UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE FÍSICA DE SÃO CARLOS

DESENVOLVIMENTO DE UM PROCESSADOR PIPELINE DEDICADO PARA EXTRAÇÃO DE BORDAS EM TEMPO REAL

Maximiliam Luppe

Dissertação apresentada ao Instituto de Física de São Carlos, da Universidade de São Paulo, para obtenção do título de Mestre em Ciências: Física Aplicada

Orientador: Prof. Dr. Valentin Obac Roda

São Carlos
1997

Luppe, Maximiliam
Desenvolvimento de uma arquitetura pipeline dedicada para extração de bordas em tempo real/Maximiliam Luppe. -- São Carlos,1997
91 p.

Dissertação (Mestrado) -- Instituto de Física de São Carlos, 1997

Orientador: Prof. Dr. Valentin Obac Roda

1. Processamento de imagens. 2. Deteção de bordas. 3. Arquitetura pipeline
I. Título

UNIVERSIDADE DE SÃO PAULO
Instituto de Física de São Carlos

Av. Dr. Carlos botelho, 1465
CEP 13560-250 - São Carlos - SP
Brasil

Fone (016) 274-3444
Fax (016) 272-2218

MEMBROS DA COMISSÃO JULGADORA DA DISSERTAÇÃO DE MESTRADO DE **MAXIMILIAM LUPPE** APRESENTADA AO INSTITUTO DE FÍSICA DE SÃO CARLOS, DA UNIVERSIDADE DE SÃO PAULO, EM 24 DE JUNHO DE 1997.

COMISSÃO JULGADORA:

Prof. Dr. Valentin Obac Roda/IFSC-USP

Prof. Dr. Eduardo Marques/ICMSC-USP

Prof. Dr. José Hiroki Saito/UFSCar

DEDICATÓRIA

Dedico este trabalho a Deus e a minha família, pois, sem eles, eu não existiria.

AGRADECIMENTOS

Agradeço ao Prof. Dr. Valentin, por ter aceitado orientar-me neste trabalho de Mestrado e pelo seu carisma.

E a todas as pessoas que estiveram ao meu lado durante este trabalho.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

ASIC	(Application-Specific Integrated Circuit)
CAD	(Computer-Aided Design)
CA/D	(Conversor Analógico-Digital)
CCD	(Charge Coupled Device)
CD/A	(Conversor Digital-Analógico)
DSP	(Digital Sinal Processor)
FFT	(Fast Fourier Transform)
FPGA	(Field Programmable Gate Array)
PAL	(Programable Array Logic)
PDA	(Personal Digital Assistant)
MIMD	(Multiple Instructions, Multiple Data)
MSPS	(Mega Samples Per Second)
TTL	(Transistor-Transistor Logic)
VLSI	(Very Large Scale Integration)

RESUMO

A detecção de bordas é um primeiro passo importante no processamento digital de imagens, pois permite separar blocos distinto presentes em uma imagem. O desenvolvimento de sistemas autônomos que realizem tarefas a partir de informações visuais necessitam que o processamento destas seja realizado em tempo real. Este trabalho descreve a implementação de um processador, baseado numa arquitetura *pipeline* e no operador de máscara chamado Cruz de Roberts, dedicado para a extração de bordas em tempo real de imagens de vídeo. Tanto a entrada como a saída dos dados são em formato de vídeo composto monocromático padrão e foram utilizados circuitos digitais discretos para a implementação do processador. Os resultados são apresentados em forma de imagens e estas são comparadas com os resultados obtidos através de programas que realizam a detecção de bordas.

ABSTRACT

Edge detection is the first important step in digital image processing which allows to separate the distinct blocks present in an image. The development of automatic systems which perform operations from visual information need real time processing. This work describes the implementation of a pipeline processor based in the Robert's Cross mask operator, dedicated to extract edges from video images in real time. Both input and output video signals are monochromatic. Common digital circuits have been used to implement the processor. The results obtained are presented as images and are compared with edge detected images obtained from comercial software.

1 INTRODUÇÃO

Atualmente os sistemas para Processamento Digital de Imagens estão ganhando cada vez mais importância em áreas que vão da multimídia às aplicações militares, da robótica aos sistemas críticos de segurança, da inspeção em linhas de produção às imagens de satélites. Requisitos básicos de capacidade de processamento, tamanho e consumo nem sempre podem ser fornecidos por sistemas de propósito geral e arquiteturas dedicadas estão sendo desenvolvidas para ultrapassarem estes limites.

Os avanços tecnológicos na fabricação de circuitos digitais com grande escala de integração (VLSI - *Very Large Scale Integration*) tem permitido a implementação de sistemas para Processamento Digital de Imagens cada vez mais complexos, e, principalmente, mais rápidos, possibilitando o desenvolvimento de sistemas para o processamento em tempo real.

O desenvolvimento de circuitos do tipo ASIC (*Application-Specific Integrated Circuit*) e FPGA (*Field Programmable Gate Array*), além de ferramentas computacionais (CAD-*tools*), tem possibilitado a rápida análise e implementação dos

sistemas de Processamento Digital de Imagens, diminuindo os custos e os erros de implementação.

Neste trabalho apresentaremos a implementação de um sistema para Processamento Digital de Imagens que será utilizado como uma forma de pré-processamento de imagens, podendo ser aplicado em outros projeto. A detecção de bordas foi tomada como exemplo, dentre as várias aplicações dos sistemas de Processamento Digital de Imagens, por ser a mais básica informação que podemos obter de uma imagem.

No Capítulo 2 faremos uma abordagem aos sistemas para Processamento Digital de Imagens, descrevendo seu histórico e sua composição, além das possíveis área de aplicação. No Capítulo 3 apresentaremos as definições de borda, onde analisaremos alguns trabalhos já expostos, e apresentaremos um detector de borda bastante conhecido, o operador Cruz de Roberts (*Roberts' Cross*), descrevendo as suas características e possíveis formas de implementação. Finalmente, no Capítulo 4, apresentaremos o desenvolvimento e implementação de um sistema para a detecção de bordas em tempo real. Este sistema é formado por um processador de arquitetura *pipeline* de 4 estágios, implementado com circuitos lógicos discretos (portas lógicas) e baseado no operador Cruz de Roberts, e por um conjunto de circuitos periféricos. Neste Capítulo também estudaremos o formato dos dados que serão utilizados pelo sistema e como este pode interferir no desenvolvimento do projeto final.

Os testes e simulações desta arquitetura foram feitos utilizando-se uma ferramenta computacional para implementação de circuitos lógicos em FPGAs. Esta ferramenta foi adquirida por meio de um convênio entre o Grupo de Instrumentação Eletrônica e Informática (GII) e a empresa Altera e está prevista a implementação da arquitetura aqui apresentada em um circuito FPGA da Altera, além de outras futuras arquiteturas para o Processamento Digital de Imagens. As informações a respeito desta implementação serão apresentadas sob forma de Apêndice.

2 PROCESSAMENTO DIGITAL DE IMAGENS

2.1 INTRODUÇÃO

Neste capítulo faremos um breve levantamento histórico das motivações que levaram ao desenvolvimento de sistemas de Processamento Digital de Imagens. Analisaremos as suas áreas de aplicação e descreveremos como um sistema de Processamento Digital de Imagens pode ser dividido.

2.2 HISTÓRICO

A visão humana tem sido objeto de estudo há longo tempo, mas somente nos últimos 40 anos é que o Processamento Digital de Imagens tem evoluído largamente e se tornado um interessante objeto de estudo.

Uma das primeiras aplicações das técnicas de processamento de imagens foi para o aperfeiçoamento das imagens jornalísticas enviadas por um cabo submarino entre Londres e Nova York. A introdução do sistema de transmissão de imagens à

cabo "Bartlane" na década de 20 reduziu o tempo necessário para enviar uma imagem através do Oceano Atlântico de mais de uma semana para menos de três horas[1][2].

De 1964, quando as imagens da Lua transmitidas pela nave espacial Ranger-7 foram processadas por um computador para corrigir vários tipos de distorções inerentes ao sistema de câmeras a bordo, até o presente, os campos de Processamento Digital de Imagens têm crescido vigorosamente. Em adição às aplicações nos programas espaciais, as técnicas de Processamento Digital de Imagens agora são usadas para resolver uma variedade de problemas, tais como contraste de imagens médicas, remoção de ruídos e interferências, correção de distorções como movimento de câmera durante a exposição de uma foto. Ainda que freqüentemente não sejam relatados, esses problemas normalmente requerem métodos capazes de realçar a informação pictórica para a análise e a interpretação humana.

Até o final da década de 70, grande parte do processamento digital de imagens era realizado por computadores de grande porte e de aplicações genéricas. A partir da década de 80, com o surgimento dos Processadores de Sinais Digitais (DSPs - *Digital Sinal Processors*) e a possibilidade de implementar Processadores Matriciais (*Array Processors*) e arquiteturas paralelas com grande escala de integração, começaram a surgir sistemas específicos para o processamento de imagens, muito mais eficientes e rápidos, capazes de processar imagens em taxas de vídeo.

Um exemplo de sistemas para Processamento Digital de Imagens da década de 80 são os sistemas modulares MaxVídeo da Datacube, baseados em barramento VME. Estes sistemas modulares eram formados de um conversor analógico digital (Digimax) com taxa de conversão de 10MHz, um módulo de armazenamento de 384x512x8*bits* (Framestore), um processador linear de pixel (VFIR), um arranjo sistólico de vizinhança para processamento de máscaras 3x3 (SNAP), um extrator de histograma em tempo real (Featuremax) e um módulo DSP de uso geral (MaxSP) [3] [4] [5].

Na FIGURA 1 temos o exemplo de uma configuração de Processamento Digital de Imagens em tempo real utilizada para a detecção de bordas através do Operador Sobel, utilizando os módulos do sistema MaxVideo. As máscaras para detectar os gradientes nas direções x e y são implementadas pelos módulos VFIR e os cálculos do gradiente e da direção das bordas são realizados por meio de uma tabela de uso geral, presente no módulo MaxSP[4].

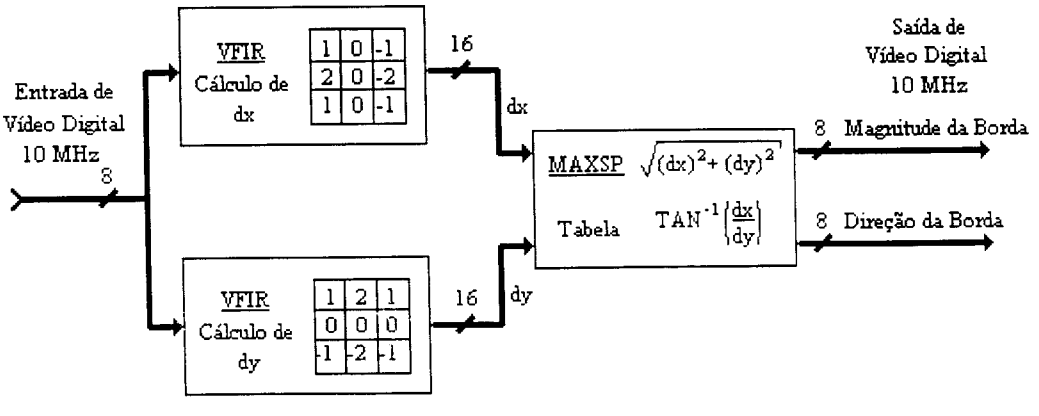

FIGURA 1 - Exemplo de aplicação dos módulos MaxVideo

Atualmente, para poder implementar complexos algoritmos de processamento digital de sinais em arquiteturas dedicadas, no menor tempo possível, metodologias de desenvolvimento e ferramentas de projeto auxiliado por computadores (CAD-*tools*) têm sido utilizadas [6][7][8]. Estas ferramentas têm permitido mapear os complexos algoritmos em circuitos ASICs (*Application-Specific Integrated Circuits*) e FPGAs (*Field Programmable Gate Arrays*), fechando um abismo que existia entre o desenvolvimento dos sistemas e a sua real implementação, além de diminuir os custo e erros de implementações.

2.3 ÁREAS DE APLICAÇÃO

Segundo Gonzalez [1], podemos distinguir dois grandes focos para as aplicações de Processamento Digital de Imagens: o primeiro seria o melhoramento das informações pictóricas para a interpretação do próprio ser humano; e a segunda seria a extração de informações em um formato acessível à interpretação automática por meio de máquinas.

No primeiro caso temos os processamentos que ajudam a corrigir erros e a melhorar imagens cuja reprodução seja crítica. Neste caso, temos as radiografias de Raio-X, onde a excessiva exposição a estes raios poderia causar danos irreversíveis à saúde do paciente. Por isso, técnicas de contraste, como filtros passa-alta e FFT (*Fast Fourier Transform* - Transformada Rápida de Fourier), e de atenuação, como filtros medianos, são extremamente úteis neste caso (FIGURA 2).

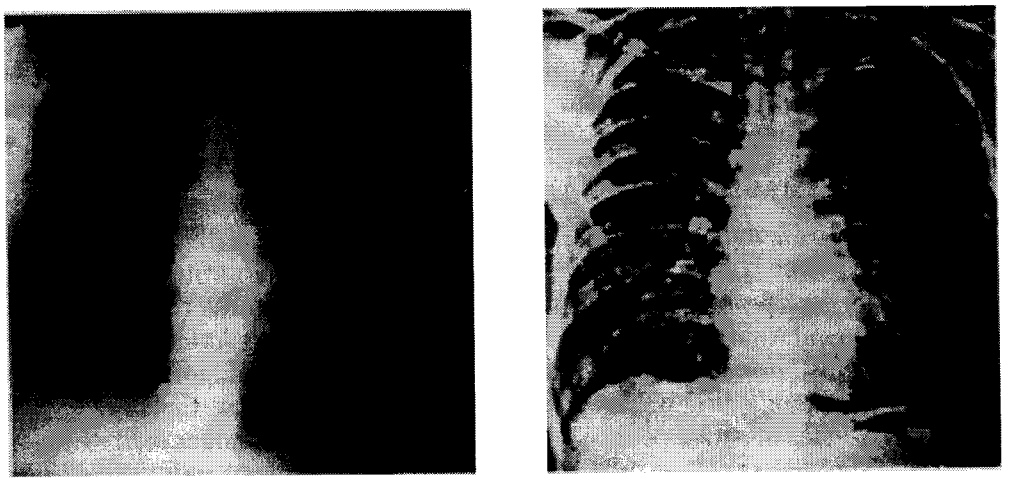

FIGURA 2 - Primeiro exemplo de restauração de uma imagem. A imagem da esquerda é a imagem de Raio-X original e a da direita é a imagem restaurada por meio do melhoramento de contraste

Um outro exemplo é a correção de distorções causadas pelo movimento de câmeras durante a exposição de uma fotografia (FIGURA 3). Estas distorções podem ser corrigidas com o auxílio de algoritmos especiais. Também temos as distorções causadas pela própria câmera, como foco irregular e distorções na lente, que também poderiam ser sanados com algoritmos especiais.

FIGURA 3 - Segundo exemplo de restauração de uma imagem. A imagem da esquerda está borrada devido a um movimento uniforme da câmera durante a exposição e a da direita é o resultado é a imagem recuperada.

A segunda maior área de aplicação das técnicas de Processamento Digital de Imagens está em resolver problemas relacionados com a percepção de máquinas. Nesse caso, o interesse foca-se em procedimentos para extrair, de uma imagem, informações em um formato específico para que a análise das imagens possa ser feita de uma modo computacional e automático.

Problemas típicos em processamento automático que rotineiramente utilizam técnicas de Processamento Digital de Imagens são reconhecimento automático de caracteres, visão automática industrial para montagem e inspeção, reconhecimento militar, processamento automático de impressões digitais, e outras mais.

O reconhecimento automático de caracteres pode ser aplicado tanto em uma empresa de correios, para fazer a seleção das cartas por meio do reconhecimento do Código de Endereçamento Postal (CEP), como em PDAs (*Personal Digital Assistants*), como o Apple Newton MessagePad [9], que reconhece a grafia e converte as letras para um formato padrão ASCII.

Na indústria, a visão computacional pode ser aplicada na linha de montagem, tanto para a inspeção de peças, como no alinhamento destas. Nestes casos existe uma grande necessidade de processamento em alta velocidade, para que estas inspeções não acarretem em atrasos na linha de montagem.

Outra aplicação do Processamento Digital de Imagens automático é a medida e análise de parâmetros de tráfego, tais como volume, velocidade e tipos de veículos

[10]. Estes sistemas permitam o monitoramento das rodovias durante todo o dia e em qualquer condição de tempo, diminuindo os custos com pessoal e os erros devidos às más condições de tempo.

2.4 ELEMENTOS DE UM SISTEMA PARA PDI

O projeto de um sistema para Processamento Digital de Imagens deve conter alguns elementos que serão descritos a seguir:

2.4.1 Aquisição da imagem

Neste item temos a conversão do sinal analógico, proveniente de um sensor em um sinal digital, por meio de um conversor analógico-digital. A escolha destes dois componentes (sensor e conversor) irá depender da aplicação do sistema de Processamento Digital de Imagens. Os sensores podem ser câmeras convencionais, que produzem uma imagem a cada 1/30s, câmeras do tipo Vidicon, que podem produzir imagem do espectro infravermelho e podem ser utilizadas para a inspeção de peças ou equipamentos cuja temperatura deva ser constantemente monitorada, sensores do tipo CCD, lineares ou bi-dimensionais, fotocaptadoras para a obtenção de imagens de Raio-X, e outros.

Os principais parâmetros do conversor são a resolução e a taxa de conversão, ou taxa de amostragem. Em sistemas convencionais de processamento de imagens normalmente são utilizados conversores de 8*bits* de resolução e com taxas de

conversão da ordem de 10MHz. A escolha destes valores deve-se ao fato de ser comum o uso de câmeras de vídeo convencionais, que trabalham com uma largura de banda da ordem de 4MHz, e que, com estes parâmetros, é possível obter imagens de 512x512x8*bits*, o que é um valor adequado para o processamento de imagens e para a visualização do ser-humano, sem que este perceba alguma degradação na imagem "digitalizada" [1][11].

2.4.2 Armazenamento

Um outro problema que surge com o aumento da taxa de amostragem e da resolução do conversor está relacionado com o armazenamento das imagens. Quanto maior for a taxa de conversão, maior será a quantidade de dados obtidos, e maior será a necessidade de memória para o armazenamento desses dados. Da mesma forma, o aumento da resolução do conversor também implicará na necessidade do aumento da capacidade de armazenamento. Por isso, deve-se equilibrar estes dois parâmetros do conversor a fim de se obter uma imagem nítida, sem perda de informações, mas que possa ser facilmente, e com o menor custo possível, armazenada. Os valores apresentados acima estão de acordo com os valores mais comuns encontrados na literatura e se adequam aos propósitos deste trabalho.

2.4.3 Processamento

O processamento total envolve várias etapas, desde a aquisição dos dados até a tomada de uma decisão, passando por pré-processamento, segmentação, descrição, reconhecimento e descrição [1][4], conforme mostra a FIGURA 4. A aquisição dos

dados é realizada por circuitos especializados (conversores), conforme foi mostrado na seção 2.4.1. Uma vez armazenado os dados, estes poderão ser processados por um sistema computacional, que poderá ser composto por um conjunto de arquiteturas dedicadas (especializadas), se houver a necessidade de tempo de processamento, ou por um computador pessoal, se houver a necessidade de baixo custo. As etapas que são realizadas por este sistema são descritas abaixo e tomam como exemplo um sistema para reconhecimento do Código de Endereçamento Postal (CEP) de correspondências em uma agência de correios[1].

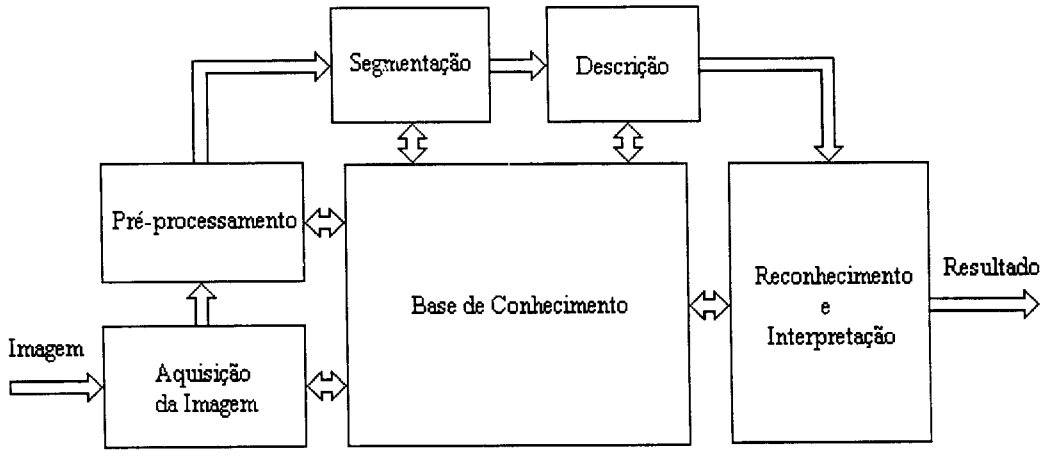

FIGURA 4 - Etapas do processamento de uma imagem

O Pré-processamento tem como função melhorar a imagem de forma a aumentar as chances de sucesso no processamento. Isto pode ser conseguido por meio de filtros medianos que servem para reduzir os efeitos do ruído, ou por meio de filtros passa-alta para aumentar o contraste da imagem e o isolamento de regiões que possuem características úteis para o sistema, ou para o ser-humano.

Se o sistema para Processamento Digital de Imagens fosse utilizado somente para o melhoramento da informação pictórica para o ser-humano, a etapa de processamento pararia aqui, indo direto para a apresentação do resultado. Além do mais, esta etapa de pré-processamento poderia ser realizada por uma arquitetura dedicada e de baixo custo, com o intuito de diminuir o tempo de processamento caso este fosse realizado por um computador pessoal. Dessa forma, esta arquitetura poderia ser utilizada tanto para o melhoramento da informação pictórica para o ser humano, como para uso em um sistema computacional e automático.

A próxima etapa do processamento é a Segmentação da imagem. Nesta etapa o sistema realmente começa a processar a imagem, com o intuito de tomar uma decisão, extraindo informações da imagem original que são importantes para o processamento automático. Aqui, a imagem original é dividida em seus objetos constituintes. Como exemplo vamos tomar o caso de reconhecimento de caracteres. Nesta etapa é feita a extração os caracteres e palavras da imagem original. Nesta etapa a imagem está sendo tratada para aumentar as chances de sucesso do processamento e ainda não é realizada nenhuma forma de reconhecimento, ou interpretação, mas somente a separação de áreas "úteis" da imagem.

Na Descrição são extraídas informações quantitativas que permitem ao sistema diferenciar os objetos extraídos na etapa anterior e separá-los em várias classes. Em termos de reconhecimento de caracteres, buracos e saliências são "descritores" importantes que ajudam as distinguir e diferenciar as várias letras do alfabeto, uma das outras.

A última parte do processo envolve o Reconhecimento e a Interpretação. O Reconhecimento é o processo de identificar e dar um nome a cada um dos objetos descritos e a Interpretação permite dar significado ao que foi reconhecido. Se foram reconhecidos vários conjuntos de caracteres, é necessário interpretar estes conjuntos e separá-los em palavras que possuam algum significado.

No caso do reconhecimento de Código de Endereçamento Postal (CEP) em correspondências, os numerais já teriam sido identificados, e, nesta etapa final, seriam separados em uma cadeia de 5 numerais, seguidos por um hífen, e depois por mais 3 numerais. A partir daqui seria, então, definido o destino da correspondência.

2.4.4 Apresentação

A última parte do sistema de Processamento Digital de Imagens trata de como os dados obtidos serão apresentados. Se o resultado final for a melhoria das informações pictóricas, os resultados deverão ser apresentados de uma forma clara para o ser-humano. Para isso podem ser utilizados monitores gráficos ou impressoras de alta qualidade. Se as imagens foram obtidas através de câmeras de vídeo, seria conveniente que os resultados fossem apresentados no mesmo formato, ou seja, sinal de vídeo, e fossem apresentados em monitores ou televisores.

Caso o resultado do processamento seja a obtenção de informações para um sistema automático, existe a necessidade de alguma forma de visualização para o ser-

humano poder intervir em caso de falha, mal-funcionamento ou erro na decisão tomada pelo sistema. Esta visualização serviria apenas como monitoramento.

2.5 CONCLUSÕES

Neste capítulo apresentamos as duas possíveis áreas para a aplicação de sistemas para Processamento Digital de Imagens: a primeira relacionada com a melhoria da informação pictórica para uso do próprio ser humano e a segunda relacionada com o processamento automático de sistemas computacionais. Foi dado um maior destaque para o segundo caso, onde apresentamos os elementos e etapas de um sistema para o Processamento Digital de Imagens automático genérico.

3 BORDAS E DETECTORES DE BORDA

3.1 INTRODUÇÃO

Neste capítulo estudaremos as características de uma borda em uma imagem e analisaremos como detectores podem ser implementados em *hardware*, verificando a necessidade do processamento em tempo real.

3.2 CARACTERÍSTICAS DE UMA BORDA

Uma borda pode ser definida como uma descontinuidade ou mudança abrupta nos níveis de intensidade luminosa de uma imagem [2]. Em geral, imagens podem conter uma variedade de tamanhos de bordas, algumas curtas e outras longas. O que é mais importante, contudo, é que estas mudanças de intensidade luminosa podem ocorrer em qualquer direção.

As bordas são importantes para os animais e para o homem, e também úteis para sistemas de visão computacional. Isto porque elas fornecem uma excelente informação a respeito da forma do objeto na imagem. Apesar disso, elas são

freqüentemente incompletas e degradadas quando comparadas com um modelo abstrato.

Modelos idealizados de borda podem ser representados pelas funções de singularidade convencionais da matemática. Por exemplo, uma mudança abrupta na intensidade pode ser definida por uma função degrau ideal, como mostra a FIGURA 5.*a*. Muito provavelmente isto é o que nós idealizamos quando pensamos num perfil de borda. Cenas com objetos sólidos feitos de superfícies planas podem exibir ângulos agudos em sua caracterização de intensidade *versus* distância. Por exemplo, a FIGURA 5.*b* mostra uma borda dente-de-serra ideal, que pode ser vista como sendo feita de duas funções de rampa, uma ascendente e outra descendente. Nós podemos também combinar os perfis de rampa e degrau para obter um perfil bem comum, modelado como mostra a FIGURA 5.*c*. Imagens contendo linhas que são bastante contrastantes podem exibir um perfil conforme mostra a FIGURA 5.*d*, onde temos um pulso feito de dois perfis de degrau.

