UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE CIÊNCIAS E TECNOLOGIA
CURSO DE MESTRADO EM ENGENHARIA ELÉTRICA

UM CIRCUITO INTEGRADO PARA DETEÇÃO DE BORDAS COM O OPERADOR DE ROBERTS

LÍRIDA ALVES DE BARROS

CAMPINA GRANDE
MAIO - 1990

UM CIRCUITO INTEGRADO PARA DETEÇÃO DE BORDAS COM O OPERADOR DE ROBERTS

LÍRIDA ALVES DE BARROS

UM CIRCUITO INTEGRADO PARA DETEÇÃO DE BORDAS COM O OPERADOR DE ROBERTS

Dissertação apresentada ao Curso de MESTRADO EM ENGENHARIA ELÉTRICA, da Universidade Federal da Paraíba, em cumprimento às exigências para obtenção do Grau de Mestre.

ÁREA DE CONCENTRAÇÃO: PROCESSAMENTO DA INFORMAÇÃO

William Ferreira Giozza, D.Ing.
(Orientador)

CAMPINA GRANDE
MAIO - 1990

B277c Barros, Lirida Alves de.
Um circuito integrado para deteção de bordas com o operador de Roberts / Lirida Alves de Barros. - Campina Grande, 1990.
184 f.

Dissertação (Mestrado em Engenharia Elétrica) - Universidade Federal da Paraíba, Centro de Ciências e Tecnologia, 1990.
"Orientação : Prof. Dr. William Ferreira Giozza".
Referências.

1. Circuitos Integrados. 2. Detecção de Bordas - Técnica. 3. Circuito - Validação e Simulação. 4. Dissertação - Engenharia Elétrica. I. Giozza, William Ferreira. II. Universidade Federal da Paraíba - Campina Grande (PB). III. Título

CDU 621.3.049.77(043)

UM CIRCUITO INTEGRADO PARA DETEÇÃO DE BORDAS
COM O OPERADOR DE ROBERTS

LÍRDIDA ALVES DE BARROS

DISSERTAÇÃO APROVADA EM 30.05.90

WILLIAM FERREIRA GIOZZA, Dr.Ing., UFPB
Orientador

JEAN ALBERT BODINAUD, Dr., USP
Componente da Banca

ARNALDO DE ALBUQUERQUE ARAÚJO, D.Sc., UFMG
Componente da Banca

JOÃO MARQUES DE CARVALHO, Ph.D, UFPB
Componente da Banca

CAMPINA GRANDE - PB
MAIO - 1990

AGRADECIMENTOS

A autora deseja agradecer:

- Ao Prof. Dr. William Ferreira Giozza, pela orientação e coordenação do trabalho.

- Ao cooperante Olivier Giordano que, através do Projeto CAPES/COFECUB 85/88, pôde oferecer relevante contribuição na execução deste trabalho.

- Ao Prof. Dr. Jean Albert Bodinaud, pelo apoio e orientação dados durante a realização do estágio na USP.

- A seus pais, Oscar e Regina, pelo carinho e incentivo dados em todas as horas.

- A seus irmãos, Lindinalva, Martins, Deusimar e, em especial, a Marcelo, pela constante colaboração.

- Aos amigos, cujas presenças sempre tornam os momentos especiais.

- Ao CNPq, pelo suporte financeiro dado, através de bolsa de estudo.

Aos meus pais

SUMÁRIO

LISTA DE FIGURAS

RESUMO

O desenvolvimento da competência para projeto e controle do processo de fabricação de circuitos integrados tem permitido a integração, em "ASICs" (Circuitos Integrados de Aplicação Específica, de funções comumente usadas em processamento digital de imagens .

Neste trabalho, são realizados estudos sobre técnicas de deteção de bordas e aspectos envolvidos no projeto de circuitos integrados. Uma arquitetura de circuito integrado "ASIC" para tarefas de deteção de bordas em tempo real com o operador de Roberts é proposta. A metodologia de projeto para o "ASIC" proposto (célula padrão e divisão hierárquica), bem como as simulações e a validação do circuito, obtidas com auxílio das ferramentas ALLIANCE e SOLO 1400, são apresentadas.

ABSTRACT

Advances in VLSI technology have made attractive to implement image processing functions algorithms in Application Specific Integrated Circuits (ASICs).

In this work, digital image edge detection techniques and aspects of integrated circuits design are studied. An architecture for real time implementation of the Roberts operator is proposed.

CMOS technology, standard-cell methodology and hierarchical approach are used in the design. ALLIANCE and SOLO 1400 CAD tools were used to simulate and validate the Robert's operator ASIC.

1 INTRODUÇÃO

Nos últimos anos, tem sido de grande importância o processamento digital de imagens por computador. Entre as tarefas envolvendo o processamento digital de imagens, inclui-se a deteção de bordas ou contornos produzida por filtros espaciais, lineares e não-lineares, passa-altas [1].

O processo de deteção de bordas implica na geração de informações sobre as fronteiras ou contornos dos objetos ou regiões presentes na imagem. Para isso, envolve a manipulação de parâmetros texturais (finura, grossura) e do comportamento dos níveis de cinza (imagens em preto e branco) da cena digitalizada. Seus objetivos principais podem ser a imediata interpretação visual da imagem ou o seu pré-processamento para tarefas de classificação, reconhecimento de padrões, segmentação e codificação [2, 3]. A Figura 1.1 apresenta a deteção de bordas em aplicação de codificação de imagens, enquanto que na Figura 1.2 a deteção de bordas é mostrada como tarefa inicial em um sistema típico de visão robótica. A codificação de imagens é uma operação necessária para a introdução de imagens em Redes Digitais de Serviços Integrados (RDSI).

Os algoritmos para deteção de borda podem ser do tipo global ou local. Os métodos globais, em princípio, usam informação de toda a imagem para estabelecer a presença de uma borda em algum ponto. Ao contrário, os métodos locais usam somente informação de uma vizinhança limitada do ponto

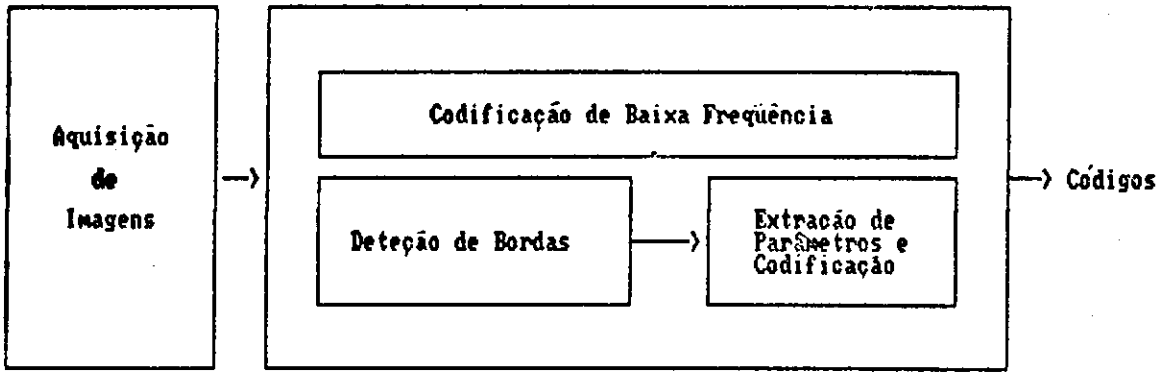


Figura 1.1: Estratégia de codificação de imagens.

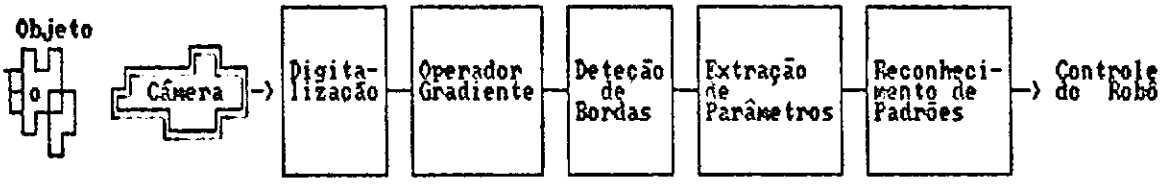


Figura 1.2: Deteção de bordas em sistema de robótica.

de interesse. Os métodos locais podem ser ainda subdivididos nas categorias de métodos de realce/limiar ("enhancement/thresholding") [4] ou de adaptação de bordas. Esta última categoria envolve a adaptação do modelo de uma borda ideal aos padrões presentes na imagem [5-7]. Os métodos de realce/limiar, por outro lado, detectam bordas usando operadores gradientes do tipo diferencial e direcional [8].

Com o avanço da tecnologia para integração de

componentes eletrônicos, tem-se tornado atrativa a implementação de funções de processamento de imagem específicas em uma única pastilha ("chip") para obter alto desempenho, minimizando tamanho e potência a nível de sistema. Este estágio do desenvolvimento resulta de uma seqüência de passos na história da tecnologia dos processos de integração.

O desenvolvimento da competência no projeto em silício e no controle do processo de fabricação permitiu a integração de famílias de sistemas complexos como microprocessadores, memórias, circuitos de interface, etc. Cada vez mais algoritmos de processamento digital de sinais podem ser implementados em um único circuito integrado. Técnicas cada vez mais complexas compartilham a arquitetura de "chips" projetados para aplicações múltiplas. Sistemas de processamento paralelo para imagens usando processadores digitais são reportados na literatura [9]. Entretanto, estas complexas arquiteturas apresentam várias limitações, tais como um grande número de processadores e necessidade de lógica de controle para obter operação em tempo real [10], além de problemas com a programação de processadores paralelos [11]. Além disso, devem ser considerados as limitações de dissipação de potência, a quantidade de pinos e o custo decorrente da área total do "chip" utilizada. Por outro lado, considerando-se o avanço das ferramentas de auxílio a projeto ("CADs"), os circuitos integrados de aplicação específica ("ASICs") podem ser desenvolvidos rapidamente em um ambiente estruturado e se apresentam como uma solução adequada para a realização de algumas técnicas fundamentais de processamento digital de sinais, em particular, de sinais de imagem [11-22].

Neste trabalho, é proposta uma arquitetura para

implementação em tempo real de uma técnica de deteção de bordas em imagens digitalizadas. Trata-se de um estudo inicial visando à integração do "chip" em um sistema maior, também integrável, que realize o processamento e codificação de imagens para interfaces de redes de computadores com integração de serviços [23]. O algoritmo de deteção de bordas escolhido para implementação foi o operador de Roberts [4] em razão de sua sensibilidade às componentes de alta freqüência, relativa imunidade ao ruído e, em especial, pela sua simplicidade, o que implica em facilidades de projeto, pequena área de silício e, conseqüentemente, menor custo final do circuito integrado. Em [24], são realizados estudos comparativos usando implementações em linguagem C que mostram o bom desempenho deste algoritmo, em relação a outros mais complexos, em tarefas de geração de imagens gradientes.

A tecnologia CMOS e a metodologia de célula padrão ("standard cell") [25] foram empregadas para concepção dos blocos básicos do circuito proposto. A implementação e as simulações do circuito foram realizadas nos sistemas de desenvolvimento ALLIANCE [26] e SOLO 1400 [27], disponíveis através do Projeto CAPES/COFECUB nº 85/88 envolvendo uma cooperação a nível internacional com o Laboratório MASI - CAO & VLSI da Universidade de Paris 6 e, a nível nacional, com o LME/IEE/USP [28]. A fundição do "ASIC" operador de Roberts está prevista para o 6º PMU - Programa Multiusuário Brasileiro a ser realizado ainda este ano.

No Capítulo 2, é apresentado um estudo sobre deteção de bordas, apresentando várias técnicas correntes na literatura.

O Capítulo 3 apresenta a arquitetura proposta para o

"ASIC" operador de Roberts e fornece a descrição do seu funcionamento.

No Capítulo 4, tem-se uma descrição dos aspectos envolvidos no projeto de um circuito integrado, são descritos os ambientes de simulação/validação do projeto e se mostram os resultados obtidos.

Finalmente, no Capítulo 5, são apresentadas as conclusões do trabalho e sugestões para trabalhos complementares.

2 O OPERADOR DE ROBERTS

Este capítulo apresenta um estudo do processo de deteção de bordas juntamente com algoritmos correntes na literatura e situa o operador de Roberts neste contexto.

2.1 Deteção de Bordas

A análise automática de uma imagem pode ser realizada através da técnica de segmentação. Esta técnica resulta na partição da imagem em regiões homogêneas em que os elementos (pixels, do inglês "picture elements") têm propriedades similares. O objetivo deste processo é separar um objeto na cena, do fundo da imagem ou de outros objetos que possam estar presentes. Os objetos podem ser distinguidos do fundo da imagem ou de outros objetos, pela presença de mudanças rápidas na intensidade luminosa da imagem (níveis de cinza) que marcam as bordas do objeto.

Uma operação na imagem que possibilita a deteção dessas mudanças abruptas dos valores dos níveis de cinza é a operação gradiente. Os algoritmos de deteção de bordas realizam um processo de diferenciação que leva em consideração fatores como continuidade das bordas do objeto, ruído, descontinuidade inerente do nível de cinza na borda do objeto e direcionalidade da borda [29-38].

A importância da deteção de bordas decorre do fato de, no processo de análise visual da imagem, as funções neurológicas envolvidas terem resposta máxima às variações de intensidade luminosa em uma imagem. São essas variações que determinam a presença de fronteiras entre regiões ou o contorno dos objetos presentes na cena. Em processamento digital de imagens, grande parte das técnicas que envolvem a análise ou reconhecimento de padrões inclui a deteção de bordas como etapa de pré-processamento ou mesmo como resultado final [11,15].

Por exemplo, uma estratégia básica de codificação de imagens, como a mostrada na Figura 1.1, tem como etapa fundamental a deteção das bordas, que representam as componentes de alta freqüência no sinal bidimensional [35]. Neste processo, as técnicas de codificação geralmente utilizam parâmetros extraídos dessas componentes de alta freqüência juntamente com atributos texturais das regiões homogêneas (componentes de baixa freqüência), para gerar os códigos e obter as taxas de compressão desejadas. De forma semelhante, os sistemas típicos de visão robótica, representados na Figura 1.2, têm como tarefa inicial a deteção dos contornos dos objetos [15].

Em ambas as aplicações apresentadas, cada etapa de processamento tem suas especificações de tempo impostas para a obtenção de um sistema estável em tempo real.

2.2 Abordagem Matemática da Deteção de Bordas

A função de imagem é representada por

$$I(x,y) \quad \therefore \quad (x,y) \in R^2 \tag{2.1}$$

onde o valor de I corresponde ao nível de cinza ou luminância da imagem na posição (x,y). Então, $I(x,y) \geq 0$ para todo (x,y). É assumido que a função imagem é diferenciável em todo ponto e em toda direção [39].

Para a determinação das bordas na imagem, é definida a derivada direcional pela seguinte equação:

$$\frac{\partial I(x,y)}{\partial \vec{e}} = \lim_{t \to 0} \frac{I(x + te1,\ y + te2) - I(x,y)}{t} \tag{2.2}$$

onde $\vec{e} = (e1,e2)$ é um vetor na direção do qual deseja-se encontrar a derivada. O ângulo entre este vetor e o eixo x é dado por φ. Então, pode-se mostrar que [39]:

$$\frac{\partial I(x,y)}{\partial \vec{e}} = \frac{\partial I(x,y)}{\partial x}\cos\varphi + \frac{\partial I(x,y)}{\partial y}\operatorname{sen}\varphi \tag{2.3}$$

onde $\frac{\partial I(x,y)}{\partial x}$ e $\frac{\partial I(x,y)}{\partial y}$ são derivadas parciais. Esta derivada tem o valor máximo (como uma função de φ) nas direções ortogonais à borda no ponto (x,y) e é zero na direção paralela à borda. A direção que produz o máximo é

$$\varphi_0 = \operatorname{arctg}\left[\frac{\partial I(x,y)}{\partial y} \Big/ \frac{\partial I(x,y)}{\partial x}\right] \qquad (2.4)$$

e o valor máximo correspondente, a magnitude do gradiente, é

$$GR(x,y) = \sqrt{\left[\frac{\partial I(x,y)}{\partial x}\right]^2 + \left[\frac{\partial I(x,y)}{\partial y}\right]^2} \qquad (2.5)$$

O vetor tendo direção φ_0 e comprimento GR é chamado vetor gradiente. Para que se decida a presença de uma borda, o gradiente deve ser comparado a algum limiar pré-estabelecido através de processos automáticos ou baseados em estatísticas da imagem (histograma, índice de homogeneidade) [40].

Os resultados acima consideram uma imagem ideal. Na prática, tem-se que a imagem é definida em uma grade ("grid") retangular com resolução finita. Esta resolução finita decorre do processo de amostragem e digitalização da imagem ideal (imagem contínua).

Adotando-se o modelo sugerido em [4], a imagem real é dada por

$$\hat{I}(i,j) \quad \therefore (i,j) \in Z^2, \qquad Z = (1,2,3,4, \ldots) \qquad (2.6)$$

$$0 \leqslant \hat{I} \leqslant \hat{I}_{máx}, \quad 1 \leqslant i \leqslant i_{máx} \quad e \quad 1 \leqslant j \leqslant j_{máx}$$

Os valores desta função podem ser obtidos pela integração de uma imagem ideal sobre um pequeno retângulo. Sejam Δx e Δy as dimensões do "grid" amostrado, então

$$\hat{I}(i,j) \approx \int_{(j-1)\Delta y}^{j\Delta y} \int_{(i-1)\Delta x}^{i\Delta x} I(x,y)\, dx dy. \qquad (2.7)$$

O sinal $\approx$ significa a aproximação do valor exato ao nível de cinza mais próximo permitido.

A idéia de diferenciação agora, leva ao cálculo de diferenças que podem apenas ser aproximações de derivadas originais. Uma conseqüência óbvia é a diferença na interpretação do processo de diferenciação. Na imagem real, as diferenças não contêm esta informação por causa da resolução finita. Em vez disso, elas dão uma idéia da magnitude da mudança de intensidade.

As diferenças correspondentes às derivadas parciais são dadas por

$$\frac{\Delta \hat{I}(i,j)}{\Delta x} \qquad e \qquad \frac{\Delta \hat{I}(i,j)}{\Delta y}$$

Analogamente, a magnitude do gradiente é agora definida por

$$\hat{GR}(i,j) = \sqrt{\left[\frac{\Delta \hat{I}(i,j)}{\Delta x}\right]^2 + \left[\frac{\Delta \hat{I}(i,j)}{\Delta y}\right]^2} \qquad (2.8)$$

Considerando-se a complexidade envolvida no cálculo de $\hat{GR}(i,j)$, faz-se, normalmente, necessário o uso de aproximações computacionalmente mais simples . Há dois métodos usados para esse fim:

$$\hat{GR1}(i,j) = \left|\frac{\Delta \hat{I}(i,j)}{\Delta x}\right| + \left|\frac{\Delta \hat{I}(i,j)}{\Delta y}\right| \qquad (2.9)$$

$$\hat{GR2}(i,j) = \max\left\{\left|\frac{\Delta \hat{I}(i,j)}{\Delta x}\right|, \left|\frac{\Delta \hat{I}(i,j)}{\Delta y}\right|\right\} \qquad (2.10)$$

onde $\hat{GR1} \geqslant \hat{GR} \geqslant \hat{GR2}$ e $\hat{GR1} = \hat{GR2}$ somente quando $\frac{\Delta \hat{I}(i,j)}{\Delta x}$ ou

$\frac{\Delta \hat{I}(i,j)}{\Delta y}$ é zero.

2.3 Algoritmos para Deteção de Bordas

Os algoritmos para deteção de borda podem ser do tipo local ou global. Os métodos globais trabalham no domínio da freqência, através de transformadas e, em princípio, usam informação de toda a imagem para estabelecer a presença de uma borda em algum ponto. Ao contrário, os métodos locais usam somente informação de uma vizinhança limitada do ponto de interesse. Os métodos locais podem ser ainda subdivididos nas categorias de métodos de realce/limiar ("enhancement/thresholding") [4] ou de adaptação de bordas. O último enfoque envolve a adaptação do modelo de uma borda ideal aos padrões presentes na imagem [5-7]. Os métodos de realce/limiar, por outro lado, detectam bordas usando operadores diferenciais ou direcionais [8]. Em ambos os casos, uma borda é admitida presente se a resposta do operador excede um limiar preespecificado.

O método de deteção de borda por realce/limiar é descrito na Figura 2.1 Neste método, o arranjo discreto da imagem $\hat{I}(i,j)$ é espacialmente processado por um conjunto de N operadores ou máscaras H(i,j) lineares para produzir um conjunto de funções gradientes

$$\hat{GR}(i,j) = \hat{I}(i,j) * H(i,j) \qquad (2.11)$$

onde * denota convolução espacial bidimensional. A Figura

2.1 ilustra um exemplo desta operação. Em seguida, a cada pixel, as funções gradientes são combinadas por um operador linear ou não linear OP(.) para criar um arranjo de bordas realçadas dado por

$$AR(i,j) = OP(\hat{GR}_k(i,j)) \quad 1 \leq k \leq N \qquad (2.12)$$

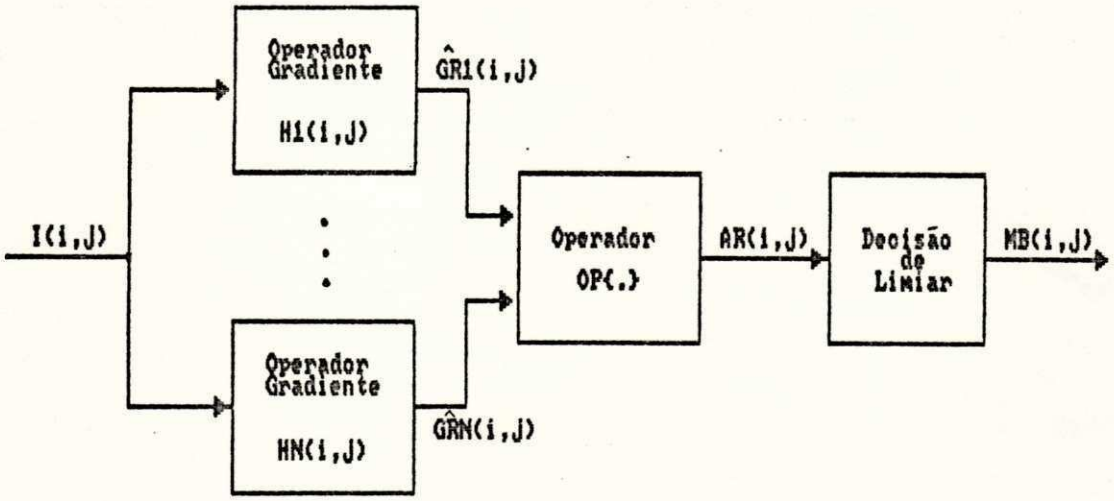


Figura 2.1: Método de deteção de bordas com realce/limiar.

