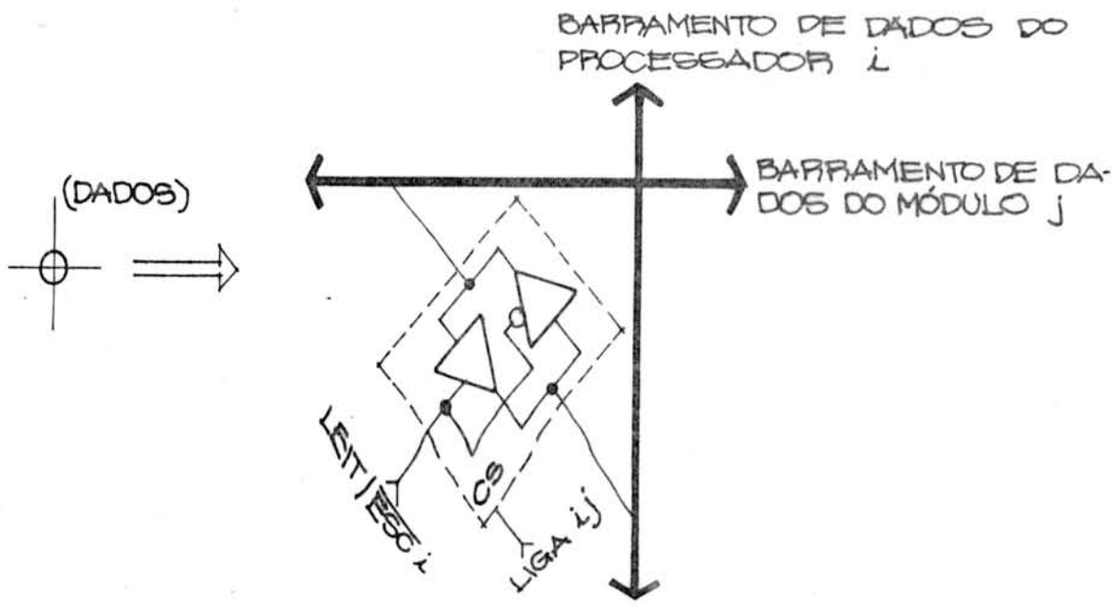


Figura 103 - Chaveamento de linhas bidirecionais

LIGAij é o mesmo usado no chaveamento das linhas unidirecionais.

7.3 Controle da matriz

7.3.1 Organização interna

A matriz de barramentos cruzados (figura 104) é formada por uma unidade operacional (a matriz de chaveamento) e uma unidade de controle. A unidade operacional contém o arranjo de chaves que, funcionalmente, direciona as requisições de acesso à memória ao módulo correto. A unidade de controle executa o chaveamento baseado na informação fornecida pelos bits de endereço que identificam o módulo de memória. Esta unidade é subdividida em três subunidades: a interface de proces-

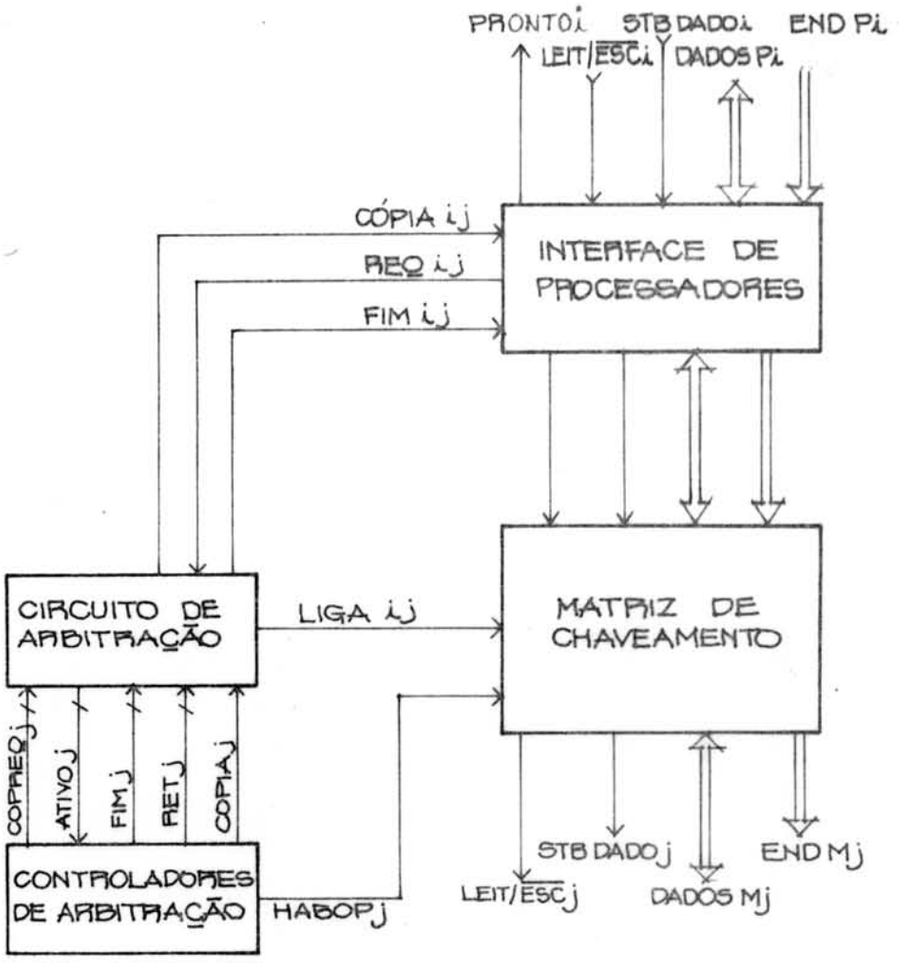


Figura 104 - Blocodiagrama da matriz de barramentos cruzados

sadores, o circuito de arbitração e os controladores de arbitração.

O problema de se estabelecer uma conexão entre um processador e um módulo de memória pode ser logicamente dividido em quatro etapas: (1) recebimento e registro dos pedidos de acesso; (2) seleção das requisições atendidas e estabelecimento da conexão; (3) informação de fim de ciclo aos processadores atendidos; (4) desconexão. A primeira etapa é executada na interface de processadores. A segunda nos circuitos de arbitração comandados pelos controladores. A terceira e quar-

ta etapas são executadas nos controladores de arbitração e nos circuitos de arbitração que enviam os sinais de fim de ciclo ao processador correto. A seguir, é descrito o funcionamento de cada uma destas partes que compõem a unidade de controle.

7.3.2 Interface de processadores

A interface de processadores é responsável pelo recebimento dos sinais provenientes de todos os processadores do sistema, pelo registro das requisições de acesso à memória, pelo armazenamento temporário dos dados lidos e pela manutenção do microprocessador em estado de espera enquanto o pedido não é deferido.

A interface é composta de m portas independentes, cada uma delas responsável pela comunicação com um dos processadores. O diagrama lógico de uma porta é visto na figura 105.

