UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

VALIDAÇÃO DE PROTÓTIPO E ANÁLISE DE FALHAS NO TESTE COM FEIXE DE ELÉTRONS: UM ESTUDO VISANDO A SUA AUTOMAÇÃO

FABIAN LUIS VARGAS

Dissertação submetida como requisito parcial
para obtenção do grau de
Mestre em Ciência da Computação

Prof. Ricardo Augusto da Luz Reis
Orientador

Drª Meryem Marzouki
Laboratoire TIM3-INPG
Co-orientador

Porto Alegre, Setembro de 1991.

Vargas, Fabian Luis
Validação de Protótipo e Análise de Falhas no Teste com Feixe de Elétrons: Um Estudo Visando a sua Automação. Fabian Luis Vargas. - Porto Alegre: CPGCC da UFRGS, 1991.
155p.: il.

Dissertação (mestrado) - Universidade Federal do Rio Grande do sul, Curso de Pós-Graduação em Ciência da Computação, Porto Alegre, 1991. Orientador: Reis, Ricardo Augusto da Luz.

Dissertação: Teste com Feixe de Elétrons, Validação de Protótipos, Análise de Falhas, Projeto Visando a Testabilidade.

AGRADECIMENTOS

Este trabalho descreve pesquisas desenvolvidas pelo autor na Universidade Federal do Rio Grande do Sul (UFRGS) e no Laboratório TIM3/INPG, Institut National Polytechnique de Grenoble, França. Em ambas universidades, o autor está endividado com inúmeras pessoas. Meryem Marzouki, Didie Conard, Jeremy D. Russell e Jacques Laurent, do laboratório TIM3, trabalharam em conjunto com o autor em vários capítulos desta dissertação e cujo tempo e esforço despendidos, apontando problemas e soluções ao longo do trabalho foram profundamente apreciados.

Um agradecimento especial é dado a duas pessoas: primeiramente a Ricardo Augusto da Luz Reis, coordenador do Curso de Pós-Graduação em Ciência da Computação e orientador do autor, cuja oportunidade de realização deste trabalho, bem como o esforço em supervisioná-lo foram encorajadores e fundamentais e por último, mas não menos importante, Bernard Courtois, diretor do Laboratório TIM3, pela aceitação do autor naquele laboratório durante a realização de parte deste trabalho.

Agradecimentos também são dados a Ingrid Jansch Pôrto e Tiaraju Vasconcellos Wagner, participantes da banca de avaliação, e a Philippe Olivier Navaux, coordenador brasileiro do Convênio CAPES-COFECUB, pelo apoio dado junto aos órgãos federais durante o período de avaliação do processo de pedido de bolsa para estudos no exterior.

Aos meus pais, Daniel e Salete,
cujo amor dedicado aos filhos é
algo assombroso,
e à Roseana, o novo amor da
minha vida...

SUMÁRIO

PARTE II
EXPERIMENTOS PRÁTICOS EM VALIDAÇÃO DE PROTÓTIPOS

LISTA DE ABREVIATURAS

A/D	Analógico/Digital
CAD	Computer Aided Design
CI	Circuitos Integrados
ES	Elétrons Secundários
GME	Grupo de Microeletrônica
INPG	Institut National Polytechnique de Grenoble
MEV	Microscópio Eletrônico de Varredura
SEM	Scanning Electron Microscope
TFE	Testador com Feixe de Elétrons
VLSI	Very Large Scale Integration

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE FOTOGRAFIAS

Fotografia 6.4 (a) Saídas do bloco analógico C3, C4, C5 em Modo de Mapeamento do Estado Lógico; (b) e (c) instantes de précarga e avaliação do capacitor C4, nos quantizadores grosseiros.

Fotografia 6.5 (a) e (b) Imagens obtidas na saída do bloco analógico com a entrada analógica colocada em 1,5 e 2,0V, respectivamente; (c) imagem da subtração (a) - (b), em contraste de tensão, mostrando as discrepâncias em áreas claras.

RESUMO

O trabalho aqui apresentado descreve algumas pesquisas em teste de circuitos integrados. Estas pesquisas consistem, por um lado, na análise de falhas e por outro, na validação de protótipos, ambas fazendo uso de técnicas de teste com feixe de elétrons.

A primeira parte deste trabalho apresenta uma revisão dos princípios do teste com feixe de elétrons, bem como descreve as pesquisas correntemente em desenvolvimento no laboratório TIM3-INPG. Também são abordados temas como o tratamento de imagem em contraste de potencial e projeto visando a testabilidade de circuitos no teste com feixe de elétrons. Quanto a este último assunto, sua inclusão neste trabalho visou apresentar, àqueles que trabalham na área de projetos de circuitos, desconhecedores dos problemas do MEV, idéias de como realizar seu projeto a fim de tornar a tarefa de depuração do protótipo pelo feixe de elétrons o mais fácil possível.

A segunda parte descreve experimentos práticos na área de validação de protótipos, onde duas técnicas pertinentes foram utilizadas e o estudo de um caso real foi apresentado. A primeira técnica é baseada na adaptação de uma ferramenta de comparação de múltiplas imagens adjacentes, que foi originalmente desenvolvida para o processo de análise de falhas. A segunda técnica utilizada faz uso de um sistema especialista que, baseado no

conhecimento adquirido do circuito, gera o diagnóstico automático de falha. Os desempenhos destas duas ferramentas são apresentados e discutidos, bem como é fornecido o diagnóstico de falha para o circuito protótipo utilizado.

Como conclusão, são propostos futuros desenvolvimentos no processo de validação de protótipo. Estes melhoramentos objetivam tanto a completa automação do processo quanto o enriquecimento da informação provida no final do processo de diagnóstico de falha, de forma a obter-se um ambiente de teste para validação de protótipos apresentando um alto grau de integração e automação.

Palavras chave: Teste com Feixe de Elétrons, Validação de Protótipos, Análise de Falhas.

ABSTRACT

The work reported herein describes some IC testing research. This research concerns on one hand, failure analysis and on the other hand IC prototype validation, both making use of e-beam testing techniques.

The first part of this work presents a review of e-beam testing as well as describes the researches currently in progress at the TIM3-INPG Laboratory. Subjects like voltage contrast image treatment and design for testability in e-beam testing are also discussed. Considering the last theme, it was included in this work in order to provide to the IC designers, whose knowledge about the SEM problems is not enough, some ideas on the way of how to accomplish their design to make the prototype validation process as easy as possible.

The second part describes practical experiments in the prototype validation domain, where two approaches were used and a real case study was presented. The first approach is based on the multiple adjacent images comparison process adaptation, firstly developed to be used in the failure analysis process. The second technique makes use of an expert system, based on the acquired knowledge of the device under test in order to provide the fault diagnosis. The performances of these two approaches are presented

and discussed, as well as, the fault diagnosis to the prototype circuit is presented.

As conclusion, it is proposed further developments in the prototype validation approach. These improvements deal with the automation of the entire process as well as the enhancement of the information provided at the end of the fault diagnosis process, in order to obtain a testing environment for prototype validation with high integration and automation degrees.

Key words: Electron Beam Testing, Prototype Validation, Failure Analysis.

INTRODUÇÃO

Com o aumento constante da complexidade de circuitos integrados (CI) VLSI, a tarefa de diagnóstico de falha com testadores tradicionais torna-se cada vez mais difícil. Métodos de teste clássicos tem sido melhorados e novas estratégias tem sido propostas a fim de se manter os tempos de desenvolvimento de CI dentro de padrões razoáveis e de se assegurar e manter a qualidade destes. Porém, apesar das técnicas de CAD e de teste serem aplicadas aos CI, os protótipos de tais circuitos não correspondem aos modelos previamente definidos. As técnicas de "built-in test" proporcionam maior testabilidade, mas por outro lado, tornam a tarefa de projeto ainda mais complexa. Assim, no final do processo, ainda é necessário se testar o funcionamento do circuito protótipo através do monitoramento de determinados nodos internos. Entretanto, os testadores com feixe de elétrons (TFE) podem proporcionar um rompimento no manuseio da complexidade destes circuitos [FAZ81]. Neste enfoque, as propriedades interessantes dos TFE fundamentam-se na capacidade destes serem usados como uma fina ponteira, alinhada e orientada sobre o objeto e sem contato físico, a qual, sob certas condições, pode ser operada em um modo "passivo" para medir sinais internos de um CI de uma forma não destrutiva e não capacitiva. Além do mais, o feixe de elétrons pode ser posicionado de uma maneira rápida e precisa sobre qualquer ponto do circuito, até mesmo sobre áreas passivadas, pois este acesso é possível tanto por acoplamento capacitivo, junto à superfície do óxido, quanto através da formação de um canal condutor entre o nodo desejado e a superfície do óxido (neste caso, elétrons com mais alta energia: 10KV, são necessários, podendo-se desta forma danificar o circuito. Normalmente são utilizados elétrons de baixa energia, isto é, de 1,5 a 2KV.). Assim, o TFE representa a única alternativa para se substituir os métodos clássicos de teste, tais como as ponteiras mecânicas, as quais tem sido usadas até o presente momento para a obtenção das medidas de sinais internos em circuitos VLSI.

É neste contexto que o CPGCC começou a investir na área do teste com feixe de elétrons. O primeiro trabalho data de 1986, com C. J. O.

Hurtado [HUR86], através da implementação de modificações em um microscópio eletrônico CAMBRIDGE [CAM72], modelo padrão, de modo a adaptá-lo ao teste de circuitos integrados. Além disso, seu trabalho também incluiu algumas experiências práticas na área de validação de protótipos através do teste de alguns circuitos NMOS projetados pelo GME.

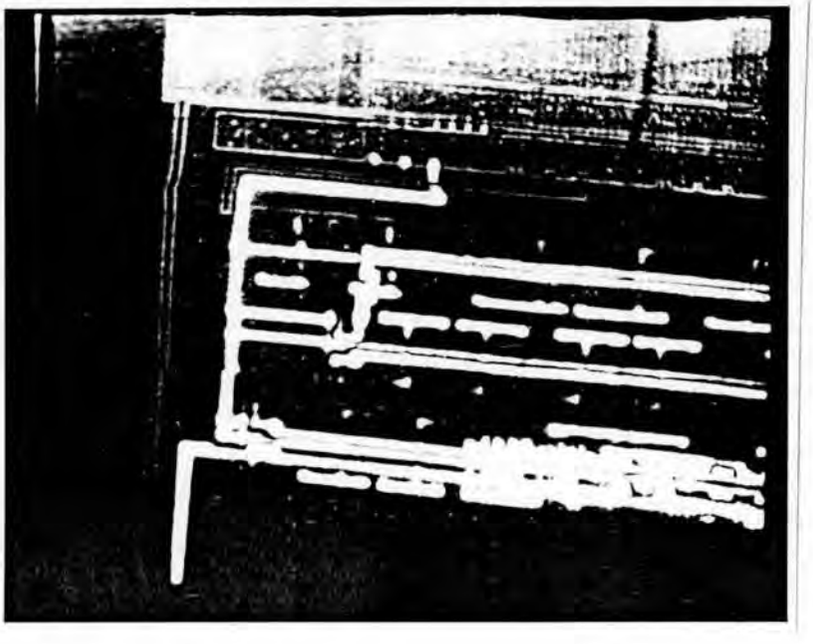
O trabalho de Hurtado teve continuidade três anos depois, em 1989, quando F. L. Vargas apresentou um trabalho individual [VAR90], recolocando em operação o MEV CAMBRIDGE e realizando a validação de dois circuitos protótipos implementados pelo GME (ver fotografias 1 e 2) [REI90]. A seqüência deste trabalho foi dada através de um estágio de seis meses no laboratório TIM3/INPG (França), onde o autor teve a oportunidade de participar do desenvolvimento de alguns projetos [VAR91]. O resultado destes trabalhos, que tiveram como objetivo principal a formação de recursos humanos e o desenvolvimento de uma capacitação técnica na área de testes com feixe de elétrons dentro do CPGCC, gerou a presente dissertação de mestrado.

Assim, a dissertação apresentada a seguir descreve os princípios do teste com feixe de elétrons e algumas das técnicas de validação de protótipos e de análise de falhas a ele aplicadas. O objetivo final é a obtenção, a partir da análise de resultados práticos destas técnicas de depuração de CI, de um ambiente integrado de teste, apresentando um alto grau de automação dos processos de aquisição das informações do circuito testado e de diagnóstico de falha.

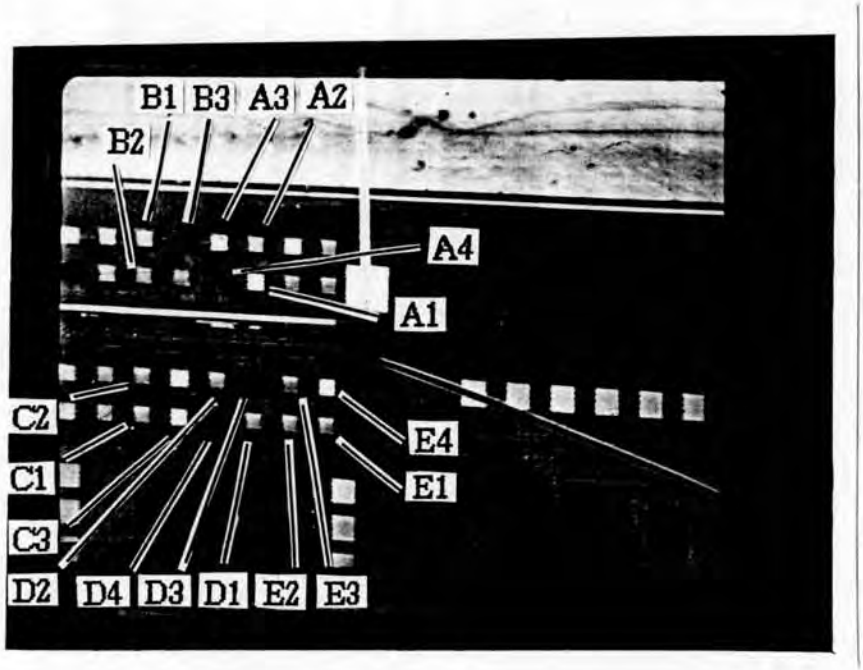
Com o objetivo de separar o estudo teórico daquele prático, o trabalho apresentado a seguir é dividido em duas partes. A primeira, apresenta o teste com feixe de elétrons, suas características principais e suas limitações (capítulo 1), um resumo das técnicas de tratamento de imagem em contraste de potencial, comentando os vários métodos de filtragem e de reconhecimento de padrões em imagens a partir da observação do circuito pelo feixe de elétrons (capítulo 2), um estudo dos processos de validação de protótipos e de análise de falhas, através da apresentação das técnicas atualmente em desenvolvimento no laboratório TIM3-INPG de Grenoble, França, e de um consórcio internacional, formado por alguns países europeus (capítulo 3). Por último, no capítulo 4, é abordado o problema do projeto visando a testabilidade no teste com feixe de elétrons, visando fornecer àqueles que trabalham na área de projeto, desconhecedores dos problemas do MEV, idéias de como realizar o

seu trabalho a fim de tornar a tarefa de depuração do circuito protótipo pelo feixe de elétrons o mais fácil possível.

A segunda parte desta dissertação apresenta um trabalho prático, onde duas técnicas para validação de protótipos são utilizadas, seus desempenhos obtidos e os resultados discutidos, bem como é fornecido o diagnóstico de falha para um circuito protótipo, implementado pelo GME (capítulo 6). No capítulo 7, são apresentados melhoramentos para uma destas técnicas de validação de protótipo. Estes melhoramentos visam o aperfeiçoamento do processo de diagnóstico de falha, de forma a obter um ambiente de teste com feixe de elétrons integrado e automatizado.



Fotografia 1 Vista geral do circuito conversor de lógica digital/códigos "HDB3/AMI". Em funcionamento, a parte codificadora do circuito [REI89].



Fotografia 2 Vista geral do circuito-teste das células da biblioteca de "gate-array", mostrando em detalhes as células: "flip-flop D", (A); "nand" 2 entradas, (B); "and" 2 entradas, (C); Decodificador, (D); "and" 3 entradas, (E).

PARTE I

O TESTE COM FEIXE DE ELÉTRONS: TEORIA E APLICAÇÕES

1 PRINCÍPIOS DO TESTE COM FEIXE DE ELÉTRONS

1.1 Introdução

Um testador com feixe de elétrons é contituído por um microscópio eletrônico de varredura (MEV) conectado a um sistema de CAD e a unidades de tratamento de imagem, processamento de dados e de controle do CI, conforme pode ser visto na figura 1.1.

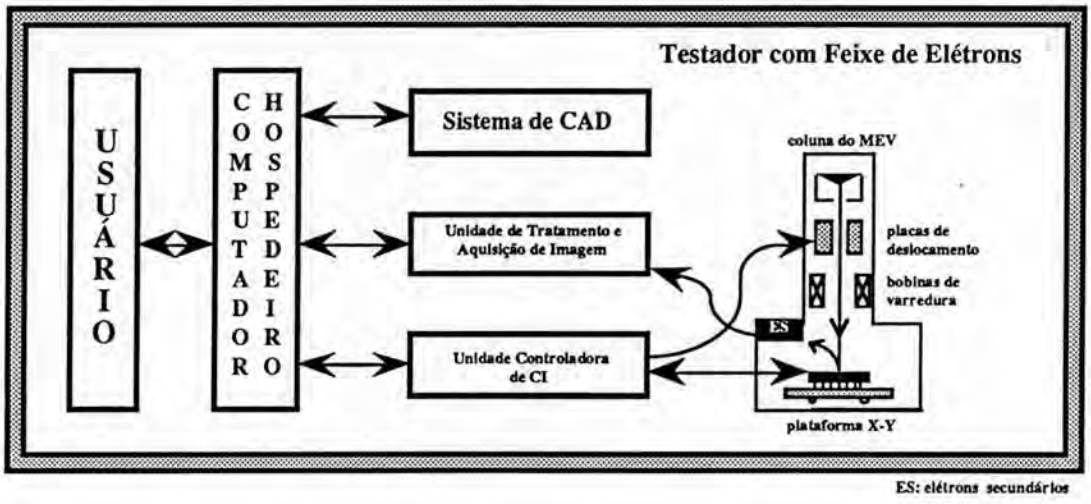


Figura 1.1 Vista geral de um Sistema Testador de Feixe de Elétrons

Dentro da câmara de vácuo do MEV, os circuitos são montados e excitados pelo controlador de CI. Para se realizar uma medida, o feixe de elétrons primários é orientado em direção à superfície do circuito através de um sistema de lentes. Os elétrons secundários, de baixa energia, gerados neste processo, são absorvidos por um coletor sobre o qual uma alta tensão positiva é aplicada. O sinal resultante amplificado é então mostrado em um tubo de raios catódicos. Para medidas de alta freqüência (acima de 100KHz), a coluna de feixe de elétrons é equipada com placas de desvio, controladas por uma unidade temporizadora sincronizada com o sinal de relógio do circuito protótipo. Este procedimento é necessário porque a largura de banda do tubo

de raios catódicos é limitada, fazendo com que a relação sinal-ruído da imagem gerada seja bastante baixa.

Os sistemas de CAD são usados nas fases de projeto e de teste de CI. A unidade de processamento de dados é usada para manipular a informação provida por um lado pelo MEV e por outro pelo sistema de CAD, e a unidade de processamento de imagem é encarregada da aquisição e do tratamento das informações fornecidas pelo MEV. A última unidade é o controlador de CI, cuja tarefa é controlar as entradas e as saídas primárias do CI através da simulação do seu ambiente de operação e da análise dos sinais de saída.

1.2 Fenômeno do Contraste de Tensão

As tensões geradas no interior de circuitos integrados, que são operados dentro de um microscópio eletrônico, modulam o sinal secundário detectado. Este fenômeno é chamado de "contraste de tensão" e é causado pela influência do potencial do ponto de emissão de elétrons secundários, no dispositivo, através da interação do seu campo elétrico com o do detector de elétrons secundários.

Devido à alta tensão do detector (300V), quase todos os elétrons secundários gerados a partir de trilhas conectadas à massa do sistema são captados pelo coletor. Se, entretanto, a superfície do circuito apresenta trilhas com potencial de 5V, fortes campos eletrostáticos são gerados ao redor de sua vizinhança. Estes campos próximos interagem com o do detector, podendo anular completamente o seu efeito.

A figura 1.2 mostra este fenômeno. Os elétrons secundários gerados a partir de trilhas conectadas à massa são expostos a um campo eletrostático que os impulsiona em direção ao coletor. Entretanto, os elétrons secundários provenientes das conexões em potencial 5V tem que passar através de um campo de retardo, o qual força alguns deles a retornarem em direção às trilhas de origem. Desta forma, o detector recebe uma menor quantidade de elétrons secundários a partir de conexões em 5V e, como resultado, estes pontos são mostrados como sendo mais escuros do que aqueles em potencial de 0V.

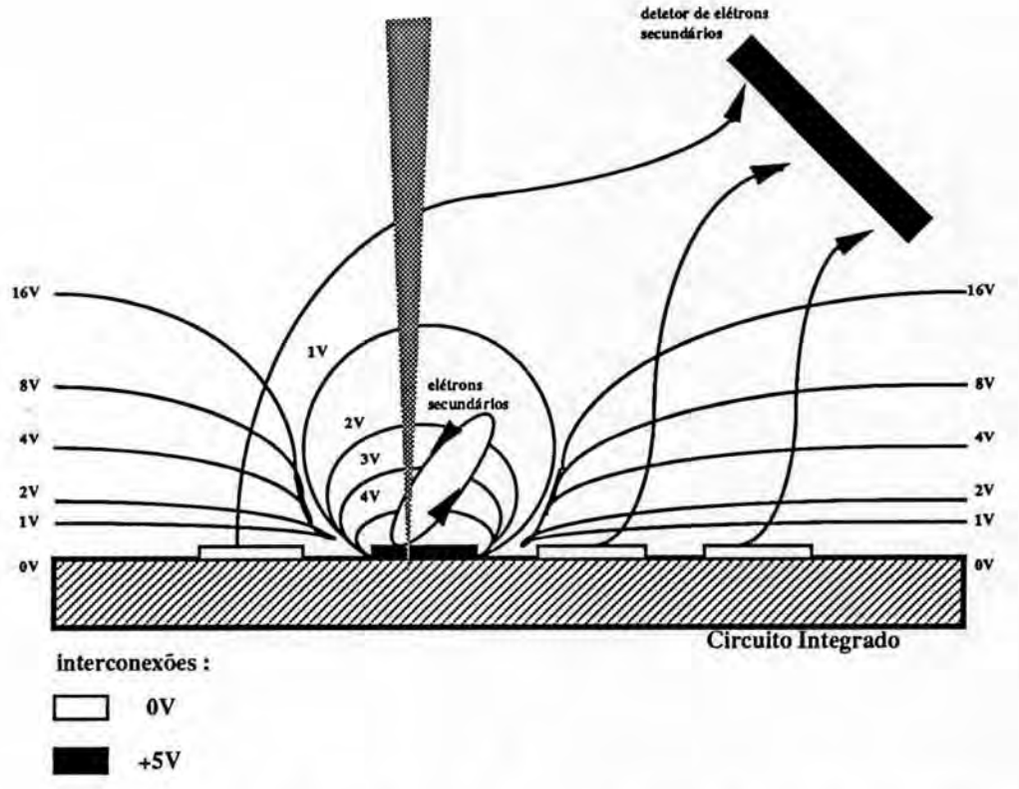


Figura 1.2 Esquema da distribuição de potencial sobre trilhas de 0V e 5V.

1.3 Técnicas de Observação

Considerando o teste com feixe de elétrons, vários métodos de observação de um dispositivo podem ser obtidos. Isto é, um feixe de elétrons pode ser usado como sendo uma ponteira para medição ou injeção de sinais internos, sob uma forma contínua ou pulsada, no modo de varredura fixo ou variável. Estes métodos de observação são descritos como se segue:

Modo de Forma de Onda em Tempo Real: quando o modo de varredura é cortado, o feixe de elétrons é fixado sobre o eletrodo de um dispositivo; neste caso, a corrente de elétrons secundários varia (na mesma proporção) de acordo com a variação da tensão: um sinal de nível alto para uma tensão baixa e vice-versa.

Modo de Código de Tensão: se o feixe de elétrons é posto no modo de varredura usual, a corrente de elétrons secundários varia de acordo com as respectivas variações de tensão, e como conseqüência, pode-se

observar faixas no vídeo na freqüência de varredura da TV. Se a freqüência de operação do CI coincide com um múltiplo inteiro de um inteiro da freqüência de varredura de uma linha do vídeo ou da freqüência de varredura de um quadro, a imagem permanece fixa na tela. Com este modo de observação em tempo real, vários MHz do sinal medido podem ser observados, mas sempre limitados a uma única seqüência de tempo.

Modo de Imagem Estroboscópica: para se obter este modo, antes de tudo, a tensão do dispositivo deve ser periódica, isto é, a mesma fase deve aparecer a cada ciclo. Se os pulsos de elétrons tocarem a superfície do circuito sempre na mesma fase do dispositivo, os pulsos de elétrons encontrarão o dispositivo sempre no mesmo estado interno.

As fotografias 1.1 e 1.2 ilustram um exemplo de aplicação das técnicas de observação em código de tensão e em imagem estroboscópica, repectivamente. O exemplo escolhido é o circuito CONVERSO, um conversor A/D de 4 bits, que foi utilizado durante a parte prática do trabalho de validação de protótipos e a ser descrito, em detalhes, no capítulo 6.

Modo de Forma de Onda Estroboscópica: este modo pode ser obtido através do corte da varredura do feixe de elétrons, isto é, fixando-o sobre um determinado ponto e ao mesmo tempo, deslocando-se a diferença de fase entre as freqüências do pulso de elétrons e do dispositivo, respectivamente.

Modo de Mapeamento do Estado Lógico: este modo é usado para varrer linhas verticais no tempo. No modo de imagem estroboscópica usual, a diferença de fase entre os pulsos do feixe de elétrons e dos sinais internos do dispositivo é fixa durante a tomada de uma fotografia. Mas no mapeamento do estado lógico, a fase é fixa durante a varredura horizontal, e é deslocada por um dado incremento na próxima varredura horizontal. No dispositivo, o feixe de elétrons varre ao longo da linha horizontal enquanto que a varredura vertical do vídeo é feita proporcionalmente ao deslocamento da fase, assim, as variações dos estados lógicos através das trilhas são mostrados verticalmente na tela. A figura e a fotografia 1.3 ilustram esta técnica de observação. Na fotografia 1.3, observa-se para as trilhas C3, C4 e C5 a seguinte seqüência de valores lógicos, durante 2 instantes diferentes de medição: 000 e 100, respectivamente, de acordo com o instante de avaliação

definido pelo sinal de relógio, isto é, CK = 0. O circuito mostrado nesta foto é o mesmo apresentado nas fotografias 1.1 e 1.2: o conversor A/D de 4 bits.

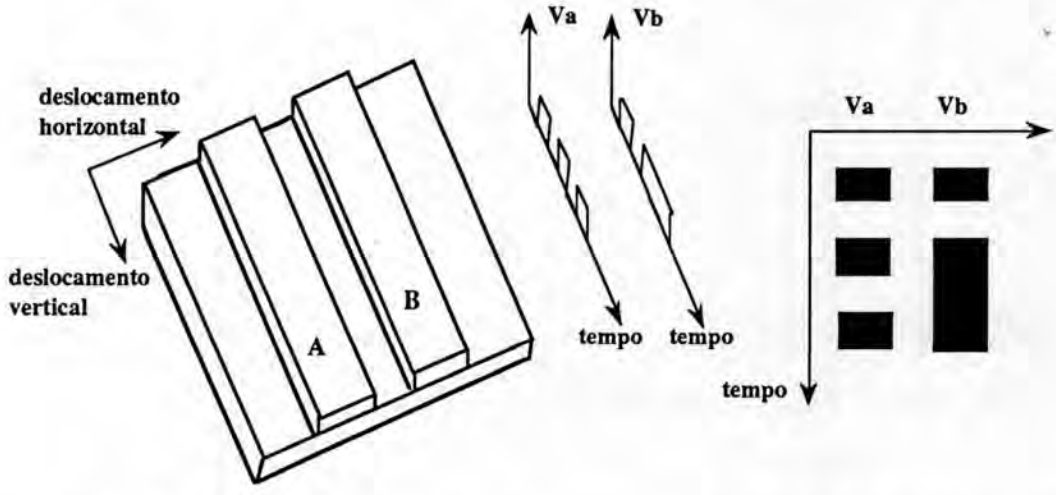


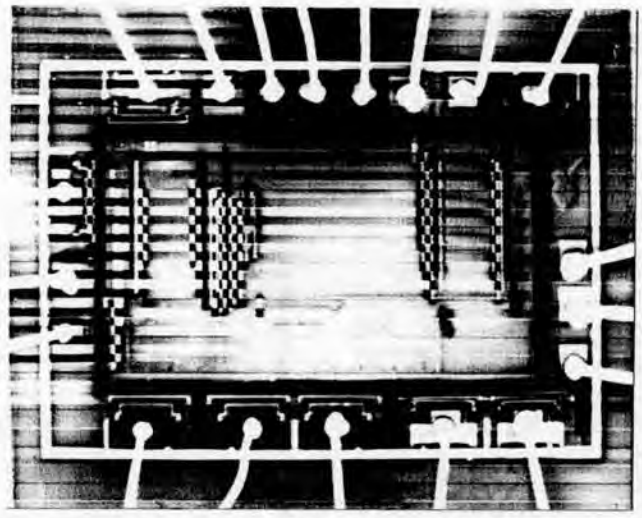
Figura 1.3 Princípio do Modo de Mapeamento do Estado Lógico: sinais aplicados a cada uma das trilhas e respectivos códigos gerados.

1.4 Limitações do Feixe de Elétrons

O feixe de elétrons pode afetar as características do dispositivo testado. Este fenômeno pode ser manifestado de diferentes maneiras:

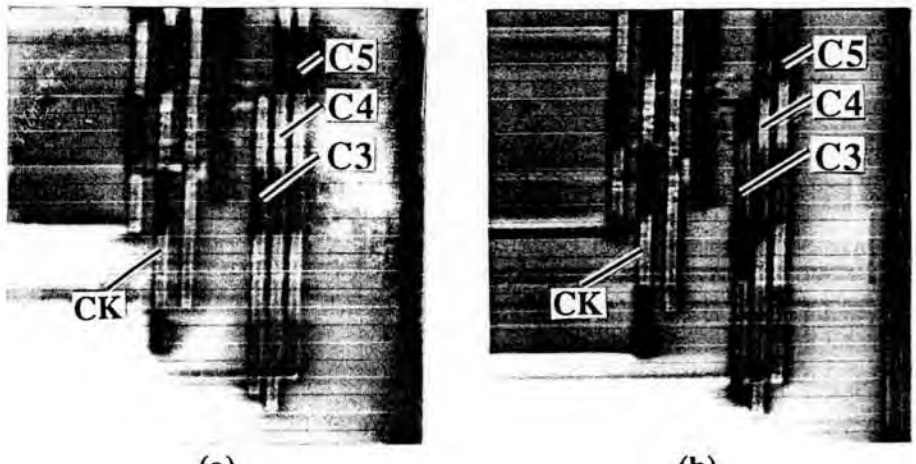
Contaminação da superfície induzida pelo feixe de elétrons: uma superfície observada pelo feixe de elétrons é degradada durante o tempo no qual ela é atingida pelo feixe de elétrons, resultando em uma imagem mais escura. Esta degradação é devido à contaminação da superfície através da absorção de moléculas de hidrocarbono na região irradiada, onde elas são injetadas pelo feixe de elétrons.

Deslocamento da tensão de "threshold" de transistores MOS: se os elétrons primários alcançarem a camada de óxido fino, ou seja, o "gate" de um transistor , eles gerarão pares de elétrons-lacunas. Estas lacunas serão "ocupadas" no óxido fino, o que causa o deslocamento da tensão de "threshold". A variação da tensão de "threshold" depende da tensão do "gate" do transistor durante a irradiação do feixe de elétrons.



Fotografia 1.1 Vista geral do circuito CONVERSO em Modo de Código de Tensão.

Fotografia 1.2 Visualização das saídas do bloco analógico C3, C4, C5, no Modo de Contraste de Tensão Estroboscópico com a entrada analógica do Conversor A/D fixada em 2,2V.



(a) (b)

Fotografia 1.3 Mapeamento do Estado Lógico de 3 conexões (C3, C4, C5) em 2 instantes diferentes de medição: (a) valor lógico medido em t_1: C3: 0, C4: 0, C5: 0; (b) valor lógico medido em t_2: C3: 1, C4: transição: 0/5V, C5: 0.

2 TRATAMENTO DE IMAGEM EM CONTRASTE DE POTENCIAL

2.1 Técnicas de Seleção de "Threshold"

O uso de técnicas de seleção de "threshold" durante o processo de segmentação de imagem tem sido extremamente estudado e uma variedade delas foram propostas para automatizar este processo [WES78].

Pode-se definir técnica de seleção de "threshold" como o processo de escolha de um nível de cinza t, de tal forma que os níveis de cinza maiores que t são mapeados com o rótulo "fundo" da imagem (ou seja, nível de cinza 1) e todos os demais níveis são mapeados com o rótulo "objeto" (nível de cinza 0) [WES78].

No caso de um procedimento automático de seleção de nível de "threshold" de uma imagem, várias técnicas foram desenvolvidas segundo a dependência do nível t em relação aos seguintes parâmetros: posição do "pixel", valor do nível de cinza, definição de um atributo (por exemplo, a intensidade média da vizinhança centrada sobre o ponto considerado).

Assim, a relação do nível t com estes parâmetros permite definir os seguintes tipos de linearização do valor de "threshold" [WES78]:

- Se t depende somente do **nível de cinza**, então a seleção de "threshold" é dita linearização **global**;
- Se t depende do **nível de cinza** e de um **atributo**, então a linearização é **local**;
- Se t depende do **nível de cinza**, de um **atributo** e da **posição do "pixel"** analisado, então a linearização é dita **dinâmica**.

2.1.1 Seleção de "Threshold" Global

2.1.1.1 Baseada em Histogramas de Níveis de Cinza

As primeiras técnicas de seleção automática de "threshold" global foram baseadas na análise do histograma do nível de cinza da imagem. Existem duas técnicas principais:

A primeira, [DOY62], trata de imagens cujos objetos são conhecidos e cujas áreas são invariáveis. Assim, se as áreas claras e escuras contidas em uma imagem não variam, o valor de "threshold" poderia ser obtido sempre no mesmo ponto. Por exemplo, supondo que uma imagem sempre possuísse 20% de seus "pixels" mapeados como objeto e o resto como "fundo", então a imagem poderia ter o seu valor de "threshold" determinado na sua 80ésima percentagem, ou seja, nos seus 20% mais elevados níveis de cinza.

A segunda técnica, proposta por Prewitt [PRE66], envolve a procura de "vales" (ou antinodos) no histograma, ou seja, trata-se da procura dos níveis de cinza cujos valores são intermediários entre o nível de cinza do objeto (1) e o do fundo da imagem (0). Suas posições relativas na imagem são as da "fronteira" entre o objeto e o fundo.

2.1.1.2 Baseada em Propriedades Locais

Os valores de propriedades locais tem sido usados de duas maneiras [MAS78]: primeiro visando melhorar a "forma" dos histogramas de níveis de cinza, por exemplo, tornando-os fortemente bimodais e segundo, para calcular diretamente o nível de "threshold" da imagem.

Métodos para Melhorar a Forma de Histogramas: este método tem por objetivo tornar os vales de histogramas mais profundos a fim de facilitar o uso das técnicas apresentadas por [PRE66]. Trata-se de um método no qual os pontos sobre a curva do histograma não são calculados igualmente. Quanto maior é a diferença entre um dado operador e o valor definido pelo ponto sobre a curva, maior é o peso dado a este ponto. Assim, pontos no interior de um objeto e sobre o fundo da imagem (picos do histograma) são marcados fortemente, enquanto que os pontos na fronteira entre o objeto e o

fundo (vales do histograma) são fracamente contados. Como resultado deste processo, a curva do histograma passa a apresentar picos mais agudos e vales mais profundos, facilitando a seleção do vale como nível de "threshold".

Métodos para Selecionar Níveis de "Threshold": a técnica descrita a seguir faz uso de valores de propriedades locais, tal como a determinação do valor do gradiente para cada um dos pontos sobre a curva do histograma de uma imagem em nível de cinza, a fim de computar diretamente o valor de "threshold" da imagem. Assim, esta técnica envolve a aplicação de gradiente à imagem e para tanto somente os pontos tendo os maiores valores de gradiente são "histogramados". Neste caso, estes pontos recaem sobre ou próximos aos contornos dos objetos e apresentam um histograma de nível de cinza bimodal. Neste caso, o fundo do vale representa o nível de cinza no qual os contornos dos objetos são máximos. Se o histograma selecionado é bimodal, a média do nível de cinza dos pontos filtrados pode ser um limite razoável para se definir como valor de "threshold" a fim de se separar os objetos do fundo da imagem.