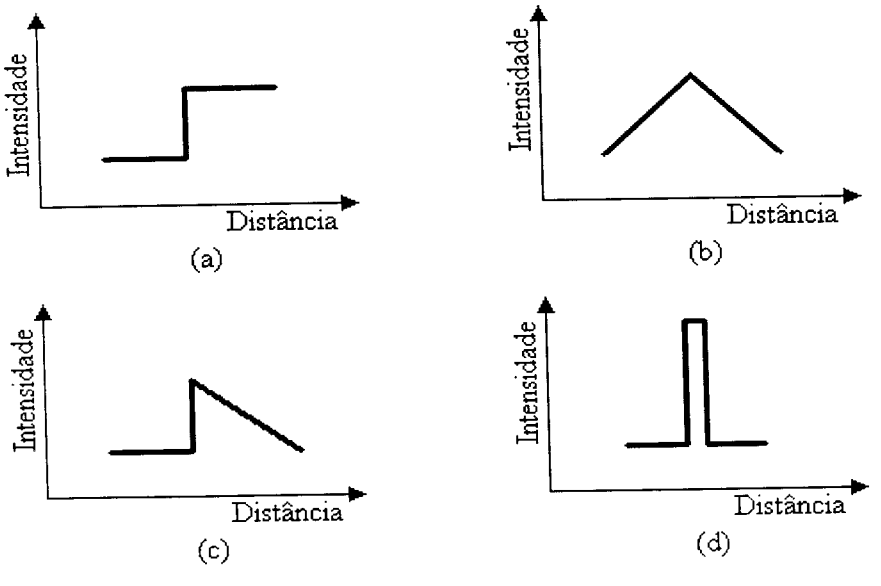

FIGURA 5 - Exemplos de perfis ideais de bordas

A origem destes perfis unidimensionais claramente relaciona-se à geometria particular de imagens tridimensionais. Para poliedros rígidos e opacos, as bordas numa imagem ocorrem na interseção das superfícies. Diferentes tipos de bordas podem ser observadas em um mapa de profundidade, conforme mostrado na FIGURA 6. Esta bordas são os gráficos abaixo de cada objeto e "traçam" a distância do observador ao ponto mais próximo na superfície do objeto a um dado ângulo α. Note que, nas FIGURA 6.*a* e *b*, a varredura não inclui descontinuidade, uma vez que o gráfico é liso. A borda côncava (α_1) na FIGURA 6.*c* e a borda convexa (α_2) na FIGURA 6.*d* resultam de uma descontinuidade em *r*, com o sinal da tangente definindo o tipo. Bordas ocultas ocorrem quando há uma descontinuidade na profundidade, como visto na FIGURA 6.*d*, onde as duas superfícies envolvidas do objeto não se tocam (α_3). Uma quinta categoria de borda seria a "borda de contorno", que delineia a borda entre o objeto e o seu fundo de imagem. Isso ocorreria na FIGURA 6.*a*, *b*, *c* e *d* se o "rastreamento" continuasse para a extrema direita ou esquerda do poliedro. Em geral, ainda não foi determinada uma maneira de transformar perfis de bordas em mapas de profundidade.

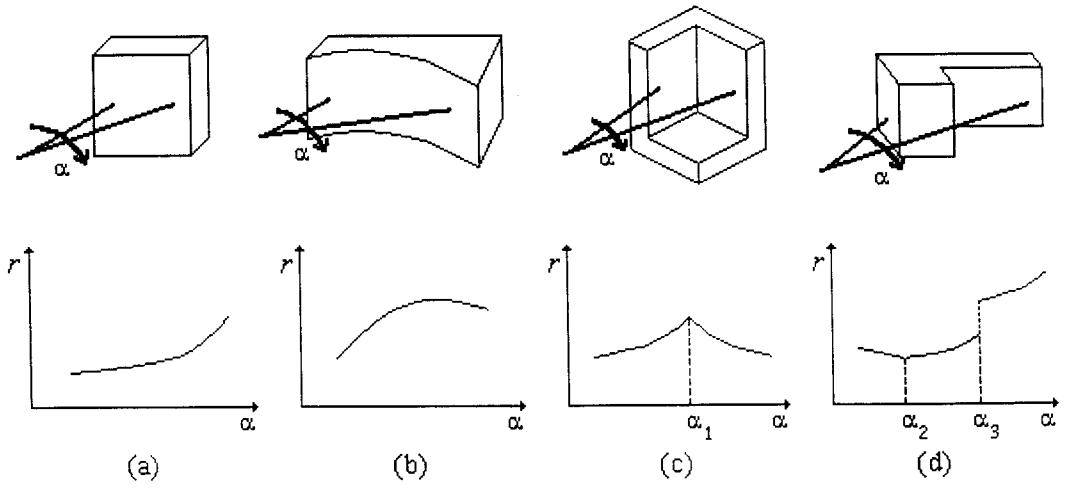

FIGURA 6 - Mapas de profundidade

Certos perfis específicos em imagens podem algumas vezes serem relacionados a tipos particulares de superfícies de objetos. Por exemplo, um pico ou um degrau com um pico sobreposto é provavelmente o resultado de uma borda convexa. Do outro lado, um dente-de-serra ou um degrau com um dente-de-serra sobreposto corresponderia mais a uma borda côncava. Finalmente um degrau, um pico negativo ou um degrau com um pico negativo sobreposto ocorrem tanto em bordas côncavas como em convexas na cena original.

Apesar de existir uma grande distância entre os perfis de borda ideais e os que são encontrados na natureza, estes podem ser utilizados para o estudo de detetores de bordas, como em [12], onde encontramos um estudo do comportamento destes perfis ideais de borda, quando é aplicado o gradiente normalizado do operador Gaussiano.

3.3 DETETORES DE BORDA

Muito já se tem estudado a respeito da detecção de borda [13-21]. O que podemos notar é que, sendo a borda uma variação abrupta nos níveis de intensidade de uma imagem, é comum o uso de operadores diferenciais para realizar a sua detecção. Outra característica é que, dada uma imagem qualquer, detector de borda eficiente deve necessariamente ter a capacidade de distinguir contraste em diferentes ângulos.

3.3.1 Operador Cruz de Roberts

O método mais comum de diferenciação em aplicações de processamento de imagens é o gradiente. Para uma função **f**(x,y), o gradiente de **f** nas coordenadas (x,y) é definido como o vetor

$$\nabla \mathbf{f} = \begin{bmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{bmatrix} \tag{1}$$

A magnitude deste vetor,

$$\nabla f = \text{mag}(\nabla \mathbf{f}) = \left[\left(\frac{\partial f}{\partial x} \right)^2 + \left(\frac{\partial f}{\partial y} \right)^2 \right]^{1/2} \tag{2}$$

é a base de várias aproximações para diferenciação de imagens e cujo valor determina a existência, ou não, de uma borda. Considere a região de imagens na FIGURA 7 , onde os *I*s denotam os valores dos níveis de cinza. A Eq. (2) pode ser aproximada no ponto $I_{2,3}$ de várias formas. A mais simples é usar a diferença ($I_{2,3}$ - $I_{2,4}$) na direção *x* e ($I_{2,3}$ - $I_{3,3}$) na direção *y*, combinada como:

$$\nabla f \approx \left[\left(I_{2,3} - I_{2,4} \right)^2 + \left(I_{2,3} - I_{3,3} \right)^2 \right]^{1/2} \tag{3}$$

$I_{1,1}$	$I_{1,2}$	$I_{1,3}$	$I_{1,4}$	$I_{1,5}$	$I_{1,6}$
$I_{2,1}$	$I_{2,2}$	$I_{2,3}$	$I_{2,4}$	$I_{2,5}$	$I_{2,6}$
$I_{3,1}$	$I_{3,2}$	$I_{3,3}$	$I_{3,4}$	$I_{3,5}$	$I_{3,6}$
$I_{4,1}$	$I_{4,2}$	$I_{4,3}$	$I_{4,4}$	$I_{4,5}$	$I_{4,6}$

FIGURA 7 - Pedaço de uma imagem, onde Ii,j corresponde aos níveis de intensidade do sinal de vídeo

Ao invés de se utilizar quadrados e raízes quadradas, nós podemos obter resultados similares usando valores absolutos:

$$\nabla f \approx \left|I_{2,3} - I_{2,4}\right| + \left|I_{2,3} - I_{3,3}\right| \tag{4}$$

O gradiente também pode ser calculado utilizando-se uma aproximação para o máximo valor entre os dois gradientes:

$$\nabla f \approx max\left\{\left|I_{2,3} - I_{2,4}\right|, \left|I_{2,3} - I_{3,3}\right|\right\} \tag{5}$$

Outra aproximação para as equações anteriores é usar o cálculo do gradiente nas diagonais, ao invés de usar o gradiente nas direções x e y, resultando nas diferenças cruzadas:

$$\nabla f \approx \left[\left(I_{2,3} - I_{3,4}\right)^2 + \left(I_{2,4} - I_{3,3}\right)^2\right]^{1/2} \tag{6}$$

ou, utilizando-se valores absolutos:

$$\nabla f \approx \left|I_{2,3} - I_{3,4}\right| + \left|I_{2,4} - I_{3,3}\right| \tag{7}$$

e

$$\nabla f \approx max\left\{\left|I_{2,3} - I_{3,4}\right|, \left|I_{2,4} - I_{3,3}\right|\right\} \tag{8}$$

As equações acima, também conhecidas como operadores não-lineares $\Lambda(x,y)$, podem ser implementadas utilizando-se máscaras de tamanho 2x2, conforme mostra a FIGURA 8. Estas máscaras são denominadas W_1 e W_2 e são conhecidas como operadores de gradiente Cruz de Roberts [13].

1	0
0	-1

W_1

0	1
-1	0

W_2

FIGURA 8 - Máscaras W1 e W2 do operador Cruz de Roberts

Este detetor de bordas foi utilizado por Roberts no seu trabalho de reconhecimento automático de objetos tridimensionais. De acordo com Roberts, "a escolha do operador diferencial é muito crítica e várias variações foram utilizadas. Três critérios principais foram utilizados para se escolher qual operador seria utilizado. As bordas produzidas deveriam ser bem definidas, o fundo deveria produzir o menor ruído possível e a intensidade das linhas produzidas deveria corresponder o mais próximo possível à habilidade humana em perceber as bordas na imagem original. A definição das bordas depende do número de amostras usadas pelo operador. O ruído de fundo parece ser reduzido utilizando-se operadores simétricos

em x e y. De forma a fazer com que bordas iguais tenham gradientes iguais, os valores de intensidade da imagem podem ser sujeitos a uma mudança de forma a fazer com que as diferenças de intensidade sejam proporcionais à habilidade humana de percebê-las. De acordo com a teoria psicofísica, a raiz quadrada da intensidade deve ser utilizada de forma a obter o efeito desejado."

Desta forma, Roberts utilizou o operador descrito pela Eq. (6), só que ao invés de utilizar os valores de intensidade $I_{i,j}$, utilizou a raiz quadrada deste, $\sqrt{I_{i,j}}$, a fim de obter o efeito psicofísico desejado.

3.3.2 Implementações

A forma mais rápida de se implementar qualquer operador para a detecção de bordas é por meio de programas (implementações em *softwares*). Este tipo de implementação permite o rápido desenvolvimento e análise dos operadores, mas a desvantagem está no fato de que sistemas baseados nesse tipo de implementação não são adequados quando se necessita de processamento em tempo real. Sistema para o processamento em tempo real devem ser capazes de trabalhar com uma grande quantidade de dados (uma simples imagem pode conter 256k*bytes*) em uma alta taxa de transferência (até 20MSPS). Isto pode ser remediado por meio do uso de computadores de grande porte, mas teria como desvantagem o alto custo do sistema.

Outra forma de implementar operadores para a detecção de bordas é por meio de arquiteturas dedicadas, do tipo *pipeline* [6-8][22-23], *Data-flow* [25], ou mesmo

arquiteturas MIMD (*Multiple Instructions, Multiple Data* - Múltiplas Instruções, Múltiplos Dados) [26], que podem ser implementadas em circuito tipo ASIC [22] ou em VLSI [23-26]. Estas implementações permitem o processamento de imagens em tempo real e a sua complexidade irá depender do operador a ser implementado.

A implementação em *hardware* de operadores como a base ortogonal nano-dimensional proposta por Frei e Chen [20] é um tanto complexa, por envolver cálculos de raiz quadrada. Esta base permite obter vários operadores para a detecção de ponto, borda e linha por meio da combinação linear dos seus elementos. Para solucionar este problema, Haralick apresentou um conjunto completo de operadores baseados nos polinômios de Chebyshev generalizados para detecção de borda, linha e ponto em uma imagem[21]. Estes operadores efetuam apenas somas e subtrações e multiplicações por 1, 2 e 4, o que torna a implementação em *hardware* destes operadores de uma forma mais simples. Da mesma forma que os operadores ortogonais propostos por Frei e Chen, os operadores de gradiente mais comuns podem ser expressos pela combinação linear destes operadores propostos por Haralick. Estes novos operadores foram utilizados por Majumdar *et al.*[22] em um circuito ASIC para implementar o operador Sobel para detecção de bordas.

Com isso podemos verificar que se os elementos de um operador forem limitados a 0, ±1, ±2, ±4, a convolução entre o operador e a imagem pode ser realizada de uma forma bem simples utilizando-se apenas registradores de deslocamento e somadores, não necessitando de circuitos de cálculo complexos.

3.4 PROPOSTA DE UMA ARQUITETURA

A arquitetura proposta para a detecção, ou extração, de bordas de uma imagem é baseada no operador Cruz de Roberts apresentado anteriormente. A escolha deste operador deve-se à sua simplicidade e facilidade de implementação utilizando-se circuitos lógicos comuns. Abaixo temos a sua representação.

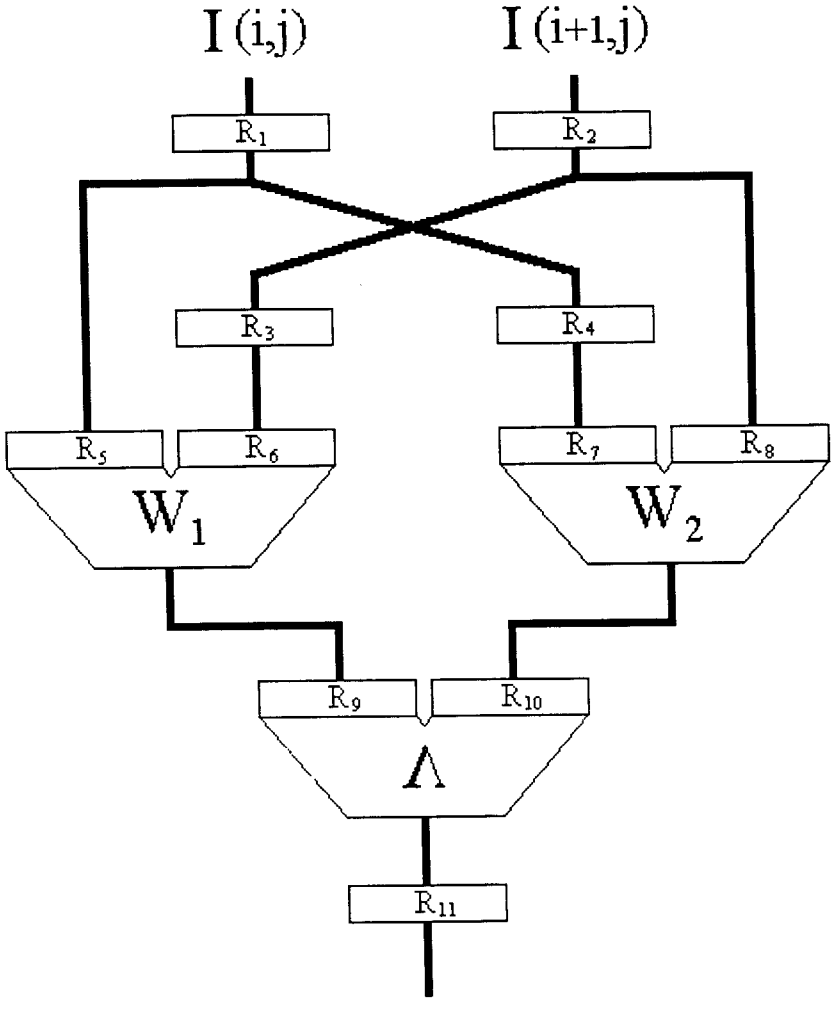

FIGURA 9 - Esquema da arquitetura para implementação do operador Cruz de Roberts. Nessa FIGURA W_1 e W_2 representam as operações de convolução entre a imagem e as máscaras W_1 e W_2, Λ representa a implementação do operador não linear e R são registradores.

Este processador possui basicamente três unidades aritméticas, sendo duas responsáveis pelos cálculos de convolução das máscaras W_1 e W_2 com a imagem e uma para o cálculo do operador não linear Λ.

Podemos perceber que deverão ser fornecidas à arquitetura duas linhas ($I_{i,j}$ e $I_{i+1,j}$) da imagem. Como a imagem é obtida através de um sinal de vídeo entrelaçado, estas linhas estarão em campos diferentes. Isto implica que o campo anterior sempre terá que estar disponível para o processamento do campo atual. Isto pode ser conseguido armazenando-se o campo anterior em uma memória.

Os quatro primeiros registradores (R_1 - R_4) realizarão os atrasos necessários para a obtenção dos elementos $I_{i,j}$, $I_{i+1,j+1}$ e $I_{i+1,j}$, $I_{i,j+1}$ da imagem, que serão enviados aos operadores de máscara W_1 e W_2 para o cálculo do gradiente. Os resultados destas duas convoluções são enviados ao operador não linear Λ, que determinará a existência ou não de uma borda nessa região da imagem. Todos os demais registradores (R_5-R_{11}) têm como função manter a seqüência correta dos dados dentro da arquitetura.

O processador, além da arquitetura proposta e da memória, irá necessitar de um conversor analógico-digital que possa trabalhar numa alta taxa de conversão, um conversor digital-analógico, para que o resultado possa ser visualizado e de uma lógica de controle que possa controlar todos. No próximo capítulo apresentamos a implementação desta arquitetura.

3.5 CONCLUSÕES

Neste capítulo abordamos um estudo sobre bordas e detectores de borda, por ser esta uma importante informação a ser obtida de uma imagem. Foi descrito o operador Cruz de Roberts, um detector bastante conhecido e de que forma ele poderia ser implementado em *hardware*, para fazer parte de um sistema de Processamento Digital de Imagens que atuasse como forma de pré-processamento, para poder ser utilizado em outro projetos.

A arquitetura do processador mostrou ser bem simples devido à simplicidade do operador Cruz de Roberts, mas a escolha do operador não linear poderá implicar num aumento da complexidade do processador.

4 ARQUITETURA PARA DETECÇÃO DE BORDAS EM TEMPO REAL

4.1 INTRODUÇÃO

Baseando-se nos elementos de um sistema de Processamento Digital de Imagens, explicaremos as partes que compõem o sistema para processamento de imagens em tempo real desenvolvido, conforme o esquema geral mostrado na FIGURA 10. A primeira parte deste sistema consiste na aquisição e armazenamento dos dados. Como os dados, neste caso uma imagem, são obtidos através de um sinal de vídeo composto monocromático padrão, analisaremos inicialmente o formato desse sinal de vídeo e as informações que ele fornece. Depois, apresentaremos a parte do sistema que faz a conversão desse sinal para o formato digital, ou seja, o "digitalizador", juntamente com as memórias, que farão o armazenamento dos dados, e o circuito de controle para a aquisição dos dados.

Na segunda parte, apresentaremos a implementação da arquitetura proposta no capítulo anterior, ou seja, o processador. Esta arquitetura, como já foi mencionado anteriormente, é do tipo *pipeline* e está baseada no operador Cruz de Roberts. Conforme vimos também, o operador não linear pode ser descrito de três formas

diferentes. Apresentaremos essas três formas, como elas podem ser implementadas e qual foi a implementação utilizada.

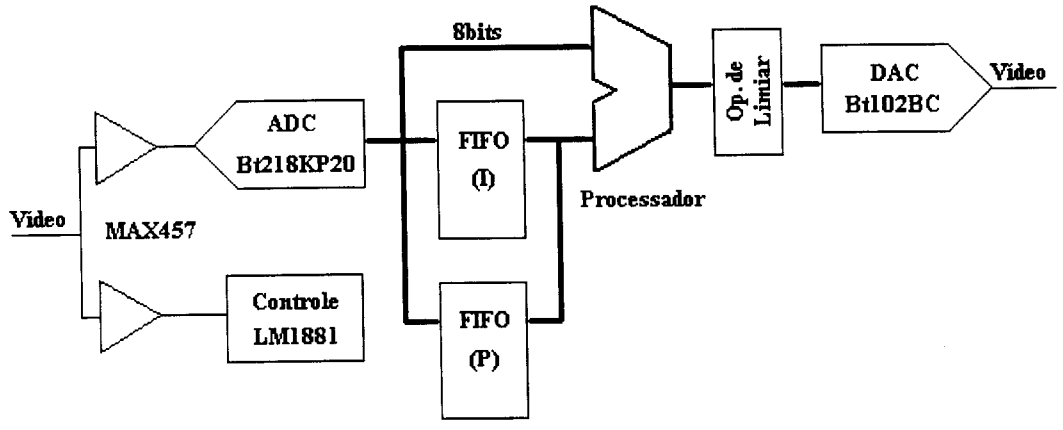

FIGURA 10 - Esquema geral do sistema para Processamento Digital de Imagens

As duas últimas parte do sistema constam da comunicação e da apresentação dos resultados. A comunicação se preocupa apenas em como transferir os dados até o sistema e do sistema para um monitor que possa apresentar os resultados.

4.2 AQUISIÇÃO DOS DADOS

4.2.1 Sinal de Vídeo Composto Monocromático Padrão

O sinal de vídeo composto padrão é um sinal analógico de $1V_{p-p}$ x 75Ω e recebe este nome porque contém as informações de luminância (Y), crominância (C) e sincronização, necessárias para formar uma imagem colorida completa [27][28]. Como utilizaremos apenas o sinal de vídeo monocromático, que será referido só por sinal de vídeo, não entraremos em detalhes com relação à crominância, informação esta que diz respeito às cores de uma imagem. E como existem vários tipos de sinais

de vídeo que possuem variações com respeito ao número de linhas, descreveremos apenas o padrão adotado no Brasil, que possui 525 linhas por quadro completo de imagem.

Um quadro completo de uma imagem possui 525 linhas que são divididas em dois campos, cada um com 262,5 linhas. Os campos são denominados par e ímpar, sendo que o campo par apresenta as linhas pares e o campo ímpar, as linhas ímpares. Na FIGURA 11 abaixo podemos ver como as linhas são distribuídas entre os dois campos. Cada campo é apresentado a cada 1/60s, o que implica que cada imagem completa será apresentada a cada 1/30s e cada uma das linhas dos campos apresentará um período de 63,5μs (1/30s ÷ 525 linhas), também conhecido por 1H.

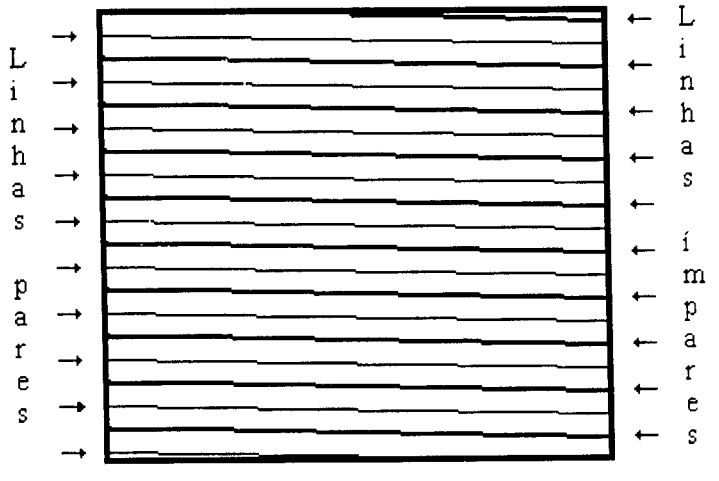

FIGURA 11 - Linhas de vídeo de um quadro completo

Cada linha possui as informações de sincronismo horizontal, crominância, no caso de um sinal colorido, e luminância, conforme mostra a FIGURA 12 abaixo.

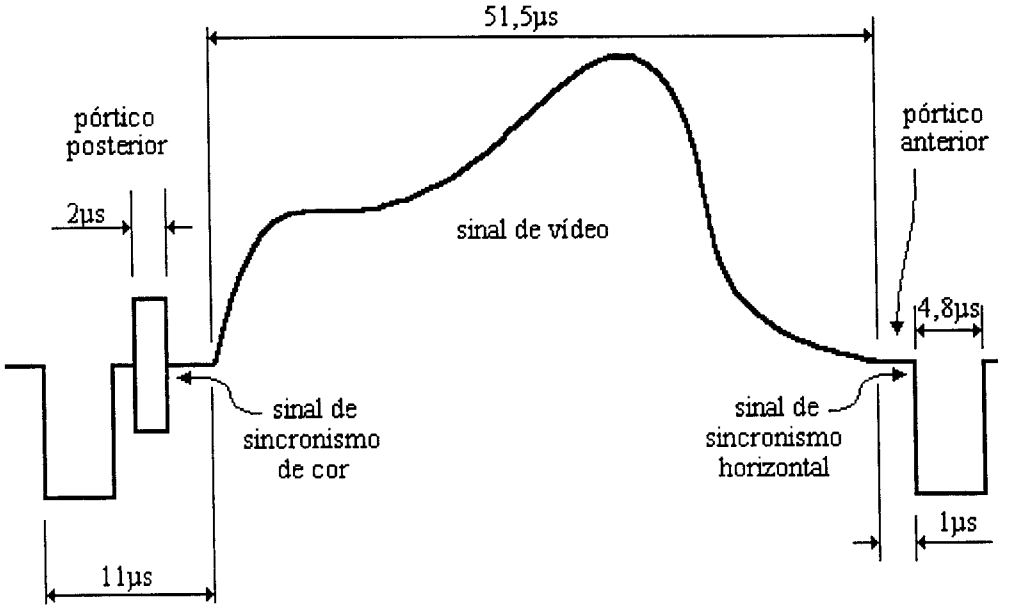

FIGURA 12 - Formato de uma linha de vídeo. Os valores apresentados são valores aproximados.

O período inicial de ~12µs de cada linha contém os sinais de sincronismo horizontal e de crominância. Inicialmente temos um período inicial de 1µs onde o sinal de vídeo fica no nível de preto, conhecido pórtico anterior ou *front porch*, que representa o final da última linha. A seguir temos o sinal de sincronismo horizontal, que tem um amplitude de -0,3V e um período de 4,8µs. Este sinal serve para sincronizar o sinal de vídeo com o monitor, permitindo que o circuito do monitor retorne o feixe de elétrons até o início da próxima linha. Depois do sinal de sincronismo horizontal, temos um período de 6,2µs que serve de referência de preto para o sinal de luminância e é conhecido por pórtico posterior ou *back porch*. Durante este período é enviado o sinal de sincronismo de cor, chamado *burst*, que tem uma duração de 2µs e a função de sincronizar o oscilador local de cor do monitor.

Então, durante os 12μs iniciais da linha de vídeo são enviados os sinais de fim de linha, nível de preto e de sincronismo de cor. Todo este período inicial é conhecido por retraço horizontal ou *horizontal blanking* e nenhuma informação de luminância é enviada.

A informação de vídeo propriamente dita é enviada como um simples sinal de luminância, cuja amplitude varia de 0V (nível de preto) até 0,7V (nível de branco), com os níveis de cinza entre estes dois extremos.

