JOSÉ EDUARDO COGO CASTANHO

UMA ESTAÇÃO DE TRABALHO PARA VISÃO COMPUTACIONAL

Este exemplar corresponde à redação final da Tese defendida por José Eduardo Cogo Castanho e aprovada pela Comissão Julgadora em 19/2/90

UNIVERSIDADE ESTADUAL DE CAMPINAS

CAMPINAS - 1990

JOSÉ EDUARDO COGO CASTANHO

UMA ESTAÇÃO DE TRABALHO PARA VISÃO COMPUTACIONAL

Dissertação apresentada à Faculdade de Engenharia Elétrica da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica

Orientador
Clésio Luis Tozzi

UNIVERSIDADE ESTADUAL DE CAMPINAS
Campinas - SP.
1990

Aos meus pais

Agradecimentos:

Ao Alemão, por ter enfrentado a difícil missão de fazer o protótipo funcionar.

Aos amigos Zé Raimundo e Lotufo, por terem me iniciado nos segredos dos fios e soquetes (vulgo "hardware").

Ao Sérgio, pelo saudável hábito de criticar a mediocridade humana

Ao Clésio, pelo incentivo, apoio, e principalmente pela paciência e amizade.

Este trabalho recebeu apoio das seguintes entidades:

SID INFORMÁTICA através do projeto ESTRA

CNPQ

UNICAMP

ÍNDICE

RESUMO

É apresentada uma estação de trabalho para visão computacional com a discussão de seus aspectos de projeto e implementação. A imagem é obtida de uma câmara de vídeo P&B. Também são discutidos aspectos de arquitetura para melhoramento do desempenho computacional.

São avaliadas também as necessidades de utilização de processadores especializados e algoritmicamente dedicados para processamento da imagem em tempo real. A fim de ilustrar a discussão, um processador para deteção de bordas em tempo real é apresentado nos seus aspectos de arquitetura e implementação.

ABSTRACT

It is presented an image processing workstation and aspects of design and implementation are discussed. The image is acquired from a B&W TV camera. Architectural aspects for improved performance are also discussed.

The needs of employing algorithmically dedicated processors for real time image processing are evaluated. In order to illustrate the discussion it is presented a real time processor for edge detection in its aspects of architecture and implementation.

CAPÍTULO I

INTRODUÇÃO

I.1 - INTRODUÇÃO

A visão computacional (ou visão artificial, ou ainda processamento de imagem) consiste basicamente em analisar e interpretar uma imagem e é uma área de pesquisa em pleno desenvolvimento [1]. A estrutura primária de informação em visão computacional é uma matriz numérica representando os níveis de intensidade luminosa dos pontos da imagem espacialmente quantizada.

A primeira etapa do processo da visão computacional atua sobre esta matriz obtendo a partir dela partes ou características da imagem que são convertidas em dados simbólicos. Esta etapa é frequentemente divida em processamento de imagem e segmentação da imagem. No primeiro se colocam as operações que processam uma imagem e cujo resultado ainda é uma imagem, entretanto, mais adequada ao processamento subsequente, envolvendo por exemplo eliminação de ruído, melhoria de contraste, etc. A segmentação da imagem consiste em retirar ou identificar em uma imagem partes relevantes para a sua interpretação tais como contornos ou áreas. Todo o processamento envolvido nesta etapa é conhecido como processamento de baixo nível e utilizam-se algoritmos e processos bem determinados.

A segunda etapa da visão computacional, a análise da cena propriamente dita, envolve a obtenção de uma estrutura relacional a partir das informações obtidas na segmentação da imagem. Os nós da estrutura relacional representam as partes obtidas na análise da imagem e os arcos as relações existentes entre as partes. Esta estrutura é usada para obter comparações com estruturas que descrevem objetos previamente conhecidos.

Das etapas do processo da visão, a de baixo nível envolve principalmente processamento numérico, tais como transformações, convoluções, etc. A segunda etapa é essencialmente simbólica e envolve busca de grafos e subgrafos, bem como o tratamento de estruturas relacionais e técnicas de inteligência

artificial.

Assim, o campo da visão computacional abrange um largo espectro de operações e algoritmos já usados amplamente para a solução de vários problemas em áreas distintas. Neste trabalho, os termos visão computacional e processamento de imagem serão empregados indistintamente, sendo mais comum o emprego do termo processamento de imagem por questões associadas ao desenvolvimento histórico do trabalho.

A visão computacional tinha, até recentemente, seu campo de aplicação restrito a algumas poucas aplicações especiais tais como militares, industriais, sensoriamento remoto ou médicas, principalmente devido ao custo do equipamento empregado. Atualmente, com a grande difusão e a contínua redução do custo dos equipamentos de vídeo (câmaras e terminais) bem como dos sistemas computacionais (principalmente memórias e CPU's), tem ocorrido uma grande popularização das pesquisas e aplicações na área.

Uma característica própria da visão computacional é o volume de processamento exigido. Contribuem para isso a grande quantidade de dados presentes desde o início do processo e também o tipo das operações aplicadas sobre eles. Um exemplo comum é a operação de convolução sobre uma imagem qualquer; para cada um dos pontos da imagem são repetidas operações de multiplicação e acumulação envolvendo vários pixels vizinhos ao ponto de interesse. Os processos costumam ser, dessa maneira, extremamente morosos quando são utilizados processadores convencionais, dificultando por exemplo aplicações que necessitem respostas rápidas ou operação em tempo real. Situações nas quais é necessário a obtenção de respostas rápidas são encontradas por exemplo em aplicações industriais, militares, ou visão robótica. Sob esse aspecto é inevitável a busca de soluções para a redução do tempo de resposta dos sistemas de visão, ou pela busca de aumento de eficiência dos algoritmos, ou pelo aumento do desempenho dos sistemas para processamento.

Para as aplicações hoje existentes é difícil encontrar soluções para o aumento de desempenho através de otimização dos algoritmos empregados em virtude "de estes estarem de certa forma definidos" e por já terem sido

trabalhados por longo tempo. A solução para a aceleração do processamento passa a ser, então, a produção de sistemas computacionais com maior capacidade. Os algoritmos para processamento de imagem em baixo nível apresentam características tais como grande repetibilidade e independência de execução (sobre diferentes dados) que os torna especialmente adequados para implementação em arquiteturas paralelas (SIMD, MIND, pipelines, etc). Em particular, as operações efetuadas sobre vetores numéricos, se prestam a execução em máquinas do tipo SIMD. Estas máquinas por sua vez podem ser implementadas na forma de pipelines explorando ao máximo a repetibilidade das operações. Uma grande diversidade de arquiteturas com essa finalidade têm sido propostas e implementadas [2],[3]. A utilização de processadores algoritmicamente orientados também se mostra adequada a obtenção de alto desempenho em aplicações específicas [4],[5]. Com a especialização do processador para a execução de um algoritmo, ou de uma classe de algoritmos, é possível desenvolver cada parte do circuito para executar a sua função de forma ótima. Obviamente a especialização diminui a flexibilidade de aplicação dos processadores, mas permite a obtenção do desempenho desejado, além de uma vantajosa redução dos custos. As soluções para certas aplicações podem ser tratadas academicamente com a utilização de computadores de grande porte ou supercomputadores, mas têm seu emprego prático inviabilizado devido as dimensões e custo do equipamento. Pode-se citar como exemplo a identificação de alvos para fins militares, onde o equipamento deve ser transportado a bordo de um veículo qualquer. Nesses casos, é inevitável a adoção de circuitos dedicados com tamanho reduzido [6]. A popularização da tecnologia VLSI tem provovado um sensível aumento no uso de processadores dedicados, sendo que estes mostram-se, em diversos casos, bastante adequados a integração.

1.2 - OBJETIVOS DO TRABALHO

Este trabalho descreve uma proposta de implementação de uma estação de trabalho para suporte ao desenvolvimento de "hardware" e software para processamento de imagem, incluindo como tópicos principais do projeto um sistema de aquisição e visualização de imagens e de um processador pipeline para extração de bordas de objetos em uma imagem multitonal, passível de implementação em VLSI.

A estação de trabalho aqui descrita foi denominada SPI (Sistema de Processamento de Imagem) e foi concebida com o objetivo de propiciar um ambiente básico para o desenvolvimento de estudos em visão computacional, tanto em "hardware" como em software, bem como servir de base para aplicações. Basicamente o SPI é constituído de elementos que permitem a aquisição e visualização de imagens de vídeo, ou seja, um digitalizador das imagens fornecidas por uma câmara de TV, um terminal de vídeo para a visualização das imagens digitalizadas, um sistema de memória que permite o armazenamento das imagens e de um processador de uso geral que permita o desenvolvimento de programas e aplicações.

Atualmente, com o desenvolvimento da tecnologia e difusão de sistemas de vídeo é, em geral, possível encontrar sistemas comerciais, com as características descritas acima, a um custo relativamente reduzido. Muitos desses sistemas podem ser encontrados na forma de placas de expansão para microcomputadores pessoais, como por exemplo PC's, ou em placas para barramentos padrão VME. Tais sistemas apresentam, contudo, algumas desvantagens com respeito principalmente ao desempenho, devido ao barramento lento ou ao baixo poder computacional dos processadores usados. Alguns fabricantes desenvolveram sistemas equipados com processadores dedicados, em especial com o emprego de processadores digitais de sinais buscando, assim, contornar o baixo desempenho computacional. Ocorre que o aumento de desempenho obtido com essa solução nem sempre é suficiente, ou está restrito a algumas poucas aplicações. O principal motivo de tais limitações é a baixa capacidade de transferência de dados através dos barramentos, e ainda a incapacidade de

satisfazer todos os níveis de processamento, exigidos em visão computacional, com um único tipo de arquitetura.

Baseado na idéia de um equipamento voltado para processamento de imagem, cujas características permitissem alto desempenho associado a um relativo grau de flexibilidade, e que estas contornassem os problemas já mencionados dos equipamentos comerciais, desenvolveu-se o SPI estruturado sobre dois barramentos especialmente empregados para a transferência de dados de imagem, com uma alta banda de passagem permitindo assim, a rápida transferência de grandes quantidades de dados da memória para os processadores especializados acoplados a estes barramentos. As características específicas do barramento concebido permitem a obtenção de um grande desempenho pela utilização de processadores especialmente projetados para as mesmas.

Na arquitetura proposta para o SPI um microcomputador é responsável pelo controle dos processos no sistema, inclusive aqueles executados pelos processadores especializados, pelo processamento de alto nível (processamento simbólico), pela interface com o usuário e operações de suporte tais como memória de massa. Adotou-se um microcomputador tipo PC devido a sua grande difusão no mercado e a grande quantidade de ferramentas para desenvolvimento de software. A interface entre o SPI e o microcomputador foi feita , entretanto, intencionalmente simples para permitir a migração segura para qualquer outro tipo de barramento.

Com o intuito de aumentar o desempenho global do sistema, a arquitetura foi desenvolvida de modo a receber, sem qualquer alteração, um processador ou um conjunto de processadores especializados na execução das etapas de baixo nível do processo de visão computacional. Estes processadores são essenciais quando se deseja atender aplicações que demandem respostas em tempo real, como por exemplo algumas aplicações industriais em linhas de montagem.

Para exemplificar esta classe de processadores escolheu-se para projeto um processador para executar a função de deteção de bordas. A função deteção de bordas é, por sua vez, baseada no emprego de nucleos de convolução de dimensão 3x3, bastante usados em inúmeras outras aplicações de processamento de imagem. Embora visando uma aplicação específica, é possivel derivar a

partir deste projeto, com pequenas modificações, processadores para diversas classes de algoritmos. Neste trabalho é discutido o projeto do processador para deteção de bordas no seu aspecto arquitetural e sua integração é tema central de outro trabalho desenvolvido em consonância com este [7].

A obtenção do desempenho especificado só é possível pelo emprego de paralelismo, em especial de pipeline de operações.

O projeto do processador buscou explorar as caracaterísticas especificas do do barramento de imagem do SPI a fim de obter um alto desempenho. Entre outras vantagens obtidas dessa abordagem pode-se citar a total independência para acesso à memória para obtenção de dados e armazenamento de resultados, resultando num fluxo ótimo de processamento.

1.3 - Organização do Trabalho

No capítulo 2 é apresentada uma breve discussão genérica dos elementos de uma estação de trabalho voltada para o processamento de imagem e a estrutura de interligação desses elementos. São apresentados os fundamentos dos terminais de vídeo e "frame grabbers" e um estudo das forma de organização da memória de imagem levando em conta as características do processamento de imagem.

No capítulo 3 é realizada uma descrição a nível funcional do projeto do Sistema para Processamento de Imagem (SPI). São tecidas considerações a respeito das opções de implementação da estrutura e tecnologia empregadas. Em especial, discute-se a adoção de um barramento de imagem com banda de passagem larga como forma de evitar disputas para acesso à memória e eventualmente possibilitar a adoção de processamento paralelo.

No capítulo 4 é apresentado uma visão da estrutura algoritmica da aplicação pretendida, ou seja, a deteção de bordas, bem como uma avaliação

das necessidades da utilização de "hardware" específico. Na sequência é realizado um estudo das possíveis estruturas do processador dedicado, procurando analisar as relações existentes entre custo-flexibilidade.

No capítulo 5 é feito um estudo das possibilidades de implementação prática de cada estágio do processador e como estas afetam o projeto.

No capítulo 6 são tecidos comentários a respeito das vantagens e desvantagens do sistema proposto. São relizados, também alguns comentários a respeito das dificuldades de implementação relativas a infraestrutura e tecnologia disponíveis. Por fim, avalia-se as possibilidades de continuidade de desenvolvimento do sistema e direções a serem tomadas.

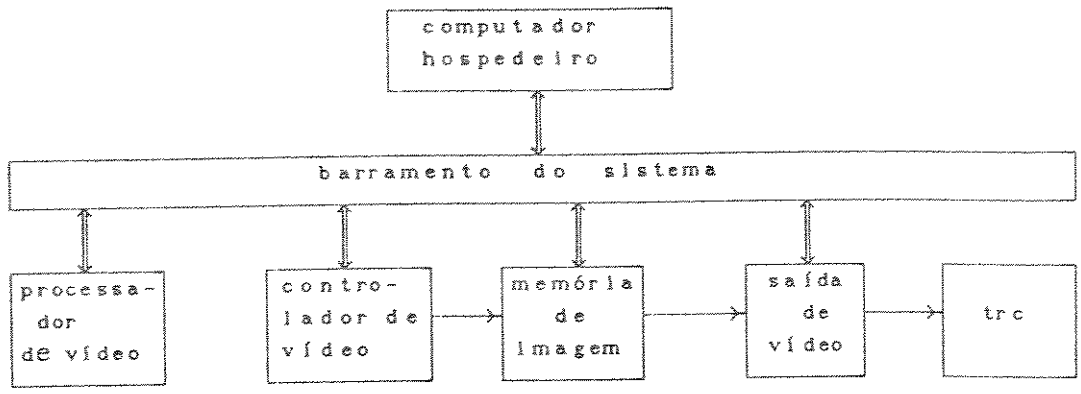
Capítulo II

Uma estação de trabalho para processamento de imagem.

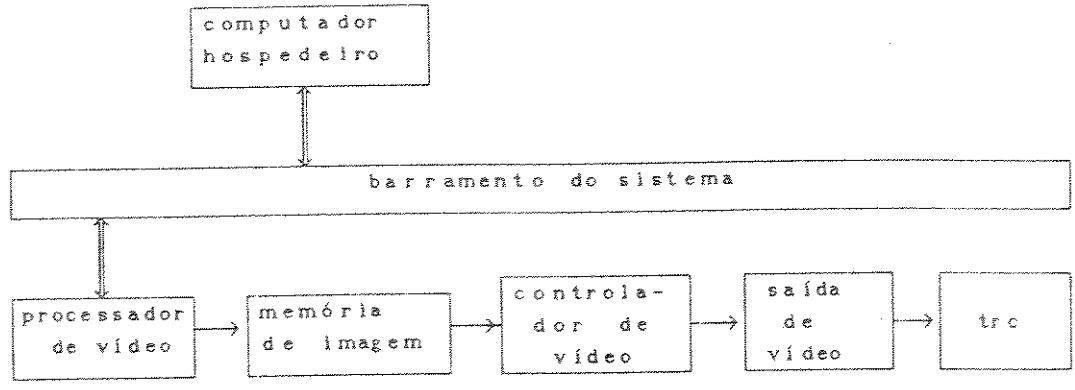
II.1 - INTRODUÇÃO

Uma estação de trabalho para aplicações em processamento de imagem é, sob a maioria dos aspectos, bastante similar a uma dedicada ao processamento gráfico, diferindo, basicamente, pela necessidade de um dispositivo de aquisição de imagens no aspecto de "hardware" e pela disponibilidade de programas para manipulação das imagens no aspecto do "software". Por sua vez, o "software" necessário para manipulação das imagens muito se assemelha, em seu nível primitivo (primitivas básicas como por exemplo "pixel-op", "windows", "storage", "retrieval", etc), com o "software" para manipulação gráfica. Em síntese, uma estação para processamento de imagem opera, em geral, sobre imagems reais e uma estação gráfica opera sobre imagens sintetizadas. Por dispositivos de aquisição de imagens são entendidos todos os dispositivos que obtêm imagens através de sensores de luz, tais como digitalizadores de vídeo, "scanners", etc, cujo sinal de saída é colocado na forma digital para posterior armazenamento na memória do sistema de processamento.

Na literatura especializada é frequente encontrar-se referência a duas formas mais comuns de interligação dos elementos componentes de uma estação gráfica. Os diagramas de blocos da fig.II.1 ilustram as duas possibilidades ao mesmo tempo que representa os principais elementos componentes do sistema. Esta concepção de estação gráfica inclui como elementos principais: um processador de propósito geral (processador hospedeiro), voltado ao processamento de alto nível das aplicações; um processador especializado, voltado ao processamento de rotinas gráficas básicas; e um sistema de visualização de imagem (designado como terminal gráfico). O processador especializado, em geral denominado processador gráfico (ou "pixel engine") deve ser capaz de realizar as rotinas mais comuns em computação gráfica tais como por exemplo a geração de primitivas gráficas (ponto, círculo, reta, etc) ou operações de movimento de blocos de memória, com a finalidade de acelerar o processamento e manter o processador hospedeiro livre para execução de aplicações. O processador gráfico pode ser baseado em um processador comum, especialmente programado, ou em um "hardware" especial projetado para este fim.



a) sistema com memória com duas portas, para o controlador de vídeo.

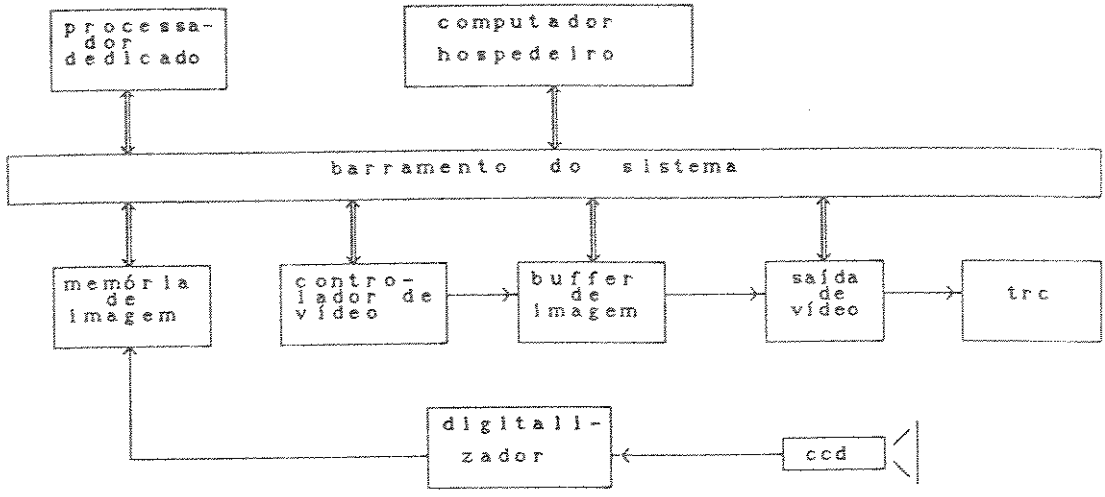


b) sistema de vídeo no qual a memória de vídeo está ligada somente ao controlador de vídeo

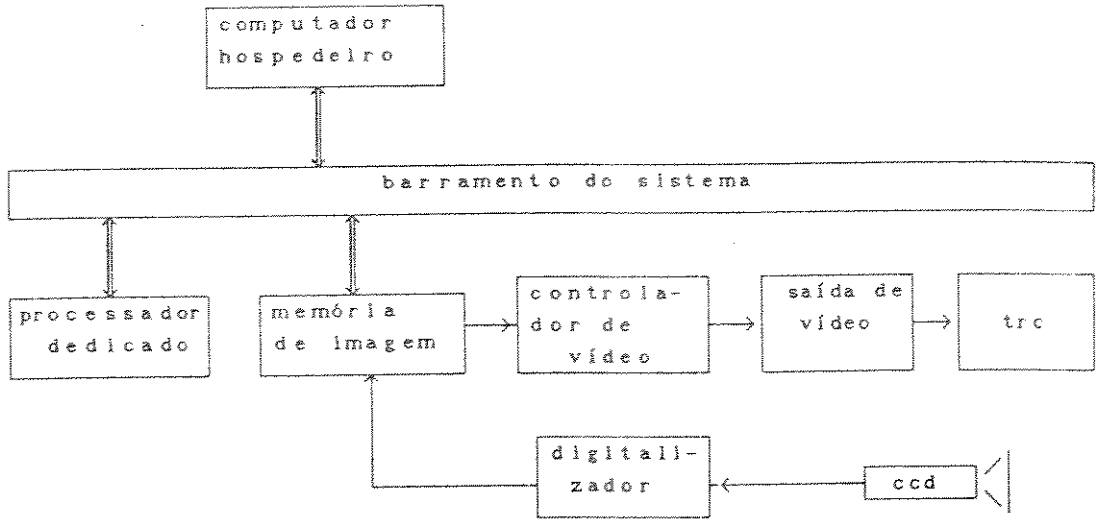
fig.II.1 - diagrama em blocos dos esquemas mais comuns de interligação de elementos em um sistema de vídeo.

Por outro lado, não existe, na literatura, um consenso a respeito da estrutura de uma estação para processamento de imagem. Este fato ocorre principalmente pela pequena disseminação destas estações devido ao alto custo que apresentavam até recentemente e que de maneira geral conduzia sempre a uma solução específica para cada problema. Entretanto, dado a semelhança entre os componentes básicos de uma estação de trabalho para aplicações gráficas e uma outra para processamento de imagem, é natural basear-se na estrutura apresentada acima para derivar a estrutura básica, isto é, os requisitos mínimos exigidos para uma estação empregada em processamento de imagem.

Em processamento de imagem os algoritmos mais básicos são normalmente simples, porém o tratamento completo da imagem implica na execução destes algoritmos milhares de vezes. Por exemplo, para uma imagem de 512x 512 pontos, com múltiplos níveis de intensidades luminosa, o cáculo do histograma implica no tratamento de 262.144 pontos. Deste modo o desempenho computacional é muito importante pois os algoritmos são extremamente custosos do ponto de vista computacional. O emprego de processadores dedicados soluciona em grande parte os problemas de velocidade de processamento em determinadas aplicações. Por outro lado, não é, em geral, possível cobrir todas as etapas do processamento de imagem com um único processador dedicado em virtude da grande diversidade de algoritmos envolvidos em cada uma delas. É usual, então, que o processador dedicado realize algumas tarefas básicas e comuns a vários algoritmos, ao passo que o processador hospedeiro realize as tarefas menos frequentes e particulares a cada um deles. Os algoritmos executados pelos processadores dedicados são usualmente de caráter numérico e referidos como de baixo nível ao passo que os algoritmos executados pelo processador hospedeiro são de



a) sistema com memória com duas portas, para o controlador de vídeo.



b) sistema de vídeo no qual a memória de vídeo está ligada somente ao controlador de vídeo

fig.II.2 - Possíveis estruturas de uma estação de trabalho para processamento de imagem derivadas da estrutura da fig.II.1a.

caráter simbólico e referidos como processamento de alto nível. Assim, a alternativa apresentada na fig.II.1a parece mais indicada ao processamento de imagens uma vez que permite o acesso direto do processador hospedeiro à memória de imagem possibilitando um desempenho melhor e maior flexibilidade. Na fig.II.2 são apresentados dois possíveis esquemas de interligação para os componentes de uma estação de processamento de imagem derivados da fig.II.1a.

Na fig.II.2a o esquema de ligação inclui uma memória de imagem exclusiva para o sistema de aquisição. Caso a imagem deva ser mostrada no terminal de vídeo, ela deve ser transferida de um "buffer" para o outro. Essa operação consome tempo e dificulta a visualização em tempo real, importante para aplicações que exigem monitoramento.

Considerando o esquema apresentado na fig.II.2b mais conveniente, uma vez que o mesmo permite a monitoração da imagem adquirida e através de "hardware" e "software" adequados a sobreposição de imagens reais e sintéticas, é conveniente para a compreensão do projeto que se façam algumas discussões iniciais sobre técnicas de geração de imagens em terminais do tipo "raster", já bastante estudadas e difundidas, para em seguida verificar-se a integração do dispositivo digitalizador, bem como os processadores dedicados as funções escolhidas.

II.2 - O PRINCÍPIO DO TERMINAL "RASTER"

O terminal "raster" tem seu princípio baseado no emprego de um tubo de raios catódicos para a geração de imagens, da mesma forma que a televisão comercial. O feixe de elétrons do TRC é deslocado rapidamente da esquerda para a direita e lentamente de cima para baixo. A composição destes movimentos faz com que o feixe de elétrons, ao atingir a tela de fósforo do TRC, descreva uma trajetória que corresponde a um conjunto de linhas horizontais dispostas muito próximas umas das outras, preenchendo quase toda a área da tela. No instante em que o feixe de elétrons atinge a camada de fósforo da tela, esta passa a emitir luz com uma intensidade que é proporcional a intensidade do feixe. A modulação da intensidade do feixe de elétrons causa áreas de maior ou de menor luminosidade nas tela do TRC, permitindo a formação de imagens. No terminal "raster", ao contrário da televisão comercial, a modulação do feixe de elétrons é quantizada, levando a formação de uma imagem composta de pontos, ou em outras palavras, de um reticulado. A intensidade e/ou a côr de cada um desses pontos é armazenada em uma memória, geralmente do tipo RAM semicondutora.

Um circuito controlador se encarrega de fornecer os sinais para a sincronização dos movimentos de varredura, além de buscar a informação armazenada na memória de imagem para modular a intensidade do feixe de elétrons. Os movimentos horizontal e vertical são usualmente referidos como varredura horizontal e vertical respectivamente. A volta do feixe de elétrons ao término da varredura de um linha para início da varredura de outra não é instantânea e é denominada retraço horizontal. Durante o retraço horizontal o feixe de elétrons é apagado. Da mesma forma, a volta do feixe de elétrons para início da varredura de um novo quadro é denominado retraço vertical e o feixe também mantém-se apagado. É importante ressaltar que a duração do retraço vertical equivale, em geral, o tempo de varredura de várias linhas.

II.2.1 - A RESOLUÇÃO DA IMAGEM E A TEMPORIZAÇÃO DA VARREDURA

A definição ou a qualidade da imagem gerada é função da quantidade de pontos pelos quais ela é formada. Obviamente, quanto maior a quantidade de pontos, tanto na horizontal quanto na vertical, maior a definição da imagem. Assim, é desejável que a imagem seja formada pelo maior número de pontos possíveis. Entretanto, a quantidade de pontos está sujeita a uma série de restrições de temporização para a implementação e para o funcionamento do terminal. A resolução da imagem também acarreta consequências na organização e tipo da memória utilizada para armazenamento da imagem. À medida que a resolução aumenta, diminui o tempo de acesso para a obtenção dos dados na memória, sendo necessário a utilização de componentes mais rápidos ou então a adoção de estruturas de circuito mais complexas para a obtenção dos requisitos de temporização com um consequente aumento de custo.

II.2.1.1 - Tempo de geração de imagem ou de regeneração

A camada de fósforo ao ser atingida pelo feixe de elétrons emite luz durante um período de tempo limitado após o qual cessa a emissão. Assim, para manter a imagem visível, esta deve ser regenerada constantemente através de repetidas varreduras do feixe de elétrons. A taxa de regeneração da imagem situa-se na faixa de de 30 a 60 imagens por segundo dependendo da persistência do fósforo. A taxa de regeneração das imagens está associada, também, a capacidade da visão humana de captá-la e retê-la. Os valores próximos de 30Hz permitem a retenção, mas provocam em determinados tipos de cena, como as que são formadas por uma grande quantidade de linhas finas, uma incômoda sensação de cintilamento. Com uma frequência de 60Hz o cintilamento desaparece, mas a banda de passagem dos tubos de raios catódicos deve ser maior para uma mesma resolução da imagem, encarecendo o sistema. Um sistema que procura minimizar a influência do cintilamento sem entretanto aumentar a frequência de regeneração da imagem foi proposto nos primórdios da televisão comercial. Nesse sistema de varredura a imagem, ou quadro, é decomposta em dois campos, um com as linhas

pares e outro com as ímpares, que são mostradas alternadamente e por isso é denominada de varredura entrelaçada. A utilização desse expediente permite que a frequência de varredura horizontal permaneça baixa para uma determinada quantidade fixa de linhas. Por outro lado, cada um dos campos é regenerado com a metade da frequência de varredura vertical, fazendo com que o cintilamento esteja ainda presente mas não com a mesma intensidade. É importante ressaltar que, no caso da televisão comercial, o cintilamento provocado pelo entrelaçamento, em virtude do tipo de imagens presente, é praticamente insignificante, mas no caso de terminais gráficos o cintilamento é muito mais perceptível. Para os terminais empregados em processamento de imagem o cintilamento também possui uma influência menor.

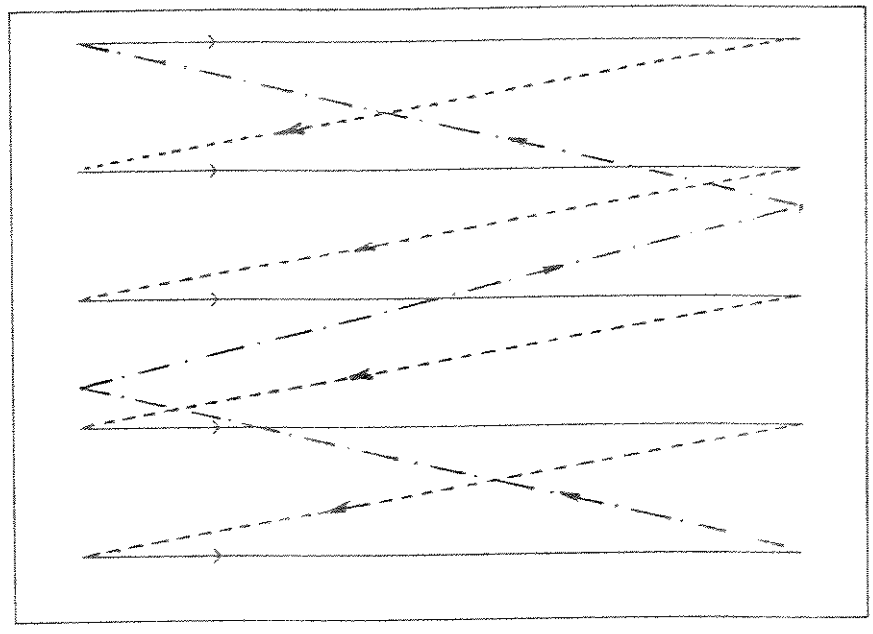


fig.II.3 - Representação da varredura do feixe de eletrons na tela.

