UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ARQUITETURAS E ALGORITMOS PARA UM ANALISADOR DE INTERCONEXÕES

por

OSMAR BRUNE

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

Prof. Tiaraju Vasconcellos Wagner
Orientador

Porto Alegre, setembro de 1988.

CATALOGAÇÃO NA FONTE

BRUNE, Osmar

Arquiteturas e Algoritmos para um Analisador de Interconexões.

Porto Alegre, PGCC da UFRGS, 1988.

1 v.

Diss. (mestr. ?? ??) UFRGS - CPGCC

Porto Alegre, BR-RS, 1988.

Dissertação:

AGRADECIMENTOS

Agradeço aos meus pais, que sempre me incentivaram a seguir na direção que escolhi; ao Tiaraju, pela dedicação com que auxilia e orienta a mim, e a todos que o procuram; à Hildegard, pelo carinho e por mostrar-me que vida é movimento; ao amigo Rui, pelas excelentes discussões, e a tantos outros que me acompanharam e tornaram mais interessante minha viagem neste planeta.

SUMÁRIO

GLOSSÁRIO

- Resistência

Resistência elétrica, cuja unidade de medida é o "ohm".

- Número, endereço ou índice de um ponto de teste

Aquele que indentifica o ponto de teste, a fim de diferenciá-lo dos demais ou de selecioná-lo para um determinado objetivo.

- Número, endereço ou índice de uma rede

Aquele que identifica a rede, a fim de diferenciá-la das demais ou de selecioná-la para um determinado objetivo. Este número coincide com o número do ponto de teste identificador da rede.

- i-ésima rede

Rede de ordem i, dentro da cadeia ordenada de redes na descrição de um padrão ou UST. A ordem é crescente. Não confundir com a rede cujo endereço é i.

- i-ésimo ponto de teste

Ponto de ordem i, dentro da cadeia ordenada de pontos na descrição de uma rede. A ordem é crescente. Não confundir com o ponto cujo endereço é i.

LISTA DE CONVENÇÕES

- Uma abreviatura, função matemática ou símbolo especial não definido na sua lista correspondente, será definido no contexto próximo de sua utilização.

LISTA DE ABREVIATURAS

- AI Analisador de interconexões
- BASTIDOR Bastidor de 1024 pontos
- BS Banco de sensores
- BUFF_BAST Buffer de bastidor
- C Condução
- DFD Diagrama de fluxo de dados
- DIG_LEIT Digitalmente legível
- DIG_PRG Digitalmente programável
- ENDX Bloco de endereçamento X
- ENDY Bloco de endereçamento Y
- ERR_CD Imprecisão da medida no teste de condução
- ERR_IS Imprecisão da medida no teste de isolação
- FALSE Valor booleano "falso"
- FORMATO Formato da listagem de erros
- I Isolação
- IBM/PC Microcomputador da linha IBM/PC
- LIM_CORR Limite de corrente
- LIM_ERR_COND_REDE Limite de erros de condução por rede
- LIM_ERROS_COND Limite de erros de condução
- LIM_ERROS_ISOL Limite de erros de isolação
- LIM_INF Ponto limite inferior
- LIM_SUP Ponto limite superior
- MAX_ACEIT_RCD Máxima resistência de condução aceitável

- MAX_TIP_RCD Máxima resistência de condução típica
- MCEM Módulo Central de Endereçamento e Medidas
- MIN_ACEIT_RIS Mínima resistência de isolação aceitável
- MIN_TIP_RIS Mínima resistência de isolação típica
- MUX Multiplexador de sensores
- NC Não-condução
- NI Não-isolação
- NULL Elemento não existente, tal como fim de uma lista encadeada
- OFF Aberta (chave), ou valor booleano "falso"
- ON Fechada (chave), ou valor booleano "verdadeiro"
- OPERADOR Operador do AI
- OUTPT Dispositivo de saída de listagens de erros
- PADRÃO Padrão em teste
- PAD_CRG Padrão de carga
- PAD_MEM Padrão em memória
- PLACA Placa de 64 pontos de teste
- PONTO Ponto de teste
- RCD Resistência de Condução
- RIS Resistência de Isolação
- T_CIC_CD Tempo de ciclo do teste de condução
- T_CIC_IS Tempo de ciclo do teste de isolação
- TCOND Teste de condução
- TINT Teste de interconexões

- TISOL Teste de isolação

- TRUE Valor booleano "verdadeiro"

- UST Unidade sob teste

- VCD Tensão de condução

- VIS Tensão de isolação

LISTA DE FUNÇÕES MATEMÁTICAS

- int(z)

Parte inteira de z. É o próprio z, quando este for um inteiro, ou o inteiro imediatamente inferior a z, quando este for fracionário.

- rds(z)

Arredondamento superior de z. É o próprio z, quando este for um inteiro, ou o inteiro imediatamente superior a z, quando este for fracionário.

- dec(vb)

Conversão decimal do vetor binário vb.

- bin(vd)

Conversão binária de um vetor decimal vd.

- vb1 & vb2

Operação "e", bit a bit, entre 2 vetores binários, e que resulta num terceiro vetor binário.

- vb1 | vb2

Operação "ou", bit a bit, entre 2 vetores binários, e que resulta num terceiro vetor binário.

- ~vb

Operação "não", sobre cada bit do vetor binário vb, resultando no vetor binário complemento de vb.

- !vl

Negação do valor booleano, ou lógico, vl.

- i * j

Produto de i por j.

- i / j

Divisão de i por j.

- i - j

Subtração de números ou de conjuntos, dependendo do tipo dos símbolos i e j no contexto.

- i + j

Adição de números ou de conjuntos, dependendo do tipo dos símbolos i e j no contexto.

- [k,z]

Intervalo fechado, entre k e z. Se aplicado sobre valores inteiros, simboliza o conjunto finito de

$$z - k + 1$$

valores inteiros:

$$\{k,k+1,...,z-1,z\}.$$

- [k,z)

Intervalo aberto à direita, entre k e z. Se aplicado sobre valores inteiros, simboliza o conjunto finito de

$$z - k$$

valores inteiros:

$$\{k,k+1,...,z-2,z-1\}.$$

- (k,z]

Intervalo aberto à esquerda, entre k e z. Se aplicado sobre valores inteiros, simboliza o conjunto finito de

z - k

valores inteiros:

{k+1,k+2,...,z-1,z}.

- SOMA(i,k,z,f(i))

Somatório dos valores f(i), para todo valor inteiro de i no conjunto [k,z].

- PROD(i,k,z,f(i))

Produtório dos valores f(i), para todo valor inteiro de i no conjunto [k,z].

- C(z,2)

Combinações de z, 2 a 2. Pode ser avaliado por

z * (z - 1) / 2.

- log2(z)

Logaritmo na base 2 de z.

- L2(z)

Definida como:

rds(log2(z)).

- exp2(z)

2 elevado a potência z.

LISTA DE SÍMBOLOS ESPECIAIS

- n

Número de pontos de teste do AI. Nas análises de complexidade dos algoritmos, admite-se que todos os n pontos estão alocados na UST, para o teste de interconexões.

- nr

Número de redes, em um PADRÃO ou UST.

- nr(i)

Número de redes formadas por i pontos, em um PADRÃO ou UST, com i = 1,2,...,n.

- np(i)

Número de pontos envolvidos em redes de i pontos, em um PADRÃO ou UST, com i = 1,2,...,n.

- s

Número de sensores de corrente empregados.

- V_TESTE

Tensão programável aplicada durante as medidas.

- S(i)

Designação de um dos "s" sensores.

- S

Conjunto dos s sensores.

S = {S(i) ; i = 0,1,2,...,s-1}.

- P(i)

Designação de um ponto de teste cujo número, endereço ou índice é i, com i = 0,1,2,...,n-1.

- P

Conjunto dos n pontos de teste.

$$P = \{P(i) ; i = 0,1,2,...,n-1\}.$$

- X(i)

Designação da chave do conjunto X que corresponde ao ponto P(i).

- X

Conjunto das n chaves que conectam cada ponto de teste do conjunto P a um sensor do conjunto S, através de chaves multiplexadoras do conjunto M, a ser definido adiante.

$$X = \{X(i) ; i = 0,1,2,...,n-1\}.$$

- Y(i)

Designação da chave do conjunto Y que corresponde ao ponto P(i).

- Y

Conjunto das n chaves que conectam cada ponto de teste do conjunto P à tensao V_TESTE:

$$Y = \{Y(i) ; i = 0,1,2,...,n-1\}.$$

- M(i,j)

Chave multiplexadora que conecta a saída da chave X(i) à entrada do sensor S(j).

- M

Conjunto das n*s chaves multiplexadoras que conectam a saída das chaves de X aos sensores de S.

$M = \{M(i,j) ; i = 0,1,2,...,n-1; j = 0,1,2,...,s-1\}$.

- I_S(i)

Corrente medida no sensor S(i).

- I_S

Conjunto de correntes medidas em todos os s sensores.

$I\_S = \{I\_S(i) ; i = 0,1,2,...,s-1\}$.

- x

Especifica o sub-conjunto de chaves de X no estado ON (fechadas). Em outras circunstâncias, pode especificar um sub-conjunto de X que terá seu estado alterado, para ON ou para OFF.

- y

Especifica o sub-conjunto de chaves de Y no estado ON (fechadas). Em outras circunstâncias, pode especificar um sub-conjunto de Y que terá seu estado alterado, para ON ou para OFF.

- ccd

Especifica a complexidade ou número de ciclos com medida de resistência do TCOND.

- cis

Especifica a complexidade ou número de ciclos com medida de resistência do TISOL.

LISTA DE PROPRIEDADES SOBRE SÍMBOLOS ESPECIAIS

- np(i) = nr(i) * i;

- nr = SOMA(i,1,n,nr(i))

- n = SOMA(i,1,n,np(i)) = SOMA(i,1,n,nr(i) * i)

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Este trabalho aborda um estudo de algoritmos e arquiteturas de um Analisador de Interconexões. Várias alternativas possíveis são discutidas e uma análise de custo e desempenho é feita. Alguns dos algoritmos e arquiteturas propostos parecem ser novos se comparados à literatura publicada. Um dos algoritmos foi completamente simulado para auxiliar a análise de desempenho e para demonstrar a interface com o usuário em uma aplicação comercial.

ABSTRACT

This work deals with a study of algorithms and architectures of an Interconnection Analyzer. Several possible alternatives are discussed and an analysis of cost and performance is carried out. Some of the proposed algorithms and architectures seems to be new when compared to the published literature. One of the algorithms was fully simulated to help the performance analysis and to demonstrate the user interface in a commercial application.

1 - INTRODUÇÃO

1.1 - Definição de Analisador de Interconexões

Um Analisador de Interconexões é um equipamento automatizado de testes, cuja função principal é o teste de interconexões.

O teste de interconexões visa detectar dois tipos de defeitos em unidades sob teste:

a) circuitos-abertos, entre pontos de teste que deveriam estar conectados.

b) curto-circuitos, ou fugas, entre pontos de teste que não deveriam estar conectados.

A unidade sob teste é comparada com um padrão, descrito em memória, e os defeitos são reportados.

1.2 - Requisitos de um Analisador de Interconexões

Alguns dos requisitos básicos de um Analisador de Interconexões são:

a) o teste de interconexões deve ser executado rapidamente, consumindo, tipicamente, poucos segundos para dezenas de milhares de pontos de teste.

b) o teste de interconexões deve reportar erros compactamente, que sejam facilmente compreendidos e localizados, e deve possuir uma cobertura aceitável.

c) o custo do Analisador de Interconexões deve ser compatível com o dos equipamentos similares no mercado.

d) a confiabilidade do equipamento, bem como a facilidade de manutenção, devem ser aceitáveis, oferecendo auto-diagnósticos para isolar possíveis defeitos do Analisador de Interconexões.

e) deve existir um conjunto poderoso de funções auxiliares,

permitindo auto-programação de padrões, nomeação simbólica de pontos de teste, manipulação de arquivos e outras.

f) os parâmetros e características elétricas do Analisador de Interconexões devem ser compatíveis com a necessidade de várias classes de usuários, e por isso vários parâmetros devem ser programáveis.

1.3 - Unidades Sob Teste Comumente Testadas por um AI

As USTs testadas por um AI não devem ter componentes montados, uma vez que o objetivo é apenas testar, rapidamente, sub-sistemas eletro-eletrônicos que apenas possuam interconexões de baixa resistência elétrica entre determinados pontos de teste, e que possuam interconexões de alta resistência entre outros pontos de teste.

Como exemplos de tais USTs, tipicamente testadas por AIs, pode-se citar:

a) placas nuas de circuito impresso.

b) barramentos.

c) bastidores de chaveamento na indústria telefônica.

d) montagens "wire-wrap".

e) cablagens.

f) conectores.

g) outras.

1.4 - Vantagens da Aplicação do Teste de Interconexões

O teste de um sub-sistema de interconexões, sobre uma amostragem suficientemente alta de pontos de teste, deve ser feito antes da incorporação deste sub-sistema em um sistema maior.

Por exemplo, deve-se testar uma placa de

circuito-impresso antes de montar os componentes sobre ela.

O custo de reparo de um defeito torna-se maior à medida que o equipamento passa pelos estágios de produção e o defeito não é detectado.

No caso de placas de circuito impresso, alguns estudos indicaram que a identificação e reparo de defeitos em placas nuas pode custar em torno de 1% do que custaria após o equipamento ter sido despachado para o campo /MAY 84/.

Com a crescente complexidade dos sistemas eletrônicos, surgiu uma verdadeira "explosão de interconexões" /ERI 79/. A presença de defeitos nestas interconexões, com o seu aumento em número e densidade, também aumentou significativamente, o que justifica a necessidade do teste de interconexões.

O teste de interconexões, em sistemas com mais do que algumas dezenas de pontos de teste, seria totalmente lento e cansativo, além de inconfiável, se feito manualmente. O Analisador de Interconexões o automatiza e o acelera milhares ou milhões de vezes.

1.5 - Motivação da Dissertação

Vários foram os motivos que levaram o mestrando a escolher este assunto para sua dissertação, entre os quais pode-se citar:

a) experiência anterior do mestrando, no projeto de um Analisador de Interconexões, junto à empresa Ábaco Sistemas de Informação Ltda., de Porto Alegre, RS.

b) desejo de explorar o campo de equipamentos automáticos de teste, pouco desenvolvido no país, desenvolvendo o início do projeto de um Analisador de Interconexões competitivo, em custo e performance, com os equipamentos do mercado externo.

c) existe um mercado promissor, possibilitando a industrialização e comercialização de Analisadores de Interconexões.

d) desejo de otimizar os resultados obtidos na experiência de projeto citada em "a)", especialmente reduzindo o tempo de teste. Outras características também foram consideradas, tais como a cobertura, a estrutura da listagem de erros, o conjunto de funções auxiliares, o conjunto de parâmetros elétricos e sua programabilidade, e a confiabilidade do AI.

1.6 - Escopo da Dissertação

Esta dissertação visa estudar arquiteturas e algoritmos tecnicamente viáveis para implementar Analisadores de Interconexões que possuam um tempo de teste de interconexões aceitável, ou que possuam um custo aceitável, ou que possuam ambas as características.

As arquiteturas serão detalhadas apenas até o nível de blocos lógicos, cuja função será especificada comportamentalmente, num nível de abstração relativamente alto. Embora tenham sido desenvolvidos esquemas até o nível lógico, estes não se encontram anexos. Estes esquemas foram utilizados para avaliar a viabilidade técnica e econômica com uma precisão maior.

Os algoritmos também serão descritos em alto nível de abstração, de tal maneira que seja possível compreender os seus funcionamentos e avaliar suas complexidades.

A simulação foi utilizada para um dos algoritmos, para permitir a avaliação da sua correção e complexidade exata. Escolheu-se o algoritmo que oferecia maiores dificuldades para a análise formal de complexidade, o que não acontece para os demais algoritmos, que foram analisados formalmente.

2 - CONCEITOS, PARÂMETROS E OBSERVAÇÕES PRELIMINARES

Grande parte dos conceitos, parâmetros e observações contidos neste capítulo serão utilizados nos próximos capítulos. Outra parte discute aspectos que não serão analisados com maior profundidade nesta dissertação, por escaparem do seu escopo central.

2.1 - Conceitos Primários

2.1.1 - Unidade sob Teste

É o equipamento eletro-eletrônico submetido ao teste de interconexões pelo AI.

2.1.2 - Pontos de Teste

Nodos elétricos escolhidos na UST, entre os quais deseja-se verificar a correção das interconexões. São identificados pelo seu número, endereço ou índice e, opcionalmente, por um nome simbólico atribuído pelo usuário. O conjunto de pontos de teste escolhido deve ser suficiente para verificar todas as interconexões de interesse.

2.1.3 - Condução

Dois pontos de teste estão ligados por uma interconexão do tipo condução, ou Ç se a resistência elétrica entre eles for menor do que uma resistência limite chamada RCD, tipicamente valendo poucos ohms.

2.1.4 - Isolação

Dois pontos de teste, ou duas redes, estão ligados por uma interconexão do tipo isolação, ou I, se a resistência elétrica entre eles for maior do que uma resistência limite chamada RIS, tipicamente valendo vários Mega-ohms.

2.1.5 - Não-Condução

Dois pontos de teste estão ligados por uma interconexão do tipo não-condução, ou NÇ se não estiverem ligados por uma interconexão do tipo C.

2.1.6 - Não-Isolação

Dois pontos de teste estão ligados por uma interconexão do tipo não-isolação, ou NI, se não estiverem ligados por uma interconexão do tipo I.

2.1.7 - Rede

Conjunto de pontos de teste P(i) ligados por interconexões de baixa resistência. Um dos pontos do conjunto, escolhido como ponto identificador da rede, deve estar ligado por interconexões do tipo C com qualquer outro ponto deste conjunto.

A figura 2.1 exibe uma UST com 8 pontos de teste, e 3 redes. O número de cada ponto de teste aparece ao lado de sua representação. Interconexões representadas entre 2 pontos como um traço contínuo têm resistência nula, as não representadas têm resistência infinita, e as demais têm especificado o valor da resistência. Admite-se que

RIS >> RCD.

As 3 redes desta figura são formadas, respectivamente, pelos pontos:

- 0,1,2,3,5.
- 4.
- 6,7.

Os pontos identificadores destas redes são 0, 4 e 6.

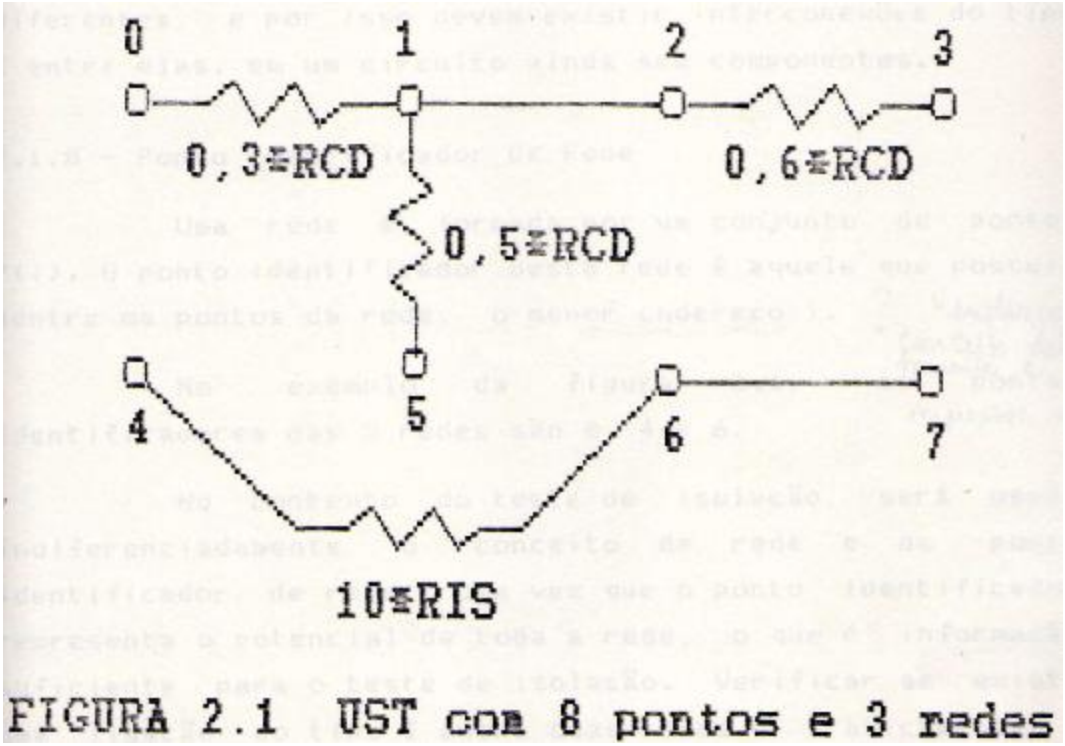

FIGURA 2.1 UST com 8 pontos e 3 redes

Pode-se afirmar que a resistência elétrica entre 2 pontos quaisquer de uma rede é menor do que

2 * RCD,

pois a resistência entre o ponto identificador da rede e qualquer outro ponto da rede é menor do que RCD. Isto porque a resistência entre dois pontos, B e Ç é menor do que a soma da resistência entre os pontos A e B com a resistência entre os pontos A e C. Imagine-se que A seja o ponto identificador da rede, e B e Ç dois outros pontos quaisquer da rede.

Uma rede representa, do ponto de vista elétrico, um conjunto de pontos de teste que terão aproximadamente o mesmo potencial elétrico quando o circuito estiver alimentado. Como exemplo de rede, podemos citar o conjunto de pontos de teste sobre uma mesma trilha de circuito impresso.

Redes distintas poderão ter potenciais elétricos diferentes, e por isso devem existir interconexões do tipo I entre elas, em um circuito ainda sem componentes.

2.1.8 - Ponto Identificador de Rede

Uma rede é formada por um conjunto de pontos P(i). O ponto identificador desta rede é aquele que possui, dentre os pontos da rede, o menor endereço i.

No exemplo da figura 2.1, os pontos identificadores das 3 redes são 0, 4 e 6.

No contexto do teste de isolação, será usado indiferenciadamente o conceito de rede e de ponto identificador de rede, uma vez que o ponto identificador representa o potencial de toda a rede, o que é informação suficiente para o teste de isolação. Verificar se existe uma ligação do tipo I entre duas redes é praticamente o mesmo do que verificar se existe uma ligação do tipo I entre os dois pontos identificadores destas redes, desde que

$$RIS >> RCD.$$

2.1.9 - Rede Combinatória:

Conjunto de pontos de teste P(i) ligados, 2 a 2, por interconexões do tipo C.

Uma rede combinatória sempre é também uma rede, mas o contrário nem sempre é verdadeiro.

A figura 2.2 mostra uma UST formada por 2 redes de 4 pontos cada uma. A primeira rede, formada pelos pontos 0, 1, 4 e 5, não é uma rede combinatória, pois a resistência entre os pontos 4 e 5 é maior do que RCD. A segunda rede, formada pelos pontos 2, 3, 6 e 7, é uma rede combinatória.

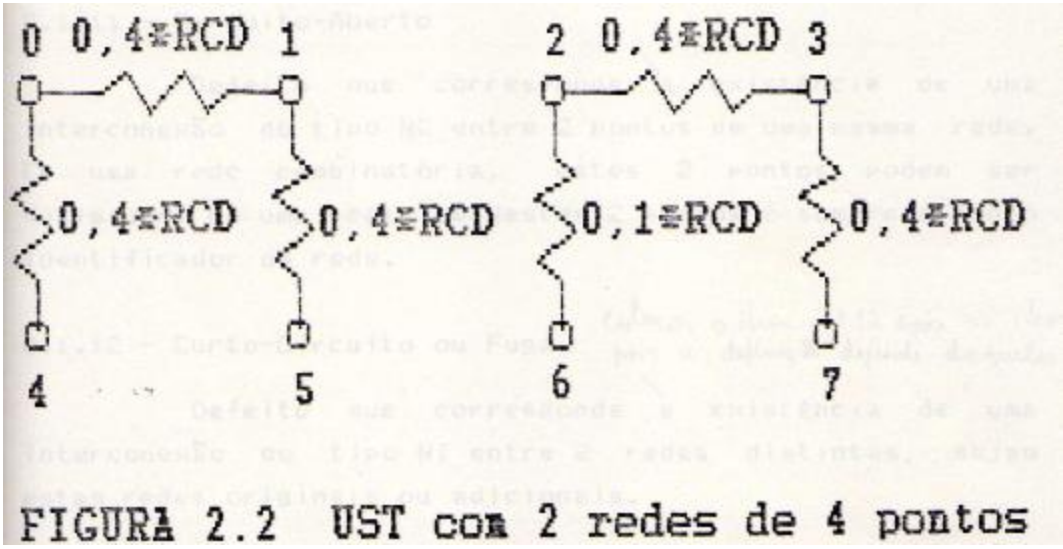

FIGURA 2.2 UST com 2 redes de 4 pontos

Em uma rede combinatória, assegura-se que a resistência entre quaisquer 2 pontos da rede é menor do que

RCD,

enquanto que numa rede assegura-se apenas que a resistência entre quaisquer 2 pontos da rede é menor do que

2 * RCD,

e apenas que a resistência entre o ponto identificador da rede e outro qualquer da rede será menor do que

RCD.

2.1.10 - Padrão

Trata-se de uma descrição, em arquivo ou memória, de uma UST considerada boa, afim de ser comparada, pelo teste de interconexões, com as USTs semelhantes que saem da linha de produção. Esta descrição é a lista das redes que formam a UST boa.

O padrão pode ser editado ou então obtido automaticamente, conectando-se ao AI uma UST sabidamente

perfeita e realizando uma função auxiliar de autoprogramação.

2.1.11 - Circuito-Aberto

Defeito que corresponde a existência de uma interconexão do tipo NC entre 2 pontos de uma mesma rede. Em uma rede combinatória, estes 2 pontos podem ser quaisquer. Em uma rede, um destes 2 pontos é sempre o ponto identificador da rede.

2.1.12 - Curto-Circuito ou Fuga

Defeito que corresponde a existência de uma interconexão do tipo NI entre 2 redes distintas, sejam estas redes originais ou adicionais.

2.1.13 - Rede Original

Trata-se de uma rede que encontra-se descrita no padrão.

2.1.14 - Rede Adicional

Trata-se de uma rede que se originou da ruptura de uma rede original, devido a um ou mais defeitos do tipo circuito-aberto na rede original.

Cada defeito do tipo circuito-aberto normalmente origina uma rede adicional.

A figura 2.3 exibe uma rede original, formada pelos pontos 0, 1, 2, 3, 4, 5, 6 e 7.

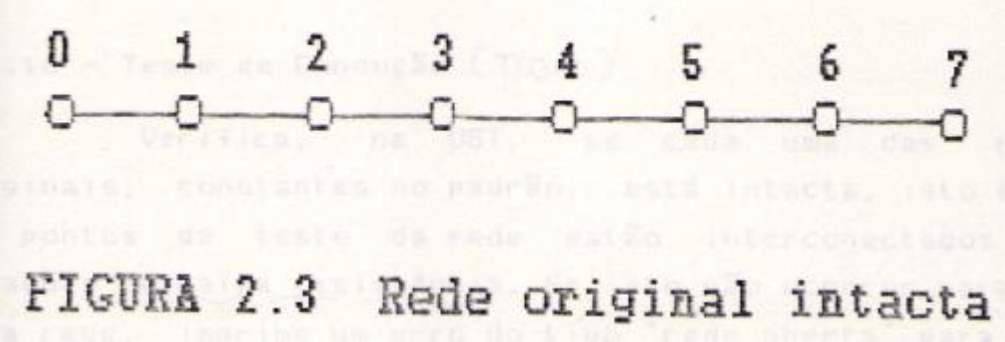

FIGURA 2.3 Rede original intacta

A figura 2.4 exibe a rede da figura 2.3, mais duas redes adicionais geradas por dois defeitos do tipo circuito-aberto. A nova rede original, que ainda possui o mesmo ponto identificador, é formada pelos pontos 0, 1 e 2. As duas redes adicionais, cujos pontos identificadores são 3 e 6, são formadas, respectivamente, pelos pontos 3, 4 e 5, e pelos pontos 6 e 7.

