Universidade Federal de Santa Catarina

Curso de Pós-Graduação em Ciência da Computação

Projeto do Sistema de Comunicação de um Multicomputador

por

Cesar Albenes Zeferino

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Ciência da Computação.
Orientador: Prof. Altamiro Amadeu Suzim

Florianópolis, maio de 1996

Projeto do Sistema de Comunicação de um Multicomputador

Cesar Albenes Zeferino

Esta dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação, Especialidade Sistemas de Computação, e aprovada em sua forma final pelo Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Santa Catarina.

Prof. Altamiro Amadeu Suzim, Dr.
Orientador, DELET, UFRGS

Prof. Murilo Silva de Camargo, Dr.
Coordenador do Curso, INE, UFSC

Banca Examinadora:

Prof. Altamiro Amadeu Suzim, Dr.
Presidente, DELET, UFRGS

Prof. Thadeu Botteri Corso, M.Sc.
INE, UFSC

Prof. Luis Fernando Friedrich, Dr.
INE, UFSC

Prof. Vitório Bruno Mazzola, Dr.
INE, UFSC

AGRADECIMENTOS

A todos aqueles que de uma forma ou de outra contribuiram à minha formação no Curso de Pós-Graduação em Ciência da Computação (CPGCC) e, principalmente, na realização deste trabalho.

Aos professores Hermann A. H. Lücke, *in memorium*, e Altamiro A. Suzim, pela oportunidade e pela dedicação na condução dos trabalhos.

À toda equipe do Projeto Nó// e aos colegas do curso de mestrado.

À Verinha e à Valdete, a dupla administrativa do CPGCC.

Aos integrantes do Laboratório de Instrumentação Eletrônica (LINSE).

Ao CNPq, pelo apoio financeiro.

Aos professores José R. Pinheiro, Humberto Pinheiro e Augusto C. Zeferino, meu tio, que me incentivaram para que eu seguisse este caminho.

Aos Engenheiros de Camobi, Thomaz, Alexandre, Pedro, Gláucio, Rizzati, Douglas e Marcos, que caminharam comigo na busca de um objetivo comum.

À minha namorada, Jaine, pelo seu amor, apoio, compreensão, revisão do texto e pela sua presença.

Aos meus pais, Albenes e Elenita, e à minha irmã, Enelise, pelo amor, apoio financeiro, orações e revisão do texto.

À Deus.

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

RESUMO

A busca por sistemas de computação capazes de atingir elevadas performances de processamento tem levado os pesquisadores e cientistas a propor e desenvolver diferentes modelos de arquiteturas de computadores de alto desempenho. O Projeto Nó// (lê-se nó paralelo), do qual participam grupos de pesquisa das Universidades Federais de Santa Catarina e do Rio Grande do Sul, também insere-se nesse contexto. Esse projeto visa o desenvolvimento de um ambiente completo para programação paralela, incluindo a construção de um multicomputador com rede de interconexão dinâmica. O presente trabalho vem colaborar com a concepção desse multicomputador, através do projeto do sistema de comunicação necessário à interação entre os processadores da máquina. Em um primeiro instante, realiza-se uma revisão da literatura a respeito das arquiteturas de alto desempenho e apresenta-se dois modelos de máquina propostos para o multicomputador Nó//. Após, faz-se um estudo comparativo desses dois modelos, visando determinar, a partir de uma análise de custo e desempenho, aquele mais adequado à construção de um primeiro protótipo do multicomputador. Por fim, descreve-se os projetos físico e lógico do sistema de comunicação para o modelo de máquina definido.

ABSTRACT

The pursuit of high performance computing systems leads the scientists to propose several machine models. The Nó// Project (reads Parallel node) aims to develop a parallel programming environment, which includes the design of a multicomputer with a dynamic interconnection network. This project involves research groups of the Federal Universities of Santa Catarina and Rio Grande do Sul. This work presents the design of a internode communication system for the Nó// multicomputer. In a first moment, it is made a literature review about some high performance architectures. Afterwards, two machine models proposed for the Nó// multicomputer are presented. Then, it is performed a comparative study between these two models, analysing cost and performance, to establish the most adequate to build a first multicomputer prototype. Finally, the physical and logical designs of the communication system for the machine prototype are described.

INTRODUÇÃO

Desde a invenção dos primeiros computadores mecânicos, até os dias de hoje (1996), cientistas e pesquisadores têm buscado a construção de máquinas com capacidades de processamento cada vez maiores. A queda sustentada dos custos de *hardware* durante as últimas décadas proporcionou que fosse dado um grande passo nesse sentido. A velocidade com a qual a tecnologia tem evoluído é tamanha, que aquilo que hoje é considerado um padrão, amanhã tornar-se-á, inevitavelmente, obsoleto.

Atualmente, existem diversos modelos de arquiteturas de computadores que apresentam alta *performance* de processamento, como os computadores vetoriais, os processadores matriciais e os multiprocessadores, entre outros. Alguns deles são chamados de "supercomputadores", pois proporcionam índices máximos de desempenho. Outros são menos poderosos, mas oferecem relações custo × desempenho altamente satisfatórias.

Dentro do contexto da Universidade Federal de Santa Catarina (UFSC), está em andamento um projeto para a construção de um ambiente completo de programação paralela, destinado ao desenvolvimento de aplicações em alto desempenho. Esse projeto, chamado Projeto Nó// (lê-se nó paralelo), conta, ainda, com a colaboração da Universidade Federal do Rio Grande do Sul (UFRGS), envolvendo pesquisadores das áreas de Arquitetura de Computadores, Sistemas Operacionais e Simulação.

Entre os objetivos principais do Projeto Nó// estão incluídos o desenvolvimento de um sistema operacional distribuído e a construção de um multicomputador com rede de interconexão dinâmica. Essa máquina apresentará diversos processadores interligados de modo a cooperarem entre si para a solução de um mesmo problema. O trabalho aqui apresentado

vem colaborar com a concepção desse multicomputador através do projeto do sistema de comunicação que proverá meios para a interação entre os processadores da máquina.

O trabalho é dividido em cinco capítulos. No primeiro, far-se-á uma revisão da literatura, onde será apresentado um estudo sobre arquiteturas de computadores, com ênfase àquelas que visam a alta *performance* de processamento. Inicialmente, apresentar-se-á um histórico da evolução dos computadores, desde a era mecânica até a última geração de computadores eletrônicos. Após, serão apresentados os principais modelos de arquiteturas de alto desempenho, onde serão descritos os computadores vetoriais, os processadores matriciais, os multiprocessadores e os multicomputadores. No final do capítulo, mostrar-se-ão algumas topologias de redes de interconexão utilizadas nesses computadores.

O segundo capítulo descreverá o Projeto Nó// e dará ênfase aos aspectos relacionados à arquitetura da máquina paralela definida para o projeto. Inicialmente, será apresentado um modelo geral de multicomputador com rede de interconexão dinâmica, para o qual serão mostradas duas propostas específicas de arquitetura. Ainda, nesse capítulo, será apresentada uma visão global do ambiente de programação paralela, através da descrição da sua estrutura geral.

No capítulo seguinte, mostrar-se-ão decisões básicas tomadas para a concepção do projeto. Serão definidos o protocolo de nível físico, o padrão de barramento e o modelo de arquitetura da máquina paralela.

No quarto capítulo, será efetuada uma descrição detalhada do projeto do sistema de comunicação. Serão mostrados os projetos físico e lógico dos adaptadores de compõe o sistema de comunicação.

O último capítulo apresentará resultados de experimentos de simulação realizados sobre o projeto lógico dos adaptadores de comunicação. Ao concluir, serão feitas algumas considerações sobre o projeto e suas perspectivas futuras.

CAPÍTULO 1

ARQUITETURAS DE COMPUTADORES

Introdução

Este capítulo apresenta um estudo sobre arquiteturas de computadores, com ênfase àquelas que visam ao alto desempenho. Seu objetivo não é esgotar tal assunto, e sim situar o trabalho proposto no contexto ao qual ele se refere.

Durante o decorrer do trabalho, o termo "arquitetura" será utilizado de forma ampla, englobando não apenas os aspectos arquiteturais do computador, mas, principalmente, aqueles relacionados à sua organização.

Inicialmente, é feita uma revisão histórica do desenvolvimento dos computadores, onde são mostradas as principais contribuições dadas pelos cientistas, desde os primeiros computadores mecânicos até a última geração de computadores eletrônicos. Após, na segunda seção, são apresentadas as principais arquiteturas de alto desempenho, incluindo os computadores vetoriais, os processadores matriciais, os multiprocessadores e os multicomputadores. Por fim, são descritas algumas topologias de redes de interconexão utilizadas em máquinas com múltiplos processadores.

1.1 - Histórico

A história da computação iniciou-se na primeira metade do século XVII, quando Pascal construiu um contador mecânico que realizava operações de adição e subtração. Esse feito ocorreu no ano de 1642, dando início à era dos computadores mecânicos. Desde então, diversos trabalhos foram realizados no sentido de construir máquinas para propósitos específicos e geral.

Com o advento da eletrônica digital, na primeira metade deste século, começaram a surgir os primeiros computadores eletrônicos, mais rápidos e mais confiáveis que os seus equivalentes mecânicos. A evolução no campo da eletrônica permitiu o desenvolvimento de máquinas cada vez mais poderosas, caracterizando cinco gerações de computadores eletrônicos, cada qual baseada em uma tecnologia específica.

Este sub-capítulo apresenta a história da evolução dos computadores, mostrando a contribuição dada pelos cientistas e pesquisadores ao desenvolvimento da computação.

1.1.1 - Os computadores mecânicos

A era dos computadores mecânicos iniciou em 1642 com os trabalhos de Blaise Pascal, filósofo e cientista francês que construiu um somador/subtrator mecânico. Seguindo o caminho de Pascal, Gottfried Leibniz desenvolveu, em 1671, uma calculadora capaz de efetuar operações de multiplicação e divisão, assim como de adição e subtração. Posteriormente, em 1801, Joseph Jacquard construiu uma máquina para automação do processo de tecelagem programada através de cartões perfurados.[1]

Uma importante contribuição para o desenvolvimento da computação, durante a era dos computadores mecânicos, foi dada pelo inglês Charles Babbage. Em 1821, esse cientista propôs uma máquina para avaliação de polinômios através do método das diferenças finitas. Após concluído o primeiro protótipo, o qual era capaz de resolver polinômios de segundo grau, Babbage projetou, mas não construiu, um modelo mais poderoso que poderia avaliar polinômios de até o sexto grau. Ele projetou, ainda, um computador de propósito geral constituído por um módulo armazenador, uma unidade aritmética e um dispositivo de entrada e saída via cartões perfurados. Esse computador, chamado Máquina Analítica, também não foi concluído, pois o seu projeto estava muito além da tecnologia disponível na época.[2]

Mais de um século depois dos trabalhos de Babbage, em 1841, o alemão Konrad Zuse construiu o primeiro computador mecânico de propósito geral totalmente operacional - chamado Z3. Também no mesmo período, Howard Aiken iniciou o desenvolvimento de uma

---

[1] STALLINGS, William. **Computer organization and architecture** : principles of structure and function. 3. ed. New York : Macmillan, 1993. p. 15-17.

[2] PERROT, Ronald H. **Parallel programming**. Great Britain : Addison-Wesley, 1987. p. 3-4.

máquina baseada no projeto de Babbage. Essa máquina tornou-se operacional em 1944 e era denominada Mark I.[3]

1.1.2 - Os computadores eletrônicos

O início da era dos computadores eletrônicos deu-se com o uso da válvula de tubo de vácuo como bloco construtivo básico. Após, vieram o transistor e os circuitos integrados em diferentes escalas. O uso de cada uma dessas tecnologias caracterizou a existência de cinco gerações de computadores eletrônicos, conforme é mostrado na Tabela 1.1.

Tabela 1.1 - Gerações de computadores eletrônicos.

Geração	Período	Tecnologia
1	1945-54	Válvula de tubo de vácuo
2	1955-64	Transistor
3	1965-74	Integração em pequena e em média escala (SSI e MSI)
4	1975-90	Integração em larga e em muito-larga escala (LSI e VLSI)
5	1991-presente	integração em ultra-larga escala (ULSI)

Fonte: HWANG, Kai. **Advanced computer architecture** : parallelism, scalability, programmability. 1993. p. 5.

1.1.2.1 - A primeira geração

O primeiro computador eletrônico de propósito geral foi o ENIAC (*Electronic Numerical Integrator And Computer*), desenvolvido, na Universidade da Pensilvânia, em 1946, para ser utilizado em projetos militares do governo dos Estados Unidos. Essa máquina marcou o início da primeira geração de computadores eletrônicos, da qual destacam-se também o EDSAC *(Electronic Delay Storage Automatic Computer*) da Universidade de Cambridge, o EDVAC *(Electronic Discrete Variable Arithmetic Computer*) da Universidade da Pensilvânia, o IAS (*Institute for Advanced Studies*) de Princeton, os computadores UNIVAC (*Universal Automatic Computer*) I, II e série 1100 da Sperry-Rand Corporation e os computadores da série 700 da IBM.[4, 5]

---

[3] STALLINGS, op. cit., p. 20-21.

[4] Id., p. 22-33.

[5] PERROT, op. cit., p. 4.

Durante o período da primeira geração, John von Neumann, um matemático que participou de diversos projetos, incluindo o ENIAC, o EDVAC e o IAS, propôs um novo modelo de computador baseado no conceito de programa armazenado.[6] Nesse modelo, o computador é constituído por quatro unidades funcionais básicas, a saber:

- memória principal, onde são armazenados os dados e as instruções do programa;
- unidade de controle, responsável pela interpretação e execução das instruções;
- unidade lógica e aritmética, responsável pelo processamento dos dados;
- dispositivo de entrada e saída (I/O - *Input/Output*), para comunicação com o meio externo.

A unidade de controle é agrupada com a unidade lógica e aritmética em uma terceira, chamada unidade central de processamento (CPU - *Central Processing Unit*). A Figura 1.1 apresenta a estrutura proposta por von Neumann.

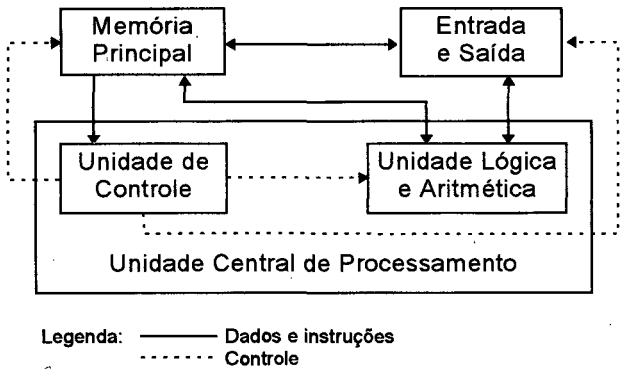


Figura 1.1 - O modelo de von Neumann

Esse modelo de máquina foi largamente utilizado no desenvolvimento de computadores dessa e de outras gerações, sendo ainda aplicado nos computadores atuais (1996).

Além do uso de válvulas, os computadores da primeira geração foram caracterizados pela utilização de memórias a relé, aritmética de ponto fixo, acesso à entrada e

[6] STALLINGS, op. cit., p. 23-25.

saída controlado exclusivamente pela CPU, programação em linguagens de baixo nível e sistemas do tipo mono-usuário.[7]

1.1.2.2 - A segunda geração

A segunda geração de computadores eletrônicos começou no final dos anos 50, quando o transistor passou a ser utilizado como bloco construtivo básico. Inventado nos laboratórios da Bell Telephone, em 1947, o transistor era mais compacto, com maior confiabilidade e menor consumo de potência e geração de calor que as válvulas. Seu uso possibilitou a construção de computadores com maior número de componentes ativos e, conseqüentemente, com maior capacidade de processamento.[8, 9]

As máquinas dessa geração foram construídas utilizando aritmética de ponto flutuante, memórias ferromagnéticas, processadores de entrada e saída e acesso multiplexado à memória. A nível de *software* e aplicações, foram introduzidas linguagens de programação de alto nível com compiladores, bibliotecas de subrotinas e monitores de processamento em lote.[10]

A segunda geração se estendeu até a metade dos anos 60 e marcaram esse período os computadores IBM série 7000, o DEC PDP-1, o CDC 1604 e o UNIVAC LARC.[11, 12]

1.1.2.3 - A terceira geração

Desde o início da era da eletrônica digital, os esforços dos pesquisadores sempre se voltaram no sentido de reduzir o tamanho dos circuitos eletrônicos. A substituição das válvulas de tubo de vácuo pelos transistores foi o primeiro grande passo dado nesse sentido. Depois disso, no final dos anos 50, surgiram os circuitos integrados, o que possibilitou o agrupamento de centenas de transistores em um único componente. A terceira geração de computadores eletrônicos foi caracterizada pelo uso desses circuitos com integração em pequena e em média escala (SSI - *Small Scale Integration*; MSI - *Medium Scale Integration*).

---

[7] HWANG, op. cit., p.5.

[8] STALLINGS, op. cit., p. 33.

[9] PERROT, op. cit., p.4.

[10] HWANG, op. cit., p. 5.

[11] STALLINGS, op. cit., p. 34.

[12] HWANG, op. cit., p. 5-6.

Outros aspectos que caracterizaram os computadores dessa geração foram a introdução do uso de memórias *cache* e de processamento em *pipeline*, para minimizar o gape de velocidade entre a CPU e a memória principal, e o uso de técnicas de multiprogramação. Além disso, foram desenvolvidos sistemas operacionais de tempo compartilhado e aplicações multi-usuário.[13]

Dos computadores desenvolvidos durante o período da terceira geração (1965-1974) destacam-se a família de *mainframes* IBM System/360, o minicomputador DEC PDP-8, o CDC 6600 e o TI ASC.[14, 15]

1.1.2.4 - A quarta geração

A quarta geração de computadores eletrônicos foi marcada pelo uso de circuitos integrados em larga e em muito-larga escala (LSI - *Large Scale Integration*; VLSI - *Very Large Scale Integration*) com a utilização de memórias semicondutoras.

Buscando modelos alternativos àquele proposto por von Neumann, foram desenvolvidas diversas máquinas baseadas nos conceitos de paralelismo e *pipelining*, como multiprocessadores, multicomputadores e supercomputadores vetoriais. Também foram desenvolvidos sistemas operacionais, linguagens, compiladores e ambientes para processamento paralelo. Essa geração teve início em 1975 e prolongou-se até 1990. Entre os computadores mais representativos estão o VAX 9000, o Cray X-MP, o IBM 3090 e o BBN TC2000.[16]

1.1.2.5 - A quinta geração

A quinta geração constitui-se na mais recente geração de computadores eletrônicos e foi iniciada em 1991. Utilizando circuitos integrados em ultra-larga escala (ULSI - *Ultra Large Scale Integration*), as máquinas dessa geração vêm sendo desenvolvidas com base no conceito de processamento massivamente paralelo (MPP - *Massively Parallel Processing*). Além desse, um outro conceito emergente é o de processamento heterogêneo, no qual é

---

[13] Ibid.

[14] STALLINGS, op. cit., p. 37.

[15] HWANG, op. cit., p. 6.

[16] Id., p. 5-6.

utilizada uma rede de computadores heterogêneos para solução de problemas de larga escala. Entre alguns dos sistemas mais representativos dessa geração estão o Fujitsu VPP500, o Cray/MPP, o TMC/CM-5 e o Intel Paragon.[17]

1.2 - Arquiteturas de alto desempenho

Os computadores convencionais têm arquitetura baseada no modelo proposto por von Neumann. Nesse modelo, as instruções de um programa são executadas seqüencialmente em um ciclo constituído por seis etapas, conforme a Figura 1.2.

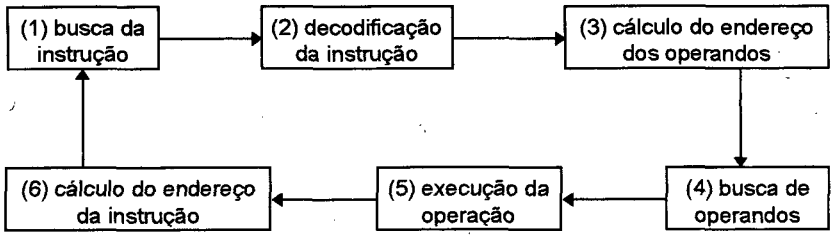


Figura 1.2 - Ciclo de execução de uma instrução

As etapas de busca da instrução e dos operandos constituem-se em operações de acesso à memória principal. Uma vez que o ciclo de memória é muito mais lento que o ciclo de máquina, o fluxo de dados entre o processador e a memória, durante essas duas etapas, limita o tempo de execução de uma instrução. Esse fato caracteriza-se na principal restrição ao desempenho de um computador seqüencial e é conhecido como "gargalo de von Neumann".

A taxa de processamento das instruções em um computador é também limitada pelas capacidades: (i) do processador, que executa apenas uma instrução por vez; (ii) da memória, que permite no máximo uma operação de leitura ou escrita por ciclo; e (iii) do sistema de entrada e saída, que transfere somente um dado por ciclo. Entretanto, de acordo com ALMEIDA e ÁRABE[18], esses fatores limitantes ao desempenho do computador podem ser contornados pelo uso das seguintes técnicas:

---

[17] Ibid.

[18] ALMEIDA, Virgilio A. F. & ÁRABE, José Nagib C. **Introdução à supercomputação**. Rio de Janeiro : LTC, 1991. p. 12-20.

- redução do tempo do ciclo de máquina com o desenvolvimento da tecnologia dos componentes;
- redução do ciclo de instrução por meio da superposição de algumas etapas, como por exemplo: realização simultânea das etapas de busca de instrução e de operandos com a etapa de execução da operação;
- aumento da taxa de transferência de dados entre o processador e a memória; o que pode ser feito através de uma estrutura hierárquica de memória composta por diversos níveis: registradores, *cache*[19], memórias principal e secundária, conforme a Figura 1.3. Nessa estrutura, os dados referenciados com maior freqüência são armazenados em memórias mais rápidas, mas de menor capacidade. Os dados menos utilizados são mantidos em memórias de maior capacidade de armazenamento, porém, mais lentas que as primeiras. O custo por *bit* de memória aumenta com a redução do tempo de acesso.

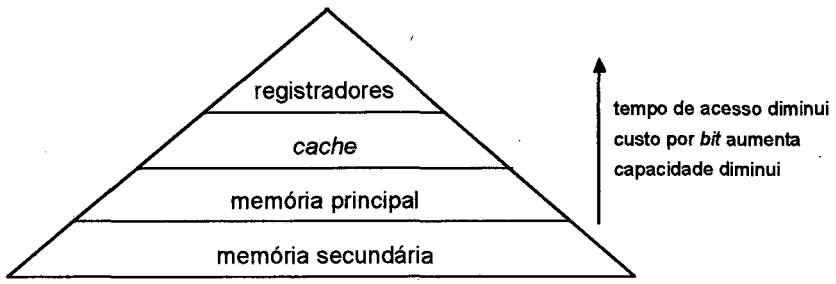


Figura 1.3 - Uma estrutura hierárquica de memória.

O uso dessas técnicas leva a um incremento considerável à performance de uma máquina. Porém, algumas aplicações, como a simulação de sistemas de parâmetros distribuídos, processamento de dados de sinais amostrados e simulação em tempo real de sistemas dinâmicos, requerem capacidades de processamento ainda maiores. Diversos modelos alternativos ao de von Neumann têm sido propostos com o objetivo de atender às necessidades

[19] "A *cache* é uma memória de pequena capacidade e de rápido acesso colocada entre a CPU e a memória principal. [...] Nela são mantidas cópias de porções da memória principal. Quando a CPU tenta ler uma palavra da memória é realizado um teste para determinar se a palavra está na *cache*. Se estiver, o dado é enviado para a CPU. Se não, um bloco de memória principal, com um determinado número de palavras, é lido para a *cache* e, então, a palavra é enviada para a CPU. Devido ao fenômeno da localidade, quando um bloco de dados é buscado para dentro da *cache* para satisfazer a uma única referência de memória, é muito provável que futuras referências venham a ocorrer às outras palavras do bloco." (STALLINGS, 1993, p. 156).

dessas e de outras aplicações. Esses modelos baseiam-se nos conceitos de *pipelining* e de paralelismo e têm permitido o desenvolvimento de diversas famílias de computadores digitais de alta velocidade, entre os quais estão incluídos os computadores vetoriais, os processadores matriciais e os multiprocessadores.[20]

O presente sub-capítulo trata dessas arquiteturas, as quais são descritas a partir da definição dos conceitos nos quais se baseiam. São mostradas também algumas metodologias utilizadas na literatura para classificar os computadores segundo as suas arquiteturas.

1.2.1 - Paralelismo e *pipelining*

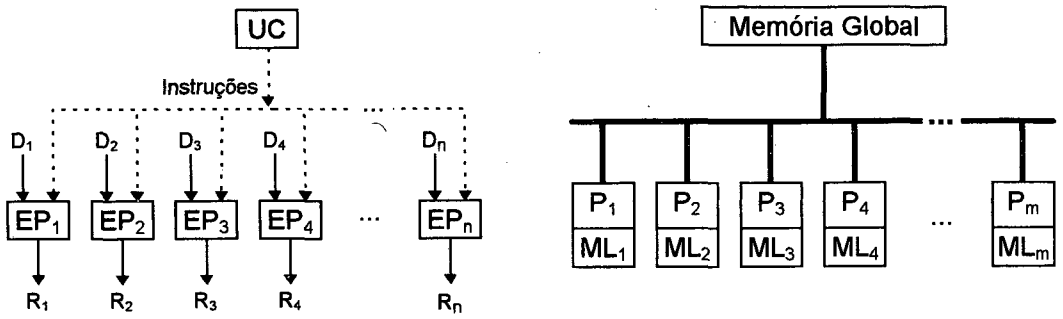
No contexto desse trabalho, paralelismo refere-se à replicação das unidades processadoras de um computador. Tal replicação possibilita a execução simultânea da mesma função pelas unidades sobre diferentes conjuntos de dados, o que caracteriza um paralelismo espacial síncrono. Permite, também, que múltiplas unidades cooperem entre si para a solução de um mesmo problema, sendo que cada unidade tem a responsabilidade de processar uma parte específica do problema. Tem-se, nesse caso, um paralelismo espacial assíncrono.

Nas máquinas que exploram o paralelismo espacial síncrono (Figura 1.4.a), como os processadores matriciais, existem vários elementos de processamento (EPs) que executam a mesma instrução sob o comando de uma única unidade de controle (UC). Cada elemento de processamento opera sobre um conjunto de dados (D) próprio de modo a produzir seus resultados (R).

Já nas máquinas que exploram o paralelismo espacial assíncrono (Figura 1.4.b), como os multiprocessadores, existem diversos processadores (P) que cooperam entre si para a solução de um mesmo problema. Nessa arquitetura, em especial, a cooperação ocorre através do acesso a uma memória global compartilhada, sendo que cada processador pode ter sua própria memória local (ML).

---

[20] KARPLUS, Walter J. Vector processors and multiprocessors. In: HWANG, Kai & DEGROOT, Douglas. **Parallel processing for supercomputers and artificial inteligence**. New York : McGraw-Hill, 1989. p. 3-5.



(a) Paralelismo espacial síncrono

(b) Paralelismo espacial assíncrono

Figura 1.4 - Arquiteturas que exploram o paralelismo espacial: (a) síncrono e (b) assíncrono.

O *pipelining*[21] baseia-se na exploração do paralelismo a nível temporal. Segundo FERNÁNDES e AMORIM:

> "O princípio de *pipelining* propõe a divisão de uma função genérica em uma seqüência de *k* sub-funções que possam ser implementadas por *k* módulos de *hardware* dedicados e autônomos, denominados estágios. Cada estágio é capaz de receber dados do estágio anterior, operá-los e transmitir o resultado para o estágio seguinte. Dessa forma, sucessivas execuções da função podem ser conduzidas pelas sub-funções operando por superposição". [22]

A Figura 1.5 apresenta um *pipeline* com cinco estágios cadenciados por um relógio com período *t* igual ao tempo do estágio mais lento. Cada instância da função f_i (i = 1, 2, 3, ..., n) leva cinco ciclos de relógio para ser executada. Entretanto, após a execução da primeira, haverá sempre uma nova instância pronta na saída do *pipeline*.

---

[21] O termo *pipelining* pode ser traduzido para a língua portuguesa como "processamento em duto" ou "processamento em linha de canalização". Contudo, costuma-se utilizar nas publicações nacionais especializadas o seu original em inglês.

[22] FERNÁNDES, Edil S. T. & AMORIN, Claudio L. de. **Arquiteturas paralelas avançadas**. In: VI Escuela Brasileño Argentina de Informática (1993 : Córdoba). Córdoba : Programa Argentino - Brasileño de Investigación y Estudios Avanzados en Informática, jul. 1993. p. 119.

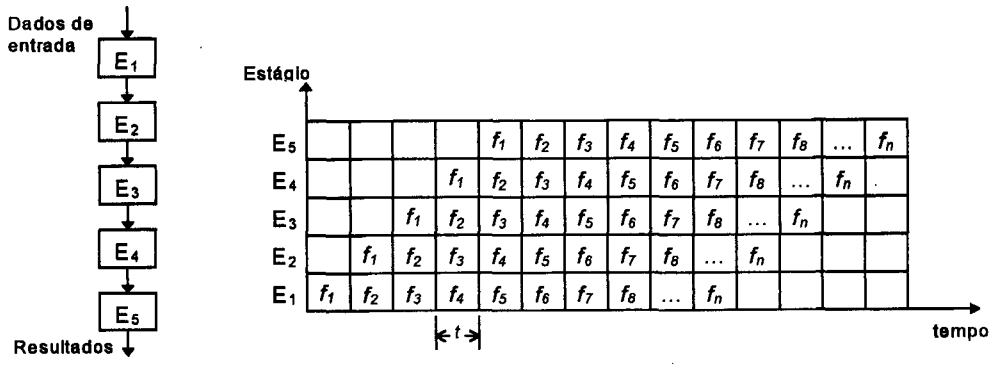


Figura 1.5 - Execução de uma função em um *pipeline* com cinco estágios.

1.2.2 - Computadores vetoriais

Os computadores vetoriais constituem-se na mais importante classe de arquitetura de alto desempenho para resolução de problemas numéricos. Essas máquinas apresentam uma notável relação custo × desempenho, com taxas de processamento que chegam a ser duas ordens de grandeza superiores às atingidas pelos computadores seqüenciais.[23]

Esses computadores utilizam estruturas *pipeline* em ***hardware*** para implementar instruções específicas para operação com vetores, chamadas instruções vetoriais. Em geral, essas instruções são agrupadas em quatro classes de operações (*op*), conforme os tipos dos operandos utilizados e dos resultados que retornam:[24]

- *vetor* ← *vetor (op) vetor* - Ex.: soma vetorial, produto interno, média;
- *vetor* ← *vetor (op) escalar* - Ex.: soma (vetor + escalar), divisão (vetor/escalar);
- *vetor* ← *(op) vetor* - Ex.: raiz quadrada vetorial, seno vetorial;
- *escalar* ← *(op) vetor* - Ex.: somatório, máximo.

Um exemplo clássico de operação vetorial é a soma de dois vetores. Uma operação de adição em ponto flutuante pode ser dividida em quatro etapas elementares: (i) ajuste dos expoentes; (ii) soma das mantissas; (iii) normalização do resultado; e (iv) arredondamento do resultado. Essas quatro etapas podem ser implementadas como estágios de um *pipeline*, conforme a Figura 1.6.

---

[23] ALMEIDA, op. cit., p. 24.

[24] FERNÁNDES, op. cit., p. 124.

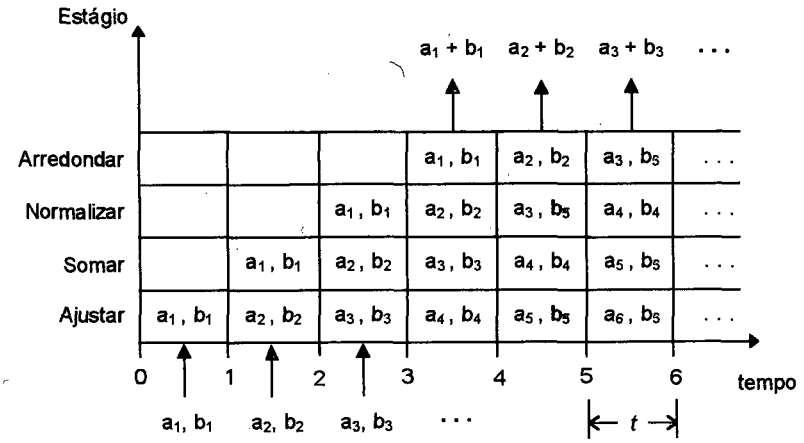


Figura 1.6 - *Pipeline* para adição de vetores.

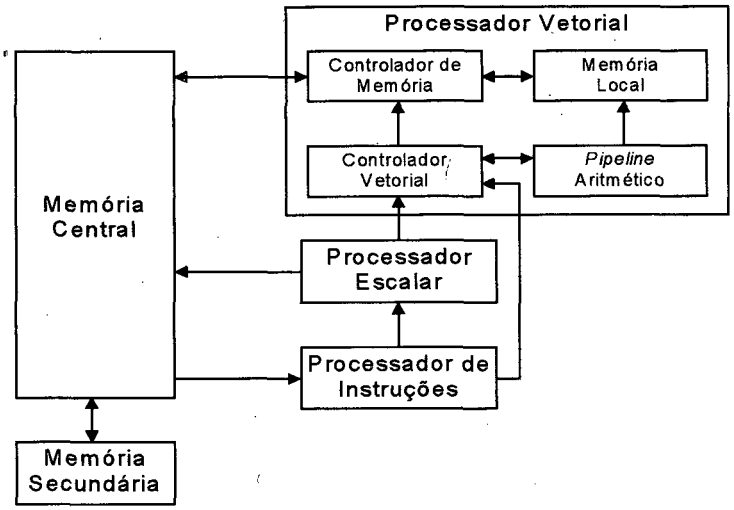
Nesse *pipeline*, cada operação de adição leva quatro ciclos para ser executada. Entretanto, enquanto o primeiro par de operandos é processado pelo segundo estágio, um novo par é introduzido na entrada do *pipeline*. Dessa forma, após completada a primeira operação, novos resultados são produzidos a cada ciclo do *pipeline*. Já em um processamento seqüencial, tal sobreposição não ocorre e uma nova operação só pode ser iniciada após o término da anterior.

Nos computadores vetoriais existem diversas unidades funcionais em *pipeline*, com diferentes números de estágios. De forma geral, esse tipo de arquitetura é organizado conforme a Figura 1.7, com uma unidade para o processamento de instruções, processadores escalar e vetorial e uma memória hierarquizada.

O processador de instruções realiza a busca e decodificação das instruções de programa. Cada instrução, após decodificada, é avaliada e enviada ao processador adequado ao seu tipo: escalar ou vetorial. O processador escalar realiza operações tanto em ponto fixo, como em ponto flutuante e dispõe de registradores internos para o armazenamento de resultados intermediários.

O processador vetorial possui diversas estruturas em *pipeline* que implementam unidades mono ou multifuncionais, as quais executam, respectivamente, uma ou várias operações aritméticas. A maioria dos processadores vetoriais possui, pelo menos, uma unidade para adição e outra para multiplicação em ponto flutuante. Além dessas, alguns modelos apresentam unidades para divisão e para extração da raiz quadrada. Em geral, um processador

vetorial pode conter diversas unidades funcionais para esses e outros fins, cada uma com capacidade de operar de forma independente. O conjunto de unidades funcionais forma o núcleo do processador vetorial.



Fonte: KARPLUS, Walter J. **Vector processors and multiprocessors**. 1989. p. 7.

Figura 1.7 - Organização geral de um computador vetorial.

A memória dos computadores vetoriais é estruturada de forma hierárquica, com registradores, *buffers*, memórias principal e secundária. A memória principal é decomposta em diversos módulos intercalados, de modo a permitir o acesso simultâneo a mais de uma palavra de memória e aumentar a banda passante. Os *buffers* são utilizados para o armazenamento de blocos de instruções e permitem a minimização do acesso à memória principal. Os bancos de registradores formam as memórias locais dos processadores escalar e vetorial.