O arranjo de bordas realçadas AR(i,j) fornece uma medida da descontinuidade da imagem no pixel analisado. Uma decisão da existência de borda é tomada com base na amplitude AR(i,j) com relação a um limiar L. Se

$$AR(i,j) > L$$

assume-se a presença de borda e se

$$AR(i,j) \leq L$$

não é indicada a presença de borda. A decisão de borda é normalmente arquivada em um mapa de bordas binário MB(i,j) onde um valor 1 indica uma borda e um valor 0 indica ausência de borda.

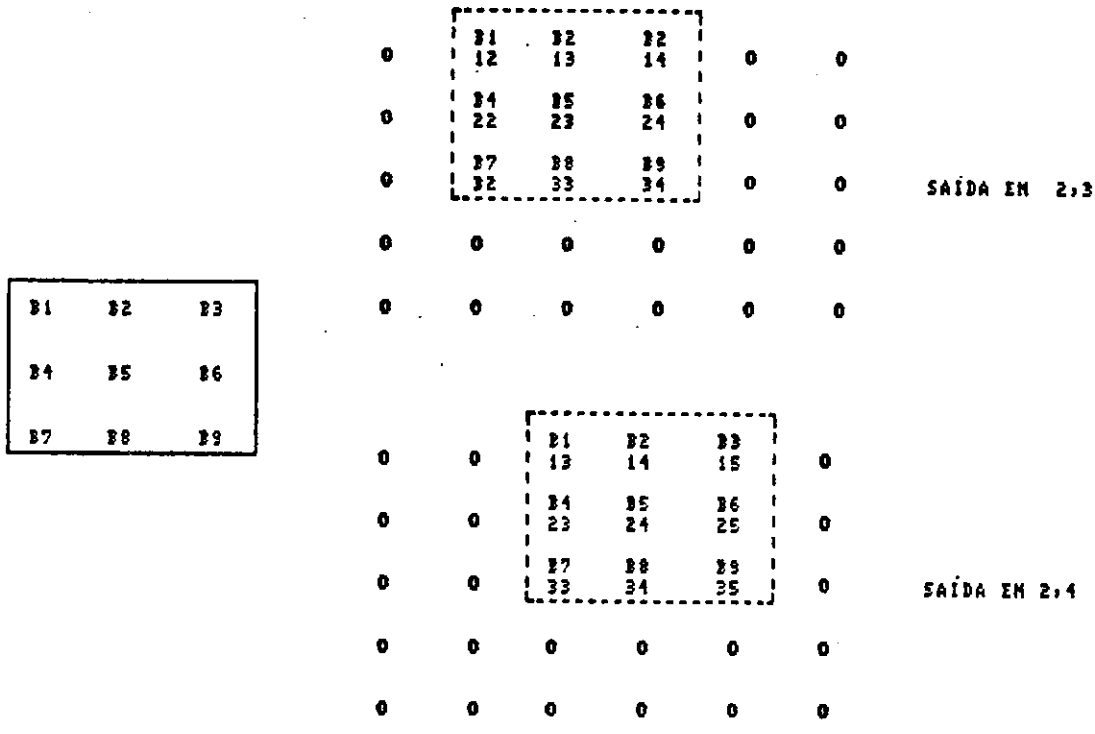


Figura 2.2: Exemplo de convolução de máscara sobre uma imagem.

Há dois tipos de operadores espaciais do tipo realce/limiar: operadores direcionais e operadores diferenciais.

a) Os operadores direcionais:

Os operadores direcionais são um conjunto de máscaras representando aproximações discretas para bordas ideais de várias orientações. Alguns algoritmos comumente usados são o operador de Prewitt [41, 42], o operador de Kirsh [43] e as máscaras de 3 e 5 níveis apresentadas por Robinson [44]. Os dois últimos operadores estão relacionados com os operadores de Prewitt [41] e de Sobel [45], respectivamente. Com estes operadores, o arranjo realçado é obtido como o máximo dos arranjos gradientes. Assim,

$$AR(i,j) = \text{máx}\ (|\hat{GR}_k\ (i,j)|) \qquad (2.13)$$

Em [46], Overington e Greenway apresentam um método que aumenta a precisão na determinação dessas direções reduzindo os níveis de quantização dos ângulos calculados.

b) operadores diferenciais:

Os operadores diferenciais realizam a diferenciação discreta de um arranjo de imagem para produzir um campo gradiente. Este grupo de operadores inclui os operadores Prewitt [41, 47], Sobel [45], Laplace [31] e Roberts [48].

Os operadores de Laplace não apresentam respostas polarizadas em direções ideais, sendo utilizados nos casos em que não há a preocupação com a orientação da imagem. Os operadores de Sobel e Prewitt são máscaras de pixel 3X3. Estas máscaras aproximam as derivadas parciais nas direções ortogonais x e y.

$$H_y = \begin{matrix} 1 & 0 & -1 \\ c & 0 & -c \\ 1 & 0 & -1 \end{matrix} \qquad H_x = \begin{matrix} 1 & -c & -1 \\ 0 & 0 & 0 \\ 1 & c & 1 \end{matrix} \qquad (2.14)$$

Com o operador de Prewitt c = 1 e com o operador de Sobel c = 2. Combinações lineares e não lineares das saídas das duas máscaras ortogonais geram o gradiente da imagem em cada ponto. A direção da borda pode ser obtida a partir de uma operação tangente inversa das saídas das duas máscaras.

Operadores locais com diferentes critérios para deteção de bordas foram estudados em [49-53].

Os métodos de adaptação de bordas consistem na adaptação de um modelo de borda ideal, uma função rampa ou um determinado padrão de impulso bidimensional, em algumas regiões da imagem. Uma borda é admitida presente se a adaptação for suficientemente boa. Em [5, 54], é apresentado um operador local que aproxima modelos com forma de disco. Outros métodos são sugeridos em [55-58] e se baseiam em um modelo de facetas para a imagem. Nevatia [59] e Abramatic [60] apresentam simplificações para o operador Hueckel. Shipman et alii [61] sugerem um algoritmo adaptativo para deteção de bordas difusas.

Alguns algoritmos para deteção de borda incorporam propriedades texturais da imagem, tais como orientação, grossura ("coarseness") e finura ("finess") da textura. [62-68].

.O algoritmo apresentado em [69] considera a resposta do sistema visual humano, utilizando um limiar variante no espaço e dependente de uma função logarítmica do

comportamento dos níveis de cinza da vizinhança do ponto analisado. Os princípios fisiológicos e psicofísicos do sistema visual humano podem também ser utilizados para eliminar bordas espúrias, geradas pela maioria dos detetores convencionais [70-72].

Alguns pesquisadores têm empregado modelos aleatórios para representar um pixel e classificá-lo como sendo de uma região de fronteira ou não. Em [73], é proposto o uso de modelos marcovianos, enquanto que em [74, 75] usam-se modelos autoregressivos causais e derivadas de segunda ordem para gerar campos de contorno. Outros métodos neste sentido são apresentados em [76, 83].

2.4 O Operador de Roberts

Roberts [31, 48] apresentou um método simples de diferenciação bidimensional baseado nas diferenças cruzadas em uma janela 2x2. O valor de cada pixel da imagem com bordas realçadas é definido como

$$AR(i,j) = \sqrt{([I(i,j)-I(i+1,j+1)]^2 + [I(i,j+1)-I(i+1,j)]^2)} \quad (2.15)$$

Outras aproximações são:

$$AR(i,j) = |I(i,j)-I(i+1,j+1)| + |I(i,j+1)-I(i+1,j)| \quad (2.16)$$

e

$$AR(i,j) = \max [|I(i,j)-I(i+1,j+1)|, |I(i,j+1)-I(i+1,j)|] \quad (2.17)$$

O operador de Roberts pode ser representado pelas seguintes máscaras [48]:

$$H1 = \begin{matrix} 1 & 0 \\ 0 & -1 \end{matrix} \qquad .H2 = \begin{matrix} 0 & 1 \\ -1 & 0 \end{matrix} \qquad (2.18)$$

O resultado da operação deve ser associado ao pixel que ocupa a posição superior esquerda da janela. As bordas detectadas apresentam larguras que variam de um a dois pixels. A obtenção de borda com largura de um pixel é uma característica ideal de um método de deteção de bordas [44, 84-87]. Esta propriedade não existe na maioria dos operadores diferenciais ou direcionais e, no caso do operador de Roberts, é função das dimensões das máscaras utilizadas (janelas 2 X 2). As informações de borda estreitas aumentam a precisão nos processos posteriores de aplicação de limiar, definição de contornos, análise visual ou reconhecimento de padrões.

O algoritmo do operador de Roberts apresenta boa sensibilidade às componentes de alta freqüência e relativa imunidade ao ruído [4]. A sua simplicidade implica em facilidades de projeto e implementação [88-90]. Estudos comparativos usando implementações em linguagem C, mostraram o bom desempenho deste algoritmo, em relação a outros mais complexos, em tarefas de geração de imagens gradientes [24].

Na Figura 2.3, tem-se uma imagem com bordas em várias direções. As Figuras 2.4 e 2.5 apresentam, respectivamente, os mapas de pixels desta imagem exemplo

antes e depois do processamento com o operador de Roberts.

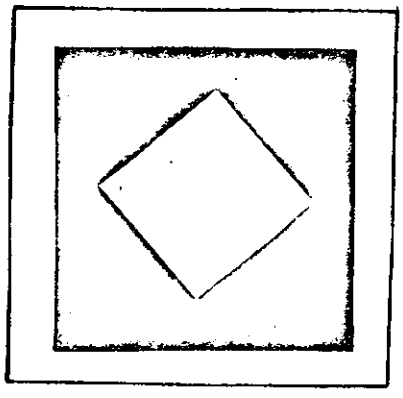


Figura 2.3: Imagem exemplo.

FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF 00 00 00 00 00 00 00 00 00 00 00 00 FF FF
FF FF 00 00 00 00 00 00 00 00 00 00 00 00 FF FF
FF FF 00 00 00 00 00 FF FF 00 00 00 00 00 FF FF
FF FF 00 00 00 00 FF FF FF FF 00 00 00 00 FF FF
FF FF 00 00 00 FF FF FF FF FF FF 00 00 00 FF FF
FF FF 00 00 FF FF FF FF FF FF FF FF 00 00 FF FF
FF FF 00 FF FF FF FF FF FF FF FF FF FF 00 00 FF FF
FF FF 00 FF FF FF FF FF FF FF FF FF 00 00 00 FF FF
FF FF 00 00 FF FF FF FF FF FF 00 00 00 00 FF FF
FF FF 00 00 00 FF FF FF FF 00 00 00 00 00 FF FF
FF FF 00 00 00 00 FF FF 00 00 00 00 00 00 FF FF
FF FF 00 00 00 00 00 00 00 00 00 00 00 00 FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

Figura 2.4: Mapa de pixels da imagem.

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 FF FF FF FF FF FF FF FF FF FF FF FF FF 00 00
00 FF 00 00 00 00 00 00 00 00 00 00 00 FF 00 00
00 FF 00 00 00 00 FF FF FF 00 00 00 00 FF 00 00
00 FF 00 00 00 FF FF 00 FF FF 00 00 00 FF 00 00
00 FF 00 00 FF FF 00 00 00 FF FF 00 00 FF 00 00
00 FF 00 FF FF 00 00 00 00 00 FF FF 00 FF 00 00
00 FF FF FF 00 00 00 00 00 00 00 FF 00 FF 00 00
00 FF FF 00 00 00 00 00 00 00 FF FF 00 FF 00 00
00 FF FF FF 00 00 00 00 00 FF FF 00 00 FF 00 00
00 FF 00 FF FF 00 00 00 FF FF 00 00 00 FF 00 00
00 FF 00 00 FF FF 00 FF FF 00 00 00 00 FF 00 00
00 FF 00 00 00 FF FF FF 00 00 00 00 00 FF 00 00
00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

Figura 2.5: Imagem gradiente gerada com o operador de Roberts.

3 ARQUITETURA DO CIRCUITO OPERADOR DE ROBERTS

Este capítulo apresenta a arquitetura proposta para a implementação do "ASIC" operador de Roberts. São apresentados um diagrama de tempo detalhando o funcionamento do circuito e as considerações de sistema feitas para o uso deste circuito.

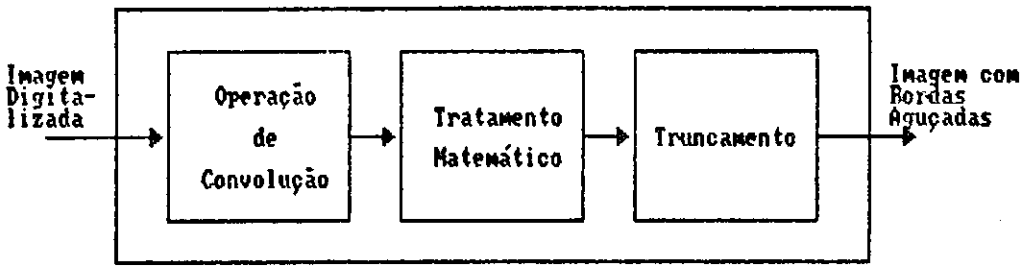
3.1 Apresentação da Arquitetura

Para a implementação em "hardware" do operador de Roberts, é proposta a arquitetura da Figura 3.1. Esta arquitetura realiza a operação representada pela equação 2.16 (Cap. 2). Os cálculos realizados para os pixels na última coluna ou na última linha devem ser desconsiderados, uma vez que não é possível completar a janela corretamente. Nesta arquitetura, não é imposta nenhuma restrição quanto ao tamanho da imagem, o que permite obter mesmo desempenho tanto em imagens pequenas quanto em imagens de maior resolução.

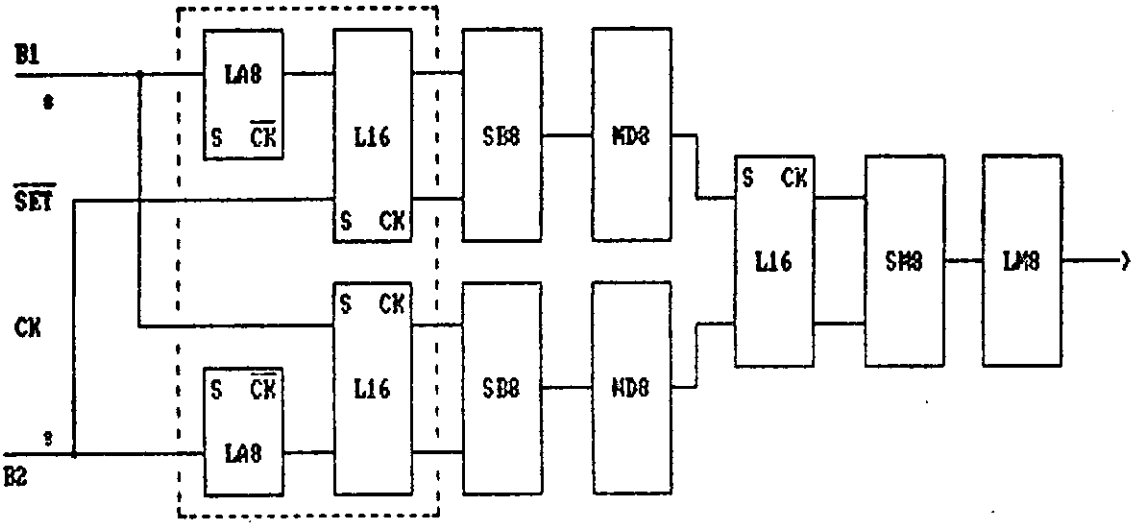
O circuito está dividido em três partes principais. O primeiro bloco é responsável pela alocação dos dados nas suas posições dentro da máscara. O segundo bloco realiza as operações matemáticas envolvidas na diferenciação cruzada definida por Roberts [48, 90]. Por último, há o bloco que

limita o resultado a um máximo de 255 (FF_H).

Os sinais $\overline{SET}$ e CLOCK (Fig. 3.1.b) são responsáveis pela habilitação e pelo sincronismo do circuito, respectivamente.



(a)



(b)

Figura 3.1: Arquitetura proposta para a implementação do operador de Roberts:
a) Diagrama em blocos.
b) Detalhamento da arquitetura.

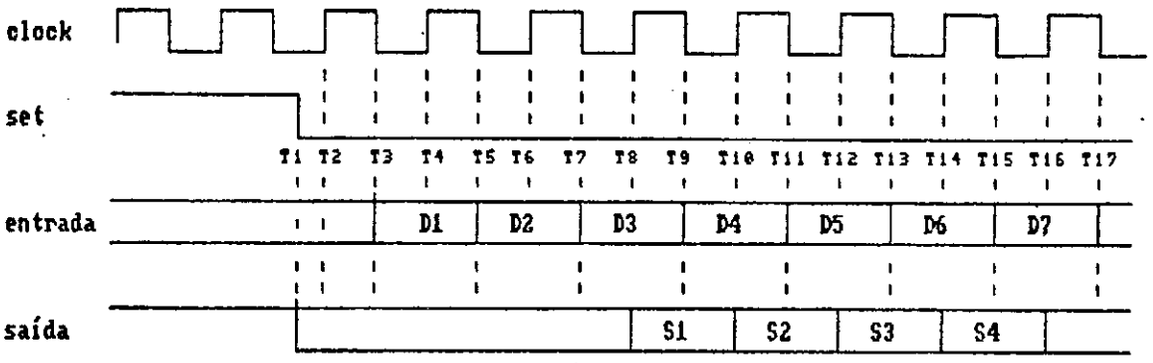
3.2 Descrição do Funcionamento

Na arquitetura proposta (Fig. 3.1), os barramentos B1 e B2 devem receber os valores dos níveis de cinza dos pixels da i-ésima e da i+1-ésima linhas, respectivamente.

O posicionamento dos dados de entrada nestes barramentos deve ocorrer a cada transição para baixo do sinal CLOCK, após a habilitação do circuito. O valor a ser associado ao primeiro pixel processado deve ser recuperado na quarta transição para baixo do sinal CLOCK, enquanto que os demais valores devem ser recuperados a cada transição para baixo subseqüente deste sinal. Um diagrama de tempo ilustrando o funcionamento é apresentado na Figura 3.2.

3.3 Detalhamento da Arquitetura Proposta

A Figura 3.1.b detalha a arquitetura proposta para o operador de Roberts. Os dados, introduzidos dois a dois, nos barramentos B1 e B2 correspondem à codificação em 8 bits dos níveis de cinza dos pixels da imagem a ser processada. Este número de bits permite a codificação de níveis de cinza entre 0 (00_H) e 255 (FF_H), que é a faixa usual [40]. A estrutura de "latches" no interior do retângulo pontilhado realiza a operação de convolução da máscara sobre a imagem. Na máscara, a cada descida do sinal CLOCK, são descartados os pixels sob as posições 11 e 21, os pixels que ocupavam as posições 12 e 22 são transferidos para as posições 11 e 21, respectivamente, e são recebidos novos pixels para as posições 12 e 22. Esta operação é ilustrada na Figura 3.3.




```
EI = ESTADO INTERNO
SA = SAIDA DE BLOCO
SUP = SUPERIOR
SUP = INFERIOR

T1 - HABILITAÇÃO DO CIRCUITO

T4 - I11 = EI(L16INF)
     I21 = EI(L16SUP)

T4 - I11 = EI(LA8SUP)
     I21 = EI(LA8INF)

T5 - I11 = SA(LA8SUP)= EI(L16SUP)
     I21 = SA(LA8INF)= EI(L16INF)
     I12 = EI(L16INF)
     I22 = EI(L16SUP)

T6 - I11 E I22 = SA(L16SUP)
     I12 E I21 = SA(L16INF)
     I12 = EI(LA8INF)
     I22 = EI(LA8SUP)
     INÍCIO PRIMEIRO CÁLCULO SUBTRAÇÃO

T7 - I12 = SA(LA8SUP) = EI(L16SUP)
     I22 = SA(LA8INF) = EI(L16INF)
     I13 = EI(L16INF)
     I23 = EI(L16SUP)
     FIM PRIMEIRO CÁLCULO MÓDULO
     PRIMEIRO CÁLCULO MÓDULO = EI(L16)

T8 - I12 E I23 = SA(L16SUP)
     I13 E I22 = SA(L16INF)
     I13   EI(LA8INF)
     I23   EI(LA8SUP)
     INICIO SEGUNDO CÁLCULO SUBTRAÇÃO
     PRIMEIRO CÁLCULO MÓDULO = SA(L16)
     INÍCIO PRIMEIRA SOMA

T9 - I13 = SA(LA8SUP) = EI(L16SUP)
     I23 = SA(LA8INF) = EI(L16INF)
     I14 = EI(L16INF)
     I24 = EI(L16SUP)
     FIM SEGUNDO CÁLCULO MÓDULO
     SEGUNDO CÁLCULO MÓDULO = EI(L16)
     PRIMEIRO RESULTADO - I11 PROCESSADO

T11- SEGUNDO RESULTADO  - I12 PROCESSADO

T13- TERCEIRO RESULTADO - I13 PROCESSADO
```

Figura 3.2: Diagrama de tempo ilustrando o funcionamento do circuito operador de Roberts.

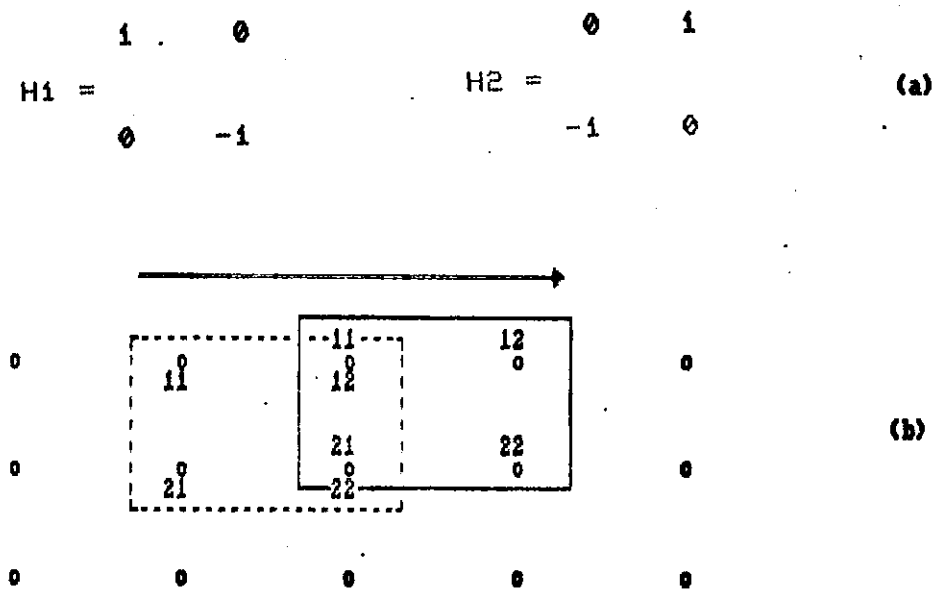


Figura 3.3: Operador de Roberts:
a) Máscaras.
b) Exemplo de convolução das máscaras.