No sistema não entrelaçado, toda as linhas são apresentadas em ordem do topo até a base. Uma especificação bastante comun para a frequência de regeneração da imagem é 60Hz, isto é, a imagem é completamente "desenhada" na tela 60 vezes em um segundo. No modo entrelaçado a taxa de regeneração da imagem é usualmente 30Hz, embora um novo campo seja exibido a cada 1/60 de segundo.

II.2.1.2 - Resolução vertical

As frequências de varredura horizontal e vertical determinam a quantidade de linhas a ser mostrada na tela e consequentemente, a resolução vertical da imagem. Para uma dada frequência de varredura horizontal, a resolução vertical é maior, isto é, a quantidade de linhas é maior quanto mais longo for o tempo de varredura vertical ou de regeneração da tela. Por outro lado, para uma dada frequência de varredura vertical, o número de linhas é maior quanto maior é a frequência de varredura horizontal.

Nos padrões de TV a quantidade de linhas situa-se entre 525 e 612 linhas. Terminais de vídeo para computação gráfica apresentam, entretanto, uma larga faixa de opções, sendo possível encontrar terminais com mais de 1024 linhas. Terminais de vídeo para processamento de imagem são encontrados, também com diferentes resoluções horizontais, mas um valor bastante comum é o de 512 linhas que se aproxima da resolução dos sistemas de televisão comerciais.

II.2.1.3 - Resolução horizontal

A resolução horizontal é determinada pela frequência de varredura horizontal e pelo tempo de duração do pixel. Para uma dada frequência de varredura horizontal, a resolução horizontal é maior, isto é, existe uma maior quantidade de pixels por linha quanto menor o tempo de duração do pixel. Por outro lado, para um dado tempo de geração do pixel, a resolução horizontal é

maior quanto menor for a frequência de varredura horizontal.

II.2.1.4 - Tempo de geração de um pixel

É importante conhecer o tempo de geração de um pixel a fim de determinar a frequência com que um novo pixel deve ser buscado na memória de imagem para ser enviado a saída de vídeo. A frequência de acesso à memória, ou a frequência de geração de pixels vai determinar o tipo de tecnologia a ser usada, bem como a estrutura do sistema de vídeo e da memória.

O tempo de pixel pode ser determinado a partir do tempo de regeneração, tempos de retraço vertical e horizontal, número de linhas mostradas por imagem e número de pixels mostrados por linha.

A fig.II.4 mostra como o tempo total de um quadro é distribuído entre os tempos de retraço vertical e horizontal e tempo de linha ativa.

Para se determinar a duração de cada pixel, pode-se seguir a equação II.1 dada abaixo onde os valores de retraço vertical e horizontal são geralmente valores nominais para cada monitor. Na tabela II.1 são fornecidos valores típicos para diferentes resoluções de tela.

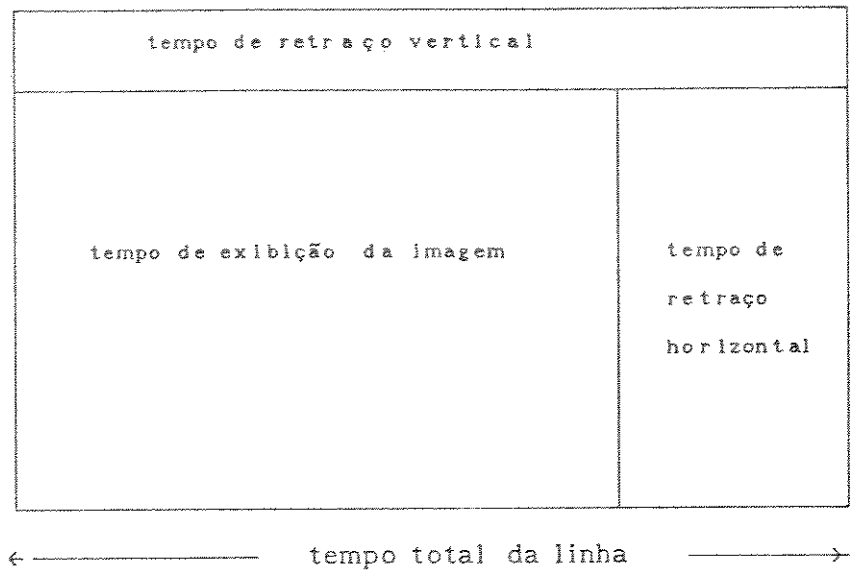


fig.II.4 - Distribuição dos tempos de traço e retraços em relação a composição de um quadro.

equação II.1

$$\left[\frac{1}{\text{taxa de regeneração}} - \text{retraço vertical}\right] \Big/ \text{linhas visíveis por tela} - \text{retraço horizontal} \Big/ \text{pixels visíveis por linha} = \text{tempo de pixel}$$

obs:

1 - a norma RS170-A especifica o tempo de linha como sendo 63,556us.

2 - Para sistemas entrelaçados, usar duas vezes o tempo de retraço vertical. RS170-A especifica o retraço vertical de 20 linhas por campo ou 40 linhas por tela.

TABELA II.1

área visível pixelsXlinhas	fr	e?	trv(us)	trh(us)	ttl(us)	tp(ns)
512x512	30	sim	1203	10,9	60,4	96.7
512x485	30	sim	1271	10,9	63,56	102,8
1024x1024	30	sim	1250	7	30,11	22,57
512x512	60	não	1250	7	30,11	45,14
1024x1024	60	não	600	4	15,69	11,42

fr = frequência de regeneração

e? = entrelaçado?

trv = tempo de retraço vertical

trh = tempo de retraço horizontal

ttl = tempo total da linha horizontal

tp = tempo de um pixel

*obs: os tempos são especificados em microsegundos com exceção do tempo de pixel que é dado em nanosegundos

II.2.2 - O CONTROLADOR DE VÍDEO

De um modo geral, o circuito controlador de vídeo tem as funções de controlar a transferência dos dados armazenados na memória para o monitor de vídeo, juntamente com a geração dos sinais de sincronismo. Assim, o circuito controlador deve gerar os endereços para acesso as posições de memória que contêm a intensidade de cada pixel, bem como os sinais de comando para a transferência dos dados. Alguns circuitos controladores podem executar funções que envolvem a manipulação dos endereços da memória tais como "pan", "scroll" e janelas. Os dados lidos na memória são enviados a um circuito de saída de vídeo que é constituído, normalmente, por "look-up-tables" e conversores digital/analógico.

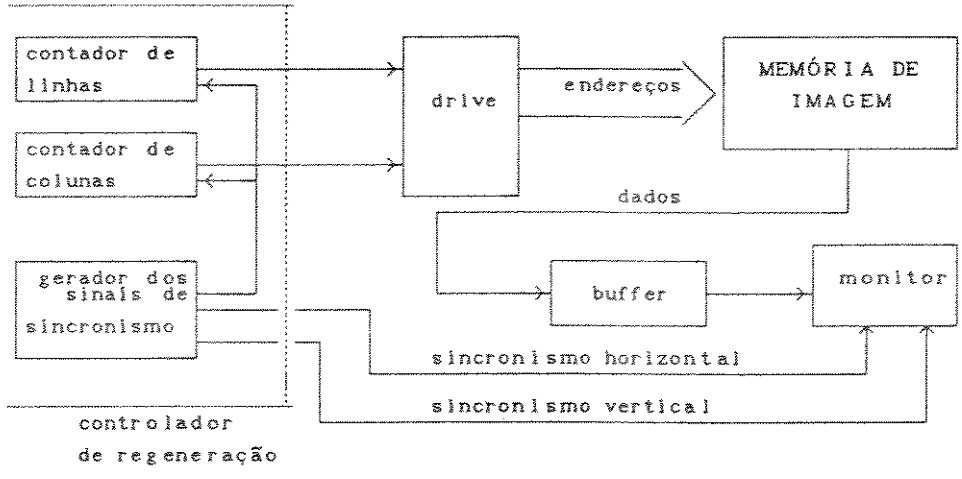


fig.II.5 - Diagrama de blocos básico de um controlador de vídeo.

Existem dois grandes obstáculos a serem superados na implementação dessas funções, ambos relacionados à implementação da memória. O primeiro relaciona-se a taxa de acesso à memória para a obtenção dos pixels no tempo correto. O segundo refere-se a resolução do conflito entre o controlador e o processador de vídeo para acesso ao barramento da memória de vídeo.

A taxa de acesso à memória é inversamente proporcional ao tempo de duração de cada pixel, isto é, quanto maior a resolução desejada, maior a frequência de acesso à memória. Dessa forma, as memórias devem permitir acessos aos dados de forma bastante rápida. Para terminais de alta resolução, entretanto, é difícil a obtenção de memórias que atendam aos requisistos de velocidade exigidos, sendo necessário encontrar alternativas para o acesso direto do controlador a cada pixel individualmente.

O outro fator que contribui para agravar o problema de tempo de acesso à memória é a necessidade de atualização ou consulta dos dados contidos na memória de imagem.

Existe um grande variedade de controladores de vídeo, sendo que há uma tendência dos fabricantes em incorporar cada vez mais recursos de processamento aos controladores. Na realidade, os atuais controladores podem ser melhor definidos como processadores de vídeo com recursos para a regeneração da imagem. Uma descrição bastante detalhada de diversos controladores de vídeo, além de comparações e aplicacões dos mesmos pode ser encontrada em Kane [8] e Berardi [9].

II.3. - A INTERFACE CÂMARA - COMPUTADOR.

Pode-se divisar diversas maneiras de viabilizar a interface de uma câmara de vídeo com um computador. Basicamente, a tarefa da interface consiste na conversão de um sinal analógico proveniente de uma câmara de vídeo para um sinal digital que é armazenado na memória do computador, na sequência correta de formação da imagem "raster". Não serão abordadas, aqui, todas as possibilidades de realização da interface, mas sim os princípios gerais que a norteiam, bem como os principais problemas. Cabe ressaltar que a interface para a aquisição de imagens de vídeo digitalizadas é comumentemente referida como "frame grabber", em publicações especializadas e por fabricantes, distinguindo essa denominação dos demais elementos de aquisição de imagem, cuja fonte não seja um sinal de vídeo.

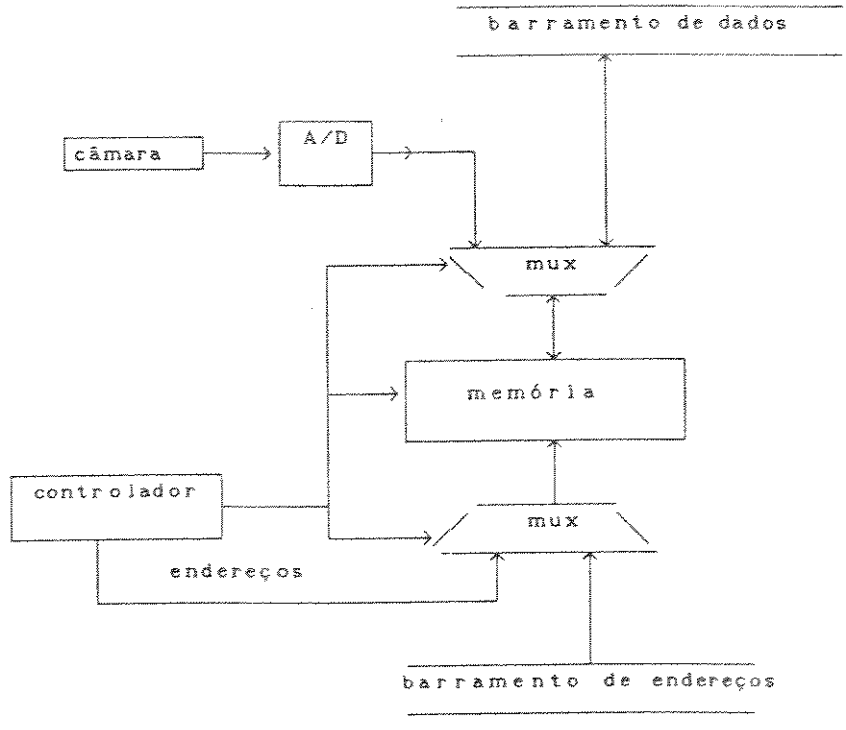


fig.II.6 - Diagrama de blocos de uma interface para aquisição de imagens.

O sinal de vídeo fornecido por uma câmara é, essencialmente, o mesmo que é enviado pelo controlador para o terminal de vídeo, e já discutido nos itens anteriores. É composto de sinais de sincronismo, que delimitam o fim de uma linha de varredura e final de uma tela. Entre os sinais de sincronismo horizontal se encontra a informação de vídeo propriamente dita e que deve ser digitalizada. A fig.II.7 ilustra o aspecto de um sinal fornecido por uma câmara de vídeo, considerando o tempo entre dois sinais de sincronismo horizontal.

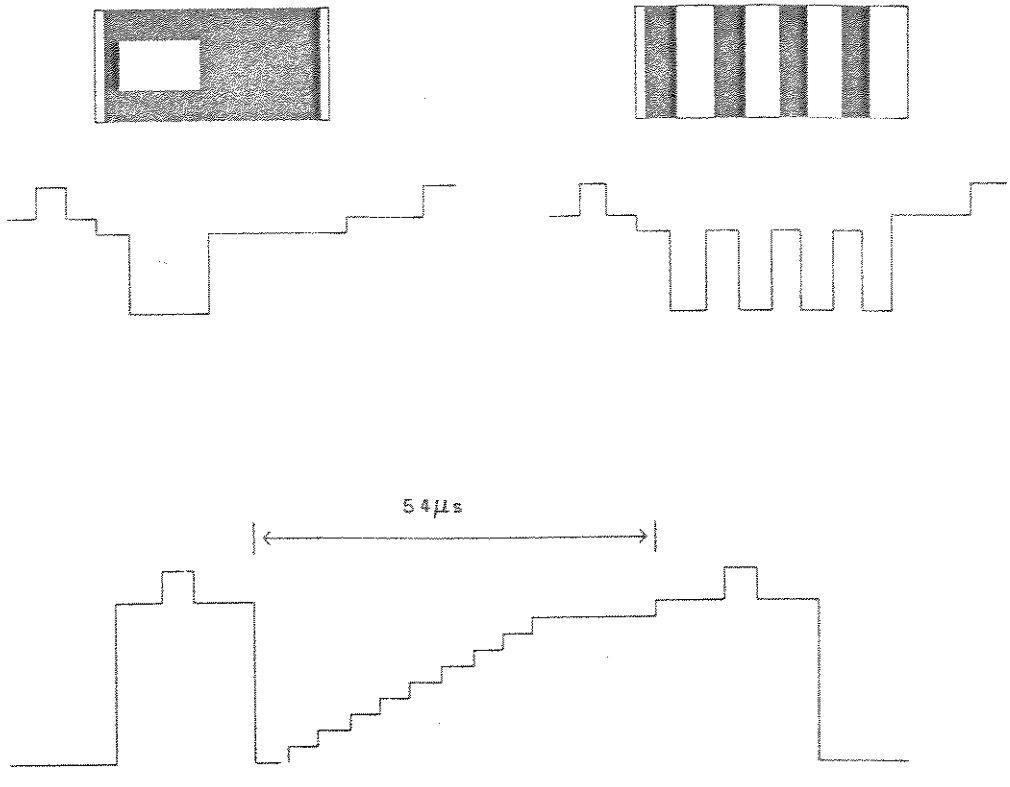


fig.II.7 - sinal de vídeo ilustrando o intervalo entre dois pulsos de sincronismo horizontal que contém a informação visual, com imagens de teste e o sinal correspondente.

II.3.1 - SINCRONIZAÇÃO E APAGAMENTO

É possível fazer uma analogia entre um aparelho receptor de TV e um digitalizador. Ambos devem receber um sinal de vídeo e interpretá-lo, com a diferença básica que o aparelho receptor posiciona a informação recebida na tela de fóforo ao passo que o digitalizador posiciona a informação na memória de imagem.

Dado que os sinais de vídeo provenientes da câmara se sucedem initerruptamente, o digitalizador deve receber os sinais de sincronismo para identificar o término de cada uma das linhas e assim posicionar corretamente na memória cada elemento da imagem. Da mesma forma é necessário informar ao digitalizador do término de um quadro ou campo de exploração vertical para que ele possa posicionar cada linha no local correto da memória. Este processo é levado a cabo pelos sinais de sincronismo horizontal (para a identificação do término/início de linha) e vertical (para a identificação do término/início de quadro ou campo).

O ciclo de varredura horizontal dura em sistemas comerciais de TV por volta de 64μs. Excluído o tempo ocupado pelo sinal de sincronismo e o tempo de retorno do feixe sobram aproximadamente 54 μs de sinal de vídeo com informação válida. No restante do tempo, o sinal de vídeo não contém qualquer informação de imagem e durante este período não deve ser realizada a conversão e armazenamento do sinal na memória.

O reduzido tempo da duração de cada linha é responsável pelas maiores dificuldades técnicas a para implementação de uma interface. Uma das dificuldades refere-se a resolução horizontal e taxa de conversão e a outra refere-se a taxa de transferência dos dados para a memória. Estes dois problemas serão abordados nas próximas secções.

II.3.2 - Conversão A/D

A conversão analógica/digital possui dois parâmentros de avaliação: a taxa de amostragem e a resolução ou quantização do valor amostrado. A taxa de amostragem é responsável pela resolução espacial (horizontal) da imagem adquirida ao passo que a quantização revela a precisão da representação da intensidade luminosa de cada ponto amostrado. Para se obter uma resolução horizontal alta é necessário que a taxa de conversão A/D seja também alta. Por exemplo, no caso de uma imagem com resolução horizontal de 512 pontos por linha, cada pixel terá uma duração de aproximadamente 100ns, ou, uma taxa de conversão de aproximadamente 10MHz. Os conversores A/D que operam nessa faixa de taxa de conversão devem ser do tipo "flash". Portanto, são necessários componentes caros e de difícil aquisição no mercado. Além disso, a alta taxa de operação aumenta a presença de ruído no resultado.

II.3.3 - Taxa de transferência

A taxa de transferência de dados do digitalizador para a memória é igual a taxa de conversão A/D e portanto igualmente alta, acarretando também uma série de problemas na implementação e organização da memória. Os problemas e soluções encontrados são semelhantes aos presentes no circuito controlador de vídeo e resultam principalmente da dificuldade de obtenção de componentes que satisfaçam as especificações de tempo de acesso (ou ciclo de acesso) e da necessidade de partilhar o barramento da memória com outros elementos do sistema (processadores, co-processadores, dispositivos de DMA, ou outros controladores).

Algumas implementações de baixo custo contornam o problema da velocidade de aquisição de dados através do uso de um artifício que permite a aquisição

de uma única imagem pela intercalação de amostragens lentas em diversos quadros consecutivos, aumentando, assim o período de amostragem total. Cada uma das amostragens realizadas sobre um determinado quadro é espacialmente deslocada em relação as amostragens dos outros quadros. Dessa forma, combinando o resultado da amostragem de diversas telas de baixa resolução em apenas uma, obtém-se uma imagem de alta resolução. Este método tem o incoveniente de só ser útil na obtenção de imagens estáticas, ou com movimentos muito pequenos, sem mudanças significativas de uma tela para outra. Existe a possibilidade de realizar a obtenção de imagens não estáticas através deste método e posteriormente restaurar a imagem corrompida pelo movimento através do processamento adequado [10]. Entretanto, o custo computacional exigido pela operação não justifica o procedimento, a não ser por razões econômicas incontornáveis.

A solução típica para o problema, entretanto, consiste em paralelizar os dados obtidos do conversor A/D através de um registrador de deslocamento com entrada serial e saída paralela. A operação realizada é exatamente a inversa da realizada no controlador de regeneração. Após um conjunto de dados, ou pixels, ser paralelizado, ele é enviado a memória a uma taxa menor que a inicial, embora em palavras maiores. O processo pode ser melhor entendido através do esquema da fig.II.8.

A solução apresentada acima pode ser considerada padrão, embora possam existir pequenas variações na forma de implementação, principalmente relacionadas a tecnologia dos componenentes usados. O problema da taxa de tramsferência pode ser analisado também do ponto de vista de organização da memória e da influência que esta tem sobre os diversos elementos do sistema e por isso voltará a ser abordado na secção relacionada a organização da memória.

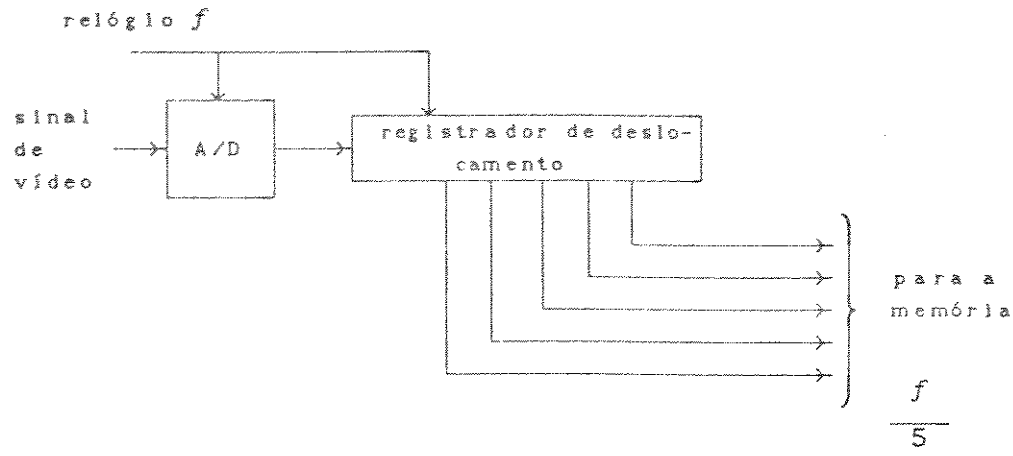


fig.II.8 - Esquema de paralelização do sinal de vídeo em um digitalizador para redução da taxa de transferência de dados para a memória. O sinal de vídeo é convertido e inserido no registrador de deslocamento a uma frequência f e enviado a memória a uma frequência $f/5$.

II.3.4 - FUNCIONAMENTO DO "FRAME GRABBER"

O princípio básico de operação e construção de uma interface para aquisição de imagens de uma câmara de vídeo pode ser entendindo a partir diagrama de blocos simplificado da fig.II.9.

A partir da deteção do pulso de sincronismo vertical inicia-se a amostragem e conversão do sinal de vídeo com uma taxa dada pelo relógio do sistema e cuja valor foi previamente calculado para a resolução desejada. A cada período de relógio um contador é incrementado e sua saída é usada para endereçar uma posição de memória para receber o pixel amostrado naquele período de relógio. A chegada de um pulso de sincronismo horizontal ou vertical inibe a contagem a fim de evitar que se armazenem dados não válidos

na memória durante o retraço. O divisor por quatro habilita a aquisição apenas durante o período de dois campos (um quadro), sendo que no terceiro pulso de sincronismo vertical, um sinal é enviado a um circuito de controle que habilita o acesso da memória pelo processador hospedeiro. Uma outra alternativa poderia fazer uso da transferência dos dados por um circuito auxiliar para a memória do hospedeiro, ao contrário de considerar a memória da interface diretamente alocada no mapa de memória do processador.

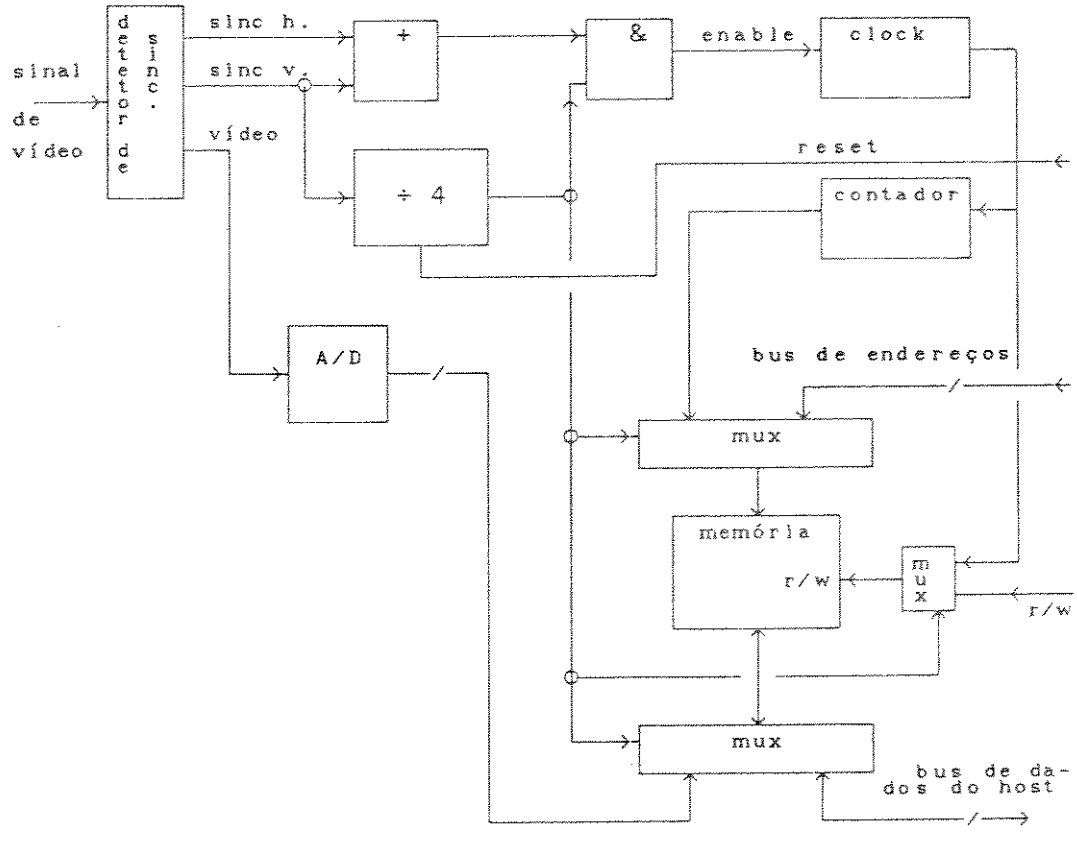


fig.II.9 - Esquema simplificado de uma interface de um digitalizador de vídeo.

O digitalizador descrito acima apresenta diversas simplificações com relação a implementação prática, mas descreve com razoável clareza os conceitos básicos envolvidos.

Um problema a ser considerado quando do projeto de um "frame grabber" relaciona-se a origem ou fonte dos sinais de sincronismo horizontal e vertical. Existem pelo menos três formas destes sinais serem gerados. O mais comum é a própria fonte do sinal de vídeo fornecer os sinais de sincronismo junto com a própria informação de vídeo, na forma de um sinal de vídeo composto ou mesmo separados. No caso do sinal de vídeo composto, o "frame grabber" deve interpretar o sinal enviado pela fonte (câmara ou outro dispositivo) e separar os sinais de sincronismo horizontal da mesma maneira que o faz um receptor de TV. Caso o sinal enviado pela fonte seja separado só há a necessidade de interpretação. Uma segunda forma de geração dos sinais de sincronismo ocorre quando o próprio circuito do "frame grabber" é responsável pela geração, enviando estes para a fonte de vídeo. Esta situação é adequada no caso de haver mais de uma fonte de vídeo, as quais, por sua vez, devem estar em sincronismo, e pode ocorrer por exemplo no caso de obtenção de imagens estereocópicas, em que duas imagens são capturadas simultaneamente. Uma terceira forma é baseada no uso de um gerador independente dos sinais de sincronismo, permitindo a sincronização de diversas fontes e "frame grabbers" e corresponde a uma solução intermediária entre as duas primeiras, sendo, também, mais versátil. A possibilidade de separação dos sinais de sincronismo no "frame grabber" permite a este obter imagens de um grande número de fontes, pois a maioria dos dispositivos existentes gera sinal de vídeo composto (câmaras, vídeo cassetes, etc). A geração de um sinal de sincronismo no "frame grabber" ou em um gerador independente é fundamental em certas aplicações e permite maior controle do formato da imagem gerada, mas requer fontes de vídeo especialmente adaptadas que permitam a operação com sincronismo externo.

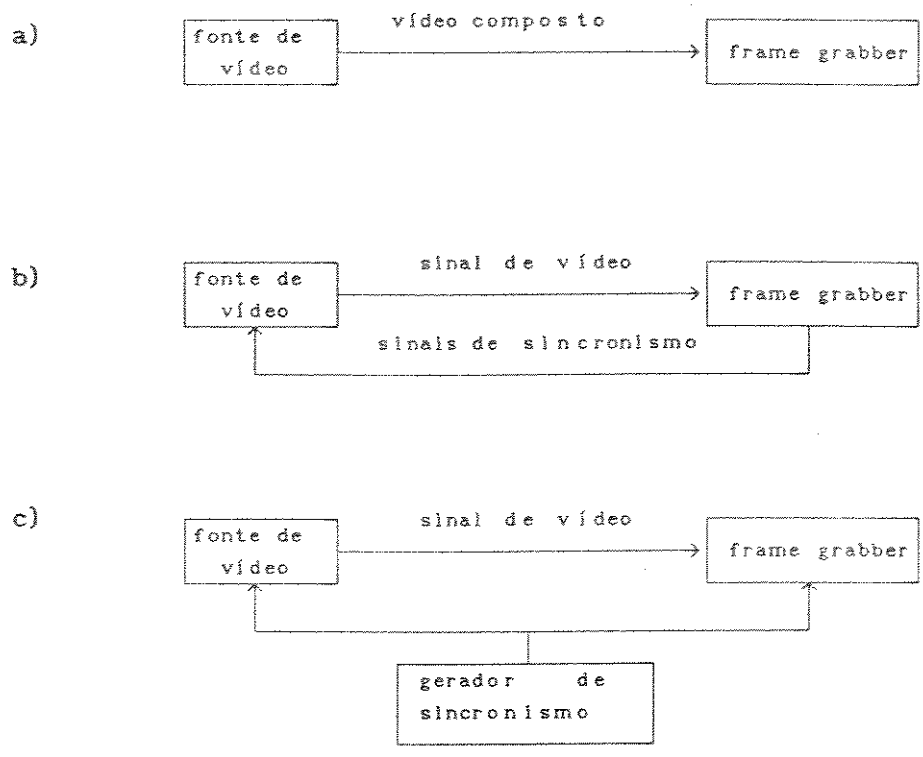


fig.10 - Possibilidades de origem da geração dos sinais de sincronismo. a) geração na fonte; b) geração no frame grabber; c) geração independente da fonte e do frame grabber.

II.4 - MEMÓRIA DE IMAGEM.

Como visto anteriormente, dois problemas principais devem ser resolvidos para a implementação da memória de imagem: a alta taxa de transferência de dados e a resolução de conflitos de acesso dos vários módulos do sistema a uma memória partilhada. As interfaces de vídeo (tanto de aquisição como de visualização) requerem uma alta taxa de transferência de dados, induzindo a organização da memória de imagem com palavras largas, e/ou exige o emprego de componentes rápidos. O processador de próposito geral não requer grande atenção quanto à organização da memória, entretanto, requer especial atenção quanto ao problema da disputa pelo acesso ao barramento da memória. O processador dedicado por sua vez, seja ele um processador gráfico ou um processador de imagem, possui exigências maiores, tanto no aspecto de organização da memória como no de taxa de transferência de dados e, consequentemente, resolução de conflitos no barramento.

O uso de componentes rápidos tende a minimizar ambos os problemas citados acima, entretanto, apesar da continua queda nos preços das pastilhas de memória, esta solução tende a encarecer o sistema, sem que se obtenha necessariamente o resultado esperado. Diversos fabricantes de memórias buscaram resolver alguns dos problemas apresentados através da introdução de componentes especialmente projetados para as aplicações visadas, principalmente no tocante aos dispositivos de saída gráfica.