Para que o monitor possa diferenciar entre um campo e outro, é enviado o sinal de sincronismo vertical entre os dois campos. O formato deste sinal pode ser visto na FIGURA 13.

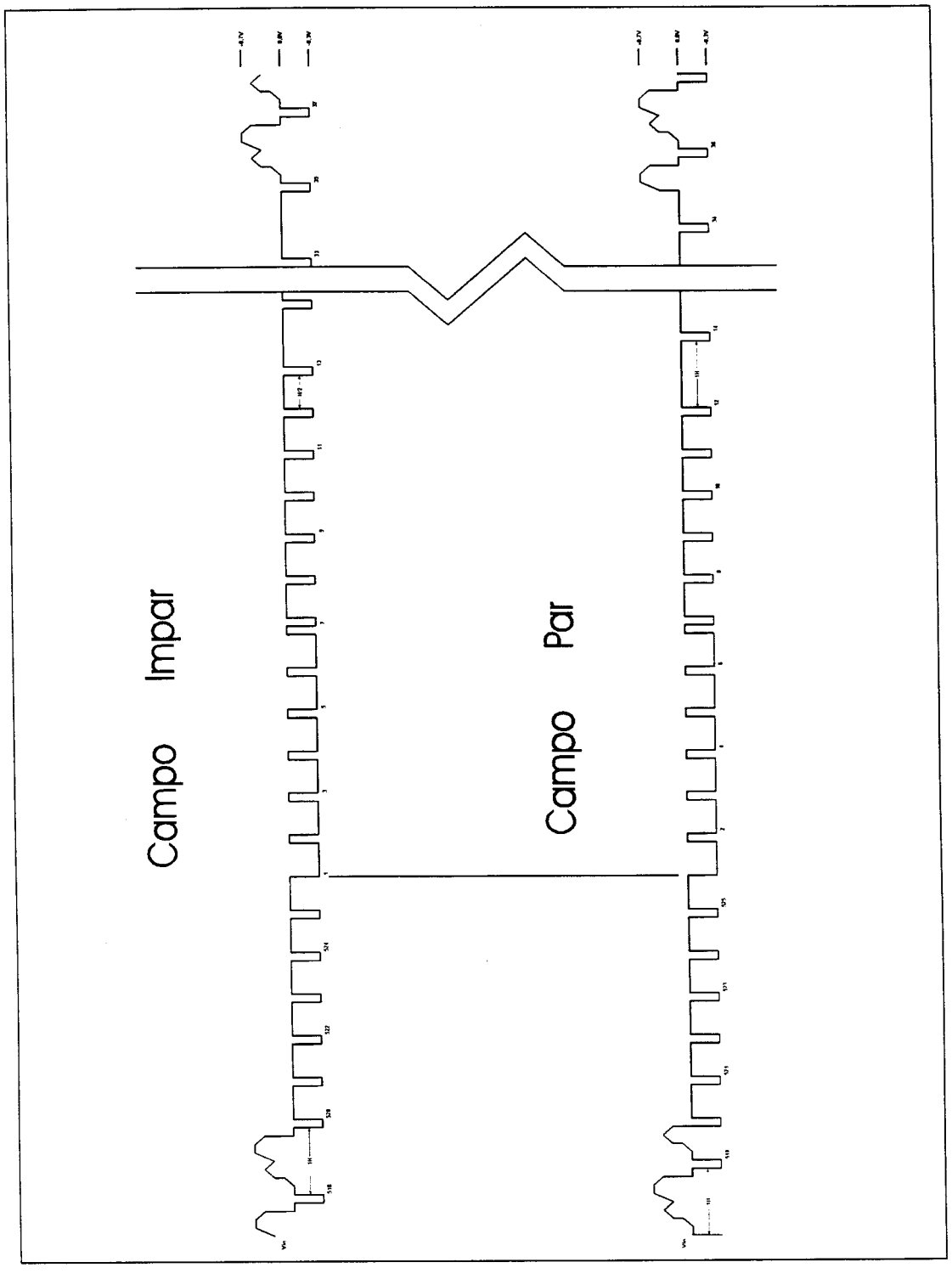

FIGURA 13 - Formato do sinal de sincronismo vertical

Este sinal possui uma amplitude de -0,3V e um período de 3H, distando de 4H do pulso de sincronismo horizontal da última linha do campo par (linha número 518) e de 3,5H do pulso de sincronismo horizontal da última linha do campo ímpar (linha número 519). Para que a malha integradora do monitor, que detecta este sinal, possa funcionar corretamente, sem perda de sincronismo horizontal, foram introduzidos pulsos extras nesse sinal, sendo que esses pulsos devem possuir as seguintes características:

1 - O intervalo entre o início do apagamento vertical e o início do pulso serrilhado deve conter 6 pulsos, correspondentes aos números 520 a 525, a fim de sincronizar o oscilador vertical do monitor.

No final do campo ímpar, o sincronismo vertical é feito com os pulsos 521, 523 e 525 e, ao final do campo par, com os pulsos 520, 522 e 524. Esses pulsos ficam afastados de ½H e sua largura deve ser a metade daquela exibida por um pulso de sincronismo horizontal, a fim de se manter a mesma tensão residual na rede integradora, no sistema de sincronismo vertical, do monitor.

Por esse motivo, tais pulsos recebem o nome de pulsos equalizadores, pois realmente equalizam as cargas residuais da rede integradora, para ambos os campos.

2 - O pulso serrilhado de sincronismo vertical deverá conter 6 descontinuidades, correspondentes aos números de 1 a 6, com freqüência igual à dos pulsos equalizadores e largura suficientemente pequena, a fim de reduzir o tempo de subida dos pulsos integrados, na saída da rede, minimizando o efeito de dente-de-serra no sinal integrado. Como o detetor de sincronismo horizontal utiliza a borda de

subida do sinal de sincronismo, a inversão desses pulsos, em comparação com os pulsos de sincronismo horizontal, não causa problemas.

3 - No intervalo de 3H, imediatamente após o fim do pulso serrilhado, deve existir mais seis pulsos equalizadores (de 7 a 12), idênticos aos anteriores.

Logo após os pulsos equalizadores de saída, reinicia-se o envio dos pulsos de sincronismo horizontal, cuja quantia pode variar entre 5 e 9, antes do término do apagamento vertical. Haverá, desse modo, uma distância de H entre o pulso 14 e o último equalizador de saída para o campo par, e de ½H entre o pulso 13 e o último equalizador de saída para o campo ímpar.

4.2.2 Digitalizador

Dentre os vários tipos de conversores analógico-digitais (CA/D), o conversor do tipo *flash* é o mais indicado para "digitalizações" de vídeo, por permitir altas taxas de conversão[11]. Por esse motivo, foi utilizado o circuito CA/D Bt218KP20, da BrookTree [29], um circuito conversor digital-analógico de 8*bits* do tipo *flash*, específico para aplicações em "digitalização" de sinais de vídeo, com uma taxa de conversão de até 20MSPS (veja FIGURA 14).

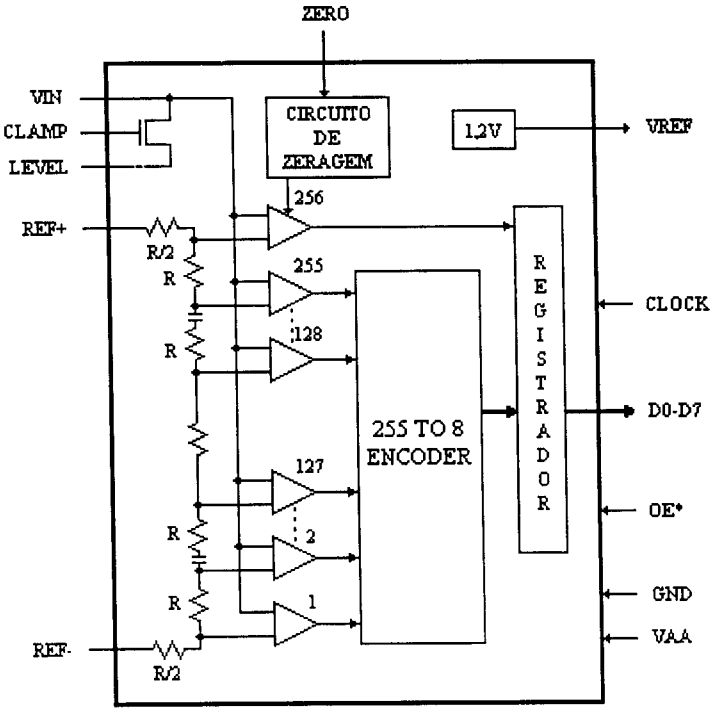

FIGURA 14 - Diagrama interno do conversor A/D Bt218KP20

Este conversor contém 256 comparadores de alta velocidade, um codificador de 255 para 8, um registrador de saída e uma ponte divisora de resistores. Dos 256 comparadores, 255 são utilizados para "digitalizar" o sinal analógico; um comparador adicional é utilizado para gerar o *bit* de TRANSBORDO (*OVERFLOW*). Possui ainda uma fonte interna de referência de 1,2V.

A saída dos dados é compatível com o tipo TTL e é síncrona com a borda de subida do pulso de CLOCK, sendo que o valor é amostrado na borda de descida do sinal de CLOCK e o sinal "digitalizado" fica disponível na segunda borda de subida, conforme mostra a FIGURA 15. O pino OE* permite colocar a saída em *tri-state*, assincronamente.

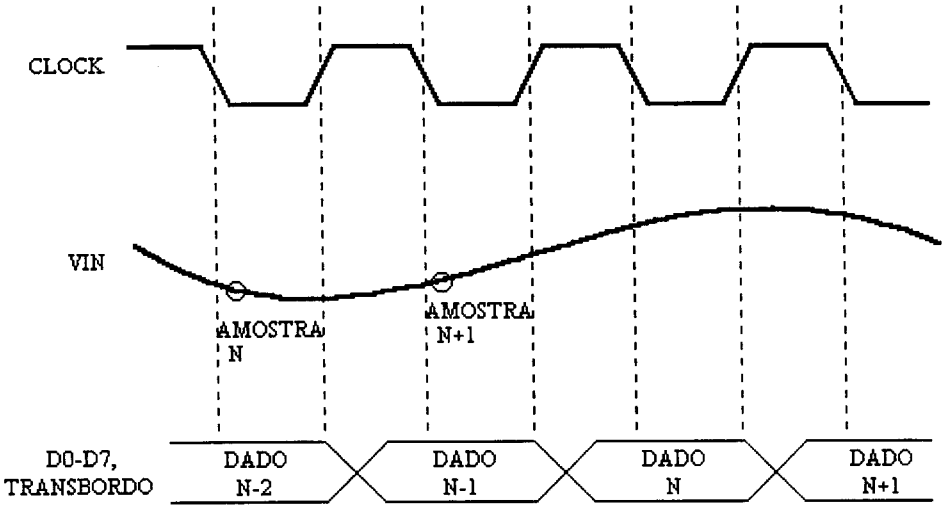

FIGURA 15 - Diagrama de tempo da aquisição de dados

Os valores dos pinos de referência REF+ e REF- são flexíveis, o que permite a "digitalização" de qualquer sinal de vídeo sem a necessidade de amplificadores de entrada.

O pino ZERO é utilizado periodicamente para "zerar" os comparadores. Os comparadores possuem um *offset* inicial devido às tolerâncias de fabricação. Capacitores são, então, utilizados para corrigir este erro inicial. Como os capacitores se descarregam, estes precisam ser recarregados periodicamente. Durante este período de recarga, a saída permanece com o último valor amostrado antes do pulso de ZERO.

As entradas CLAMP e LEVEL são utilizadas somente em aplicações onde o sinal de vídeo não possui um nível contínuo (DC). Quando o pino CLAMP está em nível "1", a entrada VIN é forçada para o nível de tensão presente em LEVEL,

restaurando o nível contínuo do sinal de vídeo. Se o sinal de vídeo já possuir um nível contínuo, o pino LEVEL deverá ser ligada à entrada VIN.

Na FIGURA 16 podemos ver o circuito utilizado para fazer a conversão analógico-digital do sinal de vídeo. A resistência de 75Ω é necessária para fazer o casamento de impedâncias, típica para sinais de vídeo, enquanto que a resistência de 50Ω prevê o isolamento na entrada VIN de qualquer ruído vindo dos pulsos de CLOCK. A tensão de referência foi obtida a partir do próprio circuito, sendo utilizado apenas uma resistência variável de 200Ω.

O sinal de CLOCK foi obtido a partir de um oscilador controlado por um filtro cerâmico comercial de 10,7MHz. Este valor, além de ser o valor utilizado em vários outros projetos [6][7][23], está de acordo com o valor mínimo necessário para a conversão de sinais de vídeo, uma vez que este possui uma largura de banda da ordem de 4,5MHz.

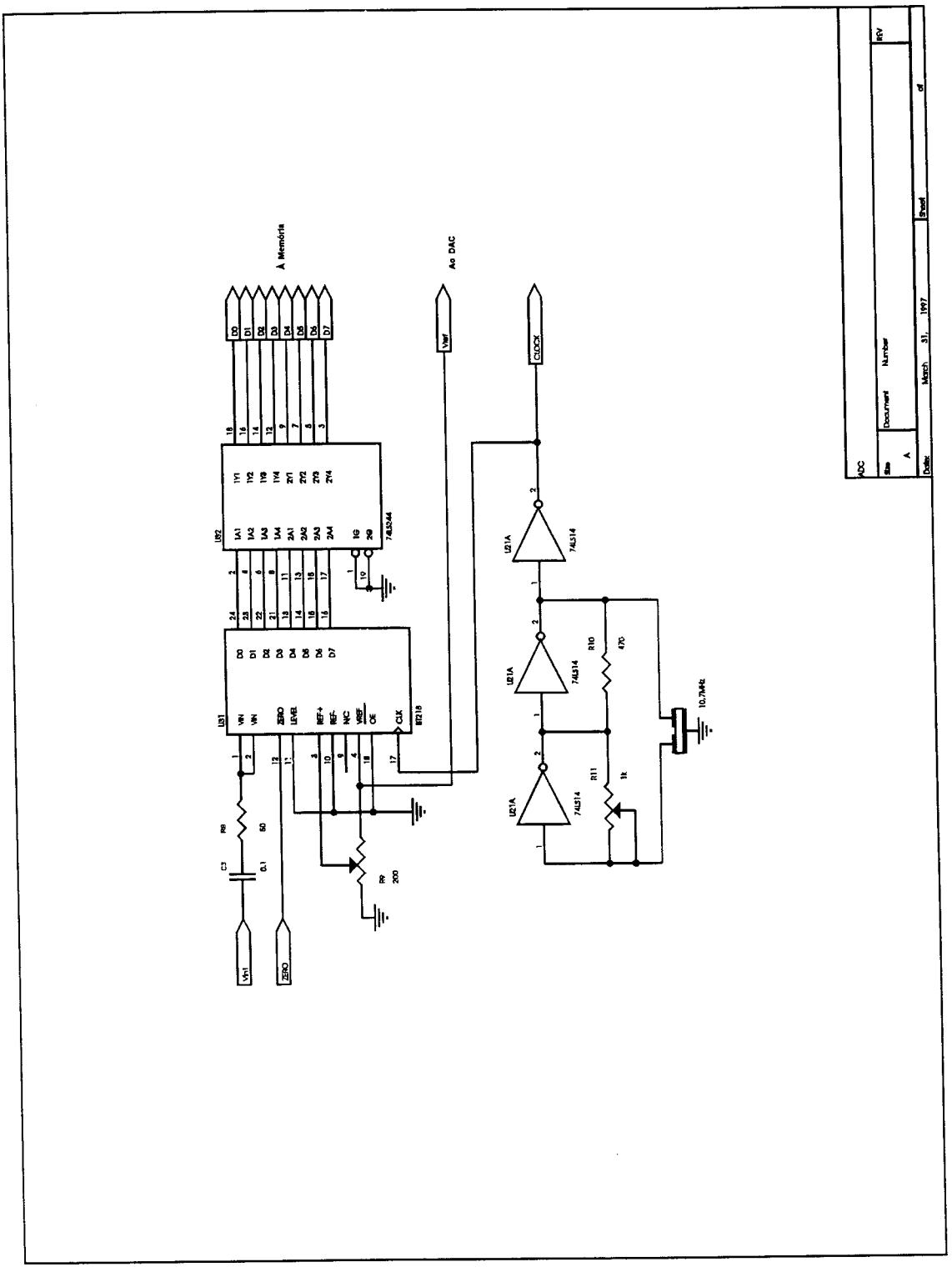

FIGURA 16 - Circuito do Conversor A/D

4.2.3 Armazenamento

Trabalhar com sinal de vídeo entrelaçado impõe a necessidade de armazenamento de cada um dos campos que compõem uma imagem. Enquanto que o sinal de vídeo não entrelaçado permite fazer a separação das linhas de uma imagem por meio de *delay lines*, baseados em registradores de deslocamento (FIGURA 17), o sinal de vídeo entrelaçado obriga o armazenamento de um campo inteiro, para que possamos ter o mesmo efeito

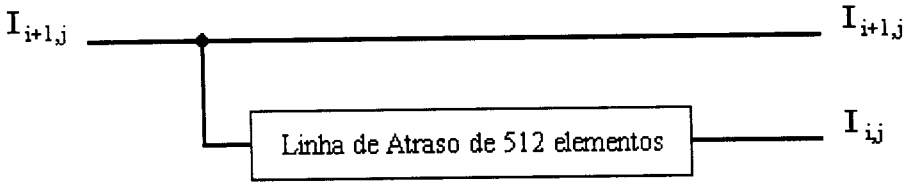

FIGURA 17 - Atrasos no sinal de vídeo não entrelaçado

Esta separação de linhas é necessária para fornecer ao processador de imagem uma pequena janela de 2x2 da imagem para que este possa fazer o processamento. No caso de se trabalhar com sinal de vídeo não entrelaçado, precisaríamos apenas de um *delay line* de 512 elementos. Como o trabalho está baseado em um sinal de vídeo entrelaçado, o *delay line* deve ser de um campo inteiro. Isto pode ser conseguido por meio de duas memórias do tipo FIFO (*First Input-First Output)*. Foram utilizadas duas memórias MSM518221 [30], com capacidade de 256k*bytes* (2M*bits*) cada e tempo de acesso de 30ns.

As principais características desta memória são:

1 - Os controles de escrita e leitura são independentes, o que permite que estas duas operações sejam feitas assincronamente e em taxas diferentes.

2 - Apesar de ser uma memória FIFO dinâmica, esta possui um circuito interno de *self-refresh*, o que torna desnecessária a utilização de circuito externos, fazendo com que pareça ser totalmente estática.

3 - A existência dos pinos OE (*Output Enable*) e IE (*Input Enable*) permite que os apontadores internos sejam modificados sem que sejam feitas escritas ou leituras, propriamente ditas, ou que estas operações só sejam realizadas em determinadas posições da memória.

Devido à sua configuração interna, os dados só podem ser lidos após, pelo menos, 71 pulsos de escrita. Como esta memória é utilizada para armazenar um campo inteiro, antes deste ser lido, isto não se torna um problema.

Na FIGURA 18 podemos ver o circuito utilizado, assim como a sua pinagem.

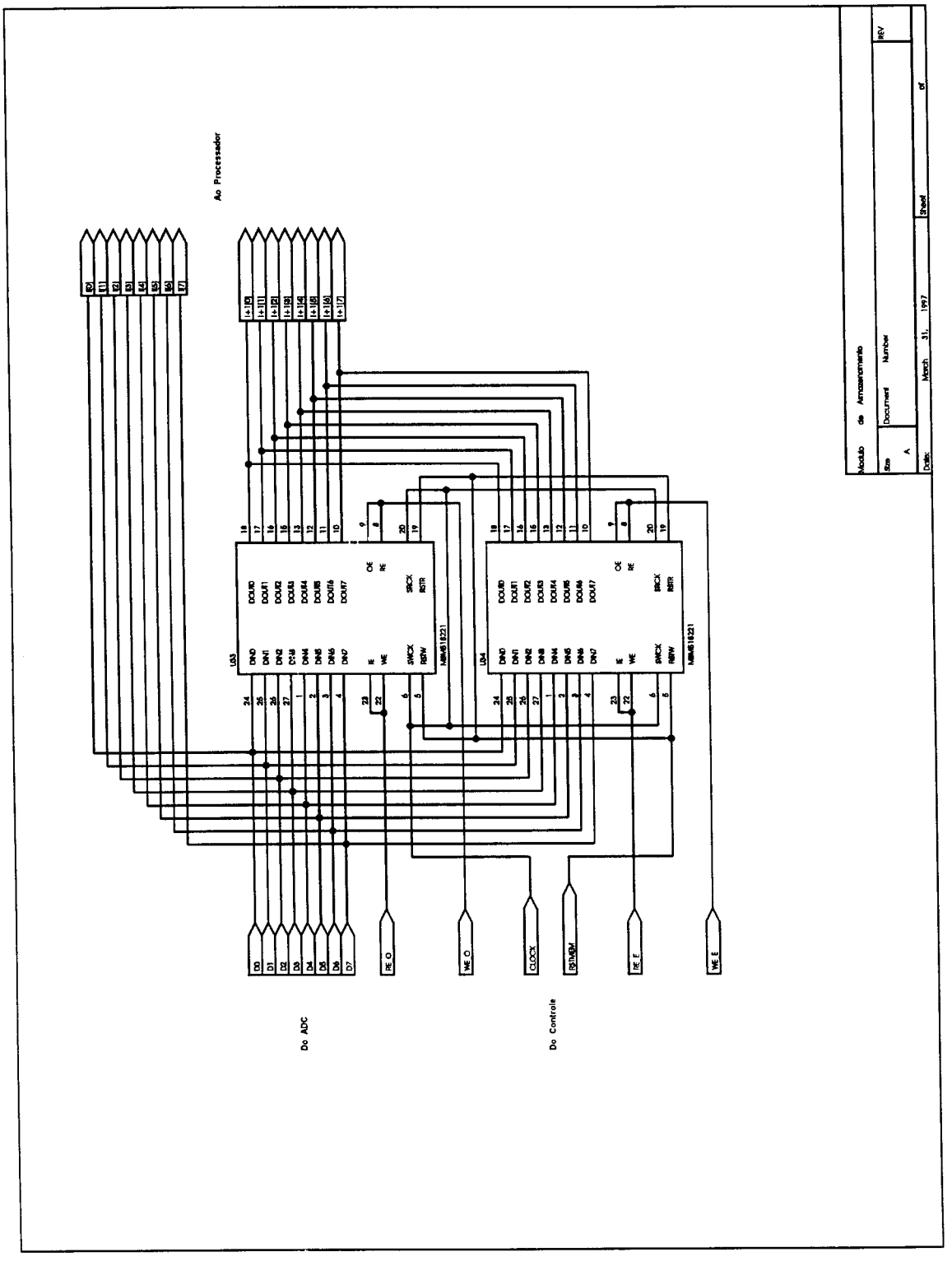

FIGURA 18 - Circuito do armazenamento de dados

A função dos pinos é descrita abaixo:

Pinos de Escrita:

WE - *Write Enable* - Realiza a escrita

IE - *Input Enable* - Habilita a escrita

SWCK - *Serial Write Clock* - Pulso de escrita

RSTW - *Write Reset Clock* - Zera o ponteiro de escrita

D_{in}0-7 - Dados de entrada

Pinos de Saída:

RE - *Read Enable* - Realiza a leitura

OE - *Output Enable* - Habilita a leitura

SRCK - *Serial Read Clock* - Pulso de leitura

RSTR - *Read Reset Clock* - Zera o ponteiro de leitura

D_{out}0-7 - Dados de saída

Quando o pino de entrada WE estiver em estado alto, o ponteiro de escrita será incrementado a cada pulso no pino SWCK, e a escrita propriamente dita, só ocorrerá se o pino IE também estiver em estado alto. Se este estiver em estado baixo, o ponteiro de escrita continuará a ser incrementado a cada pulso em SWCK, mas nada será escrito. Quando o pino WE estiver em estado baixo, o ponteiro de escrita ficará parado. A escrita ocorrerá sempre na borda de subida do pulso em SWCK. O pino RSTW "zerará" o ponteiro de escrita, sempre que este pino estiver em estado alto.

Quando o pino de saída RE estiver em estado alto, o ponteiro de saída será incrementado a cada pulso no pino SRCK, e a leitura propriamente dita, só ocorrerá se o pino OE também estiver em estado alto. Se este estiver em estado baixo, o ponteiro de leitura continuará a ser incrementado a cada pulso em SRCK, mas nada será lido. Quando o pino RE estiver em estado baixo, o ponteiro de leitura ficará parado. A leitura ocorrerá sempre na borda de subida do pulso em SRCK. O pino RSTR zerará o ponteiro de leitura, sempre que este pino estiver em estado alto.

4.2.4 Controle

Para realizar o controle de leitura e escrita das memórias, assim como o controle do conversor analógico digital, foi desenvolvido um circuito lógico especial. A primeira parte deste circuito consiste de um separador de sincronismo de vídeo, o LM1881 [31], da National Semicondutors, que extrai os sinais de sincronismo vertical e horizontal, de *burst*, e de tipo de campo a partir de um sinal de vídeo composto padrão, e a segunda parte é composta de um conjunto de portas lógicas e contadores que utilizam os sinais fornecidos pelo separador de sincronismo para gerar os sinais de controle das memórias e do conversor analógico-digital.

Na FIGURA 19 podemos ver a ligação do circuito separador de sincronismo e os sinais gerados a partir de um sinal de vídeo, que é fornecido ao pino 2 do circuito por meio de um capacitor de 100nF. No pino 1 deste circuito temos a saída do sinal de sincronismo horizontal /COM (ativo baixo), cuja largura é de 4,7μs durante os pulsos de sincronismo horizontal e de 2,4μs durante os pulsos

equalizadores e serrilhado do sincronismo vertical. Ao final do pulso de sincronismo horizontal, temos o pulso de *burst* /BURST, também ativo baixo, que está presente no pino 5 e tem uma largura de aproximadamente 4μs. Este sinal é gerado a cada transição positiva do sinal de sincronismo horizontal.

No primeiro pulso serrilhado do sinal de sincronismo vertical, são gerados os pulsos de sincronismo vertical (/VS) e de campo (O_E), presentes nos pinos 3 e 7 do circuito, respectivamente. O sinal de sincronismo vertical gerado é ativo baixo e possui uma largura típica de 230μs, enquanto que o sinal de campo gerado será alto quando o campo for ímpar e baixo quando o campo for par.