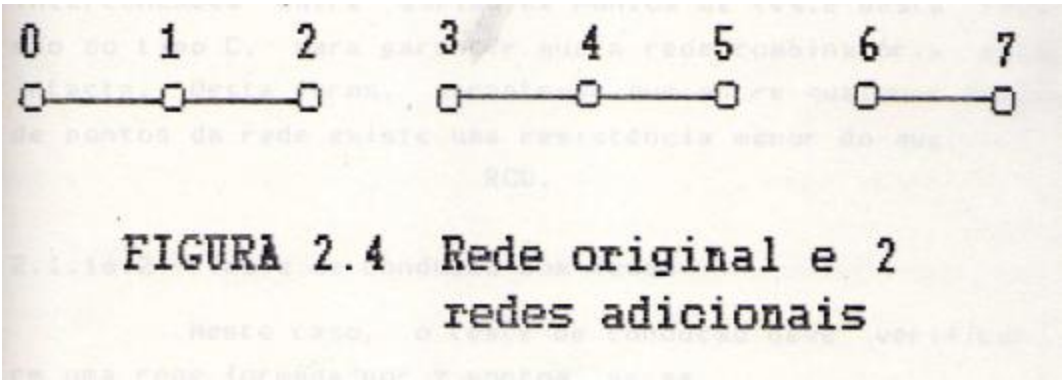

FIGURA 2.4 Rede original e 2 redes adicionais

2.1.15 - Teste de Interconexões

É a principal e mais executada função do AI. Compara-se um padrão selecionado com a UST conectada ao AI, e são listados erros do tipo "rede aberta", gerados devido a defeitos do tipo circuito-aberto, e do tipo "fuga entre redes", gerado devido a defeitos do tipo curto-circuito. O teste de interconexões é composto pela seqüência de dois

testes: condução e isolação.

2.1.16 - Teste de Condução

Verifica, na UST, se cada uma das redes originais, constantes no padrão, está intacta, isto é, se os pontos de teste da rede estão interconectados por ligações de baixa resistência. Se isto não ocorrer para uma dada rede, imprime um erro do tipo "rede aberta" para esta rede.

O TCOND passa uma lista de redes atualizada, incluindo possíveis redes adicionais, para o teste de isolação.

2.1.16.1 - Teste de Condução com Redes Combinatórias

Neste caso, o teste de condução deve verificar, em uma rede combinatória formada por z pontos, se as

$$C(z,2)$$

interconexões entre duplas de pontos de teste desta rede são do tipo Ç para garantir que a rede combinatória está intacta. Desta forma, garante-se que entre qualquer dupla de pontos da rede existe uma resistência menor do que

$$RCD.$$

2.1.16.2 - Teste de Condução com Redes

Neste caso, o teste de condução deve verificar, em uma rede formada por z pontos, se as

$$z - 1$$

interconexões entre o ponto identificador da rede e os demais

$$z - 1$$

pontos desta rede são do tipo Ç para garantir que a rede está intacta. Desta forma, garante-se que entre qualquer dupla de pontos da rede teremos uma resistência menor do que

2 * RCD.

Embora o teste de condução com redes combinatórias garanta um resultado melhor, sua complexidade inerente é

z / 2

vezes maior do que para o teste de condução com redes, o que se reflete no tempo de teste, principalmente para grandes valores de z.

As desvantagens do teste de condução com redes podem ser praticamente removidas se o parâmetro RCD for programado conscientemente, em função dos parâmetros ERR_CD, MAX_TIP_RCD e MAX_ACEIT_RCD, conforme será discutido na seção 2.5, onde também serão analisados os efeitos da imprecisão da medida sobre a cobertura do teste de condução.

Por estes motivos, todos os algoritmos do teste de condução serão feitos utilizando o conceito de redes, e não o de redes combinatórias.

2.1.17 - Teste de Isolação

Verifica, na UST, se cada uma das redes passadas pelo TCOND, seja original ou adicional, está isolada de todas as demais redes. Se isto não ocorrer para alguma rede, imprime um erro do tipo "fuga entre redes" para esta rede.

O efeito da imprecisão da medida sobre a cobertura do teste de isolação é analisado na seção 2.6.

2.2 - Parâmetros Programáveis

Abaixo serão listados parâmetros elétricos e de controle para o teste de interconexões, os quais poderão ser ajustados pelo usuário, interagindo com o sistema.

2.2.1 - Padrão em Memória

PAD_MEM é o padrão carregado na memória, e que pode ser utilizado para o teste de interconexões.

2.2.2 - Padrão de Carga

PAD_CARGA é o padrão que será automaticamente carregado para a memória (PAD_MEM) sempre que o equipamento for energizado.

2.2.3 - Pontos Limite Inferior e Superior

LIM_INF e LIM_SUP, definem o intervalo

[LIM_INF,LIM_SUP],

contido no intervalo

[0,n - 1],

que é o conjunto total dos pontos de teste disponíveis. O objetivo é restringir o escopo do teste de interconexões, para diminuir o tempo do teste no caso de a UST não utilizar todos os pontos de teste disponíveis. O número de pontos de teste usado será

LIM_SUP - LIM_INF + 1,

sendo que a velocidade será ligeiramente maior se

LIM_INF = 0.

Para efeito de análise de complexidade dos algoritmos, no capítulo 5, assume-se que todos os n pontos são utilizados.

2.2.4 - Resistência de Isolação

RIS é a resistência limite entre a isolação e a não-isolação.

Seu valor pode ser calculado a partir dos parâmetros MIN_TIP_RIS e MIN_ACEIT_RIS, ou pode ser fornecido diretamente pelo usuário. Parece mais razoável e segura a primeira opção, de forma que, provavelmente, RIS não será programável pelo usuário, mas sim, calculado e

usado internamente pelo AI. Se a segunda opção for escolhida, serão desnecessários os parâmetros MIN_TIP_RIS e MIN_ACEIT_RIS. O cálculo interno de RIS será discutido na seção 2.6.

2.2.5 - Mínima Resistência de Isolação Aceitável

MIN_ACEIT_RIS é a mínima resistência aceitável entre duas redes distintas para que uma UST possa ser considerada boa, e para que o sistema que a incorpora funcione corretamente. O objetivo deste parâmetro é evitar que fugas reais sejam ignoradas.

2.2.6 - Mínima Resistência de Isolação Típica

MIN_TIP_RIS é a mínima resistência típica que costuma ocorrer entre duas redes distintas em uma UST boa. O objetivo deste parâmetro é evitar que fugas falsas sejam detectadas.

2.2.7 - Resistência de Condução

RCD é a resistência limite entre a condução e a não-condução.

Seu valor pode ser calculado a partir dos parâmetros MAX_TIP_RCD e MAX_ACEIT_RCD, ou pode ser fornecido diretamente pelo usuário. Parece mais razoável e segura a primeira opção, de forma que, provavelmente, RCD não será programável pelo usuário, mas sim, calculado e usado internamente pelo AI. Se a segunda opção for escolhida, serão desnecessários os parâmetros MAX_TIP_RCD e MAX_ACEIT_RCD. O cálculo interno de RCD será discutido na seção 2.5.

2.2.8 - Máxima Resistência de Condução Aceitável

MAX_ACEIT_RCD é a máxima resistência aceitável entre dois pontos de uma rede para que uma UST possa ser

considerada boa, e para que o sistema que a incorpora funcione corretamente. O objetivo deste parâmetro é evitar que circuitos-abertos reais sejam ignorados.

2.2.9 - Máxima Resistência de Condução Típica

MAX_TIP_RCD é a máxima resistência típica que costuma ocorrer entre dois pontos de uma rede em uma UST boa. O objetivo deste parâmetro é evitar que circuitos-abertos falsos sejam detectados.

2.2.10 - Tensão de Isolação

VIS é a tensão de teste aplicada durante o teste de isolação. Existe um compromisso entre os valores de VIS e RIS. Para grandes valores de RIS, normalmente são necessários grandes valores de VIS.

2.2.11 - Tensão de Condução

VCD é a tensão de teste aplicada durante o teste de condução. Existe um compromisso entre os valores de VCD e RCD. Para pequenos valores de RCD, normalmente são necessários pequenos valores de VCD.

2.2.12 - Tempo de Ciclo do Teste de Isolação

T_CIC_IS é o tempo de ciclo programado para o teste de isolação, gasto para a estabilização de uma medida de alta resistência. Depende das capacitâncias e indutâncias parasitas da UST e do interface AI/UST, bem como dos parâmetros LIM_CORR e VIS.

2.2.13 - Tempo de Ciclo do Teste de Condução

T_CIC_CD é o tempo de ciclo programado para o teste de condução, gasto para a estabilização de uma medida de baixa resistência. Depende das capacitâncias e indutâncias parasitas da UST e do interface AI/UST, bem

como dos parâmetros LIM_CORR e VCD.

2.2.14 - Limite de Corrente

LIM_CORR é o limite de corrente que pode ser aplicado sobre uma interconexão qualquer da UST, a fim de protegê-la de danos por sobrecorrente.

2.2.15 - Formato da Listagem de Erros

FORMATO seleciona uma das opções de listagem para os erros encontrados.

2.2.16 - Dispositivo de Saída da Listagem de Erros

OUTPT seleciona a unidade de saída dos erros.

2.2.17 - Limite de Erros de Isolação

LIM_ERROS_ISOL é o limite de erros listados pelo teste de isolação. Se tal limite for alcançado, o teste de isolação é abortado neste ponto.

2.2.18 - Limite de Erros de Condução

LIM_ERROS_COND é o limite de erros listados pelo teste de condução. Se tal limite for alcançado, o teste de condução não será abortado, pois o teste de isolação precisa conhecer todas as redes adicionais detectadas pelo teste de condução.

2.2.19 - Limite de Erros de Condução por Rede

LIM_ERR_COND_REDE é o limite de defeitos do tipo circuito-aberto que serão detectados em cada uma das redes, pelo teste de condução. Isto é feito para impedir listagens de erros e tempos enormes para o teste de condução, quando erros extremamente grosseiros ocorrerem. Se este limite for atingido em alguma rede, a resolução desta rede é abortada

no teste de condução, e a

LIM_ERR_COND_REDE-ésima

rede adicional detectada será marcada como "SUSPEITA" sempre que ocorrer nas listagens de erro. Isto porque podem existir outras redes adicionais não descobertas, admitidas como pertencentes a esta rede adicional. Além disso, fugas envolvendo estas redes adicionais ignoradas pelo teste de condução, não serão detectadas pelo teste de isolação. Isto não é grave, pois os erros cometidos para que tal ocorra serão bastante grosseiros, exigindo uma grande reforma da UST, seguida de outro teste.

2.2.20 - Operador do AI

OPERADOR é o nome do operador do AI.

2.3 - Aquisição de Dados

Um analisador de interconexões é composto de um sistema de aquisição de dados e de um computador que controla este sistema. Os dados descritivos da UST são comparados a um padrão residente na memória do computador, e as diferenças são listadas como erros.

Os dados são obtidos através da leitura de resistências elétricas. Estes dados, que simbolizam as interconexões, são binários, indicando apenas se a resistência lida entre 2 pontos ou 2 conjuntos de pontos é maior ou menor do que uma resistência de referência.

No caso do teste de condução esta resistência de referência é RCD, e a interconexão pode ser do tipo Ç se a resistência lida for menor do que RCD, ou NÇ em caso contrário.

No caso do teste de isolação esta resistência de referência é RIS, e a interconexão pode ser do tipo I, se a resistência lida for maior do que RIS, ou NI, em caso contrário.

2.3.1 - Inexistência de Informação Geométrica

Uma vez que os dados adquiridos são valores de resistências elétricas, não pode-se obter a descrição geométrica da UST, pois a uma única descrição de UST como lista de redes podem corresponder muitas topologias diferentes.

A figura 2.5 mostra um PADRÃO e uma UST que, se comparados, seriam iguais do ponto de vista de lista de redes, mesmo tendo uma topologia bem diferente. Não pode-se detectar, através dos dados adquiridos pelo AI, nenhuma diferença entre este PADRÃO e esta UST, pois do ponto de vista de resistências elétricas, não existe erro nas interconexões.

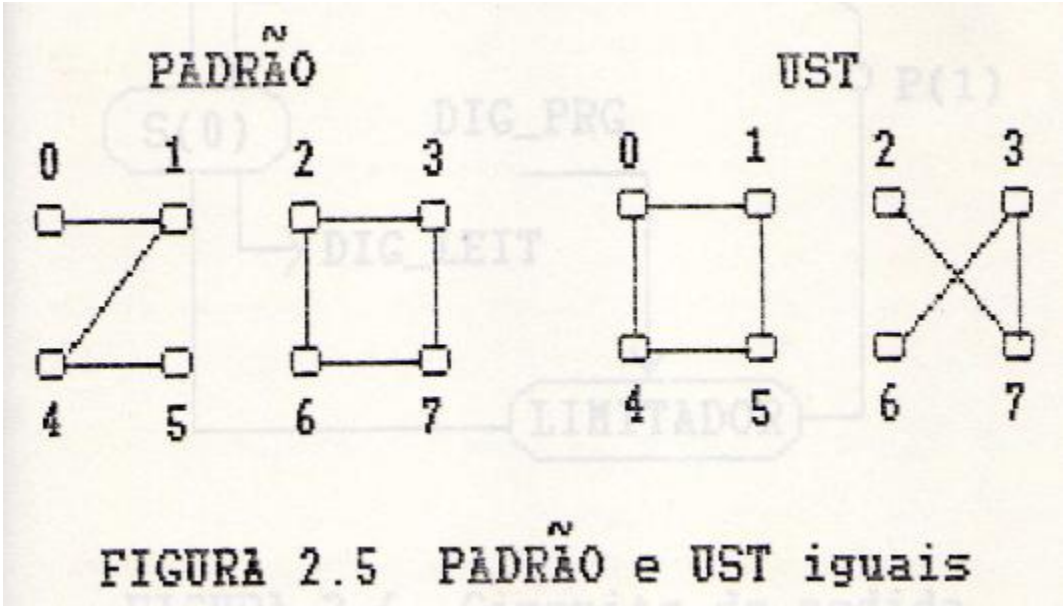

FIGURA 2.5 PADRÃO e UST iguais

O AI deveria adquirir outro tipo de dados para reconhecer informações geométricas. Um sistema de aquisição de dados com esta capacidade, no entanto, é extremamente caro e não é suficientemente genérico para ser utilizado em vários tipos de USTs. O propósito nesta dissertação é

analisar apenas as interconexões do ponto de vista de resistência elétrica.

2.3.2 - Medida de Resistência entre 2 Pontos Fixos

A resistência entre 2 pontos fixos, P(0) e P(1), pode ser medida utilizando o circuito da figura 2.6.

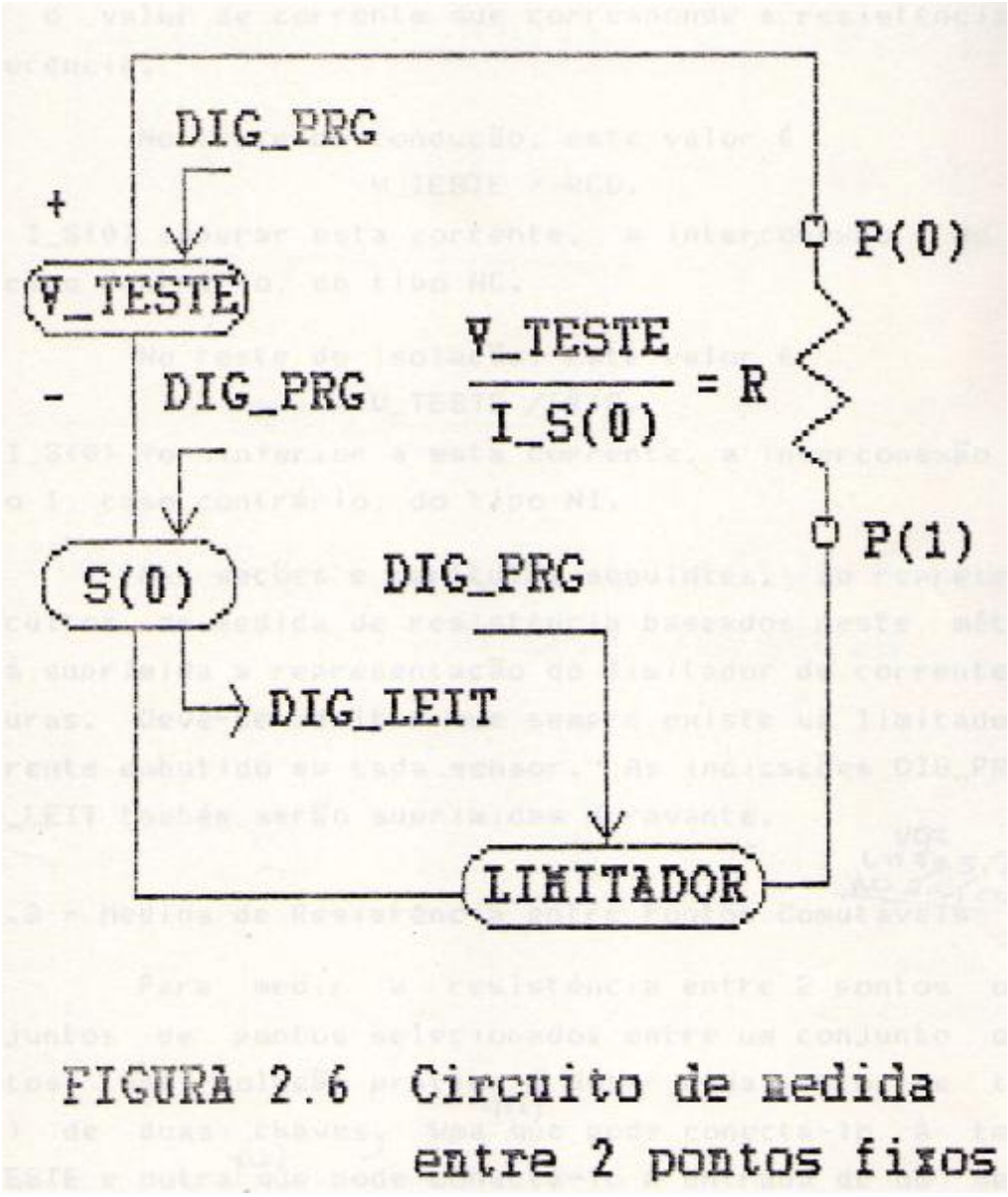

FIGURA 2.6 Circuito de medida entre 2 pontos fixos

Pode-se fazer uma analogia entre os pontos P(0) e P(1) e as ponteiras de um ohmímetro.

A tensão V_TESTE é programada digitalmente, bem como o limitador de corrente, para proteger a interconexão, simbolizada pela resistência R.

O sensor de corrente S(0) também é programado, com o valor de corrente que corresponde a resistência de referência.

No teste de condução, este valor é

V_TESTE / RCD.

Se I_S(0) superar esta corrente, a interconexão é do tipo Ç caso contrário, do tipo NC.

No teste de isolação, este valor é

V_TESTE / RIS.

Se I_S(0) for inferior a esta corrente, a interconexão é do tipo I, caso contrário, do tipo NI.

Nas seções e capítulos seguintes, ao representar circuitos de medida de resistência baseados neste método, será suprimida a representação do limitador de corrente nas figuras. Deve-se admitir que sempre existe um limitador de corrente embutido em cada sensor. As indicações DIG_PRG e DIG_LEIT também serão suprimidas doravante.

2.3.3 - Medida de Resistência entre Pontos Comutáveis

Para medir a resistência entre 2 pontos ou 2 conjuntos de pontos selecionados entre um conjunto de n pontos, uma solução prática é dotar cada ponto de teste P(i) de duas chaves. Uma que pode conectá-lo à tensão V_TESTE e outra que pode conectá-lo à entrada de um sensor de corrente. Deve-se evitar o fechamento simultâneo das duas chaves de um mesmo ponto, o que obviamente resultaria em curto-circuito e acionaria o limitador de corrente.

A figura 2.7 mostra um circuito que possibilita

este tipo de medidas.

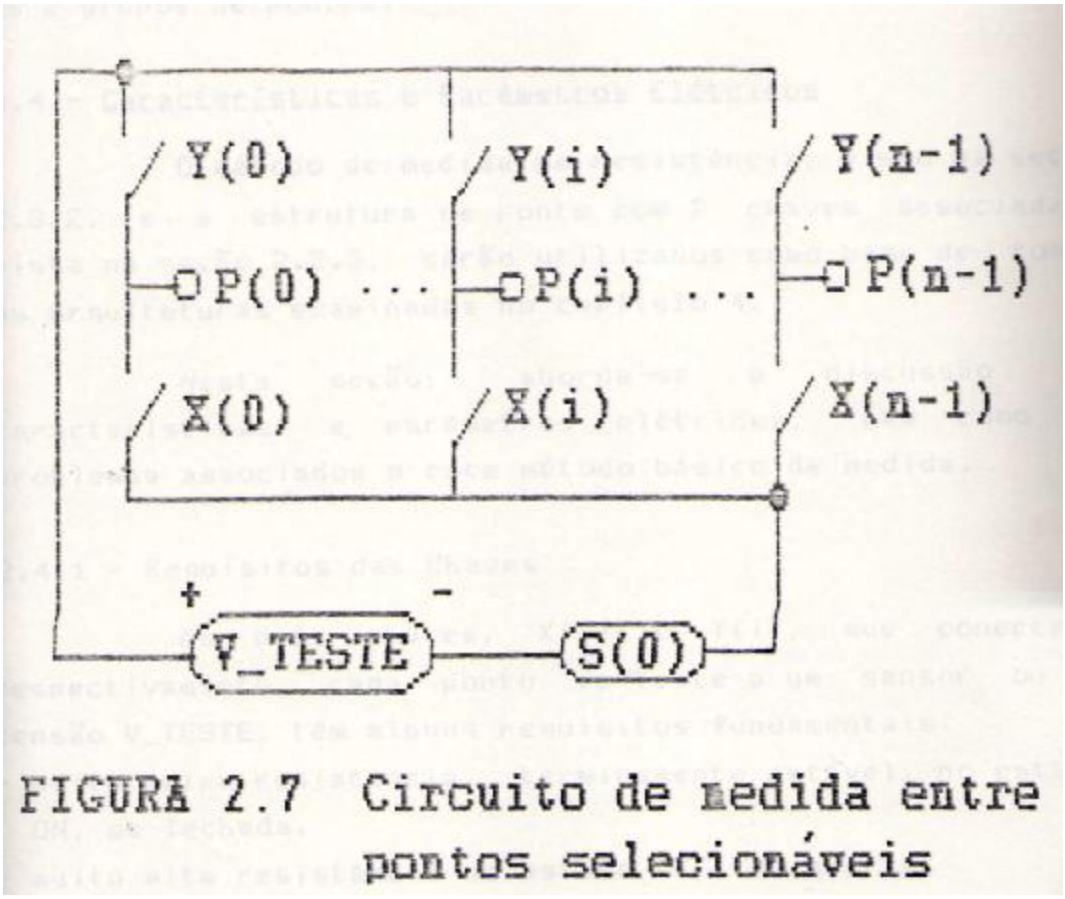

FIGURA 2.7 Circuito de medida entre pontos selecionáveis

Por exemplo, para determinar a resistência entre o ponto P(5) e o ponto P(34), pode-se fechar as chave X(5) e Y(34), e ler a corrente no sensor, após um tempo de estabilização da medida. Ou então poderia-se fechar as chaves X(34) e Y(5), uma vez que o valor de uma resistência independe do sentido da corrente.

Para medir a resistência entre dois sub-conjuntos de pontos, como por exemplo:

{P(3),P(6),P(8)} e {P(1),P(5),P(95)},

pode-se fechar as chaves X(3), X(6) e X(8), o que coloca em contato os pontos P(3), P(6) e P(8), e as chaves Y(1), Y(5) e Y(95), o que coloca em contato os pontos P(1), P(5) e P(95). A seguir, espera-se pela estabilização da medida, e

então mede-se a corrente e calcula-se a resistência entre os 2 grupos de pontos.

2.4 - Características e Parâmetros Elétricos

O método de medida de resistência, visto na seção 2.3.2, e a estrutura de ponto com 2 chaves associadas, vista na seção 2.3.3, serão utilizados como base de todas as arquiteturas examinadas no capítulo 4.

Nesta seção, aborda-se a discussão de características e parâmetros elétricos, bem como de problemas associados a este método básico de medida.

2.4.1 - Requisitos das Chaves

As duas chaves, X(i) e Y(i), que conectam, respectivamente, cada ponto de teste a um sensor ou à tensão V_TESTE, têm alguns requisitos fundamentais:

- muito baixa resistência, termicamente estável, no estado ON, ou fechada.
- muito alta resistência no estado OFF, ou aberta.
- boa isolação entre o circuito de comando da chave, que determina se a chave está aberta ou fechada, e o caminho de corrente, através da chave.
- baixa corrente de comando e comutação.
- tempo de comutação aceitável.
- boa tolerância a polarizações reversas impostas por cargas parasitas na UST.
- deve suportar altas tensões, dependendo do tipo de UST e de seus requisitos.
- deve suportar corrente razoáveis, dependendo do tipo de UST e de seus requisitos.
- encapsulamento compacto, se possível na forma de "arrays" de chaves.
- baixo custo, pois o custo da chave é um dos mais significativos no custo global do AI. Deve-se desenvolver diferentes tipos de placas, com diferentes tipos de

chaves, para aplicações que necessitem de chaves mais ou menos perfeitas, para assim reduzir o custo.

2.4.2 - Efeito e Limitação das Cargas Parasitas

Capacitâncias e indutâncias parasitas são características em qualquer sistema de interconexões. Os efeitos destas cargas parasitas, durante o teste de interconexões, são:

- aumento do tempo de estabilização de uma medida, a partir do momento em que as chaves são posicionadas para uma medida de resistência.
- possibilidade de aplicação de tensões reversas sobre chaves, ao comutar chaves ligadas a cargas parasitas enquanto estas tiverem energia armazenada.

Devido a estes motivos, deve existir um limite tolerável para estas cargas parasitas, para que não seja prejudicada a confiabilidade do teste de interconexões.

2.4.3 - Necessidade de Limitação de Corrente

Um limitador de corrente é colocado em série com cada sensor a fim de limitar a corrente aplicada sobre as interconexões da UST, e assim protegê-la de possíveis danos. Esse limitador também protege o próprio AI.

Além disso, é recomendável limitar a corrente fornecida pela fonte de tensão V_TESTE.

A corrente limite em cada sensor e a da fonte V_TESTE podem ou não ser programáveis, dependendo da flexibilidade exigida.

2.4.4 - Cálculo do Tempo de Estabilização da Medida

Para executar uma medida de resistência deve-se fechar determinadas chaves dentro dos conjuntos X e Y, aguardar o tempo de estabilização da medida e só então ler

a corrente nos sensores.

Admite-se que todos as medidas de um mesmo teste, de isolação ou de condução, terão um tempo de espera constante pela estabilização, chamado de tempo de ciclo do teste em questão (T_CIC_IS e T_CIC_CD). Para que os resultados de todas medidas sejam confiáveis, o tempo de ciclo programado deve ser maior do que o máximo tempo de estabilização das várias medidas necessárias para este teste. Normalmente utiliza-se estimativas com boa margem de segurança, que podem ser verificadas testando uma UST boa, e aumentando o tempo de ciclo até que não sejam mais listados erros falsos.

Desta forma, o tempo total de cada teste será múltiplo do tempo de ciclo selecionado, desprezando os tempos de software, endereçamento das chaves e leitura dos sensores. Na prática, pode-se diminuir estes últimos, enquanto que o tempo de estabilização da medida é uma característica estanque.

Uma solução é dotar os sensores de maior complexidade, para que indiquem quando a medida está estável, assim que isto acontecer. Desta maneira, cada medida gastaria um tempo levemente superior ao mínimo necessário para sua estabilização.

No escopo desta dissertação, no entanto, para facilitar as estimativas de tempo de teste, admite-se um tempo de ciclo constante, maior ou igual ao pior tempo de estabilização.

Os seguintes fatores influem sobre o tempo de estabilização de uma medida de resistência entre 2 conjuntos de pontos de teste:

- as cargas parasitas, que são a causa fundamental da demora de estabilização. Quanto maior forem as capacitâncias e indutâncias parasitas, maior será o tempo de estabilização.

- a tensão de teste V_TESTE.
- o limite de corrente LIM_CORR.
- a resistência que está sendo medida.

Capacitâncias parasitas costumam prejudicar mais o teste de isolação, pois neste normalmente é utilizada uma alta tensão V_TESTE (VIS). A figura 2.8 mostra a representação simplificada do circuito em uma medida do teste de isolação, com uma carga capacitiva parasita associada. A fim de estimar um tempo de ciclo seguro para o teste de isolação, admite-se que CMAX é o valor de uma carga capacitiva máxima existente entre quaisquer dois sub-conjuntos de pontos de teste.