A solução geral, entretanto, tende a ser um compromisso entre a organização interna da memória, entre o uso de componentes adequados àquela organização e também da própria organização e estrutura de interligação dos componentes do sistema.

A seguir são apresentados as características mais adequadas das memórias semicondutoras para aplicações em projetos de memória de imagem e as implicações na estrutura e organização da mesma.

II.4.1 - Memórias semicondutoras

Não é objetivo deste trabalho realizar umas descrição minuciosa do funcionamento das memórias semicondutoras, mas referenciar as características que mais se adequam ao emprego em memórias de imagem.

As memórias semicondutoras são usualmente classificadas em duas grandes classes: memórias estáticas e memórias dinâmicas. Ao contrário das memórias estáticas, as memórias dinâmicas devem ter seu conteúdo regenerado a intervalos regulares sob risco de perda da informação armazenada. As memórias estáticas ainda permitem um ciclo de operação (leitura ou escrita) mais rápido, porém consomem mais, são mais caras e têm encapsulamento menos denso. Memórias de imagem são usualmente de grande dimensão e por isso há uma tendência pelo uso de memórias dinâmicas, devido ao menor custo e maior densidade de encapsulamento.

O grande problema apresentado pelas memórias dinâmicas para emprego em memórias de vídeo é o elevado tempo de recuperação ("recovery time": tempo ocioso, necessário entre dois acessos consecutivos). Tipicamente, o tempo de ciclo é quase o dobro do tempo de acesso (aproximadamente 150ns para as memórias mais comuns). Além disso, o projeto com DRAM's é mais complexo devido a necessidade da regeneração periódica dos dados.

Para aumento de desempenho, os fabricantes agregaram diversas formas de acesso a fim de acelerar a transferência de dados.

Modo acesso paginado:

No modo página um determinado endereço é decodificado e mantido pelo sinal RAS. A leitura de qualquer célula de uma linha é então permitida apenas com a variação do endereço das colunas e do sinal CAS. O acesso as células não necessita ser a endereços em sequência. Através desta técnica diminui-se o tempo de ciclo de leitura/escrita de um dado. Até 256 bits de dados, uma página de memória, podem ser lidos dessa maneira.

Modo Nibble:

Este modo permite até quatro bits de dado serem acessados em cada ciclo RAS. Neste modo a memória é organizada internamente em quatro arranjos (matrizes) de células de memória, embora externamente se comporte como um único arranjo. Ao se fornecer um endereço para a memória, quatro bits de dados, um de cada arranjo, é armazenado internamente em um latch, e as duas linhas menos significativas do endereço são usadas para selecionar um deles através de multiplexação. Pela pulsação do sinal de CAS, sem que nenhum endereço adicional seja fornecido, é possível acessar sequencialmente os três bits restantes no latch. É possível através desta técnica obter taxas de acesso de 80ns.

Modo Ripple:

Neste modo a memória opera de maneira similar ao modo página, porém o circuito de decodificação de endereços das colunas é implementado com circuitos estáticos mais velozes [11], fazendo com que os tempos de ciclo sejam da ordem 40ns contra 125ns no modo página.

Modo coluna estática:

Este modo é também uma variante do modo paginado em que o circuito de decodificação é estático, mas o sinal CAS não pulsa. Uma mudança de endereço de coluna traz o dado correspondente prontamente.

Embora tais técnicas de acesso melhorem as taxas de transferências, nenhum outro acesso pode ser feito durante estas. Visando solucionar este problema e tendo em vista que a saída de vídeo é tradicionalmene implemtada com um registrador de deslocamento, a Texas Instruments propôs em 1981 um dispositivo de memória "dual port" denominada vídeo RAM ou VRAM.

II.4.1.1 - Vídeo RAM's:

Em adição a uma RAM dinâmica comum uma VRAM típica inclui um registrador de acesso serial de 256 bits, como ilustrado na fig.II.11. O registrador serial, denominado SAM (Serial Access Memory), comporta-se de maneira similar a um registrador de deslocamento e permite a entrada e saída de dados. Um ciclo especial interno transfere uma linha completa do arranjo da memória dinâmica para o registrador SAM. Após a transferência da linha da RAM para o SAM, ou vice-versa, é possível realizar-se o acesso independente à RAM dinâmica e ao registrador de deslocamento. Embora uma VRAM não seja uma memória "dual port" real, comporta-se como tal a maior parte do tempo de operação, proporcionando um elevado ganho na banda de passagem.

O endereço do primeiro bit do registrador serial a ser transferido pode ser especificado, pois o registrador é constituído de um conjunto de latches em paralelo com lógica endereçável de escrita e leitura. Um contador pré-programável aponta o bit a ser transferido e varre o registrador circularmente.

Um sinal adicional, DT, é necessário para as transferências entre a DRAM e o registrador SAM. Dependendo se o sinal DT é ativado antes ou depois do sinal RAS, ocorre uma transferência de dados para o SAM, ou um ciclo normal de acesso à memória. O registro SAM da VRAM é carregado pela leitura de uma linha da memória com o sinal DT ativo no início do ciclo (fig. II.12). Se o sinal WR está ativo na borda ativa do sinal RAS, o conteúdo do registrador SAM é escrito na linha de memória selecionada. O endereço presente na borda ativa do sinal CAS é usado para selecionar o endereço inicial da transferência serial. No modo de leitura, os dados serão deslocados a partir daquele endereço. No modo de escrita, o próximo dado deslocado será escrito no registrador SAM naquele endereço.

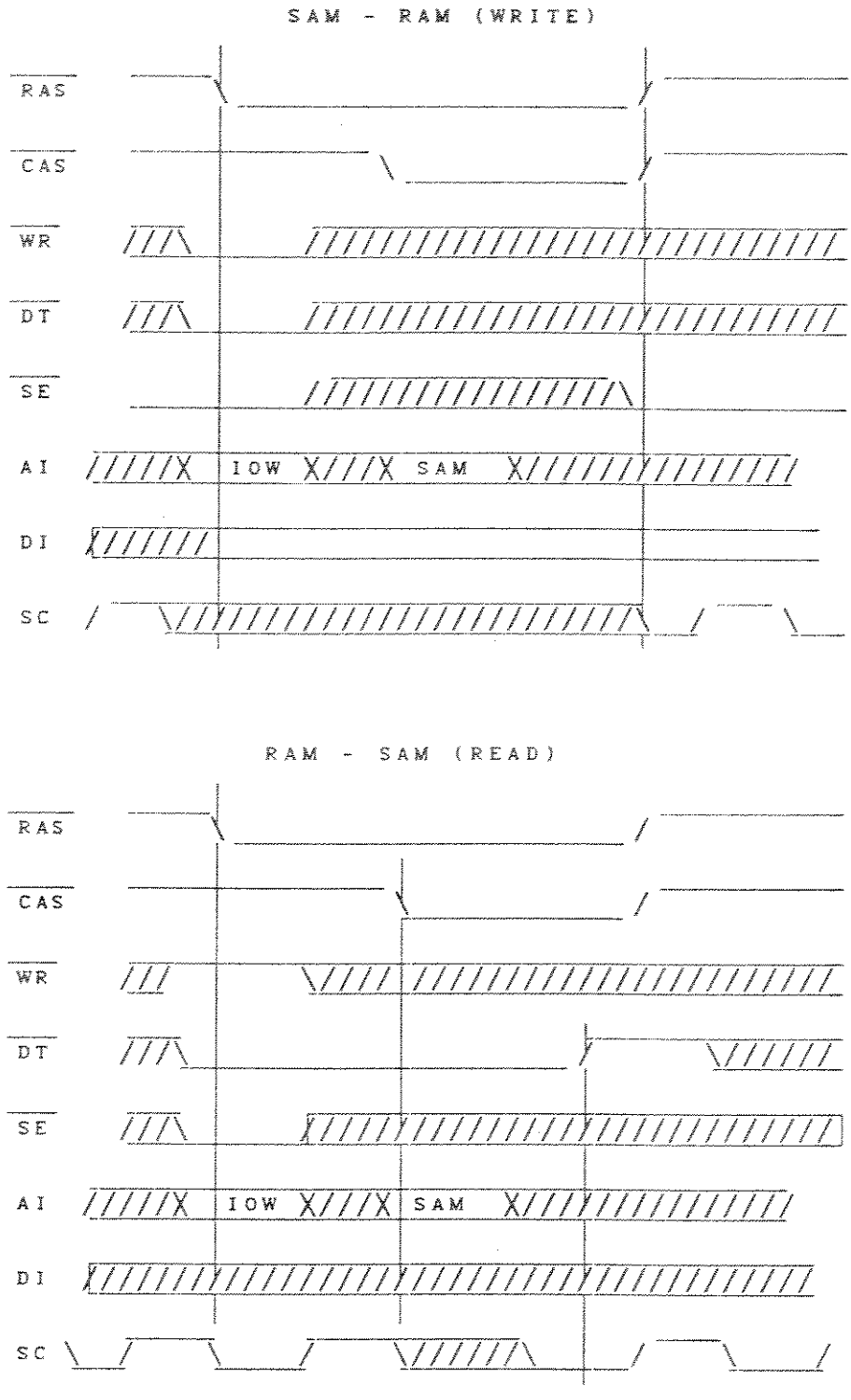


fig.II.12 - Transferência de leitura e escrita no registrador SAM.

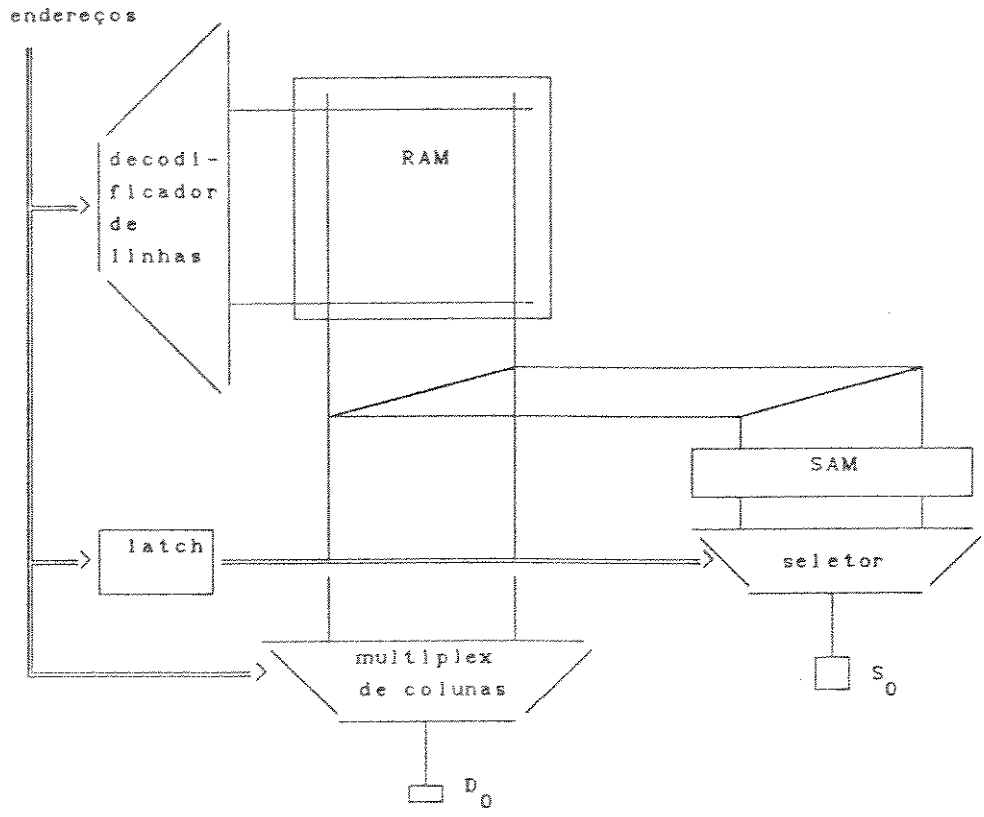


fig.II.11 - Diagrama de blocos de uma VRAM.

O registrador de deslocamento SAM.

O registrador SAM comporta-se como um simples registrador de deslocamento. A borda positiva do clock desloca os dados para fora ou armazena os dados de entrada (fig.II.13). Se a borda do clock ocorre quando o sinal SE não está ativo, o contador é incrementado, mas o dado endereçado não é lido ou escrito. É importante notar que após uma transferência RAM-SAM, o novo dado estará disponível na saída serial somente após a primeira borda positiva seguida da desativação do sinal DT.

As VRAM's apresentam, ainda, os diversos modos de endereçamento das RAM's dinâmicas comuns e a necessidade de regeneração periódica dos dados. Alguns fabricantes adicionaram

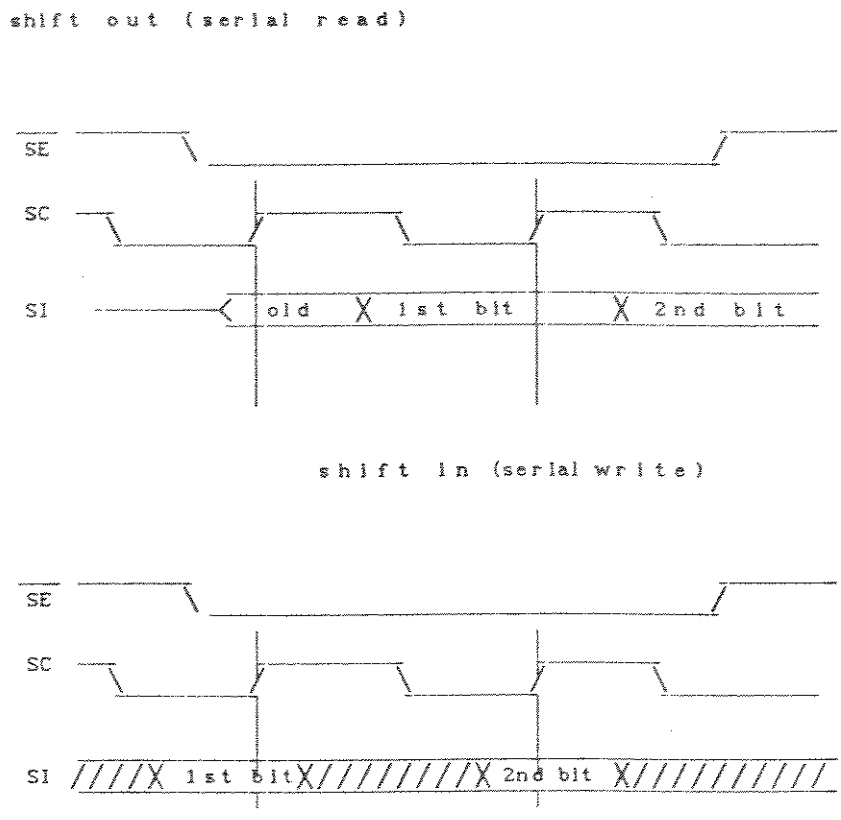


fig.II.13 - Temporização do registrador SAM.

caracteríticas especiais, úteis em aplicações gráficas, como por exemplo escrita mascarável, dependente da temporização do sinal de escrita, ou operações mais sofisticadas, como operações lógicas internas entre o conteúdo da célula e o dado sendo escrito. Uma descrição mais detalhada dessas características e da própria operação das VRAM's pode ser encontrada em Nicoud [12].

Comparada com uma DRAM de mesma capacidade, uma VRAM possui um encapsulamento maior, mais longo e mais largo. A potência consumida é também maior, mas o preço por pastilha é aproximadamente o mesmo. Levando em consideração que uma DRAM comum possui aproximadamente quatro vezes a capacidade de uma VRAM, a placa de memória realizada com VRAM's é

aproximadamente quatro vezes mais cara, oito vezes maior e consome oito vezes mais potência que uma placa de memória realizada com DRAM's. Entretanto, ao se considerar a lógica e os registradores de deslocamento necessários em uma placa de vídeo, o uso de VRAM's é favorecido, especialmente para telas de alta resolução.

II.4.2 - A ORGANIZAÇÃO DAS MEMÓRIAS DE IMAGEM

O processador hospedeiro é o módulo que menos restrições estabelece para o projeto do sistema de memória. Uma característica desejável é que o endereçamento da memória tenha correspondência direta com o mapeamento da imagem na tela , isto é, que a cada endereço na memória corresponda um pixel de imagem e que a correspondência seja em ordem direta. Dessa maneira, ao primeiro pixel na tela corresponde o primeiro endereço na memória de imagem, ao segundo pixel corresponde o segundo endereço e assim sucessivamente. Esta organização é interessante apenas pela sua representação uniforme que facilita a descrição da imagem em termos de estrutura de dados, mas não é impositiva.

O conjunto controlador de regeneração e digitalizador impõe restrições bem maiores ao sistema de memória. O problema da velocidade de transferência cresce com a resolução da imagem e é solucionada tradicionalmente com o acesso paralelo a diversos pixels, na memória de imagem, e em seguida, transferindo-os para um registrador de deslocamento no circuito de vídeo que então se encarrega da serialização. O uso de componentes (memórias) rápidos diminui a quantidade de pixels a ser acessado em paralelo facilitando a implememtação.

Em geral, a memória de vídeo é organizada de forma que o endereçamento corresponda ao formato de varredura da tela. Esta forma de organização facilita o acesso e a utilização dos modos de acesso rápido (paginado, ripple, nibble, etc) e consequentemente melhoram a taxa de transferência. Diversas

alternativas de implementação são possíveis e uma análise detalhada a respeito do assunto pode ser encontrada em Fallin [11] e em Witton [13]. A solução do problema da disputa pelo acesso à memória é uma extenção da solução anterior. A medida que mais posições de memória são acessadas simultaneamente pelo controlador de regeneração, uma maior porcentagem do tempo ficará livre para acesso dos processadores. As memórias VRAM's levam esta solução ao extremo pela incorporação do registrador de deslocamento na própria pastilha de memória, e representam o estado de arte em memórias para interfaces de vídeo. Entretanto, é importante ressaltar que a simples resolução do problema de taxa de transferência não resolve, necessariamente, o problema da disputa pelo acesso entre o controlador de regeneração e os outros elementos do sistema. O uso de componentes rápidos também contribui para a solução deste problema, pois diminui o tempo de cada ciclo de acesso e consequentemente o tempo de ocupação do barramento.

A utilização de dois buffers de imagem com canais de acesso separados é um recurso que diminui os problemas de conflitos na disputa pelo acesso à memória. Enquanto a imagem em um dos buffers é mostrada na tela pelo controlador de regeneração, a imagem do outro buffer pode ser atualizada pelos processadores sem qualquer disputa pelo acesso ao barramento. Ao término da operação de atualização realizada pelo processador, os dois buffers são trocados, isto é, a imagem recém atualizada passa a ser exibida na tela enquanto que a outra passa ao controle do processador para atualização. Esta técnica é denominada "double buffering" ou memória "ping-pong" e possui como desvantagem, além de usar o dobro de memória, a necessidade das operações de atualização serem repetidas em ambos os buffers.

O encapsulamento das memórias é um aspecto importante na implementação das soluções propostas acima. Assim, memórias organizados em palavras largas facilitam o acesso a diversos pixels em paralelo. Pelo mesmo motivo, o aumento da capacidade de armazenamento das pastilhas de memórias, causa dificuldades na implementação, embora esta seja uma tendência verificada no mercado [14]. O seguinte exemplo ajuda a esclarecer: suponha um terminal monocromático de 512x512 pixels de resolução a 30Hz, implementado com uma única pastilha de memória de 256Kx1; a memória de imagem deve ser acessada a cada 100ns, impedindo o acesso, à memória de imagem, de qualquer outro dispositivo,

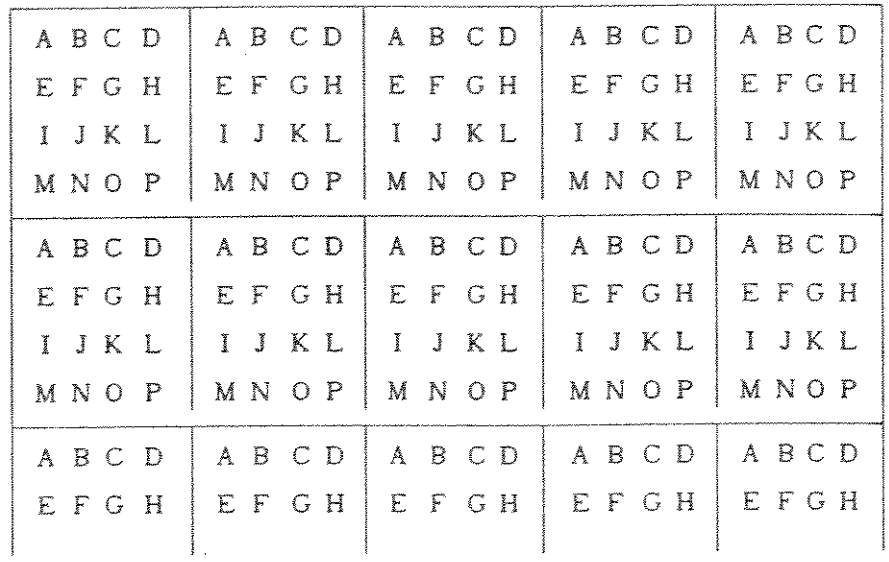
durante a varredura. Ao contrário, o uso de pastilhas com organização interna diferente, por exemplo, quatro pastilhas de 64Kx1 ou uma de 64Kx4, permite acessos a cada 400ns, propiciando oportunidade de acessos de outros dispositivos nos intervalos entre um acesso e outro.

O uso de processadores especializados para geração de primitivas gráficas ou processamento de imagem, pode exigir, da mesma forma que os controladores de regeneração, uma alta taxa de transferência de dados para atualização dos dados da imagem. A solução, pela própria semelhança do problema, parece ser o acesso a vários pixels em paralelo. É comum a utilização do tempo de retraço do feixe para a atualização do conteúdo da memória de imagem otimizando o processo.

Embora o acesso paralelo a diversos pixels na mesma ordem de varredura da tela solucione os dois problemas principais, da taxa de transferência de dados e da disputa do barramento, ele pode não ser a melhor solução para a atualização do conteúdo da memória de imagem.

A geração de primitivas gráficas e, principalmente, processamento de imagem de baixo nível envolve a manipulação de algoritmos locais, isto é, que manipulam endereços dos pixels localizados em uma área limitada em torno de um pixel. Assim, para esse tipo de processamento a organização da memória em linhas na mesma ordem do rastreamento não é a mais eficiente. Pode-se exemplificar a ineficiência da organização da memória em linhas pela geração de vetores verticais ou inclinados, ou o sombreamento de um polígono. Para a atualização dos dados de um vetor vertical é necessário atualizar um pixel em cada acesso à memória de imagem, pois cada pixel do vetor está localizado em uma linha diferente. Assim, com a memória organizada em grupos de acesso linear, a atualização da memória é otimizada apenas para primitivas que tenham seus pixels localizados em uma mesma linha e portanto vários pixels das primitivas são acessados simultaneamente na memória. Para a otimização da atualização de primitivas que possuam seus pixels espalhados por diferentes linhas é interessante que a organização da memória se faça na forma de blocos de acesso paralelo constituídos de matrizes bi-dimensionais ou vetores verticais. Caso a primitiva gerada cubra uma única célula ou matriz, o processador pode gerar a primitiva por completo e realizar a atualização de

uma única vez, através do acesso paralelo de todos os pixels daquela célula, aumentando a velocidade do processo.



□ células matriciais de memória.

fig.II.14 - mapeamento pastilha (A até P)-por-pixel parauma matriz organizada simetricamente.

A fig.II.14 mostra o mapeamento pixel por pastilha de memória para 16 pastilhas de 64Kxl em uma tela de 1024x1024 pixels. Células consecutivas formadas por 16 pastilhas ao longo de uma linha de varredura são semelhantes; as células tem organização simétrica e somente quatro pastilhas aparecem em qualquer linha. Uma matriz usada desta maneira perde a vantagem da leitura paralela pelo controlador de regeneração e não pode ser implementada. Se pastilhas com capacidade de armazenamento menor são usadas, o tamanho do registrador de deslocamento de saída de vídeo aumenta significativamente, mas o projeto é exequível.

A fig.II.15 mostra uma matriz na qual os grupos de quatro pastilhas são deslocados uma linha em cada célula, isto é, são escalonados. Cada pastilha aparece apenas uma vez em cada série de 16 pixels sequenciais em cada linha de rastreamento, e, assim, a varredura pode ser suportada. Como os 16 pixels encontram-se em quatro diferentes células, o controlador de vídeo deve enviar quatro diferentes endereços para acessar os pixels ao longo da linha de rastreamento.

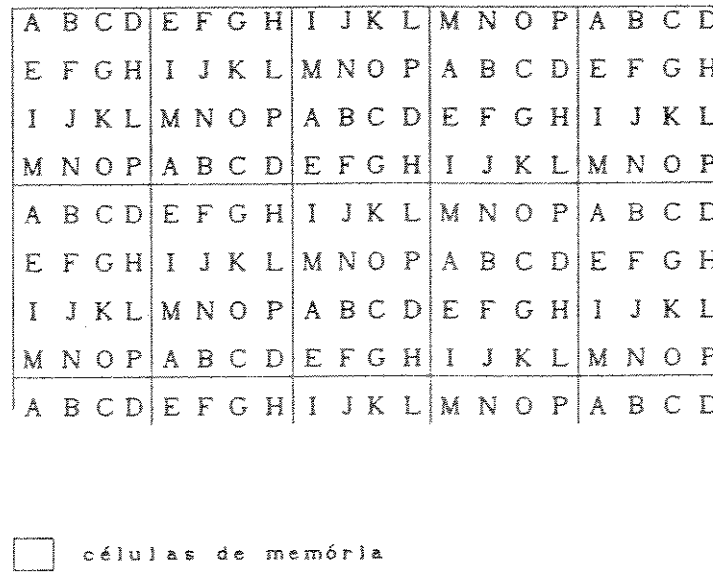


fig.II.15 - mapeamento pastilha (A até P) por pixel para organização da memória com células escalonadas.

O controlador de regeneração acessa uma memória organizada em matrizes sobre células com fronteiras regularmente limitadas. O desempenho é melhor se o processador de imagem puder ler e escrever em células com fronteiras arbitrárias. Em qualquer das duas organizações apresentadas, simétrica ou escalonada, cada pastilha contém somente um pixel de qualquer área arbitrária de dimensão 4x4 na tela. Através da manipulação adequada de endereços, o acesso a matrizes com fronteiras arbitrárias pode ocorrer.

Em especial, sistemas dedicados de processamento de imagem, que fazem uso de paralelismo para a aceleração do processamento requerem que a organização da memória permita acessos a blocos com formatos e fronteiras arbitrários. Uma imagem pode ser representada por uma matriz de pontos de imagem MxN onde cada elemento **I(i,j)** da matriz, para 0<i<M e 0<j<N, é um inteiro ou um conjunto de inteiros que representam côr e/ou intensidade de uma porção da imagem. Uma grande quantidade de operações ou algoritmos para processamento de imagem requerem sistemas que permitam acesso simultâneo a uma fila e/ou bloco de **I(*,*)**. Por exemplo diversos algoritmos fazem acesso a um ponto de imagem e seus 8 pontos vizinhos, os quais constituem um bloco de pixels de dimensão 3x3. Deve-se notar que o local dentro da imagem em que um bloco é acessado é aleatório. Dessa forma, um processador de imagem terá desempenho melhor se o sistema de memória permitir acesso simultâneo a um bloco **BL** de imagem de dimensão **pq** onde pq<<MN e, é definido por:

$$BL(i,j) = \{\ I(\ i+a,\ j+b\ )\ /\ \ 0<a<p,\ \ 0<b<q\ \},$$

$$0 < i < M-p,\quad 0 < j < N-q.$$

Repare que se "**a**" ou "**b**" são constantes, tem-se o acesso, respectivamente, a uma linha ou coluna de dimensão **p** ou **q**. Para implementar um sistema de memória com essas características é necessário que hajam **m** módulos de memória que possam ser acessados simultaneamente e com endereçamento independente onde **m** > **pq**. Sistemas desse tipo são apresentados por Woorhis [15] e por Park [16]. De fato, **p** e **q** são parâmetros para o projeto do sistema e são definidos na especificação do processador de imagem.

Capítulo III

SPI - um sistema para desenvolvimento de pesquisa em processamento de imagem

III.1 - INTRODUÇÃO:

Neste capítulo é apresentada e discutida a concepção de uma estação de trabalho para suporte ao desenvolvimento de "software" e "hardware" em processamento de imagem.

Os objetivos principais da implementação desse sistema são propiciar um ambiente com os recursos básicos para o desenvolvimento de aplicações em processamento de imagem e a obtenção de uma arquitetura que permita o emprego de processadores especializados de alto desempenho com a exploração de paralelismo nas operações. O projeto foi desenvolvido tendo em mente a obtenção de uma solução versátil, simples, sem a utilização de componentes sofisticados, mas garantindo, tanto quanto possível, a obtenção de alto desempenho.

Mais especificamente, o sistema proposto foi projetado com os seguintes requisitos:

- realizar captura e digitalização de imagens de câmeras de vídeo;
- realizar o display da imagem em terminal de vídeo;
- memória de imagem para armazenamento das imagens a serem processadas;
- um estrutura de barramentos que permite o acesso aos "buffers" de memória de imagem com uma alta taxa de transferência de dados;
- e ainda suporta processadores especializados que permitam a aceleração do processamento de algoritmos, ou, etapas de algoritmos para processamento de imagem.

A arquitetura proposta segue em linhas gerais aquela descrita no capítulo II. Adicionalmente, incluiu-se dois barramentos especializados para a transferência exclusivamente de dados de imagem com uma alta banda de passagem. Neles podem ser acoplados processadores especializados para tarefas de processamento de imagem e um sistema de aquisição e visualização de imagems. O uso de dois barramentos de imagem independentes, embora

dispendioso, tem como vantagens a facilidade de aquisição ou visualização de duas imagens simultaneamente e, principalmente, da obtenção de um alto desempenho devido a quase inexistência de disputa pelo acesso ao barramento da memória. Além disso, um processador especialmente projetado, que tenha acesso a ambos os barramentos, pode utilizar um "buffer" de imagem localizado em um barramento como fonte de dados e o "buffer" localizado em outro barramento para armazenar os resultados. Essa configuração permite, através de um projeto adequado, a sobreposição das operações de busca de dados, execução e armazenamento de resultados, conduzindo a uma grande eficiência no processamento. Todos os elementos do sistema são controlados por um computador hospedeiro, no caso específico desta implementação, por um microcomputador PCxt. Cada elemento do sistema, inclusive o próprio barramento de imagem, possui uma interface direta com o barramento PC. Assim, o micro PCxt atua como árbitro do barramento de imagem, estabelecendo modos de operação bem definidos nos quais determinados elementos atuam como mestres e outros como escravos, evitando a ocorrência de conflitos.

A arquitetura geral do sistema é apresentada no diagrama de blocos da fig.III.1. O sistema é constituído por quatro módulos principais: processador hospedeiro, processador de imagem, controlador de regeneração/digitalizador e sistema de memória. A descrição da função e das características de cada módulo é feita nas próximas secções.