Um endereço da memória global é decodificado e registrado como requisição de acesso na descida do sinal STBEND. Os sinais REQil, ... REQin especificam o módulo sendo requisitado pelo processador i. Estes sinais são dirigidos aos circuitos de arbitração onde são confrontados com as demais requisições de outras portas. Enquanto um dos sinais de requisição (mutuamente exclusivos) está ativo, o sinal PRONTO permanece desativado. Ele somente é ativado quando da chegada do sinal FIMij, como resultado do apagamento do registro da requisição recém atendido. O apagamento é feito assincronamente para que no próximo pulso de relógio uma nova requisição ao módulo possa ser selecionada para atendimento.

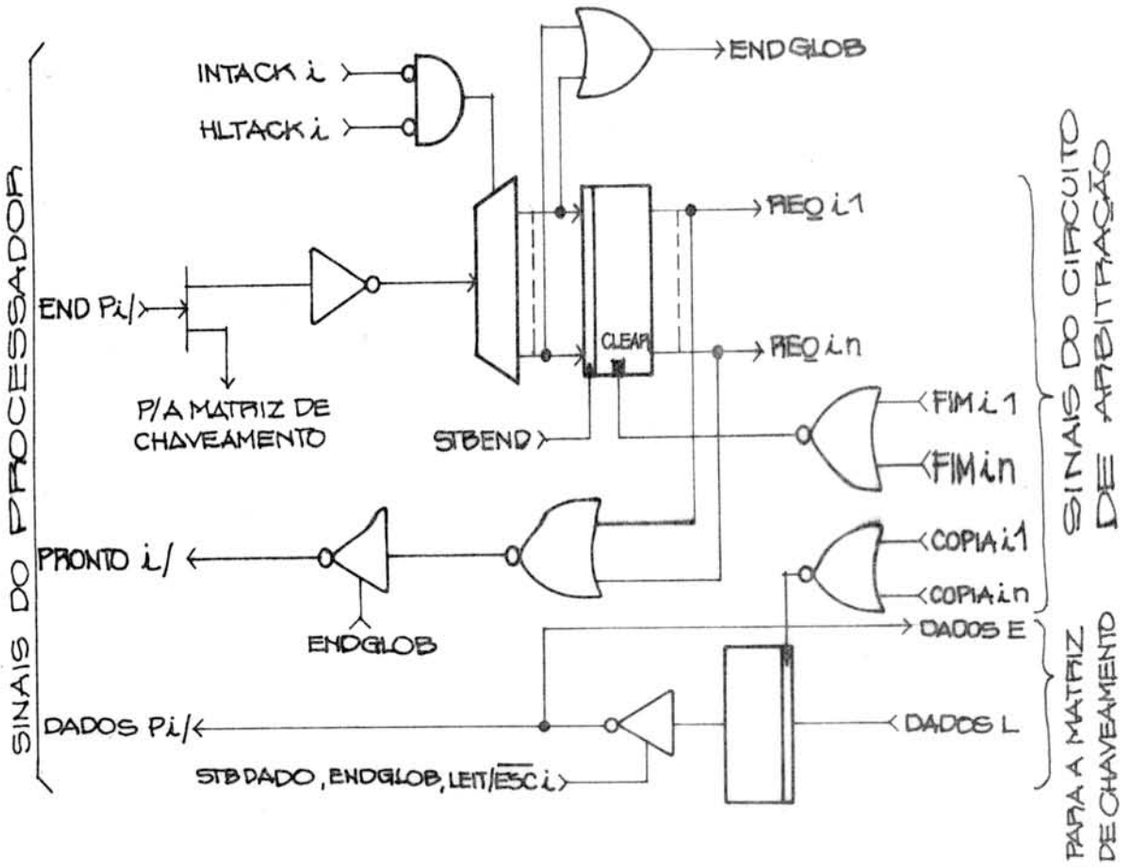


Figura 105 - Diagrama lógico de uma porta da interface de processadores

Durante um ciclo de interrupção ou parada as transições do sinal STBEND não são registradas como requisições. Além disso, o sinal INTACK coloca as saídas da interface para o processador no terceiro estado de modo a não interferirem no recolhimento do vetor de interrupção.

7.3.3 Circuitos de arbitração

O problema de resolução de conflitos num sistema centralizado é subdividido numa série de árbitros que operam em paralelo, porém assincronamente um em relação aos outros.

As requisições são agrupadas por módulo e a arbitração feita dentro de cada subgrupo. O diagrama lógico do árbitro j é mostrado na figura 106.