Alguns computadores vetoriais possuem múltiplos processadores com capacidade escalar e vetorial ou, então, diversas unidades escalares e vetoriais independentes. Na Tabela 1.2 são apresentados seis modelos de computadores vetoriais.

Tabela 1.2 - Modelos de computadores vetoriais.

Fabricante	Modelo	No. máx. de processadores	Ciclo de máquina (ns)	Cap. máx. de memória (Mbytes)	Pico de Performance (Mflops)	* OBS.
Cray	Y-MP	8	6,0	256	2.700	
NEC	SX-3 / 44	4	2.9	2.048	22 K*	22 Gflops
Fujitsu	VP-400	1/1*	15,0/7,0*	256	1.140	escalar/vetorial
Hitachi	S-820 / 80	1/1*	8,0/4,0*	512	3.000	escalar/vetorial
IBM	3090	6	14,5	2.048	828	
DEC	VAX 9000	4 / 4 *	não obtido	512	500	escalar/vetorial

Fonte: ALMEIDA, Virgilio A. F. & ÁRABE, José Nagib C. **Introdução à supercomputação**. 1991, p. 56-75.

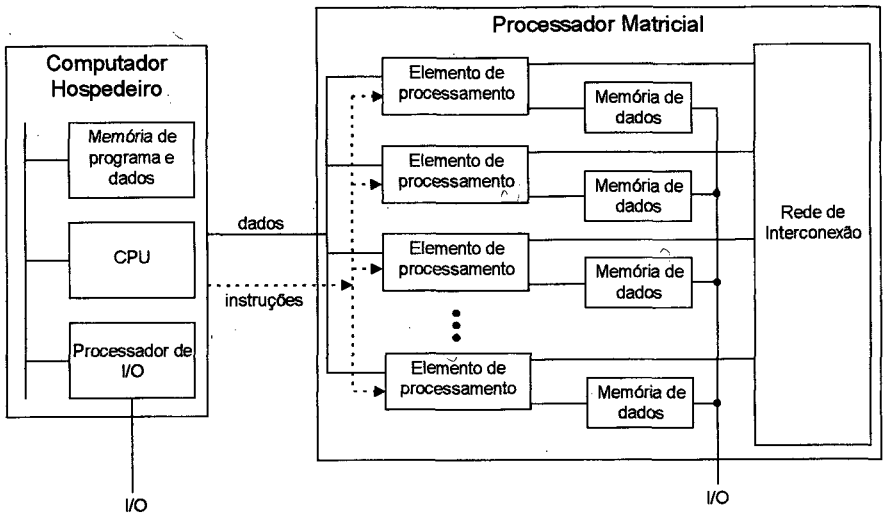
1.2.3 - Processadores matriciais

Um processador matricial é um computador vetorial implementado como um conjunto de elementos de processamento idênticos, interconectados de alguma forma e sincronizados por uma mesma unidade de controle. Todos os processadores realizam a mesma instrução sobre diferentes dados e cada um deles pode ser mascarado, ou não, para a execução de alguma instrução específica. Conforme Walter J. KARPLUS,

> "Um processador matricial é projetado para operar conjuntamente a um computador convencional (o hospedeiro) de modo a melhorar sua performance em algumas aplicações numéricas. Ele é acoplado ao computador hospedeiro através de uma unidade de interface, a fim de atuar como um periférico. A alta performance é alcançada através do uso de paralelismo e/ou *pipelining*". [25]

A Figura 1.8 apresenta um modelo realístico de um processador matricial.

[25] KARPLUS, op. cit., p. 9-10.



Fonte: QUINN, Michael J. **Parallel computing** : theory and practice. 1994, p. 62.

Figura 1.8 - Organização geral de um processador matricial.

Nesse modelo, o computador hospedeiro executa o programa de forma seqüencial e, quando encontra uma instrução cujo operando é um vetor, transfere a tarefa de executar a instrução para o processador matricial, o qual é formado por diversos elementos de processamento. Cada um desses elementos possui uma pequena memória local na qual é armazenada uma parte do vetor de dados a ser manipulado em paralelo pelo arranjo de processadores. A instrução é difundida a todos os elementos do arranjo e executada por aqueles que não estão mascarados. A habilidade de mascarar os processadores permite a implementação de estruturas de controle do tipo *if...then...else* e também manter a sincronização.[26]

Os elementos de processamento são ligados entre si através de uma rede de interconexão, o que permite a troca de resultados de operações executadas. A estrutura da rede de interconexão pode ter a forma de uma grelha, um hipercubo, ou outras. O sub-capítulo 2.3 apresenta os tipos de redes de interconexão utilizadas nos computadores de alto desempenho.

Em geral, os processadores matriciais são construídos com múltiplos elementos de processamento integrados em um único *chip* e interconectados por meio de uma rede do tipo

[26] QUINN, Michael J. **Parallel computing** : theory and practice. 2. ed. Singapore : McGraw-Hill, 1994. p. 62.

grelha. Em um segundo nível, diversos *chips* são interconectados por uma rede de dimensão maior, em grelha ou em outra topologia.

Os processadores matriciais apresentam algumas limitações que restringem sua aplicabilidade. Segundo FERNÁNDES e AMORIM:

> "As principais limitações dos processadores matriciais residem no modo síncrono de operação e na relativa complexidade de programação, assim como na lenta comunicação entre processadores que não sejam vizinhos. [...] Os processadores matriciais são considerados computadores especializados e orientados para certas áreas de aplicação tais como processamento de imagem e reconhecimento de padrões, previsão meteorológica e programação matemática". [27]

A Tabela 1.3 apresenta alguns modelos representativos de processadores matriciais. São destacados o número de elementos de processamento, o tamanho da memória local de cada processador, a estrutura da rede de interconexão utilizada e a performance máxima.

Tabela 1.3 - Modelos de processadores matriciais.

Fabricante	Modelo	No. máx. de elementos de processamento	Cap. máx. de memória por processador	Estrutura da rede de interconexão	Pico de performance
MasPar Computer Corporation	MP-1	16.384 (32 por *chip*)	16 Kbytes	Grelha X-Net e um crossbar multiestágio	1.3 Gflops
Thinking Machines Corporation	CM-2	65.536 (16 por *chip*)	1 Mbits	Hipercubo de dimensão 12, onde cada vértice é formado por um *chip* com uma grelha 4 × 4	28 Gflops
Active Memory Technology	DAP600	4.096 (64 por *chip*)	1 Kbits	Grelha 64 × 64, onde em cada cruzamento há um *chip* com uma grelha 8 × 8	560 Mflops

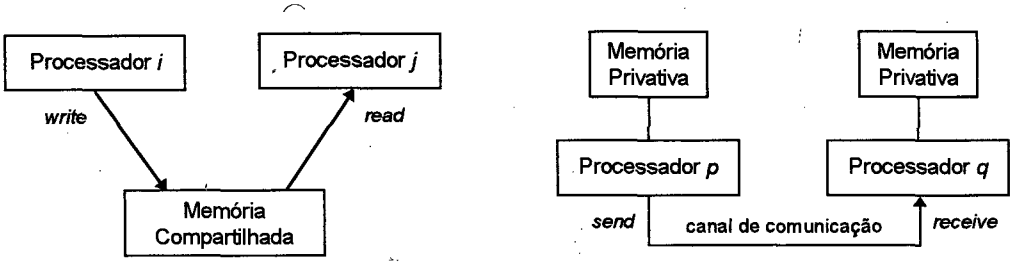
Fonte: HWANG, Kai. **Advanced computer architecture** : parallelism, scalability, programmability. 1993. p.33.

[27] FERNÁNDES, op. cit., p. 108.

1.2.4 - Multiprocessadores e multicomputadores

Os multiprocessadores e os multicomputadores são máquinas paralelas com múltiplos processadores que cooperam entre si para a resolução de um mesmo problema. Diferente dos processadores matriciais, onde uma unidade de controle difunde as instruções a todos os elementos do arranjo de processadores, nessas arquiteturas, cada processador tem a sua própria unidade de controle e executa uma parte do programa paralelo.

A diferença fundamental entre um multiprocessador e um multicomputador está no nível em que ocorre a cooperação entre os processadores da máquina. No primeiro, os processadores interagem entre si através do acesso à uma memória compartilhada, que pode estar fisicamente centralizada ou distribuída entre os processadores, conforme a Figura 1.9.a. No segundo, a interação se dá através da troca de mensagens e não existe compartilhamento de memória, cada processador possui uma memória local privativa, de acordo com a Figura 1.9.b. Por esses motivos, os multiprocessadores são considerados como arquiteturas fortemente acopladas, enquanto os multicomputadores são tidos como fracamente acoplados.



(a) multiprocessador (b) multicomputador

Figura 1.9 - Comunicação entre processadores em um: (a) multiprocessador; (b) multicomputador.

Outro aspecto diferencial reside no fato de que um multiprocessador pode ter no máximo algumas dezenas de processadores, enquanto que em um multicomputador podem existir milhares de processadores interconectados. Isso ocorre porque em um multiprocessador existe a concorrência por um recurso comum, a memória, e o aumento do número de processadores leva a uma saturação do uso desse recurso. Essas máquinas apresentam baixa extensibilidade, ou seja, a performance não aumenta na mesma proporção que o número de

processadores. Já os multicomputadores apresentam uma capacidade maior de extensibilidade, ao custo de uma gerência mais complexa de recursos.[28]

Em geral, cada processador tem associado algumas unidades tais como co-processador numérico, *cache*, interfaces de comunicação e, em alguns casos, memória local. A esse conjunto, dá-se o nome de "nodo" ou "nó".

1.2.4.1 - Multiprocessadores com memória centralizada

Um multiprocessador com memória centralizada consiste de um conjunto de processadores (P) e módulos de memória (M) interligados por meio de uma rede de interconexão. Os módulos de memória formam um espaço de endereçamento global compartilhado por todos os processadores. A estrutura geral desse tipo de arquitetura é mostrada na Figura 1.10.

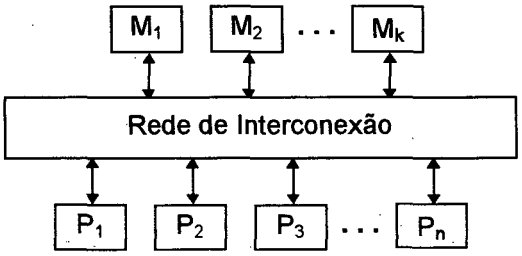


Figura 1.10 - Estrutura de um multiprocessador com memória compartilhada centralizada.

Nesse modelo de multiprocessador, o tempo de acesso a uma palavra na memória é igual para todos os processadores da máquina, ou seja, o acesso à memória é uniforme. Por isso, esse tipo de arquitetura é também chamado de UMA (*Uniform Memory Access*).

Um exemplo de multiprocessador UMA é o Symmetry S-81 da Sequent Computer Systems. Essa máquina possui até 30 nodos processadores conectados através de um barramento a um banco de memórias e a dispositivos de entrada e saída. Cada nodo é formado por uma CPU Intel 80386, um co-processador aritmético Intel 80387, um acelerador de ponto

[28] KITAJIMA, João Paulo F. W. **Programação paralela utilizando mensagens**. Porto Alegre : Instituto de Informática da UFRGS, 1995. p. 3.

flutuante Weitek WTL 1167 e uma *cache* de 64 Kbytes. A memória principal pode ter de 8 a 240 Mbytes.[29]

Um outro exemplo de máquina UMA é o modelo 900/VF da IBM. Esse multiprocessador apresenta 6 processadores ES/9000, com capacidade vetorial, conectados através de uma rede *crossbar* a canais de I/O e a uma memória principal de 1 Gbyte.[30]

1.2.4.2 - Multiprocessadores com memória distribuída

Nesse tipo de arquitetura, a memória compartilhada é fisicamente distribuída entre os processadores da máquina. Cada processador (P) tem a sua própria memória local (ML) e pode acessar a de um outro através da rede de interconexão. O conjunto de memórias locais forma o espaço de endereçamento global, compartilhado por todos os processadores da máquina. A Figura 1.11 apresenta a estrutura geral desse tipo de multiprocessador.

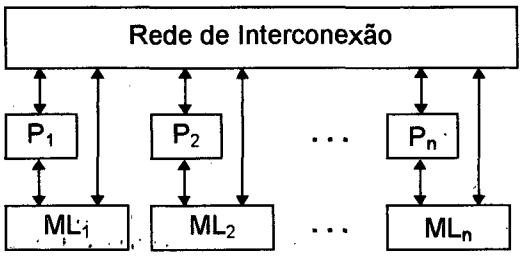


Figura 1.11 - Estrutura de um multiprocessador com memória compartilhada distribuída

Nesse modelo, o tempo de acesso a um endereço na memória depende da sua localização. Devido ao atraso introduzido pela rede de interconexão, o acesso a um dado é mais lento se ele está localizado na memória local de um outro processador. Pelo fato do acesso à memória ser não uniforme, esse tipo de arquitetura é também chamado de NUMA (*Non-Uniform Memory Access*).

Um exemplo de multiprocessador NUMA é o TC2000 da BBN System and Technologies. Essa máquina apresenta 128 nodos processadores interconectados por meio de uma rede *Butterfly*. Cada nodo é formado por uma CPU Motorola 88100, três *chips* Motorola 88200 para *cache* de dados e de instruções, uma memória local de 4 a 16 Mbytes e interfaces

[29] QUINN, op. cit., p. 68-69.

[30] HWANG, op. cit., p. 23.

para a rede *Butterfly* e para um barramento VME de entrada e saída. Essa interface VME oferece acesso à uma larga faixa de dispositivos de I/O, incluindo conversores A/D e D/A, dispositivos de armazenamento secundário e processadores gráficos.[31]

1.2.4.3 - Multicomputadores

Um multicomputador é constituído por múltiplos nodos associados por meio de uma rede de interconexão. Cada nodo é um computador autônomo e possui processador (P), memória local (ML) privativa e, em alguns casos, discos e/ou periféricos de I/O. Em um multicomputador não existe compartilhamento de memória e a interação entre os nodos ocorre por meio da troca de mensagens pela rede de interconexão. Como não ocorre acesso a dados em memória remota, esse tipo de arquitetura é também chamado de NORMA (*NO-Remote Memory Access*). A estrutura geral de um multicomputador é apresentado na Figura 1.12.

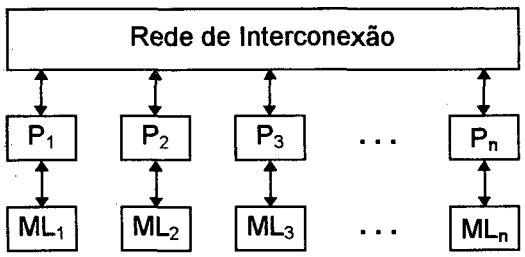


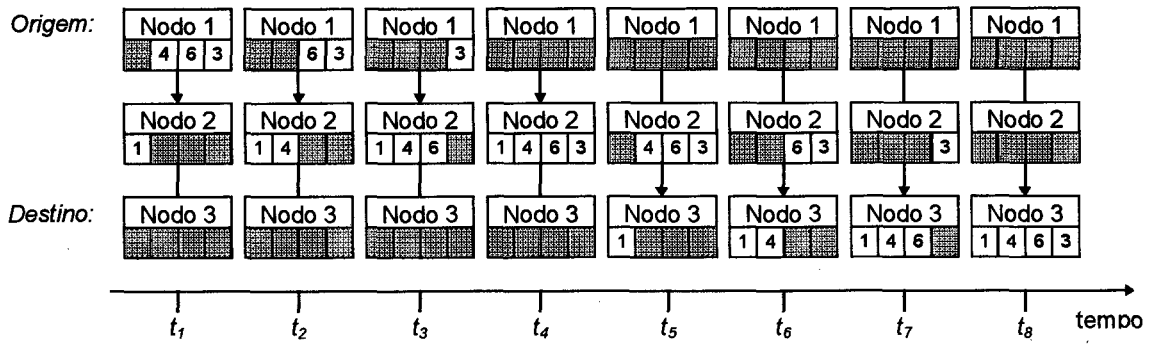
Figura 1.12 - Estrutura de um multicomputador.

Os multicomputadores podem ser agrupados em duas gerações, conforme o esquema de comunicação internodo utilizado: "armazena-e-repassa" ou "circuito chaveado".

Os multicomputadores da primeira geração baseiam-se no esquema "armazena-e-repassa" para implementar um mecanismo de chaveamento, ou roteamento, de mensagens por *software*. Uma mensagem enviada de um nodo origem para um nódo destino não adjacente na rede de interconexão passa por diversos nodos intermediários. Cada um desses nodos armazena a mensagem inteira em um *buffer* e, então, a repassa para o nodo seguinte no caminho da comunicação. Quanto maior for a distância entre os nodos origem e destino, maior será o tempo necessário para a transferência da mensagem. Além disso, esse mecanismo tem

[31] QUINN, op. cit., p.70-72.

desvantagens: o tamanho das mensagens é limitado pela capacidade do *buffer* e os nodos são interrompidos a cada comunicação.



Fonte: QUINN, Michael J. **Parallel computing** : theory and practice. 1994. p. 73.

Figura 1.13 - Transferência de uma mensagem entre dois nodos não adjacentes através de um mecanismo de roteamento por *software*.

Na Figura 1.13 é exemplificada a transferência de uma mensagem entre dois nodos não adjacentes. No exemplo, uma mensagem é transferida do nodo 1 para o nodo 3 através de um nodo intermediário. No instante t_1, o nodo 1 inicia o envio da mensagem para o nodo 2. Esse, após receber a mensagem completa, em t_4, reenvia os dados recebidos para o nodo 3. A comunicação entre os nodos é finalizada apenas no instante t_8.

Na segunda geração de multicomputadores, é utilizada a técnica de "circuito chaveado", através de um mecanismo de chaveamento de mensagens por *hardware*. Cada nodo tem associado um dispositivo de roteamento, chamado módulo de conexão direta. A transferência de mensagens entre nodos não adjacentes ocorre por meio de um circuito estabelecido pelos dispositivos de roteamento dos nodos intermediários. Sendo assim, a mensagem flui diretamente do nodo origem para o destino, sem interromper outros nodos e nem tampouco ser armazenada nestes. O referido mecanismo não é tão dependente da distância entre os nodos quanto o anterior.

Na Figura 1.14, é apresentada uma organização de multicomputador com rede de interconexão em grelha e chaveamento de mensagens por *hardware*. A comunicação entre dois nodos não adjacentes envolve apenas os módulos de conexão direta dos nodos intermediários.

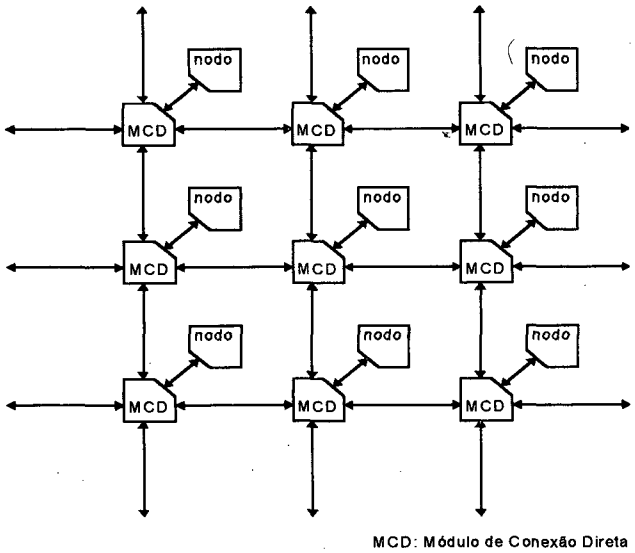


Figura 1.14 - Organização de um multicomputador com rede de interconexão em grelha e mecanismo de roteamento de mensagens por *hardware*.

Em geral, os multicomputadores utilizam redes de interconexão estáticas, como a grelha ou o hipercubo. Alguns modelos, entretanto, adotam redes dinâmicas, como o *crossbar*, para estabelecer a comunicação entre os processadores da máquina.

Um exemplo de multicomputador da primeira geração é o Intel iPSC/1. Esse multicomputador apresenta até 128 nodos interconectados por uma rede hipecúbica de dimensão 7. Cada nodo possui um processador i80286, um co-processador numérico i80287, uma memória local de 512 Kbytes e oito portas de I/O. Dessas, sete são utilizadas para formar o hipercubo e a oitava estabelece uma conexão Ethernet com o hospedeiro, um microprocessador Intel 310. Todas as operações de I/O são realizadas através desse hospedeiro.[32]

Entre os multicomputadores da segunda geração destacam-se o nCUBE/2, o Intel Paragon XP/S e o SuperNode1000 da Parsys. O nCUBE/2 contém até 8192 nodos interconectados por uma rede hipercúbica de dimensão 13. Cada nodo possui um processador com tecnologia CISC de 64 bits e uma memória local de até 64 Mbytes.[33]

---

[32] HWANG, op. cit., p. 368-369.

[33] QUINN, op. cit., p. 74-75.

O Paragon XP/S da Intel é um multicomputador com uma rede em grelha que interconecta três tipos de nodos: (i) de computação; (ii) de serviço; e (iii) de I/O. Os nodos de computação e de serviços são baseados no processador i860XP e têm de 16 a 128 Mbytes de memória local. Os nodos de I/O utilizam processadores i80386 e possuem de 16 a 64 Mbytes de memória. Cada nodo tem associado um *chip* que implementa o módulo de conexão direta para roteamento de mensagens. Os nodos de serviço são utilizados para diagnóstico do sistema e manipulação de interrupções. Já os nodos de I/O oferecem interfaces VME, SCSI, HIPPI e Ethernet.[34]

O SuperNode1000 é parte integrante de uma família de multicomputadores da Parsys. O modelo básico dessa família possui até 36 nodos, enquanto os modelos maiores podem atingir até 1024 nodos. Cada nodo possui um processador Transputer T800, até 4 Mbytes de memória local e quatro canais de comunicação. Os nodos são interconectados por intermédio de redes dinâmicas do tipo *crossbar* e de um barramento auxiliar de controle. Essa estrutura, com rede dinâmica e quatro canais por nodo, permite o estabelecimento de diversas topologias estáticas, conforme a aplicação a ser executada. Uma configuração básica pode ter até 36 nodos, distribuídos em três grupos: (i) 1 nodo de controle, responsável pela gerência da rede; (ii) até 3 nodos de serviço, que implementam servidores de memória e de disco; e (iii) 16 ou 32 nodos de trabalho, responsáveis pelo processamento efetivo da aplicação. Configurações maiores, como o SuperNode1000, são formadas pela conexão de configurações básicas por meio de *crossbars* suplementares.

1.2.5 - Classificação dos computadores

Existem diversas metodologias para classificação das arquiteturas de computadores. A mais utilizada na literatura é a taxonomia proposta por Flynn, a qual divide os computadores em quatro classes, conforme a multiplicidade dos fluxos de instruções e de dados. Outra metodologia, mais abrangente ainda, é a classificação de Duncan, que permite a inclusão de modelos de máquinas não considerados por Flynn.

---

[34] HWANG, op. cit., p. 372-373.

1.2.5.1 - A taxonomia de Flynn

Michael J. FLYNN propôs uma metodologia baseada em uma visão macroscópica da estrutura de um computador. Para tal, ele assumiu o dispositivo de I/O como sendo uma "caixa-preta", onde não há limitações de performance na execução dos programas de interesse ou, então, que as limitações de I/O são iguais para todas as configurações de memória. Ele considerou, também, a existência de um conjunto ideal de instruções com tempos de execução uniformes. FLYNN baseou a sua taxonomia na unicidade e na multiplicidade dos fluxos de instruções e de dados.[35] Disso derivaram quatro classes, a saber:

- SISD (*single instruction stream - single data stream*): onde existe um único fluxo de instruções que opera sobre um fluxo de dados igualmente único (Figura 1.15.a). Nessa classe estão incluídas as máquinas seqüenciais convencionais, onde existe somente uma unidade de controle que decodifica seqüencialmente as instruções, que por sua vez manipulam um conjunto de dados.

- SIMD (*single instruction stream - multiple data stream*): onde existe um único fluxo de instruções que opera sobre múltiplos fluxos de dados (Figura 1.15.b). Nessa categoria se enquadram as arquiteturas matriciais, que são computadores constituídos por um arranjo de processadores sincronizados pelo mesmo controlador. Todos os processadores executam a mesma instrução, mas cada um é capaz de buscar e manipular seu próprio dado.

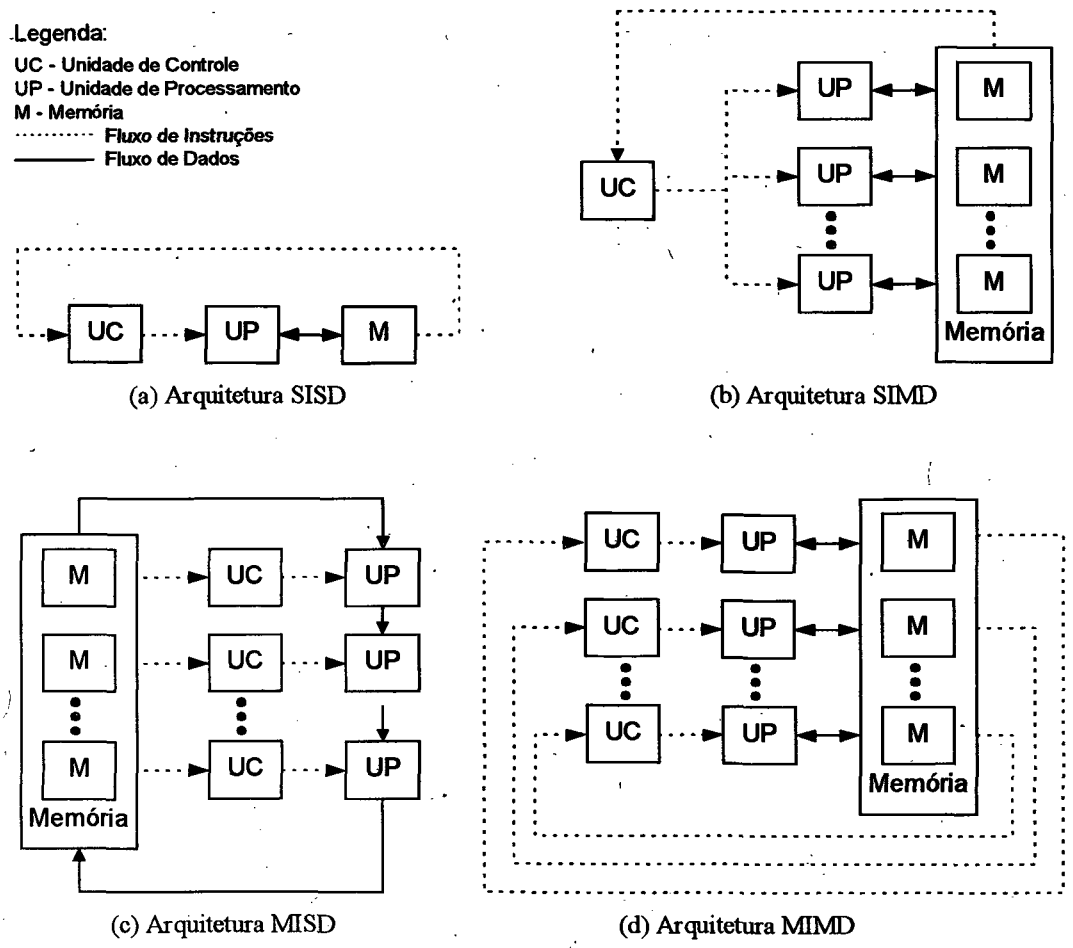
- MISD (*multiple instruction stream - single data stream*): onde existem múltiplos fluxos autônomos de instruções que operam sobre um fluxo de dados único (Figura 1.15.c). Há uma certa polêmica na literatura com relação a existência de alguma máquina MISD. Alguns autores afirmam que não há nenhum modelo de computador incluso nessa categoria. Segundo outros, as máquinas sistólicas são um exemplo de arquitetura MISD. Essas máquinas são computadores paralelos compostos por diversos processadores que operam em *pipelining* sobre o mesmo fluxo de dados. Cada processador modifica o dado

---

[35] FLYNN, Michael J. Some computer organizations and their effectiveness. **IEEE Transactions on Computers**, v. c-21, n. 9, p. 948-960, sep. 1972. p. 948.

antes de passá-lo para o próximo, o qual pode realizar uma operação diferente no dado.

- MIMD (*multiple instruction stream - multiple data stream*): onde existem múltiplos fluxos autônomos de instruções que operam sobre múltiplos fluxos de dados (Figura 1.15.d). Nessa classe está incluída a maioria dos sistemas multiprocessados, nos quais existem um conjunto de processadores que executam fluxos independentes de instruções sobre conjuntos de dados próprios.



Fonte: NAVAUX, Philippe. **Multiprocessadores** : organização e programação. p. 7-8.

Figura 1.15 - A taxonomia de Flynn.

1.2.5.2 - A taxonomia de Duncan

DUNCAN baseou-se na taxonomia de FLYNN para desenvolver uma metodologia de classificação mais abrangente, a qual permite a inclusão de arquiteturas paralelas não consideradas por FLYNN. Ele propôs três classes de computadores paralelos, a saber:[36]

- Síncronas: onde as instruções são executadas sincronamente a um relógio único e operam sobre um ou mais conjuntos de dados. Nessa categoria incluem-se os computadores matriciais SIMD; as máquinas sistólicas MISD; e os supercomputadores vetoriais, compostos por várias unidades em *pipeline* que operam concorrentemente.
- MIMD: idêntica a classe original definida por Flynn.
- Baseadas no paradigma MIMD: onde, além de existirem múltiplos fluxos de instruções que operam sobre múltiplos fluxos de dados, há, também, outros comportamentos particulares. Enquadram-se nessa categoria as máquinas MIMD/SIMD, que são computadores MIMD com partes que funcionam como SIMD; as arquiteturas baseadas em fluxos de dados, nas quais a disponibilidade dos dados determina a instrução a ser executada; e os computadores sistólicos assíncronos, conhecidos por máquinas *wavefront*.

1.3 - Redes de interconexão

Segundo Laxmi N. BHUYAN, "uma rede de interconexão é um conjunto de elementos de chaveamento e de ligação que permite a comunicação de dados entre processadores e memórias em um multiprocessador ou entre processadores em um multicomputador".[37]

A topologia de uma rede de interconexão pode ser estática ou dinâmica. Em uma rede estática os elementos são conectados por meio de ligações ponto-a-ponto fixas, que não mudam durante a execução do programa. Já uma rede dinâmica é formada por canais

---

[36] KITAJIMA, op. cit., p. 5.

[37] BHUYAN, Laxmi N. Interconnection networks for parallel and distributed processing. **Computer**, v. 20, n. 6, p. 9-12, june 1987. p. 9.

chaveados que são configurados dinamicamente, conforme a demanda do programa em execução.

De um modo geral, redes estáticas são usadas na interconexão de nodos em multicomputadores e redes dinâmicas na conexão de processadores aos módulos de memória em multiprocessadores. Além desses, os processadores matriciais também utilizam redes estáticas ou dinâmicas para interconectar seus elementos de processamento. Alguns modelos de multicomputador, como o SuperNode da Parsys, fazem uso de redes de interconexão dinâmicas, ao invés de estáticas.

1.3.1 - Propriedades das redes de interconexão

Uma rede de interconexão pode ser representada por um grafo, onde os nodos são os processadores da máquina e os arcos representam os caminhos de comunicação entre pares de processadores, conforme o exemplo da Figura 1.16.

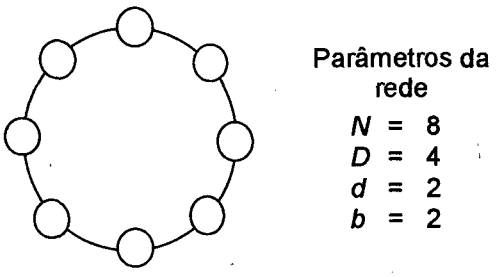


Figura 1.16 - Representação gráfica de uma rede de interconexão em anel.

A performance de uma rede de interconexão depende da sua funcionalidade, latência, largura de banda, complexidade de *hardware* e extensibilidade. A funcionalidade refere-se a como a rede suporta o roteamento de dados e a manipulação de interrupções, entre outros. A latência da rede é fornecida pelo maior atraso na transferência de uma unidade de mensagem, enquanto a largura de banda é definida pela taxa máxima de transferência em Mbyte/s. A complexidade de *hardware* tem reflexo direto no custo de implementação da rede, incluindo fios, chaves, conectores e interfaces. A extensibilidade, ou escalabilidade, refere-se à

capacidade da rede de aumentar a sua performance com o incremento modular dos recursos da máquina.[38]

Esses fatores podem ser avaliados a partir da análise de alguns parâmetros extraídos do grafo da rede, entre os quais destacam-se:

- tamanho da rede (N): é definido pelo número de nodos no grafo;
- diâmetro da rede (D): é determinado pelo número de arcos entre os dois nodos mais distantes na rede. O diâmetro da rede está associado ao tempo máximo para transmissão de uma mensagem entre dois nodos e deve ser o menor possível;
- grau do nodo (d): é obtido pelo número de arcos incidentes em um nodo. Esse parâmetro reflete o número de portas de I/O por nodo e, portanto, o seu custo. É desejável que seu valor seja pequeno e constante, de modo a garantir a extensibilidade do sistema;
- largura da bisseção (b): é o número de arcos que devem ser removidos a fim de dividir a rede em duas metades iguais. Como cada arco corresponde a um canal com um determinado número de fios, esse parâmetro fornece, juntamente com a capacidade de transferência por fio, uma indicação da máxima largura de banda da rede.

Um quinto parâmetro utilizado na avaliação de uma rede de interconexão é o comprimento máximo do arco. Idealmente, para assegurar a extensibilidade do sistema, o seu valor deve ser uma constante independente do tamanho da rede. Além disso, algumas propriedades importantes incluem a simetria da rede e a homogeneidade dos nodos, entre outros.

[38] HWANG, op. cit., p.80.

1.3.2 - Redes estáticas

Uma rede estática é caracterizada pela existência de ligações ponto-a-ponto fixas entre pares de nodos. Cada nodo é conectado a um pequeno número de nodos vizinhos. A comunicação entre nodos não adjacentes ocorre através do roteamento da mensagem pelos nodos intermediários ao caminho da comunicação.

Existem diversas estruturas de redes estáticas que são utilizadas tanto em multicomputadores como em processadores matriciais. Algumas das principais topologias encontradas na literatura são apresentadas nesta seção.[39]

1.3.2.1 - Grelha

Uma rede em grelha de dimensão k, conforme o exemplo da Figura 1.17, é formada por um arranjo k-dimensional de ordem n com tamanho $N = n^k$ nodos. O diâmetro da rede é dado por $k(n\text{-}1)$ e a largura de bisseção é igual a $n^{k\text{-}1}$. Essa rede apresenta uma assimetria dada pela não-uniformidade no número de ligações por nodo. O máximo grau do nodo é $2k$, sendo que os nodos externos têm graus menores.

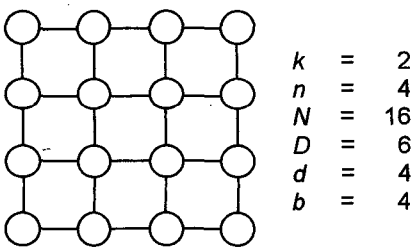


Figura 1.17 - Rede em grelha bidimensional 4 × 4.

As redes em grelha mais comumente utilizadas são as bidimensionais. Diversos processadores matriciais, como o Illiac IV, o MP-1 da MasPar e o DAP600 da AMT; e multicomputadores, como o Intel Paragon XP/S, utilizam esse tipo de rede ou alguma das suas variações.

[39] As relações mostradas a seguir, que fornecem os valores para os parâmetros da rede, foram originalmente apresentadas por QUINN (1994, p. 61) e HWANG (1993, p. 88).