2.1.2 Seleção de "Threshold" Local

Ullmann [ULL74] propôs uma técnica para selecionar o nível de "threshold", baseada nos níveis de cinza da vizinhança de um ponto. Somente os pontos rotulados por **n** em uma vizinhança de 5X5 de um ponto **p** contribuem para a decisão do nível de "threshold" no ponto **p**:

o n n n o
n n o n n
n o **p** o n
n n o n n
o n n n o

Baseado em resultados experimentais, Ullmann utilizou duas regras para selecionar o "threshold" em um ponto: a regra escolhida dependia do valor do nível de cinza mais alto na vizinhança de **p**, denotado por $\mathbf{n_p}$. A regra (a) era aplicada quando $\mathbf{n_p} \leq 40$ e a regra (b), quando $\mathbf{n_p} > 40$. As duas regras são:

(a) O ponto **p** é um ponto pertencente ao objeto se para algum ponto **n** da vizinhança tem-se

$$p - n < r$$

onde **r** é um valor de "threshold" pré-definido; caso contrário o ponto **p** pertence ao fundo da imagem.

(b) O ponto **p** é um ponto pertencente ao objeto se, para ao menos um de seus vizinhos **n**, tem-se

$$p < n/\mu$$

onde μ é o valor de uma constante pré-definida; caso contrário o ponto **p** pertence ao fundo da imagem.

2.1.3 Seleção de "Threshold" Dinâmica

Chow e Kaneko [CHO72] apresentaram uma técnica de seleção de "threshold" dinâmica que utiliza os mesmos princípios da técnica de seleção de "threshold" local, descrito no ítem 2.1.2, para serem aplicadas sobre vizinhanças de 7X7 "pixels".

Nesta técnica, um histograma é apresentado para cada vizinhança (modelado por uma ou duas distribuições normais, de acordo com sua forma: unimodal ou bimodal). A partir deste histograma, a seleção de "threshold" dinâmica é determinada da seguinte forma:

• para cada histograma cuja variação de nível de cinza excede um valor pré-definido de "threshold", são estimados alguns parâmetros das distribuições de componentes da sequinte maneira:

- para cada uma das vizinhanças, é determinado o nível de "threshold" para os seus pontos centrais, exatamente da mesma forma que aqueles apresentados em 2.1.2.

- os valores de "threshold" definidos para cada ponto central de janela (vizinhança) são usados na interpolação de polinômios a fim de determinar o valor de "threshold" para cada um dos outros pontos pertencentes à imagem.

2.2 Técnicas de Detecção de Cantos

As técnicas de detecção de cantos são empregadas para a localização automática de objetos poligonais em imagens. Estas técnicas se dividem em dois grupos, segundo a natureza dos dados analisados:

- o primeiro grupo procura os cantos de polígonos sobre imagens binárias (imagens codificadas sobre dois níveis de cinza), onde os dados se apresentam sob a forma de contornos fechados de espessura de 1 "pixel". Para tanto, são exigidas técnicas de segmentação de imagem, cujo objetivo é a extração dos contornos binários da imagem inicial, codificada em níveis de cinza. Este grupo é denominado de **Técnicas de Detecção Binária.**
- o segundo grupo procura os cantos de polígonos diretamente sobre as imagens em níveis de cinza. Neste caso as técnicas de segmentação, que fazem vários tratamentos de imagem tais como filtragem do sinal em níveis de cinza e extração de contornos binários, são dispensadas. Este grupo é denominado de **Técnicas de Detecção em Nível de Cinza.** A figura 2.1 apresenta o diagrama geral das técnicas de detecção de cantos, bem como alguns dos métodos mais utilizados durante o processo.

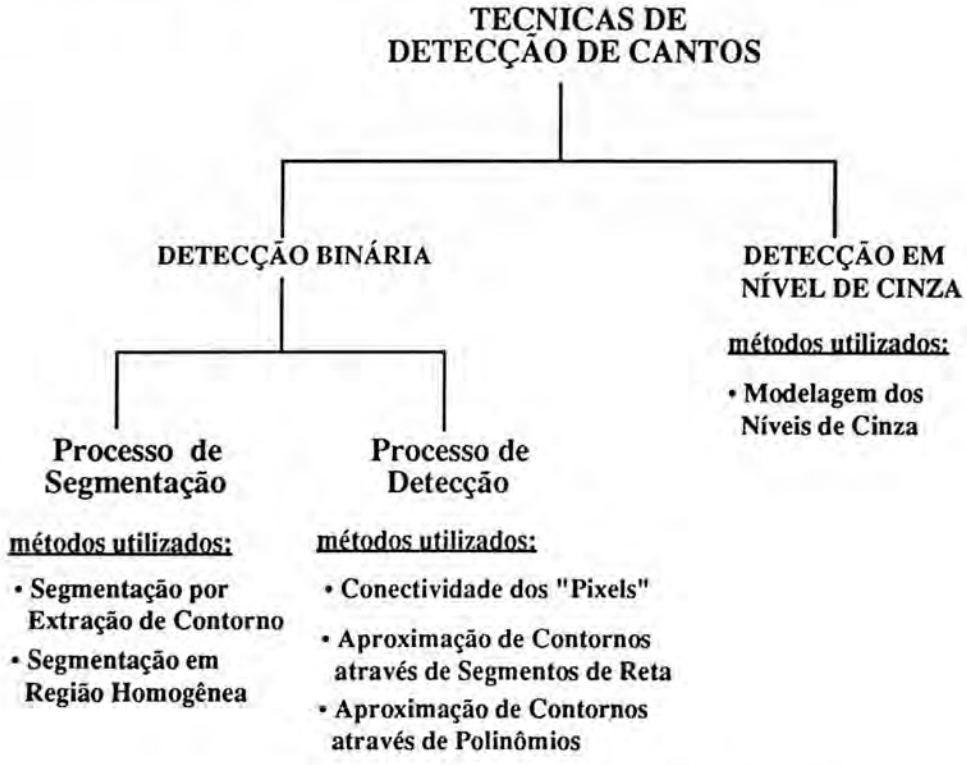


Figura 2.1 Técnicas de Detecção de Cantos

2.2.1 Técnicas de Detecção Binária

Estas são as técnicas mais utilizadas em reconhecimento de formas atualmente. Elas consistem em procurar, sobre a imagem de contornos binários, os segmentos de reta cujos pontos de quebra apresentam uma curva importante. Para tanto, utiliza-se uma codificação de contorno baseando-se na **conectividade dos "pixels"** ou então numa **aproximação de contornos através de segmentos de reta** ou **através de polinômios.** Procedimentos anteriores de segmentação de imagem são fundamentais para um bom resultado das técnicas de detecção binárias.

2.2.1.1 Processo de Segmentação de Imagem

O processo de segmentação constitui-se na primeira etapa de análise de uma imagem. Sua função principal é a separação dos diversos elementos de uma imagem em regiões bem definidas, representadas através de seus contornos fechados. Estes contornos podem ser obtidos através de suas fronteiras, no caso da utilização de técnicas de **segmentação por extração de contorno**, ou então podem ser diretamente caracterizados pelos "pixels" que os constituem, no caso de uma **segmentação em região homogênea.**

2.2.1.1.1 Segmentação por Extração de Contorno

O processo de segmentação por extração de contornos consiste em detectar as variações bruscas de níveis de cinza sobre a imagem. Com este objetivo, as etapas de **determinação, redução e binarização das linhas de contorno** são realizadas:

Determinação dos Contornos: obtém-se através da diferenciação da imagem. Para tanto, realiza-se a derivação da imagem, uma ou duas vezes, de acordo com o operador utilizado: gradiente ou laplace.

No caso do operador utilizado ser o gradiente, este se apresenta da forma seguinte:

para uma função f(x,y), o gradiente:

$$G = \begin{pmatrix} \frac{df(x,y)}{dx} \\ \frac{df(x,y)}{dy} \end{pmatrix}$$

é um vetor orientado no sentido da variação máxima.

Assim, este vetor é representado para as imagens digitalizadas I(x,y) através de seu módulo:

$$G = f(x,y) - f(x+1,y) + f(x,y) - f(x,y+1)$$

e por sua direção, caracterizada pela tangente:

$$D(x,y) = \text{arctg}\left(\frac{f(x,y) - f(x,y+1)}{f(x+1,y) - f(x,y)} \right)$$

No caso do operador laplaciano, este vem a ser o cálculo da derivada segunda da função que representa a imagem. A vantagem deste operador é que ele fornece os contornos com espessura de apenas 1 "pixel". Isto é devido ao fato de que o método laplaciano apresenta uma única passagem por zero. Entretanto, seu maior inconveniente é a grande sensibilidade ao ruído. Isto faz necessário o uso de técnicas de convolução de imagem através de filtros passa-baixo para atenuação do ruído. A figura 2.2 mostra o processo de extração de contornos dos operadores de laplace e de gradiente em função de dois objetos contidos em uma imagem.

Redução dos Contornos: esta etapa consiste em reduzir as linhas do contorno a um único "pixel" de espessura. [CAN83] propôs um método chamado "método de supressão de pontos não-máximos". Este método consiste em eliminar do contorno todos os pontos cujos vizinhos apresentem valores de amplitude de gradiente maiores que os deles.

Binarização das Linhas de Contorno: esta etapa consiste em reduzir o conjunto de níveis de cinza a apenas dois níveis de maneira a facilitar a análise dos elementos componentes da imagem.

Geralmente se efetua a binarização da imagem a fim de se eliminar as flutuações devido ao ruído ou à textura dos objetos.

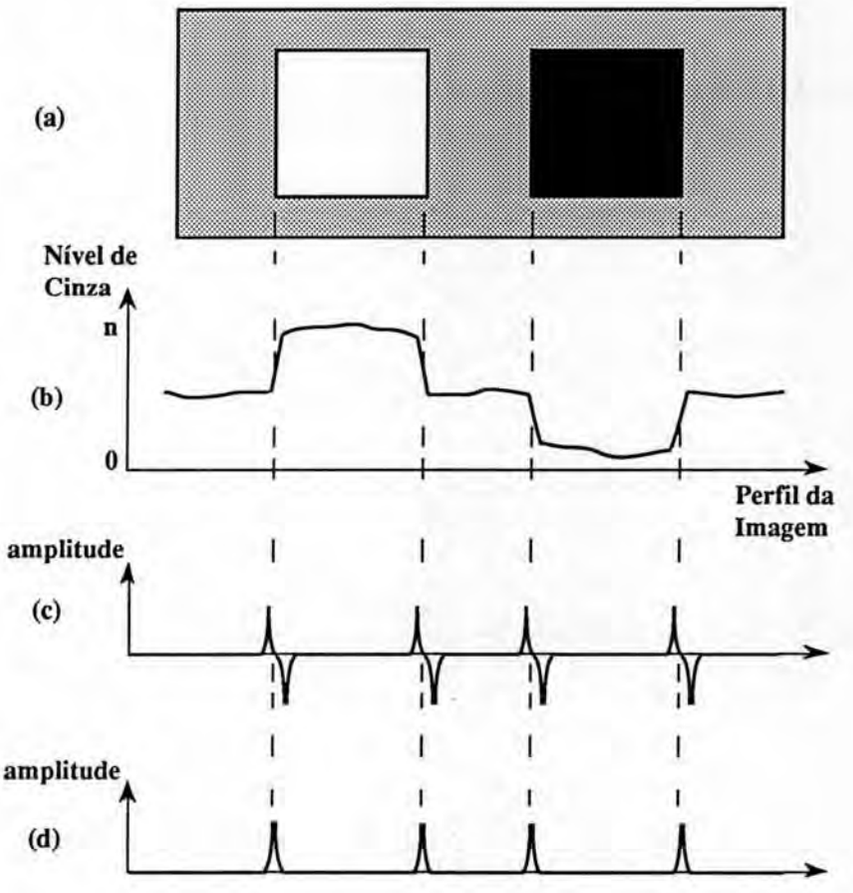
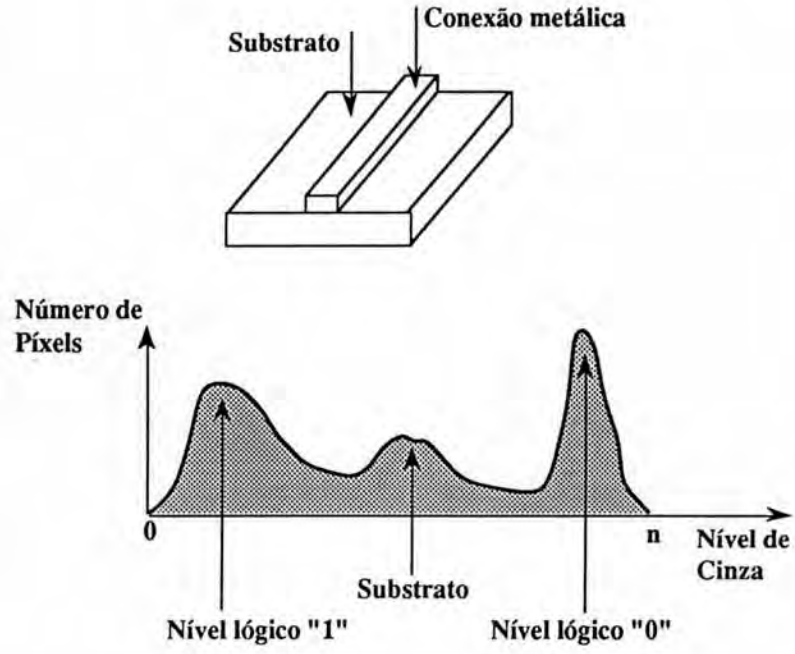


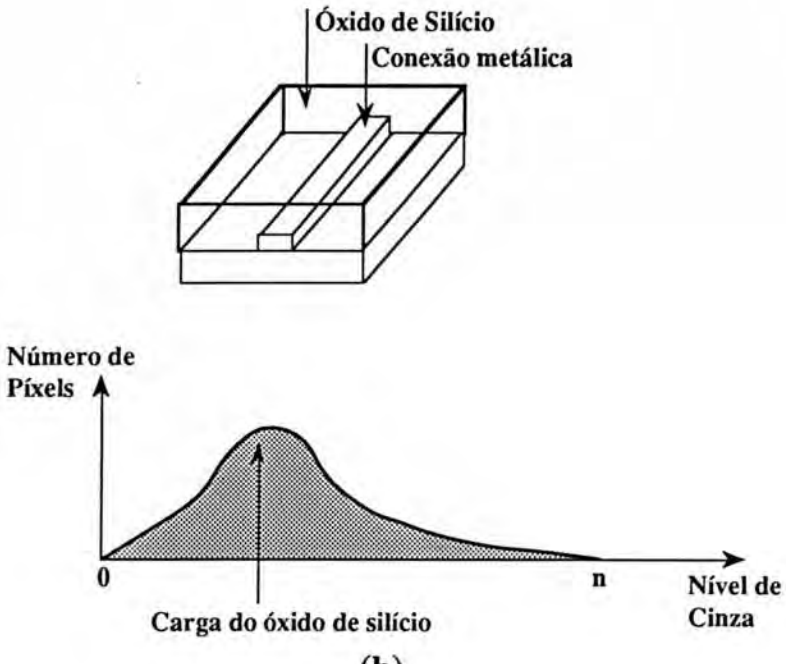
Figura 2.2 Derivação dos contornos de uma imagem. (a) Imagem, (b) Perfil da Imagem, (c) Laplace, (d) Gradiente.

Técnicas baseadas na análise de histogramas de níveis de cinza são utilizadas para fazer uma linearização adaptativa em função das classes de níveis de cinza.

A título de ilustração, a figura 2.3 apresenta o perfil típico dos histogramas de níveis de cinza para circuitos não-passivados e passivados. A partir da análise destas curvas, são obtidos processos de seleção automática de níveis de "threshold" a fim de binarizar as linhas de contorno dos objetos.



(a)



(b)

Figura 2.3 Perfil geral dos histogramas para os circuitos (a) não-passivados e (b) passivados.

2.2.1.1.2 Segmentação por Região Homogênea

A técnica de segmentação por região homogênea baseia-se na análise de histogramas de níveis de cinza. Para tanto, duas etapas são exigidas: a **linearização do histograma** dos níveis de cinza da imagem e a **definição dos contornos em uma imagem homogênea.**

Linearização do Histograma: o histograma apresenta a freqüência de aparição de cada nível de cinza na imagem. Assim, para uma imagem com **n** "pixels" em um determinado nível de cinza entre **k** níveis discretos, a freqüência de aparição **Pk** se exprime através da fórmula

$$Pk = \frac{nk}{n} = \frac{\text{número de "pixels" no nível k}}{\text{número total de "pixels" da imagem}}$$

Uma imagem composta de objetos cinza claro (trilhas em 0V) sobre um fundo cinza escuro (substrato) faz aparacer dois grupos de nível de cinza sobre o histograma, como pode ser visto na figura 2.4.

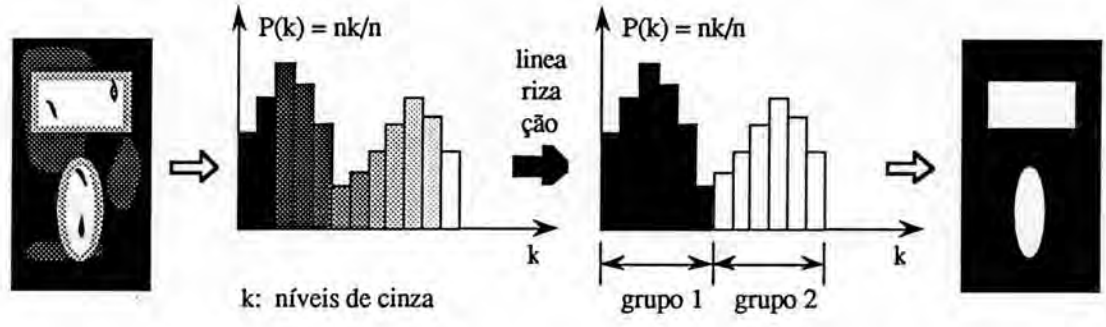


Figura 2.4 Linearização do Histograma

Para se fazer a extração de um objeto a partir de uma imagem, deve-se escolher um valor limite entre os dois grupos de níveis de cinza como valor de "threshold", no caso, S. Assim, o objeto é definido pela condição seguinte:

PARA cada "pixel" de intensidade I(x,y)
SE I(x,y) > S então OBJETO
senão FUNDO
FSE
FPARA.

Pode-se notar através do histograma que o limite S é o valor mínimo entre os dois picos. Ele representa a fronteira entre o objeto e o fundo da imagem (no caso de uma imagem formada por n níveis de cinza, S seria a minoria de "pixels" que assume valores de níveis de cinza intermediários entre os do objeto e os do fundo da imagem).

Neste caso, um procedimento automático de linearização do histograma em níveis de cinza consiste em, a partir de uma etapa de pesquisa, identificar o valor mínimo S entre os dois picos, no caso de um histograma bimodal.

Definição do Contorno de uma Imagem Homogênea: a detecção de contornos em regiões homogêneas de uma imagem segmentada pode ser obtida por métodos de diferenciação como descritos na etapa de determinação dos contornos, no processo de segmentação por extração de contorno, ou então por outros métodos mais rápidos, baseados na vizinhança direta dos "pixels" constituintes da região homogênea. Segundo estes métodos, o contorno pode ser memorizado e codificado através de uma lista de valores constituintes de uma cadeia numérica. Esta codificação descreve cada "pixel" do contorno tendo sempre em vista a posição relativa do "pixel" precedente àquele que se deseja codificar.

Freedman [FRE77] definiu um código que determina para cada uma das 8 direções possíveis, um valor numérico para os pontos conexos a um ponto central, conforme pode ser visto na figura 2.5. No ítem (c) da figura 2.5 pode ser visto um exemplo de codificação do objeto E definido no ítem (a).

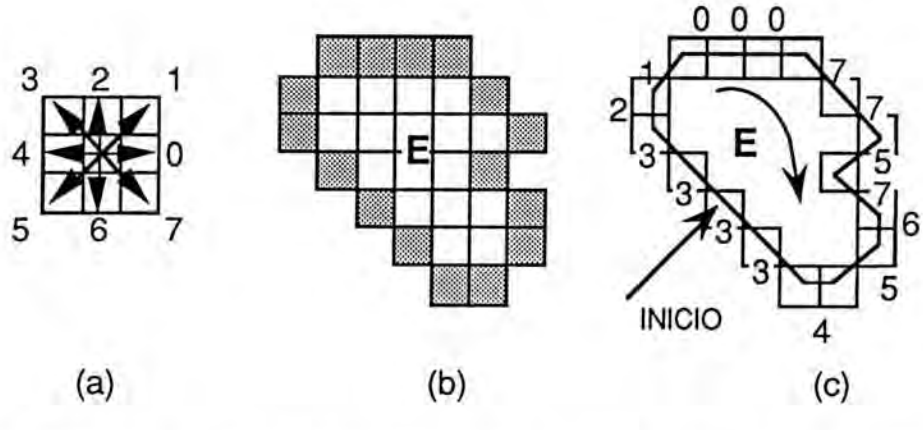


Figura 2.5. Código de FREEMAN. **(a)** Definição do Código; **(b)** Contorno do objeto modelado; **(c)** Exemplo de modelagem.

Esta codificação dá uma representação do objeto, para um conjunto de arcos elementares, tal como:

$$E = \{arc(i)\} = \{3345675770001233\}$$

2.2.1.2 Detecção de Cantos sobre Imagens Binárias

Os detetores de cantos sobre imagens binárias são classificados segundo a modelagem do contorno. Três tipos de modelagem são utilizados: **codificação por ângulo, aproximação por segmentos de reta** e **aproximação polinomial.**

Codificação por Ângulo: esta é uma técnica de codificação de FREEMAN que caracteriza os ângulos entre os arcos. As máscaras utilizadas e um exemplo de codificação para o contorno do objeto **E** da figura 2.5.b são mostrados na figura 2.6.

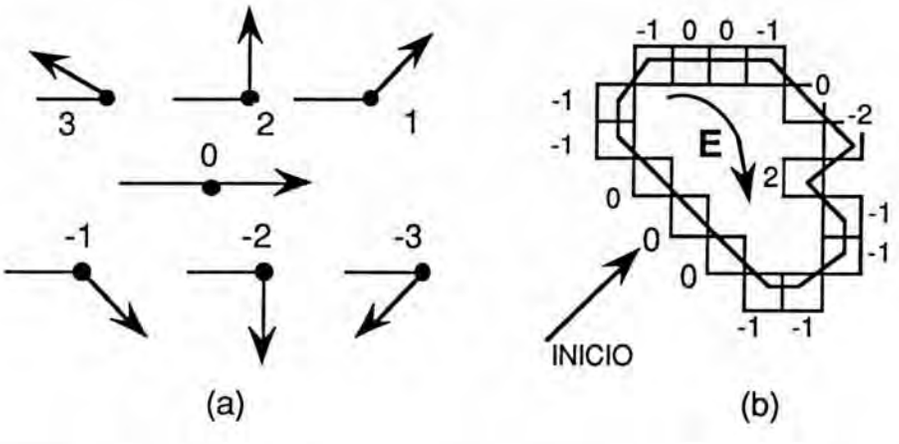


Figura 2.6 Codificação por Ângulo. **(a)** Definição do Código; **(b)** Exemplo de Codificação.

Esta codificação dá uma representação do objeto, para um conjunto de arcos elementares, de acordo com a cadeia numérica:

$$E = \{arc(i)\} = \{00\text{-}1\text{-}1\text{-}100\text{-}10\text{-}22\text{-}1\text{-}1\text{-}1\text{-}10\}$$

Esta técnica caracteriza-se por levar em conta somente os 3 "pixels" adjacentes, o que não permite determinar os cantos sobre objetos cujas bordas apresentam numerosas flutuações.

Aproximação por Segmentos de Reta: esta técnica consiste em determinar a curvatura do objeto através de segmentos de reta. Tal método associa a cada "pixel" do contorno um segmento de reta que liga as extremidades de S "pixels" adjacentes.

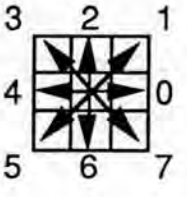
A representação é a seguinte:

Os segmentos de S "pixels" são representados em notação de FREEMAN através da expressão:

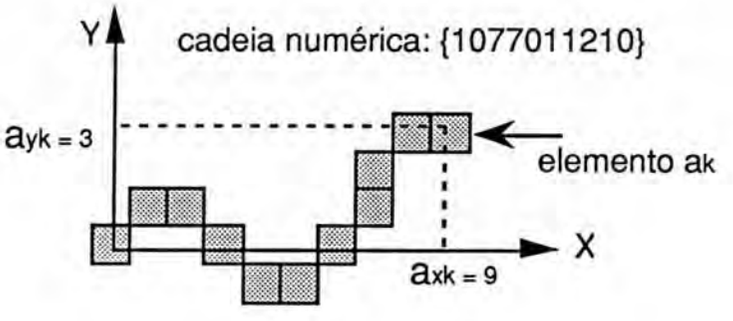
$$A = a_1, ..., a_i, ...a_S \qquad \text{onde } ai \ C \ \{0,...,7\}$$

A tabela representada na figura 2.7.a mostra a relação entre o código numérico $\mathbf{a_i}$ e suas projeções sobre os eixos **X** e **Y**, notadas repectivamente $\mathbf{ax_i}$ e $\mathbf{ay_i}$.

ai	0	1	2	3	4	5	6	7
axi	1	1	0	-1	-1	-1	0	1
ayi	0	1	1	1	0	-1	-1	-1



(a)



(b)

Figura 2.7 Projeção de uma cadeia numérica sobre os eixos X e Y. **(a)** tabela de projeção; **(b)** exemplo de projeção.

Aproximação Polinomial: este método consiste em modelar os contornos de um objeto através de um conjunto de funções polinomiais. Esta técnica não restitui toda a informação mas assegura uma transformação entre uma representação discreta (modo "pixel") e um conjunto de funções contínuas que facilitam a análise da curvatura do contorno. Através deste processo, os cantos são representados pela curvatura dos polinômios ajustados ao contorno [MED87].

2.2.2 Técnicas de Detecção de Cantos em Nível de Cinza

Estas técnicas, introduzidas por [BEA78], tem a vantagem de dispensar a etapa de segmentação da imagem. Entretanto, apresentam o inconveniente de serem extremamente sensíveis ao ruído pois elas se baseiam na análise direta dos níveis de cinza.

Zuniga [ZUN83] propôs um método no qual, considerando a sua vizinhança, cada "pixel" é modelado localmente (ver figura 2.8). Para tanto, ele utiliza uma função polinomial bicúbica para modelar o "pixel" corrente e sua vizinhança, que é definida pela função:

$$f(x,y) = K_1 + K_2x + K_3y + K_4x^2 + K_5xy + K_6y^2 + K_7x^3 + \\ + K_8x^2y + K_9xy^2 + K_{10}y^3$$

onde os coefecientes $\mathbf{K_i}$ são determinados pela média quadrática dos níveis de cinza dos "pixels" a serem modelados.

Para a modelagem local do "pixel" corrente, a vizinhança é composta pela matriz quadrada de "pixels" centrada sobre o "pixel" corrente. A dimensão de tal matriz depende dos objetos a analisar. Geralmente ela é inferior a 10, para não inviabilizar os cálculos em termos de consumo de tempo.

Segundo [ZUN83], um "pixel" é identificado como sendo um elemento pertencente ao contorno de um objeto, se e somente se:

1. A primeira derivada direcional da função f (função que modela localmente cada "pixel " tendo sempre em conta a sua vizinhança) é diferente de 0 ;

2. A segunda derivada da função f é igual a 0.

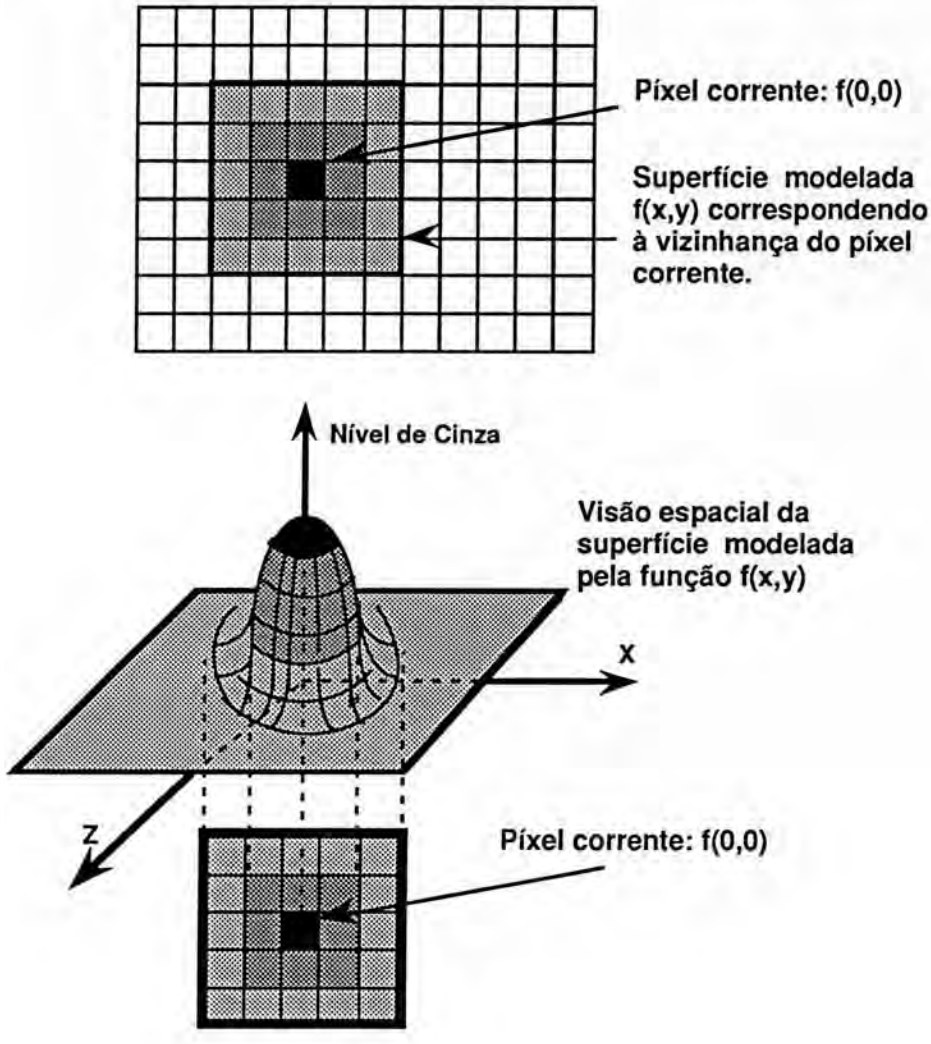


Figura 2.8. Modelagem local do "pixel" corrente.

Além de identificar os "pixels" pertencentes ao contorno de um objeto, [ZUN83] também definiu os parâmetros para detecção de cantos; ou seja, um determinado "pixel" é considerado como sendo um canto, se e somente se (ver figura 2.9):

1. O "pixel" em análise, além de outros dois pontos (P_1 e P_2) situados a uma mesma distância do "pixel" analisado, e em lados opostos em relação a este, sejam todos pertencentes à linha de contorno do objeto.

2. A variação incremental da direção do gradiente ao longo do contorno do objeto (mais precisamente: $\phi(P_1) - \phi(P_2)$) for maior que um determinado valor de "threshold", onde $\phi(P_1)$ e (P_2) é a direção do gradiente nos pontos P_1 e P_2, respectivamente.

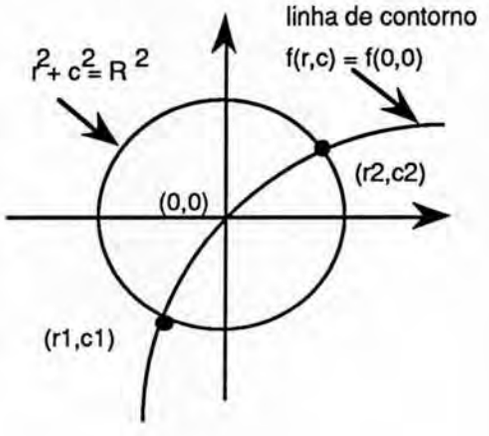


Figura 2.9. Determinação dos pontos simétricos ao "pixel" corrente sobre a curva que representa o contorno do objeto.

Esta técnica é computacionalmente bastante árdua em função da necessidade de se calcular a intersecção entre a curva cúbica f(r,c) = f(0,0) (linha de contorno) e a curva quadrática $r^2 + c^2 = R^2$ a fim de se determinar os pontos P_1 e P_2 a uma distância R da origem ("pixel" corrente).

3 VALIDAÇÃO DE PROTÓTIPO E ANÁLISE DE FALHAS COM TESTE DE FEIXE DE ELÉTRONS

3.1 Introdução

O projeto de um sistema integrado de teste atualmente em desenvolvimento no laboratório TIM3 tem por objetivo aliar uma ferramenta de teste, o microscópio eletrônico de varredura, a técnicas de localização de falhas em circuitos complexos. Este projeto baseia-se na utilização de métodos de alto nível, tais como sistemas especialistas e técnicas de comparação automática de imagens, e seu objetivo principal é a automação do processo de diagnóstico de falha.

Neste contexto, este projeto conduz a dois temas independentes de pesquisa.

O primeiro, é a validação de protótipos de circuitos cuja estrutura é conhecida, sendo que as informações sobre o processo de concepção estão disponíveis, tais como diagramas de blocos, resultados de simulações, etc.

O segundo tema de pesquisa concerne à análise de falhas sobre circuitos cuja estrutura não é conhecida, de forma que estes são comparados com outros circuitos de referência, assumidos como sendo perfeitos.

Nos dois casos, o circuito sob teste é submetido a uma avaliação através do feixe de elétrons do microscópio, a fim de se obter as informações necessárias a um sistema de aquisição e tratamento de imagem para que, então, métodos específicos de tratamento dos resultados da observação forneçam o diagnóstico automático da(s) falha(s).

3.2 Validação de Protótipos Baseada no Conhecimento

3.2.1 Ambiente de Teste

No que concerne ao assunto "validação de protótipos baseada no conhecimento", está em desenvolvimento um ambiente de teste baseado em técnicas de sistemas especialistas [MAR91a], [CON90]. Em particular, este ambiente é integrado basicamente por duas unidades: Sistema de Aquisição de Informações (SAI) e Sistema de Tratamento de Informações (STI).

O Sistema de Aquisição de Informações é composto por 3 elementos básicos:

1. um emulador (TESSIE), que assegura o condicionamento do circuito em análise através da simulação do seu ambiente de utilização;

2. um microscópio eletrônico de varredura (CAMECA ST15), utilizado em contraste de potencial e munido de um subsistema de controle, permitindo observar os estados lógicos e elétricos do circuito em teste;

3. um órgão de captura e tratamento de imagens do circuito observado pelo microscópio. Detalhes desta ferramenta de captura e tratamento de imagens serão dados mais adiante, nos ítens 3.4 e 6.3.1.

O Sistema de Tratamento de Informações deve preparar os dados para o Sistema Especialista e interagir com o seu meio exterior. Como dados principais, o STI recebe, provenientes do SAI, as imagens do circuito em teste, e provenientes das ferramentas de CAD, os resultados das simulações do funcionamento correto do circuito, bem como uma descrição de sua estrutura. O STI também admite como dados certas hipóteses de falha sobre o circuito em análise, fornecidas pelo usuário. A figura 3.1 mostra o diagrama de blocos do ambiente de teste para validação de protótipos.

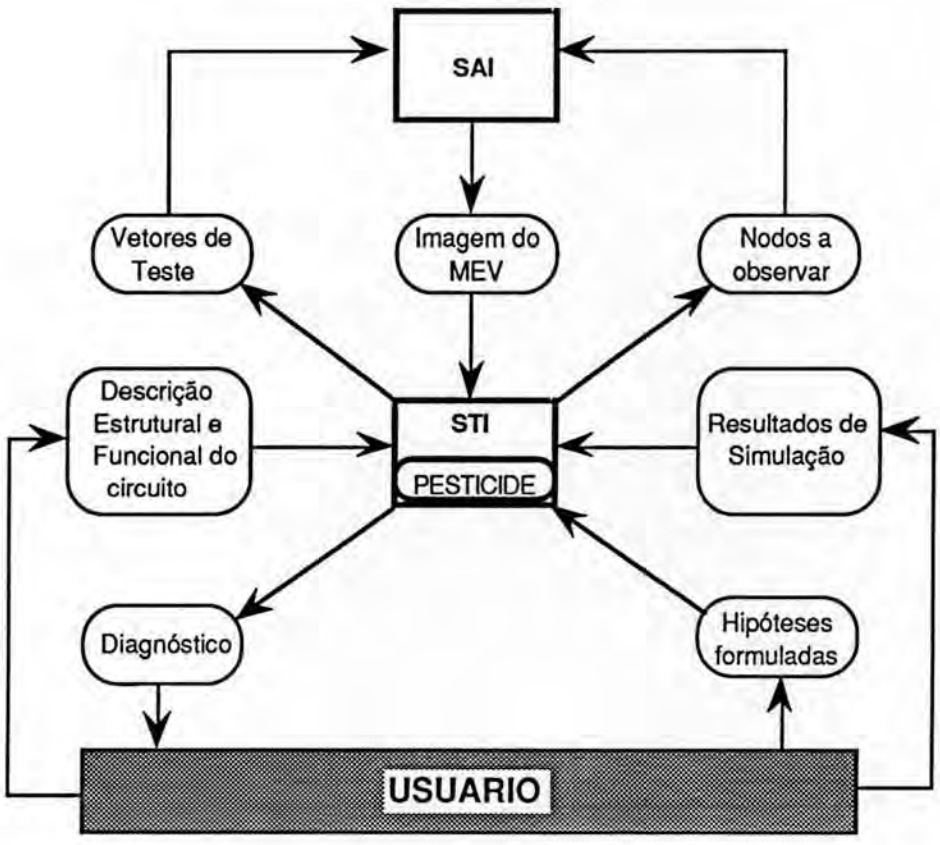


Figura 3.1 Ambiente de teste para validação de protótipos [MAR91a].