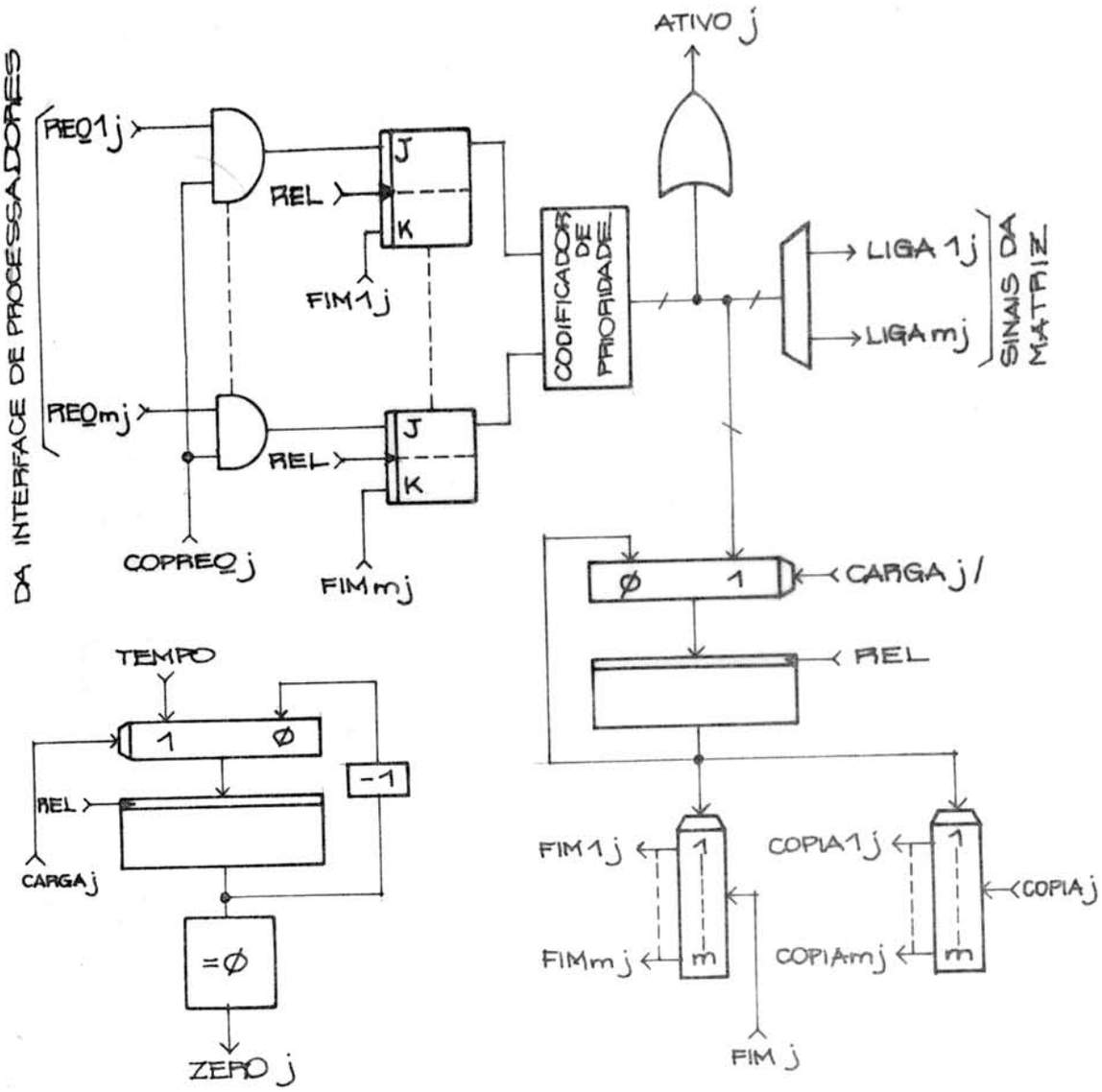


Figura 106 - Diagrama lógico do árbitro

Cada módulo de memória é associado a um circuito de arbitração independente dos demais. As requisições provenientes dos vários processadores e endereçadas a um mesmo módu-

lo são dirigidas ao seu árbitro. Os índices 1 a m nos sinais REQ1j, ..., REQmj indicam a fonte da requisição.

Um árbitro de requisições independentes, resolve os acessos conflitantes que chegam a qualquer tempo. Este tipo de árbitro pode ser implementado através de uma máquina seqüencial síncrona[16,28]. Esta é a técnica adotada para todos os árbitros de módulo. A sincronização torna-o um pouco mais lento, porém facilita a contagem do tempo de manutenção da ligação.

O árbitro apresentado difere daquele usado para as memórias multiporta. Naquele, todas as requisições presentes participavam do processo de seleção a cada fim de ciclo. Neste, é interposto um "buffer" entre as linhas de requisição e o codificador de prioridade (ver figura 106).

O circuito funciona da seguinte maneira: quando o processador i requisita um acesso ao módulo j é acionada a linha REQij. Este sinal tenta ligar um dos flip-flops do"buffer", entretanto ele é bloqueado pela porta lógica E na ausência do sinal COPREQj, o qual é gerado pela unidade de controle somente quando não há nenhuma requisição a ser atendida. Quando isto acontece, todos os flip-flops são habilitados a copiar as requisições presentes no próximo pulso de relógio. Basta que exista uma requisição para desabilitar a entrada do buffer até o fim do ciclo de memória. As saídas dos flip-flops alimentam um codificador de prioridade que escolhe uma das requisições para atendimento. Esta é decodificada gerando os sinais LIGAij. Da saída do decodificador partem as linhas LIGA1j,..., LIGAmj que irão atuar diretamente sobre as chaves associadas ao módulo j. Portanto, é o codificador quem, em última análi-

se, decide qual das requisições é atendida.

A conexão se mantém por um número inteiro de períodos de relógio até que o contador decrescente chega a zero.Neste instante é gerado o sinal FIMj que desfaz a conexão atual desligando o flip-flop correspondente no "buffer" de entrada,e informando a interface de processadores para gerar o PRONTO. O sinal FIMj é dirigido, por meio de um demultiplexador, unicamente ao flip-flop que armazena a requisição atendida. As linhas de controle do demultiplexador são armazenadas em um registrador para que elas se mantenham estáveis durante toda a duração do sinal FIMj. Sem isso, o sinal FIMj ativaria o PRONTO na interface do processador recém selecionado para atendimento.

É importante salientar que a implementação particular escolhida para o árbitro elimina o problema de um processador de baixa prioridade sofrer longos tempos de espera, coisa comum em árbitros com esquema de prioridade fixa. Apesar do fato de sua prioridade ser fixa (determinada pelo codificador de prioridade), o "buffer" faz com que ele se comporte como uma fila a qual é preenchida de tempos em tempos (quando está vazia). Uma vez preenchida, total ou parcialmente as requisições são atendidas uma a uma segundo uma prioridade fixa de 1 a m. Desta forma garante-se que nenhum processador irá esperar mais que m ciclos de memória para ser atendido. O esquema de prioridade implementado não é fixo, e também não é circular.Ele pode ser considerado como "quase-circular"[20].

7.3.4 Algoritmo de controle

A existência de um árbitro para cada módulo de memória não significa que seja necessário um controlador para cada circuito de arbitração. O controle pode ser exercido de forma global por uma única unidade que geraria os sinais necessários a todos os árbitros e interfaces. Neste caso todos os acessos não conflitantes são apresentados simultaneamente à memória, ou seja: o chaveamento da matriz é feito a intervalos regulares pois a operação é totalmente síncrona. A vantagem imediata é a redução do número de componentes, porém a sincronização dos módulos traz consigo um inconveniente grave: uma requisição que chega em meio a um ciclo de memória não pode ser deferida imediatamente mesmo que ela se refira a um módulo ocioso. Sempre é preciso esperar o fim do ciclo atual. Conseqüentemente o processador sofrerá um atraso desnecessário. Por esta razão é preferível fazer com que os módulos operem assincronamente um em relação aos outros. As requisições são atendidas de imediato quando se referem a módulos ociosos, e o sistema só é atrasado quando houver conflito.

Portanto, a solução adotada é um controlador para cada árbitro, sendo o seu algoritmo de controle mostrado na figura 107.

Ao fim de um ciclo de memória o controlador verifica se existe alguma requisição pendente na fila. Em caso afirmativo a ligação é estabelecida automaticamente pelo codificador e a unidade de controle espera apenas o final deste ciclo indicado pelo sinal ZERO. Se a fila está vazia, o controlador habilita o seu preenchimento com as requisições armazenadas

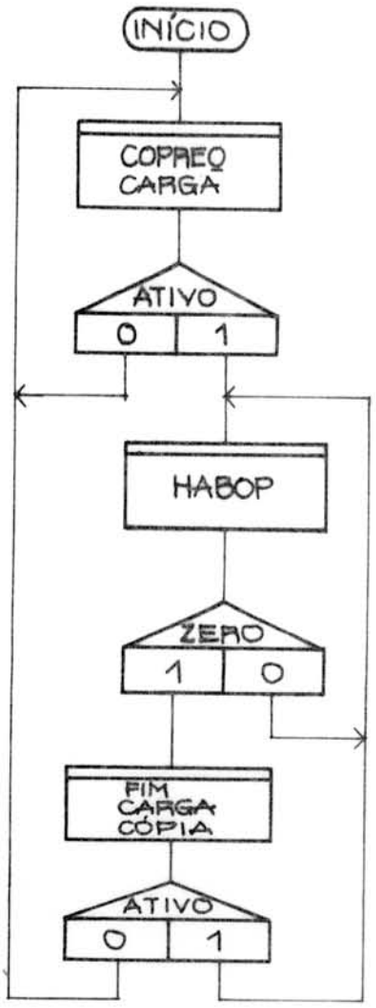


Figura 107 - Algoritmo de controle do árbitro

na interface de processadores e seleciona imediatamente a de mais alta prioridade para atendimento.