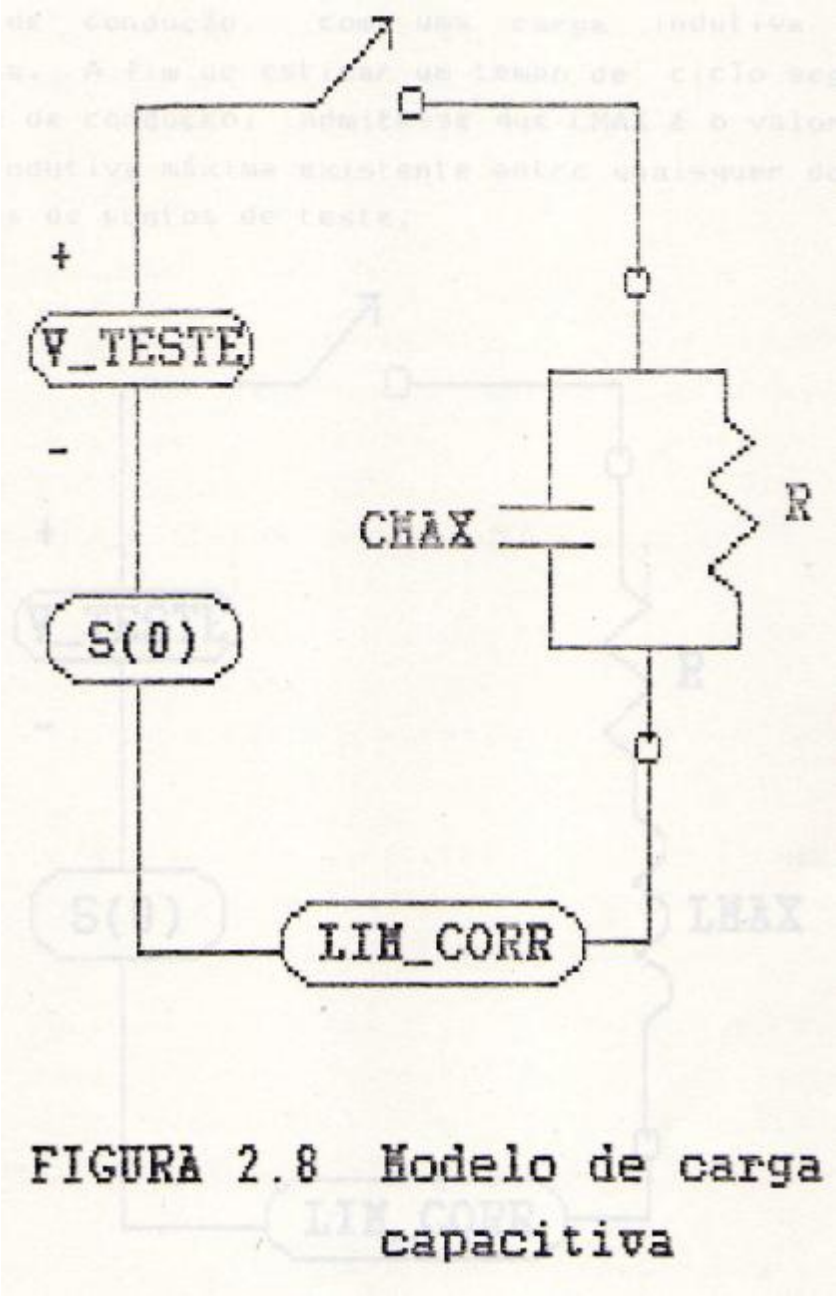

FIGURA 2.8 Modelo de carga capacitiva

Após fechar a chave, o tempo de carga de CMAX, ou tempo de estabilização máximo, será aproximadamente

CMAX * V_TESTE / LIM_CORR,

quando a resistência medida, R, for muito grande.

Indutâncias parasitas costumam prejudicar mais o teste de condução, pois neste normalmente é utilizada uma baixa tensão V_TESTE (VCD). A figura 2.9 mostra a

representação simplificada do circuito em uma medida do teste de condução, com uma carga indutiva parasita associada. A fim de estimar um tempo de ciclo seguro para o teste de condução, admite-se que LMAX é o valor de uma carga indutiva máxima existente entre quaisquer dois sub-conjuntos de pontos de teste.

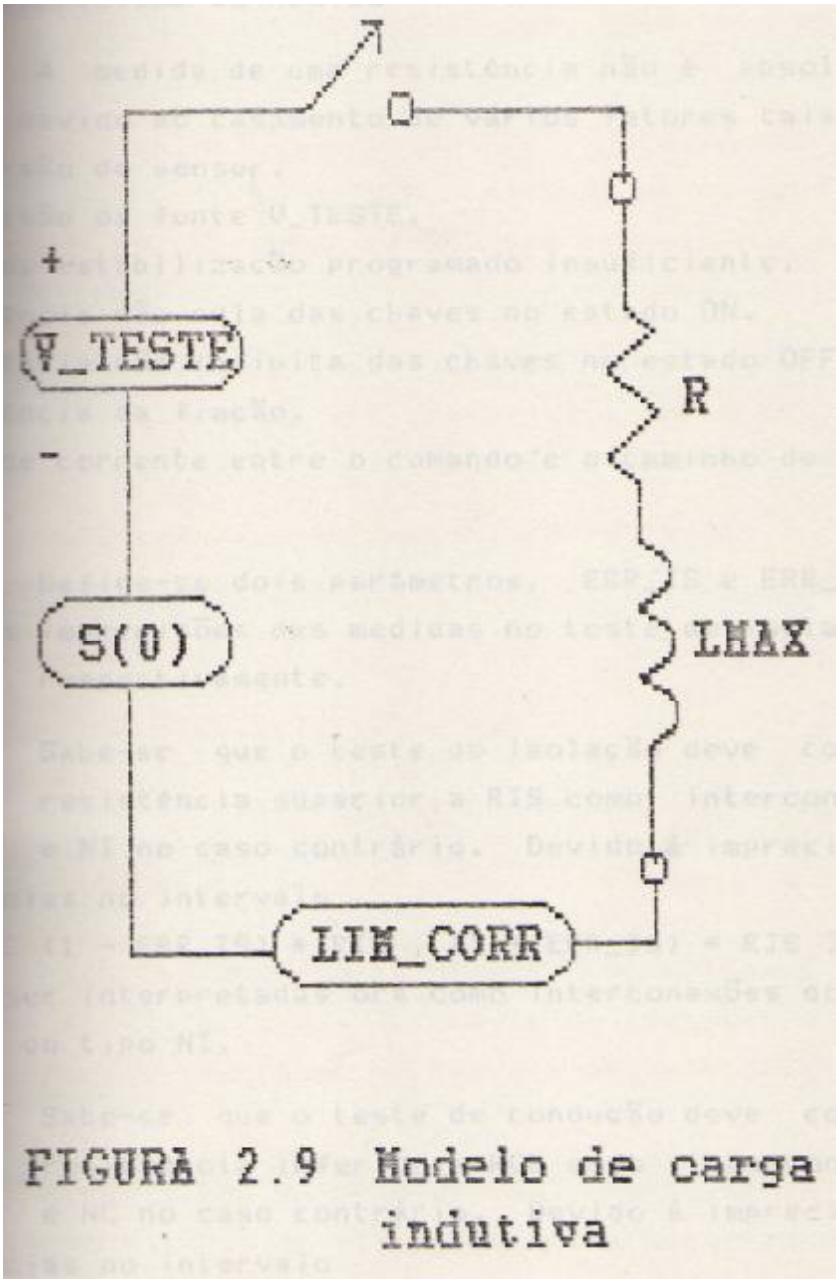

FIGURA 2.9 Modelo de carga indutiva

Após fechar a chave, o tempo de carga de LMAX, até que a corrente LIM_CORR seja atingida, ou o tempo de estabilização máximo, será aproximadamente

LMAX * LIM_CORR / V_TESTE,

quando a resistência medida, R, for muito pequena.

2.4.5 - Imprecisão da Medida

A medida de uma resistência não é absolutamente precisa, devido ao casamento de vários fatores tais como:

- imprecisão do sensor.
- imprecisão da fonte V_TESTE.
- tempo de estabilização programado insuficiente.
- resistência não nula das chaves no estado ON.
- resistência não infinita das chaves no estado OFF.
- resistência da fiação.
- fugas de corrente entre o comando e o caminho de corrente da chave.

Define-se dois parâmetros, ERR_IS e ERR_CD, como sendo as imprecisões das medidas no teste de isolação e de condução, respectivamente.

Sabe-se que o teste de isolação deve considerar qualquer resistência superior a RIS como interconexão do tipo I, e NI no caso contrário. Devido à imprecisão, as resistências no intervalo

[(1 - ERR_IS) * RIS , (1 + ERR_IS) * RIS]

poderão ser interpretadas ora como interconexões do tipo I, ora como do tipo NI.

Sabe-se que o teste de condução deve considerar qualquer resistência inferior a RCD como interconexão do tipo Ç e NC no caso contrário. Devido à imprecisão, as resistências no intervalo

[(1 - ERR_CD) * RCD , (1 + ERR_CD) * RCD]

poderão ser interpretadas ora como interconexões do tipo Ç ora como do tipo NC.

A amenização deste problema pode ocorrer através da programação adequada dos parâmetros RCD e RIS, como será visto, respectivamente, nas seções 2.5 e 2.6.

2.4.6 - Pares Compatíveis VCD - RCD

Cada um destes dois parâmetros é programável, dentro de um determinada faixa. No entanto, existe um dependência entre eles.

Para baixos valores de RCD, normalmente deve-se ajustar baixos valores de VCD, afim de permitir a leitura de resistências em torno de RCD sem atingir o limite de corrente LIM_CORR.

Para altos valores de RCD, normalmente deve-se ajustar altos valores de VCD, para que a corrente de medida possa atingir uma faixa suficientemente sensível do sensor.

O AI poderá, automaticamente, listar e consistir todas as possibilidades de ajuste de VCD, para um dado RCD, ou vice-versa.

2.4.7 - Pares Compatíveis VIS - RIS

Cada um destes dois parâmetros é programável, dentro de um determinada faixa. No entanto, existe um dependência entre eles.

Para baixos valores de RIS, normalmente deve-se ajustar baixos valores de VIS, afim de permitir a leitura de resistências em torno de RIS sem atingir o limite de corrente LIM_CORR.

Para altos valores de RIS, normalmente deve-se ajustar altos valores de VIS, para que a corrente de medida possa atingir uma faixa suficientemente sensível do sensor.

O AI poderá, automaticamente, listar e consistir todas as possibilidades de ajuste de VIS, para um dado RIS, ou vice-versa.

2.4.8 - Chaveamento da tensão V_TESTE

A tensão V_TESTE precisa ser comutada, entre os testes de condução e isolação, para os valores VCD e VIS.

Este processo tem um atraso associado, que consome tempo no teste. O tempo gasto é da ordem de décimos de segundo até um segundo, para cada comutação.

O número de chaveamentos desta tensão deve ser reduzido, sempre que possível. Pode-se, por exemplo, pensar em testar simultaneamente várias USTs pequenas e que possuam os mesmos parâmetros VCD e VIS. Executa-se primeiro o teste de condução para todas as USTs, e depois o teste de isolação para todas as USTs. O número de chaveamentos é reduzido tantas vezes quanto for o número de USTs simultaneamente testadas.

2.5 - Faixa Indeterminada de Condução

Existe uma faixa, ou intervalo real, em torno do valor RCD, em que a resistência de uma interconexão poderá ser interpretada ora como sendo do tipo Ç ora como do tipo NC. Esta indeterminação pode causar dois tipos de problemas:

a) um defeito do tipo circuito-aberto existente em uma rede pode não ser detectado, se uma interconexão do tipo NC for interpretada, equivocadamente, como sendo do tipo C.

b) um defeito do tipo circuito-aberto inexistente pode ser detectado, se uma interconexão do tipo C for interpretada, equivocadamente, como sendo do tipo NC.

O problema "a)", sem dúvida, é mais grave do que o "b", embora este último cause trabalho adicional ao usuário.

Estes problemas se devem a 2 motivos:

- uso de teste de condução com rede, ao invés de com rede combinatória.
- imprecisão da medida.

O uso do teste de condução com rede garante que a resistência entre qualquer das

$$C(z,2)$$

dupla de pontos numa rede com z pontos será menor do que

$$2 * RCD,$$

caso tenha sido constatado, através de medidas, que cada uma das

$$z - 1$$

resistências entre o ponto identificador da rede e os demais pontos da rede é menor do que RCD.

O teste infere que as

$$C(z,2) - (z-1)$$

interconexões não medidas são do tipo Ç o que pode não ocorrer se algumas delas estiverem no intervalo

$$[RCD , 2 * RCD].$$

Devido à imprecisão da medida, este intervalo de indeterminação é aumentado, para

$$[RCD * (1 - ERR\_CD) , 2 * RCD * (1 + ERR\_CD)].$$

Se uma interconexão entre dois pontos quaisquer de uma rede estiver neste intervalo, não se pode prever se ela vai ser interpretada como sendo do tipo C ou do tipo NC.

Se a resistência desta interconexão estivesse no intervalo

$$[RCD , 2 * RCD * (1 + ERR\_CD)],$$

desejaria-se que ela fosse interpretada como sendo do tipo NÇ pois em caso contrário ocorreria o problema "a)".

Se a resistência desta interconexão estivesse no intervalo

$$[RCD * (1 - ERR\_CD) , RCD),$$

desejaria-se que ela fosse interpretada como sendo do tipo

C pois em caso contrário ocorreria o problema "b)".

A ocorrência deste tipo de problemas é rara, mas pode ser solucionada pela programação criteriosa dos parâmetros MAX_ACEIT_RCD e MAX_TIP_RCD.

Para evitar a ocorrência do problema "a)", deve-se fazer com que

MAX_ACEIT_RCD > 2 * RCD * (1 + ERR_CD).

Para evitar a ocorrência do problema "b)", deve-se fazer com que

MAX_TIP_RCD < RCD * (1 - ERR_CD).

Para tanto, basta programar os valores de MAX_ACEIT_RCD e MAX_TIP_RCD. O sistema calcula automaticamente um valor de RCD que tenta satisfazer as duas condições acima. Caso não seja possível satisfazê-las simultaneamente, o sistema automaticamente substitui o valor de MAX_TIP_RCD por um menor, o que dá margem à ocorrência do problema "b)", mas garante que o problema mais grave, o "a)", não ocorrerá.

2.6 - Faixa Indeterminada de Isolação

Existe uma faixa, ou intervalo real, em torno do valor RIS, em que a resistência de uma interconexão poderá ser interpretada ora como sendo do tipo I, ora como do tipo NI. Esta indeterminação pode causar dois tipos de problemas:

a) um defeito do tipo curto-circuito existente entre 2 redes pode não ser detectado, se uma interconexão do tipo NI for interpretada, equivocadamente, como sendo do tipo I.

b) um defeito do tipo curto-circuito inexistente pode ser detectado, se uma interconexão do tipo I for interpretada, equivocadamente, como sendo do tipo NI.

O problema "a)", sem dúvida, é mais grave do que

o "b", embora este último cause trabalho adicional ao usuário.

Estes problemas se devem a imprecisão da medida, que gera um intervalo de indeterminação em torno de RIS:

[RIS * (1 - ERR_IS) , RIS * (1 + ERR_IS)].

Se uma interconexão entre duas redes quaisquer estiver neste intervalo, não se pode prever se ela vai ser interpretada como sendo do tipo I ou do tipo NI.

Se a resistência desta interconexão estivesse no intervalo

[RIS * (1 - ERR_IS) , RIS],

desejaria-se que ela fosse interpretada como sendo do tipo NI, pois em caso contrário ocorreria o problema "a)".

Se a resistência desta interconexão estivesse no intervalo

(RIS , RIS * (1 + ERR_IS)],

desejaria-se que ela fosse interpretada como sendo do tipo I, pois em caso contrário ocorreria o problema "b)".

A ocorrência deste tipo de problemas é rara, mas pode ser solucionada pela programação criteriosa dos parâmetros MIN_ACEIT_RIS e MIN_TIP_RIS.

Para evitar a ocorrência do problema "a)", deve-se fazer com que

MIN_ACEIT_RIS < RIS * (1 - ERR_IS).

Para evitar a ocorrência do problema "b)", deve-se fazer com que

MIN_TIP_RIS > RIS * (1 + ERR_IS).

Para tanto, basta programar os valores de MIN_ACEIT_RIS e MIN_TIP_RIS. O sistema calcula automaticamente um valor de RIS que tenta satisfazer as duas condições acima. Caso não seja possível satisfazê-las simultaneamente, o sistema automaticamente substitui o valor de MIN_TIP_RIS por um maior, o que dá margem à

ocorrência do problema "b)", mas garante que o problema mais grave, o "a)", não ocorrerá.

2.7 - Descrição do Padrão

Um AI pode manipular vários padrões. O padrão em uso corrente está na memória do controlador do AI, enquanto que os demais residem em disco.

A descrição do padrão em memória (PAD_MEM) poderá apresentar alto grau de redundância, a fim de acelerar a execução dos algoritmos pela utilização de mais memória.

A descrição de um padrão em disco é compacta, de forma a permitir o armazenamento de muitos padrões. Cada padrão em disco pode ter 2 ou 3 arquivos associados:

a) arquivo com extensão .PAR, que armazena o valor dos parâmetros programados para este padrão, e que foram ajustados antes da programação do padrão.

b) arquivo com extensão .RED, que armazena uma lista ordenada de redes, sendo que cada rede é uma lista ordenada de pontos de teste.

c) arquivo opcional com extensão .NAM, que armazena nomes simbólicos atribuídos a pontos de teste selecionados.

Além disso, existe um arquivo de diretório de padrões, o DIRET.PAD, e um arquivo geral de configuração, o AI.CNF.

O arquivo DIRET.PAD é necessário na medida em que os nomes de padrões poderão ser mais extensos do que os permitidos no sistema operacional usado.

O arquivo AI.CNF guarda dados configuracionais, tais como o nome do padrão de carga (PAD_CRG) e a senha. A senha controla o acesso à determinadas funções auxiliares, conforme será visto no capítulo 7.

2.8 - Interfaceamento entre AI e UST

Para realizar o teste de interconexões, nomeação de pontos por caneta, ou auto-programação de padrão, é necessário interconectar os pontos de teste do AI aos pontos de teste da UST, numa correspondência de 1 para 1.

Um ponto de teste, no AI, é identificado pelo seu número ou endereço, no intervalo

[0 , n - 1].

O usuário poderá atribuir nomes simbólicos aos pontos de teste onde isto for desejável.

Por ocasião do teste, não é necessário utilizar todos os pontos disponíveis, sendo possível programar um intervalo

[LIM_INF , LIM_SUP]

contido no intervalo

[0 , n - 1],

o que acelera o teste e aumenta a flexibilidade.

As interconexões entre AI e UST devem ter resistência de contato bem menor do que RCD, e não devem introduzir capacitâncias ou indutâncias intoleráveis.

Algumas alternativas de interfaceamento, ou "fixtures" comuns são:

- camas de pregos.
- cabos com conectores.
- placas extensoras.

Para o teste de placas de circuito-impresso nuas, é comum o uso de camas-de-pregos. Alguns AIs apresentam alto grau de sofisticação mecânica no sistema de interfaceamento, apresentando inclusive carga e descarga automáticas /CIR SD/.

3 - ESPECIFICAÇÃO INFORMAL DO TESTE DE INTERCONEXÕES

O teste de interconexões é, dentre as funções do AI, a mais importante e a mais utilizada. As demais funções são auxiliares.

As arquiteturas e algoritmos estudados nos capítulos seguintes propõem-se, basicamente, a otimizar o custo do AI e a performance do TINT.

As arquiteturas serão suficientes também para o suporte de funções auxiliares dependentes do sistema de aquisição de dados, tais como autoprogramação, autodiagnósticos e nomeação simbólica de pontos via caneta, embora este estudo não seja prioridade nesta dissertação.

A especificação informal do TINT visa definir os objetivos globais deste teste e dos dois testes em que se subdivide: TCOND e TISOL.

3.1 - Diagrama de Fluxo de Dados do TINT

O diagrama de fluxo de dados do TINT é exibido pela figura 3.1. Dados e módulos serão descritos informalmente, sem preocupação com sua estrutura ou detalhamento.

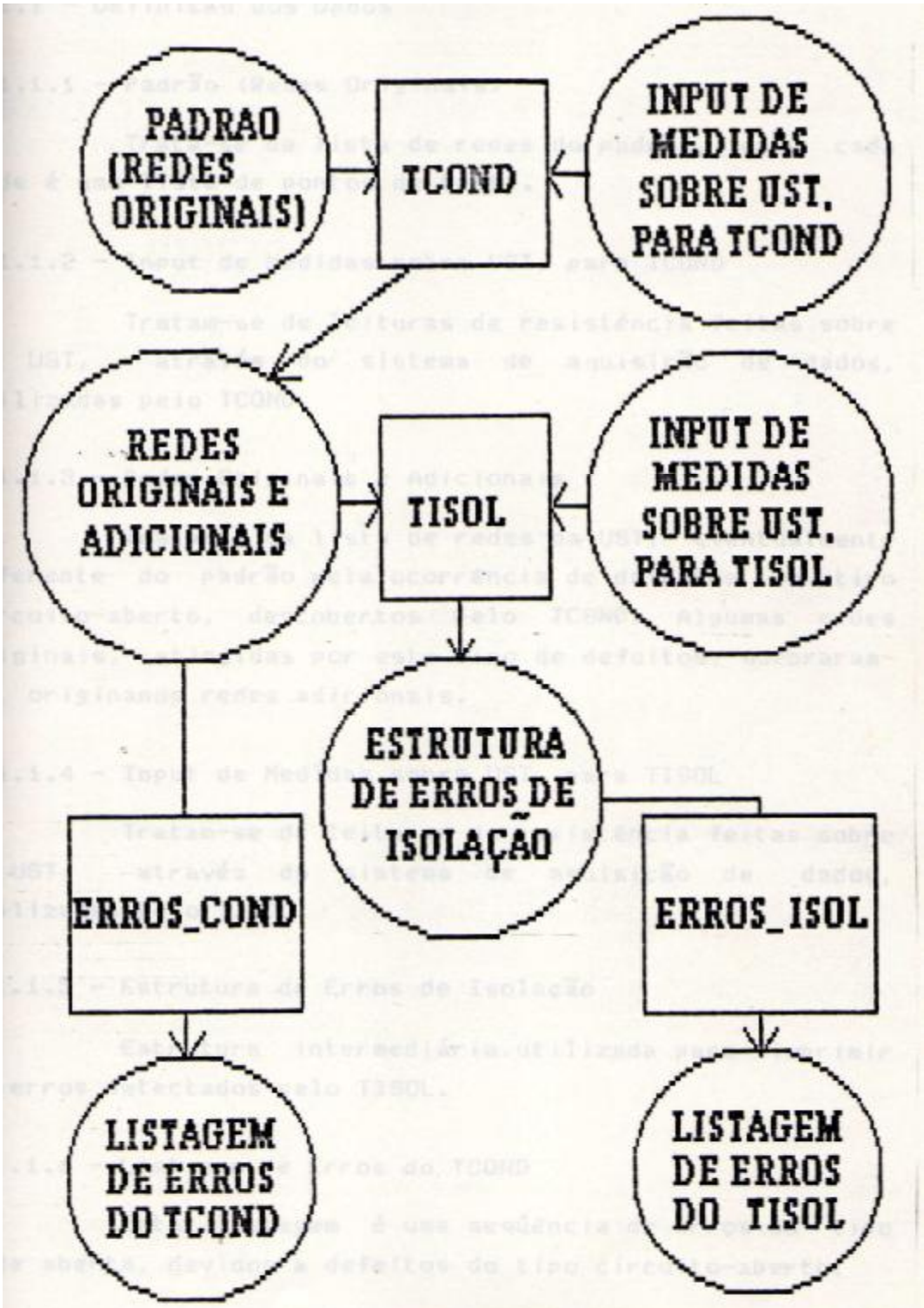

FIGURA 3.1 DFD do TINT

3.1.1 - Definição dos Dados

3.1.1.1 - Padrão (Redes Originais)

Trata-se da lista de redes do padrão, onde cada rede é uma lista de pontos de teste.

3.1.1.2 - Input de Medidas sobre UST, para TCOND

Tratam-se de leituras de resistência feitas sobre a UST, através do sistema de aquisição de dados, utilizadas pelo TCOND.

3.1.1.3 - Redes Originais e Adicionais

Trata-se da lista de redes da UST, eventualmente diferente do padrão pela ocorrência de defeitos do tipo circuito-aberto, descobertos pelo TCOND. Algumas redes originais, atingidas por este tipo de defeitos, quebraram-se, originando redes adicionais.

3.1.1.4 - Input de Medidas sobre UST, para TISOL

Tratam-se de leituras de resistência feitas sobre a UST, através do sistema de aquisição de dados, utilizadas pelo TISOL.

3.1.1.5 - Estrutura de Erros de Isolação

Estrutura intermediária utilizada para imprimir os erros detectados pelo TISOL.

3.1.1.6 - Listagem de Erros do TCOND

Esta listagem é uma seqüência de erros do tipo rede aberta, devidos a defeitos do tipo circuito-aberto.

Cada erro do tipo rede aberta corresponde a um ou mais defeitos do tipo circuito-aberto em uma rede. Normalmente, cada defeito do tipo circuito-aberto provoca o

aparecimento de uma rede adicional em uma rede original.

A figura 3.2 mostra um exemplo de erro do tipo rede aberta, com dois defeitos do tipo circuito-aberto e duas redes adicionais originadas por estes defeitos.

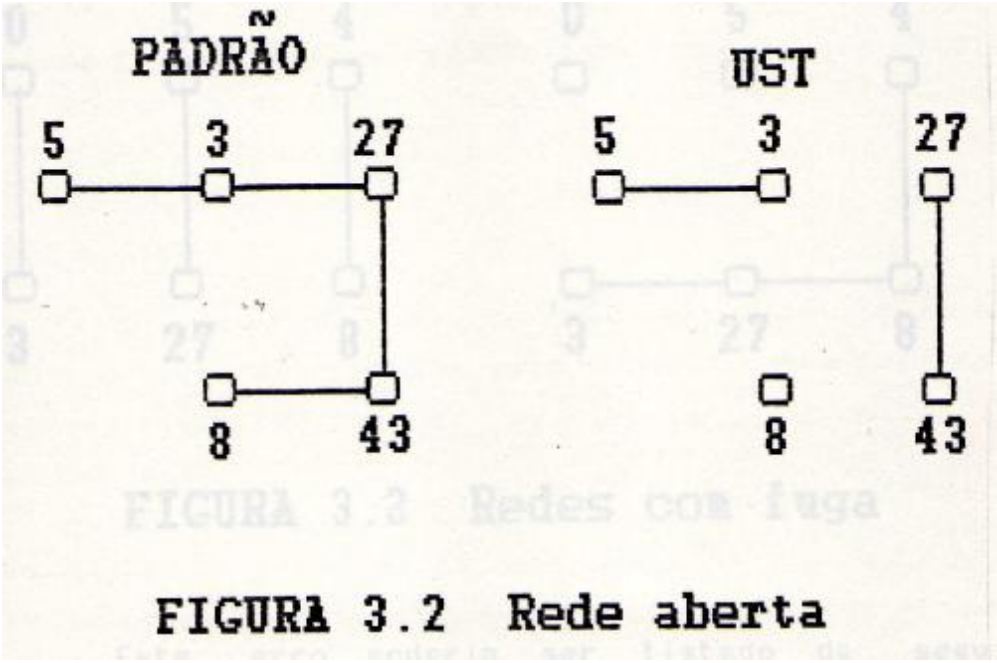

FIGURA 3.2 Rede aberta

Este erro poderia ser listado da seguinte maneira:

- rede 3 aberta - redes adicionais: 8, 27

3.1.1.7 - Listagem de Erros do TISOL

Esta listagem é uma seqüência de erros do tipo fuga entre redes, devidos a defeitos do tipo curto-circuito.

Cada erro do tipo fuga entre redes corresponde a um ou mais defeitos do tipo curto-circuito entre uma rede de referência e outras redes.