1.3.2.2 - Toroidal

A rede toroidal é uma variante da grelha e é obtida pela interligação dos nodos externos das mesmas linhas e colunas, conforme a Figura 1.18. Essa topologia apresenta parâmetros melhores que os da grelha. O diâmetro da rede é menor, $D = n$, onde n é a ordem do arranjo, e a largura de bisseção é maior, $b = 2n^{k-1}$. Além disso, o grau do nodo é igual a $2k$ para todos os nodos, o que resulta em uma rede simétrica.

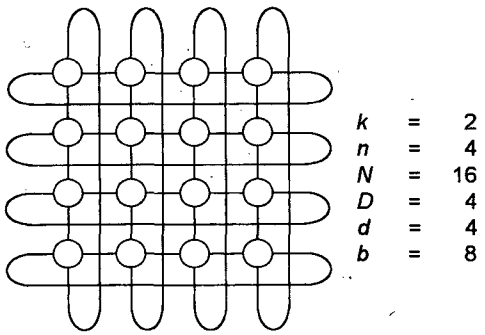


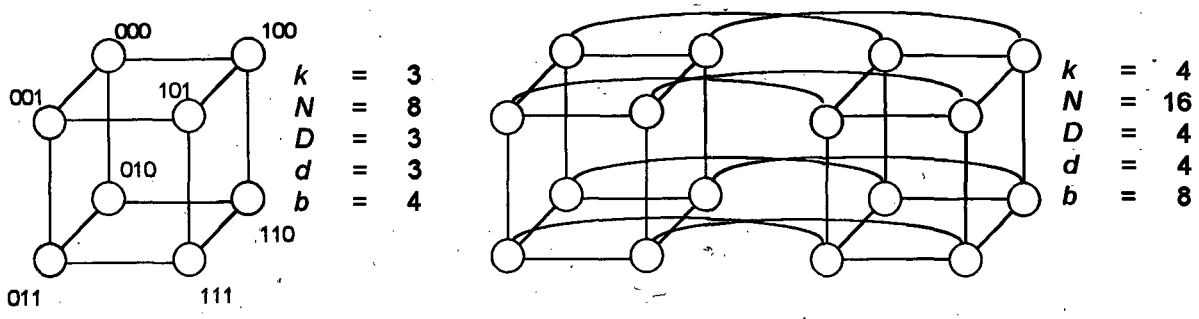
Figura 1.18 - Rede toroidal bidimensional.

A comparação da rede em grelha da Figura 1.17 com a rede toroidal da Figura 1.18 mostra a redução do diâmetro da rede de 6 para 4 e o aumento da largura de bisseção de 4 para 8. Isso leva a uma latência menor e a uma banda passante maior, o que resulta em um desempenho melhor.

1.3.2.3 - Hipercubo

Uma rede hipercúbica de dimensão k tem $N = 2^k$ nodos interconectados de forma que cada nodo é ligado a k nodos vizinhos, o que produz uma rede simétrica com grau de nodo $d = k$. A largura de bisseção é 2^{k-1} e o diâmetro da rede é igual a k. Apesar da simetria da rede, o grau do nodo cresce com o aumento da sua dimensão, o que dificulta a extensão da mesma.

Os nodos do hipercubo são numerados de 0 a 2^k-1. Dois nodos são adjacentes se seus números, em notação binária, diferem de apenas um *bit*. Na Figura 1.19 são apresentados dois exemplos de redes hipercúbicas com diferentes dimensões.



(a) 3-Cubo (b) 4-Cubo

Figura 1.19 - Dois exemplos de hipercubo: (a) 3-Cubo; e (b) 4-Cubo.

Entre os multicomputadores comerciais que utilizam redes hipercúbicas destacam-se o iPSC/1, com um hipercubo de dimensão 7, e o nCUBE/2, com uma rede hipercúbica de dimensão 13.

1.3.2.4 - Cubo conectado por ciclos - CCC

A rede cúbica conectada por ciclos, ou CCC, é uma variante do hipercubo, na qual cada vértice é formado por um ciclo de k nodos, onde k é a dimensão da rede. Dessa forma, para uma mesma dimensão, tem-se uma rede maior, $N = k2^k$, com o mesmo grau de nodo, $d = k$, do hipercubo. Entretanto, a largura de bisseção é a mesma, $b = 2^{k-1}$, e o diâmetro da rede é ainda maior, $D = 2k$, que no hipercubo. A Figura 1.20 apresenta uma rede CCC de dimensão 3.

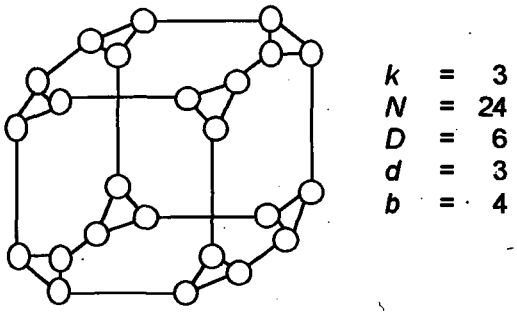


Figura 1.20 - 3-Cubo conectado por ciclos.

1.3.2.5 - Árvore binária

Uma rede em árvore binária é formada por uma estrutura hierárquica composta por n níveis, indexados de 0 a n-1, conforme a Figura 1.21. Cada nível tem 2^i nodos, onde i é o

índice do nível, e o tamanho da rede é 2^n-1. O grau do nodo máximo é igual a 3 para qualquer tamanho de rede, o que facilita a sua extensão. O diâmetro da rede é $D = 2(n\text{-}1)$ e a largura de bisseção é sempre igual a um. Uma árvore binária é referenciada pela sua profundidade, dada por n-1.

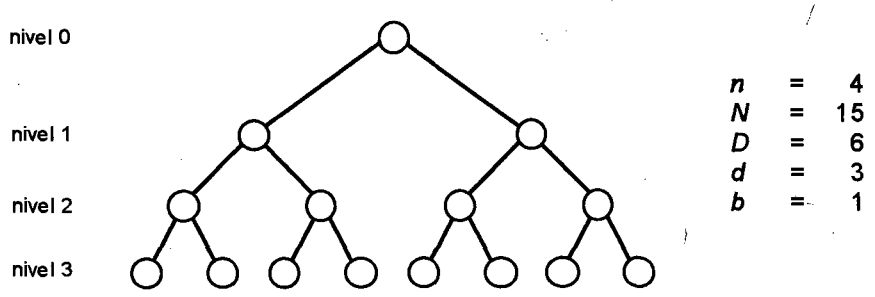


Figura 1.21 - Rede em árvore binária com profundidade 3.

1.3.2.6 - Sumário das redes estáticas

A Tabela 1.4 apresenta um resumo das redes estáticas descritas acima. São mostradas as relações que fornecem os valores dos parâmetros: tamanho, diâmetro da rede, grau do nodo e largura de bisseção; além da simetria.

Tabela 1.4 - Características das redes estáticas.

Nome da rede	Tamanho da rede (N)	Diâmetro da rede (D)	Grau do nodo (d)	Largura de bisseção (b)	Simetria	OBS.
Grelha	n^k	$k(n\text{-}1)$	$2k$	n^{k-1}	não	k: dimensão; n: ordem
Toroidal	n^k	n	$2k$	$2n^{k-1}$	sim	k: dimensão; n: ordem
Hibercubo	2^k	k	k	2^{k-1}	sim	k: dimensão
CCC	$k2^k$	$2k$	k	2^{k-1}	sim	k: dimensão
Árvore binária	2^n-1	$2(n\text{-}1)$	3	1	não	n: número de níveis

Das redes estáticas apresentadas, a árvore binária é a mais favorável à extensão, pois o grau do nodo é constante. Entretanto, a largura de bisseção é muito baixa e o diâmetro da rede é grande. O hipercubo tem largura de bisseção maior, mas o grau do nodo cresce com o tamanho da rede, o que dificulta a sua extensão. Entre as redes descritas, a toroidal é a que

apresenta as melhores relações, pois o grau do nodo é uniforme e constante para uma mesma dimensão, o diâmetro da é da mesma ordem do arranjo da rede e a largura de bisseção é satisfatória.

Além dessas, outras redes estáticas são encontradas na literatura, como por exemplo: o arranjo linear, o anel, o anel cordal, a rede completamente conectada, a pirâmide e a rede de mistura-e-troca (*shuffle-exchange*). Não é objetivo do presente trabalho apresentar um estudo exaustivo sobre tal assunto. Dessa forma, a descrição dos diferentes tipos de redes estáticas fica limitada ao apresentado até este ponto.

1.3.3 - Redes dinâmicas

As redes dinâmicas apresentam estrutura reconfigurável conforme a demanda do programa em execução e são mais utilizadas em aplicações de propósito geral, cujos padrões de comunicação são imprevisíveis. Os três tipos básicos de redes dinâmicas incluem: o barramento, as redes multiestágio e o *crossbar*[40], descritos a seguir.

1.3.3.1 - Barramento

Um barramento é formado por um conjunto de fios e conectores que estabelecem um meio de comunicação comum a todos os elementos por ele conectados, conforme a Figura 1.22. Esse tipo de rede constitui-se em um sistema de tempo compartilhado, onde apenas uma transferência fonte-destino pode ocorrer em um determinado momento. Múltiplas requisições competem pelo seu uso e devem ser atendidas uma após a outra.

A maior vantagem do barramento sobre os outros tipos de redes dinâmicas é o seu baixo custo. Entretanto, a banda passante é muito limitada, o que restringe o desempenho do sistema.

---

[40] O termo *crossbar* pode ser traduzido como "barramentos cruzados". Contudo, nesse texto, será utilizado o seu original em inglês.

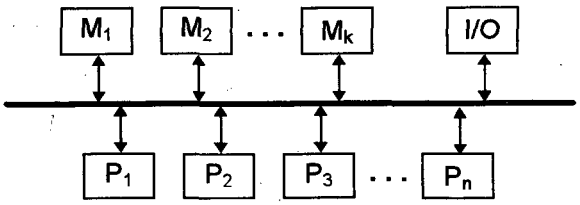


Figura 1.22 - Um multiprocessador com rede de interconexão em barramento.

Multiprocessadores que utilizam redes em barramento têm no máximo algumas dezenas de processadores, em geral, até 30. Contudo, o uso de múltiplos barramentos, ao invés de um barramento único, leva ao aumento da banda passante da rede e permite incrementar o número de processadores para até poucas centenas.[41]

Alguns exemplos de barramentos digitais utilizados nos sistemas multiprocessados incluem o VMEbus e o Futurebus[42]. Entre os modelos de multiprocessadores que baseiam seus sistemas de interconexão no barramento estão o Sequent Balance 8000 e o Encore Multimax.

1.3.3.2 - Redes multiestágio

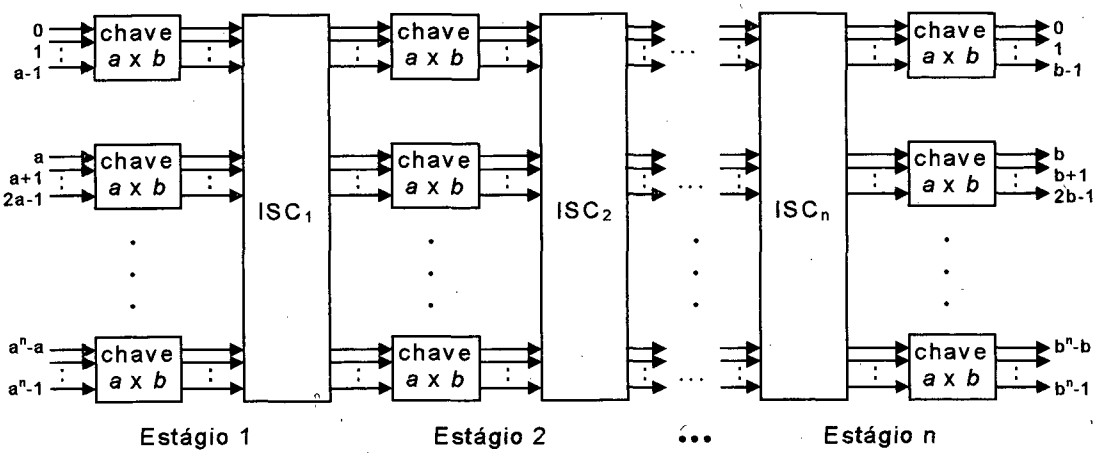
Uma rede multiestágio é formada por um conjunto de chaves $a \times b$ agrupadas em n estágios interconectados por meio de ligações fixas, conforme a Figura 1.23. As chaves são configuradas dinamicamente e permitem o estabelecimento de um caminho direto entre qualquer par entrada-saída. A estrutura das conexões interestágio (ISC - *Interstage Conection*) pode ser de diversos padrões, como: *perfect shuffle*, *butterfly*, *multiway shuffle*, *crossbar* e *cube connection*.[43] Além do tamanho das chaves e do padrão das conexões interestágio, uma rede multiestágio pode ser caracterizada pela existência ou não de um controle centralizado.

---

[41] MUDGE, Trevor N.; HAYES, John. P.; WINSOR, Donald. C. Multiple bus architecture. **Computer**, v. 20, n. 6, p. 42-48, june 1987. p. 42.

[42] GIACOMO, Joseph Di. **Digital bus**. New York : McGraw-Hill, 1990. p. 1.3, 7.1.

[43] HWANG, op. cit., p. 91.



Fonte: HWANG, Kai. **Advanced computer architecture** : parallelism, scalability, programmability. 1993. p. 91.

Figura 1.23 - Estrutura genérica de uma rede multiestágio.

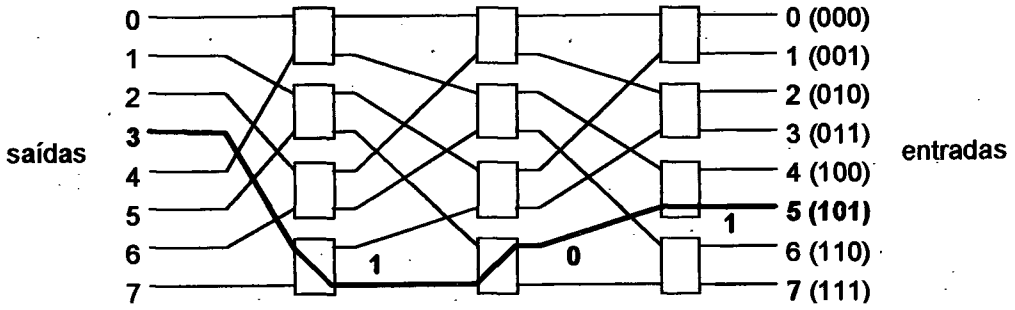
Uma rede multiestágio pode ser bloqueante, rearranjável ou não-bloqueante. No primeiro tipo, conexões simultâneas de mais de um par de nodos podem provocar conflitos no uso dos canais de comunicação da rede. No segundo, todas as conexões possíveis entre entradas e saídas podem ser realizadas por intermédio do rearranjo das conexões existentes. No terceiro e último tipo, todas as conexões possíveis são estabelecidas sem o bloqueio e sem a perturbação das outras conexões da rede.[44]

Entre os diversos modelos de redes multiestágio estão incluídas as redes Ômega, Delta, Baseline, SW Banyan e Benes. A Figura 1.24 apresenta uma rede Ômega 8 × 8 com três estágios interconectados segundo o padrão *perfect shuffle*, sendo que cada estágio é formado por quatro chaves 2 × 2. Essa rede tem controle descentralizado e o chaveamento é efetuado a partir de um rótulo gerado pelo processador requisitante. Tal rótulo define uma representação binária do caminho a ser estabelecido na rede. A conexão de uma chave no *i*-ésimo estágio é realizada pelo *i*-ésimo *bit* desse rótulo binário, contado a partir do *bit* mais significativo. Se esse *bit* é igual a 0, a entrada é conectada à saída superior da chave. Do contrário, se igual a 1, é estabelecida uma conexão com a saída inferior.[45] No exemplo da Figura 1.24, a entrada 3 é conectada à saída 5 através do rótulo binário 101.

---

[44] FENG, Tse-yun. A survey of interconnection networks. **Computer**, v. 14, n. 12, p. 12-27, dec. 1981. p. 29.

[45] BHUYAN, op. cit., p.10.



Fonte: BHUYAN, Laxmi N. Interconnection networks for parallel and distributed processing. **Computer**, v. 20, n. 6, p. 9-12, june 1987. p. 10.

Figura 1.24 - Uma rede Ômega 8 × 8.

Entre os computadores que utilizam redes de interconexão multiestágio estão incluídos o BBN TC2000 e o IBM RP-3.

1.3.3.3 - Redes *crossbar*

Uma rede *crossbar* assemelha-se a um comutador telefônico, onde diversos pontos de cruzamento estabelecem conexões entre entradas e saídas, conforme a Figura 1.25.

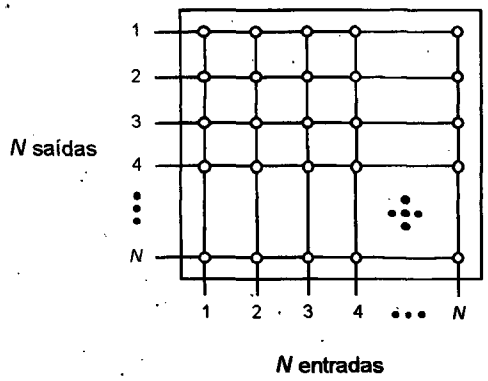


Figura 1.25 - Uma rede *crossbar* $N \times N$.

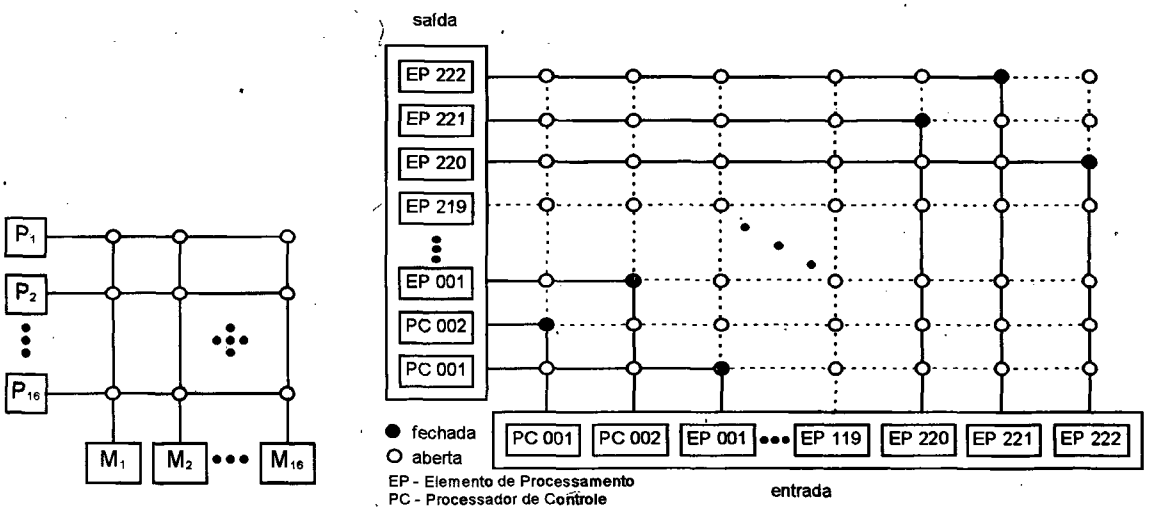
Esse tipo de rede é o que apresenta maiores largura de banda e capacidade de interconexão. Um *crossbar* normal permite a interconexão um-para-um de N entradas a N saídas sem contenção. Já *crossbar* generalizado possibilita, também, a conexão de uma entrada a mais de uma saída para realização de difusão.[46] O *crossbar* é o tipo de rede ideal para uso em

[46] SAWCHUCK, Alexander A. et alli. Optical crossbar networks. **Computer**, v. 20, n. 6, p. 50-60, june 1987. p. 50.

processamento paralelo, pois é reconfigurável e não bloqueante. Porém, seu custo de implementação é elevado e a complexidade de *hardware* cresce com o quadrado do número de processadores.

A rede *crossbar* 16 × 16 do multiprocessador C.mmp, mostrada na Figura 1.26.a, conecta 16 processadores PDP 11 a 16 módulos de memória, sendo que cada módulo pode ser acessado por um processador.

No multicomputador VPP500 da Fujitsu é utilizado um *crossbar* 224 × 224 para realizar conexões do tipo processador-a-processador, conforme a Figura 1.26.b. Além desses, outros exemplos de máquinas que utilizam redes de interconexão *crossbar* incluem o Cray Y-MP e o SuperNode da Parsys.



(a) Rede crossbar tipo processador-a-memória utilizada no multiprocessador C.mmp construído na Universidade Carnegie-Mellon (1972).

(b) Rede crossbar tipo processador-a-processador utilizada no processador paralelo vetorial VPP500 da Fujitsu (1992).

Fonte: HWANG, Kai. **Advanced computer architecture** : parallelism, scalability, programmability. 1993. p. 94.

Figura 1.26 - Dois exemplos de redes *crossbar* para interconexão: (a) processador-a-memória; (b) processador-a-processador.

Redes crossbar têm sido construídas através do uso de microeletrônica e de tecnologias óticas. *Crossbars* eletrônicos apresentam capacidades de reconfiguração[47] maiores que dos *crossbars* óticos, porém, a largura de banda da comunicação é menor.

1.3.3.4 - Sumário das redes dinâmicas

Na Tabela 1.5 são mostradas as relações que fornecem as características de latência mínima, largura de banda, complexidade de fios e chaves e capacidades de conectividade e roteamento das redes dinâmicas acima apresentadas.

Tabela 1.5- Características das redes dinâmicas.

Características	Barramento	Multiestágio	*Crossbar*
Latência mínima	Constante	$O(\log_k n)$	Constante
Largura de banda	de $O(w/n)$ a $O(w)$	de $O(w)$ a $O(n.w)$	de $O(w)$ a $O(n.w)$
Fios	$O(w)$	$O(n.w.\log_k n)$	$O(n^2.w)$
Chaves	$O(n)$	$O(n.\log_k n)$	$O(n^2)$
Conectividade e roteamento	Somente um-para-um	Algumas permutações e difusões	Todas as permutações
OBS.	n processadores conectados ao barramento de largura w *bits*	rede $n \times n$ com chaves $k \times k$ e largura de linha igual a w *bits*	rede $n \times n$ com largura de linha igual a w *bits*

Fonte: HWANG, Kai. **Advanced computer architecture** : parallelism, scalability, programmability. 1993. p. 95.

Dos três tipos de redes dinâmicas, o barramento é o que apresenta o menor custo de implementação. Entretanto, a largura de banda e a tolerância a falhas também são as menores. Em algumas máquinas são utilizados barramentos duais, ou múltiplos, para aumentar o desempenho e a confiabilidade do sistema.

Num extremo oposto ao barramento estão as redes *crossbar*, que apresentam as maiores larguras de banda e capacidades de roteamento e conectividade. Porém, o custo de implementação dessas redes é o mais elevado, aumentando com o quadrado do número de processadores. Em geral, as redes *crossbar* apresentam custos proibitivos para a construção de

[47] A capacidade de reconfiguração está relacionada, conforme Alexander A. SAWCHUCK, "ao tempo necessário para mudar completamente as configurações de todas as chaves da rede, de modo a estabelecer um padrão de configuração completamente diferente". (SAWCHUCK, 1987, p. 51)

grandes sistemas, embora *crossbars* volumosos possam ser implementados a partir do particionamento em unidades menores e mais baratas.

Entre os dois extremos estão as redes multiestágio que apresentam as melhores características de extensibilidade para construção modular. Comparadas com o *crossbar*, as redes multiestágio têm custo de implementação menor, porém a latência na comunicação aumenta com o número de estágios.

1.4 - Resumo

Nesse capítulo, foi efetuada uma revisão sobre as arquiteturas de computadores de alto desempenho. Tal estudo foi feito com o objetivo de situar o trabalho proposto em seu próprio contexto. Realizou-se um apanhado histórico do desenvolvimento dos computadores, desde a era mecânica até a última geração de computadores eletrônicos. Foram mostrados os principais modelos de máquinas de alto desempenho e, também, as redes de interconexão por eles utilizadas.

No capítulo seguinte, será descrito um projeto que visa o desenvolvimento de um ambiente completo para programação paralela, incluindo a construção de uma máquina paralela que servirá de plataforma física. Essa máquina consistirá em um multicomputador com rede de interconexão dinâmica do tipo *crossbar*.

CAPÍTULO 2

O PROJETO NÓ//

Introdução

O Projeto Nó// (lê-se nó paralelo) visa a construção de um ambiente completo para programação paralela, reunindo grupos de pesquisa das áreas de Arquitetura de Computadores, Sistemas Operacionais e Simulação. Esse projeto está sendo desenvolvido pela Universidade Federal de Santa Catarina em cooperação com a Universidade Federal do Rio Grande do Sul.

A proposta original do Projeto Nó// é apresentada por Thadeu B. CORSO.[48] Nessa proposta, são definidos um modelo de multicomputador com rede de interconexão dinâmica e um mecanismo de troca de mensagens que leva a uma grande redução da complexidade dos problemas associados à comunicação.

No presente capítulo, é feita uma descrição do Projeto Nó//, com ênfase aos aspectos relacionados ao modelo de máquina paralela.

2.1 - A arquitetura do multicomputador Nó//

O modelo geral de arquitetura proposto para o multicomputador Nó// consta de um conjunto de nodos interconectados por meio de uma rede dinâmica do tipo *crossbar*, conforme mostrado na Figura 2.1.

---

[48] CORSO, Thadeu B. **Ambiente para programação paralela em multicomputador.** Florianópolis : Relatório Técnico UFSC-CTC-INE Nº 1, 1993.

Nesse modelo, um nodo especializado, chamado nó de controle, tem a função de comandar o *crossbar*, reconfigurando-o dinamicamente, conforme a demanda do programa paralelo em execução nos demais nodos, chamados nós de trabalho.[49] Tal reconfiguração é efetuada com o auxílio de um mecanismo adicional de controle entre os nós de trabalho e o nó de controle. Esse mecanismo fornece meios para os primeiros enviarem as suas requisições de serviço para o último. Entre os serviços que o nó de controle disponibiliza aos nós de trabalho estão o estabelecimento e o cancelamento de uma conexão com outro nodo da máquina.

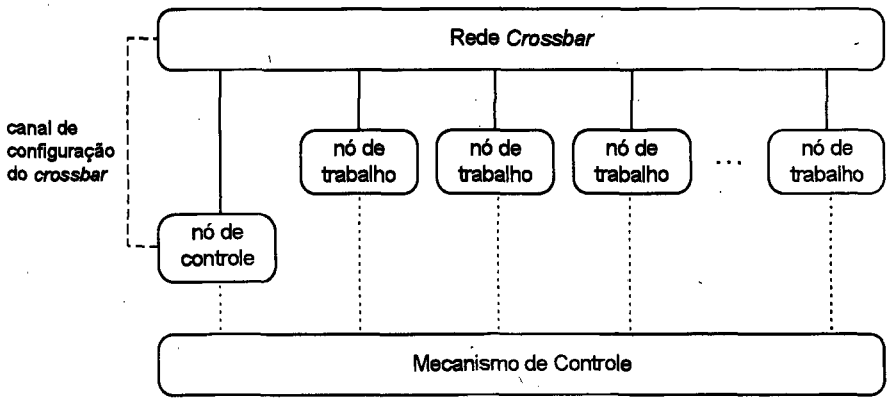


Figura 2.1 - Estrutura genérica do multicomputador Nó//.

Cada nodo possui um processador com memória privativa e um *hardware* para o suporte à comunicação, o qual estabelece as ligações com a rede de interconexão e com o mecanismo de controle, conforme a Figura 2.2.

Quanto maior o número de ligações (ou canais) por nodo, maior será a capacidade de reconfiguração da rede. Em uma máquina com quatro canais por nodo, por exemplo, é possível estabelecer qualquer topologia estática com grau do nodo até 4, como a grelha, o 4-cubo, a árvore binária, entre outros.

---

[49] Na terminologia apresentada no relatório original do Projeto Nó// (CORSO, 1993) os nodos processadores recebem a denominação de "nós". Neste capítulo, bem como nos subseqüentes, serão adotados os dois termos. Quando se tratar de uma referência geral, sem a determinação da função (de controle ou de trabalho), utilizar-se-á o termo "nodo". Do contrário, quando for especificada a função do processador, serão utilizados os termos "nó de controle" e "nó de trabalho".

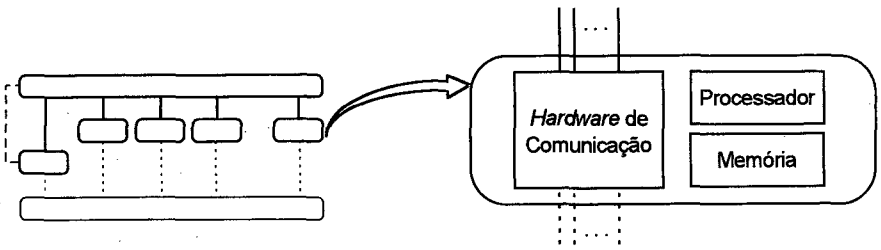


Figura 2.2 - Estrutura dos nós de trabalho.

O *hardware* de comunicação do nó de controle apresenta, além dessas ligações, um canal para a configuração do *crossbar*, pelo qual são estabelecidas e canceladas as conexões.

A rede de interconexão *crossbar*, com dimensão $N \times N$, permite que sejam estabelecidas $N/2$ ligações simultâneas e exclusivas entre pares de nodos. O estabelecimento de uma conexão é feito sem perturbar as outras conexões em uso.

O mecanismo adicional de controle pode ser implementado de diversas formas. Nas seções a seguir, são apresentadas duas propostas de arquitetura para o multicomputador Nó//, cada uma com um mecanismo de controle baseado em uma estrutura específica. Na primeira, existe um barramento compartilhado por todos os nodos, por meio do qual são transferidas pequenas mensagens de controle entre os nós de trabalho e o nó de controle. Na segunda, o mecanismo de controle baseia-se em um conjunto de linhas de interrupção entre os nós de trabalho e o nó de controle. A operação conjunta dessas linhas com conexões temporárias no *crossbar* permite a troca de mensagens de controle entre os nodos.

2.1.1 - Arquitetura baseada em um barramento compartilhado

Na proposta original do Projeto Nó//, é apresentada uma arquitetura de multicomputador com um mecanismo adicional de controle baseado em um barramento compartilhado por todos os nodos da máquina, chamado barramento de serviço, conforme a Figura 2.3.

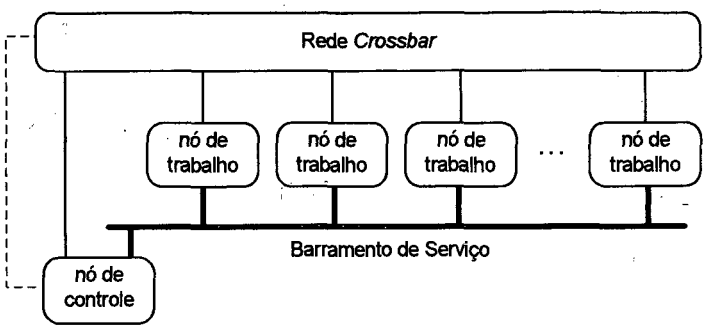


Figura 2.3 - Arquitetura baseada em um barramento compartilhado.

Através do barramento de serviço são transferidas mensagens de controle entre os nós de trabalho e o nó de controle, as quais transportam os comandos e as requisições necessários ao estabelecimento de um protocolo para a reconfiguração da rede de interconexão. Os comandos são enviados pelo nó de controle, enquanto as requisições são emitidas pelos nós de trabalho.

Durante o funcionamento normal da máquina, o nó de controle executa um ciclo de *polling* sobre o barramento de serviço. Ele endereça seqüencialmente cada um dos nós de trabalho, emitindo-lhes um comando de permissão. Para cada comando de permissão enviado é esperado o retorno da requisição de serviço do nó de trabalho endereçado.

Cada nó de trabalho, ao receber um comando de permissão, pode retornar a requisição correspondente ao serviço desejado - conexão ou desconexão - ou com uma requisição nula, caso nenhum tipo de serviço seja necessário. O atendimento de um serviço solicitado é confirmado pelo envio de um comando de reconhecimento ao nó de trabalho requisitante.

Esse protocolo é representado na Figura 2.4, onde o nó de controle envia um comando de permissão aos nós de trabalho da máquina e recebe a requisição de serviço de cada nodo endereçado. No exemplo, o primeiro nodo requisita o atendimento de um serviço de conexão com um outro nodo, o segundo não deseja qualquer tipo de serviço e o último solicita uma desconexão. Por fim, o nó de controle envia um comando de reconhecimento aos nodos requisitantes, pelos quais confirma o atendimento dos serviços solicitados.

Esse mecanismo tem a vantagem de possibilitar o aumento do número de nós de trabalho sem qualquer alteração no *hardware* do nó de controle. Entretanto, o *overhead*

associado ao estabelecimento, ou cancelamento, de uma conexão cresce com o número de nodos, pois cada nó de trabalho deve esperar a sua vez no ciclo de *polling* para enviar a sua requisição de serviço.

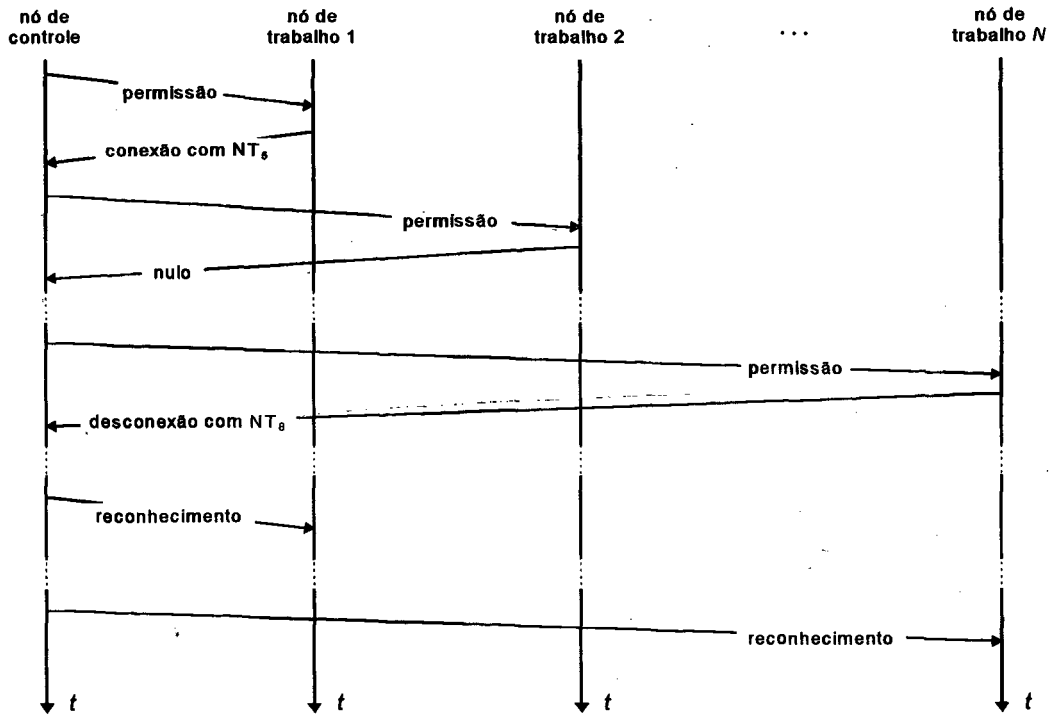


Figura 2.4 - Protocolo de reconfiguração para o mecanismo baseado em um barramento compartilhado.

2.1.2 - Arquitetura baseada em linhas de interrupção

Com o objetivo de reduzir o *overhead* associado ao atendimento de um serviço pelo nó de controle, foi proposta uma segunda arquitetura para o multicomputador Nó//, mostrada na Figura 2.5.

Nessa arquitetura, o mecanismo de controle baseia-se em um conjunto de linhas de interrupção, que, operando em conjunção com conexões temporárias no *crossbar*, permite estabelecer um protocolo eficiente para a troca de mensagens de controle entre os nodos da máquina e a subseqüente reconfiguração da rede de interconexão.