A figura 108 ilustra o atendimento de duas requisições (REQ1 e REQ2) a um mesmo módulo de memória. REQ1 tem prioridade sobre REQ2, por isto é atendida em primeiro lugar, apesar de ter chegado depois.

Quando é atendida a última requisição da fila o árbitro leva 3/2 t_{REL} para habilitar a requisição 1 (medidos a partir do instante em que é gerado FIM). Já o tempo de habilitação da requisição 2 é 1/2 t_{REL}. Um período de relógio é usado para preencher a fila.

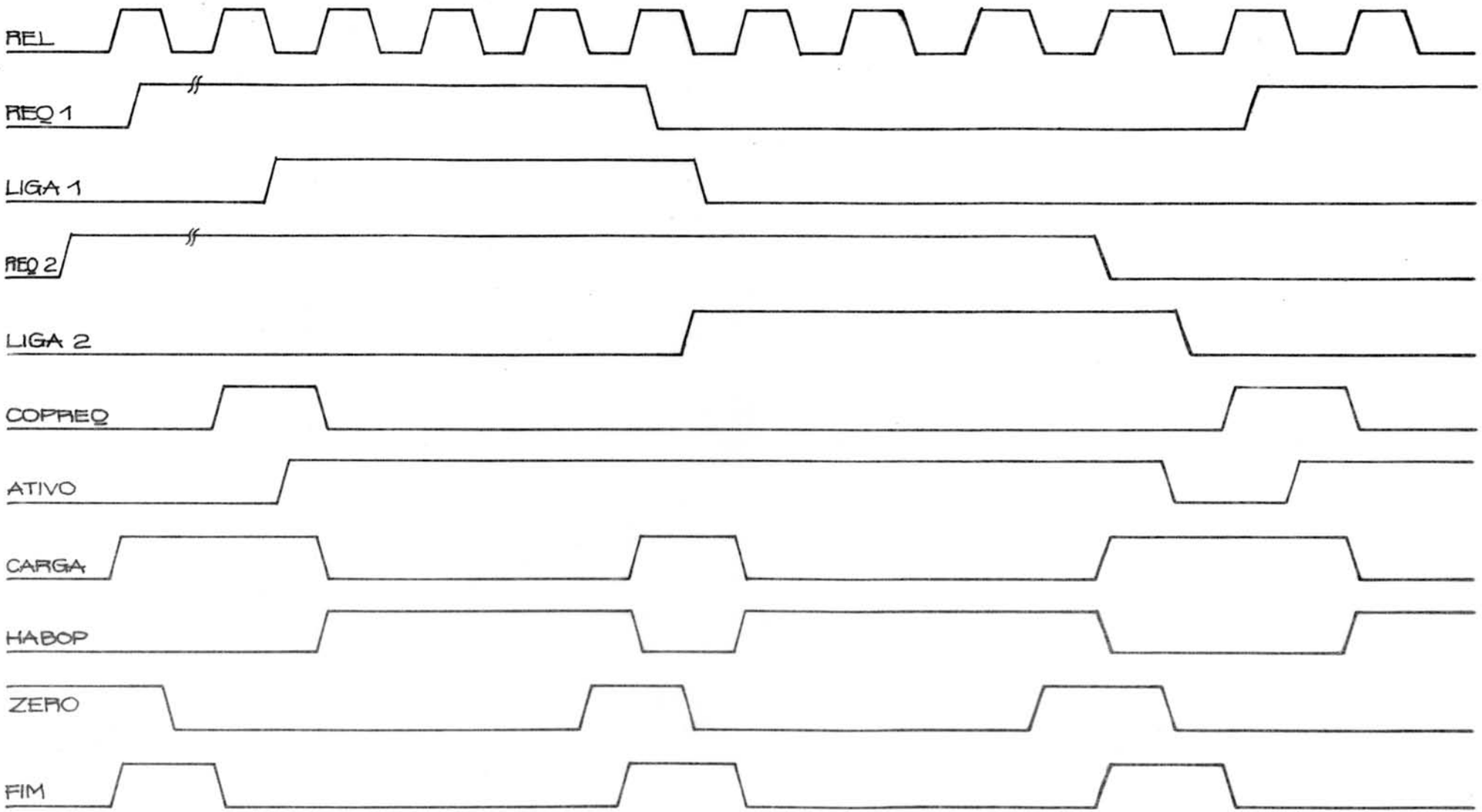


Figura 108 - Diagrama de tempos no atendimento de duas requisições

7.4 Mecanismo para acessos indivisíveis

O mecanismo de sincronização ao nível do hardware é implementado através de um módulo especial de memória. Nele, um determinado número de endereços é designado para esta função e não podem ser usados como posições comuns de memória. Funcionalmente o módulo se comporta de maneira idêntica àquele usado com múltiplos barramentos/memórias multiporta. Uma operação de leitura sobre uma variável semáforo causa a escrita automática de 1's sem a intervenção do processador.

A implementação do mecanismo requer algumas modificações no algoritmo de controle e no módulo em questão. Em primeiro lugar, é preciso identificar o endereço de uma variável semáforo. Isto pode ser feito na interface de processadores ou no módulo. A identificação no módulo é a mais indicada por exigir apenas um decodificador. Em compensação, duas novas linhas são adicionadas entre a chave e o módulo (figura 109). A

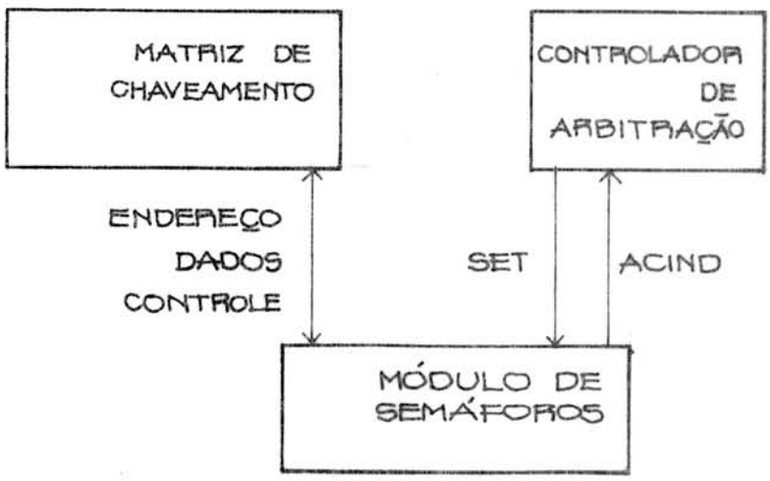


Figura 109 - Linhas adicionais para acessos indivisíveis

linha ACIND pede ao controlador um ciclo de escrita forçada, e a linha SET comanda a escrita. As modificações no módulo estão mostradas na figura 110.