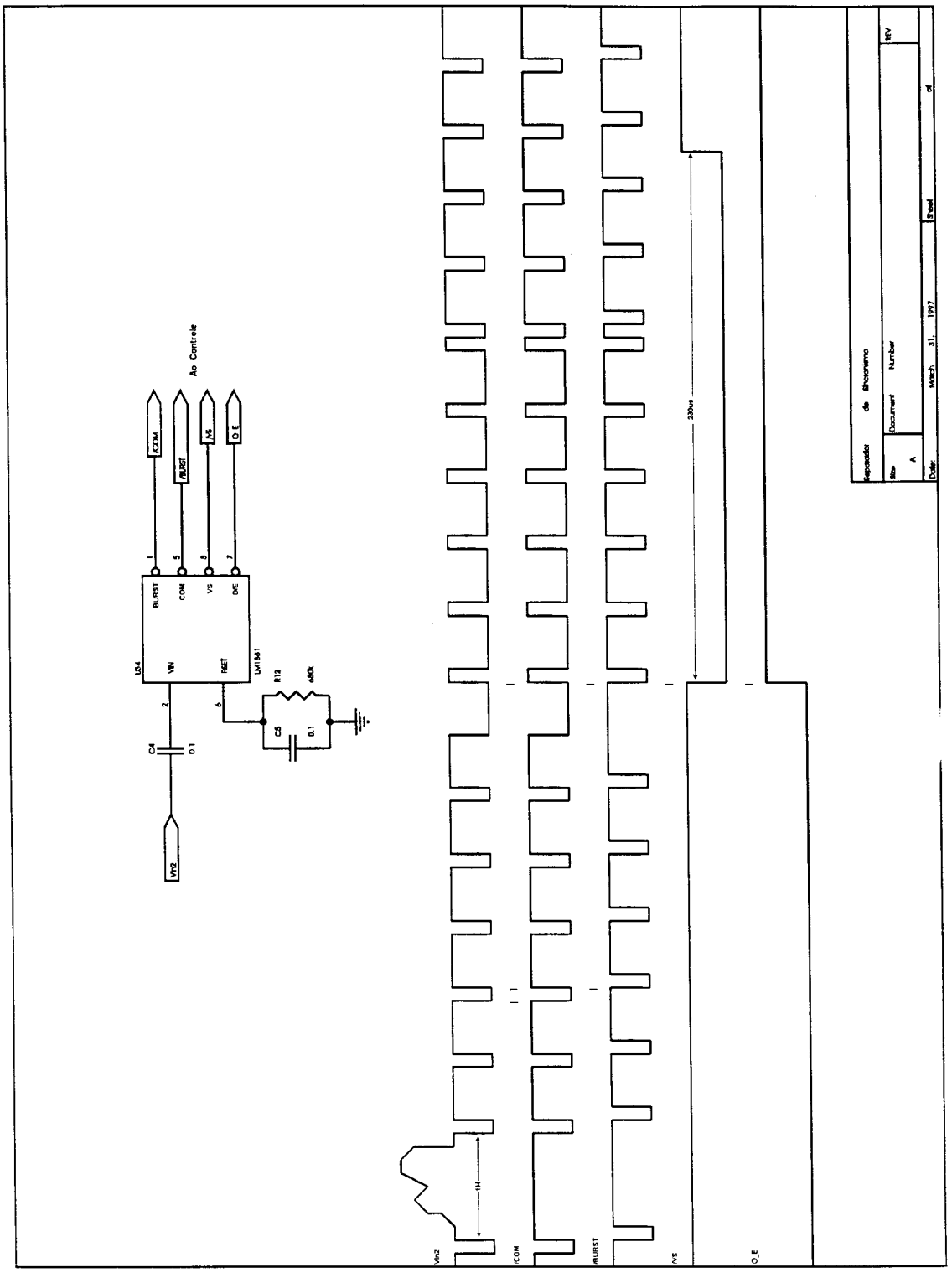

FIGURA 19 - Circuito separador de sincronismo e formas de onda

A segunda parte do circuito de controle pode ser vista na FIGURA 20. Este circuito tem as seguintes funções:

- Gerar o sinal de ZERO para a "zeragem" dos comparadores do conversor
- Excluir os pulsos de serrilhado e equalizadores
- Gerar o sinal de inicialização dos ponteiros de leitura e escrita das memórias
- Gerar os sinais de leitura e escrita para as memórias
- Limitar em 512 a quantidade de conversões por linha

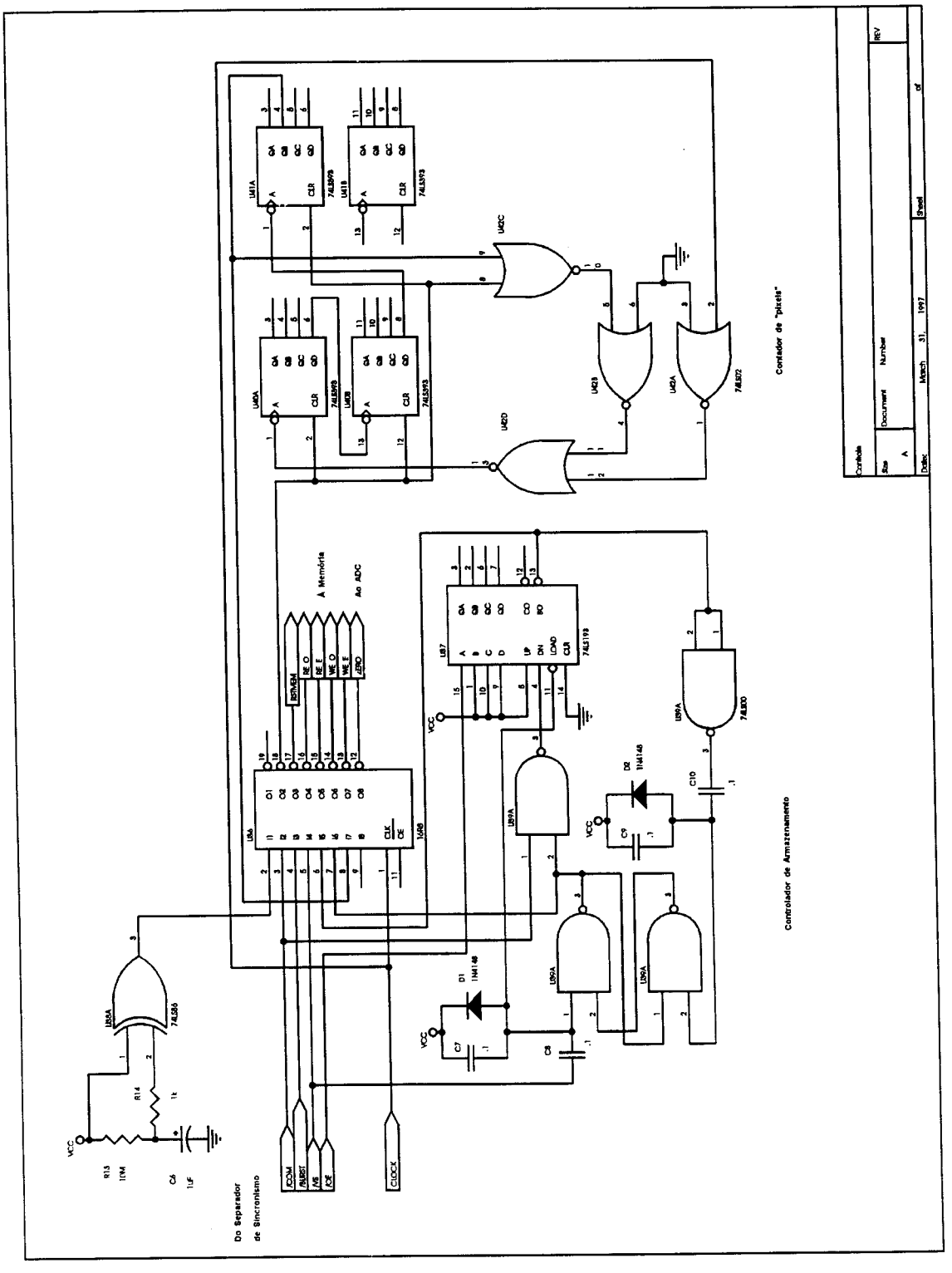

FIGURA 20 - Lógica de controle

As portas lógicas NÃO-E (74LS00), juntamente com o contador (74LS193), fazem a exclusão dos pulsos serrilhado e equalizadores. O pulso /VS ativa o flip-flop, formado pelas portas NÃO-E A e B, e habilita o contador a fazer a leitura dos dados presentes na sua entrada paralela. Enquanto o flip-flop estiver ativado, este habilitará a contagem decrescente dos valores lidos pelo contador, por meio da porta NÃO-E C e do sinal /COM. Quando a contagem chegar a zero, o pino /BO do contador será ativado e zerará o flip-flop, terminando a exclusão dos pulsos.

Para que haja um perfeito sincronismo entre as linhas do campo par e do campo ímpar, é necessária a exclusão de mais uma linha do campo ímpar, o que é feito ligando-se o *bit* menos significativo da entrada paralela do contador ao sinal O_E.

O período ativo do flip-flop é denominado SYNCNEW e o pulso fornecido pelo pino /BO do contador é denominado /H, e estes dois sinais são enviados para a PAL 16R8 que irá gerar os sinais para o controle das memória e do conversor.

Os contadores (74LS393), juntamente com as portas NÃO-OU (74LS02), habilitam a conversão de 512 pontos por linha. Quando começa uma nova linha, a PAL 16R8 envia o sinal /COUNT_E para a porta NÃO-OU C, que por meio das portas A, B e D habilitará a contagem dos contadores. Quando a contagem chegar em 512, o contador desabilitará a contagem e enviará o sinal de término de contagem Q12 para a PAL 16R8. O sinal /COUNT_E, durante o sinal de sincronismo horizontal, é utilizado para zerar os contadores.

As equações da PAL 16R8 são:

```
TITLE        Controle da memória FIFO
PATTERN      PALFIFOR.PDS
REVISION     4.10
AUTHOR       Maximiliam Luppe
COMPANY      LIE/FFI/IFSC/USP
DATE         10/10/96

CHIP         PALFIFOR     PAL16R8

;PINS   1     2     3      4      5    6      7         8    9    10
        CLK   POR   COM    BURST  O_E  H   SYNCNEW   Q12   NC   GND

;PINS  11     12     13     14     15     16      17          18      19  20
        NC    ZERO   WE_E   WE_O   RE_E   RE_O   RSTMEM   COUNT_E   NC  VCC

EQUATIONS

        WE_O :=        O_E * /SYNCNEW * COM * /POR * /Q12
        RE_E :=        O_E * /SYNCNEW * COM * /POR * /Q12
        WE_E :=       /O_E * /SYNCNEW * COM * /POR * /Q12
        RE_O :=       /O_E * /SYNCNEW * COM * /POR * /Q12
                    + /O_E *  SYNCNEW * COM * /POR * /Q12 * /H
        COUNT_E :=  /(        /SYNCNEW * COM * /POR
                    + /O_E *  SYNCNEW * COM * /POR * /H)
        RSTMEM :=     /BURST * /H
        ZERO :=       /COM
                    +  POR
```

O sinal POR (*Power-On Reset*) é gerado pela porta OU EXCUSIVO (74LS86) e tem a função de proteger os circuito durante a estabilização da tensão de alimentação.

4.3 PROCESSADOR

O processador proposto foi implementado em uma arquitetura tipo *pipeline*, constituída basicamente de 4 estágios:

1 estágio de distribuição dos dados internamente

2 estágios para a subtração em módulo

1 estágio para a determinação do resultado final

4.3.1 Primeiro Estágio: Distribuição

O primeiro estágio (FIGURA 21) consiste de quatro registradores (*latches*) que realizam o atraso necessário para que os elementos diagonais estejam adequadamente disponíveis para o segundo estágio. Neste estágio todos os registradores estarão sincronizados e teremos um atraso total de 2 ciclos de clock.

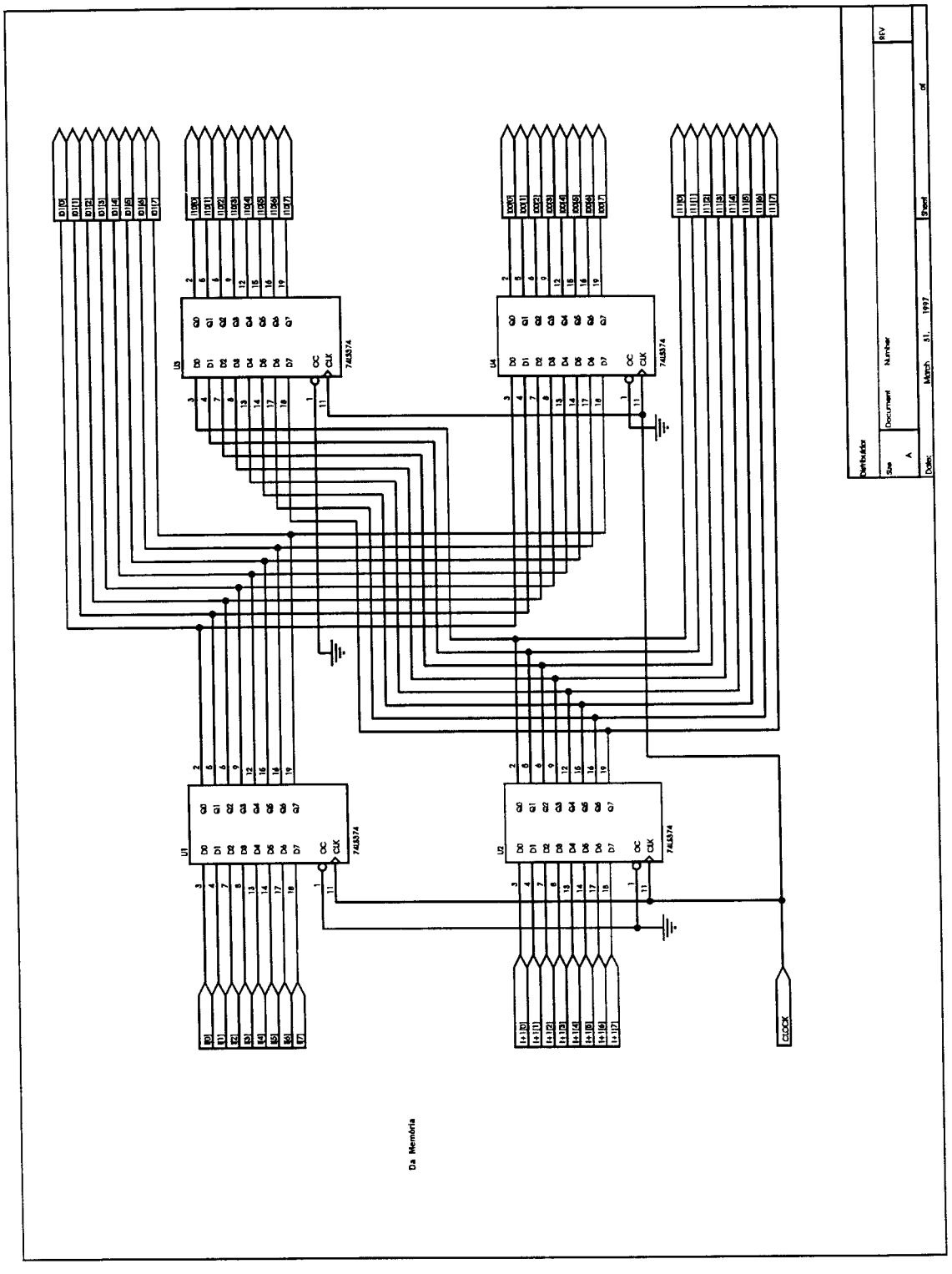

FIGURA 21 - Primeiro estágio do processador

4.3.2 Segundo Estágio: Máscaras

O segundo estágio (FIGURA 22) realiza a subtração dos elementos diagonais. Como não está implementado o cálculo da direção do gradiente, não houve a necessidade de se preocupar com o sinal do resultado da aplicação das máscaras W1 e W2, sobre a imagem. Desta forma, a subtração foi implementada através de um somador, onde a menor parcela da soma está em complemento de um. Para que isto fosse possível, este estágio foi dividido em duas partes: a primeira parte realiza o complemento de um da menor parcela, e a segunda parte realiza a soma das duas parcelas, mais 1 (*carry*). A primeira parte foi implementada com um comparador de 8*bits* (74LS684*), uma porta NÃO (1/6 74LS14) e com 16 portas OU-EXCLUSIVO de duas entradas (4 74LS86). As portas OU-EXCLUSIVO podem ser utilizadas para funcionarem como inversores controlados, bastando para isso utilizar uma de suas entradas para o controle e a outra para o dado. Se a porta de controle estiver em nível "0", o dado não será invertido, e se estiver em nível "1", este será invertido. O sinal de controle vem do pino /P>Q do comparador e vai direto para um conjunto de 8 portas OU-EXCLUSIVO e invertido para o outro conjunto de 8 portas OU-EXCLUSIVO. Desta forma teremos sempre o complemento de um da menor parcela.

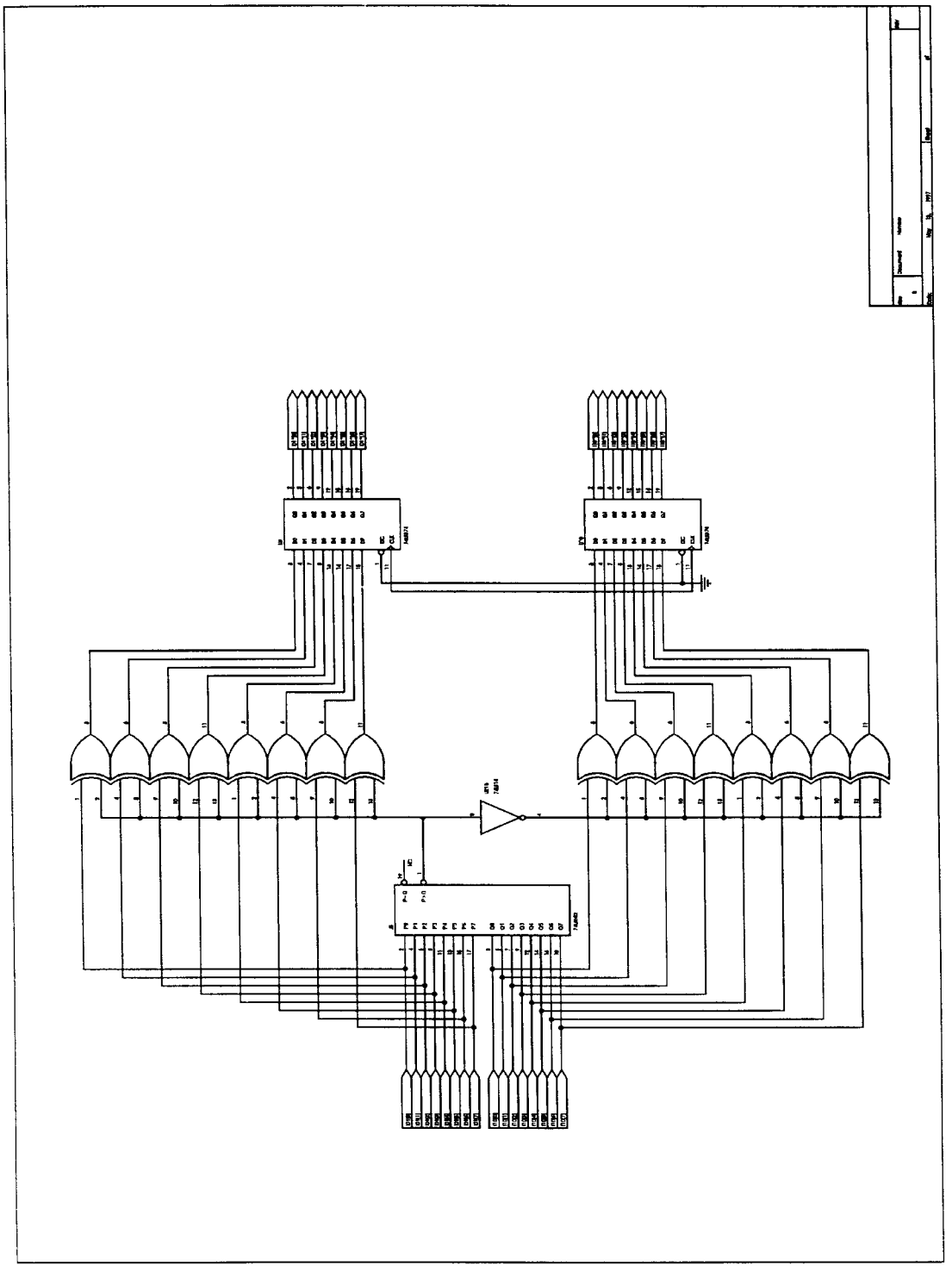

FIGURA 22 - Primeira parte do circuito para o cálculo de W

A segunda parte foi implementada com dois somadores de 4*bits* (74LS283), que realizam a soma propriamente dita. Para que o resultado esteja correto, é necessário somar mais 1, para que realizemos a subtração em complemento de dois. Esta parte do segundo estágio é a mais crítica devido ao tempo de atraso dos somadores, que são do tipo cascata, e tendem a ter um atraso da ordem de 41ns. Por este motivo, este estágio teve que ser dividido em duas partes, com um registrador entre elas, mais um registrador na saída, o que causa um atraso total de 2 ciclos de clock.

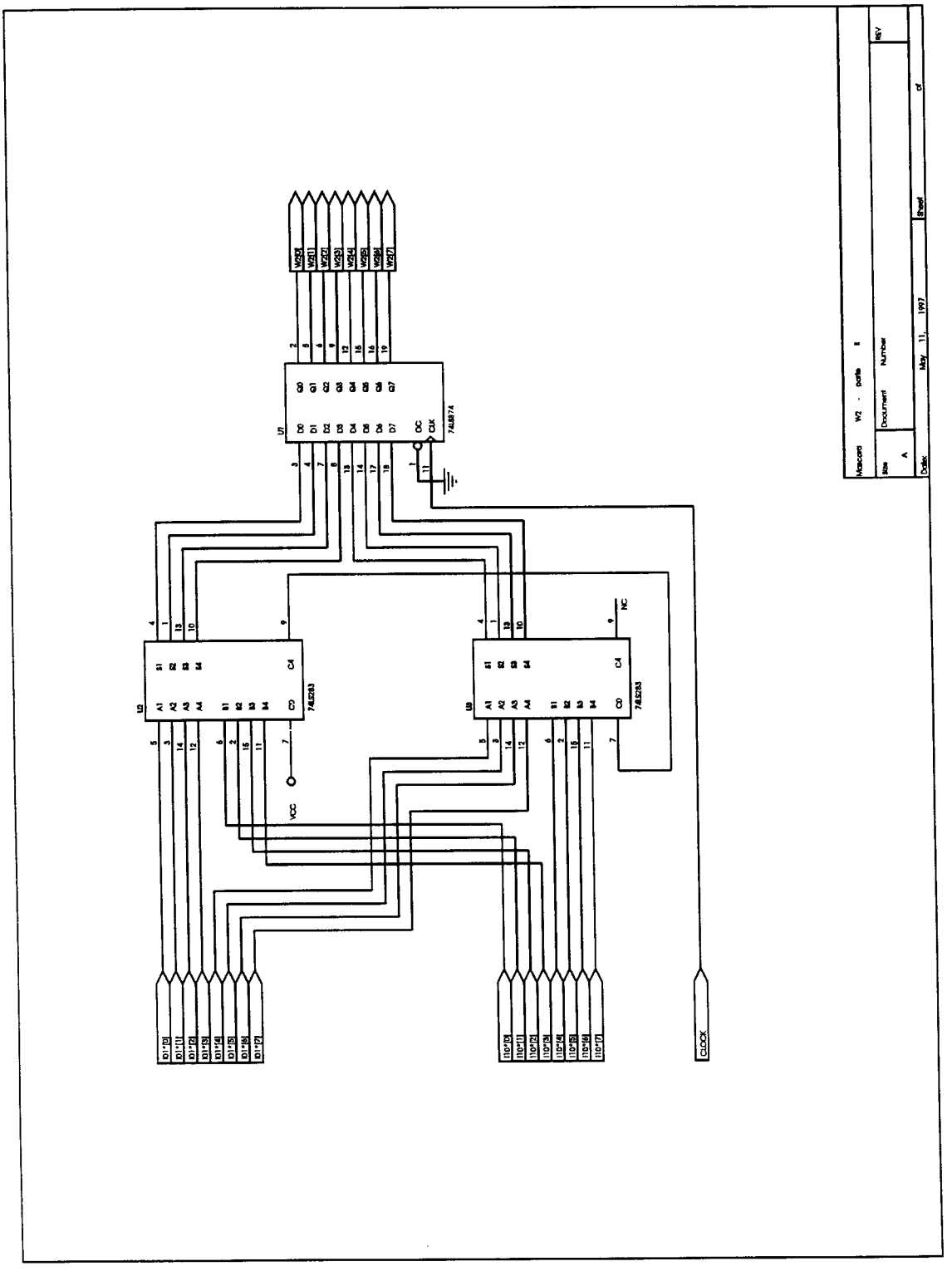

FIGURA 23 - Segunda parte do circuito para o cálculo de W

4.3.3 Terceiro Estágio: Operador não Linear

O terceiro e último estágio do processador realiza o cálculo do operador não linear Lambda(i,j). Dentre as três possíveis implementações [Eqs. (6), (7) e (8)], somente a do valor médio [Eq. (6)] não pôde ser implementada, por exigir circuitos complexos para cálculos de raiz quadrada e quadrado de um número, além de uma divisão por $\sqrt{2}$, necessária para a normalização. É claro que este cálculo do valor médio poderia ser facilmente implementado por meio de uma tabela, do tipo *look-up* (FIGURA 24), utilizando-se, por exemplo, uma memória do tipo EPROM, mas isso não foi feito devido ao fato de que não foram encontradas memórias EPROMs com tempos de acesso menores do que 120ns e do fato de que a utilização de uma tabela tornaria a implementação desta arquitetura em uma FPGA impossível.