3.2.2 Sistema Especialista

O sistema especialista, batizado com o título de PESTICIDE (Prolog-Written Expert System as a Tool for Integrated CIrcuits DEbugging), tem como função principal fornecer ao usuário o diagnóstico concernente ao circuito em teste [MAR89a], [MAR91a].

O circuito em análise deve ser fornecido ao sistema especialista através de um arquivo de entrada, no qual ele é decomposto, hierarquicamente, em 3 componentes: blocos, conexões, e interfaces-bloco (portas de entrada/saída). Segundo esta modelagem, 3 bases de conhecimentos são obtidas:

1. Base de Conhecimentos Estruturais, contendo os parâmetros dos 3 componentes básicos (os blocos, as conexões e as interfaces-bloco), sendo entendido que estes parâmetros identificam "situações estáticas" do circuito em teste;

2. Base de Conhecimentos Funcionais, contendo as informações sobre o funcionamento do circuito protótipo;

3. Base de Conhecimentos Comportamentais, contendo as informações sobre o desenvolvimento do teste em curso, principalmente no que se refere aos instantes de interfaces blocos. Estas duas últimas bases de conhecimento, funcional e comportamental, identificam "situações dinâmicas" do circuito.

Para explorar estas bases de conhecimentos, 3 módulos são utilizados:

1. Um módulo de verificação, que assegura a consistência de dados entre as bases de conhecimentos;

2. Um módulo de detecção, que identifica falhas em conexões;

3. Um módulo de localização, que procura pelos blocos defeituosos (o circuito em teste é modelado segundo a estrutura de vários blocos interconectados).

Os componentes de PESTICIDE, bases de conhecimentos e módulos que exploram estas informações, são controlados e manipulados por um interpretador PROLOG.

3.2.2.1 Modelagem do Circuito

Neste modelo, o dispositivo em teste é decomposto hierarquicamente em três componentes básicos: o bloco, a interface-bloco e a conexão. (Ver figura 3.2).

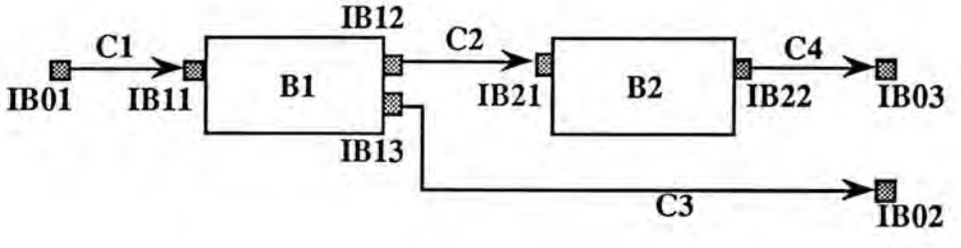


Figura 3.2. Modelagem de um dispositivo para depuração

• **Bloco**: os parâmetros do bloco são:

- Seu nome;

- Sua lista de interfaces, indicando quais são as interfaces blocos que o conectam ao meio exterior, bem como o seu tipo: entrada, saída ou bidirecional.

• **Interface-Bloco**: a interface-bloco permite conectar os blocos entre eles mesmos e aos "pads" do circuito. A maior razão para a inclusão destas interfaces-bloco na modelagem de um circuito é a possibilidade de identificação de falhas em conexões e trilhas, bem como a detecção de falhas em circuitos seqüenciais graças aos parâmetros de tempo. Os parâmetros associados às interfaces-bloco são:

- Seu *nome* ;

- Seu *estado* (certo, errado, ou não observado, de acordo com a sessão de teste corrente);

- Seu *valor lógico* ,

- Seu *instante de medição* , ou seja, o momento correto para medição do valor lógico de uma interface-bloco, de acordo com sinal de relógio do dispositivo. Este parâmetro é muito importante no caso de circuitos seqüenciais, pois é ele que definirá os instantes corretos de medição, de forma a varrer o circuito sequencial como se este fosse um circuito combinacional;

- Seu *tipo* , ou seja, entrada ou saída de um bloco, ou bidirecional;

- Seu *atraso* , medido em ciclos de relógio, representando o tempo após o qual o valor de uma interface-bloco pode ser levado em consideração, ou seja, medido. Este *atraso* é avaliado como sendo o tempo de funcionamento do bloco mais o tempo de estabilização do sinal em sua saída, ou seja, na sua interface-bloco (neste caso, tipo da interface = saída), ou então o tempo de propagação da trilha mais o tempo de estabilização do sinal na entrada do bloco, (neste caso, tipo da interface = entrada).

- Seu *tempo de manutenção* , ou seja, este parâmetro define o tempo que a interface mantém em sua saída o mesmo valor do estado lógico.

• **Conexão**: a conexão tem como parâmetros associados:

- Seu nome;

- Uma lista das interfaces-bloco que estão conectadas em suas extremidades de origem ou de destino.

3.2.2.2 Organização dos Conhecimentos do Sistema Especialista

Os três componentes básicos (bloco, interface-bloco e conexão) formam a base de conhecimento do sistema especialista através da composição de um conjunto de informações a respeito da estrutura, funcionalidade e comportamento do circuito em teste. A fim de organizar este tipo de conhecimento, três bancos de dados foram gerados: o banco de dados "estrutural", o "funcional" e o "comportamental".

• **Base de Conhecimento Estrutural:** a base de conhecimento estrutural contém todas as informações a respeito do circuito sob teste:

- definição dos blocos, incluindo os seus dois parâmetros;
- definição das interfaces-bloco, restringindo-se aos seus parâmetros estruturais: *nome* , *tipo* , *atraso* e *tempo de manutenção* ;
- definição das conexões, com seus dois parâmetros.

• **Base de Conhecimento Funcional:** a base de conhecimento funcional contém duas informações adicionais concernentes às interfaces-bloco:

- uma condição, definindo para a interface-bloco bidirecional qual o seu tipo: entrada ou saída. Esta condição pode, por exemplo, referir-se a outras interfaces-bloco já definidas;
- um cone de cobertura para cada interface-bloco de saída. (considerando-se um mesmo bloco, define-se "cone de cobertura" como sendo a informação adicional, cuja função é expressar a dependência funcional entre cada uma de suas interfaces-bloco de saída e as interfaces-bloco de entrada, que sobre elas atuem).

• **Base de Conhecimento Comportamental:** a base de conhecimento comportamental contém informações a respeito do teste corrente, consistindo dos seguintes parâmetros:

- resultados de medições dos valores lógicos de interfaces-bloco, qualificados pelo *nome* da interface-bloco e pelos seus atributos "comportamentais": seu *estado* (certo, errado ou não observado), seu *valor lógico*, seu *tipo* (entrada ou saída) e seu *instante de medição*;

- uma hipótese, que é formulada a respeito do teste corrente. Esta hipótese pode ser:
  - Combinacional simples, modelo de falha simples, restrito a circuitos combinacionais;
  - Combinacional múltipla, modelo de falha múltipla, restrito a circuitos combinacionais;
  - Sequencial simples, modelo de falha simples, normalmente aplicado a circuitos sequenciais;
  - Sequencial múltipla, modelo de falha múltipla, normalmente aplicado a circuitos sequenciais.

Esta hipótese não é uma restrição no processo de obtenção do diagnóstico, mas sim uma forma de se obter uma simplificação nesta tarefa no sentido de se reduzir o tempo de execução do programa. Caso nenhuma hipótese tenha sido formulada, PESTICIDE assumirá como modelo de falha o caso mais geral: falha múltipla em circuitos sequenciais.

3.2.2.3 Processo de Detecção de Falhas

O processo de detecção de falhas tanto em blocos quanto em conexões exige a definição dos estados lógicos das interfaces blocos. Para que esta definição seja considerada válida, é necessário que a observação dos blocos ou das conexões seja feita em "bons" momentos, ou seja, no instante de medição válido, definido para cada interface-bloco, a fim de que o sistema especialista dê como resposta resultados corretos.

3.2.2.3.1 Definição do Instante Válido de Medição

Para se definir o exato instante de medição de um bloco ou de uma conexão, são considerados os seguintes parâmetros das interfaces-bloco: *atraso* , *tempo de manutenção* e o seu *instante de medição* .

Os parágrafos seguintes apresentam as fórmulas que regem os cálculos para determinação do exato instante de medição quando se considera um bloco ou uma conexão:

• **Através de um Bloco:** as entradas e saídas de um bloco devem satisfazer às relações abaixo:

1) instante da interface bloco de saída ≥ instante da interface bloco de entrada + atraso da interface bloco de saída

2) instante da interface bloco de saída ≤ instante da interface bloco de entrada + atraso da interface bloco de saída + tempo de manut. da interface bloco de entrada

Estas relações são suficientes quando existe um e somente um membro do cone de cobertura para cada saída do bloco. No caso de existir mais de um membro no cone de cobertura, estas relações são revistas de forma a satisfazerem as novas condições:

1') instante da interface bloco de saída ≥ MAX(instante das interface bloco de entrada) + atraso da interface bloco de saída

2') instante da interface bloco de saída ≤ MAX(instante das interface bloco de entrada) + atraso da interface bloco de saída + MIN(tempo de manut. das interface bloco de entr.)

• **Através de uma Conexão:** As relações **1** e **2** apresentadas para um bloco podem facilmente ser adaptadas para conexões. Por outro lado, as relações apresentadas em **1'** e **2'** devem ser desprezadas. Isto ocorre devido ao fato de que somente uma interface bloco é conectada a cada uma das extremidades de uma conexão, caso contrário, um curto circuito é detectado. Assim, as relações que definem o instante válido para medição do valor lógico de uma conexão são:

1) instante da interf. bloco de saída da conexão ≥ instante da interf. bloco de entr. da conexão + delay da interf. bloco de saída da conexão

2) instante da interf. bloco de saída da conexão ≤ instante da interf. bloco de entr. da conexão + delay da interf. bloco de saída da conexão + tempo de manut. da interface bloco de entrada

3.2.2.3.2 Detecção de Falhas em Blocos

A escolha da estratégia para detecção de falhas em blocos é feita pelo usuário durante o processo de teste. Estas estratégias são baseadas em regras clássicas de propagação de erros em circuitos lógicos.

Três observações importantes devem ser levadas em consideração antes de se apresentar estas estratégias de detecção de falhas:

1. Circuitos sequenciais são considerados, mas técnicas especiais para eliminação de elos de realimentação não são utilizadas. Isto ocorre porque a partir do momento que se define instantes diferentes para observação de interfaces-bloco, torna-se possível observar o circuito depurado em um modo de funcionamento passo a passo. Desta forma, é possível se ter a qualquer momento da observação, a imagem instantânea do circuito sob teste, tornando-se desnecessária a identificação de realimentações internas ao circuito.

2. Um circuito sequencial é considerado como tal se o particionamento deste circuito apresentar blocos interconectados de forma que realimentações possam ocorrer entre blocos, através de conexões. Neste caso, este particionamento representa um sistema sequencial de blocos. Feita esta consideração, circuitos sequenciais podem ser representados como se fossem combinacionais.

3. Quando circuitos puramente combinacionais são observados, os seguintes parâmetros são desconsiderados:

- *atraso* e *tempo de manutenção* de uma interface-bloco;
- instante válido de medição de um instante de interface-bloco.

Feitas estas considerações, quatro métodos possíveis para localização de falhas em blocos são adotados:

- **Estratégia de Localização de Falha "Combinacional Simples" (SCF):** Quando esta hipótese é selecionada, o sistema especialista procura, primeiramente, todas as interfaces blocos cujo parâmetro "estado" tenha sido declarado "errado", e então, para cada um deles, aplica a seguinte estratégia:

- examina o bloco que tem esta interface como uma de suas saídas;

- se nenhum dos membros do cone de cobertura (entradas do bloco) desta interface-bloco de saída apresenta erro, então este bloco contém uma falha e manifesta um erro do tipo "direto" em uma de suas saídas primárias. Entende-se por "erro direto" como sendo aquele provocado por um bloco que apresenta ao menos uma de suas saídas erradas e todas as suas entradas corretas. No caso de um bloco por onde uma falha apenas se propagou, tem-se um "erro indireto".

- **Estratégia de Localização de Falha "Combinacional Múltipla" (MCF):** Quando esta hipótese é formulada, o sistema especialista procura, primeiramente, todos os blocos que manifestam um erro do tipo "direto" em uma de suas saídas, exatamente da mesma maneira que no caso anterior. A seguir, o processo continua através dos blocos que são assumidos como "errados". Estes blocos são aqueles em cujas saídas o sistema especialista detetou um erro. Assim, a estratégia resume-se em:

- para cada um dos blocos declarados "errados", procura suas saídas onde um erro foi detetado;

- para cada uma destas interfaces blocos, examina o(s) bloco(s) que têm esta interface-bloco como entrada;

- se este bloco ainda não foi declarado como errado, assuma-o como tal;

- aplica esta estratégia a todos os blocos que o seguem em direção à saída do circuito.

- **Estratégia de Localização de Falha "Seqüencial Simples" (SSF) e Estratégia de Localização de Falha "Seqüencial Múltipla" (MSF):** Estas estratégias de teste não serão discutidas pois o processo de execução é exatamente o mesmo que nas duas anteriores, respectivamente, com exceção de que os tempos de aquisição dos estados lógicos das interfaces-bloco obedecem aos instantes válidos de medição, definidos no ítem 3.2.2.3.1.

Note que o processo de localização de falhas é interativo: o usuário faz perguntas à PESTICIDE; se o sistema especialista tem conhecimentos suficientes sobre o circuito e sobre o teste corrente ele pode,

então, prover as respostas diretamente, caso contrário, PESTICIDE pede por novas informações tais como o estado lógico de uma interface bloco ainda não observada. Futuros desenvolvimentos permitirão ao sistema especialista que este peça a aplicação de uma determinada seqüência de vetores de teste.

3.2.2.3.3 Detecção de Falhas em Conexões

Esta estratégia deteta tanto curto-circuitos quanto rupturas em uma conexão, bem como a ocorrência de sinais não suficientemente amplificados. A estratégia usada é a seguinte: se todas as interfaces-bloco ligadas à conexão examinada não tiverem o mesmo valor lógico, então esta conexão apresenta uma falha. Note que, por suposição, o valor lógico é observado em um instante válido, neste caso, é necessário aplicar as estratégias de definição de um instante válido de medição apresentadas no ítem 3.2.2.3.1..

A figura 3.3 ilustra o processo de detecção de falha em uma conexão. Neste exemplo, a interface-bloco IB1 é conectada em avanço na entrada da conexão e as interfaces-bloco IB2 à IB5 são conectadas na saída desta mesma conexão. Assumindo que a expressão "VAL(IB/t)" representa o valor lógico da interface IB observado no instante "t", então a conexão da figura 3.3 é correta, se e somente se, as relações dadas na figura 3.4 são verdadeiras, onde o "delay(IB)" representa o parâmetro *atraso* associado à interface-bloco IB.

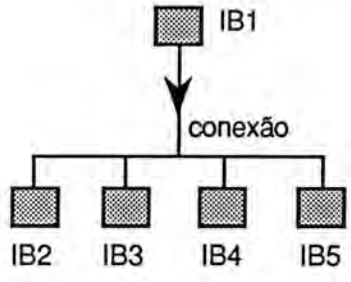


Figura 3.3 Modelagem de uma conexão para depuração.

Os anexos I e II apresentam alguns exemplos de arquivos de entrada para o sistema especialista PESTICIDE. Estes arquivos correspondem à descrição estrutural, funcional e comportamental do circuito CONVERSO, a ser descrito no ítem 6.2.1 do capítulo 6.

VAL(IB1/t) = VAL(IB2/t + delay(IB2))
" .....
" .....
" .....
" VAL(IB5/t + delay(IB5))

Figura 3.4 Relações para validar uma conexão.

É importante mencionar que a ligação entre PESTICIDE e a ferramenta que faz a aquisição e o processamento da imagem do circuito sob teste ainda não está completa, embora o trabalho no Laboratório TIM3 prossiga neste sentido.

3.2.3 Interface PESTICIDE-Ambientes de Projeto e Simulação

Este ítem é dedicado à apresentação de uma interface cujo objetivo é manipular os dados fornecidos por uma ferramenta de auxílio à concepção de circuitos integrados VLSI, HILO3, de forma a moldá-los de acordo com as necessidades exigidas pelo sistema especialista PESTICIDE.

Para tanto, a partir da linguagem de descrição de hardware utilizada por HILO3, um compilador gera, em duas fases, uma tabela de modelos e a seguir uma árvore de blocos. A partir deste instante, outro programa específico de software passa a atuar gerando, simultaneamente, os arquivos contendo as descrições estruturais e funcionais do circuito descrito por HILO3. Estes arquivos fazem parte das bases de conhecimentos ditas estáticas e serão, em seguida, exploradas pelo sistema especialista.

3.2.3.1 Apresentação de HILO3

HILO3 é um simulador de circuitos digitais que facilita a concepção e o teste de novos circuitos. Trata-se de um sistema integrado de ferramentas de concepção que apresentam facilidades para:

- descrição estrutural e comportamental de circuitos;
- simulação e geração automática de vetores de teste.

Assim, HILO3 é capaz de:

- modelar qualquer tipo de circuito digital;
- estimular o modelo através da aplicação de sinais (formas de ondas) nas suas entradas;
- prevêr o comportamento do circuito graças aos seus estímulos aplicados ao modelo;
- gerar uma lista de vetores de teste que servirá ao usuário para programar um testador automático (TESSIE), utilizado para testar o circuito após sua concepção.

HILO3 utiliza as linguagens HDL e WDL: a primeira, para modelar os circuitos e a segunda, para determinar as formas de ondas aplicadas ao modelo, no caso de uma simulação. Desta forma, a presença de dois compiladores faz-se necessária: um compilador para cada uma das respectivas linguagens. A figura 3.5 apresenta a estrutura modular do sistema HILO3.

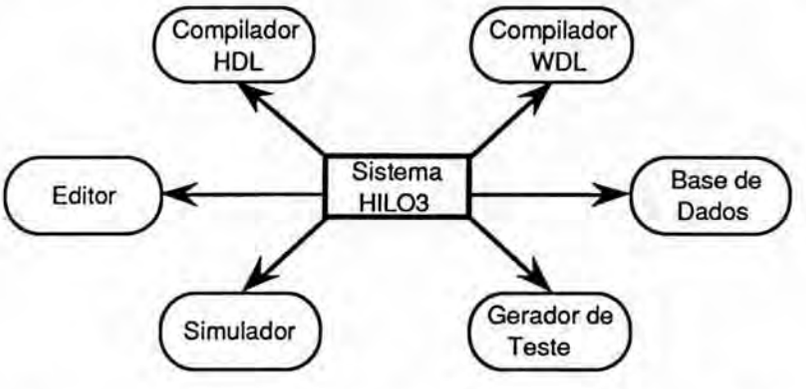


Figura 3.5 Estrutura Modular do Sistema HILO3.

3.2.3.2 Interface PESTICIDE/HILO3

Este ítem é dedicado à apresentação do processo de interfaceamento entre PESTICIDE e HILO3 [BEN89]. Neste sentido, serão vistas as informações que fazem o fluxo de dados entre estes dois módulos, bem como a estrutura de software necessária para tal. No final do processo, obtém-se a geração das bases de conhecimentos estáticas, ou seja: arquivos que descrevem estrutural e funcionalmente o circuito protótipo.

3.2.3.2.1 Informações Fornecidas por HILO3

Do ponto de vista de PESTICIDE, as informações que devem ser fornecidas por HILO3 se limitam ao aspecto funcional. Neste caso, o programa que faz o interfaceamento destes dados gera, nada mais do que uma lista de declarações de diferentes modelos de circuitos utilizados durante a definição do circuito principal. Assim, para cada modelo do circuito principal, HILO3 fornece as seguintes informações:

- nome do modelo;

- informações sobre o modelo tais como: tipo, tecnologia utilizada, atraso,...

- eventualmente uma lista de parâmetros. O circuito será então parametrizado e a passagem de parâmetros deve ser assegurada a partir do momento da definição dos instantes do modelo (ex.: lista de parâmetros que define as condições de passagem de uma interface-bloco bidirecional de entrada para saída ou vice-versa).

- lista de conexões associadas ao modelo;

- lista das interfaces-bloco, associadas a cada uma destas conexões;

- lista de sub-circuitos que, eventualmente, formam este modelo (inexistente, se este é um circuito elementar);

- entradas do circuito em termos de conexões e eventualmente o aspecto funcional de certas conexões (por exemplo, conexões primitivas do modelo).

3.2.3.2.2 Informações Necessárias ao Sistema Especialista

Ao nível da base de conhecimentos estática, as informações necessárias à PESTICIDE são as seguintes:

- os blocos utilizados na modelagem do circuito principal. Para cada bloco, deverá ser fornecido seu **nome** e a **lista de interfaces bloco** a ele associadas;

- as interfaces bloco utilizadas na modelagem do circuito. Para cada uma delas, deverá ser especificado seu **nome, tipo, atraso** e **tempo de manutenção;**

- as conexões utilizadas ao longo da definição do circuito. Para cada uma delas, os seguintes parâmetros devem ser mencionados: **nome, lista de interfaces bloco** que conectam suas extremidades;

- eventualmente, a condição que exprime a passagem de uma interface bloco do estado unidirecional para o bidirecional ou vice-versa, neste último caso, especificando se é uma interface bloco de entrada ou saída;

- o cone de cobertura para cada um dos blocos, ou seja para cada uma das saídas dos blocos definidos, identificar o seu respectivo cone de cobertura.

3.2.3.2.3 Árvore de Blocos

Árvore de blocos consiste na primeira etapa do processo de interfaceamento dos dados entre o sistema HILO3 e PESTICIDE. O objetivo desta etapa é a geração de um texto intermediário, na forma de uma árvore de blocos, a partir do texto fonte HDL, sem erros. A figura 3.6 apresenta um exemplo de geração da árvore de blocos a partir de um dado circuito.

A partir desta estrutura de dados intermediários, no formato de uma árvore de blocos, programas específicos (GEN.C e AFFICHE.C), conforme visto na figura 3.7, geram as bases de conhecimento estrutural e funcional para serem explorados pelo sistema especialista. Para tanto, o método que implementa este processo é dividido em duas fases: a primeira gera uma **tabela de modelos** que constituem o circuito; e a segunda, a partir desta tabela de modelos, constrói a **árvore de blocos.**

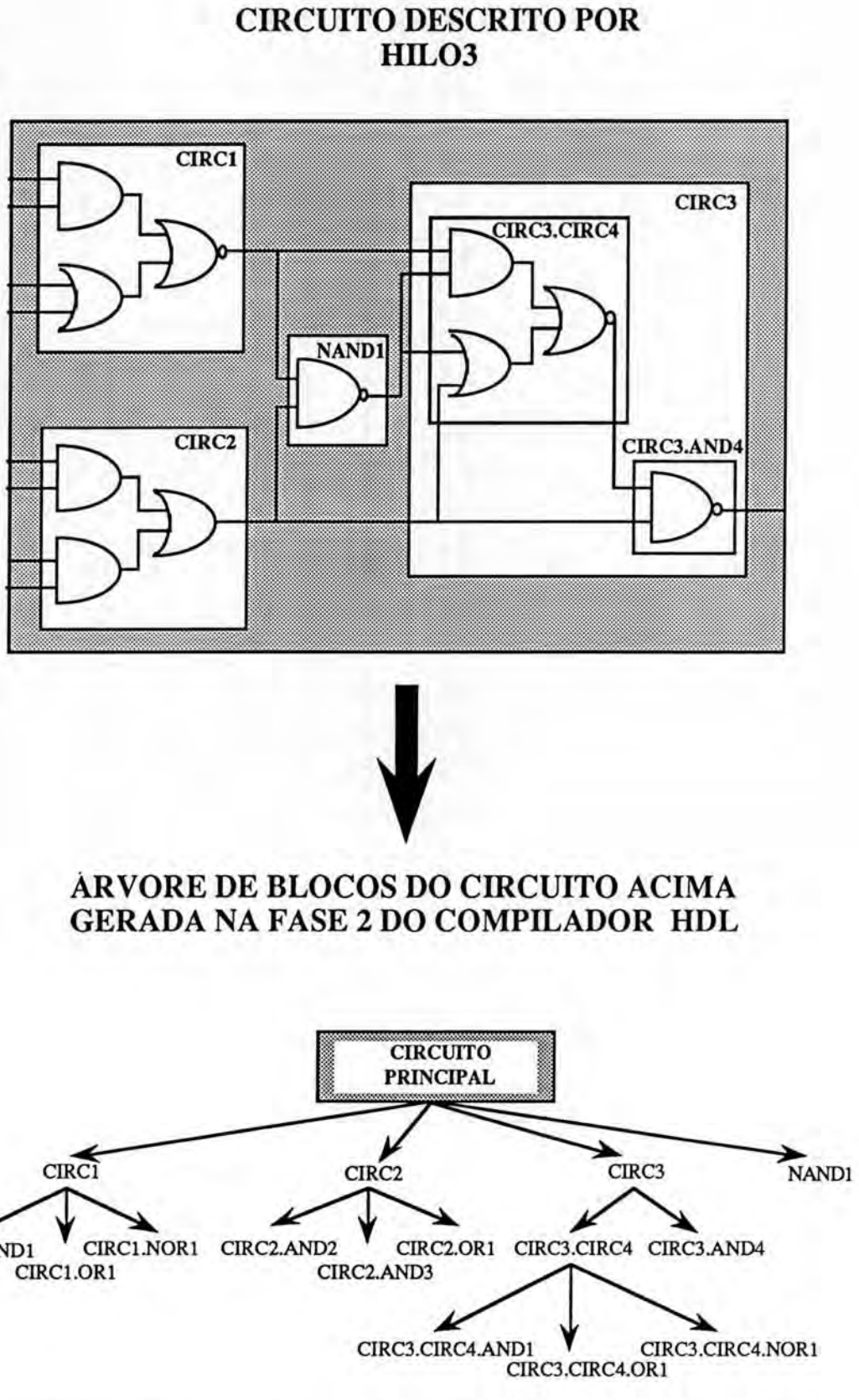


Figura 3.6 Exemplo de geração da árvore de blocos a partir de um circuito.

A primeira fase consiste em percorrer o arquivo fonte, fornecido por HILO3, a fim de gerar a tabela de modelos. Neste instante, o programa que executa esta tarefa verifica:

- se os modelos utilizados já estão todos declarados;
- se o número de parâmetros de cada modelo corresponde ao mesmo número dos parâmetros declarados para este modelo.

Uma vez a tabela de modelos construída e a partir desta, a segunda fase consiste em gerar a árvore de blocos que constitui o circuito. A raiz da árvore será destinada ao último modelo definido na tabela, a saber, o modelo principal. A geração dos sub-nodos (nodos-filhos) é tal que para cada um deles será atribuído um sub-bloco do modelo principal. Desta forma, a geração da árvore é feita descendo-se hierarquicamente através de seus sub-nodos, até a decomposição total do circuito em blocos. As folhas da árvore serão, necessáriamente, as portas lógicas predefinidas (de número de modelo igual à 0). Neste momento, a estrutura de dados está pronta para a geração das bases de conhecimentos estrutural e funcional.

3.2.3.2.4 Geração das Bases de Conhecimento Estáticas

Esta é a segunda etapa do processo de intefaceamento dos dados entre o sistema HILO3 e PESTICIDE.

Cada nodo da árvore gerada no ítem anterior representa um bloco do circuito. Associadas a este mesmo nodo, estão as informações do nome e das interfaces-bloco que se conectam ao bloco definido neste nodo. Para cada uma destas interfaces-bloco, informações adicionais, contendo seu nome, tipo, atraso e tempo de manutenção são deduzidas. Em seguida, os programas geradores das bases de conhecimento (GEN.C e AFFICHE.C) [BEN90] geram uma lista cujos atributos são o nome de cada uma das interfaces-bloco e o conjunto de conexões associado a cada uma delas. Como última etapa, os programas obtém, também a partir da árvore de blocos, os cones de cobertura para cada uma das saídas do blocos do circuito. No final deste processo, os arquivos contendo as descrições estrutural e funcional do circuito estão gerados e prontos para serem explorados pelo sistema especialista.

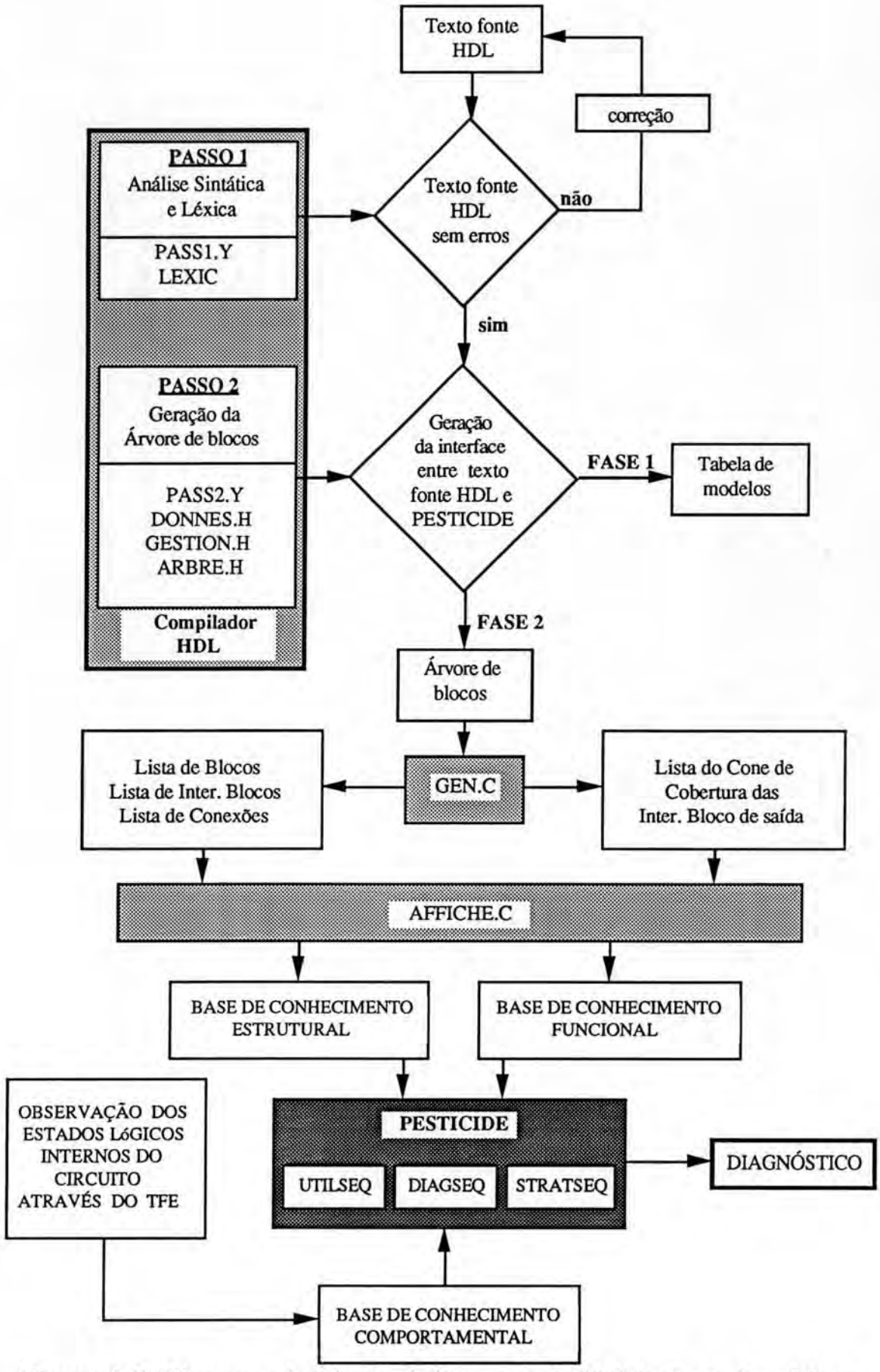


Figura 3.7 Diagrama de Blocos do Processo de Validação de Protótipos

A figura 3.7 apresenta o diagrama de blocos detalhado do processo global de validação de protótipos. O exemplo contido neste diagrama é o do circuito CONVERSO, conversor A/D de 4 bits, a ser apresentado no ítem 6.2.1 do capítulo 6.

Os programas "PASS1.Y", "LEXIC" e "PASS2.Y", "DONNES.H", "GESTION.H", "ARBRE.H" constituem os passos 1 e 2 do compilador HDL, respectivamente. Quanto aos programas "UTILSEQ", "DIAGSEQ" e "STRATSEQ" são programas escritos em Prolog, contendo as regras de inferência que definem as estratégias de detecção de falhas, tanto em blocos quanto em conexões. Os dois últimos arquivos contém as regras de inferência relativas aos circuitos síncronos tipo combinacional e sequencial, respectivamente. Durante o processo de diagnóstico automático da falha, PESTICIDE busca nestes arquivos as informações necessárias para executar a estratégia de teste determinada pelo usuário.

3.3 Validação de Protótipos Baseada em Dicionário de Falhas

3.3.1 Introdução

O projeto ADVICE (Automatic Design Validation of Integrated Circuits using E-Beam Testing) tem por objetivo o desenvolvimento de um sistema de teste com feixe de elétrons completo, integrado e automático [MAR91a].

Este projeto é constituído por três parceiros industriais:

- BTRL (British Telecom Research Laboratories), Inglaterra.
- CNET (Centre National d'Etudes des Télecommunications), França.
- CSELT (Centro Studi E Laboratori Telecommunicazioni), Itália.

e dois parceiros universitários:

- TIM3/IMAG (Laboratoire TIM3 - Groupe d'Architecture des Ordinateurs - Institut d'Informatique et Mathématiques Appliquées de Grenoble), pela França.
- TCDU (Trinity College DUblin), pela Irlanda.

O sistema ADVICE, desenvolvido em uma máquina DEC Vaxstation II/GPX, oferece ao usuário um sistema gráfico iterativo para a validação de protótipos de CI observados através do MEV utilizado no modo de contraste de potencial.

Com este objetivo, o projeto ADVICE se propõe a encontrar soluções para problemas relacionados com a utilização do MEV no teste de circuitos integrados. Estes problemas são assim discriminados:

- Posicionamento do feixe e aquisição de medidas;
- Técnicas de detecção de falha de concepção do circuito sob teste;
- Estratégia de teste, de maneira a minimizar o número de vetores de teste a serem gerados, bem como o número de medidas a serem efetuadas.

3.3.2 Informação Proveniente do Sistema de CAD

Os dados fornecidos pelas ferramentas de CAD são utilizados, por ADVICE, na comparação com os valores medidos pelo MEV durante o processo de depuração.

Como no caso de PESTICIDE, ADVICE também utiliza o sistema HILO3 para obter a descrição lógica do circuito. Além disso, a descrição a nível de máscaras é utilizada e as linguagens escolhidas são CIF ou GDSII. O sistema HILO3 foi escolhido pelos mesmos motivos que levaram a sua adoção por PESTICIDE, na validação de protótipos baseada no conhecimento (ver ítem 3.2.3.1). Tais motivos valem a pena serem lembrados:

Ambiente de simulação completo, proporcionando a:

- descrição lógica do circuito;
- simulação do circuito em dois tipos:
  - simulação livre de falhas ("fault-free simulation");
  - simulação de falhas.
- geração de vetores de teste;
- geração de dicionário de falhas.

Assim, os dados expressos e/ou obtidos através de HILO3 serão comparados àqueles fornecidos pelo MEV a fim de se obter o diagnóstico do circuito em teste.

A utilização das informações provenientes do sistema de CAD permite automatizar as tarefas de posicionamento do feixe de elétrons e de obtenção das medidas. A execução destas tarefas é realizada em três etapas:

- identificação dos pontos a observar sobre o desenho de máscaras;
- posicionamento "grosseiro", através do deslocamento da platina do microscópio;
- verificação da exatidão do posicionamento e ajuste, se necessário, através de um posicionamento mais "fino", utilizando-se técnicas de tratamento de imagem.

As duas primeiras etapas deste processo utilizam como dados as coordenadas dos pontos a observar sobre o desenho de máscaras e os transformam em sinais de comando de deslocamento da platina do microscópio. A terceira etapa faz a correspondência da imagem observada do circuito e de seu desenho de máscaras.

Uma vez as medições obtidas, as formas de ondas lógicas e/ou elétricas são comparadas com os resultados da simulação do funcionamento correto do circuito.

3.3.3 Descrição Sinóptica do Processo

O processo de validação de protótipos apresentado pelo sistema ADVICE é mostrado na figura 3.8, onde pode-se ver a descrição de uma sessão de teste em termos de seqüência de operações a efetuar. Estas operações são descritas a seguir.

As duas primeiras etapas (geração da seqüência de teste e simulação lógica, tradução e carga dos resultados dentro da estrutura interna de dados) são consideradas pré-processos e, como tal, são executadas fora da sessão normal de teste.

