Jener Toscano Lins e Silva

# Instrumentação virtual para microscopia de varredura

**Recife**

**2002**

Universidade Federal de Pernambuco
Programa de Pós-graduação em Engenharia Elétrica

# Instrumentação virtual para microscopia de varredura

Dissertação

submetida à Universidade Federal de Pernambuco

como parte dos requisitos para obtenção do grau de

Mestre em Engenharia Elétrica

Jener Toscano Lins e Silva

Recife, março de 2002.

**Silva, Jener Toscano Lins e**

**Instrumentação virtual para microscopia de varredura / Jener Toscano Lins e Silva. - Recife : O Autor, 2002.**

**xiii, 143 folhas : il., fig., tab.**

**Dissertação (mestrado) - Universidade Federal de Pernambuco. CTG. Engenharia Elétrica, 2002.**

**Inclui bibliografia e anexos.**

**1. Engenharia - Engenharia eletrônica. 2. Instrumentação virtual - Conceitos e aplicações. 3. Sistemas e sistema de tempo real - Microscopia de varredura. 4. Projeto do LDN/UFPE-2001 - Placa de aquisição de dados e varredura PCI - Composição da placa, PLD e S5920. I. Título.**

| | | |
|---|---|---|
| **567.3** | **CDU (2.ed.)** | **UFPE** |
| **577.78** | **CDD (22.ed.)** | **BC2005-275** |

**PARECER DA COMISSÃO EXAMINADORA DE DEFESA DE DISSERTAÇÃO DE MESTRADO DE**

# JENER TOSCANO LINS E SILVA

TÍTULO

**"INSTRUMENTAÇÃO VIRTUAL PARA MICROSCOPIA DE VARREDURA"**

**A comissão examinadora composta pelos professores: EDVAL JOSÉ PINHEIRO SANTOS, DES/UFPE, TOMAZ DE CARVALHO BARROS, DES/UFPE e JOSÉ SÉRGIO DA ROCHA NETO, DEE/UFPB, sob a presidência do primeiro, consideram o candidato JENER TOSCANO LINS E SILVA A P R O V A D O**

Recife, 08 de março de 2002.

**EDVAL JOSÉ PINHEIRO SANTOS**

**TOMAZ DE CARVALHO BARROS**

**JOSÉ SÉRGIO DA ROCHA NETO**

A meus Pais.

# Agradecimentos

Agradeço,

Àqueles que sempre me deram apoio incondicional e me conduziram nos estudos para que pudesse chegar aqui, meu pai Jair Barbosa Lins e Silva e minha mãe Terezinha de Jesus Toscano Lins e Silva.

A Sandra Low por ter me encorajado a fazer este mestrado, torcendo sempre no meu sucesso profissional e ao amigo e professor Edval José Pinheiro Santos que além de orientar todo este trabalho, soube diante a confiança depositada em mim, com sua tolerância, sabedoria e experiência, me ajudar nos momentos mais críticos em que eu

A Victor Garcia da AMCC por ter gentilmente esclarecido as dúvidas sobre o funcionamento do controlador do barramento PCI, ao convênio FACEPE/SENAI, que possibilitou uma bolsa de estudos à nível de mestrado, projeto FINEP/RECOPE e o projeto CNPq / Instituto do Milênio.

Aos colegas de trabalho, em particular, a Alberto de Moraes Barbosa pela inestimável ajuda na realização não só deste trabalho, mas durante o cumprimento dos créditos curriculares, a Khyale Santos Nascimento pela ajuda na confecção do layout da placa, Victor Miranda, Marcílio Feitosa, Renato Cintra e Isnaldo Coelho. E a todos que me apoiaram direto ou indiretamente na realização deste trabalho.

**J**ENER **T**OSCANO **L**INS E **S**ILVA

*Universidade Federal de Pernambuco*

*8 de março de 2002*

Resumo da Dissertação apresentada à UFPE como parte dos requisitos necessários para obtenção do grau de Mestre em Engenharia Elétrica.

# Instrumentação virtual para microscopia de varredura

Jener Toscano Lins e Silva

março/2002

Orientador: Edval J. P. Santos, Rh.D.
Área de Concentração: Eletrônica
Palavras-chaves: Instrumentação Virtual, Microscopia de Varredura, Barramento PCI
Número de páginas: xiii+143

O microcomputador é hoje uma ferramenta indispensável em qualquer tipo de atividade tecnológica, não só pela capacidade de analisar dados, mas também pela possibilidade de gerar e coletar sinais. Este trabalho se refere a implementação de instrumentação virtual capaz de fazer a aquisição de sinais e o controle de um microscópio de varredura, seja ele eletrônico, de força atômica, capacitivo ou de tunelamento. A instrumentação virtual consiste basicamente de uma placa periférica de microcomputador e uma interface gráfica. O circuito implementado utiliza um algoritmo de tempo real gravado em um PLD para gerenciar a escrita e leitura em dois bancos de memória independentes. Enquanto uma memória é escrita através do conversor analógico-digital, a outra tem seu conteúdo transferido para a memória principal do micro através de um controlador comercial de barramento PCI. Os bancos de memória são sincronizados através da utilização de um semáforo binário. O usuário pode comandar a placa facilmente, através da interface gráfica criada com o aplicativo *Delphi® ou Kylix®*. A placa oferece grande potencial para diversas outras aplicações. Por exemplo, ela pode ser utilizada como ferramenta de ensino, onde o aluno tem a oportunidade de trabalhar com os conceitos de programação orientada a objeto, barramento PCI e tempo real. Outras aplicações são a litografia eletrônica e o desenvolvimento de nanomanipuladores.

Abstract of Dissertation presented to UFPE as a partial fulfillment of the requirements for the degree of Master in Electrical Engineering.

# Virtual Instrumentation for Scanning Microscopy

Jener Toscano Lins e Silva

March/2002

Supervisor: Edval J. P. Santos, Ph.D.
Area of Concentration: Electronic
Keywords: Virtual Instrumentation, Scanning Microscope, PCI Bus
Number of pages: xiii+143

The microcomputer is nowadays an indispensable tool in any type of technological activity, not only for its data analysis capability, but also for the possibility of generating and collecting signals. This work refers to the implementation of virtual instrumentation capable of signal acquisition and control of a scanning microscope, be it electronic, atomic force, capacitive or tunneling. The virtual instrumentation consists basically of a peripheral microcomputer board and a graphical user interface. The implemented circuit uses a real time algorithm programmed in a PLD to manage the reading and writing in two independent memory banks. While one memory is written through the analog-to-digital converter, the other has its content transferred to the main memory through a commercial PCI controller. The memory banks are synchronized through the use of a binary semaphore. The user can command the board easily, through the graphical interface developed in *Delphi*® *or Kylix*®. The board offers great potential for several applications. For instance, it can be used as teaching tool, where the student has the opportunity to work with object oriented programming concepts, PCI bus and real time. Other applications are electronic lithography and the development of nanomanipulators.

# Conteúdo

# -Listci d.G TcitíGlâiS

# Lista de Figuras

# Ccvpitjiilo 1

# Introdução

Instrumentação virtual é uma técnica bastante flexível de se implementar tanto instrumentos tradicionais, como instrumentos especiais. Uma vez que com o aparecimento de ambientes de desenvolvimento orientados a objeto, tais como *Delphi*® e *Kylix*®, o trabalho para a construção de interfaces gráficas ficou reduzido. Além disso, o uso do microcomputador como equipamento hospedeiro ajuda a baratear o custo do sistema.

A aplicação escolhida para esse projeto é o sistema de litografia por feixe de elétrons, objeto de convênio entre o Laboratório de Dispositivos e Nanoestruturas da UFPE e o Instituto de Problemas da Tecnologia de Microeletrônica da Academia Russa de Ciências. Em particular, um dos objetos do convênio é o estudo de técnicas rápidas para placas controladoras de microscópio de varredura. Um primeiro aspecto importante é a utilização do barramento PCI *(Peripheral Gomponent Interconnect).* O barramento PCI aumenta a velocidade de transferência de dados entre o processador e o periférico, oferecendo um melhor desempenho nas aplicações mais exigentes, quando comparado com os barramentos mais tradicionais como o ISA/EISA *(Industry Standard Architecture/Extended ISA)* [42].

Uma outra aplicação considerada é o desenvolvimento de uma ferramenta educacional. Uma vez que, conceitos de programação orientada a objeto, tempo real e barramento PCI, podem ser explorados com a placa desenvolvida.

Pode-se questionar a necessidade de se desenvolver uma placa de aquisição de dados e geração de varredura. No entanto, os seguintes aspectos devem ser considerados:

- Placas comerciais são tipicamente lentas para a aplicação almejada;

- As soluções comerciais mais rápidas são extremamente caras;
- Placas dedicadas podem ser otimizadas para a aplicação desejada;
-

diversos sistemas;

de ensino;

valor.

Em resumo, o objetivo principal deste trabalho é projetar e implementar um circuito eletrônico que possa ser conectado ao barramento PCI de um microcomputador e ser utilizado para aquisição de sinais e geração de varredura. A placa PCI denominada "placa LDN/UFPE-2001" deve ter o potencial de apresentar altas taxas de aquisição e varredura para poder controlar um microscópio de varredura, com intuito de fazer aquisição de imagem e litografia. A placa LDN/UFPE-2001 também deverá ser uma ferramenta de ensino.

## 1.1 Instrumentação Virtual

Existem basicamente duas idéias a serem consideradas, quando se fala de instrumentação virtual. A primeira idéia consiste na interação do computador com o mundo externo, através de interface gráfica amigável e utilizando uma placa com conversores AD e DA para monitoramento e controle. A segunda idéia consiste em utilizar um programa com interface gráfica que simule o comportamento de instrumentos reais. Esse último recurso oferece a possibilidade de simulação de medidas complexas, apresentando para o usuário, resultados equivalentes aos que seriam obtidos com o instrumento real. Levando em consideração ambas as idéias, é possível realizar um sistema misto, mais sofisticado e mais flexível, possibilitando o uso dual: um para experimento e o outro para desenvolvimento do processo de medida [21].

Atualmente, aplicativos de aquisição de dados são utilizados dentro dos conceitos de instrumentação virtual para emulação de um instrumento físico. Instrumentação

virtual permite que engenheiros projetem instrumentos de uso comum com funcionalidade definida pelo usuário e assim adequá-los para uma aplicação particular. Diferente dos outros tipos de aplicativos *(software)* de aquisição de dados, a instrumentação virtual combina facilidade de uso e flexibilidade, com a possibilidade da reutilização de módulos de desenvolvimento para novas aplicações [30].

Este trabalho utiliza o primeiro conceito de instrumento virtual. O sistema é composto por microcomputador, interface gráfica e placa de aquisição de dados e geração de varredura, veja o diagrama apresentado na Figura 1.1. Os sensores são dispositivos de entrada, responsáveis pela captura dos sinais externos. Em diversas aplicações, o nível de tensão desses sinais é tipicamente baixo, sendo necessário amplificá-los e condicioná-los. Sinais analógicos são convertidos para a forma digital pelo conversor AD presente na placa de aquisição, para que o computador possa tratá-los. Ao receber esses dados, o computador armazena em sua memória, enviando o resultado do processo para a interface gráfica, para que o mesmo seja apresentado ao usuário através do monitor de vídeo. Se for necessária uma intervenção no ambiente externo, um sinal é enviado do computador para o controlador presente na placa de aquisição. Para essa tarefa o sinal digital do computador deve ser convertido da forma digital pctFct 3. SM3. forma analógica. Isso é feito através do conversor DA também presente na placa de aquisição.

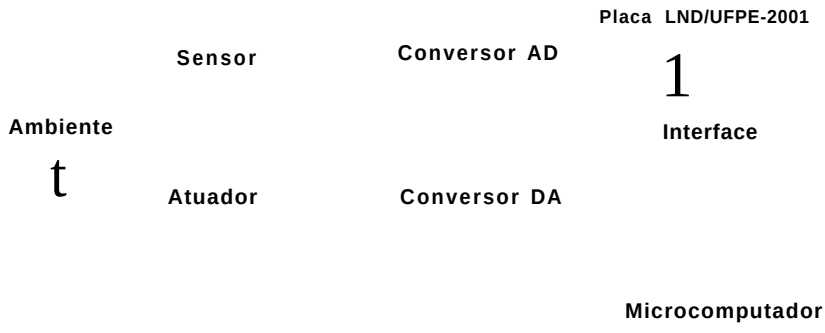



Figura 1.1: Conceito de instrumentação virtual usado no trabalho.

### 1.1.1 Aplicações utilizando instrumentação virtual

É possível acompanhar, nos últimos anos, o crescente uso dos microcomputadores para os mais diversos fins, desde simples tarefas de escritório até complicados controles de processos industriais. Isso se deve ao contínuo barateamento dos equipamentos, combinado com o aumento da capacidade de processamento e disponibilidade de aplicativos.

O surgimento da *internet* veio consolidar o uso dos microcomputadores em redes locais, permitindo a troca de informações de maneira cada vez mais eficiente. A rápida expansão do uso da *internet* como ferramenta de compartilhamento de informações, juntamente com ferramentas modernas de controle, aquisição e distribuição de dados via rede de computadores tem impulsionado o controle e acompanhamento de ensaios em ambientes remotos [27].

O sistema de instrumentação virtual faz uso dos microcomputadores para medir determinada grandeza e/ou estabelecer controle, sendo bastante utilizado em processos de controle na área industrial, na medicina e nas aulas práticas em laboratórios.

Exemplos de aplicação:

- tema virtual de medida e controle apresentado na Figura 1.2, em que sinais do meio externo são captados por sensores. O circuito de condicionamento de sinal consiste de amplificadores de sinais, filtros e retificadores. Um conversor A/D é usado para transformar o sinal analógico em digital, de maneira que a informação possa ser processada, misturada, comparada, manipulada e/ou armazenada, conforme sua aplicação. A informação armazenada em memória, pode ser visualizada na tela do computador. O resultado do processamento pode ser colocado novamente no formato analógico para o controle do processo [15].

- para a análise de sistema biológico através da aquisição de dados do pH. O controle instrumental, aquisição, armazenamento, processamento e apresentação dos dados experimentais são feitos através da placa de aquisição de dados, uma interface gráfica e aplicativo *(software)* de análise gráfica e numérica. Os eletrodos, usados para medição do pH, desenvolvem um potencial elétrico na saída

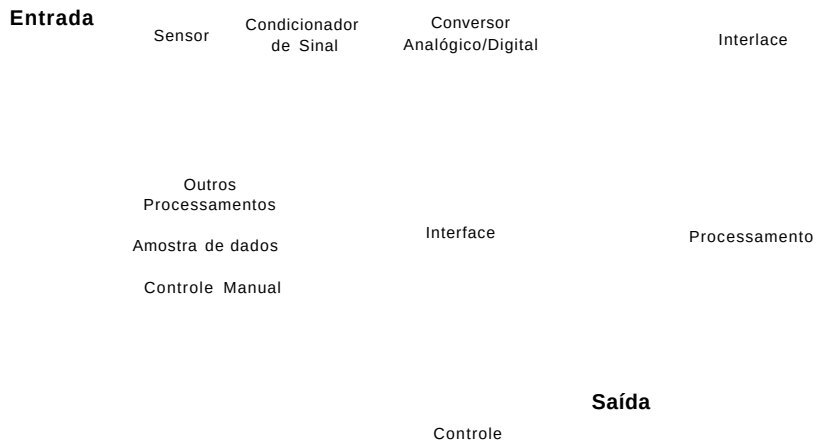



Figura 1.2: Esquema típico de um instrumento de medida e controle

proporcional ao pH observado. O aplicativo foi desenvolvido utilizando o sistema de programação gráfica LabVIEW® (National Instruments) para a coleta e análise dos dados.

- O microcomputador com eletrodos acoplados também foi utilizado para o estudo de mitoerôndias. O *software* de análise gráfica e numérica é responsável pelo controle da instrumentação, armazenamento dos dados, processamento e apresentação dos dados experimentais. Nessa aplicação também foi utilizado o ambiente LabVIEW® [13] para fazer aquisição de dados, controle dos instrumentos e monitoramento de processos.

-

de um sistema de alto desempenho para aquisição e processamento digital de sinais biomédicos, com a finalidade de fazer o processamento de sinais digitais eletroencefalográficos em tempo real, dentro do projeto de interface cérebro-computador.

-

torar ou acionar variáveis de um processo, com um simples teclado de computador, ou com um painel sensível ao toque, visualizando na tela barras e gráficos em tempo real da variável selecionada. Através do IHM é possível ainda, gerar relatórios e se comunicar através da rede [7].

- medidas, pode-se construir uma estação de trabalho baseada em computador para o ensino de eletrônica. A estação possibilita a integração de um conjunto de funcionalidades, permitindo ao aluno desenvolver grande parte do trabalho laboratorial associado a sua aprendizagem e complementa o equipamento tradicional de bancada de um laboratório de eletrônica [32]. Um exemplo disto, é o trabalho de Grimoni e Lopes [22] que apresenta uma plataforma experimental para sistemas de energia elétrica, constituída de sensores de corrente e de tensão, desenvolvido de forma modular, permitindo ao aluno fazer vários experimentos. Outro exemplo é o trabalho de Nascimento e Santos [31] que desenvolveram uma placa de geração de forma de onda arbitrária.

A placa LDN/UFPE-2001 é uma placa PCI que deve ser instalada na placa mãe de um micro e juntamente com uma interface gráfica forma um instrumento virtual, conforme será discutido mais adiante.

## 1.2 Laboratório virtual

O desenvolvimento e uso de laboratórios virtuais onde estudantes e pesquisadores controlam instrumentos remotamente, existindo uma metodologia de coleta e análise de dados, se mostra como uma das recentes atividades promissoras que une diferentes tecnologias para a elaboração de sistemas poderosos e seguros. Já existem, em diversas partes do mundo, laboratórios virtuais que fazem uso da *internet* para compartilhar informações entre equipes de trabalho na área industrial, entre alunos e pesquisadores na área acadêmica [69, 75] e entre equipes médicas [35]. Esses laboratórios são implementados em diversas áreas do conhecimento e com os mais diferentes propósitos. Podem existir apenas para consultas ao banco de dados de universidade bem como para simular ou controlar experiências e monitorar processos de controle de forma remota. O uso de laboratório virtual para aulas práticas é uma forma didática recente e muito empregada em diversas universidades, tais como: Universidade Nacional de Singapura (NUS), Universidade de Stanford, Universidade de Padova e a Universidade Norueguesa de Ciência e Tecnologia.

A troca de informações médicas sobre pacientes entre hospitais e grupos de especialistas para avaliação, discussão e diagnóstico de doença é hoje uma realidade e conquista, a cada dia, maior atenção dos especialistas e desenvolvedores de sistemas. Como exemplo, pode-se citar o InCor (Instituto do Coração das Clínicas da Universidade de São Paulo) que utiliza um sistema de monitoramento de sinais vitais em tempo real de forma remota para pacientes recentemente operados [11].

No trabalho de Fernandez et al [19, 72] é apresentado um Laboratório Virtual que permite o acompanhamento e o controle de experimentos e ensaios remotamente, via *internet*. O experimento inicial foi feito utilizando apenas um microcomputador equipado com uma placa de aquisição de dados. Para possibilitar o acesso remoto ao experimento foram desenvolvidos dois programas utilizando o ® primeiro programa é executado no computador diretamente ligado ao experimento, através da placa de aquisição de dados e controla tanto a execução das medidas, quanto a recepção e transmissão dos dados pela *internet*. O segundo programa deve ser executado no computador do usuário remoto e é responsável por enviar os parâmetros a serem utilizados no experimento bem como visualizar o gráfico com os resultados do mesmo.

Uma aplicação prática desse recurso é o monitoramento da qualidade da água através da *internet*. O método clássico de monitoramento da qualidade da água se faz através de uma amostra coletada por uma pessoa qualificada, normalmente usando instrumento paramétrico. Esse método de monitoramento traz diversas desvantagens pois, exige freqüentes visitas de pessoas ao local, acarretando em possíveis erros humanos e atrasos. No caso de contaminação, a amostra terá que ser levada ao laboratório químico para análises mais precisas. O sistema de monitoramento da qualidade da água desenvolvido por Torán et al [40] pode ser utilizado em vários tipos de computadores e oferece os seguintes recursos automatizados: notificação de alarme por correio eletrônico, registro das medidas efetuadas, geração dinâmica de relatórios HTML e gráficos em tempo real.

## 1.3 Sistemas de tempo real

Um sistema de tempo real pode ser definido como um sistema computacional que deve reagir a estímulos oriundos do seu ambiente em prazos específicos, sob pena de ocorrer uma falha temporal. O comportamento correto de um sistema de tempo real, não depende só da integridade dos resultados obtidos, mas também dos instantes de tempo em que são produzidos. Um sistema de tempo real deve ser então capaz de oferecer garantias de correção temporal para o fornecimento de todos os serviços que apresentem restrições temporais [6] [12] [17].

Dependendo do grau de restrição de tempo, sistemas de tempo real podem ser classificados como firme ou flexível. No sistema de tempo real firme, as restrições temporais devem ser satisfeitas sem ultrapassar o prazo limite. Como é o caso dos sistemas responsáveis pelo monitoramento de pacientes em hospitais, sistemas de supervisão e controle em plantas de industriais e os sistemas embarcados em robôs e veículos, tais como, automóveis e sondas espaciais. No sistema de tempo real flexível, as restrições temporais podem ser satisfeitas em um prazo médio, i.e, o prazo limite pode ser ultrapassado, desde que na média ele seja respeitado. É o caso dos *videogames*, multimídias e teleconferências pela *internet*, em que é admissível o salto esporádico de algum quadro, durante apresentação de vídeo [17].

Além das restrições temporais dos componentes críticos, sistemas embarcados de tempo real devem ser projetados de forma que todas as tarefas sejam executadas dentro dos prazos, para tal torna-se necessário incluir um núcleo de sistema operacional que garanta a execução concorrente de tarefas de acordo com as restrições temporais [12].

Os sistemas de tempo real têm se tornado mais comuns nos últimos anos, afetando cada vez mais pessoas. Desde terminais bancários de caixas eletrônicos a aparelhos eletrodomésticos, diariamente as pessoas são beneficiadas pelos serviços que esse tipo de sistema oferece. Os sistemas de tempo real devem oferecer confiabilidade no seu funcionamento, evitando o prejuízo para as pessoas que utilizam os sistemas e dependem deles [12].

Dessa forma, torna-se necessário à utilização de mecanismos para lidar com os problemas que potencialmente possam afetar os sistemas. Tolerância a falhas é um

desses mecanismos. Diferente da prevenção de falhas, tolerar as falhas do sistema implica em reconhecer que as falhas são inevitáveis; tendo origem em erros de projetos ou implementação, desgaste do material ou colapsos na fonte de energia. Um sistema tolerante a falhas dispõe de alternativas que permitam ao sistema manter o funcionamento desejado mesmo na ocorrência de falhas. As falhas podem ocorrer tanto no *software*, como no *hardware*. Para se adquirir tolerância a falhas, é necessário o uso de redundância de componentes, de *software* ou *hardware* [12].

Um estudo das técnicas de tolerância a falhas a nível de *software* sem a necessidade de utilizar recursos dedicados de *hardware* foi realizado por Fernandes [18] e Burns et al [12]. As técnicas estudadas estão orientadas para utilização em microcomputadores comerciais do tipo IBM-PC e para uma classe de aplicações que envolve o controle de processo do tipo não crítico em tempo real, assim como conceitos, terminologias, técnicas de detecção e recuperação de erros, constituindo os meios para melhoria da segurança de funcionamento dos sistemas. A placa LDN/UFPE-2001 emprega a técnica de tolerância a falha por *software* (veja Secção 3.9).

### 1.3.1 Monitoramento em tempo real

O monitoramento em tempo real consiste na observação contínua dos parâmetros, que caracterizam o estado funcional de um sistema ou equipamento, diagnosticando tendências de falhas e desajustes, viabilizando a programação das datas mais oportunas para as ações corretivas, antes que os problemas se tornem críticos para a qualidade dos serviços oferecidos aos usuários. Menezes [26] propõe um modelo de monitoramento de baixo custo para os equipamentos de supervisão e controle.

No programa de leitura do controlador de barramento PCI com a memória SRAM da placa LDN/UFPE-2001, os parâmetros de entrada do controlador são monitorados em cada estado, de maneira a garantir o seu perfeito funcionamento.

### 1.3.2 Núcleo de programação concorrente

Aplicativos para aplicações de tempo real apresenta uma complexidade maior do que aplicativos para tarefas puramente seqüenciais. Sistemas de tempo real monitoram eventos do ambiente externo; estes eventos são assíncronos porque podem ocorrer a

qualquer tempo e são potencialmente concorrentes porque um evento pode ocorrer enquanto outro está sendo processado. Qualquer programa para processar eventos múltiplos, concorrentes e assíncronos são naturalmente complexos. Para realizar a cooperação ordenada entre os processos é necessário que haja:

- evento causado por outro processo, sendo então necessário o sincronismo destes processos;

- nas um deles de cada vez. Por exemplo, um não deve modificar uma tabela, enquanto o outro a está lendo. O mecanismo de alocação de recursos deve garantir a inexistência de *deadlock*. Um processo está em *deadlock* quando ele espera por um evento que nunca vai ocorrer, como por exemplo, quando entre dois processos cada um espera que o outro libere um recurso para poder continuar [12].

O núcleo é um conjunto de rotinas responsável pela sincronização, comunicação e exclusão mútua entre processos. A principal função de um núcleo é permitir o compartilhamento da UCP (Unidade Central de Processamento) entre os processos, de tal forma que cada processo tenha sua "UCP virtual". Além disso, deve prover mecanismos de comunicação entre processos. Um núcleo de programação concorrente mais sofisticado facilita a implementação dos processos, porém o próprio núcleo é mais difícil de ser implementado. Além disso, o núcleo deve ser eficiente. Um modelo lógico de um sistema de programação concorrente é apresentado na Figura 1.3.

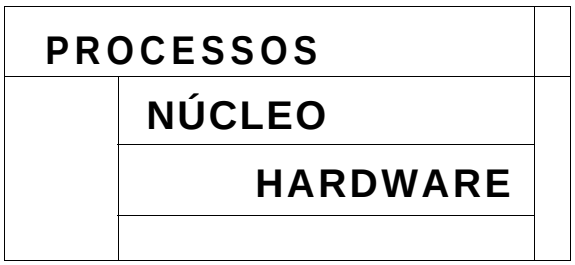



Figura 1.3: Sistema de programação concorrente

A placa LDN/UFPE-2001 dispõe de dois bancos de memória que são escritos (ou lidos) concorrentemente. A comutação dos bancos de memória de escrita e leitura é permitida apenas quando houver a finalização das duas tarefas que são independentes entre si. Esse controle é feito com duas bandeiras "A" e "B", conforme será discutido adiante.

Para evitar que o UCP fique ociosa quando nenhum dos processos está pronto para execução, pode-se acrescentar o processo "ocioso", com prioridade 0, esse processo está sempre pronto para execução. Supõe-se que a UCP seja capaz de executar todos os processos em tempo hábil, ou seja, o processo "ocioso" deve ser executado alguma vez. Isto garante que mesmo os processos de baixa prioridade serão executados, não sendo necessário alterar sua prioridade dinamicamente [17].

### 1.3.3 Suporte para aplicação de tempo real

Em aplicações de tempo real, as exigências temporais dependem do auxílio do sistema operacional. Nas aplicações em que as exigências de tempo são rigorosas, o sistema operacional é substituído por um único núcleo de tempo real, não incluindo serviços, tais como: sistemas de arquivos ou gerenciamento sofisticado de memória. Esse núcleo de tempo real simplificado oferece uma funcionalidade mínima, apresentando um excelente comportamento temporal em razão da sua simplicidade interna.

Aplicações de tempo real ficam mais fáceis de serem construídas, quando é possível aproveitar os serviços de um sistema operacional. Assim, não é necessário que o programador se preocupe com a gerência dos recursos básicos do processador, memória, controlador de disco, etc.

Sistemas operacionais comuns dificilmente atendem às exigências de tempo real, pois sua construção visa atender o propósito funcional e não temporal. Por exemplo, a questão da previsibilidade temporal de uma tarefa não é um fator presente nestes sistemas operacionais comuns. Os mecanismos de *caches*, memória virtual, partição de tarefas do processador, entre outros melhoram o desempenho do sistema, porém tornam difícil fazer afirmações sobre o comportamento de uma tarefa, diante das restrições de tempo. Um sistema operacional de tempo real é aquele que atende não só as exigências funcionais, mas também as restrições temporais.

O suporte para aplicação de tempo real varia em tamanho e funcionalidade de acordo com sua aplicação. Basicamente classifica-se em dois tipos: núcleos de tempo real e sistemas operacionais de tempo real.

O núcleo de tempo real é constituído de um pequeno "núcleo" com funcionalidade mínima. Podendo ser utilizado, por exemplo, no controlador de uma máquina industrial, enquanto que o sistema operacional de tempo real é utilizado para propósito geral, com um núcleo adaptado para melhorar o comportamento temporal. O desempenho temporal do núcleo adaptado, varia de sistema para sistema, pois, alguns são totalmente projetados para tempo real, enquanto outros recebem poucas otimizações.

O sistema computacional pode ser organizado em camadas, de modo a facilitar o desenvolvimento de todos os serviços e ter um bom comportamento temporal. Na Figura 1.4, o leitor poderá perceber que a medida que subimos na estrutura de camadas, os serviços tornam-se mais sofisticados e o comportamento temporal menos previsível.

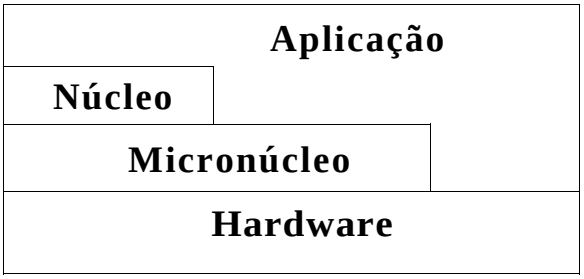



Figura 1.4: Camadas de um sistema computacional

Acima do *hardware* existe um "micronúcleo" responsável pelos serviços básicos, como por exemplo, alocação e liberação de memória física, instalação de novos acionadores de dispositivos e mecanismo para sincronização de tarefas. O núcleo é responsável pelos serviços de sistema de arquivos e protocolos de comunicações. Assim, a aplicação possui uma variedade de serviços a sua disposição, porém quando essas aplicações exigem um comportamento temporal melhor, pode-se acessar diretamente o micronúcleo ou até mesmo o próprio *hardware* [17].

## 1.4 O sistema operacional Linux

O sistema operacional *Linux* foi criado por Linus Torvalds em 1991, na Universidade de Helsinque, na Filândia. Este sistema operacional é uma variante do *Unix* e apresenta inúmeras vantagens, tais como: excelente estabilidade, eficiência, sem restrição de licença, estando o código fonte disponível na *internet*. Além disso, o *Linux* pode ser executado em diversos tipos de computadores, tais como: IBM PCs e máquinas compatíveis.

Apesar de ser um sistema operacional relativamente recente, possui todas as características de um sistema *Unix* moderno, pois contém várias implementações de interface gráfica sobre o sistema *X Windows*, rede, banco de dados, liguagem de programação, depuradores e uma variedade de aplicativos. Seu núcleo é encarregado de fazer o gerenciamento de recursos e periféricos, tais como: memória, discos rígidos, arquivos, impressoras, CD-ROMs, etc. O sistema pode ser embarcado, podendo ser executado com pouca memória RAM (memória volátil de leitura e escrita) [6].

Hoje o *Linux* é desenvolvido por milhares de pessoas no mundo todo, onde cada uma contribui gratuitamente para a evolução do núcleo e dos aplicativos. O próprio Linus Torvalds ainda trabalha no seu desenvolvimento e ajuda na coordenação entre os desenvolvedores. O suporte ao sistema se destaca como sendo bastante eficiente e rápido, em relação aos sistemas comerciais disponíveis no mercado.

O *Linux* convencional não é apropriado para aplicações de tempo real, pois além de não ser baseado em micronúcleo, segue o estilo do núcleo *Unix* tradicional. Porém o núcleo do *Linux* possui um recurso que facilita aplicações de tempo real: o código fonte é aberto. Isso possibilita o estudo do seu comportamento temporal. A mesma facilidade não acontece com outros sistemas operacionais de tempo real, os quais possuem normalmente um núcleo que se comporta, como uma caixa preta.

Existem vários distribuidores do *Linux*, os principais são: Conectiva, Red Hat, Debian, Corel e Slackware, onde cada distribuidor diferencia basicamente pela organização do sistema de arquivos e os aplicativos incluídos [60].

Desta forma, o *Linux* tem potencial para se tornar uma excelente plataforma de desenvolvimento para uma grande variedade de aplicações. Em particular, com o desenvolvimento do *Kylix*® é possível desenvolver aplicativos para *Windows* e *Linux*.

### 1.4.1 O Sistema Operacional Linux de Tempo Real

O *Linux* de Tempo Real é um sistema operacional onde um micronúcleo de tempo real trabalha junto com um núcleo convencional *Linux* [8]. Esse sistema foi desenvolvido no Instituto de Tecnologia do Novo México com o propósito de fazer uso de serviços convencionais do *Linux,* junto com tarefas de tempo real em uma mesma máquina, não necessitando fazer modificações no *Linux.* Para que isso aconteça, o micronúcleo de tempo real considera o núcleo convencional como uma tarefa com 1jâixâprioridade, sendo particionada e executada, quando não houver tarefas de tempo real esperando para ser processadas. O *Linux* de Tempo Real recorre ao *Linux* também para realizar várias funções de um sistema operacional convencional, tais como: comunicação com a rede, operação no sistema de arquivos e controle de processos. Além disso, o *Linux* é responsável pela inicialização do *Linux* de Tempo Real e pela maioria dos acionadores de dispositivos *(drivers).*

Visando diminuir as mudanças no núcleo convencional, é adicionado no micronúcleo de tempo real uma emulação de *hardware* para controle de interrupção. Assim, quando o *Linux* convencional desabilitar as interrupções, o *software* emulador irá enfileirar as interrupções que foram passadas pelo micronúcleo de tempo real, para serem processadas posteriormente.

Uma aplicação de tempo real típica consiste de tarefas de tempo real incorporadas às tarefas do sistema convencional do *Linux,* na forma de módulos adicionais do núcleo, sendo responsáveis por funções como: registro de dados em arquivos, atualização da tela, comunicação via rede, entre outras funções, sem restrições temporais. Assim, essas facilidades podem ser aplicadas a placa LDN/UFPE-2001, onde as tarefas e o micronúcleo de tempo real são carregados como módulos adicionais ao núcleo convencional. Mas é importante lembrar que esta solução não suporta requisitos temporais durante a inicialização do sistema, o que ocorre com todos sistemas operacionais de tempo real.

O micronúcleo de tempo real oferece poucos serviços, tais como: tarefas com agendamento baseado em prioridades fixas e alocação estática de memória. A comunicação de tarefas de tempo real utiliza memória compartilhada e sincronização, podendo ser feita através da desabilitação das interrupções de *hardware.* A disponibilidade de

novos serviços para tarefas de tempo real pode ser obtida através do mecanismo de módulos de núcleo do *Linux*. Porém, o comportamento das tarefas de tempo real ficará mais difícil de prever à medida que aumenta a complexidade destes serviços [17].

O *Linux* de Tempo Real permite que o usuário escreva o seu próprio sistema de agendamento *(scheduling),* podendo ser testadas diferentes políticas de agendamento, selecionando a que melhor se adapta a aplicação [6].

O núcleo original do *Linux* é executado juntamente com o *Linux* de Tempo Real, através da técnica de pré-empção, necessitando de um mecanismo de comunicação para que se possa fazer uma chamada ao *Linux,* utilizando uma tarefa de tempo real. Assim, para que haja comunicação entre os processos *Linux,* ou entre o núcleo convencional e as tarefas de tempo real, são criados no *Linux* de Tempo Real "*buffers FIFOs (First-In, First-Out)".* Quando os FIFOs são acessados, as interrupções são desabilitadas [6].

A dificuldade no desenvolvimento de sistemas de tempo real, principalmente pela necessidade de *hardware* especializado, tornam o projeto muitas vezes inviável, devido ao alto custo. Para diminuir tais dificuldades, uma alternativa é utilizar sistemas distribuídos, através de um protocolo de comunicação de grupo confiável, conforme é discutido na referência de Figueiredo et al [20].

## 1.5 Microscopia de varredura

A principal motivação para o desenvolvimento da placa LDN/UFPE-2001 é sua utilização no controle de um microscópio de varredura. Para isso faz-se necessário fazer um resumo sobre os principais tipos de microscópios de varredura.

### 1.5.1 Microscópio Eletrônico de Varredura (MEV)

No início do século XX, foi descoberto por H. Busch que campos magnéticos poderiam ser utilizados para construir lentes para feixes de elétrons. Essa descoberta foi fundamental para a invenção do microscópio eletrônico por E. Ruska. Nesse tipo de microscópio é utilizado o conceito de varredura. A imagem do objeto é formada pela detecção de pequenas áreas do objeto em um arranjo matricial. A relação entre o objeto e sua imagem, passa de uma função geométrica para uma função temporal [47, 58].

O MEV é um microscópio versátil que permite a obtenção de informações estruturais e químicas das amostras [66]. A microscopia de varredura oferece contribuições a diversas áreas do conhecimento, podendo ser empregada na indústria e em pesquisas acadêmicas [57, 61].

Para a realização da microscopia de varredura, podemos utilizar, em princípio, qualquer interação entre um estímulo e a matéria, que resulte em uma resposta que possa ser captada por um sensor. Exemplifiquemos pela descrição do MEV: um feixe de elétrons com cerca de *20keV*, gerado em um canhão de elétrons (cátodo) é condensado por um conjunto de lentes eletromagnéticas denominadas de lentes condensadoras. Este feixe é focalizado sobre o objeto, e mediante bobinas defietoras, percorre a superfície do objeto, fazendo uma varredura sobre uma pequena região da mesma. O feixe de elétrons incidente é denominado de feixe primário. O feixe primário ao penetrar a superfície do objeto, excita átomos e moléculas, os quais emitem elétrons denominados de secundários com energia em torno de 50eV. Estes elétrons são captados por um detector cuja resposta modula o brilho de um tubo de raios catódicos, que é varrido em sincronismo com o feixe eletrônico. Portanto, a cada ponto da amostra corresponde um ponto da tela, e nela é mapeada a resposta do objeto ao feixe de excitação. O aumento é obtido pela relação entre a área varrida sobre a amostra, e a área da tela. Na Figura 1.5 o leitor poderá visualizar o princípio de funcionamento do microscópio eletrônico de varredura [54].