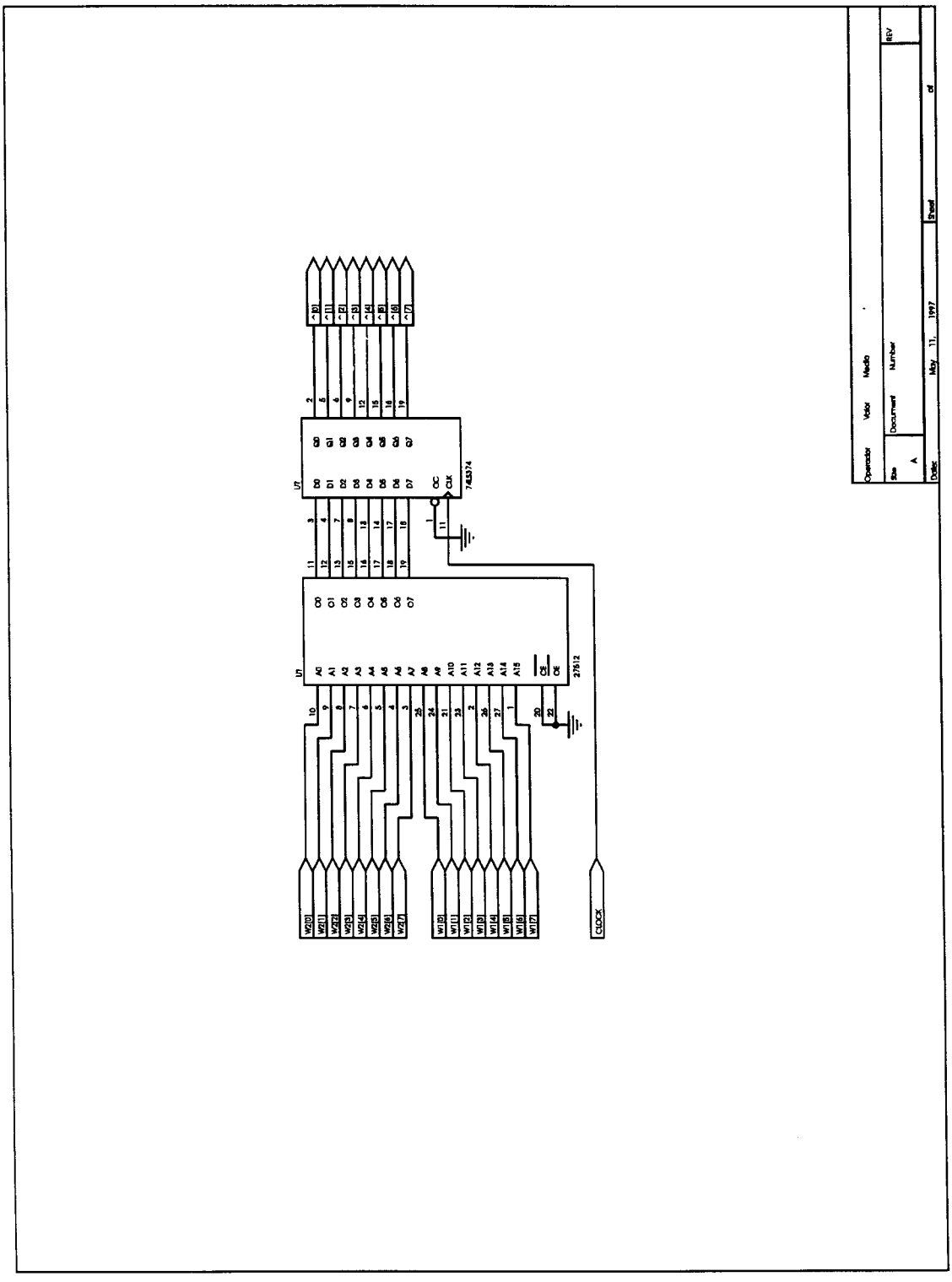

FIGURA 24 - Circuito para o cálculo do operador não linear $\sqrt{x^2 + y^2}$

A implementação do operador não linear $max(x,y)$ (FIGURA 25) foi feita por meio de um comparador de 8*bits* (74LS684*), e dois seletores quádruplos de 2 entradas e uma saída (74LS157). O comparador realiza a comparação do resultado de cada um dos ramos do processador, ou seja, do resultado da aplicação das máscaras W1 e W2 sobre a imagem, e, por meio do pino /P>Q indica aos seletores qual o maior resultado.

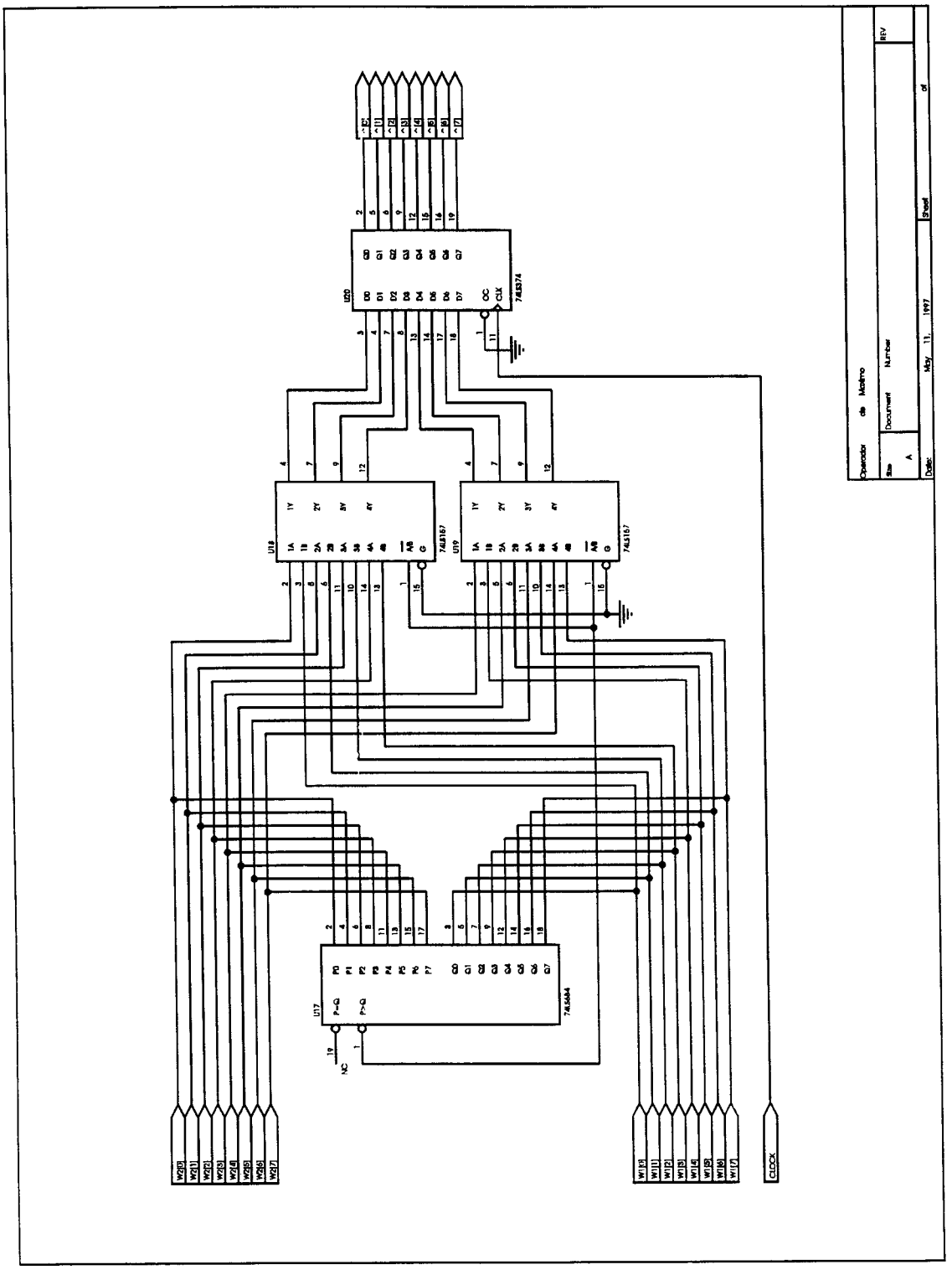

FIGURA 25 - Circuito para o cálculo do operador não linear $max(x,y)$

A implementação do operador de soma (FIGURA 26) pode ser feita utilizando apenas dois somadores de 4*bits* (74LS283), e a divisão por 2, para a normalização, pode ser realizada por meio de um deslocamento dos *bits* do resultado da soma, chamado *wired shift*, onde o *bit* mais significativo do resultado final é o pino de transbordo (*carry*) do segundo somador, ao invés do pino S4, e o *bit* menos significativo é o pino S2 do primeiro somador (segundo *bit* da soma), ao invés do pino S1 (FIGURA 26). Em ambas as implementações deste terceiro estágio, temos no final um registrador, que estabiliza o dado final do processador proposto, resultando em um atraso de 1 ciclo de clock.

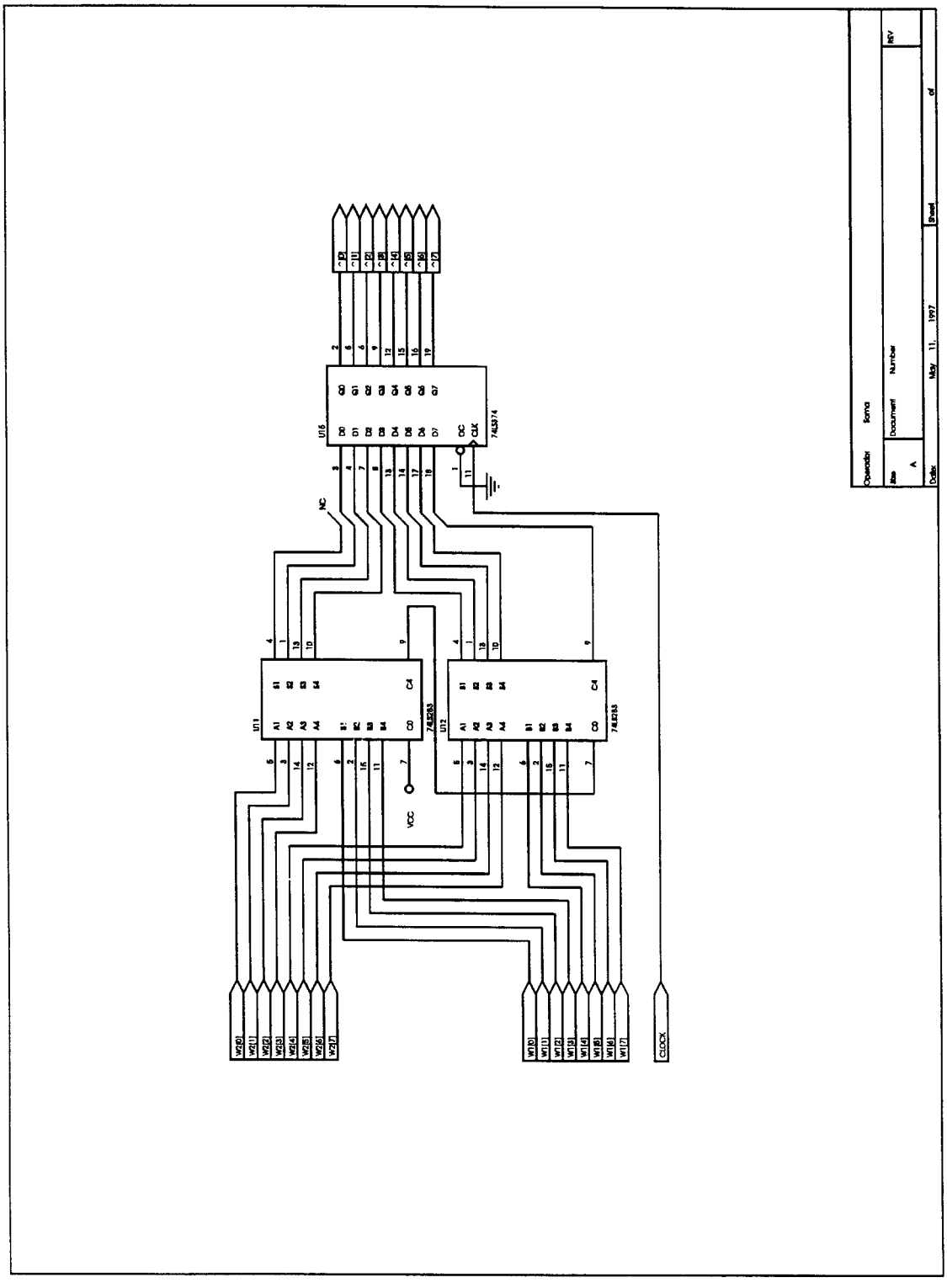

FIGURA 26 - Circuito para o cálculo do operador não linear $x+y$

O atraso total do processador implementado é de 5 ciclos de clock, ou seja, o cálculo para a detecção de bordas de uma dada imagem é de aproximadamente 467ns.

4.4 COMUNICAÇÃO E APRESENTAÇÃO DOS RESULTADOS

Como foi mencionado anteriormente, a comunicação trata apenas de como os dados são transferidos para o sistema implementado e como é feita a saída deste. Para a entrada dos dados, ou seja, do sinal de vídeo, é utilizado um amplificador operacional específico para vídeo , o MAX457 [32], da Maxim, e que é montado em uma configuração de seguidor não inversor [33] (amplificador de ganho igual a 1 - FIGURA 27). Apesar de o "digitalizador" não necessitar deste amplificador, ele foi utilizado como um distribuidor, ou seja, para separar o sinal de vídeo para o "digitalizador" e para o circuito de controle, mantendo a impedância de entrada igual a 75Ω. A não utilização dessa configuração poderia acarretar num desbalanceamento de carga na entrada, ou seja, teríamos um cabo coaxial de 75Ω ligado diretamente a duas entradas de 75Ω.

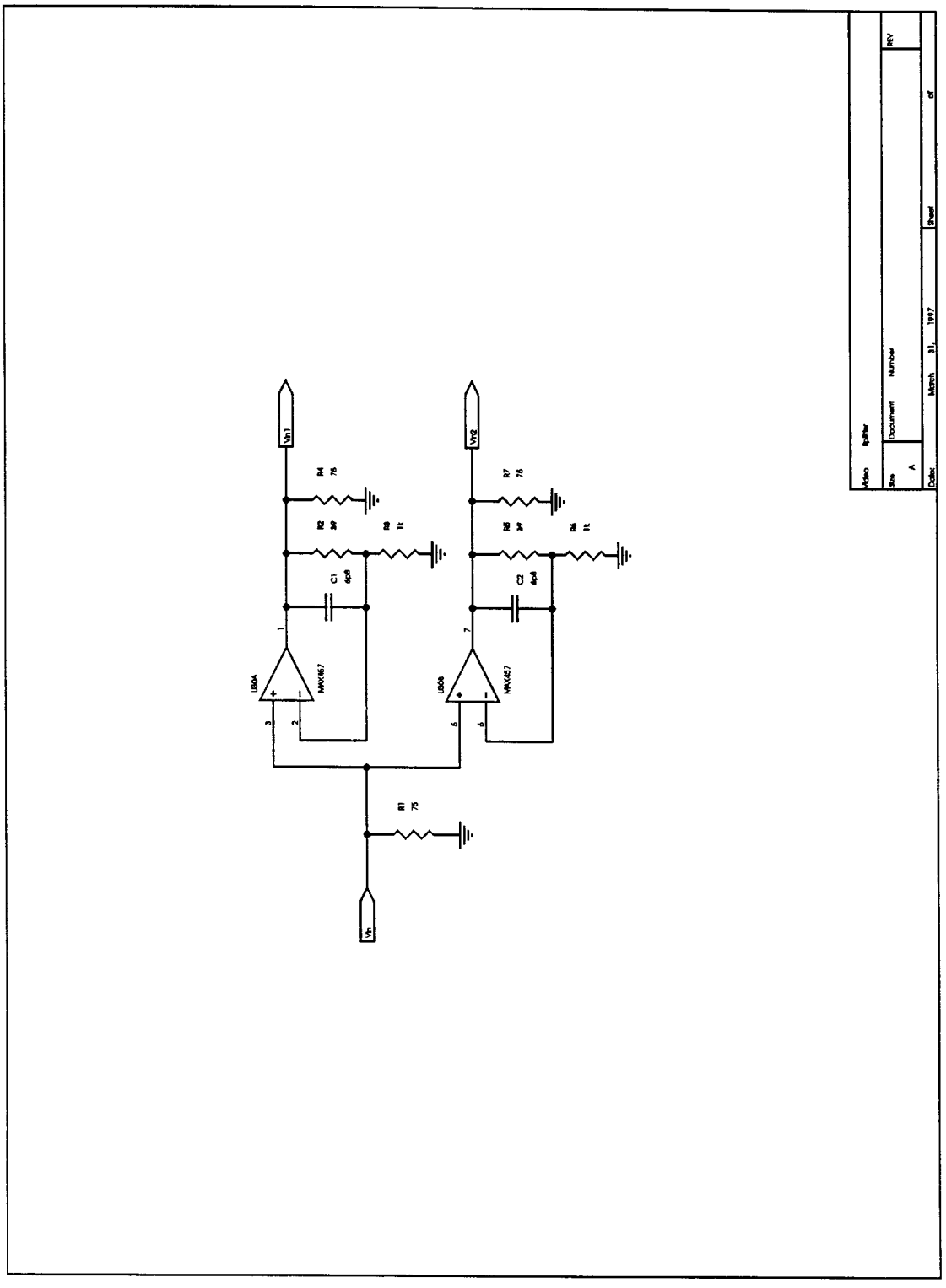

FIGURA 27 - Amplificador de entrada

Para a saída dos dados, foi optado manter o formato em sinal de vídeo. Isso permite a rápida visualização dos resultados em um monitor com entrada para vídeo composto, além de ser possível a utilização deste sinal por um computador, por meio de uma placa "digitalizadora" de vídeo tipo *Video Blaster* ou *Video Spigot*.

A conversão do sinal digital fornecido na saída do processador implementado pode, antes de ser apresentado, passar por um circuito detector de limiar, conforme mostra a FIGURA 28.

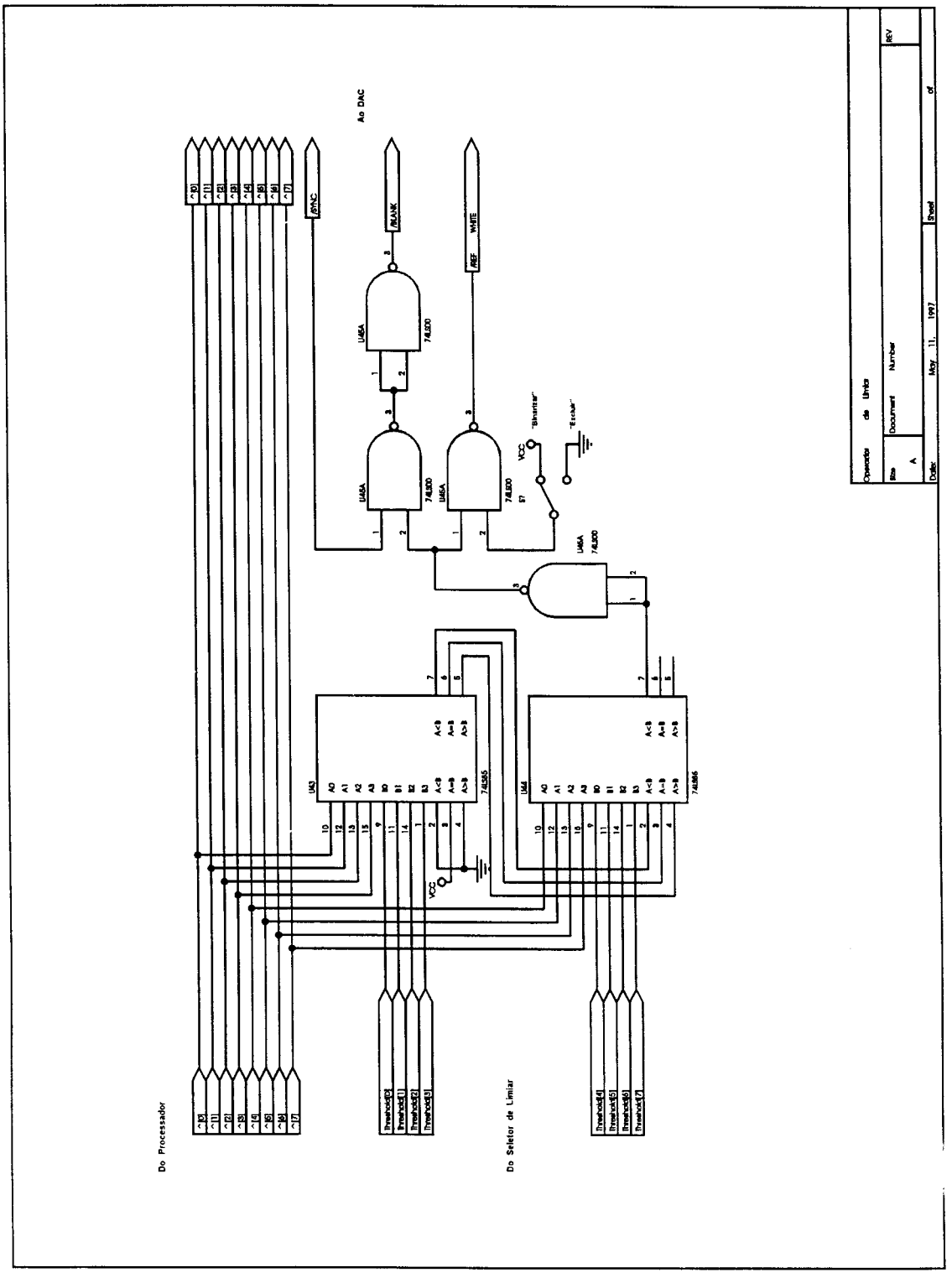

FIGURA 28 - Circuito para realizar um *threshold* final na imagem

Nesse circuito, temos uma entrada para os dados vindos do processador, que serão comparados com um valor de limiar, previamente fornecido por um seletor de limiar. Em função desse valor, podemos eliminar algumas bordas falsas, fazendo igual a zero as bordas que tiverem um gradiente menor que o valor de limiar, ou até mesmo "binarizar" a imagem final.

Para a conversão do sinal digital em um sinal de vídeo, foi utilizado um converso digital-analógico (CD/A) Bt102 [29], da BrookTree. A opção por este conversor se deve ao fato da alta taxa de conversão necessária.

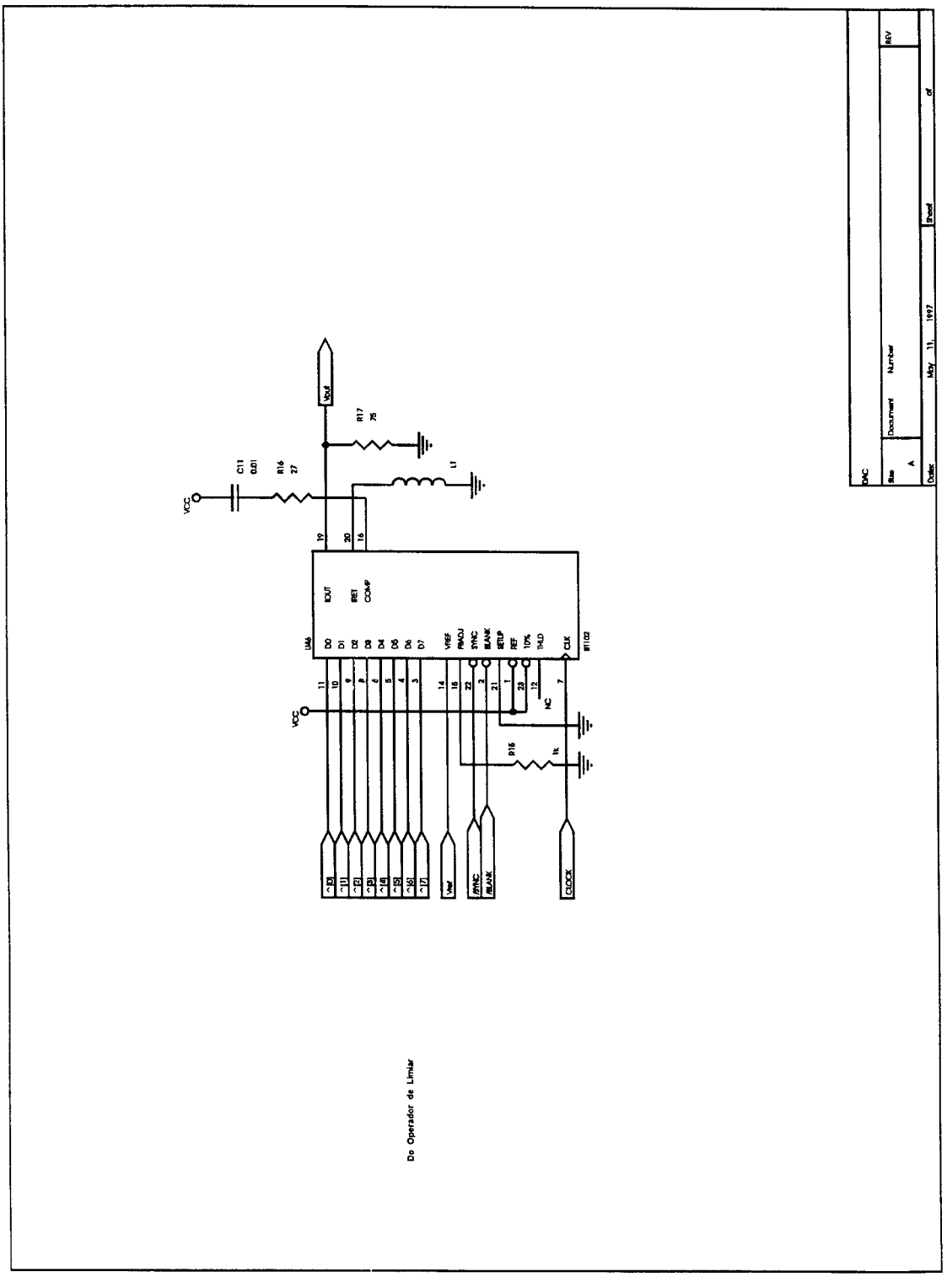

FIGURA 29 - Conversor digital-analógico

Para que o sinal permanecesse no formato de sinal de vídeo composto, foi necessária a introdução do sinal de sincronismo por meio da soma do sinal de sincronismo do sinal de vídeo de entrada com o sinal obtido na conversão digital-analógico. Como o tempo de atraso total é de 7 clocks entre o sinal de entrada e o de saída, ou seja, aproximadamente 0,7μ, pudemos fazer essa soma, sem que causasse problemas na visualização do resultado final. A opção dessa implementação foi devido ao fato de que a utilização de circuitos que gerassem um sinal de sincronismo aumentaria o custo final do sistema, além de torná-lo mais complexo.

4.5 RESULTADOS

Na FIGURA 30 abaixo podemos ver a foto do protótipo montado.

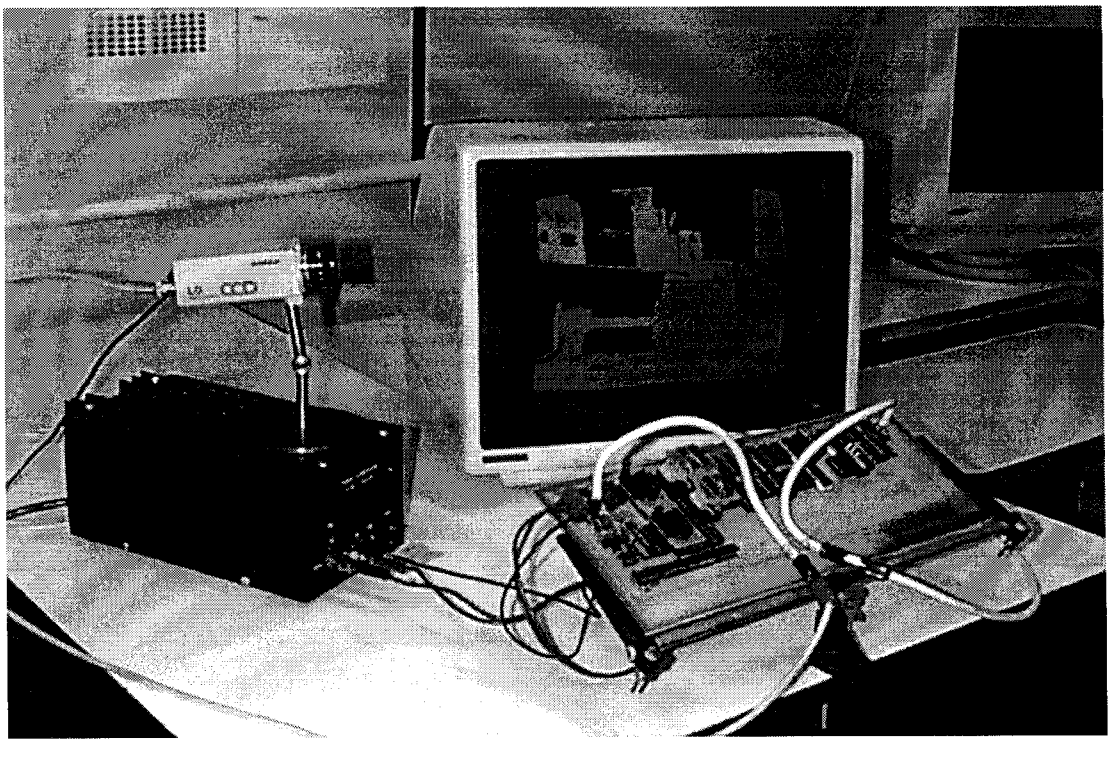

FIGURA 30 - Foto do protótipo do sistema montado. Nesta foto podemos ver o protótipo montado embaixo à direita, juntamente com a câmera de vídeo à esquerda e o monitor.