Em seguida, a seqüência de teste é transmitida à unidade controladora de CI a fim de se efetuar um primeiro teste, sem o uso do microscópio. Este teste é efetuado ao nível de entradas/saídas primárias do circuito e a unidade controladora de CI funciona no modo analisador lógico.

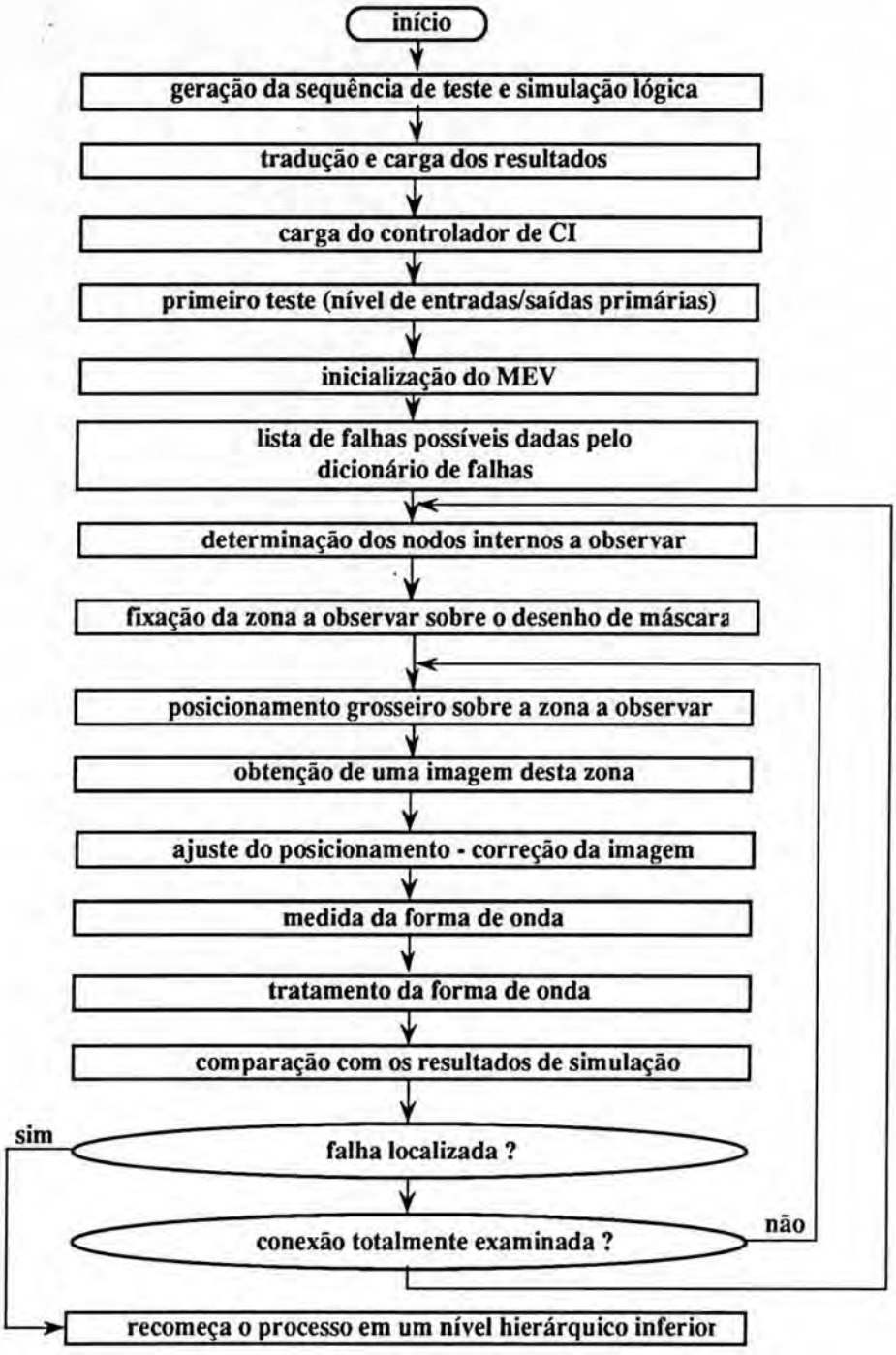


Figura 3.8 Seqüência de operações de uma sessão de teste ADVICE.

O passo seguinte é a inicialização do microscópio e a utilização do dicionário de falhas para uma primeira tentativa de localização de falhas entre os nodos a serem observados no interior do circuito.

Os processos de observação e de aquisição de medidas podem então ser iniciados, fornecendo as formas de onda, que são tratadas (eliminação de ruído) e então comparadas com os resultados de simulação.

Se uma falha é localizada, então tenta-se refinar o diagnóstico, reinicializando-se as mesmas etapas desde o começo do processo, em um nível hierárquico inferior. Se nenhuma falha é localizada, tenta-se observar outros nodos internos do circuito, de mesmo nível hierárquico, definidos pelo dicionário de falhas.

Por outro lado, se nenhuma falha está definida no conjunto de falhas expresso pelo dicionário de falhas, ou então se as observações seguintes mostram que as falhas definidas não são em realidade aquelas procuradas, o algoritmo começa uma análise da estrutura do circuito, à partir de suas saídas errôneas, seguindo em direção às entradas primárias até encontrar uma célula que será susceptível de conter a falha. A partir do momento em que esta célula é identificada, seu conteúdo é examinado, de modo a obter-se um diagnóstico mais preciso, em um nível hierárquico inferior. O algoritmo pára a procura no instante em que é encontrada uma conexão ou uma célula primitiva (de acordo com a descrição do sistema HILO3) defeituosa. Por célula primitiva, entende-se como sendo aquela que não é decomposta em níveis hierárquicos inferiores.

O objetivo deste tipo de diagnóstico é minimizar o tempo e o custo do processo de localização de falhas, através da escolha de um dicionário de falhas. Esta escolha reduz o número de pontos a serem observados com o microscópio, restringindo-os, em uma primeira etapa, àqueles definidos no dicionário de falhas.

Um outro ponto importante a ser mencionado é o modo pelo qual são determinados os nodos a serem observados. Neste caso, três modos de operação estão disponíveis no sistema ADVICE:

- Modo manual: neste caso, o usuário do sistema decide, ele próprio, quais serão os pontos a observar, em função dos resultados da

comparação dos valores medidos através do MEV com àqueles definidos em simulação.

• Modo assistido: o usuário determina os pontos a observar em função de informações obtidas previamente, como por exemplo, uma pré-localização de falhas através de um dicionário de falhas.

• Modo automático: modo no qual a determinação dos pontos a serem observados se faz sem a intervenção do usuário.

3.4 Análise de Falhas

No que concerne ao tema "análise de falhas", a metodologia de teste em desenvolvimento no laboratório TIM3 é baseada na comparação das imagens de dois circuitos cuja estrutura interna é desconhecida: o defeituoso e o de referência, supostamente livre de falhas [COL89], [CON90], [CON91].

Para a localização de um defeito em um circuito cuja estrutura interna não é conhecida, é fortemente sugerida a seguinte condição para a realização do teste:

- É necessário o conhecimento de alguns vetores de teste que provocam a manifestação do(s) defeito(s). Isto pode ser obtido através de um simples teste funcional.

O circuito pode ser combinacional ou sequencial. Assim, a observação de uma mesma área sobre cada um dos dois circuitos, o defeituoso e o de referência, para um mesmo vetor de teste, permitirá a identificação de pontos de discrepância entre as duas imagens. Estes pontos podem ser posteriormente definidos como pontos onde há uma falha ou simplesmente como o caminho por onde a falha se propagou.

Seja um circuito combinacional, com uma falha simples; a observação de uma seqüência de imagens relativas à aplicação de uma seqüência de vetores de teste, que provocam a manifestação de um defeito, permitirá que a falha seja localizada na intersecção dos pontos que apresentam discrepâncias entre as imagens comparadas. Assim, no final deste processo, a intersecção destes pontos conterá apenas um único ponto, que é aquele onde

está a falha. No caso de falha múltipla, pode-se fazer uso da **freqüência de aparição** de um ponto que apresenta discrepâncias. Isto é, um dado vetor de teste t_1 pode conduzir à manifestação de uma falha f_1; por outro lado, um outro vetor de teste t_2 pode, por sua vez, conduzir à manifestação de outra falha f_2. Assim, no final da aplicação de uma seqüência de vetores, os pontos defeituosos corresponderão ao conjunto dos pontos que apresentaram discrepâncias mais freqüentemente.

No que concerne aos circuitos sequenciais, a localização de uma falha é mais complicada pois neste caso a falha pode ser memorizada e a partir daí, induzir o surgimento de erros quando da aplicação dos vetores de teste seguintes sem que, necessariamente, estes novos vetores induzam, realmente, à manifestação de um defeito. Para evitar que este problema ocorra, somente a primeira das imagens que apresenta pontos discrepantes é utilizada no processo de localização da falha. Os vetores de teste seguintes são desprezados e uma nova seqüência de vetores é aplicada até o momento da captura de outra imagem contendo discrepâncias.

3.4.1 Processo de Análise de Falhas Baseado em Técnicas de Comparação de Imagens

Para alcançar o objetivo que é a comparação das imagens do circuito defeituoso e as do circuito de referência e a obtenção das diferenças entre elas, o processo de análise de falhas passa por três fases distintas [CON91]:

- Inicialização dos Parâmetros do Microscópio;
- Deslocamento do Suporte e Aquisição de Imagem;
- Sobreposição das Imagens.

3.4.1.1 Inicialização dos Parâmetros do Microscópio

Esta etapa caracteriza os parâmetros físicos do microscópio e a descrição geométrica dos circuitos. Ela consiste em ajustar os parâmetros tais como a luminosidade, o contraste e o astigmatismo de maneira a obter a melhor qualidade de imagem possível. Em seguida, os dois circuitos são localizados

sobre o suporte da câmara de vácuo do MEV, cada um definido por uma origem comum e um ângulo de rotação em relação ao referencial do suporte. A terceira etapa da fase de inicialização é a definição da zona de observação, definida por dois pontos sobre o circuito de referência. O número de imagens a serem adquiridas para a cobertura desta zona, dependerá do grau de aumento dado pelo microscópio.

4.4.1.2 Deslocamento do Suporte e Aquisição de Imagem

Estas duas funções são controladas automaticamente a partir de uma estação de trabalho que utiliza os parâmetros calculados na fase anterior (origem comum dos circuitos, ângulo de rotação em relação ao referencial do suporte, zona a observar).

O objetivo desta etapa é obter a sobreposição perfeita das duas imagens, e para tanto, a tarefa principal consiste em corrigir os defeitos de alinhamento em translação e os efeitos de rotação. O processo automático de deslocamento da mesa suporte e aquisição das imagens é visto na figura 3.9.

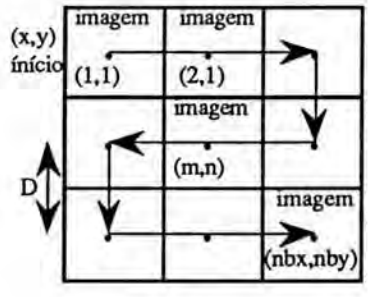


Figura 3.9 Processo automático de deslocamento e aquisição de imagens

3.4.1.3 Sobreposição das Imagens

O processo de sobreposição automática de imagens é uma operação fundamental que deve ser confiável e rápida, de forma a oferecer ao analista uma ferramenta de comparação de imagens eficaz dentro do domínio de um sistema de teste por feixe de elétrons automático e integrado. Com este objetivo, esta ferramenta é gerada utilizando-se técnicas a base de segmentação e de detecção de cantos sobre imagens binárias.

O processo de sobreposição automática de imagem por detecção de cantos sobre imagens binárias consiste em separar cada imagem, a de referência e a contendo o defeito em contornos fechados, a fim de aplicar um algorítmo que procura por ângulos retos.

Com este objetivo, os seguintes procedimentos são executados de modo a tratar e isolar a informação útil (estes procedimentos serão detalhados nas próximas páginas):

- Melhoramento da imagem;
- Linearização automática;
- Eliminação de ruídos;
- Extração de contornos.

Em seguida, os cantos são localizados e memorizados em uma tabela contendo os seguintes atributos:

- Coordenadas;
- Orientação;
- Convexidade.

A partir destes dados, o cálculo do deslocamento é feito tomando-se como referência as posições relativas dos cantos nas imagens dos circuitos defetuoso e de referência. A seguir, os processos de correção de coordenadas e de sobreposição de imagens são executados na seqüência abaixo:

- Cálculo de deslocamento;
- Correção em translação da imagem de referência;
- Sobreposição das imagens a comparar.

3.4.1.3.1 Tratamento e Seleção das Informações Úteis

Melhoramento da Imagem: as imagens obtidas em contraste de potencial oferecem geralmente uma qualidade medíocre, não podendo ser utilizadas diretamente nas etapas seguintes do processo de sobreposição de imagens. Os fatores que levam a esta baixa qualidade são:

• primeiro, a utilização da estroboscopia do feixe de elétrons introduz uma degradação da imagem inversamente proporcional à duração do tempo de exposição do feixe.

• segundo, outro fator limitante da qualidade das imagens é associado à natureza da superfície varrida: depassivação e contaminação do circuito a ser observado. A depassivação deve ser realizada de maneira uniforme, a fim de evitar a variação de luminosidade devido à carga e descarga da camada de óxido fino sobre o circuito. O efeito destas zonas não depassivadas é a aparição de listas ou pontos escuros sobre a imagem obtida. Quanto à contaminação, esta é uma degradação da superfície em função duração do tempo de exposição ao feixe de elétrons. Este é um fenômeno cumulativo, e ao nível de imagem apresenta-se como sendo a degradação do contraste até o seu escurecimento completo.

Desta forma, constata-se que a qualidade da imagem é frágil, que depende do material, de sua manipulação e da tarefa de preparação do circuito. Para o melhoramento da qualidade desta imagem, são utilizadas **técnicas de integração** e **transformação de imagens no domínio espacial.**

• Integração de Imagens: esta técnica consiste em melhorar a relação sinal/ruído da imagem no momento da captura através da média entre várias imagens. Assim, se definirmos por $m_i(x,y)$ cada imagem e $r_i(x,y)$ o ruído a ela associado, o sinal original f(x,y) da imagem não alterada é satisfeito pela relação:

$$m_i(x,y) = f(x,y) + r_i(x,y)$$

onde i = 1, 2, 3,..., n imagens amostradas.

O cálculo da média aritmética destas n imagens permite avaliar a nova relação sinal/ruído [PAP65]:

seja

$$E(m) = \overline{m_i(x,y)} = \frac{1}{n} \sum_{i=1}^{n} [m_i(x,y) = f(x,y) + r(x,y)]$$

com

$$r(x,y) = \frac{1}{n} \sum_{i=1}^{n} b_i(x,y)$$

Tais operações mostram que o cálculo de E(m) não modifica a imagem original f(x,y). Por outro lado, o ruído resultante apresenta uma variação dada por:

$$\partial_r = \frac{\partial}{\sqrt{n}}$$

onde ∂ representa o ruído típico de uma imagem não amostrada.

Assim, a nova relação sinal/ruído da imagem integrada, $m_i(x,y)$ passa a ser dado por:

$$S/R = \frac{f(x,y)}{\partial_r} = \frac{f(x,y)}{\partial} \cdot \sqrt{n}$$

O termo $f(x,y)/\partial$ é a relação sinal/ruído de uma imagem comparada com seu próprio ruído. Como conclusão, esta última relação mostra que a operação da média das imagens fornece uma melhora de $\sqrt{n}$ em relação ao sinal/ruído original da imagem observada.

De acordo com experiências práticas, o número de imagens amostradas no laboratório é igual a 10. Este valor oferece um bom compromisso entre a carga e a contaminação do circuito, e o tempo de aquisição e a eliminação do ruído.

• Transformação de Imagens no Domínio Espacial: esta técnica de filtragem espacial consiste em uma operação direta sobre os "pixels" e seu atributo principal: o nível de cinza. Este procedimento opera uma transformação que considera para cada "pixel" a sua vizinhança. Para tanto, uma técnica denominada de "Média dos Vizinhos" é utilizada.

Este filtro utiliza a média não ponderada do nível de cinza dos vizinhos próximos ao "pixel" que deseja-se modelar para reduzir o ruído em

imagens obtidas à alta freqüência (como no caso de técnicas de estroboscopia). O nível de cinza de cada "pixel" é substituído pela média dos valores de seus 8 vizinhos, como pode ser visto na figura 3.10; onde i e j são as coordenadas dos "pixels" pertencentes ao conjunto E de 8 vizinhos; f(x,y) representa o nível de cinza de cada um destes pontos e g(x,y) é o nível de cinza do "pixel" corrente, ou seja, é a imagem filtrada.

$$g(x,y) = \frac{1}{n} \sum_{(i,j)\, C\, E} f(i,j)$$

V V V
V ⓪ V
V V V

Figura 3.10 Filtragem pela "Média dos Vizinhos".

Linearização Automática: esta etapa consiste em determinar, automaticamente, um limite t a fim de binarizar a imagem. Para tanto, este procedimento baseia-se na análise de histogramas de níveis de cinza. Assim, ao final desta etapa a imagem é reduzida a apenas dois níveis de cinza (ver ítem 2.1.1, Seleção de "Threshold" Global).

Eliminação de Ruídos: a tarefa principal deste procedimento é preparar a melhor imagem binária para a etapa final de detecção de cantos. Sua ação de filtragem permite linearizar os contornos dos objetos eliminando os ruídos e facilitando a procura de cantos sobre as flutuações das bordas dos objetos. Atenção especial deve ser dada ao fato de que este procedimento pode alterar demasiadamente a forma dos objetos.

Extração de Contornos: o processo de evidenciação dos contornos é realizado por um extrator de contorno. Ao final deste processo obtém-se os contornos fechados dos objetos, com uma espessura de apenas um "pixel", e cuja imagem codificada é fornecida em três níveis de cinza:

- preto para o "fundo" da imagem;
- branco para o "objeto";
- cinza médio para o interior do objeto.

3.4.1.3.2 Detecção e Memorização de Cantos

A detecção de cantos é feita sobre os contornos binários espaçados de um "pixel". A imagem é formada por três níveis de cinza. Seu princípio consiste em determinar sobre a imagem, os ângulos retos compostos de duas cadeias numéricas: segmentos de seis "pixels" contíguos orientados em direções ortogonais. Durante esta etapa, é gerada uma tabela de cantos contendo os atributos: **coordenadas, orientação** e **concavidade.** A função desta lista é possibilitar a formação de pares de cantos segundo os seguintes critérios:

- Os tipos de canto devem ser idênticos (tipo 1,2,3, ou 4). (Ver figura 3.11);
- Os cantos devem possuir o mesmo ângulo (côncavo ou convexo).

3.4.1.3.3 Correção de Coordenadas e Sobreposição de Imagens

Nesta etapa é realizada a sobreposição das imagens a comparar. Para tanto, os processos de **cálculo do deslocamento** e de **correlação das imagens** são executados através da determinação das diferenças das imagens.

Fazendo-se uso da tabela de cantos definida na etapa anterior, o cálculo de deslocamento é feito a partir da determinação da diferença da posição relativa de cada par de cantos, segundo os três critérios básicos:

I. orientação dos cantos (tipos 1, 2, 3 ou 4, conforme visto na figura 3.11);
II. concavidade dos cantos;
III. posição (a diferença de posição, dada em "pixels", entre os cantos a comparar, deve ser inferior ao passo* entre duas conexões). Esta

---

* **passo**: distância mínima entre duas conexões; seu valor depende da tecnologia utilizada durante a fase de projeto.

condição é necessária pois corre-se o risco de se comparar cantos idênticos pela forma, mas diferentes pela posição.

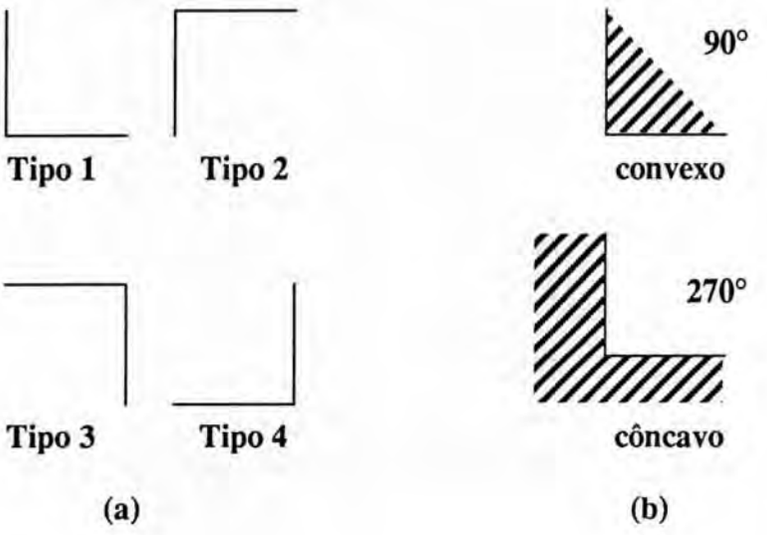


Figura 3.11 Tipos de cantos, definidos de acordo com dois critérios: (a) sua orientação, (b) sua convexidade.

Estas diferenças são obtidas separadamente, nas duas direções x e y. As diferenças mais frequentes são retidas para serem utilizadas na correção das imagens a comparar. Com este objetivo, realiza-se um cálculo baseado em curvas de histogramas que representam o número de pares de cantos em função de suas respectivas diferenças, dadas em "pixels". Dentro desta representação, os valores máximos das curvas do histograma fornecem os índices de deslocamento mais prováveis para se efetuar a correlação em translação das imagens.

4 PROJETO VISANDO A TESTABILIDADE NO TESTE COM FEIXE DE ELÉTRONS

4.1 Introdução

A complexidade e o alto grau de seqüencialidade dos circuitos VLSI criaram grandes dificuldades na tarefa de verificação dos parâmetros de projeto. A adoção do MEV, funcionando especialmente no modo estroboscópico, tornou ainda mais rígidas as regras para um projeto visando a testabilidade quando se deseja utilizar, além das técnicas comumente empregadas dentro do processo de teste de circuitos integrados, este equipamento, operando em modo de contraste de potencial.

O objetivo deste capítulo é apresentar algumas regras de projeto de CI que forneçam àqueles que trabalham nesta área, desconhecedores dos problemas do MEV, idéias de como realizar seu projeto a fim de tornar a tarefa de depuração do protótipo pelo feixe de elétrons o mais fácil possível.

Devido ao fato de que os maiores problemas estão relacionados à qualidade das medidas em circuitos passivados, sobre condutores colocados em camadas inferiores, e em áreas altamente perturbadas por campos induzidos por outros condutores carregados, dois conjuntos básicos de regras são determinados: o primeiro refere-se à **topologia do circuito**, e o segundo, ao **processo de seleção dos pontos a serem observados** pelo feixe de elétrons [LEE89], [MEL88]. Estas regras possuem um objetivo comum: melhorar a qualidade das medidas lógicas e elétricas realizadas no circuito ao longo do processo de teste.

4.2 Regras relacionadas à topologia do circuito

As regras relacionadas ao projeto topológico visam prover, ao "layout" do circuito, pontos de teste que permitam uma boa qualidade do sinal

medido, ao mesmo tempo que tentam reduzir ao máximo o impacto das restrições que o TFE impõe ao estilo de projeto.

Devido às técnicas de acoplamento capacitivo, circuitos passivados podem ser observados no MEV. Entretanto, somente circuitos depassivados são normalmente usados na fase de validação de projeto. Porém, às vezes não é possível a utilização de circuitos depassivados por motivos tais como:

- circuitos depassivados não estão disponíveis ou então quando é perigoso realizar o processo de corrosão da camada de passivação, a fim de se alcançar o ponto desejado;

- o efeito de contaminação pode danificar o circuito.

Para se contornar estes problemas, algumas regras de projeto podem ser seguidas a fim de facilitar a tarefa do TFE:

- uma máscara pode ser criada para definir uma pequena área de metal sobre o óxido de passivação e assim prover a segunda placa de um capacitor. Desta forma, poderá ser realizada uma medição por acoplamento capacitivo mais estável, de melhor qualidade. (Ver figura 4.1.b).

- durante a fase de projeto, aberturas podem ser deixadas no óxido, de forma a alcançar pontos de teste pré-definidos, considerados de crucial importância na observação ao longo do processo de depuração.

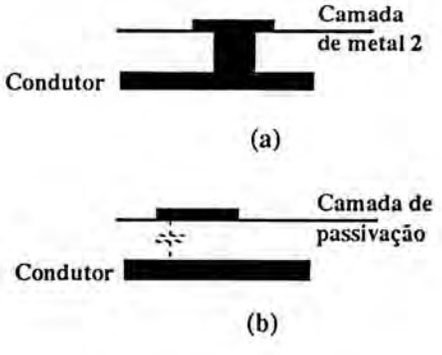


Figura 4.1 "Pads" de teste. (a) Diretamente conectado ao condutor enterrado; (b) capacitivamente acoplado ao condutor enterrado.

Caso o circuito seja depassivado, duas medidas podem ser colocadas em prática durante a fase de projeto:

- todos os nodos a serem observados devem possuir acesso à camada de metal 2 através de pequenos "pads" internos, onde o feixe de elétrons será posicionado. (Ver figura 4.1.a).

- a fim de se aumentar a qualidade destas medições é proposta a realização, próximo destes "pads", de pequenas áreas metálicas, ligadas à massa do circuito, de forma a reduzir a interferência dos campos eletrostáticos das vizinhanças no ponto a ser medido.

Um outro ponto muito importante a ser considerado é o da geração da seqüência de teste a ser aplicada ao circuito. Isto é, deve-se gerar um conjunto de vetores de teste que excitem o circuito de forma que seja provocada a manifestação da falha.

Para cada período da seqüência de teste, duas etapas podem ser distinguidas: a de **inicialização**, a fim de colocar o circuito em um estado interno predefinido, e a de **excitação**, propriamente dita.

Com o intuito de "encurtar" a etapa de inicialização, tornando-a mais fácil e rápida, algumas estruturas podem ser incluídas no circuito durante a fase de projeto. Estas estruturas visam aumentar a observabilidade e a controlabilidade, e podem ser classificadas da seguinte maneira:

- Estruturas de "reinicialização";
- Estruturas de particionamento;
- Estruturas de varredura.

Estruturas de "reinicialização", são as mais simples, mas de fundamental importância, pois elas permitem conduzir dispositivos sequenciais para estados conhecidos, definidos como o ponto inicial para a evolução do teste subseqüente.

Estruturas de particionamento, cujo objetivo é dividir o circuito em blocos funcionais, facilmente controláveis e observáveis. A partir desta condição, o circuito particionado é testado em blocos separados, facilitando a geração da seqüência de teste e isolando as possíveis falhas em seus

respectivos blocos, evitando assim, a sua propagação para outras partes do circuito.

Estruturas de varredura, quando adotadas, merecem um cuidado todo especial quanto as suas extensões. Isto é, estruturas de varredura muito grandes requerem uma seqüência de relógio muito longa para carga/descarga dos registradores deslocadores, violando desta forma as limitações do equipamento de teste e de tempo de execução da tarefa.

4.3 Regras relacionadas à seleção dos pontos de teste

O objetivo deste conjunto de regras é fornecer meios que facilitem ao projetista a tarefa de seleção dos pontos a serem testados. Estas regras são divididas em três subclasses:

- regras relacionadas ao diagnóstico;
- regras relacionadas ao reconhecimento de padrões;
- regras relacionadas à qualidade do sinal medido.

Regras relacionadas ao diagnóstico: estas regras objetivam a definição dos pontos e do caminho de propagação de seus respectivos sinais a serem medidos. Para tanto, são definidas todas as entradas e saídas de cada uma das células, seus níveis de hierarquia no dispositivo, além dos seus instantes válidos de medição.

Feitas estas considerações, os mesmos conceitos que conduziram à seleção dos nodos a serem monitorados e seus resultados armazenados durante o processo de simulação podem ser aplicados na estratégia de seleção dos pontos a serem observados pelo feixe de elétrons.

Regras relacionadas ao reconhecimento de padrões: estas regras visam otimizar o tempo e a área necessária, para que o processo de tratamento de imagem realize a tarefa de reconhecimento de padrões.

Estes problemas são bem evidenciados em estruturas regulares :

- nas quais o intervalo de repetição é inferior ao erro de posicionamento do feixe;

- cuja área é maior que a largura da janela de observação.

Assim, estruturas altamente regulares deveriam ser evitadas ao máximo, procurando-se variar a forma de padrões repetidos, principalmente daqueles que apresentam os menores intervalos de repetição.

Regras relacionadas à qualidade do sinal medido: estas regras são utilizadas a fim de obter-se uma melhor qualidade dos sinais medidos em forma de onda. Como recomendações importantes, pode-se citar:

- se já existirem "pads" de teste no circuito, inclua-os no conjunto de pontos a serem observados;

- selecione tantos pontos quanto possível no nível de metal 2;

- evite pontos sobre um barramento circundado por outros barramentos conduzindo sinais em alta freqüência a fim de evitar efeitos de sobreposição de campos gerados por conexões vizinhas;

- evite pontos de cruzamento entre metal 1 e metal 2, pois a corrente de elétrons secundários pode ser seriamente afetada pelo campo elétrico gerado pelos condutores do cruzamento.

5 CONCLUSÃO DA PARTE I

Na primeira parte do trabalho foram tratados assuntos relativos à teoria e aplicações do teste com feixe de elétrons. Assim, temas como:

- os seus princípios básicos, relacionados ao fenômeno físico da emissão de elétrons;
- tratamentos de imagem aplicados especificamente àquelas obtidas através do contraste de potencial do MEV;
- técnicas de validação de protótipo e de análise de falhas, bem como o projeto visando a testabilidade, adaptados ao teste com feixe de elétrons,

foram apresentados e discutidos.

Conforme visto no capítulo 1, de acordo com as características e as técnicas de observação do TFE, suas vantagens em relação às tradicionais ponteiras mecânicas no que se refere aos problemas de controlabilidade e observabilidade do circuito em teste são evidentes.

Por outro lado, devido à grande quantidade de informação fornecida pelo MEV, o problema de gerenciamento e de tratamento desta tornou-se uma tarefa árdua e complexa. Nesta situação, é necessário, cada vez mais, o desenvolvimento de técnicas que automatizem os processos de depuração do circuito testado, mais especificamente, os processos de validação de protótipo e de análise de falhas.

Quanto à definição das técnicas de automação do processo de teste com feixe de elétrons, esta diz respeito a dois pontos importantes:

- *escolha correta das técnicas de tratamento de imagem* (seleção de "threshold" e detecção de cantos) de forma a utilizar de maneira adequada os recursos de "hardware" disponíveis no local de pesquisa, isto é, o

tipo do computador, a capacidade de memória do sistema, a existência de processadores dedicados ao tratamento de imagem ou não, etc;

• *definição de uma metodologia de teste e de diagnóstico de falha* realista e confiável dentro de tempos de execução aceitáveis. Assim, problemas práticos, tais como o acesso do feixe de elétrons aos pontos internos do circuito testado, o tempo de realização desta tarefa e a confiabilidade das medições realizadas, devem ser fortemente considerados. Além disso, é importante citar que, o projeto visando a testabilidade com feixe de elétrons assume um papel muito importante no processo de escolha da metodologia de teste mais adequada, pois a quantidade (e qualidade) das informações fornecidas para o diagnóstico de falha preciso são baseadas na possibilidade de acesso aos sinais internos do circuito testado.

PARTE II

EXPERIMENTOS PRÁTICOS EM VALIDAÇÃO DE PROTÓTIPOS

6 ESTUDO DE UM CASO REAL EM VALIDAÇÃO DE PROTÓTIPOS

6.1 Introdução

Neste capítulo são descritos experimentos práticos na área de validação de protótipos. Duas técnicas são utilizadas, PESTICIDE e MAICOP, seus desempenhos obtidos e os resultados discutidos, bem como é fornecido o diagnóstico de falha para o circuito protótipo utilizado.

Como uma das conclusões destes experimentos práticos, são propostos futuros desenvolvimentos no processo de validação de protótipo baseado no conhecimento. Estes melhoramentos são descritos no capítulo 7 e objetivam tanto a completa automação do processo quanto o enriquecimento da informação provida no final do processo de diagnóstico de falha, de forma a obter-se um ambiente de teste para validação de protótipos apresentando um alto grau de integração e automação.

Este capítulo é dividido em duas partes distintas. A primeira, "Avaliação do Desempenho do Sistema Especialista", apresenta o circuito protótipo a ser validado, suas características estruturais e funcionais, bem como descreve os experimentos práticos realizados com o sistema especialista PESTICIDE. Estes experimentos práticos tiveram como objetivo avaliar o desempenho do sistema especialista baseado no conhecimento através da simulação de falhas em um circuito protótipo e da geração do respectivo diagnóstico de falha.

A segunda parte, "Usando o TFE para Depurar um Circuito Protótipo", descreve, por sua vez, os experimentos práticos desenvolvidos sobre uma nova técnica para validação de protótipos: MAICOP, Multiple Adjacent Images Comparison Process". Esta nova técnica é baseada na adaptação de uma ferramenta, originalmente desenvolvida para o processo de análise de falhas, cuja função era a comparação de múltiplas imagens adjacentes. Estes experimentos objetivaram avaliar a nova técnica de

comparação de imagens para validação de protótipos através da análise de seu desempenho durante o processo de depuração de um circuito real e do fornecimento de seu diagnóstico de falha. Para tanto, imagens foram comparadas, duas a duas; cada imagem, contendo as informações relativas aos nodos internos do circuito para diferentes vetores de entrada. No final do processo, é gerado um conjunto de imagens resultantes das comparações, contendo os pontos de discrepâncias entre dois vetores de entrada distintos para o circuito em teste. Baseado na análise deste conjunto de imagens e no conhecimento estrutural e funcional do circuito protótipo, o diagnóstico de falha é proposto.

Ao longo da realização do trabalho prático descrito neste capítulo, quatro artigos [MAR91b], [MAR91c], [RUS91], [MAR91d] foram publicados em três congressos de âmbito nacional e internacional.

6.2 Avaliação do Desempenho do Sistema Especialista

Esta etapa teve como objetivo avaliar o desempenho do sistema especialista PESTICIDE. Para tanto, as seguintes tarefas foram implementadas ao longo do experimento:

- simulação de falhas em trilhas e em blocos (ver anexos I e II);
- simulação de falha simples: injetou-se falhas em 10 pontos distintos do circuito;
- simulação de falha múltipla: injetou-se de 2 a 10 falhas múltiplas no circuito;
- variação do número de instantes de medição sobre cada uma das interfaces-bloco definidas para o circuito; para cada interface-bloco, foram realizadas de 1 a 15 medições (ver definição de *interface-bloco* e de seu *instante de medição* no ítem 3.2.2.1, capítulo 3).
- variação das formas de modelagem do dispositivo dentro das bases de conhecimento do sistema especialista, particionando o circuito de várias formas e em vários níveis hierárquicos. Assim, o circuito foi particionado:
  - ora como um modelo combinacional, ora como seqüencial (sem alteração do comportamento original do circuito);
  - em um, dois, e três níveis hierárquicos.

6.2.1 O Circuito Protótipo

Este circuito trata-se de um conversor A/D de 4 bits, denominado de CONVERSO [DOS89]. Ele foi projetado e processado, respectivamente, pela Universidade Federal do Rio Grande do Sul (Brasil) e ES2 (França). A fotografia 2.1 (página 27) mostra uma vista geral do circuito protótipo. Suas principais características são:

- • Aplicação: processamento de imagem em sinal de vídeo;
- • Regras de Projeto: segundo normas fornecidas pela ES2, tecnologia CMOS 2μ, 2 níveis de metal (regras ECDM20);
- • Área: 3,5 mm^2;
- • Número de portas lógicas da parte digital: 40;
- • Freqüência máxima de operação: 10 MHz.

O circuito é composto por três blocos funcionais (figura 6.1):

• **ANA**: bloco analógico, cuja função é realizar uma pré-conversão da entrada analógica, traduzindo-a em 6 sinais digitais. Estes sinais são enviados para o bloco digital para serem manipulados em duas etapas: 3 sinais em cada etapa;

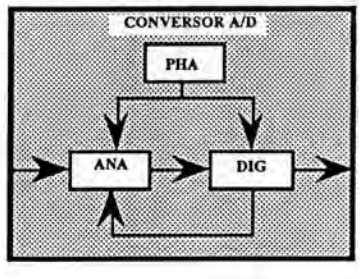


Figura 6.1 Diagrama de bloco do circuito CONVERSO

• **DIG**: bloco digital do conversor. Este bloco é encarregado de manipular a informação (6 sinais) proveniente da parte analógica, fornecendo em suas saídas o sinal digital correspondente. Depois de manipular os 3 primeiros sinais, este bloco fornece instruções para a parte analógica a fim de que esta forneça os 3 últimos sinais. Em cada etapa, a parte digital provê em suas saídas primárias 2 bits: primeiramente os 2 bits mais significativos (MSB) e em seguida, os 2 menos significativos (LSB);

• **PHA**: bloco gerador de fase, cuja função é gerar duas fases de relógio não sobrepostas, a partir de um único sinal de entrada;

6.2.1.1 Parte Analógica

Esta unidade tem como função traduzir o sinal analógico de sua entrada primária em 6 sinais discretos (C_0 a C_5). Estes sinais são enviados à parte digital em 2 etapas: C_3 a C_5 na primeira e, de acordo com o sinal de comando da parte digital, os sinais C_0 a C_2 na segunda etapa. Os parágrafos seguintes descrevem a implementação e as características físicas do conversor.