Na Figura 1.6, é apresentado um diagrama do microscópio eletrônico de varredura em desenvolvimento no Instituto de Problemas da Tecnologia de Microeletrônica da Academia Russa de Ciências.

### 1.5.2 Microscópio de Varredura por Sonda (SPM)

Um microscópio de varredura por sonda *(Scanning Probe Microscope)*, SPM, é na realidade um grupo de instrumentos compostos basicamente de sonda sensora, cerâmicas piezelétricas para posicionar o objeto e fazer varreduras, circuitos de realimentação para controlar a posição vertical da sonda e um computador para geração de varredura, armazenamento de dados e conversão de imagens por meio de aplicativos específicos fim.

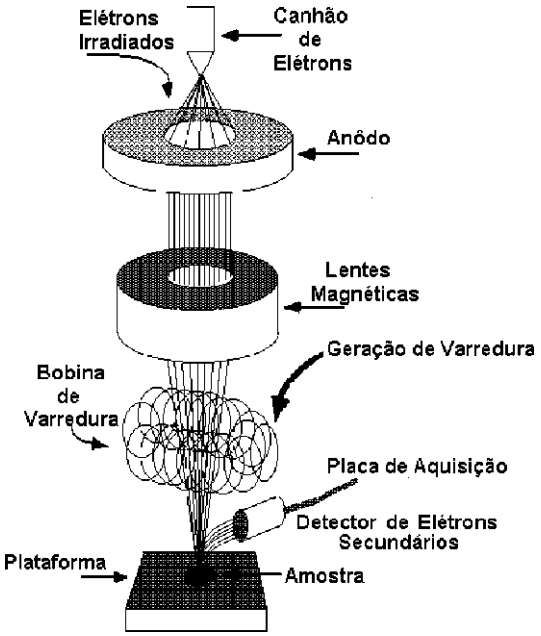



Figura 1.5: Princípio de funcionamento do microscópio eletrônico de varredura

Há diversos tipos de microscópios de sonda. Os mais conhecidos são:

- De tunelamento ou STM *(Scanning Tunneling Microscope);*
- De força atômica ou AFM *(Atomic Force Microscope);*
-

O STM foi inventado por Gerd Binnig e Heinrich Rohrer, da IBM de Zurich, em 1981 e foi o primeiro instrumento capaz de gerar imagens de superfícies com resolução atômica. Em 1986, os inventores ganharam o Prêmio Nobel de Física junto com E. Ruska. Como o STM depende da existência de uma corrente de tunelamento, ele só é aplicável para a visualização de superfícies condutoras, podendo ser utilizado tanto no vácuo, como em atmosfera. Com o STM podem ser estudadas a topografia de materiais em escala atômica e as forças que agem entre a sonda e a amostra [14].

Modificando o microscópio de tunelamento para que ele funcione como um perfilômetro (aparelho para medir rugosidade em escala microscópica) Binnig, Quate e Gerber, desenvolveram o AFM em 1986.

O componente essencial do SPM é a ponta sensora, com a qual se consegue sondar as amostras e obter imagens com ampliações muito altas, de forma tal que podem

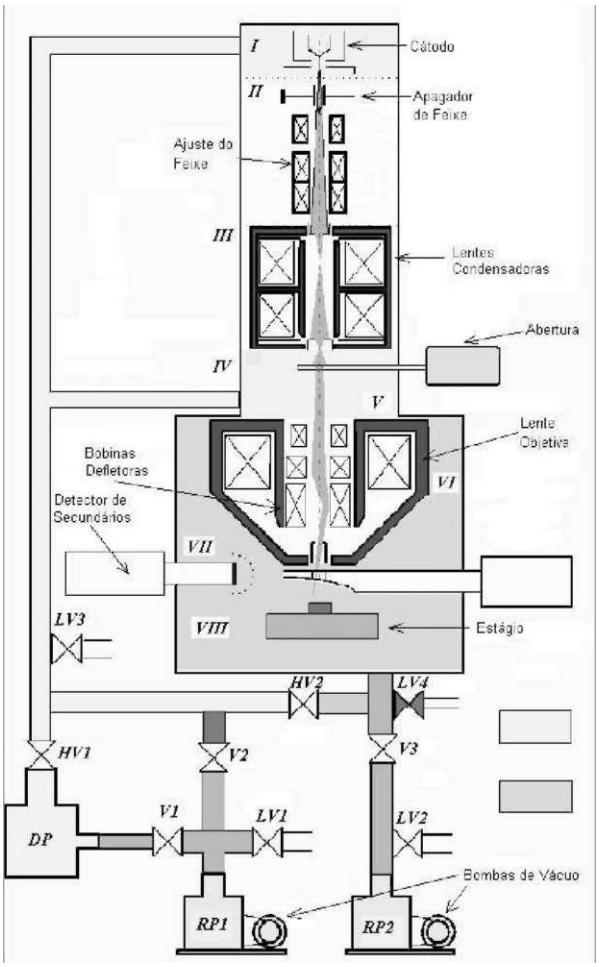



Figura 1.6: Diagrama esquemático do microscópio eletrônico de varredura em desenvolvimento no Instituto de Problemas da Tecnologia de Microeletrônica da Academia Russa de Ciências

ser medidas distâncias com resolução de até 0,1Â (1Â =10"¹⁰m). No microscópio de tunelamento, a ponta sensora é uma ponta metálica para que haja tunelamento de elétrons entre ela e a amostra. No microscópio de força atômica, o sensor de força pode ser em forma de ponta condutora ou isolante e para o SNOM, o sensor é uma fibra óptica.

O SNOM, ou microscópio óptico de campo próximo, pode ser considerado como a combinação da microscopia de varredura por sonda com microscopia óptica convencional. O SNOM melhora a resolução da óptica convencional em pelo menos uma ordem de grandeza, superando o limite de difração. Ele pode alcançar uma resolução espacial de até 25nm, que é uma resolução pior que a resolução do microscópio de força atômica. É por isso que esta técnica é utilizada como complementar ao AFM. O princípio é simples, a amostra a ser examinada é varrida com uma fibra óptica, que

tem uma abertura muito estreita (algumas centenas de ângstroms de diâmetro na sua extremidade) e que está recoberta por uma camada metálica opaca. Pela fibra passa luz visível que é refletida pela amostra, ou passa através da amostra para um detector. A intensidade do sinal óptico detectado em cada ponto da varredura constitui um conjunto de dados que irá reproduzir uma imagem da superfície da amostra com resolução entre *2ònm e òOnm,* com a única condição de que a distância entre a fonte de luz e a amostra seja da ordem de 50Â [74, 47, 70, 14].

Na Tabela 1.1 a seguir, é apresentado um quadro comparativo dos diversos tipos de microscópios [74].

Tabela 1.1: Quadro comparativo dos diversos tipos de microscópios.

| Características | aumento | meio | imagem | danos |
|---|---|---|---|---|
| Microsc. óptico | $IO^3$ | ar, líquidos | 2-D | nenhum |
| Varredura laser | $10^4$ | ar | 2-D | mínimos |
| Feixe de íons | $IO^5$ | vácuo | 2-D | graves |
| MEV ou SEM | $10^6$ | vácuo | 2-D | alguns |
| SPM | $10^9$ | liq., ar, vácuo | 3-D | mínimos ou nenhum |

Na Tabela 1.2 o leitor poderá observar a proposta dos dez principais tipos de microscópio SPM desenvolvidos no período de 1981 a 1991 [14]. Com esses microscópios é possível desenvolver técnicas de nanotecnologia, permitindo não só manipular os componentes da matéria, mas também, medir o seu tamanho. Um exemplo da importância da nanotecnologia é o controle de qualidade na produção de revestimentos de disquetes, discos rígidos, CDs e DVDs, bem como o uso de materiais com características bem determinadas em circuitos eletrônicos [47].

Outras aplicações estão ainda na fase de pesquisa. Uma das mais recentes, chamada *"dip-pen nanolithography"* desenvolvida na *Northwestern University,* utiliza-se uma ponta de AFM para escrever diretamente em uma superfície por ação capilar, usando o mesmo princípio de 4000 anos da caneta tinteiro. A diferença é que a linha traçada tem apenas 30nm de largura [55].

Richard Superfine e seus colegas e estudantes na *University of North Carolina, Chapei Hill* desenvolveram um nanomanipulador que combina um SPM a uma interface de realidade virtual. O resultado é um sistema de "telepresença" que opera com uma diferença de escala de um milhão para um. Um movimento manual de 1cm pro-

Tabela **1**.2: Alguns tipos de microscópios SPM.

| Microscópio: | Ano | Proposta |
|---|---|---|
| de Tunelamento | 1981 | estudar a morfologia de superfícies condutoras e semicondutoras com resolução atômica |
| Óptico de Campo Próximo | 1982 | fornecer imagens ópticas de superfícies com 50nm de resolução lateral |
| de Capacitância | 1984 | estudar a varação de capacitância com *500nm* de resolução lateral |
| Térmico | 1985 | estudar a temperatura de superfícies com 50nm de resolução lateral |
| de Força Atômica | 1986 | estudar a morfologia de superfícies condutoras, semi-condutoras e isolantes com resolução nanométrica |
| de Força Magnética | 1987 | estudar as características magnéticas de superfícies com lOOnm de resolução lateral |
| de Fotoemissão Inversa | 1988 | fornecer espectros de luminescência com resolução lateral nanométrica |
| Acústico de Campo Próximo | 1989 | realizar medidas acústicas de baixa freqüência com escala de *10nm* |
| de Potencial Químico | 1990 | estudar variações de potencial químico com resolução lateral atômica |
| Kelvin Probe Force Microscope | 1991 | estudar potenciais de contato com resolução lateral de *10nm* |

duz um movimento da ponta de 10nm. O sistema incorpora um mecanismo de realimentação de força, permitindo ao usuário "sentir" moléculas individuais conforme elas são empurradas pela superfície. Pode-se imaginar uma indústria de nanoconstrução em que minúsculas estruturas e máquinas são montadas peça a peça na realidade virtual do nanomanipulador [55].

Podemos notar que nestas duas décadas, grandes avanços do conhecimento tecnológico foram alcançados com a descoberta dos microscópios de varredura por sonda. Hoje temos a oportunidade de visualizar o mundo demasiadamente pequeno da matéria em escalas atômicas, afim de estudar detalhadamente suas propriedades. Outros tipos de microscópios eletrônicos podem ser encontrados em [64, 68].

## 1.6 Organização

Essa dissertação de mestrado está dividida em quatro capítulos:

**Capítulo 1,** que compreende essa introdução, na qual é feita um breve comentário sobre conceito e aplicações de instrumentação virtual. Em seguida é feita uma abordagem conceituai sobre Sistemas de Tempo Real e Linux de Tempo Real. Mais adiante é visto os tipos mais comuns de microscopia de varredura, utilizados na ciência, tais como, microscopia eletrônica, de força atômica e de tunelamento.

**Capítulo 2,** onde são abordadas as restrições do projeto desenvolvido, suas implicações na definição da tecnologia utilizada e o modo funcional do processador *Pentium,* utilizado no teste da placa LDN/UFPE-2001.

**Capítulo 3,** onde é apresentado o projeto da placa de aquisição de dados e geração de varredura, descrevendo o seu funcionamento. Aqui é discutido o funcionamento do PLD em diagrama de blocos e a linguagem VHDL. Em seguida é discutido o funcionamento do controlador de barramento PCI, responsável pela interface do barramento PCI com o mundo externo. Mais adiante o leitor encontrará os resultados das simulações lógicas dos blocos que contém o PLD. A interface gráfica utilizada para o funcionamento da placa também é comentada neste capítulo, além dos resultados experimentais.

**Capítulo 4,** tem-se as conclusões e sugestões para melhorar o desempenho da placa.

No fim deste trabalho, são incluídos os apêndices A, B e C. O **apêndice "A"** contém as listagens dos programas do PLD em VHDL. No **apêndice "B"** estão os símbolos de esquemáticos gerado pelo ambiente de desenvolvimento *MAX+plus II da Altera Corp.,* e finalmente no **apêndice "C"** está a descrição das pinagens e tabelas verdade dos diversos componentes que compõem a placa LDN/UFPE-2001.

# Ccipitulo 2

# Considerações sobre o projeto

A utilização do computador para aquisição de dados traz diversas vantagens, tais como: maior velocidade na aquisição de dados, maior integração entre simulação e análise dos resultados experimentais, operação automática, facilidade de aquisição e controle em períodos longos, reduzida probabilidade de erro, facilidade de implementação de novos programas de análise, menor custo. Esse tipo de interface entre o microcomputador e o mundo externo tem sido utilizada nas mais diversas aplicações, entre as quais pode-se citar: sistema de aquisição para radiação solar [29], sistemas de medição de energia elétrica [24], sistemas de medição de baixo custo [16], etc. Embora existam placas disponíveis comercialmente, observamos dois problemas: nem sempre a solução comercial é adequada à aplicação de interesse, ou o preço da solução comercial é proibitivo. Com isso em mente, desenvolvemos uma placa de aquisição e geração de varredura para ser utilizada em um sistema de microscopia de varredura (eletrônica, força atômica, tunelamento) e outros fins didáticos. Entre os aspectos didáticos que podem ser considerados, tem-se a própria atividade de confecção e funcionamento da placa, o desenvolvimento da interface gráfica e implementação de conceitos avançados de aquisição, tais como: arquitetura cliente servidor [10], sistemas de medição orientada a objeto [45]. Neste capítulo estão apresentadas as tecnologias em que o projeto está baseado.

## 2.1 Principais tipos de barramento

Os microcomputadores modernos utilizam cada vez mais o barramento PCI *(Peripheral Component Interconnect).* O uso do barramento PCI resolve várias limitações dos barramentos mais antigos, tais como, ISA e EISA (Extended ISA) e vem se tornando o padrão de fato na indústria de microcomputadores domésticos e industriais. Os principais tipos de barramento que foram desenvolvido ao longo da história dos microcomputadores compatíveis com a arquitetura IBM, são os seguintes: ISA, MCA, EISA, VESA, PCI [51].

### 2.1.1 ISA - *Industry Standard Architecture*

Padrão de barramento criado em 1984 para os micros IBM PC/AT. Nesse barramento pode-se realizar transferências de dados de 8 ou 16 *bits* operando a 8MHz. Apesar de estar ultrapassado, este padrão ainda é utilizado na indústria.

### 2.1.2 MCA - *MicroChannel Architecture*

Barramento proprietário de 32 *bits* lançado pela IBM em 1987 para os computadores da linha PS/2. Projetado visando multiprocessamento, este barramento permite que as placas de expansão se identifiquem para o sistema, evitando, desta forma, os conflitos que surgem nas configurações manuais de endereço necessárias nos barramentos convencionais. O MCA não é compatível com as placas de expansão ISA.

### 2.1.3 EISA - *Extended Industry Standard Architecture*

16

32

barramento. Projetado como resposta ao MCA, o EISA é compatível com as placas de expansão ISA.

### 2.1.4 VESA *Local Bus*

O padrão VESA foi desenvolvido por um grupo de fabricantes denominado *Video Electronic Standards Association.* Surgiu como uma opção para acomodar periféricos,

principalmente as placas controladoras de vídeo e de disco, capazes de executar transferências de dados de 32 *bits*. Permaneceu durante alguns anos como uma alternativa boa e barata para viabilizar a melhoria da performance do sistema computacional como um todo.

### 2.1.5 PCI - *Peripheral Component Interconnect*

Motivado pelas limitações técnicas do barramento ISA, de *8MHz* e 16 *bits*, a Intel, em 1992, introduziu a especificação do barramento PCI, que permite a transferência de palavras de 32 ou 64 *bits*, a *33MHz*. O PCI possui a característica de universalidade, podendo ser aproveitado por qualquer processador em qualquer arquitetura de máquina IBM PC. Atualmente esse barramento trabalha com relógio *(clock)* de 33 64$MHz$ 132 264 versão da placa LDN/UFPE-2001 opera com 8 *bits*, a uma frequência de relógio de 33$MHz$ 33

32

132

Outra novidade do barramento PCI foi a possibilidade de expansão através de pontes PCI-PCI que cria um barramento secundário isolado eletricamente do barramento raiz, permitindo a utilização de mais placas de expansão. O leitor poderá visualizar na Figura 2.1 a interligação de diversos barramentos PCI.

No barramento PCI quatro sinais de interrupção estão disponíveis nas baias *(slots)*, são eles: INTA#, INTB#, INTC# e INTD#. Assim, o controlador PCI fica responsável por fazer a ligação elétrica conveniente entre a INT# escolhida pelo projetista da placa e a IRQ que será alocada pelo aplicativo de gerenciamento *plug and play*, mais detalhes na próxima secção. Estas quatro interrupções são roteadas para cada uma das baias *(slots)*, sem que haja conflitos, caso duas placas optem por usar o mesmo sinal. É conveniente que placas que usem apenas um sinal de interrupção, utilizem a INTA#.

32

12

operacional tenham a flexibilidade de mapear dispositivos PCI sem que haja qualquer

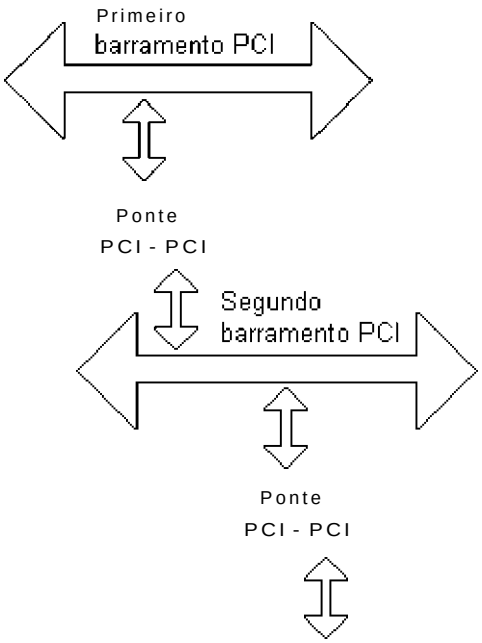



Figura 2.1: Interligação de diversos barramentos PCI

tipo de conflito com placas ISA.

Ao contrário da técnica ISA, a tecnologia PCI não permite que se utilize um canal de *DMA* (Acesso Direto à Memória) fixo para um determinado periférico. Assim a tecnologia PCI, permite que vários periféricos (um de cada vez) compartilhem um mesmo canal. Quando alguma placa ISA exigir *DMA*, estando esta placa conectada a uma baia *(slot)* ligado ao PCI por ponte PCI-ISA, é implementado um mecanismo especial para fazer transferências por *DMA*, ficando transparente para o aplicativo.

Para cada *slot* lógico, a especificação PCI reserva 256 endereços de E/S consecutivos, sendo que os 64 primeiros compõem um cabeçalho para a identificação e programação dos recursos exigidos pelo dispositivo. Dependendo do *chip-set* PCI utilizado na placa mãe do sistema, apenas o cabeçalho poderá ser acessado. Os outros 192 endereços estão disponíveis ao projetista para uso geral [48].

## 2.2 Arquitetura plug and play

Os sistemas antigos de microcomputadores exigiam que os usuários tivessem conhecimentos técnicos. Nessa época, o endereçamento de dispositivos de entrada e saída, nível de interrupção e canais de DMA (Acesso Direto à Memória) eram determinados por intermédio de *jumpers* e chaves do tipo *dip-switch.* A configuração incorreta desses dispositivos poderiam ocasionar conflito elétrico no barramento e consequente travamento do micro. Com a popularização dos microcomputadores, usuários predominantemente leigos passaram a ter que fazer a instalação de periféricos em seu computador. Para solucionar este problema, foi trazida para a arquitetura IB.Vl-PC e compatíveis uma tecnologia que permite a autoconfiguração desses periféricos, denominada *plug and play,* que quer dizer, "conecte e opere".

A intenção dessa tecnologia é permitir que o *firmware* - *BIOS* (Sistema Básico de Entrada e Saída), ou o sistema operacional, instale e configure automaticamente qualquer combinação de placas de expansão ou dispositivos. Com essa facilidade a configuração por *jumpers* ou chaves passa a ser substituída por um programa de gerenciamento, de forma a permitir reconfigurações, durante a execução de um aplicativo. Através dessa tecnologia, cada periférico *plug and play* informa ao sistema operacional quem ele é e quais recursos do sistema ele precisa, tornando a configuração mais simples e rápida. Para isso, o microcomputador precisa ter um sistema operacional *plug*

*BIOS*

féricos.

*BIOS*

operacional e aplicativos compartilhem dispositivos, através de formatos de identificação dos dispositivos, como por exemplo: tipo de placa, fabricante, versão, etc. e também recursos exigidos do sistema, tais como, interrupção, canal de DMA e endereço de memória ou dispositivos.

O processo *plug and play* inicia com a identificação da configuração do micro¬

*BIOS*

*BIOS*

dispositivos, durante a inicialização do computador e armazena as informações dos recursos exigidos.

Durante a inicialização do microcomputador, o processo de configuração básico do acionador *(driver)* do periférico é feito pelo BIOS, enquanto que o restante do processo fica por conta do sistema operacional. Para um sistema ser *plug and play* é necessário que ele contenha um *BIOS plug and play,* o qual dispõe de serviços padronizados de gerenciamento de recursos, para ser usado pelo sistema operacional ou aplicativos.

*BIOS*

através do "cabeçalho de instalação" definido numa área da memória. O cabeçalho é uma estrutura de 64 *bytes,* contendo as rotinas de serviços oferecidos pelo *BIOS,* código do dispositivo, código do fabricante, versão, entre outras informações [48].

**Acesso aos Registradores de Configuração:**

O acesso aos registradores de configuração dos respectivos dispositivos é feito apenas

*BIOS*

operacional *plug and play.* O mecanismo de acesso ao espaço de endereçamento da baia *(slot)* pode ser realizado de duas formas diferentes. A primeira forma é programando uma palavra de controle de 32 *bits* no intervalo de endereço de E/S *CF*8h a *CFFh.* Os dados armazenados neste intervalo contêm campos que especifica o barramento a ser acessado, o número do dispositivo, a função e o índice do registrador de

*CFCh*

pela palavra de controle. Na Figura 2.2, o leitor poderá visualizar esse mecanismo de endereçamento rápido.

Uma outra maneira é proteger o espaço de configuração PCI, através de uma escrita apropriada no endereço *CF * h.* Desta forma cada baia *(slot)* é endereçada com 256 endereços, ou seja, a primeiro baia é endereçada a partir de *C*000h, a segunda

*C*100*h* *C*200*h*

comando fica mapeado no quinto e sexto endereço consecutivo de cada baia, podendo ser acessado pelas instruções *IN* ou *OUT.* Na Figura 2.3, o leitor poderá visualizar o espaço de endereçamento gerado por esse método.

O *chipset* PCI da placa-mãe do microcomputador é responsável pela escolha do mecanismo adotado. A segunda forma tem a vantagem de possuir uma programação bem simples, porém não é indicado nas arquiteturas com múltiplos processadores, pois, o sistema para efetuar a sincronização dos ciclos de barramento, fica bastante

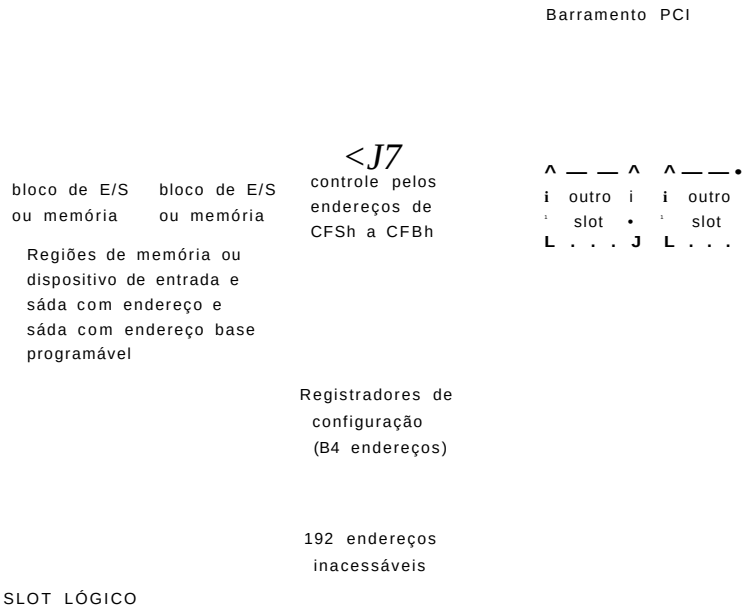



Figura 2.2: Espaço de endereçamento rápido

lento e caro.

O sistema operacional e os aplicativos podem ter acesso aos dados resultantes da configuração de recursos definidos pelo gerenciamento *plug and play,* através do pedido de interrupção INT *lAh,* oferecido pelo BIOS [48].

## 2.3 Microprocessador Pentium

A placa LDN/UFPE-2001 foi testada utilizando um microcomputador com microprocessador *Intel Pentium* e sistema operacional *Windows.* É interessante fazer uma breve revisão sobre os recursos oferecidos pelo microprocessador *Pentium,* juntamente com o sistema operacional *Windows 95.*

### 2.3.1 O microprocessador Pentium

Do ponto de vista do programador, o microprocessador *Pentium* é compatível com os microprocessadores *Intel* 80386 e 80486. Eles têm os mesmos modos de operação com as mesmas características, tais como, proteção de memória, multitarefa e memória virtual, acessando até 4 *gigabytes* (32 *bits* de endereço) de memória RAM e 64 *terabytes* com memória virtual. Diversas características técnicas tornam o *Pentium* muito mais

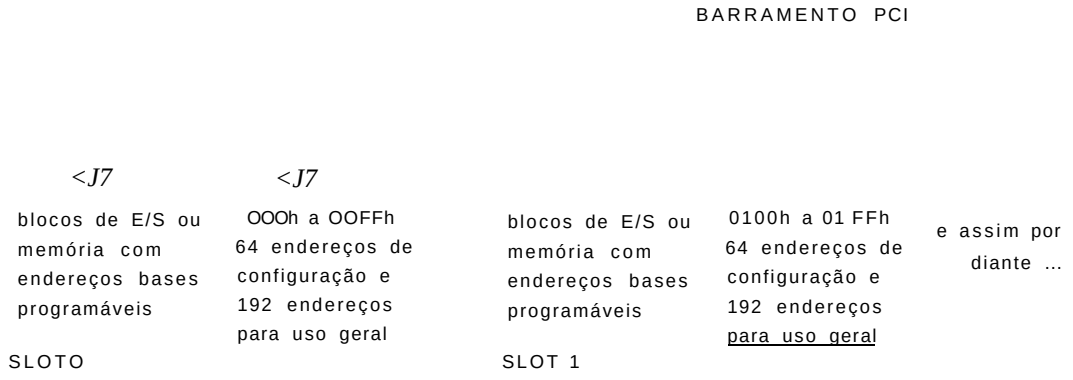



Figura 2.3: Espaço de endereçamento simples

rápido que os seus antecessores, por exemplo:

- Barramento de dados de 64 *bits* - o acesso à memória é feito a 64 *bits* por vez, o que significa uma maior velocidade, pois pode transportar simultaneamente dois dados de 32 *bits;*

- de 16 *kilobytes,* sendo 8 *kilobytes* para instruções e 8 *kilobytes* para dados, aumentando o seu desempenho;

- memória *cache,* os conteúdos dos ramos do desvio, antes de ser concluída a instrução atual de desvio. Desta forma a memória *cache* atua muito mais rápido;

- 80486

  o *Pentium* processa duas informações simultaneamente por pulso de relógio;

  *Pentium* numa mesma placa-mãe, melhorando a desempenho do sistema;

  a cinco vezes mais rápido que os seus antecessores, tornando-os mais eficientes para aplicações científicas;

[1] Memória especial para acelerar a execução de instruções.

- Instrução de identificação - o *Pentium* trás um nova instrução denominada *CPUID,* que identifica sua características. Assim, um programa tem como identificar o processador que está sendo utilizado, facilitando sua configuração.

### 2.3.2 Segmento e deslocamento *(offset)*

Em termos do programa é mais comum representar endereços em forma de segmento e deslocamento *(offset).* A área da memória está dividida em pequenos bancos, onde segmento especifica o banco e o deslocamento *(offset)* é o endereço relativo deste banco. Tomemos como exemplo, o endereço inicial *F*0211000h do registrador de

1

*F0211:000h,* ou seja, endereço inicial (segmento) *F*0211h + deslocamento *(offset) 000h.*

### 2.3.3 Modo protegido de memória

O processador *Pentium* possui dois modos de operação bem distintos: o modo real e o modo protegido. No modo real, a capacidade de endereçamento não é mais que *1Mb* de memória, o que é insuficiente para interfaces gráficas mais avançadas. Esse valor advém do fato que o endereço é obtido deslocando de *4bits* para a esquerda, o conteúdo do registrador de segmento (16biís) e adicionando a esse valor, o conteúdo do registrador de deslocamento *(offset),* formando um endereço de *20bits* ($2^{20}$ ~ 1Mb). A partir do 80286 foi intoduzido o modo protegido. No modo protegido, é possível ter áreas de memória isoladas uma das outras, pois é introduzido o conceito de *descritor de segmentos,* o qual contém informação de proteção de memória. Nesse modo a capacidade de endereçamento é de $2^{32}$ ~ 4Gb. As principais características do modo protegido usado no *Pentium* são:

-

memória RAM do que realmente se tem. A memória extra conseguida através dessa técnica é armazenada em um arquivo do disco rígido, denominado "arquivo

64

*terabytes* de tamanho;

- Proteção de memória - como o *Pentium* acessa muitas áreas de memória, podemos carregar diversos programas simultaneamente. Através da proteção de memória, o *Pentium* é capaz de isolar cada programa em uma área de memória bem definida, de maneira que um programa não invada a área de memória que esteja sendo utilizada por um outro programa;

-

mente onde se encontra cada programa carregado em memória. Dessa forma, pode-se executar automaticamente uma instrução de cada programa, parecendo para o usuário que os programas estão sendo executados simultaneamente, uma vez que que a execução de uma instrução pelo *Pentium* é extremamente rápida (da ordem de *Ins).*

Como o *Pentium* pode acessar várias áreas da memória virtual, é necessário evitar que um programa altere a área do outro, ocasionando uma falha de proteção ou travamento no microcomputador. Para que isso não ocorra é usado o recurso de proteção de memória, onde cada área protegida é designada por um nível de privilégio

0 3 0 3

poderá observar o diagrama dos privilégios da proteção de memória.

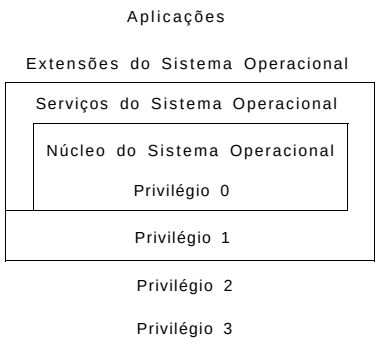



Figura 2.4: Privilégios da proteção de memória

Instruções e dados armazenados em uma área de memória com um determinado

0

3

diretamente a área de memória dos programas, por ter um nível de privilégio maior, porém o *Pentium* não permite que programas acessem diretamente a área de memória

do sistema operacional, por não terem o privilégio necessário. Logo, programas podem acessar outros programas que tenham privilégio menor, mas nunca o contrário.

### 2.3.4 Memória Virtual

Quando a memória RAM do microcomputador está ficando toda ocupada, o *Pentium* comanda uma troca entre uma área da memória RAM que esteja sendo menos utilizada, com uma área do arquivo de troca que esteja vazia no disco rígido. Desta forma, uma porção da memória RAM passará et ficar vazia, não sendo emitida mensagem de erro por falta de espaço na memória RAM. Esse arquivo de troca é a memória virtual. Quando o *Pentium* necessita de um dado na memória virtual, ele faz outra troca, recuperando o dado armazenado no disco rígido. Por outro lado se o arquivo de troca ficar cheio, não haverá como escaparmos da mensagem de erro por falta de memória, a menos que o usuário feche um ou mais aplicativos que estejam executando.

A técnica de memória virtual traz uma solução para falta de memória RAM, mas também traz uma desvantagem. Como a comunicação com o disco rígido é muito lenta em relação ao acesso direto a memória RAM, essa técnica de troca de informação entre a memória RAM e o disco rígido, deixa o sistama mais lento. Assim, deve-se usar uma quantidade maior de memória RAM, diminuindo essas trocas.

Existem duas técnicas de se implementar o recurso de memória virtual: *segmentação* e *paginação:*

- e a memória RAM podem variar de 1 *byte* a 4Gb, que é o limite de memória RAM endereçável pelo *Pentium.* Como o *Pentium* permite que a memória seja dividida em até 16.384 ($2^{14}$) blocos, é possível acessar até 64Tb de memória virtual, caso utilize blocos de 4Gb.

- Na paginação, a memória RAM é dividida em blocos fixos de 4Kb denominados *páginas.* A fixação do tamanho bloco facilita e agiliza o uso da memória virtual. Quando o *Pentium* solicita um dado ou instrução que não se encontra na RAM, mas no arquivo de troca, é transferido um bloco de apenas 4Kb. No caso da

memória virtual que usa o método de segmentação, o bloco pode ser maior (até *4Gb),* tornando a transferência mais lenta [42].

### 2.3.5 Arquitetura dos registradores

16

tivos. Estes registradores são agrupados, conforme a sua utilização. Na Figura 2.5 o leitor poderá visualizar esses registradores.

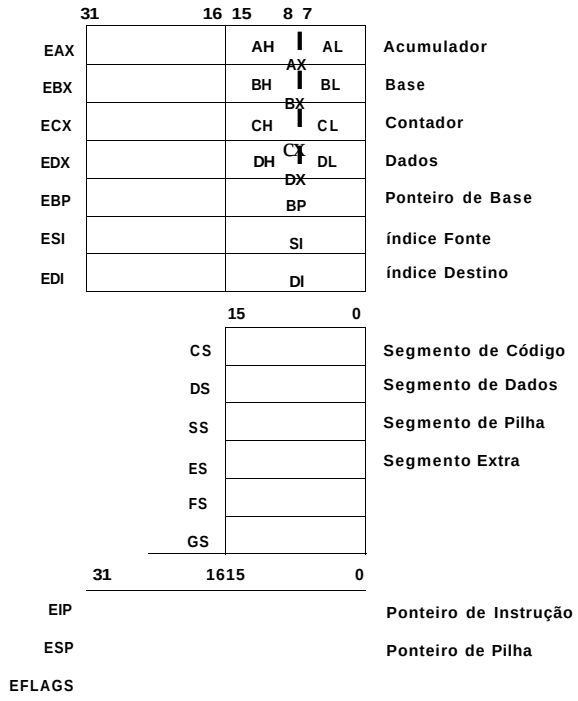



Figura 2.5: Registradores básicos do Pentium

- Registradores de uso geral - foraiado por 8 registradores, cuja utilização é livre para o programador, podendo ser utilizado como operando para instruções lógicas, aritméticas e de cálculo de endereço. Os registradores são de *32 bits* e chamados de: EAX, EBX, ECX, EDX, EBP, ESP e EDI. Cada registrador é

16

8086

8

8

8

- temas a flexibilidade para escolher entre diversos modelos de organização de
  16
  lizados como índices numa tabela em memória. Essa tabela armazena o endereço base para cada segmento e outras informações de acordo com o acesso
  6
  minam, em um dado instante, quais segmentos de memória estão à disposição. Cada registrador está associado a um tipo de acesso à memória: código, dados, ou pilha;

- 32
  EIP, ESP e EFLAGS. O ponteiro de instruções (EIP), contém o deslocamento *(offset)* do endereço da próxima instrução a ser executada pelo *Pentium* dentro do segmento de código (CS). O ponteiro de instrução não está diretamente à disposição do programador. Ele é alterado com a execução das instruções de transferência de controle (desvios, retornos, etc), interrupções e exceções. O ponteiro de pilha (ESP) armazena o deslocamento *(offset)* do topo da pilha, no segmento de pilha atual. Os registradores indicadores de status (EFLAGS) fornecem o tipo de resultado produzido por uma operação lógica ou aritmética. Desvios condicionais e chamadas a sub-rotinas verificam o estado destes indicadores e agem de acordo com o resultado dessa verificação [1].

### 2.3.6 Modos de endereçamento

O *Pentium*® possui uma variedade de modos de endereçamento. Os principais modos são:

- tração, pois, o dado a ser manipulado já vem no campo operando da instrução, não sendo necessário buscá-lo na memória principal. A vantagem desse método é a velocidade de execução da instrução.

  Exemplo: MOV EAX, 12345678h (mova para o registrador EAX de 32 *bits* o valor 12345678 hexadecimal).

Neste método, por outro lado, o valor máximo do dado manipulado é limitado pelo tamanho do campo operando. Outra desvantagem é a necessidade de alteração do valor da variável no campo operando em cada execução que exija um valor diferente.

- 

indica o endereço de memória onde está o dado. O endereço pode ser de todo o dado ou parte inicial do dado quando o mesmo ocupa mais de uma célula de memória. Este modo é também um modo simples de acesso, pois para buscar o dado requer apenas uma referência à memória principal. Porém, mais lento que o modo imediato, já que faz referência à memória.

Exemplo: MOV [7000h], FFh (mover o conteúdo FF hexadecimal para o endereço de memória 7000 hexadecimal).

Com este método, é possível implementar rotinas em que o dado varie de valor a cada execução, através de um programa que utilize uma instrução indicando o endereço do dado previamente armazenado na memória principal.

Uma desvantagem deste modo é a limitação de memória a ser utilizada imposta pelo tamanho do campo operando

- 

o endereço de uma célula que contém o endereço do dado desejado, existindo assim um duplo endereçamento para o acesso do dado desejado.

Esse método elimina o problema da limitação da capacidade de endereçamento do dado do modo direto, pois, com o endereço armazenado na memória é possível estender seu acesso. O modo indireto permite que se faça a movimentação de uma relação de dados para novas posições de memória, bastando para isso, modificar o endereço de acesso ao dado, sem alterar o valor do campo operando.