Os "latches" de 16 bits recebem os novos dados e posicionam os pixels das diagonais para a operação de subtração. A estrutura utilizada para a formação dos "latches" é mostrada na Figura 3.4.

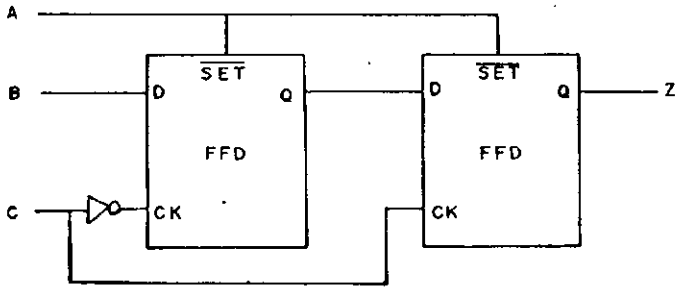


Figura 3.4: Estrutura do "latch" elementar.

Os blocos SB8 (Fig. 3.5) são subtratores de 8 bits

que realizam a deleção da diferença entre os níveis de cinza dos pixels nas diagonais da janela atual. Ou seja, os blocos realizam as operações

$$\hat{I}_{i,j} - \hat{I}_{i+1,j+1} \quad e \qquad (3.1)$$

$$\hat{I}_{i,j+1} - \hat{I}_{i+1,j}. \qquad (3.2)$$

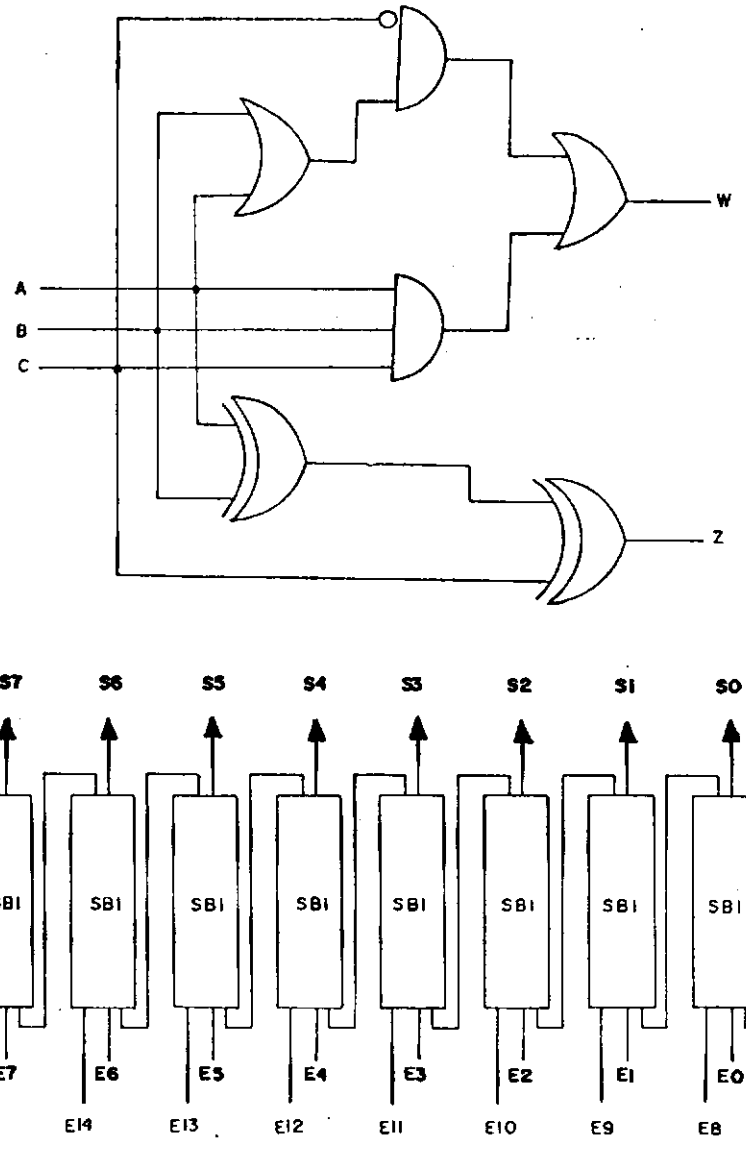


Figura 3.5: Bloco subtrator de 8 bits (S88):
a) Detalhe da implementação do subtrator elementar.
b) Diagrama completo.

Como o resultado de uma operação de subtração pode ser negativo e a operação de Roberts considera apenas o seu valor absoluto, faz-se necessária a introdução do bloco MD8 que define o módulo em complemento de dois da saída do bloco SB8. Diagramas ilustrando a sua formação são fornecidos na Figura 3.6.

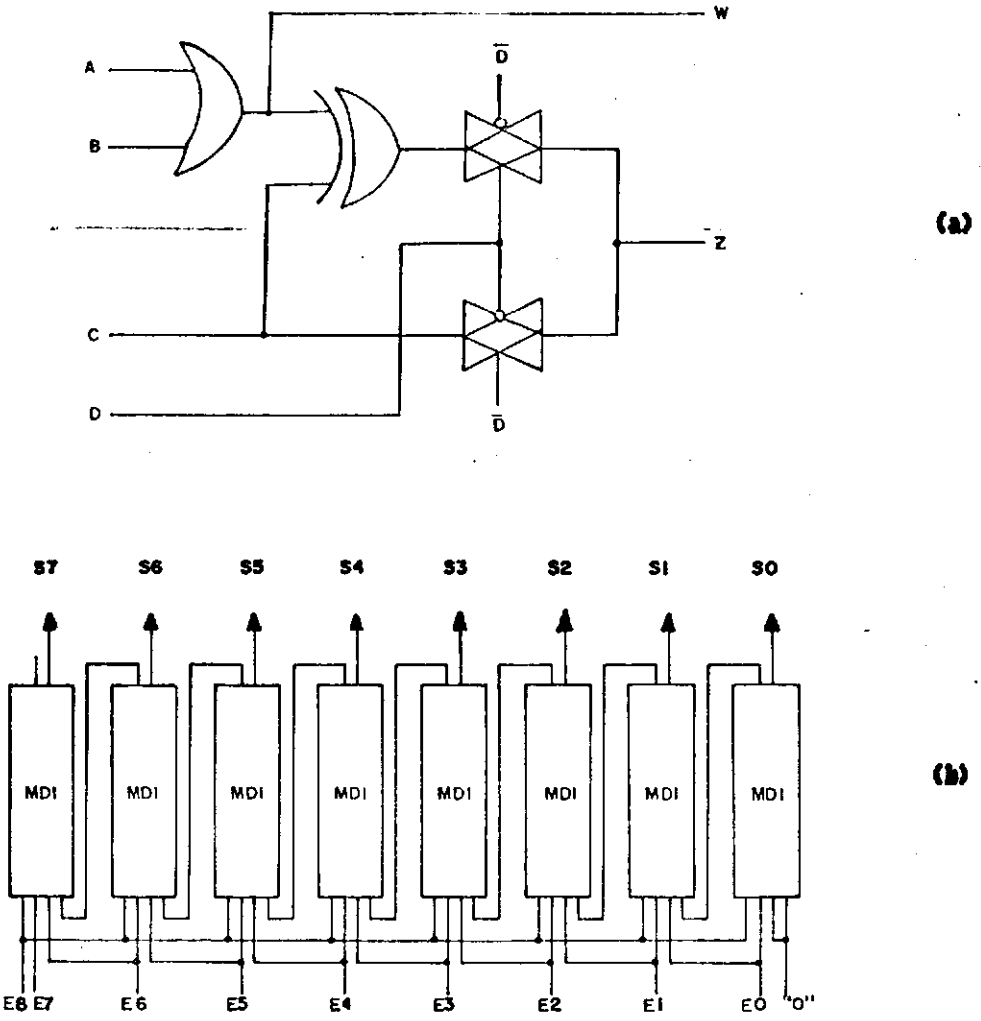


Figura 3.6: Bloco módulo de 8 bits (MD8):
a) Detalhe do operador módulo elementar.
b) Diagrama completo.

O bloco SM8 faz a aproximação da operação de Roberts pela soma, isto é, ele fornece

$$|\hat{I}_{i,j} - \hat{I}_{i+1,j+1}| + |\hat{I}_{i,j+1} - \hat{I}_{i+1,j}| \quad (3.3)$$

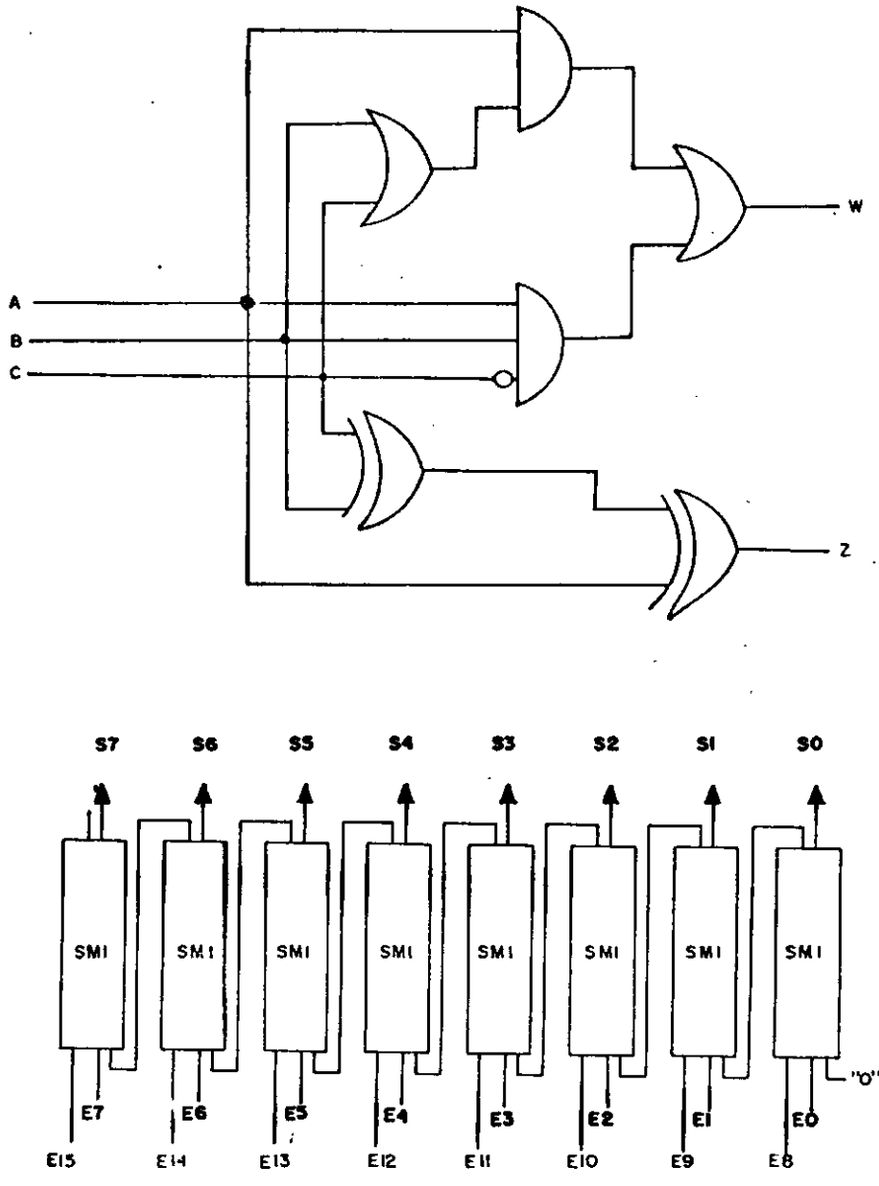


Figura 3.7: Bloco somador de 8 bits (SM8)
a) Detalhe da implementação do somador elementar.
b) Diagrama completo.

que é o valor de nível de cinza a ser atribuído ao pixel processado. O bloco SM8 é um somador de 8 bits formado a partir de somadores elementares conforme mostrado na Figura 3.7.

Uma vez que o valor dado na expressão 3.3 pode exceder 255 (FF) que é o maior número com representação binária em 8 posições, surge a necessidade do bloco LM8 para fazer o truncamento. Este bloco é formado por uma combinação de portas lógicas OR, como indicado na Figura 3.8.

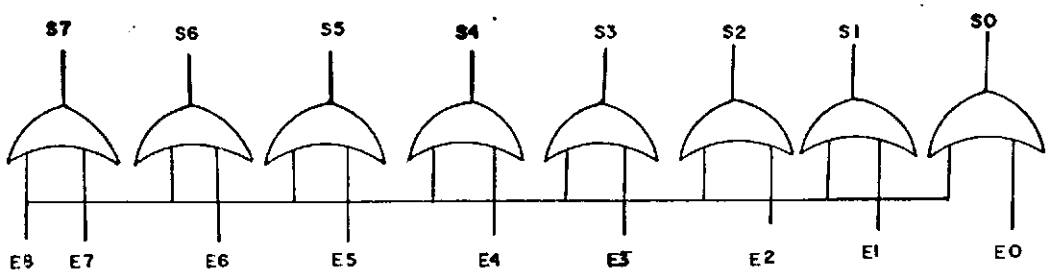


Figura 3.8: Bloco de truncamento (LM8).

3.4 Considerações sobre o Sistema

A arquitetura proposta para o "chip" considera sua utilização no sistema mostrado na Figura 3.9. A memória M1 contém a imagem a ser processada, enquanto que a memória M2 deve armazenar a imagem resultante da detecção de bordas. Fisicamente, estas memórias M1 e M2 podem ser uma só, permitindo operação "in-place". O bloco CONTROLADOR tem a função de sincronizar a operação do conjunto. Este bloco deve fornecer os sinais SET e CLOCK, além de gerar os

endereços e os sinais C1 e C2 que são utilizados para as operações de leitura e escrita nas memórias.

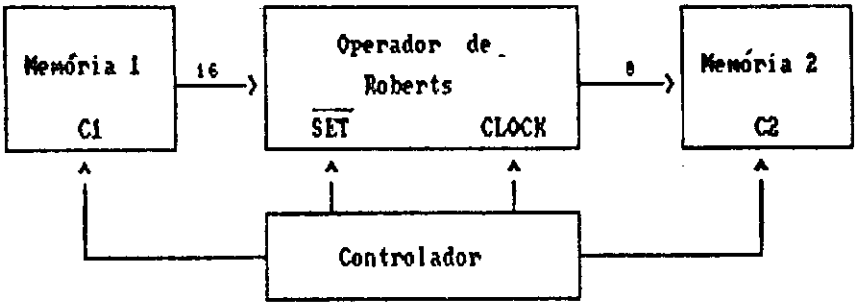


Figura 3.9: O sistema operador de Roberts.

4 IMPLEMENTAÇÃO

Neste capítulo, é dada uma visão geral dos aspectos envolvidos no projeto de um circuito integrado, apresenta-se a metodologia utilizada no projeto do operador de Roberts e são fornecidos os resultados obtidos, incluindo uma comparação dos sistemas ALLIANCE e SOLO 1400.

4.1 A Implementação de um Circuito Integrado

As etapas de fabricação de um circuito integrado podem ser representadas, de uma maneira geral, pelo fluxograma da Figura 4.1.

Numa primeira etapa, é feita a especificação das funções e condições de operação do circuito a nível do sistema. Numa segunda etapa, é gerado um circuito conceitual que é, então, simulado para verificar se realiza as funções especificadas anteriormente. Uma vez validado através de simulação ou protolipagem, é feito o desenho das máscaras.

Depois de testadas e validadas as máscaras, procede-se a fabricação das pastilhas, usando-se o processo escolhido quando da especificação do sistema. Após ser feita a difusão, as pastilhas resultantes são montadas e, então, são realizados testes para verificar a conformidade com as especificações. Estes testes incluem testes funcionais e de

confiabilidade, de modo que o resultado seja, não somente um circuito que opere funcionalmente de acordo com as especificações, mas que as atenda, também, sob o ponto de vista da qualidade e da confiabilidade. São as informações desses testes que permitem a correção de erros de concepção e o aprimoramento das técnicas de simulação e prototipagem usadas na validação do projeto.

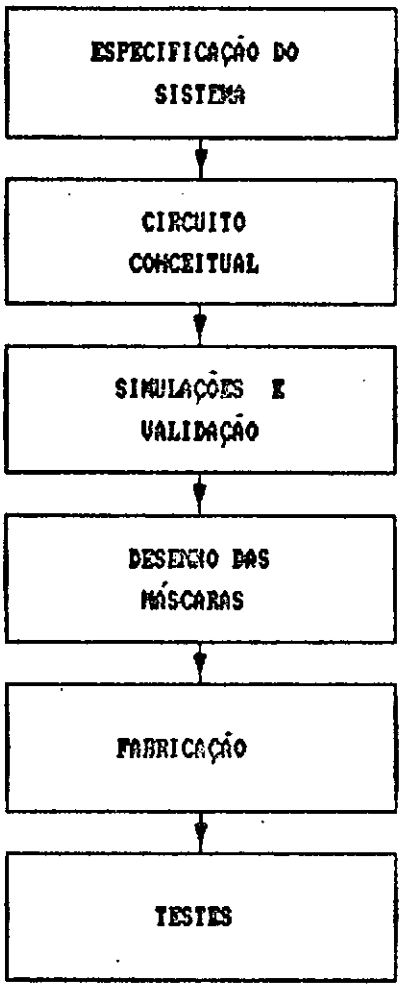


Figura 4.1: Fluxograma para projeto de um circuito integrado.

Uma metodologia comumente usada no projeto de "ASICs" é a divisão hierárquica [25]. Esta metodologia permite a realização de projetos complexos em um curto intervalo de tempo. Cada bloco é dividido de tal modo que as várias partes que o compõem possam ser projetadas em paralelo. Os modelos podem ser estabelecidos para os vários níveis hierárquicos do sistema completo. O primeiro nível em que se pode estabelecer um modelo é o nível do próprio sistema. Em seguida, podem ser estabelecidos modelos para os níveis de hierarquia mais baixa, como os vários níveis de subsistema (grandes blocos funcionais, registros, portas lógicas, etc.), o nível de estrutura dos dispositivos eletrônicos e o nível de processos.

Os blocos são normalmente denominados células. A célula de um sistema é um circuito que desempenha uma função cuja complexidade depende do nível de hierarquia em que se está trabalhando. Assim, uma célula pode ser constituída pelo circuito de uma simples porta NAND, ou por uma complexa memória ROM que contém um microprograma.

A criação das células normalmente se faz a partir de células simples, as quais vão sendo agregadas de modo a constituir células mais complexas. Muitas vezes são dados nomes específicos para as células de acordo com o nível hierárquico em que são definidas como, por exemplo, macro-células, bloco de células, hiper-células, etc. [91].

As células que constituem os sistemas podem ser de dois tipos principais: células dedicadas e células padronizadas.

Entende-se por célula dedicada, um circuito destinado

a atender uma especificação particular ligada a uma aplicação particular. A célula dedicada tem suas funções mais simplesmente implementadas de modo a otimizar o desempenho e a ocupação da área do circuito, sendo utilizada apenas na aplicação específica para que foi criada.

A célula padronizada, por sua vez, é concebida para ser usada como bloco de construção de um número muito grande de sistemas diferentes, o que implica geralmente numa diminuição do custo e do tempo de projeto. Essas células padronizadas são otimizadas para a execução de uma dada função e são validadas individualmente.

Do ponto de vista funcional, distinguem-se as células como sendo células internas (para tratamento de sinais) ou células de entrada e saída ("pads"). Essa distinção é feita porque os critérios usados em sua concepção são distintos. No caso das células internas, interessa otimizar seu desempenho até um limite determinado pelo processo de fabricação, procurando-se limitar seu consumo de energia e suas dimensões, aumentando sua velocidade, etc. As células de entrada e saída, entretanto, não estão limitadas somente pelo processo de fabricação, mas seu projeto deve levar em consideração também os circuitos externos aos quais estarão ligadas.

Para as matrizes de portas ("gate arrays"), as células internas padronizadas são todas iguais ou são constituídas por blocos de células elementares com um número limitado de células diferentes. É o caso, por exemplo, de células destinadas à realização de circuitos lógicos e das células de memória, as quais serão organizadas pelo projetista de acordo com suas necessidades. As células de matrizes de portas devem ser otimizadas para apresentar o

melhor desempenho possível, de modo a não limitar demais as aplicações das matrizes finais. A otimização deve levar em conta principalmente a área ocupada pela célula, sua conectividade e seu desempenho dinâmico. Embora possam ser otimizadas, as células das matrizes de portas sempre tendem a ser pouco econômicas em área e os circuitos construídos com elas têm, em geral, desempenho pior que os totalmente dedicados.

Para os circuitos com célula padrão ("standard cell"), as células internas padronizadas surgem como uma forma de melhorar o desempenho e a ocupação de área dos circuitos, ao mesmo tempo em que se mantém uma razoável flexibilidade e rapidez de projeto. Estas células são projetadas, em geral, com mesma altura para facilitar operações como roteamento e existem na forma de uma biblioteca de células dedicadas a funções específicas e de uso geral, como é o caso das famílias lógicas e dos módulos analógicos [92, 93]. A célula padrão foi escolhida para a implementação do "ASIC" proposto neste trabalho.

As especificações do sistema colocado na pastilha devem levar em consideração vários fatores:

- A divisão do subsistema em partes ou componentes de hierarquia inferior, denominados células. Deverão ser definidos tantos níveis hierárquicos quanto necessários para a elaboração conveniente do sistema.

- O loteamento da superfície da pastilha, que consiste em se dividir a pastilha de modo a fazer caberem os módulos que constituirão o sistema com a menor área.

- A definição das portas de entrada e saída, que

são responsáveis pela comunicação do "chip" com o mundo exterior, que consiste no estabelecimento dos sinais que se comunicarão com os circuitos externos à pastilha e suas características elétricas, tipo "fan-in", "fan-out", etc.

- A avaliação do tamanho da pastilha que deve ser a menor possível.

- A metodologia de projeto, que é função de fatores como complexidade do sistema e disponibilidade de ferramentas de auxílio ao projeto. Um circuito muito complexo dificilmente poderá ser construído com matrizes de portas ("gate arrays"), pois o número de portas contido nessas matrizes é limitado, a eficiência de interconexão é muito baixa e as alternativas de escolha de configurações de circuitos são muito limitadas, o que compromete a otimização do desempenho do circuito. A metodologia de célula padrão, por outro lado, está associada à existência de ferramentas que realizam a alocação e interconexão das células no "layout" final [94-98].

4.2 O Projeto do Circuito Operador de Roberts

Para o ASIC proposto para implementar o operador de Roberts, foram utilizadas as metodologias de divisão hierárquica e de célula padrão. O nível 0 (nível mais alto) consistiu-se do circuito completo mostrado na Figura 3.1.b.