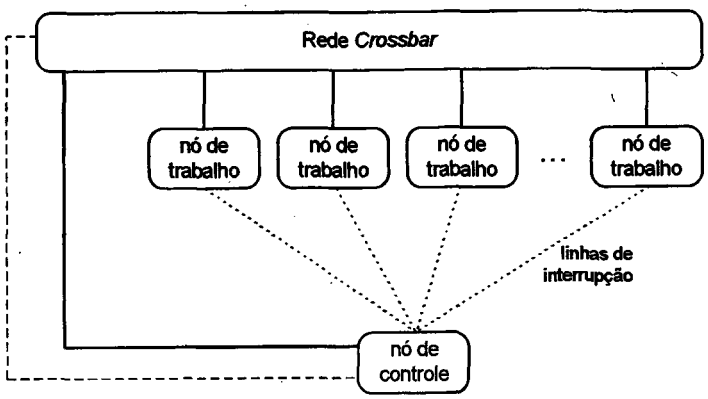


Figura 2.5 - Arquitetura baseada em linhas de interrupção.

Conforme mostra a Figura 2.6, cada nó de trabalho apresenta um par de linhas de interrupção associadas ao nó de controle. Essas linhas permitem ao primeiro interromper o segundo e vice-versa.

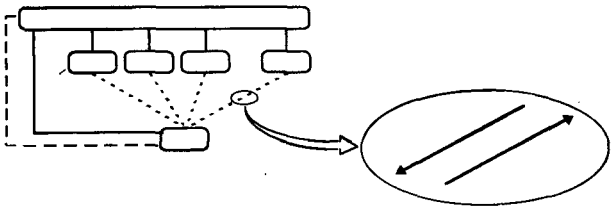


Figura 2.6 - Linhas de interrupção entre um nó de trabalho e o nó de controle.

Durante o funcionamento normal da máquina, representado na Figura 2.7, quando um nó de trabalho deseja o atendimento de um serviço, ele interrompe o nó de controle, enviando-lhe um sinal pela linha de interrupção apropriada. O nó de controle, por sua vez, atende a interrupção e estabelece uma conexão no *crossbar* com o nodo requisitante. Uma vez estabelecida a conexão, o nó de controle envia um comando de permissão para o nó de trabalho que gerou a interrupção, o qual retorna com a requisição para o serviço desejado. Por fim, o nó de controle cancela a conexão utilizada, busca atender o serviço solicitado e, após efetuá-lo, interrompe o nó de trabalho requisitante, para informa-lo do atendimento do serviço.

Nesse mecanismo, o *overhead* associado ao atendimento de um serviço é menor que aquele visto na seção 2.1.1, pois os nodos de trabalho não precisam esperar o recebimento de uma permissão para enviar as suas requisições. Entretanto, a complexidade do *hardware*

aumenta com o número de nodos, já que o nó de controle deve apresentar um canal para as linhas de interrupção de cada nó de trabalho.

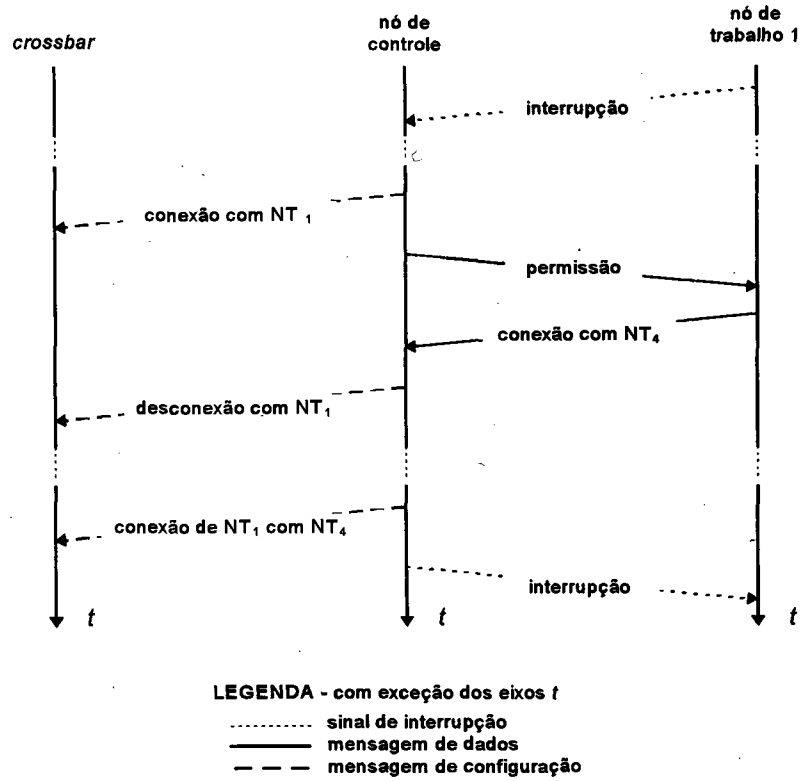
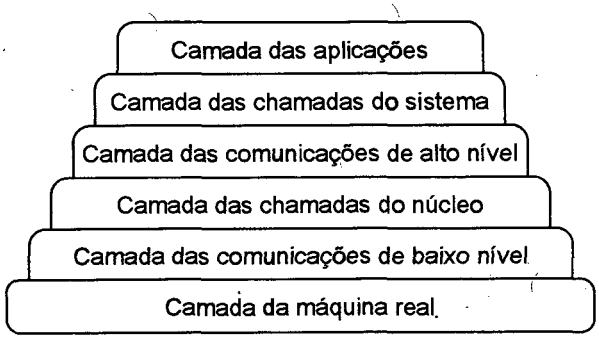


Figura 2.7 - Protocolo de reconfiguração para o mecanismo baseado em linhas de interrupção.

2.2 - O ambiente de programação paralela

O ambiente de programação paralela Nó// é estruturado segundo uma hierarquia de camadas, onde cada camada intermediária provê serviços à camada imediatamente superior e, para realizá-los, faz uso dos serviços da camada imediatamente inferior. A interface entre duas camadas subseqüentes é estabelecida por operadores que representam os serviços prestados pela camada inferior e esperados pela camada superior.[50] Na Figura 2.8, é apresentada a estrutura em camadas do ambiente de programação Nó//.

[50] CORSO, op. cit., p. 13-14.



Fonte: CORSO, Thadeu B. Ambiente para programação paralela em multicomputador. 1993. p. 14.

Figura 2.8 - As camadas do ambiente de programação paralela.

Conforme Thadeu B. CORSO[51], o papel de cada uma das camadas do ambiente de programação paralela pode ser assim resumido:

- Camada das aplicações: é representada pelos programas escritos sobre uma interface de programação Unix, estendida com serviços específicos do mecanismo de troca de mensagens. Este mecanismo baseia-se no modelo das redes de processos comunicantes[52] para criar um ambiente para programação paralela;
- Camada das chamadas do sistema: essa camada oferece, à camada das aplicações, serviços equivalentes aos oferecidos pelo sistema Unix, acrescidos de serviços de comunicação do modelo das redes de processos comunicantes;
- Camada das comunicações de alto nível: essa camada provê serviços de comunicação para a troca de mensagens volumosas através de canais no *crossbar*;
- Camada das chamadas do núcleo: entre os serviços oferecidos por essa camada estão a conexão e desconexão de canais entre nós que desejam comunicar-se pela rede de interconexão dinâmica;
- Camada das aplicações de baixo nível: essa camada fornece serviços que auxiliam a conexão e desconexão de canais entre os nodos da máquina através

[51] id., p. 14-19.

[52] HOARE, C. A. R. **Communicating sequential process.** Prentice-Hall, 1985.

da troca de mensagens sobre o mecanismo de controle utilizado, segundo o seu próprio protocolo;

- Camada da máquina real: essa camada é representada pelos recursos físicos disponibilizados pelo modelo de multicomputador adotado.

2.3 - Resumo

Nesse capítulo foi apresentado o Projeto Nó//, que visa a construção de um ambiente completo para programação paralela. Foi dada ênfase aos aspectos relacionados ao modelo de máquina a ser utilizado, através da descrição de duas propostas de arquitetura para o multicomputador Nó//. Apresentou-se, também, uma visão geral do ambiente de programação paralela; onde foi mostrado, de forma resumida, o papel de cada uma das camadas que compõe esse ambiente.

O modelo de ambiente de programação estabelecido no Projeto Nó// permite que a rede de interconexão seja reconfigurada dinamicamente conforme a demanda do programa paralelo em execução. Um exemplo de arquitetura de multicomputador, semelhante ao modelo apresentado na seção 2.1, com rede de interconexão dinâmica e um barramento auxiliar de controle, é o SupeNode. Entretanto, no modelo de execução dessa máquina não é permitida a reconfiguração dinâmica da rede de interconexão. A rede é configurada antes da carga do programa paralelo e mantém-se inalterada até o término da sua execução.

CAPÍTULO 3

PROJETO DO NÍVEL FÍSICO DE COMUNICAÇÃO

Introdução

Este capítulo apresenta os aspectos gerais estabelecidos para o projeto do sistema de comunicação do multicomputador Nó//. São descritos o protocolo de comunicação de nível físico, a metodologia de projeto de *hardware* e o padrão de barramento de interface com o processador. Além disso, é mostrado um estudo comparativo dos dois modelos de arquitetura propostos para o multicomputador Nó// e a escolha do modelo a ser utilizado na construção do primeiro protótipo da máquina.

3.1 - Protocolo de comunicação de nível físico

Como protocolo de comunicação de nível físico foi escolhido o protocolo de ligação INMOS (também chamado protocolo Transputer-Link). Esse protocolo estabelece uma comunicação serial, assíncrona, *full-duplex* e com controle de fluxo *byte*-a-*byte*. Cada *byte* é transmitido sob a forma de um pacote de dado e o seu recebimento é confirmado por meio de um pacote de reconhecimento, conforme a Figura 3.1.

(a) Pacote de dado (b) Pacote de reconhecimento

Figura 3.1 - Pacotes de (a) dado e (b) reconhecimento.

O pacote de dado é formado por: (i) dois *bits* em nível alto que indicam o início do pacote (*start bit*); (ii) oito *bits* de dado, referentes ao *byte* de dado e transferidos do menos

significativo para o mais significativo; e (iii) um *bit* em nível baixo que indica o fim do pacote (*stop bit*). Já o pacote de reconhecimento é formado por um *bit* em nível alto (*start bit*) e outro em nível baixo (*stop bit*). Cada pacote de dado transmitido tem seu recebimento confirmado por meio de um pacote de reconhecimento.

Esse protocolo é definido pela INMOS, sendo utilizado para prover a comunicação entre os seus produtos (como, por exemplo, o *crossbar* IMS C004[53] e o processador IMS T800[54]) e desses com o meio externo.

A implementação do protocolo de ligação INMOS é feita através do uso do adaptador de ligação IMS C011[55] (*Link Adaptor*), conforme descrito a seguir.

3.1.1 - O adaptador de ligação IMS C011

Esse componente consiste em um sistema de interconexão de alta velocidade que fornece comunicação *full-duplex* com microprocessadores e periféricos. Ele realiza a conversão de um canal de ligação serial bidirecional em um ou dois barramentos paralelos de 8 *bits*, de acordo com o modo de operação adotado. A transferência de dados é feita a uma velocidade de 10 ou 20 Mbits/s, sendo que a recepção é realizada de forma assíncrona, o que permite uma comunicação independente da fase do relógio.

O adaptador de ligação pode operar em dois modos. No primeiro, O IMS C011 realiza a conversão entre um canal de ligação serial bidirecional e dois barramentos paralelos unidirecionais de 8 *bits*, um de entrada e outro de saída. Oferece, ainda, linhas para o controle de transferência de dados. No segundo modo, o IMS C011 provê uma interface entre um canal de ligação serial e um barramento bidirecional de 8 *bits*. Apresenta, também, registradores de entrada e saída e duas linhas de interrupção, uma para indicar a disponibilidade de um dado na entrada e outra para informar que o *buffer* de saída encontra-se vazio.

No projeto dos adaptadores de comunicação dos nodos do multicomputador Nó//, o IMS C011 é utilizado para implementar os canais de ligação com o *crossbar*, sendo

---

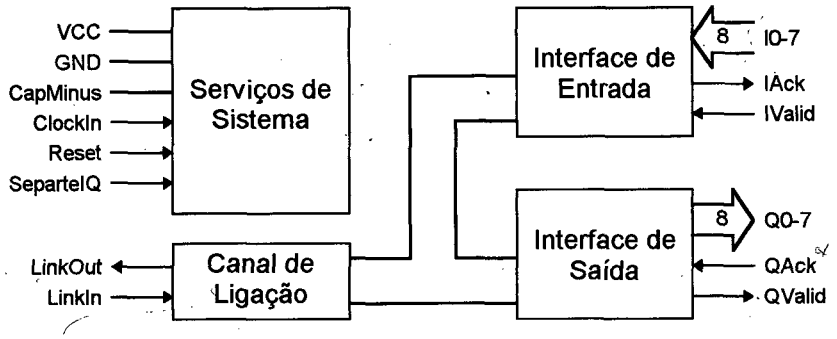
[53] INMOS. IMS C004 programmable link switch. In: **INMOS Engineering Data**, 1989. p. 479-501.

[54] INMOS. **The transputer application notebook : architecture and software.** Melksham : Redwood Press Limited, 1989.

[55] INMOS. IMS C011 : link adaptor. In: **INMOS Engineering Data**, 1988. p. 389- 412.

configurado para funcionar no primeiro modo de operação. Por esse motivo, os parágrafos a seguir restringem-se à descrição do funcionamento do adaptador de ligação nesse modo.

No modo 1 de operação, o IMS C011 é organizado conforme a Figura 3.2, com interfaces de entrada e de saída, um canal de ligação serial e os serviços de sistema necessários à lógica de partida e manutenção do componente. A Tabela 3.1 descreve cada uma das linhas mostradas nessa figura.



Fonte: INMOS. IMS C011 : link adaptor. In: **INMOS: Engineering Data**, 1989. p. 390.

Figura 3.2 - O IMS C011 no modo 1 de operação.

Tabela 3.1 - Descrição da pinagem do IMS C011.

Pino	Entrada/Saída	Função
VCC, GND		fonte de alimentação
CapMinus		capacitor externo para alimentar o relógio interno
ClockIn	entrada	relógio de entrada
Reset	entrada	reinicialização do sistema
SeparateIQ	entrada	seleciona a velocidade de transferência de dados (no Modo I)
LinkIn	entrada	entrada do canal de ligação serial
LinkOut	saída	saída do canal de ligação serial
I0-7	entrada	barramento paralelo de entrada (8 *bits*)
IValid	entrada	indica dado válido em I0-7
IAck	saída	reconhece recebimento do dado em I0-7
Q0-7	saída	barramento paralelo de saída (8 *bits*)
QValid	saída	indica dado válido em Q0-7
QAck	entrada	reconhece leitura do dado em Q0-7

Fonte: INMOS. IMS C011 : link adaptor. In: **INMOS: Engineering Data**, 1989. p. 391.

3.1.1.1 - Serviços de sistema

Para um perfeito funcionamento do IMS C011, alguns requisitos, relativos aos serviços de sistema, devem ser atendidos:

- *VCC* e *GND*: a fonte de alimentação deve ser desacoplada do circuito integrado por um capacitor de baixa indutância com capacitância igual a 100 nF (por exemplo, de cerâmica) entre os pinos *VCC* (= 5V) e *GND*;
- *CapPlus* e *CapMinus*: a alimentação para o relógio interno requer um capacitor de desacoplamento entre os pinos *CapPlus* e *CapMinus*. Esse capacitor deve ter resistência e indutância baixas e uma capacitância igual a 1μF. Recomenda-se o uso de um capacitor de cerâmica com impedância menor que 3 Ohms entre 100 KHz e 10 MHz;
- *ClockIn*: o sinal de relógio deve ter uma freqüência de 5 MHz (T = 200 ns) e ser derivado de um cristal oscilador;
- *SeparateIQ*: seleciona velocidade de 10 Mbits/s, se conectado ao VCC, e de 20 Mbits/s, se conectado ao *ClockIn*.
- *Reset*: um pulso de reinicialização com, no mínimo, 8 *ClockIn* (1.6 ms) leva o IMS C011 para o seguinte estado: (i) a saída *LinkOut* é mantida baixa; (ii) as saídas de controle (*IAck* e *QValid*) são mantidas baixas; e (iii) o estado de *Q0-7* torna-se indefinido. Além disso, a linha *LinkIn* deve ser mantida em nível baixo durante o pulso de reinicialização, exigindo um circuito como o da Figura 3.3.

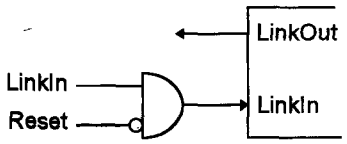


Figura 3.3 - Circuito para amarrar a linha *LinkIn* ao *Reset*.

3.1.1.2 - Canal de ligação serial

O canal de ligação é formado por duas linhas seriais unidirecionais, uma de entrada e outra de saída. Ele não é sincronizado com o sinal de relógio *ClockIn*, sendo insensível à

fase. Desse modo, dois canais com relógios independentes podem ser interligados, desde que os relógios sejam nominalmente idênticos e estejam dentro das especificações.

Para distâncias menores que 300 mm, a conexão entre dois canais de ligação pode ser feita diretamente. Para distâncias maiores, devem ser utilizados circuitos casadores de impedância ou *buffers*. A Figura 3.4 ilustra uma conexão direta entre dois canais de ligação.

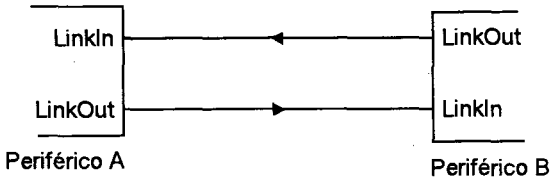


Figura 3.4 - Conexão direta entre dois canais de ligação IMS C011.

3.1.1.3 - Interface de entrada

A interface de entrada apresenta um barramento paralelo de 8 *bits* (*I0-7*) e duas linhas para o controle do fluxo de transmissão de dados (*IValid* e *IAck*). A linha *IValid* serve para o periférico transmissor indicar ao IMS C011 que existe, nas linhas *I0-7*, um dado válido pronto para ser transmitido. Já a linha *IAck*, quando em nível alto, indicá que o periférico receptor recebeu o pacote de dado e retornou com o pacote de reconhecimento, estando pronto para receber um novo dado.

A seguir é descrito o procedimento para a transmissão de um dado em um periférico através do canal de ligação do IMS C011:

- dado em *I0-7*: o periférico coloca um *byte* de dado no barramento de entrada *I0-7*. Após um tempo mínimo de estabilização de 5 ns, leva *IValid* para o nível alto, indicando ao IMS C011 que há um *byte* válido no barramento de entrada e que esse *byte* está pronto para ser transmitido. (Obs: O dado deve ser mantido no barramento de entrada, bem como a linha *IValid* deve ficar em nível alto, até que o IMS C011 leve a linha *IAck* para o nível alto);
- transmissão do dado: após *IValid* ter sido levado para o nível alto, o IMS C011 demora de 0,8 a 2 *bits* para iniciar a transmissão do pacote de dado pela linha serial de saída (para uma velocidade de 10 Mbits/s, 1 *bit* = 100 ns);

- reconhecimento do dado: quando o *start bit* do pacote de reconhecimento é recebido na linha de entrada serial, o IMS C011 demora até 3 *bits* para levar a linha *IAck* para o nível alto. (Obs: o periférico deve retornar com a linha *IValid* para o nível baixo, finalizando a transmissão do dado).
- fim da transmissão: a transmissão do dado se completa quando o IMS C011 retorna com o sinal *IAck* para o nível baixo. (Obs: nenhum dado novo pode ser colocado no barramento *I0-7* até a descida da linha *IAck*).

3.1.1.4 - Interface de saída

A interface de saída apresenta um barramento paralelo de 8 *bits* (*Q0-7*) e duas linhas para o controle do fluxo de recepção de dados (*QValid* e *QAck*). A linha *QValid* indica ao periférico receptor que existe, nas linhas *Q0-7*, um novo dado recebido e pronto para leitura. A linha *QAck* permite ao periférico indicar que a leitura do dado em *Q0-7* foi realizada e que ele está pronto para receber um novo dado.

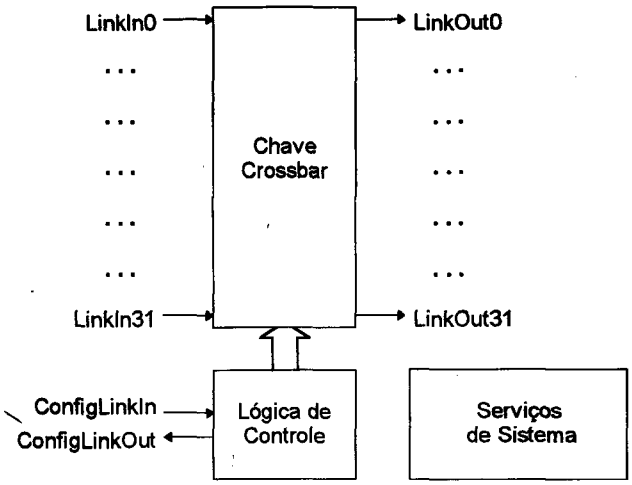
A seguir é descrito o procedimento para a recepção de um dado por um periférico através do canal de ligação do IMS C011:

- recepção do dado: após receber o *start bit* do pacote de dado, o IMS C011 demora, no mínimo, 11.5 *bits* para receber o pacote completo e apresentar o *byte* de dado no barramento de saída.
- dado em *Q0-7*: após apresentar o dado recebido no barramento de saída, o IMS C011 demora, no mínimo, 15 ns para levar *QValid* para o nível alto, indicando a existência de um novo dado válido em *Q0-7*;
- reconhecimento do dado: o periférico, após ler o dado no barramento de saída, leva *QAck* ao nível alto. O IMS C011 demora de 0.8 a 2 bits para iniciar a transmissão do pacote de reconhecimento;
- fim da recepção: após a subida de *QAck*, o IMS C011 demora, no mínimo, 1.8 bits para retornar *QValid* para o nível baixo, finalizando a recepção do dado. (OBS: o periférico deve retornar *QAck* para o nível baixo imediatamente após a descida de *QValid*).

3.1.2 - O *crossbar* IMS C004

Conforme definido no Projeto Nó//, o modelo de *crossbar* a ser utilizado para prover a interconexão entre os nodos do primeiro protótipo é o IMS C004 da INMOS. Esse componente apresenta 32 linhas de entrada e outras 32 de saída, as quais podem ser comutadas de modo a estabelecer até 16 ligações exclusivas e simultâneas entre pares de nós.

O *crossbar* IMS C004 é organizado conforme a Figura 3.5. Ele apresenta 32 pares de linhas de entrada e saída para os canais de ligação serial, um par de linhas de entrada e saída para a configuração das conexões e os serviços de sistema necessários à lógica de partida e manutenção do mesmo.



INMOS. IMS C004 programmable link switch. In: **INMOS Engineering Data**, 1989. p. 479.

Figura 3.5 - O *crossbar* IMS C004.

A Tabela 3.2 apresenta as mensagens que devem ser enviadas através do canal de configuração do *crossbar* de modo a estabelecer ou cancelar conexões, entre outros.

Tabela 3.2 - Mensagens de configuração do *crossbar* IMS C004.

Mensagem de configuração	Função
[0][entrada *i*][saída *j*]	Conecta a entrada *i* à saída *j*.
[1][ligação *p*][ligação *q*]	Conecta o canal de ligação *p* ao canal de ligação *q*, através da conexão da entrada do primeiro canal à saída do segundo e da entrada desse à saída daquele.
[2][saída j]	Pergunta à qual entrada a saída *j* está conectada. O IMS C011 responde com a entrada, sendo que o *bit* mais significativo do *byte* de resposta indica se a saída está conectada (nível alto) ou desconectada (nível baixo).
[3]	Esse *byte* de comando deve ser enviado ao fim de cada seqüência de configuração que estabelece uma conexão. O IMS C004 então está pronto para aceitar dados nas entradas conectadas.
[4]	Reinicializa a chave crossbar. Todas as saídas são mantidas em nível baixo. Isso também acontece quando um pulso, com no mínimo 1,6 ms é aplicado ao pino *Reset*.
[5] [saída j]	Esse comando desconecta a saída *j*.
[6][ligação p][ligação q]	Desconecta a saída do canal ligação *p* e a saída do canal de ligação *q*.

Fonte: INMOS. IMS C004 programmable link switch. In: **INMOS Engineering Data**, 1989. p. 489.

A documentação fornecida pela INMOS nada diz a respeito da possibilidade de se realizar difusão de uma entrada para *N* saídas. Entretanto, supõe-se que o envio de sucessivas mensagens de configuração do tipo [0][entrada i][saída j] permite que diversas saídas sejam conectadas a uma mesma entrada.

3.2 - Projeto do *hardware* de comunicação

O projeto do *hardware* de comunicação baseou-se no uso de circuitos integrados de aplicação específica (ASICs - *Application Specific Integrated Circuits*) do tipo arranjo de portas lógicas programáveis em campo (FPGA - *Field Programmable Gate Array*). Esses dispositivos apresentam milhares de portas associadas por meio de uma rede de interconexão configurada conforme a lógica definida no projeto.

Os FPGAs permitem a integração de centenas de circuitos SSI e MSI comerciais em um único *chip*, propiciando uma redução do número de componentes. Com isso, obtém-se produtos com maior confiabilidade, menor custo de desenvolvimento e manutenção facilitada.

Alguns modelos apresentam uma característica muito importante para os projetistas de *hardware*: podem ser apagados eletricamente, ou por luz ultravioleta, e serem reutilizados no mesmo, ou em outro, projeto. Por esse motivo, são também chamados de EPLDs (*Erasable Programmable Logic Devices*), ou seja, dispositivos lógicos programáveis e apagáveis.

Existem vários fornecedores de FPGAs no mercado internacional, como a Altera, Xilinx e QuickLogic, entre outros. Em geral, todos eles oferecem ambientes completos para o desenvolvimento desses dispositivos, com ferramentas para o projeto, depuração, simulação e programação dos componentes.

Face à disponibilidade da plataforma MAX+PLUS II[56] da Altera, decidiu-se por adotar esse ambiente no projeto dos adaptadores de comunicação. O seu uso foi feito sob a autorização da chefia do Laboratório de Instrumentação Eletrônica (LINSE) e dos professores responsáveis pelo Projeto Universitário Altera no Departamento de Engenharia Elétrica da UFSC.

3.2.1 - O ambiente de desenvolvimento Altera MAX+PLUS II

O MAX+PLUS II constitui-se em um ambiente para o desenvolvimento de projetos em FPGAs fornecido pela Altera Corporation. Esse ambiente pode ser executado tanto sob Windows ou Windows NT em um PC, como em estações de trabalho SPARCstation da Sun Microsystems, HP 9000 Series 700 ou DEC Alpha AXP.[57]

Segundo a própria Altera:

> "O sistema MAX+PLUS II completo suporta diversas as famílias de FPGAs da Altera. Ele apresenta uma interface gráfica intuitiva que acelera o projeto através de entradas sob a forma de gráficos, textos e formas de onda; e, ainda, oferece uma biblioteca com mais de 300 componentes TTL e macrofunções personalizadas. O MAX+PLUS II inclui a linguagem de descrição de *hardware* da Altera (AHDL - *Altera Hardware Description Language*), que suporta máquinas de estado, equações booleanas e tabelas da verdade. Inclui, também, o padrão industrial VHDL[58] e Verilog HDL. [...] O MAX+PLUS II provê compilação altamente automatizada, particionamento do projeto em

[56] MAX+PLUS - *Multiple Array MatriX Programable Logic User System.*

[57] ALTERA. **Choosing the right MAX+PLUS II development tools**. San Jose : Altera Corporation, 1994.

[58] VHDL - *Very High Speed Integrated Circuit* (VHSIC) *Hardware Description Language*.

> mais de um dispositivo, verificação do projeto, estimativa de atrasos para caminhos críticos, localização automática de erros, programação e verificação do dispositivo. "[59]

3.3 - O padrão de barramento para interface com o PC

O padrão de barramento escolhido para a interface dos adaptadores de comunicação com a placa-mãe PC foi o ISA-16 *bits* (*Industrial Standard Architecture*). Embora existam outros padrões com largura de 32 *bits* e capacidades de transferência maiores - como o EISA, o VESA e o PCI, a referida escolha justifica-se pelo fato de o padrão ISA apresentar a melhor disponibilidade de documentação para o projeto e o menor custo de implementação. Além disso, atende aos objetivos do Projeto Nó//, uma vez que sua preocupação inicial não é o desempenho e sim a validação do modelo de ambiente de programação paralela.

3.3.1 - O barramento ISA

O barramento ISA-16 *bits* constitui-se em uma extensão ao ISA-8 *bits*, oferecendo maiores capacidades para transferência de dados e endereçamento de memória, entre outros. Os principais atributos do barramento ISA-16 *bits*, relevantes ao projeto dos adaptadores de comunicação para os nodos do multicomputador Nó//, são:

- barramento de dados de 16 *bits* para transferências tanto em 8 *bits* como em 16 *bits*;
- barramento de endereços de 24 *bits* para endereçamento de memória tanto em 20 *bits* como em 24 *bits*;
- quatro canais de DMA (*Direct Memory Access*) para transferências em 8 *bits* (canais 0-3) e outros três para transferências em 16 *bits* (canais 5-7);
- 15 níveis de requisição de interrupção (IRQ 0-1; IRQ 3-15);
- freqüência de barramento de 10 MHz;[60]

---

[59] Ibid.

[60] Alguns modelos de PC apresentam barramentos ISA-16 *bits* com freqüências diferentes (de 8 a 12.5 MHz), conforme KLOTH (1994).

além de outros atributos irrelevantes para o projeto.[61]

3.4 - Análise de custos

Nesta seção, realiza-se uma análise do custo para a implementação de um conjunto de placas de comunicação que permita a construção de um protótipo do multicomputador Nó//. Esse protótipo terá oito nós de trabalho e um nó de controle, interconectados por meio de uma rede *crossbar* e de um mecanismo auxiliar de controle. Faz-se a análise de custo tanto para a arquitetura com mecanismo de controle baseado em um barramento compartilhado, como para aquela baseada em linhas de interrupção.

3.4.1 - Arquitetura baseada em um barramento compartilhado

Através de um projeto físico preliminar para essa arquitetura, definiu-se os principais circuitos integrados necessários à construção dos adaptadores de comunicação dos nós de trabalho e do nó de controle, conforme as Tabelas 3.3 e 3.4.[62]

Tabela 3.3 - Custo para a implementação do adaptador de comunicação dos nós de trabalho.

Nome	Descrição	Preço unitário	Qtd.	Preço total
IMS C011	adaptador de ligação	R$ 30,00	2	R$ 60,00
74_541	*driver*	R$ 2,00	1	R$ 2,00
74_245	*transceiver*	R$ 1,20	3	R$ 3,60
74_574	registrador	R$ 2,60	7	R$ 18,20
EPM 7068	FPGA	R$ 30,00	1	R$ 30,00
Total				R$ 113,80

[61] Maiores informações sobre o barramento ISA podem ser encontradas em EGGEBRECHT (1991), KLOTH (1994) e TISCHER (1991).

[62] Os preços apresentados foram obtidos a partir da consulta à distribuidores dos componentes listados.

Tabela 3.4 - Custo para a implementação do adaptador de comunicação do nó de controle.

Nome	Descrição	Preço unitário	Qtd.	Preço total
IMS C011	adaptador de ligação	R$ 30,00	3	R$ 90,00
74_541	*driver*	R$ 2,00	2	R$ 4,00
74_245	*transceiver*	R$ 1,20	3	R$ 3,60
74_574	registrador	R$ 2,60	9	R$ 23,40
EPM 7068	FPGA	R$ 30,00	1	R$ 30,00
Total				R$ 151,00

Dessa maneira, o custo mínimo para a construção dos adaptadores de comunicação, desconsiderando gastos com componentes discretos, placa e outros, é de R$ 1.061,00.[63]

3.4.2 - Arquitetura baseada em linhas de interrupção

Da mesma forma que para arquitetura anterior, realizou-se um projeto físico preliminar, o que possibilitou a definição dos principais circuitos integrados necessários à construção dos adaptadores de comunicação dos nós de trabalho e do nó de controle, conforme as Tabelas 3.5 e 3.6.

Tabela 3.5 - Custo para a implementação do adaptador de comunicação dos nós de trabalho.

Nome	Descrição	Preço unitário	Qtd.	Preço total
IMS C011	adaptador de ligação	R$ 30,00	1	R$ 30,00
74_541	*driver*	R$ 2,00	1	R$ 2,00
74_245	*transceiver*	R$ 1,20	2	R$ 2,40
74_574	*registrador*	R$ 2,60	4	R$ 10,40
EPM 7068	FPGA	R$ 30,00	1	R$ 30,00
Total				R$ 74,80

[63] 113,80 x 8 nós de trabalho + 151,00 x 1 nó de controle = 910,40 + 151,00 = 1.061,40

Tabela 3.6 - Custo para a implementação do adaptador de comunicação do nó de controle.

Nome	Descrição	Preço unitário	Qtd.	Preço total
IMS C011	adaptador de ligação	R$ 30,00	2	R$ 60,00
74_541	*driver*	R$ 2,00	3	R$ 6,00
74_245	*transceiver*	R$ 1,20	2	R$ 2,40
74_574	registrador	R$ 2,60	7	R$ 18,20
74_573	*latch*	R$ 1,50	1	R$ 1,50
74_30	*NAND* - 8 entradas	R$ 1,50	1	R$ 1,50
EPM 7068	FPGA	R$ 30,00	1	R$ 30,00
Total				R$ 119,60

Desse modo, o custo mínimo para a construção dos adaptadores de comunicação, desconsiderando gastos com componentes discretos, placas e outros, é de R$ 718,00.[64]

Para outras configurações de máquina, com 16 e 31 nós de trabalho, têm-se os seguintes custos:

Tabela 3.7- Custo para a implementação de um conjunto de adaptadores de comunicação para três configurações de máquina.

Número de Nós de Trabalho	Custo da arquitetura baseada em um barramento	Custo da arquitetura baseada em linhas de interrupção[65]	Diferença de custo relativa entre as duas arquiteturas [66]
8	R$ 1.061,00	R$ 718,00	32,3%
16	R$ 1.971,00	R$ 1.326,00	32,7%
31	R$ 3.527,80	R$ 2.467,20	30,0%[67]

A partir dos custos estimados para a construção de um conjunto de adaptadores de comunicação, pode-se verificar que a segunda arquitetura apresenta um custo de

[64] 74,80 x 8 nós de trabalho + 119,60 x 1 nó de controle = 598,40 + 119,60 = 718,00

[65] No cálculo do custo do adaptador de comunicação do nó de controle para essa arquitetura, foram considerados os custos referentes à implementação dos canais de interrupção para os nós de trabalho adicionais.

[66] Obtida pela relação percentual entre a diferença absoluta de custo das duas arquiteturas e o custo da arquitetura baseada em um barramento.

[67] Esse valor é ligeiramente menor pelo fato de, na segunda arquitetura, ocorrer a não utilização de um dos 32 canais de interrupção disponíveis.

implementação cerca de 30% menor que o da primeira. Diferentes configurações de máquinas apresentam relações muito próximas a essa. Assim, a arquitetura baseada em linhas de interrupção mostra-se mais econômica que aquela baseada em um barramento compartilhado.

3.5 - Análise de desempenho

A análise de desempenho foi realizada a partir de resultados de simulação obtidos sobre modelos desenvolvidos para as duas arquiteturas de máquina. Esses modelos foram elaborados por uma equipe de trabalho objetivando determinar taxas de ocupação dos recursos físicos da máquina (como o *crossbar* e o nó de controle) e a capacidade de transferência de dados do sistema de interconexão.

Foi definida uma aplicação baseada em chamadas aleatórias de sistema. Tal aplicação foi executada sobre os modelos de simulação das duas máquinas com diferentes cargas de trabalho e configurações da rede (8 ou 31 nós de trabalho).