Os quantizadores fino e grosseiro, a cadeia de resistores e a matriz de chaves formam a parte analógica do circuito CONVERSO. Na figura 6.2 é mostrado o seu diagrama elétrico [DOS89]. Por simplicidade, as chaves CMOS e NMOS foram representadas no diagrama por símbolos de chaves comuns, comandadas pelos sinais indicados. O quantizador grosseiro é composto por 3 comparadores de balanço de cargas ligados às derivações primárias da cadeia de resistores e à tensão de entrada através de chaves comandadas pelas fases 1 e 2, respectivamente. O quantizador fino é formado por outros 3 comparadores idênticos aos do comparador grosseiro, mas com uma das chaves de entrada substituída pelo conjunto de 4 chaves da matriz. Estas chaves são comandadas por sinais vindos da parte digital, sincronizados com a fase 1, e conectam os comparadores às derivações secundárias do divisor resistivo. Além disso,um comparador para testes está incluído na parte analógica. Seus dois terminais de entrada e o terminal de saída estão conectados ao exterior do circuito através de "pads" independentes. Este comparador extra compartilha das linhas de alimentação e de relógio dos demais comparadores. A figura 6.3 mostra o esquema elétrico a nível de transistores de um comparador [DOS89].

Os capacitores dos comparadores são implementados em polissilício e apresentam forma retangular, com dimensão de 180 X 90 μm^2. Conforme visto na figura 6.3, à direita do capacitor do comparador do quantizador grosseiro (a) estão os dois inversores, a chave de autozeramento (b) e o transistor de compensação (c). No lado esquerdo do capacitor estão as duas chaves CMOS de entrada.

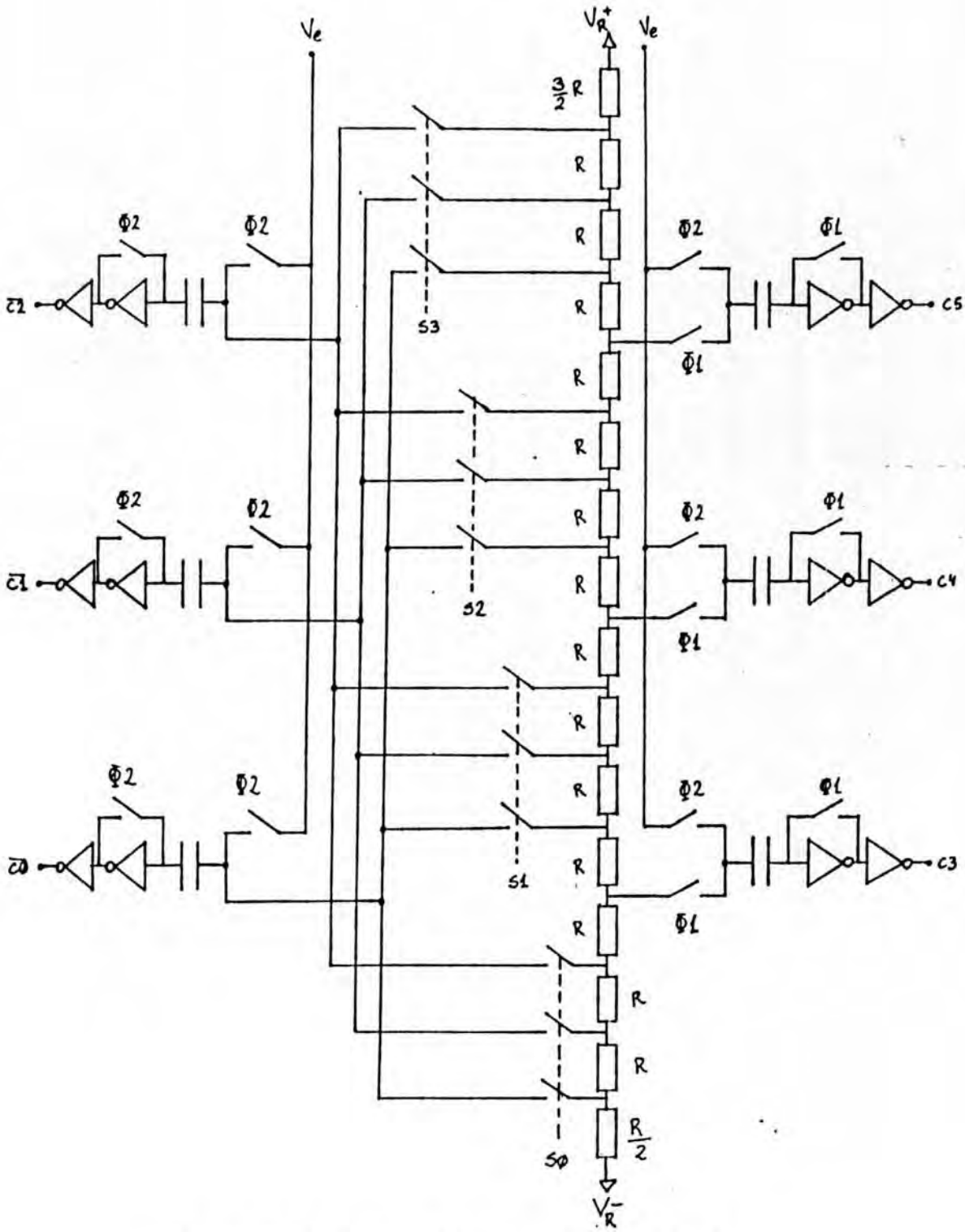


Figura 6.2 Diagrama Elétrico da Parte Analógica

Quanto aos comparadores do quantizador fino, estes se diferenciam dos demais pela ausência de uma das chaves, que é substituída por parte da matriz. Nos dois "layouts", as tensões de alimentação e os sinais de comando das chaves, comuns a todos os comparadores de cada quantizador,

são conduzidos por linhas de metal 2 orientadas perpendicularmente ao comprimento dos comparadores.

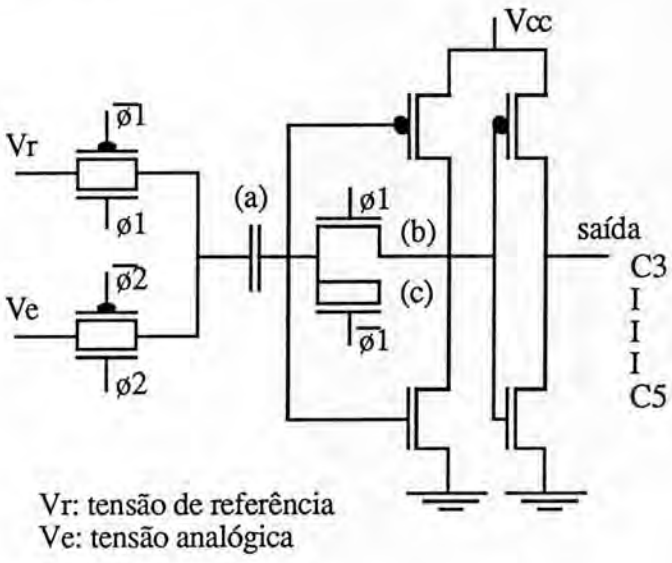


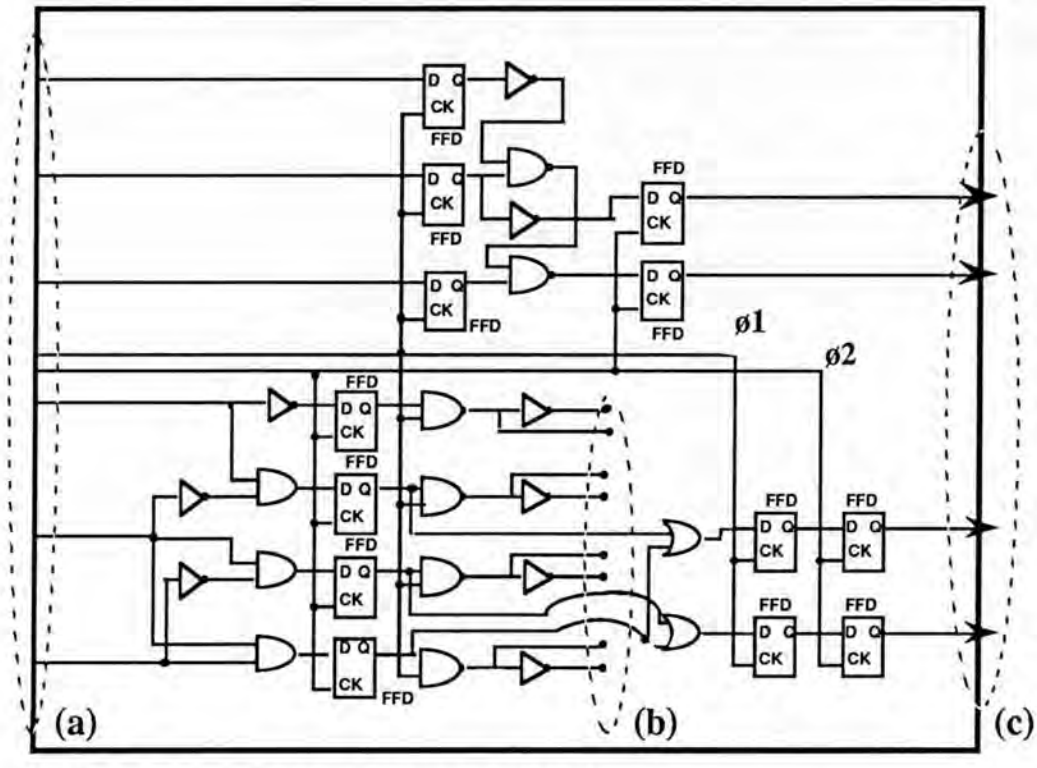
Figura 6.3 Esquema elétrico de um comparador do quantizador grosseiro.

A cadeia de resistores possui 15 derivações igualmente espaçadas entre si: três primárias, ligadas ao quantizador grosseiro, e 12 secundárias, ligadas ao quantizador fino através de chaves. O primeiro resistor da cadeia, ligado à tensão de referência positiva, e o último, ligado à tensão de referência negativa tem valores, respectivamente, 1,5 e 0,5 vezes o valor dos demais resistores. Esta cadeia de resistores, cujo valor total de sua impedância é de 12,5 KΩ, é implementada em polissilício, sob a forma de uma faixa ininterrupta, dobrada várias vezes sobre si mesma. As derivações são tomadas nas dobras da faixa, as quais delimitam os 15 segmentos centrais (iguais) do resistor. Segmentos menores, incluídos nas duas extremidades para formar o primeiro e o último resistor, completam a cadeia.

6.2.1.2 Parte Digital

A parte digital é bastante simples, sendo implementada, a partir da biblioteca de células TRANCA [REI88], em um único bloco de 33 células. Sua função é controlar e manipular a informação proveniente da parte analógica, fornecendo ao mesmo instante, o sinal digital correspondente em suas saídas primárias.

A figura 6.4 apresenta o diagrama lógico da parte digital do circuito CONVERSO. Os sinais C_0 a C_5 **(a)** vêm dos quantizadores. ø1 e ø2 são sinais de relógio, provenientes de um circuito gerador de fases. A saída digital do conversor é formada pelos sinais B_0 a B_3 **(c)**. O comando da matriz de chaves CMOS da parte analógica **(b)** é feito pelos sinais S_{n0} a S_{n3}, que comandam as chaves NMOS, e S_{p0} a S_{p3}, que comandam as chaves PMOS. É através da seleção destas chaves que a parte digital controla o "ajuste fino" realizado pela parte analógica. Este ajuste é realizado da seguinte maneira: quando os sinais C_3 a C_5 são recebidos pela parte digital, esta fornece, simultaneamente, os dois bits mais significativos do sinal digital e os sinais S_{n0} a S_{n3} e S_{p0} a S_{p3} à parte analógica, através da seleção de um par de chaves CMOS/NMOS. Ao final desta operação, a parte digital recebe os sinais C_0 a C_2 provenientes dos quantizadores finos, fornecendo em suas saídas primárias os dois bits menos significativos do sinal digitalizado.



(a) sinais provenientes do bloco analógico;
(b) sinais de controle para as chaves dos comparadores, no bloco analógico;
(c) saídas primárias: sinais digitais.

Figura 6.4 Diagrama Lógico da Parte Digital

A geração das duas fases não-sobrepostas necessárias para o comando do conversor a partir do sinal de relógio externo é feita pelo gerador de fases mostrado na figura 6.5.

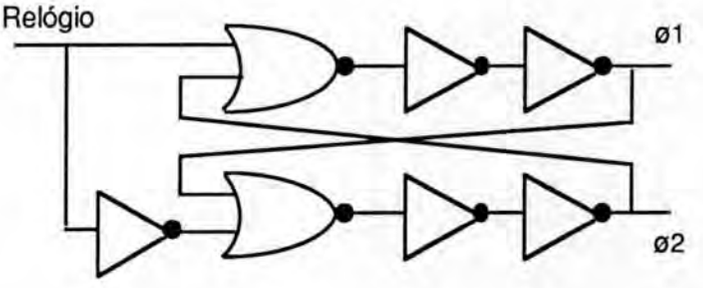


Figura 6.5 Gerador de Fases.

O circuito apresenta a possibilidade de tornar a operação do conversor independente do correto funcionamento do gerador de fases, pois estas podem ser enviadas para fora do circuito através de "pads" de saída. Se os sinais forem satisfatórios, serão redirecionados para o interior do circuito através de dois pads de entrada de fases, que permitem também o comando do conversor por sinais externos, caso o gerador não funcione adequadamente.

6.2.1.3 Planta Baixa

As figuras 6.6 e 6.7 mostram a planta baixa e o "layout" final do circuito, cuja área total apresenta as dimensões de 2356 X 1479 μm^2.

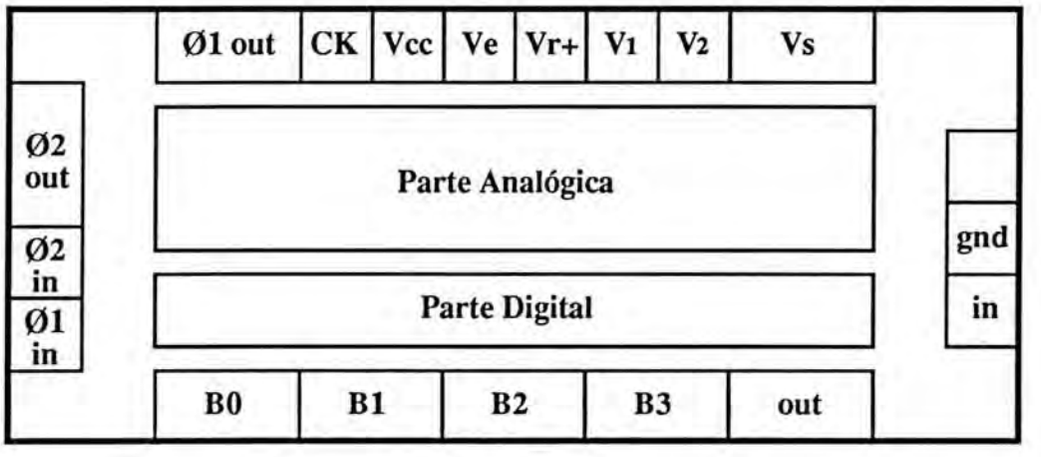


Figura 6.6 Planta Baixa do Circuito CONVERSO

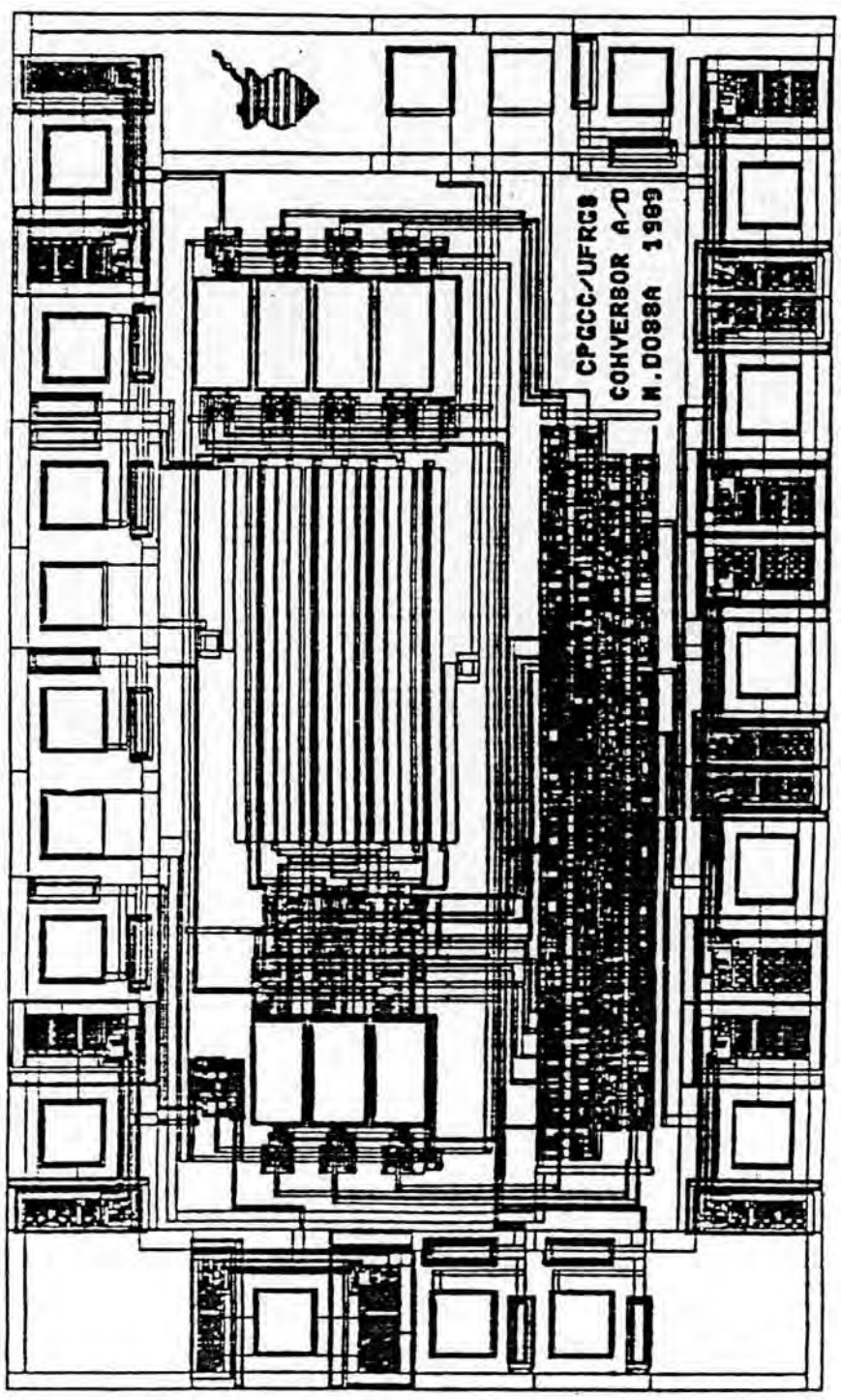


Figura 6.7 "Layout" final do circuito CONVERSO.

O gerador de fases e o comparador de teste encontram-se dentro da área da parte analógica. Linhas independentes ligam as partes analógica e digital aos "pads" de alimentação, solução bastante providencial,

visto que para conversores de média à grande resolução, a parte digital gera considerável ruído, passando a interferir na parte analógica que apresenta sensibilidade muito alta.

6.2.2 Modelagem do Circuito

Com o objetivo de avaliar o desempenho de PESTICIDE, o circuito protótipo foi modelado, dentro das bases de conhecimento do sistema especialista, em vários níveis hierárquicos, e dentro deles, o grau de modularidade também sofreu grandes alterações.

Considerando-se esta questão, duas observações devem ser feitas.

A primeira é relacionada com a entrada primária do conversor A/D. A fim de modelar um bloco analógico usando PESTICIDE, optou-se pela geração de uma tabela contendo 16 níveis de entrada, cada um definindo um pequeno intervalo de tensão ao longo de uma escala de valores contínuos de 0 à 5V. Desta forma, a entrada primária analógica foi modelada em 16 níveis discretos de sinal que foram representados por 4 entradas primárias.

A segunda observação faz referência à modelagem do circuito. Graças ao conhecimento que o usuário adquiriu sobre o circuito em fase de validação, este pôde ser modelado de duas diferentes maneiras, dependendo do particionamento escolhido. A primeira forma de particionamento considerou o circuito como sendo um sistema sequencial de blocos enquanto que a segunda considerou-o como um sistema combinacional de blocos.

As figuras 6.8.a e 6.8.b mostram, respectivamente, o particionamento sequencial e combinacional do circuito protótipo dentro da base de conhecimento. Em ambos os casos, a hipótese de falha múltipla foi assumida e o modelo que representa o conversor foi descrito em dois níveis hierárquicos. De acordo com as figuras 6.8.a e 6.8.b, o particionamento sequencial e combinacional do primeiro nível hierárquico considerado para o circuito modelado é mostrado, em detalhes, nas figuras 6.9 e 6.10, respectivamente.

Os apêndices I e II contêm as bases de conhecimento estrutural, funcional e comportamental geradas para ambos os casos (figuras 6.8.a e 6.8.b). O primeiro apêndice mostra a base de conhecimento gerada considerando-se uma *falha em um bloco,* de acordo com a modelagem do circuito mostrada na figura 6.8.a e o segundo apêndice contém as bases de conhecimento geradas para uma *falha em uma trilha,* de acordo com a modelagem do circuito descrita na figura 6.8.b.

Devido a um teste funcional prévio aplicado ao circuito e graças às entradas de fase externas, providas pelo protótipo, verificou-se que o bloco gerador de fases não possuía falhas. Desta forma, durante a tarefa de modelagem do circuito, os nodos internos deste bloco não foram considerados.

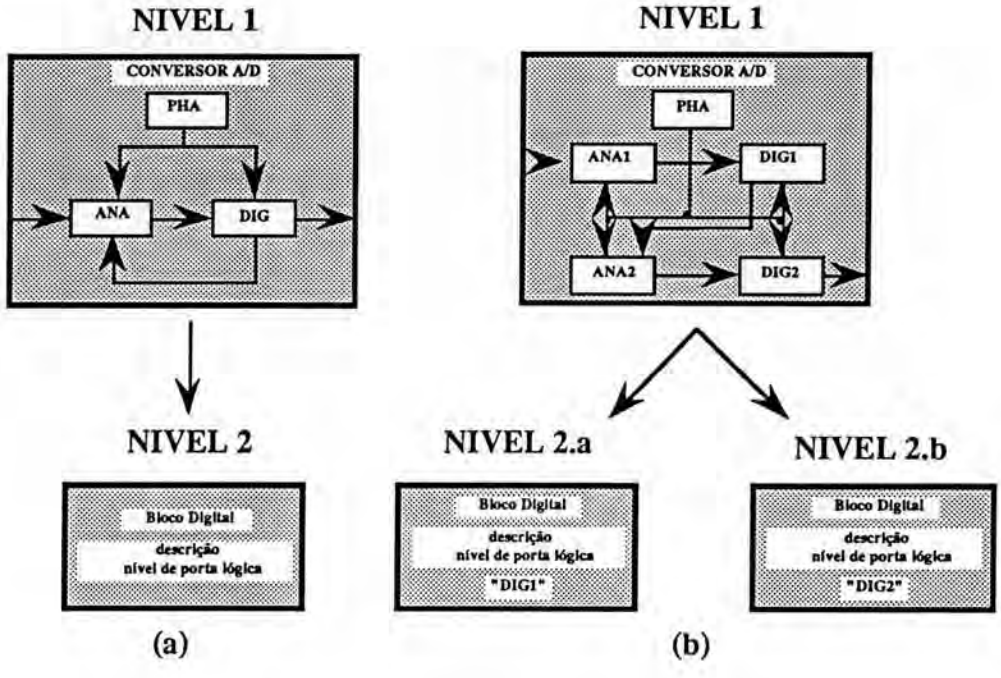


Figura 6.8 Níveis de descrição dentro das bases de conhecimento do sistema PESTICIDE. (a) para a estratégia de localização de falha "seqüencial múltipla", (b) para a estratégia de localização de falha "combinacional múltipla".

De acordo com o primeiro modelo do circuito, mostrado na figura 6.8.a, o primeiro nível das bases de conhecimento descreve o circuito do ponto de vista "conversor" e seus "três blocos componentes": analógico, digital e gerador de fases, fornecendo, ao mesmo tempo, as interconexões entre

eles. O segundo nível das bases de conhecimento provê a descrição interna do bloco digital. Considerando estes dois níveis de modelagem, o primeiro é bem enquadrado no caso da estratégia de localização de falha "seqüencial múltipla" e o segundo no caso da estratégia de localização de falha "combinacional múltipla" (considerando-se o bloco digital como sendo um circuito combinacional).

Agora, assumindo-se as mesmas considerações para a modelagem do circuito apresentada na figura 6.8.b, conclui-se que ambos, o primeiro e o segundo nível de descrição, são bem aplicáveis ao caso da estratégia de localização de falha "combinacional múltipla".

6.2.3 Resultados Obtidos

O experimento prático foi realizado através da definição de dois modelos de falha para o circuito. O primeiro modelo considera uma falha em uma conexão e o segundo, uma falha em um bloco.

Com relação à representação seqüencial ou combinacional do circuito, as figuras 6.9 e 6.10, respectivamente, mostram estes dois modelos de falha com os respectivos caminhos de propagação para cada um dos casos simulados.

Durante o processo de validação de protótipo, PESTICIDE executa dois procedimentos independentes. O primeiro procedimento procura por falhas em trilhas e conexões, e o segundo em blocos. Esta seqüência de teste é assim executada porque uma trilha defeituosa pode mascarar uma falha em um bloco. Em outras palavras, considera-se uma falha em uma trilha que está conectada à saída de um bloco. Esta situação pode mascarar a saída do bloco através do fornecimento, a PESTICIDE, de um valor errôneo em sua saída. Como resultado, o sistema especialista declara este bloco como sendo um "bloco contendo falha" e todo o caminho conectado a suas saídas como um "caminho de propagação do erro".

Terminado o processo de simulação de um caso real, PESTICIDE forneceu o nome da trilha contendo uma falha e em seguida, o nome do bloco defeituoso, bem como o caminho de propagação do erro, o qual era conectado à saída do bloco.

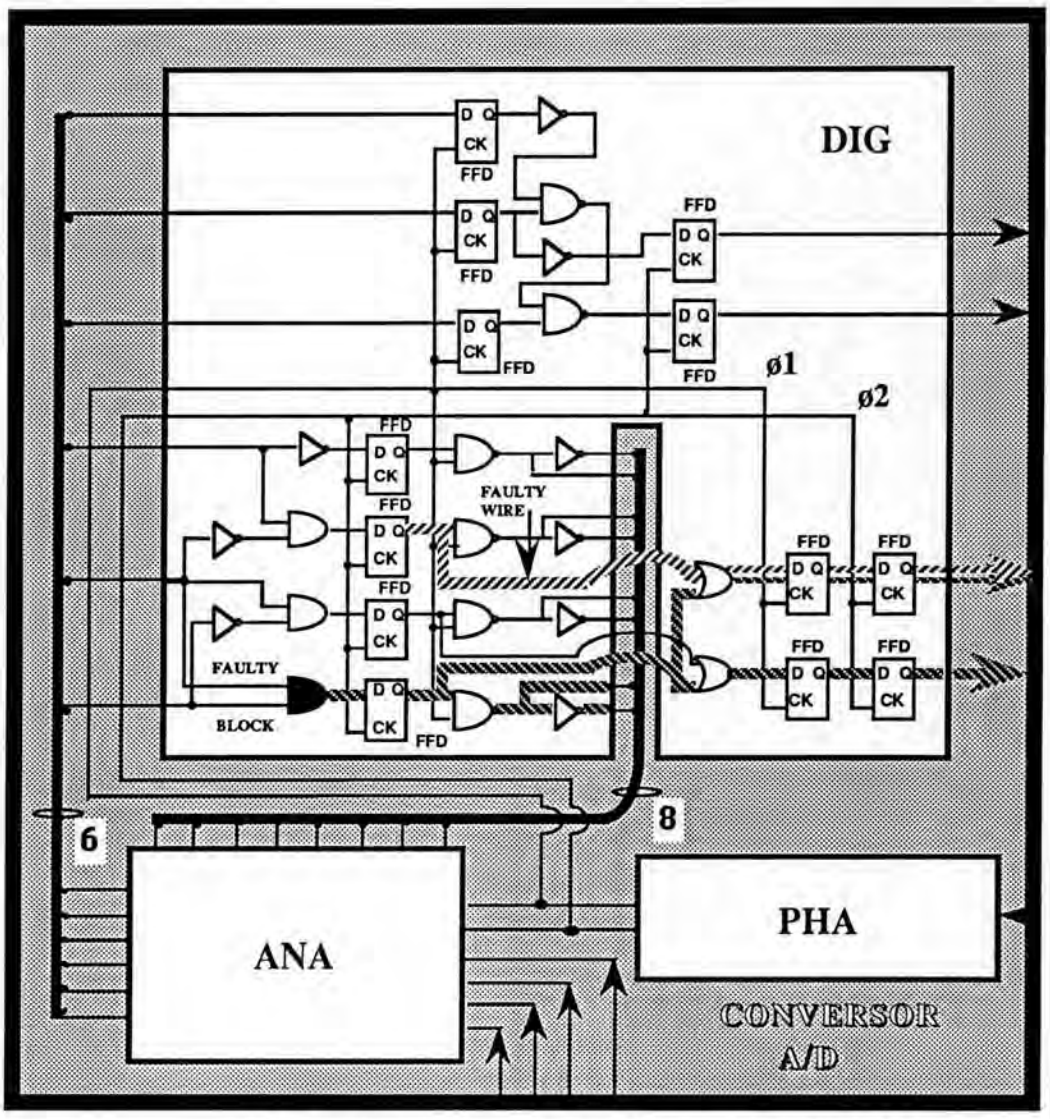


Figura 6.9 Particionamento seqüencial do circuito protótipo

Esta situação descrita é bastante interessante no que diz respeito ao processo de validação de protótipos baseado no teste com feixe de elétrons, porque a superfície que deve ser analisada através do TFE, procurado-se por falhas no seu interior, pode ser fortemente reduzida, desde que se saiba onde se encontra a área contendo o defeito e seu respectivo caminho de propagação.

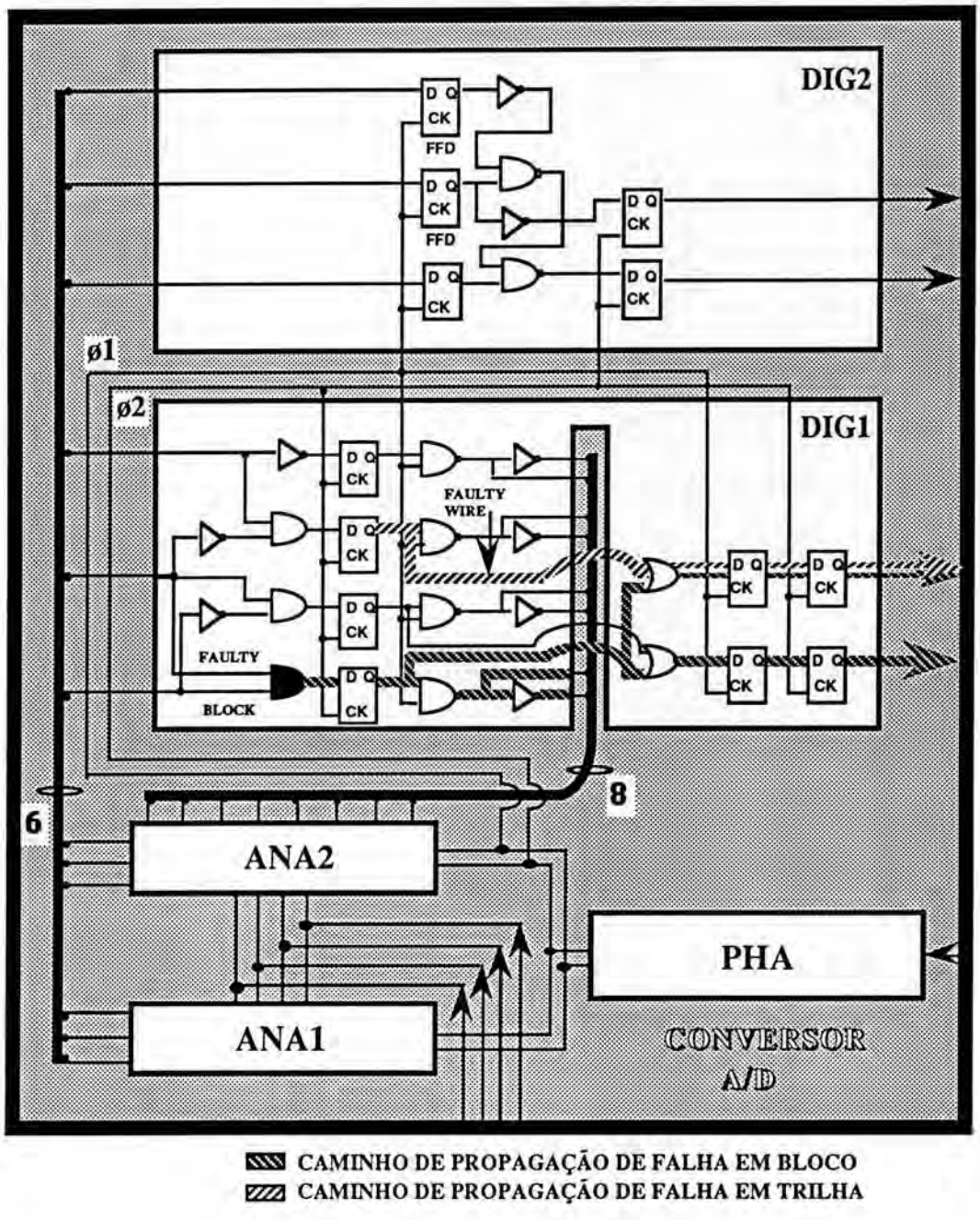


Figura 6.10 Particionamento combinacional do circuito protótipo

6.2.4 Desempenhos Obtidos*

A tabela 6.1 mostra alguns dos resultados obtidos para a análise do desempenho de PESTICIDE com o circuito descrito na figura 6.9.

* Os trabalhos práticos foram realizados em um computador BULL DPX 5000

Esta tabela descreve o comportamento das estratégias de localização de falhas para vários números de interfaces-bloco erradas a fim de se comparar as estratégias seqüencial e combinacional. Esta tabela define o número total de blocos (33) e interfaces-bloco (104) do circuito, o número total de blocos e interfaces-bloco errôneos em cada um dos casos simulados, e o consumo de tempo da CPU para cada uma das estratégias definidas, a fim de obter-se o diagnóstico de falha. Assim, depois de se fazer estas considerações, os próximos parágrafos fornecem algumas conclusões a respeito do desempenho das estratégias.

Considera-se a primeira linha da tabela 6.1: nenhum bloco defeituoso, o consumo de tempo da CPU é quase o mesmo entre as estratégias de localização de falha combinacional simples e múltipla (tempo de CPU $\approx$ 0,07 seg.) e as estratégias de localização de falha sequencial simples e múltipla (tempo de CPU $\approx$ 1,7 seg.), pois o processo de procura dos blocos defeituosos é o mesmo em ambas estratégias (simples e múltipla);

Agora considera-se somente as colunas SCF e MCF da tabela 6.1: as linhas de dois a seis e de sete a onze representam dois caminhos diferentes de propagação da falha. O primeiro grupo contém o incremento dos blocos defeituosos através de um caminho serial e o segundo, através de um caminho paralelo. Esta última forma de propagação do erro consome maior tempo da CPU na estratégia SCF do que na estratégia MCF. Por outro lado, o primeiro grupo apresenta um comportamento inverso. Isto é devido ao fato de que, para a estratégia MCF, PESTICIDE propaga um erro descendo-se diretamente através da árvore de blocos, cuja raiz é o bloco defeituoso, e declarando cada bloco ao longo deste caminho como sendo um "provável candidato a conter falha"* pois este se encontra ao longo do caminho de propagação da falha; enquanto que na estratégia SCF, PESTICIDE inicia o processo a partir de cada uma das interfaces-bloco erradas e sobe através da árvore de blocos, até se alcançar aquele que contém a falha. Neste último caso, o sistema especialista indica somente um bloco como sendo defeituoso, nenhum outro bloco pode ser declarado como sendo um "provável candidato a conter falha" pois trata-se de uma estratégia de localização de falha simples.