A figura 3.3 mostra um exemplo de erro do tipo fuga entre redes, com dois defeitos do tipo curto-circuito. Alguns defeitos do tipo circuito-aberto também foram

incluídos para ressaltar também o envolvimento de redes adicionais em curto-circuitos.

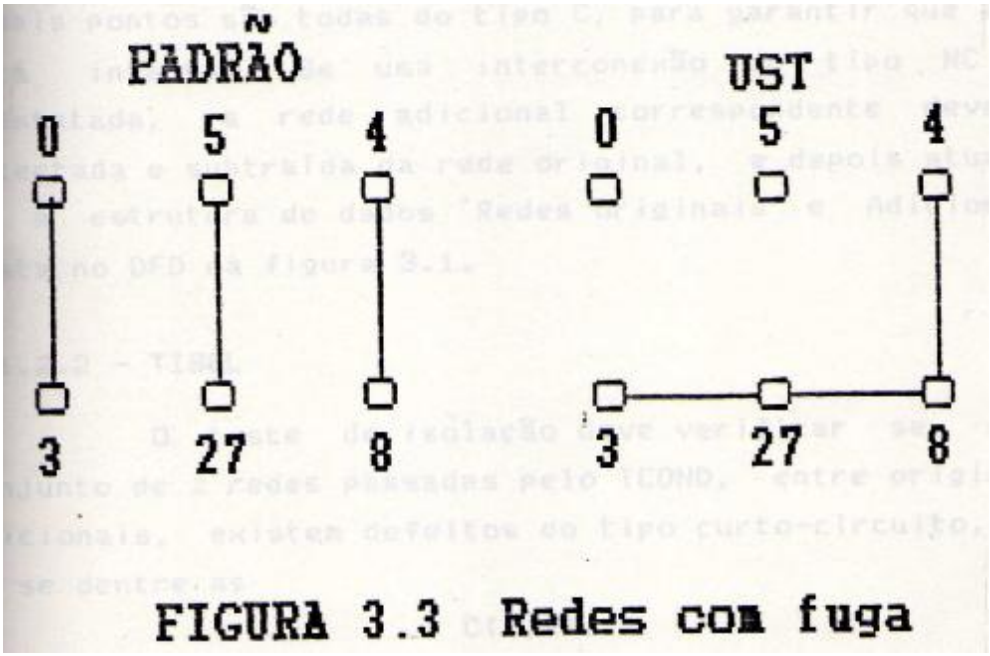

FIGURA 3.3 Redes com fuga

Este erro poderia ser listado da seguinte maneira:

- rede 3(0) em fuga contra redes: 4(4), 27(5)

O número entre parêntesis, ao lado de cada rede, indica a rede original a que pertence esta rede. Isto é útil no caso em que a rede é uma rede adicional, devido a defeitos do tipo circuito-aberto. Note que, neste exemplo, isto ocorre para as redes 3 e 27.

3.1.2 - Definição dos Módulos

3.1.2.1 - TCOND

O teste de condução deve verificar se cada uma das redes constantes no padrão está perfeita, sem defeitos do tipo circuito-aberto.

O TCOND utiliza o conceito de rede, e não de rede combinacional, no escopo desta dissertação. Desta forma, o

TCOND deve apenas verificar, em uma rede de z pontos, se as

$$z - 1$$

interconexões entre o ponto identificador da rede e os demais pontos são todas do tipo Ç para garantir que a rede está intacta. Se uma interconexão do tipo NC for constatada, a rede adicional correspondente deve ser detectada e subtraída da rede original, e depois atualiza-se a estrutura de dados "Redes Originais e Adicionais", vista no DFD da figura 3.1.

3.1.2.2 - TISOL

O teste de isolação deve verificar se, em um conjunto de z redes passadas pelo TCOND, entre originais e adicionais, existem defeitos do tipo curto-circuito, isto é, se dentre as

$$C(z,2)$$

interconexões entre pares de redes existem algumas do tipo NI. Neste caso, as descrições dos erros devem ser incluídas na estrutura de dados "Estrutura de Erros de Isolação".

Do ponto de vista do TISOL, o conceito de rede e de ponto identificador de rede se confunde, pois ambos representam um mesmo potencial elétrico, quando o circuito é energizado.

3.1.2.3 - ERROS_COND

Este módulo é responsável pela listagem dos erros detectados pelo TCOND.

3.1.2.4 - ERROS_ISOL

Este módulo é responsável pela listagem dos erros detectados pelo TISOL.

3.2 - Exemplos de Listagem de Erros

A figura 3.4 mostra um PADRÃO e uma UST que

produzem listagens de erros vistas nas seções de 3.2.1 até 3.2.4. A diferença entre as 4 listagens é devida a programação dos limites de erros LIM_ERROS_ISOL, LIM_ERROS_COND e LIM_ERR_COND_REDE.

Admitiu-se que alguns pontos foram nomeados simbolicamente:

0 = VCC

23 = GND

20 = AUX_GND

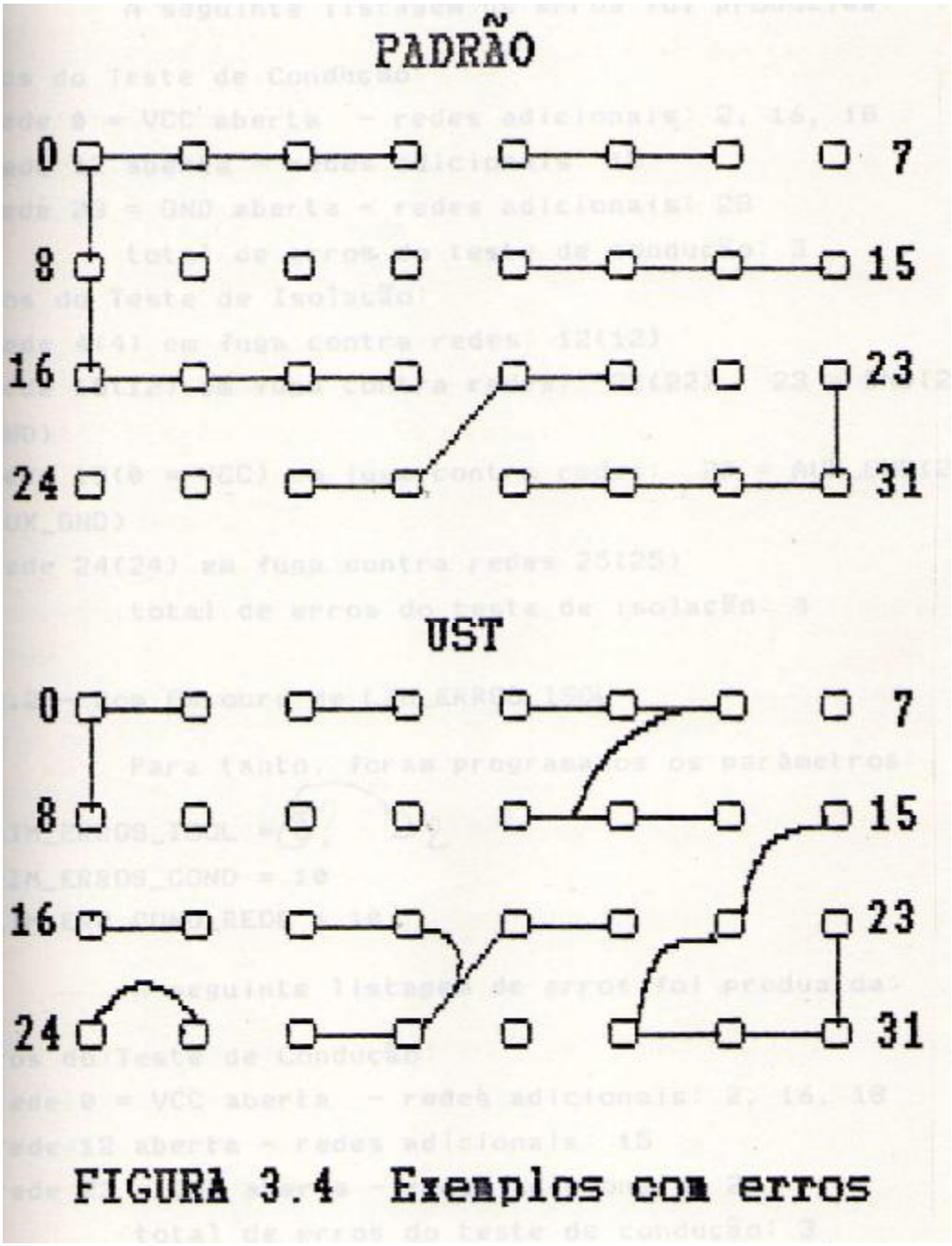

FIGURA 3.4 Exemplos com erros

3.2.1 - Sem Estouro de Limites de Erros

Para tanto, foram programados os parâmetros:

- LIM_ERROS_ISOL = 10
- LIM_ERROS_COND = 10
- LIM_ERR_COND_REDE = 10

A seguinte listagem de erros foi produzida:

Erros do Teste de Condução:
- rede 0 = VCC aberta - redes adicionais: 2, 16, 18
- rede 12 aberta - redes adicionais: 15
- rede 23 = GND aberta - redes adicionais: 28

total de erros do teste de condução: 3

Erros do Teste de Isolação:
- rede 4(4) em fuga contra redes: 12(12)
- rede 15(12) em fuga contra redes: 22(22), 23 = GND(23 = GND)
- rede 18(0 = VCC) em fuga contra redes: 20 = AUX_GND(20 = AUX_GND)
- rede 24(24) em fuga contra redes 25(25)

total de erros do teste de isolação: 4

3.2.2 - Com Estouro de LIM_ERROS_ISOL

Para tanto, foram programados os parâmetros:

- LIM_ERROS_ISOL = 2
- LIM_ERROS_COND = 10
- LIM_ERR_COND_REDE = 10

A seguinte listagem de erros foi produzida:

Erros do Teste de Condução:
- rede 0 = VCC aberta - redes adicionais: 2, 16, 18
- rede 12 aberta - redes adicionais: 15
- rede 23 = GND aberta - redes adicionais: 28

total de erros do teste de condução: 3

Erros do Teste de Isolação:
- rede 4(4) em fuga contra redes: 12(12)
- rede 15(12) em fuga contra redes: 22(22), 23 = GND(23 = GND)

limite de erros do teste de isolação estourado: 2

3.2.3 - Com Estouro de LIM_ERROS_COND

Para tanto, foram programados os parâmetros:

- LIM_ERROS_ISOL = 10
- LIM_ERROS_COND = 2
- LIM_ERR_COND_REDE = 10

A seguinte listagem de erros foi produzida:

Erros do Teste de Condução:
- rede 0 = VCC aberta - redes adicionais: 2, 16, 18
- rede 12 aberta - redes adicionais: 15

limite de erros do teste de condução estourado: 2

Erros do Teste de Isolação:
- rede 4(4) em fuga contra redes: 12(12)
- rede 15(12) em fuga contra redes: 22(22), 23 = GND(23 = GND)
- rede 18(0 = VCC) em fuga contra redes: 20 = AUX_GND(20 = AUX_GND)
- rede 24(24) em fuga contra redes 25(25)

total de erros do teste de isolação: 4

3.2.4 - Com Estouro de LIM_ERR_COND_REDE

Para tanto, foram programados os parâmetros:

- LIM_ERROS_ISOL = 10
- LIM_ERROS_COND = 10
- LIM_ERR_COND_REDE = 2

A seguinte listagem de erros foi produzida:

Erros do Teste de Condução:
- rede 0 = VCC aberta - redes adicionais: 2, 16-->SUSPEITA
- rede 12 aberta - redes adicionais: 15
- rede 23 = GND aberta - redes adicionais: 28

total de erros do teste de condução: 3

Erros do Teste de Isolação:
- rede 4(4) em fuga contra redes: 12(12)
- rede 15(12) em fuga contra redes: 22(22), 23 = GND(23 =

GND)

- rede 24(24) em fuga contra redes 25(25)

total de erros do teste de isolação: 3

Nota-se que, devido à limitação de LIM_ERR_COND_REDE em 2, não foi detectada a rede adicional 18, que assumiu-se falsamente pertencer a rede adicional 16, marcada como SUSPEITA. Por este motivo, também não foi detectada a fuga entre a rede adicional 18 e a rede 20.

3.3 - Dependências entre o TISOL e o TCOND

Existe uma dependência, não necessária, mas aconselhável, do TISOL para os resultados produzidos pelo TCOND. Esta dependência pode ser vista no DFD da figura 3.1.

O TISOL, para ter condições de funcionamento eficiente, deve conhecer todas as redes da UST, tanto as originais como as adicionais, que são descobertas pelo TCOND. Se o TISOL desconhecer as redes adicionais, sua performance degrada para todos os algoritmos e arquiteturas estudados.

Algo semelhante ocorre para o TCOND, que poderia processar várias redes em paralelo através do uso de múltiplos sensores, desde que soubesse que não existem fugas entre as redes processadas em paralelo, que são descobertas pelo TISOL.

Decidiu-se que o TISOL será feito após o TCOND pelos seguintes motivos:

- normalmente, T_CIC_IS é bem maior do que T_CIC_CD, o que sugere que deve-se tentar fazer com que a complexidade do TISOL seja mais reduzida do que a de TCOND, a fim de reduzir o tempo total do teste.
- existe uma outra opção de paralelismo para o TCOND, que é o processamento paralelo dentro de uma única rede, sem necessitar de informações produzidas pelo TISOL.

4 - ARQUITETURAS PARA ANALISADORES DE INTERCONEXÕES

Neste capítulo serão estudadas algumas arquiteturas viáveis para o AI.

Será entendido por arquitetura, um modelo, em alto nível, do hardware que implementa o AI. Serão utilizados blocodiagramas para representá-la, e os blocos componentes, por sua vez, também serão detalhados através de blocodiagramas, até chegar-se a um nível de abstração abaixo do qual a estrutura não mais será analisada. Neste nível, será feita uma descrição comportamental do bloco, mostrando a relação entre suas entradas e saídas através de uma linguagem informal ou de uma linguagem algorítmica adaptada.

Um AI, a princípio, pode ser dividido em dois grandes blocos:

- um controlador com periféricos de interação com o usuário e memória de massa.
- um sistema de aquisição de dados para medir resistências elétricas na UST.

O controlador pode ser um dos micro-computadores ou microprocessadores comuns no mercado, bem como os periféricos de interação com o usuário e a memória de massa. Não é prioridade desta dissertação detalhar a arquitetura do controlador, nem mesmo definir qual dentre os comerciais deve ser o usado. Ressalta-se apenas que o controlador escolhido deverá ter uma performance suficiente para não onerar demais o tempo do teste, e apresentar um custo aceitável.

O interesse maior nesta dissertação é definir a arquitetura do sistema de aquisição de dados, pois esta influi decisivamente sobre a complexidade dos algoritmos do TCOND e do TISOL. Entende-se por complexidade o número de medidas de resistências executado por um teste. O produto da complexidade de um teste pelo tempo de estabilização de

de suas medidas resulta aproximadamente no seu tempo total.

Será entendido por algoritmo um modelo em alto nível do software executado pelo controlador. Este comanda o sistema de aquisição de dados para a obtenção de medidas de resistência, realiza a comparação entre PADRÃO e UST, e imprime as mensagens de erro. Os algoritmos podem ser detalhados de maneira informal, por fluxogramas ou por outros meios.

4.1 - Blocodiagrama típico de um AI

O blocodiagrama da figura 4.1 visa especificar a arquitetura do AI como um todo.

Os blocos controlador, video, teclado, impressora e winchester, como já foi explicado, não serão detalhados nesta dissertação. Pode-se imaginar que sejam um computador da linha IBM/PC com seus periféricos normais.

Os demais blocos formam o sistema de aquisição de dados, que possui um conjunto de n pontos de teste, conectáveis à UST. Mostra-se uma das possíveis modularizações dos pontos de teste.

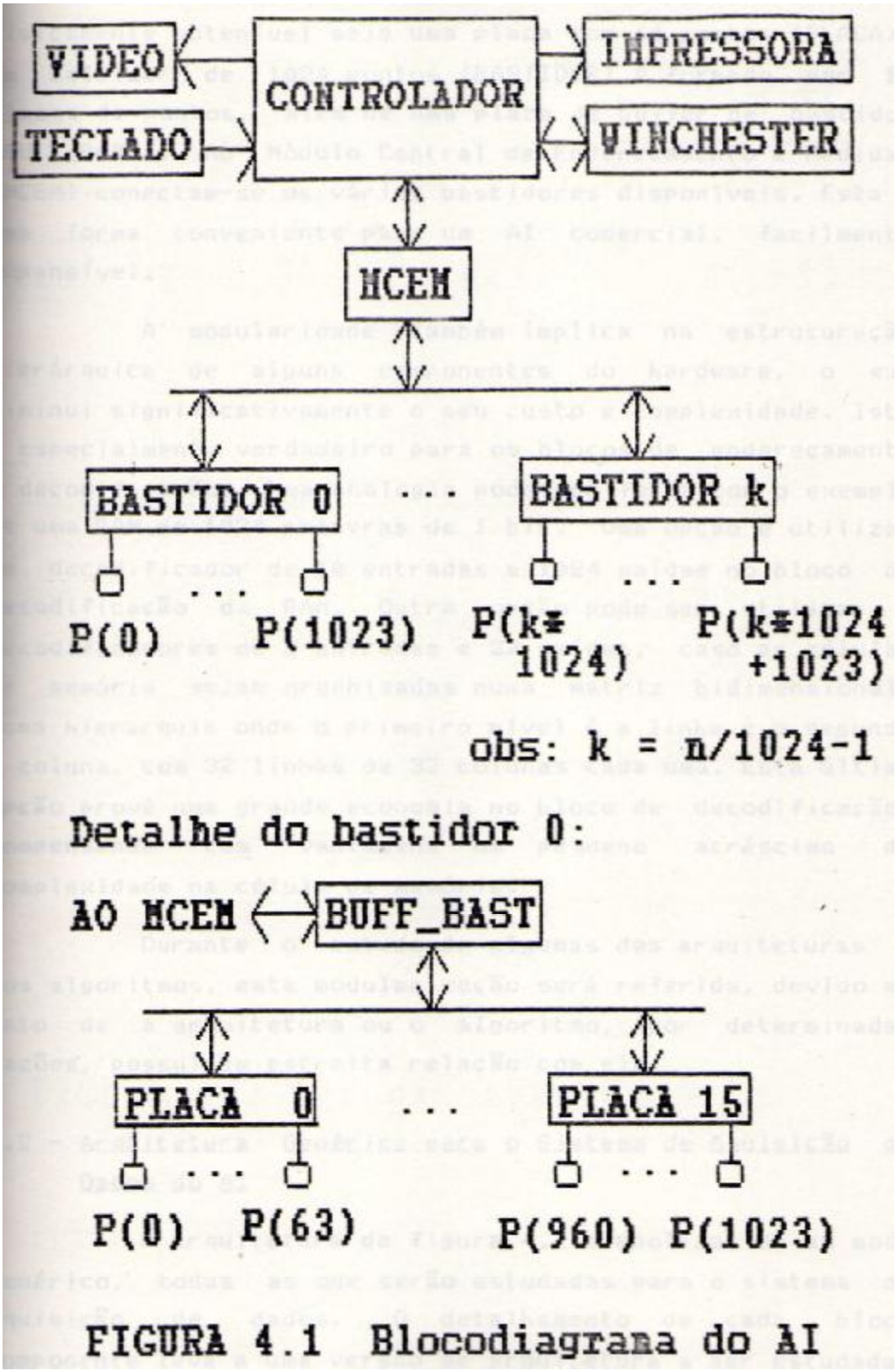

FIGURA 4.1 Blocodiagrama do AI

Assume-se que o mínimo módulo de pontos de teste fisicamente obtenível seja uma placa com 64 pontos (PLACA). Um bastidor de 1024 pontos (BASTIDOR) é formado por 16 placas de pontos, além de uma placa de buffer de bastidor (BUFF_BAST). Ao Módulo Central de Endereçamento e Medidas (MCEM) conectam-se os vários bastidores disponíveis. Esta é uma forma conveniente para um AI comercial, facilmente expansível.

A modularidade também implica na estruturação hierárquica de alguns componentes do hardware, o que diminui significativamente o seu custo e complexidade. Isto é especialmente verdadeiro para os blocos de endereçamento e decodificação. Uma analogia pode ser feita com o exemplo de uma RAM de 1024 palavras de 1 bit. Uma opção é utilizar um decodificador de 10 entradas e 1024 saídas no bloco de decodificação da RAM. Outra opção pode ser utilizar 2 decodificadores de 5 entradas e 32 saídas, caso as células de memória sejam organizadas numa matriz bidimensional, numa hierarquia onde o primeiro nível é a linha e o segundo a coluna, com 32 linhas de 32 colunas cada uma. Esta última opção provê uma grande economia no bloco de decodificação, compensando com vantagens um pequeno acréscimo de complexidade na célula de memória.

Durante o estudo de algumas das arquiteturas e dos algoritmos, esta modularização será referida, devido ao fato de a arquitetura ou o algoritmo, por determinadas razões, possuirem estreita relação com ela.

4.2 - Arquitetura Genérica para o Sistema de Aquisição de Dados do AI

A arquitetura da figura 4.2 simboliza, de um modo genérico, todas as que serão estudadas para o sistema de aquisição de dados. O detalhamento de cada bloco componente leva a uma versão de arquitetura a ser estudada.

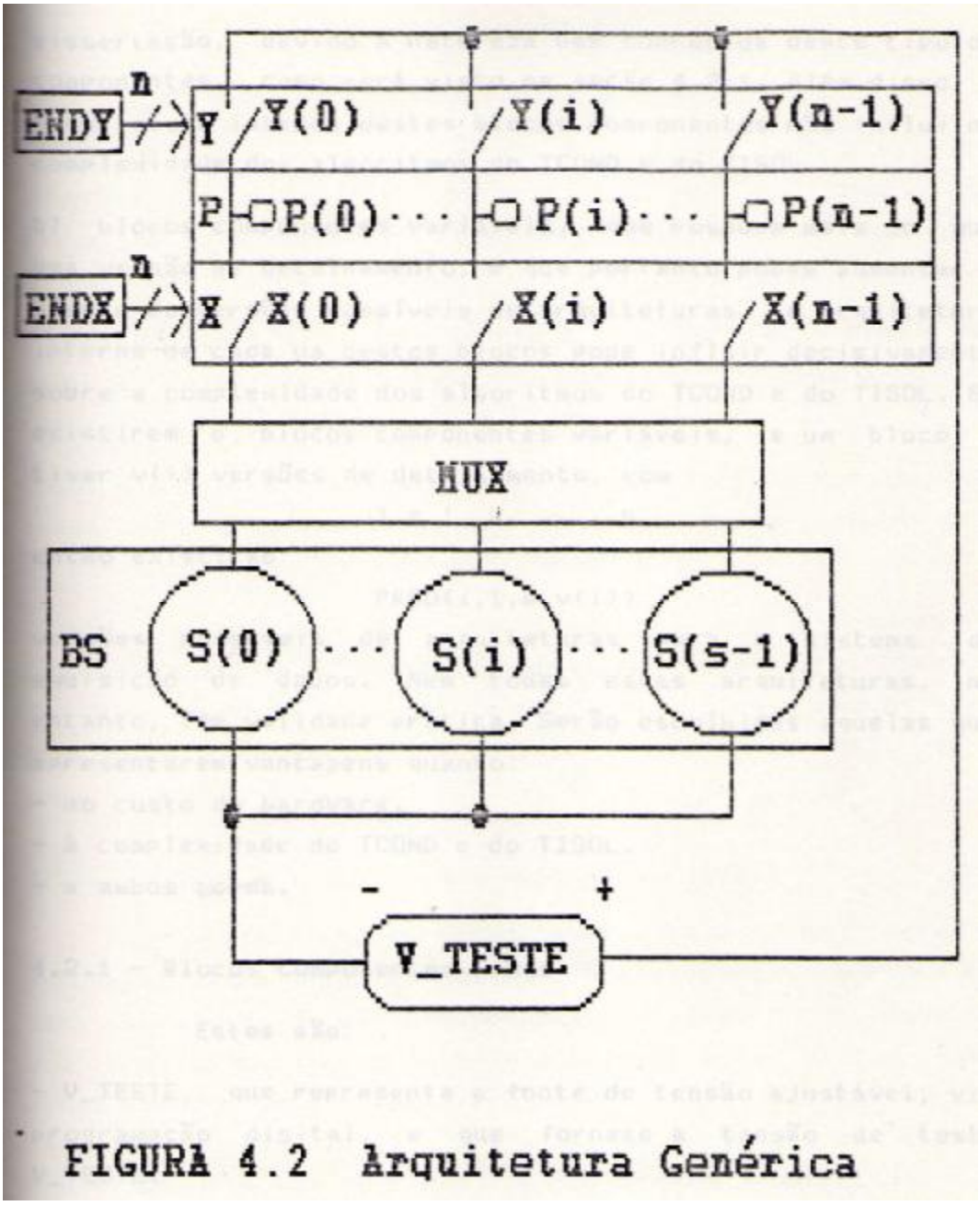

FIGURA 4.2 Arquitetura Genérica

Dentre os blocos componentes mostrados na figura 4.2, temos 2 classes principais:

a) blocos componentes fixos, que possuem apenas uma versão de detalhamento, que não será feito no escopo desta

dissertação, devido à natureza bem conhecida deste tipo de componentes, como será visto na seção 4.2.1. Além disso, a arquitetura interna destes blocos componentes não influi na complexidade dos algoritmos do TCOND e do TISOL.

b) blocos componentes variáveis, que possuem mais do que uma versão de detalhamento, e que portanto podem aumentar o número de versões possíveis de arquiteturas. A arquitetura interna de cada um destes blocos pode influir decisivamente sobre a complexidade dos algoritmos do TCOND e do TISOL. Se existirem b blocos componentes variáveis, e um bloco i tiver v(i) versões de detalhamento, com

$$i = 1, 2, ..., b,$$

então existirão

$$PROD(i,1,b,v(i))$$

versões possíveis de arquiteturas para o sistema de aquisição de dados. Nem todas estas arquiteturas, no entanto, têm validade prática. Serão escolhidas aquelas que apresentarem vantagens quanto:

- ao custo do hardware.
- à complexidade do TCOND e do TISOL.
- a ambos acima.

4.2.1 - Blocos Componentes Fixos

Estes são:

- V_TESTE, que representa a fonte de tensão ajustável, via programação digital, e que fornece a tensão de teste V_TESTE.

- P, que é o conjunto de pontos de teste P(i), apresentados fisicamente na forma de pinos de conectores ou de camas-de-pregos, por exemplo.

- X e Y, que são os conjuntos das duas chaves X(i) e Y(i) associadas a cada ponto de teste P(i).

4.2.2 - Blocos Componentes Variáveis

Estes são:

- ENDY, o bloco de endereçamento Y, que gera os n sinais de controle de estado das chaves do conjunto Y. Como cada chave tem dois estados, ON ou OFF, cada sinal de controle corresponde a um bit.

- ENDX, o bloco de endereçamento X, que gera os n sinais de controle de estado das chaves do conjunto X. Cada sinal de controle corresponde a um bit.

- BS, o banco de sensores, que é um conjunto de s sensores. Todos os s sensores podem ser utilizados simultaneamente, em circunstâncias em que isto for útil.

- MUX, o multiplexador de sensores, que determina o compartilhamento de sensores pelos pontos de teste. Através do MUX, pode-se conectar um ponto P(i), cuja chave X(i) deve estar no estado ON, a um determinado sensor S(j), se a conexão for permitida.

4.3 - Versões dos Blocos Componentes Variáveis

Nesta seção serão apresentadas as versões estudadas para cada um dos blocos componentes variáveis. Cada versão de um bloco componente variável corresponde a uma forma de detalhamento diferente para este bloco. Este detalhamento será apresentado em alto nível de abstração. Esquemas destes blocos, até o nível lógico, e em certos casos até o nível de circuitos integrados comerciais, já foram rascunhados a fim de analisar o custo e a implementabilidade física destes blocos. Estes esquemas não se encontram anexos, uma vez que na presente dissertação aborda-se a arquitetura em um nível de abstração mais alto.

4.3.1 - Versões do Bloco ENDY

4.3.1.1 - Versão ENDY/A

Nesta versão, permite-se que no máximo 1 das n linhas Y comande o estado da sua chave correspondente para o estado ON. A figura 4.3 mostra as entradas e saídas deste bloco.