Os resultados obtidos nos experimentos realizados mostraram que ambas as arquiteturas têm uma taxa de ocupação de recursos bastante reduzida, possibilitando o aumento da dimensão da máquina, e que as duas arquiteturas possuem capacidades de transferências de *bytes* muito próximas.[68, 69, 70] Na Figura 3.6, é mostrado um gráfico comparativo com resultados de simulação.

---

[68] MERKLE, Carla & BOING, Hamilcar. **Simulação do Nó//.** Florianópolis : Relatório Técnico UFSC-CTC-INE, 1996.

[69] ALVES, Valeria A. **Multicomputador Nó//: implementação de primitivas básicas de comunicação e análise de desempenho.** Dissertação de Mestrado, Curso de Pós-Graduação em Ciência da Computação, Universidade Federal de Santa Catarina, Florianópolis, SC, 1996.

[70] FREITAS Filho, Paulo F. et alli. A simulation model for the comparison of two multicomputer architectures. In: **Society for Computer Simulation Conference**, Portland, 1996.

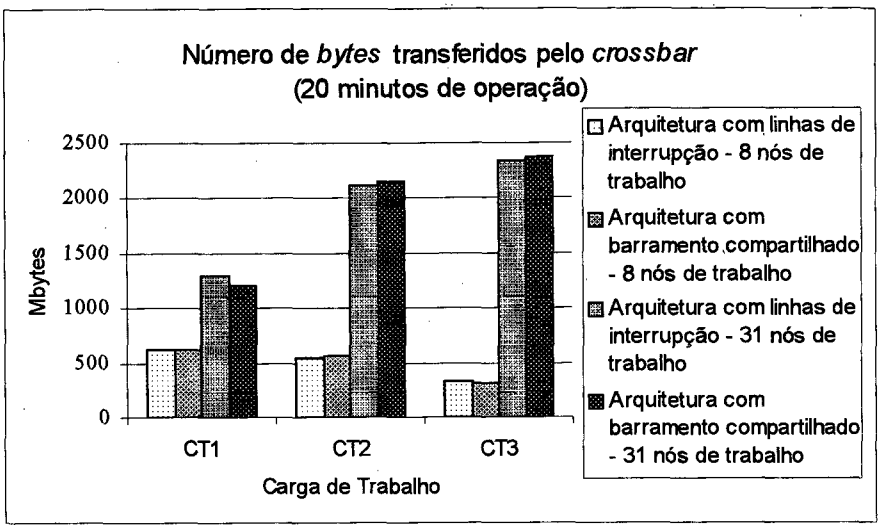


Figura 3.6 - Capacidade de transferência de dados pelo *crossbar*.

3.6 - Escolha do modelo de máquina

Considerando que a segunda arquitetura apresenta um *overhead* associado ao atendimento de um serviço menor que o da primeira e, ainda, com base nos resultados obtidos nas etapas de análise de custo e desempenho optou-se por desenvolver um protótipo inicial baseado na arquitetura com linhas de interrupção.

O capítulo a seguir apresenta a descrição do projeto dos adaptadores que integram o sistema de interconexão do multicomputador Nó//. Tal projeto foi desenvolvido sobre o modelo de arquitetura com mecanismo de controle baseado em linhas de interrupção.

CAPÍTULO 4

PROJETO DOS ADAPTADORES DE COMUNICAÇÃO

Introdução

Conforme visto no capítulo 2, o sistema de comunicação proposto para o multicomputador Nó// é formado por um conjunto de adaptadores que provê meios para a troca de mensagens entre os processadores da máquina. Dois tipos de adaptadores de comunicação foram definidos: um para os nós de trabalho e outro para o nó de controle. Ambos oferecem canais associados ao *crossbar* e ao mecanismo de controle, sendo que o segundo apresenta, ainda, um canal para a configuração da rede de interconexão. Além disso, no capítulo 3, foram definidos o modelo de arquitetura, o protocolo de nível físico e o padrão de barramento para o protótipo do multicomputador Nó//.

Este capítulo apresenta o projeto dos adaptadores que integram o sistema de comunicação do multicomputador Nó//. Inicialmente, descreve-se o modelo do protótipo da máquina a ser construído. Após, é feita uma descrição detalhada dos projetos físico e lógico dos adaptadores de comunicação para os nodos da máquina.

4.1 - O protótipo do multicomputador Nó//

O modelo de máquina definido para o protótipo do multicomputador Nó// consta de oito nós de trabalho e um nó de controle interconectados segundo a arquitetura com mecanismo de controle baseado em linhas de interrupção (Figura 4.1).

Nesse protótipo, as conexões chaveadas entre os nodos processadores provêem canais de comunicação baseados no protocolo de ligação serial INMOS. O gerenciamento dessas conexões pelo nó de controle é feito com o auxílio de linhas de interrupção que

permitem aos nós de trabalho interromper o nó de controle, e vice-versa. O uso dessas linhas, em conjunção com conexões temporárias no *crossbar*, proporciona a definição de um protocolo eficiente para o estabelecimento e cancelamento de conexões.

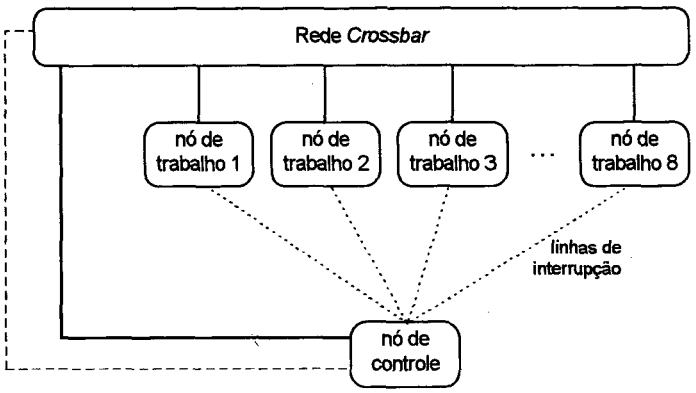


Figura 4.1 - Arquitetura do protótipo do multicomputador Nó//.

As interfaces para com o *crossbar* e o mecanismo de controle são providas por um adaptador de comunicação implementado sob a forma de uma placa de expansão conectada a um dos *slots* da placa-mãe PC. Esses adaptadores utilizam o IMS C011 para implementar canais de ligação com o *crossbar* e têm interface com o PC baseada no padrão de barramento ISA-16 *bits*. A Figura 4.2 mostra a estrutura dos nós de trabalho e de controle desse protótipo.

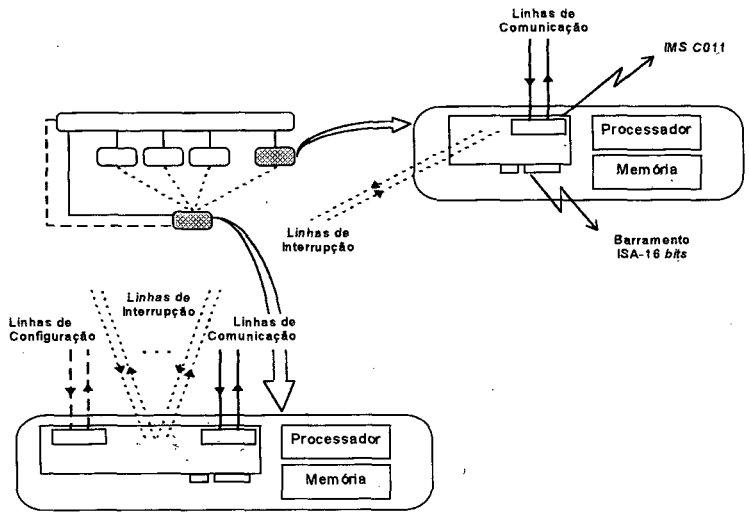


Figura 4.2 - Estrutura dos nodos do protótipo do multicomputador Nó//.

O adaptador de comunicação para os nós de trabalho, mostrado na Figura 4.3, conta com três módulos básicos que implementam as seguintes interfaces:

- interface com o *crossbar*: tem a função de prover um canal físico para comunicação via *crossbar*. Através desse canal, é feita a transferência de mensagens de dado com os outros nós de trabalho e mensagens de controle com o nó de controle. A comunicação baseia-se no protocolo de ligação INMOS, implementado via IMS C011.
- interface com as linhas de interrupção: oferece um canal bidirecional com duas linhas de interrupção. A primeira permite ao nó de trabalho interromper o nó de controle, enquanto que a segunda permite ao último interromper o primeiro;
- interface com o barramento ISA: responsável pela decodificação dos endereços da placa e geração dos sinais de controle, os quais estabelecem a comunicação entre a placa-mãe e as demais interfaces.

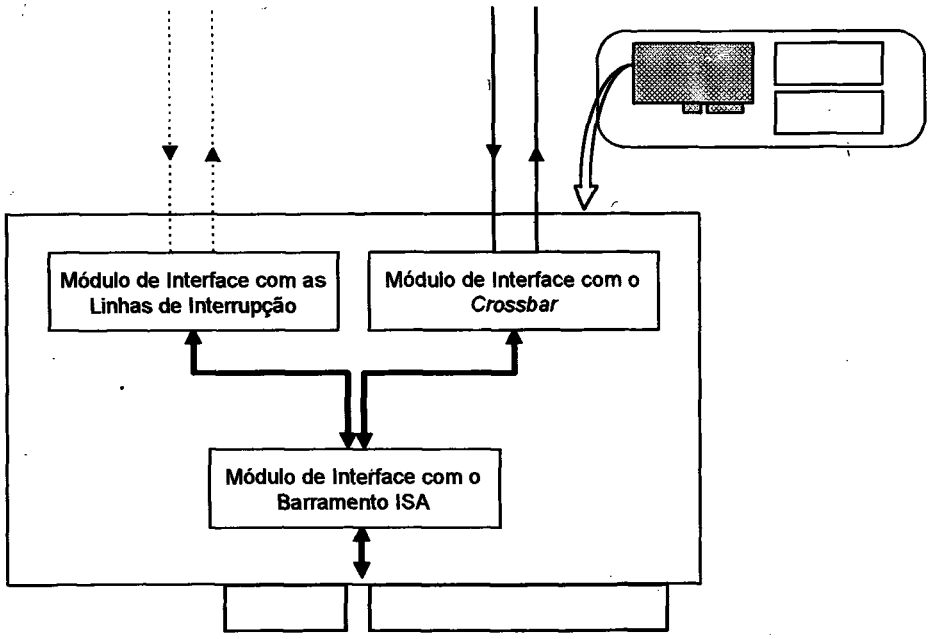


Figura 4.3 - Diagrama do adaptador de comunicação para os nós de trabalho.

O adaptador de comunicação para o nó de controle apresenta, além dessas interfaces, uma interface para a configuração do *crossbar*, pela qual o nó de controle realiza as conexões e desconexões na rede, conforme as requisições enviadas pelos nós de trabalho. As interfaces com o barramento ISA e com o *crossbar* são semelhantes às do adaptador para os

nós de trabalho. Já a interface com as linhas de interrupção tem uma estrutura "simétrica", com um par de linhas de interrupção para cada nó de trabalho. A Figura 4.4 apresenta a estrutura básica desse adaptador.

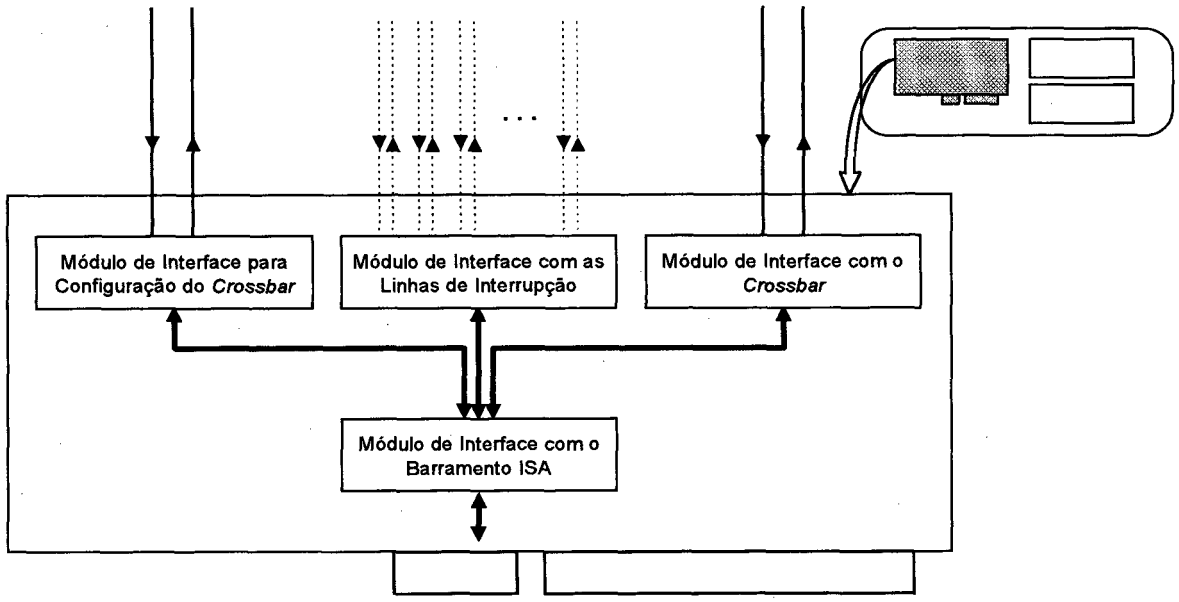


Figura 4.4 - Diagrama do adaptador de comunicação para o nó de controle.

Nas seções a seguir é feito o detalhamento do projeto dos adaptadores de comunicação para os nós de trabalho e de controle. São apresentados os projetos físico e lógico de cada módulo de interface mostrado nas Figuras 4.3 e 4.4.

4.2 - Projeto do adaptador de comunicação para os nós de trabalho

Conforme visto na Figura 4.3, o adaptador de comunicação para os nós de trabalho apresenta três módulos de interface: (i) com o *crossbar*; (ii) com as linhas de interrupção; e (iii) com o barramento ISA. Os projetos desses três módulos são descritos abaixo.

4.2.1 - Módulo de interface com o *crossbar*

Esse módulo provê um canal para comunicação serial através de uma ligação chaveada pelo *crossbar*, permitindo, ainda, a transferência de dados por meio de ciclos de acesso direto à memória (DMA).

4.2.1.1 - Projeto físico

Fisicamente, o módulo de interface com o *crossbar* é composto por um conjunto de registradores e por um adaptador de ligação IMS C011, além de um *driver* de linha e do circuito lógico de controle integrado em FPGA. Sua estrutura, mostrada na Figura 4.5, é definida de maneira a permitir que sejam realizadas transferências de dados de 16 *bits* a cada ciclo de leitura ou escrita no barramento ISA.

Esse módulo de interface é formado por um *driver* 74_541, quatro registradores 74_574 e um adaptador de ligação IMS C011, apresentado no capítulo 3. O 74_541 é um *driver* de linha que possui entradas com histerese e saídas *tri-state* controladas pelos sinais */g1* e */g2*. Ele é utilizado para reforçar e filtrar os sinais transferidos pela ligação serial. Já o 74_574 é um registrador de 8 *bits* que possui *flip-flops* tipo D sincronizados por um relógio (*clk*) sensível à borda de subida e saídas *tri-state* controladas pela linha */oc*.[71]

O protocolo para a transferência de um dado de 16 *bits* é assim definido: primeiro transfere-se o *byte* baixo (menos significativo) e após o alto (mais significativo). Dos quatro registradores desse módulo de interface, dois são dedicados ao armazenamento dos *bytes* recebidos pela entrada serial *LinkIn*, sendo denominados *rdL* (leitura do *byte* baixo) e *rdH* (leitura do *byte* alto). Os outros dois registradores são utilizados para o armazenamento de dados a serem transmitidos pela saída serial *LinkOut* e são referenciados por *wrL* (escrita do *byte* baixo) e *wrH* (escrita do *byte* alto).

---

[71] Na descrição do projeto do sistema de comunicação utiliza-se a seguinte simbologia:
- os sinais cujos nomes são iniciados por uma barra (/) são ativos em nível baixo;
- os termos *rd* e *wr* são abreviações de *read* (leitura) e *write* (escrita);
- as terminações *L* e *H* são abreviações de *Low* e *High* e indicam, respectivamente, *byte* baixo (menos significativo) e *byte* alto (mais significativo).

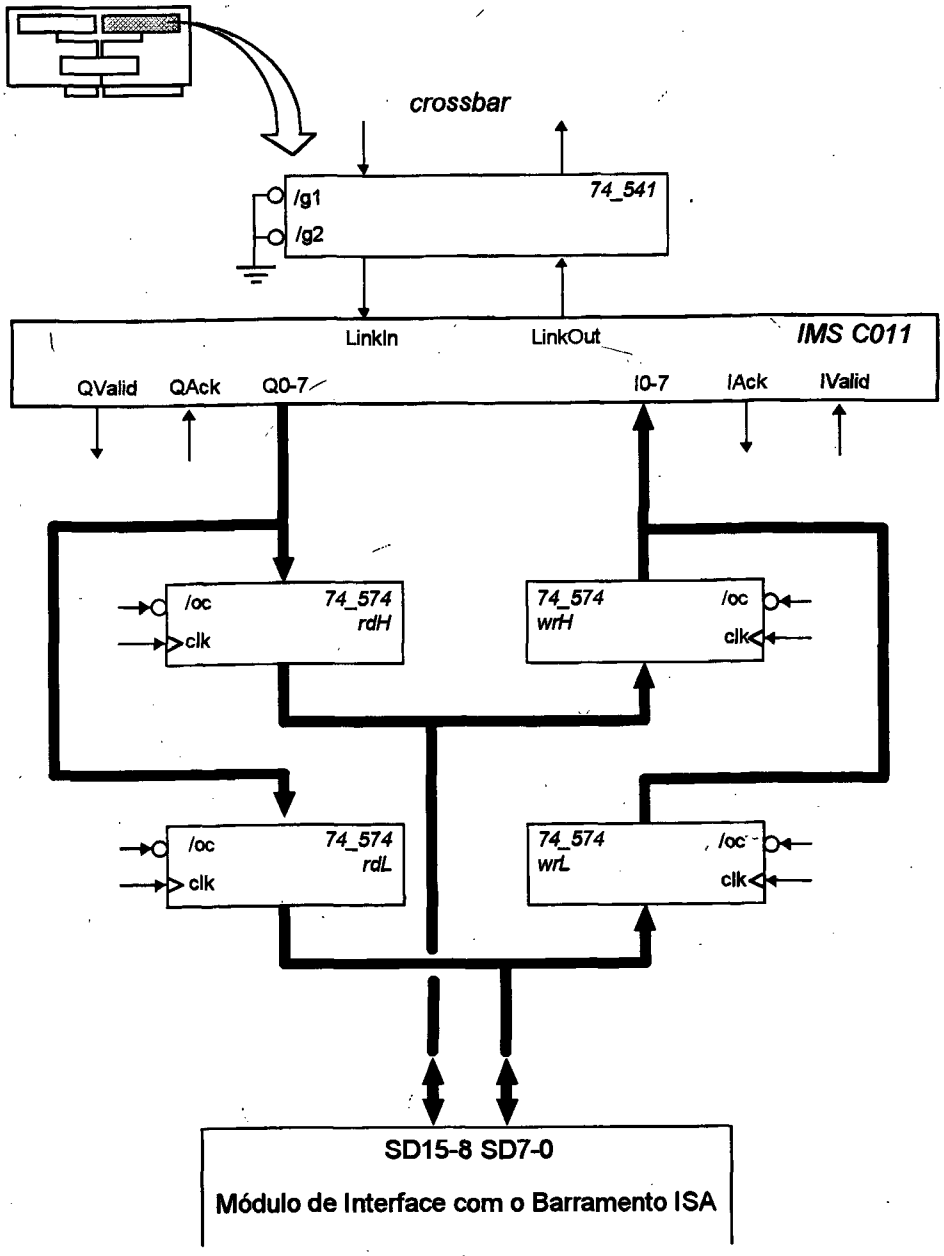


Figura 4.5 - Estrutura física do módulo de interface com o *crossbar*.

4.2.1.2 - Projeto lógico

O circuito lógico é projetado de modo a gerar os sinais de controle necessários à transferência de dados pelo IMS C011. São definidas duas máquinas de estado baseadas nos procedimentos apresentados nas seções 3.1.1.3 e 3.1.1.4. A máquina de escrita é utilizada para

comandar a transmissão dos *bytes* de dado, enquanto a máquina de leitura é destinada à recepção dos *bytes* de dado.[72]

Máquina de escrita

Os sinais de saída gerados por essa máquina de estado controlam as saídas do registradores *wrL* e *wrH*, a linha de validação do barramento *I0-7* do IMS C011 e uma linha para sinalizar que os dois *bytes* de dado foram transmitidos e que, portanto, os registradores encontram-se "vazios". Essa linha atua como uma requisição para um novo ciclo de escrita em I/O, sendo enviada ao processador, sob a forma de uma IRQ, ou ao controlador de DMA, sob a forma de uma DRQ.[73] Esses sinais são listados abaixo:

- */oc_wrL0*:[74] controla a saída *tri-state* do registrador de escrita para o *byte* baixo (*wrL*);
- */oc_wrH0*: controla a saída *tri-state* do registrador de escrita para o *byte* alto (*wrH*);
- *ivalid0*: indica ao IMS C011 a presença de um *byte* válido no barramento de entrada *I0-7* e que esse *byte* está pronto para ser transmitido;
- *obe0*: indica que o dado armazenado nos registradores de escrita foi transmitido com sucesso (*obe - output buffer empty*).

A geração dessas saídas é feita a partir de dois sinais de entrada: (i) um sinal de escrita em I/O, enviado pelo módulo de interface com o barramento ISA, é utilizado para transferir dados para dentro dos registradores; e (ii) a linha de reconhecimento de dado transmitido do IMS C011, conforme segue:

- */iowr0:* indica a ocorrência de um ciclo de escrita em I/O no endereço do módulo de interface com o *crossbar*. Esse sinal é amarrado às entradas de

---

[72] As máquinas de estado definidas no projeto lógico são máquinas de Moore. Cada estado apresenta uma configuração de saída e a transição de um estado para outro é determinada por uma condição de entradas e pelo estado atual. As transições são sincronizadas por um relógio global e só ocorrem na borda de subida (ou de descida) desse relógio.

[73] Na seção *"Seleção das requisições de escrita e de leitura"*, é explicada a forma com a qual a requisição de escrita é enviada ao barramento, como IRQ ou DRQ.

[74] A terminação 0 (zero) é utilizada para referenciar os sinais relativos ao módulo de interface com o *crossbar*, também denominado "módulo 0".

relógio (*clk*) dos registradores *wrL* e *wrH*, de modo que, durante a sua borda de subida, o dado apresentado em *SD15-0* é transferido para dentro dos registradores;

- *iack0*: indica que o *byte* transmitido foi recebido com sucesso e que o receptor está pronto para uma nova transferência.

Com base nos procedimentos para a transmissão de dados, mostrados na seção 3.1.1.3, foi desenvolvida uma máquina de estados para escrita, mostrada na Figura 4.6.

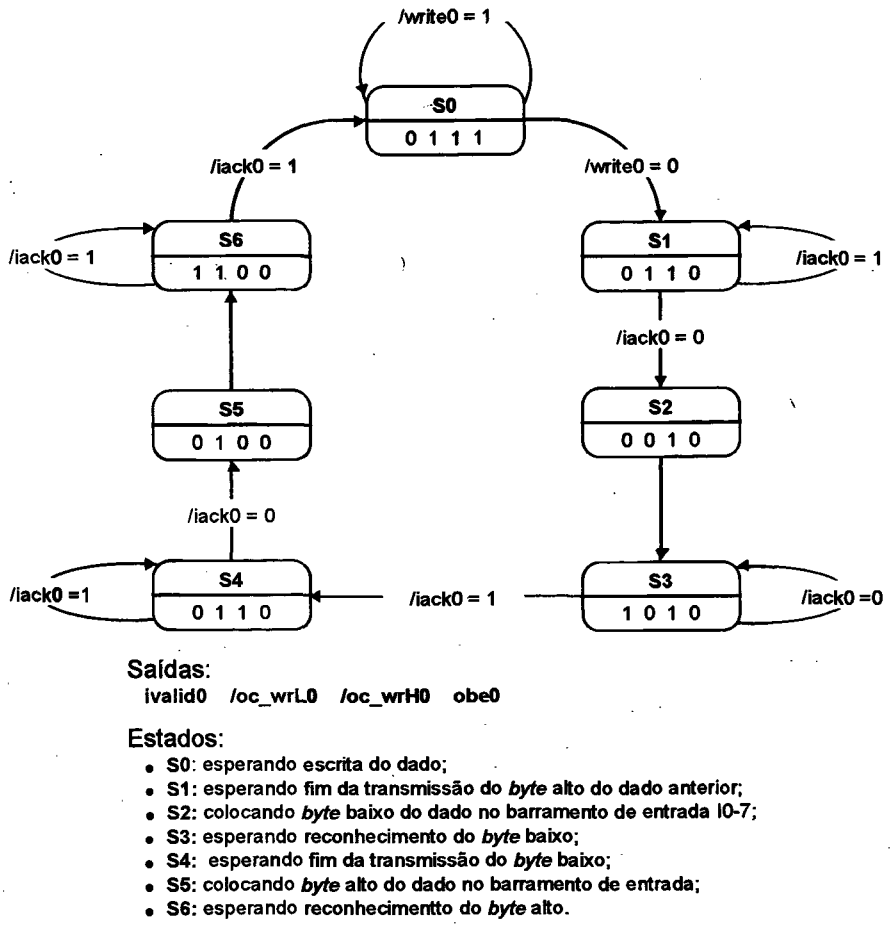


Figura 4.6 - Máquina de estados de escrita para a transmissão de dados pela interface de comunicação com o *crossbar*.

O ciclo da máquina de escrita tem início no estado S0, seja na partida ou na reinicialização do sistema, e mantém-se nele até a ocorrência de uma escrita no endereço do

módulo de interface com o *crossbar*. Nesse estado, todas as linhas de saída, com exceção de *obe*, encontram-se desativadas.

Quando no estado S0 o sinal */write0*, derivado de */iowr0*, desce para o nível baixo, ocorre uma transição para o estado S1, onde a saída *obe* é desabilitada, indicando que os registradores de escrita foram atualizados com um novo dado. A máquina se mantém nesse estado até o fim da transmissão do dado anterior, representada pela descida da entrada *iack0*.[75]

Quando *iack0* vai para o nível baixo, acontece uma transição para o estado S2 e a saída */oc_wrL0* é habilitada, fazendo com que o conteúdo do registrador *wrL* seja apresentado ao barramento *I0-7*.

No ciclo seguinte do relógio, ocorre uma transição incondicional para o estado S3, onde a saída *ivalid0* vai para o nível alto, indicando ao IMS C011 que há um novo *byte* válido no barramento *I0-7*. A partir desse momento, é iniciada a transmissão do dado e a máquina de estados fica esperando a chegada do sinal de reconhecimento para, então, preparar-se para o envio do *byte* alto.

Quando a entrada *iack0* vai para o nível alto, a máquina transita para o estado S4, desabilitando a linhas */oc_wrL0* e *ivalid0*. A evolução para o estado seguinte, S5, acontece apenas, quando essa mesma entrada retornar para o nível baixo, indicando o fim da transmissão do *byte*. Em S5, a saída */oc_wrH0* é ativada e o conteúdo do registrador *wrH* é colocado no barramento *I0-7*.

No ciclo seguinte, de forma incondicional, a máquina vai para o estado S6, onde a linha *ivalid0* é levada novamente para o nível alto, indicando ao IMS C011 para iniciar a transmissão do *byte* mais significativo do dado.

O ciclo termina com a subida da entrada *iack0*, o que indica que o dado foi transmitido com sucesso, e a máquina retorna ao estado S0. Nessa transição, *iack0* é levado para o nível baixo, */oc_wrL0* é desabilitada e a saída *obe* é reativada. A borda de subida de *obe* é utilizada para indicar ao processador, ou ao controlador de DMA, que o dado foi enviado e o módulo encontra-se pronto para uma nova escrita.

---

[75] Tal procedimento é feito de modo a respeitar a restrição imposta pelo protocolo de transmissão de dados do IMS C011, descrito na seção 3.1.1.3.

A linha */write0* é derivada de */iowr0* a fim de garantir que a ocorrência de uma borda de subida do pulso de escrita, onde ocorre o registro do dado, seja percebida pela máquina de estados. Para tal, é utilizado um *flip-flop* tipo D, conforme a Figura 4.7.

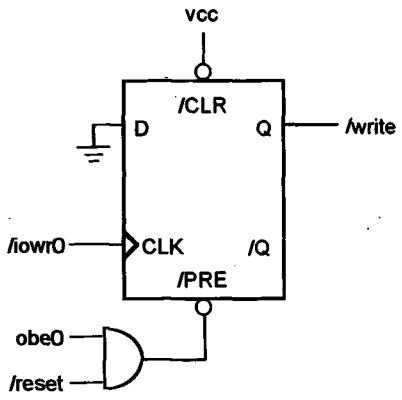


Figura 4.7 - Circuito para registrar a ocorrência de um pulso de escrita para o módulo de interface com o *crossbar*.

Nesse circuito, a saída */write* é inicializada em nível alto tanto na partida, quanto na reinicialização. Na borda de subida de */iowr0*, o conteúdo da entrada de dado D é transferido para a saída Q, levando */write* para o nível baixo. Quando essa mudança é percebida pela máquina de escrita, ocorre uma transição para o estado S1, onde a linha *obe0* é desativada. Como essa linha é amarrada ao *preset* desse *flip-flop*, a descida de *obe0* faz com que a saída */write* volte para o nível alto. Uma nova escrita só deverá ocorrer após a subida de *obe0*, no final do ciclo da máquina de estados.

Na Figura 4.8, é apresentada a estrutura do circuito transmissor de dados do módulo de interface com o *crossbar*. Além do que foi dito, deve-se ressaltar que o sinal */iowr0* é fornecido pelo módulo de interface com o barramento ISA e que a saída *obe0* é tratada por uma outra parte da lógica de controle desse módulo, onde ela é direcionada à linha de requisição adequada (IRQ ou DRQ).

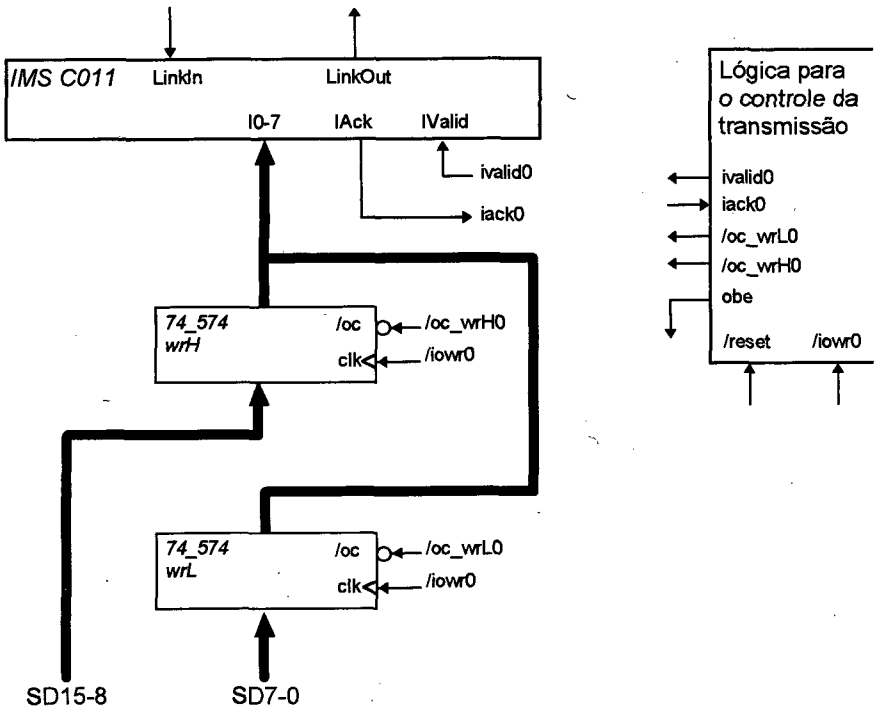


Figura 4.8 - Estrutura do circuito de transmissão do módulo de interface com o *crossbar*.

Máquina de leitura

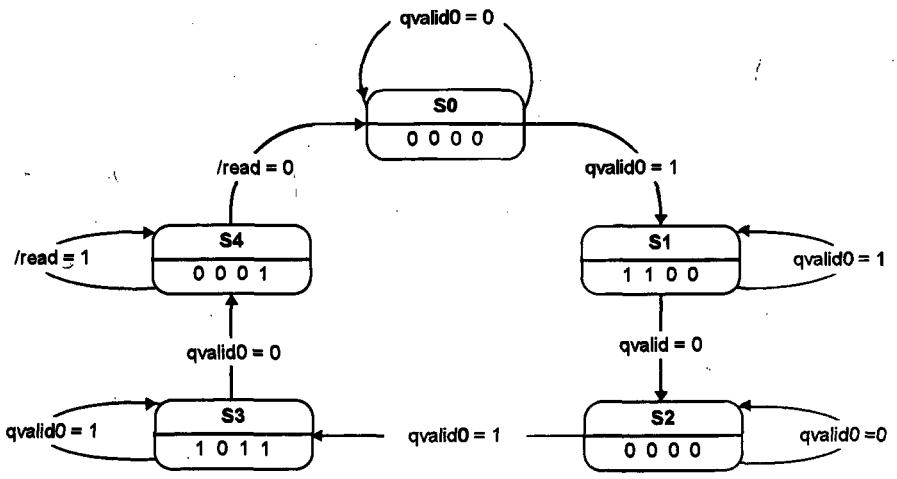
A máquina de leitura destina-se à recepção de dados pelo módulo de interface com o *crossbar*. Ela gera sinais de controle para comandar os relógios dos registradores *rdL* e *rdH*, realizar o reconhecimento do dado apresentado no barramento *Q0-7* do IMS C011 e sinalizar ao processador, ou ao controlador de DMA, o recebimento de um novo dado. Esses sinais são listados abaixo:

- *clk_rdL0*: transfere o conteúdo apresentado no barramento *Q0-7* para dentro do registrador de leitura do *byte* baixo (*rdL*);
- *clk_rdH0*: transfere o conteúdo apresentado no barramento *Q0-7* para dentro do registrador de leitura do *byte* alto (*rdH*);
- *qack0*: indica ao IMS C011 que o *byte* apresentado no barramento *Q0-7* já foi lido e que o módulo está pronto para receber um novo *byte* do transmissor;
- *ibf0*: indica ao processador, ou ao controlador de DMA, que foi recebido um novo dado e que ele encontra-se pronto para ser lido (*ibf - input buffer full*).

A geração dessas saídas é feita a partir de dois sinais de entrada: (i) a linha de validação de dado recebido pelo IMS C011; e (ii) um sinal de leitura em I/O, enviado pelo módulo de interface com o barramento ISA, conforme segue:

- *qvalid0*: indica que foi recebido um *byte* pelo IMS C011 e que ele está pronto para ser lido através do barramento *Q0-7*;
- */iord0*: indica a ocorrência de um ciclo de leitura em I/O no endereço do módulo de interface com o *crossbar*. Esse sinal é conectado às linhas de controle de saída (*/oc*) dos registradores *rdL* e *rdH*, de modo que, durante o período em que esse sinal encontra-se em nível baixo, o dado armazenado nos registradores de leitura possa ser transferido para o barramento *SD15-0*.

Com base nos procedimentos para a recepção de dados, mostrados na seção 3.1.1.4, foi desenvolvida a máquina de estados para leitura, mostrada na Figura 4.9.



Saídas:
qack0 clk_rdL0 clk_rdH0 ibf0

Estados:
- **S0**: esperando recebimento do *byte* baixo;
- **S1**: esperando fim do recebimento do *byte* baixo;
- **S2**: esperando recebimento do *byte* alto;
- **S3**: esperando fim do recebimento do *byte* alto;
- **S4**: esperando leitura do dado recebido.