A desvantagem deste modo é a maior quantidade de ciclos de memória requeridos para completar o ciclo de instrução, pois, é necessário efetuar dois acessos à memória (busca do endereço do dado e busca do dado). Uma outra desvantagem é que em um dado instante só é possível acessar endereços até o limite do campo operando.

assemelham com os modos direto e indireto, com a diferença que a célula de memória referenciada na instrução é substituída por um dos registradores do

cionado por um dos registradores do processador e não mais por uma célula da memória principal.

Este modo de endereçamento por registradores permite que o endereçamento seja feito utilizando uma quantidade menor de *bits*, uma vez que, estes registradores existem em quantidade menor que as células de memória. Assim, o tamanho geral da instrução fica reduzido. Outra vantagem oferecida por este modo é o seu rápido acesso pelo fato do dado está armazenado no registrador e não na memória.

Pode-se utilizar o modo de endereçamento por registrador de duas maneiras: por registrador direto, quando o registrador endereçado na instrução contém o dado a ser manipulado, ou por registrador indireto, quando o registrador referenciado armazena o endereço de uma célula de memória onde se encontra o dado.

Embora o modo de endereçamento por registrador traga várias vantagens, tais como, rapidez de execução da instrução e economia de espaço de armazenamento das instruções, essas vantagens nem sempre são utilizadas. Existem até mesmo casos em que este modo traz desvantagem, ocorrendo desperdício de instruções. Na verdade o uso do registrador só traz vantagem se proporcionar redução dos ciclos de memória.

Outra desvantagem deste modo ocorre em programas que exijam manuseio de grande quantidade de dados (envolvimento de muitos registradores), sendo necessário utilizar outros subsídios, tais como, memória principal.

- **Características dos modos de endereçamento**

de um dado relaciona-se com seu índice, permitindo uma localização dos dados mais rápida e eficiente. O endereço do dado é a soma do valor fixo do campo

Tabela 2.**1:** Registradores de Configuração PCI

| Modo de endereçamento | Definição | Vantagens | Desvantagens |
|---|---|---|---|
| Imediato | 0 campo operando contém o dado. | Rapidez na execução da instrução. | Limitação do tamanho do dado. Inadequado para o uso com dados de valor variável. |
| Direto | 0 campo operando contém o endereço do dado. | Flexibilidade no acesso a variáveis de valor diferente em cada execução do programa. | Perda de tempo, se o dado é uma constante. |
| Indireto | 0 campo operando contém o endereço do dado. | Manuseio de vetores (quando o modo indexado não está disponível). Uso como "ponteiro". | Muitos acessos à memória principal para execução. |

operando e de um valor armazenado em um dos registradores do *Pentium®,* denominado "registrador índice" que contém o valor armazenado variando para o acesso a cada elemento. Como exemplo, considere a necessidade de executar a operação sobre três contagens de 100 elementos cada, em um certo programa:

Programa:

**DO I** = 1 **TO** 100

**C(I) = A(I) + B(I)**

Este modo de endereçamento é uma evolução das técnicas desenvolvidas desde os primórdios da computação para manipulação destas estruturas de dados especiais.

•

campos na instrução: um especificando o endereço de um registrador (chamado base ou segmento) e outro contendo um valor relativo (chamado deslocamento), responsável pelo deslocamento em relação a primeira instrução.

Este modo de endereçamento tem características semelhantes ao modo indexado, pois, o endereço de acesso a uma célula de memória se obtém pela soma de

dois valores. A diferença entre estes modos está na aplicação e na forma de implementá-lo. No modo base mais deslocamento o valor do registrador de segmento se mantém fixo, enquanto o conteúdo do campo deslocamento varia em cada instrução. Já no modo indexado é o conteúdo do registrador que se altera.

O processador *Pentium*® possui seis registradores de 16 *bits* projetados especificamente com a finalidade de servir como registrador de segmento. Este modo de endereçamento reduz o tamanho das instruções, economizando memória e facilita o processo de relocação dinâmica de programas.

$2^{16}$

contendo campo de endereço de registrador-base de 4 *bits* e campo deslocamento de 12 *bits* e que este processador pode endereçar até 16 milhões de células, onde cada endereço linear de memória deverá ter 24 *bits*.

Neste caso pode-se endereçar áreas de 4096 *bytes (4Kbytes)* com um valor armazenado no registrador-base, gastando-se apenas *16 Ms* (4 + 12), ao contrário

$2^{24}$

$2^{8}$

instrução[28, 56].

Esta técnica de endereçamento será adotada no projeto da placa LDN/UFPE-2001 para endereçar a memória não-volátil contida nesta placa.

## 2.4 Inicialização de um computador

Quando o computador é ligado, todos os dispositivos são inicializados em um estado definido. Para muitos dispositivos esse estado é simplesmente zerar seus registradores, para outros é inicializar seus registradores com valores padrões definidos numa memória não volátil externa. O mesmo acontece quando o micro é inicializado a quente *(reset)*.

A arquitetura dos sistemas atuais utilizam o barramento local PCI, a BIOS PCI e a BIOS *plug and play*. A BIOS PCI realiza duas funções principais: ela fornece subrotinas através de *software* para *hardware* específico, por exemplo, placa LDN/UFPE-2001; e inicializa cada tipo dispositivo do sistema PCI, toda vez que o computador

é ligado ou inicializado. Quando isto ocorre a BIOS PCI procura em cada baia *(slot)* PCI um registrador de configuração, quando algum é detectado, o conteúdo é lido para determinar o fabricante, tipo e classe da placa, requerimentos de memória, compatibilidade e características. Quando completa a seqüência, a BIOS PCI grava no dispositivo de E/S, ou espaço de memória as atribuições para cada dispositivo PCI. Dispositivos encontrados em conflito com outros recursos de sistemas, ou que falharam durante seu auto-teste inicial são desativados. Através da Figura 2.6, o leitor poderá ter uma visão geral da interação do sistema BIOS com o controlador PCI, e.g., S5920 da AMCC [50].

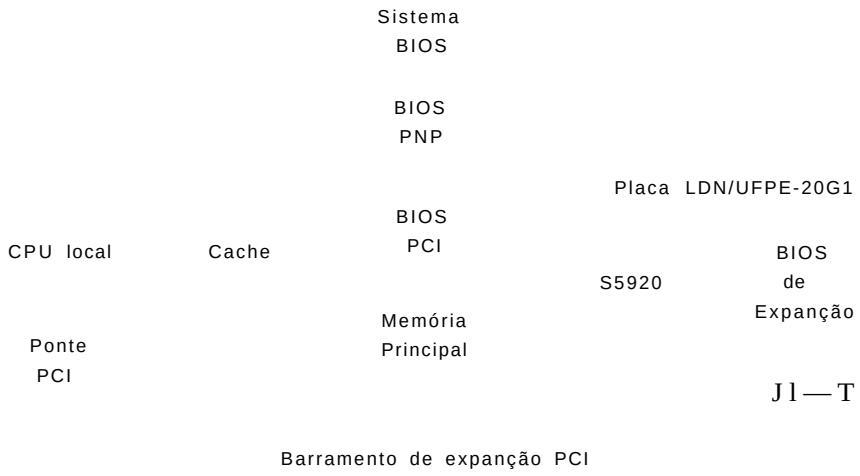



Figura 2.6: Visão geral da BIOS

## 2.5 Inicialização do controlador do barramento PCI

Para implementar pontes de barramento PCI e dispositivos de E/S é necessário configurar os Registradores de Configuração PCI. Quando vários dispositivos PCI estão presentes, estes registradores têm que ser único para cada dispositivo no sistema. O procedimento específico para selecionar a configuração de cada dispositivo envolve um sinal dedicado chamado IDSEL (ver no **apêndice "C",** pino A-26 do barramento PCI), conectado em cada baia *(slot)* PCI da placa mãe.

Após a inicialização do computador, o mesmo executa um ciclo de configuração para cada dispositivo no barramento PCI. O registrador de configuração fornece in-

formação para o *chipset* PCI, indicando memória ou dispositivo de E/S exigido.

Na placa LDN/UFPE-2001 está sendo utilizado o controlador PCI S5920 da AMCC [50]. Após a inicialização, o controlador S5920 pode ser configurado para uma aplicação específica. Para isto ele deve carregar a informação de *setup* do dispositivo contido em uma memória não volátil externa para o registrador de configuração de dispositivo. Quando se deseja usar as regiões *Pass-Thru,* o S5920 necessita de uma memória não volátil externa como dispositivo de *boot,* caso contrário as Regiões de Base de Endereço ficam desabilitadas. No entanto, outros registradores de operação PCI podem ser usados, ficando a Região de Base de Endereço-O no seu estado padrão, definindo uma região de 128 *bytes* de E/S.

Para configurar o S5920 são exigidos 64 *bytes* de informação de *setup,* o restante do dispositivo de *boot* pode ser usado para implementar uma expansão BIOS, se desejado. Algumas informações de *setup* são usadas para inicializar o registrador de configuração PCI do S5920, enquanto outras são usadas para definir os modos de operação especiais do *chip.*

O sinal de entrada "RST#" do barramento PCI (ver no **apêndice "C",** pino A-15 do barramento PCI), quando ativo aciona o sinal de saída "SYSRST#" do barramento adicional (ver no **apêndice "B",** na pinagem do controlador S5920), podendo ser usado para inicializar qualquer máquina de estado externa.

Todos os Registradores de Operação e Configuração do S5920 são inicializados em seu estado padrão no *reset.* Os valores padrão para o Registrador de Configuração serão sobrepostos pelo conteúdo da memória não volátil externa, durante a inicialização do dispositivo. A CPU irá acessar as informações do S5920 na medida que ele carrega as informações da memória não volátil até ocorrer as seguintes situações:

de *boot* válida.

A memória não volátil serial utiliza um protocolo de transferência bi-direcional à dois fios e tem a vantagem de ser de tamanho pequeno, possuindo poucos pinos

0040*h*

0041*h* *FFh*

os próximos quatro acessos são nos endereços 0050/i, 0051/i, 0052/i e 0053/i. Nestas localizações os dados têm que ser 80h (81h ou 82h), FFh, *E*8h e *10h* respectivamente, afim de que a memória não volátil externa seja considerada válida. Quando isso ocorre é lido seqüencialmente da localização 040h até 07Fh O dado é carregado no registrador de configuração PCI apropriado. Alguns dos dados de dispositivo de *boot* 045*h* mas são usados para iniciar alguns modos de operação do S5920. Ao completar esta seqüência, encerra-se o carregamento de *boot* e a configuração de acesso ao S5920 é reconhecida pelo sinal de saída "TRDY#" (ver no **apêndice "B",** pino 19 do S5920).

Dois pinos são usados para transferir dados entre o S5920 e a memória serial: um pino de saída de relógio serial (SCL) e um pino de dados serial bidirecional (SDA). A capacidade de endereçamento do dispositivo serial é de *2Kbytes*. A Figura 2.7, mostra a ligação do S5920 com a memória não volátil serial.

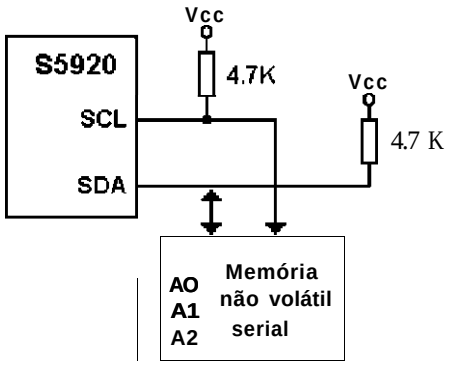



Figura 2.7: Ligação do S5920 com a memória serial

A memória não volátil pode ser acessada pela interface PCI através do Registrador de Controle de *Reset* (RCR) ou pela interface adicional, através do Registrador de Controle de *Reset* Adicional (ARCR) [50].

A placa LDN/UFPE-2001 possui a facilidade de responder a eventos externos em tempo real, graças a operação *Pass-Thru* do controlador de barramento PCI. A operação em tempo real permite o barramento PCI fazer diretamente uma leitura ou escrita do barramento externo. Para facilitar o desenvolvimento da interface gráfica é utilizado o aplicativo Delphi® / Kylix®.

## 2.6 Tecnologia de componentes empregados na placa

**Dispositivo Lógico Programável - PLD.** Os Dispositivos Lógicos Programáveis introduzido pela empresa ALTERA em 1983, são circuitos integrados configuráveis pelo usuário para implementação de funções lógicas diversas. Devido a sua alta integração, alto desempenho e baixo custo, estes dispositivos vem sendo cada vez mais utilizados pelos projetistas. Um PLD pode ser usado como uma máquina de estado (conforme foi utilizado neste projeto), ou decodificador de sinais, substituindo vários componentes discretos que implementam a mesma função. As principais motivações em se utilizar este dispositivo neste projeto foram:

- do projeto;

  a simplificar o desenvolvimento da placa de circuito impresso;

  estoque de componentes durante o desenvolvimento da placa, agilizando o tempo de montagem e diminuindo o número de componentes por placa.

O princípio de funcionamento do PLD é baseado no PAL (Matriz Lógica Programável), com estrutura de entradas programáveis e saídas fixas. Porém, o PLD é composto por vários blocos de PAL, com estruturas de somas de produtos, interligados por um arranjo de conexões programável. Assim, esses blocos poderão funcionar independentemente, ou serem interligados, para formar uma função mais complexa. O bloco básico de um PLD é uma célula lógica de uso geral formada por uma lógica combinacional e *flip-flop* programável, podendo executar a função lógica de qualquer tipo de *flip-flop* (D, T, JK ou SR) e/ou função combinacional.

A ALTERA oferece várias famílias de PLD, tais como, Clássica, MAX 5000, MAX 7000, MAX 9000, FLEX8000 e FLEX10K, onde cada família apresenta uma característica diferenciada.

Neste projeto foi utilizado o PLD da família MAX 7000. Esta família é a segunda geração da arquitetura MAX. Possui lógica variando de 600 a 5000 portas. Os dispositivos tem encapsulamento variando de 44 a 208 pinos, no formato PLCC, PGA, QFP e TQFP. O tempo de atraso lógico pino a pino pode ser de apenas *5ns*, enquanto que a freqüência dos contadores pode chegar a 175, *4MHz*. Os registradores possuem: relógio, controles de relógio e *reset* independentes. As temporizações de registradores podem ser realizadas por arranjos internos, ou sinais globais de temporizações. E oferece opções de saída com dreno aberto, sendo ideais para aplicações de alta velocidade com pontes PCI.

A sua arquitetura inclui os seguintes elementos:

• 128 macrocélulas formando 8 blocos de arranjos lógicos (LAB), onde cada bloco

8

LAB's;

• 8

• 4

1 2

O leitor poderá visualizar em diagramas de blocos a arquitetura do dispositivo, através da Figura 2.8.

Cada macrocélula do PLD pode ser individualmente configurada para operar com lógica seqüencial ou combinacional. As macrocélulas consistem de três blocos funcionais: o arranjo lógico, a matriz de seleção de termo de produto e o registrador programável, como o leitor poderá observar na Figura 2.9.

O arranjo lógico em cada macrocélula é constituído por cinco termos de produto. A matriz de seleção de termo de produto aloca estes termos, como entradas primárias para implementar funções combinacionais, ou como entradas secundárias para entradas de *relógio, preset, clear e habilita relógio (clock enable)* dos registradores.

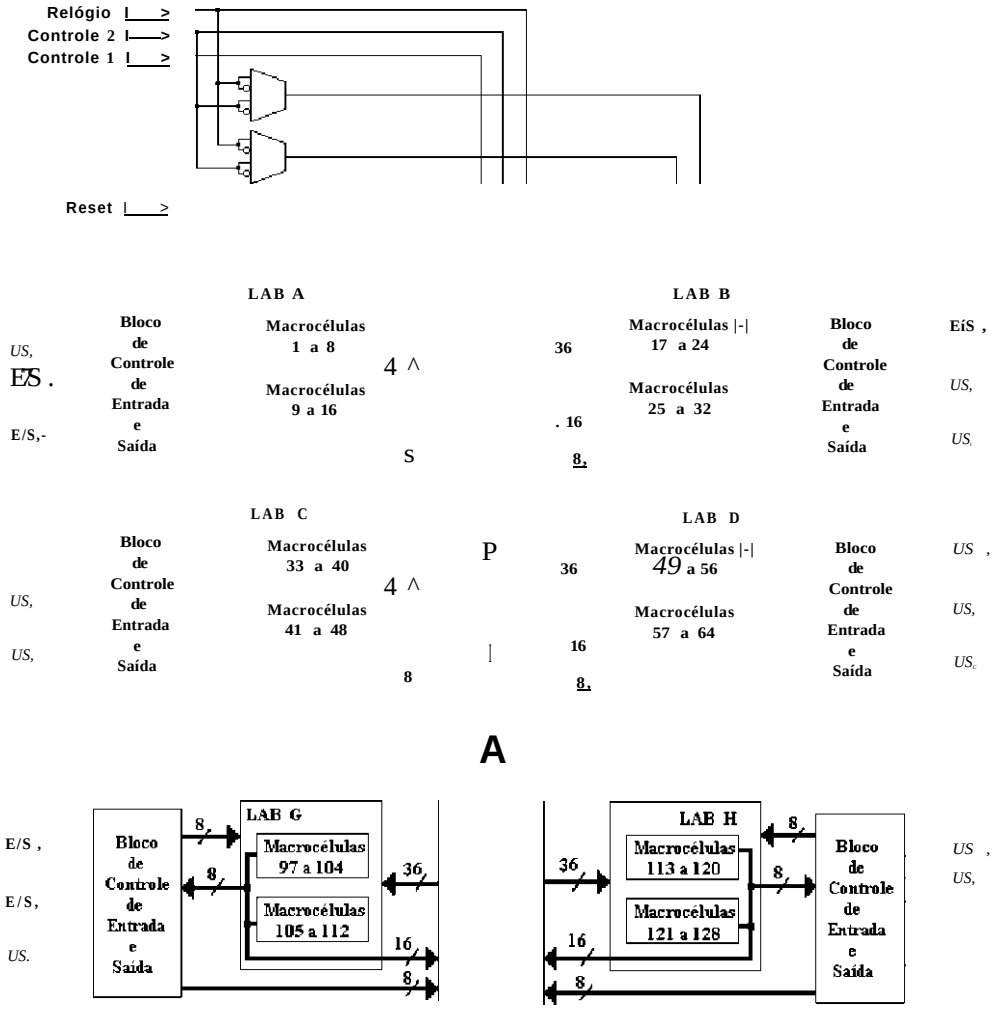



Figura 2.8: Diagrama de blocos da arquitetura do PLD EPM7128SLC84-7

Cada *flip-flop* pode ser individualmente programado para operar como tipos D, JK, T ou SR. Estes registradores podem ser programados para trabalhar com três tipos de relógio:

- Um sinal de relógio global;
- Um sinal de relógio global e habilitado por um sinal habilitação de relógio;
-

O PLD tem disponível dois tipos de expansores utilizados para incrementar recursos de lógica:

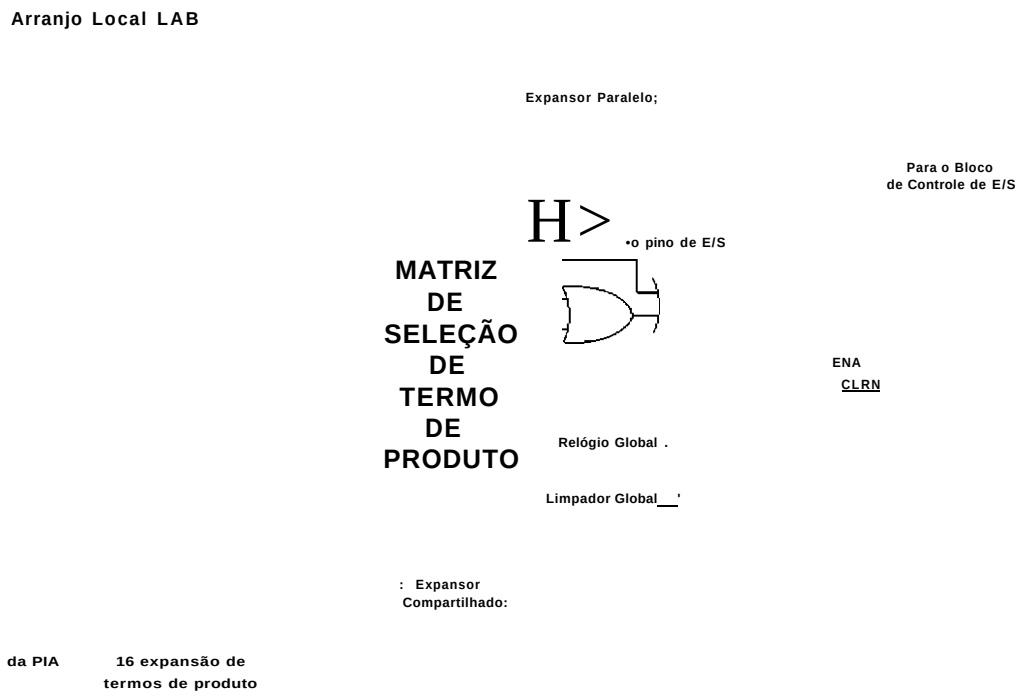



Figura 2.9: Macrocélula do PLD EPM7128SLC84-7

ao arranjo lógico e pode ser compartilhado por qualquer macrocélula no LAB, sendo usado para implementar funções lógicas complexas;

- O expansor paralelo, que são termos produto emprestados de macrocélulas adjacentes. O expansor paralelo permite que até 20 termos produto sejam conectados à porta *OU da* macrocélula.

O ambiente de programação do PLD MAX+PLUS II optimiza automaticamente a locação de termos de produtos de acordo com a lógica exigida do projeto.

O roteamento entre LABs é feito através do PIA, que consiste de um barramento global programável que conecta qualquer fonte de sinal a qualquer destino no dispositivo. Os sinais do PIA são roteados para o LAB, através de uma porta "E" de duas entradas. Uma célula EEPROM controla uma das entradas para a porta, que seleciona um sinal PIA. Desta forma, o PLD tem um atraso fixo devido a essa configuração de portas, fazendo com que o desempenho de temporização seja de fácil previsão.

O bloco de controle de entrada e saída permite que cada pino de entrada e saída seja individualmente configurado para entrada, saída ou operação bidirecional. Todos

os pinos têm um *buffer* com estado de alta impedância que é individualmente controlado por um dos sinais globais de habilitação de saída ou diretamente conectado ao terra ou $V_{cc}$.

Neste projeto foi utilizado o PLD EPM7128SLC84-7. Esse PLD tem um modo de economia de potência que suporta operações em baixa potência em determinados sinais definidos pelo usuário ou em todo o dispositivo. Este modo permite 50% a menos de dissipação energia.

Cada macrocélula pode ser programada para operação de alta velocidade ou baixa potência. Desta forma, certas partes críticas do dispositivo podem operar em alta velocidade, enquanto o restante do dispositivo opera com potência reduzida.

Vários modos de configuração podem ser programados no PLD EPM7128SLC84-7:

- Interface *MultiVolt* de entrada e saída: permite interfacear dispositivos que tenham diferentes tensões de alimentação.
-

  ao nível de sistema.

  alta velocidade nos pinos de saída do PLD [67].

responsável pela interface do barramento PCI com o barramento externo. O S5920 proporciona ao projetista uma maneira flexível e fácil de se conectar ao Barramento Local PCI, eliminando a necessidade do mesmo compreender os complexos diagramas temporais do barramento PCI e o tempo consumido na tarefa para garantir a correta especificação PCI. O S5920 converte sinais do barramento PCI para ser manipulados pelo usuário de forma simplificada através de um barramento local adicional *(Add¬ On Local Bus),* podendo ser de 8, 16 ou 32 *bits.* Os pinos do barramento local adicional do S5920 proporcionam ao projetista uma estrutura de barramento muito mais simples para interface E/S, memória ou aplicação de aquisição de dados, inclusive para projetos de portas ISA existentes no barramento PCI. O barramento pode ser operado de maneira síncrona ou assíncrona para o barramento Local PCI, podendo o usuário definir a velocidade do relógio de 8 a *40MHz.*

O S5920 é capaz de transferir dados de 32 *bits* a uma taxa de *Yò2MBps*. Suporta dois canais de barramento de nvRAM serial e pode tanto operar no modo passivo, quanto no modo ativo. No modo passivo o controle é feito por um dispositivo externo, enquanto que no modo ativo o S5920 é responsável pelo controle do sistema. A opção é feita através de um pino de entrada denominado PTMODE que em nível alto define o modo passivo e em nível baixo define o modo ativo. No projeto desta placa o S5920 trabalha no modo ativo, logo a entrada PTMODE deve está em nível lógico zero.

A arquitetura do S5920 é optimizada através de três grupos de registradores: Registradores de Configuração PCI, Registradores de Operação PCI e Registradores de Operação de Barramento Externo.

• Registradores de Configuração PCI

Tabela 2.2: Registradores de Configuração PCI

| Byte3 | Byte2 | Bytel | ByteO | Endereços |
|---|---|---|---|---|
| Identificação do Dispositivo | | Identificação do Fabricante | | 00 h |
| Status PCI | | Comando PCI | | 04 h |
| Código de Classe | | | Ident. Revisão | 08 h |
| Auto Teste | Tipo Cabeçalho | Temp.Latência | Tam.CacheLine | OC h |
| Registrador de Endereço Base 0 | | | | 10 h |
| Registrador de Endereço Base 1 | | | | 14 h |
| Registrador de Endereço Base 2 | | | | 18 h |
| Registrador de Endereço Base 3 | | | | 1C h |
| Registrador de Endereço Base 4 | | | | 20 h |
| Registrador de Endereço Base 5 | | | | 24 h |
| Espaço Reservado | | | | 28 h |
| Identificação do Subsistema | | Identificação do Fab. Subsistema | | 2C h |
| Endereço Básico ROM de Expansão | | | | 30 h |
| Espaço Reservado | | | | 34 h |
| Espaço Reservado | | | | 38 h |
| Latência Máx. | Gradiente Min. | Pino de Inter. | Linha de Inter. | 3C h |

Os códigos (somente de leitura) do fabricante e do dispositivo identificam a empresa e o tipo de controladora da placa. No caso do S5920 o código do fabricante é 10E*8h* e do dispositivo é 5920h.

O registrador de comando (leitura e escrita) é responsável pela programação de alguns parâmetros de acesso, como permissão para manusear memória e dispositivos de E/S contidos na placa, habilitação de alguns ciclos especiais de barramento, etc.

O registrador de estado (leitura e escrita parcial) informa algumas condições de velocidade de comunicação e de detecção de erros.

O identificador de revisão (leitura) informa o número da revisão do adaptador, sendo um código criado pelo fabricante. No caso do S5920 o valor é 00h.

O código da classe (leitura) identifica a função básica do dispositivo, por exemplo, multimídia, ponte PCI-outro barramento, placa de rede, etc. No caso do S5920 esse código é formado por três registradores de *8bits:* classe de base na posição de *offset 0Bh* com o conteúdo 08h, indicando periférico de sistema de base; sub-classe na posição 0Ah, com o conteúdo 80h, indicando outros periféricos de sistema e Prog I/F na posição 09h, com o conteúdo 00h, indicando outros periféricos de sistema.

O tamanho da cache (leitura e escrita) define o tamanho em *doublewords* da linha de cache para escritas na memória da placa. Usado apenas em barramento mestre.

00*h*

O temporizador de latência (leitura) especifica o tempo mínimo do relógio do barramento, que o barramento mestre pode manter a posse do barramento. Ele é decrementado pelo barramento mestre a cada pulso de relógio a partir do início da transferência de um dado. No caso do S5920 não é utilizado, portanto o seu conteúdo

00*h*

O tipo de cabeçalho (leitura) define se o dispositivo é um adaptador ou uma ponte PCI-PCI. O BIST *(Buil In Self Test* - leitura) é um registrador opcional que permite efetuar uma seqüência de auto-teste do adaptador. No caso do S5920 o conteúdo é 0£h.

Os registradores para programação de endereços bases são necessários quando o dispositivo implementa decodificadores de endereços para mapearem porções de memória e dispositivo de E/S. O bit menos significativo destes registradores é reservado para a leitura e indica o tipo se é endereço de E/S ou de memória.

Os registradores de identificação de subsistema fazem a distinção dos dispositivos que apresentam o mesmo número de identificação do fabricante e do dispositivo.

O registrador de endereço base ROM de expansão é utilizado por placas que contêm ROMs que participarão da seqüência de inicialização do computador (POST¬ *Power On Self Test)* ou que oferecem serviços de interrupção por *software.*

O registrador do pino de interrupção (leitura) indica por qual sinal elétrico do bar-

ramento, a placa LDN/UFPE-2001 gerará o pedido de interrupção (INTA#, INTB#, INTC# ou INTD#), programando-se 1 a 4, ou não exigindo recurso de interrupção, 0

O registrador de linha de interrupção (leitura e escrita) é programável pelo *software plug and play* (BIOS ou sistema operacional) e informa à placa qual linha IRQ está alocada para o adaptador.

Os registradores gradiente mínimo e latência máxima não são utilizados pelo dispositivo S5920. Portanto seus conteúdos são 00h.

Este grupo é mapeado em seis registradores de *32bits,* alocados no espaço de endereço de memória ou dispositivo de E/S, especificado pelo registrador de endereço de 0

entre os barramentos PCI e adicional.

Tabela 2.3: Registradores de Operação PCI

| Registradores de Operação PCI | Endereços |
|---|---|
| Caixa de Mensagem de Saída (OMB) | *OCh* |
| Caixa de Mensagem de Entrada (1MB) | *Wh* |
| Estado Vazio/Cheio da Caixa de Mensagem (MBEF) | 34/i |
| Controle/Estado de Interrupção (INTCSR) | 38/i |
| Controle de *reset* (RCR) | *3Ch* |
| Configuração "Pass- Thru" (PTCR) | *60h* |

O OMB define os métodos para envio de dados de comando ou parâmetros para o barramento adicional em 8, 16 ou *32bits.*

O IMB define os métodos utilizados para receber os dados de estados ou parâmetros do barramento adicional em *8,* 16 ou *32bits.* Apenas operação de leitura é feita nestes registradores.

O MBEF indica se os conteúdos da caixa de mensagem de entrada e saída estão cheios ou vazios.

O INTCSR indica o período de interrupção e identifica a fonte de solicitação da interrupção.

O RCR consiste em reinicializar os controles de acesso ao barramento adicional, bandeiras *(flag)* de estados cheias/vazias da caixa de mensagem, memória externa de escrita/leitura não volátil e FIFO de leitura *Pass-Thru.*

O PTCR controla a configuração das quatro regiões *Pass-Thru.*

O barramento de interface adicional fornece acesso a oito registradores de *32bits,* contendo dados, controle e informação de estados. Todos estes registradores são acessados através do pino "SELECT#" do barramento adicional e pinos "BE[3:0]" em conjunto com os outros controles de escrita e leitura habilitados. Todos registradores são acessados com sinais sincronizados com o relógio do barramento adicional.

Tabela 2.4: Registradores de Operação do Barramento Adicional

| Registradores de Operação Adicional | Endereços |
|---|---|
| Caixa de Mensagem de Entrada (AIMB) | *OCh* |
| Caixa de Mensagem de Saída (AOMB) | *ICh* |
| Endereço *Pass- Thru* Adicional (APTA) | *28h* |
| Dados *Pass- Thru* Adicional (APTD) | *2Ch* |
| Estado Vazio/Cheio da Caixa de Mensagem Adicional (AMBEF) | 34/i |
| Controle/Estado de Interrupção Adicional (AINT) | *38h* |
| Controle de *Reset* Adicional (ARCR) | *3Ch* |
| Configuração "Pass- Thru" Adicional (APTCR) | *60h* |

O registrador AIMB de *32bits* fornece um método para receber estados definidos pelo usuário, ou dados de parâmetros do sistema PCI. Apenas operações de leitura de

8 16 32 *bits*

uma interrupção de barramento PCI (quando desejado) se gerado uma interrupção habilitada, através do uso do registrador de controle de interrupção/estado.

32*bits*

mando, ou dado de parâmetros para a interface PCI. Operações do barramento adi-

8 16 32*bits*

esses registradores pode ser uma fonte para interrupções de barramento PCI (quando desejado) por geração de interrupção habilitada, através do uso do registrador de controle/estado de interrupção. Esse registrador também é chamado de IMB. O byte três

dessa caixa de mensagem pode também ser controlado via porta externa. A leitura desse registrador não irá afetar interrupções ou registrador de estado do AMBEF.

O registrador APTA guarda o endereço de qualquer ciclo de barramento PCI *Pass-Thru* ativo, aceito pelo S5920. Quando um dos quatro registradores de decodificação de endereços base encontra um ciclo de barramento PCI que seleciona a região definida por ele, esse registrador guarda o endereço ativo do ciclo corrente. Esse endereço é incrementado após cada transferência de dados *Pass-Thru* de *32bits*.

O registrador APTD junto com o registrador APTA é usado para executar transferências tipo *Pass- Thru*. Quando um dos registradores de decodificação de endereço base 1-4 encontra um ciclo de barramento PCI que seleciona a região definida por ele, o registrador APTA irá conter o endereço ativo atual, enquanto que o registrador APTD irá conter o dado (quando se tratar de uma escrita no barramento PCI), ou deverá ser escrito com dado (quando se tratar de uma leitura no barramento PCI). Em modo passivo estados de espera são gerados no barramento PCI até que haja a leitura ou escrita nesse registrador.

O registrador AMBEF indica se os conteúdos das caixas de mensagens então vazios ou cheios.

O registrador AINT permite escolher quais condições devem ocorrer para produzir uma interrupção na interface do barramento adicional, permite visualizar a causa para a interrupção e reconhecer (remover) a solicitação de interrupção.

O registrador ARCR fornece um método para controle de *resets* por *software* e acessos a memória não volátil.

O registrador APTCR controla a configuração das quatro regiões *Pass-Thru* PCI (PTCR). É esperado que o sistema PCI ou interface adicional local escreva neste registrador, mas não ambos. Esse registrador é também usado para inicializar qualquer operação do barramento PCI. Operação *Pass-Thru* não pode ser garantida se a atualização for feita no momento em que uma transação *Pass-Thru* estiver sendo executada.

A interface *Pass-Thru* do S5920 permite a transferência de dados de dois modos, o modo ativo (quando seu pino PTMODE encontra-se em nível lógico "0") ou modo passivo (quando seu pino encontra-se em nível lógico "1"). No primeiro caso o controle da placa é feito pelo próprio S5920. No segundo caso o controle da placa é feito por

um dispositivo externo ao S5920, no nosso caso o PLD. Acessos a região *Pass-Thru* podem ser usados para executar ciclos de barramento PCI em tempo real ou através de um FIFO interno. A operação de tempo real permite que o barramento PCI leia ou escreva diretamente os dados com o barramento externo. O S5920 permite ao projetista renomeie quatro regiões individuais de *Pass-Thru.* Apesar de cada uma dessas regiões poderem ser definidas com 8, 16 ou *32bits,* mapeadas na memória ou no espaço do sistema de entrada e saída, podendo ocupar 512Mb. Para facilitar o projeto, preferiu-se utilizar *8bits.* A Figura 2.10 mostra o diagrama em bloco da arquitetura *Pass-Thru* do S5920.

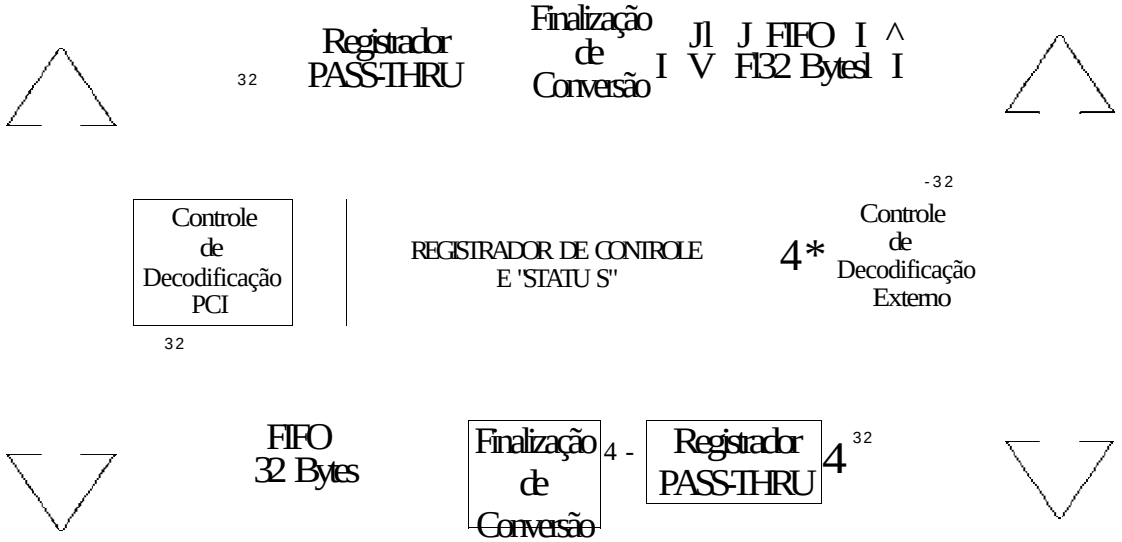



Figura 2.10: Diagrama em bloco da arquitetura *Pass-Thru* do S5920

Memória - Em sistemas embarcados os tipos de memória utilizados podem ser bastante diferentes daqueles usados em sistemas de uso geral. Em sistemas embarcados não é comum utilizar discos rígidos ou outros tipos de armazenamento externo. Dependendo da sua aplicação, o tamanho da memória pode ser muito pequeno, por exemplo, em aplicações de controle normalmente a memória de dados não passa de algumas dezenas ou milhares de *bytes.* Os sistemas embarcados são projetados para trabalhar ininterruptamente sem falhas, operando o mais rápido possível ao serem ligados. Por exemplo, não é admissível que o motorista de um carro fosse obrigado a esperar alguns minutos até que o controle do freio ABS *(Anti-Break System)* ou da injeção eletrônica estivessem operacionais após o carro ser ligado. Assim, é muito comum que memórias não voláteis sejam usadas para guardar dados de con-

figuração do sistema, de modo que os dados possam ser recuperados mesmo se o sistema for desligado e religado. As memórias não voláteis normalmente utilizadas são do tipo EEPROM (memória de leitura programável e apagável eletricamente), que são memórias cujos dados podem ser apagados *byte* por *byte* de forma independente, ou *Flash* EPROMs, cujos dados são apagados em blocos. A escolha do tipo de memória a ser usado dependerá da sua aplicação, uma vez que, as EPROMs são mais flexíveis, porém mais lentas no apagamento. As memórias voláteis são geralmente do tipo SRAM (memória estática de acesso aleatório), uma vez que os processadores empregados não dispõem de unidades de gerenciamento de memória sofisticadas o bastante para prover o *refresh* necessário à manutenção de dados nas memórias DRAM (memória dinâmica de acesso aleatório). Em sistemas embarcados é comum utilizar memórias SRAM alimentadas por baterias de maneira a manter os dados em caso de falta de energia. Existem *chips* dessas memórias que já vêm com baterias de lítio internas capazes de manter os dados por até 10anos [43] e [49]. A placa de aquisição de dados é composta por duas memórias SRAM independentes como veremos adiante.