Nas FIGURAS seguintes temos algumas imagens processadas pelo protótipo. A primeira são formas geométricas comuns. Os valores entre parênteses indicam os valores de limiar escolhidos para cada imagem, tendo como único critério retirar ruídos existentes nas imagens processadas. As outras imagens são de um alicate, de um conjunto de pequenas figuras e uma visão do laboratório.

As barras à direita das imagens processadas são devido ao fato de se ter limitado em 512 o número de pixels por linha. Para efeito de comparação, são também apresentadas as mesmas imagens originais processadas por um editor gráfico, o PhotoPaint, que possui a opção de detecção de bordas.

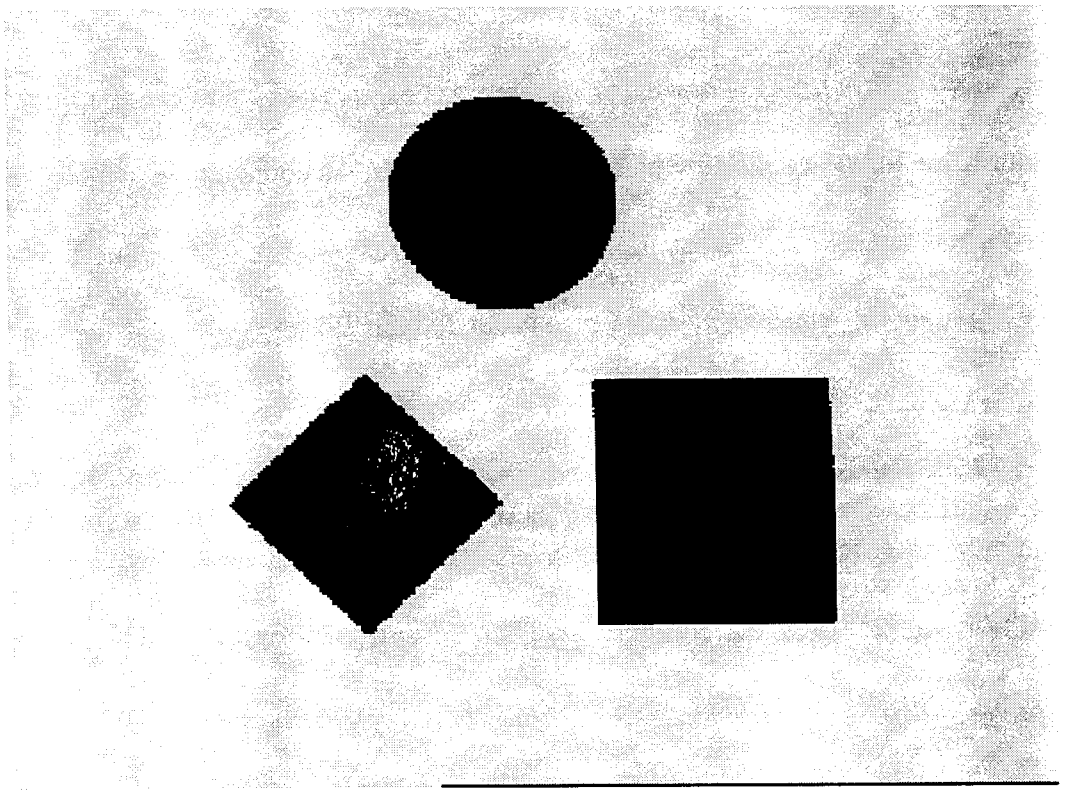

FIGURA 31 - Figuras geométricas. Imagem original.

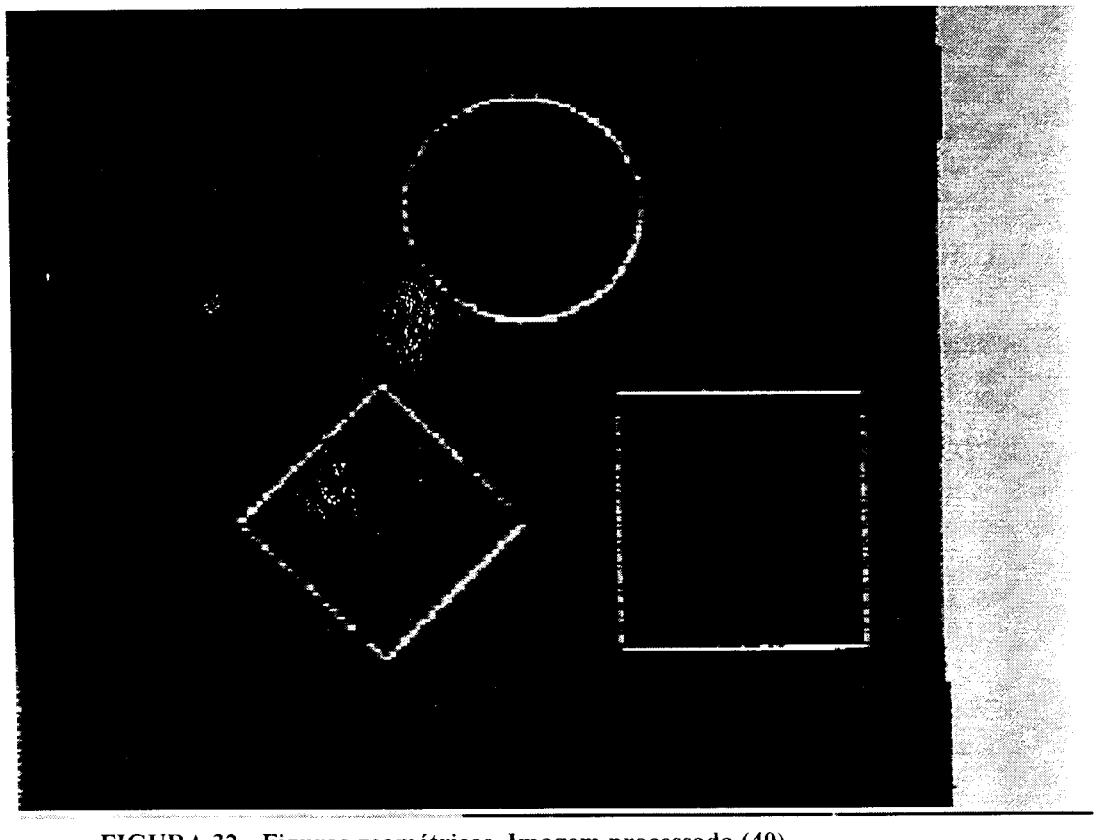

FIGURA 32 - Figuras geométricas. Imagem processada (40)

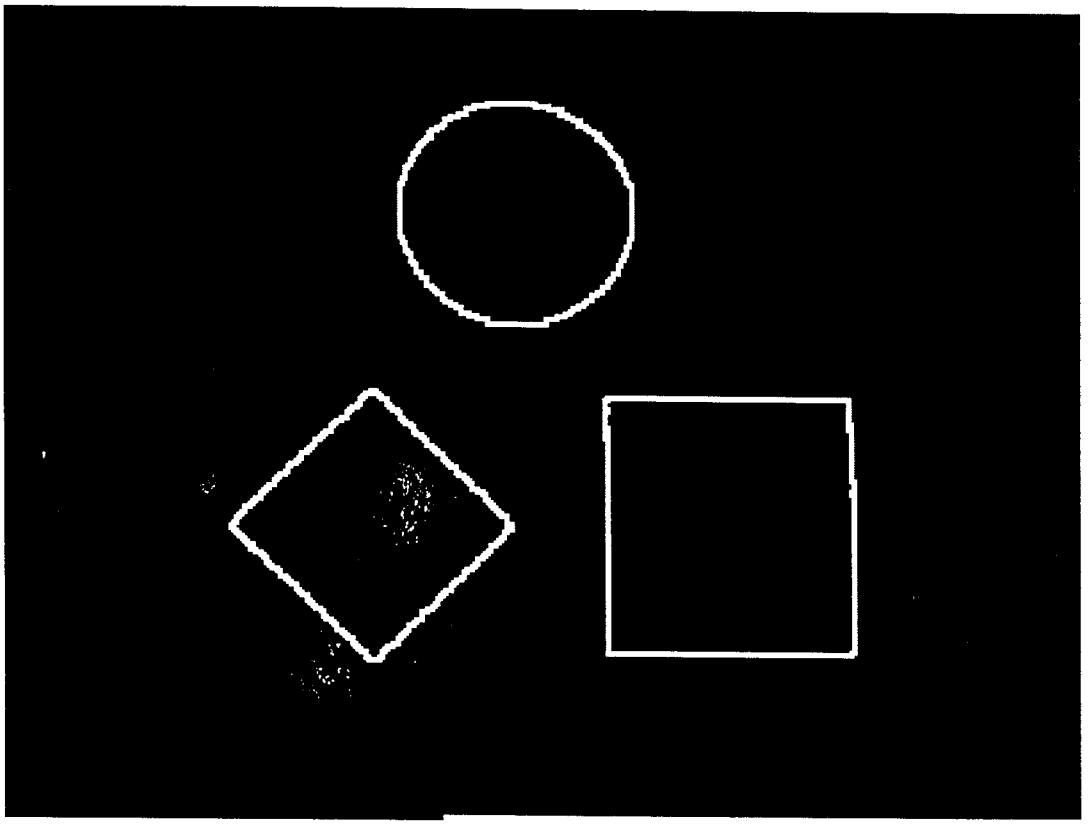

FIGURA 33 - Figuras geométricas. Imagem obtida pelo programa PhotoPaint.

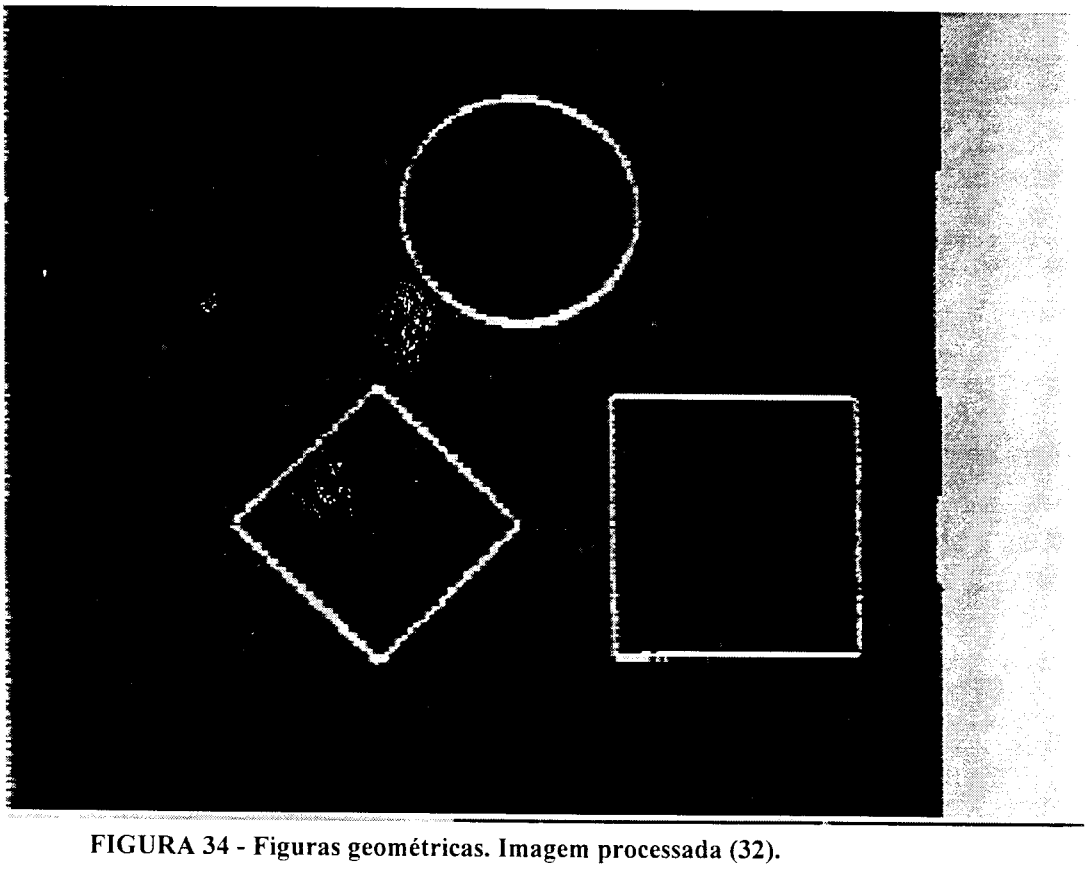

FIGURA 34 - Figuras geométricas. Imagem processada (32).

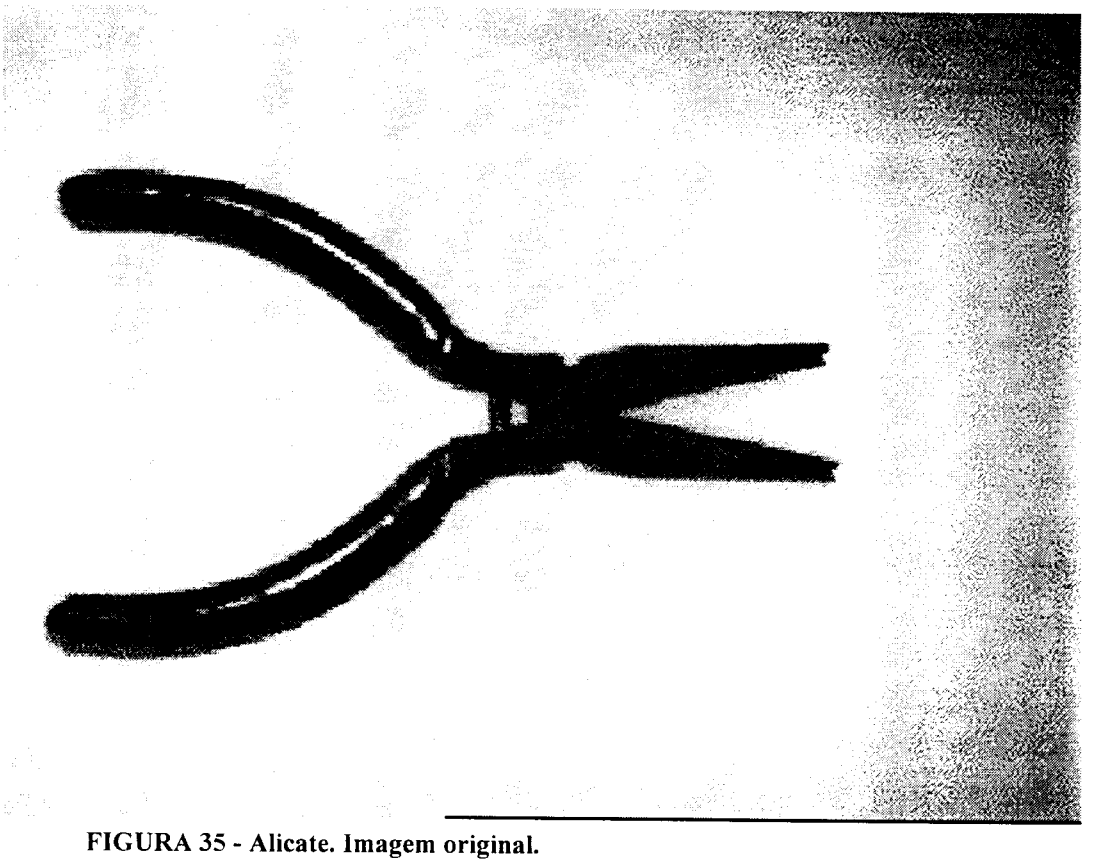

FIGURA 35 - Alicate. Imagem original.

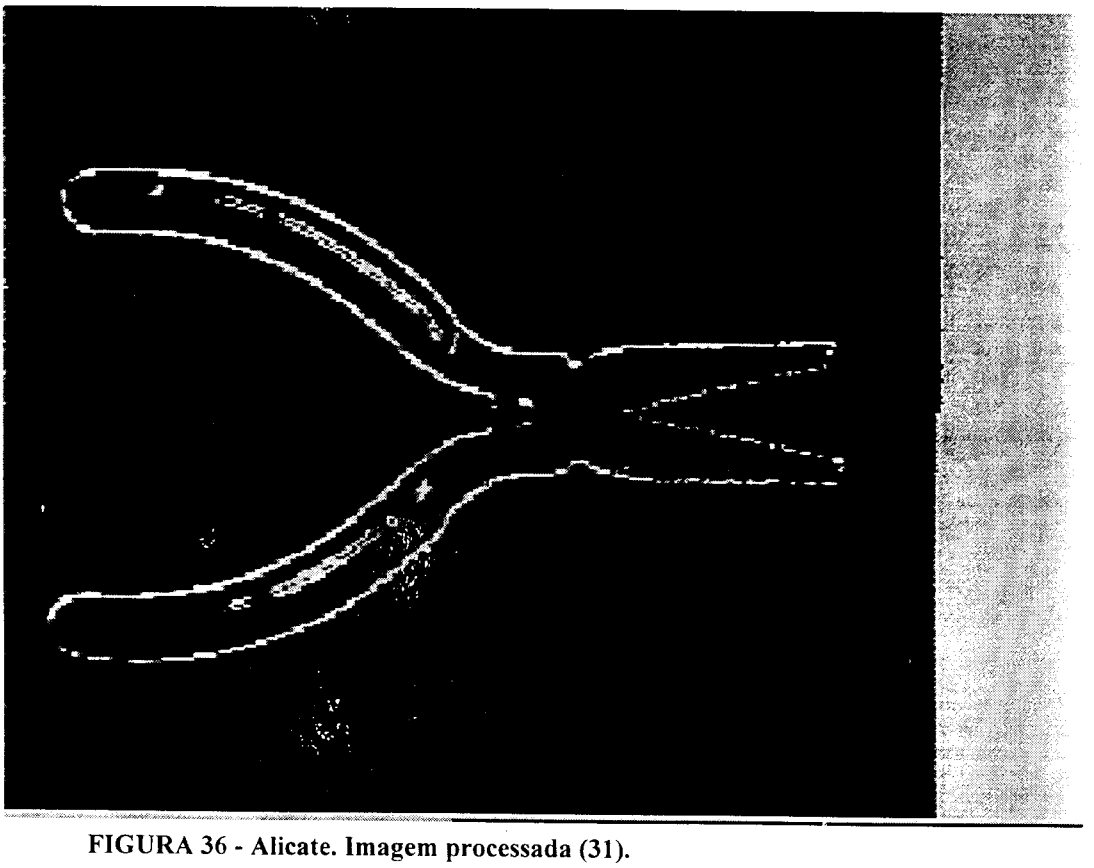

FIGURA 36 - Alicate. Imagem processada (31).

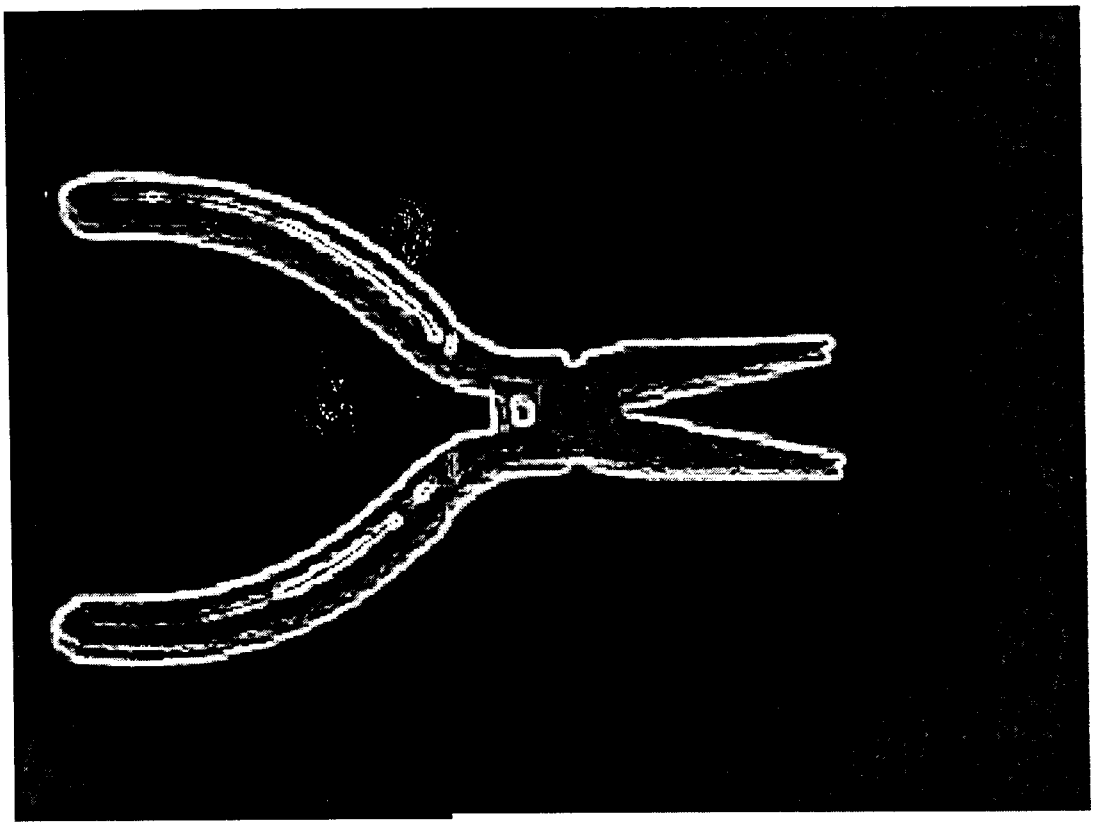

FIGURA 37 - Alicate. Imagem obtida pelo programa PhotoPaint.

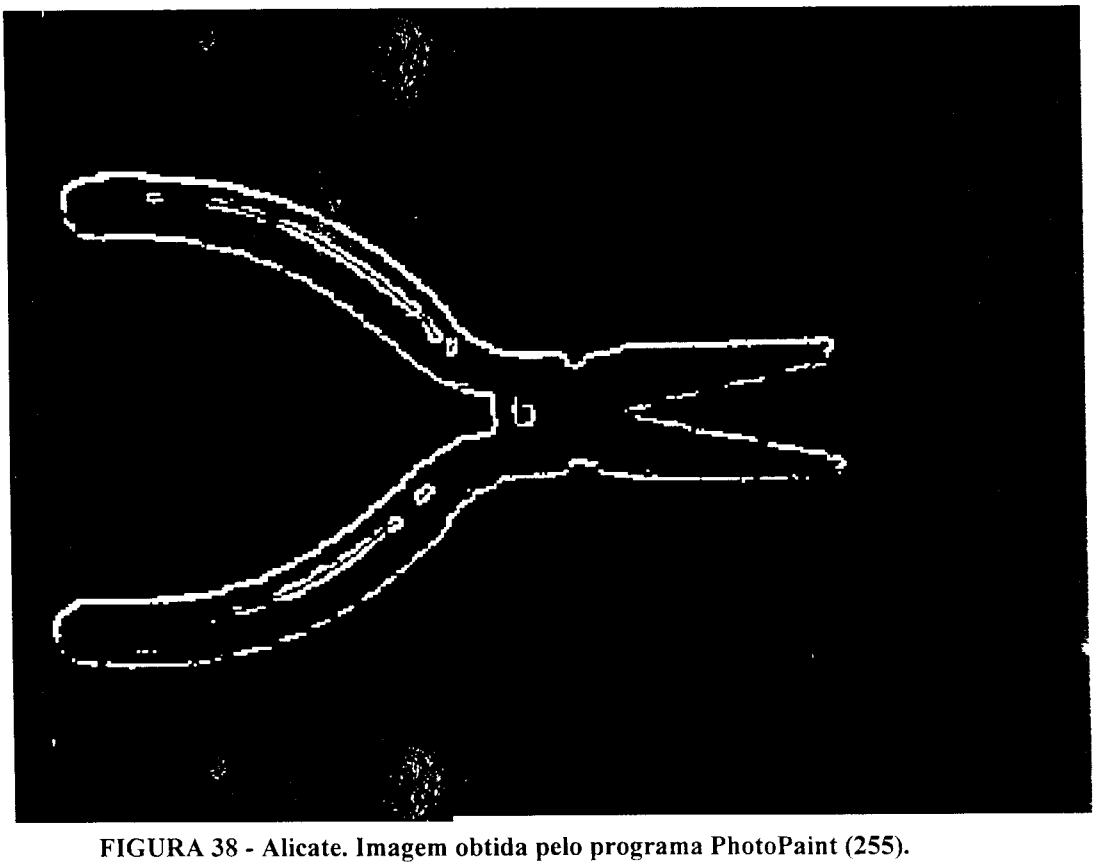

FIGURA 38 - Alicate. Imagem obtida pelo programa PhotoPaint (255).

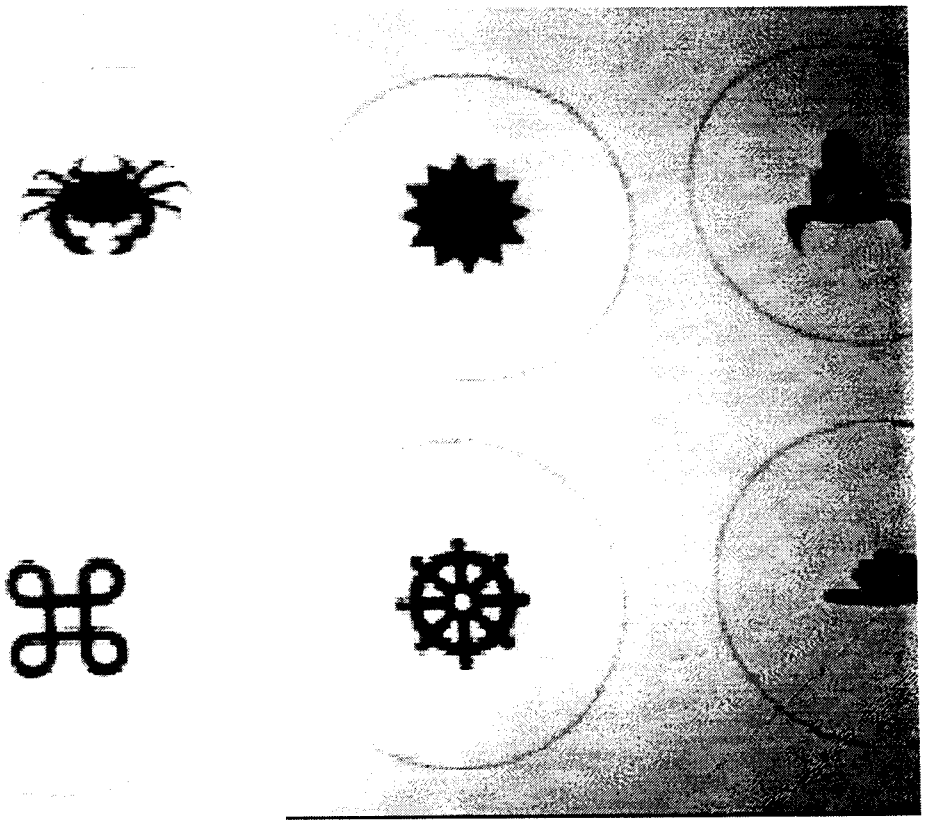

FIGURA 39 - Diversos padrões. Imagem original.