Figura 4.9 - Máquina de estados de leitura para a recepção de dados pela interface de comunicação com o *crossbar*.

O ciclo da máquina de leitura tem início no estado S0, seja na partida ou na reinicialização do sistema, e mantém-se nele até que o IMS C011 receba um *byte* e leve *qvalid0* para o nível alto. No estado inicial, todas as saídas são desabilitadas e mantidas no nível baixo.

A subida de *qvalid0* leva a máquina a evoluir para o estado S1, no qual as saídas *clk_rdL0* e *qack0* são ativadas, realizando, respectivamente, a escrita do *byte* recebido no registrador *rdL* e o seu reconhecimento. Esse estado é mantido enquanto *qvalid0* permanecer em nível alto.

A descida de *qvalid0* provoca a transição da máquina para o estado S2, onde, de forma equivalente a S0, todas as saídas são mantidas em nível baixo até a chegada de um segundo *byte*.

Uma nova subida de *qvalid0* leva a máquina ao estado S3. Nesse estado, são realizados: (i) o registro do *byte* recebido pelo C011, através da ativação da linha *clk_rdH0*; (ii) a confirmação do recebimento do *byte*, por meio da habilitação da saída *qack0*; e (iii) a sinalização da chegada de um novo dado, pela subida da linha de saída *ibf0*. Esta última é utilizada para requisitar ao processador, ou ao controlador de DMA, a leitura do dado recebido.

A transição para o estado seguinte ocorre com a descida da linha *qvalid0*. Em S4, todas as saídas, com exceção de *ibf0*, são desativadas. Esse estado se mantém até a realização de um ciclo de leitura no endereço do módulo de interface com o *crossbar*, o que faz com que a máquina retorne ao seu estado inicial.

Da mesma forma que para a máquina de escrita, o pulso de leitura deve ser registrado por meio de um *flip-flop* tipo D, garantindo, assim, que sua borda de subida seja percebida pela máquina de estados. A Figura 4.10 ilustra a implementação desse *flip-flop*.

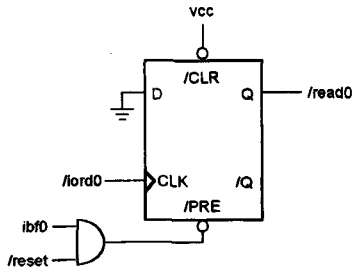


Figura 4.10 - Circuito para registrar a ocorrência de um pulso de leitura no módulo de interface com o *crossbar*.

Como no circuito da Figura 4.7, a saída */read0* também é inicializada em nível alto. A borda de subida de */iord0* faz com que ela vá para o nível baixo, onde mantém-se até a transição do estado S4 para o estado S0, onde *ibf0* é desativado. A desabilitação dessa linha leva a saída */read0* novamente para o nível alto.

O sinal */iord0* é provido pelo módulo de interface com o barramento ISA e a saída *ibf0* é tratada, posteriormente, por uma outra parte da lógica de controle, na qual ela é direcionada para a linha de requisição adequada, IRQ ou DRQ.

A estrutura do circuito receptor de dados do módulo de interface com o *crossbar* é apresentada na Figura 4.11.

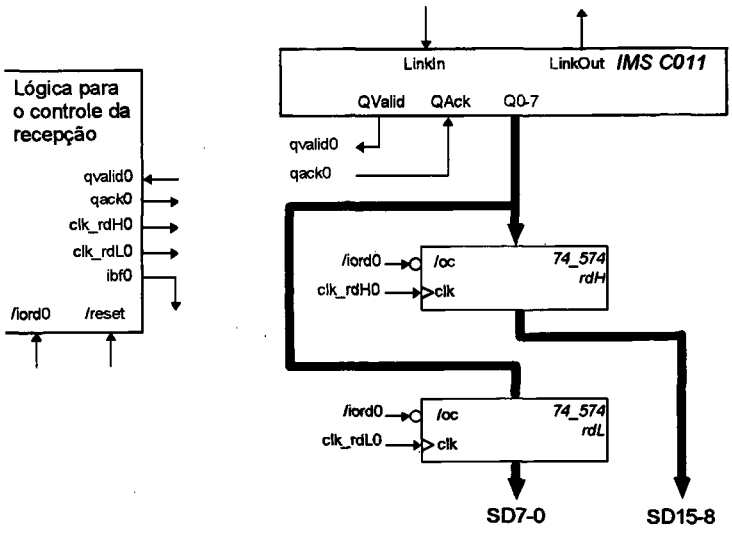


Figura 4.11 - Estrutura do circuito de recepção do módulo de interface com o *crossbar*.

Seleção das requisições de escrita e de leitura

As transferências de dados através do módulo de interface com o *crossbar* podem ser feitas tanto por ciclos de I/O, como por ciclos de acesso direto à memória (DMA).

Durante um ciclo de escrita em I/O, por exemplo, o processador primeiramente move o conteúdo de um endereço de memória para um de seus registradores e, após, executa uma instrução `OUT`, de modo a escrevê-lo no endereço do periférico. A transferência de grandes blocos de memória através desses ciclos é, em alguns casos, demasiadamente lenta.

Em uma escrita em I/O com acesso direto à memória, por exemplo, o processador transfere para o controlador de DMA a tarefa de enviar um bloco de memória para um determinado periférico. Os ciclos de DMA são realizados de maneira que a memória e o periférico sejam endereçados simultaneamente e os dados fluam do primeiro para o segundo sem passar pelo controlador. Isso leva a um incremento no desempenho dessas transferências. Alguns tipos de periféricos que utilizam transferências DMA incluem os acionadores de disco rígido, disco flexível e CD-ROM, entre outros.

O endereçamento simultâneo da memória e do periférico é obtido por meio do uso de uma linha específica para o segundo, enquanto a primeira é acessada diretamente pelo barramento de endereços. Essa linha específica é parte integrante de um canal, chamado "canal de DMA". O barramento ISA oferece quatro canais para transferências de dados de 8 *bits* e três canais para dados de 16 *bits*. Cada canal é formado por uma linha de requisição DRQ e outra de reconhecimento /DACK. A primeira é utilizada pelo periférico para solicitar ao controlador a realização de um novo ciclo de I/O com DMA. A segunda permite ao controlador endereçar o periférico durante os ciclos de transferência com acesso direto à memória.

O módulo de interface com o *crossbar* possibilita a realização de transferências de mensagens curtas e longas, por ciclos de I/O, e de arquivos, por ciclos de DMA. São oferecidos dois canais exclusivos de DMA, um para escrita e outro para leitura. No canal de escrita, a saída *obe0* da lógica de controle, normalmente direcionada para uma requisição de interrupção (IRQ), pode ser chaveada para uma linha de requisição de um dos canais de DMA de 16 *bits* (DRQ). O mesmo ocorre com a saída *ibf0* do canal de leitura.

O chaveamento da linha *obe0*, entre as requisições *irq_wr0* e *drq_wr0*, ocorre conforme a Figura 4.12. Esse circuito serve para chavear a saída *obe0* da linha de requisição *irq_wr0* para *drq_wr0*, o que é feito a partir do sinal */den_iowr0*, fornecido pela interface com o barramento ISA.

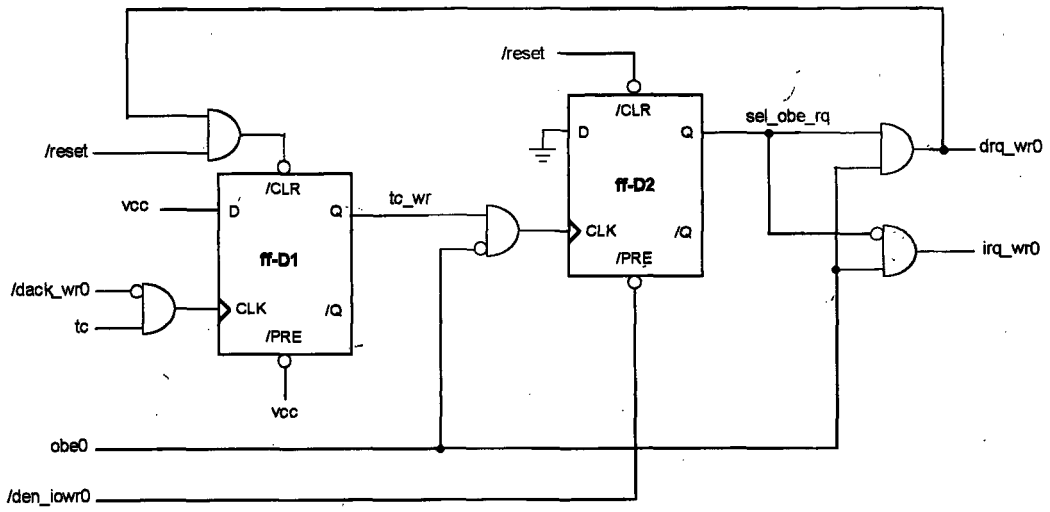


Figura 4.12 - Circuito para seleção da requisição de saída da linha *obe0*.

A partida ou reinicialização do sistema leva as saídas para o seguinte estado: *drq_wr0* é levada ao nível baixo e *irq_wr0* passa a refletir o estado da entrada *obe0*, habilitando o módulo de interface para a realização de ciclos de escrita em I/O.

Para execução de ciclos de escrita em I/O com acesso direto à memória o processador deve realizar uma escrita em um endereço específico, o que produz um pulso de nível baixo na linha */den_iowr0* da interface com o barramento ISA. Esse pulso atua no *Preset* do segundo *flip-flop*, levando sua saída para o nível alto e chaveando a entrada *obe0* de *irq_wr0* para *drq_wr0*. A partir desse momento, essa última saída passa a refletir o estado da entrada, de forma que, cada vez que um dado é transmitido, é gerada uma requisição de escrita em I/O com DMA.

O chaveamento reverso, de *drq_wr0* para *irq_wr0*, ocorre, automaticamente, com o fim da transferência. Durante a escrita do último dado, o controlador de DMA leva a linha *tc*, que indica término de contagem, para o nível alto. A combinação dessa linha com o sinal de reconhecimento */dack_wr0*, do canal de escrita em DMA, faz com que, na descida de *obe0*, seja gerada uma borda de subida na entrada de relógio do segundo *flip-flop*, levando a sua

saída para o nível baixo. Isso conduz as linhas *drq_wr0* e *irq_wr0* de volta aos seus estados iniciais. Por fim, o retorno de *drq_wr0* ao nível baixo "limpa" a saída do primeiro *flip-flop*, desabilitando *tc_wr*.

Um circuito idêntico a esse é utilizado no chaveamento da linha *ibf0*, conforme a Figura 4.13. Como ele possui funcionamento semelhante ao primeiro, não será feita a descrição da sua operação.

Os circuitos das Figuras 4.12 e 4.13 são integrados com as máquinas de estado para escrita e para leitura em um único circuito lógico. Esse circuito têm entradas provenientes da interface com o barramento ISA e do IMS C011. Suas saídas são aplicadas a essas unidades e aos registradores *wrL*, *wrH*, *rdL* e *rdH*.

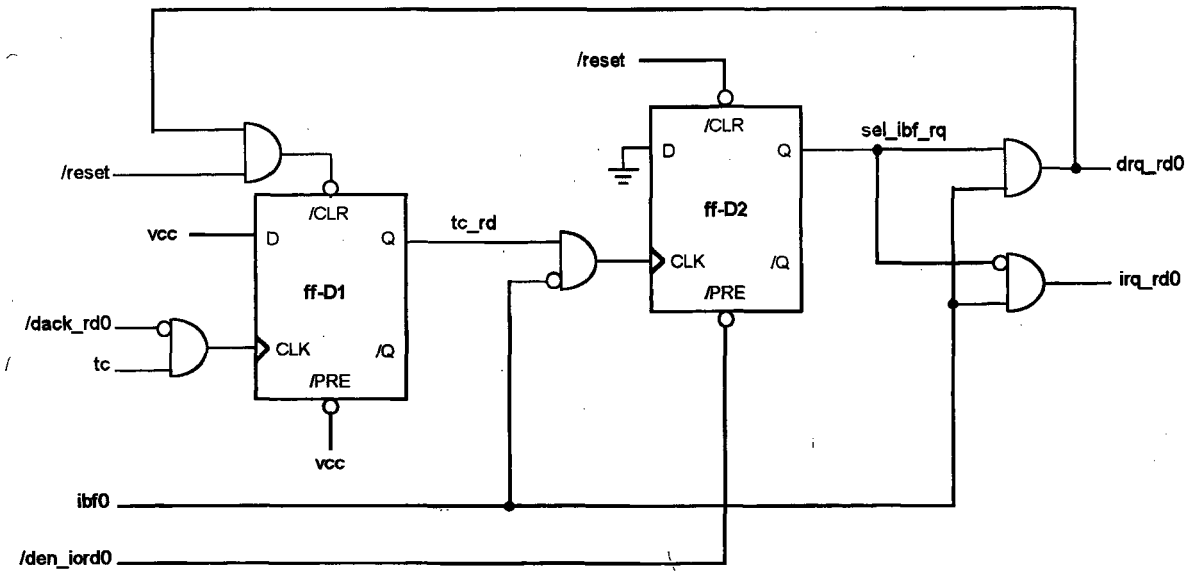


Figura 4.13 - Circuito para seleção da requisição de saída da linha *ibf0*.

A Figura 4.14 apresenta um diagrama de blocos onde são mostradas as linhas de entrada e de saída do circuito lógico do módulo de interface com o *crossbar*.

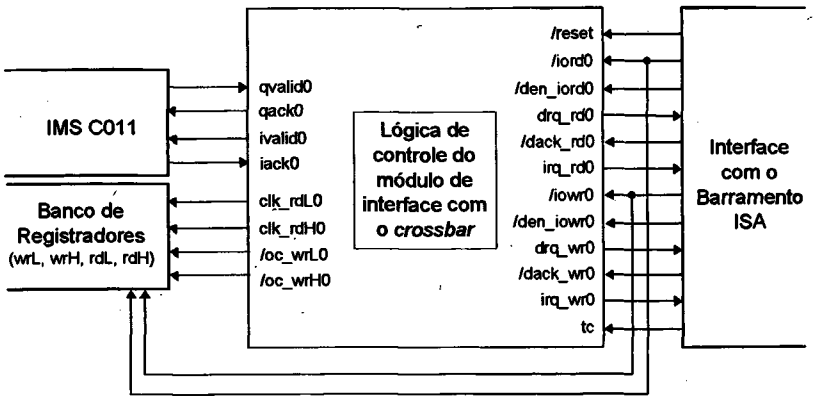


Figura 4.14 - Diagrama de blocos da lógica de controle do módulo de interface com o *crossbar*.

Já a Figura 4.15 apresenta um diagrama final do módulo de interface com o *crossbar*, mostrando seus componentes físicos e os sinais de controle que neles atuam.

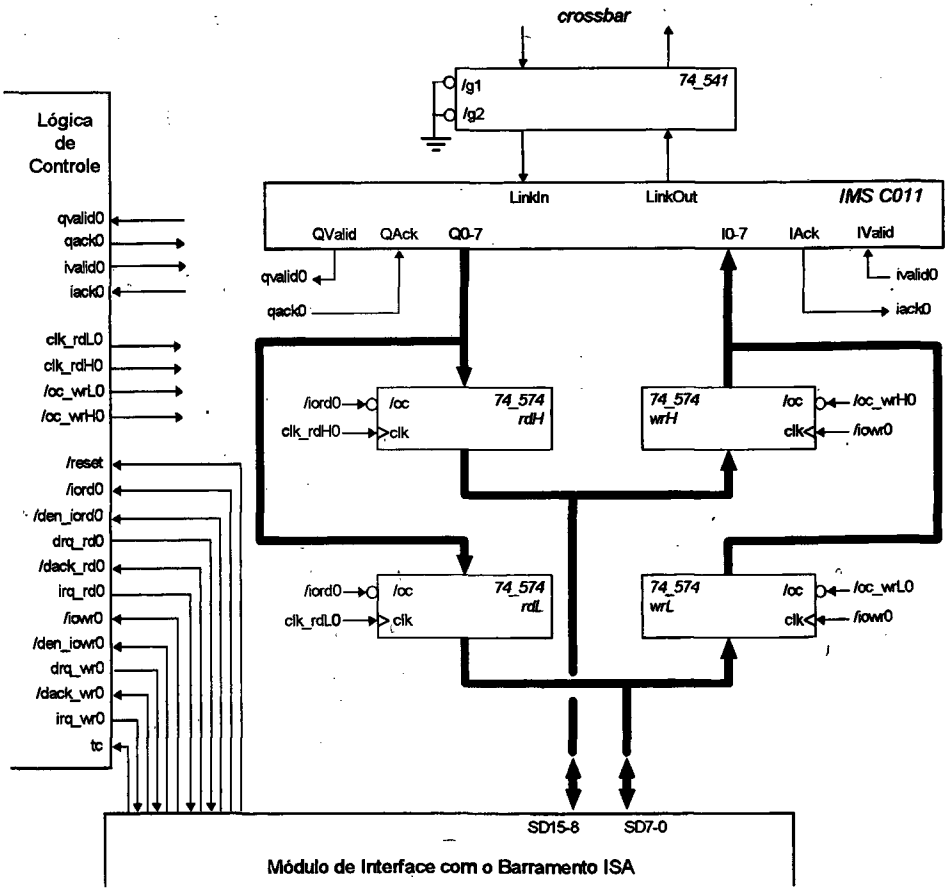


Figura 4.15 - Estrutura completa do módulo de interface com o *crossbar*.

4.2.2 - Módulo de interface com as linhas de interrupção

Esse módulo de interface implementa um canal bidirecional de interrupção para com o adaptador de comunicação do nó de controle. São definidas duas linhas de interrupção. A primeira, chamada */intr_TC*, permite ao nó de trabalho interromper o nó de controle a fim de solicitar o atendimento de algum tipo de serviço. A segunda, denominada */intr_CT*, é utilizada pelo nó de controle para interromper o nó de trabalho, confirmando o atendimento do serviço requisitado ou informando-o da realização de uma conexão, ou desconexão, à sua revelia. Essas linhas são utilizadas em conjunção com conexões temporárias no *crossbar*, através das quais são transferidas as mensagens de controle. O diagrama de estados da Figura 4.16 ilustra o funcionamento desse mecanismo.

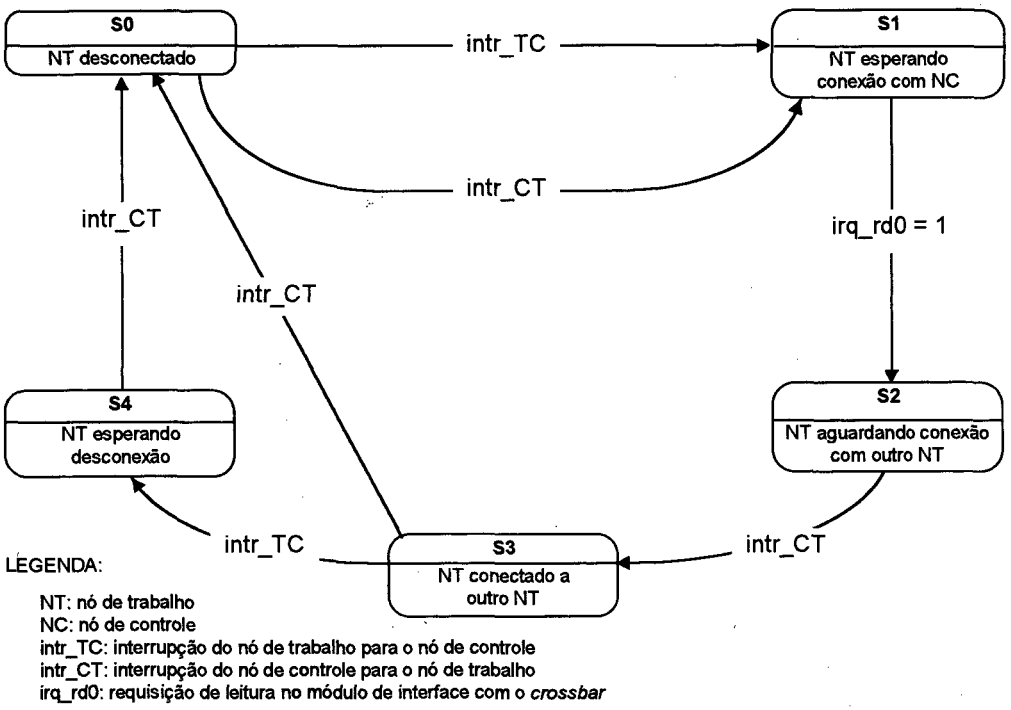


Figura 4.16 - Diagrama de estados do mecanismo de controle baseado em linhas de interrupção.

Durante o funcionamento normal da máquina, quando um nó de trabalho em processamento (estado S0) deseja conectar-se a um outro nó de trabalho, envia uma interrupção *intr_TC* ao nó de controle, indo para o estado S1. Nesse estado, o nó de trabalho fica esperando que o nó de controle estabeleça uma conexão via *crossbar* e lhe envie uma mensagem de consulta. A chegada do primeiro dado da mensagem do nó de controle provoca

uma transição para o estado S2, onde o nó de trabalho envia, em resposta, uma mensagem com a requisição para conexão com o outro nó. O nó de trabalho mantém-se nesse estado até o recebimento de uma interrupção *intr_CT*.

O nó de controle, após receber a requisição de conexão, busca atender o serviço solicitado. Se houver uma requisição recíproca do outro nó de trabalho, estabelece a conexão imediatamente. Se não, dois casos são possíveis. No primeiro, o nó de controle insere o pedido em uma tabela de pendências, onde ele é mantido até o recebimento da requisição equivalente, quando então é estabelecida a conexão. No segundo, o nó de controle interrompe o outro nó com o qual o nó de trabalho requisitante deseja conectar-se, envia-lhe uma mensagem informando-o que irá conectá-lo a um determinado nó e, após, estabelece a conexão. Em todos os casos, depois de estabelecida a conexão entre os nós de trabalho, o nó de controle envia uma interrupção a cada um dos nodos conectados.

O nó de trabalho no estado S2, após receber a interrupção *intr_CT*, vai para o estado S3, onde é realizada a troca de mensagens com o outro nodo a ele conectado. Quando a conexão não é mais necessária, os nodos enviam, cada um, uma interrupção *intr_TC* ao nó de controle e vão para o estado S4.

O nó de controle, ao receber uma interrupção de um nó de trabalho conectado, cancela a conexão em uso e envia-lhe, em resposta, uma *intr_CT*. Se, após isso, receber a interrupção do outro nodo que estava conectado, apenas retorna-lhe com uma *intr_CT*, confirmando a desconexão.

O nó de trabalho no estado S4, ao receber a *intr_CT* de confirmação de desconexão, transita para o estado inicial, encerrando o ciclo de uso de uma conexão.

Um nó de trabalho no estado S0 ou S3, ao receber uma *intr_CT* do nó de controle, coloca-se a sua disposição para, respectivamente, ser conectado ao mesmo e receber uma mensagem de controle ou liberar a conexão em uso. Se estiver no estado S0, vai para o estado S1. Se no estado S3, transita para o estado inicial.

4.2.2.1 - Projeto físico

De modo a oferecer uma boa imunidade aos ruídos do meio, os sinais de interrupção */intr_TC* e */intr_CT* são definidos como ativos em nível baixo, sendo os circuitos para detecção dessas interrupções sensíveis ao nível.

O projeto físico desse módulo de interface é simplificado pela possibilidade de se integrar sua funcionalidade em FPGA, produzindo um circuito conforme mostrado na Figura 4.17.

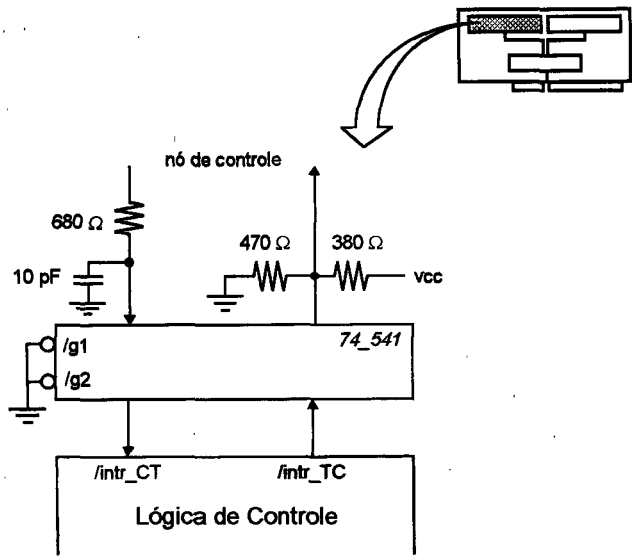


Figura 4.17 - Estrutura física do módulo de interface com as linhas de interrupção.

Nessa estrutura, utiliza-se um *driver* 74_541 para reforçar e filtrar os sinais de interrupção trocados por meio das linhas */intr_TC* e */intr_CT*. Para isso, é aproveitado o mesmo componente utilizado no módulo de interface com *crossbar*, pois, naquele projeto, o seu uso é parcial. São utilizados, ainda, os seguintes circuitos terminadores:

- divisor resistivo: garante o nível alto na linha de saída quando a mesma encontra-se inativa.
- circuito RC: aumenta a imunidade a ruídos, pela introdução de um capacitor paralelo, e limita a corrente de entrada da linha, por meio de um resistor série.

4.2.2.2 - Projeto lógico:

No projeto lógico, são implementados os circuitos de detecção da interrupção */intr_CT* e geração da saída */intr_TC* para o nó de controle, mostrados na Figura 4.18.

O circuito para detecção e geração das linhas de interrupção apresenta três *flip-flops* (dois SR e um D). Os *flip-flops* SR1 e D são utilizados para a detecção da ocorrência de um interrupção proveniente do nó de controle, */intr_CT*. Já o *flip-flop* SR2 é usado na geração da interrupção */intr_TC* para o nó de controle. Após a partida, ou reinicialização, as saídas desses circuitos são desativadas. Assim, as linhas */intr_TC* e *irq_CT* vão, respectivamente, para os níveis alto e baixo.

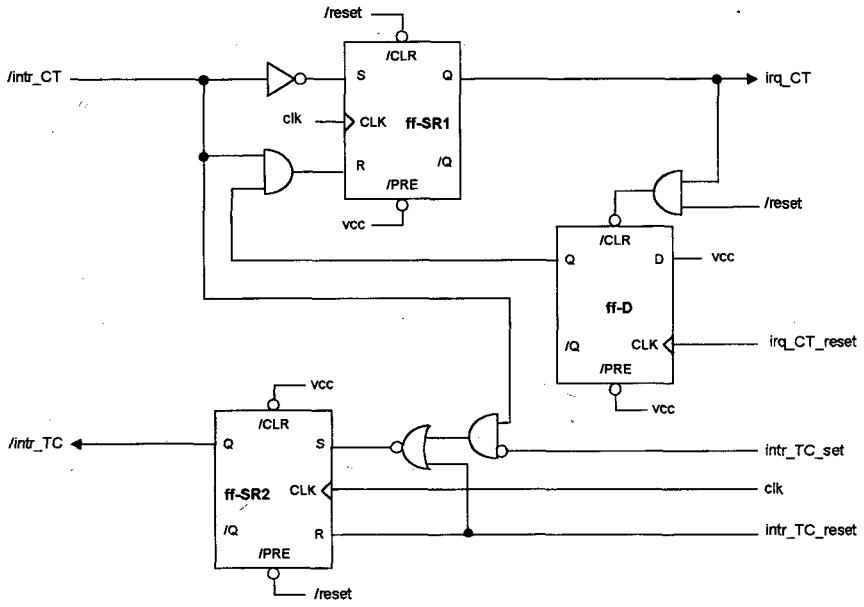


Figura 4.18 - Circuito para detecção e geração das linhas de interrupção.

Quando o nó de trabalho deseja o estabelecimento de uma conexão com um outro nó de trabalho, o seu processador realiza um ciclo de I/O em um endereço específico, o que faz com que o módulo de interface com o barramento ISA gere um pulso *intr_TC_reset* na entrada *Reset* do *flip-flop* SR1. Esse pulso provoca a descida de */intr_TC*, iniciando o envio de uma interrupção para o nó de controle.

Em seguida, o nó de trabalho mantém-se na espera pela chegada de uma mensagem do nó de controle via *crossbar*, quando, então, desativa a interrupção gerada. Isso é feito por meio de um ciclo de I/O em um determinado endereço, o que induz a interface com o

barramento ISA a produzir um pulso *intr_TC_set* que, através de um circuito combinacional, é aplicado à entrada *Set* do *flip-flop* SR1. Esse pulso leva a saída */intr_TC* a retornar para o nível alto.

O circuito combinacional utilizado na entrada *Set* do *flip-flop* SR1 é formado por duas portas lógicas. A porta *AND* permite que tanto *intr_TC_set* quanto */intr_CT* sejam aplicados nessa entrada. Já a porta *NOR* define que, nesse *flip-flop*, o *Reset* prevalece sobre o *Set*. Desse modo, garante-se que, na ocorrência simultânea das duas entradas, a saída não vá para um estado indefinido ou, então, torne-se pulsada. A entrada */intr_CT* atua efetivamente nesse circuito durante a confirmação de um serviço de desconexão solicitado pelo nó de trabalho, quando, então, não é recebida nenhuma mensagem por parte do nó de controle.

O circuito detector da interrupção */intr_CT* é formado por dois *flip-flops*, um do tipo SR para a detecção e outro do tipo D para reinicialização do primeiro. O *flip-flop* SR1 tem a linha */intr_CT* aplicada à sua entrada *Set* por meio de um porta inversora. Quando o circuito recebe um interrupção do nó de controle, a saída do *flip-flop*, inicialmente em nível baixo, é conduzida ao nível alto. A borda de subida desse sinal produz uma requisição de interrupção para o processador, chamada *irq_CT*. Essa interrupção mantém-se em nível alto até que a rotina para o seu tratamento gere um *Reset* para o *flip-flop* SR1.

A reinicialização do *flip-flop* de detecção é controlada pelo *flip-flop* D. Quando a rotina para o tratamento da interrupção leva o processador a realizar um ciclo de I/O em um endereço específico, o módulo de interface com o barramento ISA produz um pulso *irq_CT_reset* que é aplicado ao relógio (*clk*) do *flip-flop* D. Esse pulso é registrado e aplicado na entrada *Reset* do *flip-flop* SR1 através de uma porta *AND*. Essa porta, controlada pela linha */intr_CT*, garante que o *Set* prevaleça sobre o *Reset* e evita que, com a ocorrência simultânea das duas entradas, a saída torne-se indeterminada ou pulsada.

Quando a saída do *flip-flop* SR1 retorna para o seu estado original, ela automaticamente apaga o registro do pulso *irq_CT_reset*. A Figura 4.19 apresenta um diagrama completo desse módulo de interface com as linhas de interrupção.

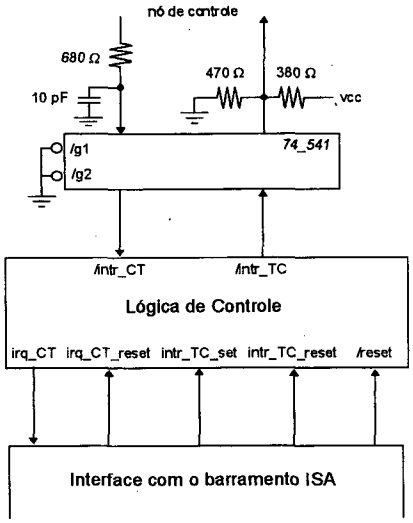


Figura 4.19 - Estrutura completa do módulo de interface com as linhas de interrupção.

4.2.3 - Módulo de interface com o barramento ISA

Esse módulo de interface realiza a decodificação dos endereços gerados para o adaptador de comunicação, bem como o controle dos componentes que realizam a isolação da placa com o barramento ISA.

4.2.3.1 - Projeto físico

Fisicamente, o módulo de interface com o barramento ISA é constituído por dois *transceivers* 74_245 e pelo FPGA que integra a parte lógica de controle e endereçamento, conforme a Figura 4.20.

Os *tranceivers* 74_245 oferecem a isolação necessária nos momentos em que a placa não é acessada através do barramento ISA. Esse componente apresenta duas interfaces bidirecionais de oito *bits* controladas pelas linhas *dir* e */g*. A primeira linha comanda o sentido do fluxo de dados entre os barramentos ligados às linhas *A1-8* e *B1-8*. Quando *dir* encontra-se em nível alto, o fluxo se dá de *A* para *B*. Do contrário, de *B* para *A*. Já a linha */g* controla a isolação dos barramentos. Quando em nível alto, *A1-8* e *B1-8* vão para um estado de alta impedância, desacoplando os barramentos a elas conectados. Quando em nível baixo, provê uma ligação direta entre os dois lados do componente.

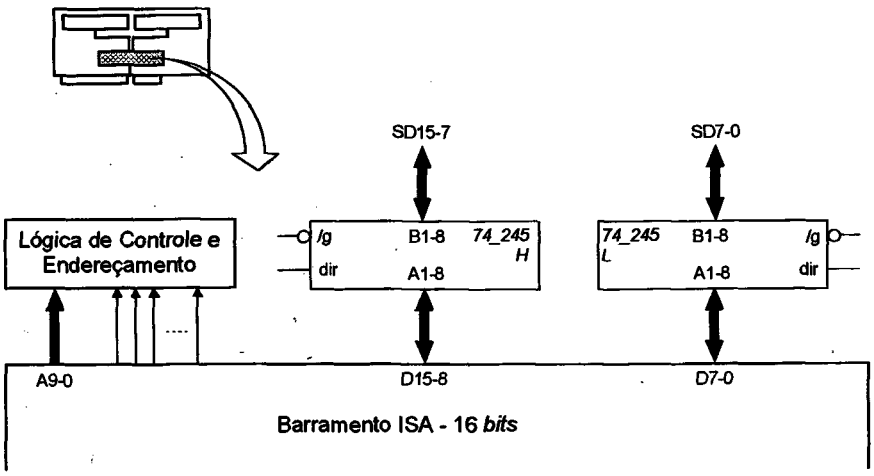


Figura 4.20 - Estrutura do módulo de interface com o barramento ISA.

Nesse projeto, */dir* é obtido diretamente da linha */iord* do barramento ISA, de modo que, durante os ciclos de leitura, os dados possam fluir dos registradores *rdL* e *rdH* para o processador ou para memória (no caso de leitura com DMA). A linha de controle */g* é fornecida pelo circuito lógico a partir da decodificação do endereço da placa.

4.2.3.2 - Projeto lógico

O projeto lógico do módulo de interface com o barramento ISA é dividido em duas unidades: uma para endereçamento da placa e outra para geração dos sinais de controle para este e para outros módulos do adaptador.

Unidade de endereçamento

A unidade de endereçamento realiza a decodificação do endereço da placa a partir da comparação do conteúdo das linhas *A9-0* do barramento de endereços com um endereço pré-definido. Essa comparação só é válida em ciclos normais de I/O. Durante ciclos de DMA, essas linhas são utilizadas no endereçamento da memória, enquanto que o periférico é endereçado pelo canal de DMA a ele reservado.

O endereço da placa adaptadora de comunicação para os nós de trabalho foi definido a partir do mapa de I/O do PC. Pela exclusão daqueles endereços reservados aos componentes e periféricos padrão (como portas paralela e serial), foram escolhidas algumas faixas de endereço passíveis de serem utilizadas no projeto. Além disso, como essa placa será utilizada em uma máquina paralela onde os nodos terão um número limitado de periféricos,

optou-se pelo uso de endereço programado em FPGA, ao invés de em micro-chaves. Isso restringe a portabilidade da placa, o que não é tão relevante; contudo, reduz o seu custo.

Dessa forma, reservou-se a faixa 250h-25Fh para o endereço da placa adaptadora dos nós de trabalho. Essa largura de faixa permite que sejam definidos 16 endereços internos para leitura e outros 16 para escrita, conforme será mostrado mais adiante. A faixa de endereço pode ser facilmente modificada pela reprogramação do FPGA.