---

* ao longo do processo de depuração, PESTICIDE declara um bloco como *provável candidato a conter falha* através da expressão "block assumed to be faulty"

Além disso, os dois casos (falha simples e múltipla) são definidos de tal forma que a hipótese simples provê menos informação do que a múltipla: neste último caso, quando um bloco apresenta uma saída errônea, é suficiente declará-lo como sendo "provável candidato a conter a falha" (mesmo que suas saídas não estejam erradas), enquanto que no caso de falha simples, este bloco deve apresentar uma entrada errada e, ao menos, uma saída errônea para pertencer ao caminho de propagação da falha. Assim, o número de caminhos a serem seguidos depende diretamente, de como os blocos estão conectados entre si (isto é, modo serial ou paralelo).

Estes diferentes tipos de comportamento, de acordo com a estratégia escolhida, são mostrados nas tabelas 6.2 e 6.3, respectivamente para as interconexões de blocos no modo serial e paralelo. A informação importante contida nestas tabelas é o número de blocos defeituosos e de blocos declarados como "prováveis candidatos a conter falhas", identificados quando da aplicação de cada uma das estratégias de localização de falha combinacional simples e múltipla. Estes diferentes resultados são explicados pelo número de caminhos a serem examinados por cada uma das estratégias, dependendo de como os blocos estão interconectados. A figura 6.11 mostra os diferentes comportamentos e fornece o número total de blocos (defeituosos e/ou declarados como "prováveis candidatos a conter falha" ou seja, blocos ao longo do caminho de propagação) e o número total de caminhos examinados. Estes números totais são detalhados da seguinte maneira:

PARALELO/SCF
(falha simples)

B1-entr. primária (bloco c/falha, 1 caminho)
B2-B1 (propagação, 1 caminho)
B3-B1 (propagação, 1 caminho)
B4-B1 (propagação, 1 caminho)

TOTAL: 4 caminhos percorridos.

PARALELO/MCF
(falha múltipla)

B1-entr. primária (bloco c/falha, 1 caminho)
B1-B2-B5 ("block assumed to be faulty", 2 caminhos)
B1-B3-B6 ("block assumed to be faulty", 2 caminhos)
B1-B4-B7 ("block assumed to be faulty", 2 caminhos)

TOTAL: 7 caminhos percorridos.

SERIAL/SCF
(falha simples)

B1-entr. primária (bloco c/falha, 1caminho)
B4-B3-B2-B1 (propagação, 3 caminhos)
B3-B2-B1 (propagação, 2 caminhos)
B2-B1 (propagação, 1 caminho)

TOTAL: 7 caminhos percorridos.

SERIAL/MCF
(falha múltipla)

B1-entr. primária (bloco c/falha, 1 caminho)
B1-B2-B3-B4-B5("block assumed to be faulty", 4 caminhos)

TOTAL: 5 caminhos percorridos.

Tabela 6.1 Consumo de tempo de CPU para as 4 estratégias de localização de falha.

# Bl.	# Int. Bl.	# Bl. Err.		# Int. Bl. Err	TEMPO CPU (seg) Estratégia			
		S	M		SCF	MCF	SSF	MSF
33	104	0	0	0	0.067	0.067	1.717	1.683
33	104	1	2	2	0.167	0.167	1.833	1.817
33	104	2	5	6	0.233	0.300	2.000	2.033
33	104	3	6	8	0.383	0.500	2.233	2.067
33	104	4	6	11	0.517	0.650	2.717	2.133
33	104	5	7	13	0.633	0.700	3.033	2.233
33	104	6	8	15	0.783	0.750	3.533	2.367
33	104	7	8	17	0.983	0.917	4.117	2.417
33	104	8	9	19	1.100	0.983	4.417	2.517
33	104	9	10	21	1.250	1.033	4.917	2.567
33	104	10	10	23	1.450	1.217	5.533	2.650

Tabela 6.2. Resultados da propagação em série.

# Bl. Err.		Tempo CPU (seg)	
Simp.	Mult.	SCF	MCF
1	1	0.133	0.083
1	2	0.133	0.083
1	3	0.117	0.117
1	4	0.117	0.133
1	5	0.117	0.150

Tabela 6.3. Resultados da propagação em paralelo.

# Bl. Err.		Tempo CPU (seg)	
Simp	Mult.	SCF	MCF
1	1	0.133	0.100
1	2	0.117	0.100
2	3	0.167	0.117
3	4	0.200	0.133
4	5	0.283	0.150

Esta análise é relativa à estratégia combinacional. Porém, as mesmas considerações podem ser feitas quanto à estratégia seqüencial. Neste caso, as seguintes observações devem ser feitas:

• As interfaces-bloco propriamente ditas são substituídas pelos seus "instantes de interfaces-bloco". Isto significa que o valor de uma interface bloco é tomado em diferentes instantes de medição, graças ao modo de tensão estroboscópica do TFE. A fotografia 6.1 mostra dois diferentes instantes das interfaces-bloco ø1 e ø2 (ver figura 6.9). Na fotografia 6.1.a, a fase ø1 é nível lógico alto (em cinza escuro na imagem), enquanto ø2 é baixo (em nível de cinza claro). Valores opostos são vistos na fotografia 6.1.b.

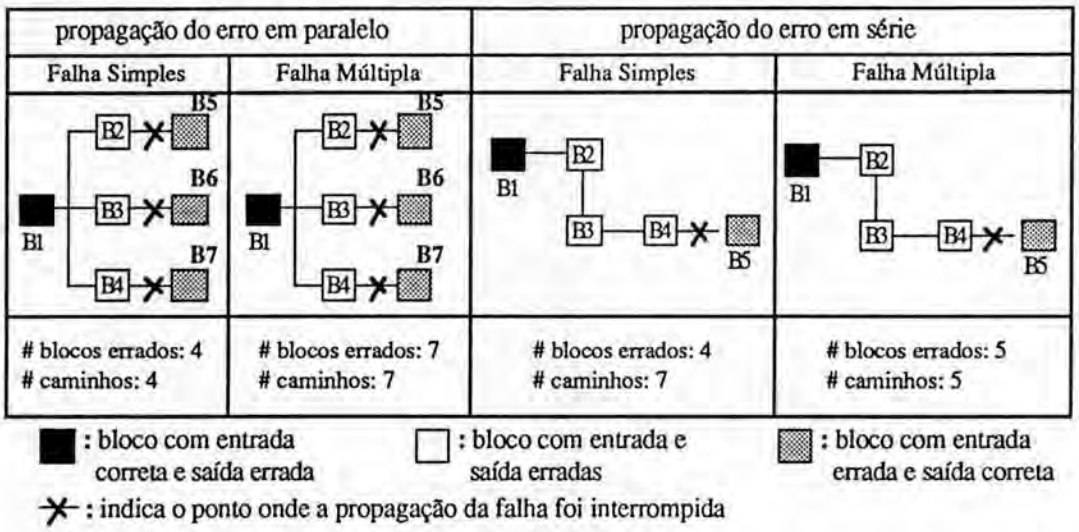


Figura 6.11 Informação provida pelas estratégias simples e múltiplas

• Considerando-se que a estratégia seqüencial deve computar a avaliação da fórmula de validação do instante de medição, ela consome maior tempo de CPU do que a estratégia combinacional.

• Considere a tabela 6.1: a mesma diferença de comportamento, como evidenciado na estratégia combinacional de falha simples e múltipla (SCF e MCF), ocorre na estratégia seqüencial, mas este fenômeno não é observado devido à computação da fórmula de verificação da validade do instante de medição. Neste cálculo, o consumo de tempo de CPU é muito maior no caso de falha simples (SSF) do que no de falha múltipla (MSF), mesmo que o número de caminhos a serem examinados seja menor. Embora mascarada, a diferença entre os casos seqüencial simples e múltiplo podem ser detetados através da divisão dos tempos SSF e MSF: os resultados são maiores para o segundo grupo de linhas (7 à 11) do que para o primeiro (2 à 6).

A fim de se observar de que maneira varia o tempo de CPU com relação à variação do número de instantes das interfaces-bloco, mais alguns experimentos foram realizados com PESTICIDE. A tabela 6.4 mostra os resultados destes experimentos: a primeira coluna apresenta o número total de instantes de medição, enquanto que a segunda mostra somente o seu fator multiplicativo. Pode-se notar na primeira linha da tabela, que constitui a referência, a existência do mesmo número de instantes de medição e de interfaces-bloco; neste caso, obteve-se o mesmo tempo de CPU que aquele

apresentado na última linha da tabela 6.1, pois este tempo de CPU foi obtido exatamente nas mesmas condições experimentais (propagação através de caminhos paralelos, com 1 instante de medição por interface-bloco). A última linha da tabela 6.4 mostra as relações dos tempos de CPU para as estratégias SSF e MSF e mostra que o consumo de tempo na estratégia seqüencial simples aumenta mais rapidamente do que na seqüencial múltipla, quando o número de instantes de medição aumenta.

Tabela 6.4 Consumo de tempo de CPU para diferentes números de instantes de medição.

Total de Instantes	Instantes /Int. Bl.	Tempo CPU (seg)		
		SSF	MSF	SSF/MSF
104	1	5.550	2.667	2.08
208	2	12.333	4.583	2.691
312	3	23.100	7.350	3.143
416	4	37.917	10.950	3.463
520	5	56.117	15.383	3.648
1040	10	207.800	50.850	4.087
1560	15	460.600	107.500	4.285

6.2.5 Conclusão da Avaliação do Desempenho do Sistema Especialista

Um trabalho prático foi realizado com o sistema especialista baseado no conhecimento, PESTICIDE. O desempenho desta ferramenta foi obtido através da simulação de falhas no circuito e da variação das formas de representação do dispositivo dentro de suas bases de conhecimento. A partir da análise dos resultados, os próximos parágrafos descrevem as conclusões obtidas.

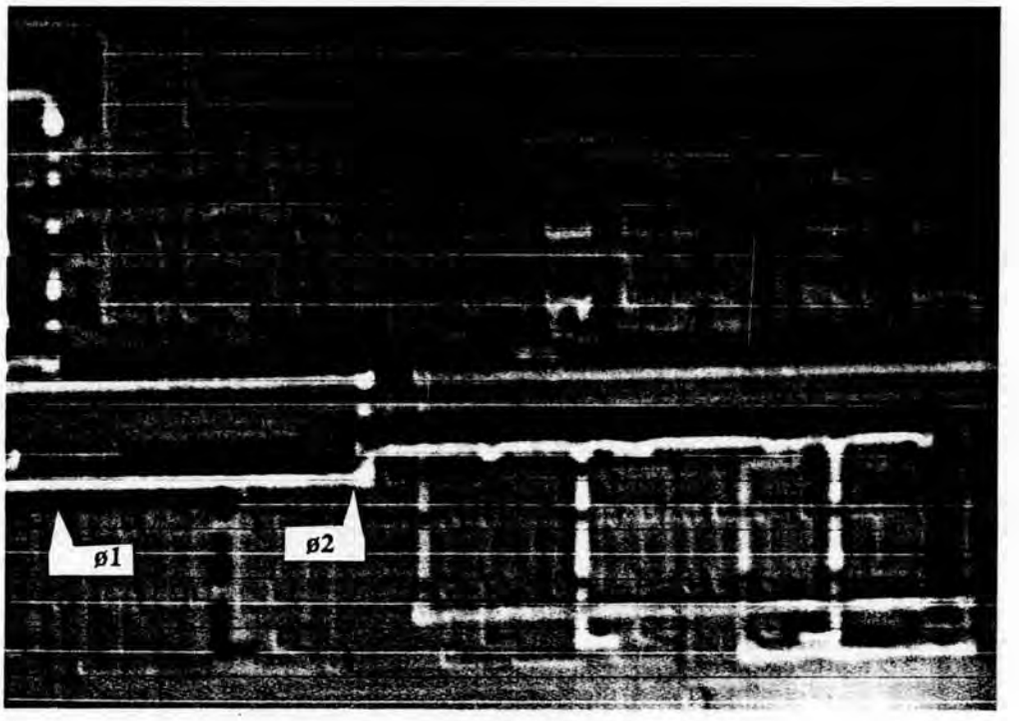
O sistema baseado no conhecimento, aqui utilizado, constitui um progresso real no estado da arte em validação de protótipos de circuitos VLSI usando-se o teste com feixe de elétrons. Ele representa a automação do processo de diagnóstico de falhas, enquanto que outras pesquisas e produtos ([CON87a], [CON87b], [GOR87], [KOM87], [KUJ86], [HEN87]) tem somente obtido a automação do "link" entre o TFE e as ferramentas de CAD, deixando o diagnóstico de falhas, propriamente dito, para o projetista do circuito.

As estratégias de falhas simples e múltipla estão fortemente ligadas à forma do caminho de propagação do erro. Assim, uma entre as duas estratégias pode ser escolhida de acordo com o circuito a ser testado. Esta situação é bastante interessante quando na aplicação em validação de protótipos a fim de obter-se um melhor desempenho no processo de depuração. Além disso, o usuário pode escolher entre as duas estratégias (simples e múltipla) dependendo de qual é a relação mais interessante: "consumo de tempo/confiabilidade da resposta", pois a estratégia de localização de falha simples provê menos informação sobre o caminho de propagação do erro do que a estratégia de localização de múltipla.

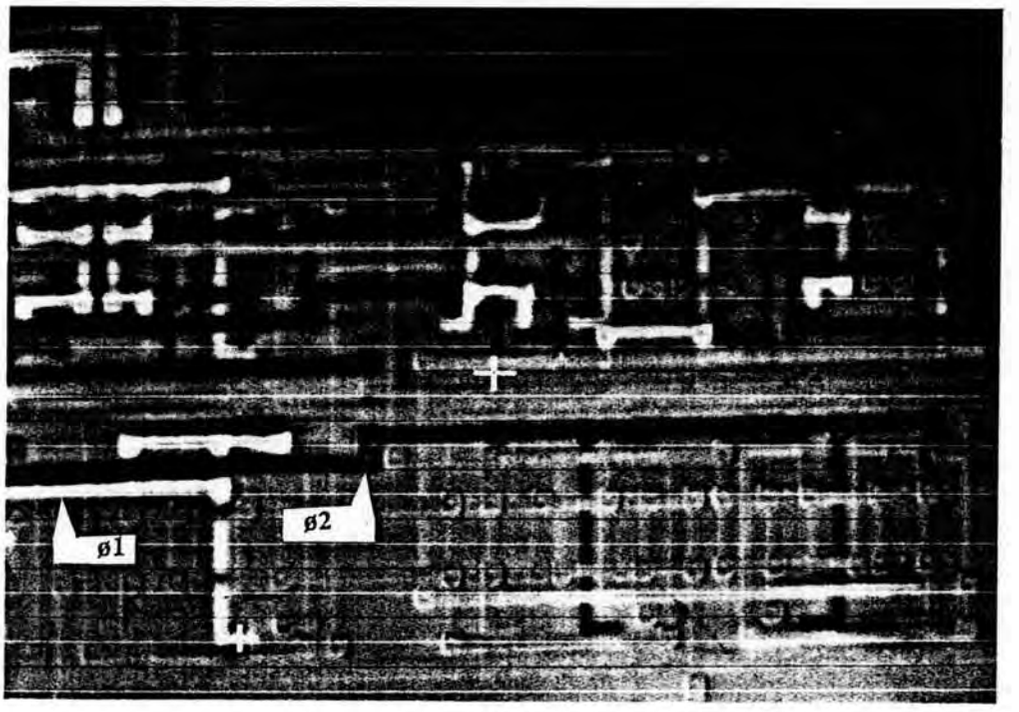
Além do mais, graças à filosofia de PESTICIDE, as tarefas de particionamento e de modelagem do circuito fornecem uma maior versatilidade para a escolha entre as estratégias combinacional e sequencial (ver a figura 6.8.b). Adotando-se a estratégia combinacional, neste caso, não somente consegue-se encurtar o tempo total de CPU, bem como reduzir o tamanho da área total do circuito que deve ser varrida à procura da(s) falha(s), pois cada um dos blocos analógico e digital é decomposto em dois outros blocos menores e menos complexos, a serem observados através do feixe de elétrons.

A fim de automatizar todo o processo de teste, o "link" entre o TFE disponível no laboratório TIM3 e PESTICIDE é mandatório. Esta condição é muito importante, caso dispositivos maiores e mais complexos venham a ser considerados.

A partir da análise dos resultados práticos obtidos, algumas modificações, visando o melhoramento desta ferramenta de validação de protótipos, são propostas mais adiante, no capitulo 7.



(a)



(b)

Fotografia 6.1. Visualização de trilhas de camadas inferiores devido ao efeito de acoplamento capacitivo durante dois instantes diferentes de interfaces-bloco: t_1: ϕ_1="1", ϕ_2="0" & t_2: ϕ_1="0", ϕ_2="1".

6.3 Usando o TFE para Depurar um Circuito Protótipo

Esta etapa teve como objetivo avaliar o desempenho de uma nova ferramenta para aplicação no processo de validação de protótipos e de gerar o diagnóstico de falha para o circuito CONVERSO [DOS89] . Trata-se de MAICOP, Multiple Adjacent Images Comparison Process, cuja função é a comparação de imagens de diferentes estados lógicos internos do circuito em fase de depuração. Para tanto, imagens foram comparadas, duas a duas; cada imagem, contendo as informações relativas aos nodos internos do circuito para diferentes vetores de entrada. No final do processo, é gerado um conjunto de imagens resultantes das comparações contendo os pontos de discrepâncias entre dois vetores de entrada distintos para o circuito em teste. Baseado na análise deste conjunto de imagens e no conhecimento estrutural e funcional do circuito protótipo, obteve-se informação suficiente para que o diagnóstico de falha para o circuito CONVERSO fosse proposto.

6.3.1 Comparação de Imagens Adjacentes Múltiplas para Validação de Protótipos

Esta nova técnica é baseada na ferramenta de comparação de imagens apresentada no ítem 3.4 do capítulo 3, cuja aplicação é feita no processo de análise de falhas [CON91]. Tal como a ferramenta discutida no ítem 3.4, MAICOP é baseada na comparação de imagens dos estados internos dos circuitos. Mas ao contrário desta, estas imagens são provenientes sempre do mesmo circuito, obtidas para diferentes vetores de entrada do dispositivo.

Algumas modificações são consideradas nesta nova técnica de comparação de imagens, pois existe somente um circuito e assim, as imagens podem ser facilmente sobrepostas. Estas modificações levam em conta o fato de que existe somente 1 circuito a ser observado, de forma que as duas imagens a serem comparadas possuem a mesma origem. Esta condição faz com que não seja necessária a realização de todas as etapas que calculam o deslocamento, tanto em translação quanto em rotação, das imagens a comparar, já que se trata da sobreposição exatamente da mesma zona do circuito.

A seguir, são definidas três etapas que apresentam as tarefas mantidas e as modificadas, de acordo com a técnica de análise de falhas vista no capítulo 3.

PASSO I: é necessário definir uma origem comum para o circuito, mas o ângulo de rotação ø é desprezado;

PASSO II: esta etapa é realizada através da movimentação da mesa e da aquisição das imagens do circuito. A área de observação é definida pelos seguintes parâmetros:

- as coordenadas da primeira imagem: $(x,y)_{start}$;
- a distância entre duas imagens contíguas: D;
- o número de imagens em ambas direções x e y: nbx e nby, a fim de cobrir-se toda a zona de observação;

mas, o cálculo destas posições não considera o fator de correção ø, já que o seu cálculo foi no passo I.

PASSO III: os processos de segmentação de imagem e de detecção de cantos não são executados. Somente um histograma de equalização é realizado a fim de assegurar a normalização do nível de cinza. Então, a imagem de referência é transladada e subtraída da imagem de teste, em todos os níveis de cinza. A imagem gerada contendo as diferenças apresentará áreas claras para as discrepâncias e áreas escuras para as não-discrepâncias. Esta etapa é detalhada na figura 6.12, e um exemplo é visto na fotografia 6.2.

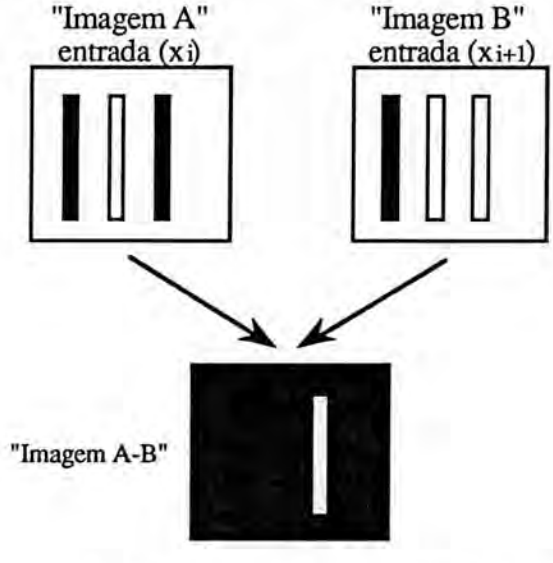


Figura 6.12 Processo de Comparação Automática de Imagem para a seqüência de entrada (x_i, x_{i+1}).

6.3.2 Equipamento de Teste

O sistema disponível para validação de protótipos no laboratório TIM3 (fotografia 6.3) consiste de um TFE CAMECA [BOU87] (figura 6.13), cujas principais características são:

- uma coluna eletrônica projetada para obter-se ótimo desempenho em baixa tensão (1KV);
- um sistema estroboscópico, permitindo trabalhar em modo dinâmico, incluindo: mapeamento do estado lógico, código de tensão, contraste de tensão estroboscópico, forma de onda, etc [MEN83], [WOL86];
- uma câmara de vácuo com dimensões suficientes para receber uma placa controladora de CI;
- uma mesa-suporte motorizada, com uma precisão de deslocamento de 1μm.

Além disso, a coluna do microscópio e o processo de aquisição são controlados a partir de uma estação de trabalho SUN 3/160, com 16M "bytes" de memória e 440M "bytes" de espaço total no disco rígido. Uma partição de 300M "bytes" é reservada ao armazenamento de um grande número de imagens produzidas. Para completar o ambiente de teste, um controlador de CI, denominado de TESSIE [BAI84], projetado e construído no próprio laboratório também está disponível.

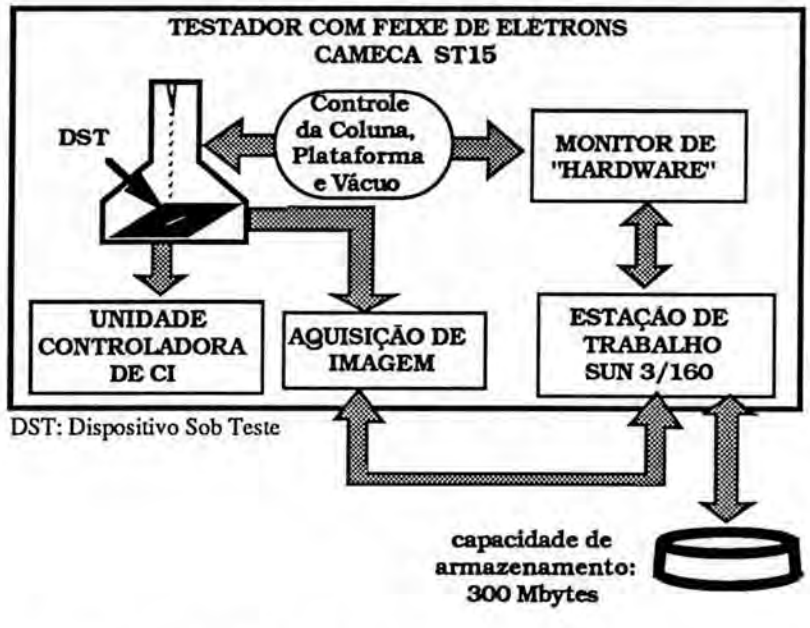
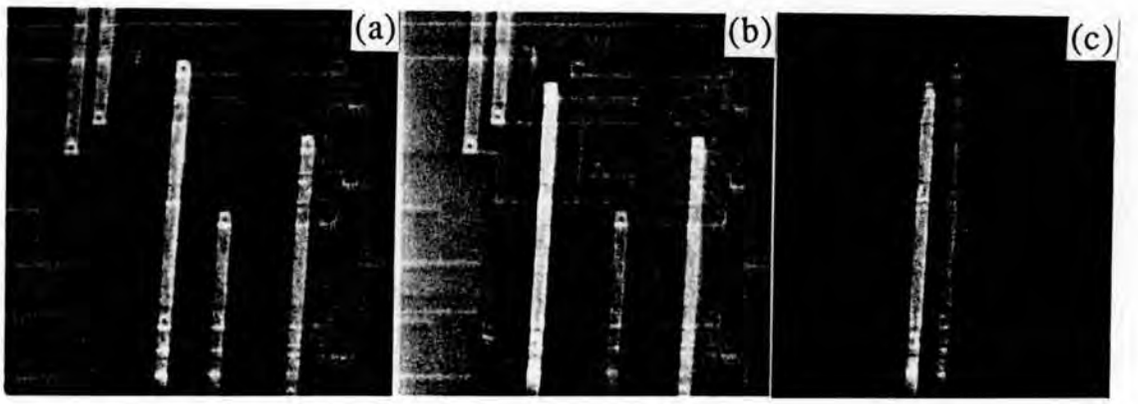
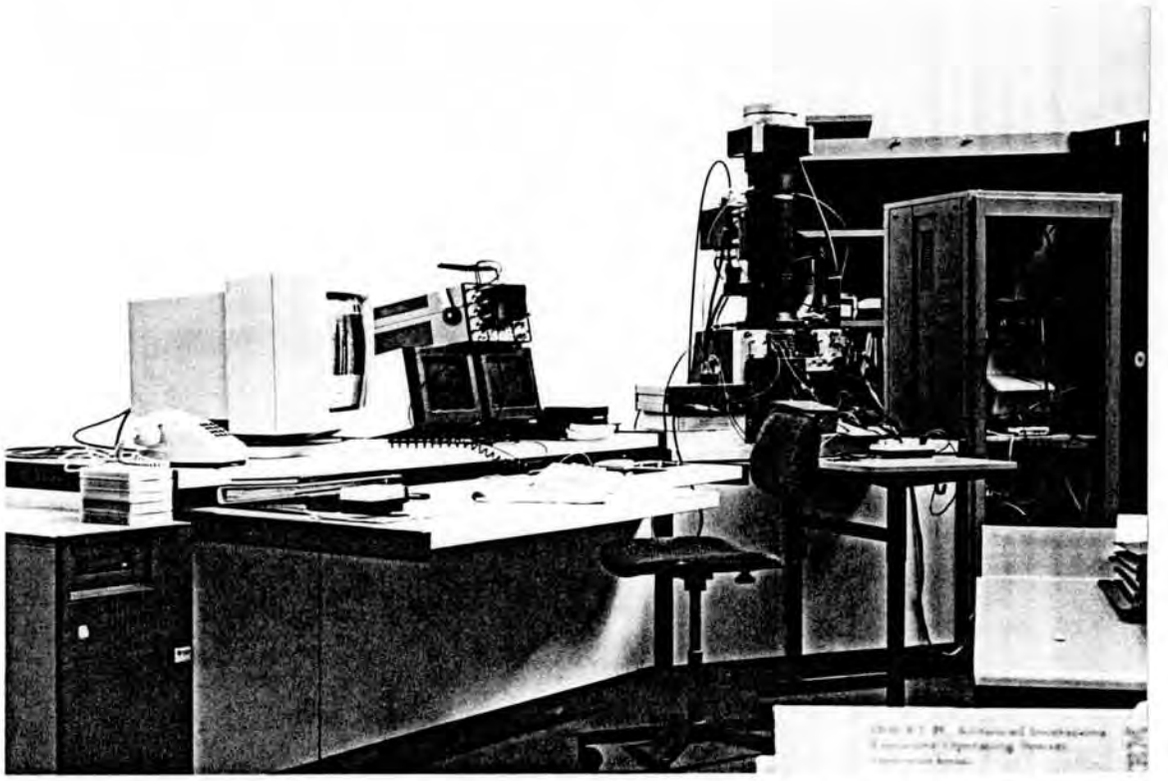


Figura 6.13 Configuração do TFE disponível no laboratório TIM3-INPG.



Fotografia 6.2. Imagens de trilhas conectando os blocos analógico e digital. (a) entrada primária Ei; (b) entrada primária Ei+1; (c) imagem da subtração (a) - (b), em contraste de tensão, mostrando as discrepâncias em áreas claras.



Fotografia 6.3. Testador de Feixe de Elétrons disponível no laboratório TIM3-INPG.

6.3.3 Modelo de Falha

Medidas convencionais sobre o protótipo do conversor mostraram uma falha "stuck-at-1" nos dois bits mais significativos da saída digital de quatro bits do conversor. Além disso, o comportamento dos dois bits menos significativos não era exatamente como aquele especificado em projeto

6.3.4 Localização da Falha

Considerando-se que a conversão é realizada em dois passos, por um processo seqüencial, a tarefa de localização da falha começou a partir da separação do elo de realimentação e pela aquisição das imagens em mapeamento do estado lógico das primeiras três saídas do bloco analógico. Devido ao fato da pequena complexidade oferecida pelas imagens, não houve a necessidade de compará-las. O objetivo desta tarefa foi verificar se os parâmetros do bloco analógico correspondiam àqueles especificados em projeto.

A primeira diferença foi observada na saída do primeiro estágio analógico, conforme pode ser observado pela fotografia 6.4.a. Nesta foto, observa-se que a mudança dos sinais sobre as trihas C3, C4 e C5, que são válidos somente quando o sinal de relógio está baixo, são respectivamente: C5 em nível lógico "0", C3 em nível lógico "1", e C4 está em transição (do nível lógico "0" para o "1"). Desta forma, concluiu-se que os sinais na saída do bloco analógico não estavam acompanhando de maneira correta as variações do sinal analógico na entrada do conversor.

Com o objetivo de confirmar este comportamento errôneo da parte analógica, efetuou-se medidas em forma de onda (fotografias 6.4.b e 6.4.c) no capacitor de diferenciação da saída analógica C4. Após uma série de medidas, confirmou-se o comportamento e a causa provável: um erro de dimensionamento do resistor divisor de tensão durante a fase de projeto.

Além disso, de posse dos dados adquiridos, foram calculados os valores exatos da entrada analógica do conversor para a alteração das saídas C3, C4 e C5. As tensões de transição observadas e os valores determinados em projeto para estas três trilhas são mostrados como se segue:

PROJETO (V)	OBSERVADO (V)	CÓDIGO DIGITAL PARA AS TRILHAS C3, C4, C5
0,0 - 1,1	0,0 - 1,9	000
1,2 - 2,3	2,0 - 2,2	001
2,4 - 3,6	2,3 - 2,8	011
3,7 - 5,0	2,9 - 5,0	111

O próximo passo era partir para a validação do bloco digital, seguindo o fluxo de dados seqüenciais para a conversão do sinal analógico. O problema era que, mesmo considerando a sua simplicidade, o comportamento exibido através das imagens era bastante complexo. A técnica de comparação de imagens foi então utilizada para tentar-se minimizar esta complexidade. Vale lembrar, que a técnica de comparação de múltiplas imagens adjacentes baseia-se na observação somente das mudanças dos estados lógicos internos do circuito, quando da aplicação de dois vetores diferentes, préviamente definidos, em suas entradas primárias. Neste instante é importante mencionar que as entradas do bloco digital, correspondentes às saídas do bloco analógico, já haviam sido todas determinadas. Esta situação era fundamental, pois o mapeamento das entradas do bloco digital era necessário para a realização da tarefa de localização de falha em seu interior.

Neste sentido, o processo de validação do protótipo continuou através da aquisição de um conjunto de imagens adjacentes do bloco digital, com entradas analógicas diferentes, demonstrando-se as diferenças vistas na fotografia 6.5(a-c). As imagens originais (6.5.a e 6.5.b) mostram os estados lógicos reais nestas trilhas para as entradas primárias do circuito colocadas em 1,5V e 2,0V, respectivamente. 6.5.c mostra a imagem resultante da subtração das duas imagens (6.5.a - 6.5.b) na qual são evidenciadas as diferenças. Estas imagens mostram as trilhas que transmitem o sinal de controle para as chaves dos comparadores finos, conectando a saída do bloco digital à entrada do bloco analógico, ou seja, o elo de realimentação, no caso de circuitos seqüenciais. A partir da análise dos pontos discrepantes destas imagens, constatou-se que os sinais que deveriam acionar as chaves uma de cada vez, deixando as outras desconectadas, realizavam a função inversa, colocando todas chaves em curto, e deixando apenas uma delas desligada. Posteriormente, identificou-se como sendo a causa desta falha, um erro na fase de roteamento dos "flip-flops" da parte digital, invertendo as ligações entre Q e Q negado.

Durante este processo de validação do protótipo, conjuntos de imagens foram adquiridos. Baseados em dados fornecidos pela simulação do circuito e analisando-se conjuntos resultantes de imagens comparadas, o procedimento para localização de falha resumiu-se na procura, ao longo do caminho de propagação da falha e de níveis de hierarquia inferiores, do lugar exato de sua localização. É importante ressaltar que este procedimento é manual; assim, o sucesso ou o fracasso do diagnóstico de falha depende do próprio usuário, de seu conhecimento sobre o circuito, e de sua experiência no processo de validação de protótipos.

No final do processo de depuração, os dois bits mais significativos "stuck-at-1" e o funcionamento dos dois bits menos significativos (LSB) que não se comportavam de acordo com os parâmetros definidos no projeto do conversor foram explicados. Sinais "stuck-at-1" foram encontrados nas saídas das duas portas lógicas "ou" da parte digital em função do erro de roteamento das saídas dos "flip-flops", e o problema de funcionamento dos LSB é relativo às más especificações dos parâmetros de projeto para o resistor do bloco analógico, além do fato das chaves, que controlam os comparadores de ajuste fino da parte analógica, estarem quase todas curto-circuitadas ao mesmo tempo, quando apenas uma delas deveria conduzir a cada sinal de controle.

6.3.5 Conclusão da Depuração do Circuito Protótipo pelo TFE

Foi apresentada uma nova técnica de validação de protótipos baseada no processo de comparação de múltiplas imagens adjacentes. Além de validar a própria ferramenta, através da obtenção de seu desempenho e da análise dos resultados, forneceu-se o diagnóstico de falha para o circuito CONVERSO.

Considerando que este processo é baseado na aquisição e no tratamento de imagem, é sempre necessário considerar fatores práticos, tais como circuitos passivados, e mesmo circuitos depassivados, os quais podem conter ainda camadas de óxido e condutores enterrados, o que dificulta consideravelmente a aquisição da imagem. Para se contornar estas limitações, outras técnicas devem ser empregadas, tais como medições em forma de onda, mapeamento do estado lógico (ver fotografia 6.4) [MEN81], projeto visando a

testabilidade no teste com feixe de elétrons [MEL88], [LEE89], etc. Este é o caso particular de circuitos analógicos para os quais as imagens em níveis de cinza não fornecem informação suficientemente acurada para se obter uma alta confiabilidade no processo de diagnóstico de falha.

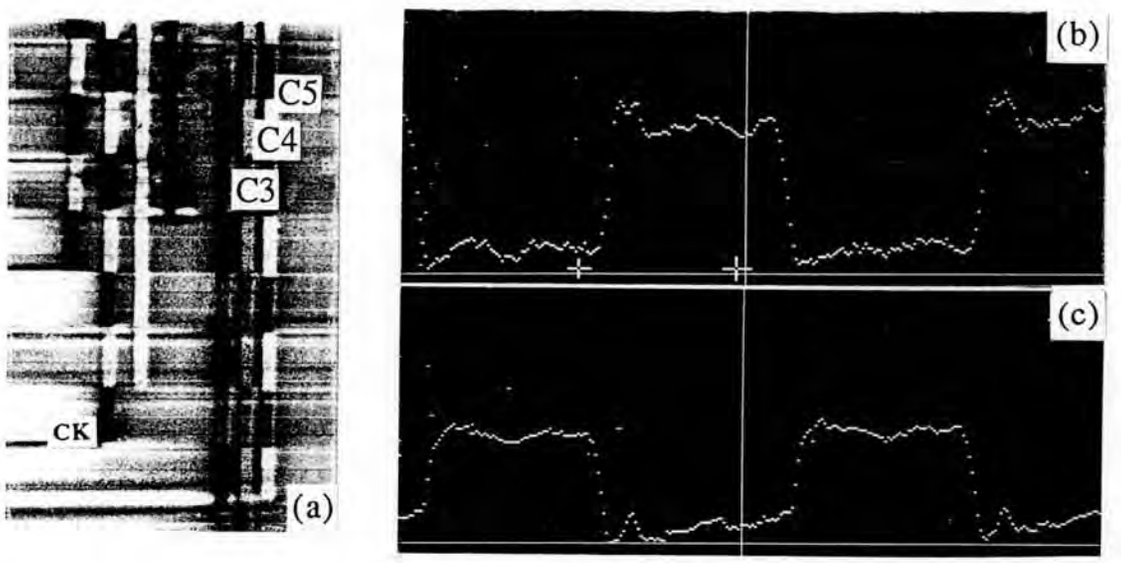
Além disso, verificou-se que o processo de comparação de múltiplas imagens adjacentes é bastante útil na validação de protótipos no sentido de reduzir-se a quantidade de dados a serem tratados. Em outras palavras, é mais fácil manusear uma quantidade reduzida de informação resultante da comparação de imagens, do que tratar cada caso separadamente, principalmente, quando é esperado que a imagem resultante forneça poucos pontos de discrepâncias, tornando a tarefa de análise dos resultados ainda mais simplificada.