**Conversor AD** - O conversor AD (Analógico-Digital) é responsável pela conversão do sinal analógico em digital. Neste trabalho é utilizado o dispositivo TDA8703. O TDA8703 é um conversor analógico-digital de *8bits,* com alta velocidade de conversão para vídeo e outras aplicações. Ele converte o sinal de entrada analógico em palavras digitais de *8bits* a uma **taxa** máxima de amostragem de *40MHz.* Todas as entradas e saídas digitais são compatíveis com TTL. Outra característica do TDA8703 é que ele não requer circuito de *sample-and-hold,* pois, internamente ele possui um *latch* que faz o armazenamento dos dados digitais convertidos, ou na transição de subida, ou de descida do relógio, conforme o gosto do usuário. A saída do dispositivo fica em alta impedância, até que o pino de controle "CE#" seja colocado em nível baixo. O resultado da conversão da amostra pode ser dado tanto em binário (com TC# em nível alto), quanto em complemento a dois (com TC# em nível baixo).

**Conversor DA** - O conversor DA (Digital-Analógico) é responsável pela conversão do sinal digital em analógico. Neste trabalho é utilizado o dispositivo TLC7524CN. Esse dispositivo CMOS, de *8bits,* permite que se faça a interface com a maioria dos microprocessadores populares de maneira facilitada. O erro de linearidade é de no máximo a metade do valor do *bit* menos significativo. Apresenta um baixo consumo

(máximo de 5mW com $V_{DD}$ = 5V), sendo um dos dispositivos mais velozes na conversão de digital para analógico *(set time* = 100ns).

## 2.7 Linguagem para descrição de hardware VHDL

A linguagem VHDL (Linguagem para Descrição de Hardware de circuito integrado com velocidade Muito alta) é uma linguagem padrão com características especiais para simulação de circuitos. Nela é possível a introdução de aspectos relativos ao comportamento do circuito (tempo de subida, tempo de descida, atrasos de portas lógicas e operação funcional). Assim como linguagens de alto-nível, VHDL permite que projetos de circuitos complexos possam ser descritos como programas de computador, e que o comportamento de circuitos eletrônicos complexos possam ser capturados dentro de um sistema de projeto para síntese automática ou para simulação. Diferente de outras linguagens de programação, VHDL é inerentemente concorrente em suas operações. VHDL é uma linguagem em aperfeiçoamento e novas versões estão sendo constantemente discutidas. Atualmente seu alcance vai desde a descrição comportamental até o nível de portas lógicas no mundo digital. No entanto, já existem versões VHDL para descrever circuitos analógicos, Analog VHDL (AVHDL), bem como tendências para usá-la como linguagem de descrição de sistemas [43].

Para programação do PLD foi utilizado o aplicativo MAX+PlusII® *Student Edition,* versão 9.23 da Altera®. Esse aplicativo permite a captura esquemática e entrada de projeto de linguagem de descrição de *hardware,* baseada em texto, incluindo VHDL e AHDL (Altera Hardware Description Language). Possui ainda, programação de projetos, compilação e suporte de verificação para o dispositivo PLD da placa de aquisição.

## 2.8 Interface com o usuário

A programação orientada a objeto é amplamente usada no desenvolvimento de sistemas de medições. Esse ambiente traz algumas vantagens, tais como, a facilidade dos instrumentos poderem ser representados como objetos e a possibilidade de criar uma hierarquia extensa dos instrumentos [21].

Para comandar a placa de aquisição de dados pode ser criada uma interface gráfica utilizando o aplicativo Delphi® no ambiente Windows ou Kylix® no ambiente Linux. Esse tipo de interface apresenta várias vantagens, tais como: execução rápida do evento solicitado, interação facilitada através do *mouse* entre o usuário e a máquina, procedimento de eventos ativos pelo usuário.

Para o projeto de um exemplo de interface gráfica da placa LDN/UFPE-2001 foi

®

1. Definição das janelas visualizadas pelo usuário;
2. Definição dos eventos de cada janela;
3. Definição dos procedimentos de eventos para os eventos;
4. Definição dos procedimentos auxiliares, responsáveis pelo funcionamento dos procedimentos de eventos.

®

os possíveis eventos, sendo reconhecível através de uma tecla pressionada, movimento do *mouse*, ou clique do seu botão. Na Figura 2.11 o leitor poderá visualizar uma possível interface gráfica para comando da placa LDN/UFPE-2001 de aquisição de dados e geração de varredura.

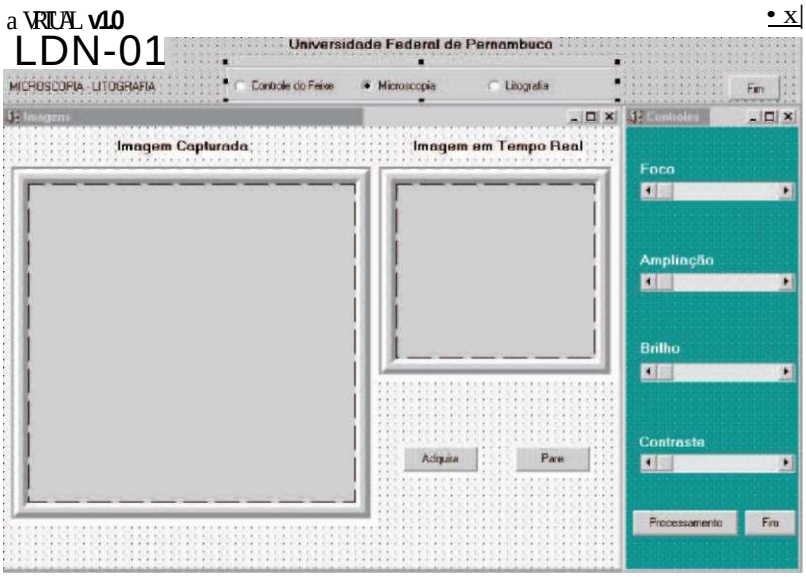



Figura 2.11: Exemplo de interface gráfica para comando da placa LDN/UFPE-2001

## 2.9 Sistemas aquisição de imagens

Atualmente sistemas de aquisição de imagens têm encontrado uma larga faixa de aplicações comerciais e científicas, trazendo avanços tecnológicos na área da biomédica, astronomia, espectroscopia, segurança, inspeção industrial. Principalmente no que diz respeito ao processamento de imagens, utilizando computadores conectados a câmeras. Em resumo, existem diversos tipos de placas comerciais para aquisição de imagens que utilizam a interface PCI, entretanto elas nem sempre apresentam desempenho adequado em certas aplicações. A placa LDN/UFPE-2001 foi projetada para apresentar um desempenho optimizado na aquisição de imagens com posicionamento da sonda.

O desenvolvimento desta aplicação em geral, tem explorado sensores ópticos de estado sólido. Esta tecnologia tem conduzido a inúmeros trabalhos revolucionários, devido ao seu baixo consumo, baixo ruído, alta resolução, alta fidelidade geométrica e alta sensibilidade para uma larga faixa de comprimentos de onda.

Trabalhos recentes nesta área enfocando detectores de estado sólido, com ênfase também em microscopia, podem ser encontrados em conferências organizadas pela SPIE - *The International Society for Optical Engineering* [73] e pela IS&T - *The Society for Imaging Science and Technology* [62].

Trabalhos no campo da microscopia multidimensional (em três ou quatro dimensões), utilizando sistemas baseados em microscopia confocal, incluindo captura, processamento e análise de imagens podem ser encontrados na Universidade de Manchester [59]. A flexibilidade da imagem confocal tem conduzido a muitas pesquisas interdisciplinares com aplicações que vão desde as ciências biológicas (células, tecidos, etc.) à física (polímeros). Para sistemas que utilizam microscopia de luz ou varredura de elétrons pode-se encontrar câmeras digitais de alta performance com uma ou três saídas do tipo digital ou analógica, coloridas ou em preto e branco [3].

### 2.9.1 Aquisição de imagens baseada em FPGAs

O FPGA *(Field Programmable Gate Array)* permite que se faça a implementação das funções PCI necessárias em um único *chip,* permitindo ainda a sua reconfiguração em protótipos.

O projeto de interface PCI também são disponíveis em blocos funcionais *(cores).* Assim grandes sistemas podem ser convenientemente projetados pela combinação de diferentes *cores,* podendo ser adquiridos gratuitamente ou comercialmente por alguns fabricantes de FPGAs.

Um projeto de interface PCI utilizando um FPGA e o recurso do microcomputador, para desempenhar a função de um neurocomputador paralelo, pode ser visto na referência [25]. Uma melhora significativa foi obtida utilizando a interface de barramento PCI, ao invés da interface ISA. A tarefa da interface é "bufferizar" os dados do barramento periférico do microcomputador com os barramentos dos neurocomputadores.

O trabalho de referência [2] trata da criação de um sistema de aquisição de imagens de vídeo, composto por um circuito integrado de tratamento de sinais e um sistema implementado em componentes de *hardware* FPGA, utilizando a linguagem VHDL. Como entrada do sistema foi utilizado o Cl Philips SAA7111, que tem como função a conversão de sinais analógicos de vídeo para sinais digitais. Para controlar a captura de imagens foi implementado em *hardware* FPGA um sistema responsável por toda a configuração do SAA7111, incluído velocidade de aquisição de imagens e o tipo de sinal de entrada e de saída. Esta configuração é feita através da troca de mensagens via comunicação serial entre o SAA711 e o FPGA, sendo necessário a implementação de sub-rotinas para o envio e recebimento de dados.

Infelizmente o sinal rápido de relógio do barramento de 33MHz do microcomputador e as severas exigências elétricas limitam o uso desses FPGAs. Assim, no projeto da placa LDN/UFPE-2001 será utilizado o controlador de barramento PCI, denominado S5920 da AMCC, podendo atingir uma frequência de *40MHz.*

### 2.9.2 Aquisição de imagens utilizando o microcomputador

O sistema computadorizado de apoio à microscopia óptica orientado a objeto atua da seguinte forma (Figura 2.12): uma amostra é colocada no microscópio óptico para observação, sua imagem primária é capturada por uma câmera de vídeo, acoplada ao microscópio óptico. O sinal elétrico proveniente da câmera é adequadamente digitalizado por uma placa de aquisição, permitindo que a imagem nela obtida venha ser

visualizada e analisada, através de um *software* de análise de imagens. A imagem obtida da câmera é enviada para um ambiente computacional, onde a imagem digital obtida poderá ser explorada de diversas maneiras (armazenando, imprimindo, realizando um processo digital sobre a imagem). Por meio de um projetor multimídia o professor poderá ainda, demonstrar e identificar as características de interesse no exato momento em que está explicando para os alunos [3] [23].

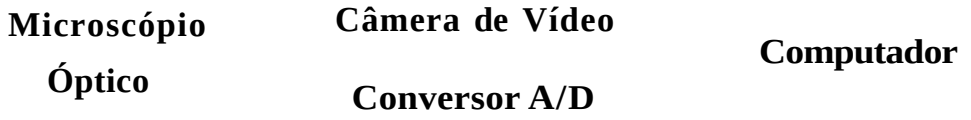



Figura 2.12: Aquisição da imagem digital

Um exemplo de aplicação é no ensino de materiais para engenharia, principalmente no que se refere as aulas práticas de metalografia, baseia-se normalmente no uso de microscópios pelos alunos e professores. Isso causa uma certa dificuldade, pois nem sempre ambos conseguem visualizar os fenômenos em destaque ao mesmo tempo. Para tentar solucionar esse problema, muitos docentes recorrem ao uso de fotografias previamente obtidas e reveladas. Entretanto, nem sempre a foto corresponde à amostra analisada, além de ter que esperar a fotografia ficar pronta. O desenvolvimento de recursos computacionais de análise de imagens vem de encontro a esses problemas. O analisador de imagens passa a ser uma ferramenta que permite ilustração real e instantânea desses conceitos, dando ao aluno maior clareza e envolvimento na interpretação de fenômenos ocorridos ou gerados no interior dos materiais em questão, e possibilitando uma análise mais aprofundada, tanto dos aspectos microscópicos, como macroscópicos.

O desenvolvimento de um sistema automático de varredura e aquisição de imagens das amostras preparadas para observação em microscópios ópticos pode ser visto em [37]. O sistema baseia-se em um microcontrolador da família MCS-51, que comanda o acionamento de três motores de passo acoplados ao mecanismo de movimentação da mesa XY do microscópio e ao seu controle de ajuste de foco. Este microcontrolador interage com o operador de um sistema através de um teclado dedicado e de um visor auxiliar, e comunica-se com um microcomputador do tipo PC via

canal de comunicação serial, constituindo-se em um elemento inteligente de controle do sistema. As imagens das amostras são captadas por uma câmera de TV acoplada ao microscópio, sendo visualizadas em um monitor de TV e digitalizados para processamento no microcomputador. O sistema foi testado com finalidades de análises de amostras sanguíneas, tendo mostrado bastante eficiente e valioso na obtenção de diagnósticos médicos.

### 2.9.3 Dimensionamento da resolução gráfica

A resolução com que a placa LDN/UFPE-2001 envia suas imagens para serem visualizadas no computador é uma característica muito importante nas aplicações em que é necessária uma alta precisão no detalhamento da representação da amostra. Essa resolução é medida normalmente na proporção 4:3, através da quantidade de linhas horizontais e colunas verticais exibidas na tela. Quanto maior for a resolução, maior será o nível de detalhamento na representação da imagem. Assim, a tela é considerada, para efeitos gráficos, como uma matriz normalmente regular de pontos chamados *pixels*. Uma resolução, por exemplo, de 600 x 800 em monitor de 15", indica que a tela é uma matriz de *600pixels* no sentido vertical por 800pixels no sentido horizontal.

Outra característica importante para obtenção do realismo na representação de imagens e maior detalhamento é o número de cores. Alguns anos atrás, podia-se encontrar placas operando com *2*, $2^4$ = 16 ou $2^8$ = 256 cores. Atualmente existem placas que operam com elevadíssimo número de cores, tais como, HI-COLOR (utilizando $2^{16}$ = 65.536 cores) TRUE COLOR (utilizando $2^{24}$ = 16.777.216 cores, 8 bits para cada cor R, G, B). Assim é possível representar com maior aproximação as quase 20

de fotos coloridas.

De um modo geral, quando se diz que uma tela possui a resolução A x B x C, significa que ela é formada por A pontos no sentido horizontal, B pontos no sentido vertical, podendo cada um desses pontos assumir C cores diferentes [44]. Por exemplo,

640x 480

640 x 480 x 4 256

tem-se uma imagem com o formato 640 x 480 x 256.

# Capítulo 3

# Projeto e realização

Neste capítulo são abordadas as considerações de projeto da placa LDN/UFPE-2001, os detalhes da arquitetura, a elaboração do leiaute e os testes experimentais realizados. A função da placa é a geração de dois sinais de varredura ( X e *Y)* para posicionamento de uma "*ponta de prova!'* que irá excitar a amostra, o sinal em resposta a essa excitação também é coletado pela placa. Uma aplicação possível é a conexão da placa com um microscópio eletrônico de varredura, como ilustrado na Figura **3.1.** O sistema funciona da seguinte forma: um feixe primário de elétrons é emitido do canhão eletrônico em direção ao objeto em estudo. Os elétrons são acelerados por uma tensão elétrica elevada em um ambiente de vácuo para minimizar as colisões com moléculas do ar. Parte deles é refletida ao se deparar com o substrato em observação, outros geram elétrons secundários ao se colidir com os átomos do objeto em estudo. Os elétrons refletidos ou emitidos são coletados por detectores específicos. Essas informações analógicas são levadas à placa de aquisição de dados que se encontra dentro do computador que por sua vez, são tratadas e enviadas para a tela do mesmo, dando condições ao usuário de poder manusear as imagens em estudo através de um *mouse* ou *joystick*. O gerador de varredura é responsável pelo desvio do feixe do canhão eletrônico através de bobinas defletoras que varre cada ponto do objeto que está sendo observado. Nesta aplicação, o feixe é a *"ponta de prova".*

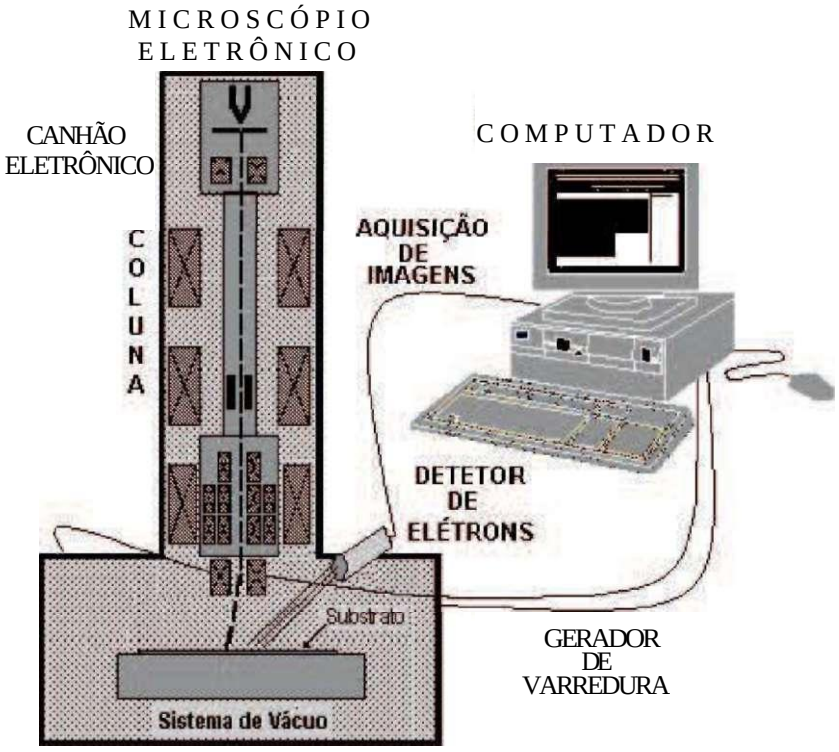



Figura 3.1: Conexão da placa com um microscópio eletrônico de varredura

## 3.1 Funcionalidade da placa

Para aumentar o desempenho, a placa dispõe de dois bancos de memórias, conforme o leitor poderá perceber na Figura 3.2. O processo de armazenamento desses sinais se dá da seguinte forma: um feixe de elétrons varre uma área do substrato, à medida que um detector coleta esses sinais de um pixel na forma analógica, converte para o forma digital e armazena em uma posição de memória em um dos bancos de memória. A tarefa de leitura é feita através do controlador de barramento PCI que leva as informações de imagens ao computador. Um PLD é utilizado para fazer o controle da conversão analógica em digital para que os sinais possam ser armazenados nas memórias e também a conversão digital em analógica, a fim de fazer a varredura do feixe de elétrons no microscópio eletrônico. Os bancos de memória se alternam a cada quadro, podendo ser a memória "A" ou "B". Enquanto se faz a escrita em um banco, no outro banco se faz a leitura.

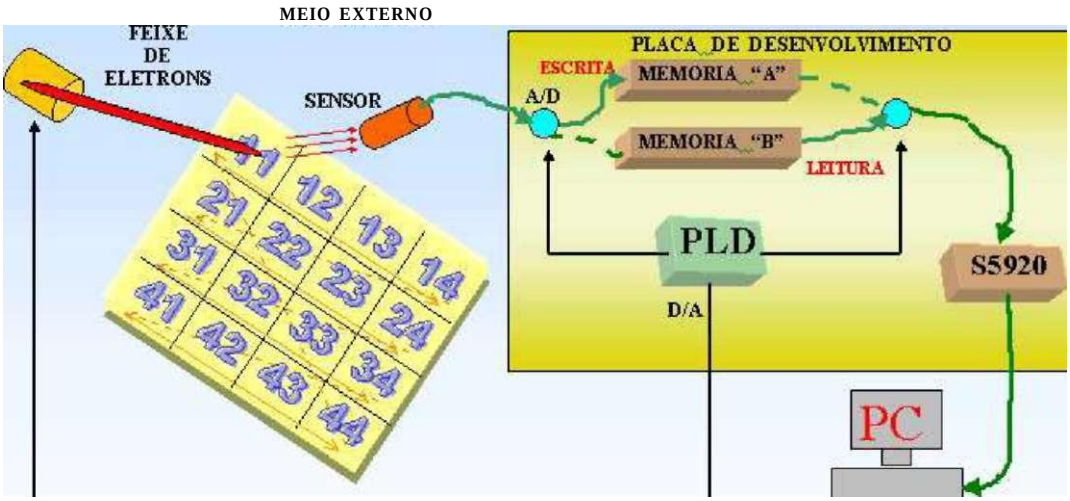



Figura 3.2: O sistema de varredura posiciona o feixe em um ponto da superfície da amostra. O sinal coletado é armazenado em um dos bancos de memória da placa.

## 3.2 Composição da placa

A placa de aquisição de dados é composta basicamente por: controlador de barramento PCI; PLD *(Programmable Logic Device);* conversores analógico-digitale digital-analógico; memória SRAM (Memória Estática de Acesso Aleatório); transceptores; *latches* e *buffers.*

A seguir são descritas as principais características de cada componente. No apêndice "B" o leitor poderá encontrar a descrição de suas pinagens e tabelas verdade.

A MCC S5920 [33]. Na Figura 3.3, o leitor poderá ver os pinos do S5920. No lado esquerdo, temos os pinos pertencentes ao barramento PCI, enquanto no lado direito, temos os pinos pertencentes ao barramento adicional ou externo. Neste projeto, os seguintes pinos foram usados:

- ADCLK: pino de entrada de relógio do S5920.
- PTATN#: pino de saída que indica ao PLD que dados *Pass-Thru* podem ser lidos ou escritos do S5920. Nesta versão inicial do projeto é utilizado o S5920 para apenas fazer a leitura na memória SRAM. Os conteúdos de endereço ou dados presentes no barramento DQ[7..0], só serão validados se o sinal PTATX estiver em nível lógico baixo.

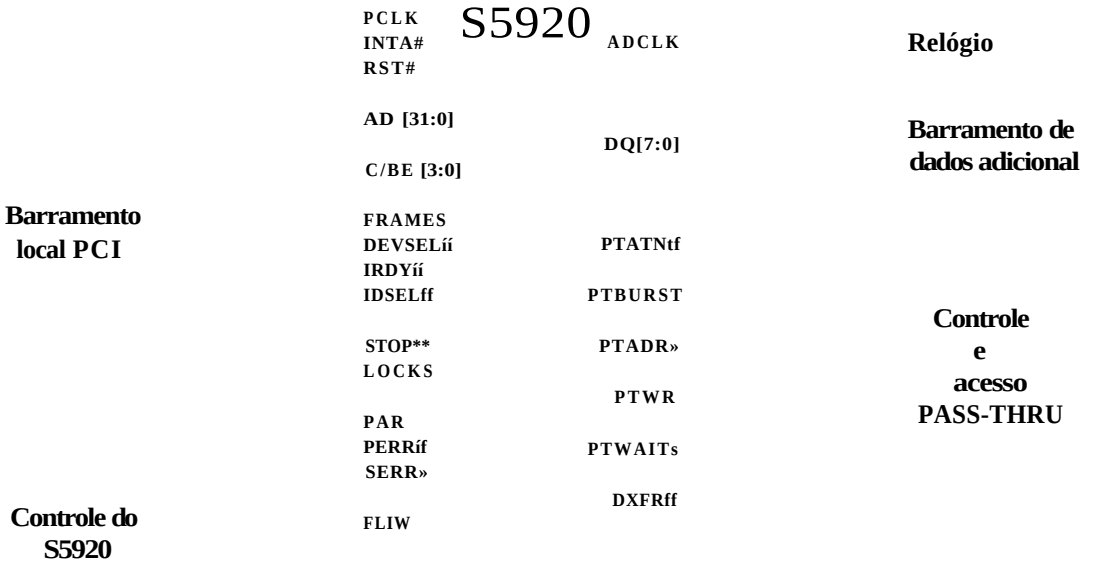



Figura 3.3: Pinos do S5920 utilizados para o funcionamento da placa

• P T B U R S T //: permite fazer uma leitura ou escrita seqüencial entre um dispositivo externo e o S5920, bastando para isso definir a faixa de endereço no qual se propõem trabalhar. A placa de aquisição nesta primeira versão não utiliza esse recurso.

•

DQ está pronto para receber o dado do respectivo endereço enviado anteriormente.

•

barramento que o S5920 coloca o endereço e recebe no ciclo posterior o respectivo dado. Para que o endereço ou dado neste barramento seja válido, o pino PTATX tem que está habilitado (em nível baixo) durante todo o período.

•

dos de espera, durante o ciclo de leitura ou escrita de periféricos lentos. A placa de aquisição nesta primeira versão não utiliza este recurso.

mento DQ possui o endereço do dado requisitado.

P T W R: pino de saída que indica aos demais periféricos se o S5920 está rea-

lizando uma operação de leitura ou escrita. Este sinal só é válido quando PTATN# é ativo.

Na Figura 3.4, o leitor poderá visualizar a sequência de sinais gerada pelo controlador de barramento PCI (S5920), numa operação *Pass-Thru* de leitura em uma das memórias SRAM da placa LDN/UFPE-2001.

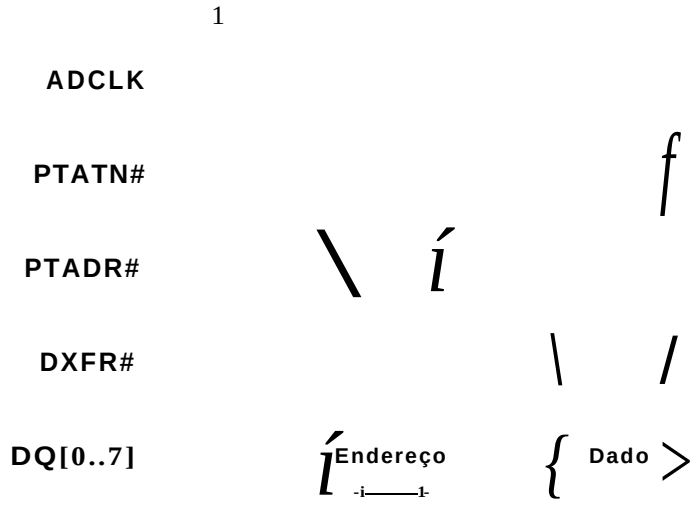



Figura 3.4: Operação *Pass-Thru* de leitura do S5920.

Inicialmente o sinal "PTATN#" é ativo em nível baixo para validação dos demais sinais. Em seguida o controlador S5920 coloca o sinal "PTADR " em nível baixo, indicando para o controlador da placa que o endereço do dado solicitado encontra-se no barramento DQ[0..7]. No ciclo 2 de relógio o S5920 desativa o sinal "PTADR
No ciclo de relógio 4 o S5920 coloca o sinal "DXFR#" em nível baixo, indicando para o controlador da placa LDN/UFPE-2001 que o barramento DQ[0..7] está pronto para receber o dado solicitado. Enquanto não concluir a leitura do quadro, o ciclo se repete, até que o sinal "PTATN" seja desabilitado.

**PLD.** O PLD (Dispositivo Lógico Programável) EPM7128SLC84-7 é o controlador da placa LDN/UFPE-2001, responsável pelo gerenciamento da placa e compõe a lógica combinacional para comunicação com o mundo externo. O PLD utilizado contém 68 portas de E/S e 128 macrocélulas, onde cada uma possui funções de relógio ("clock"), "clear" e "preset" programáveis independentemente. Com capacidade de 2500 portas e arquitetura simples, o PLD empregado na placa LDN/UFPE-2001 é um dispositivo que atende bem as exigências de coleta de imagem, podendo chegar

a uma velocidade de 151, *5MHz.* A compilação e simulação do programa do PLD foram feitas utilizando o programa MAX+PLUS II Student Edition, versão 9.23 da ALTERA Corp**[49].**

**Conversor AD.** O conversor analógico-digital TDA8703/C3 é responsável pela retenção da amostra do sinal e transformação do mesmo de analógico para digital. Esse conversor de *8bits* de alta velocidade de conversão é indicado para aplicações de vídeo, pois, consegue trabalhar a uma taxa de amostragem acima dos *40MHz.* A conversão do sinal analógico em digital é feita a partir do momento em que o PLD coloca no pino CE# do conversor nível lógico baixo.

Na Figura 3.5 é mostrada a conexão do conversor TDA8703/C3 com demais componentes para o correto funcionamento da placa. Essa configuração do relógio faz com que a conversão de analógico para digital seja feita na transição de subida do relógio. É importante perceber que o terra analógico *(AGND)* é diferente do terra digital *(DOND).*

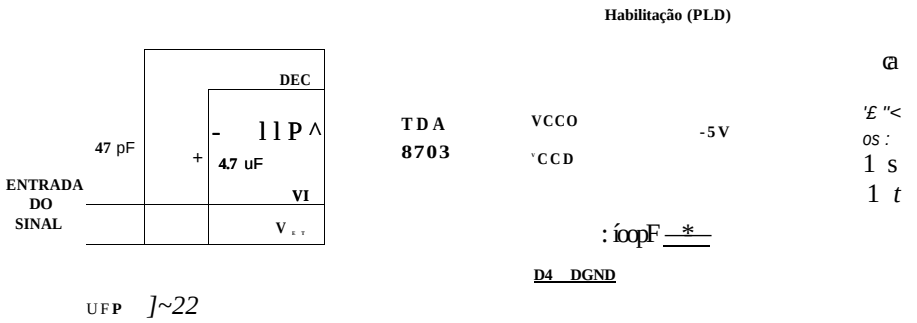



Figura 3.5: Conexão do conversor AD com o sistema

**Conversor DA.** O conversor digital-analógico TLC7524CN faz a transformação dos dados digitais para analógico na geração de varredura. Enquanto o conversor *X* é responsável pela varredura horizontal, o conversor *Y* se encarrega de fazer a varredura vertical. O conversor de *8bits* apresenta um tempo **de** propagação máximo de 80ns, atendendo a aplicação de vídeo. A conversão digital em analógico é feita colocando o sinal digital nas entradas de DBO à DB7 e pondo em nível lógico baixo ambos pinos

CS# e WR#. Conforme a Figura 3.6, para ajuste do ganho de tensão requerido são colocados dois resistores, um no pino ***REF*** e o outro no pino ***R f b*** - Para compensação de fase é colocado na saída do mesmo um capacitor de 10pF.

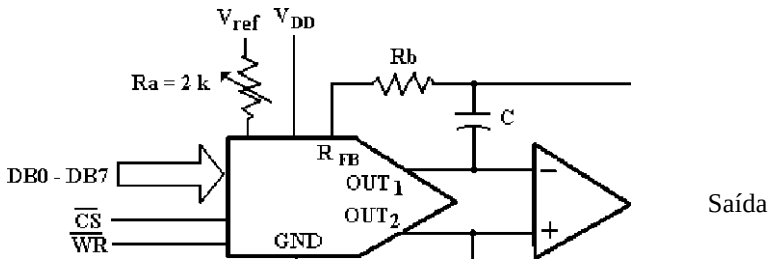



Figura 3.6: Conexão do conversor DA com o sistema.

**Memória SRAM.** A Memória SRAM TC554001FL-70 é responsável pelo armazenamento das informações de imagem. Este dispositivo possui um tempo de acesso de 70ns, que atende as aplicações de aquisição de imagens. A memória possui uma organização de 524288 **palavras** de *8bits*, ou seja, 19 vias de endereçamento para dados de *8bits*. O mesmo permite fazer uma operação de leitura com o pino R/W# em nível lógico alto, ou escrita com o pino R/W# em nível lógico baixo.

**Transceptor.** O transceptor 74ACT245B é responsável pela conexão dos diversos componentes que irão utilizar o barramento de dados comum. O mesmo é constituído de *buffers* com saída de alta impedância. Desta forma é possível conectar o conversor AD ou barramento de dados do S5920 com o barramento de dados da memória sem que haja conflito entre eles. Esse controle é feito pelo pino OE# do transceptor que é habilitado quando colocado em nível baixo pelo PLD.

**Latch.** O *latch* 71 AC17)73 é responsável pelo armazenamento do endereço para obtenção do respectivo dado durante uma operação de leitura ou escrita. Formado por oito registradores num total de 8 *bits*, quando habilidados (LE=1 e **ÜF;** v 0) guardam o endereço do dado em sua saída. Esta saída pode ser colocada em alta impedância independente do nível lógico em que se encontram as entradas OE# e D, colocando OE# em nível alto.

**Buffer.** O *buffer* faz proteção elétrica da placa com o ambiente externo e foi construido utilizando o amplificador operacional AD844. Este operacional apresenta

em aplicações de vídeo uma banda larga acima de *60MHz,* além de ser um dispositivo rápido com tempo de propagação da ordem de *100ns.*

## 3.3 Resolução gráfica da placa

A resolução da placa de aquisição depende do tamanho da sua memória SRAM, pois esta é responsável pelo armazenamento dos dados que serão enviados para a tela. Quanto maior a resolução e número de cores, maior deve ser a tamanho da memória SRAM. A placa LND/UFPE-2001 possui duas memórias independentes, podendo cada uma armazenar 524288 *bytes.*

O dimensionamento da quantidade de memória da placa de aquisição em função da resolução gráfica pode variar de acordo com a definição dos *pixels* e quantidade de tons de cinza. Na Tabela 3.1 é apresentada a quantidade de memória de vídeo necessária para exibição no monitor. Para encontrar o valor mínimo da memória (em *bytes),* basta multiplicar a quantidade de *pixels* da resolução horizontal pela quantidade de *pixels* da resolução vertical. Cada *pixel* utiliza tipicamente um *byte* para representar $2^8$ = 256 níveis de cinza diferentes.

Tabela 3.1: Quantidade de memória de vídeo

| Resolução (Tons de Cinza) | 8 bits = 256 tons |
|---|---|
| 16 x 16 | 256 *bytes* |
| 128 x 128 | 16 *kbytes* |
| 256 x 256 | 64 *kbytes* |
| 640 x 480 | 300 *kbytes* |
| 800 x 600 | 470 *kbytes* |
| 1024 x 768 | 768 *kbytes* |

Pela Tabela 3.1 pode-se notar que a placa LDN/UFPE-2001 para operar na resolução 16 x 16 necessita de uma memória SRAM com no mínimo 256 *bytes.* Por outro lado, como a memória utilizada nesta placa é de apenas 524 *kbytes,* não é possível atingir resolução de 1024 x 768, pois para isso seria necessário 768 *kbytes.*

## 3.4 Cálculo da taxa de transmissão de dados

Neste projeto é desejável que a taxa de quadros por segundo seja igual a 30 quadros. Sendo assim, o tempo de um quadro será:

$$\text{Cuadro} = \wedge = 33,\ 30ms \tag{3.1}$$

Como a taxa de transmissão dos dados depende do tempo de escrita na memória (^escrita) e do tempo de leitura do controlador S5920 na memória ($Ti_{eitura}$), prevalecerá o mais crítico (maior tempo). Assim temos que:

- O tempo de leitura será dado pelo número de pontos tela, vezes o inverso da frequência do relógio (33MHz), vezes o número de ciclos de leitura (3*ciclos)*. Assim, tem-se que:

$$\wedge\text{leitura} = -\wedge\text{pontos} * \sim 7_{/\text{relógio}} * -\wedge\text{ciclos} \tag{3-2}$$

Para demonstração de funcionamento placa, foi escolhida a menor resolução 16 x 16 (4 bits). Para um quadro de 16 x 16, temos:

$$\text{feitura} = 256 \text{ x } \wedge \text{ x } 3 \ .\text{-}.\ \mathbf{T}_{\text{leitura}} = 23{,}27fis \tag{3.3}$$

•

de propagação em cada dispositivo de escrita da placa LDN/UFPE-2001 (conversor AD, transceptor, memória, contador, conversor DA, *buffer* e tempo de estabilização do feixe de elétrons. Sendo assim, tem-se que:

$$\wedge\text{escrita} \quad -\wedge\text{pontos}^{x} (-\wedge\text{ad} + -\wedge\text{transe.} + -\wedge\text{''memória} + \wedge\text{contador} + 2\text{-da} + \text{Tjbuffer} + \wedge\text{estabilização!}) \tag{3.4}$$

16 x 16

$$T_{escrita} = 256 \text{ x } (25 + 9 + 70 + 6,\ 6 + 100 + 100 + 200) \text{ x } 10^{-9} \ .\text{-}.\ \mathbf{T}_{escrita} = \mathbf{130,\ 72}/\text{xs} \tag{3.5}$$

Assim para esta resolução (16 x 16) o tempo de um quadro será dado pelo tempo de escrita (maior tempo). Desta forma tem-se 256 quadros por segundo:

$$Taxa = Jf^{iadro} \;/.\; Taxa = \frac{}{130,72} \;.'.\; \mathbf{Taxa = 256\ quadros/s} \qquad (3.6)$$

-*16xl6

Para um quadro com resolução 128 x 128, temos:

$$\text{Tieitura} = 16384 \text{ x } ^{\wedge}\,^{\wedge}_{1()6} \text{ x } 3 \;.-.\; \mathbf{T_{leitura} = 1,49ms} \qquad (3.7)$$

$$T_{escrita} = 16384 \text{ x } 510,60 \text{ x } 10^{-9} \;.-.\; \mathbf{T_{escrita} = 8,36ms} \qquad (3.8)$$

$$Taxa = \frac{33\ 30}{8,36} \;.'.\; \mathbf{Taxa = 4\ quadros/s} \qquad (3.9)$$

128 x 128

tempos de propagação dos dispositivos de escrita (principalmente do conversor DA e *buffer,* por serem muito altos), uma vez que o tempo de escrita tem se prevalecido em relação ao tempo de leitura, para as diversas resoluções.