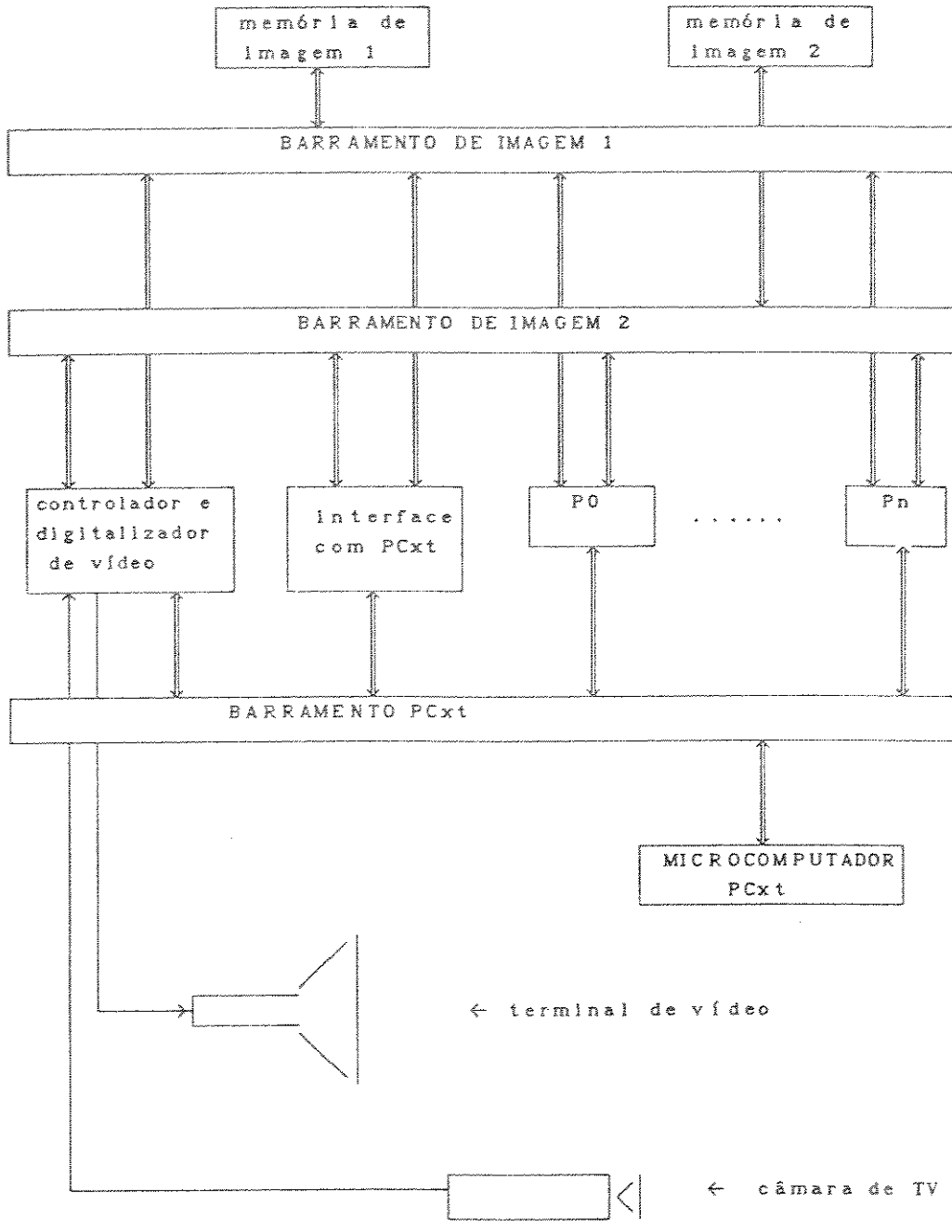


fig.III.1 - Diagrama em blocos da estação de trabalho para processamento de imagem SPI.

III.2 - O BARRAMENTO DE IMAGEM

O barramento de imagem é constituído de um conjunto de linhas para transmissão exclusivamente de dados de imagem. Na realidade, é constituído de dois barramentos idênticos, que permitem acesso separado e simultâneo a dois "buffers" de imagem. Foi concebido visando minimizar ao máximo os problemas de conflito de acesso à memória de imagem por diversos dispositivos sem lançar mão de esquemas sofisticados de arbitragem e, ao mesmo tempo, permitir uma alta taxa de transferência de dados. A alta taxa de transferência de dados é obtida pelo uso de uma palavra de dados larga, 64bits. O esquema de arbitragem é bastante simples e é, na verdade, independente do próprio barramento.

III.2.1 - Sinais

O barramento de imagem é uma barramento bastante simplificado sendo constituído de um conjunto de sinais para a transferência dos dados da imagem, um conjunto de linhas para endereços e um pequeno conjunto de sinais para o controle das transferências. Existem 64 linhas destinadas a transferência de dados possibilitando a transferência de 8 pixels simultaneamente. Dezoito linhas de endereço possibilitam o acesso a 256Kbytes de memória. Existem três linhas de controle, uma para determinar se a operação em andamento é de escrita ou leitura de dados da memória e uma para determinação do tipo de palavra, se byte ou bloco e uma para a habilitação do "buffer" de imagem. Cada um desses sinais existe em ambos os barramentos de imagem e são independentes.

DI_{0-63} - Linhas de dados (64 linhas).
AI_{0-14} - Linhas de endereço para acesso bloco (15 linhas).
AI_{15-17} - Linhas de endereço para acesso byte (3 linhas).
BL/BY - linha de controle para acesso bloco ou byte
RD/WR - linha de controle para determinação de operação de leitura ou escrita.
HB(0/1) - sinal de habilitação do buffer de imagem (um para cada buffer).

III.2.2 - Arbitragem

A arbitragem dos barramentos de imagem é exercida pelo microcomputador PCxt através da determinação de modos de operação nos quais são definidos quais elementos atuam como mestre do barramento e quais são escravos. Esse esquema de arbitragem foi escolhido pela simplicidade de implementação e pelas próprias características de operação dos elementos do sistema. Assim, de uma maneira geral, não é necessário que o processador hospedeiro dispute o acesso à memória com o digitalizador durante o processo de aquisição de uma imagem, ou, que haja a visualização de uma imagem durante um processo de análise de cena. Em outras palavras, raramente dois elementos do sistema estarão ativos ao mesmo tempo, disputando o acesso à memória de imagem; a operação de um exclui a operação de outro por um período de tempo relativamente longo.

Levando em conta as situações mais prováveis ou desejáveis para a operação do sistema, definiu-se vários casos em que cada componente teria acesso a um ou outro "buffer" de imagem, ou ainda a ambos. A forma de acesso a cada um dos "buffers" foi definida, para cada um dos elementos, do seguinte modo:

a) **Processador de Imagem:** Pode acessar cada "frame buffer" independentemente ou ambos simultaneamente. Realiza acesso a bloco.

b) **Digitalizador/display:** Pode acessar apenas um dos "frame buffers" por vez. Da mesma forma, ao Processador de Imagem, pode realizar uma operação de escrita/leitura a um bloco de 8 pixels.

c) **Computador Hospedeiro:** Pode acessar, também, apenas um dos "frame buffers" por vez. Entretanto, apenas um pixel por vez pode ser operado.

Como cada um dos dispositivos pode, em geral, acessar "buffers" diferentes em um determinado momento, tem-se dentre os mais prováveis, os modos de operação apresentados na tabela III.1.

TABELA III.1

modo	proc de imagem	ctrl/digit	proc. hosp
0	BI1	BIO	BIO
1	BIO	BI1	BI1
2	-	BI1	BIO
3	-	BIO	BI1
4	BIO/BI1	-	-
5	-	BIO/BI1	-
6	BI1	BIO	-
7	BIO	BI1	-

Tab.III.1 - Esta tabela descreve os modos de operação do barramento de imagem. Através dela é possível verificar quais elementos têm acesso a quais buffers de imagem em determinado modo de operação. (BIO - buffer de imagem 0; BI1 - buffer de imagem 1).

Cada um desses modos representa uma situação específica para o desenvolvimento de determinada função. Por exemplo, o modo de operação 0 caracteriza o uso do "buffer" 1 pelo processador de imagem enquanto o controlador de regeneração exibe a imagem no "buffer" 0 ao mesmo tempo que o processador hospedeiro a processa. No modo 5 é possível a aquisição de duas imagens provenientes de fontes diferentes. No modo 4, o processador de imagem pode estar utilizando um "buffer" para busca de operandos e o outro para armazenamento de resultados. É possível vislumbrar várias situações e funções para cada um dos modos de operação.

Segundo o descrito acima, poderá haver situações nas quais o Processador de Imagem, operando simultaneamente sobre ambos os "buffers", provocará a interrupção da visualização da imagem. Essa situação é tolerável por que quando o processador de imagem está em operação a prioridade é a obtenção de resultados tão rapidamente quanto possível, tendo a operação de visualização

uma prioridade menor. Entretanto, não é desejável que haja interrupção da visualização da imagem quando o processador hospedeiro acessar a memória porque sua operação é muito lenta e suas operações são em geral visualizáveis. Para garantir o acesso do processador hospedeiro ao conteúdo da memória de imagem sem que haja interferência no processo de visualização existem três possibilidades:

1 - Utilização do princípio de memória "duplo buffer" com a atualização de dados em um "buffer", para posterior chaveamento do "buffer" sendo utilizado para visualização.

2 - Realização dos acessos à memória de imagem apenas no retraço do feixe de elétrons.

3 - Um circuito de arbitragem específico para o processador hospedeiro e o circuito controlador de regeneração controla os acessos durante o traço do feixe, permitindo acessos intercalados, sem distúrbios na exibição da imagem.

A primeira opção consiste em executar um processo qualquer sobre um dos "buffers" enquanto o outro tem a imagem exibida. Após o término do processamento, passa-se a exibir a imagem já processada através do simples chaveamento entre os dois "buffers". Na sequência, a mesma atualização feita no primeiro "buffer" deve ser repetida no segundo. Obviamente há o inconveniente de realizar a mesma operação sobre dois "buffers".

A segunda opção consiste em monitorar o estado do controlador de regeneração e acessar a memória no momento em que este estiver realizando o retraço do feixe. Como nenhum acesso à memória é feito durante o retraço pelo controlador de regeneração não existirá perturbação na exibição da imagem. A limitação desta técnica é o pouco tempo disponível para o processador acessar a memória.

O circuito de arbitragem para disputa de acesso entre o processador hospedeiro e o controlador de regeneração utiliza a temporização do relógio de

vídeo para estabelecer janelas entre dois acessos sucessivos do controlador à memória. Nestas janelas o processador hospedeiro pode acessar a memória. Se o processador inicia o acesso fora da janela o ciclo é retardado até a próxima janela, quando então é efetivado. Com isso tem-se que, no pior caso, a cada dois ciclos de acesso do controlador de regeneração poderá haver um ciclo do processador hospedeiro. Esta opção foi implementada através de um circuito localizado no controlador de regeneração e será descrito na secção III.4.

III.2.3 - Eficiência do barramento

Como o barramento não é síncrono, sua taxa de transferência está limitada somente à tecnologia dos componentes utilizados e as características elétricas e físicas dos condutores. Usando memórias de 100ns de ciclo de operação pode-se atingir uma taxa de transferência de 80MBytes/s em cada um dos barramentos de imagem. O uso de dois barramentos simultaneamente permite atingir 160MBytes/s. Essas considerações levam em conta que seja possível operar com sinais de 10Mhz no barramento.

Em função dessa taxa de transferência é possível acessar todos os dados contidos na memória de imagem em pouco mais de 3ms. Este tempo de transferência é bem menor que o tempo de geração de uma imagem e portanto habilita, em princípio, que ocorra o processamento em tempo real das imagems armazenadas na memória. Evidentemente, para que isto ocorra é necessário que o processador (ou processadores) utilize adequadamente as características do barramento.

III.3 - CARACTERÍSTICAS DO SISTEMA DE MEMÓRIA PROJETADO:

O sistema de memória de imagem é composto de 2 "buffers", cada um dos quais capaz de armazenar uma imagem completa de 512 linhas e 512 colunas com 256 níveis de intensidade e/ou côr e acessados através de barramentos diferentes. Isto representa uma memória total com 512 Kbytes (2 x 256 Kbytes). A utilização de 2 "buffers" em barramentos diferentes apresenta diversas vantagens como a possibilidade de obtenção de dois operandos simultaneamente no caso de por exemplo operações lógicas serem realizadas sobre duas imagens, ou além disso, uma imagem pode ser mostrada no vídeo ou adquirida enquanto a outra é processada.

São usados 8 chips de memória estática de 32Kx8 bytes, 100ns de tempo de acesso, para cada "buffer". A memória está organizada para permitir o acesso linear por parte do processador hospedeiro, isto é, a um pixel de cada vez, organizados na mesma ordem de varredura da tela. O processador de imagem e o controlador/digitalizador realizam o acesso a 8 pixels de cada vez, na forma de um bloco horizontal e, portanto, endereçam linearmemnte 32K endereços. Para possibilitar o acesso nos dois formatos, byte e bloco linear existe uma sinal de controle gerado normalmente pelo PCxt que define o tipo de acesso.

A utilização de memórias estáticas deu-se não só pelo aspecto de simplicidade, mas também por que o projeto foi desenvolvido antes da especificação do processador dedicado. O projeto desenvolvido com memórias estáticas permite uma maior liberdade de atuação do processador dedicado pois não há a preocupação com a regeneração do conteúdo da memória como acontece com as memórias dinâmicas. Além disso, memórias estáticas permitem um ciclo de acesso bastante reduzido propiciando alto desempenho.

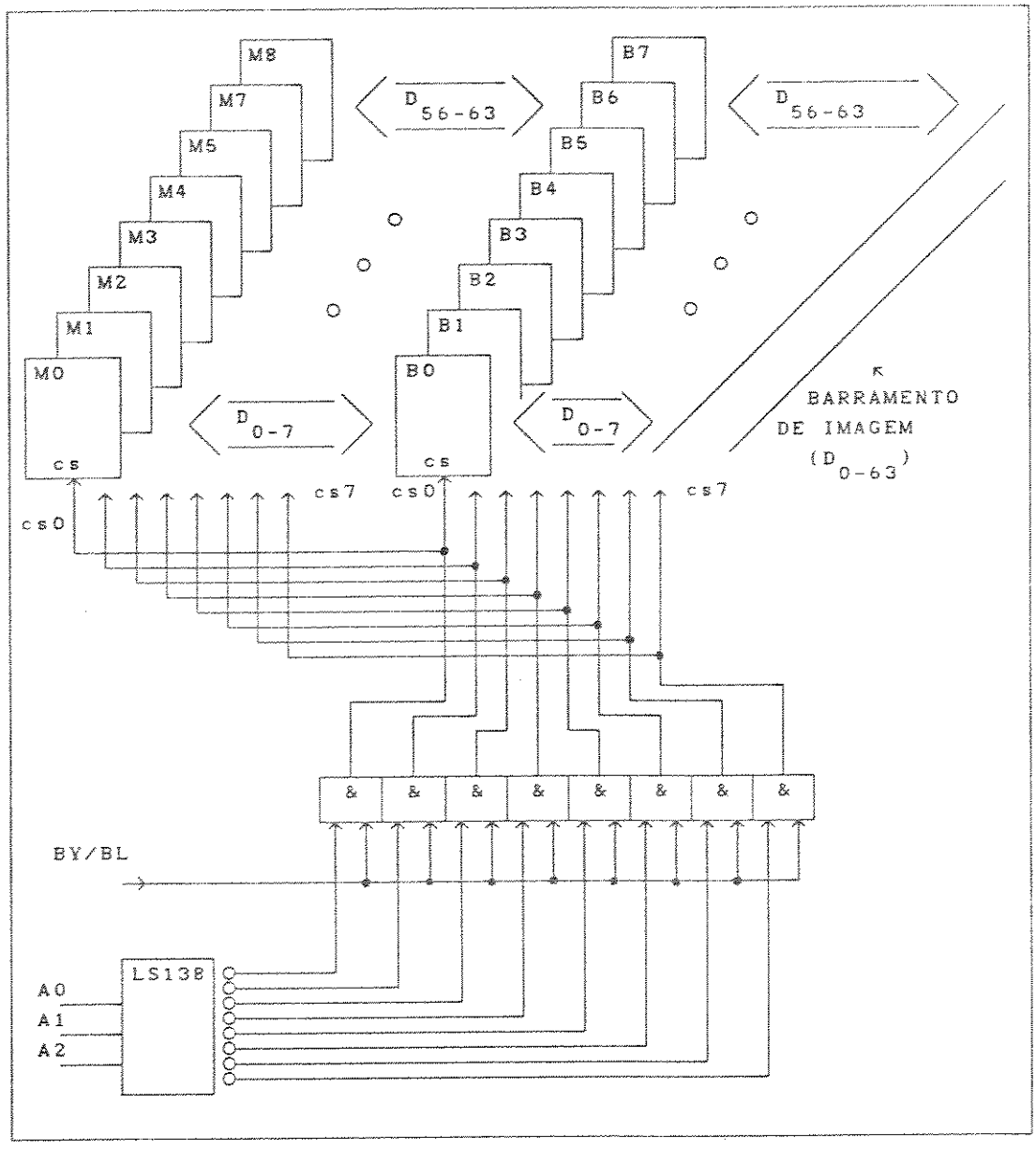


fig.III.2 - Esquema da memória de imagem para acesso modo byte (BY/BL=1) e para acesso bloco (BY/BL=0).

III.4 - CONTROLADOR DE REGENERAÇÃO-DIGITALIZAÇÃO.

O controlador de regeneração/digitalização é responsável por todas as atividades referentes a aquisição e display da imagem. Possui um circuito de controle que permite a geração dos sinais básicos de sincronismo para a geração de imagem em vídeo e que são enviados também para controle de câmeras de televisão P&B. A digitalização é feita por meio de um conversor A/D de alta velocidade tipo "flash", até o máximo de 8 bits de resolução (limitação imposta pela memória).

Display/Digitalizador:

- resolução máxima 512x512 pixels;
- 8 planos (ou 256 níveis de cinza);
- monocromático;
- saída de vídeo analógico ou digital;
- entrada de vídeo possível por até duas câmaras

No capítulo II, na apresentação da estrutura da estação de trabalho para processamento de imagem, caracterizou-se as interfaces de aquisição e visualização da imagem como elementos distintos. Embora esta caracterização seja correta, é comum a construção do controlador de digitalização e regeneração na forma de um único dispositivo. Essa configuração traz facilidades na implementação pelos diversos pontos em comum entre as duas inteFaces.

A estrutura do circuito é apresentada na fig.III.3. Existe um circuito controlador, responsável pela geração dos sinais de temporização de vídeo e pelos sinais de interface com a memória. Os sinais de temporização de vídeo são enviados aos circuitos de entrada e saída de vídeo.

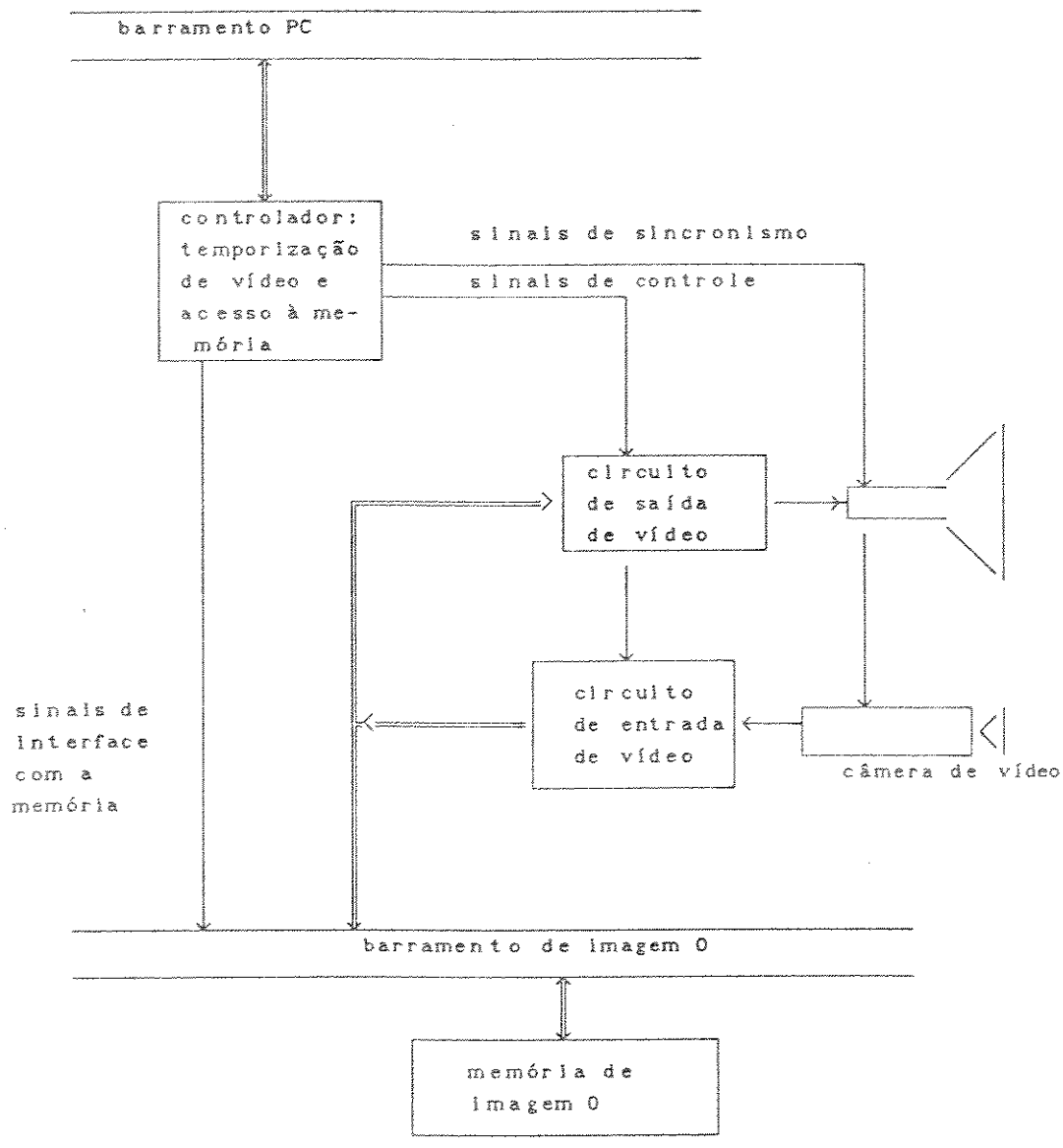


fig.III.3 - Estrutura geral do circuito de aquisição e vizualização de imagem.

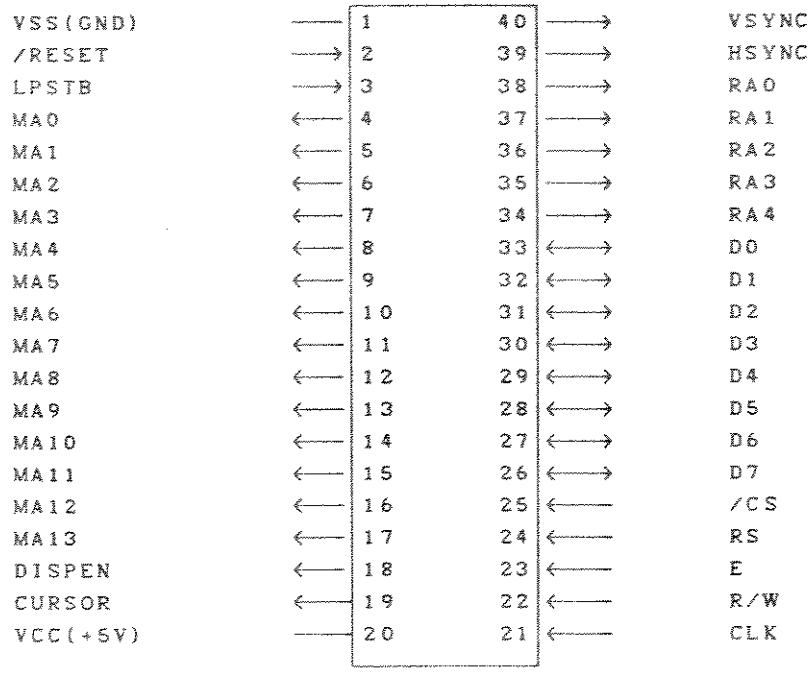
O circuito de saída de vídeo consiste principalmente de uma série de oito registradores de deslocamento com entrada paralela e saída serial, que serializam os pixels que chegam da memória em formato paralelo. Opcionalmente pode ser acrescentada uma palete na saída de vídeo para a obtenção de mais

recursos no caso de se desejar a obtenção de imagens com côr. O circuito de saída de vídeo é responsável pela compatibilização dos níveis elétricos do circuito com o monitor utilizado. Alguns monitores exigem saída TTL outros analógicos, etc.

O circuito de entrada ou de digitalização é composto de um conversor analógico/digital e um conjunto de registradores de deslocamento com entrada serial e saída paralela que são responsáveis pela paralelização do sinal de vídeo digitalizado. O sinal de vídeo é amostrado e convertido pelo conversor A/D e as amostras digitalizadas são enviadas para os registradores de deslocamento. Nos registradores de deslocamento as amostras são agrupadas em blocos de oito pixels e posteriormente armazenadas na memória.

O circuito controlador é o responsável pela temporização de todas as atividades na interface de aquisição e visualização. A base de tempo é dada por um sinal de relógio com um período igual ao tempo de duração de um pixel (aproximadamente 90ns). A geração dos sinais referentes a sincronização de vídeo e apagamento além da geração de endereços para a memória é feita por um chip controlador de regeneração de vídeo MC6845 da Motorola.

Este controlador já é bastante antigo, e embora seja bastante simples é adequado ao projeto em questão. Possui três categorias de sinais: de interface com o barramento do sistema e microprocessador, sinais de interface entre o controlador para memória de imagem e lógica do gerador de caractere, e sinais diretamente relacionados a interface com o monitor de vídeo. Os sinais de interface com o barramento do sistema permitem basicamente que se acesse os registros internos de programação de temporização do 6845. Os sinais de interface com a memória de imagem consistem de endereços para acesso à memória de imagem e para acesso ao gerador de caractere. Não é fornecida nenhuma lógica para resolução da disputa à memória entre processador e controlador. Esta deve ser implementada separadamente. O controlador 6845 fornece os principais sinais de interface com o monitor de vídeo tais como sincronismo vertical, sinal de habilitação de vídeo ("display enable"), responsável pelo



	PIN NAME	DESCRIPTION	TYPE
BUS INTERFACE	D0-D7	DATA BUS	BIDIRECI-NAL TRI-S
	/CS	CHIP SELECT	INPUT
	RS	REGISTER SELECT	INPUT
	R/W	READ/WRITE SELCT	INPUT
	E	ENABLE SYNCHRONIZATION SIGNAL	INPUT
	CLK	CHARACTEER RATE CLOCK	INPUT
	/RESET	RESET	INPUT
	VCC, VSS	POWER AND GROUND	INPUT
SCREEM MEMORY AND CHARACTER GENERATOR SIGNALS	MA0-MA13	SCREEM MEMORY ADDRES BUS	OUTPUT
	RA0-RA4	RASTER (SCAN LINE) ADDRESS SIGNALS	OUTPUT
CRT INTERFACE SIGNALS	HSYNC	HORIZONTAL SYNCHRONIZATION	OUTPUT
	VSYNC	VERTICAL SYNCHRONIZATION	OUTPUT
	DISPEN	DISPLAY (VIDEO) ENABLE	OUTPUT
	CURSOR	CURSOR ENABLE	OUTPUT
	LPSTB	LIGTH PEN STROBE	INPUT

fig.III.4 - Pinagem e sinais do controlador 6845.

apagamento do sinal de vídeo durante o retraço, sinal de habilitação de cursor e sinal de deteção de "light pen". Uma descrição dos pinos e os respctivos sinais pode ser vista na fig.III.4 e maiores detalhes podem ser encontrados nos manuais [17] ou em Kane [8].

Neste projeto o controlador é usado para gerar os sinais de sincronismo, de habilitação de vídeo e os endereços para a memória de imagem.

A operação do 6845 é controlada por uma série de registradores internos que permitem a programação das características de temporização dos sinais de vídeo definindo modos de operação. É possivel através da alteração do conteúdo dos registradores programar a duração dos sinais de vídeo bem como a sua posição no tempo, determinado as características da imagem a ser mostrada como por exemplo comprimento de linha (em número de pixels) quantidade de linhas, frequência de regeneração, etc.

Para atingir a resolução desejada de 512 por 512, o controlador 6845 deve operar no modo caractere e vídeo entrelaçado, pois as 14 linhas de endereço de memória (MA0-13) são suficientes para endereçar apenas 16k endereços. Assim, utiliza-se as linhas da lógica de endereçamento de caracteres (RA's) para atingir os 32k endereços necessários. Ainda há uma outra restrição, o registrador que determina a quantidade de linhas a ser mostrada no vídeo permite um máximo de 128 linhas de caracteres, o que força a utilização de duas linhas de endereçamento de caractere (RA's) ao contrário de apenas uma como seria natural. Dessa forma, do ponto de vista do controlador 6845, a memória de imagem fica organizada como se fosse formada por 128 linhas de 32 caracteres cada, com cada linha de caractere sendo formada por 4 linhas "raster". Cada caractere seria formado por uma matriz de 4 linhas por 8 colunas.

A operação do controlador 6845 é sincronizada por um sinal de relógio que determina a duração de um caractere que possui por sua vez um tempo de duração de oito pixels. Esse sinal de relógio determina, também, a taxa de acesso à memória de vídeo.

Como dito anteriormente, o controlador não provê nehuma forma de

resolução de conflito de acesso à memória de imagem. Também, de acordo com os modos de operação definidos para a operação do barramento de imagem, não é necessário nenhuma forma especial de arbitragem do barramento, com exceção dos modos 0 e 1. Nesses dois modos o processador hospedeiro tem permissão de acessar a memória de imagem ao mesmo tempo em que o controlador 6845 realiza a regeneração da tela. Para resolver os possíveis conflitos de acesso nessa situação utiliza-se dos relógio de caractere e de pixel para gerar janelas de acesso bem definidas nas quais ora o processador hospedeiro tem acesso à memória, ora o controlador.

A fig.III.5 mostra a relação de tempos entre os principais sinais de controle do circuito para arbitragem do barramento de imagem. Tomando por base o relógio de caractere, pode-se verificar que os sinais de endereços do 6845 tornam-se estáveis aproximadamente 160ns após a borda de descida do mesmo. Na primeira borda de subida do relógio de pixel, em seguida a estabilização dos endereços, o barramento é direcionado para o 6845 através de um multiplexador comandado pelo sinal **vídeo/bus**. Os dados da memória têm aproximadamente 180ns para ficarem estáveis e serem carregados nos registradores de deslocamento na borda de subida do sinal **shift/load**. Após o carregamento dos pixels nos registradores de deslocamento o barramento de imagem fica livre para o processador acessar a memória até o próximo acesso do controlador 6845. A duração da janela para o processador é de aproximadamente 540ns. Se o início do ciclo de acesso do PC começa durante a janela do 6845, o ciclo deve ser retardado pelo período de tempo necessário através do sinal "wait state" do barramento do PC.