Cada bloco (projetado individualmente) foi dividido em sub-blocos, gerando o nível 1 (blocos LA8,SB8,MD8,SM8 e LM8). Os blocos SB1, MD1 e SM1 foram ainda divididos para

formar o nível 2. A decomposição hierárquica utilizada é mostrada no Apêndice A.

A validação do circuito foi realizada em duas etapas: validação descendente ("top-down") e validação ascendente ("bottom-up"), obedecendo à estrutura hierárquica já definida.

O processo de 2 micras utilizado para a implementação, ilustrado na Figura 4.2, consiste de 10 níveis de máscaras:

1. N well
2. difusão ativa
3. polissilício
4. difusão N+
5. difusão P+
6. contato
7. metal1
8. via
9. metal2
10. passivação

OIS - ÓXIDO DE ISOLAÇÃO
OC - ÓXIDO DE CAMPO
OIM - ÓXIDO INTER-METAL
OG - ÓXIDO DE GATE
PS - POLISILICIO

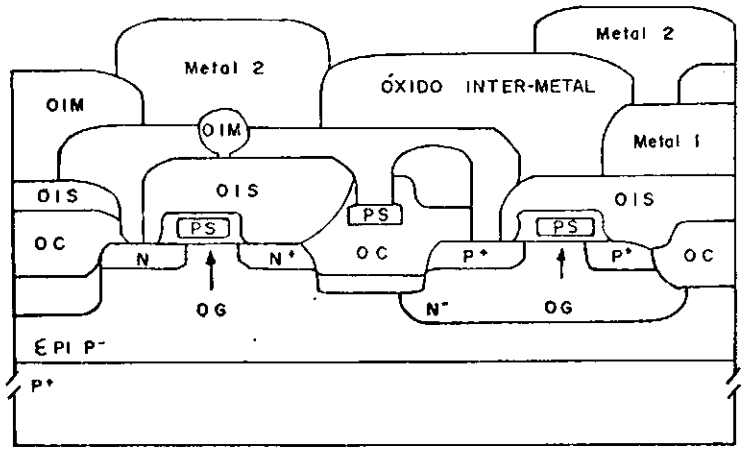


Figura 4.2: Camadas do processo CMOS

4.3 Validação Descendente

A validação descendente diz respeito à validação funcional do circuito e foi obtida através dos softwares ALLIANCE [26] e ELOISE [99], com a definição do comportamento dos blocos funcionais e simulações lógicas para os níveis 0, 1, e 2.

4.3.1 O Simulador ELOISE

O ELOISE é um simulador lógico desenvolvido no Laboratório MASI da Universidade de Paris 6 [99], construído para fornecer resultados em um tempo razoável apenas para circuitos com poucos elementos (aproximadamente 40), que podem ser indistintamente transistores ou blocos com outra função específica mais complexa. Sendo assim, as simulações para o "ASIC" proposto tiveram que ser realizadas em vários níveis hierárquicos.

4.3.2 O Sistema de Desenvolvimento ALLIANCE

O sistema de desenvolvimento de projetos ALLIANCE [26], desenvolvido na Universidade de Paris 6, inclui compilador de células, roteador, simulador lógico ELOISE, verificador de regras de interconexão, além do utilitário para simulações elétricas EPICE.

A visualização e edição das máscaras exigem uma

representação gráfica do circuito real. No modo PHYSIQUE, o editor do ALLIANCE gera a localização das máscaras e evidencia os diferentes níveis de tecnologia. Na validação de um circuito integrado, entretanto, os aspectos de "layout" (aspectos físicos) são de pouca importância, de modo que apenas as informações de interconexões são necessárias à sua descrição. O modo de funcionamento do ALLIANCE para esta abordagem é chamado LOGIQUE. A interação do projetista com o sistema ALLIANCE é ilustrada na Figura 4.3.

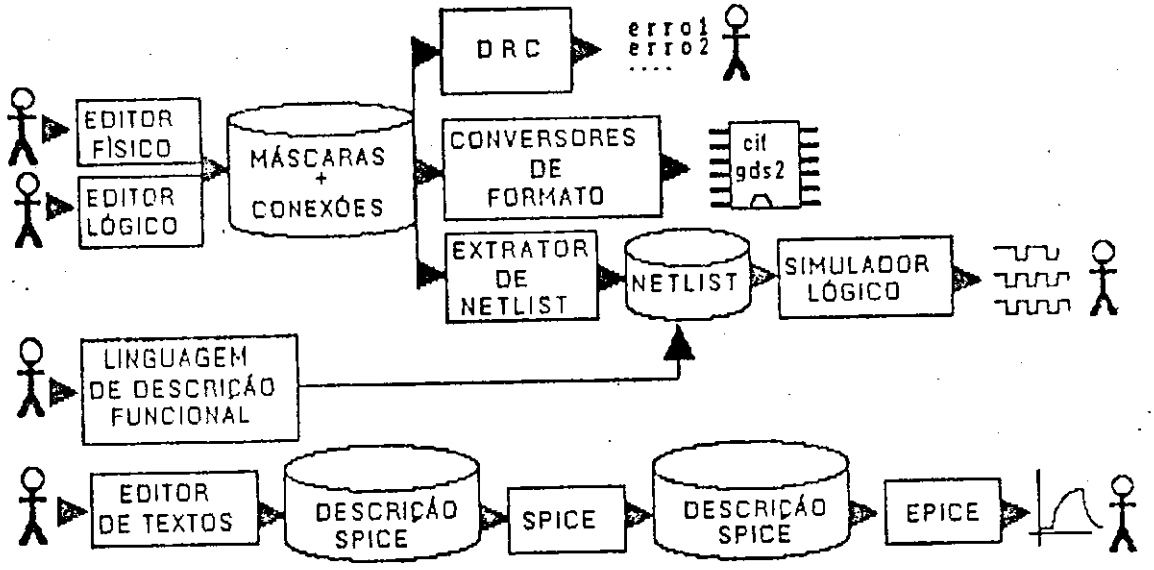


Figura 4.3: O sistema ALLIANCE.

4.3.3 Metodologia de Simulação Lógica

1 - Divisão do Sistema em Macro-blocos

De acordo com a hierarquia definida, faz-se a divisão dos blocos em sub-blocos cujas saídas são descritas como funções de suas entradas (descrição do comportamento). No

Apêndice B, são apresentadas as rotinas em linguagem C utilizadas.

2 - Realização das Simulações

Com o "software" ALLIANCE, são gerados arquivos correspondentes aos sub-blocos consistindo apenas de retângulos que representam as entradas/saídas de interesse (modo PHYSIQUE). Para que as simulações sejam realizadas, cria-se um simulador ELOISE apropriado, a partir das rotinas e dos arquivos obtidos com o "software" ALLIANCE. Cada circuito a ser simulado é gerado com a interconexão dos sub-blocos correspondentes, através do ALLIANCE (modo LOGIQUE). A apresentação dos resultados pode ser feita em forma de tabelas da verdade ou diagramas de tempo.

4.4 Validação Ascendente

A validação ascendente diz respeito à validação funcional e elétrica do circuito, levando-se em consideração aspectos como tecnologia, loteamento, roteamento, atrasos, etc.

O loteamento de uma pastilha é feito com o objetivo de se determinar qual a melhor distribuição das células e dos dutos de sinais, de modo a se otimizar a ocupação de área e melhorar o desempenho do circuito. Esta tarefa envolve os seguintes aspectos principais:

- disponibilidade de uma especificação detalhada e precisa de como se deve decompor o sistema em subsistemas e

a disponibilidade de uma descrição de suas interconexões.

- estimativa da forma e tamanho desses blocos com base na tecnologia a ser usada e em sua composição.

- alocação e interconexão desses blocos até que se obtenha uma distribuição eficiente dos mesmos na área da pastilha.

A alocação das células numa pastilha corresponde à montagem relativa de modo a formar o circuito completo do sistema da maneira mais eficiente possível, tanto do ponto de vista da ocupação da área, como do ponto de vista do desempenho funcional e paramétrico.

O roteamento consiste das fitas de metais, difusão ou polissilício que formam as interconexões das células já alocadas. O roteamento se faz em duas fases. Na primeira, definem-se os canais por onde passarão os sinais; na segunda, traçam-se as rotas dos sinais.

Todo este processo de geração, gerenciamento, alocação e interconexão das células foi realizado automaticamente com a ferramenta de "CAD" SOLO 1400.

4.4.1 O Sistema SOLO 1400

O SOLO 1400 [27] é uma ferramenta de "CAD" para projeto de circuitos integrados capaz de gerar, alocar e rotear células a partir de captura esquemática, que consiste na retirada de informações de interconexões segundo

uma edição gráfica. Todas as células são implementadas com tecnologia CMOS N well de 2 micras e dois níveis de metal desenvolvida pela "Foundry" ES2 na França. Esta tecnologia permite oferecer uma litografia de escrita sobre o "wafer" em paralelo com a tradicional litografia ótica de máscara.

A complexidade de um projeto no sistema SOLO 1400 é representada por unidades denominadas estágios. Uma porta NAND de duas entradas corresponde a aproximadamente 2,5 estágios. Um estágio contém um único transistor P e um único transistor N. A proteção contra "latch-up" (gatilhamento de transistores bipolares parasitas nas estruturas MOS) é garantida por regras de projeto, duplos anéis de guarda, espaçamento adequado entre transistores diferentes e processo de difusão consistente. O nível de imunidade a "latch-up" é de 100 mA. Na Figura 4.4, vê-se a representação do sistema SOLO 1400.

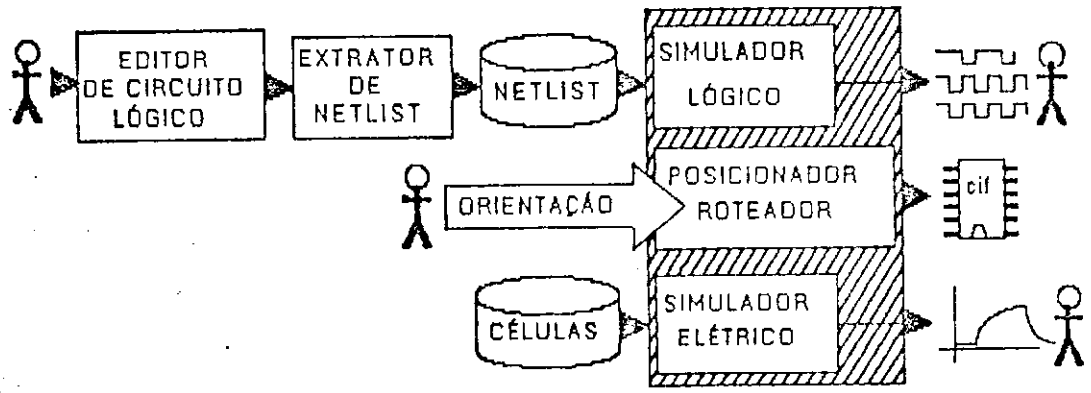


Figura 4.4: O sistema SOLO 1400.

4.4.2 Metodologia de Simulação Ascendente/Geração das Máscaras

1 - Montagem do Sistema

Para cada nível da hierarquia, a partir de células padrões disponíveis na biblioteca ("base library") do SOLO 1400, montam-se os subsistemas. Estes subsistemas são adicionados à biblioteca e chamados quando da formação de blocos de hierarquia superior.

2 - Realização das Simulações

Com a execução de programas do sistema SOLO 1400, são preparados os arquivos de entrada para a simulação de cada bloco sob condições preespecificadas de temperatura e exigências de confiabilidade. A apresentação dos resultados se dá na forma de diagramas de tempo.

3 - Geração das Máscaras

Os subsistemas gerados são conectados por um programa que produz o "layout" para os estágios e são interconectados automaticamente. A estratégia utilizada é do tipo policelular hierárquica.

4.5 Apresentação dos Resultados

4.5.1 Resultados das Simulações

Os valores dos pixels codificados de imagens que apresentam bordas em várias direções (Fig. 4.5) foram usados como entradas do "ASIC" proposto para sua validação. As Figuras 4.6 a 4.10 mostram os mapas de pixels das imagens escolhidas, juntamente com os mapas de pixels das respectivas imagens processadas.

Nas simulações para a validação ascendente, foram usadas frequências de até 9 MHz.

Todos os blocos de todos os níveis foram simulados e validados. Como ilustração, os resultados obtidos para a validação dos blocos MD1 e MD8 são fornecidos no Apêndice C.

Os resultados garantem não apenas que a implementação proposta realiza com sucesso a operação de Roberts, mas que ela é, também, capaz de gerar uma imagem gradiente em tempo real para a maioria das aplicações. Considerando-se a resolução usual de 512 × 512 pontos [100], o tempo necessário para o processamento é 29 ms. Os processadores da família TMS, por exemplo, exigem um mínimo de 8 ciclos de instrução para a realização do processamento de cada pixel, de modo que os tempos necessários para o processamento de uma imagem deste tamanho serão 419 ms (TMS32010), 408,94 ms (TMS32020), 210 ms (TMS320C25) e 126 ms (TMS320C30) [101, 102].

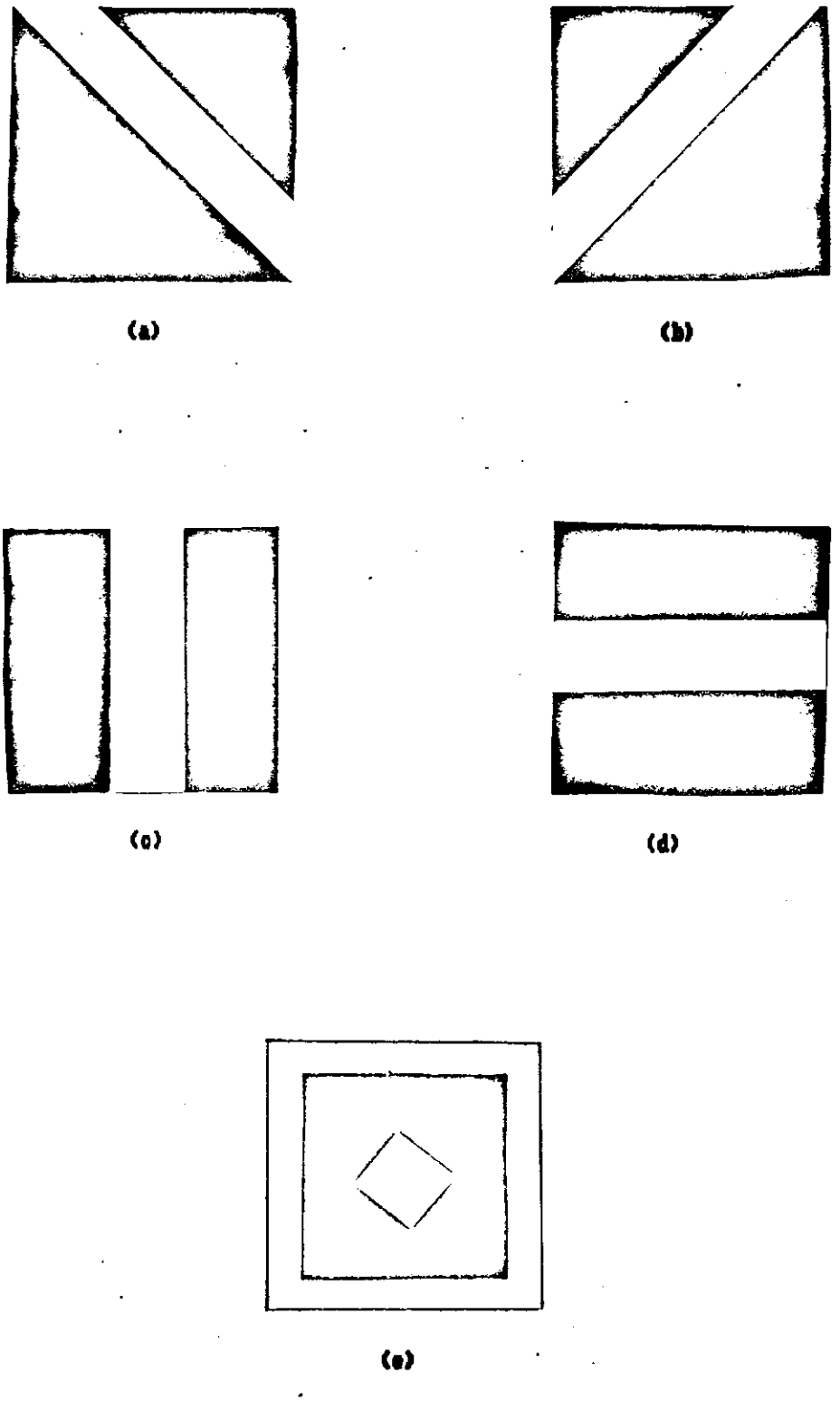


Figura 4.5: Imagens usadas para a validação do "ASIC".

FF FF FF FF 00 00 00 00 00 00 00 00 00 00 00 00
00 FF FF FF FF 00 00 00 00 00 00 00 00 00 00 00
00 00 FF FF FF FF 00 00 00 00 00 00 00 00 00 00
00 00 00 FF FF FF FF 00 00 00 00 00 00 00 00 00
00 00 00 00 FF FF FF FF 00 00 00 00 00 00 00 00
00 00 00 00 00 FF FF FF FF 00 00 00 00 00 00 00
00 00 00 00 00 00 FF FF FF FF 00 00 00 00 00 00
00 00 00 00 00 00 00 FF FF FF FF 00 00 00 00 00
00 00 00 00 00 00 00 00 FF FF FF FF 00 00 00 00
00 00 00 00 00 00 00 00 00 FF FF FF FF 00 00 00
00 00 00 00 00 00 00 00 00 00 FF FF FF FF 00 00
00 00 00 00 00 00 00 00 00 00 00 FF FF FF FF 00
00 00 00 00 00 00 00 00 00 00 00 00 FF FF FF FF
00 00 00 00 00 00 00 00 00 00 00 00 00 FF FF FF
00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF FF
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF

FF 00 00 FF FF 00 00 00 00 00 00 00 00 00 00 00
FF FF 00 00 FF FF 00 00 00 00 00 00 00 00 00 00
00 FF FF 00 00 FF FF 00 00 00 00 00 00 00 00 00
00 00 FF FF 00 00 FF FF 00 00 00 00 00 00 00 00
00 00 00 FF FF 00 00 FF FF 00 00 00 00 00 00 00
00 00 00 00 FF FF 00 00 FF FF 00 00 00 00 00 00
00 00 00 00 00 FF FF 00 00 FF FF 00 00 00 00 00
00 00 00 00 00 00 FF FF 00 00 FF FF 00 00 00 00
00 00 00 00 00 00 00 FF FF 00 00 FF FF 00 00 00
00 00 00 00 00 00 00 00 FF FF 00 00 FF FF 00 00
00 00 00 00 00 00 00 00 00 FF FF 00 00 FF FF 00
00 00 00 00 00 00 00 00 00 00 FF FF 00 00 FF FF
00 00 00 00 00 00 00 00 00 00 00 FF FF 00 00 FF
00 00 00 00 00 00 00 00 00 00 00 00 FF FF 00 FF
00 00 00 00 00 00 00 00 00 00 00 00 00 FF FF FF
00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF FF

Figura 4.6: Mapa de pixels.

a) Imagem da Fig. 4.5.a.

b) Imagem processada.

```
00 00 00 00 00 00 00 00 00 00 00 00 FF FF FF FF
00 00 00 00 00 00 00 00 00 00 00 FF FF FF FF 00
00 00 00 00 00 00 00 00 00 00 FF FF FF FF 00 00
00 00 00 00 00 00 00 00 00 FF FF FF FF 00 00 00
00 00 00 00 00 00 00 00 FF FF FF FF 00 00 00 00
00 00 00 00 00 00 00 FF FF FF FF 00 00 00 00 00
00 00 00 00 00 00 FF FF FF FF 00 00 00 00 00 00
00 00 00 00 00 FF FF FF FF 00 00 00 00 00 00 00
00 00 00 00 FF FF FF FF 00 00 00 00 00 00 00 00
00 00 00 FF FF FF FF 00 00 00 00 00 00 00 00 00
00 00 FF FF FF FF 00 00 00 00 00 00 00 00 00 00
00 FF FF FF FF 00 00 00 00 00 00 00 00 00 00 00
FF FF FF FF 00 00 00 00 00 00 00 00 00 00 00 00
FF FF FF 00 00 00 00 00 00 00 00 00 00 00 00 00
FF FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00
FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
00 00 00 00 00 00 00 00 00 00 FF FF 00 00 FF FF
00 00 00 00 00 00 00 00 00 FF FF 00 00 FF FF 00
00 00 00 00 00 00 00 00 FF FF 00 00 FF FF 00 00
00 00 00 00 00 00 00 FF FF 00 00 FF FF 00 00 00
00 00 00 00 00 00 FF FF 00 00 FF FF 00 00 00 00
00 00 00 00 00 FF FF 00 00 FF FF 00 00 00 00 00
00 00 00 00 FF FF 00 00 FF FF 00 00 00 00 00 00
00 00 00 FF FF 00 00 FF FF 00 00 00 00 00 00 00
00 00 FF FF 00 00 FF FF 00 00 00 00 00 00 00 00
00 FF FF 00 00 FF FF 00 00 00 00 00 00 00 00 00
FF FF 00 00 FF FF 00 00 00 00 00 00 00 00 00 FF
FF 00 00 FF FF 00 00 00 00 00 00 00 00 00 00 FF
00 00 FF FF 00 00 00 00 00 00 00 00 00 00 00 FF
00 FF FF 00 00 00 00 00 00 00 00 00 00 00 00 FF
FF FF 00 00 00 00 00 00 00 00 00 00 00 00 00 FF
FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Figura 4.7: Mapa de pixels.

a) Imagem da Fig. 4.5.b.

b) Imagem processada.

00	00	00	00	00	00	FF	FF	FF	FF	00	00	00	00	00	00
00	00	00	00	00	00	FF	FF	FF	FF	00	00	00	00	00	00
00	00	00	00	00	00	FF	FF	FF	FF	00	00	00	00	00	00
00	00	00	00	00	00	FF	FF	FF	FF	00	00	00	00	00	00
00	00	00	00	00	00	FF	FF	FF	FF	00	00	00	00	00	00
00	00	00	00	00	00	FF	FF	FF	FF	00	00	00	00	00	00
00	00	00	00	00	00	FF	FF	FF	FF	00	00	00	00	00	00
00	00	00	00	00	00	FF	FF	FF	FF	00	00	00	00	00	00
00	00	00	00	00	00	FF	FF	FF	FF	00	00	00	00	00	00
00	00	00	00	00	00	FF	FF	FF	FF	00	00	00	00	00	00
00	00	00	00	00	00	FF	FF	FF	FF	00	00	00	00	00	00
00	00	00	00	00	00	FF	FF	FF	FF	00	00	00	00	00	00
00	00	00	00	00	00	FF	FF	FF	FF	00	00	00	00	00	00
00	00	00	00	00	00	FF	FF	FF	FF	00	00	00	00	00	00
00	00	00	00	00	00	FF	FF	FF	FF	00	00	00	00	00	00
00	00	00	00	00	00	FF	FF	FF	FF	00	00	00	00	00	00

00	00	00	00	00	FF	00	00	00	FF	00	00	00	00	00	00
00	00	00	00	00	FF	00	00	00	FF	00	00	00	00	00	00
00	00	00	00	00	FF	00	00	00	FF	00	00	00	00	00	00
00	00	00	00	00	FF	00	00	00	FF	00	00	00	00	00	00
00	00	00	00	00	FF	00	00	00	FF	00	00	00	00	00	00
00	00	00	00	00	FF	00	00	00	FF	00	00	00	00	00	00
00	00	00	00	00	FF	00	00	00	FF	00	00	00	00	00	00
00	00	00	00	00	FF	00	00	00	FF	00	00	00	00	00	00
00	00	00	00	00	FF	00	00	00	FF	00	00	00	00	00	00
00	00	00	00	00	FF	00	00	00	FF	00	00	00	00	00	00
00	00	00	00	00	FF	00	00	00	FF	00	00	00	00	00	00
00	00	00	00	00	FF	00	00	00	FF	00	00	00	00	00	00
00	00	00	00	00	FF	00	00	00	FF	00	00	00	00	00	00
00	00	00	00	00	FF	00	00	00	FF	00	00	00	00	00	00
00	00	00	00	00	FF	00	00	00	FF	00	00	00	00	00	00
00	00	00	00	00	FF	FF	FF	FF	FF	00	00	00	00	00	00

Figura 4.8: Mapa de pixels.

a) Imagem da Fig. 4.5.c.

b) Imagem processada.

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Figura 4.9: Mapa de pixels.

a) Imagem da Fig. 4.5.d.

b) Imagem processada.