A decodificação do endereço da placa é feita pela comparação de *A9-4* com o valor 25h. Além disso, a linha *AEN*, que indica a ocorrência de ciclos DMA, deve estar desabilitada, ou seja, em nível baixo. Para tal, vale o circuito mostrado na Figura 4.21, onde o sinal de saída */cs* (*card select*) indica a seleção da placa para ciclos normais de I/O.

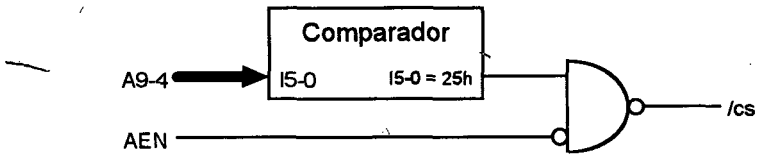


Figura 4.21 - Unidade de endereçamento do módulo de interface com o barramento ISA.

Unidade de controle

Essa unidade gera os sinais de controle esperados pelos módulo da placa, incluindo o de interface com o barramento ISA da qual ela faz parte. Esses sinais de controle são obtidos a partir da linha */cs*, proveniente da unidade de endereçamento, das linhas *A3-0* e de sinais de I/O e DMA, além do *reset* do barramento ISA.

O controle do estado dos *transceivers* de isolação do barramento é feito pelo sinal obtido pelo circuito da Figura 4.22.

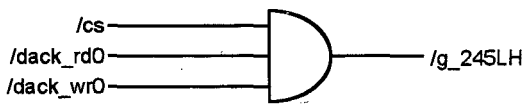


Figura 4.22 - Circuito para geração da linha de controle dos *transceivers* de isolação.

A saída */g_245LH* tornar-se-á ativa sempre que uma das entradas for para o nível baixo. A linha */cs* indica a ocorrência de um ciclo de leitura ou escrita em I/O. A entrada

/dack_rd0 sinaliza a realização de uma leitura por DMA no módulo interface com o *crossbar*, enquanto que */dack_wr0* indica uma escrita por DMA nesse mesmo módulo. Essas duas linhas são geradas em resposta às requisições */drq_rd0* e */drq_wr0* e podem ser configuradas para serem ligadas a um dos três canais de DMA para transferências de 16 *bits*. Tal configuração é feita por meio do uso de *jumpers*.

A linha */g_245LH* é ainda levada à entrada */io16* do barramento ISA, que é utilizada para indicar que a palavra de dado do ciclo de I/O em execução tem tamanho de 16 *bits*.

Os demais sinais de controle são gerados por meio de ciclos de escrita ou de leitura em I/O em endereços definidos dentro da faixa estabelecida (250h-25Fh). Eles são obtidos a partir das linhas */cs*, *A3-0*, */iord* e */iowr*, conforme a Tabela 4.1.

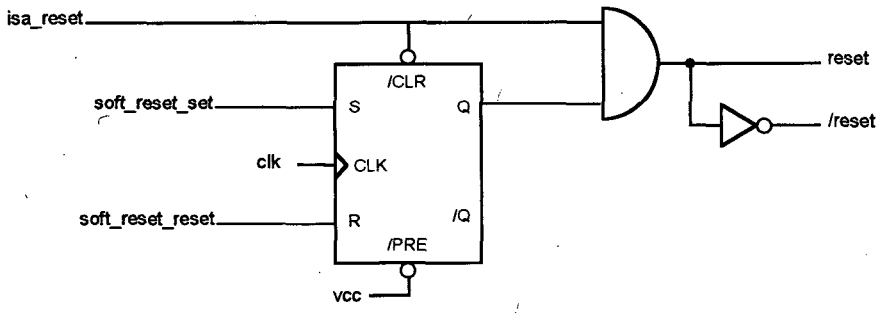
Tabela 4.1 - Sinais de controle gerados para os outros módulos do adaptador.

Sinal	/cs	A3-0	/dack _rd0	/dack _wr0	/iord	/iowr	Função
/iord0	0	0h	x	x	0	x	indica leitura no módulo de interface com o *crossbar*
	x	x	0	x	0	x	
/iowr0	0	0h	x	x	x	0	indica escrita no módulo de interface com o *crossbar*
	x	x	x	0	x	0	
/den_iord0	0	1h	x	x	0	x	habilita leitura com DMA no módulo de interface com o *crossbar*
/den_iowr0	0	1h	x	x	x	0	habilita escrita com DMA no módulo de interface com o *crossbar*
intr_TC_set	0	2h	x	x	0	x	desabilita interrupção para o nó de controle no módulo de interface com as linhas de interrupção
intr_TC_reset	0	2h	x	x	x	0	habilita interrupção para o nó de controle no módulo de interface com as linhas de interrupção
irq_CT_reset	0	3h	x	x	0	x	desabilita requisição de interrupção do nó de controle no módulo de interface com as linhas de interrupção

Legenda: x - irrelevante.

Além desses, um último sinal de controle, comum a todos os módulos da placa, é o de reinicialização do adaptador. Esse sinal pode ser gerado tanto por *hardware* como por *software*. Por *hardware* ele é obtido diretamente da linha *Reset* do barramento ISA. Já por

software, utiliza-se um *flip-flop* SR, no qual são realizados ciclos de leitura e de escrita sucessivos para gerar um pulso com largura superior à exigida pelo IMS C011 (1.6 μs). A Figura 4.23 apresenta o circuito para geração do sinal de reinicialização da placa. São fornecidas duas saídas, uma ativa em nível alto (*reset*) e outra em nível baixo (*/reset*), para suprir as necessidades dos diversos componentes.



Sinal	/cs	A3-0	/iord	/iowr	Função
soft_reset_set	0	Fh	0	x	habilita reinicialização por *software*
soft_reset_reset	0	Fh	x	0	desabilita reinicialização por *software*

Figura 4.23 - Circuito para geração do sinal de reinicialização do adaptador.

4.3 - Projeto do adaptador de comunicação para o nó de controle

O adaptador de comunicação para o nó de controle, mostrado anteriormente na Figura 4.4 apresenta quatro módulos de interface: um para comunicação via *crossbar*, um segundo para recepção e geração de interrupções para os nós de trabalho, outro para configuração do *crossbar* e, por último, um módulo para de interface com o barramento ISA. Esses quatro módulos são descritos abaixo.

4.3.1 - Módulo de interface com o *crossbar*

Esse módulo é semelhante àquele projetado para o adaptador de comunicação para os nós de trabalho. Porém, como o nó de controle utiliza o canal de comunicação via *crossbar* apenas para a troca de pequenas mensagens de controle com os nó de trabalho, não existe a necessidade do uso de transferências de dados por DMA. Assim, o projeto desse módulo é simplificado pela ausência dos circuitos para seleção das requisições de escrita e leitura. As saídas *obe0* e *ibf0* são ligadas, direta e respectivamente, às linhas para requisição de

interrupção *irq_wr0* e *irq_rd0*. Essa decisão de projeto simplifica também o módulo de interface com o barramento ISA, como será visto mais adiante.

4.3.1.1 - Projetos físico e lógico

O projeto físico desse módulo é idêntico àquele realizado para o adaptador dos nós de trabalho, mostrado em 4.2.1.1. Já o projeto lógico apresenta as mesmas máquinas de estados, menos os circuitos para seleção das requisições de escrita e de leitura. Por isso, nesta seção, mostra-se apenas a estrutura completa desse módulo de interface. O circuito da Figura 4.24 que é bastante semelhante ao da Figura 4.15.

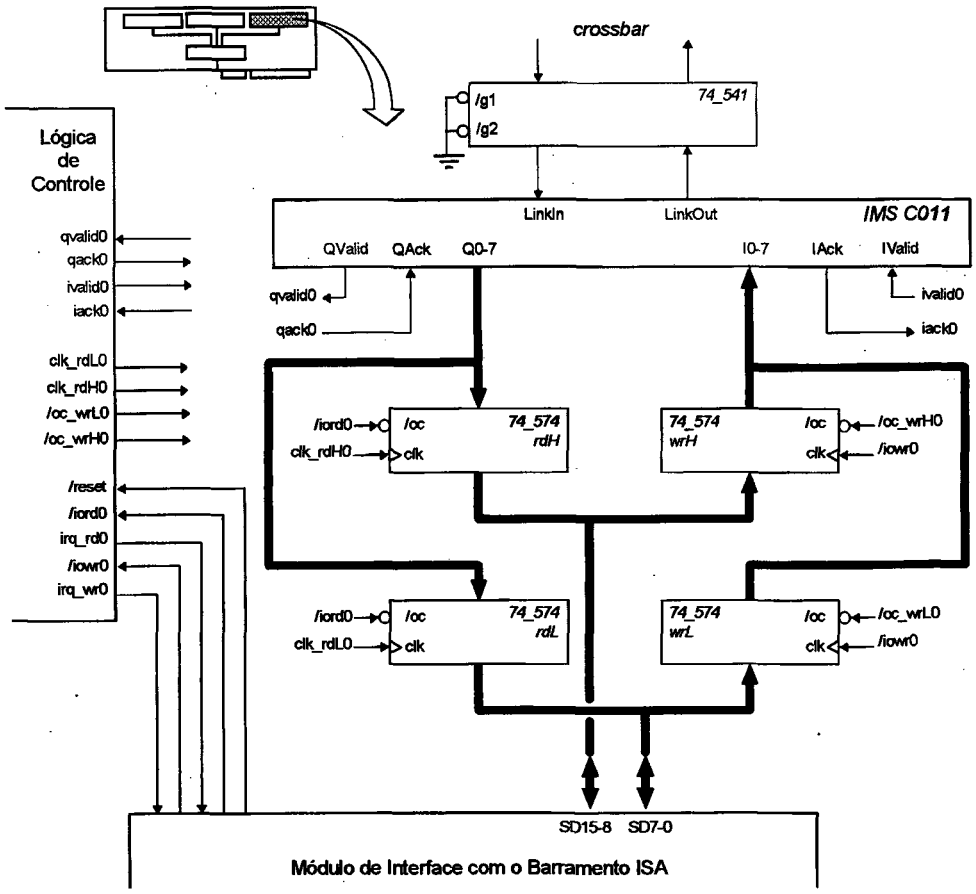


Figura 4.24 - Estrutura completa do módulo de interface com o *crossbar*.

4.3.2 - Módulo de interface com as linhas de interrupção

Esse módulo de interface apresenta oito canais bidirecionais de interrupção para com os nós de trabalho. Cada canal é formado por uma linha */intr_TC*, utilizada pelo nó de trabalho para interromper o nó de controle, e outra linha, chamada */intr_CT*, que permite ao nó de controle interromper o nó de trabalho. Por essas duas linhas são trocados pulsos ativos em nível baixo que representam requisições de interrupção.

4.3.2.1 - Projeto físico

Fisicamente, o módulo de interfaces com as linhas de interrupção é formado por duas unidades: uma de leitura (*rd*) e outra de escrita (*wr*). A unidade de leitura é formada por um *driver* 74_541, um *latch* 74_573 e uma porta *NAND* com oito entradas, implementada por um 74_30. Já a unidade de saída apresenta um registrador 74_574 e um *driver* 74_541, conforme a Figura 4.25.

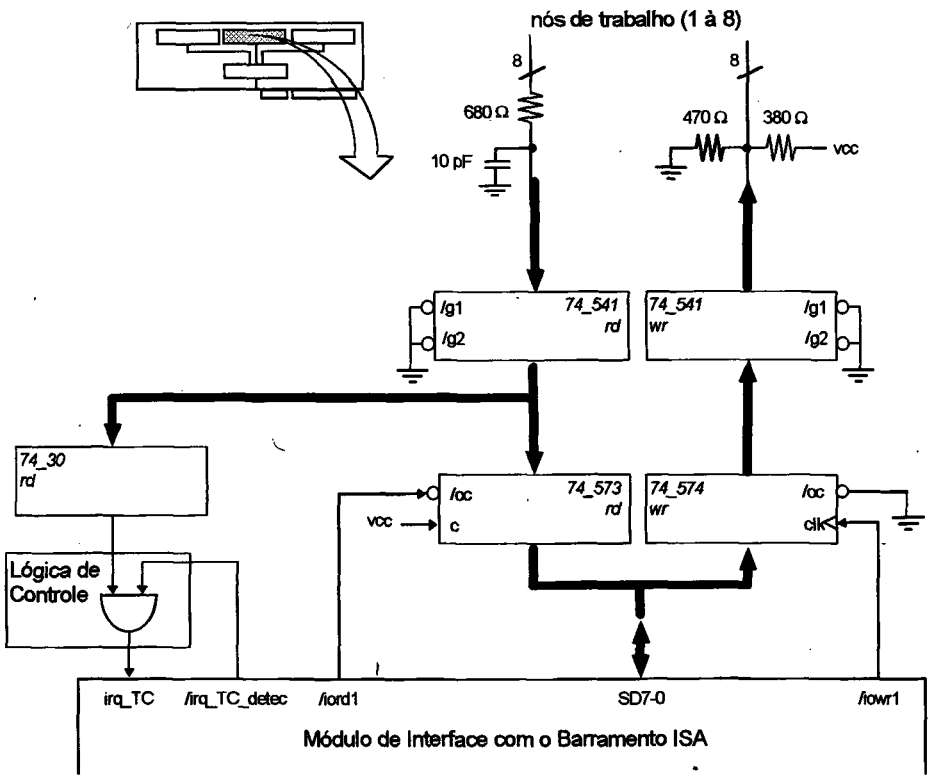


Figura 4.25 - Estrutura do módulo de interface com as linhas de interrupção.

Os *drivers* 74_541, utilizados nas duas unidades, reforçam os sinais de interrupção e minimizam os ruídos provocados pelo meio. Os demais componentes operam conforme a descrição a seguir.

Unidade de leitura

A unidade de leitura é utilizada para detectar a ocorrência de uma interrupção */intr_TC* e determinar o nó de trabalho que gerou tal interrupção.

A porta *NAND* atua de forma que, enquanto as linhas de interrupção estiverem inativas, a sua saída é mantida em nível baixo. No momento em que uma das linhas de entrada for para o nível baixo, caracterizando o início de uma interrupção */intr_TC*, a saída da porta lógica irá para o nível alto e a sua borda de subida produzirá uma requisição de interrupção *irq_CT* para o processador.

A rotina para o tratamento da interrupção *irq_CT* executa, inicialmente, um ciclo de leitura em I/O no endereço do módulo de interface com as linhas de interrupção. Esse ciclo de leitura leva a interface com o barramento ISA a produzir um pulso baixo na linha */iord1*[76], que controla a saída do *latch* 74_573.[77] Durante a leitura, a saída desse *latch,* normalmente em alta impedância, é acoplada ao barramento interno *SD7-0* e, conseqüentemente, às linhas de dado D7-0 do barramento ISA. Isso permite que o processador leia o vetor de interrupções apresentado na unidade de leitura.

Como diversas interrupções podem ser geradas simultaneamente, ou dentro de um pequeno intervalo de tempo, no momento em que o processador for realizar a leitura, pode haver mais de uma interrupção pendente. A rotina para o tratamento da interrupção *irq_TC* deve, então, selecionar um dos nós de trabalho para servir. Após a realização do serviço, é efetuado um ciclo de I/O em um endereço especifico, a fim de levar o módulo de interface com o barramento ISA a produzir um pulso *irq_TC_detect.* Esse pulso atua no circuito lógico, provocando, no caso de haver alguma interrupção pendente, uma borda de subida na linha *irq_CT.* Com isso, garante-se o atendimento de todas as interrupções, uma a uma.

---

[76] A terminação 1 deve-se ao fato desse módulo ser também referenciado de "módulo 1".

[77] A linha de controle *c* do *latch* 74_573, que habilita a saída a "acompanhar" a entrada, é amarrada ao *VCC*, o que a mantém permanentemente ativa.

Unidade de escrita

A unidade de escrita do módulo de interface com as linhas de interrupção apresenta um funcionamento mais simples que o da unidade de entrada.

Na partida ou na reinicialização do sistema, o processador do nó de controle realiza um ciclo de escrita em um endereço específico, levando o módulo de interface com o barramento ISA a gerar um pulso */iowr1*. Esse pulso atua na entrada de relógio (*clk*) do registrador 74_574, transferindo o dado apresentado no barramento *SD7-0* para a sua saída.[78] Esse dado deve ter o valor FFh, de modo que todas as linhas */intr_CT* sejam inicializadas no estado inativo.

No funcionamento normal da máquina, quando o nó de controle deseja interromper um ou mais nós de trabalho, ele deve escrever um dado no endereço da unidade de saída desse módulo. O dado deve apresentar os *bits* correspondentes aos nós de trabalho a serem interrompidos iguais a 0. Esse dado é registrado no 74_574 e mantido em sua saída até que um novo ciclo de escrita torne a mudar o seu conteúdo.

Para que as interrupções */int_CT* tenham a forma de um pulso ativo em nível baixo, após a escrita do dado de geração de uma interrupção, deve ser realizado um novo ciclo de escrita com o *bit* correspondente à interrupção gerada em 1. Por exemplo, para se interromper o oitavo nó de trabalho, devem ser realizados dois ciclos de escrita sucessivos. No primeiro, o dado deve ser igual a EFh, enquanto no segundo, a FFh.

4.3.2.2 - Projeto lógico

O projeto lógico desse módulo tem a função de evitar que a chegada de mais de uma interrupção na unidade leitura leve à paralisação do sistema, pelo atendimento de apenas uma das interrupções.

No funcionamento normal da máquina, toda vez que é finalizado o atendimento de uma interrupção, a rotina para o seu tratamento leva a interface com o barramento ISA a produzir um pulso ativo em nível baixo, chamado *intr_CT_detect*. Esse pulso entra na lógica de controle e, havendo alguma requisição pendente na saída do 74_30, gera uma nova requisição *irq_CT* para o processador, conforme a Figura 4.26.

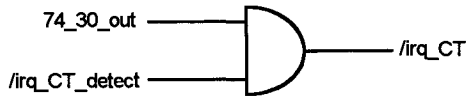


Figura 4.26 - Circuito lógico de controle do módulo de interface com as linhas de interrupção.

4.3.3 - Módulo de interface para configuração do *crossbar*

Esse módulo consiste numa simplificação do módulo de interface com o *crossbar*. Tal simplificação deriva do fato de que as mensagens para a configuração do *crossbar* são muito curtas, com no máximo quatro *bytes*, e nem sempre apresentam um número par de *bytes*. Assim, esse módulo foi projetado com apenas um registrador para escrita (*wrL*) e outro para leitura (*rdL*).[79] Além disso, o uso de DMA para a transferência de dados é totalmente desnecessário, face ao tamanho reduzido dessas mensagens.

4.3.3.1 - Projeto físico

O projeto físico do módulo de interface para a configuração do *crossbar* é mostrado na Figura 4.27. Existe um *driver* de linha 74_541, um adaptador de ligação IMS C011 e dois registradores 74_574, um para escrita e outro para leitura. O tamanho do dado é de 8 *bits*, enquanto que no outro módulo é de 16 *bits*.

---

[78] A saída do registrador é habilitada permanentemente pela ligação da linha de controle */oc* ao terra.

[79] Mantém-se o uso da terminação *L* no nome dos registradores e de seus sinais para garantir um padrão de nomenclatura.

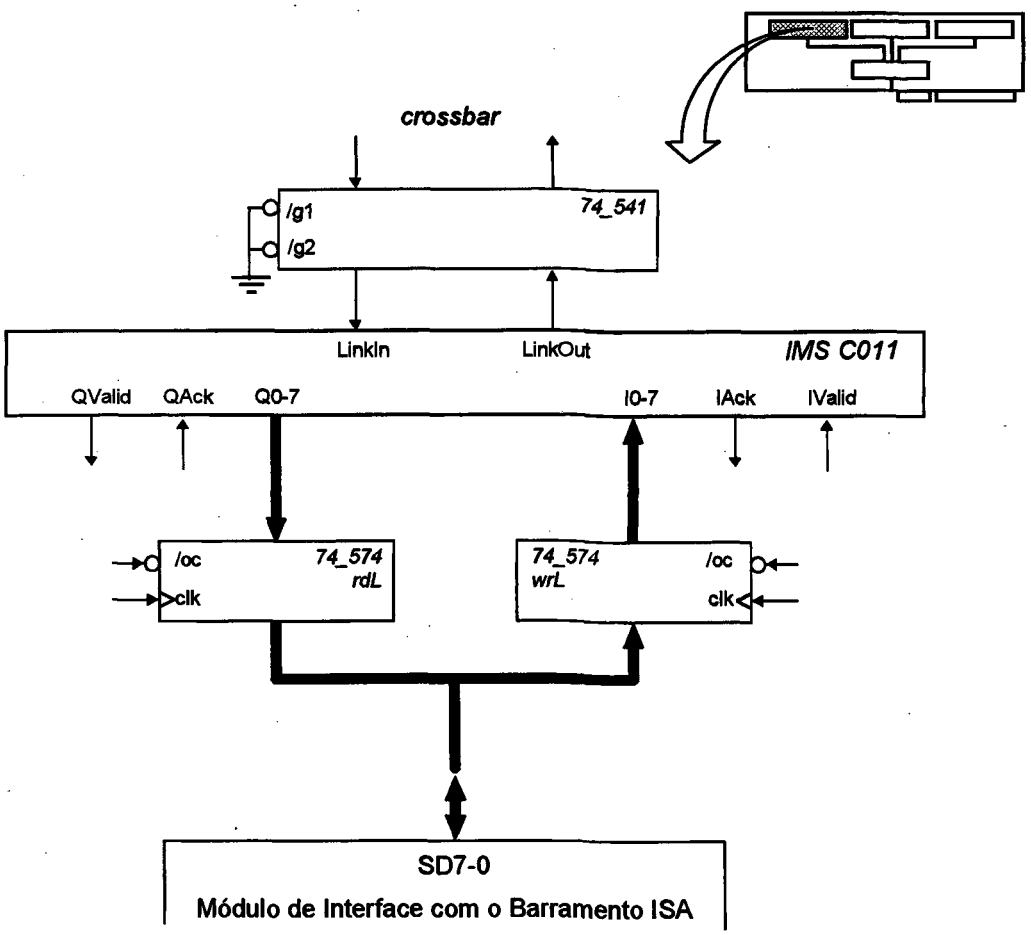


Figura 4.27 - Estrutura física do módulo de interface para configuração do *crossbar*.

4.3.3.2 - Projeto lógico

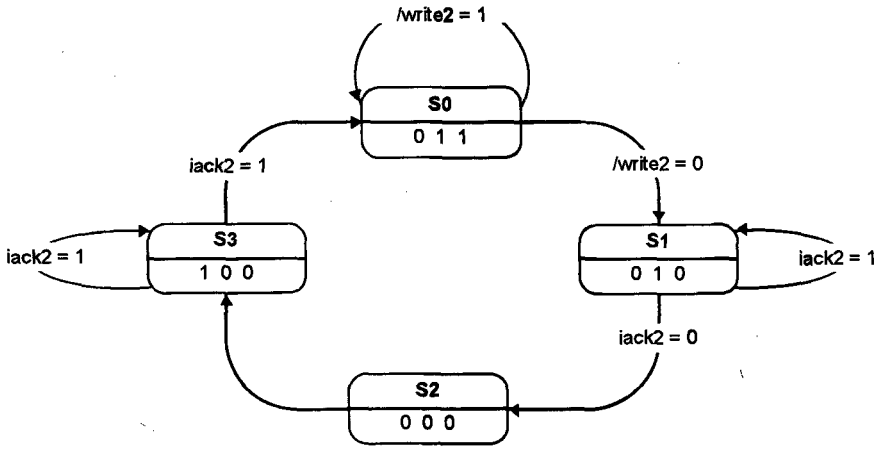
No projeto lógico desse módulo de interface são definidas duas máquinas de estados, uma para escrita e outra para leitura. A primeira é usada na transmissão das mensagens de configuração do *crossbar*, enquanto a segunda possibilita o recebimento de mensagens que informam o estado de conexão de uma saída. Essas duas máquinas são apresentadas a seguir.

Máquina de escrita

Nessa máquina, o estado das saídas *ivalid2*, */oc_wrL2* e *obe2* é condicionado às entradas */write2*, derivada de */iowr2*, e *iack2*[80]. O seu funcionamento, apresentado na Figura

[80] A terminação 2 é utilizada para identificar que estes sinais referem-se ao módulo de interface para configuração do *crossbar*, também chamado de "módulo 2".

4.28, é bastante semelhante ao da máquina de escrita para o módulo de interface com o *crossbar*, da qual ela deriva.



Saídas:
ivalid2 /oc_wrL2 obe2

Estados:
- **S0**: esperando escrita do dado;
- **S1**: esperando fim da transmissão do dado anterior;
- **S2**: colocando *byte* do dado no barramento de entrada I0-7;
- **S3**: esperando reconhecimento do *byte*;

Figura 4.28 - Máquina de estados de escrita para transmissão de dados pelo módulo de interface para configuração do *crossbar*.

A escrita de um dado no registrador *wrL* leva a máquina, do estado inicial S0, para o estado S1. Nesse estado, ela conduz a saída *obe2* ao nível baixo e fica aguardando o fim da transmissão do *byte* anterior. Quando isso acontece, com a descida de *iack2*, ocorre uma transição para o estado S2, onde */oc_wrL* é levado ao nível baixo. No ciclo seguinte do relógio, a máquina vai para o estado S3 e a saída *ivalid2* é ativada, iniciando a transmissão do dado.

O ciclo se encerra com a chegada do pacote de reconhecimento para o *byte* transmitido, quando, então, a linha *iack2* vai para o nível alto. A máquina retorna ao estado inicial, desativando as linhas *ivalid2* e */oc_wrL2* e gerando uma requisição para uma nova escrita, por meio da subida de *obe2*. Essa saída, por sua vez, é conduzida à linha de interrupção *irq_wr2*.

Além disso, é utilizado um circuito idêntico ao mostrado na Figura 4.7 para registrar a ocorrência de um pulso de escrita */iowr2*.

Máquina de leitura

Na máquina de leitura, as saídas *qack2*, *clk_rdL2* e *ibf2* têm seus estados condicionados às entradas */read2*, derivada de */iord2*, e *qvalid2*. A sua operação, mostrada na Figura 4.29, assemelha-se ao da máquina de leitura para o módulo de interface com o *crossbar*, da qual ela deriva.

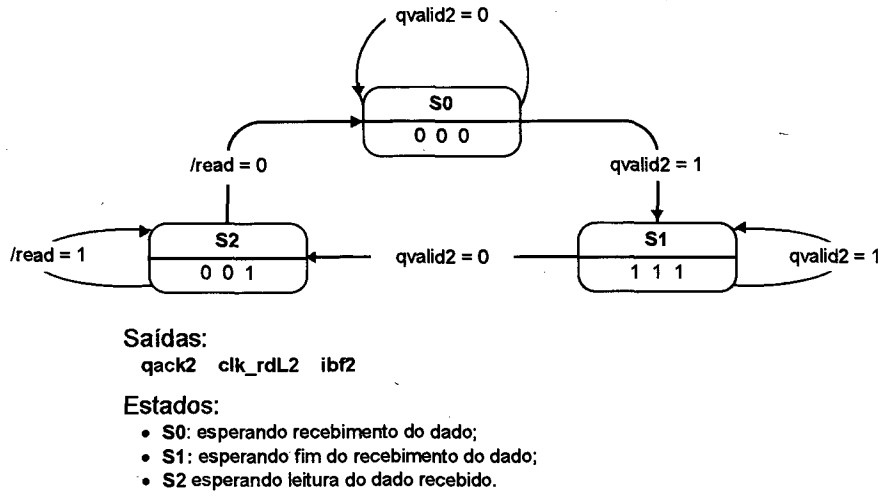


Figura 4.29 - Máquina de estados de leitura para recepção de dados pelo módulo de interface para configuração do *crossbar*.

Essa máquina inicia-se no estado S0, onde todas as saídas estão desativadas. Quando um *byte* é recebido pelo IMS C011 e a linha qvalid*2* vai para o nível alto, a máquina transita para o estado S1. Nesse estado, as saídas *qack2* e *clk_rdL2* são ativadas, realizando, respectivamente, o registro e o reconhecimento do *byte* recebido.

Quando *qvalid2* retorna ao nível baixo, finalizando a transferência do dado pelo IMS C011, a máquina evolui para o estado S2, onde *qack2* e *clk_rdL2* são desabilitadas e *ibf2* é levada ao nível alto. Essa última é amarrada à linha de interrupção *irq_rd2* e a sua subida produz uma requisição para a leitura do dado recebido. O ciclo se encerra com a leitura do dado pelo processador. A máquina volta para o estado inicial e torna-se apta a receber um novo dado.

Para essa máquina de estados, a linha de leitura */iord2* tem sua ocorrência registrada por um circuito idêntico ao da Figura 4.10.

A estrutura completa desse módulo de interface, com seus componentes e sinais de controle, é mostrada na Figura 4.30.

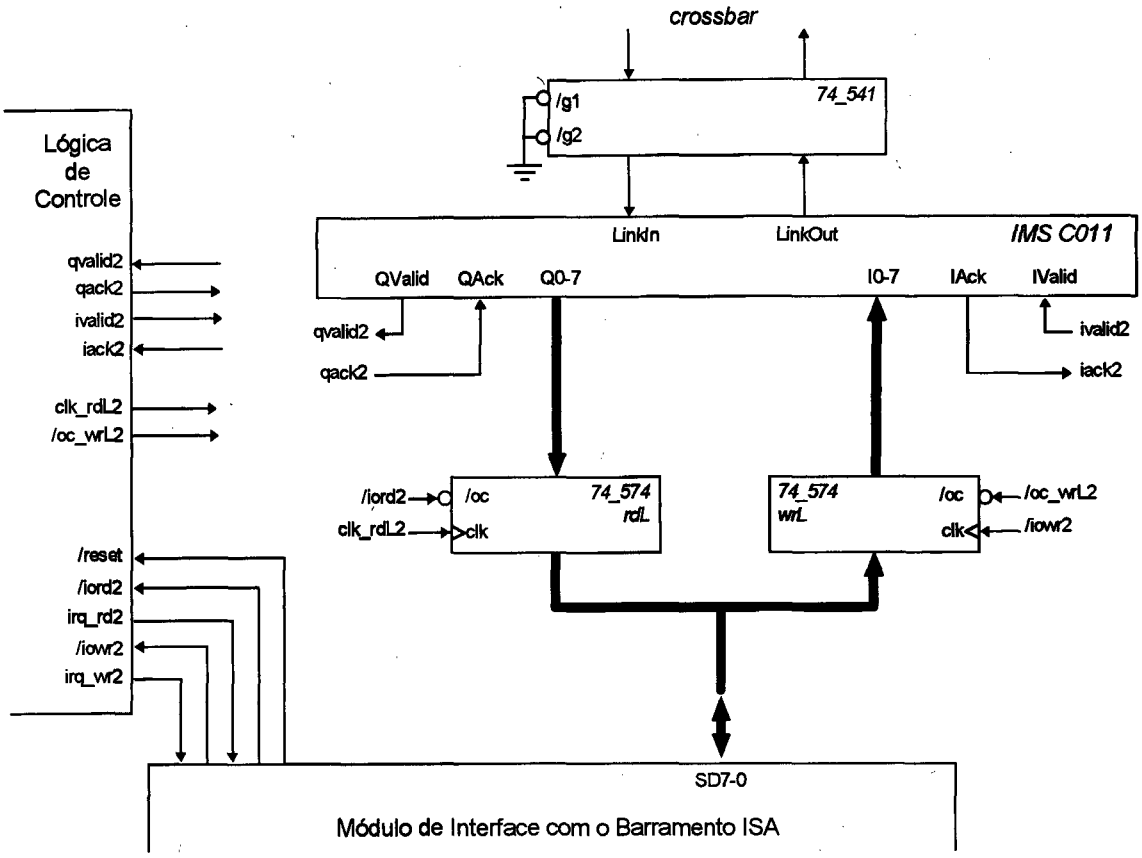


Figura 4.30 - Estrutura completa do módulo de interface para configuração do *crossbar*.

4.3.4 - Módulo de interface com o barramento ISA

O módulo de interface com o barramento ISA para o adaptador de comunicação do nó de controle é fisicamente idêntico e logicamente semelhante àquele projetado para o adaptador de comunicação dos nós de trabalho. Por isso, a presente descrição restringe-se em mostrar os aspectos diferenciais entre os dois módulos.

4.3.4.1 - Projeto lógico

A diferença entre os módulos de interface com o barramento ISA para os dois tipos de adaptador de comunicação está nos sinais de controle que são gerados para as outras interfaces da placa.

Para este adaptador, foram definidos os sinais mostrados na Tabela 4.2. Esses sinais são obtidos a partir função das linhas */cs*, *A3-0*, */iord* e */iowr*, dentro de um faixa de endereços previamente definida (280h-28Fh).

Tabela 4.2 - Sinais de controle gerados para os outros módulos do adaptador.

Sinal	/cs	A3-0	/iord	/iowr	Função
/iord0	0	0h	0	x	indica leitura no módulo de interface com o *crossbar*
/iowr0	0	0h	x	0	indica escrita no módulo de interface com o *crossbar*
/iord1	0	2h	0	x	indica leitura no módulo de interface com as linhas de interrupção
/iowr1	0	2h	x	0	indica escrita no módulo de interface com as linhas de interrupção
/irq_TC_detect	0	3h	0	x	detecta a ocorrência de uma requisição de interrupção pendente
/iord2	0	4h	0	x	indica leitura no módulo de interface para configuração do *crossbar*
/iowr2	0	4h	x	0	indica escrita no módulo de interface para configuração do *crossbar*

Legenda: x - irrelevante.

4.4 - Resumo

Esse capítulo apresentou um detalhamento dos projetos físico e lógico dos adaptadores de comunicação para os nós de trabalho e de controle que integram o sistema de comunicação do multicomputador Nó//. No capítulo seguinte, serão apresentados resultados de experimentos de simulação realizados sobre o projeto lógico do adaptador de comunicação dos nós de trabalho.

CAPÍTULO 5

SIMULAÇÃO DO PROJETO LÓGICO

Introdução

Este capítulo apresenta resultados de experimentos de simulação realizados sobre a lógica de controle e endereçamento projetada para o adaptador de comunicação dos nós de trabalho. Tais simulações foram feitas no ambiente de desenvolvimento MAX+PLUS II da Altera. O projeto lógico simulado consiste em um conjunto de sub-projetos nos quais são programadas, através do uso da linguagem de descrição de *hardware* Altera (AHDL), as lógicas combinacionais e seqüenciais a serem integradas em um único dispositivo lógico programável. Esses sub-projetos estão incluídos no anexo A.