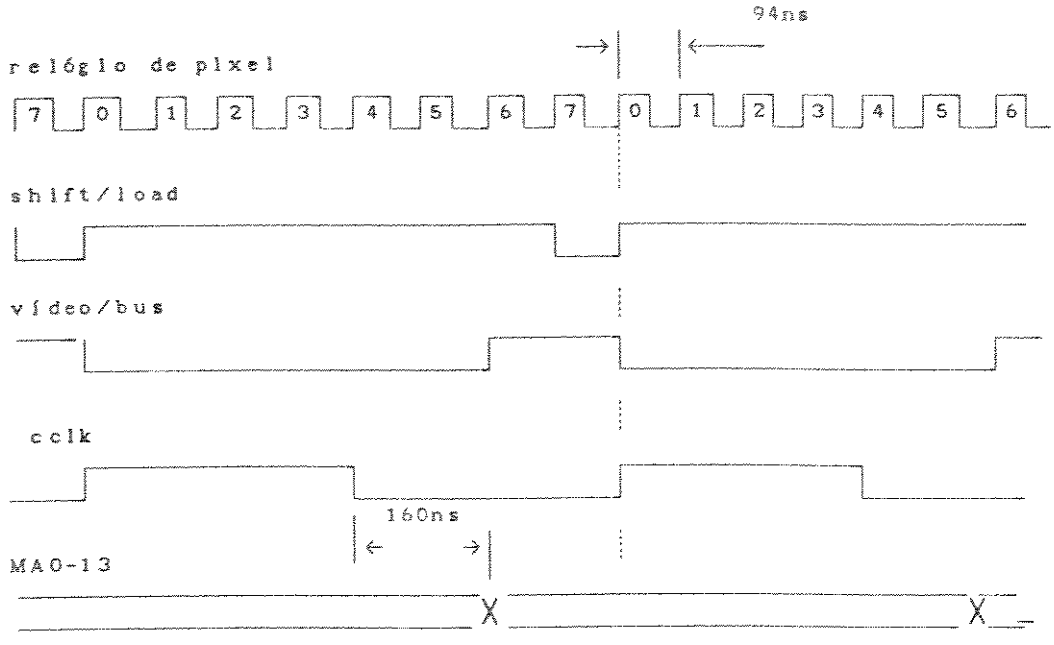


fig.III.5 - Diagrama de tempo dos principais sinais de controle para acesso à memória de imagem. CCLK é o relógio de caractere; SHIFT/LOAD é o sinal que carrega os pixels nos registradores de deslocamento (0) e comanda o deslocamento dos registradores (1); VÍDEO/BUS comanda o multiplexador que direciona o barramento de imagem para o processador (0) ou para o controlador 6845 (1); MA0-13,RA0-1 são as linhas de endereço do 6845.

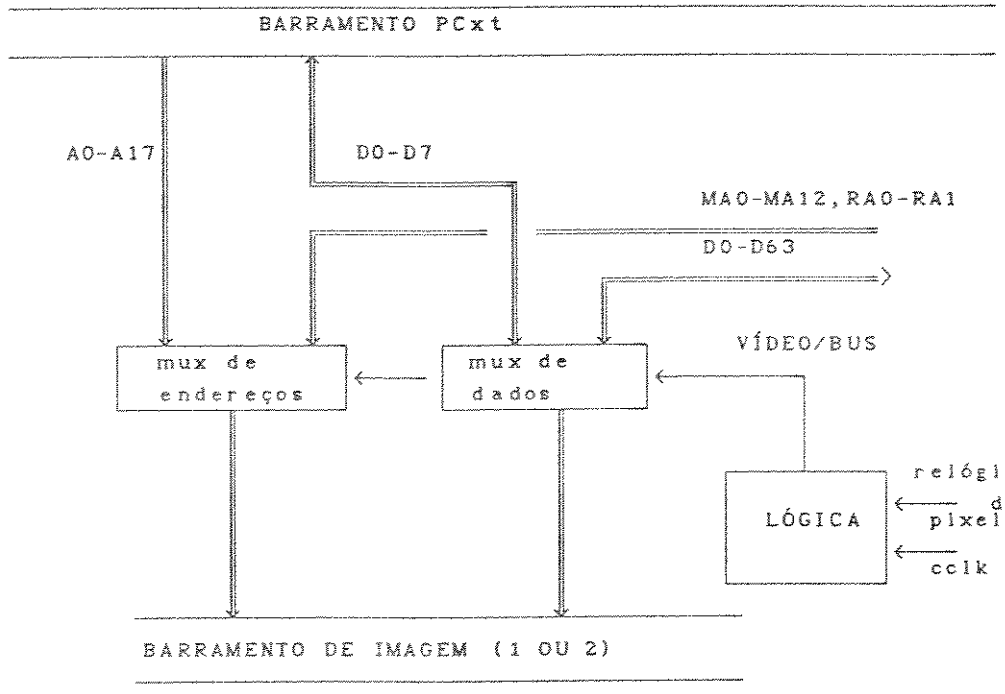


fig.III.6 - Esquema de arbitragem para acesso ao barramento de imagem pelo PCxt e controlador 6845.

III.5 - PROCESSADOR HOSPEDEIRO

O processador hospedeiro é qualquer processador de uso geral. Sua função é a de supervisionar e controlar os demais módulos do sistema e permitir a iteração com o usuário. Além disso, no caso da estação de trabalho pretendida, voltada para processamento de imagem, deve realizar o processamento de alto nível, ou seja, as funções de análise e interpretação, podendo ainda, na falta do processador de imagem realizar as operações de baixo nível.

Duas opções foram consideradas para a escolha do processador hospedeiro: um microcomputador do tipo PCxt e um sistema HOMUK. A adoção do microcomputador PC, para um primeiro protótipo deveu-se a existência de um grande número de fabricantes desse sistema no mercado, à disponibilidade de informações técnicas, e ao bom suporte proporcionado para o desenvolvimento de "software". Embora o desempenho proporcionado por um microcomputador tipo PCxt não seja compatível com o normalmente exigido para uma estação de trabalho profissional, a evolução desta família de sistemas, permite através do uso de modelos com processadores mais potentes atingir um desempenho razoável com poucas ou nenhuma mudança nas interfaces já desenvolvidas.

A implementação é realizada sobre uma expansão física do barramento interno do PC. Assim, embora o "frame buffer" e registros de controle e supervisão estejam mapeados no espaço de endereçamento lógico do processador, os elementos físicos não estão localizados no microcomputador. Em virtude do espaço de endereçamento do processador 8088 ser reduzido (apenas 1Mbyte), a interface de expansão limita o acesso ao barramento de imagem em blocos de 64K endereços com o restante sendo alcançado via registradores de endereço, isto é, o microprocessador pode endereçar somente 64kbytes de memória de imagem diretamente. Essa forma de endereçamento paginado permite salvar espaço no mapa de memória do microcomputador, sem alterar significativamente o desempenho do sistema. A descrição da interface de expansão é feita a seguir.

III.5.1 - Interface de expansão do barramento PC

A interface de expansão física do barramento do microcomputador PCxt permite a montagem do Sistema de Processamento de Imagem fora do gabinete do PC sem influenciar na operação normal do PC.

A interface é dividida em duas partes: uma de transmissão de sinais, localizada no PC e outra de recepção dos sinais, localizada no SPI. As duas são interligadas através de um cabo paralelo que conduz os sinais. A interface tem as funções de reforçar os sinais elétricos e realizar casamento de impedâncias com o cabo evitando reflexões e ruídos. Além disso, a interface controla os acessos através do barramento do PC ao SPI através do gerenciamento dos endereços. O diagrama em blocos da interface esta representado na fig.III.7.

Um circuito decodificador de endereços é responsável pela verificação de acessos do PC ao barramento de expansão. Se um acesso à memória de imagem é verificado insere-se um ciclo de espera para compensar atrasos provocados pelos "drives" e pelo cabo. O circuito de decodificação ainda é responsável pela localização da memória de imagem e de endereços de IO no mapa de memória do PCxt. Uma descrição completa da operação e das características da interface de expansão pode ser encontrada em Prado [18].

Uma implementação com interface para barramento VME poderá ser realizada futuramente. O principal atrativo dessa alternativa é a possibilidade de obtenção de um sistema com alto desempenho se considerado o potencial de multiprocessamento do sistema HOMUK.

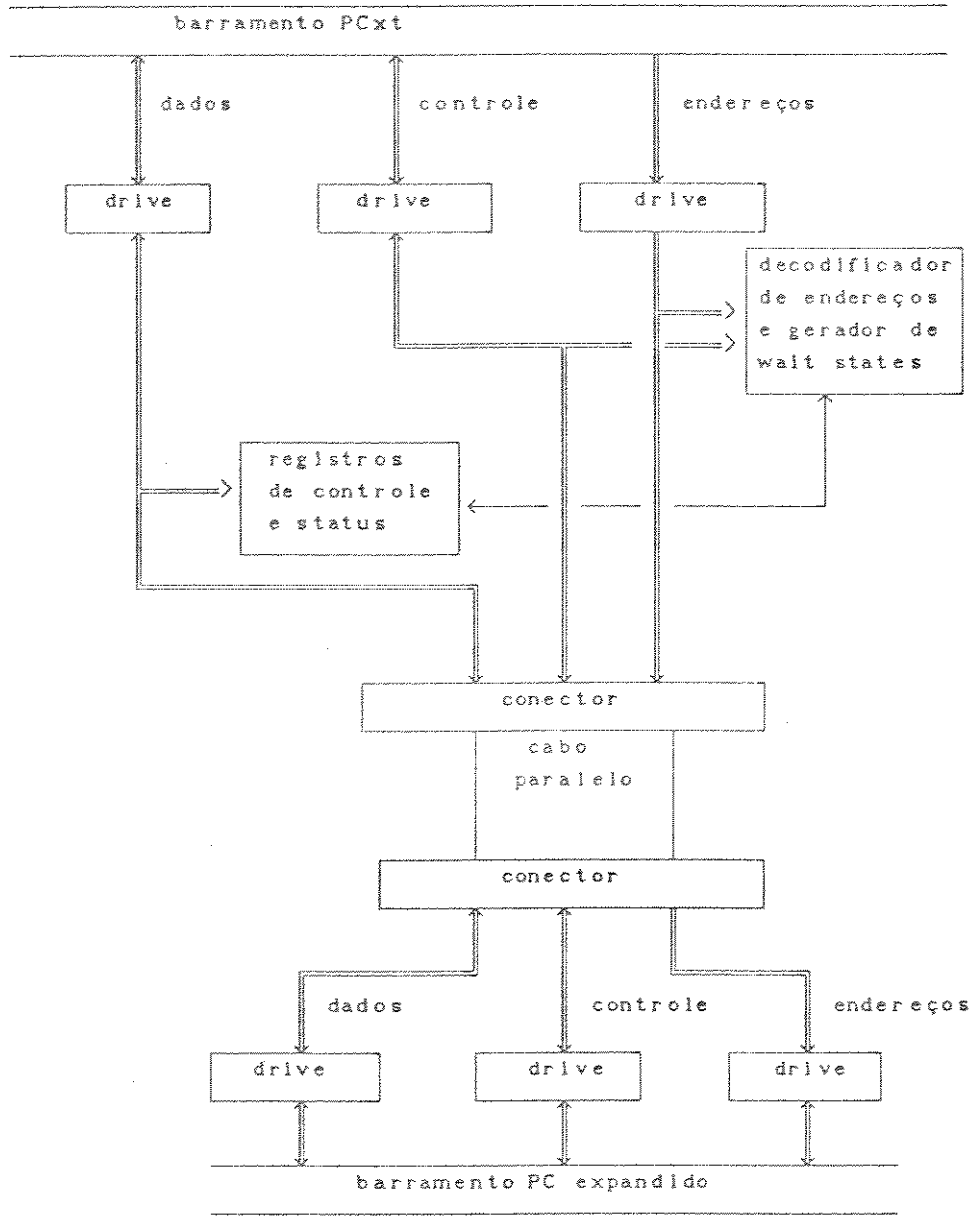


fig.III.7 - Reresentação da interface de expansão do barramento do PCxt.

III.6 - Comentários Gerais

O desenvolvimento deste projeto teve como norma o uso de componentes de tecnologia bem conhecida e de fácil aquisição no mercado nacional. Com exceção das memórias, todos os demais componetes são facilmente encontrados e constituem-se na sua maioria de componetes TTL. Existem dois motivos principais para a adoção desse procedimento. Um refere-se as dificuldades de importação de componentes existentes à época do início do projeto e o outro a problemas de estrutura de laboratório para desenvolvimento de montagem e testes.

Muitas das características do projeto refletem estes fatos. Assim, a frequência de operação do circuito não deve ser alta pois a montagem foi realizada em "wire wrap" aumentando a possibilidade de interferências de ruídos. A baixa frequência de operação também permite a utilização de componentes TTL e diminui a influências de atrasos e reflexões na fiação.

A utilização de dois barramentos com palavras largas garante um alta taxa de transferência de dados mesmo com uma baixa frequência de operação.

Entretanto, o número de "buffers" necessários para interfacear o barramento cresce bastante, assim como a potência dissipada e a área necessária nas placas. A grande quantidade de fios, bem como de conectores, torna-se uma fonte de falhas [19] principalmente no caso da montagem do protótipo em "wire-wrap".

Para minimizar os problemas advindos do grande número de sinais nos barramentos existem duas opções: diminuir a largura da palavra ou usar apenas um barramento ao invés de dois.

A diminuição da largura do barramento provoca a diminuição da taxa de transferência de dados. Esta opção é interessante desde que a diminuição da taxa de transferência seja de ordem tal que não inviabilize a operação dos processadores em tempo real. Como visto anteriormente na seção III.2.3, existe na especificação atual do projeto margem de segurança para que esta

opção seja adotada.

O estrutura com apenas um barramento também é possível mas causa maiores dificuldades no projeto. Neste caso, também o número de linhas de endereço é reduzido. Como consequência torna-se impossível endereçar simultaneamente dois pixels. Dessa forma operações que necessitem essa características, como por exemplo a aquisição de duas imagens simultaneas devem utilizar artifícios que separem os dois acessos no tempo. Isto pode ser feito através do uso de registradores temporários. No caso da sobreposição das operações de busca de operando, execução e armazenamento de resultados é necessário o uso de memórias locais tipo "interleaving".

As características das interfaces de aquisição e visualização preenchem os requisitos de um sistema com estes propósitos. Algumas características, entretanto, podem ser adicionadas, como por exemplo a possibilidade de aquisição continua de imagens, permitindo a visualização ao vivo da imagem sendo adquirida, ou ainda, a possibilidade de aquisição de imagens de fontes cujos sinais sejam de vídeo composto, ampliando as alternativas de fontes de imagem.

Capítulo IV

Um Processador Pipeline para Deteção de Bordas

IV.1 - INTRODUÇÃO.

Uma característica própria da visão computacional é o volume de processamento exigido. Contribuem para isso a quantidade de dados presentes desde o início do processo e também o tipo das operações aplicadas sobre eles. Um exemplo comum é a operação de convolução sobre uma imagem qualquer; para cada um dos pontos da imagem serão repetidas operações de multiplicação e acumulação envolvendo vários pixels vizinhos ao ponto de interesse. Os processos costumam ser, dessa maneira, extremamente morosos quando são utilizados processadores convencionais, dificultando por exemplo aplicações que necessitem respostas rápidas ou tempo real. Sob esse aspecto é inevitável a busca de soluções para a redução do tempo de resposta dos sistemas de visão.

Em virtude dos algoritmos estarem de certa forma definidos, a solução para a aceleração do processamento é produzir sistemas computacionais com maior capacidade de processamento. Os algoritmos para processamento de imagem em baixo nível apresentam características tais como grande repetibilidade e independência de execução (sobre diferentes dados) que os torna especialmente adequados para implementação em arquiteturas paralelas (SIMD, MIND, "pipelines", etc). Em particular, as operações efetuadas sobre vetores numéricos se prestam a execução em máquinas do tipo SIMD. Estas máquinas por sua vez podem ser implementadas na forma de "pipelines" explorando ao máximo a repetibilidade das operações. Uma grande diversidade de arquiteturas com a finalidade de explorar o paralelismo natural dos algoritmos de baixo nível do processamento de imagem têm sido propostas e implementadas. A utilização de processadores algoritmicamente orientados também se mostra adequado a obtenção de alto desempenho em aplicações específicas. Com a especialização do processador para a execução de um algoritmo, ou de uma classe de algoritmos, é possível desenvolver cada parte do circuito para executar a sua função de forma ótima. Obviamente a especialização diminui a flexibilidade de aplicação dos processadores, mas permite a obtenção do desempenho desejado, além de uma vantajosa redução dos custos. A especialização dos processadores também

permite a exploração de diversas características que os tornem especialmente adequados à implementação em VLSI.

Este capítulo descreve uma proposta de implementação de um processador "pipeline" para extração de bordas de objetos em uma imagem multitonal, cujo objetivo é acelerar as etapas de baixo nível do processamento utilizado para o reconhecimento de padrões em aplicações industriais. A obtenção das bordas deve ser realizada em tempo real, ou seja, no tempo de aquisição de uma imagem. As operações de alto nível são executadas em um microcomputador, após a redução do volume de informação pelo processador "pipeline" dedicado.

O processador "pipeline" proposto engloba, também, o processamento de convolução utilizando núcleos de dimensão 3X3, bastante difundidos em visão computacional. Dessa forma, embora pretendendo uma aplicação específica, é possível derivar a partir deste projeto, com pequenas modificações, um processador para toda uma classe de algoritmos.

Nas próximas secções é apresentada uma visão da estrutura algoritmica da aplicação pretendida bem como uma avaliação das necessidades da utilização de "hardware" específico. Na sequência é realizado um estudo das possíveis estruturas do processador dedicado, procurando analisar as relações existentes entre custo-flexibilidade-desempenho.

IV.2 - A Aplicação - Deteção de bordas em uma imagem.

O objetivo deste trabalho é desenvolver uma arquitetura que suporte o desenvolvimento de "hardware" especializado, com alto desempenho, para o tratamento de imagem, dirigido a visão robótica ou reconhecimento de padrões com aplicações industriais. Para validação da arquitetura proposta desenvolveu-se um projeto que limitou-se ao emprego de algoritmos simples mas suficientemente robustos e que empregam a deteção de bordas dos objetos da imagem para obter as características que permitam a identificação dos objetos de interesse. Tais algoritmos são clássicos e podem ser encontrados em livros sobre o assunto [10, [20], [21] e por não ser o objetivo principal do trabalho não serão comentados aqui. Algumas etapas do processo, que podem ser implementadas em "hardware" dedicado para melhoria de desempenho dedicado são apresentadas na fig.IV.1.

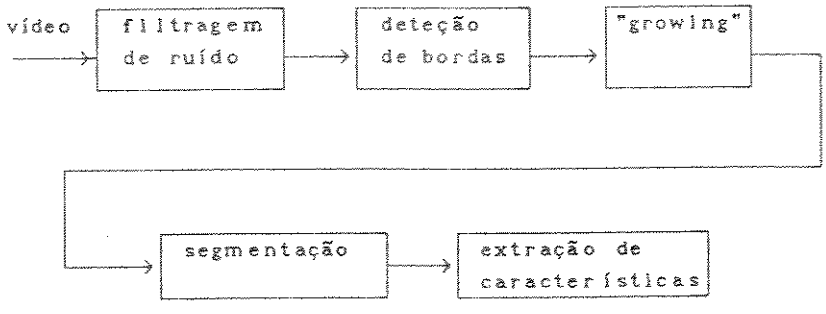


fig.IV.1 - Etapas do processo de reconhecimento de padrões.

Será proposto, neste trabalho, a implementação em "hardware" de algoritmos de deteção de bordas baseados nos operadores de Sobel. Esta escolha tem como principais motivações o fato destes operadores serem robustos, apresentarem certas facilidades para implementação em "hardware", além de serem indicados para isto por serem computacionalmente custosos.

Por outro lado, os operadores de Sobel incluem uma operação extremamente difundida no processamento de imagem que é a convolução espacial utilizando núcleos 3X3. Assim, ao mesmo tempo que se obtém um projeto com aplicação específica (deteção de bordas), também se obtém a solução para toda uma gama de problemas comuns a este tipo de processamento. O núcleo de convolução pode ser utilizado também para filtragem de ruídos, suavização da imagem, etc.

IV.2.1 - A relação hardware-algoritmo.

A deteção de bordas através dos operadores de Sobel envolve as etapas mostradas no diagrama da fig.V.2. Estas etapas podem ser implementadas diretamente em "hardware" dedicado com a mesma organização apresentada pelo diagrama, produzindo então uma estrutura "pipeline".

O algoritmo envolve a aplicação de dois operadores 3X3 sobre a imagem de entrada. Eventualmente a imagem de entrada pode ter sido pré-processada para a eliminação de ruídos ou outros meios de reconstrução de imagem como mostrado na fig IV.1. Este processamento também pode ser realizado utilizando núcleos de convolução de dimensão 3X3.

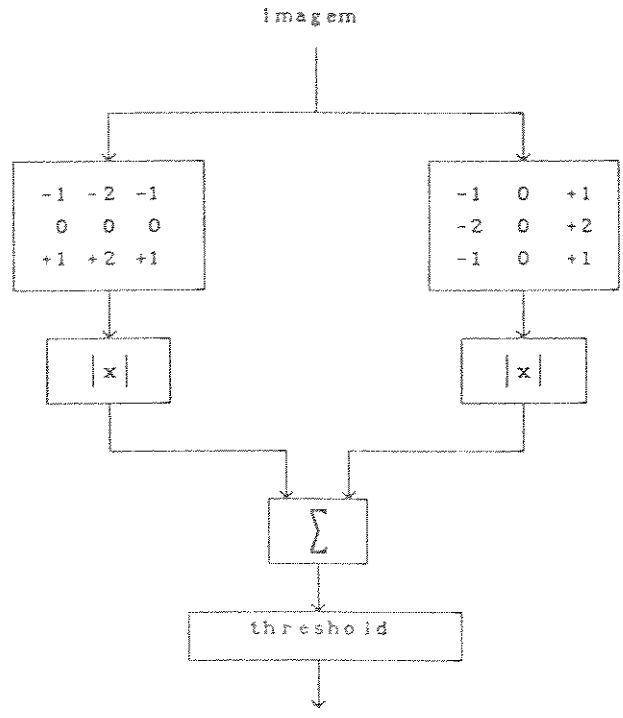


fig.IV.2 - Fluxograma da aplicação do operador de Sobel para a deteção de bordas em uma imagem.

O fluxograma da fig.IV.2 é dividida em quatro etapas que devem ser executadas sequencialmente. A convolução, realizada em duas etapas paralelamente, para deteção das bordas verticais e horizontais, a extração do módulo dos resultados da convolução (a convolução é baseada na derivada da função intensidade luminosa, sendo possíveis valores negativos) a combinação dos resultados das duas convoluções através de uma soma e finalmente a comparação dos resultados com um valor pré-determinado que define se aquela variação de luminosidade determina uma borda ou não. Após a obtenção das bordas a imagem torna-se binária, com os pixels que determinam bordas recebendo valor "1" e pixels que não representam borda recebendo valor "0". Assim, o processamento posterior fica simplificado pela sensível diminuição do volume de dados, pois cada pixel pode ser representado por um único bit.

Cada uma das etapas da deteção de bordas pode ser implementada através de circuitos dedicados, formando um circuito "pipeline". As etapas de módulo, soma e "thresholding" podem ser implementadas em um único estágio, pois são constituídas de uma única operação, por exemplo utilizando "look-up-tables". A operação de convolução, entretanto, é constituida de várias suboperações que devem ser executadas em paralelo para permitir a manutenção do fluxo de dados para os outros estágios. Ainda, algumas das sub-operações da convolução podem ser executadas por um "pipeline", alongando o número de estágios total.

Para a verificação detalhada das operações envolvidas na convolução com o núcleo 3X3, é apresentado um programa em pseudo- código para sua execução (fig.IV.3).

```
/*
convolução de um núcleo 3X3 sobre uma imagem mxn
*/

int I[m,n], K[3,3]; /* I[m,n] é a matriz imagem
 K[3,3] é o núcleo de convolução
 */

inicializa s[m,n] = 0;          /* s é uma matriz que recebe a
                                   imagem resultado
                                   */


for( x=0; x <= (m-1); x++ )
{
     for( y=0; y <= (n-1); y++
     {
          for( i=-1; i <= 1; i++ )
          {
               for( j=-1; j <= 1; j++ )
               {
                    s[x,y] = s[x,y] + I[x+i, y+j] * K[i+1, j+1];
               }
          }
     }
}
```

fig.IV.3 - pseudo-código da convolução 3X3.

Primeiro são definidas duas estruturas de dados na forma de matrizes I[m,n] e K[3,3] e que representam respectivamente uma imagem de dimensão mxn e um núcleo de convolução qualquer de dimensão 3X3.

Pelo programa acima, embora simplificado, pode-se verificar que as principais operações necessárias são soma e multiplicação, além é claro, do acesso aos dados de imagem de entrada e do armazenamento da imagem resultado. O restante das operações existentes realiza o controle da estrutura do programa.

A estrutura apresentada pelo programa se adequa a implementação em processadores "pipelines" vetoriais pois são executadas diversas operações sequencial e repetidamente sobre um vetor de dados (matriz imagem). Assim duas operações são executadas sequencialmente, multiplicação e soma, com o resultado da multiplicação sendo usado para a soma. Esta sequência de operações é executada repetidamente por um longo conjunto de "loops". Por outro lado verifica-se que este programa possui um controle simples sem quebra de "loops" ou desvios o que permite a implementação do controle diretamente em "hardware", sem muitas dificuldades e com relativa eficiência.

IV.2.1.2 - Requisitos para processamento em tempo real.

Um processador dedicado simples poderia conter um módulo para multiplicação, outro para a soma e outro para a acumulação. Como dito anteriormente, a ordem em que as operações acontecem é sempre a mesma e portanto, é interessante a adoção de uma estrutura "pipeline" para acelerar a execução do programa. A fig.IV.4 representa um processador deste tipo.

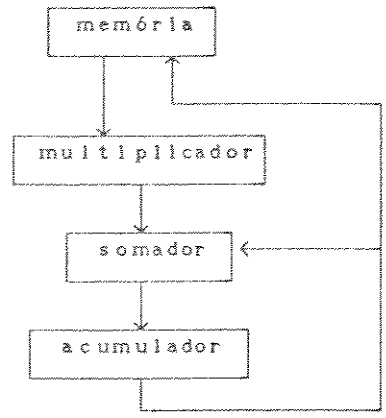


fig.IV.4 - Unidade aritmética de um processador "pipeline" simplificado para executar as operações de multiplicação e soma .

Para que este processador efetue a deteção em tempo de aquisição de imagem, isto é, em 30ms, terá que operar na própria taxa de vídeo, ou seja, o tempo de produção de um novo resultado (pixel), considerando uma imagem de 512x512 pontos, deverá ser de ~ 100ns (este valor foi aproximado apenas para simplificação dos cálculos e valores obtidos). Entretanto, para que esta taxa seja obtida é necessário que para cada resultado apresentado o processador realize pelo menos nove acessos à memória da imagem e mais nove acessos a memória em que estão os coeficientes do núcleo. Isto leva a um taxa de transferência de dados da memória para o processador de aproximadamente 180Mbytes, considerando-se valores inteiros de 8 bits para representação dos pixels e coeficientes. Para processar um pixel em 100ns, o processador deve realizar 9 multiplicações e 9 somas nesse período. Como as operações de soma e multiplicação são sobrepostas, o período de execução de cada operação deve ser de 100/9 = 11,1ns. Evidentemente, é um período de execução muito pequeno, tornando difícil a implementação de um processador como esse. Mesmo que este raciocínio seja simplificado, pois se está desconsiderando busca e decodificação de instruções, percebe-se que para realizar processamento em tempo real é necessário a adoção de outras estruturas mais eficientes.

É interessante notar que embora a estrutura do processador fictício apresentado acima seja bastente simplificada, ela ilustra os problemas

enfrentados de forma bastante clara. Além disso, esta estrutura se aproxima bastante da estrutura de alguns processadores digitais de sinal, como os da família TMS320 da Texas Instruments [22], [23], e que tem sido empregados em diversos sistemas para aceleração de processamento de imagem [24]. O ciclo de execução desses processadores está situado em torno de 100ns e por isso não permite operação em tempo real.

A observação do programa anterior (fig.IV.3) leva a conclusão que cada pixel é acessado pelo menos nove vezes durante a execução. O mesmo acontece para os coeficientes do núcleo. Uma maneira de minimizar essa excessiva quantidade de acessos a memória principal é o emprego de memórias ou registros locais. Pode-se assim manter os coeficientes armazenados em uma memória local diminuindo pela metade a frequência de acesso à memória principal. Da mesma forma, após acessar cada pixel, pode-se mantê- lo em uma memória local até que ele seja utilizado todas as vezes que for necessário, mas esta técnica só é aplicável quando o número de operações intermediárias é reduzido. A taxa de acesso à memória será reduzida então em nove vezes, igualando-se a taxa de vídeo que é de aproximadamente 10Mbytes/s. Entretanto, permanece o problema da velocidade de execução que não se altera com a diminuição da taxa de acesso a memória.

A solução do problema só é alcançada com a exploração de paralelismo de programa que pode ser feito pelo uso de replicação de processadores ou pela exploração do paralelismo temporal na execução das instruções, ou ainda de ambos.

Nas próximas seções é apresentado um processador dedicado para a execução do algoritmo de convolução com núcleo 3X3 e que emprega paralelismo em diversas formas. Após a apresentação da sua estrutura funcionamento e desempenho, outras soluções possíveis são analisadas a partir da mesma estrutura básica, mas com outras filosofias de distribuição de dados e tarefas.

IV.3 - O PROCESSADOR "PIPELINE" PARA CONVOLUÇÃO

O diagrama em blocos do módulo básico do processador dedicado para a convolução é mostrado na fig.IV.5.

Existem cinco blocos principais no processador: um registrador de deslocamento de entrada, um multiplicador, um somador, uma FIFO, e um registrador de deslocamento de saída.

O registrador de deslocamento da entrada tem a função de alinhar os pixels na entrada dos multiplicadores. O acesso do processador à memória de imagem é feito em blocos horizontais de 8 pixels. A sequência na qual esses blocos são acessados, relativamente a organização da memória, é a mesma da varredura de um feixe de elétrons em um TRC, isto é, inicia-se na parte superior esquerda da imagem, varrendo linha por linha até a parte inferior direita. Devido a organização da memória, o endereçamento é linear. O objetivo de um fluxo linear e sequencial de acesso a memória de imagem é estabelecer, com pequenas diferenças, o mesmo fluxo de dados existente em um processamento "on line", com a aquisição de imagens através de um digitalizador, sem uso de um "buffer" intermediário e assim anterciparmos alguns parâmetros para um projeto deste tipo. A diferença que existe entre as duas abordagens, sem e com "buffer", está na presença ou ausência, respectivamente, de uma sicronização com o processo de aquisição da imagem. A presença de um "buffer" também permite algumas alternativas na estrutura do sistema pois o fluxo de dados não está amarrado. Estes aspectos serão estudados posteriormente. É importante, também, que na abordagem usada o processador realiza uma única operação completa sobre toda a imagem, do começo ao fim, permitindo um fluxo constante e uma maior eficiência na sua utilização.