```
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF 00 00 00 00 00 00 00 00 00 00 00 00 FF FF
FF FF 00 00 00 00 00 00 00 00 00 00 00 00 FF FF
FF FF 00 00 00 00 00 FF FF 00 00 00 00 00 FF FF
FF FF 00 00 00 00 FF FF FF FF 00 00 00 00 FF FF
FF FF 00 00 00 FF FF FF FF FF FF 00 00 00 FF FF
FF FF 00 00 FF FF FF FF FF FF FF FF 00 00 FF FF
FF FF 00 FF FF FF FF FF FF FF FF FF FF 00 00 FF FF
FF FF 00 FF FF FF FF FF FF FF FF FF 00 00 00 FF FF
FF FF 00 00 FF FF FF FF FF FF 00 00 00 00 FF FF
FF FF 00 00 00 FF FF FF FF 00 00 00 00 00 FF FF
FF FF 00 00 00 00 FF FF 00 00 00 00 00 00 FF FF
FF FF 00 00 00 00 00 00 00 00 00 00 00 00 FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 FF FF FF FF FF FF FF FF FF FF FF FF FF 00 00
00 FF 00 00 00 00 00 00 00 00 00 00 00 FF 00 00
00 FF 00 00 00 00 FF FF FF 00 00 00 00 FF 00 00
00 FF 00 00 00 FF FF 00 FF FF 00 00 00 FF 00 00
00 FF 00 00 FF FF 00 00 00 FF FF 00 00 FF 00 00
00 FF 00 FF FF 00 00 00 00 00 FF FF 00 FF 00 00
00 FF FF FF 00 00 00 00 00 00 00 FF 00 FF 00 00
00 FF FF 00 00 00 00 00 00 00 FF FF 00 FF 00 00
00 FF FF FF 00 00 00 00 00 FF FF 00 00 FF 00 00
00 FF 00 FF FF 00 00 00 FF FF 00 00 00 FF 00 00
00 FF 00 00 FF FF 00 FF FF 00 00 00 00 FF 00 00
00 FF 00 00 00 FF FF FF 00 00 00 00 00 FF 00 00
00 FF FF FF FF FF FF FF FF FF FF FF FF FF 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

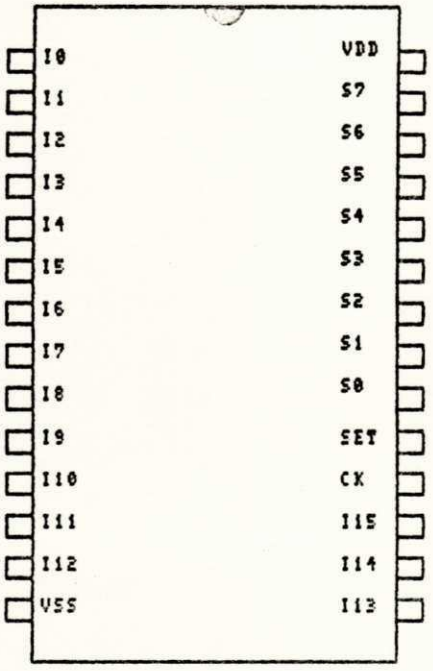
Figura 4.10: Mapa de pixels.

a) Imagem da Fig. 4.5.e.

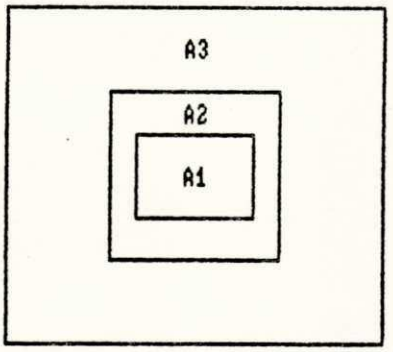
b) Imagem processada.

4.5.2 Apresentação Final do "Chip"

O resultado da implementação do circuito operador de Roberts a partir da arquitetura apresentada na Figura 3.1 foi um "chip" com 1440 estágios, 28 pinos de E/S e dimensões 4,42 x 3,46 mm2. A Figura 4.11 descreve a estrutura final do "chip".



(a)



	mm2	%A3
A1	7,22	47,1
A2	8,41	55,0
A3	15,30	100,0
AR	1,19	7,8
AP	6,89	45,0

A1 = Área do Operador
A2 = Coração do Chip
A3 = Área Total do Chip
AR = A2-A1 = Área de Roteamento para Pads
AP = A3-A2 = Área Utilizada pelos Pads

(b)

Figura 4.11: Apresentação do "chip".
a) Pinagem do "chip".
b) Utilização da área de silício.

4.5.3 Uma Comparação SOLO 1400 X ALLIANCE

Considerando-se o aspecto didático deste trabalho, torna-se interessante um estudo comparativo do desempenho de um "software" profissional que fornece geração automática de máscaras (SOLO 1400) com um "software" didático (ALLIANCE) que permite a interação do projetista no desenho das máscaras.

Não se esquecendo que os propósitos dos sistemas são distintos, apenas para efeito de ilustração, decidiu-se implementar parte do "ASIC" do operador de Roberts em ambos os sistemas. Para isso, escolheu-se o bloco MD8 e, utilizando-se a mesma metodologia de células padrões, chegou-se às estruturas mostradas no Apêndice D. O "layout" gerado pelo ALLIANCE usa uma área de 0,16 mm2, enquanto que o SOLO 1400 necessitou uma área de 0,25 mm2.

Os resultados indicam uma sensível redução (36%) de área obtida com o ALLIANCE, entretanto, o menor tempo necessário para a realização com o SOLO 1400 torna imperativa a escolha desta ferramenta em caso de grandes projetos.

Outros "layouts" são mostrados no Apêndice E.

5 CONCLUSÕES

Este trabalho tratou da definição e validação da arquitetura de um "ASIC" para implementação de uma técnica de realce para deteção de bordas em imagens digitalizadas.

Realizou-se um estudo do processo de deteção de bordas, onde foram apresentados algoritmos para esse fim correntes na literatura. O operador de Roberts foi escolhido pela sua simplicidade e eficiência. Foram descritos alguns aspectos envolvidos e metodologias normalmente usadas no projeto de um circuito integrado.

A implementação do "ASIC" operador de Roberts envolveu duas etapas. A validação descendente foi obtida através dos "softwares" ALLIANCE e ELOISE, com a definição do comportamento dos blocos funcionais e simulações em níveis hierárquicos. A validação ascendente (simulações lógicas e elétricas) foi realizada com auxílio do sistema para desenvolvimento de projetos SOLO 1400. Este sistema foi também utilizado para a geração do "layout" segundo a metodologia de célula padrão.

As simulações feitas indicaram o bom funcionamento para freqüências até 9 MHz deste "chip" que foi desenvolvido para a tecnologia CMOS 2 micras, tem 28 pinos de E/S, 1440 estágios e ocupa uma área de 4,42 x 3,46 mm2.

A arquitetura se mostra robusta e eficiente, não

impondo restrições práticas quanto ao tamanho da imagem. Isto é, o tamanho da imagem não influencia no desempenho do "chip". Considerando-se uma imagem de tamanho 512 x 512 (resolução espacial usual), o tempo necessário para o processamento é de 29 ms, possibilitando a geração de uma imagem gradiente em tempo real.

A fundição deste "chip" está prevista para o 6º PMU - Programa Multiusuário Brasileiro.

Este foi um estudo preliminar que visa à realização de sistemas integrados mais complexos para tratamento de imagens em interfaces de comunicação de redes com integração de serviços. Sugerem-se, assim, estudos visando à definição de outras arquiteturas para processamento digital de imagens, em particular, para codificação de imagens utilizando operadores gradientes como pré-processadores.

Pode-se, também, complementar o "ASIC" proposto, através da implementação de uma lógica programável que permita a escolha de um limiar para geração do mapa de bordas MB(i,j) da imagem gradiente gerada pelo operador de Roberts.

6 REFERÊNCIAS BIBLIOGRÁFICAS

1 - Rosenfeld, A. and Kak, A. Digital Image Processing. Academic Press, New York, 1976.

2 - Tousi, R., Lopes, A. and Bousquet, P. A statistical and geometrical edge detector for SAR images. IEEE Trans. on Geoscience and Remote Sensing, 26, 1988, 764-773.

3 - Rosenfeld, A. and Thurston, M. Edge and curve detection for visual scene analysis. IEEE Trans. on Computers, 20, 1971, 562-569.

4 - Abdou, J. and Pratt, W. K. Quantitative design and evaluation of enhancement/thresholding edge detectors. Proc. of the IEEE 67, 1979, 753-763.

5 - Hueckel, M. H. An operator which locates edges in digital pictures. Journal of the ACM, 18, 1971, 113-125.

6 - Mero, L. and Vassy, Z. A simplified and fast version of the Hueckel operator for finding optimal edges in picture. Proc. of the 4th IJCAI, 650-655.

7 - Rosenfeld, A. The max Roberts operator is Hueckel type edge detector. IEEE Trans. on Pattern

Analysis and Machine Intelligence, 3, 1981,101-103.

8 - Gonzalez, R. C. and Wintz, P. **Digital Image Processing**. Addison-Wesley, 1977.

9 - Ngan, K. N., Kassim, A. A. and Singh, H. S. Parallel image processing based on the TMS32010 digital signal processor. Proc. of the IEEE, 134, 1987, 119-124.

10 - Mehrez, H. et alii. On processor board built around a high speed full custom single chip. Laboratoire MASI - CAO & VLSI, 1988, 49-57.

11 - Kanopoulos, N., Vasanthavada, N. and Baker, R. L. Design of an image edge detection filter using the Sobel operator. IEEE Journal of Solid-State Circuits, 23, 1988, 358-367.

12 - Ruetz, P. A. and Brodersen, R. W. Architectures and design techniques for real-time image processing ICs. IEEE Journal of Solid-State Circuits, 22, 1987, 233-250.

13 - Jutand, F. A 13,5 MHz single chip multiformat discrete cosine transform. SPIE, 845, Visual Communications and Image Processing II, 1987, 6-11.

14 - Sugai, M. et alii. VLSI processor for image processing. Proc. of IEEE, 75, 1987, 1160-1165.

15 - C. Y. Lee, F. V. M. Cathoor, and H. J. De Man. An efficient ASIC architecture for real-time edge detection. IEEE Trans. on Circuits and Systems, 36, 1989, 1350-1359.

16 - Oflazer, K. Design and implementation of a single chip 1-D median filter. IEEE Trans. Acoust., Speech, Signal Processing, 31, 1983, 1164-1168.

17 - Demmassieux, N. et alii. VLSI architecture for a one chip video median filter. Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, 1985, 1001-1004.

18 - Harber, R. G., Bass, S. C and Neudeck, G. W. VLSI implementation of a fast rank order filtering algorithm. Proc. Int. Symp. on Circuits and Systems, 1985, 1396-1399.

19 - Atam, E., Aatre, V. K. and Wong, K. M. A fast method for real-time median filtering. IEEE Trans. Acoust., Speech, Signal Processing, 28, 1980, 415-421.

20 - Delman, D. J. Digital pipelined median filter design for real-time image processing. Proc. SPIE, 298, 1981, 184-188.

21 - Kundu, A., Singh, G. and Butner, S. VLSI implementation of two-dimensional generalized mean filter. Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, 1985, 997-1000.

22 - Roskind, J. A fast sort-selection filter chip with effectively linear hardware complexity. Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, 1985, 1519-1522.

23 - Giozza, W. F. et alii. Redes Locais: Tecnologia e Aplicações. McGraw-Hill, 1986.

24 - Barros, M. A. Filtros Espaciais: Implementação, Estudo Comparativo e Aplicações. Dissertação de Mestrado, DEE-UFPB, 1990.

25 - West, N. H. E. and Eshraghian, K. Principles of CMOS Design - A Systems Perspective. Addison-Wesley, 1985.

26 - ALLIANCE - Manuel de L'utilisateur. Laboratoire MASI - CAO & VLSI, Université Pierre et Marie Curie - Paris 6, 1986.

27 - SOLO Family Digital Databook. European Silicon Structures - ES2, 1988.

28 - Projeto CAPES/COFECUB Nº 85/88: Redes Integradas/ Microeletrônica UFPb/Paris 6, 1988.

29 - Davis, L. S. A survey of edge detection techniques Computer Graphics and Image Processing,4, 1975, 248-276.

30 - Jacobus, C. and Chien, R. T. Two new edge detectors. IEEE Trans. on Pattern Analysis and Machine Intelligence, 3, 1981, 581-592.

31 - Pratt, W. K. Digital Image Processing. Wiley-Interscience, 1978.

32 - Perkins, W. A. Area segmentation of images using edge description. Computer Graphics and Image

Processing, 13, 1982, 179-195.

33 - Levialdi, S. Finding the edge, Digital Image Processing. C. Simon and R.M. Harralick. Eds. D. Reidel, Dordrecht, 1981.

34 - Brady, M. Artificial intelligence approaches to image understanding. Pattern Recognition Theory and Applications. J. Kittler, K.S. Fu and L.F. Pau Eds. D. Reidel, Dordrecht, 1982.

35 - Kunt, M., Bénard, M. and Leonardi, R. Recents Results in High-Compression Image Coding. IEEE Trans. on Circuits and Systems, 34, 1987, 1306-1336.

36 - Hummel, R. A. Histogram modification techniques. Computer Graphics and Image Processing, 4, 1975, 209-224.

37 - Davies, E. R. Design of optimal gaussian operators in small neighbourhoods. Image and Vision Computing 5, 1987, 199-205.

38 - Goodman, J. W. Some fundamental properties of speckle. Journal of the Op. Soc. of America, 66, 1976, 1145-1150.

39 - Foglein, J. On edge gradient approximations. Pattern Recognition Letters, 1, North-Holland, 1983, 429-434.

40 - Manual of Remote Sensing - Volume II: Interpretation and applications. The American Society of

Photogrametry, 1975.

41 - Prewitt, J. M. S. Object enhancement and extraction. in Picture Processing and Psychopictorics. Lipkin, B. S. and Rosenfeld, A., Academic Press, 1970.

42 - Prewitt, J. M. S.. Object enhancement and extraction. in T. S. Huang Ed., Picture Processing and Digital Filtering. Springer-Verlag, 1975, 75-149.

43 - Kirsch, R. Computer determination of the constituent structure of biological images. Computers and Biomedical Research, 4, 1971, 315-328.

44 - Robinson, G. S. Edge detection by compass gradient masks. Computer Graphics and Image Processing, 6, 1977, 492-501.

45 - Duda, R. O. and Hart, P. E. Pattern Classification and Scene Analysis. John Wiley & Sons, 1973.

46 - Overington, I. and Grennway, P. Pratical first difference edge detection with subpixel accuracy. Image and Vision Computing, 5, 1987, 217-224.

47 - Castleman, K. R. Digital Image Processing. Prentice Hall, 1979.

48 - Roberts, L. G. Machine perception of three-dimensional solids. in Optical and Electro-optical Information Processing. Tippett, J. T. et alli, Cambridge, MA: M.I.T. Press, 1965, 159-197.

49 - Argyle, E. Techniques for edge detection. Proc. of the IEEE, 59, 1971, 286-287.

50 - Macleod, I. D. G. Comments on Techniques for edge detection. Proc. of the IEEE, 60, 1972, 344.

51 - Thompson, W. B. Textural boundary analysis. IEEE Trans. on Computers, 26, 1977, 272-274.

52 - Davis, L. S. and Mitiche, A. Edge detection in textures. Computer Graphics and Image Processing, 12, 1980, 25-39.

53 - Davis, L. S. and Mitiche, A. Edge detection in textures - maxima selection. Computers Graphics and Image Processing, 16, 1981, 158-165.

54 - Hueckel, M. H. A local visual operator which recognizes edges and lines. Journal of the ACM, 20, 1973, 634-647.

55 - Haralick, R. M. Edge and region analysis for digital image data. Computer Graphics and Image Processing, 12, 1980, 60-73.

56 - Haralick, R. M. A facet model for image data. Computer Graphics and Image Processing, 15, 1981, 634-647.

57 - Haralick, R. M. Ridges and valleys on digital image. Computer Graphics and Image Processing, 22, 1983, 28-38.

58 - Haralick, R. M. Digital step edges from zero cross-

ing of second directional derivatives. IEEE Trans. on Pattern Analysis and Machine Intelligence, 6, 1984, 58-68.

59 - Nevatia, R. Evaluation of simplified Hueckel edge line detector. Computer Graphics and Image Processing, 6, 1977, 582-588.

60 - Abramatic, J. F. Why the simplest Hueckel edge detector is a Roberts operator. Computer Graphics and Image Processing, 17, 1981, 79-83.

61 - Shipman, A. L., Bitmead, R. R. and Allen, G. H. Diffuse edge fitting and following: a location-adaptive approach. IEEE Trans. on Pattern Analysis and Machine Inteligence, 6, 1984, 96-102.

62 - Rosenfeld, A., Lee, Y. and Thomas, R. Edge and curve detection for visual scene analysis. IEEE Trans. on Computers, 20, 1971, 562-569.

63 - Rosenfeld, A. and Troy, E. B. Visual texture analysis. TR 70-116, Comp. Sc. Center, Univ. of Maryland, 1970.

64 - Rosenfeld, A. and Thurston, M. Visual texture analysis 2. TR 70-129, Comp. Sc. Center, Univ. of Maryland, 1970.

65 - Rosenfeld, A. A nonlinear edge detection technique. Proc. of IEEE, 58, 1970, 814-816.

66 - Rosenfeld, A., Thurston, M. and Lee, Y. H. Edge and curve detection, further experiments. IEEE

Trans. on Computers, 21, 1972, 677-7-1.

67 - Zucker, S. W., Rosenfeld, A. and Davis, L. S. Picture segmentation by texture discrimination. IEEE Trans. on Computers, 24, 1975, 1228-1233.

68 - Panda, D. P. and Rosenfeld, A. Image segmentation by pixel classification in (gray level, edge value) space. IEEE Trans. on Computers, 27, 1978, 875-879.

69 - Chanda, B., Chanduri, B. B. and Majunder, D. D. Some algorithms for image enhancement incorporating human visual response. Pattern Recognition, 17, 1984, 423-428.

70 - Clark, J. J. Singularities of contrast functions in scale space. Proc. First Int. Conf. Comput. Vision, 1987, 491-495.

71 - Clark, J. J. Authenticating edges produced by zero-crossing algorithms. IEEE Trans. on Pattern Analysis and Machine Intelligence, 11, 1989, 43-55.

72 - Ritcher, J. and Ullman, S. Non-linearities in cortical simple cells and the possible detection of zero crossings. Biologie Cybernetic, 53, 1986, 195-202.

73 - Pitas, I. Markovian models for image labeling and edge detection. Signal Processing, 15, 1988, 365-374.

74 - Eichel, P. and Delp, E. Sequential edge detection

in correlated random fields. Proc. Comput. Vision and Pattern Recognition, 1985, 14-21.

75 - Zhou, Y. T., Vienkateswar, V. and Chellappa, R. Edge detection and linear feature extraction using a 2-D random field model. IEEE Trans. on Pattern Analysis and Machine Intelligence, 11, 1989, 84-94.

76 - Ramer, E. U. The transformation of photographic images into stroke arrays. IEEE Trans. on Circuits and Systems, 22, 1976, 363-373.

77 - Nevatia, R. Locating object boundarys in textured environments. IEEE Trans. on Computers,25, 1976, 1170-1175.

78 - Pavlidis, T. A thinning algorithm for discrete binary images. IEEE Trans. on Pattern Analysis and Machine Intelligence, 2, 1980, 8-15.

79 - Nevatia, R. and Babu, K. R. Linear feature extraction and description. Computer Graphics and Image Processing, 13, 1980, 257-269.

80 - Ikonomopoulos, A. An approach to edge detection based on the direction of edge elements. Computer Graphics and Image Processing, 19, 1982, 179-195.

81 - Favre, A. and Keller, H. Parallel syntatic thinning by recoding of binary pictures. Computer Graphics and Image Processing, 23, 1983, 99-112.

82 - Gil, B., Mitiche, A. and Aggarwal, J. K. Experiments in combining intensity and range

edge maps. Computer Graphics and Image Processing, 21, 1983, 395-411.

83 - Lacroix, V. A three-module strategy for edge detection. IEEE Trans. on Pattern Analysis and Machine Intelligence, 11, 1989, 803-810.

84 - Stefanelli, R. and Rosenfeld, A. Some parallel thinning algorithm for digital pictures. Journal of the ACM, 18, 1971, 255-264.

85 - Hong, T. H., Dyer, C. R. and Rosenfeld, A. Texture primitive extraction using an edge-based approach. IEEE Trans. on Systems, Man and Cybernetics, 10, 1980, 659-675.

86 - Eberlein, R. B. An iterative gradient edge detection algorithm. Computer Graphics and Image Processing, 5, 1976, 245-253.

87 - Kasvand, T. Iterative edge detection. Computer Graphics and Image Processing, 4, 1975, 279-286.

88 - Araújo, A. A. Filtros Espaciais: Estudo Comparativo e Aplicação em Segmentação e Classificação de Imagens. Tese de Doutorado, DEE-UFPB, 1987.