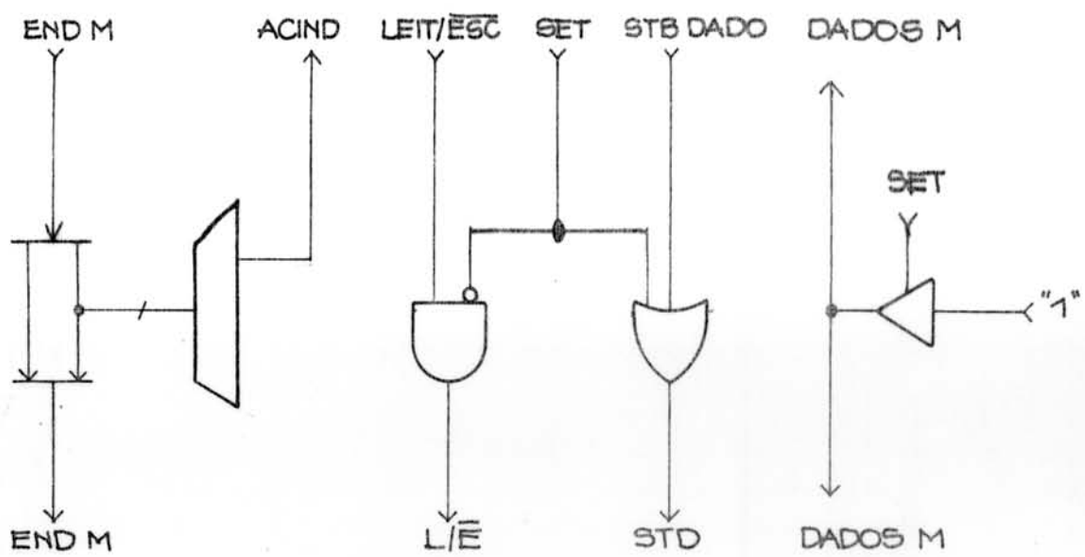


Figura 110 - Alterações no módulo de variáveis semáforo

Pode-se seguir duas estratégias quanto ao instante de envio do sinal PRONTO ao processador: liberá-lo ao final da leitura ou somente após a escrita. Seguindo-se a primeira opção, o processador pode iniciar um novo acesso em paralelo com o bloqueio do semáforo. Isto é favorável ao desempenho, porém significa que o endereço não permanecerá estável no barramento até o fim do ciclo de escrita forçada. Conseqüentemente, seria necessário um outro registrador para garantir a estabilidade do endereço. Para evitar isso, a segunda opção é escolhida. A seqüência de eventos é a seguinte: é executada a operação de leitura; o dado lido é guardado na interface de processador; é executada a operação de escrita; e, por fim, é envia-

do o sinal PRONTO juntamente com o dado lido. O processador fica em estado de espera durante todo o ciclo de escrita. Este inconveniente é amenizado pelo fato de as operações indivisíveis representarem uma fração pequena dos acessos à memória.

O algoritmo de controle é mostrado na figura 111.

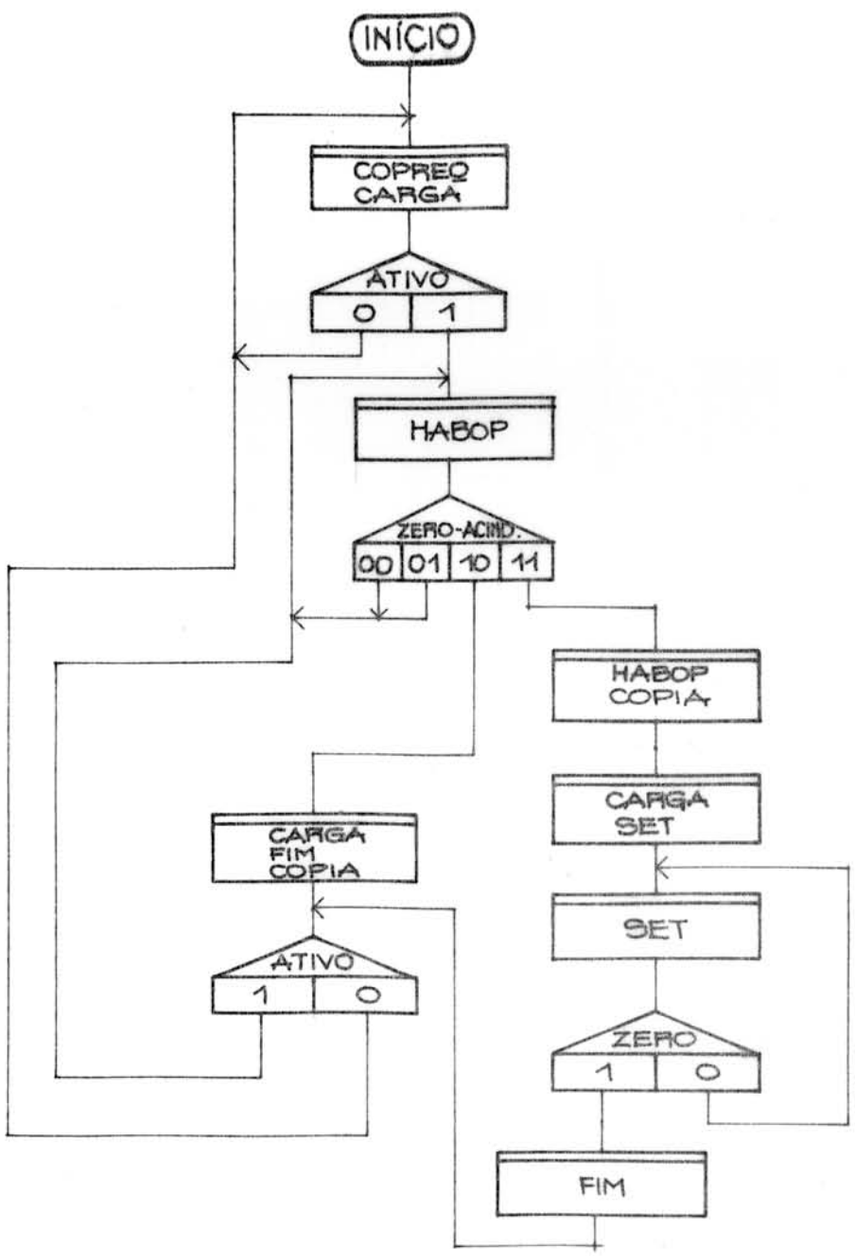


Figura 111 - Algoritmo de controle

8 CONCLUSÃO

8 CONCLUSÃO

A julgar pelo esforço devotado à pesquisa e desenvolvimento parece não haver dúvidas quanto à importância reservada aos multiprocessadores no futuro. É claro que eles não irão tomar o lugar de todas as formas de processamento eletrônico de dados pois a adequabilidade de uma arquitetura está estreitamente relacionada com o tipo de problema que ela irá resolver. Entretanto, os multiprocessadores certamente desempenharão um papel preponderante em aplicações onde é requerido um alto grau de confiabilidade, resposta rápida às requisições, flexibilidade de operação e extensibilidade modular.