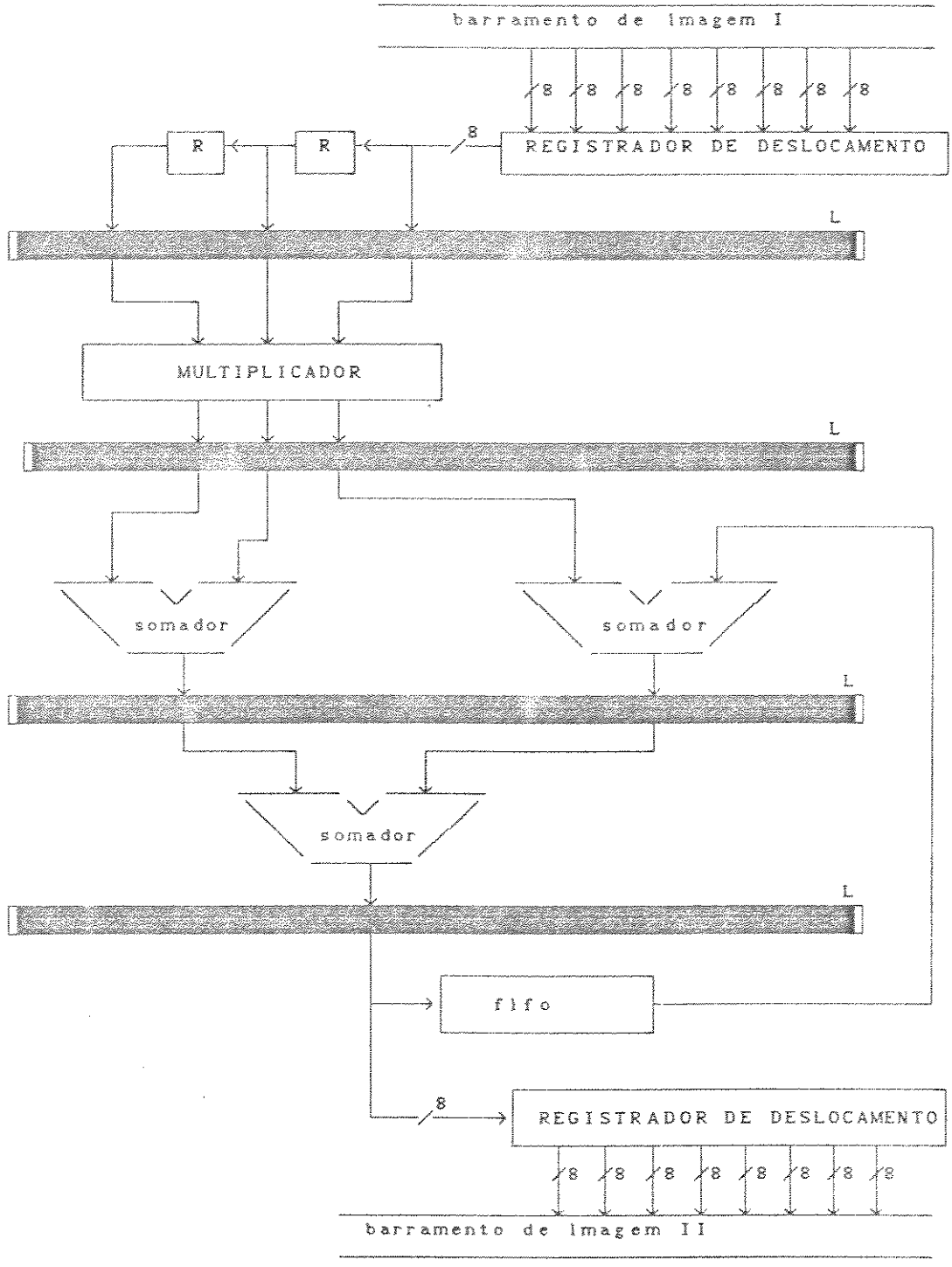


fig.IV.5 - Estrutura básica de um processador pipeline para o operador 3x3.

O processador possui quatro estágios que desempenham as seguintes funções:

1 - Registrador de deslocamento: recebe o bloco de pixels da memória e realiza o alinhamento dos pixels para a entrada do estágio seguinte. O objetivo é entrar com os pixels na ordem correta para a multiplicação com os coeficientes do núcleo. Como são realizadas três multiplicações em paralelo, a cada nova operação os pixels a serem inseridos são outros e devem ser trocados. A função deste registrador pode também ser entendida de duas maneiras. Uma como sendo similar a desempenhada por memórias locais e a outra como uma chaves "crossbar" para a distribuição de dados entre um grupo de processadores paralelos.

2 - Multiplicadores: realizam a multiplicação simultânea de uma linha do núcleo de convolução, isto é, cada três pixels que entram neste estágio são multiplicados pelos respectivos coeficientes de uma linha do núcleo de convolução. Sempre que se está processando uma nova linha os coeficientes são trocados.

3 - Somadores: realizam a somatória presente na formulação da convolução. A somatória é feita entre os valores que entram no estágio e o conteúdo dos registradores da fifo. Cada endereço da fifo armazena um resultado parcial que corresponde a soma dos três primeiros produtos de pixel pelos coeficientes e após o processamento da segunda linha corresponde a soma de seis produtos. O resultado final obtido após o processamento da terceira linha não é armazenada na fifo uma vez que deve ser armazenado na memória principal.

4 - FIFO: Sua função é armazenar os resultados intermediários obtidos a partir da computação parcial de cada linha de convolução. Possui 512 posições correspondentes ao tamanho de uma linha de imagem.

5 - Registradores de deslocamento de saída: Realizam o empacotamento do resultado final em blocos de 8 bytes para o armazenamento na memória. De uma certa maneira ele realiza a operação inversa ao registrador de deslocamento da entrada.

IV.3.1 - Variações com a estrutura "pipeline" 3x1

Até o momento foi considerada uma estrutura particular para a computação de núcleos de dimensão 3x3, que realiza três multiplicações em um único ciclo de relógio. Foi observado, também, na introdução deste trabalho que para a obtenção de processamento de uma imagem próximo ao tempo de visualização de uma imagem (60ms) é necessário a realização de no mínimo 9 multiplicações em 100ns, ou seja, o intervalo de tempo em que é gerado um novo pixel. É evidente que a estrutura 3x1 necessita, para atender esse requisito, operar a uma frequência de relógio cerca de três vezes maior que a frequência de geração de pixels. Numericamente falando, é necessário trabalhar com uma frequência de no mínimo 30Mhz. Esta frequência reduz as opções de implementação, no que se refere a escolha e emprego de tecnologia.

É necessário, portanto, que se busquem alternativas na estrutura a fim de afrouxar os requisitos de frequência de operação. Um primeiro passo é fixar a frequência de operação em um valor compatível com a tecnologia disponível. Nesse sentido, uma escolha óbvia é a própria frequência de amostragem do pixel, isto é, 10Mhz. Sem entrar, no momento, em detalhes da implementação de cada estágio, esta frequência é suficientemente baixa para permitir o emprego das tecnologias mais comuns e comercialmente acessíveis, sem restringir em demasia os parâmetros do projeto. O segundo passo é verificar quais alterações podem ser adotadas na estrutura, ou no emprego da mesma, para que se obtenha o desempenho exigido com a mesma frequência de operação proposta.

Pode-se verificar facilmente que o emprego de três desses processadores (ou mais) em paralelo permite atingir o desempenho proposto. Para viabilizar essa proposta é necessário dividir adequadamente dados de imagem entre os processadores. Primeiro, não se deve dividir a imagem entre um número muito grande de processadores para que não haja disputa pelo acesso ao barramento. Segundo, alterar ao mínimo a estrutura dos processadores originais. Terceiro, reduzir ao máximo as fronteiras entre áreas de influência de processadores para que se reduza ao máximo a multiplicidade de acessos à memória nas fronteiras.

O limite de processadores para que não ocorra disputa pelo acesso ao barramento é facilmente estabelecido supondo que cada processador realize somente um acesso a cada pixel e que cada acesso tenha um ciclo de leitura idêntico ao ciclo do processador. Assim, é possível um máximo de oito processadores pois cada ciclo de acesso é de 100ns e a cada ciclo são acessados oito pixels. Cada processador levará 800ns até que seja necessário fazer um novo acesso à memória. Entretanto, este número de processadores é dispensável um vez que a quantidade mínima para atingir o desempenho desejado é de três processadores. Note-se também, que aqui se está supondo um acesso sincronizado, ou que se estabeleça um árbitro para acesso ao barramento, de modo que cada um dos processadores acesse a memória sem interferir no acesso de outro processador.

A redução da área de fronteira é obtida pela simples atribuição a cada processador de uma única porção da imagem e pelo uso do menor número possível de processadores. Por outro lado, para modificar ao mínimo a estrutura dos processadores basta dividir a imagem em faixas horizontais. Neste caso, a única alteração necessária deverá ser feita no fluxo de controle, com a diminuição das dimensões originais da imagem para cada processador. Assim, as dimensões da imagem para cada processador, originalmente nxm, passam a ser nx(m/k), onde k é número de processadores usados. Esta abordagem possue como vantagem o fato de poder-se trabalhar com diversos processadores, não necessariamente sincronizados, apenas utilizando um árbitro para acesso ao barramento.

Pode-se adotar também, uma divisão por faixas verticais que, entretanto, forçará um alteração nas dimensões da FIFO, bem como no fluxo de controle. Ou pode-se dividir a imagem em janelas para cada um dos processadores, com as mesmas consequências.

De um modo geral, ao dividir-se a imagem em áreas específicas de atuação para cada processador cria-se uma fronteira entre essas áreas. Como o algoritmo é baseado no processamento de uma área em torno de um ponto, surge nos pontos de fronteira uma situação semelhante aos pontos que estão na borda da imagem. Naquele caso é criada uma moldura de zeros para se calcular o núcleo (no caso específico deste trabalho), mas na região de fronteira entre duas áreas adjacentes é necessário que os pixels situados nas bordas sejam partilhados por todos os processadores com influência, criando um local de sobreposição de regiões. Como neste sistema a imagem é levada a cada um dos processadores por um barramento comum, não existem grandes dificuldades para que eles recebam os dados de borda em modo "broadcast", isto é, vários processadores recebem o mesmo dado simultaneamente. No caso da utilização de uma arquitetura diferente, na qual a distribuição da imagem ocorre através de mensagens por exemplo, é necessário que haja troca de imformações entre processadores adjacentes, provocando um sobrecarga de comunicação.

Uma segunda alternativa leva em conta uma atuação sincronizada de três processadores, cada qual atuando sobre uma etapa diferente do processo, mas sobre os mesmos dados. Cada processador irá produzir resultados de linhas diferentes e intercaladas. Em outras palavras, num determinado instante, com a mesma linha da imagem como entrada, um processador estará multiplicando a primeira linha do núcleo, outro a segunda linha do núcleo e o último a terceira linha do núcleo. Consequentemente, o processador que está utilizando a terceira linha do núcleo está também produzindo um resultado. Além disso, a cada nova linha de entrada, os papéis de cada processador são trocados entre si, isto é, o processador que processava a primeira linha do núcleo passa a processar a segunda, o processador que processava a segunda linha do núcleo passa a processar a terceira e o processador que processava a terceira linha do núcleo passa a processar a primeira, num processo sequencial e cíclico.

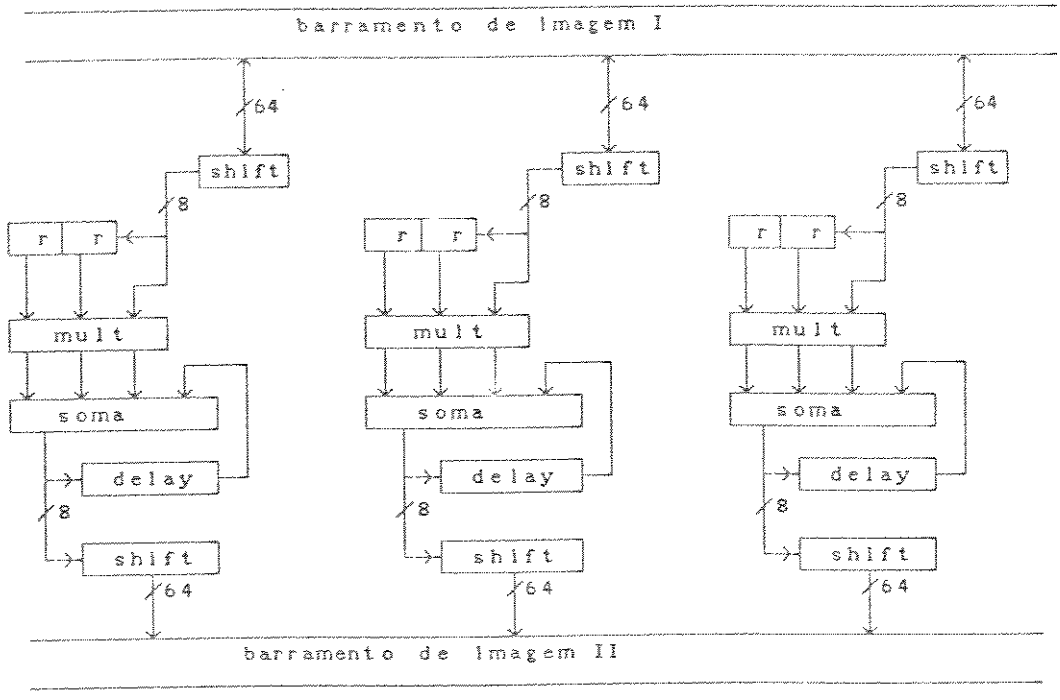


fig.IV.6 - Três processadores intercalados.

Num processo evolutivo é possível derivar um novo processador a partir deste arranjo com três processadores sincronizados, simplificando de uma maneira geral o circuito e o controle envolvidos. A descrição deste novo processador é feita nas secções seguintes.

Com o mesmo processador analisado, até o momento, não é possível obter um número muito maior de alternativas além das que foram apresentadas aqui.

IV.4 - UMA ESTRUTURA ADAPTADA: O PROCESSADOR 3x3

Baseado na última estrutura proposta, com três processadores trabalhando em paralelo e sincronizadamente, pode-se derivar um novo processador que atenda os requisitos de desempenho desejados e ao mesmo tempo obtendo uma simplificação na estrutura geral.

A idéia principal embutida no processo de obtenção deste novo processador consiste na especialização de cada uma das partes. Assim, da estrutura anterior nota-se que cada processador executa intercaladamente as mesmas funções que os outros dois. No caso, a função referida é a multiplicação de três pixels de imagem por três coeficientes do núcleo. A cada novo ciclo de relógio, cada um dos processadores tem como coeficientes para multiplicação uma nova linha do núcleo. É possível simplificar o circuito se cada processador realizar sempre multiplicações por um mesmo coeficiente.

Entretanto, torna-se agora necessário, que cada processador, após multiplicar um grupo de coeficientes de um linha de imagem por um linha do núcleo, comunique o resultado para o processador seguinte. Este por sua vez, efetuará a mesma operação com os pixels da linha seguinte, somará este resultado com o valor passado pelo processador anterior e então transfere a soma para o processador seguinte. O terceiro processador realiza então as mesmas operações com a terceira linha, obtendo o resultado final após somar o resultado obtido nele com o resultado proveniente do segundo processador. Como a operação sobre cada pixel de uma mesma coluna mas linhas diferentes ocorre após um atraso de 512 ciclos de relógio, a comunicação entre dois processadores adjacentes deve possuir uma linha de atraso correspondente para que as operações entre os processadores sejam sincronizadas corretamente. Esse atraso é efetuado pela mesma FIFO do processador 3x1 originalmente proposto, onde a realimentação para um estágio anterior é substituída pela conexão com o processador seguinte. Na fig.IV.7 é apresentado o diagrama completo do novo processador, que será referido a partir deste ponto como processador 3x3.

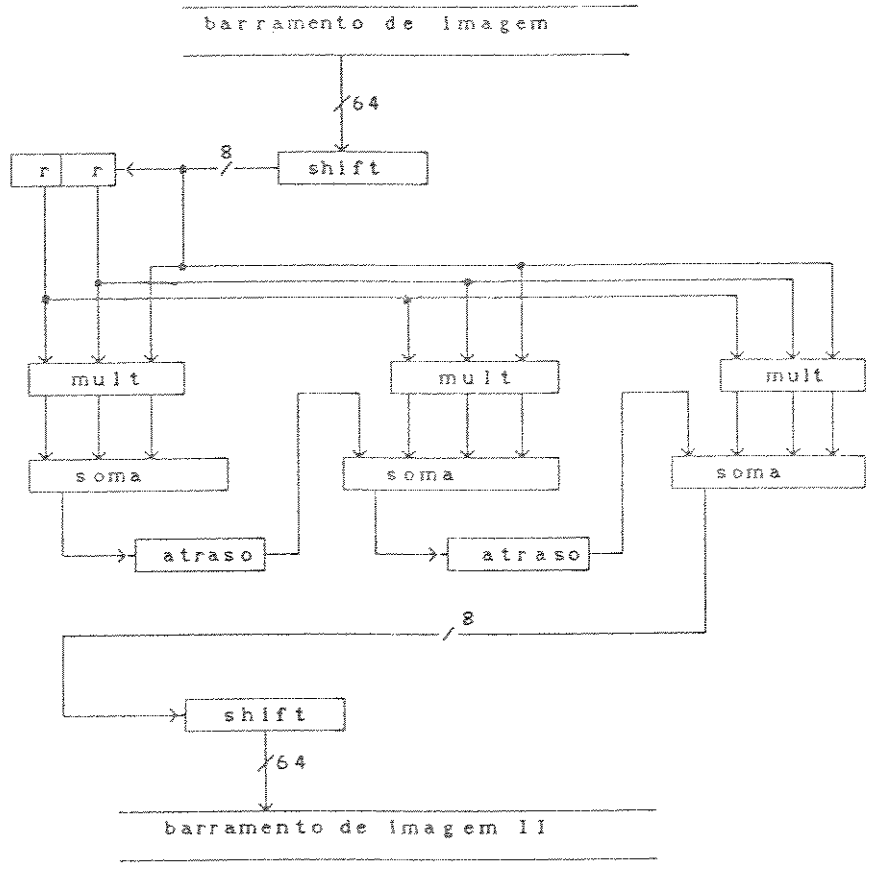


fig.IV.7 - Processador 3x3.

Verifica-se facilmente que esta nova configuração para o processador permite as seguintes simplificações:

1 - o número de FIFOs foi reduzido de 3 para 2;

2 - a largura de uma delas é reduzida pois a largura dos operandos é menor nas etapas iniciais do processamento;

3 - o número de registradores de deslocamento é reduzido de 3 para 1, tanto na entrada como na saída;

4 - os multiplicadores são substituídos por ponderadores;

5 - o controle torna-se mais simples pois não é necessário efetuar nenhuma troca de função e o "pipeline" é totalmente linear.

Com relação ao controle do processador é interessante notar que algumas funções de controle que eram explícitas no processador 3x1 foram eliminadas e estão embutidas na própria estrutura. Um exemplo é a necessidade do processador 3x1 ter de acessar três vezes a mesma posição de memória para multiplicar um mesma linha da imagem por três linhas do núcleo, uma vez que só é possível realizar três multiplicações simultâneas. Assim, o circuito de controle deve providenciar a repetição do acesso as linhas da imagem. No processador 3x3 esse controle não é necessário já que cada linha da imagem é acessada apenas uma vez.

IV.5 - Comentários Gerais

Neste capítulo estudou-se a necessidade de projeto de processadores especializados para execução de alguns algoritmos de processamento de imagem em tempo real, em especial, aqueles que utilizam a operação de convolução espacial. Foi proposto a partir desse estudo algumas arquiteturas para um circuito processador para a execução de núcleos de convolução de dimensão 3x3.

O projeto dos processadores especializados busca explorar várias características, próprias do processamento de imagem, capazes de permitir um alto desempenho na execução dos programas, muito difícil de ser obtido através de processadores de propósito geral ou mesmo de processadores progamáveis orientados para processamento digital de sinais. Entre as principais características exploradas ou analisadas pode-se citar: paralelismo temporal dos algoritmos, paralelismo espacial das imagens.

O paralelismo temporal é explorado através da utilização de uma arquitetura "pipeline" para os processadores. Assim, a alta repetibilidade de operações, proporcionada pelos algoritmos mais comuns de processamento de imagem, é aproveitada ao máximo, permitindo uma otimização na execução dos programas.

O paralelismo espacial é explorado dentro de cada processador através da realização de diversas operações sobre diversos pixels simultaneamente. Neste caso, o paralelismo espacial é do tipo vizinhança, isto é, refere-se a um grupo de pixels situados próximos uns dos outros e que são necessários para a computação dos resultados. A distribuição de diferentes regiões da imagem por entre diversos processadores foi também analisada e caracteriza a exploração do paralelismo espacial de regiões.

O processador básico para processamento do núcleo de convolução 3x3 pode ser usado para qualquer operação envolvendo tal tipo de operador. A associação de diversos destes processadores em "pipeline", cada qual executando operações

distintas, permite a acelerar de forma substancial a velocidade de processamento de um sistema voltado para o reconhecimento de padrões.

Sem muitas dificuldades, apenas pela extensão dos conceitos apresentados, é possível obter novos processadores para núcleos de convolução de maior dimensão, aumentando ainda mais as áreas de aplicação. Uma outra possibilidade bastante interessante é a modificação do circuito básico de maneira que seja possível a associação de vários circuitos básicos para o processamento de um núcleo maior.

Os processadores propostos oferecem ainda a possibilidade de operarem à taxa de vídeo. Com isso, é possível operar sobre os dados obtidos diretamente de um digitalizador, diminuindo ainda mais o tempo de resposta.

No capítulo V, cada um dos estágios do processador proposto é analisado com relação as possibilidades de implementação, comparando-se as alternativas através de fatores como desempenho, custo e dificuldade de execução.

Capítulo V -

Aspectos de Implementação

V.1 - INTRODUÇÃO

No capítulo anterior foi proposto um processador com estrutura em "pipeline" para deteção de bordas em uma imagem. Cada um dos estágios do processador pode ser implementado de diversas maneiras. Cada solução diferente implica em variações de desempenho, tecnologia, dificuldade de execução e custo. Evidentemente, a escolha, entre as diversas alternativas possíveis, de qual é a mais adequada à implementação deve levar em conta as particularidades de cada aplicação, em vista dos parâmetros citados.

Sendo os estágios do "pipeline" separados uns dos outros por "latches", a implementação de cada um deles torna-se relativamente independente. O único requisito que é comum a todos é o tempo de execução que deve ser menor ou igual ao período do relógio. Este, por sua vez, está relacionado diretamente ao desempenho e influi decisivamente na implementação.

Existem dois componentes em qualquer estágio de um "pipeline": a lógica para realizar a função requerida e o mecanismo para transferir a saída de um estágio para a entrada do estágio seguinte. O emprego de "pipeline" nos circuitos tem por finalidade o aumento de desempenho na execução de uma tarefa. O caminho para se atingir um aumento de desempenho em um "pipeline" passa por aumentar o número de estágios e diminuir o tempo do ciclo do relógio.

Pode-se diminuir o relógio básico através da diminuição do número de portas ou da lógica contida em cada estágio e correspondentemente aumentando o número de estágios. Quando isto é tentado, duas restrições de projeto tornam-se evidentes, ambas ligadas a natureza do pulso de relógio usado para gatilhar os "latches" dos estágios. Na fig.V.1 abaixo, o pulso de relógio tem dois componentes básicos, T e W. T é o tempo disponível para os sinais se propagarem da saída de um "latch" através da lógica para aquela função, até a entrada do próximo "latch". W é a largura do pulso de relógio e representa o

tempo necessário para aceitar o resultado dentro do "latch" e o tempo de estabilização da saída.

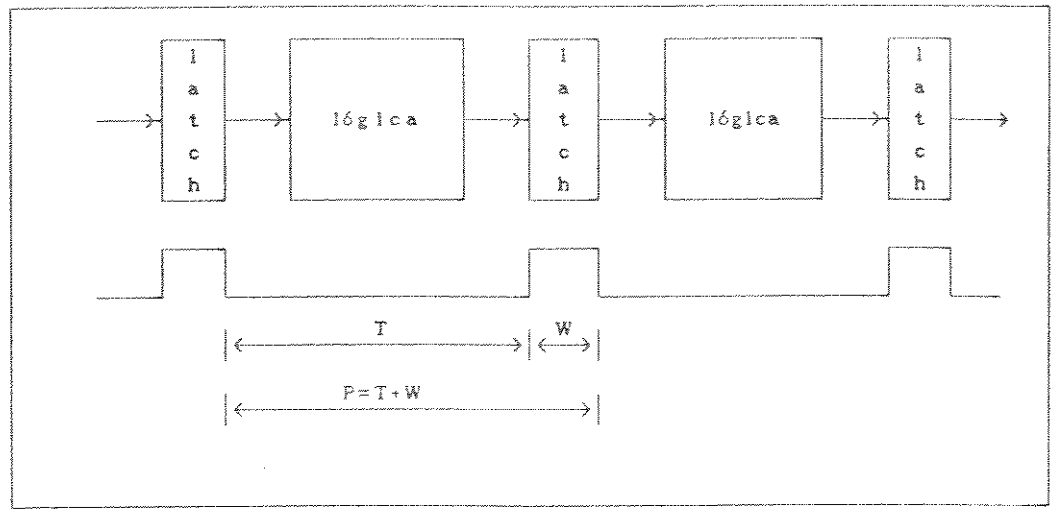


fig.V.1 - representação dos tempos em um "pipeline".

A primeira restrição de projeto mais óbvia é que o tempo Tmax para um sinal atravessar o caminho mais longo através de uma lógica não deve ser maior que T. A segunda restrição, não tão óbvia, tem a ver com a possibilidade do caminho através da lógica ser muito curto e ao mesmo tempo, se um "latch" mudar sua saída um pouco mais cedo durante W, então a mudança pode atingir o próximo "latch" e mudá-lo durante o mesmo pulso de relógio. Esta situação é conhecida por corrida crítica ("critical race"). Para evitá-la, o caminho lógico mais curto em cada estágio deve ser aumentado pela adição de circuitos não atuantes que simplificadamente introduzem um atraso na propagação do dado.

Juntas, estas duas restrições colocam limites na mínima e máxima quantidade de circutos lógicos que pode ser colocada em qualquer caminho entre

a saída de um "latch" e a entrada do próximo. Estes limites são ainda mais estreitados quando o fenômeno do "clock skew" é considerado. Idealmente todos os estágios do "pipeline" recebem o mesmo pulso de relógio ao mesmo tempo. Contudo, as diferenças nos condutores, nas cargas e circuitos de "drive" tornam praticamente impossível garantir tempos de chegada idênticos. O termo S do "skew" representa esta diferença no tempo de chegada de um mesmo pulso de relógio em diferentes estágios e deve ser levado em conta em ambas as restrições. Primeiro, se o "skew" é tal que um estágio recebe o sinal de relógio um pouco antes do seu predecessor, então o tempo disponível para a propagação lógica é reduzido. Em termos de Tmax, T e S é necessário que:

$$Tmax \leq T-S$$

A fig.V.2a mostra este caso. Do mesmo modo, a corrida crítica é piorada se o sinal de relógio para um estágio é atrasado relativamente ao mesmo sinal de relógio para o estágio anterior (fig.V.2b). Este atraso permite mais tempo para que as trocas na saída de um estágio se propague através de um caminho lógico mínimo e afete seu sucessor durante o mesmo pulso de relógio. Para evitar este problema deve-se fazer:

$$Tmin \geq W+S$$

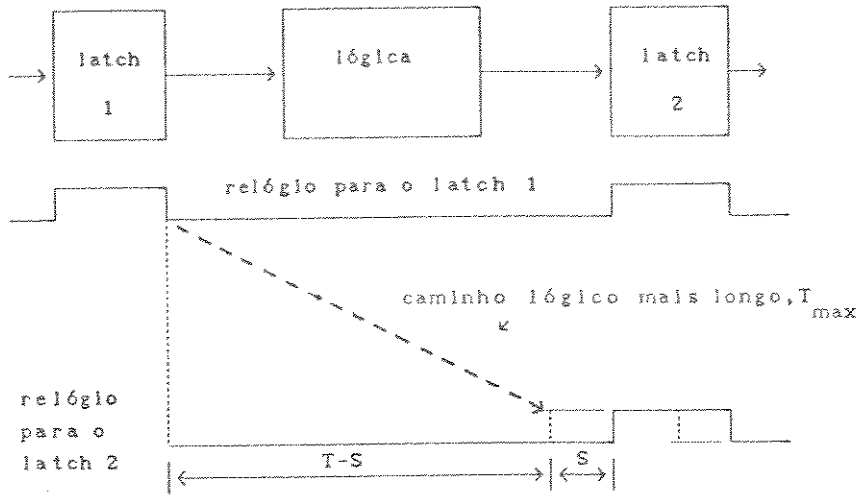


fig.V.2a - efeitos do clock skew no caminho lógico mais longo.

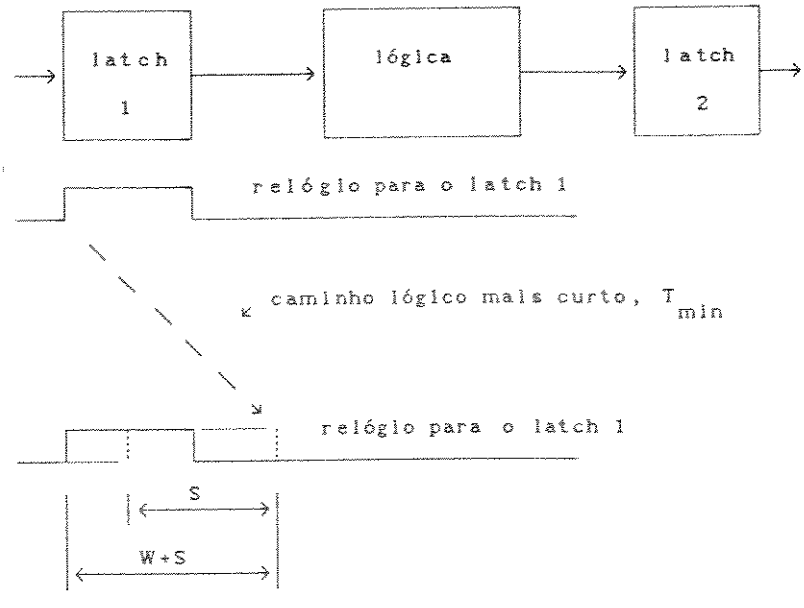


fig.V.2b - efeitos do clock skew na corrida crítica.

V.2 - O "LATCH"

Das considerações anteriores é óbvio que a chave para um alto desempenho esta no comportamento do "latch". As características necessárias para que um "latch" seja adequado são alta velocidade e consistência na temporização. Por alta velocidade de operação se entende que um tempo mínimo é empregado para aceitar um dado na entrada e armazená-lo. Isto se traduz em menores larguras de pulso, mais tempo usável pela lógica e consequentemente maior desempenho. Consistência de temporização significa que quando a saída do "latch" mudar para seguir um novo valor na sua entrada, deve fazê-lo sempre gastando a mesma quantidade de tempo, independente de qualquer saída que seja observada ou se a mudança é de 0 para 1, ou de 1 para 0. Inconsistência de temporização em "latch" é equivalente a introdução de "skew" adicional no circuito com consequente perda de tempo de processamento.

V.3 - REGISTRADOR DE DESLOCAMENTO DE ENTRADA.

Sua função é a cada novo pulso de relógio posicionar os pixels corretos para a multiplicação pelos coeficientes do núcleo. Entretanto, este posicionamento é caracterizado exatamente pelo deslocamento de um conjunto de três pixels, onde é realizada a entrada de um novo pixel e a saída de outro a cada novo pulso de relógio. Esta função pode, obviamente, ser desempenhada por um registrador de deslocamento.

Ao mesmo tempo este bloco é responsável pelo armazenamento temporário do conjunto de pixels (um bloco) transferido da memória para o processador. Estes pixels devem ser armazenados localmente durante o tempo necessário para efetuar todo o processamento no qual eles estão envolvidos, evitando acessos duplicados à memória. Na definição da estrutura deste estágio deve-se também levar em conta que a forma de acesso à memória é em blocos lineares de oito pixels. Devido as características do processamento desejado, esta forma de endereçamento cria um problema de "fronteiras" entre blocos. Para o processamento de pixels no interior do bloco necessita-se apenas dos pixels do próprio bloco, mas para o processamento de pixels situados nas extremidades, pixels 1 e 8 do bloco, é necessário a presença de pixels situados nos blocos vizinhos, anterior e posterior respectivamente. Esta última situação implica em necessidade de mais acessos à memória provocando um "overhead" no sistema. Existe ainda o problema de término de representação da imagem além de suas bordas. Neste caso, além da presença da mesma situação de fronteira ainda há a dificuldade de tratar a ausência de pixels. A solução adotada, arbitrariamente, foi envolver a imagem com uma linha, ou moldura de pixels nulos. Não existe uma representação na estrutura de dados, mas todo o processamento deve levar em conta a existência da moldura. Uma outra solução usualmente adotada é a repetição das linhas envolventes da imagem.