Uma outra consideração importante é quanto ao cálculo do tempo de leitura que foi considerado sem interrupção e DMA (Acesso Direto à Memória). Sem previsão deste tempo faz-se necessário o uso dos *flags A e B.*

## 3.5 Melhorando a resolução da placa

Neste projeto foi utilizado dois conversores DA de 8 *bits,* um responsável pela varredura horizontal do feixe e outro responsável pela varredura vertical do feixe. Para varrer 256 pontos na tela foram utilizados apenas os 4 *bits* mais significativos de cada con-

8 *bits*

pode-se obter 65.536 pontos na tela.

10 *bits*

524.288

uma melhor resolução da imagem.

## 3.6 Tempo de varredura da placa

Outra característica importante da placa LDN/UFPE-2001 é o tempo de varredura que está não só diretamente ligado à capacidade de armazenamento da memória, mas também, à velocidade de processamento dos dados pelo conversor AD. Essa velocidade é determinada pelo *set time*, ou seja, a capacidade de atualização dos quadros no tempo. Através da Tabela 3.2 o leitor poderá observar a frequência de atualização dos quadros em função da quantidade de pontos varridos na tela.

Pela Tabela 3.2, o leitor poderá perceber as diversas freqüências de varredura em função da resolução gráfica e números de tons de cinza, para uma variação de quadro a cada 10ms[1]. Frequências acima de *33MHz* (frequência do barramento da placa-mãe) não são atingidas por essa versão da placa LDN/UFPE-2001.

Pela mesma tabela é fácil perceber que é possível chegar até uma resolução de 640 x 480 em dois tons de cinza com uma frequência de varredura a 31MHz. Sendo assim, o fator de limitação no cálculo da resolução gráfica é a frequência de operação da placa LDN/UFPE-2001 e seus dispositivos e não a memória de vídeo, conforme foi mostrado na Tabela **3**.1.

Tabela 3.2: Freqüência de varredura.

| | Pontos na tela | | | Freqüência de varredura *(MHz)* (para varrer um quadro em 10ms) | | |
|---|---|---|---|---|---|---|
| Resolução | 2 Tons | 4 Tons | 8 Tons | 2 Tons | 4 Tons | 8 Tons |
| 128 x 128 | 16384 | 32768 | 49152 | 2 | 3 | 5 |
| 256 x 256 | 65536 | 131072 | 196608 | 7 | 13 | 20 |
| 640 x 480 | 307200 | 614400 | 921600 | 31 | 61 | 92 |
| 800 x 600 | 480000 | 960000 | 1440000 | 48 | 96 | 144 |
| 1024 x 768 | 786432 | 1572864 | 2359296 | 79 | 157 | 236 |

Portanto, o fator de limitação na resolução gráfica é a freqüência de atualização desses quadros no tempo de 10ms e não a memória de vídeo. Para se obter uma resolução maior, é necessário conviver com taxas mais lentas de atualização de quadro.

[1] Foi considerado 10ms como tempo de referência para atualização dos quadros.

## 3.7 Custo da placa

Conforme a Tabela 3.3 podemos verificar o custo aproximado de cada componente da placa.

Tabela 3.3: Preço aproximado de cada componente da placa.

| Item | Componentes | Descrição | Qtd | Preço |
|---|---|---|---|---|
| 1 | Placa de circuito impresso | 10 x 20 $cm^2$ | 1 | R$ 150,00 |
| 2 | PLD | EPM7128SLC84-7 | 1 | R$ 140,00 |
| 3 | Controlador PCI | S5920Q | 1 | R$ 65,00 |
| 4 | Conversor AD | TDA8703 | 1 | R$ 40,00 |
| 5 | Transceptor | 74ACT245B | 4 | R$ 10,00 |
| 6 | Latch | 74ACT573 | 4 | R$ 8,00 |
| 7 | Memória SRAM | TCS5400/FL-70 | 2 | R$ 220,00 |
| 8 | Conversor DA | TLC7524CN | 2 | R$ 16,00 |
| 9 | Buffer | AD844 | 9 | R$ 45,00 |
| 10 | Conector | BNC TCB INS SKT | 3 | R$ 63,00 |
| Total | | | | R$ 757,00 |

O preço apresentado foi para a contração de uma placa. Apesar de não serem incluídos os gastos com mão de obra e soquetes, podemos notar um baixo custo para confecção desta placa em relação a uma placa vendida comercialmente.

## 3.8 Diagramas de bloco

O funcionamento da placa LDN/UFPE-2001 ocorre da seguinte forma: o sinal de entrada analógico coletado pelo sensor é enviado para entrada do conversor AD; o controlador da placa LDN/UFPE-2001 habilita o conversor AD para que o mesmo faça a conversão do sinal, pondo em sua saída o respectivo sinal digital; o controlador seleciona a memória na qual irá escrever o resultado da conversão, através do transceptor A/B; o endereço da memória onde será armazenado o dado é definido pelo *latch* A/B de sua respectiva memória; o *latch* A/B recebe o endereço através do contador interno do controlador; no ciclo posterior o controlador habilita a memória para que possa fazer a escrita; o contador é incrementado, fazendo com que o conversor DA posicione a varredura para próxima posição a ser escrita. Um *buffer* é colocado na saída desse conversor DA, a fim de fazer a proteção do circuito com relação ao

ambiente externo.

Simultaneamente ao processo de escrita, ocorre o processo de leitura através do S5920. Isto é possível graças ao uso de duas memorias. Os dados de um quadro já carregados em uma das memórias (A ou B) é transferido para o barramento PCI, através do S5920 da seguinte forma: o S5920 coloca no barramento DQ o endereço do dado a ser lido em uma das memórias; o endereço da memória é armazenado no *latch* C/D da respectiva memória; no ciclo de relógio seguinte, o dado da memória selecionada é colocado no barramento DQ, para que o S5920, possa receber estas informações; a informação do quadro é enviada ao S5920, a medida que o mesmo envia os endereços (de 0 a 255). O leitor poderá perceber na Figura 3.7 que os transceptores permitem compartilhar o barramento único de 8 *bits* de escrita (à esquerda) e de leitura (à direita) sem que haja conflito entre as operações de leitura e escrita nas memórias. Os *latches* tem a função de armazenar o endereço do dado a ser escrito ou lido na memória. No Apêndice "C" o leitor encontrará a descrição dos sinais de entrada e saída presentes no controlador da placa LDN/UFPE-2001.

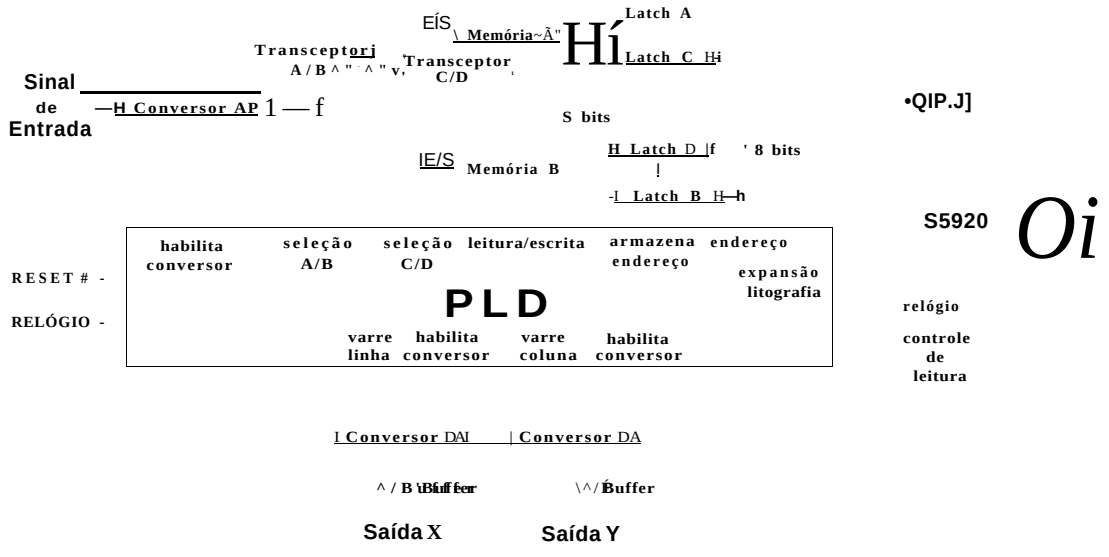



Figura 3.7: Diagrama em blocos dos componentes da placa.

O leitor poderá visualizar na Figura 3.8, a arquitetura interna do controlador da placa LDN/UFPE-2001, composta de 5 módulos:

- Controlador de escrita;
- Controlador de leitura;

- *flag A;*
- *flag B;*
- Contador de escrita.

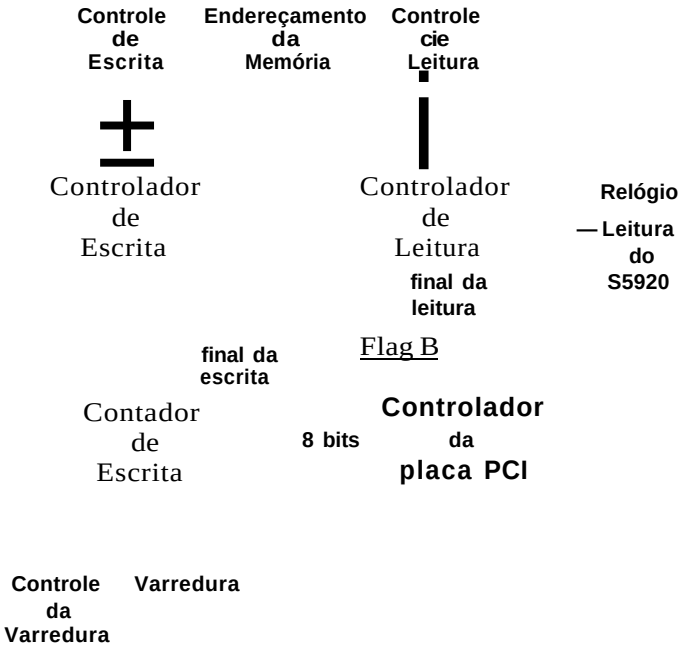



Figura 3.8: Módulos do controlador da placa LDN/UFPE-2001.

O controlador de escrita gera os sinais para o processo de escrita na memória dos dados coletados pelo sensor, incluindo o processo de conversão analógico em digital.

O controlador de leitura gera os sinais para o processo de leitura do S5920, dos dados do quadro armazenados na memória.

O *flag* A verifica a finalização da escrita do quadro na memória, informando aos controladores de escrita/leitura que a troca de memória depende da finalização do processo de leitura do S5920.

O *flag* **B** verifica a finalização do processo de leitura do S5920 na memória, informando aos controladores de escrita e leitura que a memória de leitura está liberada para fazer uma nova escrita.

Contador de escrita é responsável pela geração dos endereços de memória, onde serão escritos os dados e também gerar a varredura do feixe de elétrons do microscópio de varredura. Seus 4 *bits* menos significativos definem a coluna de varredura, enquanto

que os 4 *bits* mais significativos definem a linha de varredura. Desta forma são varridos 256 pontos de uma matriz de 16 colunas por 16 linhas.

A placa oferece outras facilidades, tais como:

- Possibilidade de escrita do S5920 na memória da placa, através do direcionamento do transceptor (C ou D);
- Expansão do barramento DQ de 8 *bits* para 16 ou 32 *bits;*
- 

(definidos pelo usuário) do substrato em estudo.

No Apêndice "C" o leitor poderá encontrar ilustrações da pinagem e arquitetura externa e interna do PLD gerado a partir do MAX+plus II Student Edition da Altera Corp [49].

## 3.9 Diagrama de estados do controlador

Pela Figura 3.9, o leitor poderá visualizar o diagrama de estados simplificado do controlador da placa LDN/UFPE-2001 sendo executado concorrentemente. Inicialmente todas as entradas e saídas do controlador são colocadas em um estado lógico pré-defmido. Por medida de segurança toda vez que ocorrer um estado não definido um programa adicional de tempo real se encarrega de colocar a placa em seu estado inicial evitando possíveis travamentos com o programa. Na mesma figura pode-se observar que enquanto o ramo esquerdo se preocupa em carregar na memória SRAM "A" os dados lidos na entrada da placa, o ramo direito se preocupa em passar os dados da memória SRAM "B" para o S5920. Quando as duas tarefas são concluídas, o processo se repete, mas com as memórias trocadas. Assim podemos ler as informações de um quadro na entrada da placa e ao mesmo tempo enviar para o S5920 as informações do quadro anterior.

Um estudo mais apurado desse fluxograma pode ser feito analisando o diagrama de estado do ciclo de escrita e o diagrama de estado do ciclo de leitura. Os sinais terminados por um  implica que os sinais são ativos em nível lógico baixo.

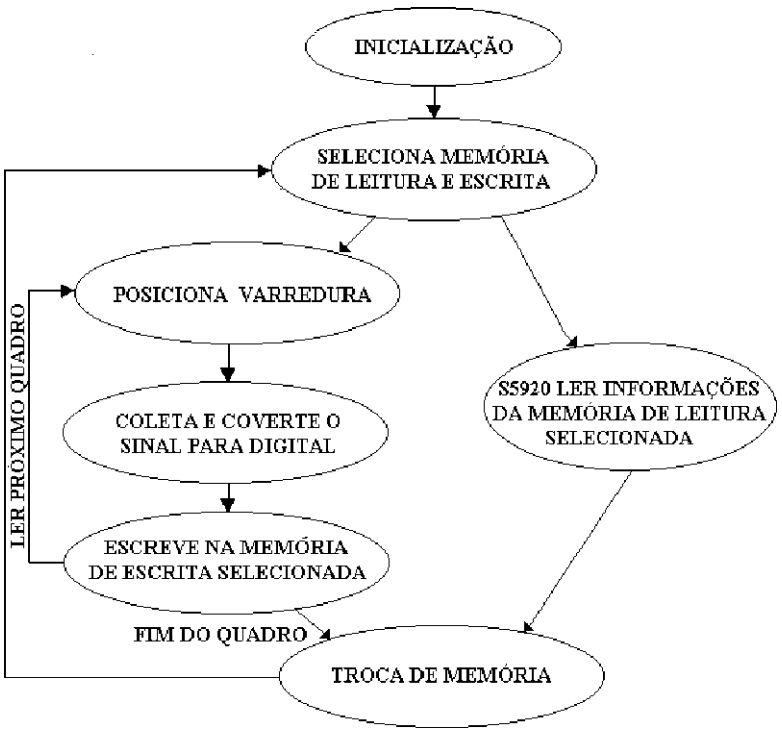



Figura 3.9: Diagrama de estados simplificado do controlador.

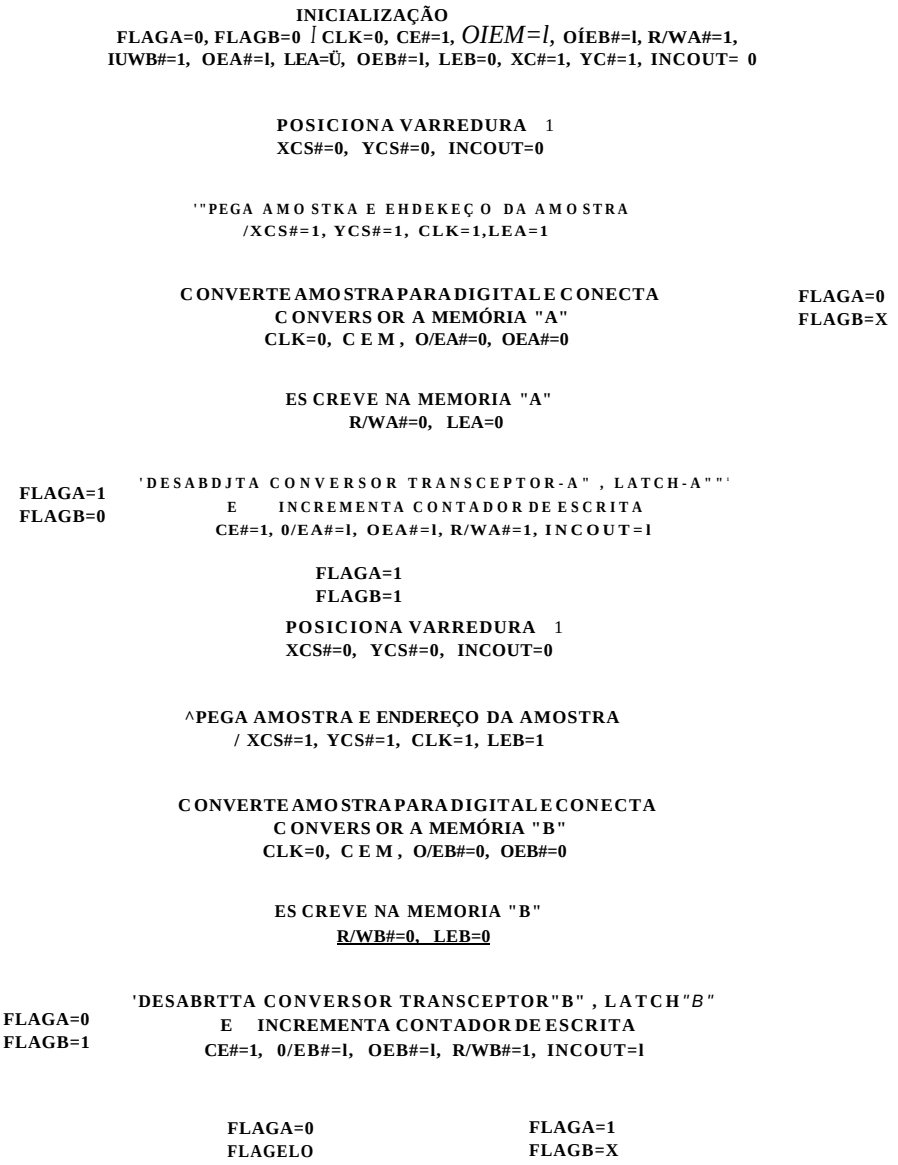
O estado inicial *(ei)* é onde estão definidos todos os níveis lógicos iniciais dos sinais presentes na placa. O sinal de RF;SF;T quando ativo (RF;SF;T 0) coloca o controlador no seu estado definido inicial, independente do estado em que se encontra.

O controlador PLD executa as tarefas de leitura e escrita em memória concorrentemente, utilizando duas bandeiras *flag A* e *flagB,* que são responsáveis pela inversão das memórias de escrita com as de leitura ao final de cada quadro. O controlador de escrita em conjunto com o contador de escrita faz o controle do *flag A,* mudando seu estado toda vez que finaliza um quadro, enquanto que o controlador de leitura é responsável pelo controle do *flagB,* mudando de seu valor ao finalizar o processo de leitura na memória.

Dependendo dos valores contidos nos *flag's* o ciclo de escrita na memória pode ocorrer na memória "A" *(flagA=0* e *flag"B=0) ou* na memória "B" *(flagA=l* e *flag"B=l).* Por segurança a troca de memória só acontece quando ambas bandeiras estiverem no mesmo nível lógico, indicando que ambas tarefas foram concluídas com sucesso. Caso um dos processos termine antes do outro, este terá de esperar que o outro conclua, para haver a troca de memória. Este método garante a integridade do

processo de escrita e leitura em memória.

O diagrama da Figura 3.10, mostra os 10 estados do ciclo de escrita, onde o estado inicial "e/' contém as condições iniciais do controlador de escrita para esta operação.




**OBS • ^° ^ᵐ·^ᵠª contagem no estado** 640 **FLAGA fica igual a 1. No final da contagem no estado 6 g o FLAGA fica igual a 0.**

Figura 3.10: Diagrama de estado do ciclo de escrita

O estado "$e_0|e_5$" posiciona a varredura para leitura do primeiro ponto. No estado "$ei|e_6$" são coletadas as informações de cada ponto da amostra. No estado "$e_2|e_7$" é feita a conversão de analógica para digital de cada ponto da amostra colhida pelo

sensor, e selecionada a a memória "A" ou "B" com o endereço apontado pelo contador de escrita. No estado "$e_3$|es" o conteúdo é escrito na memória selecionada. No estado "$e_4$|eg" o conversor AD, transceptor "A" ou "B", *latch* "A" ou "B" são desabilitados e incrementado de um, o contador de escrita (INCOUT=l).

O processo de escrita na memória se repete com o incremento do contador, até que haja a escrita completa do quadro, mudando assim, o valor do *flagA*.

O diagrama da Figura 3.11, mostra onze estados no qual o estado inicial "$e_0$" fixa as condições iniciais dos registros. Uma vez definida a memória de leitura ("A" ou "B") no estado "ei|e2", é habilitada a comunicação do PLD com o S5920 (PTATN#=0)

e3|e4

barramento "DQ" do S5920 (PTADR I), o controlador retorna ao estado inicial, caso contrário (PTAI)R 0), o S5920 armazena o endereço no *latch* "C" (memória de leitura "A"), ou no *latch* "D" (memória de leitura "B"). No estado "$e_5$|$e_6$" com o endereço retido pelo *latch* "C" ou "D" e com o sinal DXFR em nível baixo o barramento "DQ" é liberado para receber o conteúdo do respectivo endereço. No estado "$e_7$|es", a memória "A" ou "B" coloca o conteúdo do endereço contido no *latch,* no barramento "DQ". Caso haja mais conteúdos de memória para serem lidos (PTADR 0), o con-

e3|e4

inicial. Ao chegar no estado "eg|$e_{10}$" é definida o fim da leitura na memória, sendo desabilitada a comunicação do PLD com o S5920 (PTATN#=1) e setado o *flagB* (END_READ=1), indicando o fim do processo de leitura do quadro.

Para garantir o perfeito funcionamento do PLD, diante possíveis falhas que venha ocorrer durante o ciclo de leitura, no programa em VHDL (ver apêndice "A": controlador de leitura das memórias) foi determinado que; caso o programa chegue a um estado não definido, o mesmo irá para o estado de inicialização, evitando possíveis travamentos. Esta idéia não se encontra representada no diagrama de estado do ciclo de leitura, visando facilitar o seu entendimento.

## 3.10 Descrição VHDL

A programação do PLD foi feita em VHDL, utilizando o aplicativo MAX+plus II Student Edition da Altera, versão 9.23. Este aplicativo permitiu o desenvolvimento

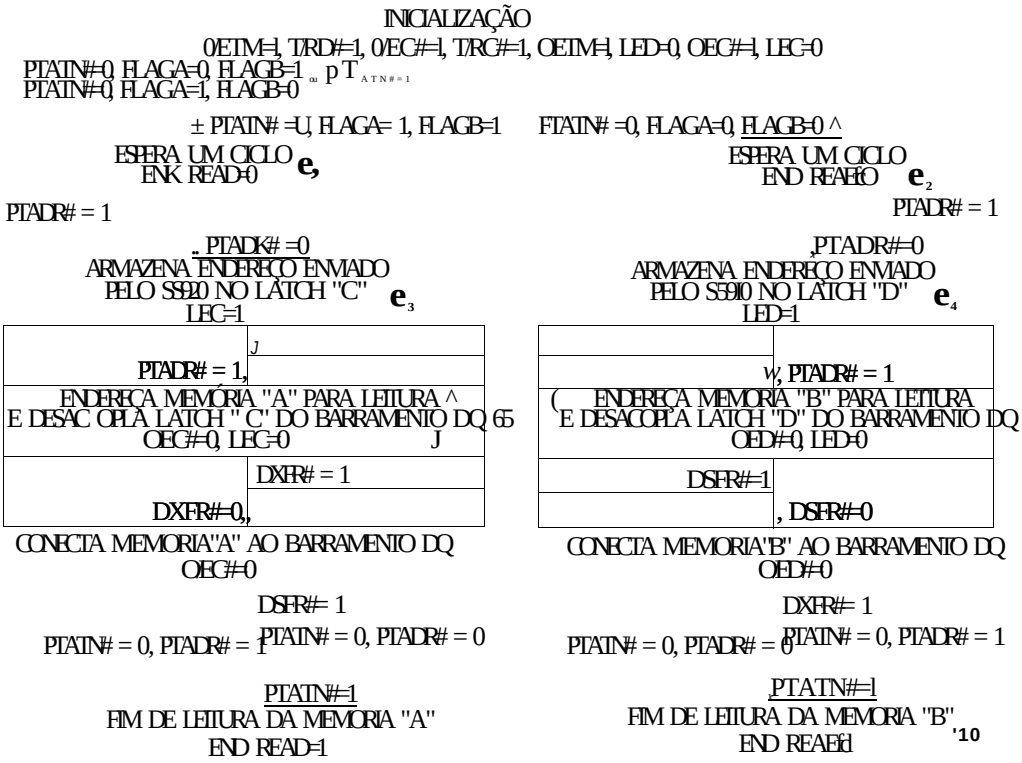



Figura 3.11: Diagrama de estado do ciclo de leitura.

da descrição de *hardware* do PLD, auxiliando na correção dos erros lógicos de programação, antes de gravar no próprio *chip*.

No Apêndice "B" o leitor poderá encontrar resumidamente os principais passos tomados para criação, compilação e simulação dos programas desenvolvidos. A descrição do programa de cada módulo contido no PLD, poderá ser encontrada no Apêndice "A". Cada módulo é identificado por um nome. O módulo principal, denominado " Fechamento do PLD da placa LDN/UFPE-2001", é o programa responsável pela ligação dos outros cinco módulos. Cada módulo define inicialmente seus sinais de entrada e saída e em seguida o nível de cada sinal para cada estado.

## 3.11 Simulação

Através do aplicativo MAX+PLUS II Student Edition [49], uma vez desenvolvido o programa do PLD, foi possível visualizar a simulação do comportamento de cada

módulo contido no PLD. A seguir é ilustrada a simulação de cada módulo.

Na Figura 3.12 é apresentada a simulação do controlador de escrita. No estado "e/', com o *RESET* ativo *(RESET_ ESCRITA* baixo), o controlador de escrita é colocado em seu estado inicial. Na transição de subida do relógio com os *flag's* em zero, o controlador vai para o estado "$e_0$". Os sinais de habilitação dos conversores DA são ativos *(XCS_INV*e *YCS_INV*baixo), colocando suas saídas em zero (posicionamento inicial do feixe de elétrons). No estado "ei", o conversor AD é habilitado *(CLK* alto), para armazenar o nível de tensão lido do sensor. Neste mesmo ciclo é também armazenado no *latch* "A" o endereço da memória "A" *(LEA* alto), onde será escrito o dado. No estado "e2", o conversor AD converte o sinal analógico em digital *(CE_INV* baixo), colocando os dados na entrada da memória "A" *(flags* em zero), através da habilitação do transceptor "A" (*0_ EA_ INV* baixo). Neste estado também é armazenado no *latch* "A" o endereço do dado a ser guardado na memória "A" *(OEA_INV* baixo). No estado "ea", é feita a escrita do dado na memória "A" *(R_ WA_INV* baixo). No estado "e4" são desabilitados o conversor AD *(CE_INV* alto), transceptor "A" *(0_EA_INV*alto), *latch "A" (OEA_INV*alto), memória "A" (R_WA_INV alto) e incrementado de um o contador de escrita *(INCOUNT* alto).

$e_0$

na memória "A", até terminar a escrita do quadro. Quando os processos de escrita (memória "A") e leitura (memória "B") terminam, os *flags* vão para nível alto, recomeçando o ciclo de escrita, mas desta vez na memória "B" (estados $e_5$, $e_6$, $e_7$, $e_8$ e eg).

Na Figura 3.13, o leitor poderá ver a simulação do contador de escrita. O sinal *"CLOCK_ COUNTER"* gerado pelo controlador de escrita é responsável pela contagem de 0 a 255, no momento em que o sinal "*CLEAR_COUNTER"* é liberado *(CLEAR_ COUNTER* alto). No final da contagem o sinal *"END_ COUNTER"* é colocado em alto, indicando o final da contagem (final da escrita na memória).

Na Figura 3.14, o leitor poderá visualizar a simulação dos *flip-flops T.* "A" e "B". Com o sinal de "*RESET FLIPFLOP'* ativo em nível baixo, fica em nível baixo, independente das demais entradas. Quando este sinal é desabilitado, o sinal de saída *"$q_a$\$q_b$"* só vai para o nível alto, se na transição de subida do relógio a entrada *"$t_a$\tb"* estiver em nível alto.

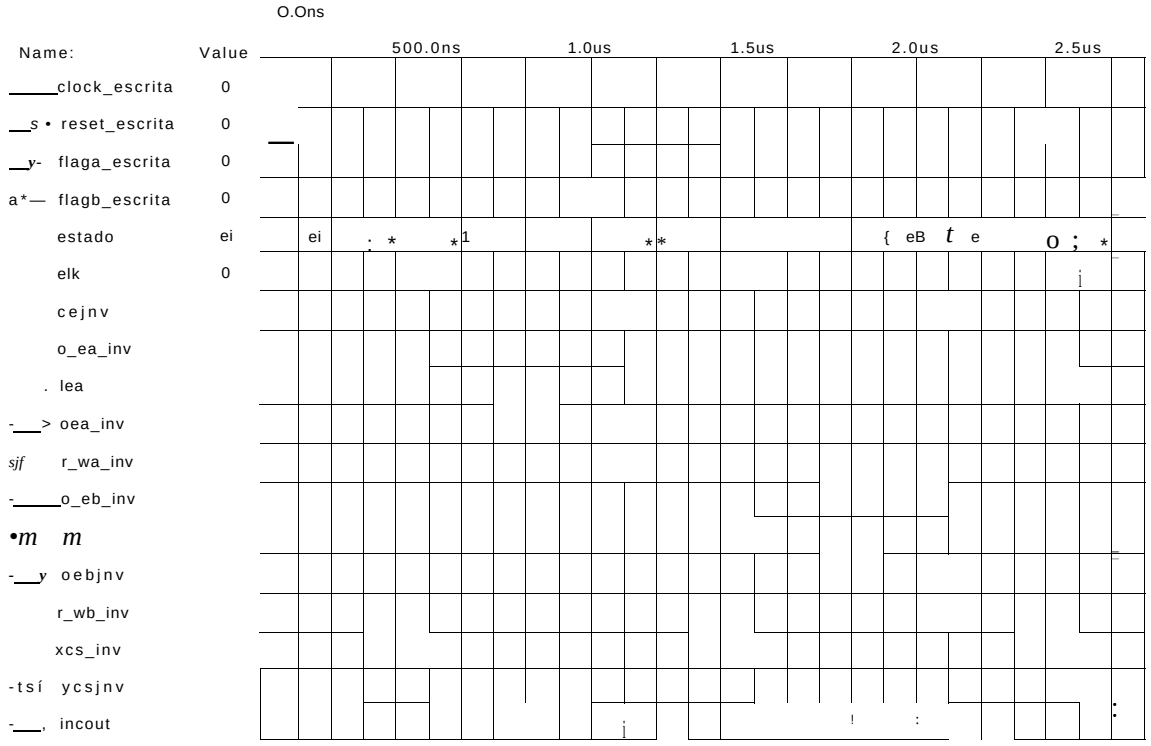



Figura 3.12: Simulação do controlador de escrita.

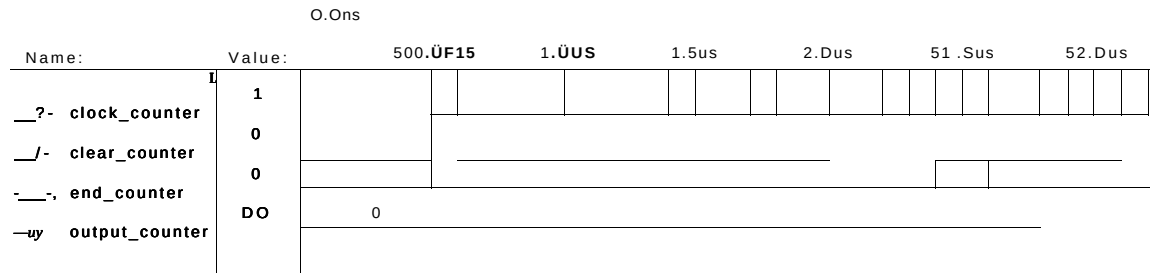



Figura 3.13: Simulação do contador.

Na Figura 3.15, é apresentada a simulação do contador de leitura. O processo de leitura inicia-se com a desabilitação do sinal "*RESET_LEITURA*" em nível alto. Estando o "*FLAGA_LEITURA*" e "FLAGB_LEITURA" em nível alto, a leitura se dá na memória "A". No estado "ei", o sinal "*PTATN_INV*" é ativado para validar o barramento "DQ". No estado "$e_3$", o sinal "*PTADR_INV*" encontra-se em nível baixo, juntamente o sinal "*PTATN_INV*" o barramento "DQ" passa a conter o endereço do dado solicitado. O endereço é armazenado no *latch* "C", através do sinal "*LEC* em nível alto. O endereçamento da memória "A" é feito pelo próprio *latch* "C", através do sinal "*OEC_INV*" em nível baixo. Com os sinais "*0_EC_INV*" em nível baixo e "*T_RD_INV* em nível alto, o transceptor "C" coloca no barramento

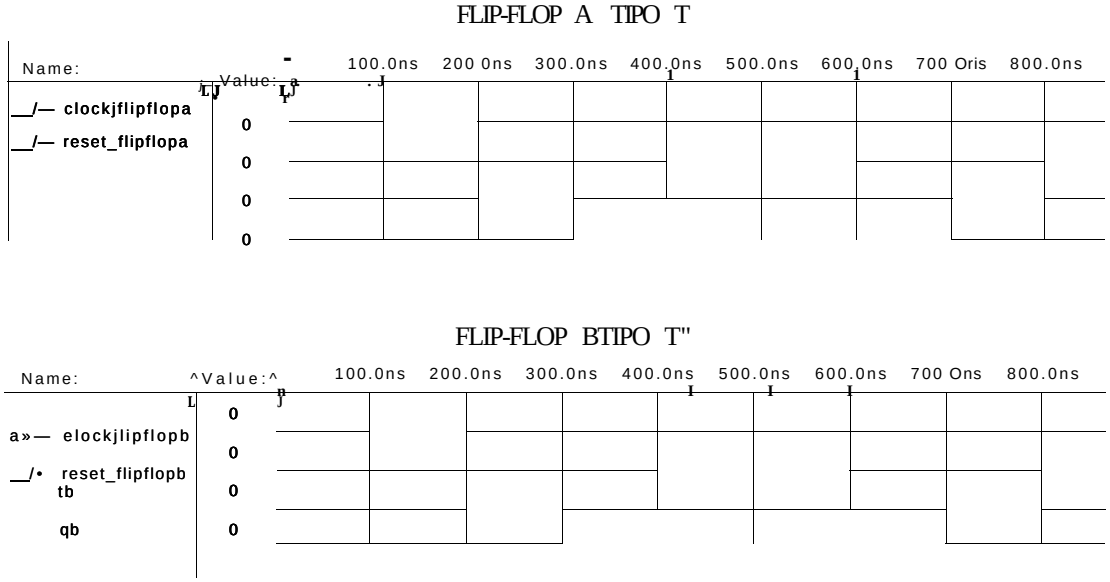



Figura 3.14: Simulação dos *flip-flops* tipo T: "A" e "B"

"DQ", o dado da memória "A" solicitado pelo S5920. O S5920 reconhece o dado, colocando o sinal "*DXFR_INV"* em nível baixo. O ciclo de leitura se repete enquanto o sinal "*PTATN_INV"* estiver em nível lógico baixo. Com a desabilitação do sinal "*PTATN_INV"* em nível alto no final da leitura do quadro, o sinal "*END_READ'* é colocado em nível alto, indicando o final da leitura do quadro, para que se faça a troca de memória. A leitura na memória "B" é feita, utilizando o *latch* "D" (sinais "*LED* e *OED_INV)* e transceptor "D" (sinais *0_ED_INVe T_RC_INV).*

Na Figura 3.16, o leitor poderá observar a simulação do PLD. Nesta simulação estão representados todos os pinos de entrada e saída do PLD, usados na placa LDN/UFPE-2001. O sinal de relógio *"CLK"* é responsável pelo funcionamento do PLD, enquanto que o sinal *"ADCLK"* é responsável pelo funcionamento do S5920. O sinal *RESET* na simulação sempre desabilitado é responsável pela inicialização do PLD.

Na simulação pode-se notar que a sequência dos sinais ocorrem da seguinte forma: O sinal *"CLK"* responsável pela retenção do sinal do sensor é ativo. O sinal "*LEA"* é ativo para reter o valor do endereço, onde será armazenado a informação do sinal adquirido. O sinal "*CE_INV"* converte o sinal de analógico para digital, enquanto o sinal "*0_EA_ INV"* conecta a saída do conversor AD ao barramento de dados da memória "A". O sinal" *OEA_ IN V"* endereça a memória "A" para que a informação seja

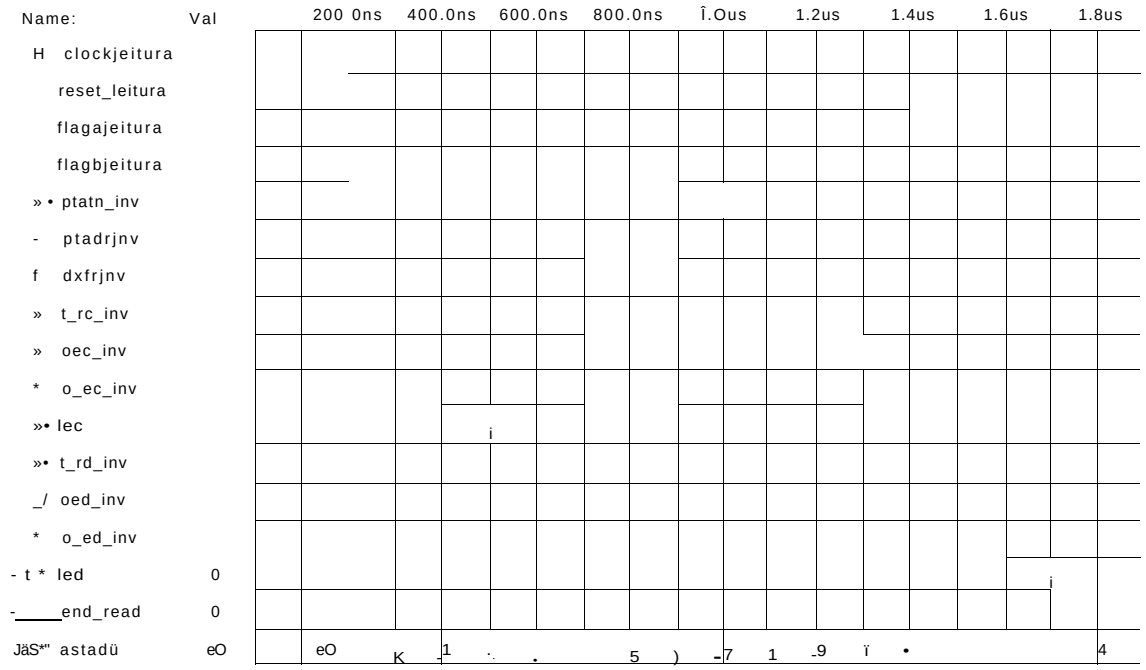



Figura 3.15: Simulação do controlador de leitura.

escrita, assim que o sinal Mv1_ //VI" for ativo. Os sinais *"0_EB_INV\ "LEB\ "OEB_INV"* e *WB_INV* para escrita na memória "B" possuem respectivamente as mesmas funções dos sinais citados para escrita na memória "A".