Com relação ao circuito CONVERSO, é importante citar que o diagnóstico de falha fornecido é assim descrito:

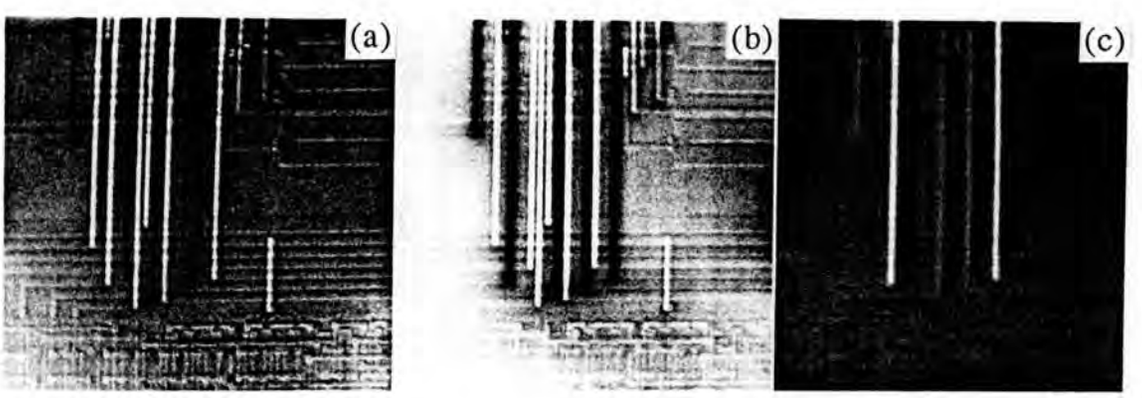
• Sinais "stuck-at-1" foram encontrados nas saídas das duas portas lógicas "ou" da parte digital em função do erro de roteamento das saídas Q e Q negado dos "flip-flops". Este problema de roteamento também afetou as chaves que controlam os comparadores de ajuste fino da parte analógica, pois quase todas as chaves estavam curto-circuitadas ao mesmo tempo, quando apenas uma delas deveria conduzir a cada sinal de controle;

• Más especificações dos parâmetros de projeto para o resistor de 12K5 do bloco analógico.

Para o futuro, a prioridade é a integração com ferramentas de concepção de alto nível, a fim de aumentar o grau de automação do processo de diagnóstico de falha [MAR91a], [MAR89a], [MAR89b]. Esta conexão pode ser realizada por um lado, com o TFE disponível no latoratório TIM3 e por outro, com o sistema especialista PESTICIDE.



Fotografia 6.4. (a) Saídas do bloco analógico C3, C4, C5 em Modo de Mapeamento do Estado Lógico; (b) e (c) instantes de précarga e avaliação do capacitor C4, nos quantizadores grosseiros.



Fotografia 6.5. (a) e (b) Imagens das trilhas que transmitem o sinal de controle para as chaves dos comparadores finos com a entrada analógica do circuito colocada em 1,5 e 2,0V, respectivamente; (c) imagem da subtração (a) - (b), em contraste de tensão, mostrando as discrepâncias em áreas claras.

7 MELHORAMENTOS NA TÉCNICA DE VALIDAÇÃO DE PROTÓTIPOS

Este capítulo apresenta alguns melhoramentos no processo de validação de protótipo baseado no conhecimento em desenvolvimento no laboratório TIM3. Estes melhoramentos objetivam tanto a completa automação do processo quanto o enriquecimento da informação provida no final do processo de diagnóstico de falha, de forma a obter-se um ambiente de teste para validação de protótipos apresentando um alto grau de integração e automação. Estes melhoramentos surgiram principalmente como resultado dos experimentos práticos discutidos no capítulo 6.

7.1 Acoplando o Testador com Feixe de Elétrons a um Sistema Baseado no Conhecimento

Devido ao aumento da complexidade dos circuitos, a necessidade da ligação do TFE a um sistema especialista é mandatória. Esta conexão fornecerá a informação necessária para PESTICIDE gerar, automaticamente, o diagnóstico de falha.

Este acoplamento pode ser realizado em três etapas:

Passo I:

A informação fornecida pelo TFE será adquirida e tratada a fim de associar-se o nome e o instante de medição de cada uma das interfaces-bloco observadas ao valor do estado lógico medido.

Passo II:

A informação será comparada com aquela fornecida pelo sistema de CAD. No final deste processo, é gerada uma lista contendo o estado das interfaces-bloco: "certo", "errado" ou "não observado".

Passo III:

A geração da base de conhecimento comportamental do dispositivo é realizada tomando-se em conta a hipótese formulada a respeito da seção de teste corrente: SCF, MCF, SSF ou MSF. No final deste processo, a base de conhecimento comportamental está pronta para ser explorada por PESTICIDE.

7.2 Informação Fornecida por PESTICIDE

Considerando a informação provida pelo sistema especialista, os seguintes ítens podem ser incluídos:

- Com relação ao parâmetro *estado* de uma interface-bloco a ser observada, o sistema especialista poderia fornecer:
  - uma lista de pontos (interfaces-bloco) a serem observadas pelo TFE;
  - a seqüência de entradas a ser gerada automaticamente pelo controlador de CI, a fim de sensibilizar o ponto desejado;

- No caso de informação insuficiente para efetuar-se o diagnóstico de falhas (por exemplo, a não observância de interfaces-bloco ou então de algum instante de uma interface-bloco) o sistema deve identificar os prováveis blocos do circuito que podem conter a falha. Assim, PESTICIDE poderia basear-se na descrição topológica do circuito a fim de prover uma listagem identificando os prováveis blocos a conter uma falha (estes blocos estão localizados ao longo do caminho de propagação da falha, desde as entradas primárias, até o ponto onde a medição foi realizada). Este diagnóstico parcial passa gradativamente através dos níveis hierárquicos inferiores e do aumento do grau de modularidade do circuito. Esta situação é ilustrada na figura 7.1, onde um exemplo é descrito. As únicas informações providas pelo TFE são os estados lógicos das entradas (corretas) dos blocos A e B e a saída do bloco C, onde um erro foi observado. Considerando esta figura, os blocos A e C são os candidatos mais prováveis para conter uma falha. O sistemas especialistas forneceriam este diagnóstico parcial baseando-se nas informações contidas:

I. na Base de Conhecimento Funcional, tais como o cone de cobertura de cada uma das interfaces-bloco de saída, onde verificou-se que a saída do bloco C, S3, tem como cone de cobertura as entradas E3 e E4.

II. na Base de Conhecimento Estrutural, identificando que as entradas E3 e E4 estão conectadas às saídas S1 e S2 do bloco A.

Agora, considerando um nível hierárquico inferior e fazendo as mesmas ressalvas, os blocos ao longo do caminho de propagação: **A1, A4, A6, A2, A5,** C1, C4 e C6 são os blocos com maior probabilidade de conter falha..

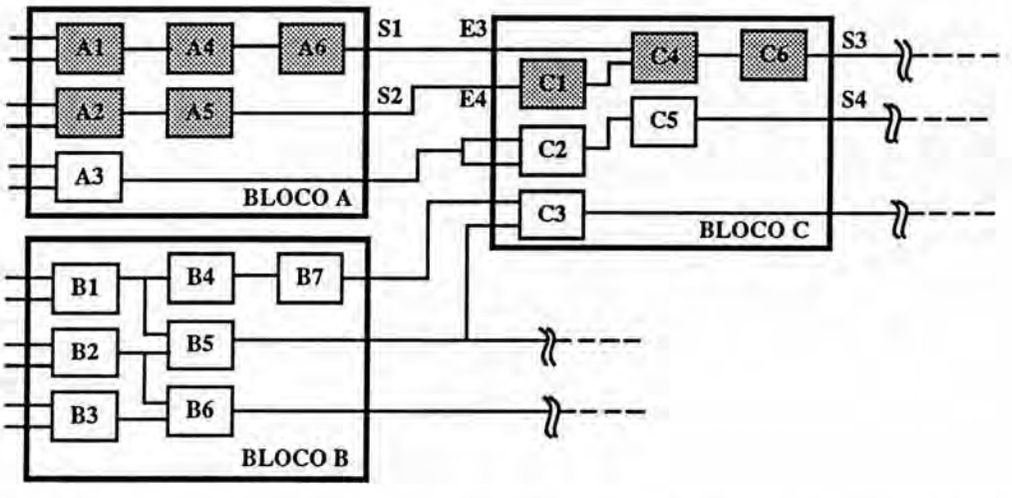


Figura 7.1 Diagnóstico de falha parcial, baseado na descrição topológica do circuito, devido à insuficiência das informações disponíveis.

7.3 Melhoramento do Modelo Baseado no Conhecimento para Representação de Dispositivos

PESTICIDE deverá ser capaz de manipular a informação provida pelos níveis de descrição mais baixos de um circuito protótipo, isto é, deve-se considerar a possibilidade de modelar-se um transistor.

Esta situação é bastante interessante quando aplicada aos casos cujo processo de diagnóstico de falha exija índices de precisão bem mais altos que o normal. Além disso, "transmission gates" e "sense amplifiers" ou estruturas similares podem ser facilmente modelados.

Uma possível representação de um transistor pode ser vista na figura 7.2. Este modelo considera que cada saída D_i do transistor assume o mesmo valor de sua entrada S_i quando um sinal de gatilho é aplicado ao transistor:

$$D_i = S_i \quad \text{quando} \quad G_{sinal} = \textbf{ativado}$$

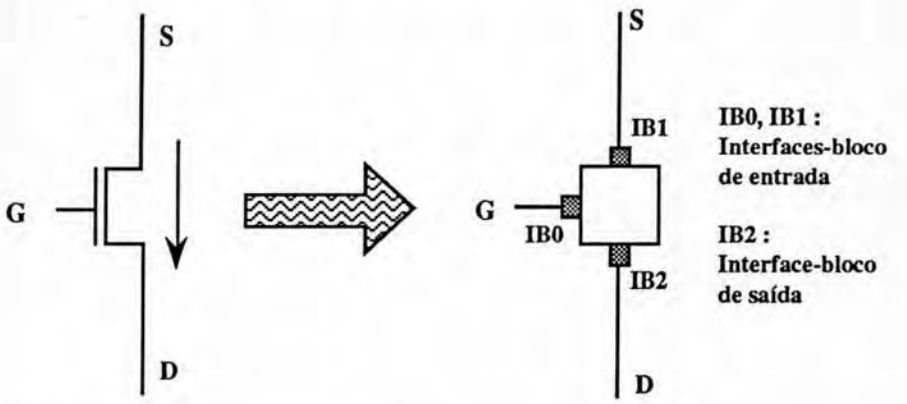


Figura 7.2 Modelando a Base de Conhecimento para Representação de um Transistor CMOS.

8 CONCLUSÃO DA PARTE II

Nesta segunda parte do trabalho, foram descritos experimentos práticos na área de validação de protótipos.

No capítulo 6, duas técnicas foram descritas, seus desempenhos analisados e os resultados discutidos. Além disso, o circuito CONVERSO foi apresentado e o seu diagnóstico de falha fornecido.

No capítulo 7, após terminados os experimentos práticos, foram propostos alguns melhoramentos na técnica de validação de protótipo baseada no conhecimento.

Quanto ao primeiro experimento prático, cujo objetivo foi avaliar o desempenho do sistema especialista, algumas conclusões podem ser obtidas:

- o desempenho de PESTICIDE mostrou-se bastante satisfatório, principalmente no que se refere à flexibilidade de modelagem do dispositivo dentro da base de dados do sistema especialista. Graças a esta característica da ferramenta e ao conhecimento que o usuário possui sobre o circuito testado, o problema da complexidade dos circuitos pode ser bastante minimizado, pois várias formas de particionamento modular e hierárquico estão à disposição do usuário.

- considerando-se o estágio atual de desenvolvimento de PESTICIDE, este não pode fornecer o diagnóstico de falha parcial, indicando aproximadamente a região onde a falha se encontra. O problema é que na prática, a ferramenta fica bastante comprometida, pois é praticamente impossível se fornecer o valor lógico de todas as interfaces-bloco à PESTICIDE. Na prática, o feixe de elétrons não tem acesso a grande maioria destes pontos devido ao problema de circuitos passivados, de trilhas enterradas, interferência de sinais em alta freqüência, etc. Assim, no caso da não observação de todas as interfaces-bloco em alguns dos níveis hierárquicos

do circuito, é proposto se melhorar a qualidade do diagnóstico de falha provido por PESTICIDE no sentido de que este forneça, baseado na descrição estrutural do circuito, resultados parciais, tais como os blocos prováveis para conter uma falha. Esta é uma medida de grande importância, caso se deseje tornar a ferramenta menos terórica e mais próxima de uma sessão de testes real.

Quanto ao segundo experimento prático, a valiação do desempenho da nova ferramenta de comparação de imagens foi realizada e o diagnóstico de falha para o circuito CONVERSO foi obtido. Neste contexto, algumas considerações podem ser feitas:

- Devido ao fato da ferramenta utilizar somente informações que são extraídas de imagens em contraste de potencial, circuitos passivados, tensões analógicas ou trilhas enterradas tornam-se barreiras quase intransponíveis para que a ferramenta obtenha as informações necessárias para a determinação do estado lógico dos diversos pontos do circuito. Desta forma, a utilização de outras técnicas auxiliares tais como medidas em forma de onda e o projeto visando a testabilidade com feixe de elétrons devem ser consideradas. É importante citar que na prática, os circuitos que não consideram o projeto visando a testabilidade com feixe de elétrons tem o seu processo de depuração seriamente afetado, pois o acesso aos pontos internos, tais como trilhas em camadas inferiores, torna-se praticamente impossível, restringindo-se ao uso de técnicas de observação por acoplamento capacitivo.

- Apesar do processo de depuração de múltiplas imagens adjacentes ter se mostrado bastante útil na validação de protótipos no sentido de reduzir-se a quantidade de informação a ser tratada, os problemas de:
  - correspondência entre os pontos sobre o desenho de máscaras e sobre o circuito físico;
  - diagnóstico de falhas;

são etapas manuais e lentas, consumindo muito tempo durante a fase de depuração do circuito. A fim de reduzir o tempo total do processo de validação de protótipo, reduzindo o número de intervenções feitas pelo usuário, estas etapas devem ser automatizadas. Para tanto, já existem no mercado ferramentas que fazem a correspondência automática entre o desenho de máscaras e o circuito físico, no interior da câmara de vácuo do MEV [TAM86]. Além disso, a conexão de MAICOP a outras ferramentas, cuja função é a localização automática de falhas, tal como PESTICIDE ou ADVICE, deve ser fortemente considerada.

CONCLUSÃO

Os temas "validação de protótipos" e "análise de falhas" foram apresentados, bem como um resumo sobre o estado da arte em âmbito internacional. O problema do projeto visando a testabilidade para o teste com feixe de elétrons também foi abordado, apresentando-se sugestões de implementação de circuito que facilitem a sua depuração.

Na segunda parte do trabalho, o estudo de um caso real foi descrito, assim como foi analisado o desempenho de dois processos de validação de protótipos: um sistema especialista, baseado no conhecimento, e um procedimento de comparação automática de imagens, originalmente utilizado somente na análise de falhas. Neste trabalho foram propostos melhoramentos na técnica de validação de protótipo, ainda em desenvolvimento no laboratótio TIM3-INPG. Estes melhoramentos visam, principalmente, aumentar o grau de automação do processo de validação de protótipos, tornando-o cada vez mais rápido e reduzindo cada vez mais o número de etapas nas quais o engenheiro de teste deve intervir no processo normal de depuração.

O objetivo final deste trabalho: busca de "know-how" e experiência na área de teste com feixe de elétrons, bem como o estudo de métodos de automação deste processo, foi plenamente alcançado. Um trabalho inicial, desenvolvido no CPGCC-UFRGS [VAR90], possibilitou o aprendizado de como testar um circuito utilizando um microscópio eletrônico e um estágio no laboratório TIM3-INPG permitiu avançar significativamente, com o desenvolvimento de uma capacitação no teste automatizado com microscópio eletrônico [VAR91]. O passo seguinte é a transferência deste conhecimento para outros pesquisadores do Grupo de Microeletrônica do PGCC, interessados nesta linha de pesquisa, bem como a aquisição, junto aos órgãos governamentais específicos, de equipamentos adequados ao trabalho de pesquisa nesta área.

ANEXO I

Arquivos de entrada do Sistema Especialista PESTICIDE: Bases de Conhecimento Estrutural, Funcional e Comportamental, geradas para a *falha em um bloco* de acordo com a modelagem do circuito CONVERSO (apresentada nas figuras 6.8.a e 6.9).

```
/---------------DESCRIPTION-LEVEL    I-------------------/
/---------ESTRUCTURAL---KNOWLEDGE-BASE-------------/

block(ext,[e0,e1,e2,e3,s0,s1,s2,s3,ck]).

block(ana,[ib00,ib01,ib02,ib03,ibf1,ibf2,ibp0,ibn0,ibp1,ibn1,inp2,ibn2,ibp
3,ibn3,ibc0,ibc1,ibc2,ibc3,ibc4,ibc5]).

block(dig,[ic0,ic1,ic2,ic3,ic4,ic5,ip0,in0,ip1,in1,ip2,in2,ip3,in3,if1,if2,ib0
4,ib05,ib06,ib07]).

block(pha,[ibck,igf1,igf2]).



blockint(e0,out,0,1).
blockint(e1,out,0,1).
blockint(e2,out,0,1).
blockint(e3,out,0,1).

blockint(s0,in,0,1).
blockint(s1,in,0,1).
blockint(s2,in,0,1).
blockint(s3,in,0,1).

blockint(ck,out,0,1).

blockint(ibck,in,0,1).

blockint(igf1,out,0,1).
blockint(igf2,out,0,1).

blockint(if1,in,0,1).
blockint(if2,in,0,1).

blockint(ibf1,in,0,1).
blockint(ibf2,in,0,1).

blockint(ib00,in,0,1).
blockint(ib01,in,0,1).
blockint(ib02,in,0,1).
blockint(ib03,in,0,1).

blockint(ib04,out,0,1).
blockint(ib05,out,0,1).
blockint(ib06,out,0,1).
blockint(ib07,out,0,1).

blockint(ibp0,in,0,1).
blockint(ibn0,in,0,1).
blockint(ibp1,in,0,1).
blockint(ibn1,in,0,1).
blockint(ibp2,in,0,1).
blockint(ibn2,in,0,1).
blockint(ibp3,in,0,1).
blockint(ibn3,in,0,1).
```

```
blockint(ip0,out,0,1).
blockint(in0,out,0,1).
blockint(ip1,out,0,1).
blockint(in1,out,0,1).
blockint(ip2,out,0,1).
blockint(in2,out,0,1).
blockint(ip3,out,0,1).
blockint(in3,out,0,1).

blockint(ibc0,out,0,1).
blockint(ibc1,out,0,1).
blockint(ibc2,out,0,1).
blockint(ibc3,out,0,1).
blockint(ibc4,out,0,1).
blockint(ibc5,out,0,1).

blockint(ic0,in,0,1).
blockint(ic1,in,0,1).
blockint(ic2,in,0,1).
blockint(ic3,in,0,1).
blockint(ic4,in,0,1).
blockint(ic5,in,0,1).



wire(w42,[e0,ib00]).
wire(w43,[e1,ib01]).
wire(w44,[e2,ib02]).
wire(w45,[e3,ib03]).
wire(w46,[ibp0,ip0]).
wire(w47,[ibn0,in0]).
wire(w48,[ibp1,ip1]).
wire(w49,[ibn1,in1]).
wire(w50,[ibp2,ip2]).
wire(w51,[ibn2,in2]).
wire(w52,[ibp3,ip3]).
wire(w53,[ibn3,in3]).
wire(w54,[ibc0,ic0]).
wire(w55,[ibc1,ic1]).
wire(w56,[ibc2,ic2]).
wire(w57,[ibc3,ic3]).
wire(w58,[ibc4,ic4]).
wire(w59,[ibc5,ic5]).
wire(w59,[igf1,ibf1,if1]).
wire(w60,[igf2,ibf2,if2]).
wire(w61,[s0,ib04]).
wire(w62,[s1,ib05]).
wire(w63,[s2,ib06]).
wire(w64,[s3,ib07]).
wire(w65,[ibck,ck]).

/------END-OF----ESTRUCTURAL---KNOWLEDGE-BASE-------/
```

```
/------------FUNCTIONAL---KNOWLEDGE-BASE-------------/

coverage(igf1,[ibck]).
coverage(igf2,[ibck]).

coverage(ibc0,[ibp0,ibn0,ibp1,ibn1,ibp2,ibn2,ibp3,ibn3,ibf2]).
coverage(ibc1,[ibp0,ibn0,ibp1,ibn1,ibp2,ibn2,ibp3,ibn3,ibf2]).
coverage(ibc2,[ibp0,ibn0,ibp1,ibn1,ibp2,ibn2,ibp3,ibn3,ibf2]).
coverage(ibc3,[ib00,ib01,ib02,ib03,ibf1,ibf2]).
coverage(ibc4,[ib00,ib01,ib02,ib03,ibf1,ibf2]).
coverage(ibc5,[ib00,ib01,ib02,ib03,ibf1,ibf2]).

coverage(ip0,[ic3,ic4,ic5,if1]).
coverage(in0,[ic3,ic4,ic5,if1]).
coverage(ip1,[ic3,ic4,ic5,if1]).
coverage(in1,[ic3,ic4,ic5,if1]).
coverage(ip2,[ic3,ic4,ic5,if1]).
coverage(in2,[ic3,ic4,ic5,if1]).
coverage(ip3,[ic3,ic4,ic5,if1]).
coverage(in3,[ic3,ic4,ic5,if1]).

coverage(ib04,[ic0,ic1,ic2,if1,if2]).
coverage(ib05,[ic0,ic1,ic2,if1,if2]).
coverage(ib06,[ic5,ic6,ic7,if1,if2]).
coverage(ib07,[ic5,ic6,ic7,if1,if2]).

/------END-OF----FUNCTIONAL---KNOWLEDGE-BASE--------/



/------------BEHAVIORAL---KNOWLEDGE-BASE------------/

instbi(ibf1,ok,_,_,in).
instbi(ibf2,ok,_,_,in).

instbi(if1,ok,_,_,in).
instbi(if2,ok,_,_,in).

instbi(igf1,ok,_,_,out).
instbi(igf2,ok,_,_,out).

instbi(ibck,ok,_,_,in).

instbi(ib00,ok,_,_,in).
instbi(ib01,ok,_,_,in).
instbi(ib02,ok,_,_,in).
instbi(ib03,ok,_,_,in).

instbi(ib04,unknown,_,_,out).
instbi(ib05,unknown,_,_,out).
instbi(ib06,erroneous,_,_,out).
instbi(ib07,erroneous,_,_,out).

instbi(ibp0,unknown,_,_,in).
instbi(ibn0,unknown,_,_,in).
```

```
instbi(ibp1,unknown,_,_,in).
instbi(ibn1,unknown,_,_,in).
instbi(ibp2,unknown,_,_,in).
instbi(ibn2,unknown,_,_,in).
instbi(ibp3,unknown,_,_,in).
instbi(ibn3,unknown,_,_,in).

instbi(ip0,ok,_,_,out).
instbi(in0,ok,_,_,out).
instbi(ip1,ok,_,_,out).
instbi(in1,ok,_,_,out).
instbi(ip2,ok,_,_,out).
instbi(in2,ok,_,_,out).
instbi(ip3,erroneous,_,_,out).
instbi(in3,erroneous,_,_,out).

instbi(ic0,unknown,_,_,in).
instbi(ic1,unknown,_,_,in).
instbi(ic2,unknown,_,_,in).
instbi(ic3,ok,_,_,in).
instbi(ic4,ok,_,_,in).
instbi(ic5,ok,_,_,in).

instbi(ibc0,unknown,_,_,out).
instbi(ibc1,unknown,_,_,out).
instbi(ibc2,unknown,_,_,out).
instbi(ibc3,ok,_,_,out).
instbi(ibc4,ok,_,_,out).
instbi(ibc5,ok,_,_,out).

instbi(e0,ok,_,_,out).
instbi(e1,ok,_,_,out).
instbi(e2,ok,_,_,out).
instbi(e3,ok,_,_,out).

instbi(ck,ok,_,_,out).

instbi(s0,unknown,_,_,in).
instbi(s1,unknown,_,_,in).
instbi(s2,erroneous,_,_,in).
instbi(s3,erroneous,_,_,in).


hypothesis(multseq).

/-------END-OF----BEHAVIORAL---KNOWLEDGE-BASE-------/
/-------------END-OF-DESCRIPTION-LEVEL   I--------------/
```

```
/----------------DESCRIPTION-LEVEL    II------------------/
/---------ESTRUCTURAL---KNOWLEDGE-BASE-------------/

block(dig,[ic0,ic1,ic2,ic3,ic4,ic5,ip0,in0,ip1,in1,ip2,in2,ip3,in3,if1,if2,ib0
4,ib05,ib06,ib07]).

block(e1,[e11,e12,e13]).
block(e2,[e21,e22,e23]).
block(e3,[e31,e32,e33]).

block(ne1,[ne11,ne12,ne13]).
block(ne2,[ne21,ne22,ne23]).
block(ne3,[ne31,ne32,ne33]).
block(ne4,[ne41,ne42,ne43]).
block(ne5,[ne51,ne52,ne53]).
block(ne6,[ne61,ne62,ne63]).

block(i1,[i11,i12]).
block(i2,[i21,i22]).
block(i3,[i31,i32]).
block(i4,[i41,i42]).
block(i5,[i51,i52]).
block(i6,[i61,i62]).
block(i7,[i71,i72]).
block(i8,[i81,i82]).

block(ou1,[ou11,ou12,ou13]).
block(ou2,[ou21,ou22,ou23]).

block(d1,[d11,d12,d13]).
block(d2,[d21,d22,d23]).
block(d3,[d31,d32,d33]).
block(d4,[d41,d42,d43]).
block(d5,[d51,d52,d53]).
block(d6,[d61,d62,d63]).
block(d7,[d71,d72,d73]).
block(d8,[d81,d82,d83]).
block(d9,[d91,d92,d93]).
block(d10,[101,d102,d103]).
block(d11,[d111,d112,d113]).
block(d12,[d121,d122,d123]).
block(d13,[d131,d132,d133]).



blockint(e11,in,_,_).
blockint(e12,in,_,_).
blockint(e13,out,_,_).

blockint(e21,in,_,_).
blockint(e22,in,_,_).
blockint(e23,out,_,_).

blockint(e31,in,_,_).
blockint(e32,in,_,_).
blockint(e33,out,_,_).
```

```
blockint(ne11,in,_,_).
blockint(ne12,in,_,_).
blockint(ne13,out,_,_).

blockint(ne21,in,_,_).
blockint(ne22,in,_,_).
blockint(ne23,out,_,_).
blockint(ne31,in,_,_).
blockint(ne32,in,_,_).
blockint(ne33,out,_,_).

blockint(ne41,in,_,_).
blockint(ne42,in,_,_).
blockint(ne43,out,_,_).

blockint(ne51,in,_,_).
blockint(ne52,in,_,_).
blockint(ne53,out,_,_).

blockint(ne61,in,_,_).
blockint(ne62,in,_,_).
blockint(ne63,out,_,_).

blockint(i11,in,_,_).
blockint(i12,out,_,_).

blockint(i21,in,_,_).
blockint(i22,out,_,_).

blockint(i31,in,_,_).
blockint(i32,out,_,_).

blockint(i41,in,_,_).
blockint(i42,out,_,_).

blockint(i51,in,_,_).
blockint(i52,out,_,_).

blockint(i61,in,_,_).
blockint(i62,out,_,_).

blockint(i71,in,_,_).
blockint(i72,out,_,_).

blockint(i81,in,_,_).
blockint(i82,out,_,_).

blockint(i91,in,_,_).
blockint(i92,out,_,_).

blockint(ou11,in,_,_).
blockint(ou12,in,_,_).
blockint(ou13,out,_,_).

blockint(ou21,in,_,_).
blockint(ou22,in,_,_).
```

```
blockint(ou23,out,_,_).

blockint(d11,in,_,_).
blockint(d12,in,_,_).
blockint(d13,out,_,_).

blockint(d21,in,_,_).
blockint(d22,in,_,_).
blockint(d23,out,_,_).

blockint(d31,in,_,_).
blockint(d32,in,_,_).
blockint(d33,out,_,_).

blockint(d41,in,_,_).
blockint(d42,in,_,_).
blockint(d43,out,_,_).

blockint(d51,in,_,_).
blockint(d52,in,_,_).
blockint(d53,out,_,_).

blockint(d61,in,_,_).
blockint(d62,in,_,_).
blockint(d63,out,_,_).

blockint(d71,in,_,_).
blockint(d72,in,_,_).
blockint(d73,out,_,_).

blockint(d81,in,_,_).
blockint(d82,in,_,_).
blockint(d83,out,_,_).

blockint(d91,in,_,_).
blockint(d92,in,_,_).
blockint(d93,out,_,_).

blockint(d101,in,_,_).
blockint(d102,in,_,_).
blockint(d103,out,_,_).

blockint(d111,in,_,_).
blockint(d112,in,_,_).
blockint(d113,out,_,_).

blockint(d121,in,_,_).
blockint(d122,in,_,_).
blockint(d123,out,_,_).

blockint(d131,in,_,_).
blockint(d132,in,_,_).
blockint(d133,out,_,_).

blockint(if1,out,_,_).
blockint(if2,out,_,_).
```

```
blockint(ic0,out,_,_).
blockint(ic1,out,_,_).
blockint(ic2,out,_,_).
blockint(ic3,out,_,_).
blockint(ic4,out,_,_).
blockint(ic5,out,_,_).

blockint(ib04,in,_,_).
blockint(ib05,in,_,_).
blockint(ib06,in,_,_).
blockint(ib07,in,_,_).

blockint(ip0,in,_,_).
blockint(in0,in,_,_).
blockint(ip1,in,_,_).
blockint(in1,in,_,_).
blovkint(ip2,in,_,_).
blockint(in2,in,_,_).
blockint(ip3,in,_,_).
blockint(in3,in,_,_).


wire(w1,[ic5,e11,i11]).
wire(w2,[ic4,e12,e22,i21]).
wire(w3,[ic3,e32,i31]).
wire(w4,[i12,e21]).
wire(w5,[i22,e31]).
wire(w6,[i32,d42]).
wire(w7,[e13,d12]).
wire(w8,[e23,d22]).
wire(w9,[e33,d32]).
wire(w10,[d13,ne62,ou11,ou21]).
wire(w11,[d23,ne52,ou12]).
wire(w12,[d33,ne42,ou22]).
wire(w13,[d43,ne32]).
wire(w14,[ou13,d52]).
wire(w15,[ou23,d62]).
wire(w16,[d53,d102]).
wire(w17,[d63,d112]).
wire(w18,[d73,ne11]).
wire(w19,[d83,ne21,i41]).
wire(w20,[d93,i51]).
wire(w21,[i52,ne22]).
wire(w22,[ne23,ne12]).
wire(w23,[i42,d122]).
wire(w24,[ne13,d132]).
wire(w25,[ne33,i61,ip0]).
wire(w26,[ne43,i71,ip1]).
wire(w27,[ne53,i81,ip2]).
wire(w28,[ne63,i91,ip3]).
wire(w29,[i62,in0]).
wire(w30,[i72,in1]).
wire(w31,[i82,in2]).
wire(w32,[i92,in3]).
wire(w33,[ic2,d72]).
```

```
wire(w34,[ic1,d82]).
wire(w35,[ic0,d92]).
wire(w36,[d123,ib05]).
wire(w37,[d133,ib04]).
wire(w38,[d113,ib06]).
wire(w39,[d103,ib07]).
wire(w40,[if1,d51,d61,d71,d81,d91,ne31,ne41,ne51,ne61]).
wire(w41,[if2,d101,d111,d121,d131,d11,d21,d31,d41]).

/-------END-OF----ESTRUCTURAL---KNOWLEDGE-BASE------/


/------------FUNCTIONAL---KNOWLEDGE-BASE-------------/

coverage(e13,[e11,e12]).
coverage(e23,[e21,e22]).
coverage(e33,[e31,e32]).

coverage(ne13,[ne11,ne12]).
coverage(ne23,[ne21,ne22]).
coverage(ne33,[ne31,ne32]).
coverage(ne43,[ne41,ne42]).
coverage(ne53,[ne51,ne52]).
coverage(ne63,[ne61,ne62]).

coverage(i12,[i11]).
coverage(i22,[i21]).
coverage(i32,[i31]).
coverage(i42,[i41]).
coverage(i52,[i51]).
coverage(i62,[i61]).
coverage(i72,[i71]).
coverage(i82,[i81]).
coverage(i92,[i91]).

coverage(ou13,[ou11,ou12]).
coverage(ou23,[ou21,ou22]).

coverage(d13,[d11,d12]).
coverage(d23,[d21,d22]).
coverage(d33,[d31,d32]).
coverage(d43,[d41,d42]).
coverage(d53,[d51,d52]).
coverage(d63,[d61,d62]).
coverage(d73,[d71,d72]).
coverage(d83,[d81,d82]).
coverage(d93,[d91,d92]).
coverage(d103,[d101,d102]).
coverage(d113,[d111,d112]).
coverage(d123,[d121,d122]).
coverage(d133,[d131,d132]).

/------END-OF-----FUNCTIONAL---KNOWLEDGE-BASE-------/
```

```
/-----------BEHAVIORAL---KNOWLEDGE-BASE-----------/

instbi(e11,ok,_,_,in).
instbi(e12,ok,_,_,in).
instbi(e13,erroneous,_,_,out).

instbi(e21,ok,_,_,in).
instbi(e22,ok,_,_,in).
instbi(e23,ok,_,_,out).

instbi(e31,ok,_,_,in).
instbi(e32,ok,_,_,in).
instbi(e33,ok,_,_,out).

instbi(ne11,ok,_,_,in).
instbi(ne12,ok,_,_,in).
instbi(ne13,ok,_,_,out).

instbi(ne21,ok,_,_,in).
instbi(ne22,ok,_,_,in).
instbi(ne23,ok,_,_,out).

instbi(ne31,ok,_,_,in).
instbi(ne32,ok,_,_,in).
instbi(ne33,ok,_,_,out).

instbi(ne41,ok,_,_,in).
instbi(ne42,ok,_,_,in).
instbi(ne43,ok,_,_,out).

instbi(ne51,ok,_,_,in).
instbi(ne52,ok,_,_,in).
instbi(ne53,ok,_,_,out).

instbi(ne61,ok,_,_,in).
instbi(ne62,erroneous,_,_,in).
instbi(ne63,erroneous,_,_,out).

instbi(i11,ok,_,_,in).
instbi(i12,ok,_,_,out).

instbi(i21,ok,_,_,in).
instbi(i22,ok,_,_,out).

instbi(i31,ok,_,_,in).
instbi(i32,ok,_,_,out).

instbi(i41,ok,_,_,in).
instbi(i42,ok,_,_,out).

instbi(i51,ok,_,_,in).
instbi(i52,ok,_,_,out).

instbi(i61,ok,_,_,in).
```

```
instbi(i62,ok,_,_,out).

instbi(i71,ok,_,_,in).
instbi(i72,ok,_,_,out).

instbi(i81,ok,_,_,in).
instbi(i82,ok,_,_,out).

instbi(i91,erroneous,_,_,in).
instbi(i92,erroneous,_,_,out).

instbi(ou11,erroneous,_,_,in).
instbi(ou12,ok,_,_,in).
instbi(ou13,erroneous,_,_,out).

instbi(ou21,erroneous,_,_,in).
instbi(ou22,ok,_,_,in).
instbi(ou23,erroneous,_,_,out).

instbi(d11,ok,_,_,in).
instbi(d12,erroneous,_,_,in).
instbi(d13,erroneous,_,_,out).

instbi(d21,ok,_,_,in).
instbi(d22,ok,_,_,in).
instbi(d23,ok,_,_,out).

instbi(d31,ok,_,_,in).
instbi(d32,ok,_,_,in).
instbi(d33,ok,_,_,out).
instbi(d41,ok,_,_,in).
instbi(d42,ok,_,_,in).
instbi(d43,ok,_,_,out).

instbi(d51,ok,_,_,in).
instbi(d52,erroneous,_,_,in).
instbi(d53,erroneous,_,_,out).

instbi(d61,ok,_,_,in).
instbi(d62,erroneous,_,_,in).
instbi(d63,erroneous,_,_,out).

instbi(d71,ok,_,_,in).
instbi(d72,ok,_,_,in).
instbi(d73,ok,_,_,out).

instbi(d81,ok,_,_,in).
instbi(d82,ok,_,_,in).
instbi(d83,ok,_,_,out).

instbi(d91,ok,_,_,in).
instbi(d92,ok,_,_,in).
instbi(d93,ok,_,_,out).

instbi(d101,ok,_,_,in).
instbi(d102,erroneous,_,_,in).
instbi(d103,erroneous,_,_,out).
```

```
instbi(d111,ok,_,_,in).
instbi(d112,erroneous,_,_,in).
instbi(d113,erroneous,_,_,out).

instbi(d121,ok,_,_,in).
instbi(d122,ok,_,_,in).
instbi(d123,ok,_,_,out).

instbi(d131,ok,_,_,in).
instbi(d132,ok,_,_,in).
instbi(d133,ok,_,_,out).


instbi(if1,ok,_,_,out).
instbi(if2,ok,_,_,out).

instbi(ib04,ok,_,_,in).
instbi(ib05,ok,_,_,in).
instbi(ib06,erroneous,_,_,in).
instbi(ib07,erroneous,_,_,in).

instbi(ic0,ok,_,_,out).
instbi(ic1,ok,_,_,out).
instbi(ic2,ok,_,_,out).

instbi(ip0,ok,_,_,in).
instbi(in0,ok,_,_,in).
instbi(ip1,ok,_,_,in).
instbi(in1,ok,_,_,in).
instbi(ip2,ok,_,_,in).
instbi(in2,ok,_,_,in).
instbi(ip3,erroneous,_,_,in).
instbi(in3,erroneous,_,_,in).

instbi(ic3,ok,_,_,out).
instbi(ic4,ok,_,_,out).
instbi(ic5,ok,_,_,out).


hypothesis(multcomb).

/------END-OF----BEHAVIORAL---KNOWLEDGE-BASE--------/
/------------END-OF-DESCRIPTION-LEVEL    II---------------/
```