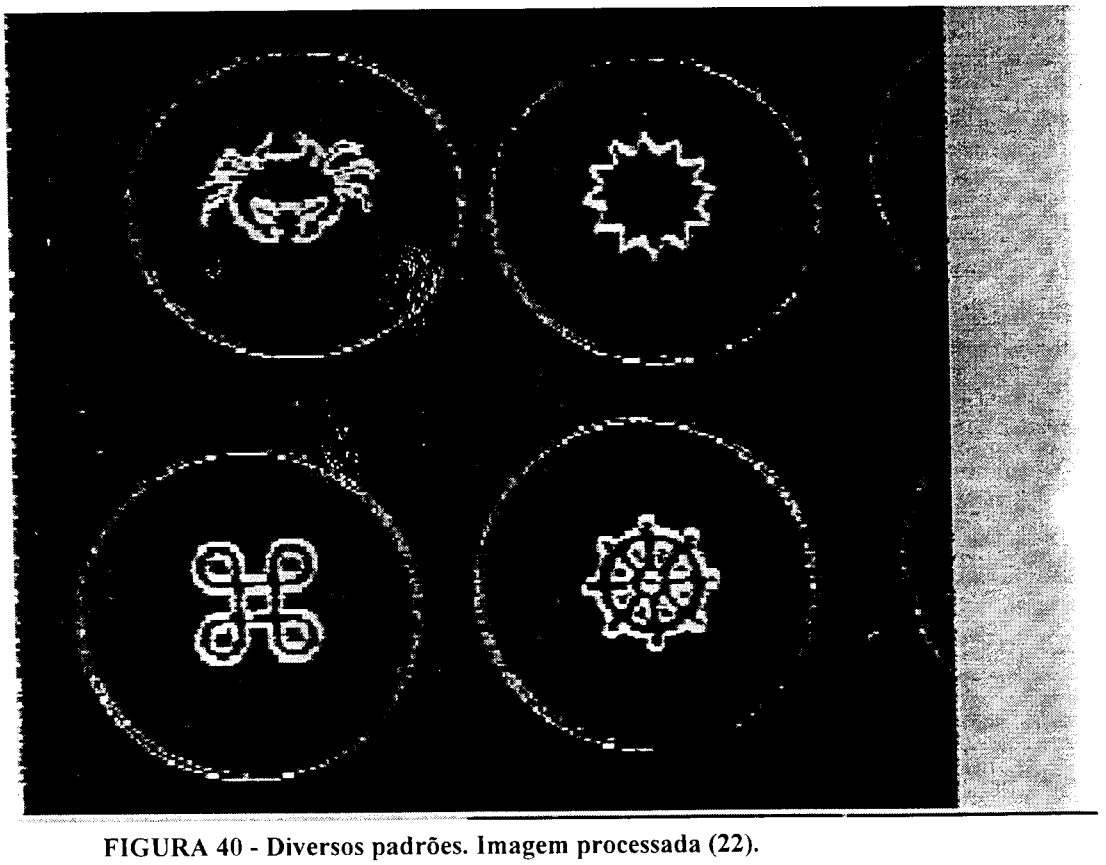

FIGURA 40 - Diversos padrões. Imagem processada (22).

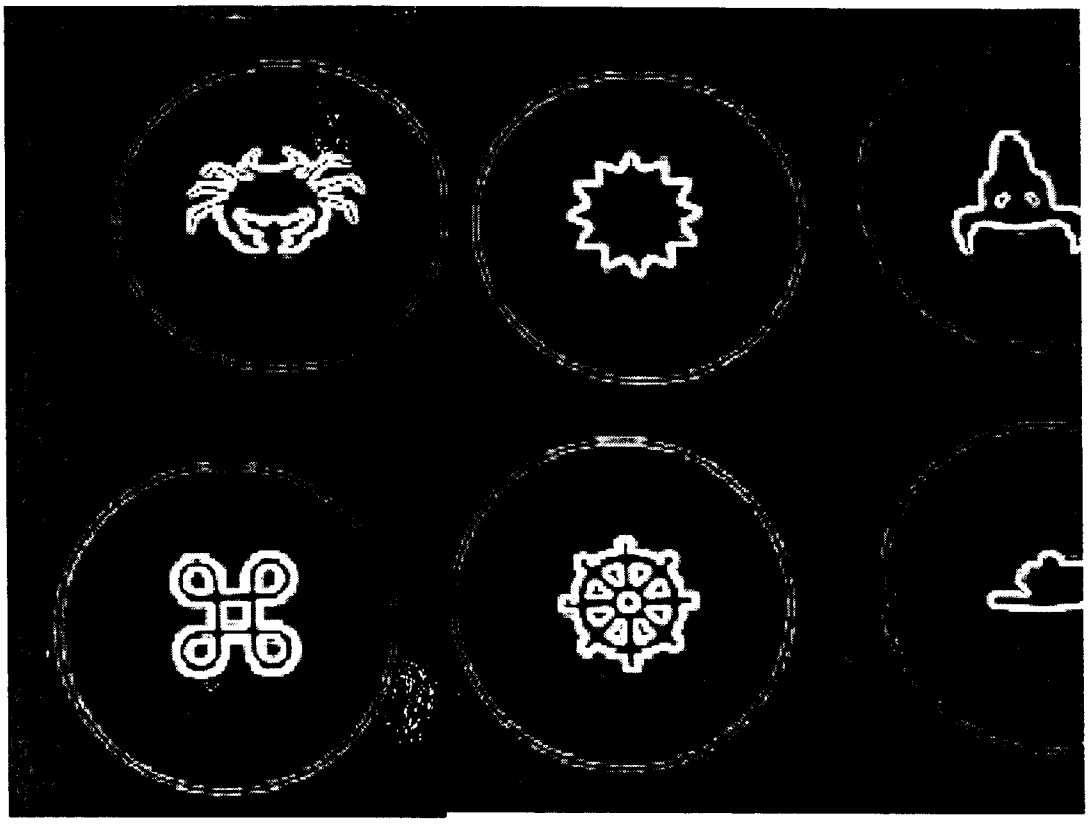

FIGURA 41 - Diversos padrões. Imagem obtida pelo programa PhotoPaint.

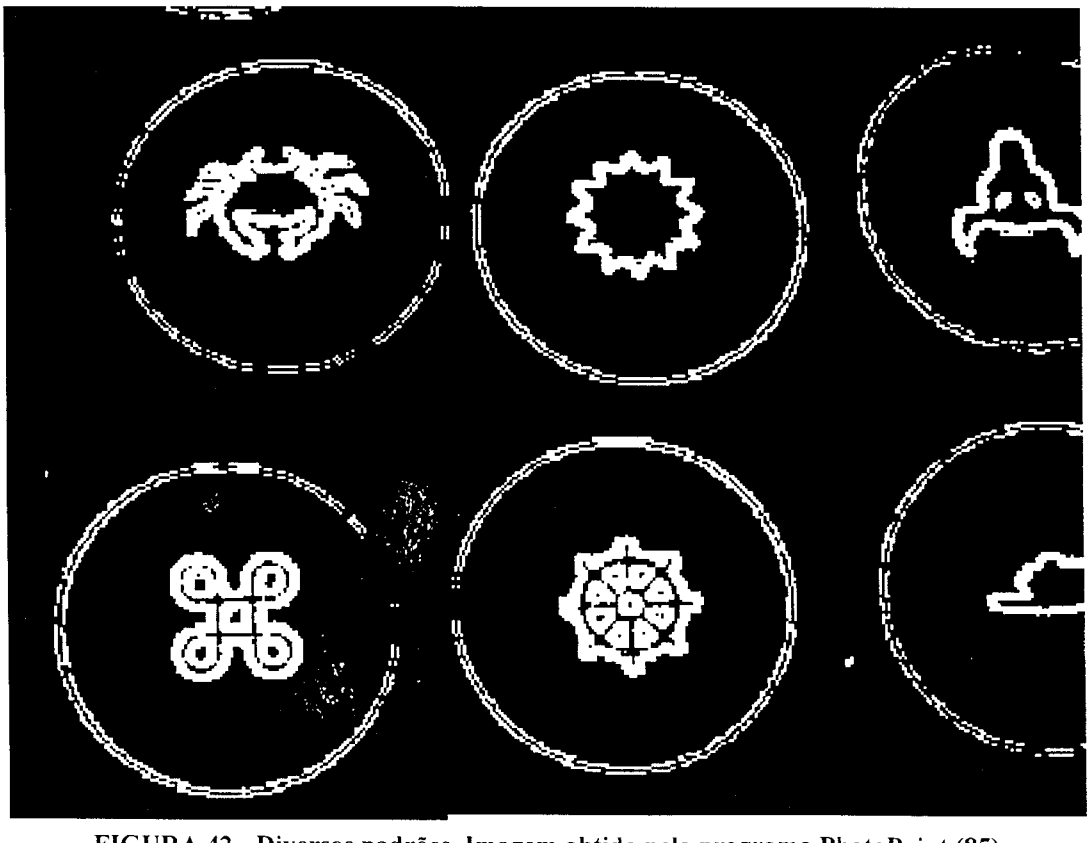

FIGURA 42 - Diversos padrões. Imagem obtida pelo programa PhotoPaint (85).

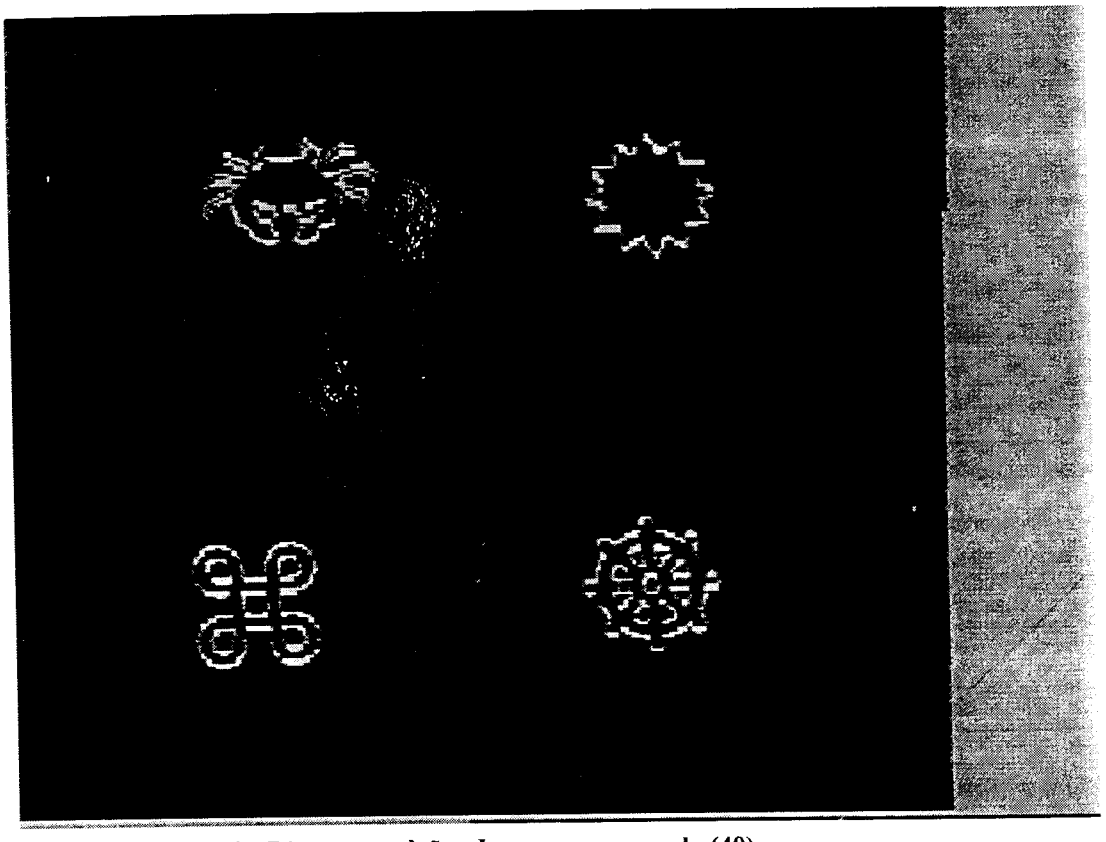

FIGURA 43 - Diversos padrões. Imagem processada (40).

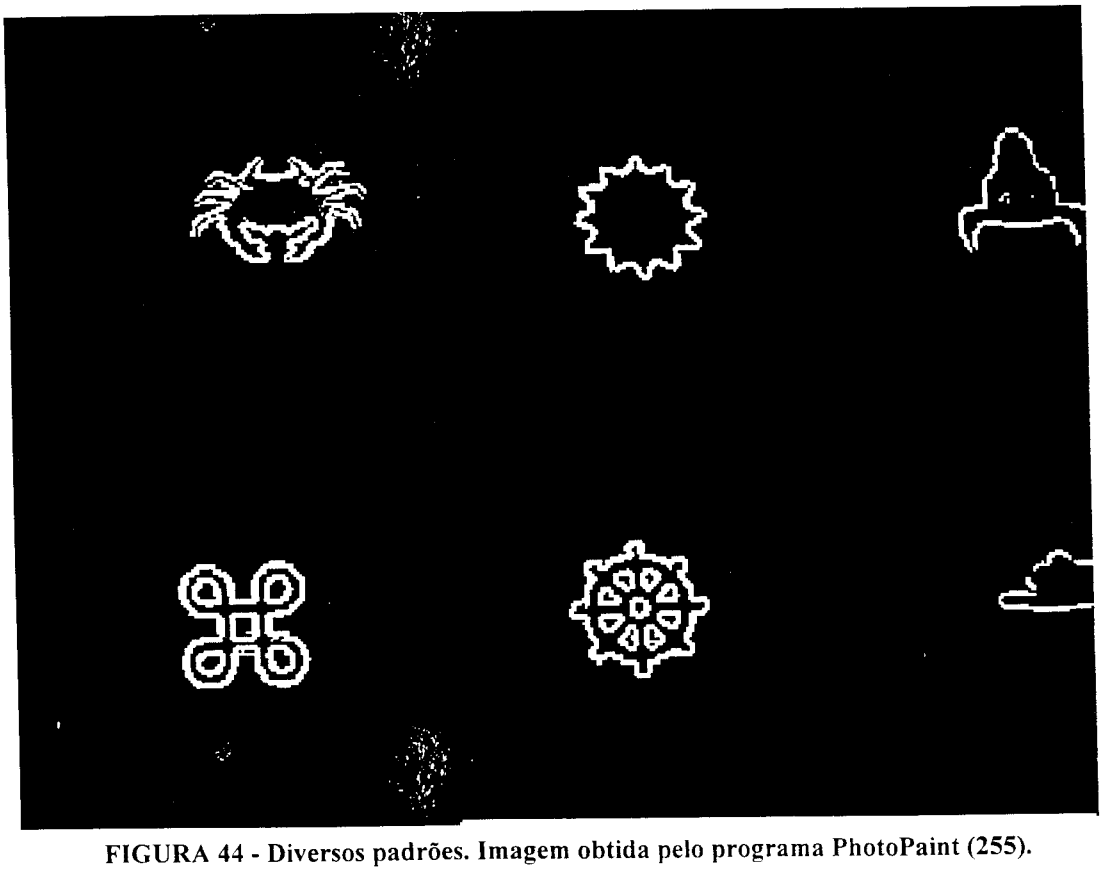

FIGURA 44 - Diversos padrões. Imagem obtida pelo programa PhotoPaint (255).

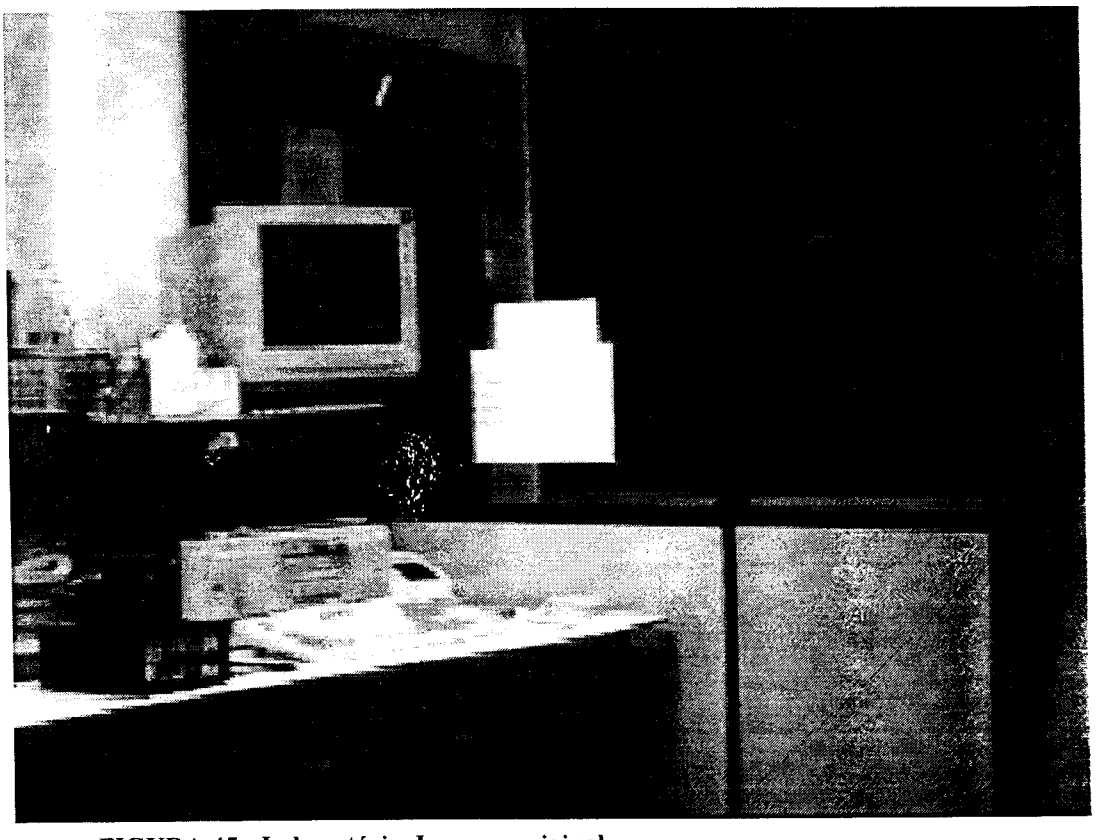

FIGURA 45 - Laboratório. Imagem original.

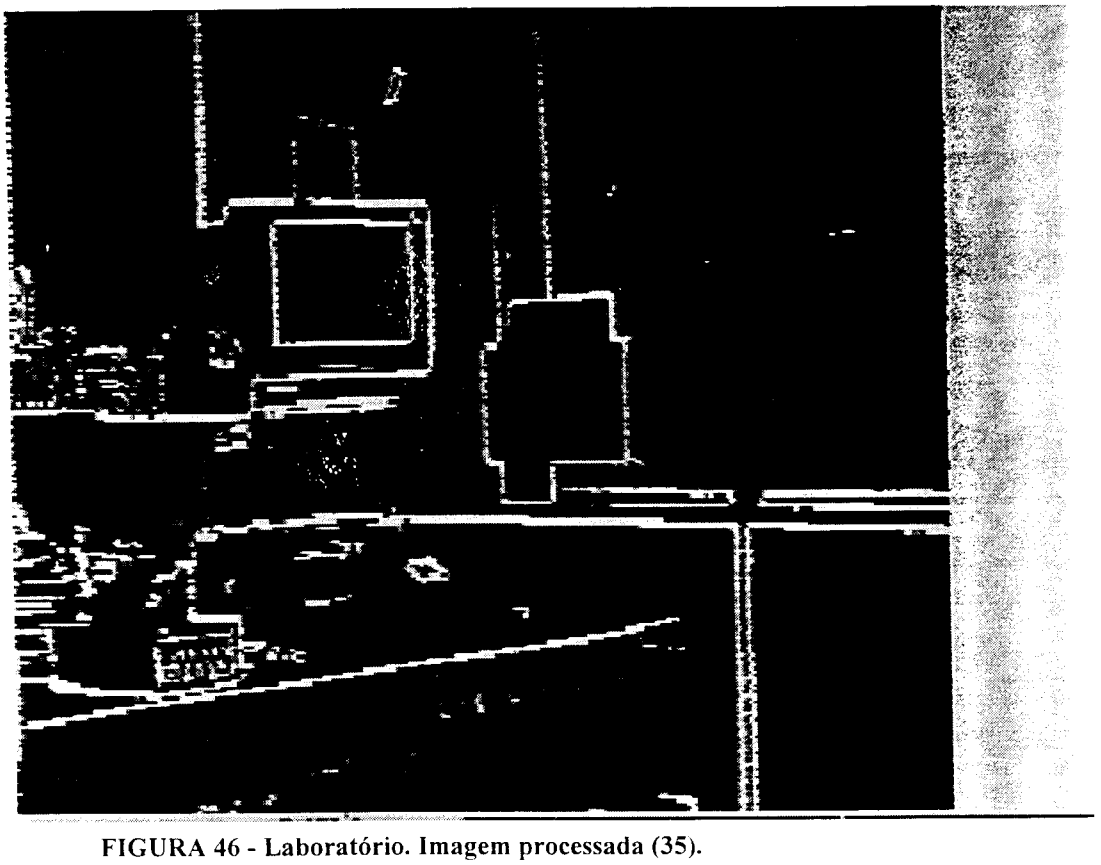

FIGURA 46 - Laboratório. Imagem processada (35).

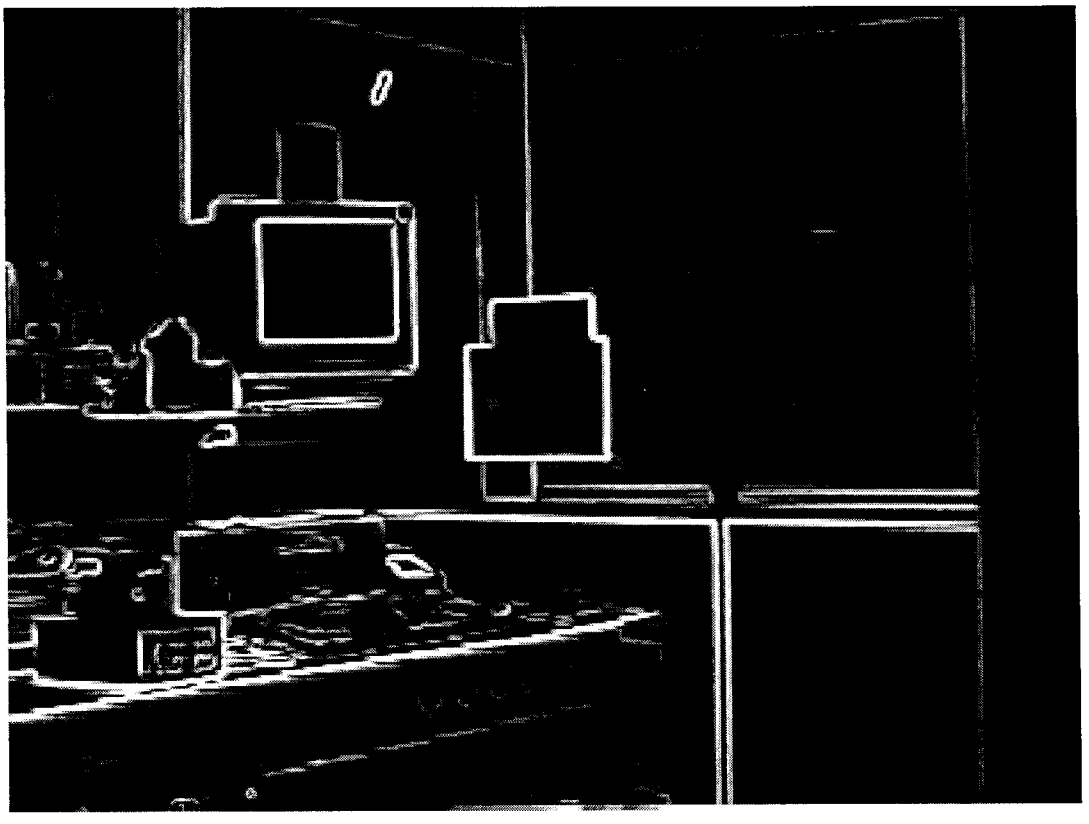

FIGURA 47 - Laboratório. Imagem obtida pelo programa PhotoPaint.

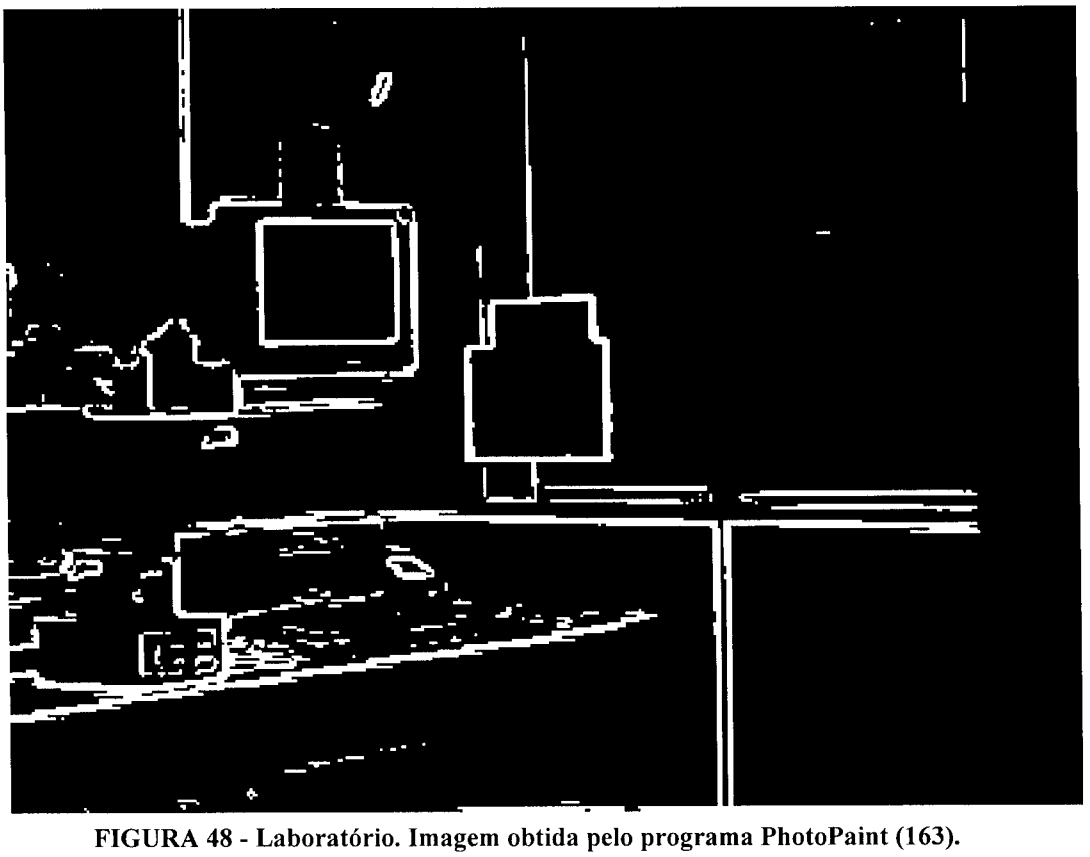

FIGURA 48 - Laboratório. Imagem obtida pelo programa PhotoPaint (163).