O sucesso de arquiteturas multiprocessadas não é projetado a um ponto futuro por mera casualidade. Se considerarmos o pequeno número de máquinas efetivamente implementadas cabe a pergunta: por que, com tantas vantagens reivindicadas, o uso de multiprocessadores não está mais difundido? A resposta a esta pergunta é encontrada observando-se as dificuldades em se atingir os objetivos propostos pela arquitetura.De uma maneira geral os problemas estão colocados em duas áreas: (1) organização do sistema de modo a permitir o compartilhamento dos recursos disponíveis, (2) desenvolvimento de software capaz de explorar efetivamente a potencialidade da arquitetura.

Este trabalho enfocou um dos problemas da área de organização do sistema: a interconexão entre os processadores e a memória. Cabe, agora, tecer algumas considerações a respeito das três estruturas propostas.

O barramento multiplexado apresenta algumas pecu-

liaridades. Ele permite, por exemplo, que ocorra atividade simultânea em mais de um módulo de memória, apesar da existência de um único caminho partilhado entre os processadores. Isto vai ao encontro do propósito de se prover paralelismo ao nível de hardware mantendo, ao mesmo tempo, a simplicidade de uma única via física.

Uma outra característica que distingue o barramento multiplexado é a possibilidade de utilização de memórias mais lentas com uma pequena degradação de performance.

Estas vantagens perdem o seu valor no caso de um grande número de janelas não ser utilizado ou se ocorrer um alto grau de interferência na memória.

Finalmente deve-se salientar que a aplicabilidade desta estrutura é limitada a pequenos sistemas. A inclusão de muitos processadores levaria a longos atrasos nas operações de leitura.

A matriz de barramentos cruzados e as memórias multiporta são estruturas que dão ênfase ao paralelismo a nível de hardware em troca de um custo mais elevado. Ambas equivalem-se em termos de capacidade. Entretanto, a descentralização da lógica de controle, no caso das memórias multiporta, reflete-se negativamente sobre a modularidade do sistema. A configuração máxima deve ser prevista muito cedo no projeto e a lógica necessária para tal implementada desde o início. Neste aspecto a matriz promete uma maior modularidade permitindo a ampliação do sistema através da adição de módulos de chaves. Entretanto, a modularidade também está condicionada aos circuitos de arbitração. Os árbitros usados no projeto da matriz não são modulares. Isto significa que eles irão limitar a configu-

ração máxima.

O número maior de linhas exigido pelas memórias multiporta é um outro fator negativo, tanto no aspecto de modularidade como pelos problemas mecânicos associados.

As considerações acima indicam a matriz de barramentos cruzados como a escolha mais apropriada em relação às memórias multiporta. Deve-se notar, porém, que o crescimento geométrico do número de chaves pode levar a matriz a um tamanho excessivamente grande.

Estudos comparativos em termos de performance foram descartados deste trabalho pelo fato de estarem condicionados a parâmetros não estabelecidos. Estes parâmetros reportam-se fundamentalmente ao sistema operacional.

Também não foram considerados no projeto mecanismos para detecção de erro nas transferências entre memória e processadores. Deve-se ter em mente, entretanto, que a detecção de erros e isolamento de falhas é fundamental em multiprocessamento a fim de que o erro não se propague pelo sistema.

Além da interconexão entre processadores e memórias, outros pontos permanecem em aberto na área de organização do sistema. A estrutura de interrupção, por exemplo, deve ser estudada em conjunto com o sistema operacional, a fim de se determinar a interrelação entre processadores e entre processadores e periféricos.

Entretanto, os maiores problemas obstaculizando o avanço do multiprocessamento não estão localizados na área de organização mas no sistema operacional. Existe uma série de questões ainda não resolvidas. É importante ter conhecimento destas questões para, se não resolvê-las, pelo menos ter cons-

ciência dos compromissos que podem ser feitos. Cita-se entre outros:

(1) a existência de múltiplos recursos compartilhados aumenta a complexidade do software. Este aumento não deve ultrapassar o ganho obtido com o paralelismo.

(2) os detalhes de paralelismo devem ser transparentes ao usuário. A máquina é quem deve controlá-los.

(3) as linguagens de programação devem conter mecanismos para a identificação de tarefas executáveis em paralelo. O sistema operacional deve ser capaz de criar e sincronizar as tarefas executadas independentemente.

(4) a disponibilidade do sistema só é possível se o software possui a capacidade de isolar falhas e reconfigurar o sistema.

Uma possível evolução deste trabalho deve estar vinculada ao projeto global de um sistema multimicroprocessador. Em função dos objetivos e restrições impostas, outras questões, além das mencionadas acima, mais diretamente relacionadas à interconexão processadores-memória merecem uma investigação mais detalhada. Citam-se, a título de exemplo, três questões a serem consideradas para se levar adiante o trabalho:

Uma modificação pode ser introduzida no sentido de permitir transferências de blocos de dados diretamente entre a memória e dispositivos periféricos de alta velocidade. Isto teria um impacto direto sobre os protocolos de arbitração pela existência de requisições críticas.

A relação de tamanho entre a memória global e a local ainda é uma questão em aberto. Seu valor deve ser fixa-

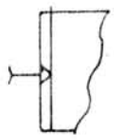
do após a determinação de quais rotinas e estruturas de dados do sistema operacional, bem como programas de aplicação, irão residir na memória local.

A efetividade de qualquer uma das estruturas de interconexão propostas está condicionada ao grau de interferência entre os processadores. Para reduzir esta interferência é interessante estudar o tráfego entre os processadores e a memória através de simulação. Esta deve reproduzir o mais fielmente possível o comportamento dos processadores ao longo do tempo. Para isto devem ser levados em consideração parâmetros tais como: designação de processos entre os processadores, gerenciamento de memória, relação entre acessos à memória local e global, tempo médio entre requisições, tempo de ciclo de memória.

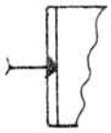
APÊNDICE: SÍMBOLOS UTILIZADOS NOS DIAGRAMAS LÓGICOS

SENSIBILIDADE DE ENTRADAS

BORDA ASCENDENTE



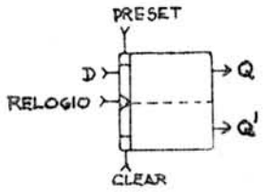
BORDA DESCENDENTE



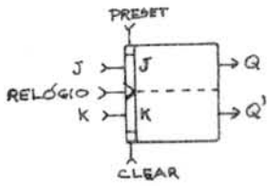
NÍVEL "1" LÓGICO

NÍVEL "0" LÓGICO

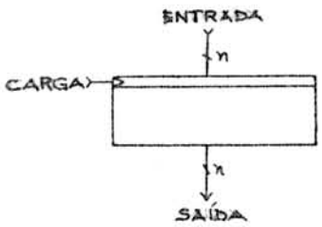
FLIP-FLOP TIPO D



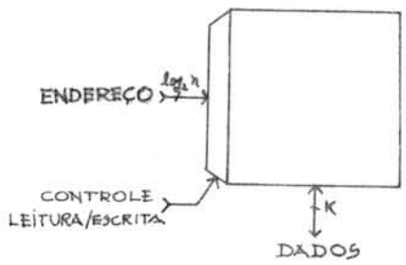
FLIP-FLOP TIPO JK



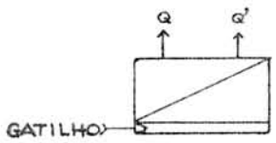
REGISTRADOR DE n BITS



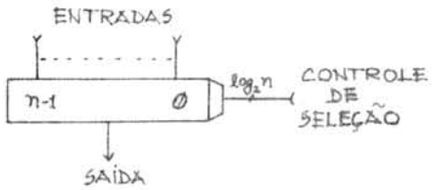
MEMÓRIA DE LEITURA/ESCRITA
(n PALAVRAS DE K BITS)