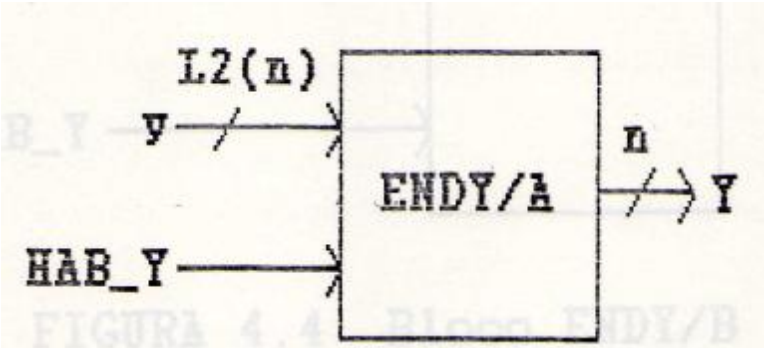

FIGURA 4.3 Bloco ENDY/A

O sinal HAB_Y, se desativado, leva todas as chaves de Y ao estado OFF. O endereço y especifica, quando HAB_Y está ativo, qual das n chaves Y estará no estado ON. Esta é a bem conhecida estrutura de um decodificador simples, facilmente implementável com componentes extremamente simples e comuns no mercado.

4.3.1.2 - Versão ENDY/B

Nesta versão, permite-se que

0 ou exp2(i),

das n linhas Y comandem o estado das suas chaves correspondentes para o estado ON, com

i = 0, 1, 2, ..., log2(n).

A figura 4.4 mostra as entradas e saídas deste bloco.

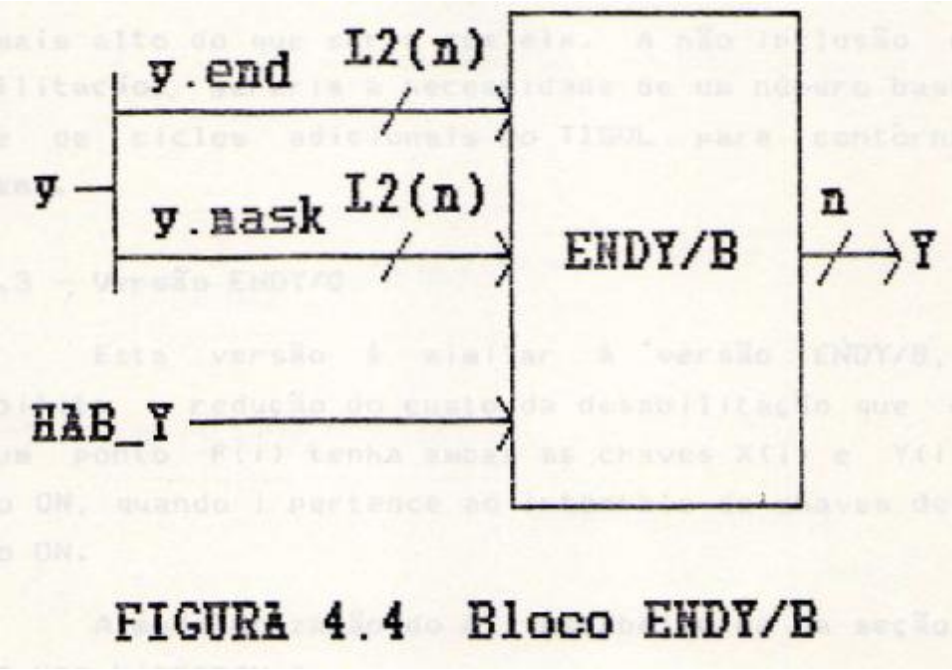

FIGURA 4.4 Bloco ENDY/B

O sinal HAB_Y, se desativado, leva todas as chaves de Y ao estado OFF. Caso contrário, estarão em ON todas as chaves cujos endereços estão no intervalo

[dec(y.end & y.mask) , dec(y.end | ~y.mask)].

O vetor binário y.mask deve possuir 1's à esquerda e 0's à direita. Como exemplo, para n = 16384, se

y.end = 11010011000000 e

y.mask = 11111111110000,

então as 16 chaves Y cujo endereço está no intervalo

[13504 , 13519]

estarão no estado ON.

A versão ENDY/B foi desenhada para trabalhar em conjunto com a versão ENDX/A do bloco ENDX. Se dentro do intervalo de endereços com chaves de Y no estado ON, estiver o endereço correspondente à única chave de X habilitada por ENDX/A, a chave de Y cujo endereço é igual a esta chave de X deve ser deixada no estado OFF, para evitar um curto-circuito óbvio, com a conseqüente leitura de uma

resistência nula. O custo desta desabilitação automática por hardware é bastante alto, tornando o custo de ENDY/B bem mais alto do que seria sem ela. A não inclusão desta desabilitação, geraria a necessidade de um número bastante grande de ciclos adicionais no TISOL para contornar o problema.

4.3.1.3 - Versão ENDY/C

Esta versão é similar à versão ENDY/B, mas possibilita a redução do custo da desabilitação que evita que um ponto P(i) tenha ambas as chaves X(i) e Y(i) no estado ON, quando i pertence ao intervalo de chaves de Y no estado ON.

A modularização do AI, estabelecida na seção 4.1, define uma hierarquia:

a) MCEM, que contém k bastidores.

b) BASTIDOR, que contém 16 placas.

c) PLACA, que contém 64 pontos, e é o módulo comercial mínimo de pontos.

d) PONTO, que corresponde a um ponto de teste.

Utilizando esta hierarquia na versão ENDY/B, o custo da desabilitação por hardware para o conflito entre uma chave X(i) e uma chave Y(i) simultaneamente no estado ON ainda é muito alto. Isso ocorre mesmo no nível PLACA, que é onde a desabilitação deve ser implementada. Como existem muitas placas, este é um custo bastante repetitivo.

A arquitetura de ENDY/C utiliza um nível hierárquico adicional, abaixo do nível PLACA, de forma que a nova hierarquia é:

a) MCEM, que contém k bastidores.

b) BASTIDOR, que contém 16 placas.

c) PLACA, que contém 64 pontos, organizados numa matriz bidimensional de 8 linhas por 8 colunas de pontos, e ainda é o módulo comercial mínimo de pontos.

d) LINHA, que contém 8 colunas de pontos.

e) COLUNA, que corresponde a um ponto dentro de cada linha.

A figura 4.5 mostra as entradas e saídas deste bloco.

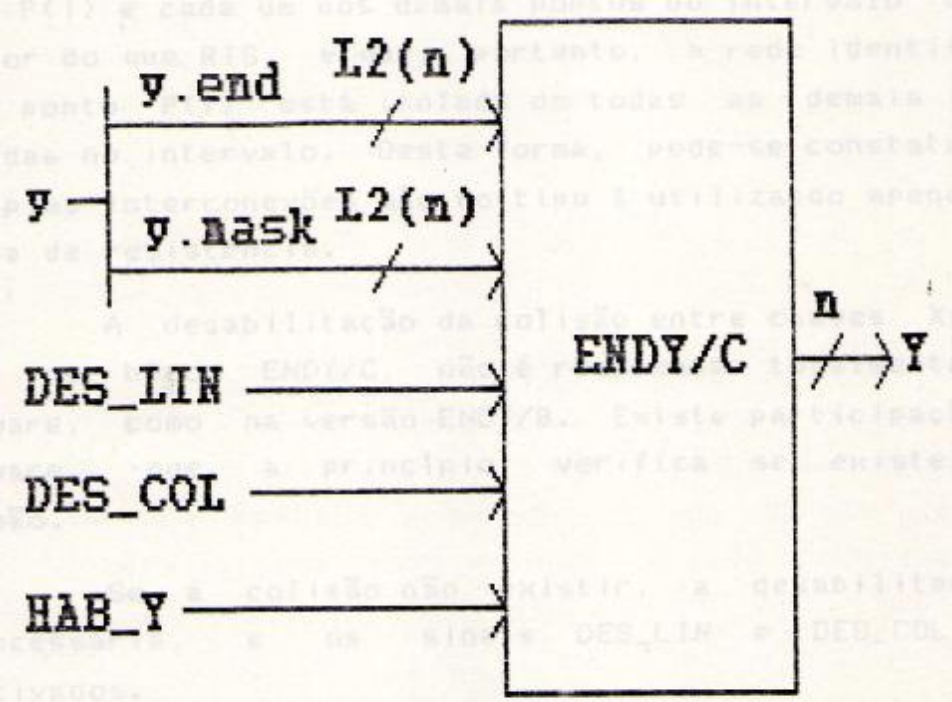

FIGURA 4.5 Bloco ENDY/C

O sinal HAB_Y, se desativado, leva todas as chaves de Y ao estado OFF. Caso contrário, estarão em ON todas as chaves de Y no intervalo

[dec(y.end & y.mask) , dec(y.end | ~y.mask)],

como também ocorre na versão ENDY/B.

Se dentro deste intervalo estiver o endereço da única chave de X no estado ON, a correspondente chave de Y deve ser desabilitada. Esta é uma situação que ocorre comumente no algoritmo do TISOL associado à esta arquitetura.

O TISOL deseja medir a resistência entre o ponto P(i), associado a chave X(i) no estado ON, e o conjunto dos pontos do intervalo acima, com a excessão do próprio P(i), quando i estiver contido neste intervalo. Se for constatado que esta resistência é maior do que RIS, existe um princípio físico que garante que a resistência entre o ponto P(i) e cada um dos demais pontos do intervalo também é maior do que RIS, e que, portanto, a rede identificada pelo ponto P(i) está isolada de todas as demais redes contidas no intervalo. Desta forma, pode-se constatar que múltiplas interconexões são do tipo I utilizando apenas uma medida de resistência.

A desabilitação da colisão entre chaves X(i) e Y(i), no bloco ENDY/Ç não é realizada totalmente por hardware, como na versão ENDY/B. Existe participação do software, que, a princípio, verifica se existe esta colisão.

Se a colisão não existir, a desabilitação é desnecessária, e os sinais DES_LIN e DES_COL são desativados.

Se a colisão existir, a medida, que se realizaria em um ciclo, passa a se realizar em dois ciclos. No primeiro ciclo, ativa-se o sinal DES_LIN e as 8 chaves de Y, pertencentes à linha da placa onde existe uma chave de X em ON, são levadas ao estado OFF. No segundo ciclo, ativa-se o sinal DES_COL e as 8 chaves de Y, pertencentes à coluna da placa onde existe uma chave de X em ON, são levadas ao estado OFF.

Unindo o resultado dos dois ciclos, pode-se dizer que a rede representada pelo ponto P(i), cuja chave X(i) está em ON, está isolada de todas as demais redes contidas no intervalo de chaves de Y habilitadas se as duas medidas resultarem em resistências maiores do que RIS. Percebe-se que a única chave de Y no intervalo habilitado que permanece em OFF em ambos os ciclos é aquela da intersecção

entre a linha e a coluna pertencentes a placa onde existe a chave X(i) em ON, isto é, a chave Y(i).

Embora esta técnica para resolver o conflito introduza um ciclo adicional, a redução do custo é bastante significativa em relação à versão ENDY/B, e o número de ciclos adicionais, 1, é bem menor do que seria se a desabilitação do conflito simplesmente não existisse. Trata-se de uma solução intermediária, entre uma opção de alto custo e performance e outra de baixo custo e performance.

4.3.1.4 - Versão ENDY/D

Nesta versão, permite-se que um número e distribuição arbitrários de linhas Y comandem o estado de suas chaves correspondentes para o estado ON. Existe um bit de memória para armazenar o estado de cada uma das n chaves de Y. A figura 4.6 mostra as entradas e saídas deste bloco.

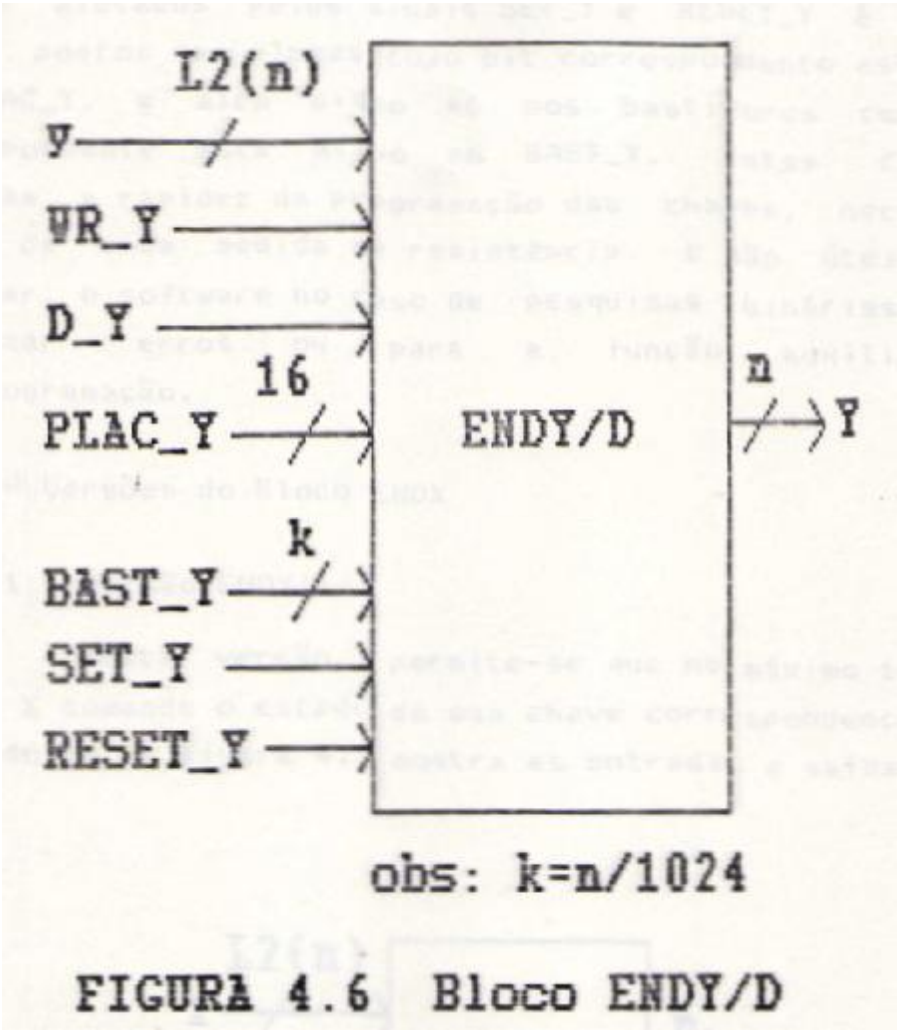

FIGURA 4.6 Bloco ENDY/D

Para alterar o estado de uma chave de Y pode-se colocar seu endereço em y, o valor do estado na linha D_Y e ativar a linha de escrita, WR_Y. Esta operação pode ser repetida quantas vezes for necessário, para configurar o estado de múltiplas chaves.

Os sinais SET_Y e RESET_Y agem sobre um sub-conjunto de chaves de Y. SET_Y leva o sub-conjunto ao estado ON, e RESET_Y leva o sub-conjunto ao estado OFF. Este sub-conjunto é especificado pelos vetores BAST_Y e PLAC_Y, que estão relacionados com a modularização discutida na seção 4.1. BAST_Y possui k bits, tantos quanto

o número de bastidores. PLAC_Y possui 16 bits, tantos quanto o número de placas por bastidor. O sub-conjunto de pontos afetados pelos sinais SET_Y e RESET_Y é formado pelos pontos das placas cujo bit correspondente está ativo em PLAC_Y, e além disso só nos bastidores cujo bit correspondente está ativo em BAST_Y. Estes comandos aumentam a rapidez da programação das chaves, necessária antes de cada medida de resistência, e são úteis para acelerar o software no caso de pesquisas binárias, para localizar erros ou para a função auxiliar de autoprogramação.

4.3.2 - Versões do Bloco ENDX

4.3.2.1 - Versão ENDX/A

Nesta versão, permite-se que no máximo 1 das n linhas X comande o estado da sua chave correspondente para o estado ON. A figura 4.7 mostra as entradas e saídas deste bloco.

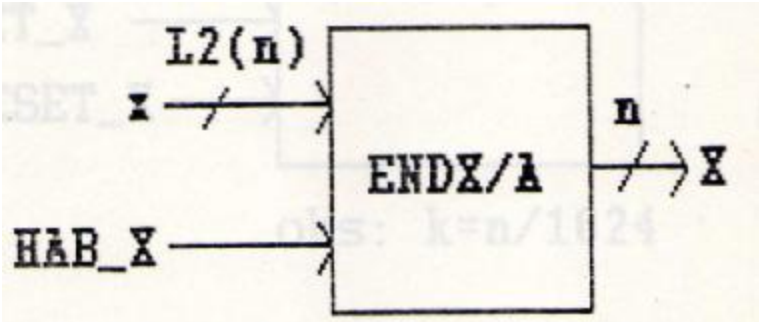

FIGURA 4.7 Bloco ENDX/A

O funcionamento é análogo ao da versão ENDY/A, descrito na seção 4.3.1.1.

4.3.2.2 - Versão ENDX/B

Nesta versão, permite-se que um número e distribuição arbitrários de linhas X comande o estado de suas chaves correspondentes para o estado ON. Existe um bit de memória para armazenar o estado de cada uma das n chaves de X. A figura 4.8 mostra as entradas e saídas deste bloco.

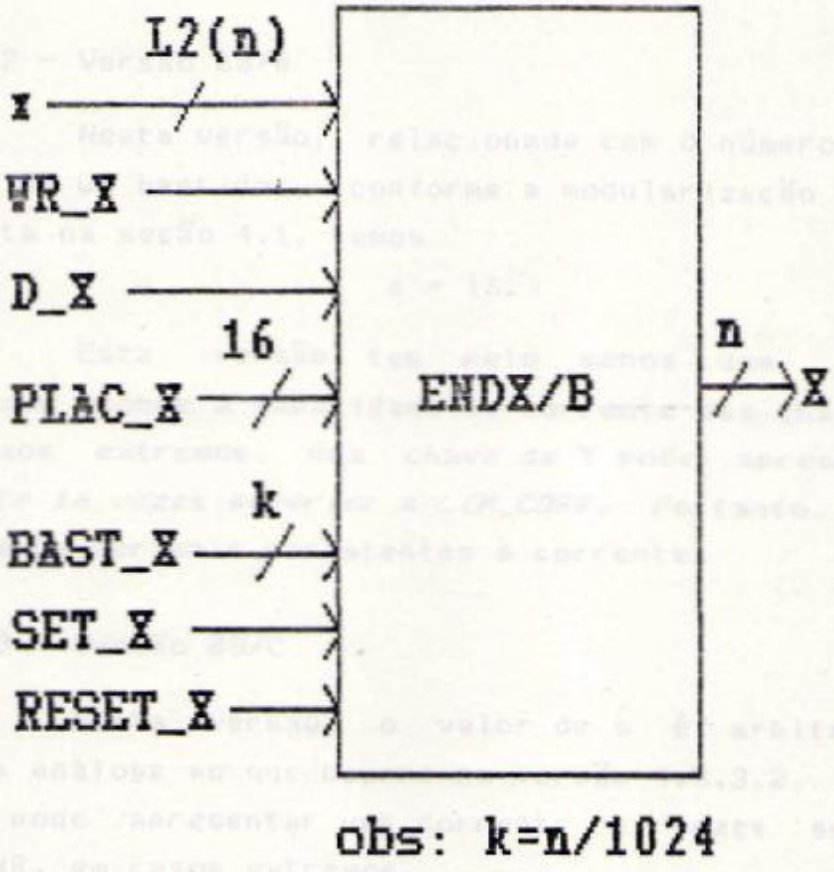

FIGURA 4.8 Bloco ENDX/B

O funcionamento é análogo ao da versão ENDY/D, descrito na seção 4.3.1.4.

4.3.3 - Versões do bloco BS

O bloco BS é um apenas um conjunto de s sensores,

que podem ser compartilhados pelos pontos de teste. Este compartilhamento é regido pelo bloco MUX. Serão definidas 3 versões, correspondentes a 3 valores particulares de s.

4.3.3.1 - Versão BS/A

Nesta versão, que é a mais simples e econômica, temos

$$s = 1.$$

4.3.3.2 - Versão BS/B

Nesta versão, relacionada com o número de placas dentro de um bastidor, conforme a modularização dos pontos descrita na seção 4.1, temos

$$s = 16.$$

Esta versão tem pelo menos uma implicação observada sobre a capacidade de corrente das chaves de Y. Em casos extremos, uma chave de Y pode apresentar uma corrente 16 vezes superior a LIM_CORR. Portanto, as chaves de Y devem ser mais resistentes à corrente.

4.3.3.3 - Versão BS/C

Nesta versão, o valor de s é arbitrário. De maneira análoga ao que ocorre na versão 4.3.3.2, uma chave de Y pode apresentar uma corrente s vezes superior a LIM_CORR, em casos extremos.

4.3.4 - Versões do bloco MUX

4.3.4.1 - Versão MUX/A

Esta versão, sempre associada à versão BS/A de BS, é extremamente simples. Qualquer um dos pontos P(i) pode ser conectado ao sensor único, simplesmente colocando-se em ON a sua chave X(i).

O bloco MUX/A não necessita de nenhum componente

além de simples interconexões para sua implementação. O terminal de saída de cada chave X(i) deve ser interconectado ao terminal de entrada do sensor.

4.3.4.2 - Versão MUX/B

Esta versão, associável à versão BS/C de BS, permite que um sensor S(j), com

$$j = 0, 1, \ldots , s-1,$$

seja conectado a qualquer ponto P(i), com i no intervalo

$$[\ j * (n / s)\ ,\ (j + 1) * (n / s) - 1\ ],$$

colocando-se a chave X(i) no estado ON.

A implementação do bloco MUX/B só utiliza interconexões. Deve-se interconectar o terminal de entrada de cada sensor S(j) aos terminais de saída das

$$n / s$$

chaves de X com endereço no intervalo acima descrito.

4.3.4.3 - Versão MUX/C

Esta versão, associável à versão BS/B de BS, permite que um sensor S(j), com

$$j = 0, 1, \ldots , 15,$$

seja conectado a qualquer ponto P(i), com i no conjunto formado pela união dos seguintes intervalos parametrizados por k:

$$[\ k * 1024 + j * 64\ ,\ k * 1024 + j * 64 + 63\ ],$$

com

$$k = 0, 1, \ldots , n / 1024 - 1.$$

Além disso, deve-se colocar a chave X(i) no estado ON.

A implementação do bloco MUX/C só utiliza interconexões. Deve-se interconectar o terminal de entrada de cada sensor S(j) aos terminais de saída das

$$64 * n / 1024$$

chaves de X no conjunto acima descrito.

Esta versão está relacionada à modularização

descrita na seção 4.1. Um sensor S(j) pode ser conectado aos pontos das placas com posição relativa j dentro de cada bastidor. Uma vantagem clara disto, em relação à versão MUX/B, é que com o aumento do número de pontos o número de sensores permanece constante, em 16. Além disso, basta que haja 1024 pontos para que se possa utilizar todos os sensores. Outras vantagens serão vistas no capítulo 5, ao se estudar um algoritmo associado a esta versão.

4.3.4.4 - Versão MUX/D

Esta versão é associada à versão BS/C de BS, e permite que um sensor S(j), com

$$j = 0, 1, \ldots, s - 1,$$

seja conectado a qualquer ponto P(i), colococando-se a chave X(i) no estado ON e a chave M(i,j) no estado ON.

Esta versão é extremamente cara, pois o multiplexador é formado por um conjunto de

$$n * s$$

chaves M(i,j). Além disso, o estado desta chaves deve ser armazenado, provavelmente em um bit de memória por chave, e deve existir um sistema de endereçamento para programar estas chaves.

A figura 4.9 mostra uma possível implementação deste bloco.

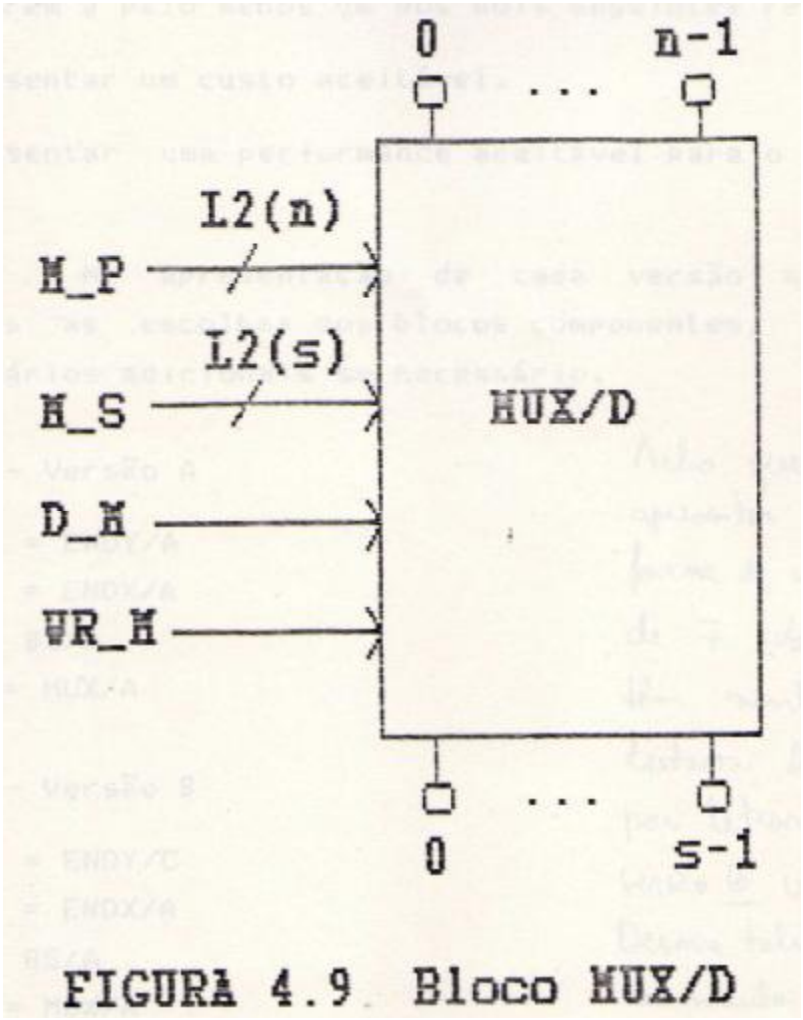

FIGURA 4.9 Bloco MUX/D

Para programar o estado de uma das chaves do multiplexador, especifica-se os endereços M_P e M_S, respectivamente do ponto e do sensor que se deseja conectar, coloca-se em D_M o estado desejado da chave, e ativa-se o sinal de escrita, WR_M. Isto deve ser feito para cada conexão ponto-sensor que se deseje alterar.

4.4 - Arquiteturas Estudadas

Combinando as versões de cada bloco componente variável obtém-se um número limitado de versões de arquiteturas. Dentre estas, muitas não são aplicáveis.

Serão selecionadas, para estudo, apenas aquelas que atenderem a pelo menos um dos dois seguintes requisitos:

- apresentar um custo aceitável.

- apresentar uma performance aceitável para o TCOND e/ou TISOL.

Na apresentação de cada versão apenas serão citadas as escolhas dos blocos componentes, e colocados comentários adicionais se necessário.

4.4.1 - Versão A

- ENDY = ENDY/A
- ENDX = ENDX/A
- BS = BS/A
- MUX = MUX/A

4.4.2 - Versão B

- ENDY = ENDY/C
- ENDX = ENDX/A
- BS = BS/A
- MUX = MUX/A

A versão ENDY/B de ENDY foi abandonada em função da versão ENDY/Ç mais barata e quase tão rápida quanto a versão ENDY/B.

4.4.3 - Versão C

- ENDY = ENDY/D
- ENDX = ENDX/A
- BS = BS/A
- MUX = MUX/A

4.4.4 - Versão D

- ENDY = ENDY/D

- ENDX = ENDX/B
- BS = BS/A
- MUX = MUX/A

4.4.5 - Versão E

- ENDY = ENDY/D
- ENDX = ENDX/B
- BS = BS/C
- MUX = MUX/B

4.4.6 - Versão F

- ENDY = ENDY/D
- ENDX = ENDX/B
- BS = BS/B
- MUX = MUX/C

4.4.7 - Versão G

- ENDY = ENDY/D
- ENDX = ENDX/B
- BS = BS/C
- MUX = MUX/D

5 - ALGORITMOS PARA O TESTE DE INTERCONEXÕES

Cada uma das versões de arquitetura do sistema de aquisição de dados, definidas na seção 4.4, possui um algoritmo associado, que utilizará os recursos da arquitetura para realizar o teste de interconexões. Estes recursos influem decisivamente na complexidade de cada algoritmo e, portanto, no tempo de teste.