ANEXO II

Arquivos de entrada do Sistema Especialista PESTICIDE: Bases de Conhecimento Estrutural, Funcional e Comportamental, geradas para a *falha em uma trilha* de acordo com a modelagem do circuito CONVERSO (apresentada nas figuras 6.8.b e 6.10).

```
/-----------------DESCRIPTION-LEVEL    I----------------/
/----------ESTRUCTURAL---KNOWLEDGE-BASE-----------/

block(ext,[e0,e1,e2,e3,s0,s1,s2,s3,ck]).

block(ana1,[ib00,ib01,ib02,ib03,ibf1,ibf2,ibc3,ibc4,ibc5]).

block(ana2,[ib00x,ib01x,ib02x,ib03x,ibf11,ibf22,ibp0,ibn0,ibp1,ibn1,inp2,
ibn2,ibp3,ibn3,ibc0,ibc1,ibc2]).

block(dig1,[ic3,ic4,ic5,ip0,in0,ip1,in1,ip2,in2,ip3,in3,if1,if2,ib06,ib07]).

block(dig2,[ic0,ic1,ic2,if11,if22,ib04,ib05]).

block(pha,[ibck,igf1,igf2]).


blockint(e0,out,0,1).
blockint(e1,out,0,1).
blockint(e2,out,0,1).
blockint(e3,out,0,1).

blockint(s0,in,0,1).
blockint(s1,in,0,1).
blockint(s2,in,0,1).
blockint(s3,in,0,1).

blockint(ck,out,0,1).

blockint(ibck,in,0,1).

blockint(igf1,out,0,1).
blockint(igf2,out,0,1).

blockint(if1,in,0,1).
blockint(if2,in,0,1).

blockint(if11,in,0,1).
blockint(if22,in,0,1).

blockint(ibf1,in,0,1).
blockint(ibf2,in,0,1).

blockint(ibf11,in,0,1).
blockint(ibf22,in,0,1).

blockint(ib00,in,0,1).
blockint(ib01,in,0,1).
blockint(ib02,in,0,1).
blockint(ib03,in,0,1).

blockint(ib00x,in,0,1).
blockint(ib01x,in,0,1).
blockint(ib02x,in,0,1).
blockint(ib03x,in,0,1).
```

```
blockint(ib04,out,0,1).
blockint(ib05,out,0,1).
blockint(ib06,out,0,1).
blockint(ib07,out,0,1).

blockint(ibp0,in,0,1).
blockint(ibn0,in,0,1).
blockint(ibp1,in,0,1).
blockint(ibn1,in,0,1).
blockint(ibp2,in,0,1).
blockint(ibn2,in,0,1).
blockint(ibp3,in,0,1).
blockint(ibn3,in,0,1).

blockint(ip0,out,0,1).
blockint(in0,out,0,1).
blockint(ip1,out,0,1).
blockint(in1,out,0,1).
blockint(ip2,out,0,1).
blockint(in2,out,0,1).
blockint(ip3,out,0,1).
blockint(in3,out,0,1).

blockint(ibc0,out,0,1).
blockint(ibc1,out,0,1).
blockint(ibc2,out,0,1).
blockint(ibc3,out,0,1).
blockint(ibc4,out,0,1).
blockint(ibc5,out,0,1).

blockint(ic0,in,0,1).
blockint(ic1,in,0,1).
blockint(ic2,in,0,1).
blockint(ic3,in,0,1).
blockint(ic4,in,0,1).
blockint(ic5,in,0,1).



wire(w42,[e0,ib00,ib00x]).
wire(w43,[e1,ib01,ib01x]).
wire(w44,[e2,ib02,ib02x]).
wire(w45,[e3,ib03,ib03x]).
wire(w46,[ibp0,ip0]).
wire(w47,[ibn0,in0]).
wire(w48,[ibp1,ip1]).
wire(w49,[ibn1,in1]).
wire(w50,[ibp2,ip2]).
wire(w51,[ibn2,in2]).
wire(w52,[ibp3,ip3]).
wire(w53,[ibn3,in3]).
wire(w54,[ibc0,ic0]).
wire(w55,[ibc1,ic1]).
wire(w56,[ibc2,ic2]).
wire(w57,[ibc3,ic3]).
wire(w58,[ibc4,ic4]).
wire(w59,[ibc5,ic5]).
```

```
wire(w59,[igf1,ibf1,if1,ibf11,if11]).
wire(w60,[igf2,ibf2,if2,ibf22,if22]).
wire(w61,[s0,ib04]).
wire(w62,[s1,ib05]).
wire(w63,[s2,ib06]).
wire(w64,[s3,ib07]).
wire(w65,[ibck,ck]).

/------END-OF----ESTRUCTURAL---KNOWLEDGE-BASE-------/


/------------FUNCTIONAL---KNOWLEDGE-BASE-------------/

coverage(igf1,[ibck]).
coverage(igf2,[ibck]).

coverage(ibc0,[ibp0,ibn0,ibp1,ibn1,ibp2,ibn2,ibp3,ibn3,ibf22]).
coverage(ibc1,[ibp0,ibn0,ibp1,ibn1,ibp2,ibn2,ibp3,ibn3,ibf22]).
coverage(ibc2,[ibp0,ibn0,ibp1,ibn1,ibp2,ibn2,ibp3,ibn3,ibf22]).
coverage(ibc3,[ib00,ib01,ib02,ib03,ibf1,ibf2]).
coverage(ibc4,[ib00,ib01,ib02,ib03,ibf1,ibf2]).
coverage(ibc5,[ib00,ib01,ib02,ib03,ibf1,ibf2]).

coverage(ip0,[ic3,ic4,ic5,if1]).
coverage(in0,[ic3,ic4,ic5,if1]).
coverage(ip1,[ic3,ic4,ic5,if1]).
coverage(in1,[ic3,ic4,ic5,if1]).
coverage(ip2,[ic3,ic4,ic5,if1]).
coverage(in2,[ic3,ic4,ic5,if1]).
coverage(ip3,[ic3,ic4,ic5,if1]).
coverage(in3,[ic3,ic4,ic5,if1]).

coverage(ib04,[ic0,ic1,ic2,if1,if22]).
coverage(ib05,[ic0,ic1,ic2,if1,if22]).
coverage(ib06,[ic5,ic6,ic7,if1,if2]).
coverage(ib07,[ic5,ic6,ic7,if1,if2]).

/------END-OF----FUNCTIONAL---KNOWLEDGE-BASE--------/


/------------BEHAVIORAL---KNOWLEDGE-BASE-------------/

instbi(ibf1,ok,_,_,in).
instbi(ibf2,ok,_,_,in).
instbi(ibf11,unknown,_,_,in).
instbi(ibf22,unknown,_,_,in).

instbi(if1,ok,_,_,in).
instbi(if2,ok,_,_,in).
instbi(if11,unknown,_,_,in).
instbi(if22,unknown,_,_,in).
instbi(igf1,ok,_,_,out).
```

```
instbi(igf2,ok,_,_,out).

instbi(ibck,ok,_,_,in).

instbi(ib00,ok,_,_,in).
instbi(ib01,ok,_,_,in).
instbi(ib02,ok,_,_,in).
instbi(ib03,ok,_,_,in).

instbi(ib04,unknown,_,_,out).
instbi(ib05,unknown,_,_,out).
instbi(ib06,erroneous,_,_,out).
instbi(ib07,ok_,_,out).

instbi(ibp0,unknown,_,_,in).
instbi(ibn0,unknown,_,_,in).
instbi(ibp1,unknown,_,_,in).
instbi(ibn1,unknown,_,_,in).
instbi(ibp2,unknown,_,_,in).
instbi(ibn2,unknown,_,_,in).
instbi(ibp3,unknown,_,_,in).
instbi(ibn3,unknown,_,_,in).

instbi(ip0,ok,_,_,out).
instbi(in0,ok,_,_,out).
instbi(ip1,ok,_,_,out).
instbi(in1,ok,_,_,out).
instbi(ip2,ok,_,_,out).
instbi(in2,ok,_,_,out).
instbi(ip3,ok,_,_,out).
instbi(in3,ok,_,_,out).

instbi(ic0,unknown,_,_,in).
instbi(ic1,unknown,_,_,in).
instbi(ic2,unknown,_,_,in).
instbi(ic3,ok,_,_,in).
instbi(ic4,ok,_,_,in).
instbi(ic5,ok,_,_,in).

instbi(ibc0,unknown,_,_,out).
instbi(ibc1,unknown,_,_,out).
instbi(ibc2,unknown,_,_,out).
instbi(ibc3,ok,_,_,out).
instbi(ibc4,ok,_,_,out).
instbi(ibc5,ok,_,_,out).

instbi(e0,ok,_,_,out).
instbi(e1,ok,_,_,out).
instbi(e2,ok,_,_,out).
instbi(e3,ok,_,_,out).

instbi(ck,ok,_,_,out).

instbi(s0,unknown,_,_,in).
instbi(s1,unknown,_,_,in).
instbi(s2,erroneous,_,_,in).
instbi(s3,ok,_,_,in).
```

```
hypothesis(multcomb).

/-------END-OF----BEHAVIORAL---KNOWLEDGE-BASE-------/
/-------------END-OF-DESCRIPTION-LEVEL   I-------------/


/---------------DESCRIPTION-LEVEL   II.A----------------/
/---------ESTRUCTURAL---KNOWLEDGE-BASE------------/

block(dig1,[ic3,ic4,ic5,ip0,in0,ip1,in1,ip2,in2,ip3,in3,if1,if2,ib06,ib07]).

block(e1,[e11,e12,e13]).
block(e2,[e21,e22,e23]).
block(e3,[e31,e32,e33]).

block(ne3,[ne31,ne32,ne33]).
block(ne4,[ne41,ne42,ne43]).
block(ne5,[ne51,ne52,ne53]).
block(ne6,[ne61,ne62,ne63]).

block(i1,[i11,i12]).
block(i2,[i21,i22]).
block(i3,[i31,i32]).
block(i6,[i61,i62]).
block(i7,[i71,i72]).
block(i8,[i81,i82]).

block(ou1,[ou11,ou12,ou13]).
block(ou2,[ou21,ou22,ou23]).

block(d1,[d11,d12,d13]).
block(d2,[d21,d22,d23]).
block(d3,[d31,d32,d33]).
block(d4,[d41,d42,d43]).
block(d5,[d51,d52,d53]).
block(d6,[d61,d62,d63]).
block(d10,[101,d102,d103]).
block(d11,[d111,d112,d113]).


blockint(e11,in,_,_).
blockint(e12,in,_,_).
blockint(e13,out,_,_).

blockint(e21,in,_,_).
blockint(e22,in,_,_).
blockint(e23,out,_,_).

blockint(e31,in,_,_).
blockint(e32,in,_,_).
blockint(e33,out,_,_).

blockint(ne31,in,_,_).
```

```
blockint(ne32,in,_,_).
blockint(ne33,out,_,_).

blockint(ne41,in,_,_).
blockint(ne42,in,_,_).
blockint(ne43,out,_,_).

blockint(ne51,in,_,_).
blockint(ne52,in,_,_).
blockint(ne53,out,_,_).

blockint(ne61,in,_,_).
blockint(ne62,in,_,_).
blockint(ne63,out,_,_).

blockint(i11,in,_,_).
blockint(i12,out,_,_).

blockint(i21,in,_,_).
blockint(i22,out,_,_).

blockint(i31,in,_,_).
blockint(i32,out,_,_).

blockint(i61,in,_,_).
blockint(i62,out,_,_).

blockint(i71,in,_,_).
blockint(i72,out,_,_).

blockint(i81,in,_,_).
blockint(i82,out,_,_).

blockint(i91,in,_,_).
blockint(i92,out,_,_).

blockint(ou11,in,_,_).
blockint(ou12,in,_,_).
blockint(ou13,out,_,_).

blockint(ou21,in,_,_).
blockint(ou22,in,_,_).
blockint(ou23,out,_,_).

blockint(d11,in,_,_).
blockint(d12,in,_,_).
blockint(d13,out,_,_).

blockint(d21,in,_,_).
blockint(d22,in,_,_).
blockint(d23,out,_,_).

blockint(d31,in,_,_).
blockint(d32,in,_,_).
blockint(d33,out,_,_).

blockint(d41,in,_,_).
```

```
blockint(d42,in,_,_).
blockint(d43,out,_,_).

blockint(d51,in,_,_).
blockint(d52,in,_,_).
blockint(d53,out,_,_).

blockint(d61,in,_,_).
blockint(d62,in,_,_).
blockint(d63,out,_,_).

blockint(d101,in,_,_).
blockint(d102,in,_,_).
blockint(d103,out,_,_).

blockint(d111,in,_,_).
blockint(d112,in,_,_).
blockint(d113,out,_,_).

blockint(if1,out,_,_).
blockint(if2,out,_,_).


blockint(ic3,out,_,_).
blockint(ic4,out,_,_).
blockint(ic5,out,_,_).

blockint(ib06,in,_,_).
blockint(ib07,in,_,_).

blockint(ip0,in,_,_).
blockint(in0,in,_,_).
blockint(ip1,in,_,_).
blockint(in1,in,_,_).
blovkint(ip2,in,_,_).
blockint(in2,in,_,_).
blockint(ip3,in,_,_).
blockint(in3,in,_,_).



wire(w1,[ic5,e11,i11]).
wire(w2,[ic4,e12,e22,i21]).
wire(w3,[ic3,e32,i31]).
wire(w4,[i12,e21]).
wire(w5,[i22,e31]).
wire(w6,[i32,d42]).
wire(w7,[e13,d12]).
wire(w8,[e23,d22]).
wire(w9,[e33,d32]).
wire(w10,[d13,ne62,ou11,ou21]).
wire(w11,[d23,ne52,ou12]).
wire(w12,[d33,ne42,ou22]).
wire(w13,[d43,ne32]).
wire(w14,[ou13,d52]).
wire(w15,[ou23,d62]).
wire(w16,[d53,d102]).
```

```
wire(w17,[d63,d112]).
wire(w25,[ne33,i61,ip0]).
wire(w26,[ne43,i71,ip1]).
wire(w27,[ne53,i81,ip2]).
wire(w28,[ne63,i91,ip3]).
wire(w29,[i62,in0]).
wire(w30,[i72,in1]).
wire(w31,[i82,in2]).
wire(w32,[i92,in3]).
wire(w38,[d113,ib06]).
wire(w39,[d103,ib07]).
wire(w40,[if1,d51,d61,ne31,ne41,ne51,ne61]).
wire(w41,[if2,d101,d111,d11,d21,d31,d41]).

/-------END-OF----ESTRUCTURAL---KNOWLEDGE-BASE------/


/------------FUNCTIONAL---KNOWLEDGE-BASE-------------/

coverage(e13,[e11,e12]).
coverage(e23,[e21,e22]).
coverage(e33,[e31,e32]).

coverage(ne33,[ne31,ne32]).
coverage(ne43,[ne41,ne42]).
coverage(ne53,[ne51,ne52]).
coverage(ne63,[ne61,ne62]).

coverage(i12,[i11]).
coverage(i22,[i21]).
coverage(i32,[i31]).
coverage(i62,[i61]).
coverage(i72,[i71]).
coverage(i82,[i81]).
coverage(i92,[i91]).

coverage(ou13,[ou11,ou12]).
coverage(ou23,[ou21,ou22]).

coverage(d13,[d11,d12]).
coverage(d23,[d21,d22]).
coverage(d33,[d31,d32]).
coverage(d43,[d41,d42]).
coverage(d53,[d51,d52]).
coverage(d63,[d61,d62]).
coverage(d103,[d101,d102]).
coverage(d113,[d111,d112]).

/------END-OF-----FUNCTIONAL---KNOWLEDGE-BASE-------/
```

```
/------------BEHAVIORAL---KNOWLEDGE-BASE------------/

instbi(e11,ok,_,_,in).
instbi(e12,ok,_,_,in).
instbi(e13,ok,_,_,out).

instbi(e21,ok,_,_,in).
instbi(e22,ok,_,_,in).
instbi(e23,ok,_,_,out).

instbi(e31,ok,_,_,in).
instbi(e32,ok,_,_,in).
instbi(e33,ok,_,_,out).

instbi(ne31,ok,_,_,in).
instbi(ne32,ok,_,_,in).
instbi(ne33,ok,_,_,out).

instbi(ne41,ok,_,_,in).
instbi(ne42,erroneous,_,_,in).
instbi(ne43,ok,_,_,out).

instbi(ne51,ok,_,_,in).
instbi(ne52,ok,_,_,in).
instbi(ne53,ok,_,_,out).

instbi(ne61,ok,_,_,in).
instbi(ne62,ok,_,_,in).
instbi(ne63,ok,_,_,out).

instbi(i11,ok,_,_,in).
instbi(i12,ok,_,_,out).

instbi(i21,ok,_,_,in).
instbi(i22,ok,_,_,out).

instbi(i31,ok,_,_,in).
instbi(i32,ok,_,_,out).

instbi(i61,ok,_,_,in).
instbi(i62,ok,_,_,out).

instbi(i71,ok,_,_,in).
instbi(i72,ok,_,_,out).

instbi(i81,ok,_,_,in).
instbi(i82,ok,_,_,out).

instbi(i91,ok,_,_,in).
instbi(i92,ok,_,_,out).

instbi(ou11,ok,_,_,in).
instbi(ou12,ok,_,_,in).
instbi(ou13,ok,_,_,out).

instbi(ou21,ok,_,_,in).
instbi(ou22,erroneous,_,_,in).
```

```
instbi(ou23,erroneous,_,_,out).

instbi(d11,ok,_,_,in).
instbi(d12,ok,_,_,in).
instbi(d13,ok,_,_,out).

instbi(d21,ok,_,_,in).
instbi(d22,ok,_,_,in).
instbi(d23,ok,_,_,out).

instbi(d31,ok,_,_,in).
instbi(d32,ok,_,_,in).
instbi(d33,ok,_,_,out).

instbi(d41,ok,_,_,in).
instbi(d42,ok,_,_,in).
instbi(d43,ok,_,_,out).

instbi(d51,ok,_,_,in).
instbi(d52,ok,_,_,in).
instbi(d53,ok,_,_,out).

instbi(d61,ok,_,_,in).
instbi(d62,erroneous,_,_,in).
instbi(d63,erroneous,_,_,out).

instbi(d101,ok,_,_,in).
instbi(d102,ok,_,_,in).
instbi(d103,ok,_,_,out).

instbi(d111,ok,_,_,in).
instbi(d112,erroneous,_,_,in).
instbi(d113,erroneous,_,_,out).

instbi(if1,ok,_,_,out).
instbi(if2,ok,_,_,out).

instbi(ib06,erroneous,_,_,in).
instbi(ib07,ok,_,_,in).

instbi(ip0,ok,_,_,in).
instbi(in0,ok,_,_,in).
instbi(ip1,ok,_,_,in).
instbi(in1,ok,_,_,in).
instbi(ip2,ok,_,_,in).
instbi(in2,ok,_,_,in).
instbi(ip3,ok,_,_,in).
instbi(in3,ok,_,_,in).

instbi(ic3,ok,_,_,out).
instbi(ic4,ok,_,_,out).
instbi(ic5,ok,_,_,out).

hypothesis(multcomb).

/------END-OF----BEHAVIORAL---KNOWLEDGE-BASE--------/
/------------END-OF-DESCRIPTION-LEVEL   II.A----------/
```

```
/----------------DESCRIPTION-LEVEL   II.B----------------/
/----------ESTRUCTURAL---KNOWLEDGE-BASE-------------/

block(dig2,[ic0,ic1,ic2,if11,if22,ib04,ib05]).


block(ne1,[ne11,ne12,ne13]).
block(ne2,[ne21,ne22,ne23]).

block(i4,[i41,i42]).
block(i5,[i51,i52]).

block(d7,[d71,d72,d73]).
block(d8,[d81,d82,d83]).
block(d9,[d91,d92,d93]).
block(d12,[d121,d122,d123]).
block(d13,[d131,d132,d133]).


blockint(ne11,in,_,_).
blockint(ne12,in,_,_).
blockint(ne13,out,_,_).

blockint(ne21,in,_,_).
blockint(ne22,in,_,_).
blockint(ne23,out,_,_).

blockint(i41,in,_,_).
blockint(i42,out,_,_).

blockint(i51,in,_,_).
blockint(i52,out,_,_).

blockint(d71,in,_,_).
blockint(d72,in,_,_).
blockint(d73,out,_,_).

blockint(d81,in,_,_).
blockint(d82,in,_,_).
blockint(d83,out,_,_).

blockint(d91,in,_,_).
blockint(d92,in,_,_).
blockint(d93,out,_,_).

blockint(d121,in,_,_).
blockint(d122,in,_,_).
blockint(d123,out,_,_).

blockint(d131,in,_,_).
blockint(d132,in,_,_).
```

```
blockint(d133,out,_,_).

blockint(if11,out,_,_).
blockint(if22,out,_,_).

blockint(ic0,out,_,_).
blockint(ic1,out,_,_).
blockint(ic2,out,_,_).

blockint(ib04,in,_,_).
blockint(ib05,in,_,_).


wire(w18,[d73,ne11]).
wire(w19,[d83,ne21,i41]).
wire(w20,[d93,i51]).
wire(w21,[i52,ne22]).
wire(w22,[ne23,ne12]).
wire(w23,[i42,d122]).
wire(w24,[ne13,d132]).
wire(w33,[ic2,d72]).
wire(w34,[ic1,d82]).
wire(w35,[ic0,d92]).
wire(w36,[d123,ib05]).
wire(w37,[d133,ib04]).
wire(w40,[if11,d71,d81,d91]).
wire(w41,[if22,d121,d131]).

/-------END-OF----ESTRUCTURAL---KNOWLEDGE-BASE------/


/-------------FUNCTIONAL---KNOWLEDGE-BASE------------/

coverage(ne13,[ne11,ne12]).
coverage(ne23,[ne21,ne22]).

coverage(i42,[i41]).
coverage(i52,[i51]).

coverage(d73,[d71,d72]).
coverage(d83,[d81,d82]).
coverage(d93,[d91,d92]).
coverage(d123,[d121,d122]).
coverage(d133,[d131,d132]).

/----END-OF------FUNCTIONAL---KNOWLEDGE-BASE--------/


/-------------BEHAVIORAL---KNOWLEDGE-BASE-----------/

instbi(ne11,ok,_,_,in).
instbi(ne12,ok,_,_,in).
```

```
instbi(ne13,ok,_,_,out).

instbi(ne21,ok,_,_,in).
instbi(ne22,ok,_,_,in).
instbi(ne23,ok,_,_,out).

instbi(i41,ok,_,_,in).
instbi(i42,ok,_,_,out).

instbi(i51,ok,_,_,in).
instbi(i52,ok,_,_,out).

instbi(d71,ok,_,_,in).
instbi(d72,ok,_,_,in).
instbi(d73,ok,_,_,out).

instbi(d81,ok,_,_,in).
instbi(d82,ok,_,_,in).
instbi(d83,ok,_,_,out).

instbi(d91,ok,_,_,in).
instbi(d92,ok,_,_,in).
instbi(d93,ok,_,_,out).

instbi(d121,ok,_,_,in).
instbi(d122,ok,_,_,in).
instbi(d123,ok,_,_,out).

instbi(d131,ok,_,_,in).
instbi(d132,ok,_,_,in).
instbi(d133,ok,_,_,out).

instbi(if11,ok,_,_,out).
instbi(if22,ok,_,_,out).

instbi(ib04,ok,_,_,in).
instbi(ib05,ok,_,_,in).

instbi(ic0,ok,_,_,out).
instbi(ic1,ok,_,_,out).
instbi(ic2,ok,_,_,out).


hypothesis(multcomb).

/-----END-OF--------BEHAVIORAL---KNOWLEDGE-BASE-----/
/------------END-OF-DESCRIPTION-LEVEL    II.B------------/
```

BIBLIOGRAFIA

[BAI84] BAILLE, G. **Testeur Logique de Circuits Integres.** Grenoble: TIM3/INPG, 1984. TR 84.

[BEA78] BEAUDET, P. R. Rotationally Ivariant Image Operator. In: INTERNATIONAL JOINT CONFERENCE ON PATTERN RECOGNITION, 4., 1978, Tóquio. **Proceedings...** Tóquio: IEEE, 1978. p.579-583.

[BEN90] BEN DAHKLIA, R. L.**Strategies de Test et Generation de Connaissances pour un Systeme Expert.** Grenoble : TIM3/INPG, 1984. Mémoire de fin d'études d'ingénieur concepteur en informatique.

[BEN89] BEN ALI, M. **Interface PESTICIDE/HILO3.** Grenoble : TIM3/INPG, 1989. Mémoire de fin d'études d'ingénieur concepteur en informatique.

[BOU87] BOURGEON, G. Electron beam tester, a tool for VLSI component analysis. In: EUROPEAN CONFERENCE ON ELECTRON & OPTICAL BEAM TESTING OF INTEGRATED CIRCUITS, 1., 1987, Grenoble. **Proceedings...** Grenoble : Microelectronic Engeneering Journal, 1987. p. 327-332.

[CAM72] CAMBRIDGE SCIENTIFIC INSTRUMENTS LIMITED. **Steoscan S600 Scanning Electron Microscope Operator's Manual.** June 1972.

[CAN83] CANNY, J. F. **Finding Edges and Lines in Images.** Massachussets: Massachussets Institute of Technology, 1983. Master Thesis.

[CHO72] CHOW, C. K.; Kaneko, T. Boundary Detection of Radiographic Images by a Threshold Method. In: IFIP CONGRESS,1971, Amsterdam. **Proceedings...**Amsterdam: North-Holland, 1972. p. 130-134.

[CON90] CONARD, D. et al. High Level Tools and Methods for Electron-Beam Debug and Failure Analysis of Integrated Circuits. In: Seminar on Intelligent Measurent Systems, July 1990, Budapest. **Proceedings...** Budapest: IEEE, 1990.

[CON91] CONARD, D. **Traitement d'Images en Analyse de Deffaillances de Circuits Integres par Faisceau d'Electrons.** Grenoble : TIM3/INPG, 1991. Tese de Doutorado.

[COL89] COLLIN, J. P.; CONARD, D.; COURTOIS, B. et al. Failure Analysis Using E - Beam. In: EUROPEAN CONFERENCE ON ELECTRON & OPTICAL BEAM TESTING OF INTEGRATED CIRCUITS, 2., 1989, Grenoble. **Proceedings...** Duisburg : Microelectronic Engeneering Journal, 1989.

[CON87a] CONCINA, S.; RICHARDSON, N . IDS 5000: an integrated diagnostic system for VLSI. In: EUROPEAN CONFERENCE ON ELECTRON & OPTICAL BEAM TESTING OF INTEGRATED CIRCUITS, 1., 1987, Grenoble. **Proceedings...** Grenoble : Microelectronic Engeneering Journal, 1987. p. 339-342.

[CON87b] CONCINA, S. ; LIU, G. Integrating design information for IC diagnosis. In: ACM - IEEE DESIGN AUTOMATION CONFERENCE, 24., 1987, Flórida. **Proceedings...** Florida: IEEE, 1987. p. 251-257.

[DOS89] DOSSA, M. K. **Estudo e Projeto de um Conversor A/D Integrado para Aplicação em um Digitalizador de Imagens em Tempo Real.** Porto Alegre: CPGCC-UFRGS, 1989. Dissertação de Mestrado.

[DOY62] DOYLE, W. Operations Useful for Similarity-Invariant Pattern Recognition. **J. of ACM**, New York, v. 9, n. 2, p. 259-267, Apr. 1962.

[FAZ81] FAZEKAS, P. Scanning Electron Beam Probes VLSI Chips. **Electronics**, New York, p. 105-112, 14 July 1981.

[FRE77] FREEDMAN, H. **Analysis of Line Drawings - Digital Image Processing and Analysis.** New York: Noordhoff -Leyden, 1977.

[GOR87] GORLICH, S.; HARBECK, H.; KEBLER, P. et al. Integration of CAD, CAT and electron-beam testing for IC-internal logic verification. In: IEEE INTERNATIONAL TEST CONFERENCE, 18., 1987, Washington. **Proceedings...** Washington: IEEE, 1987. p. 566 - 574.

[HEM87] HENNING, S. S.; KNOWLES GRAHAM,W. R.; PLOWS, S. CAD system interface for a stand-alone e-beam tester. In: EUROPEAN CONFERENCE ON ELECTRON & OPTICAL BEAM TESTING OF INTEGRATED CIRCUITS, 1., 1987, Grenoble. **Proceedings...** Grenoble : Microelectronic Engeneering Journal, 1987. p.317-325.

[HUR86] HURTADO, C. J. O. **Uma Técnica de Depuração e Teste de Circuitos Integrados Usando um Microscópio Eletrônico.** Porto Alegre: CPGCC - UFRGS, 1986. Dissertação de Mestrado

[KOM87] KOMATSU, F. ; MIYOSHI, M.; SANO, T.; et al. An electron beam test system linked with a CAD database. In : EUROPEAN CONFERENCE ON ELECTRON & OPTICAL BEAM TESTING OF INTEGRATED CIRCUITS, 1., 1987, Grenoble. **Proceedings...** Grenoble : Microelectronic Engeneering Journal, 1987. p. 267- 274.

[KUJ86] KUJI, N. ; TAMAMA, T. An automated e-beam tester with CAD interface: FINDER. In: IEEE INTERNATIONAL TEST CONFERENCE, 17., 1986, Washington. **Proceedings...** Washington: IEEE, 1986. p. 857-863.

[LEE89] LEE, W. T. Engeneering a Device for E-Beam Probing. **IEEE Design and Test of Computers**. New York, v. 6, n. 3, p. 36-49. June 1989.

[MAR89a] MARZOUKI, M.; COURTOIS, B. Debugging integrated circuits: A. I. can help. In: EUROPEAN TEST CONFERENCE, 1989, Paris. **Proceedings...** Paris: IEEE Computer Society Press, 1989. p. 184-191.

[MAR89b] MARZOUKI, M.; LAURENT, J.; COURTOIS, B. A unified use of deep and shallow knowledge in an expert system for prototype validation of integrated circuits. In: JOURNÉES INTERNATIONALES D'AVIGNON SUR LES S. E. ET LEURS APPLICATIONS, Mai 1989. **Proceedings...** Avignon: IEEE, 1989. p. 55-69.

[MAR91a] MARZOUKI, M. **Approches a Base de Connaissances pour le Test de Circuits VLSI: Application a la Validation de Protopypes dans le Cadre d'un Test Sans Contact**. Grenoble: TIM3/INPG, 1991. Tese de Doutorado.

[MAR91b] MARZOUKI, M.; VARGAS, F. L. Knowledge - Based Debugging of ASICs: Real Case Study and Performance Analysis. In: INTERNATIONAL CONFERENCE ON COMPUTER AIDED DESIGN - ICCAD'91, 1991, Santa Clara, USA. **Proceedings...** Santa Clara: IEEE, 1991.

[MAR91c] MARZOUKI, M.; VARGAS, F. L. Using a Knowledge - Based System for Automatic Debugging: Case Study and Performance Analysis. In: EUROPEAN CONFERENCE ON ELECTRON AND OPTICAL BEAM TESTING OF INTEGRATED CIRCUITS, 3., 1991, Como, Italy. **Proceedings...** Como: Microelectronic Engeneering Journal, 1991.

[MAR91d] MARZOUKI, M.; VARGAS, F. L. Debugging Prototype Circuits Using a Knowledge - Based Approach: a case study. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE MICROELETRONICA, 6., 1991, Belo Horizonte. **Anais...** Belo Horizonte : Sociedade Brasileira de Microeletrônica, 1991. p. 517-526.

[MAS78] MASON, D.; LAUDER, I.; RUTOVITZ, D.; SPOWART, G. **Measurement of C-Bands in Human Chromosomes.** Pré-edição.

[MED87] MEDIONI, G.; YASUMOTO, Y. Corner Detection and Curve Representation Using Cubic B-Spines. **Computer Vision, Graphics, and Image Processing.** n. 39, p.. 267-278, 1987.

[MEL88] MELGARA, M.; BATTU, M.; MARZOUKI, M. et al. Design for E - Beam Debuggability. In: CAVE WORKSHOP, 1988, Sintra. **Proceedings...** Sintra: IEEE, 1988.

[MEN81] MENZEL, E.; KUBALEK, E. Electron Beam Techniques for Integrated Circuits . In: SCANNING ELECTRON MICROSCOPY CONFERENCE, 1981, Chicago. **Proceedings...** Chicago: IEEE, 1981.

[MEN83] MENZEL, E.; KUBALEK, E. Review - Fundamentals of Electron Beam Testing of Integrated Circuits. **SCANNING**, New York, v. 5, p.103-122. 1983.

[PAP65] PAPOULIS, A. **Probability, Random Variables and Stochastic Processes**. New York: Mac Graw-Hill, 1965.

[PRE66] PREWITT, J. M. S.; MENDELSOHN, M. L. The Analysis of Cell Images. **Annals of the New York Academy of Sciences**, New York, v. 128, 1966, p.1035-1053.

[REI88] REIS, R. A. L. et al. An Efficient Design Methodology for Standard Cells Circuits. In: IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS, 1988, Helsinki. **Proceedings...** Helsinki : Piscataway - IEEE, 1988. p. 1213 - 1216.

[REI89] REIS, A. I.; MORAES, F. G.; REIS, R. A. L. Modem - Desenvolvimento de um ASIC para Modens de Banda Base. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE MICROELETRÔNICA, 4., 1989, Porto Alegre. **Anais...** Porto Alegre: Sociedade Brasileira de Microeletrônica, 1989. v. 2, p. 617-626.

[REI90] REIS, A. I.; VARGAS, F. L.; MORAES, F. G. et al. Teste, Depuração e Resultados de um Circuito Codificador/ Decodificador para Modems. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE MICROELETRÔNICA, 5., 1990, Campinas. **Anais...** Campinas: Sociedade Brasileira de Microeletrônica, 1990. p. 266-278.

[RUS91] RUSSELL, J. D.; VARGAS, F. L.; COURTOIS, B. E - Beam Testing Using Multiple Adjacent Image Processing for Prototype Validation. In: EUROPEAN CONFERENCE ON ELECTRON AND OPTICAL BEAM TESTING OF INTEGRATED CIRCUITS, 3., 1991, Como, Italy. **Proceedings...** Como: Microelectronic Engeneering Journal, 1991.

[TAM86] TAMAMA, T.; KUJI, N. Integrating an Electron Beam System into VLSI Fault Diagnosis. **IEEE Design & Test**, New York, n. 8, p.23-29, August, 1986.

[ULL74] ULLMAN, J. R. Binarization Using Associative Addressing. **Pattern Recognition**, New York, n. 6, p.127-135, 1974.

[VAR90] VARGAS, F. L. **Microscópio Eletrônico de Varredura como Ferramenta de Depuração e Teste de Circuitos Integrados**. Porto Alegre: CPGCC - UFRGS, 1990. 71 p. Trabalho Individual.

[VAR91] VARGAS, F. L. **Prototype Validation and Failure Analysis in E - Beam Testing: Theory and Practical Experiments**. Grenoble: TIM3/INPG, 1991. Research Report.

[WES78] WESZKA, J. S. A Survey of Threshold Selection Techniques. **Computer Vision, Graphics, and Image Processing**, New York, n. 7, p. 259-265, 1978.

[WOL86] WOLFGANG, E. Electron beam testing. **Microelectronic Engineering**, North - Holland, n. 4, p. 77-106, 1986.

[ZUN83] ZUNIGA, O. A.; HARALICK, R. M. Corner Detection Using the Facet Model. **IEEE Computer Vision and Pattern Recognition**, New York, p. 30-37, 1983.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
CURSO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

"Validação de Protótipo e Análise de Falhas no Teste com Feixe de Elétrons: Um Estudo Visando a sua Automação".

Dissertação apresentada aos Srs.

Profa. Dra. Ingrid E. S. Jansch Pôrto

Prof. Dr. Mario Lúcio Côrtes - CPqD/TELEBRÁS

Prof. Dr. Ricardo A. da L. Reis

Prof. Tiaraju Vasconcellos Wagner

Visto e permitida a impressão

Porto Alegre, 02 / 10 / 91

Prof. Dr. Ricardo A. da L. Reis
Orientador

Prof. Dr. Ricardo Augusto da L. Reis
Coordenador do Curso de Pós-Graduação
em Ciência da Computação

ESR/CCB \DIVCPGCC\ASSINA.BAN