89 - Barros, L. A. et alii. O operador de Roberts: uma implementação com tecnologia CMOS para deteção de bordas em tempo real. 3º SIBGRAP, Gramado, 1990.

90 - Chandhuri, B. B. and Chanda, B. The equivalence of best plane fit gradient with Robert's, Prewitt's and Sobel's gradient. Signal Processing, 6, 1984,

143-151.

91 - Mammana, C. I. e Mammana, A. P. Introdução ao Projeto de Circuitos Integrados. Coleção EBAI, Edição Preliminar, 1987.

92 - Amara, A. et alii. A macro cell generator for VLSI circuits. ICM'88, Algria, 1988, 5-7.

93 - Allegre, P. et alii. Génération incrémentale d'un processeur flottant 32 bits. CCVLSI-88, Halifax-Canada, 1988, 24-25.

94 - Sangiovanni, N. B. W. and Vincentelli, A. L. Computer aided design for VLSI circuits. IEEE Computer, 19, 1986, 38-60.

95 - Yoshimura, T. An efficient channel router. Proceedings of the 21 st Design Automation Conference, 1984, 610-615.

96 - Serlet, B. ALPS a generator of static CMOS layout from boolean expressions. Advanced Research in VLSI Proceedings of the 4th MIT Conference.

97 - McCormick, S. P. EXCL: A circuit extractor for IC designs. Proc. of the 21 st Design Automation Conference, 1984, 616-623.

98 - Allegre, P. et alii. Générateur de macro-cellules logiques VLSI. CCVLSI Halifax, Canada, 1988.

99 - ELOISE - Manuel de L'utilisateur. Laboratoire MASI - CAO & VLSI. Université Pierre et Marie Curie -

Paris 6, 1986.

100 - Tamura, H. and Mori, S. A data management system for manipulating large images. Proc. IEEE Workshop on Picture Data Description and Management, 1977, 45-54.

101 - TMS32010 Assembly Language Programer's Guide. Texas Instruments, 1983.

102 - Jaccoud, C. F. B. e Nascimento, R. L. Processador Digital de Sinais TMS32010. Publicação da Embratel, 1989.

APÊNDICE A

DIVISÃO HIERÁRQUICA UTILIZADA PARA IMPLEMENTAÇÃO DO "ASIC" OPERADOR DE ROBERTS

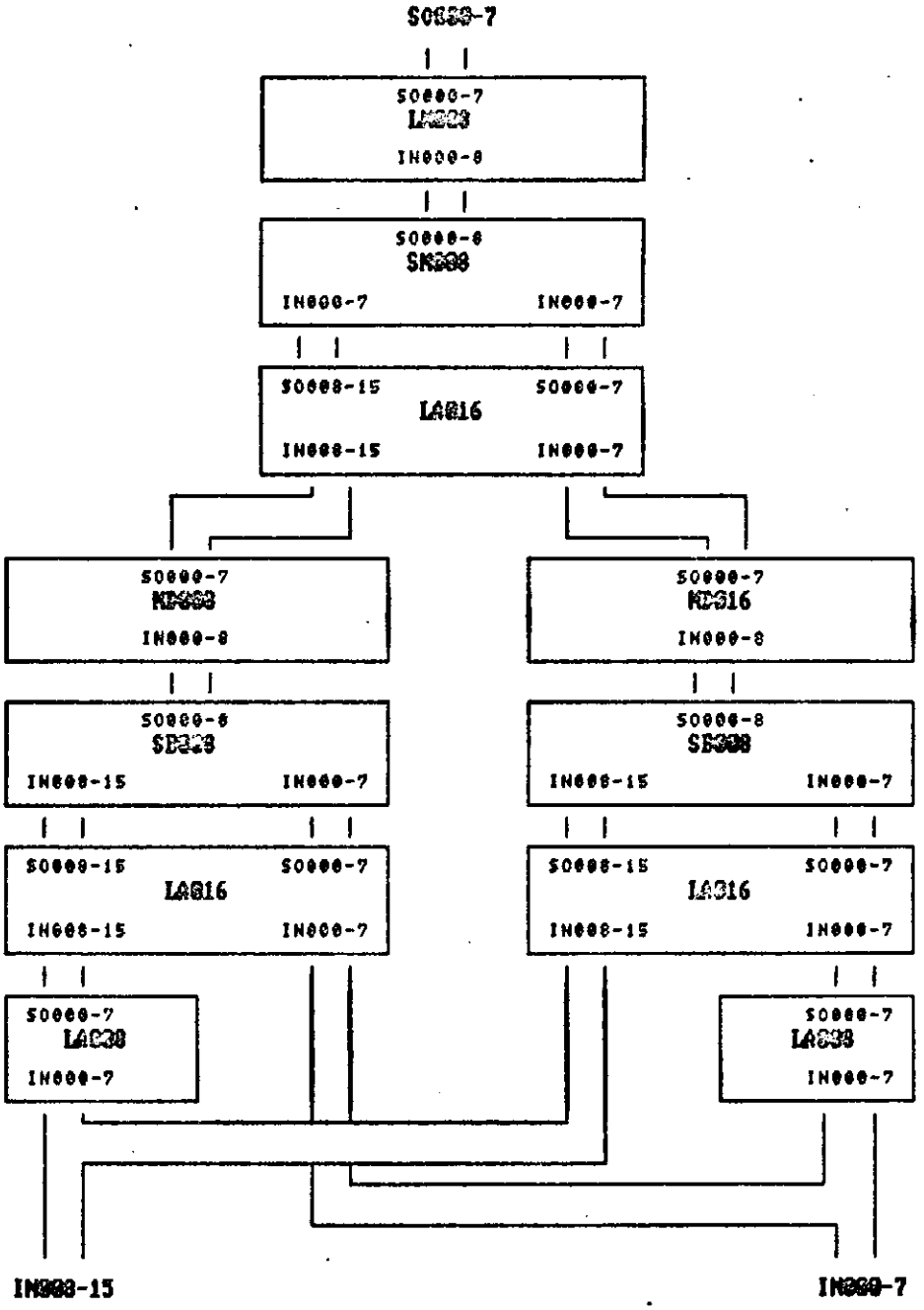


Figura A.1: Nível 0 - operador de Robérts.