Os recursos da arquitetura do sistema de aquisição de dados, do ponto de vista do algoritmo, são comandos primitivos que permitem:

a) configurar parâmetros programáveis descritos entre os da seção 2.2.

b) executar comandos de comutação nas chaves X e Y, também chamados de comandos de endereçamento, através dos sinais de entrada dos blocos ENDX e ENDY.

c) medir a corrente de cada um dos sensores.

O algoritmo é implementado pelo software, residente na memória do controlador, e a arquitetura é implementada pelo hardware do sistema de aquisição de dados, no caso particular desta dissertação. As palavras algoritmo e arquitetura possuem usos e interpretações bastante variados na literatura.

O algoritmo é composto de uma seqüência de instruções executadas pelo controlador. Estas instruções podem acessar os recursos do controlador (memória, periféricos) e também os recursos da arquitetura do sistema de aquisição de dados, através dos seus comandos primitivos. O sistema de aquisição de dados pode ser visto como mais um periférico, cuja arquitetura é de interesse maior do que a do seu controlador nesta dissertação, pois afeta decisivamente a complexidade dos algoritmos do TISOL e do TCOND.

5.1 - Descrição dos Algoritmos

Para descrever os algoritmos utiliza-se, alternadamente:

- linguagem informal.

- linguagem Ç com pequenas alterações:
  - em declarações antecipadas de funções;
  - na representação de algumas constantes numéricas;
  - no uso de um tipo "bool" para variáveis boolenas;

- comandos primitivos do sistema de aquisição de dados, representados conforme se discute a seguir.

Os comandos primitivos do sistema de aquisição de dados obedecerão as notações abaixo:

a) para configurar um parâmetro programável, usa-se a simbologia

PAR = valor

para o comando, onde PAR representa o parâmetro. A programação destes parâmetros foi omitida nos algoritmos, pois não é importante para a análise dos algoritmos e de suas complexidades.

b) para configurar chaves de X e Y, usa-se uma seqüência de comandos do tipo

linha = valor,

onde linha representa um sinal ou vetor de entradas do bloco ENDX ou ENDY. Entre estes comandos pode-se intercalar comandos do tipo

delay(valor em microssegundos),

que representam tempos necessários entre alguns comandos sucessivos. Em cada algoritmo, de acordo com os recursos da arquitetura associada, pode-se estabelecer um conjunto de rotinas compostas destes comandos, para representar em um nível mais alto os comandos de configuração das chaves X e Y.

c) a simbologia

I_S(i),

representa a corrente medida no sensor i.

A representação de valores inteiros (tipo "int") poderá ser decimal (3574), binária (01101b), ou hexadecimal (0xA04F5). Valores booleanos (tipo "bool") poderão ser representados por:

0 = OFF = inativo = FALSE

e

1 = ON = ativo = TRUE.

5.2 - Avaliação do Tempo de Teste

O tempo de teste é proporcional às complexidades dos algoritmos do TCOND e do TISOL. Este tempo pode ser representado aproximadamente pela equação

ccd * T_CIC_CD + cis * T_CIC_IS.

Algumas distorções ocorrem devido a tempos marginais, como:

- o tempo de execução do software do controlador.
- o tempo de execução dos comandos primitivos do sistema de aquisição de dados.

Estes tempos marginais podem ser reduzidos, utilizando-se um controlador de alta performance e circuitos rápidos no sistema de aquisição de dados. Os tempos de ciclo de medida de resistência, T_CIC_CD e T_CIC_IS, serão bem maiores do que estes tempos marginais, se isto for feito. Sobre os tempos de ciclo o projetista do AI não tem controle, pois os mesmos dependem de parâmetros elétricos de cada UST e do seu interface mecânico com o AI.

Por estes motivos, a avaliação dos tempos totais de TCOND e TISOL não considera os tempos marginais. Será dada ênfase à análise de complexidade dos algoritmos, isto é, ao cálculo de ccd e cis, uma vez que o tempo de cada teste é proporcional a sua complexidade, ou número de

ciclos de medida de resistência necessários para concluí-lo. o teste.

O cálculo de complexidades será feito para o caso de não haver defeitos na UST. Defeitos normalmente provocam ciclos de medida adicionais, que serão estimados qualitativamente, ou se possível quantitativamente, para cada algoritmo. Se os limites de erros, LIM_ERROS_COND, LIM_ERROS_ISOL e LIM_ERR_COND_REDE, estiverem programados com valores adequados, o aumento do tempo de teste devido a estes defeitos será pequeno, mesmo para USTs com muitos defeitos.

Para USTs com cargas parasitas preponderantemente capacitivas, espera-se valores de T_CIC_IS bem superiores aos de T_CIC_CD, de forma que, nestes casos, é mais importante otimizar o TISOL. O contrário deve ocorrer para USTs com cargas parasitas preponderamente indutivas. Normalmente, espera-se maiores valores de T_CIC_IS, como no caso de placas nuas de circuito-impresso.

5.3 - Versões de Algoritmos

Cada uma das sete versões de algoritmos será identificada por letras entre A e F, da mesma forma que a arquitetura associada foi identificada na seção 4.4.

5.3.1 - Versão A

5.3.1.1 - Descrição do TCOND/A

O TCOND deve fazer uma varredura em cada uma das redes, objetivando detectar defeitos do tipo circuito-aberto.

A Versão A possui a arquitetura mais simples, que permite, em cada ciclo, medir a resistência entre apenas 2 pontos selecionados.

Para verificar a interconexão entre dois pontos quaisquer, o TCOND/A usa a rotina básica da figura 5.1.

```
/* retorna TRUE se a interconexão é do tipo C */
bool med_cond(end1,end2)

/* endereço dos 2 pontos de teste */
int end1,end2;

{

bool medida;

/* configuração de ENDY/A */
y = end2;  delay(1);  HAB_Y = ON;

/* configuração de ENDX/A */
x = end1;  delay(1);  HAB_X = ON;

/*  espera pela estabilização da medida - este  delay  pode
ser  controlado  por  um  timer,  a  fim  de  liberar  o
processador para outras atividades */
delay(T_CIC_CD);

medida = (I_S(0) > (V_TESTE / RCD));

HAB_X = HAB_Y = OFF;

return(medida);

}
```

FIGURA 5.1 Rotina de medida do TCOND/A

Em cada rede de z pontos, mede-se as

$$z - 1$$

resistências entre o ponto identificador e os demais pontos, uma em cada ciclo. Se todas as interconexões forem do tipo Ç a rede está intacta.

O algoritmo do TCOND/A é apresentado na figura 5.2:

```
/* teste de condução - versão A */
TCOND/A()

{

/*  endereço do ponto identificador da rede  corrente,  que
está sendo verificada */
int rede;

/*  endereço  do ponto da rede corrente  cuja  interconexão
contra   o   ponto   identificador   desta   rede   está   sendo
verificada */
int ponto;

/* endereço do ponto identificador da primeira rede */
int rede_inic;

/*  retorna  TRUE  se  a interconexão entre  os  pontos  de
endereços end1 e end2 é do tipo C - ver figura 5.1 */
bool med_cond(end1,end2);
int end1, end2;

/* rotina que trata erro detectado - ver seção 5.3.1.3 */
int trata_erro(rede,ponto);
int rede, ponto;

/*  função que retorna o endereço da próxima rede,  ou NULL
se esta não existir */
int  prox_rede(rede);
int rede;

/*  função que retorna o endereço do próximo ponto  de  uma
rede, ou NULL se este não existir */
int   prox_ponto(rede,ponto);
int rede, ponto;

/* código */

rede = ponto = rede_inic;

do

   {

     while ((ponto = prox_ponto(rede,ponto)) != NULL)

            if !med_cond(rede,ponto)

                     trata_erro(rede,ponto);

   }

     while ((rede = ponto = prox_rede(rede)) != NULL);

}
```

FIGURA 5.2 Rotina TCOND/A

5.3.1.2 - Complexidade do TCOND/A

O TCOND/A deve testar nr redes. Em cada uma das nr(i) redes de i pontos utiliza-se

$$i - 1$$

ciclos de medidas para verificar se a rede está boa.

O número total de ciclos do TCOND/A é

SOMA(i,1,n,nr(i) * (i-1)),

ou

SOMA(i,1,n,nr(i) * i) - SOMA(i,1,n,nr(i)),

ou, simplesmente, ccd vale:

n - nr.

O pior caso do TCOND/A, isto é, o máximo valor de ccd, ocorre quando

nr = 1,

isto é, quando existe apenas uma rede de n pontos. Por outro lado, neste caso específico o TISOL é desnecessário.

5.3.1.3 - Efeito dos Erros na Complexidade do TCOND/A

A ocorrência de defeitos do tipo circuito-aberto em uma rede provocará, normalmente, ciclos adicionais.

Se o ponto identificador de uma rede de z pontos só estiver conectado a k destes z pontos, existe pelo menos uma rede adicional formada por

z - k

pontos. Portanto, deve-se somar

k - 1

ciclos aos

z - 1

ciclos inicialmente necessários. a fim de resolver a rede adicional. Se a rede adicional de k pontos também estiver sub-dividida em outras redes adicionais, mais ciclos adicionais serão necessários.

No pior caso, em que uma rede de z pontos se quebra em z redes de 1 ponto, o total de ciclos para resolvê-la, encontrando todas as redes adicionais, será

SOMA(i,1,z,z - i)

ou

C(z,2).

Por este motivo, utiliza-se o parâmetro LIM_ERR_COND_REDE, que faz com que sejam ignoradas as redes

adicionais em excesso ao valor deste parâmetro, oriundas de uma mesma rede original. Assim evita-se tempos excessivos no TCOND/A, e listagens de erros muito extensas, quando erros grosseiros deste tipo ocorrem.

5.3.1.4 - Descrição do TISOL/A

O TISOL deve verificar se as nr redes, entre originais e adicionais, estão isoladas entre si. A versão A, seqüencial, verifica, para cada rede, se as interconexões entre ela e as redes que a sucedem são do tipo I.

A figura 5.3 mostra a rotina básica de medida do TISOL/A:

```
/* retorna TRUE se a interconexão é do tipo I */
bool med_isol(end1,end2)

/* endereços dos dois pontos de teste */
int end1,end2;

{

bool medida;

/* configuração de ENDY/A */
y = end2;  delay(1);  HAB_Y = ON;

/* configuração de ENDX/A */
x = end1;  delay(1);  HAB_X = ON;

/*  espera  pela estabilização da medida - este delay  pode
ser   controlado   por   um   timer,   a   fim   de   liberar   o
processador para outras atividades */
delay(T_CIC_IS);

medida = (I_S(0) < (V_TESTE / RIS));

HAB_X = HAB_Y = OFF;

return(medida);

}
```

FIGURA 5.3 Rotina de medida do TISOL/A

O algoritmo de TISOL/A é apresentado na figura 5.4:

```
/* teste de isolação - versão A */
TISOL/A()

{

/*  endereço da rede de referência,  que está sendo testada
contra as demais à procura de curto-circuitos */
int rede;

/*  endereço  da rede cuja interconexão contra  a  rede  de
referência  está sendo verificada */
int  red2;

/* endereço da primeira rede */
int rede_inic;

/*   retorna  TRUE  se  a  interconexão  entre  os   pontos
identificadores  de  redes com endereços end1 e end2  é  do
tipo I - ver figura 5.3 */
boll med_isol(end1,end2);
int end1, end2;

/* rotina que trata erro detectado - ver seção 5.3.1.6 */
int trata_erro(rede,red2);
int rede, red2;

/*  função que retorna o endereço da próxima rede,  ou NULL
se esta não existir */
int  prox_rede(rede);
int rede;

/* código */

rede = red2 = rede_inic;

do

    {
       while ((red2 = prox_rede(red2)) != NULL)

              if !med_isol(rede,red2)

                        trata_erro(rede,red2);

    }

      while ((rede = red2 = prox_rede(rede)) != NULL);

}
```

FIGURA 5.4 Rotina TISOL/A

5.3.1.5 - Complexidade do TISOL/A

Para resolver a i-ésima dentre as nr redes, o TISOL/A deve medir as resistências das

$$nr - i$$

interconexões entre esta rede e as que a sucedem, dedicando um ciclo para cada medida.

Portanto, a complexidade do TISOL/A é

$$SOMA(i,1,nr,nr - i),$$

ou, cis vale:

C(nr,2).

O pior caso do TISOL/A, isto é, o maior valor de cis, ocorre quando

nr = n,

isto é, quando existem n redes de 1 ponto. Por outro lado, neste caso específico o TCOND é desnecessário.

Percebe-se que o TISOL/A tem complexidade quadraticamente proporcional a nr, o que é crítico quando nr é muito grande. USTs típicas podem ter milhares de redes.

5.3.1.6 - Efeito dos Erros na Complexidade do TISOL/A

Se o TISOL/A detectar alguma interconexão do tipo NI entre duas redes, não é necessário nenhum ciclo adicional, pois o erro já está detectado, bastando incluí-lo na estrutura de erros.

5.3.2 - Versão B

5.3.2.1 - Descrição do TCOND/B

O TCOND/B é idêntico ao TCOND/A, descrito em 5.3.1.1. Embora as arquiteturas associadas às duas versões sejam diferentes no que se refere ao bloco ENDY, esta diferença só pode ser usada para otimizar o TISOL.

Durante o TCOND/B, no máximo 1 das n chaves de Y estará no estado ON, mesmo que o bloco ENDY/C tenha capacidade de fazer com que

exp2(i)

das chaves de Y estejam fechadas simultaneamente, com

i = 0, 1, 2, ... ,log2(n).

Apenas a rotina básica de medida exibida na figura 5.1, usada no TCOND/A, deve ser modificada, no que tange ao controle do bloco ENDY/C. A figura 5.5 mostra a

rotina adaptada para o TCOND/B.

```
/* retorna TRUE se a interconexão é do tipo C */
bool med_cond(end1,end2)

/* endereço dos 2 pontos de teste */
int end1,end2;

{

bool medida;

/* configuração de ENDY/C */
y.end = end2;
y.mask = 111111...1111b;
DES_LIN = DES_COL = OFF;
delay(1);  HAB_Y = ON;

/* configuração de ENDX/A */
x = end1;  delay(1);  HAB_X = ON;

/*  espera pela estabilização da medida - este  delay  pode
ser   controlado   por   um   timer,   a   fim   de   liberar   o
processador para outras atividades */
delay(T_CIC_CD);

medida = (I_S(0) > (V_TESTE / RCD));

HAB_X = HAB_Y = OFF;

return(medida);

}
```

FIGURA 5.5 Rotina de medida do TCOND/B

5.3.2.2 - Complexidade do TCOND/B

A análise da seção 5.3.1.2 é pertinente também para o presente caso.

5.3.2.3 - Efeito dos Erros na Complexidade do TCOND/B

A análise da seção 5.3.1.3 é pertinente também para o presente caso.

5.3.2.4 - Descrição do TISOL/B

A simples modificação do bloco ENDY, passando da versão ENDY/A, no TISOL/A, para a versão ENDY/Ç no TISOL/B, pode provocar uma grande otimização.

A diferença entre as duas versões de ENDY é:

- ENDY/A só pode manter em ON uma chave de Y.
- ENDY/C pode manter em ON exp2(i) chaves de Y, com i

variando desde 1 até log2(n).

A vantagem de fechar múltiplas chaves de uma só vez parte de um princípio físico. Se k pontos forem colocados em curto-circuito, mediante o fechamento de suas k chaves de Y, e se for fechada a chave de X de um ponto de referência, o resultado da medida será a associação em paralelo das k resistências entre o ponto de referência e cada um dos k pontos em curto-circuito contra a tensão V_TESTE. Se esta resistência paralela for maior do que RIS, pode-se afirmar que cada um das k resistências que compõem a associação também é maior do que RIS. Conseqüentemente, em uma única medida, pode-se verificar a isolação de uma rede contra k redes. No TISOL/A, k medidas são necessárias para obter o mesmo resultado.

Como o valor de k pode atingir no máximo o valor n, pode-se imaginar que uma redução de n vezes na complexidade é viável, de forma que a complexidade, quadrática no TISOL/A, poderia ser linear no TISOL/B. Isto só não é possível devido a alguns conflitos, que serão discutidos por ocasião da descrição do algoritmo do TISOL/B.

A rotina básica de medida usada no TISOL/B é apresentada na figura 5.6.

```
/* retorna TRUE se a interconexão é do tipo I */
bool med_isol(end1,end2.end,end2.mask,d_lin,d_col)

/* endereços da rede de referência,  com chave de X em   ON
*/
int end1;

/*  endereço do intervalo de pontos (redes) com chaves de Y
em ON - ver seção 4.3.1.3 - ENDY/C */
int end2.end, end2.mask;

/* especificam ciclos com linha e coluna desabilitada - ver
seção 4.3.1.3 */
bool d_lin, d_col;

{

bool medida;

/* configuração de ENDY/C */
y.end = end2.end;  y.mask = end2.mask;
DES_LIN = d_lin; DES_COL = d_col;
delay(1);  HAB_Y = ON;

/* configuração de ENDX/A */
x = end1;  delay(1);  HAB_X = ON;

/*  espera  pela estabilização da medida - este delay  pode
ser   controlado   por   um   timer,   a   fim   de   liberar   o
processador para outras atividades */
delay(T_CIC_IS);

medida = (I_S(0) < (V_TESTE / RIS));

HAB_X = HAB_Y = OFF;

return(medida);

}
```

FIGURA 5.6 Rotina de medida do TISOL/B

O TISOL/B é descrito na figura 5.7.

```
/* teste de isolação - versão B */
TISOL/B()

{

/* endereço da rede de referência, que está sendo testada
contra as demais a procura de curto-circuitos; */
int rede;

/* intervalo de endereços das redes cuja interconexão
paralela contra a rede de referência está sendo verificada
*/
int  red2.end, red2.mask;

/* endereço da primeira rede */
int rede_inic;

/* função que retorna o endereço da próxima rede, ou NULL
se esta não existir */
int  prox_rede(rede);
int rede;

/* função que calcula o valor inicial de red2.end e
red2.mask, para um dado valor de rede; este valor inicial
deve especificar o menor intervalo endereçável de exp2(i)
pontos, que englobe o intervalo [rede + 1, n - 1], isto é,
todos os pontos que sucedem rede; os parâmetros red2.end e
red2.mask são pointers (call by name), pois seu valor é
alterado dentro da rotina.
*/
void inic_red2(rede,red2.end,red2.mask);
int rede, *red2.end, *red2.mask;

/* função que verifica se a rede de referência está isolada
de todas as redes no intervalo [rede + 1 , n - 1], contido
no intervalo especificado pelos valores iniciais de
red2.end e red2.mask, e coloca as fugas encontradas na
estrutura de erros; */
void ciclo_rede(rede,red2.end,red2.mask);
int rede, red2.end, red2.mask;


/* código */

rede = rede_inic;

do

    {

       inic_red2(rede,&red2.end,&red2.mask);

       ciclo_rede(rede,red2.end,red2.mask);

    }

      while ((rede = prox_rede(rede)) != NULL);

}
```

FIGURA 5.7 Rotina TISOL/B

Na figura 5.7 fica claro que a rotina ciclo_rede é chamada nr vezes, uma para cada rede original ou adicional. A rotina ciclo_rede será descrita em linguagem informal. Esta rotina deve verificar se a rede de referência (rede) está isolada de todas as redes que lhe

sucedem, contidas no intervalo

[rede + 1, n - 1].

Este intervalo, na primeira medida da rotina ciclo_rede, deve ser englobado pelo menor intervalo endereçável estipulado pelos valores iniciais dos parâmetros red2.end e red2.mask, calculados pela rotina inic_red2. O intervalo endereçado (no estado ON) de Y pode ser maior do que o intervalo das redes sucessoras, uma vez que exp2(i) pontos são endereçados por ENDY/Ç e o primeiro ponto deste intervalo deve ser múltiplo de exp2(i), conforme visto na seção 4.3.1.3.

A princípio, parece possível que a rotina ciclo_rede utilize apenas uma medida para verificar que a resistência paralela entre a rede de referência e cada ponto do intervalo endereçado de Y é maior do que RIS, e que portanto todas as interconexões entre a mesma e cada um dos pontos do intervalo são do tipo I.

Infelizmente, existem colisões que impedem que isto ocorra. No intervalo especificado por red2.end e red2.mask podem estar contidos pontos pertencentes à rede de referência, inclusive o seu ponto identificador, cujo endereço está no parâmetro rede. Se isto ocorrer, como normalmente ocorre, a medida retornada pela rotina med_isol, da figura 5.6, obviamente indicará fuga, e nem precisa ser realizada para se chegar a essa conclusão.

Mecanismos para desabilitar a chave de Y correspondente ao ponto identificador da rede de referência foram introduzidos, conforme visto na seção 4.3.1.3. Estes mecanismos implicam no uso de um ciclo de medida adicional, sempre que este ponto estiver contido no intervalo de chaves de Y habilitadas. No primeiro ciclo, temos

DES_LIN = ON e DES_COL = OFF.

Se o resultado desta medida for interconexão do tipo I, deve-se realizar o segundo ciclo, com

DES_LIN = OFF e DES_COL = ON.

Se o resultado do segundo ciclo também for interconexão do tipo I, garante-se que a rede de referência está isolada de todos os pontos do intervalo

[red2.end & red2.mask , red2.end | ~red2.mask],

com excessão do próprio ponto identificador da rede de referência. Caso este ponto não pertença ao intervalo acima, apenas um ciclo de medida, com

DES_LIN = OFF e DES_COL = OFF,

é necessário.

Este mecanismo, por questões de custo, só permite a desabilitação simultânea de um único ponto do intervalo endereçado de Y, o ponto identificador da rede de referência, cuja chave de X está em ON.

Quando dentro do intervalo endereçado de Y houver outros pontos da rede de referência, o que ocorre quando a rede é formada por mais do que um ponto, vários ciclos adicionais podem ser necessários para garantir que a rede de referência está isolada de todos os pontos deste intervalo, excetuando os pontos que lhe pertencem.

Se dentro do intervalo endereçado de Y existir apenas o ponto identificador da rede de referência, o mecanismo de desabilitação do conflito, através de 2 ciclos, manipulando as linhas DES_LIN e DES_COL, pode ser utilizado. Se dentro do intervalo existir um ou mais pontos da rede de referência, que não o ponto identificador, a medida nem precisa ser realizada, pois o resultado óbvio é uma indicação de fuga, ou interconexão do tipo NI. Neste caso, a medida não é realizada e o intervalo é quebrado em duas metades. Isso ocorre sucessivamente, configurando uma pesquisa sobre uma árvore binária. Cada nodo desta árvore binária representa um intervalo de exp2(i) chaves de Y no estado ON, e corresponde a zero, um ou dois ciclos medida. Antes de testar um nodo, deve-se verificar as condições abaixo, na seqüência indicada:

a) se dentro do intervalo endereçado de Y existe pelo menos um ponto identificador de uma rede diferente da rede de referência, e que também seja maior do que o ponto identificador da rede de referência. Se nenhum existe, não é necessário testar este nodo nem os seus nodos filhos, pois neles não existem redes sucessoras da rede de referência.

b) se dentro do intervalo endereçado de Y não existem pontos pertencentes à rede de referência, que não o ponto identificador desta rede. Se existir algum, o resultado da medida é previamente conhecido, ou seja, interconexão do tipo NI, e a medida não deve ser realizada, devendo-se migrar para o nodo filho à esquerda, se existe algum, e voltar a verificar estas condições, a partir da "a)", antes de executar a medida.

c) se dentro do intervalo existe o ponto identificador da rede de referência, e nenhum outro ponto da rede de referência, a medida correspondente ao nodo pode ser desdobrada em duas medidas, utilizando as linhas DES_LIN e DES_COL para desabilitar o conflito, como já discutido acima. Se na primeira medida for constatada uma interconexão do tipo NI, a segunda medida é desnecessária, pois o primeiro ciclo já constatou uma fuga.

d) se nenhuma das condições acima for verdadeira, uma única medida, com DES_LIN e DES_COL no estado OFF, deve ser realizada.

Se o TISOL/B verifica que o resultado da medida em um nodo da arvore binária, após ter avaliado a condição "d)", ou que o resultado de pelo menos uma de duas medidas, após ter avaliado a condição "c)", é uma interconexão do tipo NI, deve-se percorrer a seguir os seus dois nodos filhos, começando pelo da esquerda. O objetivo é isolar os pontos identificadores de redes contra as quais existe fuga. Em caso contrário, quando uma interconexão do tipo I

é constatada em um nodo, não é necessário percorrer seus nodos filhos.

Se o TISOL/B chega até um nodo folha, que endereça apenas uma chave de Y e corresponde a um ponto identificador de uma rede diferente da rede de referência, e o resultado da medida neste nodo é NI, uma fuga foi constatada entre a rede de referência e a rede correspondente a este nodo folha.

A figura 5.8 mostra o exemplo de uma UST sem defeitos, idêntica ao PADRÃO, com n = 16. Esta UST possui 6 redes, identificadas pelos pontos 0, 1, 2, 3, 7 e 12.

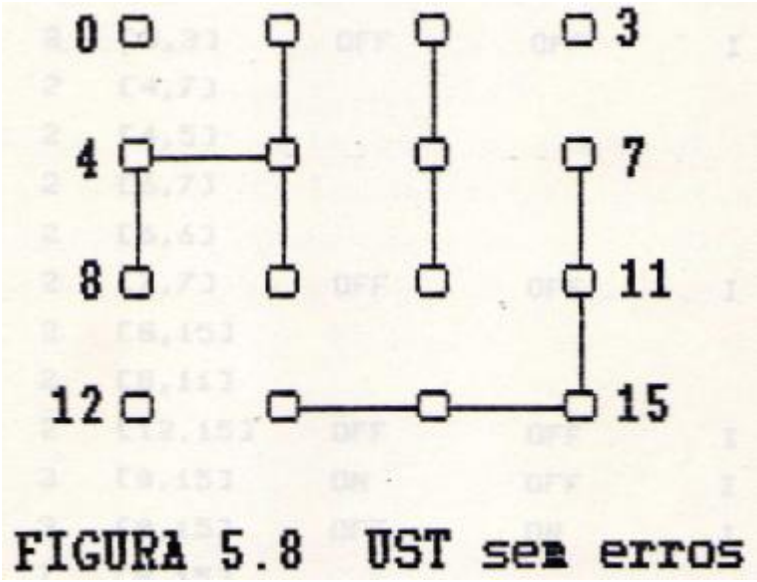

FIGURA 5.8 UST sem erros

A figura 5.9 mostra os ciclos e o valor das principais linhas de entrada de ENDX/A e ENDY/C durante o TISOL/B para confirmar que esta UST não apresenta fugas. São mostrados também os ciclos cancelados, com zero medidas associadas, devido a não verificação das condições "a)" ou "b)", acima descritas. A observação "ca" ou "cb", respectivamente, substitui o número de ordem dos ciclos cancelados.

ciclo	x	y	DES_LIN	DES_COL	medida
1	0	[0,15]	ON	OFF	I
2	0	[0,15]	OFF	ON	I
cb	1	[0,15]			
cb	1	[0,7]			
3	1	[2,3]	OFF	OFF	I
cb	1	[4,7]			
ca	1	[4,5]			
4	1	[6,7]	OFF	OFF	I
cb	1	[8,15]			
ca	1	[8,11]			
5	1	[12,15]	OFF	OFF	I
cb	2	[0,15]			
cb	2	[0,7]			
6	2	[3,3]	OFF	OFF	I
cb	2	[4,7]			
ca	2	[4,5]			
cb	2	[6,7]			
ca	2	[6,6]			
7	2	[7,7]	OFF	OFF	I
cb	2	[8,15]			
ca	2	[8,11]			
8	2	[12,15]	OFF	OFF	I
9	3	[0,15]	ON	OFF	I
10	3	[0,15]	OFF	ON	I
cb	7	[8,15]			
ca	7	[8,11]			
cb	7	[12,15]			
cb	7	[12,13]			
11	7	[12,12]	OFF	OFF	I
ca	7	[13,13]			
ca	7	[14,15]			

FIGURA 5.9 Traçado de um exemplo do TISOL/B

5.3.2.5 - Complexidade do TISOL/B

Pode-se verificar, a partir do exemplo das

figuras 5.8 e 5.9, que a distribuição particular dos pontos em redes no PADRÃO afeta a complexidade. Isto é, a complexidade exata do TISOL/B não é função apenas das variáveis n e nr, mas também da distribuição de pontos.