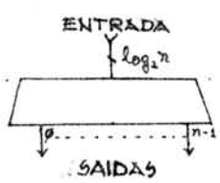
MONOESTÁVEL



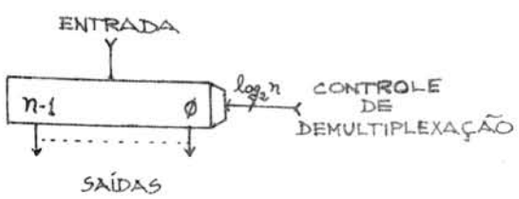
SELETOR



DECODIFICADOR



DEMULTIPLEXADOR



REFERÊNCIAS BIBLIOGRÁFICAS

REFERÊNCIAS BIBLIOGRÁFICAS

1. ADAMS,G. & ROLANDER,T. Design motivations for multiple processor microcomputer systems. Computer Design, Littleton, 17(3):81-9, Mar. 1978.

2. ANDERSON, G.A. & JENSEN, E.D. Computer interconnection structures: taxonomy, charachteristics, and examples. Computing Surveys, New York, 7(4):197-213, Dec. 1975.

3. BATCHER, K.E. Sorting networks and their applications. In: SPRING JOINT COMPUTER CONFERENCE, Atlantic City, Apr. 30 - May 12, 1968. Proceedings. Montvale,AFIPS Press, 1968. p. 307-14.

4. BURNETT, G.J. & GOFFMAN Jr., E.G. A study of interleaved memory systems. In: SPRING JOINT COMPUTER CONFERENCE, Atlantic City, May 5-7, 1970. Proceedings. Montvale, AFIPS Press, 1970. p. 467-74.

5. CHANG, D.Y. et alii. On the effective bandwidth of parallel memories. IEEE Transactions on Computers, New York, C-26(5):480-90, May 1977.

6. CHEN, C.J. & FRANK, A.A. On programmable parallel data routing networks via cross-bar switches for multiple element computer architectures. In: SAGAMORE COMPUTER CONFERENCE ON PARALLEL PROCESSING, 3, Raquette Lake, N.Y., Aug. 20-23, 1974. Proceedings. Berlin, Springer-Verlag, 1975. p. 338-69.

7. COFFMAN Jr., E.G. et alii. On the performance of interleaved memories with multiple word bandwidths. IEEE Transactions on Computers, New York, C-20(12):1570-3, Dec. 1971.

8. COLON, F.C. et alii. Coupling small computers for performance enhancement. In: NATIONAL COMPUTER CONFERENCE, New York, June 7-10, 1976. Proceedings. Montvale, AFIPS Press, 1976. p. 755-64.

9. COMTRE CORPORATION. Multiprocessors and parallel processing. New York, John Wiley, 1974.

10. COSTA, A.C.R. Microprogramação e projeto de sistemas de computação: pré-projeto de um núcleo de sistema operacional multiprogramado. Porto Alegre, CPGCC-UFRGS,1980. (Dissertação de mestrado não publicada)

11. COTTIS, R.A. et alii. The E78 Bus. Euromicro Journal, Amsterdan, 6(2):92-5, Mar. 1980.

12. DANIEL, J.M. & IRWIN, J.D. Dynamic Resolution of Memory Access Conflicts. IEEE Transactions on Computers, New York, C-20(12):1583-86, Dec. 1971.

13. DAVIS, R.L. et alii. A building block approach to multiprocessing. In: SPRING JOINT COMPUTER CONFERENCE, Atlantic City, May 16-18, 1972. Proceedings. Montvale, AFIPS Press, 1972. p. 685-703.

14. DEL CORSO, D. An experimental multimicroprocessor system with improved internal communication facilities. Euromicro Journal, Amsterdan, 4(6):326-32, Nov. 1978.

15. DENNIS, S.F. & SMITH, M.G. LSI - Implications for future design and architecture. In: SPRING JOINT COMPUTER CONFERENCE, Atlantic City, May 16-18, 1972. Proceedings. Montvale, AFIPS Press, 1872. p. 343-51.

16. DURHAM, S.J. Fast LSI arbiters supervise priorities for bus access in multiprocessor systems. Electronic Design, Rochelle Park, 27(11):128-33, May 1979.

17. EL-AYAT, K.A. The Intel 8089: an advanced integrated I/O processor. In: MICROCOMPUTER FIRMWARE AND I/O WORKSHOPS, Laurel, Mar. 6-7, Apr. 9-10, 1979. Proceedings. Long Beach, IEEE Computer Society, 1979. p. 14-22.

18. ENSLOW Jr., P.H. Multiprocessor organization - a survey. Computing Surveys, New York, 9(1):103-29, Mar. 1977.

19. FLYNN, M.J. Toward more efficient computer organizations. In: SPRING JOINT COMPUTER CONFERENCE, Atlantic City, May 16-18, 1972. Proceedings. Montvale, AFIPS Press, 1972. p. 1211-7.

20. FULLER, S.H. & HARBINSON, S.P. The C.mmp multiprocessor. Pittsburgh, Carnegie-Mellon University, 1978. (CMU-CS-78-146).

21. _____ & SIEWIOREK, D.P. Some observations on semiconductor technology and the architecture of large digital modules. Computer, Long Beach, 6(10):15-21, Oct. 1973.

22. GILOI, W. Innovative computer architectures. (Palestra realizada no CPGCC da UFRGS em 1979).

23. GOKE, L.R. & LIPOVSKY, G.J. Banyan networks for partitioning multiprocessor systems. In: ANNUAL SYMPOSIUM ON COMPUTER ARCHITECTURE, 1, Dec. 9-11, 1973. Proceedings. Florida, 1973. p. 21-8.

24. HAYES, J.P. Computer architecture and organization. New York, McGraw-Hill, 1978.

25. HELLERMAN, H. Digital computer system principles. New York, McGraw-Hill, 1967.

26. HOJBERG, K.S. An asynchronous arbiter resolves resource allocation conflicts on a random priority basis. Computer Design, Littleton, 16(8):120-3, Aug. 1977.

27. _____. One-step programmable arbiters for multiprocessors. Computer Design, Littleton, 17(4):154-8, Apr. 1978.

28. _____. Queue handling arbiter solves shared resource conflicts. Computer Design, Littleton, 18(11):123-35, Nov. 1979.

29. HOLT, R.C. et alii. Structured concurrent programming with operating systems applications. Reading, Addison-Wesley, 1978.