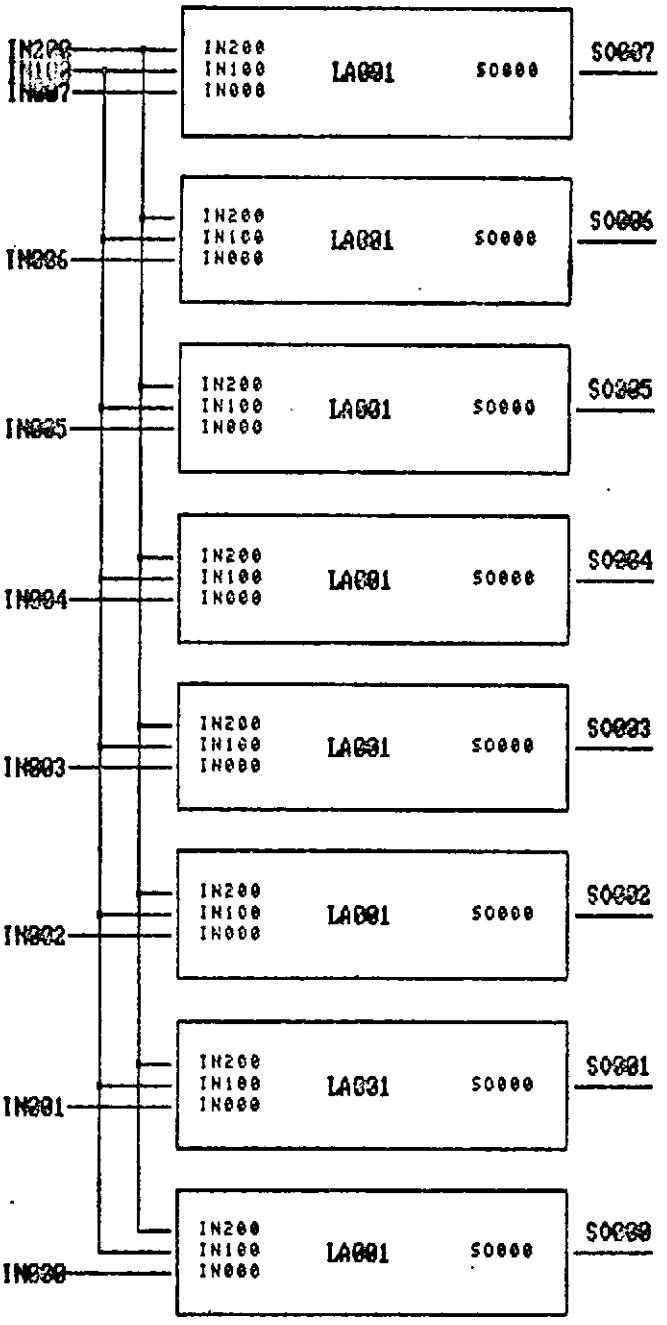


Figura A.2: Nível 1 - bloco LA8

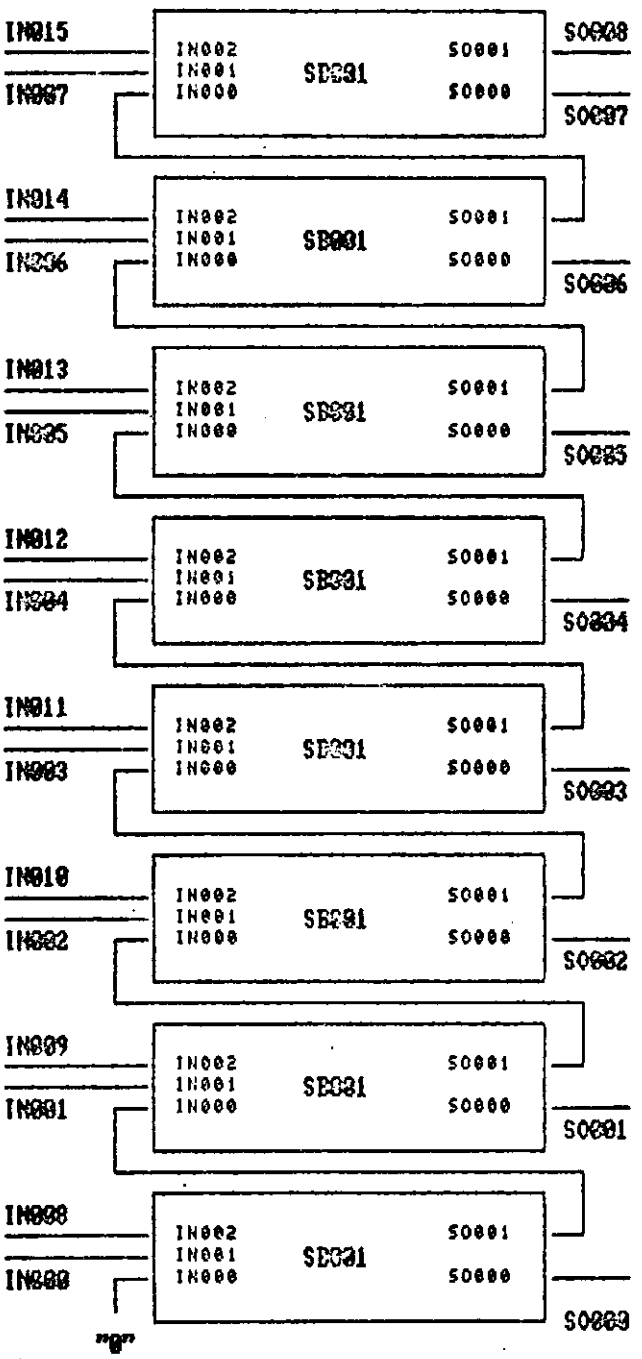


Figura A.3: Nível 1 - bloco SB8

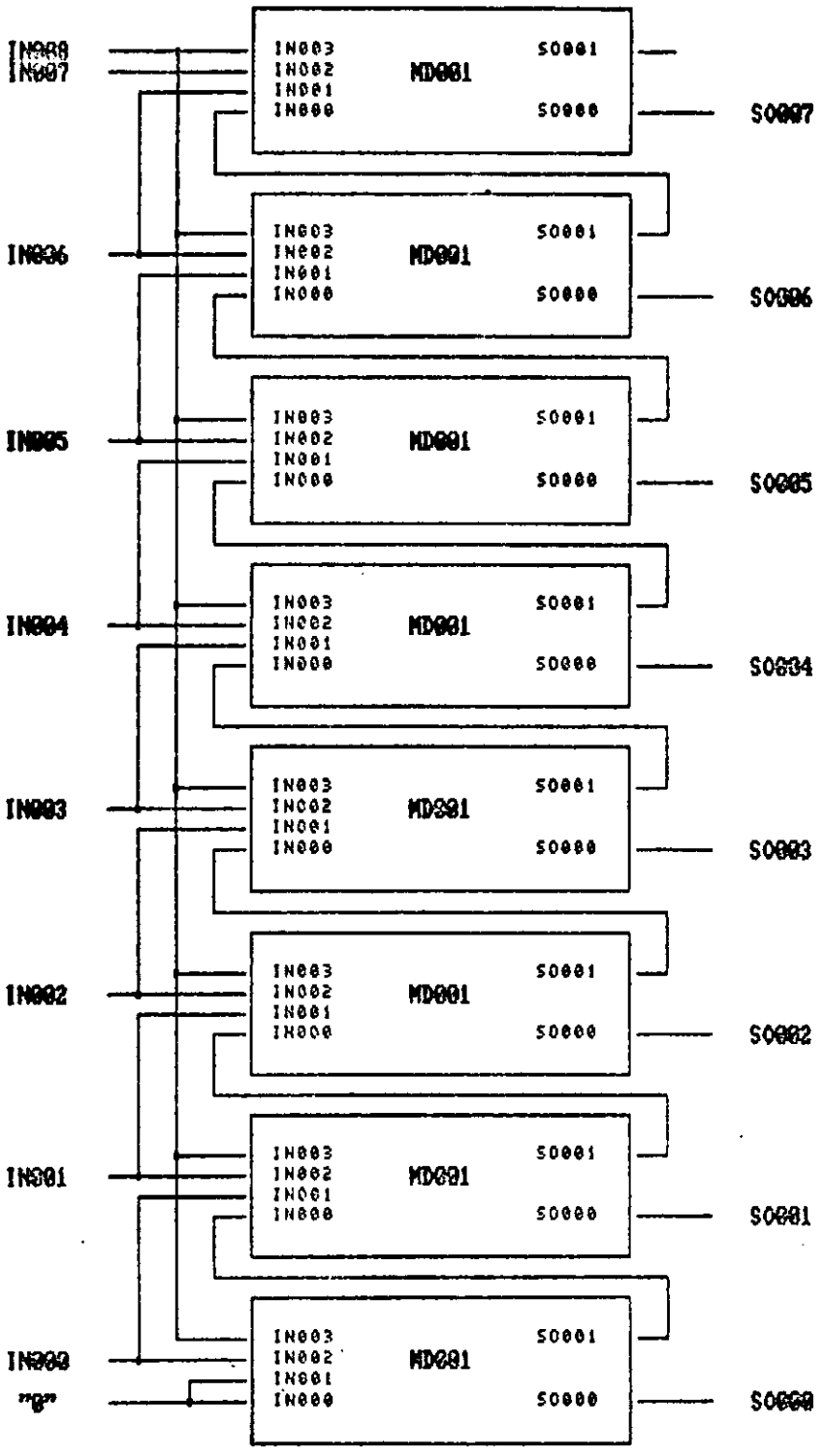


Figura A.4: Nível 1 - bloco MD8

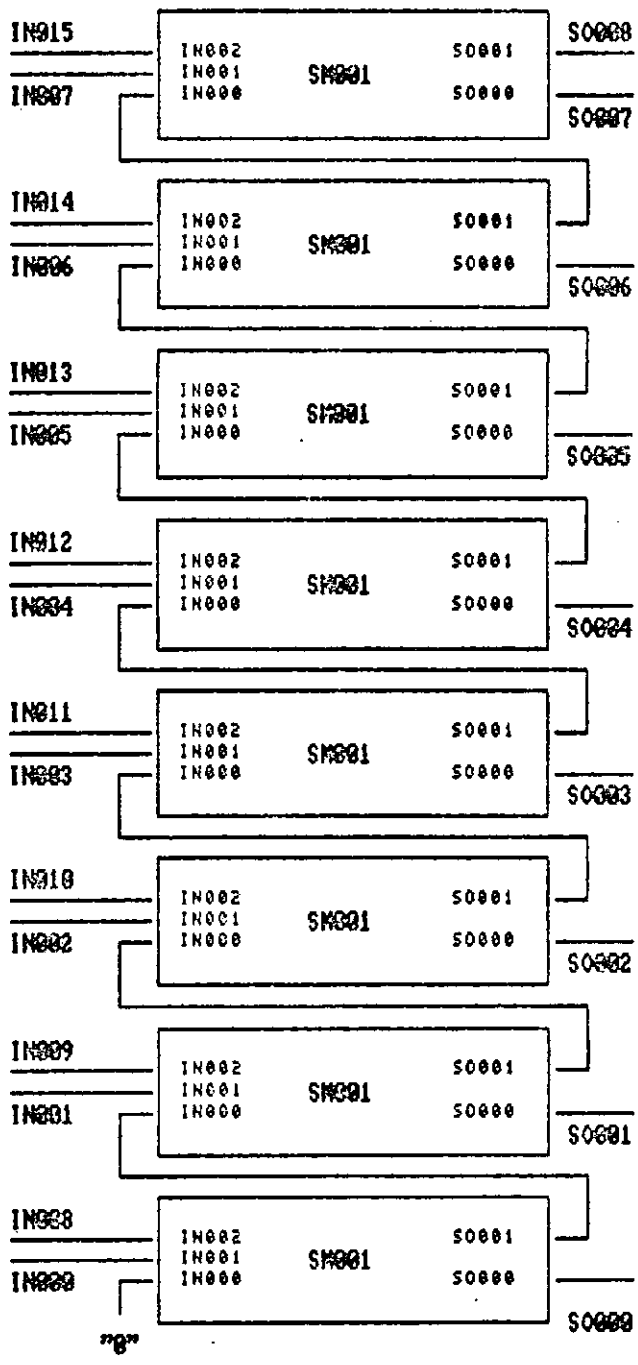


Figura A.5: Nível 1 - bloco SM8

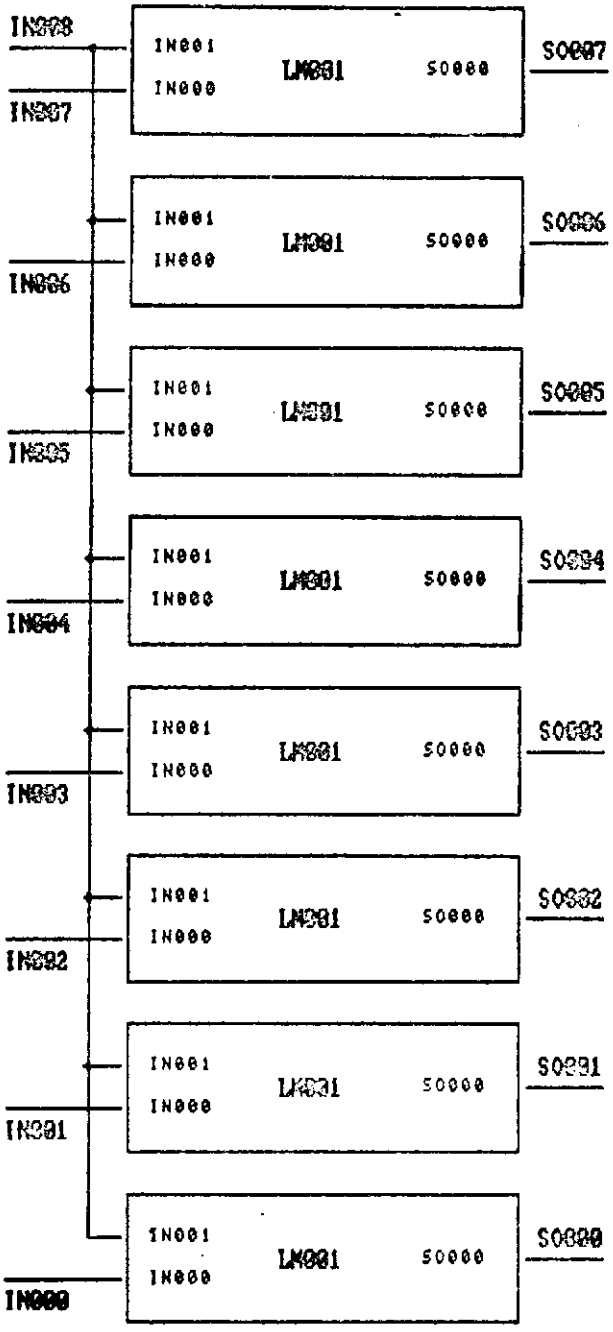


Figura A.6: Nível 1 - bloco LM8

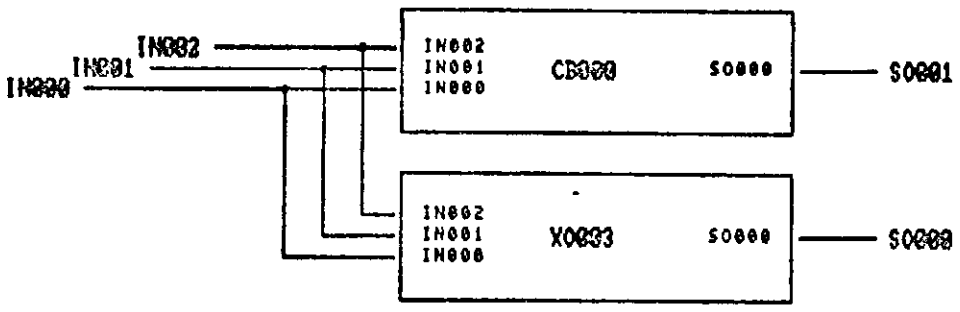


Figura A.7: Nível 2 - bloco SB1

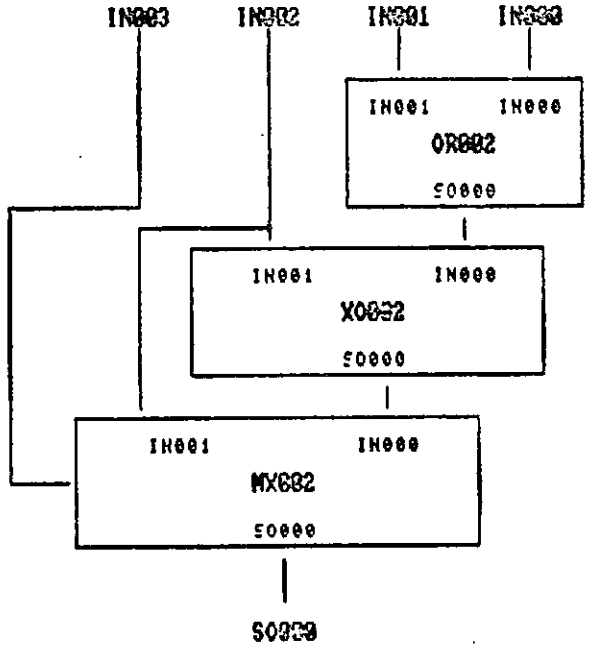


Figura A.8: Nível 2 - bloco MD1

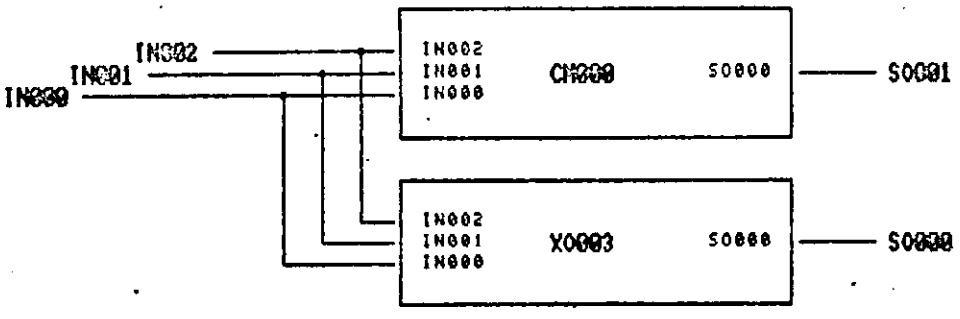


Figura A.9: Nível 2 - bloco SM1

APÊNDICE B

ROTINAS PARA SIMULAÇÕES COM O "SOFTWARE" ELOISE

Rotinas para o Nível 0:

```
behavior(LA008)
/*
Comportamento do latch de 8 bits
Bloco LA8

Entradas:
          IN000
          IN001
          IN002
          IN003
          IN004
          IN005
          IN006
          IN007
          IN100
          IN200
Saidas:
          S0000
          S0001
          S0002
          S0003
          S0004
          S0005
          S0006
          S0007

Entradas/Saidas:
          VSS
          VDD
*/
contents
{
states(8)
if (low(IN200))
   {
   S0000 = HI;
   S0001 = HI;
   S0002 = HI;
   S0003 = HI;
   S0004 = HI;
   S0005 = HI;
```

```
    S0006 = HI;
    S0007 = HI;
    return;
    )
if (high (IN100))
    (
    INTERN(0) = IN000;
    INTERN(1) = IN001;
    INTERN(2) = IN002;
    INTERN(3) = IN003;
    INTERN(4) = IN004;
    INTERN(5) = IN005;
    INTERN(6) = IN006;
    INTERN(7) = IN007;
    )
if (low (IN100))
    (
    S0000 = INTERN(0);
    S0001 = INTERN(1);
    S0002 = INTERN(2);
    S0003 = INTERN(3);
    S0004 = INTERN(4);
    S0005 = INTERN(5);
    S0006 = INTERN(6);
    S0007 = INTERN(7);
    )
)


behavior(LA016)
/*
Comportamento do latch de 16 bits
Bloco 116

Entradas:
          IN000
          IN001
          IN002
          IN003
          IN004
          IN005
          IN006
          IN007
          IN008
          IN009
          IN010
          IN011
          IN012
          IN013
          IN014
          IN015
          IN100
          IN200
```

```
Saidas:
          S0000
          S0001
          S0002
          S0003
          S0004
          S0005
          S0006
          S0007
          S0008
          S0009
          S0010
          S0011
          S0012
          S0013
          S0014
          S0015

Entradas/Saidas:
          VSS
          VDD
*/
contents
{
states(16)
if (low(IN200))
   {
   S0000 = HI;
   S0001 = HI;
   S0002 = HI;
   S0003 = HI;
   S0004 = HI;
   S0005 = HI;
   S0006 = HI;
   S0007 = HI;
   S0008 = HI;
   S0009 = HI;
   S0010 = HI;
   S0011 = HI;
   S0012 = HI;
   S0013 = HI;
   S0014 = HI;
   S0015 = HI;
   return;
   }
if (high (IN100))
   {
   INTERN(0) = IN000;
   INTERN(1) = IN001;
   INTERN(2) = IN002;
   INTERN(3) = IN003;
   INTERN(4) = IN004;
   INTERN(5) = IN005;
```

```
    INTERN(6) = IN006;
    INTERN(7) = IN007;
    INTERN(8) = IN008;
    INTERN(9) = IN009;
    INTERN(10) = IN010;
    INTERN(11) = IN011;
    INTERN(12) = IN012;
    INTERN(13) = IN013;
    INTERN(14) = IN014;
    INTERN(15) = IN015;
    }
if (low (IN100))
    {
    S0000 = INTERN(0);
    S0001 = INTERN(1);
    S0002 = INTERN(2);
    S0003 = INTERN(3);
    S0004 = INTERN(4);
    S0005 = INTERN(5);
    S0006 = INTERN(6);
    S0007 = INTERN(7);
    S0008 = INTERN(8);
    S0009 = INTERN(9);
    S0010 = INTERN(10);
    S0011 = INTERN(11);
    S0012 = INTERN(12);
    S0013 = INTERN(13);
    S0014 = INTERN(14);
    S0015 = INTERN(15);
    }
}


behavior(SB008)
/*
Comportamento do subtrator de 8 bits
Bloco SB8

Entradas:
          IN000
          IN001
          IN002
          IN003
          IN004
          IN005
          IN006
          IN007
          IN008
          IN009
          IN010
          IN011
          IN012
          IN013
```

```
          IN014
          IN015

Saidas:   S0000
          S0001
          S0002
          S0003
          S0004
          S0005
          S0006
          S0007
          S0008

Entradas/Saidas:
          VSS
          VDD
*/
contents
{
int i,j,k;
i=j=0;
plac (i,0,IN000);
plac (i,1,IN001);
plac (i,2,IN002);
plac (i,3,IN003);
plac (i,4,IN004);
plac (i,5,IN005);
plac (i,6,IN006);
plac (i,7,IN007);
plac (j,0,IN008);
plac (j,1,IN009);
plac (j,2,IN010);
plac (j,3,IN011);
plac (j,4,IN012);
plac (j,5,IN013);
plac (j,6,IN014);
plac (j,7,IN015);
k=j-i;
S0000 = lect(k,0);
S0001 = lect(k,1);
S0002 = lect(k,2);
S0003 = lect(k,3);
S0004 = lect(k,4);
S0005 = lect(k,5);
S0006 = lect(k,6);
S0007 = lect(k,7);
S0008 = lect(k,8);
}



behavior(MD008)
/*
Comportamento do modulo de 8 bits
```

```
Bloco MD8

Entradas:
          IN000
          IN001
          IN002
          IN003
          IN004
          IN005
          IN006
          IN007
          IN008

Saidas:
          S0000
          S0001
          S0002
          S0003
          S0004
          S0005
          S0006
          S0007
          S0008

Entradas/Saidas:
          VSS
          VDD
*/
contents
{
int i;
i = 0;
plac (i,0,IN000);
plac (i,1,IN001);
plac (i,2,IN002);
plac (i,3,IN003);
plac (i,4,IN004);
plac (i,5,IN005);
plac (i,6,IN006);
plac (i,7,IN007);
plac (i,15,IN008);
if (low(IN008))
  {
   S0000 = lect(i,0);
   S0001 = lect(i,1);
   S0002 = lect(i,2);
   S0003 = lect(i,3);
   S0004 = lect(i,4);
   S0005 = lect(i,5);
   S0006 = lect(i,6);
   S0007 = lect(i,7);
   }
if (high(IN008))
   {
```

```
    i = (i<0? -i:i);
    S0000 = lect(i,0);
    S0001 = lect(i,1);
    S0002 = lect(i,2);
    S0003 = lect(i,3);
    S0004 = lect(i,4);
    S0005 = lect(i,5);
    S0006 = lect(i,6);
    S0007 = lect(i,7);
    }
}


behavior(SM008)
/*
Comportamento do somador de 8 bits
Bloco SM8

Entradas:
          IN000
          IN001
          IN002
          IN003
          IN004
          IN005
          IN006
          IN007
          IN008
          IN009
          IN010
          IN011
          IN012
          IN013
          IN014
          IN015


Saidas:
          S0000
          S0001
          S0002
          S0003
          S0004
          S0005
          S0006
          S0007
          S0008

Entradas/Saidas:
          VSS
          VDD
*/
contents
```

```
{
int i,j,k;
i=j=0;
plac (i,0,IN000);
plac (i,1,IN001);
plac (i,2,IN002);
plac (i,3,IN003);
plac (i,4,IN004);
plac (i,5,IN005);
plac (i,6,IN006);
plac (i,7,IN007);
plac (j,0,IN008);
plac (j,1,IN009);
plac (j,2,IN010);
plac (j,3,IN011);
plac (j,4,IN012);
plac (j,5,IN013);
plac (j,6,IN014);
plac (j,7,IN015);
k=j+i;
S0000 = lect(k,0);
S0001 = lect(k,1);
S0002 = lect(k,2);
S0003 = lect(k,3);
S0004 = lect(k,4);
S0005 = lect(k,5);
S0006 = lect(k,6);
S0007 = lect(k,7);
S0008 = lect(k,8);
}


behavior(LM008)
/*
Comportamento do operador de truncamento
Bloco LM8

Entradas:
          IN000
          IN001
          IN002
          IN003
          IN004
          IN005
          IN006
          IN007
          IN008

Saidas:
          S0000
          S0001
          S0002
          S0003
```

```
          S0004
          S0005
          S0006
          S0007

Entradas/Saidas:
          VSS
          VDD
*/
contents
{
if (low (IN008))
   {
    S0000 = IN000;
    S0001 = IN001;
    S0002 = IN002;
    S0003 = IN003;
    S0004 = IN004;
    S0005 = IN005;
    S0006 = IN006;
    S0007 = IN007;
   }
 if (high (IN008))
    {
     S0000 = HI;
     S0001 = HI;
     S0002 = HI;
     S0003 = HI;
     S0004 = HI;
     S0005 = HI;
     S0006 = HI;
     S0007 = HI;
    }
  }


Rotinas para o Nível 1:


behavior (LA001)
/*
Comportamento do latch de 1 bit
Bloco LA1

Entradas:
          IN000
          IN100
          IN200


Saidas:
          S0000
```

```
Entradas/Saidas:
          VSS
          VDD
*/
contents
{
states(1)
if (low (IN100))
   {
   S0000 = HI;
   return;
   }
if (low (IN100))
   {
   INTERN(0) = IN000;
   }
if (high (IN100))
   {
   S0000 = INTERN(0);
   }
 }


behavior(SB001)
/*
Comportamento do subtrator elementar
Bloco SB1

Entradas:
          IN000
          IN001
          IN002
Saidas:
          S0000
          S0001

Entradas/Saidas
          VSS
          VDD
*/
contents
{
int i,j,k,l;
i=j=k=0;
plac (i,0,IN000);
plac (j,0,IN001);
plac (k,0,IN002);
l=k-(j+i);
S0000=lect(l,0);
S0001=lect(l,1);
}
```

```
behavior (MD001)
/*
Comportamento da celula modulo elementar
Bloco MD1

Entradas:
          IN000
          IN001
          IN002
          IN003

Saidas:
          S0000
          S0001

Entradas/Saidas:
          VSS
          VDD
*/
contents
{
int i,j,k;
i=j=k=0;
S0001=or2(IN000,IN001);
if (low(IN003))
  {
  S0000=IN002;
  return;
  }
if(high(IN003))
  {
  plac (i,0,IN002);
  plac (j,0,S0001);
  k=i^j;
  S0000=lect(k,0);
  }
}


behavior(SM001)
/*
Comportamento do somador elementar
Bloco SM1

Entradas:
          IN000
          IN001
          IN002
Saidas:
          S0000
          S0001
```

```
Entradas/Saidas
          VSS
          VDD
*/
contents
{
int i,j,k,l;
i=j=k=0;
plac (i,0,IN000);
plac (j,0,IN001);
plac (k,0,IN002);
l=k+j+i;
S0000=lect(l,0);
S0001=lect(l,1);
}


behavior (LM001)
/*
Comportamento da celula truncamento elementar
Bloco LM1

Entradas:
          IN000
          IN001

Saidas:
          S0000

Entradas/Saidas:
          VSS
          VDD
*/
contents
{
S0000 = or2 (IN000,IN001);
}


Rotinas para o Nível 2


behavior(CB000)
/*
Comportamento da celula CARRY
para o subtrator elementar

Entradas:
          IN000
          IN001
```

```
          IN002

Saidas:
          S0000

Entradas/Saidas:
          VSS
          VDD
*/
contents
{
var i,j,k,l;
i=j=k=l=0;
i = and3(IN000,IN001,IN002);
j = or2(IN001;IN002);
k = inv(IN000);
l = and2(k,j);
S0000 = or2(l,i);
}


behavior (CM000)
/*
Comportamento da celula CARRY
para o somador elementar

Entradas:
          IN000
          IN001
          IN002

Saidas:
          S0000

Entradas/Saidas:
          VSS
          VDD
*/
contents
{
var i,j,k,l;
i=j=k=l=0;
i = inv(IN000);
j = and3(i,IN001,IN002);
k = or2(IN001,IN002);
l = and2(IN000,k);
S0000 = or2(l,j);
}


behavior(X0003)
/*
```

```
Comportamento da funcao OU EXCLUSIVO
de tres entradas

Entradas:
          IN000
          IN001
          IN002

Saidas:
          S0000

Entradas/Saidas:
          VSS
          VDD
*/
contents
{
var i,j,k,l;
i=j=k=l=0;
plac (i,0,IN000);
plac (j,0,IN001);
plac (k,0,IN002);
l = i^j^k;
S0000 = lect(l,0);
}



behavior(OR002)
/*
Comportamento da celula OR002

Entradas:
          IN000
          IN001

Saidas:
          S0000

Entradas/saidas:
          VDD
          VSS

*/
contents
{
S0000 = or2(IN000,IN001);
}



behavior(MX002)
/*
Comportamento da celula MX002
```

```
Entradas:
          IN000
          IN001
          IN002

Saidas:
          S0000

Entradas/saidas:
          VDD
          VSS

*/
contents
{
if(low(IN002))
  {
  S0000 = IN000;
  }
if(high(IN002))
  {
  S0000 = IN001;
  }
}



behavior(X0002)
/*
Comportamento da celula X0002

Entradas:
          IN000
          IN001

Saidas:
          S0000

Entradas/Saidas:
          VDD
          VSS
*/
contents
{
int i,j,k;
i=j=k=0;
plac (i,0,IN000);
plac (j,0,IN001);
k=i^j;
S0000 = lect(k,0);
}
```

APÊNDICE C

SIMULAÇÕES COM O SOLO 1400

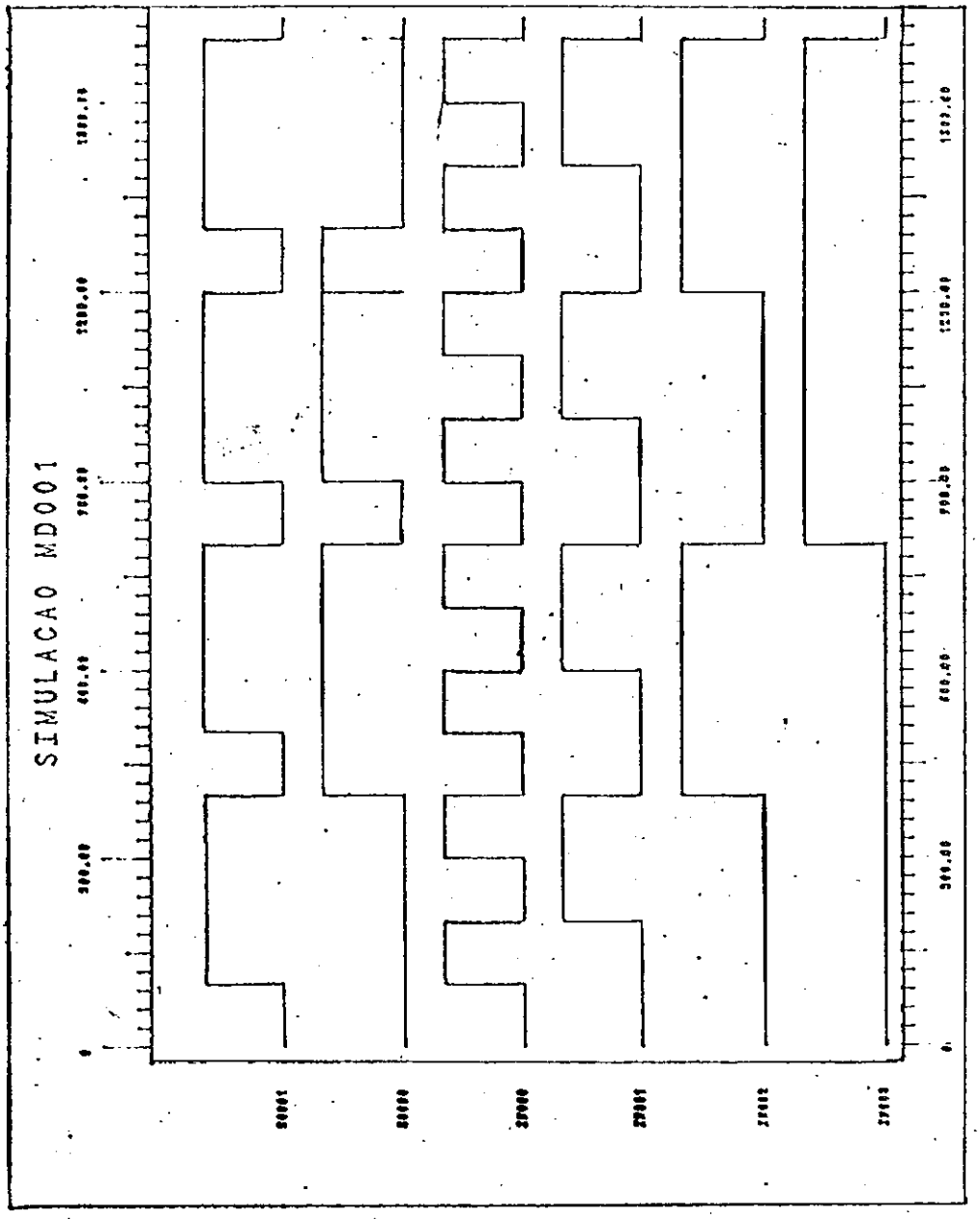


Figura C.1: Simulação do bloco MD1.

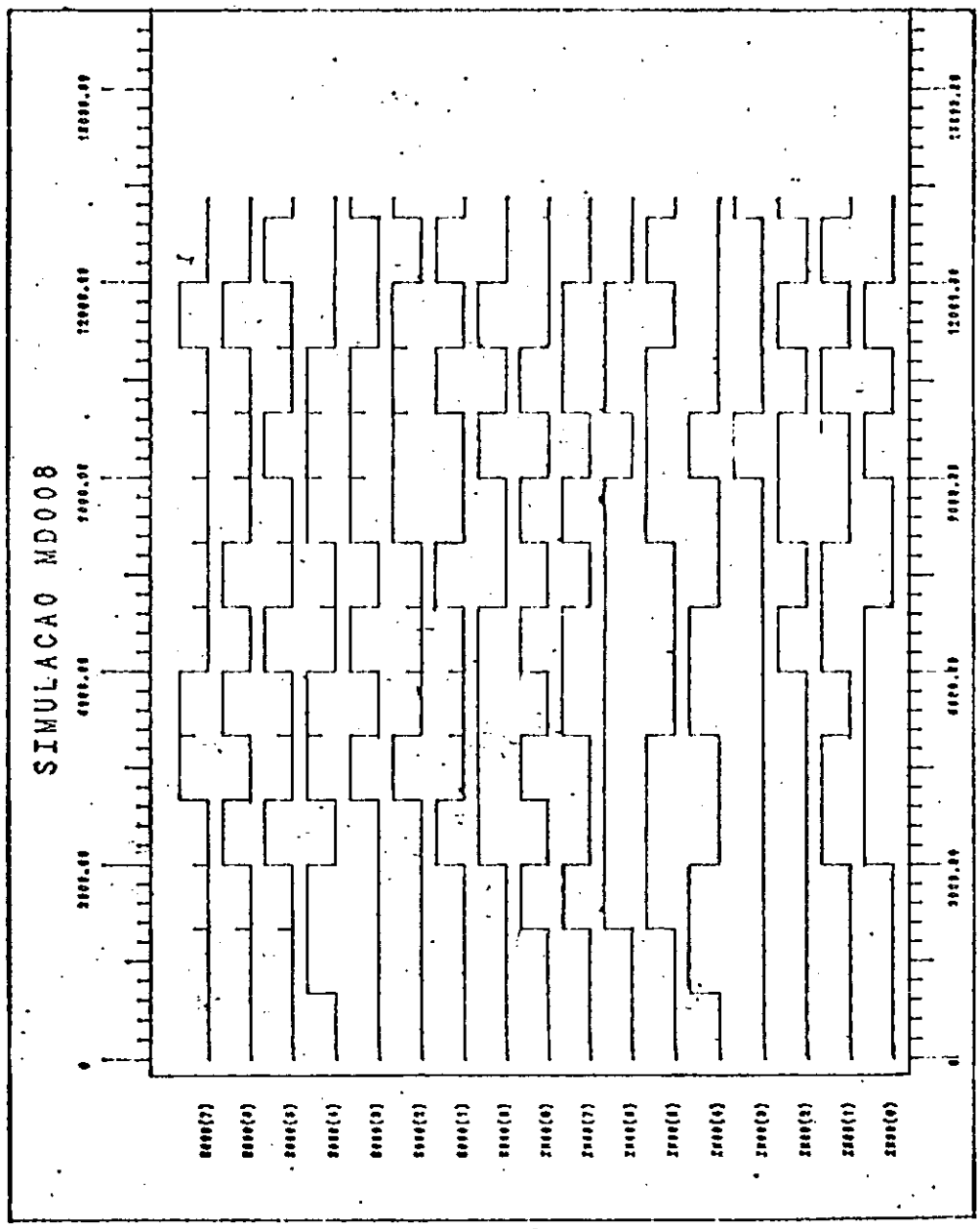
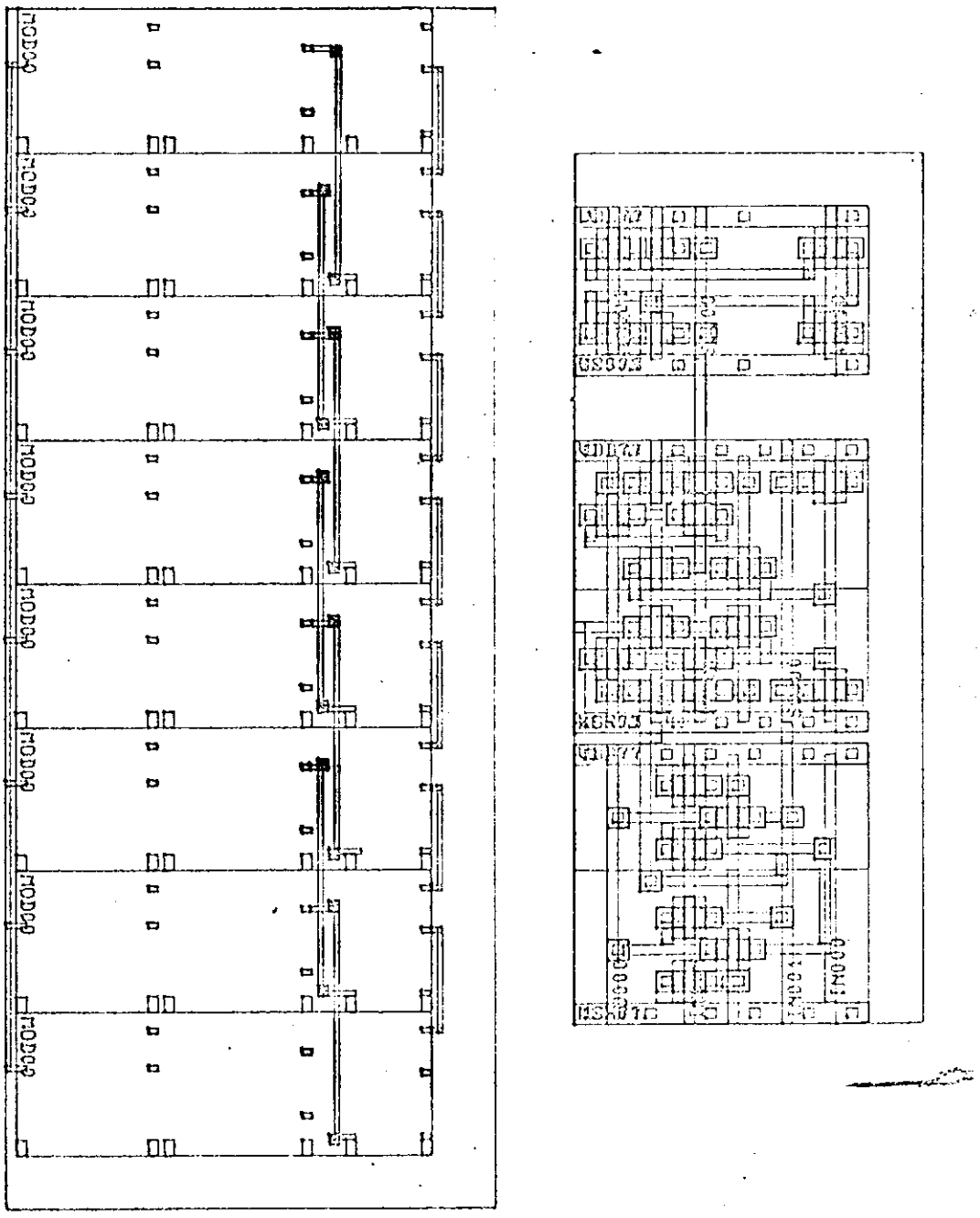


Figura C.2: Simulação do bloco MD8.