Isto torna impossível, a não ser por simulação, avaliar a complexidade exata do TISOL/B para cada uma das inúmeras UST imagináveis. Um simulador foi desenvolvido para a versão B, conforme comentado no capítulo 6, e pode produzir um saída semelhante a da figura 5.9.

É possível, no entanto, estabelecer a complexidade do pior caso, utilizando os parâmetros n e nr. O resultado será exagerado para cima, mas serve para estabelecer um limite superior para a complexidade. Utilizando as n variáveis nr(i), com

i = 1, 2, ... , n,

pode-se calcular este limite superior com maior precisão. Se toda a descrição da UST for fornecida, pode-se calcular a complexidade exata, através de simulação. As opções que utilizam apenas os valores n e nr, ou os valores n e nr(i), possibilitam a avaliação do limite superior através de equações matemáticas.

Em primeiro lugar, será analisada a equação de cis em função de n e nr(i).

Cada rede de i pontos leva, no pior caso, k(i,n) ciclos para ser resolvida. Com a ajuda do simulador, e analisando vários exemplos, chegou-se a uma equação recursiva:

k(1,n) = 2

k(i,n) = k(i - 1,n) + int(log2(n / (i - 1))) - 1.

Desta forma:

cis = SOMA(i,1,n,nr(i) * k(i,n)),

ou

cis = SOMA(i,1,n,np(i) * (k(i,n) / i)).

A relação

$$k(i,n) / i$$

pode ser vista como o número de ciclos médio para resolver cada ponto de redes de i pontos.

Se, para um determinado valor de n, for plotada a função

$$k(i,n) / i,$$

verifica-se que ela é limitada, isto é, possui um valor máximo finito para um ou mais valores de i.

O valor de i onde este máximo ocorre (admite-se que seja único) é chamado de imax, e é função de n.

A tabela 5.1 mostra estes máximos, para alguns valores de n.

TABELA 5.1 Ciclos para resolver uma rede via TISOL/B

n	imax	k(imax,n) / imax
16	2	2,5
64	3	3,67
256	4	5,0
1024	5	6,6
16384	9	9,89

Pode-se provar que o valor de cis é maximizado quando

$$np(imax) = n$$

e

$$np(i) = 0,$$

para todo i diferente de imax. Esta prova não foi incluída no texto.

Desta forma, no pior caso do TISOL/B, o valor de cis é:

$$n * k(imax,n) / imax.$$

Utilizando a tabela 5.1 para um ajuste de curvas, obtém-se

k(imax,n) / imax = 0,73 * log2(n) - 0,57.

Então, o valor máximo de cis, em função de n, é aproximadamente

0,73 * n * log2(n) - 0,57 * n.

Comparando a complexidade deste pior caso, que é exagerado, com a complexidade do TISOL/A, que é exatamente

C(n,2),

para, por exemplo, 16384 pontos de teste, chegamos ao seguinte resultado:

- pior caso do TISOL/A: 134.209.536 ciclos.
- pior caso do TISOL/B: 158.106 ciclos.

5.3.2.6 - Efeito dos Erros na Complexidade do TISOL/B

Sempre que um defeito do tipo fuga ocorrer, ciclos adicionais serão gastos até chegar ao nodo folha correspondente ao ponto identificador da rede que está em fuga contra a rede de referência.

O número máximo de ciclos adicionais devidos a uma única fuga, no caso particular de a rede de referência possuir apenas um ponto, é

2 * log2(n).

Se a rede de referência possuir mais do que um ponto, o número de ciclos adicionais devido a uma fuga contra ela não excede o valor acima.

5.3.3 - Versão C

5.3.3.1 - Descrição do TCOND/C

O TCOND/C é idêntico ao TCOND/A, descrito em 5.3.1.1. Embora as arquiteturas associadas às duas versões

sejam diferentes no que se refere ao bloco ENDY, esta diferença só pode ser usada para otimizar o TISOL.

Durante o TCOND/Ç no máximo 1 das n chaves de Y estará no estado ON, mesmo que o bloco ENDY/D tenha capacidade de fazer com que um número e distribuição arbitrários destas chaves estejam fechadas simultaneamente.

Apenas a rotina básica de medida exibida na figura 5.1, usada no TCOND/A, deve ser modificada, no que tange ao controle do bloco ENDY/D. A figura 5.10 mostra a rotina adaptada para o TCOND/C.

```
/* retorna TRUE se a interconexão é do tipo C */
bool med_cond(end1,end2)

/* endereço dos 2 pontos de teste */
int end1,end2;

{

bool medida;

/* configuração de ENDY/D */
y = end2;  D_Y = ON;
delay(1);  WR_Y = ON;
delay(1);  WR_Y = OFF;

/* configuração de ENDX/A */
x = end1;  delay(1);  HAB_X = ON;

/*  espera pela estabilização da medida - este delay pode
ser  controlado  por  um  timer,  a  fim  de  liberar  o
processador para outras atividades */
delay(T_CIC_CD);

medida = (I_S(0) > (V_TESTE / RCD));

y = end2;  D_Y = OFF;
delay(1);  WR_Y = ON;
delay(1);  WR_Y = OFF;

HAB_X = OFF;

return(medida);

}
```

FIGURA 5.10 Rotina de medida do TCOND/C

5.3.3.2 - Complexidade do TCOND/C

A análise da seção 5.3.1.2 é pertinente também para o presente caso.

5.3.3.3 - Efeito dos Erros na Complexidade do TCOND/C

A análise da seção 5.3.1.3 é pertinente também para o presente caso.

5.3.3.4 - Descrição do TISOL/C

Conforme foi visto na descrição do TISOL/B, em 5.3.2.4, um princípio físico foi utilizado para reduzir a complexidade quadrática do TISOL/A. Este princípio estabelece que, se a associação paralela de k resistências é menor do que RIS, então cada uma das k resistências também é menor do que RIS. Isto possibilitaria uma redução de k vezes na complexidade do TISOL/A, sendo que o valor de k poderia atingir o valor de n, o que tornaria linear a complexidade do TISOL/B. Algumas colisões, mencionadas na descrição do TISOL/B, impedem que isso aconteça.

No TISOL/Ç estas colisões não mais ocorrem, e o algoritmo realmente é linearizado. Uma vez que o bloco ENDY/D possibilita o endereçamento simultâneo de um número arbitrário de chaves de Y, apenas aquelas correspondentes a pontos identificadores de redes sucessoras da rede de referência são levadas ao estado ON. Desaparece o conflito existente no TISOL/B devido a pontos pertencentes à rede de referência contidos no intervalo endereçado de Y.

O bloco ENDY/D possui um bit de memória para manter a programação de cada chave de Y. Portanto, a programação destas chaves é feita de maneira similar a um acesso à memória. Pode-se acessar cada bit individualmente, através das linhas y, D_Y e WR_Y, ou grupos de bits, através das linhas BAST_Y, PLAC_Y, SET_Y e RESET_Y, conforme visto na seção 4.3.1.4.

O TISOL/C é descrito na figura 5.11.

```
/* teste de isolação - versão C */
TISOL/C()

{

/* endereço da rede de referência,  que está sendo  testada
contra as demais à procura de curto-circuitos; */
int rede;

/* endereço da primeira rede */
int rede_inic;

/*  função que retorna o endereço da próxima rede,  ou NULL
se esta não existir */
int  prox_rede(rede);
int rede;

/* função que verifica se a rede de referência está isolada
de todas as redes no intervalo [rede + 1 , n - 1], e coloca
as fugas encontradas na estrutura de erros; */
void ciclo_rede(rede);
int rede;

/* código */

rede = rede_inic;

do

     ciclo_rede(rede);

          while ((rede = prox_rede(rede)) != NULL);

}
```

FIGURA 5.11 Rotina TISOL/C

Na figura 5.11 fica claro que a rotina ciclo_rede é chamada nr vezes, uma para cada rede original ou adicional.

A rotina ciclo_rede será descrita abaixo, em linguagem informal. Esta rotina deve verificar se a rede de referência (rede) está isolada de todas as redes que lhe sucedem, contidas no intervalo

[rede + 1, n - 1].

Os pontos identificadores de redes contidos no intervalo acima têm suas chaves de Y levadas ao estado ON, o mesmo ocorrendo com a chave de X do ponto identificador da rede de referência. A medida representa a resistência paralela entre a rede de referência e cada uma das redes que a sucedem. Se o resultado for interconexão do tipo I, a resolução da rede de referência está completa.

5.3.3.5 - Complexidade do TISOL/C

Pode-se verificar, a partir da figura 5.11, que a rotina ciclo_rede é chamada nr vezes, gastando um único ciclo de medida em cada chamada, se não existirem erros. Para a última rede, a medida é desnecessária, uma vez que não existe nenhuma rede sucessora.

Portanto, para o TISOL/Ç cis vale:

nr - 1.

5.3.3.6 - Efeito dos Erros na Complexidade do TISOL/C

Se uma determinada rede estiver em fuga contra outras, são necessários ciclos adicionais para detectar estas fugas. Se a i-ésima rede for a rede de referência, deve-se descobrir quais das suas

nr - i

redes sucessoras estão em fuga contra ela. Para tanto, uma pesquisa binária é feita sobre estas redes sucessoras. O número de ciclos adicionais, devido a uma única fuga, é

2 * log2(nr - i).

5.3.4 - Versão D

5.3.4.1 - Descrição do TCOND/D

O TCOND/D é idêntico ao TCOND/A, descrito em 5.3.1.1. Embora as arquiteturas associadas às duas versões sejam diferentes no que se refere aos blocos ENDX e ENDY, estas diferenças só podem ser usadas para otimizar o TISOL.

Durante o TCOND/D, no máximo 1 das n chaves de X e de Y estarão no estado ON, mesmo que os blocos ENDX/B e ENDY/D tenham capacidade de fazer com que um número e distribuição arbitrários destas chaves estejam fechadas simultaneamente.

Apenas a rotina básica de medida exibida na

figura 5.1, usada no TCOND/A, deve ser modificada no que tange ao controle dos blocos ENDX/B e ENDY/D. A figura 5.12 mostra a rotina adaptada para o TCOND/D.

```
/* retorna TRUE se a interconexão é do tipo C */
bool med_cond(end1,end2)

/* endereço dos 2 pontos de teste */
int end1,end2;

{

bool medida;

/* configuração de ENDY/D */
y = end2;  D_Y = ON;
delay(1);  WR_Y = ON;
delay(1);  WR_Y = OFF;

/* configuração de ENDX/B */
x = end1;  D_X = ON;
delay(1);  WR_X = ON;
delay(1);  WR_X = OFF;

/*  espera pela estabilização da medida - este  delay  pode
ser   controlado   por   um   timer,   a   fim   de   liberar   o
processador para outras atividades */
delay(T_CIC_CD);

medida = (I_S(0) > (V_TESTE / RCD));

y = end2;  D_Y = OFF;
delay(1);  WR_Y = ON;
delay(1);  WR_Y = OFF;

x = end1;  D_X = OFF;
delay(1);  WR_X = ON;
delay(1);  WR_X = OFF;

return(medida);

}
```

FIGURA 5.12 Rotina de medida do TCOND/D

5.3.4.2 - Complexidade do TCOND/D

A análise da seção 5.3.1.2 é pertinente também para o presente caso.

5.3.4.3 - Efeito dos Erros na Complexidade do TCOND/D

A análise da seção 5.3.1.3 é pertinente também para o presente caso.

5.3.4.4 - Descrição do TISOL/D

A versão D possui uma arquitetura mais poderosa

do que a versão Ç no que tange ao endereçamento das chaves de X. O bloco ENDX/B é estruturalmente idêntico ao bloco ENDY/D, isto é, um número arbitrário de chaves de X e de Y podem ser levadas simultaneamente ao estado ON. A seguir será descrito como o TISOL/D explora este recurso adicional para diminuir ainda mais a complexidade linear do TISOL/C.

Chama-se de r(i) a i-ésima rede, com

$$i = 1, 2, 3, \ldots, nr.$$

O TISOL deve verificar se a interconexão entre cada dupla de redes r(i) e r(j) é do tipo I, com i diferente de j.

O algoritmo TISOL/D, em cada ciclo de medida, divide as nr redes em 2 conjuntos de

$$nr/2$$

redes. As chaves de X associadas aos pontos identificadores de redes de um destes conjuntos são levadas ao estado ON, e o mesmo ocorre com as chaves de Y associadas aos pontos identificadores de redes do outro conjunto. Desta forma, em cada medida, verifica-se a resistência entre os 2 conjuntos de redes. Se a resistência for maior do que RIS, pode-se concluir que cada uma das redes de um dos conjuntos está isolada de cada uma das redes do outro conjunto. Isto corresponde à verificação simultânea de

$$(nr / 2) * (nr / 2)$$

interconexões, superando o grau de paralelismo do TISOL/Ç que verifica no máximo

$$nr - 1$$

interconexões simultaneamente. O grau de paralelismo do TISOL/D não corresponde à redução de complexidade em relação a versão seqüencial, TISOL/A, pois o TISOL/D repete a verificação de algumas interconexões ao longo de seus ciclos de medida.

Para verificar se as redes r(i) e r(j) estão isoladas entre si, deve-se colocar a rede r(i) em um dos 2 conjuntos, e a rede r(j) no outro conjunto. Devem ser realizadas tantas medidas tal que esta condição seja

satisfeita para cada dupla de i e j, com i diferente de j.

O exemplo da figura 5.13, com 8 redes, mostra a execução de TISOL/D.

ciclo	X	Y
1	r(1)-r(2)-r(3)-r(4)	r(5)-r(6)-r(7)-r(8)
2	r(1)-r(2) r(7)-r(8)	r(5)-r(6) r(3)-r(4)
3	r(1) r(6) r(7) r(4)	r(5) r(2) r(3) r(8)

FIGURA 5.13 Traçado de um exemplo do TISOL/D

Nota-se que em cada ciclo troca-se de posição nr/4 redes de um conjunto com nr/4 redes do outro conjunto. A posição relativa das redes que migram de um conjunto para o outro, que é a mesma nos dois conjuntos, é determinada por uma divisão binária sucessiva dos conjuntos ao longo dos ciclos. As redes que migram são aquelas que ficam à direita do eixo de partição binária, indicado na figura 5.13 pelo desaparecimento do hífen em relação ao ciclo anterior.

5.3.4.5 - Complexidade do TISOL/D

Através do exemplo da figura 5.13, pode-se verificar que o algoritmo TISOL/D aplica divisões binárias sucessivas até que todos os sub-grupos possuam apenas uma rede. Por esse motivo, a complexidade do TISOL/D, cis, vale:

$$\log_2(nr).$$

5.3.4.6 - Efeito dos Erros na Complexidade do TISOL/D

Se em um dos ciclos de medida constata-se uma resistência menor do que RIS, é necessário isolar as redes culpadas por esta fuga. Isto significa que pelo menos uma das nr/2 redes de um dos 2 conjuntos está com fuga contra pelo menos uma das nr/2 redes do outro conjunto.

Supondo que exista apenas 1 fuga, entre 1 rede de

um conjunto e 1 rede do outro conjunto, é necessário realizar 2 pesquisas binárias, uma em cada conjunto, para isolar estas 2 redes. Cada pesquisa binária utiliza

$$2 * \log2(nr / 2)$$

ciclos adicionais devido a esta fuga, totalizando

$$4 * \log2(nr / 2)$$

ciclos adicionais para isolar as 2 redes.

5.3.5 - Versão E

5.3.5.1 - Descrição do TCOND/E

Os algoritmos do TCOND, desde TCOND/A até TCOND/D, não apresentaram nenhuma evolução. As mudanças de arquitetura nestas versões ocorreram apenas nos blocos ENDX e ENDY, o que permitiu apenas a otimização do TISOL. A partir da versão E, até a versão G, serão verificadas otimizações no TCOND, devido a mudanças nos blocos BS e MUX. Por outro lado, estas mudanças não afetarão a performance do TISOL, a não ser no caso de existirem defeitos do tipo fuga na UST.

Para a otimização do TISOL, utilizou-se um princípio físico: a associação paralela de resistores resulta menor do que o menor dos seus resistores componentes. Para o TCOND, poderia-se pensar na aplicação do princípio físico dual deste: a associação em série de resistores resulta maior do que o maior dos seus resistores componentes. Entretanto, o princípio de medida, que utiliza duas chaves para cada ponto, e o desconhecimento da geometria da rede, conforme discutido na seção 2.3, impedem a utilização deste princípio físico para o TCOND.

No TCOND, o grau de paralelismo pode ser aumentado se várias medidas forem feitas simultaneamente, o que é possível se houver múltiplos sensores. O número total de medidas continua sendo o mesmo da versão TCOND/A, mas se existem s sensores, a complexidade e o tempo podem ser

reduzidos s vezes, se em cada ciclo todos os s sensores forem de fato utilizados, sem redundância.

Uma das opções para paralelizar o TCOND é processar em paralelo diversas redes, utilizando um sensor em cada rede. Infelizmente, um grave problema dificulta a implementação segura desta opção. É necessário, antes de realizar o processamento paralelo de s redes, ter certeza de que não existem fugas entre estas s redes. A figura 5.14 mostra um PADRÃO e uma UST, contendo duas redes cujo processamento em paralelo causaria problemas.

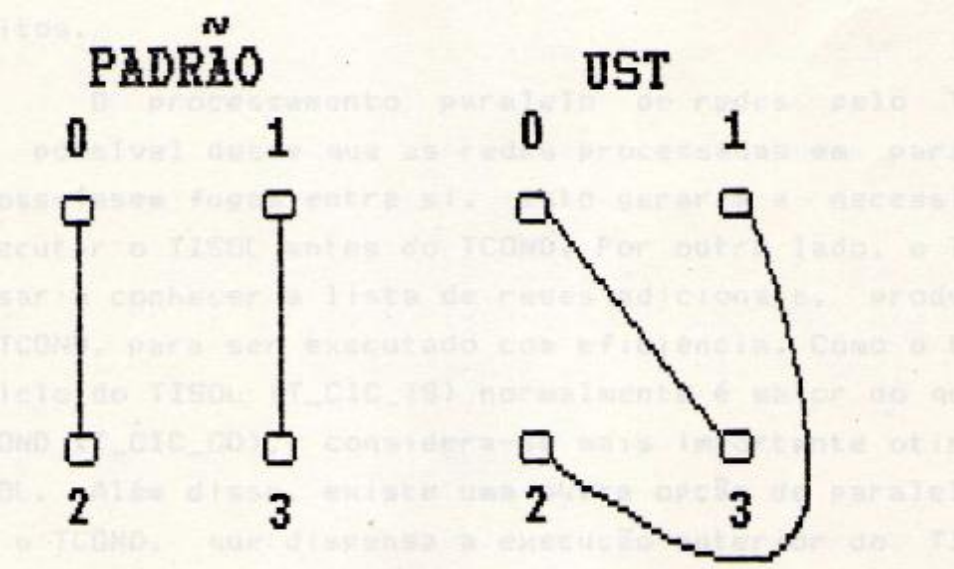

FIGURA 5.14 Redes com defeitos cruzados

Para testar as duas redes em paralelo, aloca-se o sensor S(0) para a rede 0 e o sensor S(1) para a rede 1. Fecha-se as chaves X(0) e Y(2) e mede-se a corrente em S(0), para determinar a resistência entre os pontos 0 e 2. Fecha-se também as chaves X(1) e Y(3) e mede-se a corrente em S(1), para determinar a resistência entre os pontos 1 e 3. Ocorre um imprevisto. As correntes medidas são altas o suficiente, indicando que as interconexões são do tipo C,

quando na verdade ambas são do tipo NC. A corrente de S(0), ao invés de passar pelas chaves Y(2) e X(0), passa pelas chaves Y(3) e X(0). A corrente de S(1), ao invés de passar pelas chaves Y(3) e X(1), passa pelas chaves Y(2) e X(1).

Além de não detectar os dois defeitos do tipo circuito-aberto, não serão detectadas as duas fugas. Isto porque as duas redes adicionais, 2 e 3, não serão passadas para o TISOL, que portanto não poderá detectar as fugas em que estas redes adicionais se envolvem. Como resultado final, o teste de interconexões diria que a UST está perfeita, embora existam 2 circuitos-abertos e 2 curto-circuitos.

O processamento paralelo de redes pelo TCOND seria possível desde que as redes processadas em paralelo não possuíssem fugas entre si. Isto geraria a necessidade de executar o TISOL antes do TCOND. Por outro lado, o TISOL precisaria conhecer a lista de redes adicionais, produzida pelo TCOND, para ser executado com eficiência. Como o tempo de ciclo do TISOL (T_CIC_IS) normalmente é maior do que o do TCOND (T_CIC_CD), considera-se mais importante otimizar o TISOL. Além disso, existe uma outra opção de paralelismo para o TCOND, que dispensa a execução anterior do TISOL. Por estes motivos, mantém-se a decisão de executar o TISOL depois do TCOND, conforme estabelecido no DFD da figura 3.1.

A outra opção de paralelismo no TCOND, que será utilizada nas versões E, F e G, consiste em realizar múltiplas medidas, simultaneamente, em uma única rede.

Para resolver uma rede, fecha-se a chave de Y correspondente ao ponto identificador da rede e fecha-se até s chaves de X, correspondentes a s outros pontos da rede, em cada ciclo de medidas. Cada uma destas s chaves de X deve conectar o seu ponto correspondente a um dos s sensores. Cada sensor deve ser conectado no máximo a um ponto. As s leituras de corrente em cada ciclo correspondem

à determinação simultânea de s interconexões.

A conectabilidade entre um sensor e um ponto é determinada pelo bloco MUX. Normalmente, cada sensor pode ser conectado a pontos contidos em um intervalo de endereços. Portanto, se em uma rede particular de z pontos, todos os

$$z - 1$$

pontos, que não o ponto identificador, se concentrarem em um único intervalo de endereços, o que permite a sua conexão a apenas um sensor, então os demais

$$s - 1$$

sensores não poderão ser utilizados, e de nada servirão. O ideal seria que os pontos da rede se distribuíssem uniformemente sobre os intervalos de alocação dos s sensores, permitindo que todos fossem usados em cada ciclo.

O algoritmo descrito nesta seção é válido para as versões E, F e G. As variações do bloco MUX nestas 3 versões afetarão as complexidades destes algoritmos.

5.3.5.2 - Complexidade do TCOND/E

Para resolver uma rede de z pontos, são necessários no mínimo

$$rds((z - 1) / s)$$

ciclos, com s medidas em cada ciclo, quando todos os sensores puderem ser utilizados em cada ciclo.

Na versão MUX/B, um sensor S(j) pode ser alocado por pontos no intervalo

$$[j * (n / s) , (j + 1) * (n / s) - 1].$$

A complexidade do TCOND/A vale

$$n - nr,$$

e pode ser reduzida s vezes se

$$nr(i) = 0,$$

para todo i diferente dos valores

$$1, s + 1, 2 * s + 1, 3 * s + 1, ...,$$

e se, em cada ciclo de medidas, houver s pontos não testados da rede que possam ser conectados a s sensores diferentes.

Pode-se estabelecer que a complexidade mínima do TCOND/E é

SOMA(i,1,n,nr(i) * rds((i - 1) / s)),

quando as condições descritas acima ocorrem.

Por outro lado, a complexidade máxima é a mesma do TCOND/A, ou

n - nr,

e ocorre quando a particular distribuição dos pontos nas redes de uma UST impede o uso de mais de um sensor em cada ciclo.

5.3.5.3 - Efeito dos Erros na Complexidade do TCOND/E

A existência de defeitos do tipo circuito-aberto em redes resulta no uso de ciclos adicionais. Se for constatado que, em uma rede de z pontos, o ponto identificador está desconectado de k entre os

z - 1

demais pontos, pode-se afirmar que existe no mínimo uma rede adicional de k pontos.

É necessário, portanto, resolver também esta rede adicional, a fim de descobrir se a mesma não está decomposta em mais outras redes adicionais. A resolução desta rede adicional de k pontos utiliza o mesmo algoritmo de resolução da rede original. Portanto, para resolver esta rede de k pontos, o número de ciclos pode variar entre

rds((k - 1) / s)

e

k - 1.

5.3.5.4 - Descrição do TISOL/E

As mudanças nos blocos BS e MUX não determinam

melhoria de performance nem diferenças significativas no algoritmo do TISOL/D. Observa-se que uma melhoria de performance, em relação ao TISOL/D, ocorre quando houver defeitos do tipo fuga na UST, conforme será visto em 5.3.5.6.

5.3.5.5 - Complexidade do TISOL/E

A análise da seção 5.3.4.5 também é pertinente no presente caso.

5.3.5.6 - Efeito dos Erros na Complexidade do TISOL/E

Em 5.3.4.6, verificou-se que, em caso de fuga, o TISOL/D deve fazer duas pesquisas binárias para isolar as duas redes em fuga, uma em cada um dos 2 conjuntos de nr/2 redes. Um destes conjuntos corresponde às redes endereçadas pelas chaves de Y, e o outro às redes endereçadas pelas chaves de X. Cada uma destas 2 pesquisas consome

$$2 * \log2(nr / 2)$$

ciclos adicionais devido à fuga.

No caso do TISOL/E, a pesquisa para isolar a rede em fuga do conjunto endereçado pelas chaves de Y continua gastando o mesmo número de ciclos, mas a outra pesquisa, para isolar a rede em fuga do conjunto endereçado pelas chaves de X, necessita de um número menor de ciclos. Esta pesquisa binária não precisa partir da raiz, isto é, do conjunto total das nr/2 redes endereçadas pelas chaves de X. Como existem s sensores, a pesquisa só é realizada sobre as

$$nr / (2 * s)$$

redes ligadas ao sensor que indicou uma resistência menor do que RIS, admitindo-se que as nr/2 redes endereçadas pelas chaves de X se distribuam uniformemente sobre os s sensores. Portanto, a pesquisa binária para descobrir a rede em fuga dentre as redes endereçadas pelas chaves de X consome no mínimo

$$2 * \log2(nr / (2 * s)),$$

e no máximo

$$2 * \log2(nr / 2)$$

ciclos adicionais devido à rede em fuga.

Portanto, o número total de ciclos adicionais, para ambas as pesquisas, vale no mínimo

$$2 * \log2(nr / 2) + 2 * \log2(nr / (2 * s)),$$

ou

$$4 * \log2(nr / 2) - 2 * \log2(s)$$

ciclos. Em relação ao TISOL/D, economiza-se no máximo

$$2 * \log2(s)$$

ciclos adicionais devidos à fuga, e no mínimo nenhum ciclo.

5.3.6 - Versão F

5.3.6.1 - Descrição do TCOND/F

A análise da seção 5.3.5.1, para o TCOND/E, também é pertinente neste caso. As diferenças entre as arquiteturas das versões E e F são pequenas. Na versão F temos o valor de s fixo, em 16. Além disso, o bloco MUX distribui os intervalos de conectabilidade de pontos a sensores de maneira levemente diferente nas duas versões.

Estas diferenças não alteram a idéia básica do algoritmo do TCOND nas duas versões. Algumas alterações podem ocorrer a nível de complexidade.

5.3.6.2 - Complexidade do TCOND/F

A complexidade do TCOND/F tem limites mínimo e máximo iguais àqueles estudados na seção 5.3.5.2, para o TCOND/E. O TCOND/F, no entanto, normalmente possui uma complexidade média menor, conforme será discutido a seguir, de maneira informal.

O bloco MUX/C permite que um sensor S(j) seja alocado por pontos em um dos seguintes intervalos:

[0 * 1024 + j * 64 , 0 * 1024 + j * 64 + 63],

[1 * 1024 + j * 64 , 1 * 1024 + j * 64 + 63],

[2 * 1024 + j * 64 , 2 * 1024 + j * 64 + 63],

...

Isto significa que um sensor é conectável a pontos pertencentes a placas de mesma posição relativa dentro dos vários bastidores, conforme a modularidade descrita na seção 4.1. Portanto sempre existem 16 sensores, para qualquer configuração de AI com mais do que 1024 pontos, pois existem 16 placas por bastidor.

Esta organização possui a vantagem de possuir uma expansibilidade fácil, sem introdução de novos sensores. Além disso, oferece a possibilidade de utilizar mais sensores do que na versão E, caso a UST utilize menos do que n pontos, isto é, se o parâmetro LIM_INF não for programado em 0 ou se o parâmetro LIM_SUP não for programado em n - 1. Estes parâmetros foram discutidos na seção 2.2.3.