A solução do problema de fronteiras leva em conta, nesta proposta, que cada bloco é acessado sequencialmente. Para o processamento do último pixel de um bloco (entendido que o processamento é realizado no pixel do centro do

núcleo) é feito um novo acesso à memória e a leitura do bloco seguinte. Todo bloco é mantido em registros locais de modo que que ele se concatena linearmente com o final do bloco anterior. Deve-se observar que não é necessário manter todo o bloco anterior para que se faça o processamento, mas apenas os dois últimos pixels. Por outro lado, para o processamento do primeiro pixel do bloco recém carregado é necessário a presença do último pixel do último bloco. Para isto é reservado apenas um registro. Com este procedimento obtém-se um comportamento bem ordenado no fluxo dos pixels e pode-se implementar a estrutura da maneira ilustrada na fig.V.3.

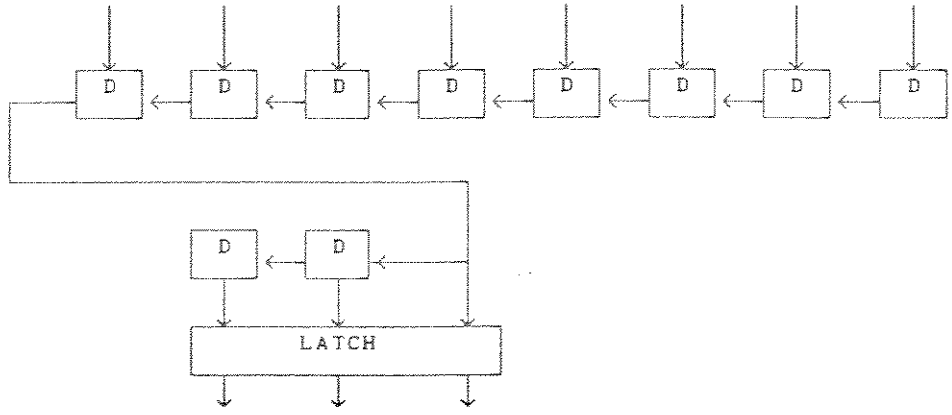


fig.V.3 - possível implementação do primeiro estágio.

Em se tratando de implementação física é possível mais de uma alternativa. A primeira e mais óbvia é o uso de simples registradores que seriam concatenados para a composição de um registrador de deslocamento no qual a estrutura de controle deverá ser realizada através de fiação. Uma segunda alternativa é o emprego de registradores de deslocamento com entrada paralela que seriam usados para carregar os blocos da memória. Eles seriam conectados de forma que cada bit de um pixel (D0, D1, ... , Dn) seria ligado em um registrador de deslocamento diferente e todos os bits com um mesmo peso

significativo seriam ligados em um mesmo registrador de deslocamento. Para esses oito registradores não há a necessidade de implementação de controle uma vez que os chips já o possuem internamente (CI's padrões). Entretanto os dois registradores restantes não podem ser implementados dessa forma pois não existem CI's padrões com 10 registradores (ou dois, se optar-se por implementação separada), e portanto eles devem ser implementados da mesma maneira que na primeira alternativa apresentada.

É possível verificar que algumas situações de exceção ocorrem na estrutura proposta, ou seja, situações nas quais a operação normal de deslocamento dos pixels não pode ser executada.

A primeira se verifica na inicialização, quando é necessário introduzir a moldura. Isto é feito com a introdução de zeros nos registradores mais a esquerda, responsáveis pela passagem dos dados para o estágio seguinte. No mesmo instante é feita a leitura do primeiro bloco da memória. Uma segunda operação é então necessária consistindo de um deslocamento à esquerda no bloco lido para que o primeiro pixel seja posicionado corretamente no segundo registrador.

A segunda situação ocorre quando é processado o oitavo pixel de um bloco e este é posicionado no segundo registrador. Nesta situação, para se realizar o processamento é necessário que se faça a busca de um novo bloco na memória de maneira que seja suprido o primeiro pixel do próximo bloco. Entretanto a leitura de um novo bloco no interior do registrador só é possível se este estiver desocupado. Consequentemente o carregamento e o deslocamento (do oitavo pixel) não podem ocorrer ao mesmo tempo. Se tanto o carregamento quanto o deslocamento forem realizados em um tempo adequadamente reduzidos eles poderão ser efetuados em um mesmo período de relógio com pequenas defasagens de tempo. Se o contrário for verdadeiro, o carregamento e o deslocamento deverão ser efetuados em períodos de relógio diferentes, acarretando uma não linearidade no fluxo interno de dados que implicaria em sofisticação e aumento da complexidade do controle do "pipeline", além da diminuição da eficiência de processamento.

Finalmente a terceira situação de exceção é encontrada no esvaziamento ("flushing") do "pipeline". Quando do processamento do último pixel do último bloco, é necessário que se leve em conta a moldura da imagem. Esta é colocada fazendo-se com que a cada pulso de relógio um zero seja inserido nos registradores de deslocamento mais à direita. Assim quando o oitavo pixel do último bloco estiver sendo processado não haverá carregamento de um novo bloco e a moldura nula será utilizada como dado.

Estas três situações devem ser tratadas adequadamente no controle do "pipeline" com a geração adequada dos sinais para cada estágio.

V.4 - ESTÁGIO MULTIPLICADOR

A implementação de um estágio multiplicador para a multiplicação de dois números inteiros de ponto fixo é bastante custosa ponto de vista de complexidade de circuito, principalmente se for necessário um alto desempenho.

Uma opção interessante para a multiplicação de dois números inteiros de ponto fixo é através do uso de somadores CSA ("carry-save adder") [25].

Tradicionalmente, a multiplicação de dois números de ponto fixo é feita através de repetidas operações de deslocamento-e-soma, usando uma unidade lógica e aritmética com as funções de soma e deslocamento incorporadas. O número de deslocamentos-soma necessárias é proporcional à largura dos operandos. Está execução sequencial faz com que a multiplicação seja um processo bastante moroso. Pela exame da matriz de multiplicação de dois números na fig.V.4 fica claro que o processo de multiplicação equivale a adição de múltiplas cópias de multiplicandos deslocados, tais como os seis mostrados na fig.V.4.

Multiplas adições podem ser realizadas com somadores em árvores multinível. O somador com propagação de "carry" ("carry propagation adder - CPA") adiciona os números da entrada, A e B, para produzir um número de saída, chamado de soma A+B. Um somador salva-"carry" ("carry-save adder -CSA") recebe três números na entrada, A, B e D, e produz dois números, o vetor soma S e o vetor "carry" C. Matematicamente, A+B+D = S $\oplus$ C onde $\oplus$ é operação lógica ou-exclusivo bit a bit e + é a adição aritmética.

Um somador de propagação de "carry" pode ser implementado com uma cascata de somadores completos com a saída de "carry" de um estágio anterior conectada a entrada de "carry" de um estágio seguinte. Um somador salva-"carry" pode ser implementado com um conjunto de somadores completos, com todas as entradas de "carry" servindo como entrada para o terceiro número D, e todas as saídas de "carry" servindo como saídas para o vetor C. Em outras palavras, as linhas de

"carry" de todos os somadores completos não são interconectadas em um somador salva-"carry".

$$
\begin{array}{rcl}
 & a_5\ a_4\ a_3\ a_2\ a_1\ a_0 & = A \\
x) & b_5\ b_4\ b_3\ b_2\ b_1\ b_0 & = B \\
\hline
 & a_5b_0\ a_4b_0\ a_3b_0\ a_2b_0\ a_1b_0\ a_0b_0 & = W_1 \\
 & a_5b_1\ a_4b_1\ a_3b_1\ a_2b_1\ a_1b_1\ a_0b_1 & = W_2 \\
 & a_5b_2\ a_4b_2\ a_3b_2\ a_2b_2\ a_1b_2\ a_0b_2 & = W_3 \\
 & a_5b_3\ a_4b_3\ a_3b_3\ a_2b_3\ a_1b_3\ a_0b_3 & = W_4 \\
 & a_5b_4\ a_4b_4\ a_3b_4\ a_2b_4\ a_1b_4\ a_0b_4 & = W_5 \\
 & a_5b_5\ a_4b_5\ a_3b_5\ a_2b_5\ a_1b_5\ a_0b_5 & = W_6 \\
\hline
P_{11}\ P_{10}\ P_9\ P_8\ P_7\ P_6\ P_5\ P_4\ P_3\ P_2\ P_1\ P_0 & & = A\ X\ B = P
\end{array}
$$

fig.V.4- Matriz multiplicação de dois números de 6 bits.

Agora pode-se mostrar como usar várias CSAs para a adição de vários números. Esta, por sua vez, serve para o propósito da multiplicação "pipeline". Este "pipeline" é projetado para multiplicar dois números de seis bits, como ilustrado na fig.V.5. O "pipeline" consiste de cinco estágios. O primeiro estágio é para a geração de todos os 36 (6X6) produtos imediatos $\{a^ib^j \mid 0\le i\le 5 \text{ e } 0\le j\le 5\}$, que formam 6 linhas de multplicandos deslocados $\{W^i \mid i=1,2,\ldots,6\}$. Os seis números entram em dois CSAs no segundo estágio. No total, 4 CSAs são interconectados formando uma árvore de três níveis de somadores salva-"carry" (do estágio 2 para o estágio 4 do "pipeline"). Esta árvore de CSAs junta seis números em dois: o vetor soma S e o vetor "carry" C. O estágio final é uma CPA ("carry" antecipado pode ser adicionado se o operando é longo) que adiciona os dois números S e C para produzir o resultado final, o produto P=AxB. Se a árvore CSA for usada para múltiplas adições de números de um bit, obtém-se a bem conhecida árvore de Wallace.

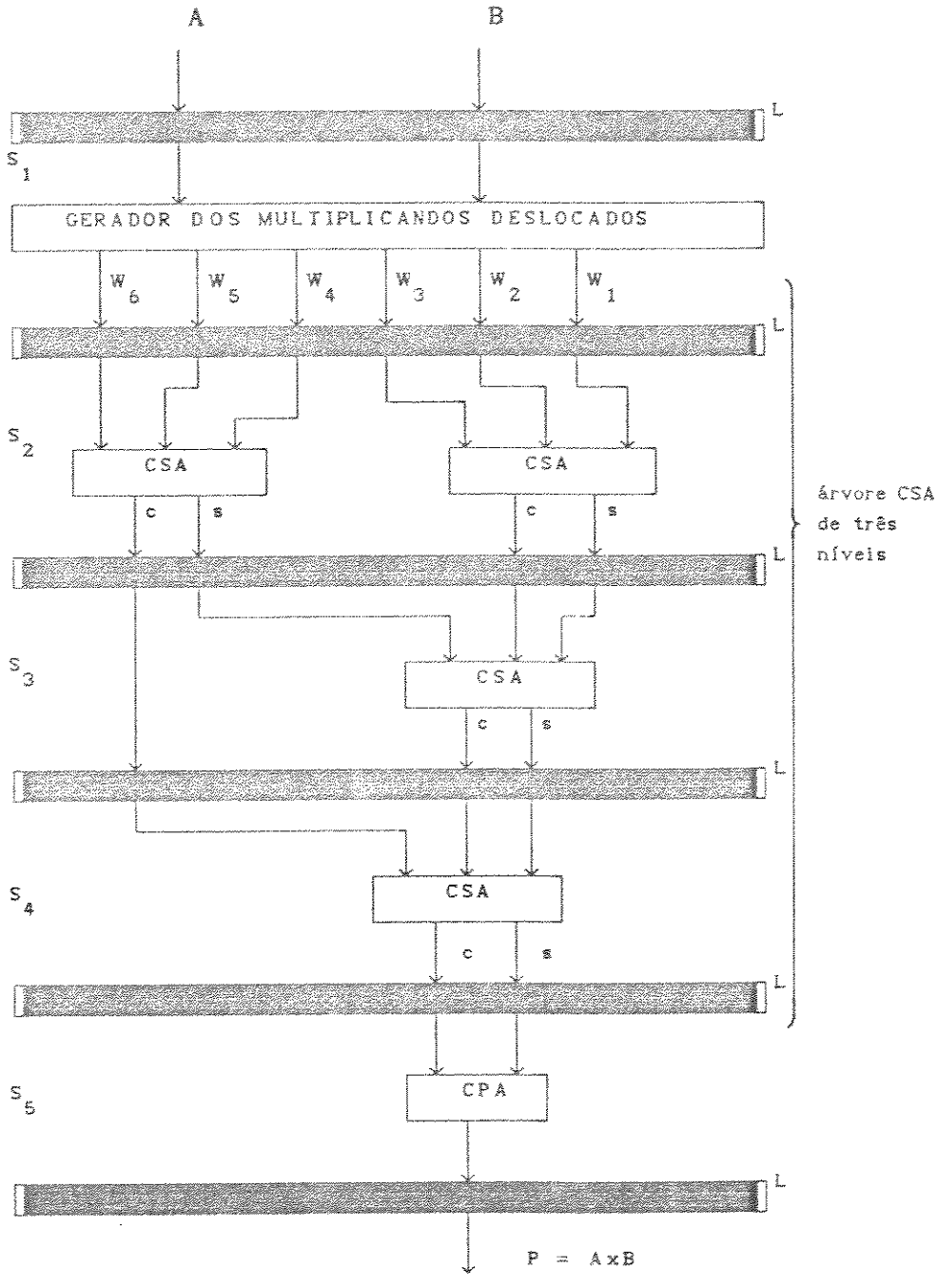


fig.V.5 - Multiplicador "pipeline" construído com uma árvore CSA.

O emprego de multiplicadores baseados em "look-up-tables" também é possível. Entretanto "look-up-tables" possuem uma gama de aplicações menor já que é difícil ou mesmo custoso implementar multiplicadores para uma faixa grande de valores, pois o tamanho das memórias cresce exponencialmente com a quantidade de bits necessária para a representação dos números. Uma maneira de atenuar este crescimento é utilizar memórias de escrita e leitura que permitam o carregamento de novas tabelas sempre que necessário referenciar um valor não contido na faixa de valores existentes originalmente. Embora consiga-se limitar o tamanho das memórias empregadas, surge um tempo adicional de troca dos valores da tabela, tempo este que pode não ser aceitável.

A utilização de "look-up-tables" pode ser bastante interessante em casos de multiplicações por coeficientes formados por constantes, ou ponderações. Nas ponderações a faixa de valores a ser armazenada nas tabelas é limitada já que um dos multiplicandos é constante. Portanto, é necessário armazenar na tabela apenas o conjunto dos valores resultantes quando uma variável é multiplicada por um constante e o número de posições na tabela está relacionada com a quantidade de bits necessária para a representação da variável. Um outro fator importante a ser levado em consideração no emprego de "look-up-tables" é o tempo de acesso das memórias que devem ser compatíveis com os outros estágios do "pipeline". Atualmente existe uma grande variedade de memórias rápidas com diversos formatos e este problema não chega a ser um impecilho a maioria dos projetos, embora permaneça como restrição de projeto.

Um caso especial de ponderação é aquele definido por constantes que são potências de dois. Ponderações deste tipo podem ser realizadas através de simples operações de deslocamento, simplificando de maneira acentuada a implementação. As multiplicações por potências de dois são realizadas com tantos deslocamentos à esquerda quanto for o valor do expoente, isto é, no caso de uma multiplicação por oito são realizados três deslocamentos à esquerda. Nos bits menos significativos são introduzidos zeros. O mesmo procedimento deve ser seguido para as divisões mudando-se a direção dos deslocamentos.

Registradores de deslocamento podem ser usados para implementar as

ponderações. Um determinado valor é inserido no registrador e é deslocado a cada pulso de relógio tantas vezes quanto for o valor da potência da ponderação. Este método possui o incoveniente de necessitar diferentes tempos de execução para diferentes ponderações.

Uma solução mais simples consiste em efetuar os deslocamentos através das mudanças nas ligações, como ilustrado na fig.V.6a. A vantagem é que o tempo de multiplicação é reduzido a um valor praticamente nulo, bem como o custo de implementação. Como restrição existe a pouca flexibilidade, uma vez que a implementação é "hardwired". Pode-se atenuar a rigidez usando-se um conjunto chaves para efetuar os deslocamentos como ilustrado na fig.V.6b.

Deve-se observar que o bit mais significativo representando o sinal em complemento de dois não muda. Note também que os bits mais significativos são perdidos resultando em erro de transbordo. Para atenuar o erro é possível prover o circuito com uma lógica de saturação. Para eliminar completamente o erro é necessário aplicar o número de bits empregados na representação dos resultados.

Para realizar ponderação utilizando representação em complemento de um pode-se utilizar a mesma estrutura. Para a multiplicação por um coeficiente negativo,como algumas vezes é necessário em processamento de imagem, é preciso obter a representação negativa do número após a operação de multiplicação. Na representação em complemento de um a tarefa é muito simples exigindo-se apenas um conjunto de portas ou-exclusivo ligadas da forma apresentada na fig.V.7.

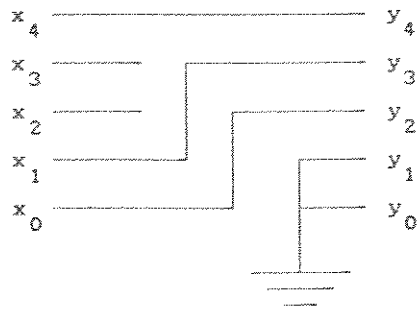


fig.V.6a- multiplicação por X4

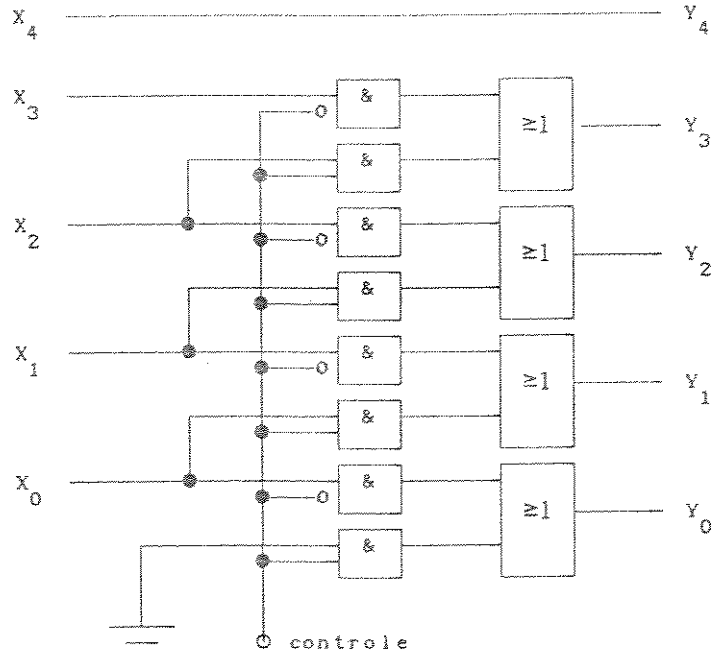


fig.V.6b- multiplicação por 1 e 2. Se o controle=1, y=x.2 se não y=x.

Se for necessário obter o complemento do número deve-se colocar o sinal de controle em "1". Se a representação numérica for em complemento de dois, além do complemento é necessário somar uma unidade ao número, acarretando portanto um custo adicional ao circuito. O sinal de controle pode ser acionado pelo próprio bit de sinal do coeficiente.

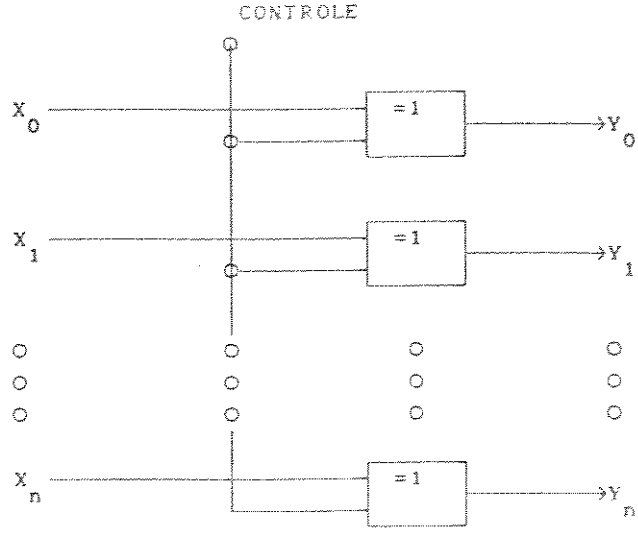


fig.V.7 - circuito para complemento, controlado pelo bit de sinal.

Existe uma quantidade razoável de algoritmos, entre os mais conhecidos para processamento de imagem, nos quais as ponderações com potências de dois estão presentes. Dessa maneira a implementação desses algoritmos em "hardware" se torna bastante simplificada se utilizarmos as alternativas mostradas anteriormente.

V.5 - O ESTÁGIO SOMADOR

Este estágio é formado pela associação de elementos somadores. A estrutura básica para realização da soma entre dois bits é chamada de meio-somador e se uma lógica é acrescentada para levar em conta o transporte, então o elemento se torna um somador completo. Assim como qualquer função lógica não trivial, um somador pode ser implementado de diversas maneiras.

Os somadores são classificados em geral pela maneira como manipulam os dados, isto é, serialmente ou em paralelo.

a) somadores seriais.

Um somador para operação no modo serial é facilmente construído apartir do somador completo conectando um "flip-flop" entre o "carry-out" e o "carry-in".

As duas palavras a serem adicionadas são aplicadas aos terminais de entrada em bit sincronismo com o bit menos significativo entrando primeiro. O flip-flop recebe o carry de uma dada posição e o aplica para a próxima posição mais significativa.

Para evitar interferência entre as palavras é necessário inibir a propagação do carry durante a chegada dos bits menos significativos de cada nova palavra através da aplicação de um pulso de reset por cada ciclo de palavra.

Se números negativos são considerados, o somador apresentado operará sem qualquer alteração se for considerada representação em complemento de dois. Se a representação for em complemento de um será necessário realimentar a saída de carry para a entrada.

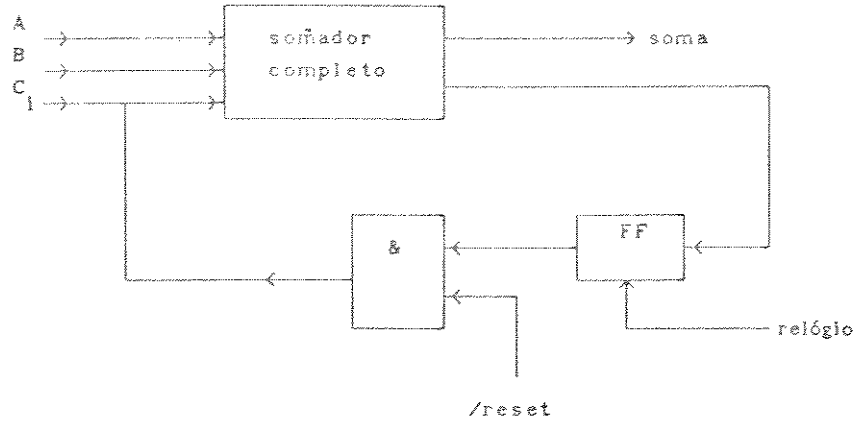


fig.V.8 - somador serial feito a partir de um somador completo .

b) somadores paralelos.

Um somador paralelo, com vários bits sendo somados simultaneamente, pode ser construído agrupando-se diversos somadores simples, na qual a saída do carry de cada somador simples é conectada a entrada de "carry" do somador que está na posição imediatamente mais significativa. É óbvio que para circuitos síncronos, com o mesmo relógio e palavras de mesma largura, o somador paralelo é mais rápido que o serial, embora mais custoso. A configuração apresentada possui entretanto, um atraso na propagação do "carry" da posição menos significativa até a mais significativa. Para a influência do "carry" do estágio menos significativo ser sentida no estágio mais significativo deve-se considerar o tempo de propagação dele através de todos os estágios intermediários. Portanto, para um somador paralelo e outro serial, com a mesma tecnologia, o ciclo de execução (ou o relógio do sistema) do paralelo será mais lento. Para evitar este inconveniente usa-se uma técnica conhecida como "carry look-ahead". Esta técnica utiliza uma lógica extra para calcular o

"carry" necessário para a n-ésima soma diretamente das entradas, diminuindo o tempo de propagação. Utilizando-se esta técnica para pequenos grupos de bits pode-se manter o tempo de propagação do "carry" relativamente constante independente do número de bits da palavra, embora haja um custo adicional no "hardware".

O somador paralelo se apresenta como sendo mais adequado para implementação neste projeto por que permite a obtenção do desempenho desejado sem restringir a escolha da tecnologia, ao passo que a utilização do somador serial implicaria num relógio de operação elevado, para o mesmo desempenho, diminuindo as alternativas para a tecnologia de componentes empregada.

Além do tipo do somador escolhido, a forma como estes serão utilizados é muito importante neste projeto. Isto se deve ao fato do estágio somador ser responsável pela somatória de diversos valores (nove ao todo). Parte desta somatória é realizada em paralelo, parte iterativamente através de operações de acumulação.

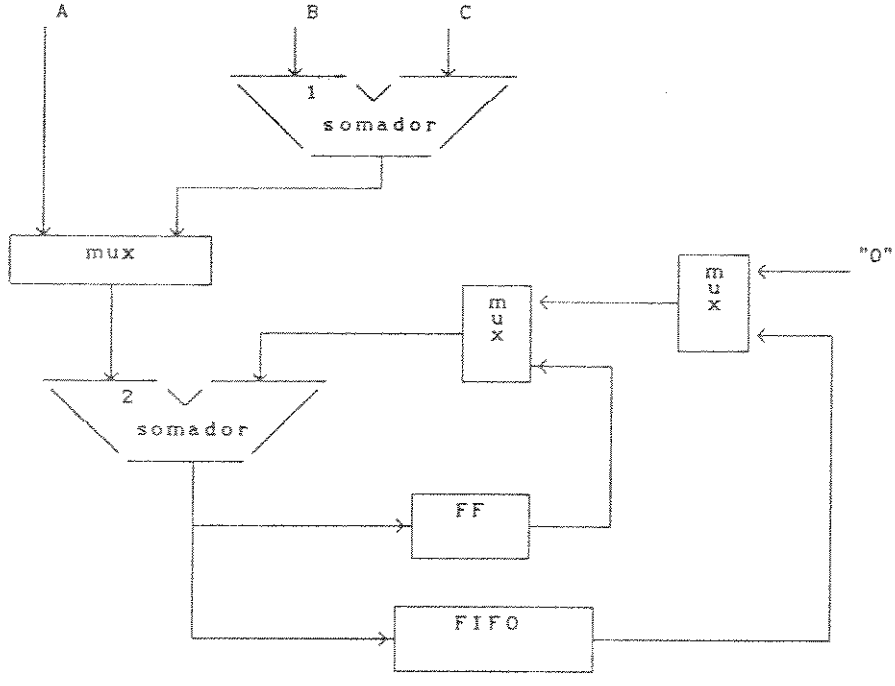


fig.V.9 - estrutura do estágio somador utilizando somente dois elementos somadores.

Na proposta aqui apresentada, a soma é feita parcialmente sobre quatro valores, dois a dois numa estrutura de árvore. A estrutura em árvore é a mais simples e eficiente para se realizar a soma dos quatro valores em um único passo, mas não é a única possível. Pode-se utilizar apenas dois somadores, realizando a operação em dois passos. A organização do estágio é aquela apresentada no diagrama em blocos da fig.V.9.

A somatória é realizada da seguinte forma:

1 - soma-se B e C no somador 1, e A é o valor do topo da fifo no somador 2.

2 - soma-se os resultados anteriores no somador 2.

Repare que as duas somas intermediárias devem ser armazenadas para a segunda operação. No caso da soma de B e C não é necessário um registrador uma vez que estes valores estão mantidos pelo "latch" existente entre os estágios. A escolha dos valores a serem somados em cada passo é determinada pelos multiplexadores. A operação soma completa dos quatro valores pode, dependendo dos elementos somadores utilizados ser feita em único ciclo de relógio do "pipeline" ou pode ser feita em dois pulsos de relógio. No primeiro caso têm-se uma decomposição de um ciclo de relógio em sub-ciclos implicando nos necessários cuidados com a temporização das operações. O segundo caso leva a uma decomposição do estágio somador em dois outros, alterando o comportamento geral do "pipeline".

Uma outra abordagem para a implementação do estágio somador incorpora o desenvolvimento do conceito de recorrência [26], presente na soma acumulativa. A recorrência é um tipo de função com realimentação que aceita como sua entrada um conjunto de valores a(1), a(2),.....,a(n) e produz como saída outro conjunto de valores x(1), x(2),...., x(n) onde para algum m>0, cada x(i) é uma função de a(i) e de x(i-1) até x(i-m). Isto é:

$$x(i) = f(a(i), x(i-1),\ldots,x(i-m))$$

O desenvolvimento do estágio somador baseado no estudo de recorrências pode alterar por completo a estrutura do processador apresentado aqui.

Para o nível de desempenho e custo desejados neste projeto a estrutura em árvore composta por quatro somadores paralelos, utilizando representação em complemento de um, parece ser a mais adequada. É possível encontrar componentes comerciais que satisfaçam estes requisitos com uma certa facilidade.

V.6 - A FIFO

A FIFO representa um atraso necessário para sincronizar a soma de pixels vizinhos, mas em linhas diferentes. Consequentemente seu comprimento é exatamente o comprimento de uma linha de pixels , no caso presente 512. Este tamanho apresenta-se como uma primeira dificuldade para implementação da FIFO. Em geral, as FIFO's encontradas comercialmente possuem comprimentos pequenos tais como 8 ou 16 posições dificultando a implementação da FIFO desejada em termos de espaço ocupado e também custo. Contudo, esta opção é simples e direta em termos de projeto.

O tamanho relativamente grande da FIFO viabiliza uma solução alternativa com o uso de memórias RAM's rápidas e de pequeno tamanho. Uma primeira opção recai sobre as memórias "dual-port" cujo uso é comum em processadores "pipeline", sendo, geralmente, denominadas denominadas "register files". Uma porta é conectada a saída de um estágio ao passo que a outra é conectada a entrada do próximo. A primeira porta é configurada para escrita e a segunda para leitura, enquanto que a unidade de controle providencia os dois endereços necessários. Genericamente falando, as memórias "dual- port" possuem pelos menos três características que as FIFO's não possuem. Primeiro, permite que um determinado conjunto de valores seja repetidamente usado sem que seja necessário regenerá-los. Segundo, pode ser usado para embaralhar dados previamente ordenados, ou reordená-los de maneiras diversas. Parte do "pipeline" pode gerar dados em um determinada ordem, mas o restante do "pipeline" pode necessitar dos dados em outra. Um controle separado de endereços pode proporcionar este tipo de reordenação. A terceira característica e mais comum aplicação, é como atraso variável. Um atraso variável K pode ser obtido usando ciclicamente uma sequência crescente de endereços para a porta de escrita e o mesmo endereço para a porta de leitura mas atrasado k ciclos. A fig.V.10 mostra o diagrama de tempo para um atraso de dois ciclos.