APÊNDICE D

"LAYOUTS" PARA O BLOCO MD8



(a) (b)

Figura D.1: "Layout" gerado com o ALLIANCE.
(a) Bloco MD8.
(b) Bloco MD1.

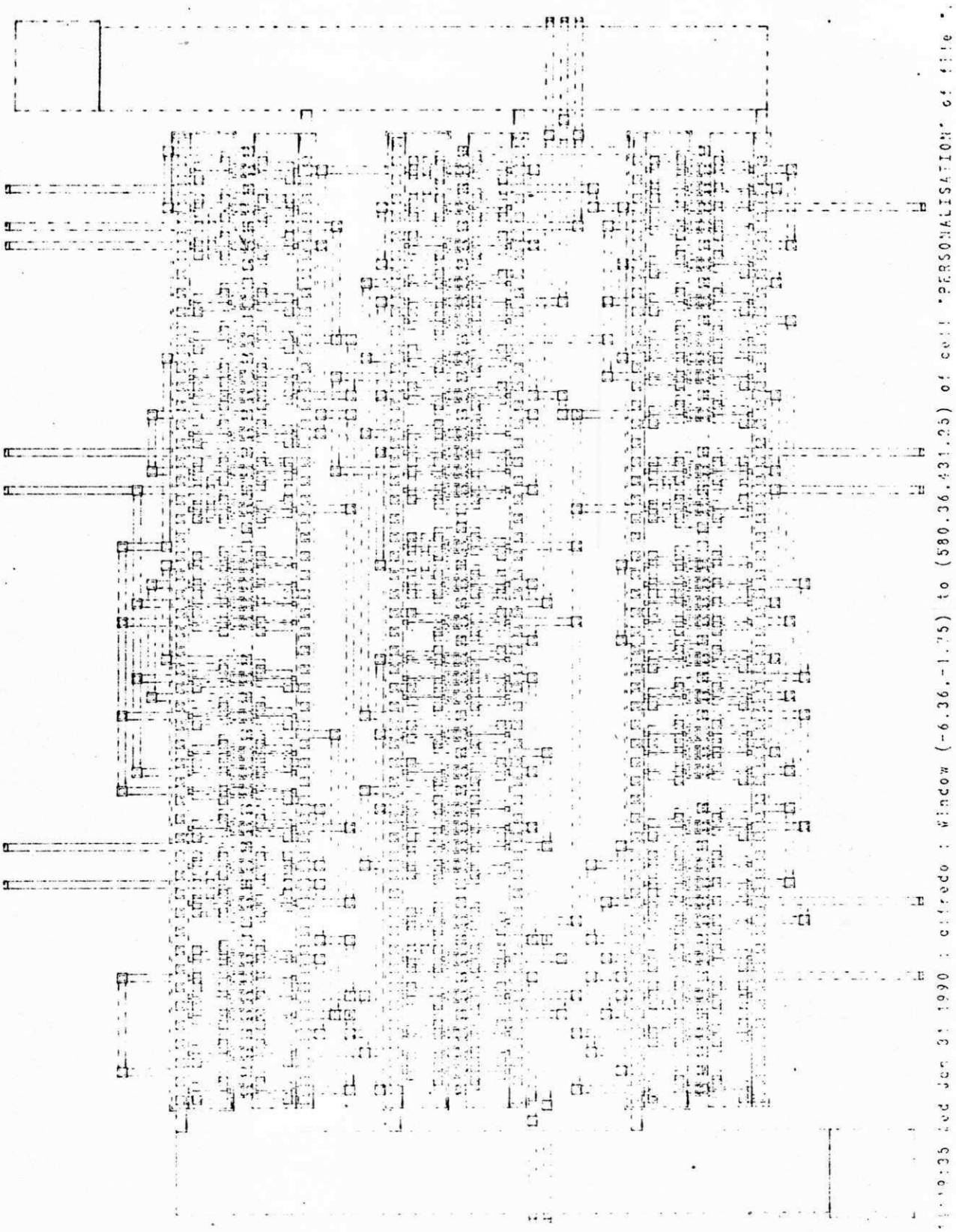


Figura D.2: "Layout" do bloco MD8 gerado com o SOLO 1400.

APÊNDICE E

"LAYOUTS" DE CÉLULAS DO OPERADOR DE ROBERTS

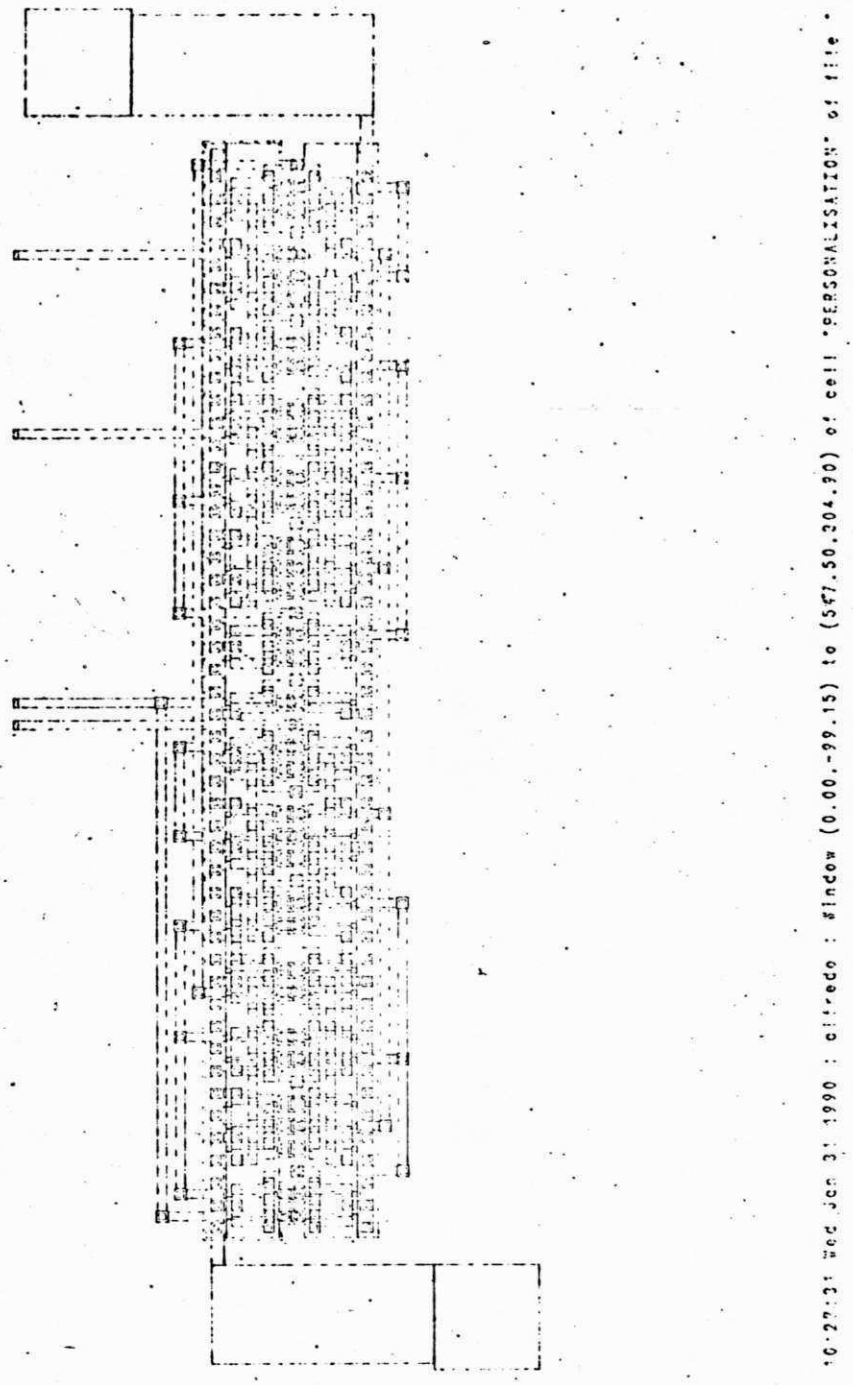


Figura E.1: "Layout" da célula LA1 gerado com o SOLO 1400.

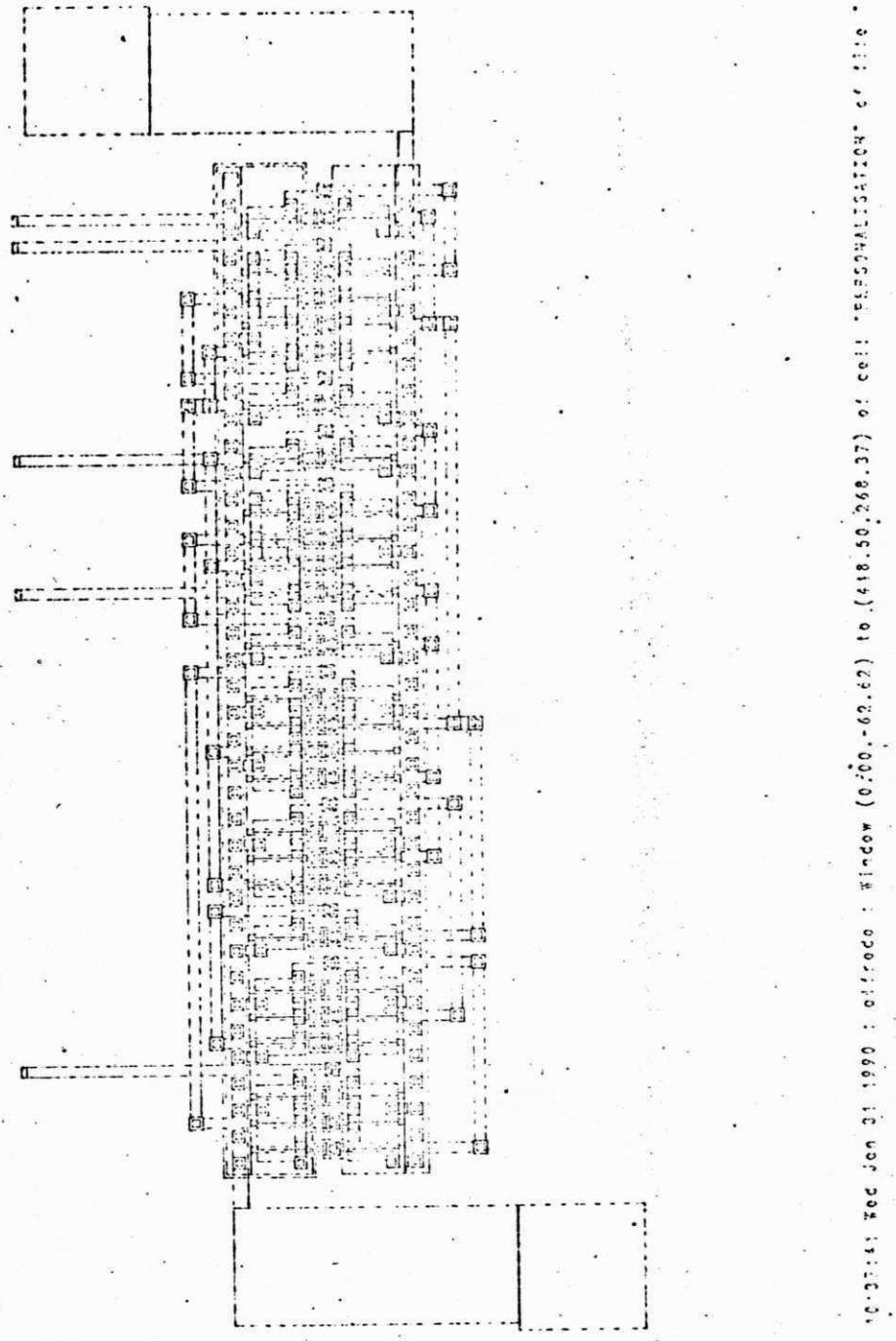


Figura E.2: "Layout" da célula SB1 gerado com o SOLO 1400.

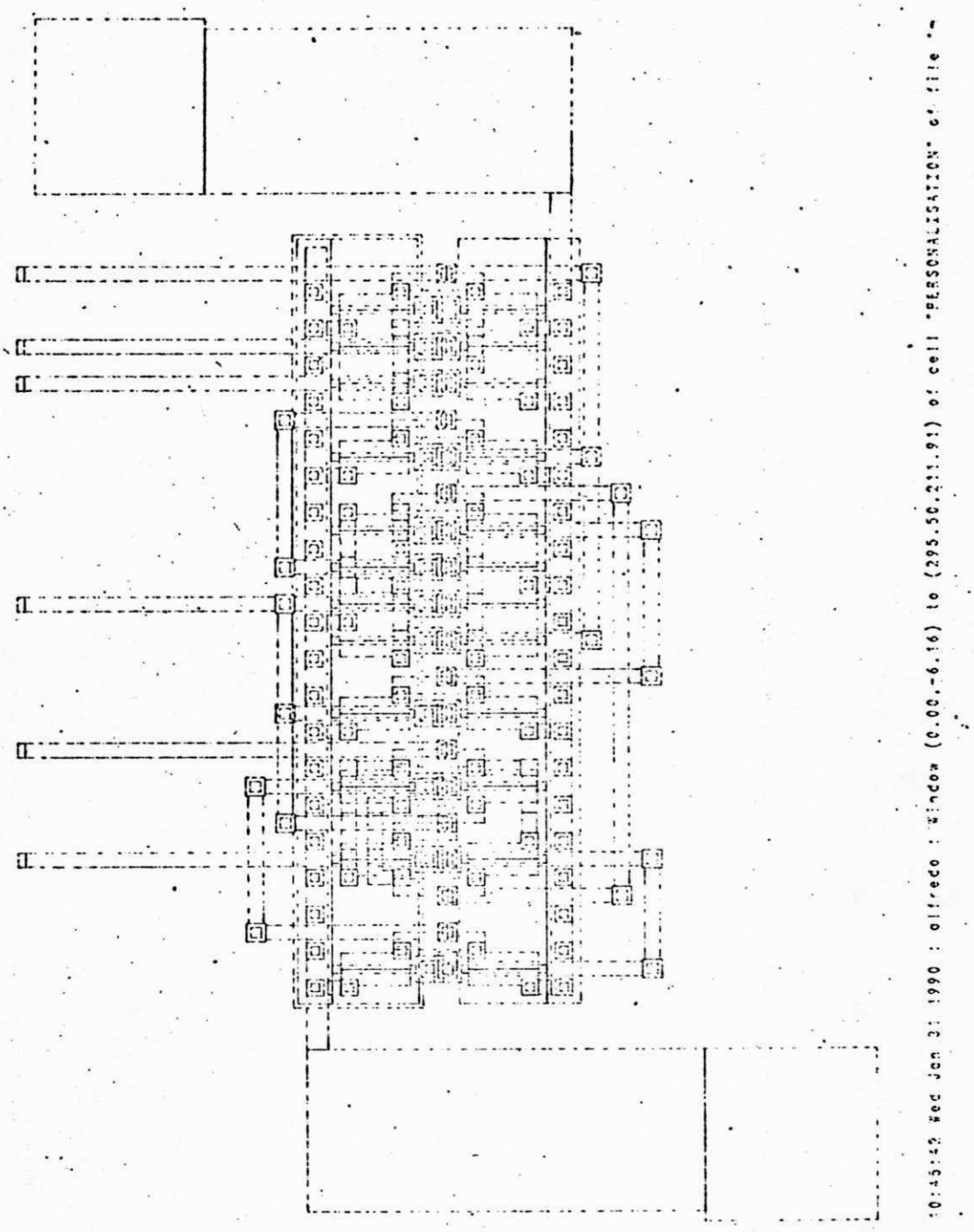


Figura E.3: "Layout" da célula MD1 gerado com o SOLO 1400.

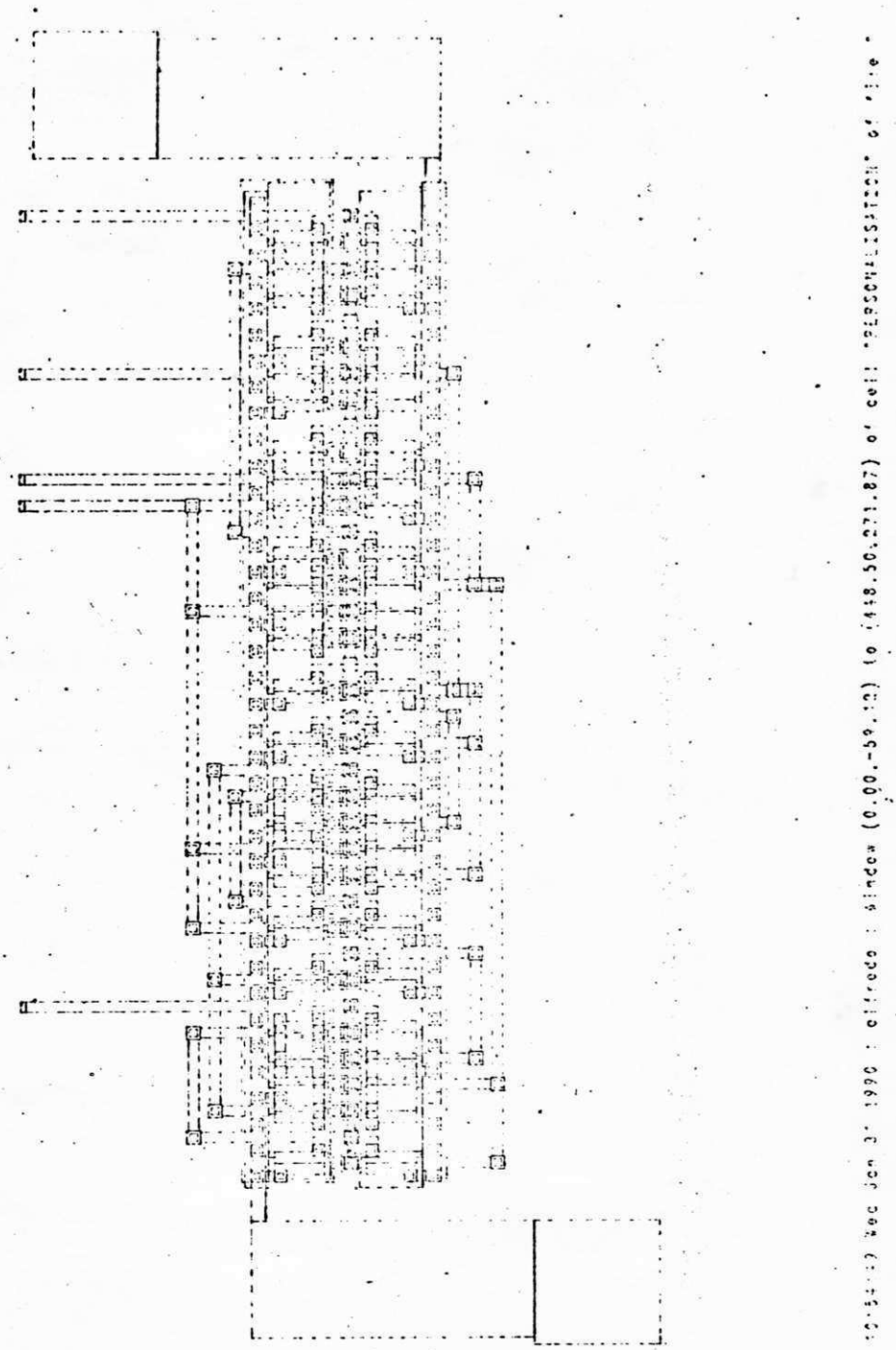


Figura E.4: "Layout" da célula SM1 gerado com o SOLO 1400.