O processo de leitura na memória "B" ocorre simultaneamente com o processo de escrita na memória "A". Sendo assim, o sinal "*PTATN_INV* é ativo, validando o barramento "DQ" do S5920. Posteriormente, o sinal "*PTADR_INV"* também é colocado em nível baixo, informando ao PLD que o S5920 colocou um endereço válido no barramento "DQ". O PLD armazena no seu latch "D", através do sinal *"LED* O sinal "*OED_INV* é habilitado em nível baixo para endereçar a memória "B". O dado requisitado é colocado na saída da memória "B". Com a desabilitação do sinal *"PTADR_INV* o sinal *"DXFR_INV"* é habilitado em nível baixo, validando o barramento "DQ" para o S5920 receber o dado. Desta forma, o PLD ativa o sinal *"0_ED_INV* com o sinal "*T_RD_INV* em nível alto, para que o barramento "DQ" do S5920 receba o dado. A busca pelo próximo dado continua, até a leitura completa do quadro. Quando isto ocorre, o sinal *"PTATN_INV"* retorna ao seu estado inicial (nível alto) para que seja feita a troca da memória de leitura.

O processo de leitura na memória "A" é análogo ao descrito acima, variando os respectivos sinais *"LEC, OEC_INV, 0_EC_INV, T_RC_INV*

Ao término de cada escrita/leitura os conversores DA são ativados com os sinais *"XCS_INV* e *YCS_INV* colocando um novo valor de tensão em suas saídas para que seja posicionado o feixe de elétrons do microscópio de varredura. Os sinal *"ADR"* formado por oito bits é responsável pelo endereçamento das memórias na escrita. O sinal *" VAR"* responsável pela geração das 256 posições de varredura, é o mesmo sinal *"ADR"*.

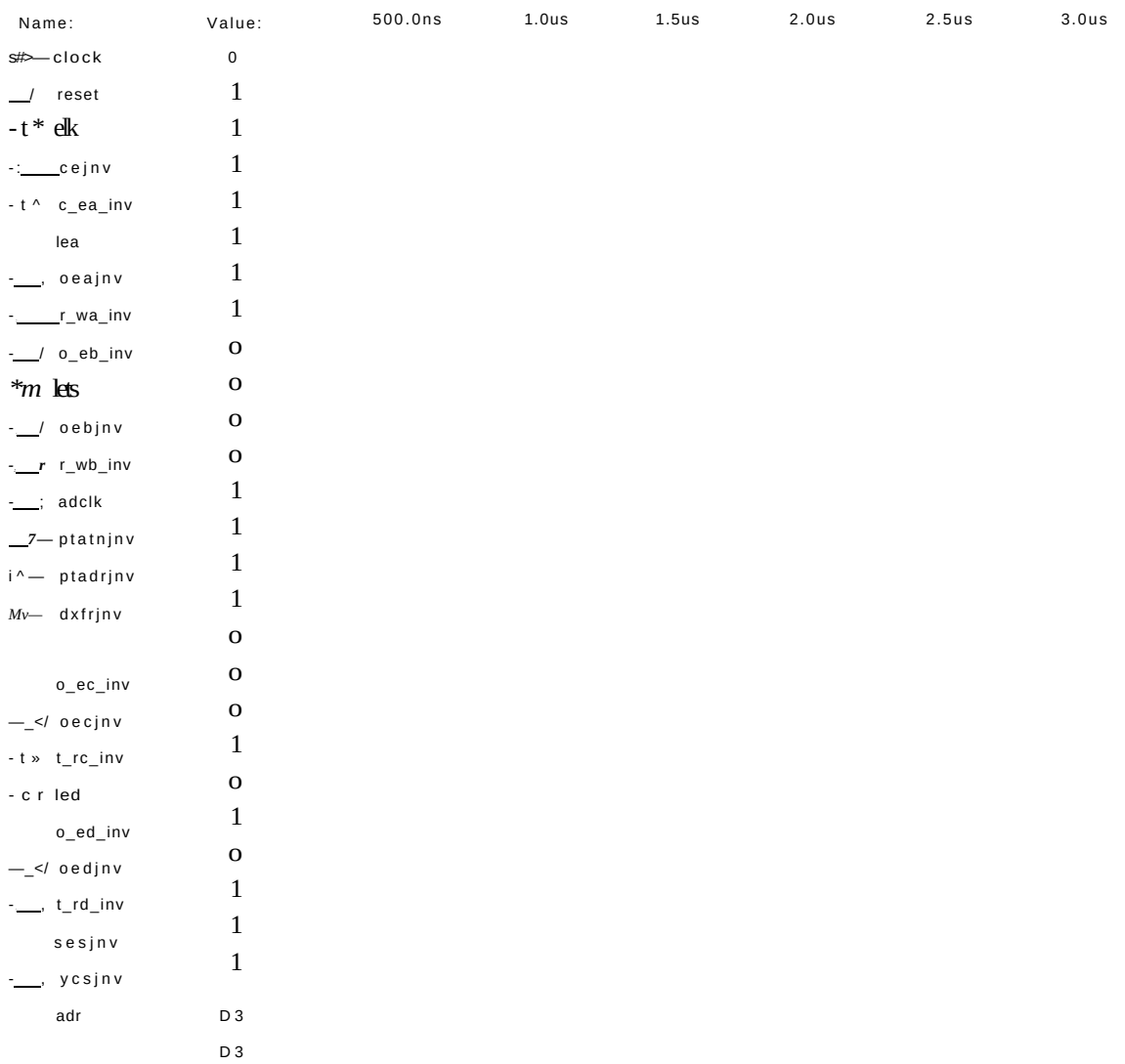



Figura 3.16: Simulação do PLD

## 3.12 Resultados experimentais

Nesta secção são apresentados o leiaute, a montagem, a instalação e testes.

### 3.12.1 Montagem da placa LDN/UFPE-2001

A placa de circuito impresso de dupla face com furos metalizados, foi projetada utilizando o aplicativo TraxMaker3 PRO da Protel International Ltd. Para evitar que ruídos externos interferisse no correto funcionamento da placa LDN/UFPE-2001, linhas de grade foram colocadas e aterradas com capacitores de acoplamento nos espaços vazios da placa. Através da Figura 3.17, o leitor poderá visualizar o leiaute da placa LDN/UFPE-2001.

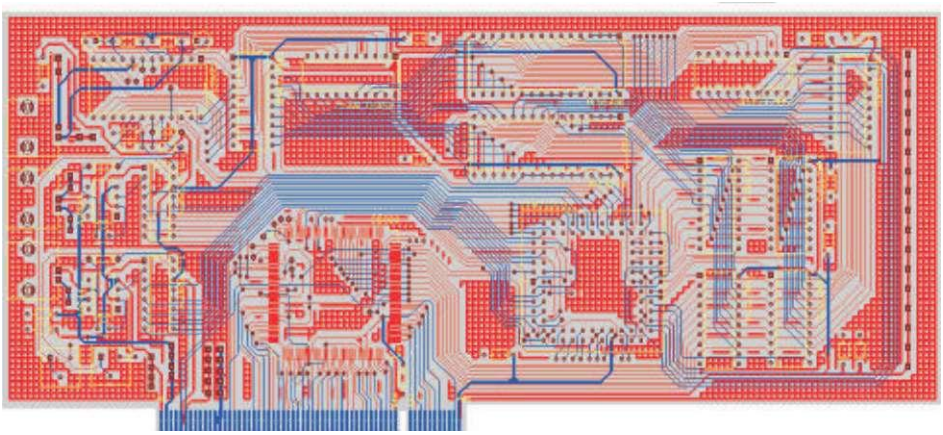


Figura 3.17: Leiaute da placa LDN/UFPE-2001.

A soldagem direta dos componentes da placa sensíveis ao calor foi evitado, utilizando soquetes adequados para cada tipo de encapsulamento. Através da Figura 3.18, o leitor poderá visualizar a fotografia da placa LDN/UFPE-2001 real, tirada com uma câmera digital.

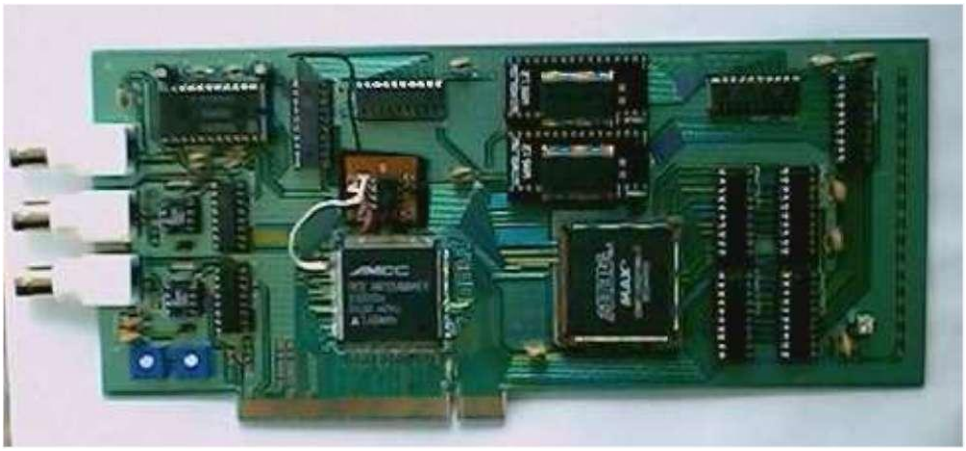


Figura 3.18: Fotografia da placa LDN/UFPE-2001.

Toda a soldagem da placa LDN/UFPE-2001 foi feita manualmente pelo autor.

Para fixação do dispositivo S5920, foi usado um ferro de solda de 20 *W*, tendo que afinar a sua ponteira. Um microscópio óptico foi utilizado para examinar a correta fixação dos 160 pinos deste dispositivo.

A necessidade do uso de uma EEPROM para inicialização do dispositivo S5920 em modo ativo teve que ser implementado pelo autor. Sendo assim, um pequeno circuito adicional, contendo uma memória serial não volátil (24C08) e dois resistores de 4, 70, foi fixado a placa LDN/UFPE-2001.

### 3.12.2 Programação e testes do PLD

A programação do PLD foi feita utilizando o aplicativo MAX+plus II da Altera Corp [49]. Com a descrição em VHDL de cada bloco contido no PLD, o programa foi depurado. Corrigidos alguns erros de parâmetros, o programa foi compilado. Após essa etapa foi feita a simulação de cada bloco, conforme mostrado anteriormente. Para programação do *chip* EPM7128SLC84-7, diagramas de símbolos, constando as características de cada bloco com suas respectivas ligações, foram criados e compilados. Finalmente o programa foi gravado no *chip* e testado na placa de desenvolvimento da Altera.

Um gerador de onda quadrada de 1 Hz, com amplitude de 5 V, foi colocado na entrada de *clock* do PLD. Uma chave com níveis de tensão 0 ou 5 V, foi colocado na entrada de *RESET* do PLD. Três chaves também com níveis de tensão 0 ou 5 V, foram colocadas nas entradas *"PTATNf,* *"PTADR#"* e *"DXFR#"* do PLD. Nas saídas ylDi?[0..7] foram colocados oito LEDs, afim de acompanhar a sua contagem.

### 3.12.3 Teste e configuração do S5920

**Teste do S5920**

Com a placa LDN/UFPE-2001 inserida no computador, através do Windows 95®, ao clicar: *Iniciar -> Configurações -> Painel de controle -> Sistema -> Gerenciador de dispositivo -> Outros dispositivos -> PCI system peripheral -> Recursos,* pode-se capturar a tela apresentada na Figura 3.19.

10

rupção de memória *BARO* com intervalo de *F*0210000 a *F*021007F e *BARI* com

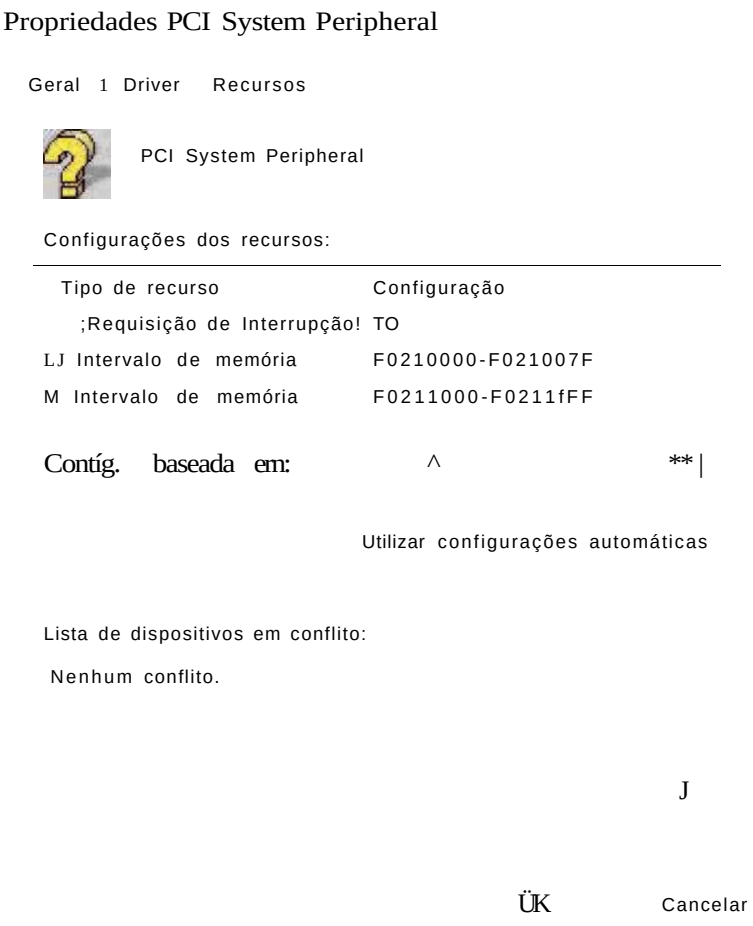



Figura 3.19: Detecção da placa LDN/UFPE-2001.

intervalo de F0211000 a *F0211FFF,* configurado pelo BIOS do computador.

Para testar o dispositivo S5920 na placa LDN/UFPE-2001, foi utilizado o aplicativo WinDriver, versão de demonstração 5.03 (http://www.jungo.com). Esse programa permitiu fazer de maneira simplificada a leitura e escrita dos conteúdos dos registradores de configuração do S5920. Para verificação desses registradores foram feitos os seguintes procedimentos: conexão da placa LDN/UFPE-2001 na baia *(slot)* do computador; escolha do cartão PCI, através da lista dos componentes PCI instalados no computador, conforme o leitor poderá observar na Figura 3.20. Através deste menu foi possível obter informações sobre a descrição da placa LDN/UFPE-2001 solicitada, como por exemplo; código do fabricante e do dispositivo, barramento, baia e função).

O WinDriver permitiu a fácil visualização dos conteúdos dos registradores de configuração do S5920, conforme visto na Figura 3.21.

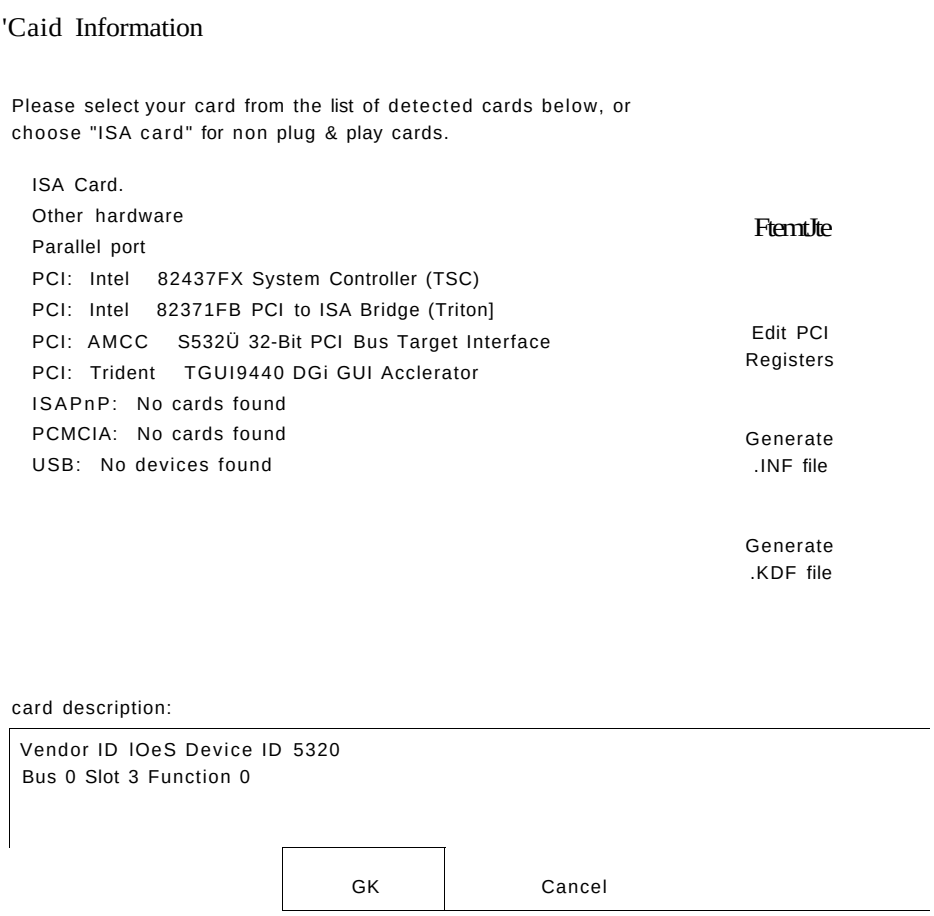



Figura 3.20: Seleção da placa LDN/UFPE-2001.

Através deste aplicativo, ao acessar a tela de registradores de configuração PCI, pode-se escrever o dado *FFFFFFFFh* no registrador *BARI,* obtendo *FFFFFOOOh,* visualizado na Figura 3.22, conforme esperado. Segundo o manual AMCC do S5920, esse resultado indica que o tamanho da memória reservada para escrita é de 4 *kbytes,* ou 1 *kDWORD.*

Foi possível também alterar o conteúdo do registrador do endereço base-1, conforme Figura 3.23.

Além disso, o aplicativo permitiu fazer o diagnóstico e a depuração dos conteúdos gerados, podendo ser utilizado para compilar e executar o código do *driver* da placa.

Programação da EEPROM

A programação da EEPROM serial 24C08 ou nvRAM (RAM não volátil), foi feita com o programador de EPROMs SWC-271GEP, adquirido na página da Macsym Tecnologia Eletrônica Ltda. O mapa com os conteúdos de inicialização de memória

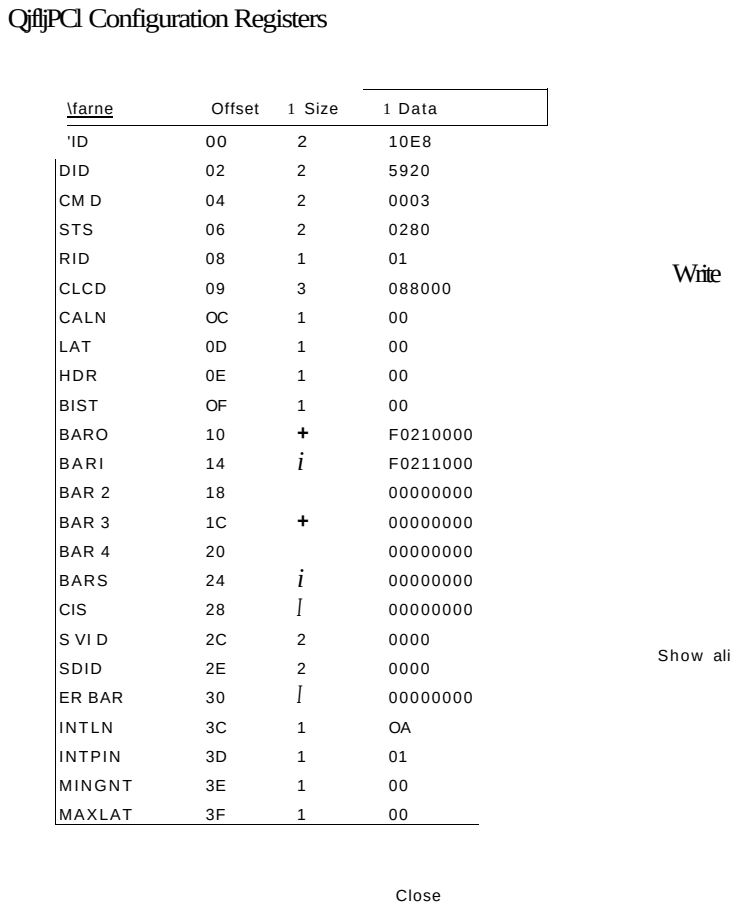



Figura 3.21: Conteúdo dos registradores de configuração do S5920.

foi enviado para esta empresa, para que pudesse ser gravado na EEPROM. Testes do S5920, utilizando esta EEPROM foram refeitos de modo garantir o funcionamento correto da placa LDN/UFPE-2001. Na Figura 3.24, o leitor poderá visualisar o conteúdo desta nvRAM.

Os primeiros endereços de *offset* 40h e 41/i, possuem o conteúdo *10E8h* que é o registrador de identificação do fabricante AMCC. Os próximos endereços 42h e 43h, possuem o conteúdo 5920h que é o registrador de identificação do dispositivo S5920.

No endereço 48h encontra-se o conteúdo 01h, indicando a primeira revisão de configuração.

Os endereços 49h a 4Bh, possuem o conteúdo 08h 80h 00h (outros sistemas periféricos) que identifica a função básica do dispositivo.

4Eh 00h

única.

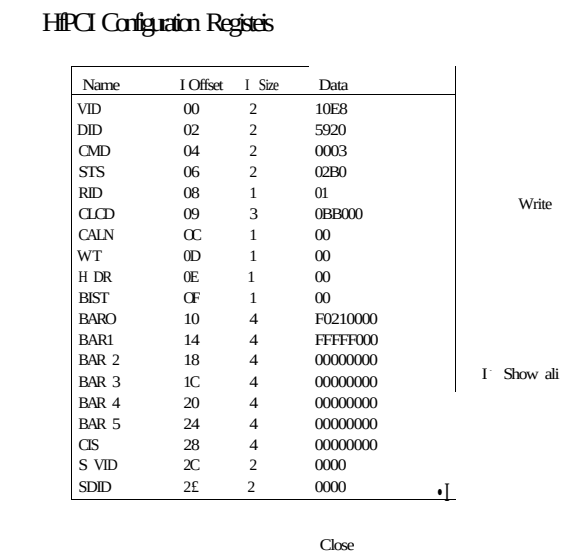



Figura 3.22: Registrador de configuração da placa LDN/UFPE-2001.

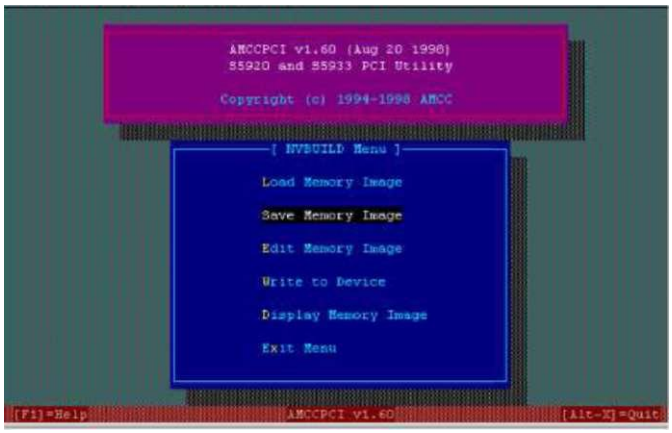



Figura 3.23: Interface de leitura e escrita dos registradores do S5920.

O endereço 4Fh possui o conteúdo 00/i, deixando o auto-teste do barramento externo desabilitado.

Os endereços 50h a 53h possuem o conteúdo \0hE8h FFh 80h, tornando os conteúdos da EEPROM válidos.

Os endereços 6Ch e 6Dh possuem o conteúdo 00h 00h, não sendo necessário fazer a distinção da identificação do fabricante do subsistema.

6Eh 6Fh 00h 00h

a distinção da identificação do subsistema.

Os endereços 70h a 73h possuem o conteúdo 00h 00h 00h, deixando desabilitado o registrador de endereço base da ROM de expansão.

Edição do buffer de trabalho — Cursor : 0000 — 16:36:46

| End. | Conteúdo em Hexadecimal | | | | | | | | | | | | | | | | ASCII |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0000 | 93 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | |
| 0010 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | |
| 0020 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | |
| 0030 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | |
| 0040 | E8 | 10 | 20 | 59 | FF | FF | FF | FF | 01 | 00 | 80 | 08 | FF | FF | 00 | 00 | . . Y. . . . . |
| 0050 | 80 | FF | E8 | 10 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | |
| 0060 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | 00 | 00 | 00 | |
| 0070 | 00 | 00 | 00 | 00 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | 01 | FF | |
| 0080 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | |
| 0090 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | |
| 00H0 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | |
| 00B0 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | . . |

ALT Freenche Copia coHpara prOcura (;Ursor TAB

Figura 3.24: Mapa de conteúdo da nvRAM

O endereço 7Ch possui o conteúdo *FFh,* indicando a rotina de interrupção para o S5920.

O endereço 7Dh possui o conteúdo 01h, indicando que o dispositivo gerará pedido de interrupção *"ÍNTA/-".*

Os demais endereços da EEPROM não são usados, ficando com seus conteúdos *FFh*

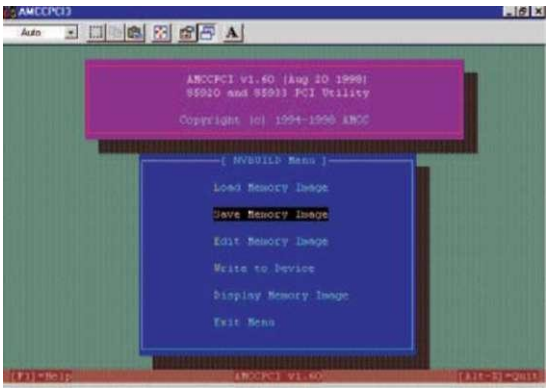



Figura 3.25: Menu de construção da memória nvRAM.

### 3.12.4 Configurando a nvRAM com o aplicativo AMCC PCI

Com o aplicativo AMCC PCI, versão 1.60, foi possível configurar a memória nvRAM para inicialização da placa LDN/UFPE-2001 "à quente", possibilitando sua operação no modo *Pass-Thru.* Para isto foram adotados os seguintes procedimentos: executando o aplicativo, inicialmente foi mostrada uma lista com os dispositivos conectados ao barramento PCI; selecionando através dos menus o dispositivo 5920 e a memória nvRAM 24C08 na qual deseja-se programar, foi carregado o conteúdo da nvRAM na

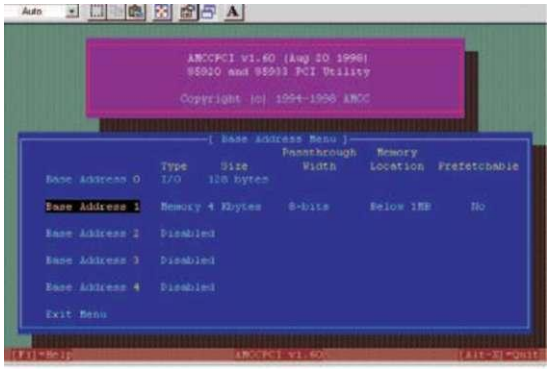



Figura 3.26: Menu de edição do endereço base-1.

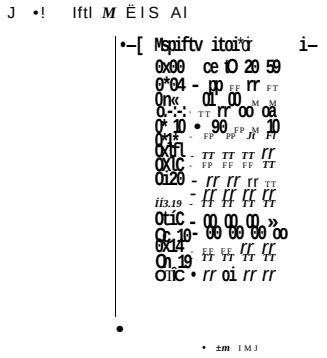


Figura 3.27: Conteúdos atualizados da memória nvRAM.

memória do computador.

Retornando ao menu de construção, foi salvo o conteúdo em hexadecimal numa área do disco rígido do computador, com a opção *"Save Memory Imagé'* indicado na Figura 3.25

Retornando ao menu de construção, através da opção *"Edit Memory Imagé'* foi definido a base de endereço-1, com as seguintes características: tipo de memória, com locação abaixo de *1MByte, sem prefetchable,* tamanho de memória de 4 *kBytes* (lk DWORDS), com 8 *bits* de largura. Após confirmado esses dados foi obtida a seguinte tela, Figura 3.26:

Retornando mais uma vez ao menu de construção foi feito a escrita desta configuração na memória nvRAM, através da opção " *Write to Device".*

As informações atualizadas contida na memória nvRAM, puderam ser visualizadas através da opção *"Display Memory Imagem",* conforme o leitor poderá ver na Figura 3.27.

### 3.12.5 Subsistema de varredura

0 subsistema de varredura consiste do PLD (contador), conversor DA com um operacional rápido. Para realizar o teste foi programada um contador de oito *bits* no PLD, os quatro bits menos significativos foram utilizados para gerar o sinal de varredura X e os quatro *bits* mais significativos são convertidos para gerar o sinal *Y*. Na Figura 3.28 é apresentado os sinais gerados e na Figura 3.29 tem-se uma demonstração de operação

*XY*

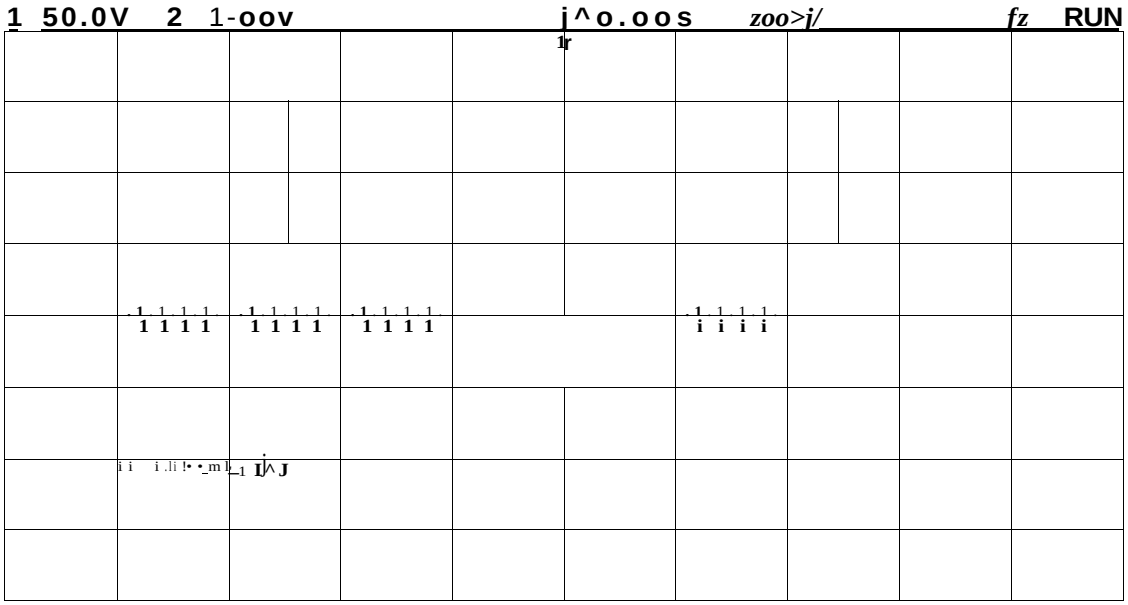



Figura 3.28: Geração de sinais de varredura para controle da posição do feixe eletrônico.

## 3.13 Dificuldades encontradas

Este trabalho de mestrado, diante as diversas dificuldades enfrentadas durante o desenvolvimento do mesmo, principalmente no caracter prático trouxe vários aprendizados.

Foi percebido pelo autor que na prática, a construção de uma placa LDN/UFPE-2001 não é tão simples quanto se parece. As dificuldades aparecem à medida que os novos desafios vão surgindo. Assim em frente a esse confronto foi adquirido pelo autor maturidade e auto-confiança.

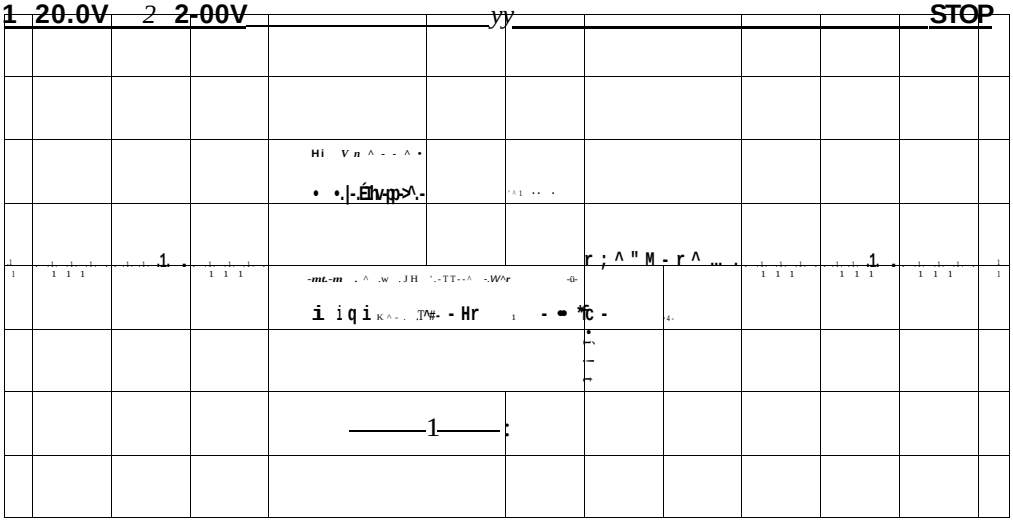



Figura 3.29: Demonstração de varredura em uma matriz 16 x 16 utilizando um osciloscópio.

Durante o curso de mestrado, o autor teve a oportunidade de apresentar oralmente os seus trabalhos desenvolvidos: em vários seminários apresentados no LDN (Laboratório de Dispositivos e Nanoestruturas); palestras no auditório do SENAI Joseph Turton Júnior e também no concresso COBENGE 2001 em Porto Alegre, no qual o autor fez uma apresentação oral, utilizando o recurso do *Power Point,* exposição com painel e publicação do artigo: "Placa de desenvolvimento PCI para instrumentação virtual utilizando o *chip* AMCC S5920 [38]".

No aspecto teórico, o autor pôde conhecer novos conceitos, tais como, sistemas embarcados e sistemas de tempo real, sistema operacional Linux e seus comandos, Linux de Tempo Real, instrumentação virtual, programação de PLD/FPGA, características do barramento PCI *plug and play,* arquitetura do dispositivo S5920, entre outros assuntos.

Do ponto de vista prático, o autor estudou características dos diversos dispositivos eletrônicos, tais como, o microcontrolador PIC 18C858 da Microchip, com 1536 *kbytes* de memória RAM interna e conversor AD interno de 8 *bits* com velocidade de *AQMHz.* O microcontrolador iria ser utilizado inicialmente na placa LDN/UFPE-2001, mas por não estar disponível comercialmente, o projeto foi implementado utilizando o conversor AD (TDA8703), o PLD (EMP7128SLC84-7) e a memória SRAM (TC55400/FL-70), conforme foi mencionado nas seções anteriores. O autor buscou os dispositivos

com taxas de transferênica de dados mais rápidos para atender as exigências de imagem. Porém não foi possível encontrar comercialmente conversor DA e amplificador operacional com *set time* inferior a 100ns.

Outra dificuldade encontrada foi na hora da compra de componentes não mais comecializados (caso dos *latches),* tendo que fazer alterações no projeto da placa LDN/UFPE-2001. Assim toda a placa teve que ser reprojetada, com componentes disponíveis comercialmente.

O funcionamento do circuito foi alcançado aos poucos, diante vários testes efetuados com a placa LDN/UFPE-2001. Com a programação do PLD verificada na simulação e na placa de desenvolvimento, foi necessário testar cada um de seus módulos internos, utilizando uma frequência de relógio inferior, para que fosse possível acompanhar o comportamento do circuito a cada ciclo de relógio, a fim de encontrar algumas falhas do projeto. Para auxiliar esta pesquisa foram utilizados os seguintes equipamentos de bancada: multímetro digital, gerador de forma de onda e osciloscópio digital.

O leiaute da placa de circuito impresso teve que ser desenhada manualmente, uma vez que, o aplicativo não possuia em sua biblioteca as características do controlador S5920. Outra dificuldade encontrada foi para confeccionar esta placa com as características de dupla face e furo metalizado.

O *chip* S5920 com encapsulamento PQFP e 160 pinos foi soldado na placa manualmente pelo autor, utilizando um ferro de solda comum de 20 W de potência. Esta técnica exigiu um cuidado redobrado, visto que o dispositivo não suporta altas temperaturas na soldagem de seus terminais. Além da dimensão de seus pinos, consideravelmente pequenos, houve uma preocupação em não por em risco possíveis maus contatos, ou curto-circuitos entre eles. Para reparar estas falhas, foi utilizado o microscópio óptico do LDN.

Diante dessas experiências adquirida, o autor encontra-se mais capacitado para superar novos desafios, de maneira mais eficiente.

# Capítulo 4

# Conclusões e trabalhos futuros

## 4.1 Conclusões

A placa desenvolvida faz uso de uma arquitetura concorrente para aumentar seu desempenho. Enquanto um banco de memória é escrito, o outro é lido. Técnicas de tempo real são utilizadas para que o processo ocorra transparentemente. Observa-se que neste sistema o limite de desempenho é o conversor DA.

Esse trabalho apresenta um detalhamento da interface PCI e o comportamento do sistema operacional durante a inicialização do computador, incluindo o funcionamento *plug-and-play*.