4.6 CONCLUSÕES

Neste capítulo apresentamos uma descrição do formato do sinal de vídeo e a implementação do sistema para Processamento Digital de Imagens baseado no operador Cruz de Roberts. Foram descritas todas as partes constituintes do sistemas e como elas foram implementadas

A escolha do sinal de vídeo entrelaçado impôs a necessidade de se utilizar uma memória de grande capacidade de armazenamento como uma linha de atraso. Uma vez aceita esta imposição, pôde-se desenvolver um sistema de aquisição de dados que pode ser utilizado com processadores baseados em máscaras de dimensões maiores do que 2x2, tais como os operadores de Prewitt e Sobel, utilizando um formato de sinal de vídeo comum.

As imagens obtidas demonstram o funcionamento do sistema montado e mostra que existem melhorias a serem feitas, tais como eliminação de ruídos e um melhor controle do número de pixels por linha.

5 CONCLUSÃO

Apresentamos a implementação de um sistema para Processamento Digital de Imagens em tempo real. O sistema foi baseado em um processador com arquitetura *pipeline*, dedicado para extração de bordas e baseado no operador Cruz de Roberts.

As áreas para aplicação do Processamento Digital de Imagens são bastante amplas, e a detecção de borda, apesar de ser um primeiro passo importante no processamento de imagens, é apenas um pequeno exemplo das possibilidades de aplicação.

Os sistemas para Processamento Digital de Imagens podem ser implementados das mais diversas formas, tanto em *hardware* como em *software*. A escolha da forma de implementação irá depender da aplicação. As áreas que necessitam de processamento em tempo real requerem que o processamento seja feito em taxas de transmissão de vídeo, o que pode ser conseguido por meio de supercomputadores, ou por meio de arquiteturas dedicadas.

Os avanços tecnológicos nas áreas de integração em larga escala tem permitido o rápido desenvolvimento de arquiteturas dedicadas para o processamento

digital de imagens, e o uso de ferramentas computacionais tem efetivado este rápido desenvolvimento, permitindo diminuir os custos e erros de implementações.

O sistema aqui apresentado pôde ser inteiramente montado e testado com a ajuda de uma ferramenta computacional, o que permitiu identificar alguns erros de implementação.

O uso do sinal de vídeo monocromático entrelaçado permite que o sistema seja utilizado com fontes de sinais baseadas neste mesmo padrão, que é um padrão utilizado pela maioria das câmeras de vídeo existentes. Apesar disto, houve a necessidade de se desenvolver um circuito lógico que permitisse a utilização desse padrão de vídeo.

O sistema ainda necessita de algumas melhorias. É necessário um maior controle do número de pixels por linha e do número de linhas a serem processados, para que se possa processar pequenas áreas de interesse da imagem, e não toda a imagem.

A simplicidade do operador utilizado permitiu que a implementação do processador fosse feita com circuitos lógicos discretos, e a obtenção de bons resultados. Apesar disto, este é um operador muito simples, muito sensível a ruídos e que foi inicialmente utilizado para extração de bordas de poliedros opacos.

Fica em aberto a possibilidade de desenvolvimento de outros processadores para o Processamento Digital de Imagens, utilizando as ferramentas computacionais da Altera e o próprio sistema desenvolvido, modificando-se apenas o processador.

6 BIBLIOGRAFIA

[1] Gonzalez, R. C. e Woods, R. E., "Digital Image Processing," Addison-Wesley Publishing Company, 1993

[2] Levine, M. D., "Vision in Man and Machine," McGraw-Hill Publishing Company, New York, 1985

[3] Wilson, A., "Electronic Imaging '85: Digitizing the World," Digital Design, vol. 15, no. 9, pp. 50-54, 1985

[4] Siegel, S., "VME Building Blocks Make Image Processing a Snap," Digital Design, vol. 15, no. 10, pp. 48-52, 1985

[5] Wilson, A., "Array Processors: The Best Way to Process Images," Digital Design, vol. 16, no. 1, pp. 47-52, 1986

[6] Ruetz, P. A. e Brodersen, R. W., "Architectures and Design Techniques for Real-Time Image-Processing IC's," IEEE Journal of Solid-State Circuits, vol. SC-22, no. 2, pp. 233-250, 1987

[7] Lee, C.; Catthoor, F. V. M. e De Man, H. J., "An Efficient ASIC Architecture for Real-Time Edge Detection," IEEE Transactions on Circuits and Systems, vol. 36, no. 10, pp. 1350-1359, 1989

[8] Barros, M., "Uma Metodologia de Projeto e Implementação de Operadores para Processamento Digital de Imagens em Tempo Real Usando Field Programmable Gate Arrays (FPGA)," http://www.visgraf.impa.br/sibgrapi96/trabs/abst/a68.html

[9] Apple Newton MessagePad, Personal Computer World, vol. 16, no. 9, pp. 448-454, 1993

[10] Fathy, M. e Siyal, M. Y., "A Window-based Edge Detection Technique for Measuring Road Traffic Parameters in Real-Time," Real Time Imaging, **1**, pp. 297-305, 1995

[11] Wilson, A. C., "Video Conversion With ADCs and DACs," Digital Design, vol. 16, no. 2, pp. 57-62, 1986

[12] Williams, D. J. e Shah, M., "Edge Characterization Using Normalized Edge Detector," CVGIP: Graphical Models and Image Processing, vol. 55, no. 4, pp. 311-318, 1993

[13] Roberts, L. G., "Machine Perception of Three-Dimensional Solids," em Tippett, J. T., et al. (eds.), "Optical and Electro-Optical Information Processing," MIT Press, Cambridge, Mass., pp. 159-197, 1965

[14] Prewitt, J. M. S., "Object Enhancement and Extraction," em Lipkin, B. S. e Rosenfeld, A. (eds.) "Picture Processing and Psychopictorics," Academic, New York, pp. 75-149, 1970

[15] Bernsen, A. C., "An Objective and Subjective Evaluation of Edge Detection Methods in Images," Philips Journal Research, no. 46, pp. 57-94, 1991

[16] Ziou, D. e Tabbone, S., "A Multi-Scale Edge Detector," Pattern Recognition, vol. 26, no. 9, pp. 1305-1314, 1993

[17] Nalwa, V. S. e Binford, T. O., "On Detecting Edges," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-8, no. 6, pp. 699-714, 1986

[18] Canny, J., "A Computational Approach to Edge Detection," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-8, no. 6, pp. 679-698, 1986

[19] Clark, J. J., "Authenticating Edges Produced by Zero-Crossing Algorithms," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-11, no. 1, pp. 43-57, 1989

[20] Frei, W. e Chen, C., "Fast Boundary Detection: A Generalization and a New Algorithm," IEEE Transactions on Computers, vol. C-26, no. 10, pp. 988-998, 1977

[21] Haralick, R. M., "Digital Step Edges from Zero Crossing of Second Directional Derivatives," in Chellappa, R. e Sawchuck, A. A., "Digital Image Processing and Analysis - Volume 2: Digital Image Analysis," IEEE Computer Society Press, pp. 39-49, 1985

[22] Majumdar, B.; Sankarayya, N., e Majumdar, A. K., "An ASIC design for edge detection in real time," Microprocessing and Microprogramming 36, pp.55-69, 1992/93

[23] Kanopoulos, N.; Vasanthavada, N. e Baker, R. L., "Design of an Image Edge Detection Filter Using the Sobel Operator," IEEE Journal of Solid-State Circuits, vol. 23, no. 2, pp. 358-367, 1988

[24] Yamashina, M.; Enomoto, T.; Kunio, T.; Tamitani, I.; Harasaki, H.; Nishitani, T.; Satoh, M. e Kikuchi, K., "A Microprogramable Real-Time Video Signal Processor (VSP) LSI," IEEE Journal of Solid-State Circuits, vol. SC22, no. 6, pp. 1117-1123, 1987

[25] Quénot, G. e Zavidovique, B., "A Data-Flow Processor for Real-Time Low-Level Image Processing," http://www.limsi.fr/Individu/quenot/publis/abstracts /asic91a.html

[26] Houzet, D. "Real Time Image Processing with a MIMD Computer," Real-Time Imaging, no. 2, pp. 383-392, 1996

[27] Revista Nova Eletrônica, Ano VI, no. 71, pp. 72-77, 1983

[28] Revista Nova Eletrônica, Ano VI, no. 72, pp. 76-80, 1983

[29] BrookTree Home Page, WWW page http://www.brooktree.com

[30] OKI Semiconductors, WWW page http://www.okisemi.com

[31] National Semiconductors, WWW page http://www.national.com

[32] Maxim Integrated Products, Inc, WWW page http://www.maxim-ic.com

[33] Sonderbrink, J., "Dual Video Amplifier," Elektor Eletronics,vol. 19, no. 207, pp.36-37, 1993

APÊNDICE

Apresentamos aqui algumas figuras obtidas do programa MAXPLUS2, da Altera. Este programa é destinado à programação de circuitos FPGA e foi utilizado para simular o processador apresentado nesta Dissertação. A primeira figura mostra todo o circuito do processador; as seguintes mostram cada um dos blocos do processador e as duas últimas mostram a simulação do mesmo.

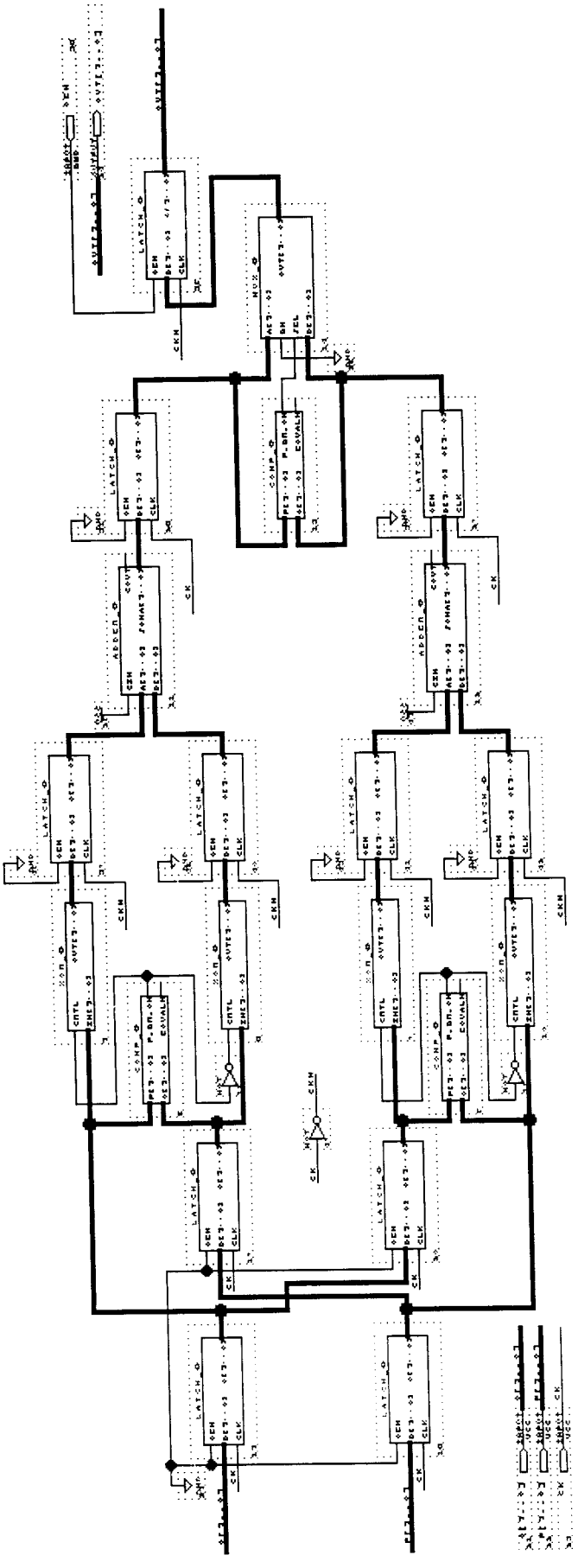

Figura A - Circuito do processador Cruz de Roberts

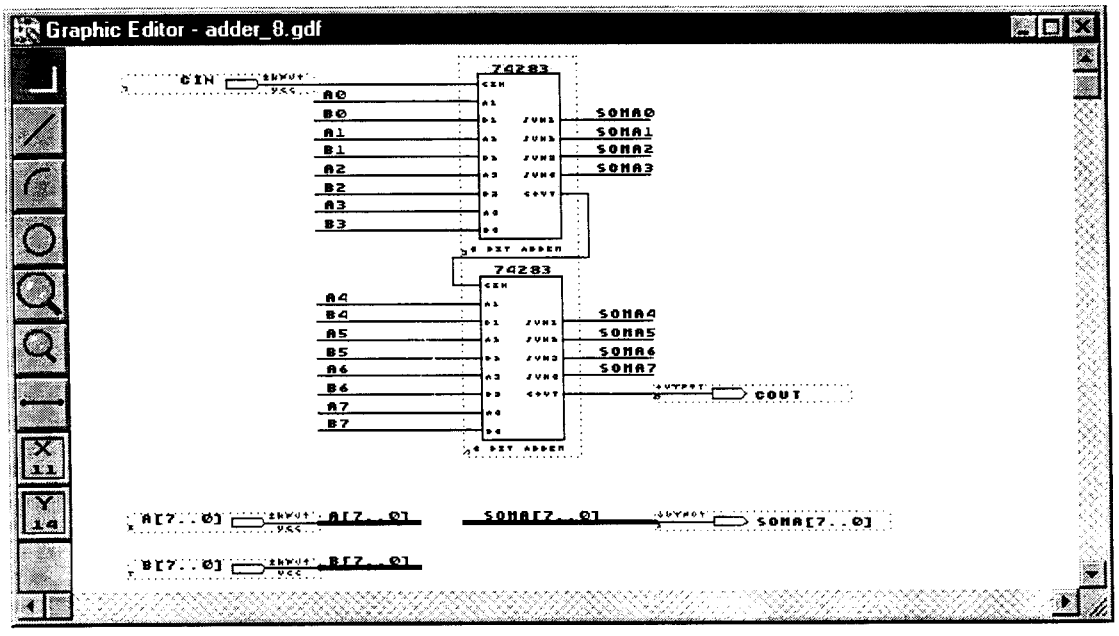

Figura B - Circuito somador de 8*bits*.

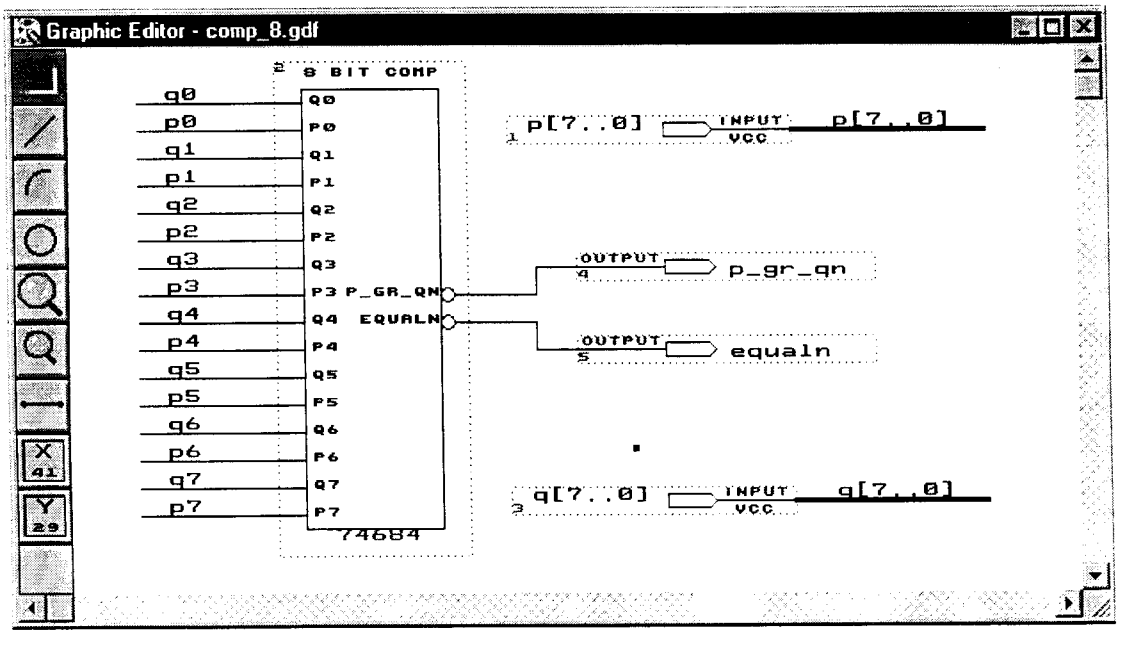

Figura C - Circuito comparador de 8*bits*.

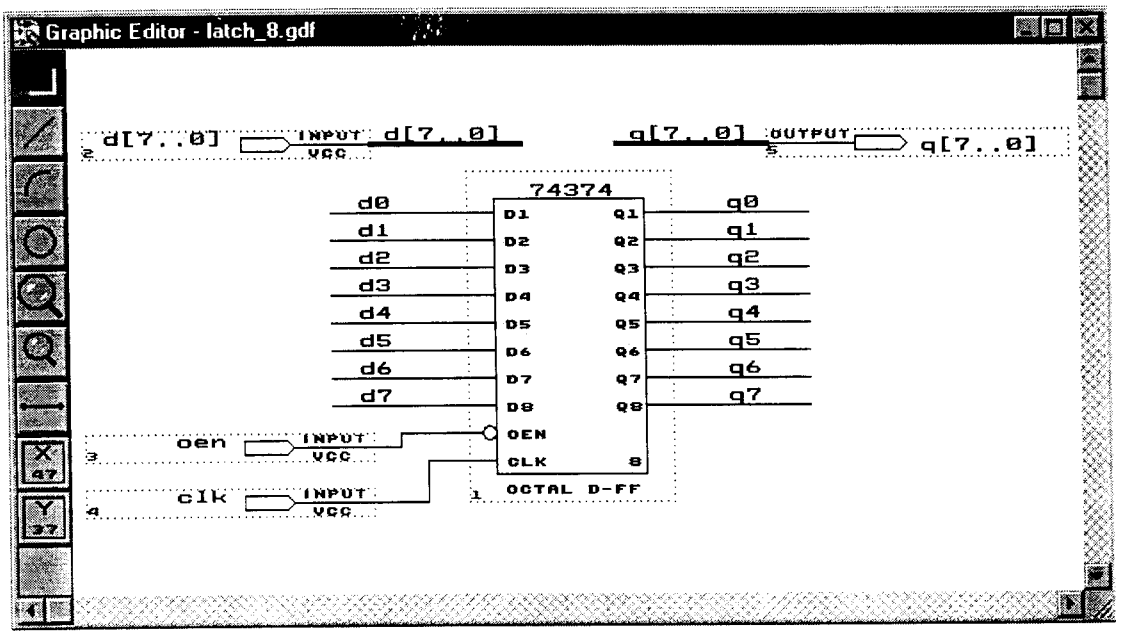

Figura D - Registrador de 8*bits*.

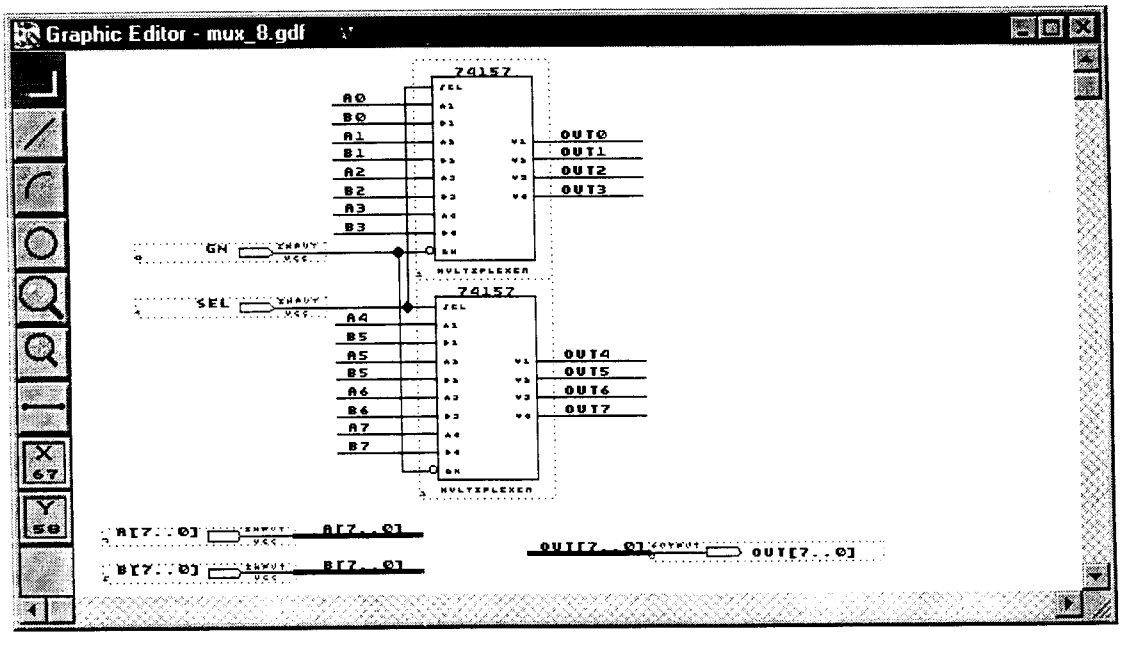

Figura E - Circuito multiplexador 2-1 de 8*bits*.

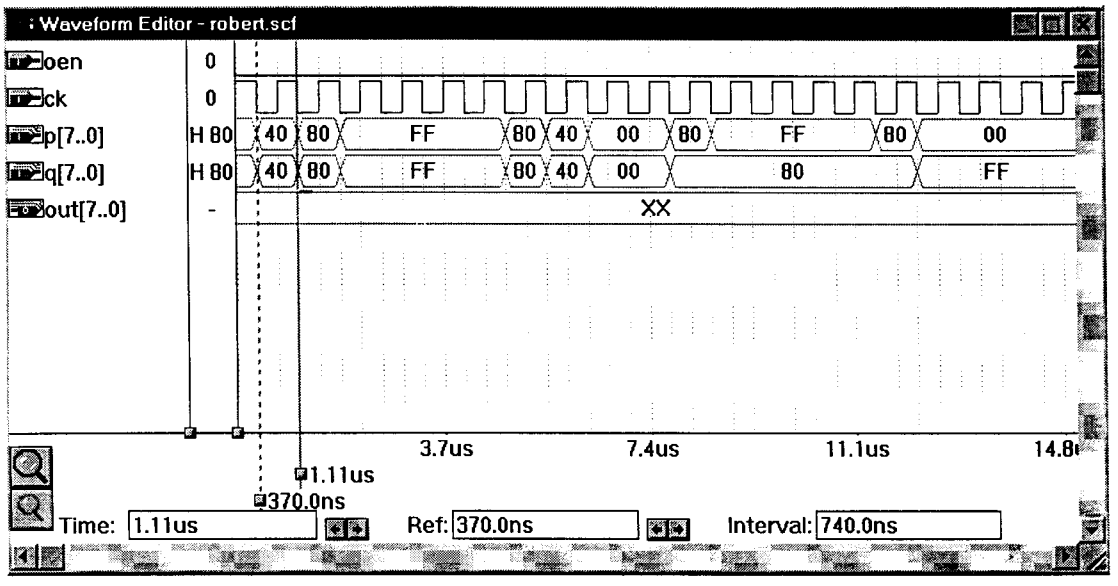

Figura F - Entrada de dados para simulação. As entradas **p[7..0]** e **q[7..0]** são as entradas das linhas de vídeo $I_{i,j}$ e $I_{i+1,j}$.

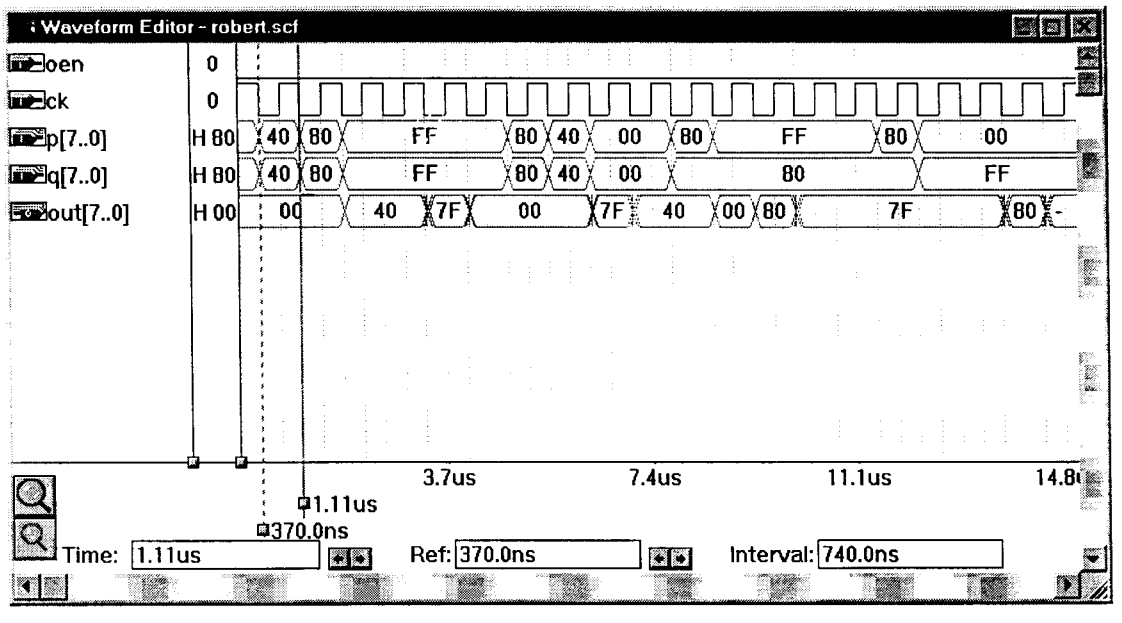

Figura G - Resultado da simulação. Podemos ver na saída **out[7..0]** o resultado da simulação.