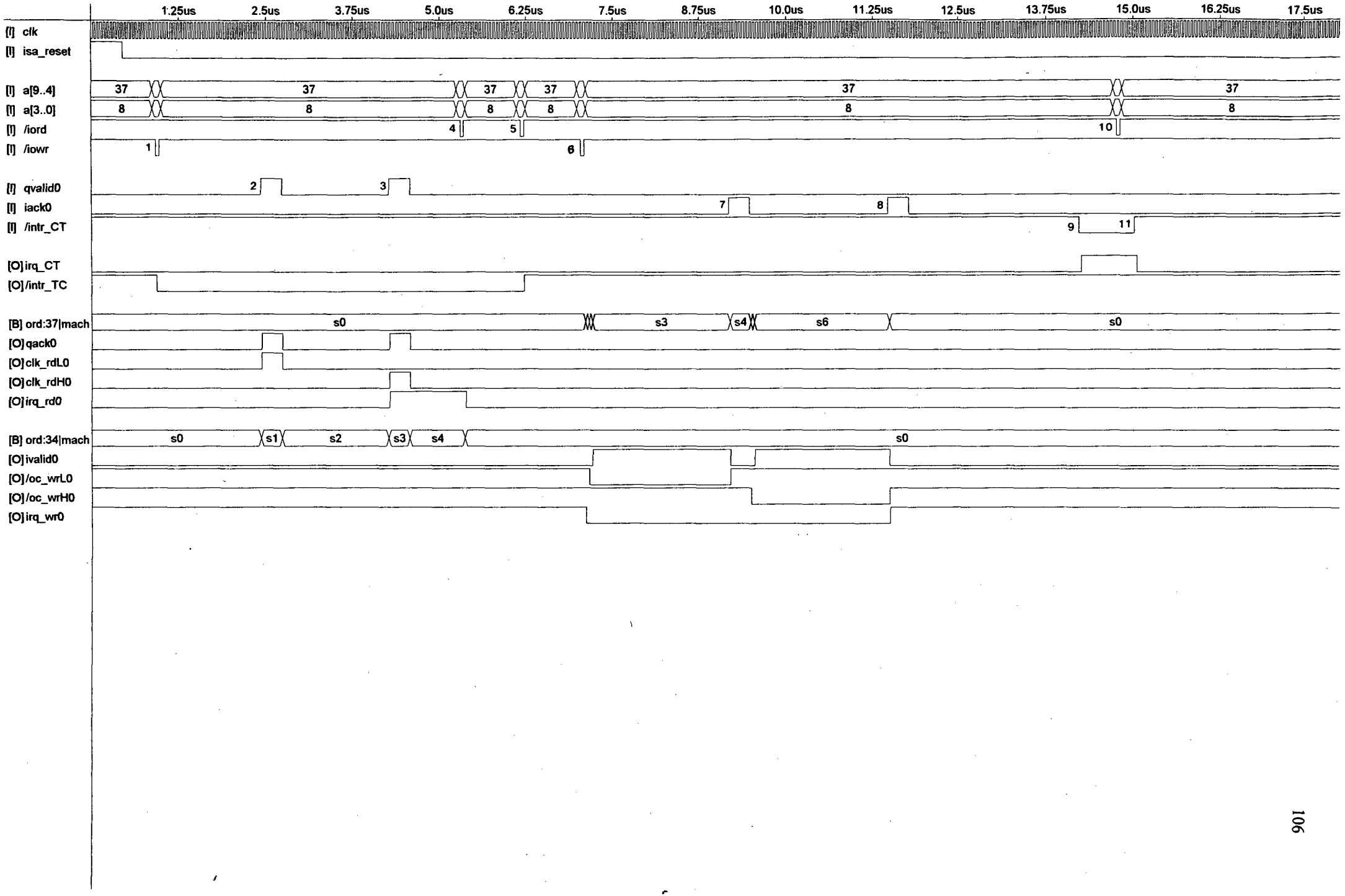
Os experimentos de simulação baseiam-se na aplicação de um vetor de entrada ao projeto lógico e na obtenção de um vetor de saída. A interface gráfica do ambiente permite que esses vetores sejam visualizados como formas de onda. Os diagramas a seguir apresentam alguns resultados obtidos dentro de um conjunto de experimentos realizados para a verificação e validação do projeto lógico. São mostrados:

- o mecanismo solicitação de uma conexão e confirmação de seu estabelecimento;
- a recepção e a transmissão de mensagens de dados através de uma conexão estabelecida por meio de ciclos normais de I/O e de ciclos de I/O com acesso direto à memória;
- reinicialização do adaptador de comunicação via *software*.

5.1 - Estabelecimento de uma conexão

O diagrama da página 106 representa o mecanismo de estabelecimento de uma conexão solicitada por um nó de trabalho. Os instantes relevantes ao seu entendimento são destacados e descritos abaixo:

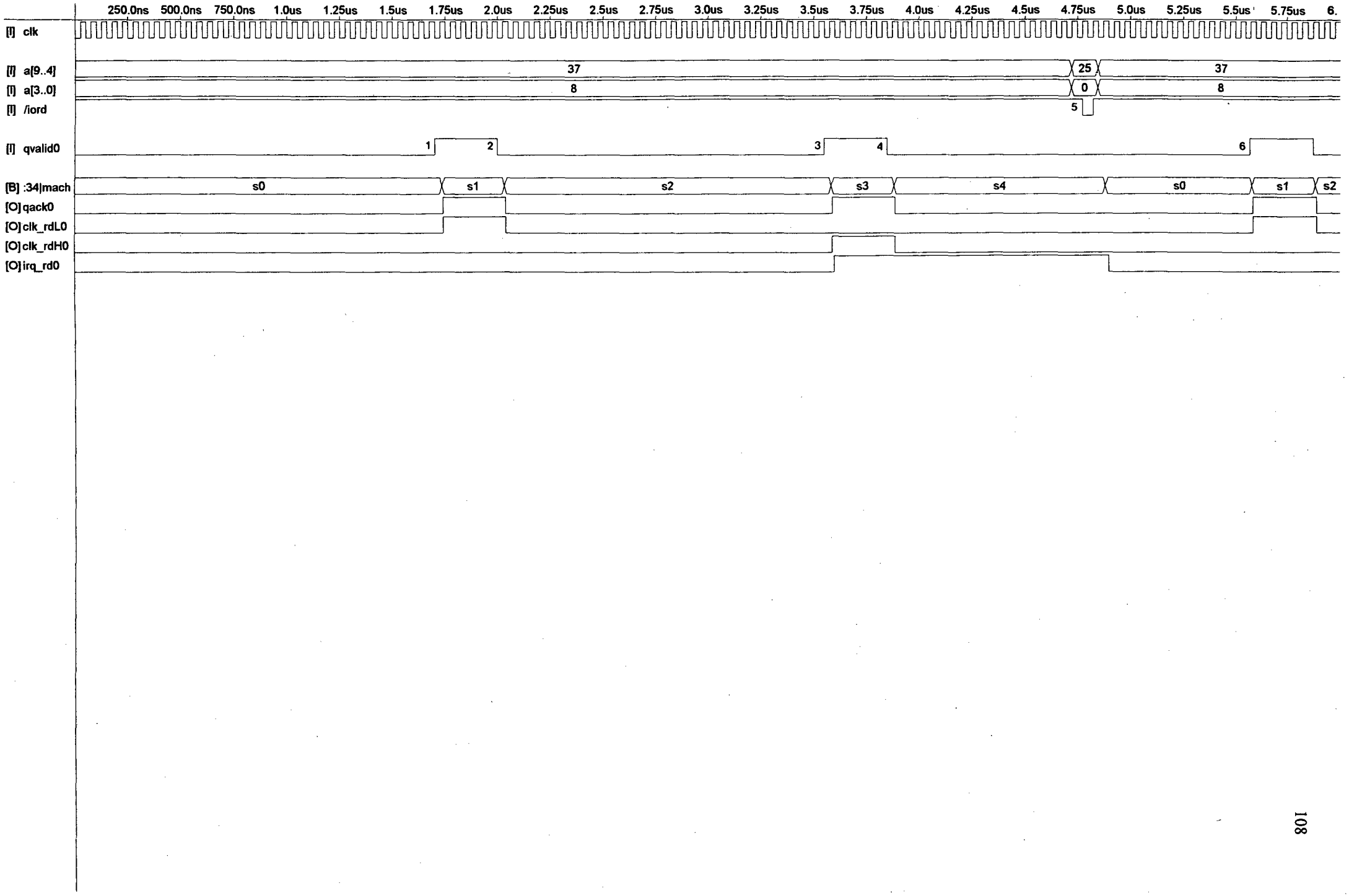
1. O processador do nó de trabalho realiza um ciclo de escrita no endereço 252h, iniciando a geração de um pulso de interrupção */intr_TC*.
2. O módulo de interface com o *crossbar* recebe o primeiro *byte* da mensagem de consulta enviada pelo nó de controle em resposta à interrupção gerada;
3. O módulo de interface com o crossbar recebe o segundo e último *byte* da mensagem de consulta enviada pelo nó de controle;
4. O processador do nó de trabalho realiza um ciclo de leitura no endereço 250h para ler o dado enviado pelo nó de controle;
5. O processador do nó de trabalho realiza um ciclo de escrita no endereço 252h, desabilitando o pulso */intr_TC;*
6. O processador do nó de trabalho realiza um ciclo de escrita no endereço 250h a fim de enviar a sua requisição de conexão para o nó de controle;
7. O primeiro *byte* da mensagem de requisição é transmitido ao nó de controle;
8. O segundo e último *byte* da mensagem de requisição é transmitido ao nó de controle;
9. O módulo de interface com as linhas de interrupção recebe a interrupção */intr_CT* do nó de controle, que confirma o estabelecimento da conexão solicitada. Então, ele gera uma requisição *irq_CT* ao processador;
10. Em resposta à *irq_CT*, o processador realiza um ciclo de leitura no endereço 253h, visando desabilitar a requisição recebida; e
11. O pulso da interrupção */intr_CT* chega ao seu fim, possibilitando a desabilitação de *irq_CT*.


1.25us
2.5us
3.75us
5.0us
6.25us
7.5us
8.75us
10.0us
11.25us
12.5us
13.75us
15.0us
16.25us
17.5us
[I] clk
[I] isa_reset
[I] a[9..4]
[I] a[3..0]
[I] /iord
[I] /iowr
[I] qvalid0
[I] iack0
[I] /intr_CT
[O] irq_CT
[O] /intr_TC
[B] ord:37|mach
[O] qack0
[O] clk_rdL0
[O] clk_rdH0
[O] irq_rd0
[B] ord:34|mach
[O] ivalid0
[O] /oc_wrL0
[O] /oc_wrH0
[O] irq_wr0
37
8
1
2
3
4
5
6
7
8
9
10
11
s0
s1
s2
s3
s4
s6

5.2 - Recepção de dados

O diagrama da página 108 representa a recepção de mensagens de dados através de ciclos normais de I/O utilizando uma conexão preestabelecida. Os instantes relevantes ao seu entendimento são destacados e descritos abaixo:

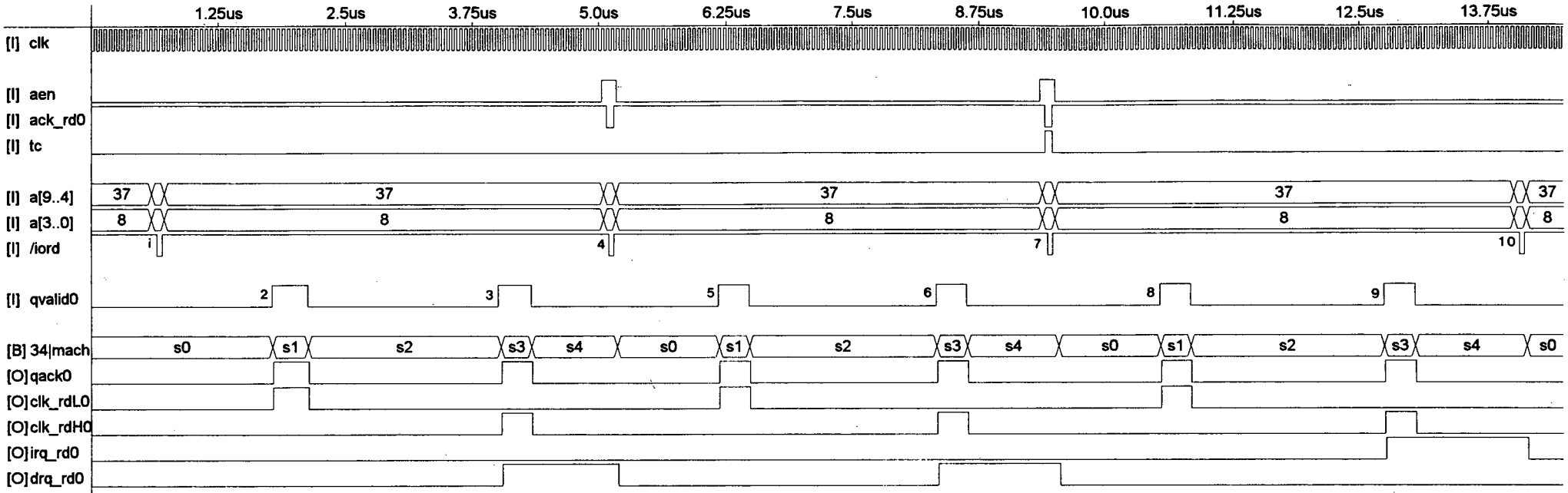
1. O IMS C011 recebe o primeiro *byte* de um dado e leva a linha *qvalid0* para o nível alto. Em resposta, a máquina de estados de leitura ativa as linhas *clk_rdL0* e *qack0*, registrando e reconhecendo o *byte* recebido;
2. Com a subida de *qack0*, o IMS C011 envia o pacote de reconhecimento e desativa a linha *qvalid0*. A máquina de leitura faz descer as linhas de saída previamente ativadas;
3. O IMS C011 recebe o segundo *byte* do dado e leva a linha *qvalid0* novamente para o nível alto. Em resposta, a máquina de estados de leitura reativa a linha *qack0*, bem como *clk_rdH0* e *irq_rd0*, realizando, respectivamente, o reconhecimento e o registro do *byte* e a requisição para leitura do dois *bytes* recebidos;
4. Com a subida de *qack0*, o IMS C011 envia o pacote de reconhecimento e desativa a linha *qvalid0*. A máquina de leitura faz descer as linhas de saída *qack0* e *clk_rdH0* ;
5. O processador realiza um ciclo de leitura no endereço 250h, pelo qual ele lê o dado recebido pelo IMS C011. Nesse momento, *irq_rd0* retorna para o nível baixo; e
6. Um novo dado começa a ser recebido.


250.0ns 500.0ns 750.0ns 1.0us 1.25us 1.5us 1.75us 2.0us 2.25us 2.5us 2.75us 3.0us 3.25us 3.5us 3.75us 4.0us 4.25us 4.5us 4.75us 5.0us 5.25us 5.5us 5.75us 6.
[I] clk
[I] a[9..4]
37
25
37
[I] a[3..0]
8
0
8
[I] /iord
5
[I] qvalid0
1
2
3
4
6
[B] :34|mach
s0
s1
s2
s3
s4
s0
s1
s2
[O] qack0
[O] clk_rdL0
[O] clk_rdH0
[O] irq_rd0

5.3 - Recepção de dados com DMA

O diagrama da página 110 representa a recepção de mensagens de dados através ciclos de I/O com acesso direto à memória utilizando uma conexão preestabelecida. Os instantes relevantes ao seu entendimento são destacados e descritos abaixo:

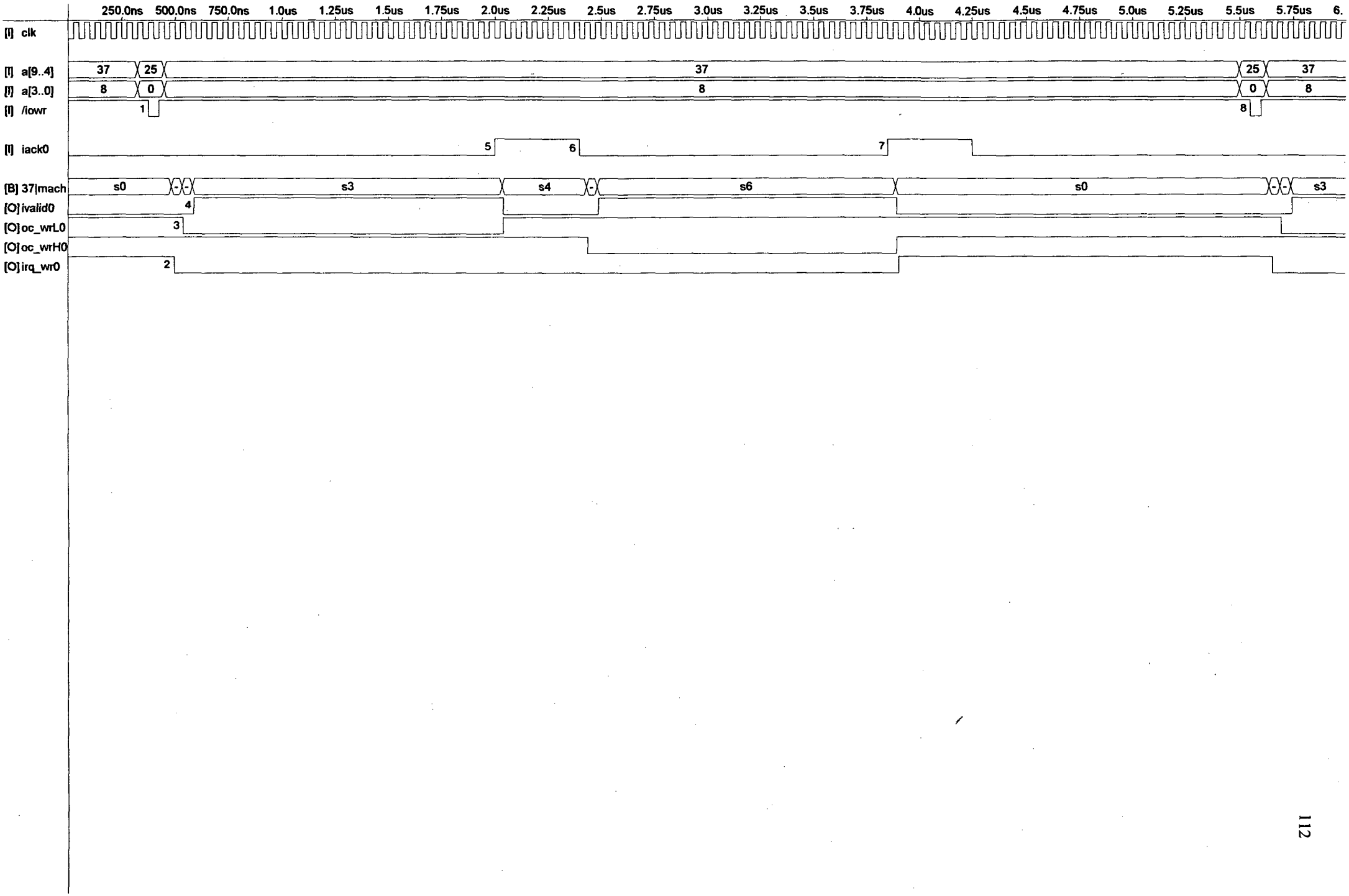
1. O processador realiza um ciclo de leitura no endereço 251h, habilitando a recepção de dados com DMA pelo módulo de interface com o *crossbar.*
2. O IMS C011 recebe o primeiro *byte* do primeiro dado da mensagem a ser transferida por DMA. A máquina de estados registra e reconhece o recebimento do *byte*;
3. O IMS C011 recebe o segundo *byte* do primeiro dado da mensagem. A máquina de estado registra e reconhece o recebimento do *byte*. Além disso, envia um requisição de leitura *drq_rd0* ao controlador de DMA;
4. Em resposta ao recebimento da *drq_rd0* o controlador de DMA realiza um ciclo de leitura em I/O e escrita em memória, endereçando o módulo de interface com o *crossbar* através da linha *dack_rd0*;
5. O IMS C011 recebe o primeiro *byte* do último dado da mensagem a ser transferida por DMA;
6. O IMS C011 recebe o segundo *byte* do último dado da mensagem;
7. O controlador de DMA realiza um ciclo de leitura em I/O e escrita em memória, endereçando o módulo de interface com o *crossbar* através da linha *dack_rd0*. Também ativa a linha *tc* durante a leitura do dado, indicando o fim da transferência e desabilitando a leitura por DMA.
8. O IMS C011 recebe o primeiro *byte* de uma nova mensagem, sem DMA;
9. O IMS C011 recebe o segundo *byte* da mensagem e gera uma *irq_rd0* ao processador; e
10. Em resposta, o processador realiza um ciclo de leitura no endereço 250h.


1.25us
2.5us
3.75us
5.0us
6.25us
7.5us
8.75us
10.0us
11.25us
12.5us
13.75us
[I] clk
[I] aen
[I] ack_rd0
[I] tc
[I] a[9..4]
[I] a[3..0]
[I] /iord
[I] qvalid0
[B] 34|mach
[O] qack0
[O] clk_rdL0
[O] clk_rdH0
[O] irq_rd0
[O] drq_rd0
37
8
s0
s1
s2
s3
s4

5.4 - Transmissão de dados

O diagrama da página 112 representa a transmissão de mensagens de dados através de ciclos normais de I/O utilizando uma conexão preestabelecida. Os instantes relevantes ao seu entendimento são destacados e descritos abaixo:

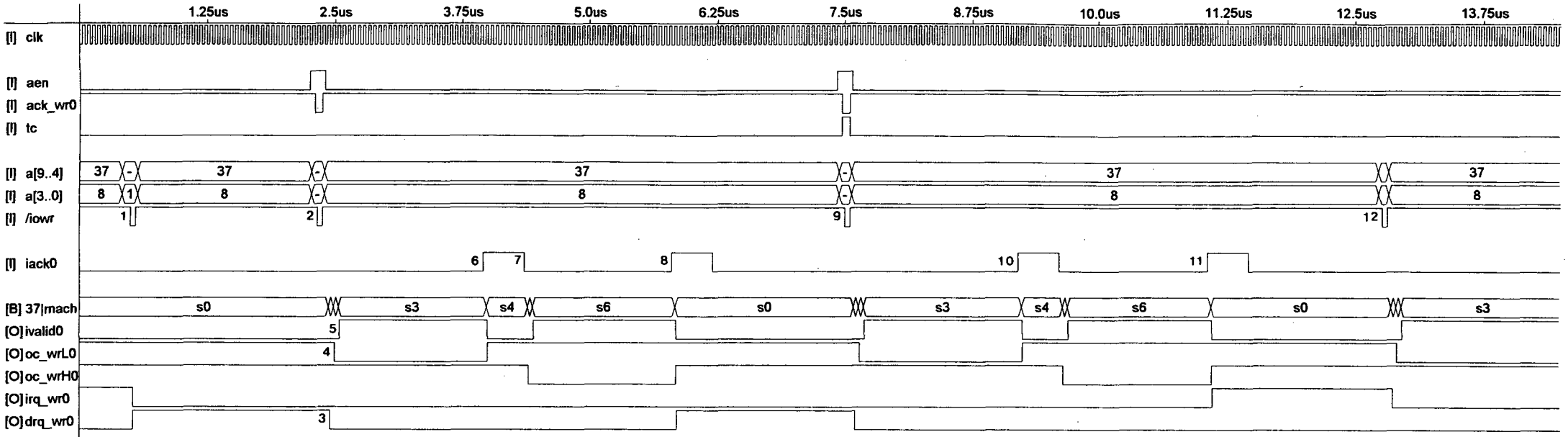
1. O processador realiza um ciclo de escrita em I/O no endereço 250h, escrevendo os dois *bytes* do dado no módulo de interface com *crossbar*;
2. A máquina de escrita faz a linha *irq_wr0* ir para o nível baixo, suspendendo a requisição para escrita nesse módulo;
3. A máquina de escrita habilita a saída do registrador *wrL*, colocando o primeiro *byte* do dado na entrada do IMS C011;
4. A máquina de escrita ativa a linha *ivalid0*, iniciando a transmissão do dado;
5. O IMS C011 recebe o pacote de reconhecimento para o *byte* enviado e ativa a linha *iack0*. Com isso, a máquina de estados desabilita as linhas */oc_wrL0* e *ivalid0*;
6. O IMS C011 desativa a linha *iack0*, finalizando a transmissão do primeiro *byte*. A máquina de estados reativa a linha *ivalid0*, iniciando a transmissão do *byte* armazenado em *wrH*; e
7. O IMS C011 reativa a linha *iack0*, indicando o recebimento do dado enviado. A máquina de estados desabilita as linhas *ivalid0* e */oc_wrL0*; e faz a linha *irq_wr0* ir para o nível alto, requisitando um nova escrita, realizada em 8.


250.0ns 500.0ns 750.0ns 1.0us 1.25us 1.5us 1.75us 2.0us 2.25us 2.5us 2.75us 3.0us 3.25us 3.5us 3.75us 4.0us 4.25us 4.5us 4.75us 5.0us 5.25us 5.5us 5.75us 6.
[I] clk
[I] a[9..4]
[I] a[3..0]
[I] /iowr
[I] iack0
[B] 37|mach
[O] ivalid0
[O] oc_wrL0
[O] oc_wrH0
[O] irq_wr0
37
25
37
25
37
8
0
8
0
8
s0
s3
s4
s6
s0
s3

5.5 - Transmissão de dados com DMA

O diagrama da página 114 representa a transmissão de mensagens de dados através ciclos de I/O com acesso direto à memória utilizando uma conexão preestabelecida. Os instantes relevantes ao seu entendimento são destacados e descritos abaixo:

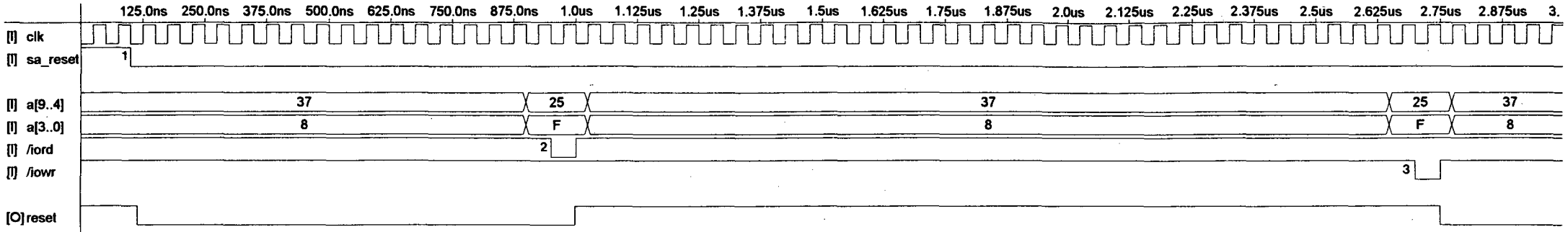
1. O processador realiza um ciclo de escrita no endereço 251h, habilitando a transferência de dados com DMA pelo módulo de interface com o *crossbar*. A linha *irq_wr0* vai para o nível baixo, enquanto *drq_wr0* é ativada, solicitando a realização de um ciclo de escrita em I/O com DMA;
2. O controlador de DMA endereça o módulo de interface com *crossbar* por meio da linha */dack_wr0* e realiza um ciclo de leitura em memória e escrita em I/O. Com isso, ele transfere o primeiro dado da mensagem a ser enviada por DMA através desse módulo;
3. A máquina de escrita desativa a requisição *drq_wr0*;
4. A máquina de escrita apresenta o *byte* armazenado no registrador *wrL* ao IMS C011;
5. A máquina de escrita ativa a linha *ivalid0*, iniciando a transmissão do primeiro *byte*;
6. O IMS C011 reconhece o recebimento do primeiro *byte* transmitido;
7. O IMS C011 finaliza a transmissão do primeiro *byte* e a máquina de escrita inicia o envio do segundo;
8. O IMS C011 reconhece o recebimento do segundo *byte* enviado e a máquina de escrita requisita uma nova escrita com DMA; e
9. O controlador realiza a escrita do último dado a ser enviado por DMA. Após, ocorrem ciclos normais de escrita em I/O (10, 11, 12).


1.25us
2.5us
3.75us
5.0us
6.25us
7.5us
8.75us
10.0us
11.25us
12.5us
13.75us
[I] clk
[I] aen
[I] ack_wr0
[I] tc
[I] a[9..4]
[I] a[3..0]
[I] /iowr
[I] iack0
[B] 37|mach
[O] ivalid0
[O] oc_wrL0
[O] oc_wrH0
[O] irq_wr0
[O] drq_wr0
37
8
s0
s3
s4
s6

5.6 - Reinicialização por *software*

O diagrama da página 116 representa a reinicialização do adaptador de comunicação por *software*. Os instantes relevantes ao seu entendimento são destacados e descritos abaixo:

1. Após a partida ou reinicialização do sistema, a saída *reset vai para o estado inativo;*
2. O processador inicia uma reinicialização por *software* através da leitura no endereço 25Fh; e
3. O processador termina uma reinicialização por *software* pela da escrita no endereço 25Fh;


125.0ns 250.0ns 375.0ns 500.0ns 625.0ns 750.0ns 875.0ns 1.0us 1.125us 1.25us 1.375us 1.5us 1.625us 1.75us 1.875us 2.0us 2.125us 2.25us 2.375us 2.5us 2.625us 2.75us 2.875us 3.
[I] clk
[I] sa_reset
1
[I] a[9..4]
37
25
37
25
37
[I] a[3..0]
8
F
8
F
8
[I] /iord
2
[I] /iowr
3
[O] reset

CONCLUSÃO

Nesse trabalho foi apresentado o projeto do sistema de comunicação de um multicomputador com rede de interconexão dinâmica. A referida máquina é parte integrante do Projeto Nó//, que objetiva a construção de um ambiente completo para programação paralela.

Inicialmente, foi feita uma revisão da literatura, a partir de um estudo sobre as arquiteturas de computadores que visam ao alto desempenho. Após, apresentou-se o projeto Nó//, descrevendo-se os aspectos relacionados ao multicomputador que servirá de plataforma física ao ambiente de programação paralela. No terceiro capítulo, foram mostradas algumas decisões de projeto tomadas no sentido de viabilizar o desenvolvimento de um protótipo inicial do multicomputador.

O quarto capítulo descreveu o projeto dos adaptadores que integram o sistema de comunicação do multicomputador. Esses adaptadores provêem meios para implementação de um mecanismo para a troca de mensagens entre os nodos da máquina. Foram projetados dois tipos de adaptador. O primeiro, destinado aos nós de trabalho, apresenta módulos de interface para com o *crossbar* e o mecanismo de controle baseado em linhas de interrupção. O segundo, oferece, além dessas duas, uma interface para o nó de controle programar o *crossbar* e realizar a reconfiguração da rede. Foram detalhados os projetos físico e lógico dos módulos de interface que compõe cada um dos adaptadores mencionados

Por fim, no quinto capítulo, foi apresentado um conjunto de resultados de simulação obtidos a partir de experimentos realizados sobre o projeto lógico descrito no capítulo anterior.

O projeto apresentado nesse texto encontra-se completo e pronto para implementação, restando apenas a conclusão dos diagramas de *lay-out* para as placas de circuito impresso dos adaptadores de comunicação.

Têm-se como perspectivas futuras a implementação desse projeto, a fim de se construir, em um breve período de tempo, um protótipo inicial da máquina. Dessa forma, o presente trabalho vem colaborar com o próprio desenvolvimento do Projeto Nó//, possibilitando que futuros trabalhos em *software* venham a ser realizados diretamente sobre uma máquina real.

A nível de *hardware,* pode-se esperar, ainda, uma evolução no sentido de migrar-se para outros padrões de barramento, como por exemplo o barramento PCI, que oferecem taxas para transferência de dados bem maiores que as do barramento ISA. Além disso, é previsto um aumento no número de canais para o *crossbar* no adaptador de comunicação para os nós de trabalho.

REFERÊNCIAS BIBLIOGRÁFICAS

1 ALMEIDA, Virgilio A. F. & ÁRABE, José Nagib C. **Introdução à supercomputação**. Rio de Janeiro : LTC, 1991.

2 ALTERA. **Choosing the right MAX+PLUS II development tools**. San Jose : Altera Corporation, 1994.

3 ALVES, Valeria A. **Multicomputador Nó//: implementação de primitivas básicas de comunicação e avaliação de desempenho.** Dissertação de Mestrado, Curso de Pós-Graduação em Ciência da Computação, Universidade Federal de Santa Catarina, Florianópolis, SC, 1996.

4 BHUYAM, Laxmi. N.; AGRAWAL, Dharma. P. Design and performance of generalized interconection networks. **IEEE Transactions on Computers**, v. c-32, n., p. 1081-1090, dec. 1983.

5 BHUYAN, Laxmi N. Interconnection networks for parallel and distributed processing. **Computer**, v. 20, n. 6, p. 9-12, june 1987.

6 CAMPOS, Rodrigo A. **Um sistema operacional fundamentado no modelo cliente-servidor e um simulador multiprogramado para multicomputador**. Dissertação de Mestrado, Curso de Pós-Graduação em Ciência da Computação, Universidade Federal de Santa Catarina, Florianópolis, SC, 1995.

7 CORSO, Thadeu B. **Ambiente para programação paralela em multicomputador.** Florianópolis : Relatório Técnico UFSC-CTC-INE Nº 1, 1993.

8 EGGEBRECHT, L. C. **Interfacing to the IBM personal computer.** 2 ed, USA : SAMS, 1991.

9 FENG, Tse-yun. A survey of interconnection networks. **Computer**, v. 14, n. 12, p.

12-27, dec. 1981.

10 FERNÁNDES, Edil S. T. & AMORIN, Claudio L. de. **Arquiteturas paralelas avançadas**. In: VI Escuela Brasileño Argentina de Informática (1993 : Córdoba). Córdoba : Programa Argentino - Brasileño de Investigación y Estudios Avanzados en Informática, jul. 1993.

11 FLYNN, Michael J. Some computer organizations and their effectiveness. **IEEE Transactions on Computers**, v. c-21, n. 9, p. 948-960, sep. 1972.

12 FREITAS Filho, Paulo F. et alli. A simulation model for the comparison of two multicomputer architectures. In: **Society for Computer Simulation Conference**, Portland, 1996.

13 GIACOMO, Joseph Di. **Digital bus**. New York : McGraw-Hill, 1990.

14 HAYES, John P. **Computer architecture and organization**. 2. ed. New York : McGraw-Hill, 1987.

15 HOARE, C. A. R. **Communicating sequential process.** Prentice-Hall, 1985.

16 HWANG, Kai. **Advanced computer architecture** : parallelism, scalability, programmability. Singapore : McGraw-Hill, 1993.

17 INMOS. IMS C004 programmable link switch. In: **INMOS Engineering Data**, 1989. p. 479-501.

18 INMOS. IMS C011 : link adaptor. In: **INMOS Engineering Data**, 1988. p. 389-412.

19 INMOS. **The transputer applications notebook : architecture and software.** Melksham : Redwood Press Limited, 1989

20 INTEL. **Peripheral components.** USA : Intel, 1994.

21 KARPLUS, Walter J. Vector processors and multiprocessors. In: HWANG, Kai & DEGROOT, Douglas. **Parallel processing for supercomputers and artificial inteligence**. New York : McGraw-Hill, 1989, p. 3-30.

22 KITAJIMA, João Paulo F. W. **Programação paralela utilizando mensagens**. Porto Alegre : Instituto de Informática da UFRGS, 1995.

23 KLOTH, A. **Bussystem des PC.** Germany : Franzis, 1994.

24 MERKLE, Carla & BOING, Hamilcar. **Simulação do Nó//.** Florianópolis : Relatório Técnico UFSC-CTC-INE, 1996.

25 MERKLE, Carla. **Ambiente para execução de programas paralelos escritos na linguagem SuperPascal em um multicomputador com rede de interconexão dinâmica.** Dissertação de Mestrado, Curso de Pós-Graduação em Ciência da Computação, Universidade Federal de Santa Catarina, Florianópolis, SC, 1996.

26 MONTEZ, Carlos B. **Um sistema operacional com micronúcleo distribuído e um simulador multiprogramado de multicomputador**. Dissertação de Mestrado, Curso da Pós-Graduação em Ciência da Computação, Universidade Federal de Santa Catarina, Florianópolis, SC, 1995.

27 MUDGE, Trevor N.; HAYES, John. P.; WINSOR, Donald. C. Multiple bus architecture. **Computer**, v. 20, n. 6, p. 42-48, june 1987.

28 NAVAUX, Philippe. **Multiprocessadores** : organização e programação. Curso ministrado na UFPR, Curitiba.

29 PERROT, Ronald H. **Parallel programming**. Great Britain : Addison-Wesley, 1987.

30 QUINN, Michael J. **Parallel computing** : theory and practice. 2. ed. Singapore : McGraw-Hill, 1994.

31 REED, Daniel A. & FUJIMOTO, Richard M. **Multicomputer networks** : message-based parallel processing. MIT Press, 1987.

32 SAWCHUCK, Alexander A. et alli. Optical crossbar networks. **Computer**, v. 20, n. 6, p. 50-60, june 1987.

33 STALLINGS, William. **Computer organization and architecture** : principles of structure and function. 3. ed. New York : Macmillan, 1993.

34 TISHER, M. **PC intern - system programming**. USA : Abacus, 1991.

35 ZEFERINO, C. A.; LÜCKE, H. A. H.; SILVA, V. A. Um multicomputador com sistema experimental de comunicação. In. **VII Simpósio Brasileiro de Arquitetura de Computadores - Processamento de Alto Desempenho**, Canela, RS, 1995. p. 137-150.