O uso do controlador PCI S5920 da AMCC facilita o projeto de interface do barramento PCI com o meio externo, pois torna os detalhes do barramento PCI transparente para o projetista e permite a transferência de dados em tempo real a uma taxa alta, sendo assim, um dispositivo ideal para projeto de aquisição de imagens.

O uso do aplicativo para criação de interfaces gráficas Delphi® *ou* Kylix® facilita o desenvolvimento de interfaces para operação da placa.

O sistema projetado e desenvolvido pode ter grande utilização com fins educativos e é uma prova de conceito de como se implementar um sistema de litografia eletrônica controlado por microcomputador. Essa aplicação foi inspirada no Convênio entre o Laboratório de Dispositivos e Nanoestruturas - LDN/UFPE e o Instituto da Tecnologia de Microeletrônica da Academia Russa de Ciências.

## 4.2 Trabalhos futuros

Segue aqui algumas sugestões para melhoramento da placa LDN/UFPE-2001:

- Incorporar a nvRAM ao leiaute da placa;
- Expandir o barramento de endereçamento da memória para fazer uso de toda a capacidade de memória instalada na placa;

sinais na placa;

de varredura para que não seja necessariamente sequencial.

# Apêndice A

# Listagens VHDL

## A.1 Contador para escrita

```
-- Projeto Placa PCI de Aquisição de Dados
-- contador.vhd - Contador para escrita
-- Versão 1.0 - 28/02/2002
-- Autor: Jener Toscano

ENTITY contador IS
    PORT
    (
        clock_counter       : IN    bit;
        clear_counter       : IN    bit;
        end_counter         : OUT   bit;
        output_counter      : OUT INTEGER RANGE 0 TO 255
    );
END contador;

ARCHITECTÜRE a OF contador IS
    SIGNAL   internal_output_counter  :  INTEGER RANGE 0 TO 255;
    SIGNAL   internal_end_counter     :  bit;
```

```
BEGIN


    PROCESS (clock_counter, clear_counter)
    BEGIN
        IF clear_counter = '0' THEN
          internal_output_counter <= 0;
        ELSIF (clock_counter'EVENT AND clock_counter = '1') THEN
          IF internal_output_counter = 254 THEN
            internal_end_counter <= '1';
          ELSE
            internal_end_counter <= '0';
          END IF;
          internal_output_counter <= internal_output_counter + 1;
        END IF;
    END PROCESS;
        output_counter <= internal_output_counter;
    end_counter <= internal_end_counter;
END a;
```

## A.2 Controlador de escrita nas memórias

```
-- Projeto Placa PCI de Aquisição de Dados
-- escrita.vhd - Controlador de Escrita nas Memórias
-- Versão 1.0 - 28/02/2002
-- Autor: Jener Toscano

ENTITY escrita IS
    PORT
    (
        clock_escrita       : IN    bit;
        reset_escrita       : IN    bit;
        flaga_escrita       : IN    bit;
        flagb_escrita       : IN    bit;
        clk         : OUT   bit;
        ce_inv          : OUT   bit;
        o_ea_inv        : OUT   bit;
        o_eb_inv        : OUT   bit;
        r_wa_inv        : OUT   bit;
        r_wb_inv        : OUT   bit;
        oea_inv         : OUT   bit;
        lea         : OUT   bit;
        oeb_inv         OUT   bit;
        leb         : OUT   bit;
        ycs_inv         : OUT   bit;
        xcs_inv         : OUT   bit;
        incout          : OUT   bit
    );
END escrita;
```

```
ARCHITECTURE arc_escrita of escrita is
  TYPE estados is (ei, eO, el, e2, e3, e4, e5, e6, e7, e8, e9);
  SIGNAL estado: estados;
BEGIN
  PROCESS(clock_escrita,reset_escrita)
  begin
    if (reset_escrita='0') then
      estado <= ei;
        elk      <= >0>;
        ce_inv       <= '1';
        o_ea_inv     <= 'V ;
        o_eb_inv     <= 'V ;
        r_wa_inv     <= '1';
        r_wb_inv     <= 'V ;
        oea_inv      <= 'V ;
        lea      <= >0>;
        oeb_inv      <= 5i5;
        leb      <= >0>;
        ycs_inv      <= 5i5;
        xcs_inv      <=
        incout       <= >0>;
    elsif (clock_escrita=5l5 and clock_escrita5event) then
      case estado is
        when ei =>
            estado <=e0;
            xcs_inv <=505;
            ycs_inv <=505;
            incout <=505;
        when eO =>
            estado <=el;
            xcs_inv <='15;
            ycs_inv <='15;
```

```
        elk   <=1;
        lea   <=1;
    when el =>
        estado <=e2;
        elk <='0';
        ce_inv <='0';
        o_ea_inv <='0';
        oea_inv <='0';
    when e2 =>
        estado <=e3;
        r_wa_inv <='0';
        lea <='0';
    when e3 =>
        estado <=e4;
        ce_inv  <='l';
        o_ea_inv <='1';
        oea_inv <='1';
        r_wa_inv <='1';
        incout <='1';
    when e4 =>
      if (flaga_escrita ='1' and flagb_escrita'='0') then
        estado <= e4;
        ce_inv  <='l';
        o_ea_inv <='1';
        oea_inv <='1';
        r_wa_inv <='1';
        incout <='!';
      elsif (flaga_escrita ='1' and flagb_escrita='l') then
          estado <= e5;
          xcs_inv <='0';
          ycs_inv <='0';
          incout <='0';
```

```
  elsif (flaga_escrita ='0') then
        estado <= eO;
        xcs_inv <='0';
        ycs_inv <='0';
        incout <='0';
  end if;
when e5 =>
    estado <=e6;
    xcs_inv  <='1';
    ycs_inv  <='1';
    cXk ^— 1'5
    leb <='1';
when e6 =>
    estado <=e7;
    elk <='0';
    ce_inv <='0';
    o_eb_inv <='0';
    oeb_inv <='0';
when e7 =>
   estado <=e8;
    r_wb_inv <='0';
    leb <='0';
when e8 =>
   estado <=e9;
    ce_inv  <='l';
    o_eb_inv <='1';
    oeb_inv  <='1';
    r_wb_inv <='1';
    incout <='1';
when e9 =>
  if (flaga_escrita ='0' and flagb_escrita='l') then
    estado <= e9;
```

```
            ce_inv <='1';
            o_eb_inv <='1';
            oeb_inv  <='l';
            r_wb_inv <='1';
            incout <='1';
          elsif (flaga_escrita ='0' and flagb_escrita='0') then
              estado <= eO;
              xcs_inv  <='0';
              ycs_inv  <='0';
              incout <='0';
          elsif (flaga_escrita ='1') then
                estado <= e5;
                xcs_inv  <='0';
                ycs_inv  <='0';
                incout <='0';
          end if;
        when others =>
      end case;
    end if;
  end process;
end arc_escrita;
```

# A.3 FlipFlop tipo T

## A.3.1 FlipFlop A

```
-- Projeto Placa PCI de Aquisição de Dados
-- flipflopA.vhd - FlipFlop tipo T
-- Versão 1.0 - 28/02/2002
— Autor: Jener Toscano

ENTITY flipflopa IS
    PORT
    (
        ta           : IN    bit;
        clock_flipflopa : IN    bit;
        reset_flipflopa : IN    bit;
        qa           : OUT   bit
    );

END flipflopa;

ARCHITECTURE arc_flipflopa OF flipflopa IS
    SIGNAL  internal_qa : bit;
BEGIN
    PROCESS (clock_flipflopa, reset_flipflopa)
    BEGIN
        IF reset_flipflopa = '0' THEN
            internal_qa <= '0';
        ELSIF (clock_flipflopa'EVENT AND clock_flipflopa = '1') THEN
            if (ta='l' and internal_qa='l') then
                internal_qa <= '0';
            elsif (ta='l' and internal_qa='0') then
                internal_qa <= '1';
            END IF;
```

```
END IF;
END PROCESS;
qa <= internal
ND arc_flipflopa;
```

### A.3.2 FlipFlop B

```
-- Projeto Placa PCI de Aquisição de Dados
-- flipflopb.vhd - FlipFlopb tipo T
-- Versão 1.0 - 28/02/2002
— Autor: Jener Toscano

ENTITY flipflopb IS
    PORT
    (
        tb           : IN    bit;
        clock_flipflopb : IN    bit;
        reset_flipflopb : IN    bit;
        qb           : OUT   bit
    );

END flipflopb;

ARCHITECTURE arc_flipflopb OF flipflopb IS
    SIGNAL  internal_qb : bit;
BEGIN
    PROCESS (clock_flipflopb, reset_flipflopb)
    BEGIN
        IF reset_flipflopb = '0' THEN
            internal_qb <= '0';
        ELSIF (clock_flipflopb'EVENT AND clock_flipflopb = '1') THEN
            if (tb='1' and internal_qb='1') then
                internal_qb <= '0';
            elsif (tb='1' and internal_qb='0') then
                internal_qb <= '1';
            END IF;
        END IF;
```

```
    END PROCESS;
    qb <= internal_qb;
ND arc_flipflopb;
```

## A.4 Controlador de leitura das memórias

```
-- Projeto Placa PCI de Aquisição de Dados
-- leitura.vhd - Controlador de leitura das Memórias
-- Versão 2.0 - 28/02/2002
-- Autor: Jener Toscano

ENTITY leitura IS

    PORT
    (
        clock_leitura       : IN    bit;
        reset_leitura       : IN    bit;
        flaga_leitura       : IN    bit;
        flagb_leitura       : IN    bit;
        ptatn_inv           : IN    bit;
        ptadr_inv           : IN    bit;
        dxfr_inv            : IN    bit;
        o_ed_inv            : OUT   bit;
        t_rd_inv            : OUT   bit;
        o_ec_inv            : OUT   bit;
        lec                 : OUT   bit;
        t_rc_inv            : OUT   bit;
        oed_inv             : OUT   bit;
        oec_inv             : OUT   bit;
        led                 : OUT   bit;
        end_read            : OUT   bit
    );

END leitura;
```

```
ARCHITECTURE arc_leitura of leitura is
  TYPE estados is (eO, el, e2, e3, e4, e5, e6, e7, e8, e9, elO);
  SIGNAL estado: estados;

BEGIN
  PROCESS(clock_leitura,reset_leitura)
  begin
    if (reset_leitura"5"0") then
      estado <= eO;
            lec      <= >0>;
            oec_inv <= "i";
            o_ec_inv     <=
            t_rc_inv     <= "i";
            led      <= >0>;
            oed_inv <= "i";
            o_ed_inv     <= "i";
            t_rd_inv     <= "i";
    elsif (clock_leitura="l" and clock_leitura'event) then
      case estado is
        when eO =>
            if (ptatn_inv="0" and flaga_leitura="1" and flagb_leitura="l") then
              estado <=el;
              end_read <="0";
            elsif (ptatn_inv="0" and flaga_leitura="0" and flagb_leitura"5"0") then
              estado <=e2;
              end_read <="0";
            elsif ((ptatn_inv="l") or (ptatn_inv="0" and flaga_leitura="l" and
flagb_leitura= "0") or (ptatn_inv="0" and flaga_leitura"5"0" and
flagb_leitura= "1")) then
              estado <= eO;
                lec      <= "0";
                oec_inv <= "1";
```

```
        o_ec_inv      <=    '1';
        t_rc_inv      <=    '1';
        led       <= >0>;
        oed_inv <=
        o_ed_inv      <=    '1';
        t_rd_inv      <=    '1';
    end if;


when el =>
    if (ptatn_inv='0' and ptadr_inv='1' and dxfr_inv='l') then
      estado  <=el;
    elsif (ptatn_inv='0' and ptadr_inv='0' and dxfr_inv='l') then
      estado <=e3;
      lec     <=  '1';
    elsif (ptatn_inv='l') then
      estado <= eO;
        lec       <= '0';
        oec_inv <= '1';
        o_ec_inv      <=  '1';
        t_rc_inv     <= '1';
        led       <= '0';
        oed_inv <= '1';
        o_ed_inv     <= '1';
        t_rd_inv     <= '1';
    end if;


when e3 =>
    if (ptatn_inv='0' and ptadr_inv='0' and dxfr_inv='l') then
      estado <=e3;
    elsif (ptatn_inv='0' and ptadr_inv='1' and dxfr_inv='l') then
```

```
        estado  <=e5;
        oec_inv     <='0';
        lec     <='0';
      elsif (ptatn_inv='l') then
        estado <= eO;
          lec       <= '0';
          oec_inv <= '1';
          o_ec_inv      <= '1';
          t_rc_inv     <= '1';
          led       <= '0';
          oed_inv <= '1';
          o_ed_inv      <= '1';
          t_rd_inv     <= '1';
      end if;

    when e5 =>
      if (ptatn_inv='0' and ptadr_inv='1' and dxfr_inv='l') then
        estado  <=e5;
        oec_inv  <='0';
        lec <='0';
      elsif (ptatn_inv='0' and ptadr_inv='l' and dxfr_inv='0') then
        estado  <=e7;
        o_ec_inv  <='0';
      elsif (ptatn_inv='l') then
        estado <= eO;
          lec       <= '0';
          oec_inv <= '1';
          o_ec_inv      <= '1';
          t_rc_inv     <= '1';
          led       <= '0';
          oed_inv <= '1';
          o_ed_inv      <= '1';
```

```
        t_rd_inv    <= '1';
    end if;

when e7 =>
    if (ptatn_inv='0' and ptadr_inv='0' and dxfr_inv='0') then
      estado <=e7;
      o_ec_inv <='0';
    elsif (ptatn_inv='0' and ptadr_inv='0' and dxfr_inv='l') then
      estado <=e3;
      lec   <=1>;
    elsif (ptatn_inv='l') then
      estado <= e9;
      end_read   <='1';
    elsif (ptatn_inv='0' and ptadr_inv='1') then
      estado <= eO;
        lec      <= '0';
        oec_inv <= '1';
        o_ec_inv     <= '1';
        t_rc_inv    <= '1';
        led      <= '0';
        oed_inv <= '1';
        o_ed_inv    <= '1';
        t_rd_inv    <= '1';
    end if;

when e9 =>
    estado <= eO;
        lec          <= '0';
        oec_inv <= '1';
        o_ec_inv     <= '1';
        t_rc_inv    <= '1';
        led      <= '0';
```

```
        oecLinv <=  '1';
        o_ed_inv    <= "1";
        t_rd_inv    <= "1";

when e2 =>
    if (ptatn_inv="0" and ptadr_inv="1" and dxfr_inv='l') then
      estado <=e2;
    elsif (ptatn_inv="0" and ptadr_inv="0" and dxfr_inv='l') then
      estado <=e4;
      led    <= '1';
    elsif (ptatn_inv="l") then
      estado <= eO;
        lec     <= "0";
        oec_inv <= "1";
        o_ec_inv    <= '1';
        t_rc_inv    <= '1';
        led     <= "0";
        oed_inv <= "1";
        o_ed_inv    <= '1';
        t_rd_inv    <= "1";
    end if;


when e4 =>
    if (ptatn_inv="0" and ptadr_inv="0" and dxfr_inv='l') then
      estado <=e4;
    elsif (ptatn_inv="0" and ptadr_inv="1" and dxfr_inv='l') then
      estado <=e6;
      oed_inv   <="0";
      led   <="0";
    elsif (ptatn_inv="l") then
      estado <= eO;
```

```
        lec      <= '0';
        oec_inv <= '1';
        o_ec_inv     <=  '1';
        t_rc_inv     <= '1';
        led      <= '0';
        oed_inv <= '1';
        o_ed_inv     <= '1';
        t_rd_inv     <= '1';
    end if;

when e6 =>
    if (ptatn_inv='0' and ptadr_inv='1' and dxfr_inv='l') then
      estado <=e6;
      oed_inv <='0';
      led <='0';
    elsif (ptatn_inv='0' and ptadr_inv='1' and dxfr_inv='0') then
      estado <=e8;
      o_ed_inv <='0';
    elsif (ptatn_inv='l') then
      estado <= eO;
        lec      <= '0';
        oec_inv <= '1';
        o_ec_inv     <=  '1';
        t_rc_inv     <= '1';
        led      <= '0';
        oed_inv <= '1';
        o_ed_inv     <=  '1';
        t_rd_inv     <= '1';
    end if;

when e8 =>
    if (ptatn_inv='0' and ptadr_inv='0' and dxfr_inv='0') then
```

```
          estado <=e8;
          o_ed_inv <="0";
        elsif (ptatn_inv="0" and ptadr_inv="0" and dxfr_inv='l') then
          estado <=e4;
          led   <=1>;
        elsif (ptatn_inv="l") then
          estado <= elO;
          end_read   <="1";
        elsif (ptatn_inv="0" and ptadr_inv="1') then
          estado <= eO;
            lec      <= "0";
            oec_inv <= "1";
            o_ec_inv     <= '1';
            t_rc_inv    <= '1';
            led      <= "0";
            oed_inv <= "1";
            o_ed_inv     <= "1";
            t_rd_inv     <= "1";
        end if;

    when elO =>
        estado <= eO;
            lec      <= "0";
            oec_inv <= "1";
            o_ec_inv     <= '1';
            t_rc_inv     <= '1';
            led      <= "0";
            oed_inv <= "1";
            o_ed_inv     <= '1';
            t_rd_inv     <= "1";
    when others =>
end case;
```

```
    end if;
  end process;
end arc_leitura;
```

## A.5 Fechamento do PLD da placa PCI

```
-- Projeto Placa PCI de AquisiGIo de Dados
-- placapci.vhâ - Fechamento do PLD da Placa PCI
— Verslo 1.0 - 28/02/2002
— Autor: Jener Toscano

entity placapci is
port (
  1*6 S6"t          : in bit;
  clock             : in bit;
  ptatn_inv         : in bit;
  ptadr_inv         : in bit;
  dxfr_inv          : in bit;
  clk               : out bit;
  ce_inv            : out bit;
  o_ea_inv          : out bit;
  o_eb_inv          : out bit;
  r_wa_inv          : out bit;
  r_wb_inv          : out bit;
  oea_inv           : out bit;
  lea               : out bit;
  oeb_inv           : out bit;
  leb               : out bit;
  o_ed_inv          : out bit;
  t_rd_inv          : out bit;
  o_ec_inv          : out bit;
  lec               : out bit;
  t_rc_inv          : out bit;
  oec_inv           : out bit;
  oed_inv           : out bit;
  led               : out bit;
```

```
  xcs_inv       : out bit;
  ycs_inv       : out bit;
  adr           : out bit_vector (7 downto 0)
  var           : out bit_vector (7 downto 0)
  var_gnd       : out bit_vector (7 downto 0)
  adclk     : out bit
  );
end placapci;


architecture arc_placapci of placapci is
component contador
port (
        clock_counter       : IN    bit;
        clear_counter       : IN    bit;
        end_counter         : OUT   bit;
        output_counter      : OUT   bit_vector(7 downto 0 )
 );
end component;

component flipflopa
port (
        ta              : IN    bit;
        clock_flipflopa     : IN    bit;
        reset_flipflopa     : IN    bit;
        qa              : OUT   bit
 );
end component;

component flipflopb
port (
        tb                IN    bit;
```

```
        clock_flipflopb      : IN    bit;
        reset_flipflopb      : IN    bit;
        qb                   : OUT   bit
 );
end component;

component escrita
port (
    clock_escrita               : IN    bit;
        reset_escrita        ; IN    bit;
        flaga_escrita        ; IN    bit;
        flagb_escrita        ; IN    bit;
        elk          : OUT bit;
        ce_inv           : OUT   bit;
        o_ea_inv         : OUT   bit;
        o_eb_inv         : OUT   bit;
        r_wa_inv         : OUT   bit;
        r_wb_inv         : OUT   bit;
        oea_inv          : OUT   bit;
        lea          : OUT   bit;
        oeb_inv          : OUT   bit;
        leb          : OUT   bit;
        ycs_inv          : OUT   bit;
        xcs_inv          : OUT   bit;
        incout           : OUT   bit

end component;

component leitura
 port (
        clock_leitura        : IN    bit;
        reset_leitura        : IN    bit;
```

```
        flaga_leitura       : IN    bit;
        flagb_leitura       : IN    bit;
        ptatn_inv       : IN    bit:
        ptadr_inv       : IN    bit
        dxfr_inv        : IN    bit
        o_ed_inv        : OUT   bit
        t_rd_inv        : OUT   bit
        o_ec_inv        : OUT   bit
        lec         : OUT   bit;
        t_rc_inv        : OUT   bit;
        oed_inv         : OUT   bit;
        oec_inv         : OUT   bit;
        led         : OUT   bit;
        end_read        : OUT   bit
  );
end component;

-- declaracao dos sinais utilizados para a montagem da placa
 signal counter_flipflopa       : bit;
 signal leitura_flipflopb       : bit;
 signal escrita_counter         : bit;
 signal flipflopa_escrita       : bit;
 signal flipflopb_leitura       : bit;
 signal internal_var            : bit_vector(7 downto 0);
begin

 bl_contador        : contador
port map (escrita_counter, reset, counter_flipflopa, internal_var);
 b2_flipflopa       : flipflopa
port map (counter_flipflopa, clock, reset, flipflopa_escrita);
 b3_flipflopb       : flipflopb
port map (leitura_flipflopb, clock, reset, flipflopb_leitura);
```

```
 b4_escrita          : escrita
port map (clock, reset, flipflopa_escrita, flipflopb_leitura, elk, ce_inv,
o_ea_inv, o_eb_inv, r_wa_inv, r_wb_inv, oea_inv, lea, oeb_inv, leb, ycs_inv,
xcs_inv, escrita_counter); b5_leitura          : leitura
port map (clock, reset, flipflopa_escrita, flipflopb_leitura, ptatn_inv,
ptadr_inv, dxfr_inv, o_ed_inv, t_rd_inv, o_ec_inv, lee, t_rc_inv, oed_inv,
oec_inv, led, leitura_flipflopb);
 adclk    <= clock;
 var      <=internal_var;
 var_gnd <= ("00000000");
 adr      <= internal_var;
end arc_placapci;
```

# Apêndice B

# MAX+PLUS II da ALTERA: programação e símbolos

## B.1 Criando um arquivo novo

1. Escolher New *(File Menu),* selecione *Text Editor File.*
2. Se necessário maximizar a tela do editor.
3. Escolher *Save As (File Menu),* escolhendo o nome do arquivo com extensão .vhd

   Obs.: Salvar o arquivo em disco. O nome do arquivo deve ser o mesmo da entidade *(ENTITY).*
4. Escolher *Project Set Project to Current File* ou *Project Name (File menu)*
5. Ativar o comando *Syntax Coloring (Options menu).* Este comando auxilia a edição de programas tipo texto enfatizando com cores diferentes as palavras chaves, comentários, variáveis e sinais, tornando fácil a visualização da edição do programa.

Após a edição do programa, chegou a hora de verificarmos possíveis erros de sintax.

1. Escolher *Project Save & Check (File menu).*
2. Se tudo estiver ok, compile o programa. Acionar *START* para compilação ou vá até *File menu* e escolha *Save & Compile.* Uma vez salvo, checado a sintaxe e compilado o circuito para um chip especificado, pode-se simular o circuito.

## B.2 Simulação

### B.2.1 Gerando ondas

1. Escolher *New* para criar um arquivo de simulação.
2. Selecionar *Waveform Editor File.*
3. Escolher OK.
4. Se necessário maximizar a tela.
5. Escolher *Enter Nodes from SNF (Node menu).*
6. Escolher *List* para listar as entradas disponíveis e saídas.
7. Escolher nós de tela *Available & Groups* e selecionar os nós.
8. Copiar os sinais para a janela da direita, use => para copiar.
9. Escolher OK.
10. Escolher *Save As,* dando um nome ao arquivo, concluindo com OK.
11. Gerar as ondas necessárias.
12. *Save (File menu).* Escolher *Glose* se quiser sair do programa Simular circuito.

### B.2.2 Simular Circuito

1. Escolher *Simulator (MAX+PLUS II menu).*
2. Escolher *Start.*

# Apêndice C

# Pinagem dos componentes

- Controlador da placa LDN/UFPE-2001 - PLD Altera EPM7128S
- Controlador do barramento PCI - AMCC S5920
-

**Controlador da placa LDN/UFPE-2001 - PLD Altera**

**Função dos sinais de saída do PLD**

CE# - coloca o sinal digital convertido em sua saída.

direciona dados. Em nível baixo, o sentido do dado é de A para B. Em nível alto, o sentido do dado é de B para A.

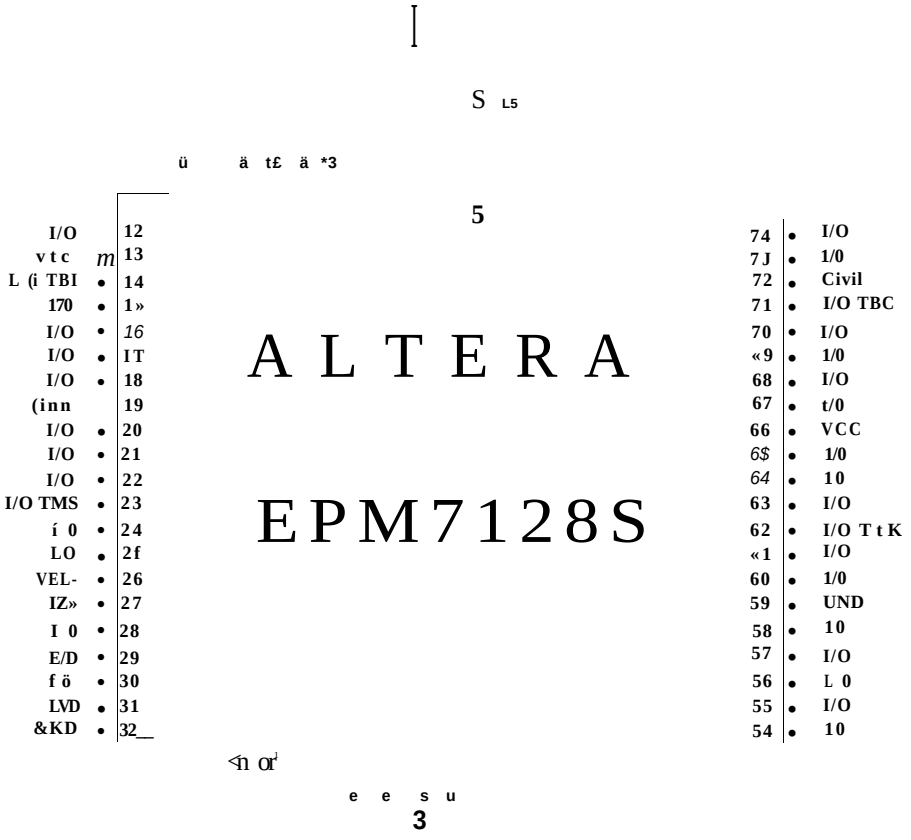



Figura C l : Pinagem do controlador da placa LDN/UFPE-2001.

- MEMÓRIA SRAM: R/W (A, B)# - ativa memória para escrita em nível lógico baixo.
- LATCH: OE (A, B, C, D)# - retém dado da entrada quando em nível baixo. LE (A, B, C, D) - coloca o dado retido na entrada em sua saída. Quando em nível lógico alto.
-

para analógica. (X, Y) CS# e (X,Y) WR# - habilita o conversor DA em nível baixo.

S5920:

Função dos sinais de entrada do PLD

válido.

- PTADR# - informa ao PLD que o barramento DQ contém o endereço do dado requisitado quando em nível baixo juntamente com o sinal PTATN#.
-

bits contidos no barramento DQ quando em nível baixo juntamente com o sinal PTATN#.

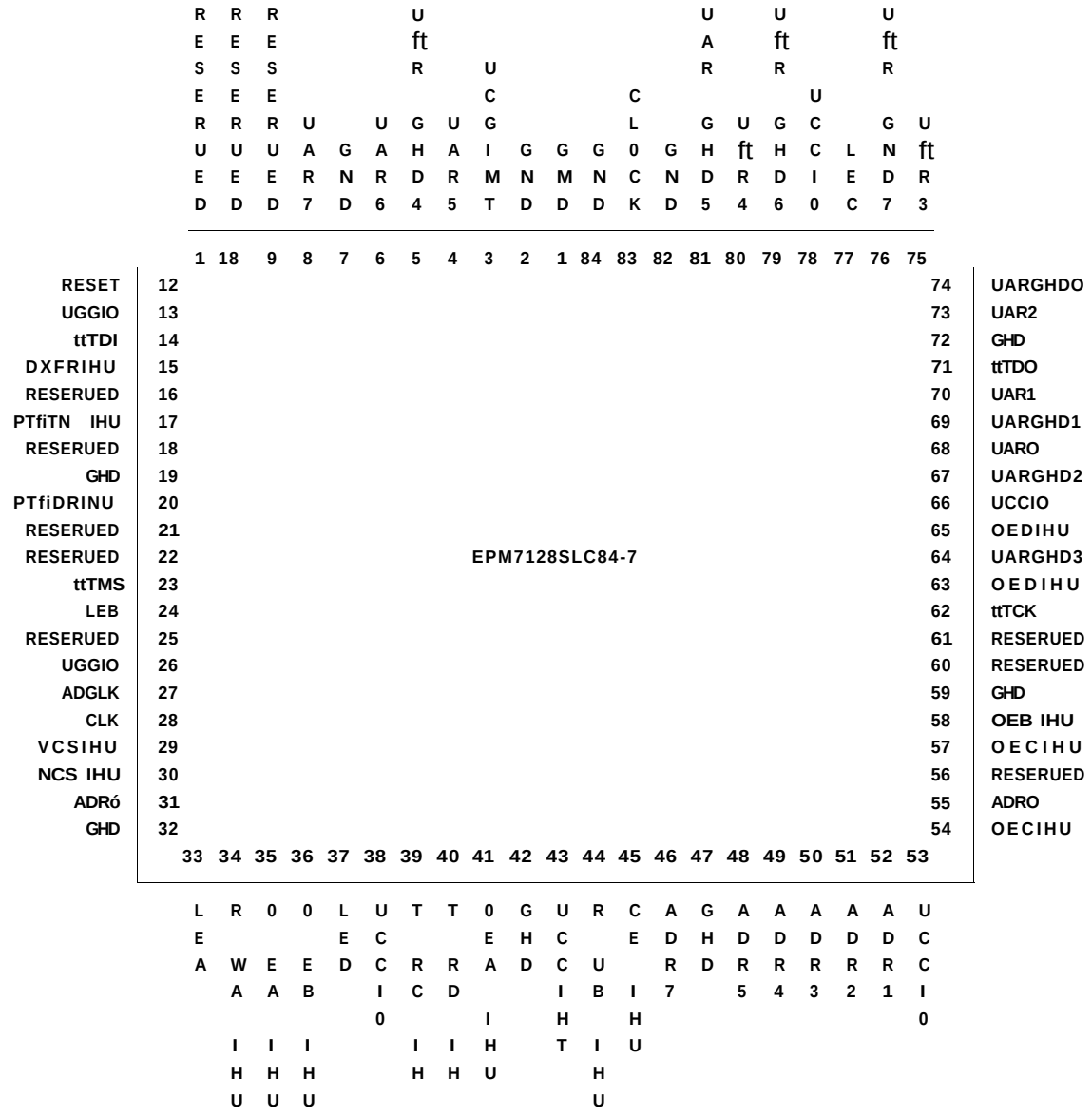



Figura C.2: Pinagem do PLD programado.

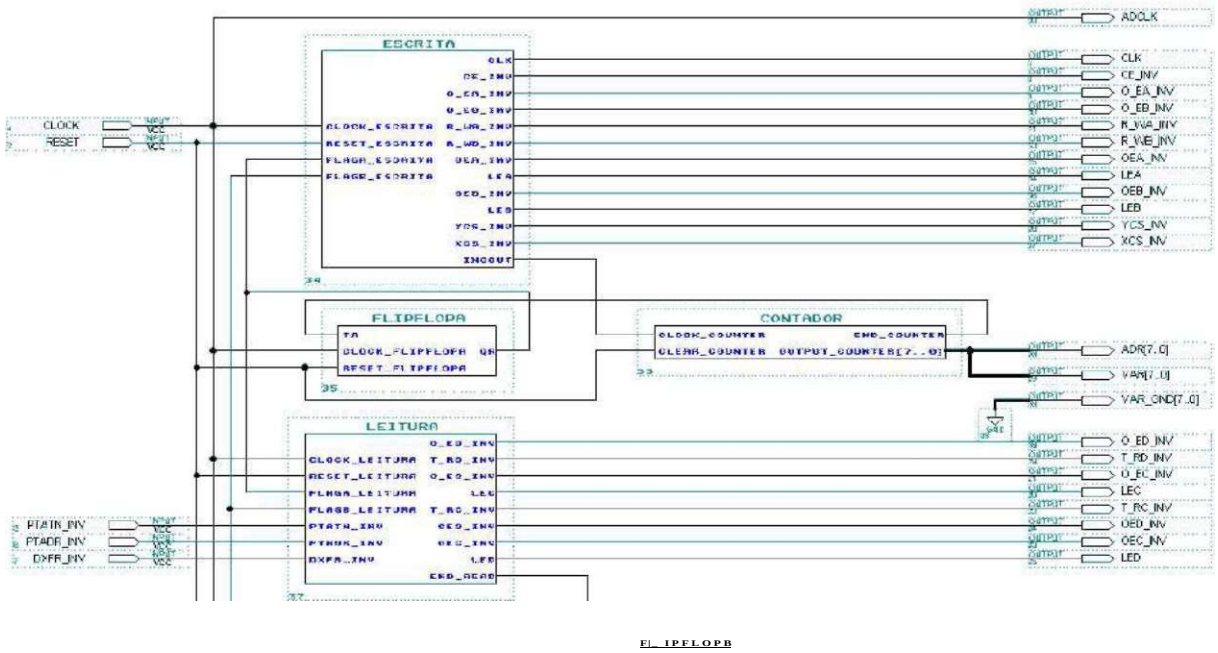



Figura C.3: Módulos do controlador da placa LDN/UFPE-2001.

placap=i2@1 RESETJNV
placapi:i2<g2 y CLOCK
;plat;apci2@12 PTATNINV
:placapci2(g1 S PTADRJNV
iplacapcl2@° DXFR INV

CLK pl3capcl2@&3
CÉJiW placapci2fg|S1
0_FAJ\V placapci2@S<]
0_EB_INV placapci2@5S
R_WA_INV plecapci2@57
R_vV8_INV plscaptl2@SE
"ÕEÀJNV j9l«eapelí@54
LEA pl8capc!2@4-5
Oi B Nv pl3capcl2@44
LEB "" plaeapd2@30
0_ED_INV placapci2@3J
T_RD_ NV p lacapclZ@34
0_EC_INV plecapci2@35
LEC piscapcl2(gï7
T_RC_ NV pleeapcl2@36
OEC INV plecape!2@4-0
Oi I) INV placapcl2@4-1
11 :; placapci2@39
XCSJNV plscapci2@£5
YCSJNV plecapci2@77
ADR7. 0]
VAR[" .0]
VAR_GKC7. C]
ADCLK plaeapcl2(810

Figura C.4: Comunicação do controlador da placa LDN/UFPE-2001 com o mundo externo.

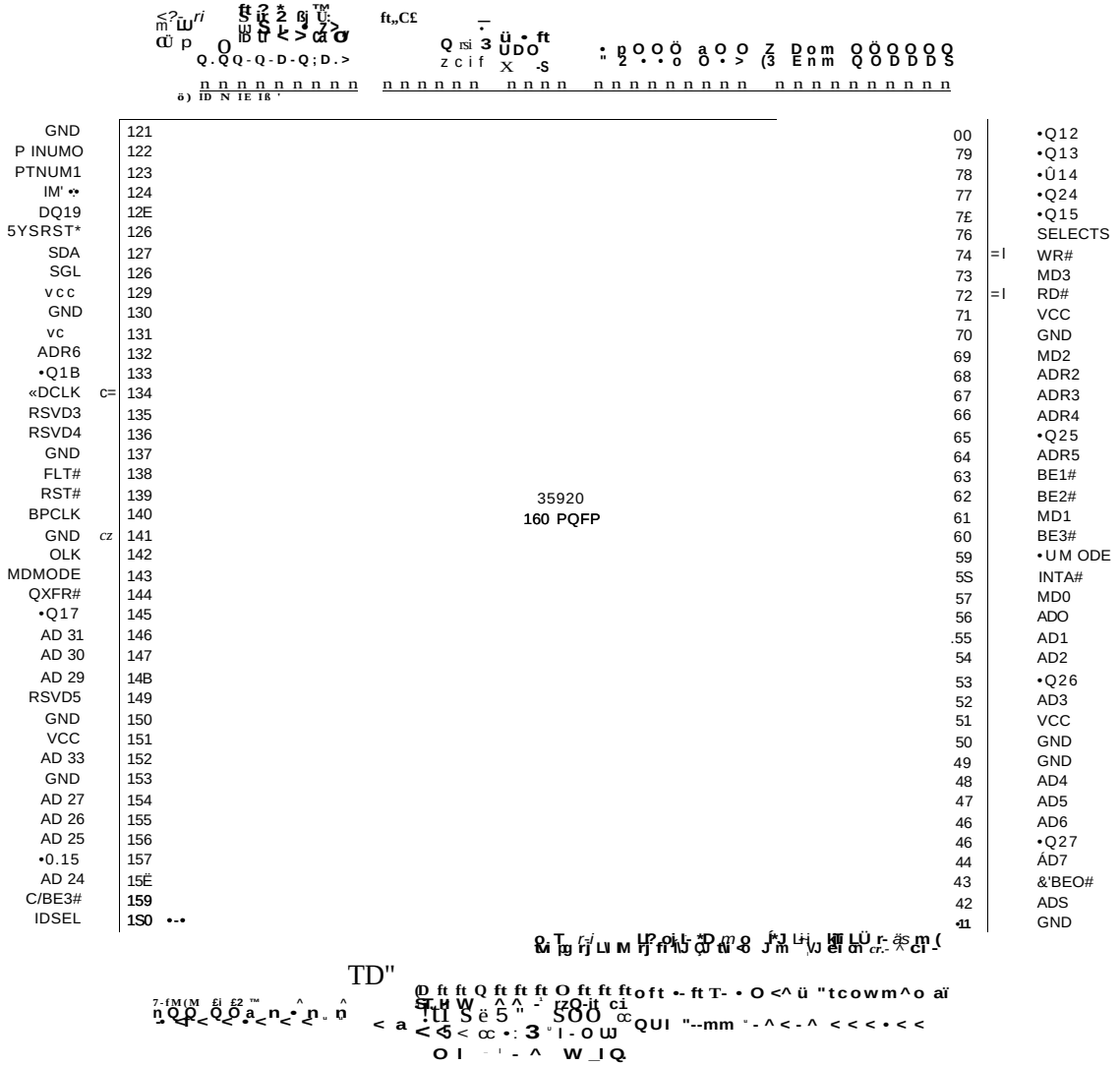



Figura C.5: Pinagem do controlador de barramento PCI.