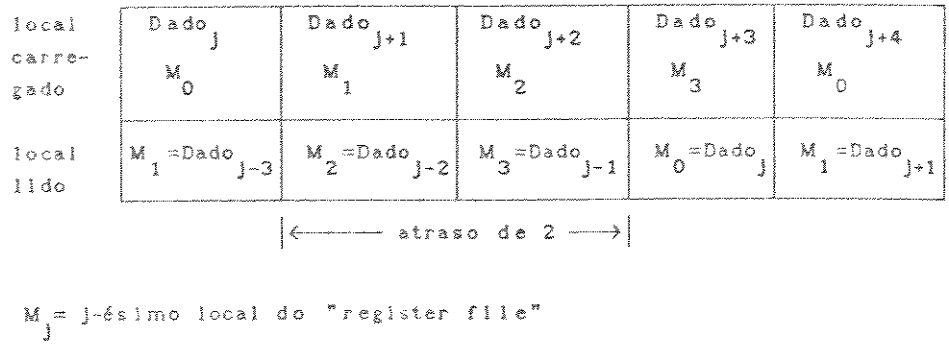


fig.V.10 - endereçamento de um register file para indução de atraso.

É exatamente neste modo de operação que as memórias dual- port são úteis neste processador. No caso, o atraso desejado é de 512 ciclos. A própria característica dessa implementação, atraso variável, introduz maior flexibilidade ao projeto podendo-se acrescentar a possibilidade de processamento de imagens de tamanhos diferentes ou mesmo de janelas dentro de uma imagem. O processamento de janelas pode ser realizado através da limitação dos endereços fornecidos pela unidade de controle à memória de imagem e também pela alteração conveniente do atraso do "register file".

As memórias "dual-port" podem ser encontradas em forma de circuitos comerciais com toda a lógica necessária para controle já incorporada ao CI. Entretanto, no caso de se desejar um projeto mais adaptado as necessidades do projeto, pode-se empregar memórias RAM comuns, provendo a necessária lógica de controle para solucionar a disputa pelo acesso aos dados. O principal requisito para uma memória poder ser usada dessa forma é que o tempo de acesso para um ciclo de escrita e um de leitura contíguos seja menor ou igual ao ciclo de relógio do "pipeline". Se não for possível obter uma memória com estas características pode-se fazer a implementação com dois bancos de memória separados, como mostra a fig.V.11. Em determinado momento um banco é habilitado para escrita enquanto o outro é habilitado para leitura, simulando desta maneira o funcionamento de um memória "dual port", mas sem as mesmas

restrições de temporização. No instante que todos os dados de um banco foram acessados é feito um chaveamento entre os dois bancos invertendo-se as habilitações de escrita e leitura em cada um dos bancos.

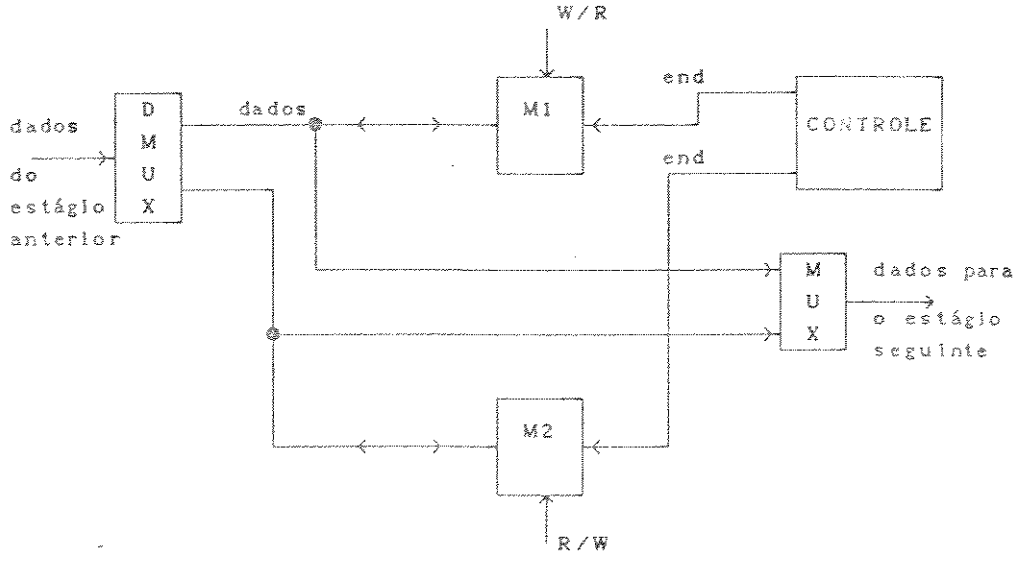


fig.V.11 - memória dual port implementada por memórias ram's comuns.

V.7 - MONTAGEM

Como descrito no início da secção V, "pipelines" são sensitivos ao "clock skew" e variações nos atrasos lógicos. Na prática existem três fatores para esses problemas: a ligação ou conexões físicas entre componentes, variação nas características dos componentes e diferenças na velocidade dos componentes devido as variações de carga. As duas últimas são evitadas ou diminuídas pela cuidadosa separação de componenentes e o desenvolvimento de boas técnicas de aterramento, ou seja, um problema de montagem, onde colocar cada componente e como interconectá-los.

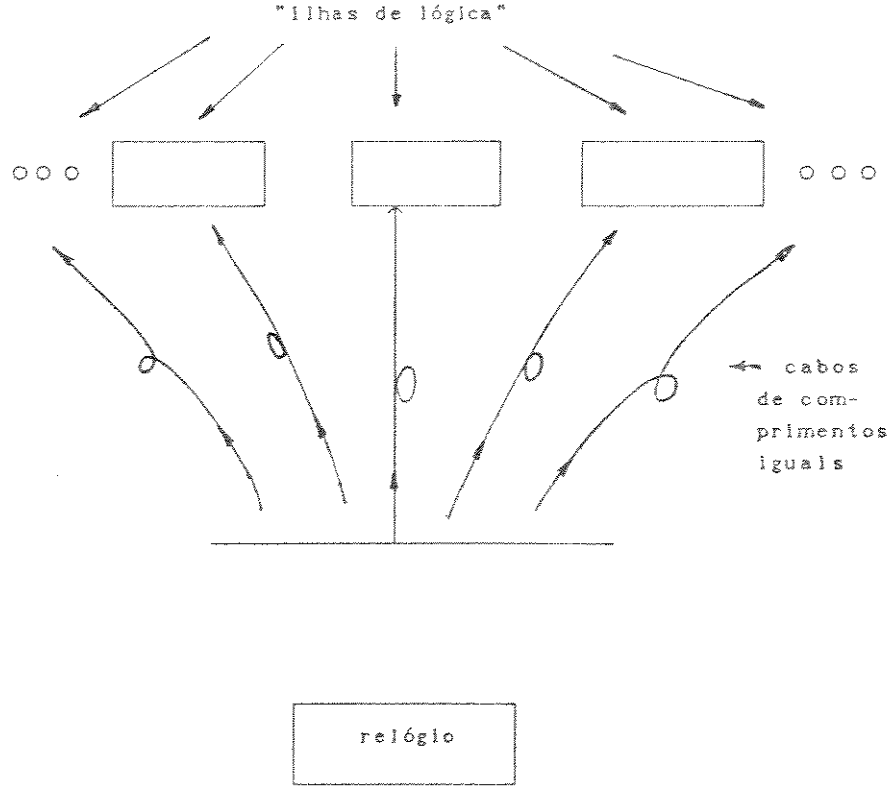


fig.V.12 -Fonte de relógio central.

Este problema de montagem mostra-se claro na distribuição do relógio do sistema. Na maioria dos projetos, particularmente para grandes "pipelines" síncronos, há uma única fonte de relógio que deve ser distribuída para todos os estágios de uma maneira uniforme. A maioria dos projetos usa variações ou combinações de dois esquemas mais comuns. No primeiro, partes lógicas comuns são agrupadas em ilhas, com um único ponto de entrada do sinal de relógio por ilha. Cabos de comprimento uniforme conectam cada ilha a um sinal de relógio central, independente da distância real entre cada ilha (fig.V.12). Um procedimento similar pode ser usado recursivamente dentro de cada ilha. Esta distribuição pode ser observada nos atuais sistemas computacionais de alta velocidade, onde os processadores principais são arranjados ao longo de um círculo com o gerador de relógio no centro.

Outra abordagem para a distribuição do sinal de relógio usa um "timing chain". Com tal arranjo, o pulso de relógio entra somente no primeiro estágio (fig.V.13). O devido cuidado deve ser tomado para produzir circuitos geradores de relógio com alta repetibilidade e atrasos consistentes. A frequência de relógio deve ser escolhida para que o período entre quaisquer dois pulsos seja maior que o maior atraso possível de qualquer temporização de qualquer estágio.

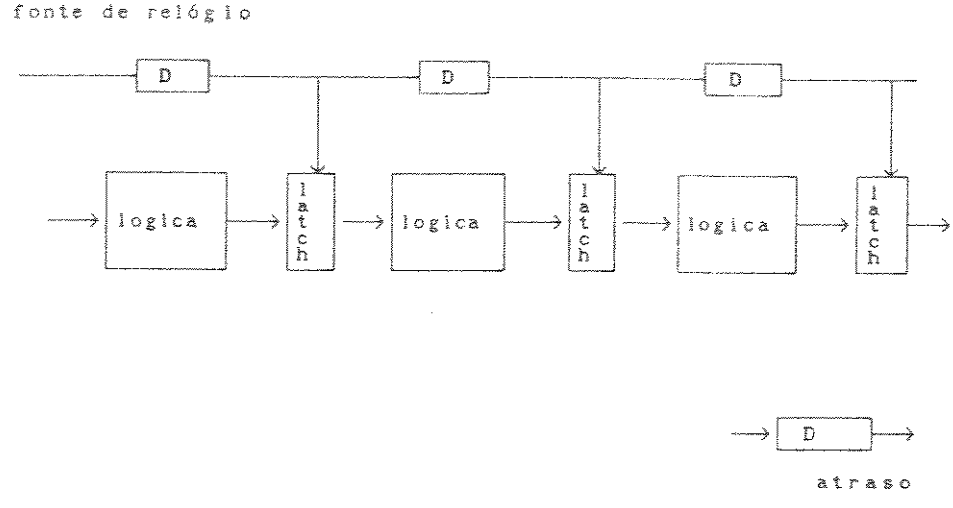


fig.V.13 - "Timing chain".

Diversas variações dessas abordagens são possíveis. Por exemplo, dentro de uma ilha individual um "timing chain" relativamente curto pode ser usado para distribuir o sinal de relógio do ponto de entrada para os estágios individuais. Em casos onde uma ilha representa um ou mais estágios completos, um "timing chain" pode interconectar os estágios da mesma maneira como eles são usados, isto é, o sinal de relógio principal alimentando somente a primeira ilha.

Uma outra possibilidade é distribuir o sinal de relógio à semelhança de uma estrela para cada estágio e usar um "timing chain" para selecionar qual sinal de relógio passará para o "latch" do estágio (fig.V.14).

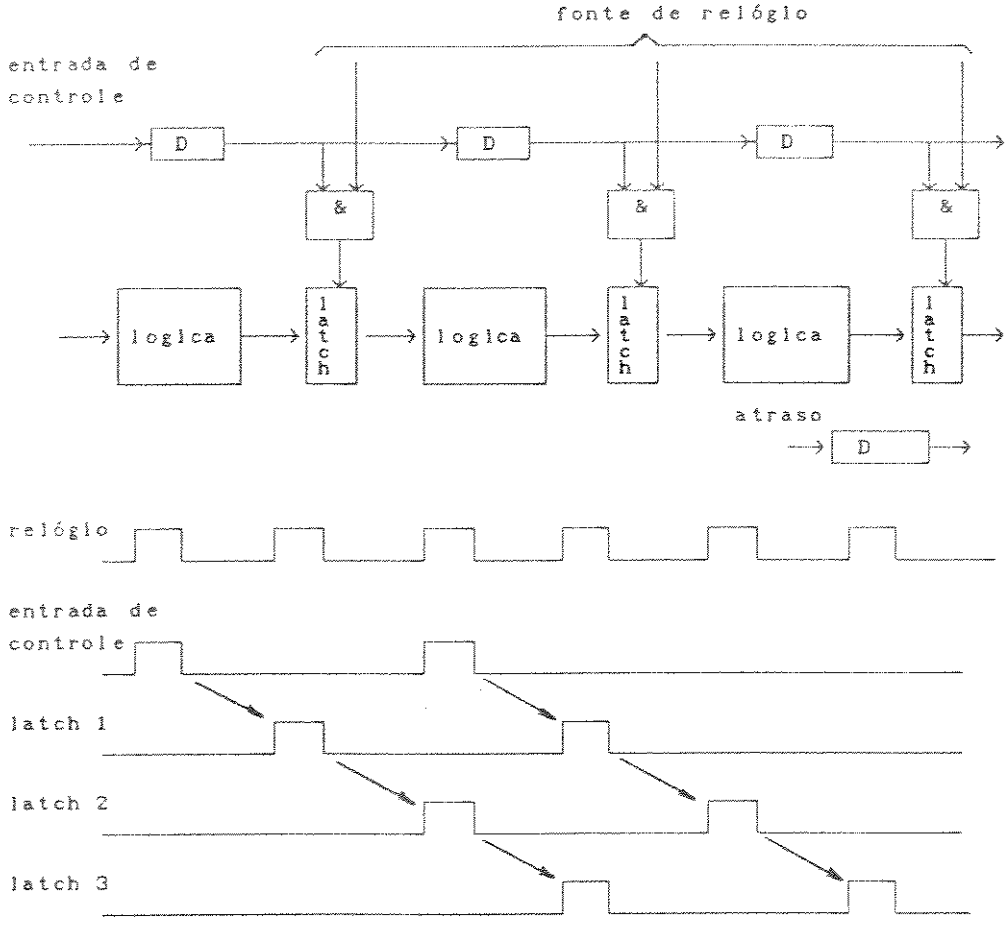


Fig.V.14 - "Timing chain" usado para controle.

Os atrasos na corrente são deliberadamente ajustados para serem menores que o período do relógio, com a saída do pulso um pouco mais larga que a largura do relógio real. Como é um pulso central de relógio que ativa o estágio, o problema previamente mencionado com variações nas características dos atrasos e assim no "clock skew" são bastante reduzidas. Uma vantagem adicional desta configuração é que agora a fonte inicial do "timing chain" pode ser independente do relógio; em particular, ele pode ser gerado somente quando um dado está entrando o "pipeline", assim servindo como um controle ao invés de uma função de temporização.

Outro fator importante a ser discutido na montagem de sistemas "pipeline" refere-se aos atrasos introduzidos por fiações, incluindo seus diversos níveis, tais como ligações entre componentes de uma placa, entre placas , ou mesmo entre cabines ou racks contendo um conjunto de placas. Estes atrasos são devidos a velocidade limitada de propagação dos sinais elétricos na fiação. Em alguns sistemas estes atrasos são tão significativos que se a lógica fosse toda feita de componentes perfeitos, sem atrasos, o desempenho do sistema somente poderia dobrar. A fiação entre componentes praticamente iguala os atrasos dos componentes lógicos.

O tratamento destes atrasos é feito através de uma distribuição adequada dos estágios nas placas e também dos componentes. Pode-se por exemplo colocar um estágio por placa minimizando os efeitos das ligações internamente aos estágios ou pode-se distribuir vários estágios entre diversas placas, minimizando os efeitos das ligações entre os estágios como representado na fig.V.15.

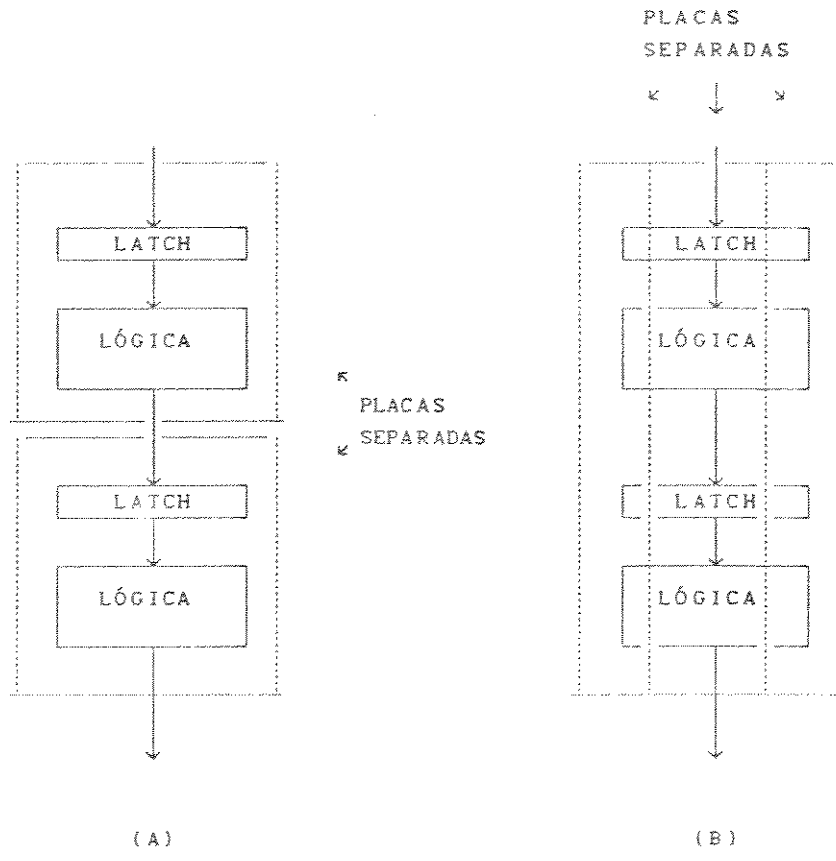


fig.V.15 - posicionamento lógico em "pipelines": a) um estágio por placa; b) uma fatia por de cada estágio por placa.

V.8 - COMENTÁRIOS

Neste capítulo buscou-se apresentar uma visão das principais formas de implementação possíveis para cada estágio do processador "pipeline" para convolução de imagem proposto no capítulo IV. Também foram apresentados alguns conceitos básicos para a implementação de qualquer estágio de um "pipeline", relativos a temporização dos sinais. Tais conceitos tornam-se mais importantes a medida que a frequência de operação do circuito aumenta, mas devem ser observados mesmo em circuitos de baixas frequências onde a tecnologia de componentes ou de montagem não é a ideal.

A análise desenvolvida para a implementação de cada um dos estágios abordou principalmente aspectos de estrutura de implementação e não de tecnologia de componentes. Cada uma das possibilidades oferecidas podem, entretanto, ser realizada com diferentes tecnologias de componentes. O aspecto da tecnologia utilizada está ligado ao custo, desempenho, e também da disponibilidade de componentes. Ainda, algumas estruturas se adaptam melhor a algumas tecnologias que outras.

É possível observar que, na maioria dos casos, um aumento da complexidade da estrutura de cada estágio conduz a uma maior flexibilidade do circuito. Assim, a adoção de ponderações no lugar de multiplicadores leva a uma maior simplicidade na implementação do estágio, mas restringe as aplicações do circuito. Da mesma forma, o uso de uma memória "dual port" para a implementação das FIFOs aumenta a complexidade do circuito, mas permite a utilização do processador em imagens de diferentes resoluções.

A repetibilidade de estruturas (somadores, multiplicadores, "latches"), a possibilidade de implementação de cada estágio com baixos níveis de complexidade, e um fluxo de controle simples, torna bastante atraente a utilização de tecnologia VLSI para a implementação de circuitos integrados dedicados.

Trabalho nesse sentido vem sendo desenvolvido por Costa [3] que

especificou e detalhou o "lay-out" de um CI com a estrutura aqui proposta. A implemetação em CI permitiu a utilização de multiplicadores baseados em CSA, tornando o processador adequado para a execução de diversos núcleos. A FIFO é implementada externamente ao CI, garantindo maior flexibilidade e menor custo, pois reduz bastante a área do CI. A tecnologia utilizada, CMOS com dimensões mínimas de 2μ, permite uma frequência de operação de até 16MHz. Portanto, dentro da especificação para operação em tempo real.

Capítulo VI

Conclusão

1 - Conclusão

O trabalho aqui apresentado está inserido numa linha de pesquisa visando capacitar o Departamento de Automação e Computação da Faculdade de Engenharia Elétrica da Unicamp na área de processamento de imagem. Nesse sentido, contribui com estudos na área de sistemas para processamento de imagens bem como a implementação de uma estação de trabalho para fins de pesquisa na área.

Além dos dispositivos básicos para processamento de imagem, interfaces de aquisição e visualização, a arquitetura da estação incorpora processadores especializados para execução de algoritmos de convolução com núcleos de 3x3. Tais processadores levam em conta paralelismo das operações para obtenção de alto desempenho no processamento.

Foi possível, dessa maneira, caracterizar uma arquitetura baseada no emprego de um barramento com banda de passagem larga e processadores dedicados, que permite a obtenção de alta velocidade de computação, habilitando o seu uso em aplicações de reconhecimento de padrões em tempo real.

A título de conclusão final, é feito aqui um resumo dos comentários já apresentados nos capítulos anteriores.

O desenvolvimento deste projeto teve como norma o uso de componentes de tecnologia bem conhecida e de fácil aquisição no mercado nacional. Com exceção das memórias todos os demais componentes são facilmente encontrados e constituem-se na sua maioria de componetes TTL. Existem dois motivos principais para a adoção desse procedimento. Um refere-se as dificuldades de importação de componentes existentes à época do início do projeto e o outro a problemas de estrutura de laboratório para desenvolvimento de montagem e testes.

Muitas das características do projeto refletem estes fatos. Assim, a

frequência de operação do circuito não deve ser alta pois a montagem foi realizada em "wire wrap" aumentando a possibilidade de interferências de ruídos. A baixa frequência de operação também permite a utilização de componentes TTL e diminui a influência de atrasos e reflexões na fiação.

A utilização de dois barramentos com palavras largas garante um alta taxa de transferência de dados mesmo com uma baixa frequência de operação. Entretanto, o número de buffers necessários para interfacear o barramento cresce bastante, assim como a potência dissipada e a área necessária nas placas. O grande número de componentes dificulta a montagem e os testes.

As características das interfaces de aquisição e visualização preenchem os requisitos de um sistema com estes propósitos. Algumas características, entretanto, podem ser adicionadas, como por exemplo a possibilidade de aquisição continua de imagens, permitindo a visualização ao vivo da imagem sendo adquirida, ou ainda, a possibilidade de aquisição de imagens de fontes cujos sinais sejam de vídeo composto, ampliando as alternativas de fontes de imagem.

O projeto dos processadores especializados buscou explorar várias características, próprias do processamento de imagem, capazes de permitir um alto desempenho na execução dos programas, muito difícil de ser obtido através de processadores de propósito geral ou mesmo de processadores progamáveis orientados para processamento digital de sinais. Entre as principais características exploradas ou analisadas no presente projeto pode-se citar: paralelismo temporal dos algoritmos, paralelismo espacial das imagens.Também explorou-se a especialização de funções do processador ou de partes dele como método para melhoria de desempenho pela otimização da implemtação de cada parte para execução de determinada função.

Sem muitas dificuldades, apenas pela extensão dos conceitos apresentados, é possível obter novos processadores para núcleos de convolução de maior dimensão, aumentando ainda mais as áreas de aplicação. Uma outra possibilidade bastante interessante é a modificação do circuito básico de maneira que seja possível a associação de vários circuitos básicos para o processamento de um

núcleo maior.

Os processadores propostos oferecem ainda a possibilidade de operarem à taxa de vídeo. Com isso, é possível operar sobre os dados obtidos diretamente de um digitalizador, diminuindo ainda mais o tempo de resposta.

As diversas possibilidades de implementação de cada um dos estágios do processador proposto foram analisadas comparando-se as alternativas através de fatores como desempenho, custo e dificuldade de execução. Através dessa análise pode-se verificar que a especialização de funções pode levar a uma simplificação na implementação, embora tenha como contrapartida uma perda na flexibilidade de aplicação dos circuitos.

A repetibilidade de estruturas (somadores, multiplicadores, "latches"), a possibilidade de implementação de cada estágio com baixos níveis de complexidade, e um fluxo de controle simples, torna bastante atraente a utilização de tecnologia VLSI para a implementação de circuitos integrados dedicados.

Num projeto de engenharia é importante a análise do mesmo sob o ponto de vista de atualidade tecnológica. Sob este aspecto o trabalho desenvolvido apresenta e analisa soluções atuais para os problemas de arquitetura e implementação. Por outro lado, a implementação de um protótipo está sujeita a disponibilidade de componentes e ferramentas para a efetivação das soluções propostas. Este fato reflete-se no presente caso, onde muitas das soluções efetivamente adotadas, embora sub-ótimas, justificam-se principalmente pelos recursos disponíveis. A incorporação de tecnologias de componentes mais modernas ao projeto deve acrescentar benefícios, sem invalidar as soluções analisadas.

Ao lado dos resultados já obtidos, o trabalho apresentado abre campo para o desenvolvimento de pesquisa em duas importantes direções. A primeira refere-se ao desenvolvimento de novos processadores dedicados que permitam executar uma maior variedade de algoritmos de processamento de imagem, e ao estudo de como estes poderiam ser interconectados de maneira a cooperarem

entre si para a melhoria de desempenho. O segundo refere-se a continuidade do estudo das alternativas de implementação dos processadores no sentido de empregar tecnologia VLSI. O trabalho desenvolvido por Costa [7] representa um dos caminhos a serem percorridos pelo emprego da tecnologia VLSI em processamento de imagem, explorando seus benefícios imediatos, como redução do número de componentes, menor consumo, confiabilidade, etc. Outras vantagens podem, entretanto, ser obtidas com o emprego da tecnologia VLSI. A principal delas, o alto nível de integração, abre a possibilidade de utilização de um grande número de processadores para a solução de problemas com grande eficiência. Por outro lado, diversos problemas devem ser solucionados para atingir tal objetivo, sendo que estes, de uma forma geral, consistem dos problemas clássicos do processamento paralelo. Em particular, é de grande interesse o desenvolvimento de técnicas que permitam um mapeamento sistemático de algoritmos diretamente para arquiteturas paralelas e consequentemente sua implementação em hardware.

BIBLIOGRAFIA:

[1] - Rosenfeld. A. "Computer Vision: Basic Principles". Proceedings of the IEEE, vol 76, no. 8, August 1988.

[2] - Danielsson, Per-Erik, S. Levialdi. "Computer Architectures for Pictorial Information Systems". Computer, November, 1981.

[3] - Maresca, M. et al. "Paralell Architectures for Computer Vision". Procedings of the IEEE, vol 76, no. 8, August 1988.

[4] - Pratt, William K. "Algorithmic-Based Machine-Vision Computing". Digital Design. October 25, 1986.

[5] - Ruetz, Peter A. and Robert W. Brodersen. "An Image-Recognition System Using Algorithmically Dedicated Integrated Circuits". Machine Vision and Applications. January, 1988.

[6] - Bursky, Dave. "CMOS four-chip set processes images at 20MHz date rates". Eletronic Design. May 28, 1987.

[7] - Costa, Henrique Sérgio G. da. "Processador Pipeline de Imagem - CI.01". Relatório Técnico nº 036/89 Faculdade de Engenharia Elétrica de Campinas - UNICAMP. Campinas 1989.

[8] - Kane, Gerry. "CRT Controler Handbook". Osborne/MacGraw Hill.

[9] - Berardi, Paulo C. "Projeto e implementação de hardware e software para um placa gráfica de média/alta resolução com capacidade de processamento local". Tese de mestrado. Faculdade de Engenharia. UNICAMP. 1989.

[10] - Gonzalez, Rafael C., Paul Wintz. "Digital Image Processing". 1977, Addison-Wesley Publishing Company.

[11] - Fallin, John." Aplication Note: CHMOS DRAMS in Graphics Aplications". Solutions, Intel. May/June 1984.

[12] - Nicoud, Jean-Daniel. "Video RAMs: Structure and applications". IEEE Micro, vol.8, no.1, Feb. 1988.

[13] - Witton, Mary C. "Memory Design for Raster Graphics Aplications". IEEE CG&A, March 1984.

[14] - Gulley, David W." Match DRAM organization to memory requirements". Computer Design. pág. 73 - 80, vol. 22, nº 14, Dec 1983.

[15] - Voorhis, Van, David C. e Thomas H. Morrin. "Memory Systems \for Image Processing". IEEE Transactions on Computers, vol c-27, nº2, February 1978.

[16] - Park, Jong W. "An Efficient Memory System for Image Processing". IEEE Transactions on Computers, vol c-35, nº.7, July 1986.

[17] - MC6845 - Motorola Inc. 1981.

[18] - Prado, Paulo H. M. "Sistema para processamento de imagem - SPI". Relatório Técnico nº 006/89 Faculdade de Engenharia Elétrica - UNICAMP. Campinas 1989.

[19] - Thurber, Kenneth J. et al. "A systematic approach to the design of digital bussing structures". AFIPS Conference Procedings, vol. 41, 1972 FJCC.

[20] - Ballard, Dana H., Christopher M. Brown. "Computer Vision". Prentice Hall.

[21] - Madrigal, R. I., José M. U.. "Vision Artificial por Computador". Paraninfo 1986.

[22] - "First Generation TMS320 User's Guide". Texas Instruments. 1988.

[23] - "Digital Signal Processing Aplications with the TMS320 Family". Texas Instruments, 1986.

[24] - Mendes, Celso Luiz e outros. "Análise de arquiteturas para processamento de imagens". Anais do I Simpósio Brasileiro de Arquitetura de Computadores - Processamento Paralelo, SBC, maio de 1987.

[25] - Hwang, Kay and Fayé A. Briggs. "Computer Architecture and Paralell Processing". MacGraw-Hill Company. 1987.

[26] - Kooge, Peter M. "The Architecture of Pipelines Computers". 1982, MacGraw-Hill Book Company.

Referências Complementares

As referências listadas a seguir complementam as citadas diretamente no texto e ajudam a situar melhor o trabalho desenvolvido.

Cantoni, Virginio and Stefano Levialdi. "Multiprocessor Computing for Images". Proceedings of the IEEE, vol 76, no. 8, August 1988.

Ciarcia, Steve. "Build a Gray-Scale Video Digitizer - Part 1". Byte, May 1987.

Ciarcia, Steve. "Build a Gray-Scale Video Digitizer - Part 2". Byte, JUNE 1987.

Duin, Robert P. W. and Pieter P. Jonker. "Processor Arrays Compared to Pipelines for Cellular Image Operations". Em Multicomputer Vision. Academic Press Limited, 1988.

Freeny, S.L. "Special-Purpose Hradware for Digital Filtering". Proceedings of the IEEE, vol 63, no. 4, April 1975.

IBM PCxt Technical Hardware Reference.

Mérigot A. "Designing Memories for Cellular Processors". Em Multicomputer Vision. Academic Press Limited, 1988.

Owczarczyk, J. "Design of PE structure of a paralell imagem processor". Eletronics Letters, January 29, 1987. vol 23, no. 3.

Potter, J. L.. "Image Processing on Massively Parallel Processor". Computer, January 1983. Vol 16, nº 1.

Preston Jr., Kendall. "Cellular Logic Computers for Pattern Recognition". Computer, January 1983. Vol 16, nº 1.

Ramamoorthy, C. V. "Pipeline Architecture". Computing Surveys, Vol. 9, nº 1, March 1977.

Robertson, William. "Interface your PC to the Multibus". EDN. January 8, 1987

Rosenfeld, Azriel. "Paralliel Image Using Cellular Arrays". Computer, January 1983. Vol 16, nº 1.

Stout, Q. F. "Mapping Vision Algorithms to Paralell Architectures". Proceedings of the IEEE, vol 76, no. 8, August 1988.

Taub, Herbert and Donald Shilling. "Digital Integrated Eletronics".MacGraw Hill, Inc. 1977.

"VRAM" , Mos Memory Texas Manual, 1984.

Yalamanchili S. and J. K. Aggarwall. "Analysis of a Model for Paralell Image Processing". Pattern Recognition. Vol 18 nº 1. 1985.

Yalamanchili S. and J. K. Aggarwall. "A System Organization for Paralell Imagem Processing". Pattern Recognition. Vol 18 nº 1. 1985.