30. IEEE Inc. IEEE standard digital interface for programmable instrumentation. New York, 1975. (IEEE STD 488-1975)

31. INTEL CORPORATION. Component Data Catalog. Santa Clara, 1979.

32. _____. MCS-86 product description. Santa Clara, 1979.

33. JASWA, R. Designing interrupt structures for multiprocessor systems. Computer Design, Littleton, 17(9): 101-10, Sept. 1978.

34. JENSEN, E.D. The Honeywell experimental distributed processor - an overview. Computer. Long Beach, 11(1): 28-37, Jan. 1978.

35. KAUTZ, W.H. et alii. Cellular interconnection arrays. IEEE Transactions on Computers, New York, C-17(5): 443-51, May 1968.

36. LEHMAN, M. A survey of problems and preliminary results concerning parallel processing and parallel processors. In: BELL, C.G. & NEWELL, A. Computer structures: readings and examples. New York, McGraw-Hill, 1971. p. 456-69.

37. LE MAIR, I. Indexed mapping extends microprocessor addressing range. Computer Design, Littleton, 19(8): 111-8, Aug. 1980.

38. LIEBOWITZ, B.H. Multiple processor minicomputer systems-part I: design concepts. Computer Design, Littleton, 17(10):87-95, Oct. 1978.

39. LIPOVSKY, G.J. & DOTY, K.L. Developments and directions in computer architecture. Computer, Long Beach, 11(8): 54-67, Aug. 1978.

40. _____ & TRIPATHI, A. A reconfigurable varistructure array processor. In: PARALLEL PROCESSING CONFERENCE, s.l. Proceedings. 1977.

41. MAHR, W. & PATZELT, R. Improvements of the multiprocessing capabilities of microprocessor busses. Euromicro Journal, Amsterdan, 4(4):207-19, July 1978.

42. MOTOROLA SEMICONDUCTOR PRODUCTS. MC68000; advance information. Austin, 1979.

43. NADIR, J. & McCORMIC, B. Bus arbiter streamlines multiprocessor design. Computer Design, Littleton, 19(6): 103-9, June 1980.

44. OLIVEIRA, R.V. Estudo e realização de um sistema multiprocessador. Rio de Janeiro, PUC, 1977. (Tese de mestrado)

45. PEARCE, R.C. et alii. Asynchronous arbiter module. IEEE Transactions on Computers, New York, C-24(9):931-2,Sept. 1975.

46. PIRTLE, M. Intercommunications of processors and memory. In: FALL JOINT COMPUTER CONFERENCE, Anaheim, Nov. 14-16, 1967. Proceedings. Montvale, AFIPS Press, 1967. p. 621-33.

47. PRICE, R.J. Multiprocessing mode easy. In: NATIONAL COMPUTER CONFERENCE, Anaheim, June 5-8, 1978. Proceedings. Montvale, AFIPS Press, 1978. p. 589-96.

48. R.C.A. CD22100 COS/MOS 4x4 cross point switch with control memory. s.l., 1977.

49. SATYANARAYANAN, M. Commercial multiprocessing systems. Computer, Long Beach, 13(5):75-96, May 1980.

50. SEARLE, B.C. & FREBERG, D.E. Tutorial: microprocessor application in multiple processor systems. Computer, Long Beach, 8(10):22-30, Oct. 1975.

51. SECHOVSKY, H. & JURA, S. Asynchronous speed independent arbiter in a form of a hardware control module. In: NATIONAL COMPUTER CONFERENCE, New York, June 7-10, 1976. Proceedings. Montvale, AFIPS Press, 1976. p. 777-82.

52. SELIGMAN, L. LSI and minicomputer system architecture. In: SPRING JOINT COMPUTER CONFERENCE, Atlantic City, May 16-18, 1972. Proceedings. Montvale, AFIPS Press, 1972. p. 767-73.

53. SIEWIOREK, D.P. Process coordination in multimicroprocessor systems. In: WORKSHOP ON THE MICROARCHITECTURE OF COMPUTER SYSTEMS, Nice, June 23-5, 1975. Proceedings. Oxford, North-Holland, 1975. p. 1-8.

54. _____, & DURHAM, I. Multimicroprocessor structures: trends, experience and problems. In: INFOTECH STATE OF THE ART REPORT. Microcomputer systems. Maidenhead, 1978. v.1.

55. SMITH, A.J. Multiprocessor memory organization and memory interference. Communications of ACM, New York, 20(10):754-61, Oct. 1977.

56. SWAN, R.J. et alii. The implementation of the Cm* multimicroprocessor. In: NATIONAL COMPUTER CONFERENCE, Dallas, June 13-16, 1977. Proceedings. Montvale, AFIPS Press, 1977. p. 645-55.

57. TANG, C.K. Cache system design in the tightly coupled multiprocessor system. In: NATIONAL COMPUTER CONFERENCE, New York, June 7-10, 1976. Proceedings. Montvale, AFIPS Press, 1976. p. 749-53.

58. THOMPSON, C.D. Generalized connection networks for parallel processor intercommunication. IEEE Transactions on Computers, New York, C-27(12):1119-25, Dec. 1978.

59. THURBER, K.J. Computer Communication Techniques. Computer Architecture News, ACM SIGARCH. 7(3):7-16, Oct. 15, 1975.

60. _____. Interconnection networks. A survey and assessment. In: NATIONAL COMPUTER CONFERENCE, Chicago, May 6-10, 1974. Proceedings. Montvale, AFIPS Press, 1974. p. 909-19.

61. _____. Parallel processor architectures. Computer Design. Littleton, 18(1):89-97, Jan. 1979.

62. _____, et alii. A systematic approach to the design of digital bussing structures. In: FALL JOINT COMPUTER CONFERENCE, Anaheim, Dec. 5-7, 1972. Proceedings. Montvale, AFIPS Press, 1972. p. 719-40.

63. WITTEN, I.H. Computer buses. Wireless World, London, 85(1518):32-5, Feb. 1979.

64. _____. Computer buses - 2. Wireless World, London, 85(1519):60-2, Mar. 1979.

65. WULF,W.A. & HARBINSON, S.P. Reflections in a pool of processors - an experience report on C.mmp/Hydra. In: NATIONAL COMPUTER CONFERENCE, Anaheim, June 5-8, 1978. Proceedings. Montvale, AFIPS Press, 1978. p. 939-51.

66. YENCHARIS, L. ... But the right bus evokes the system's best. Electronic Design, Rochelle Park, 28(10):125-30, May 1980.

67. ZILOG. Z8001 CPU, Z8002 CPU; Product especification. Cupertino, 1979.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Pós-Graduação em Ciência da Computação da UFRGS

INTERCONEXÃO DE PROCESSADORES E MEMÓRIAS PARA MULTIMICROPROCESSADORES

TESE APRESENTADA AOS SRS.

Philippe Navaux

Dalcidio Moraes Claudio

Francisco Bernardo Moss Silva

Visto e permitida a impressão.

Porto Alegre, 25/05/81.

Coordenador do Curso de Pós-Graduação em Ciência da Computação