S5920

**Revision Level**
When Applicable

**Package Option**
**Q**= 160-pin PQFP

**Device Number**

Figura C.6: Identificação do código

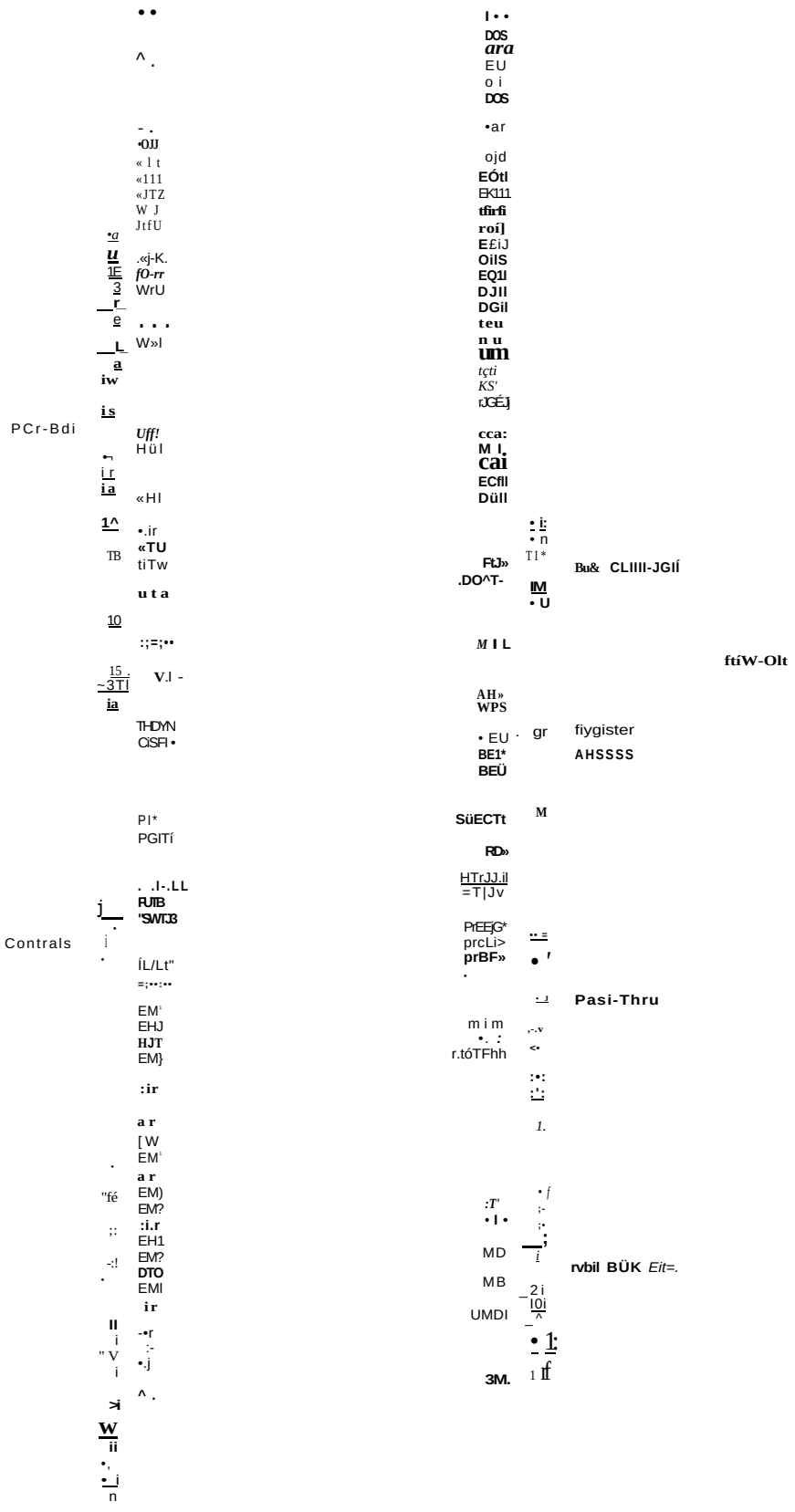



Figura C.7: Pinagem do controlador S5920Q.

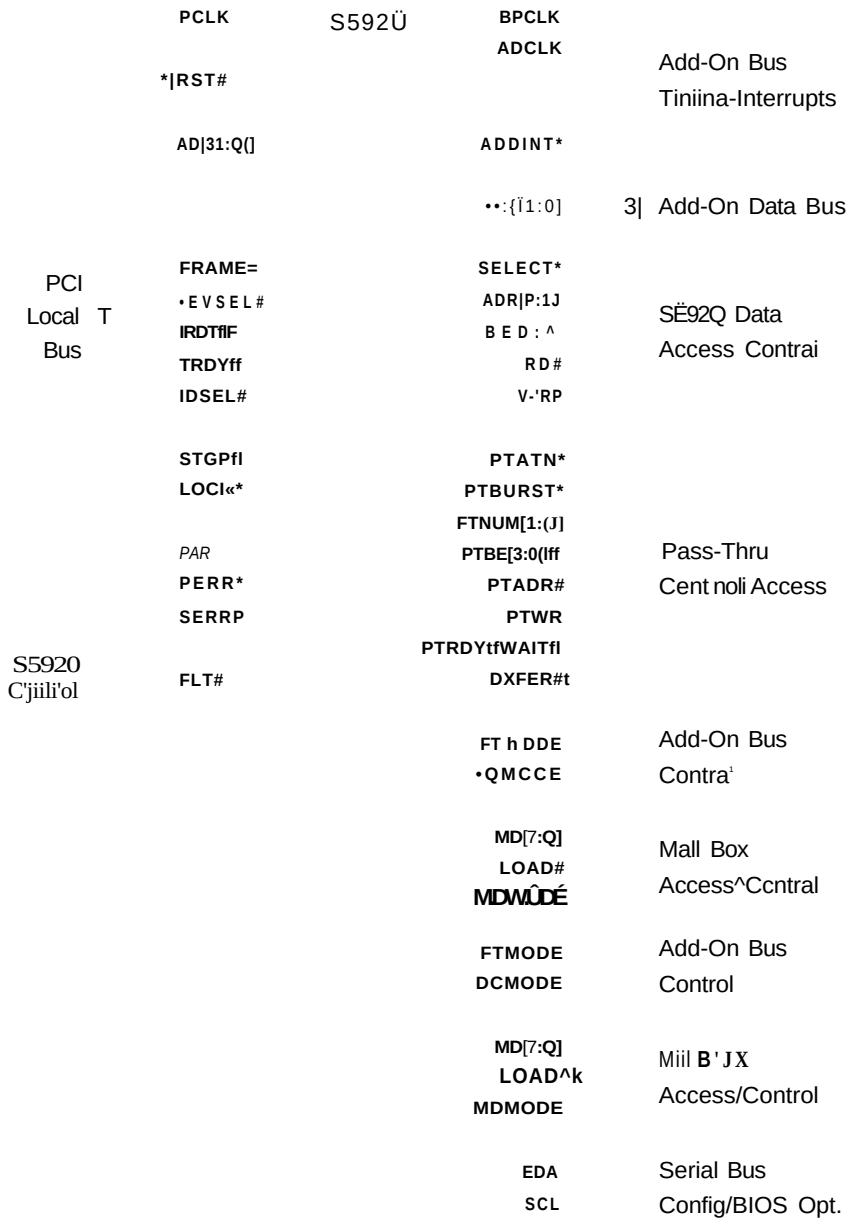



Figura C.8: Pinagem de entrada e saída do S5920Q.

| SYMBOL | PIN | DESCRIPTION |
|---|---|---|
| D1 | 1 | 1 output: bif 1 |
| DÛ | 2 | -I. u v ulpul: bit •• ,LSB. |
| | 'ł | analog around |
| | J | reference *?ll-ige bottom (decoupling) |
| DEC | 5 | •Ktrv.uplitH input (H H M I stabil teuton loop dscouplingj |
| n.c. | 6 | ii.-[ conn-idgd |
| * « | 7 | poBttiv* supply vrrtags teranalog circuits (+5 V] |
| VI | B | ,irral:\i vollag* input |
| Vil | 9 | inference ullage (op [decoupling i |
| n *v* | 10 | n<<l connected |
| QJUF | 11 | ovsrflc^/und^rlo'^ data output |
| D7 | 12 | •Lil.n •. ulpul bit 7 ik-ISEi |
| Do | 15 | •J.ÏÏ : output bite |
| OS | U | dota Output bil ' |
| LH | 15 | J _ 11 : OUlpubit *4* |
| CLK | 1Í | clock input |
| CLK | 17 | . .. M\ — M^:ii.i h . 1 - - input |
| Vfccfj | 19 | pcisrtivB 5uppJy vciltage lordgrtal circuits i + 5 Vj |
| V | 19 | cdBva supply «lt^5» lorouiput stages t+5 *V)* |
| DQND | 2D | •Jruitol ground |
| TT | 21 | input for *tj*T.-'$* complement output rTTL levelinput, aDlnisLOWt |
| TE | *22* | chip ítiabl* in pui. ( TTL Isval input, aclive LQWj |
| D3 | 23 | •l. ii- • bu? |
| *02* | 2J | dak] culpui; bit 2 |

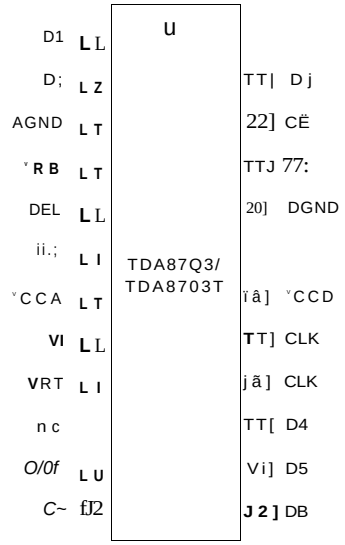



Figura C.9: Descrição e pinagem do conversor AD.

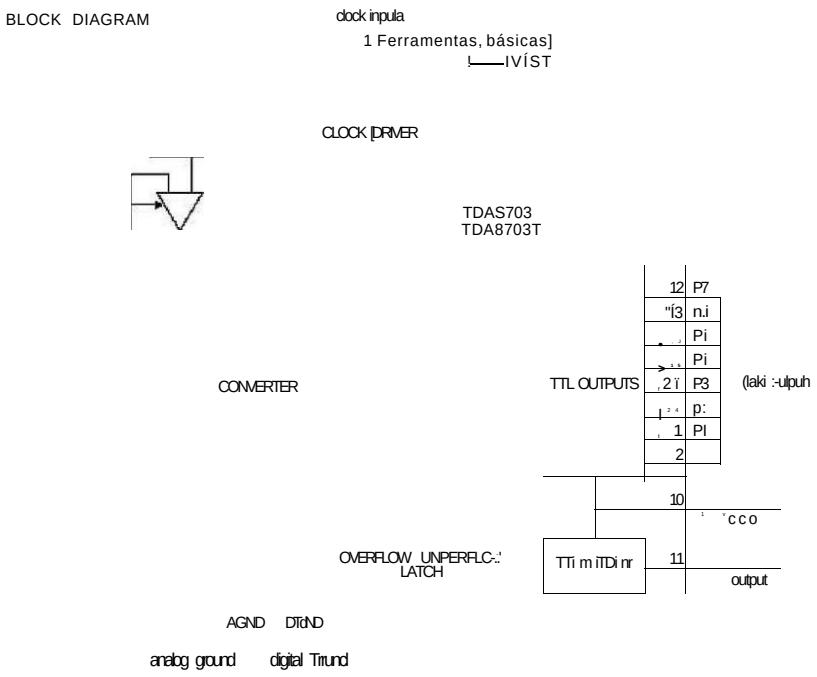
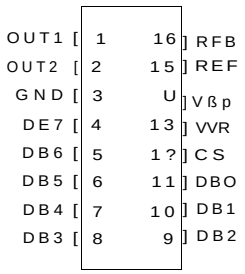



Figura CIO: Diagrama em blocos interno do conversor AD.

| KEY PERFORMANCE SPECIFICATIONS | |
|---|---|
| Resolution | S Bits |
| Linearity error | 1/2 LSB Max |
| Power dissipation at V Q D = 5 V | 5 mW Max |
| Setting time | 100 ns Max |
| Propagation delay time | 80 ns Max |

. N, OR PW PACKAGE
(TOP VIEW)




FN PACKAGE
(TOP VIEW)

Figura C.11: Tabela de especificação e pinagem do conversor DA.

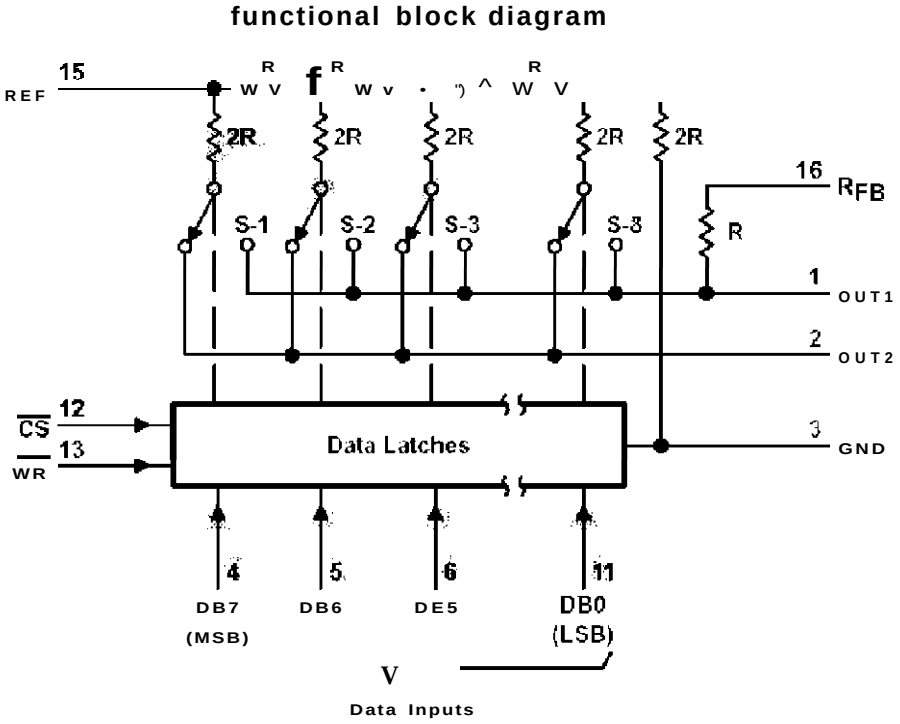
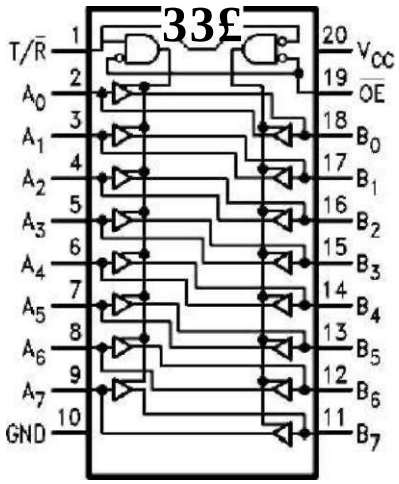



Figura C.12: Diagrama interno do conversor DA.

## Connection Diagram




### Pin Descriptions

| Pin N Wims | Description |
|---|---|
| Ô Ê | Output Ena&fe Input |
| T'ÏÏ | Transmit/Receive Input |
| | Sidg A 3-STATE Inputs or 3-STATE Outputs |
| Bo-By | SidG B 3-STATE Inputs or 3-STATE Outputs |

### Truth Table

| Inputs | | Outputs |
|---|---|---|
| OE | T7R | |
| L | L | Bus B Data to Bus A |
| L | H | Bus A Data to Bus B |
| H | X | HIGH-Z State |

H = HIGH Voltage Levai
L = LOW Voltage Level
K= Immaterial

Figura C.13: Pinagem/Descrição dos pinos/Tabela verdade.

# CONNECTION DIAGRAM

## 8-Pin Plastic (N), and Cerdip (Q) Packages

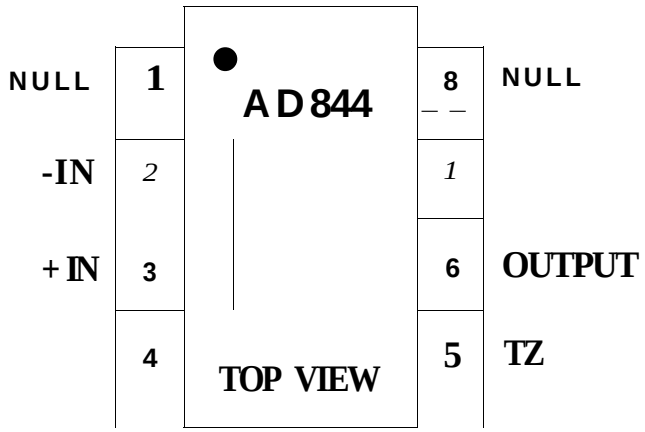



Figura C.14: Pinagem do amplificador operacional.

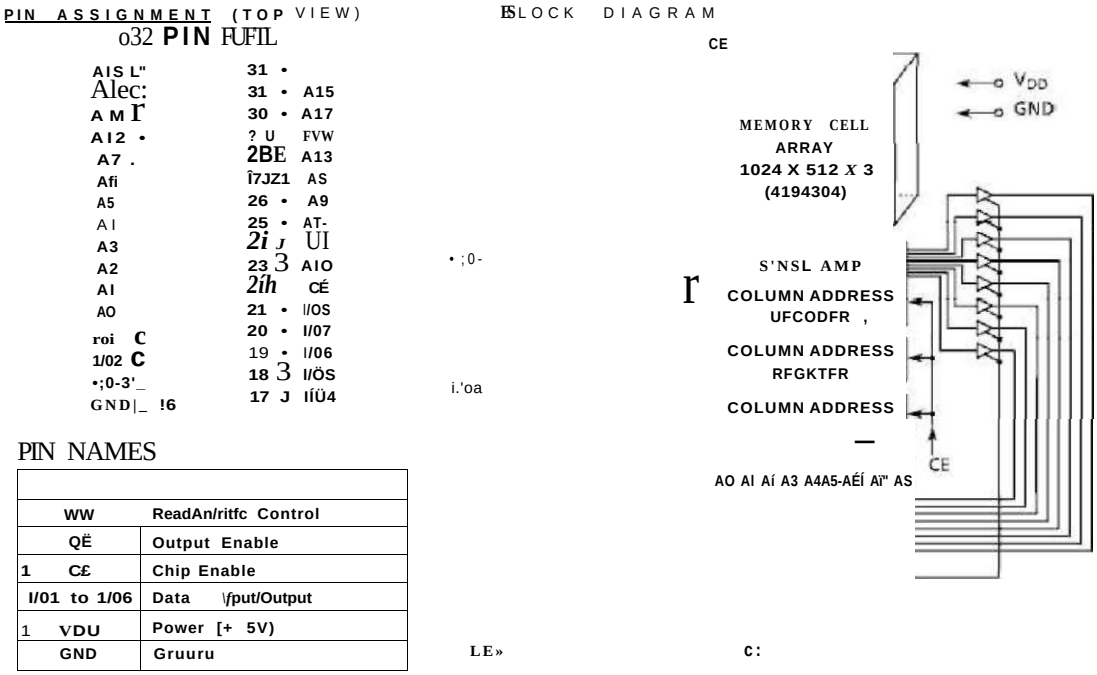



Figura C.15: Pinagem/Diagramaem blocos/Descrição dos pinos.

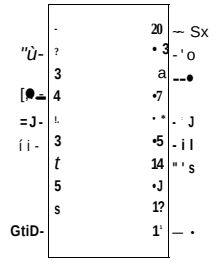
OPL-KAUQN MPPE

| OPERATION MODE | CE | OE | | I/01 to I/OS | A' r 3 |
|---|---|---|---|---|---|
| fteaû | L | 1 | h | | ta o" |
| Write | L | y. | L | DIM | IDDÛ |
| Output Disabled | L | H | H | HiglvZ | IQDÛ |
| | | | X | High-Z | |

**Note: X = don't care. H = logie high. L = lúgii: iöw.**

Figura C.16: Modos de operação da memória.

## Connection Diagram




## Pin Descriptions

| Pin Names | Description |
|---|---|
| Do-D, | Data Inputs |
| LE | Latch Enable Input |
| ÖE | 3-STATE Output Enable Input |
| O0-O7 | 3-STATE Latch Outputs |

## Truth Table

| Inputs | | | Outputs |
|---|---|---|---|
| OE | LE | D | On |
| L | H | H | H |
| L | H | L | L |
| L | L | X | o。 |
| H | X | X | Z |

Figura C.17: Pinagem/Descrição dos pinos/Tabela verdade.

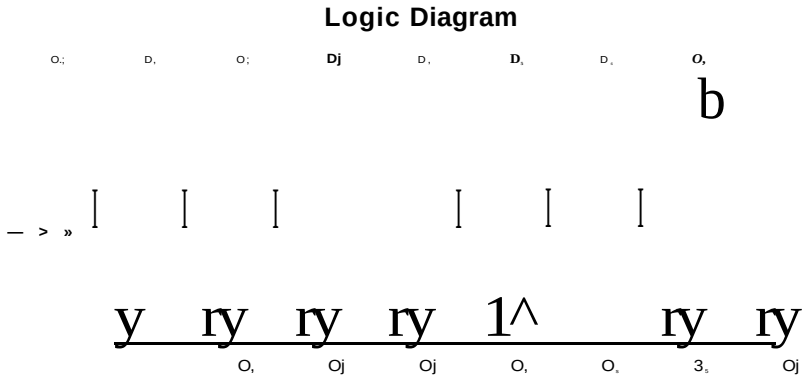



Figura C.18: Diagrama lógico do *latch.*

| PINO | LADO "B" | LADO "A" | PINO | LADO "B" | LADO "A" | PINO | LADO "B" | LADO "A" | PINO | LADO "B" | LADO "A" |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 12 V | TRST# | 25 | 3,3V | AD[24I | 49 | Terra | AD[09I | 71 | AD[59I | AD[58I |
| 2 | TCK | 12V | 26 | C/BE[3]# | IDSEL | 50 | Não conectado | | 72 | AD[57] | Terra |
| 3 | Terra | TMS | 27 | AD[23I | 3,3V | 51 | Não conectado | | 73 | Terra | AD[56I |
| 4 | TODO | TDI | 28 | Terra | AD[22] | 52 | AD[08] | C/BE{0]# | 74 | AD[55] | AD[54] |
| 5 | 5V | 5V | 29 | AD[211 | AD[20I | 53 | AD[07I | 3,3V | 75 | AD[53I | 5V(I/0) |
| 6 | 5V | INTA# | 30 | AD[19] | Terra | 54 | 3,3V | AD[06] | 76 | Terra | AD[52] |
| 7 | INTF3# | INTC# | 31 | 3,3V | AD[181 | 55 | AD[05I | AD[04I | 77 | AD[511 | AD[50I |
| 8 | INTD# | 5V | 32 | AD[171 | AD[1B1 | 56 | AD[03I | Terra | 78 | AD[49I | Terra |
| 9 | PRSNT1# | Reservado | 33 | C/BE[2]# | 3,3V | 57 | Terra | AD[02] | 79 | 5V(I/0) | AD[48] |
| 10 | Reservado | 5V (I/O) | 34 | Terra | FRAME# | 58 | AD[011 | AD[00I | 80 | AD[47I | AD[46I |
| 11 | PRSNT2# | Reservado | 35 | IRDY# | Terra | 59 | 5V (I/O) | 5V(I/0) | 81 | AD[45] | Terra |
| 12 | Terra | Terra | 36 | 3,3V | TRDY# | 60 | ACKB4# | REQ64# | 82 | Terra | AD[44I |
| 13 | Terra | Terra | 37 | DEVSEL# | Terra | 61 | 5V | 5V | 83 | AD[43] | AD[42] |
| 14 | Reservado | Reservado | 38 | Terra | STOP# | 62 | 5V | 5V | 84 | AD[411 | 5V(I/0) |
| 15 | Terra | RST# | 39 | LOCK# | 3,3V | 63 | Reservado | Terra | 85 | Terra | AD[40I |
| 16 | CLK | 5V (I/O) | 40 | PERR# | SDONE | 64 | Terra | C/BE{7I# | 86 | AD[39I | AD[38I |
| 17 | GND | GNT# | 41 | 3,3V | SBO# | 65 | C/BEf6I# | C/BE{5I# | 87 | AD[37I | Terra |
| 18 | REQ# | GND | 42 | SERR# | Terra | 66 | C/BE{4]# | 5V(I/0) | 88 | 5V(I/0) | AD[36] |
| 19 | 5V (I/O) | Reservado | 43 | 3,3V | PAR | 67 | Terra | PAR64 | 89 | AD[35I | AD[34I |
| 20 | AD[31] | AD[30] | 44 | C/BE[1] | AD[15] | 68 | AD[63] | AD[62] | 90 | AD[33] | Terra |
| 21 | AD[32I | 3,3V | 45 | AD[141 | 3,3V | 69 | AD[611 | Terra | 91 | Terra | AD[32I |
| 22 | GND | AD[28] | 46 | GND | AD[13] | 70 | 5V(I/0) | AD[60] | 92 | Reservado | Reservado |
| 23 | AD[27] | AD[26] | 47 | AD[12] | AD[11] | | | | | | |
| 24 | AD[25I | Terra | 48 | AD[101 | GND | | | | | | |

Figura C.19: Descrição dos pinos do barramento PCI.

# Bibliografia


[1] ANTONAKOS, J. L. **The pentium microprocessor.** Editora Prentice-Hall do Brasil Ltda., Rio de Janeiro, 1997. 539p.

[2] ANTÔNIO, V. H. M. **Um sistema de aquisição de imagens baseado em FPGAs.** 8° Simpósio Internacional de Iniciação Científica da Universidade de São Paulo - SIICUSP, novembro 2000.

[3] ARAÜJO, J.G.C. **Sistema computadorizado de apoio à microscopia óptica orientada a objetos.** Dissertação de mestrado, Universidade Federal de Pernambuco, março 1999.

[4] AZEVEDO, M. F. **Operação e avaliação de multímetros analógicos e** ,o

Iniciação Científica da Universidade de São Paulo - SIICUSP, novembro 2000.

[5] BANON, G. J. F.; BARRERA, J. **Bases da morfologia matemática para análise de imagens binárias.** 2a ed. São José dos Campos - SP, 225p. 1998.

[6] BARABANOV, M. **A Linux-based Real Time Operating System.** Master's Thesis, New Mexico Institute of Mining and Technology. 36p. June 1997. URL: http://www.rtlinux.org/ rtlinux/

[7] BARROS, *1'.:* FARINELI, C. A. **Gerente de negócios e gerente técnico e de novos produtos da advantech brasil s/a, automação com pes.** URL: http: / / www.selco.com.br/ciencia/index.htm

[8] BECK, M. et al. **Linux kernels internals.** Addison-Wesley, 2nd, 1998.

[9] BERTIN, M.; JUNIOR, V. P. **Desenvolvimento de sistema de alto desem-** ,o

Simpósio Internacional de Iniciação Científica da Universidade de São Paulo - SIICUSP, novembro 2000.

[10] BERTOCCO, M. et al. **A client-server arquitecture for distributed measurement systems.** IEEE-Instrumentation and Measurement, Technology, Conference, St. Paul, Minnesota, USA, p.18-21, May 1998. URL: http://www.dei.unipd.it/ mat/odl/papers/1998/imtc/remote.04.pdf

[11] BORGES, A. P. **Instrumentação inteligente associada a um laboratório virtual.** Dissertação (Mestrado), Escola Politécnica da Universidade de São Paulo, novembro 2000.

[12] BURNS, A.; WELLINGS, A. **Real-Time Systems and their Programming Languages.** 2a edição, Adison-Wesley, 1997. 575 p.

[13] CAZZARO, F.; RIGOBELLO, M.P.; BINDOLI, A. **Personal computer control of electrochemical detectors utilized for mitochondrial studies.** Computer Methods and Programs in Biomedicine, v.51, p.141-151, November 1996.

[14] CHINAGLIA, E. F. **Caracterização nanoestrutural de filmes finos de Ti, Zr e Hf.** Tese de Doutorado. Instituto de Física da Universidade de São Paulo, novembro 2002.

[15] CRUZ, N. S. **Interface humano para sistemas de aquisição de dados e processamento digital de sinal.** Relatório da disciplina de projeto, Universidade de Coimbra, 42p. Dezembro 2000. URL: http://berta.fis.uc.pt/pdfs/Thesis003.pdf

[16] DtANTONA, G.; FERRERO, A.; OTTOBONI, R. **Improvement of metrological performance for low-cost DSP-based boards with analog interface circuits.** IEEE Transactions on Instrumentation and Measurement, v.48, n.6, p. 1278-1281, December 1999.

[17] FARINES, J. M.; SILVA, J.; OLIVEIRA, R. S. **Sistemas de Tempo Real.** Volume 1. IME-USP, São Paulo, 201p., 2000.

[18] FERNANDES, S. M. M. **Estudo e implementação de mecanismo de tolerância a falhas em software por meio de blocos de recuperação.** Dissertação (Mestrado), Universidade Federal de Pernambuco, dezembro 1995.

[19] FERNANDEZ, R. O.; PEREZ-LISBOA, M. **Laboratório virtual aplicado**

, o

Universidade de São Paulo - SIICUSP, novembro 2000.

[20] FIGUEIREDO, J. O. C.; WEBER, T. S. **Comunicação de grupo confiável em tempo real para o sistema operacional RT-Linux.** 4ª Semana Acadêmica do PPGC, Universidade Federal do Rio Grande do Sul, agosto 1999. URL: http://wwwinf.ufrgs.br/pos/SemanaAcademica/Semana99/joseotavio/joseotavio.html.

[21] GRIMALDI, D.; MARINOV, M. **Distributed measurement systems.** Measurement, Universita Della Calabria, Rende, Italy, v.30, n.4, p.279-287, December 2001.

[22] GRIMONI, J. A. B.; LOPES, V. J. S. **A utilização do Software L A BV I E W no ensino experimental de sistemas de energia elétrica.** Anais do Ibero-American Summit on Engineering Education, Universidade do Vale do Paraíba - São José dos Campos, São Paulo, março 2003.

[23] JUNIOR, M. V.; LIMA, A. **Utilização de analizadores de imagens como recurso para uma nova metodologia de ensino de materiais para engenharia.** 29º Congresso Brasileiro de Ensino de Engenharia Porto Alegre - RS, setembro 2001.

[24] KUNG, C. H. et al. **Fuzzy-based adaptive digital power metering Using a genetic algorithm.** IEEE Transactions on Intrumentation and Measurement, v.47, n.l, p.183-188, February 1998.

[25] KUUSILINNA, K.; HÀMÀLÀINEN, T.; SAARINEN, J. **Field programmable gate array-based P C I interface for a coprocessor system.** Microprocessors and Microsystems, v.22, n.7, p.373-388, January 1999.

[26] MENEZES, H. B. **Um modelo de predição de falhas e garantia de qualidade com monitoramento em tempo real.** Dissertação (Mestrado) - Universidade Federal de Pernambuco, Dezembro 1995.

[27] MONTEIRO, M. P.; DIAS, L. M. M. **A Tecnologia da Informação no Processo Ensino- Aprendizagem de Engenharia.** 29° Congresso Brasileiro de Engenharia Porto Alegre, 2001.

[28] MONTEIRO, M. A. **Introdução à Organização de Computadores.** Livros Técnicos e Científicos Editora S.A., Rio de Janeiro, 2002, p.324-344.

[29] MUKARO, R.; CARELSE, X. F. **A microcontroller-based data acquisition system for solar radiation and environmental monitoring.** IEEE Transactions on Intrumentation and Measurement, v.48, n.6, p. 1232-1238, December 1999.

[30] NAFALSKI, A.; DOAN, T.; GOL, O. **Application of a virtual instrument concept in magnetic measurements.** Journal of Magnetism and Magnetic Materials, v.160, p.154-156, July 1996.

[31] NASCIMENTO, K. S. **Placa ISA para geração de forma de onda arbitrária.** Iniciação Científica sob orientação do Prof. Edval J. P. Sanos, Departamento de Eletrônica - UFPE, 2001.

[32] PACHECO, O. R.; LOUREIRO, ('.: NETO, N. M. **Instrumentação virtual baseada em computador para o ensino da eletrônica.** 1° Simpósio Ibérico de Informática Educativa, Dep. Eletrônica e Telecomunicações - Universidade de Aveiro, Setembro 1999.

[33] **PCI Products Data Book.** AMCC, Applied Micro Circuits Corporation, Section 2: S5920 PCI Target Interface. 145p. URL: http://hsi.web.cern.ch/HSI/s-link/devices/pci-slink/pciprod.pdf

[34] PESSOA, L. F. C. **Uma metodologia para diagnóstico automático da filariose utilizando imagens microscopias digitalizadasa.** Tese (Doutorado)- Universidade Federal de Pernambuco, Setembro 1992.

[35] REGGIANI, L. **A vida pelos bytes.** Info Exame, São Paulo, p.48-58, outubro 2000.

[36] RIGOBELLO, M. P. et al. **Virtual instrumentation for pH measurements in biological systems.** Computer Methods and Programs in Biomedicine, v.60, p.55-64, July 1999.

[37] SILVA, A. H. S. C. **Sistema automático de leitura de amostras em microscópios ópticos.** Dissertação (Mestrado), Universidade Federal de Pernambuco, julho 1992.

[38] SILVA, J. T. L.; SANTOS, E. J. P. **Placa de desenvolvimento PCI para instrumentação virtual utilizando o *chip* AMCC S5920.** 29° Congresso Brasileiro de Ensino de Engenharia, Porto Alegre - RS, setembro 2001.

[39] TANER, A. H.; BRIGNELL, J. E. **Virtual instrumentation and intelligent sensors.** Sensor and Actuators, University of Southampton, n.61, p.427-430, June 1997. URL: http://eprints.ecs.soton.ac.uk/886/

[40] TORÁN, F. et al. **Design of a virtual instrument for water quality monitoring across the internet.** Sensors and Actuators, v.76, p.281-285, June 2001.

[41] TORRES, C. L. **Projeto e implementação de um sistema de automação para um expansocolapsômetro.** Dissertação (Mestrado) - Universidade Federal de Pernambuco, Abril 1999.

[42] TORRES, G. **Hardware Curso Completo.** Axcel Books do Brasil Editora, 4a edição, Rio de Janeiro, 2001. p.297-356.

[43] VALDERRAMA, C. A. et al. **Hardware/Software co-desing: projetando hardware e software concorrentemente.** IME - USP São Paulo, 2000. 246p.

[44] VASCONCELOS, L. **Como montar, configurar e expandir seu PC 486/Pentium.** volume 1, Hardware Básico, Editora LVC, Rio de Janeiro, 1994. 300p.

[45] YANG, Q.; BUTLER, C. **An Object-Oriented Model of Measurement Systems.** IEEE Transactions on Instrumentation and Mensurement, v.47, n.l, **p.104-107.** February 1998.

[46] YODAIKEN, V. **The RTLinux Manifesto.** Department of Computer Science, New Mexico Institute of Technology, November 1999. URL: http://rtlinux.cs.nmt.edu/ rtlinux/papers/rtmanifesto/index.html

[47] ZANETTE, S. L; CARIDE, A. O. **As novas imagens da matéria.** Ciências Hoje, v.27, n.162, julho 2000, p.32-37. URL: http://cienciahoje.uol.com.br/rn

[48] ZELENOVSKY, R.; MENDONÇA, A. **PC: um guia prático de hardware e interfaceamento.** MZ Editora **Ltd**a., 3a edição, Rio de Janeiro - 2002. p. **1**15¬ 425.

[49] URL: http://www.altera.com

[50] URL: http://www.amcc.com

[51] URL: http://atlas.ucpel.tche.br/ feo

[52] URL: http://www.clubedohardware.com.br/pnpl.html

[53] URL: http://www.clubedohardware.com.br/pnp2.html

[54] URL: http://www.coppe.ufrj.br/^notimat/materia/Vol2N2/artigol/micxx_4.htm

[55] URL: http://www.cs.unc.edu/Research/nano/

[56] URL: http://www.di.uminho.pt/amp/coa/nodel3.html

[57] URL: http://www.fiocruz.br/ccs/novidades/out04/simposio_fer.htm

[58] URL: http://www.fc.ul.pt/centros/microscopia/

[59] URL: http://gonzo.sci.man.ac.uk/confocal

[60] URL: http://www.guiadohardware.tcinet.com.br/curso/linux/index.asp

[61] URL: http://www.ifi.unicamp.br/ccjdr/teses/apreseiitacao.php3?filename=IF1262

62] URL: http:// www.imaging.org/

63] URL: http://www.lasid.ufba.br/eventos/wola99/resumos/tarsis.html

64] URL: http://www.lnls.br/principal.asp?idioma=l&conteudo=15&opcaoesq=10

65] URL: http://www.selco.com.br/ciencia/index.htm.

66] URL: http://www.materiais.ufsc.br/lcm/web-MEV/MEV_index.htm

67] URL: http:// cubiccyclonium.com/literature/ds/m7000.pdf

68] URL: http://www.metalmat.ufrj.br/pos_recobrimentos_microscopia.php

69] URL: http://www.nina.ecse.edu/shur/remote/

70] URL: http://www.sbfl.sbfisica.org.br/eventos/ebee/viii/procs/medeirosribeiro.pdf

71] URL: http://www.shimadzu.com.br/analitica/aplicacoes/SPM/SPM.pdf

72] URL: http:// sim.lme.usp.br/labvirtual

73] URL: http://www.spie.org/

74] URL: http://vénus.rdc.puc rio.br/abramo/introducao.html

75] URL: http://vlab.ee.nus.edu.sg/vlab/