Considerando, por exemplo, n = 16384, pode-se comparar as versões E e F com ambas utilizando 16 sensores. Para que a versão E possa fazer uso de todos os 16 sensores, a UST deve possuir 16384 pontos. USTs menores não aproveitam todos os sensores. A versão F aproveita todos os 16 sensores desde que a UST tenha ao menos 1024 pontos. Este é um aspecto prático importante, uma vez que um usuário normalmente utiliza um AI para diferentes tipos de USTs. Os parâmetros LIM_INF e LIM_SUP podem diminuir o tempo do teste se delimitarem precisamente os pontos limites inferior e superior da UST.

5.3.6.3 - Efeito dos Erros na Complexidade do TCOND/F

A análise da seção 5.3.5.3 também é pertinente neste caso.

5.3.6.4 - Descrição do TISOL/F

A análise da seção 5.3.5.4 também é pertinente neste caso.

5.3.6.5 - Complexidade do TISOL/F

A análise da seção 5.3.4.5 também é pertinente no presente caso.

5.3.6.6 - Efeito dos Erros na Complexidade do TISOL/F

A análise da seção 5.3.5.6 também é pertinente neste caso.

5.3.7 - Versão G

5.3.7.1 - Descrição do TCOND/G

A análise da seção 5.3.5.1, para o TCOND/E, também é pertinente neste caso. As diferenças entre as arquiteturas das versões E e G ocorrem no bloco MUX. Estas diferenças não alteram a idéia básica do algoritmo do TCOND nas duas versões. Alterações ocorrem a nível de complexidade.

5.3.7.2 - Complexidade do TCOND/G

A inclusão do bloco MUX/D na arquitetura da versão G faz com que a distribuição de pontos dentro de cada rede não influa sobre a alocabilidade dos sensores em cada ciclo de medidas. O bloco MUX/D possibilita flexibilidade total neste aspecto, pois qualquer sensor S(j) pode ser conectado a qualquer ponto P(i), fechando-se as chaves X(i) e M(i,j).

Na versão TCOND/G, a resolução de uma rede de z pontos consome exatamente

rds((z - 1) / s)

ciclos. E o TCOND/G, portanto, consome

SOMA(i,1,n,nr(i) * rds((i - 1) / s))

ciclos, o que também equivale à complexidade mínima de TCOND/E e TCOND/F.

5.3.7.3 - Efeito dos Erros na Complexidade do TCOND/G

A análise da seção 5.3.5.3 também é pertinente neste caso. A única diferença é que, se uma rede adicional de k pontos for constatada, serão consumidos exatamente

rds((k - 1) / s)

ciclos para resolvê-la, a menos que esta rede se quebre em outras redes adicionais.

5.3.7.4 - Descrição do TISOL/G

A análise da seção 5.3.5.4 também é pertinente neste caso.

5.3.7.5 - Complexidade do TISOL/G

A análise da seção 5.3.4.5 também é pertinente no presente caso.

5.3.7.6 - Efeito dos Erros na Complexidade do TISOL/G

A análise da seção 5.3.5.6 também é pertinente neste caso. A única diferença é que a economia de ciclos adicionais para localizar uma fuga, em relação ao TISOL/D, é exatamente

2 * log2(s).

5.4 - Comparação de Arquiteturas e Algoritmos

Nesta seção serão utilizadas tabelas para comparar as 7 versões, no que tange ao custo de suas arquiteturas e à complexidade de seus algoritmos.

Na tabela 5.2 compara-se o custo das arquiteturas, para os 4 blocos componentes variáveis. Os

custos são avaliados por números inteiros, que refletem apenas uma proporcionalidade ou uma ordem de evolução, dentro do mesmo bloco componente. Não se pretende uma proporcionalidade exata. Também não se deve comparar o custo de blocos componentes de classe diferente através destes números. Isto é, se o custo de ENDX/B é 3, e o custo de BS/B é 16, não deve-se concluir que o custo de ENDX/B é menor do que o custo de BS/B. Uma excessão pode ser feita para comparar as classes de blocos ENDX e ENDY, devido à semelhança funcional entre ambas.

Os blocos ENDX e ENDY serão avaliados, de acordo com suas versões, com os seguintes valores:

- ENDX/A: 1
- ENDX/B: 3
- ENDY/A: 1
- ENDY/C: 2
- ENDY/D: 3

O bloco BS será avaliado pelo número de sensores, e o bloco MUX, pelo número de chaves utilizado no multiplexador.

TABELA 5.2 Comparação de Custos

Versão	ENDX	ENDY	BS	MUX
A	1	1	1	0
B	1	2	1	0
C	1	3	1	0
D	3	3	1	0
E	3	3	s	0
F	3	3	16	0
G	3	3	s	n * s

Na tabela 5.3 compara-se as complexidades, mínima, máxima ou exata, do TCOND e do TISOL em cada

versão.

TABELA 5.3 Comparação de complexidades

Versão		
A	TCOND - exata:	n - nr
	TISOL - exata:	C(nr,2)
B	TCOND - exata:	n - nr
	TISOL - máxima:	0,73 * n * log2(n) - 0,57 * n
C	TCOND - exata:	n - nr
	TISOL - exata:	nr - 1
D	TCOND - exata:	n - nr
	TISOL - exata:	rds(log2(nr))
E	TCOND - mínima:	SOMA(i,1,n,nr(i)*rds((i-1)/s))
	TCOND - máxima:	n - nr
	TISOL - exata:	rds(log2(nr))
F	TCOND - mínima:	SOMA(i,1,n,nr(i)*rds((i-1)/s))
	TCOND - máxima:	n - nr
	TISOL - exata:	rds(log2(nr))
G	TCOND - exata:	SOMA(i,1,n,nr(i)*rds((i-1)/s))
	TISOL - exata:	rds(log2(nr))

Em termos de complexidade dos algoritmos, as versões a partir da C apresentam boa possibilidade de implementação. A versão G pode ser muito cara, devido ao número de chaves do multiplexador. Entre as versões E e F, a F deve ser preferida, devido aos motivos apresentados na seção 5.3.6.2.

6 - AMBIENTE DE SIMULAÇÃO

Concebeu-se um ambiente de simulação para algoritmos do teste de interconexões. O simulador possui um conjunto mínimo de funções auxiliares, discutidas no capítulo 7, para facilitar a simulação dos algoritmos. Também são usados parâmetros discutidos na seção 2.2.

Este simulador pode ser visto como um protótipo do software definitivo do AI, pois a conversão do algoritmo simulado para o algoritmo definitivo não exige muito esforço adicional, e existe um número significativo de funções auxiliares total ou parcialmente implementadas. O interface com o usuário, orientado a menus, também já se volta para um produto final.

Simula-se apenas um dos algoritmos, o de versão B, embora outros pudessem ser adaptados ao ambiente de simulação, com grande reaproveitamento de código.

Um algoritmo simulado do TINT, na verdade, só possui uma diferença para o algoritmo definitivo. A obtenção de medidas, que na versão definitiva ocorre através da interação entre o controlador e o hardware de aquisição de dados, na versão simulada ocorre através da interação do controlador com um arquivo que modela as interconexões da UST. O algoritmo simulado chama as rotinas de endereçamento e medidas, mas estas na verdade acessam dados em memória, previamente trazidos de disco. Em cada ciclo de medidas pode-se imprimir informações que permitem visualizar o traçado do algoritmo, e pode-se medir o tempo de execução de trechos do software, a fim de estimar a performance do controlador. Várias opções de acompanhamento do traçado são disponíveis.

Apenas um dos algoritmos, o de versão B, foi simulado. Escolheu-se apenas esta versão pois sua análise é a mais complexa, e a simulação foi útil para se chegar a equação que estabelece a sua complexidade máxima. Além

disso, é o único meio de obter a complexidade exata para qualquer uma das inúmeras USTs, uma vez que a complexidade deste algoritmo depende da distribuição particular dos pontos e redes em uma UST.

6.1 - Utilização do Simulador

O simulador coloca a disposição um conjunto de funções auxiliares, além da função principal, que é o teste de interconexões. A descrição de cada função, e do seu uso, encontra-se no capítulo 7. Existem, dentre as funções auxiliares, algumas que só são úteis para fins de simulação, e que não aparecerão na versão definitiva. Simula-se apenas um dos algoritmos, o de versão B, embora outros pudessem ser adaptados ao ambiente de simulação, com grande reaproveitamento de código.

Nenhuma função dependende do hardware de aquisição de dados foi implementada, a não ser o teste de interconexões, que foi simulado. Entre as funções auxiliares dependentes do hardware, pode-se citar a auto-programação, os auto-diagnósticos e a nomeação simbólica de pontos por caneta. Algumas funções auxiliares não dependentes do hardware, e de menor importância para a simulação, também não foram implementadas, tais como a nomeação simbólica de pontos e o backup de padrões.

6.2 - Depuração da Versão B

Várias opções de acompanhamento do traçado dos algoritmos TCOND/B e TISOL/B podem ser obtidas pelo ajuste do parâmetro OUTPT, que permite o uso de impressora ou vídeo, a exibição ou não do traçado de cada ciclo e dos seus tempos de software, e dos totalizadores da complexidade e dos tempos de software.

Uma destas opções, a que oferece maiores detalhes, imprime, em seqüência:

a) o traçado do TCOND, que mostra em cada ciclo de medida:

- a chave de X endereçada;
- a chave de Y endereçada;
- o valor da medida, isto é, interconexão do tipo C ou NC.
- o tempo adicional de software, em microssegundos, que o controlador utiliza, e que não se superpõe com o tempo de estabilização da medida.

b) totalizadores do TCOND:

- número de ciclos de medida;
- tempo total de software do controlador;
- tempo total do teste, e "overhead" do tempo de software;

c) o traçado do TISOL, que mostra em cada ciclo de medida:

- tipo de ciclo: normal, coluna desabilitada, linha desabilitada, ou cancelado com o motivo do cancelamento.
- a chave de X endereçada;
- o intervalo de chaves de Y endereçadas;
- o valor da medida, isto é, interconexão do tipo I ou NI.
- o tempo adicional de software, em microssegundos, que o controlador utiliza, e que não se superpõe com o tempo de estabilização da medida.
- o tempo de software, em microssegundos, que o controlador utiliza enquanto espera-se a passagem do tempo de estabilização da medida. Este tempo de software não afeta o tempo de total do teste, caso for menor do que o tempo de estabilização em cada ciclo.

d) totalizadores do TISOL:

- número de ciclos de medida;
- tempo total de software do controlador;
- tempo total do teste, e "overhead" do tempo de software;

e) totalizadores de ambos os testes:

- número de ciclos de ambos os testes;
- tempo total do software do controlador;
- tempo total do teste de interconexões, e "overhead" do tempo de software;

f) listagem de erros do TCOND.

g) listagem de erros do TISOL.

7 - FUNÇÕES AUXILIARES DO ANALISADOR DE INTERCONEXÕES

O Analisador de Interconexões possui uma função principal, que é o teste de interconexões. Além disso, existe um conjunto de funções auxiliares, para programação do padrão, manutenção de arquivos padrões, diagnósticos, e outras.

Um simulador, com a implementação ou citação da maior parte das funções auxiliares necessárias em um AI, foi discutido no capítulo 6. A interface com o usuário deste simulador é orientada a menus.

Neste capítulo todas as funções do AI incluídas no simulador serão discutidas quanto ao seu efeito e uso. Algumas destas funções não estão implementadas, mas apenas citadas, porque não são importantes para a simulação ou porque dependem do hardware de aquisição de dados. Dentre as funções dependentes do hardware, apenas o teste de interconexões foi simulado. Algumas funções, como a programação de USTs, desaparecerão no produto final, pois só têm utilidade para a simulação.

7.1 - Menu Principal

- Estado: Implementado.
- Acesso: Não necessita de senha.
- Função: Possibilita a chamada de todas as funções do AI.
- Chamada: É o menu de mais alta hierarquia, apresentado logo após a energização do equipamento.

7.1.1 - Executar Teste de Interconexões

- Estado: Simulação implementada.
- Acesso: Não necessita de senha.
- Função: Executa o teste de interconexões, que compara o padrão exibido no parâmetro PAD_MEM à UST conectada ao AI, listando os erros encontrados na UST. Para efeito de simulação, a UST é representada por arquivos, e seu nome é

exibido no parâmetro UST_MEM.

- Chamada: Teclar F1, a partir do Menu Principal.

7.1.2 - Alterar Parâmetros

- Estado: Implementada.
- Acesso: Não necessita de senha.
- Função: Permite a programação de um ou mais parâmetros ajustáveis a partir do menu principal, tais como dispositivo de saída de erros, formato da listagem de erros, limites de erros, operador e padrão de carga. O parâmetro UST de carga é útil apenas para a simulação.
- Chamada: Pressionar a tecla F2, a partir do menu Principal. Será exibida a tela correspondente, onde os valores atuais de cada parâmetro são colocados à direita de seu campo de entrada. O usuário altera os parâmetros de acordo com o seguinte procedimento:

    - Para selecionar o parâmetro a ser alterado, o cursor deve ser levado até o campo de entrada associado, através das teclas de movimentação de cursor: "seta para baixo", "seta para cima", HOME e END.
    - Dentro de um campo de entrada, as teclas "seta para a esquerda", "seta para a direita" e "backspace" movimentam o cursor sem apagar os caracteres sobre os quais passa o cursor.
    - Para transmitir um campo de entrada, coloque o cursor na coluna seguinte ao final do texto, pressione a tecla ENTER ou RETURN, e a seguir a tecla F1, para confirmar a entrada. Será feita uma consistência sobre o campo transmitido. Se nenhum erro for detectado, o parâmetro é atualizado e o cursor passa para o campo de entrada seguinte. Caso contrário, um erro é listado na janela "MSG" e o cursor permance no campo de entrada atual.
    - Para negar a transmissão de um campo de

entrada, pode-se teclar F2, que apagará o campo atual e colocará o cursor no campo seguinte, tendo o mesmo efeito da tecla "seta para baixo".

- Para pedir ajuda, a respeito de algum parâmetro, posicione o cursor no seu campo de entrada e tecle F3. Uma mensagem de "help" aparecerá na janela "MSG".

7.1.3 - Carregar Novo Padrão para a Memória

- Estado: Implementada.
- Acesso: Não necessita de senha.
- Função: Carrega para a memória (PAD_MEM) um novo padrão especificado pelo usuário, e seus parâmetros associados.
- Chamada: Teclar F3, a partir do menu Principal. A seguir o sistema solicita o nome do padrão que deve ser carregado. Digite e transmita o nome, seguido de F1 para executar, ou de F2 para cancelar a função.

7.1.4 - Carregar Nova UST para a Memória

- Estado: Implementada. Temporária, para simulação.
- Acesso: Não necessita de senha.
- Função: Carrega para a memória (UST_MEM) uma nova UST especificada pelo usuário, e seus parâmetros associados.
- Chamada: Teclar F4, a partir do menu Principal. A seguir o sistema solicita o nome da UST que deve ser carregada. Digite e transmita o nome, seguido de F1 para executar, ou de F2 para cancelar a função.

7.1.5 - Arquivos

- Estado: Implementada.
- Acesso: Não necessita de senha.
- Função: Executa um série de funções relativas aos arquivos padrões e USTs, residentes em disco. As funções sobre arquivos USTs são temporárias, úteis apenas para

simulação.

- Chamada: Teclar F5, para chamar este menu a partir do menu Principal, ou F6, para chamá-lo a partir do menu Programação de Padrão, ou F3, para chamá-lo a partir do menu Programação de UST.

7.1.5.1 - Diretórios

- Estado: Implementada.
- Acesso: Não necessita de senha.
- Função: Permite observar o diretório global dos padrões e USTs, ou obter informações específicas sobre um padrão ou UST.
- Chamada: Teclar F1, para chamar este menu a partir do menu Arquivos.

7.1.5.1.1 - Diretório de Padrões

- Estado: Implementada.
- Acesso: Não necessita de senha.
- Função: Lista o nome de todos os arquivos constantes no diretório de padrões.
- Chamada: Teclar F1 a partir do menu Diretórios. Esta listagem poderá ter várias páginas, sendo necessário pressionar qualquer tecla para passar para a página seguinte.

7.1.5.1.2 - Diretório de USTs

- Estado: Implementada. Temporária, para simulação.
- Acesso: Não necessita de senha.
- Função: Lista o nome de todos os arquivos constantes no diretório de USTs.
- Chamada: Teclar F2 a partir do menu Diretórios. Esta listagem poderá ter várias páginas, sendo necessário pressionar qualquer tecla para passar para a página seguinte.

7.1.5.1.3 - Padrão Específico

- Estado: Implementada.
- Acesso: Não necessita de senha.
- Função: Lista o nome de um padrão selecionado e dos 3 arquivos físicos que lhe correspondem, cada um com seu tamanho, em bytes.
- Chamada: Teclar F3 a partir do menu Diretórios, digitar e transmitir o nome do padrão, seguido de F1 para confirmar, ou de F2 para negar a operação.

7.1.5.1.4 - UST Específica

- Estado: Implementada. Temporária, para simulação.
- Acesso: Não necessita de senha.
- Função: Lista o nome de uma UST selecionada e dos 2 arquivos físicos que lhe correspondem, cada um com seu tamanho, em bytes.
- Chamada: Teclar F4 a partir do menu Diretórios, digitar e transmitir o nome da UST, seguido de F1 para confirmar, ou de F2 para negar a operação.

7.1.5.2 - Deletar/Recuperar

- Estado: Implementada.
- Acesso: Necessita de senha.
- Função: Deletar arquivos, padrões e USTs, lógica e não fisicamente. Recuperar estes arquivos, em caso de deleção indevida.
- Chamada: Teclar F2, a partir do menu Arquivos, digitar a senha, transmití-la e teclar F1. Será exibido o menu Deletar/Recuperar.

7.1.5.2.1 - Deletar Padrão

- Estado: Implementada.
- Acesso: Não necessita de senha.
- Função: Marca como "deletado" o padrão selecionado, sem

excluir fisicamente seus arquivos relacionados. Isto permite a recuperação posterior deste padrão, em caso de engano do usuário.
- Chamada: Teclar F1 a partir do menu Deletar/Recuperar, digitar o nome do padrão, transmití-lo, confirmar via tecla F1, ou negar via tecla F2.

7.1.5.2.2 - Deletar UST

- Estado: Implementada. Temporária, para simulação.
- Acesso: Não necessita de senha.
- Função: Marca como "deletada" a UST selecionada, sem excluir fisicamente seus arquivos relacionados. Isto permite a recuperação posterior desta UST, em caso de engano do usuário.
- Chamada: Teclar F2 a partir do menu Deletar/Recuperar, digitar o nome da UST, transmití-lo, confirmar via tecla F1, ou negar via tecla F2.

7.1.5.2.3 - Recuperar Padrão

- Estado: Implementada.
- Acesso: Não necessita de senha.
- Função: Tenta recuperar um padrão anteriormente deletado. A operação terá sucesso se, entre a deleção e a recuperação, não tiver sido programado nenhum novo padrão nem tiver sido chamada a função Eliminar Padrões Deletados.
- Chamada: Teclar F3 a partir do menu Deletar/Recuperar, digitar o nome do padrão, transmití-lo, confirmar via tecla F1, ou negar via tecla F2.

7.1.5.2.4 - Recuperar UST

- Estado: Implementada. Temporária, para simulação.
- Acesso: Não necessita de senha.
- Função: Tenta recuperar uma UST anteriormente deletada. A operação terá sucesso se, entre a deleção e a recuperação, não tiver sido programada nenhuma nova UST nem tiver sido

chamada a função Eliminar USTs Deletadas.

- Chamada: Teclar F4 a partir do menu Deletar/Recuperar, digitar o nome da UST, transmití-lo, confirmar via tecla F1, ou negar via tecla F2.

7.1.5.3 - Listar

- Estado: Implementada.
- Acesso: Necessita de senha.
- Função: Listar arquivos padrões e USTs.
- Chamada: Teclar F3 a partir do menu Arquivos, digitar a senha, transmití-la e teclar F1. Será exibido o menu Listar.

7.1.5.3.1 - Listar Padrão

- Estado: Implementada.
- Acesso: Não necessita de senha.
- Função: Lista o padrão selecionado e seus parâmetros associados.
- Chamada: Teclar F1 a partir do menu Listar, digitar o nome do padrão, transmití-lo, e confirmar via tecla F1, ou negar via tecla F2. A seguir, digitar "I", para saída na impressora, ou "V", para saída em vídeo, ou "A", para abortar a impressão.

7.1.5.3.2 - Listar UST

- Estado: Implementada. Temporária, para simulação.
- Acesso: Não necessita de senha.
- Função: Lista a UST selecionada e seus parâmetros associados.
- Chamada: Teclar F2 a partir do menu Listar, digitar o nome da UST, transmití-lo, e confirmar via tecla F1, ou negar via tecla F2. A seguir, digitar "I", para saída na impressora, ou "V", para saída em vídeo, ou "A", para abortar a impressão.

7.1.5.4 - Backup

- Estado: Não implementada.
- Acesso: Necessita de senha.
- Função: Realiza "backups", em disquete, de arquivos padrões e USTs.
- Chamada: Teclar F4 a partir do menu Arquivos, digitar a senha, transmití-la e teclar F1. Será exibido o menu Backup.

7.1.5.4.1 - Backup de Padrão

- Estado: Não implementada.
- Acesso: Não necessita de senha.
- Função: Faz o backup do padrão selecionado.
- Chamada: Teclar F1 a partir do menu Backup, digitar o nome do padrão, transmití-lo, e confirmar via tecla F1, ou negar via tecla F2. A seguir, digitar o nome do arquivo backup, no drive "A:".

7.1.5.4.2 - Backup de UST

- Estado: Não implementada.
- Acesso: Não necessita de senha.
- Função: Faz o backup da UST selecionada.
- Chamada: Teclar F2 a partir do menu Backup, digitar o nome da UST, transmití-lo, e confirmar via tecla F1, ou negar via tecla F2. A seguir, digitar o nome do arquivo backup, no drive "A:".

7.1.5.5 - Renomear

- Estado: Implementada.
- Acesso: Necessita de senha.
- Função: Renomear arquivos padrões e USTs.
- Chamada: Teclar F5 a partir do menu Arquivos, digitar a senha, transmití-la e teclar F1. Será exibido o menu Renomear.

7.1.5.5.1 - Renomear Padrão

- Estado: Implementada.
- Acesso: Não necessita de senha.
- Função: Renomeia o padrão selecionado.
- Chamada: Teclar F1 a partir do menu Renomear, digitar o nome do padrão, transmití-lo, e confirmar via tecla F1, ou negar via tecla F2. A seguir, digitar o novo nome, transmití-lo, e confirmar via F1, ou negar via F2.

7.1.5.5.2 - Renomear UST

- Estado: Implementada. Temporária, para simulação.
- Acesso: Não necessita de senha.
- Função: Renomeia a UST selecionada.
- Chamada: Teclar F2 a partir do menu Renomear, digitar o nome da UST, transmití-lo, e confirmar via tecla F1, ou negar via tecla F2. A seguir, digitar o novo nome, transmití-lo, e confirmar via F1, ou negar via F2.

7.1.5.6 - Eliminar Deletados

- Estado: Implementada.
- Acesso: Necessita de senha.
- Função: Eliminar, fisicamente, todos os arquivos padrões e USTs deletados logicamente.
- Chamada: Teclar F6 a partir do menu Arquivos, digitar a senha, transmití-la e teclar F1. Será exibido o menu Eliminar Deletados.

7.1.5.6.1 - Eliminar Padrões Deletados

- Estado: Implementada.
- Acesso: Não necessita de senha.
- Função: Elimina, fisicamente, todos os padrões marcados como deletados no diretório de padrões.
- Chamada: Teclar F1 a partir do menu Eliminar Deletados.

7.1.5.6.2 - Eliminar USTs Deletadas

- Estado: Implementada. Temporária, para simulação.
- Acesso: Não necessita de senha.
- Função: Elimina, fisicamente, todas as USTs marcadas como deletadas no diretório de USTs.
- Chamada: Teclar F2 a partir do menu Eliminar Deletados.

7.1.6 - Programação de Padrão

- Estado: Implementada.
- Acesso: Necessita de senha.
- Função: Gerar um arquivo padrão, que será incluído no diretório de padrões e poderá ser utilizado no teste de interconexões. Esta é uma das mais importantes funções auxiliares.
- Chamada: Teclar F6, a partir do menu Principal. A seguir, digitar a senha, transmití-la, e teclar F1 para confirmar a operação, ou F2 para negá-la.

7.1.6.1 - Preparar Parâmetros

- Estado: Implementada.
- Acesso: Não necessita de senha.
- Função: Ajustar os parâmetros que estarão associados ao padrão que será programado.
- Chamada: Teclar F1 a partir do menu Programação de Padrão. Será exibida a tela correspondente. A entrada dos parâmetros é feita de forma similar àquela discutida na seção 7.1.2, para a função Alterar Parâmetros.

7.1.6.2 - Programação por Edição

- Estado: Implementada.
- Acesso: Não necessita de senha.
- Função: A partir de um arquivo texto, editado pelo usuário, com extensão ".R", para descrever as redes, o sistema gera um padrão em disco.

- Chamada: Teclar F2 a partir do menu Programação de Padrão, e a seguir F1 para confirmar que os parâmetros estão ajustados, ou F2 para cancelar a função. Digitar o nome do padrão a ser programado, transmití-lo, e confirmar via F1, ou negar via F2. A seguir, digitar o nome do arquivo ".R", transmití-lo, e confirmar via F1, ou negar via F2.

7.1.6.3 - Autoprogramação

- Estado: Não implementada.
- Acesso: Não necessita de senha.
- Função: Gerar um padrão a partir de uma UST boa conectada ao AI.
- Chamada: Após a conexão da UST boa ao AI, teclar F3 a partir do menu Programação de Padrão, seguido de F1 para confirmar que os parâmetros estão ajustados, ou de F2 para cancelar a função. Digitar o nome do padrão e transmití-lo, confirmando via F1, ou negando via F2.

7.1.6.4 - Programação Aleatória

- Estado: Implementada. Temporária, para simulação.
- Acesso: Não necessita de senha.
- Função: Gerar um padrão pseudo-aleatório, a partir um arquivo básico com extensão ".ALP", que indica o número de redes e seu tamanho, como parâmetros de controle do usuário.
- Chamada: Teclar F4 a partir do menu Programação de Padrão, seguido de F1 para confirmar que os parâmetros estão ajustados, ou de F2 para cancelar a função. Digitar o nome do padrão e transmití-lo, confirmando via F1, ou negando via F2. Digitar a seguir o nome do arquivo ".ALP" e transmití-lo, confirmando via F1, ou negando via F2.

7.1.6.5 - Nomeação de Pontos

- Estado: Implementada.

- Acesso: Não necessita de senha.
- Função: Atribuir nomes simbólicos alfanuméricos aos pontos de teste de um padrão.
- Chamada: Teclar F5 a partir do menu Programação de Padrão.

7.1.6.5.1 - Nomeação por Edição

- Estado: Não implementada.
- Acesso: Não necessita de senha.
- Função: Atribuir nomes simbólicos a pontos de teste de um padrão existente, a partir de um arquivo editado, com extensão ".N".
- Chamada: Teclar F1 a partir do menu Nomeação de Pontos, digitar o nome do padrão e transmití-lo, confirmando via F1, ou negando via F2. Digitar a seguir o nome do arquivo ".N" e transmití-lo, confirmando via F1, ou negando via F2.

7.1.6.5.2 - Nomeação por Caneta

- Estado: Não implementada.
- Acesso: Não necessita de senha.
- Função: Atribuir nomes simbólicos a pontos de teste de um padrão existente, apontando-os através de uma caneta especial.
- Chamada: Teclar F2 a partir do menu Nomeação de Pontos, digitar o nome do padrão e transmití-lo, confirmando via F1, ou negando via F2. A seguir, para nomear cada ponto:
    - aponte o ponto com a caneta;
    - digite F1, para confirmar que o ponto está apontado;
    - após a localização do endereço do ponto, o sistema o imprime. O usuário digita o nome do ponto e o transmite, seguido de F1 para confirmar, ou de F2 para negar.

Para encerrar a nomeação de pontos por caneta, digitar ESC.

7.1.6.5.3 - Rotina NOM3

- Estado: Não implementada.
- Acesso: Não necessita de senha.
- Função: Atribuir nomes simbólicos a pontos de teste de um padrão existente, utilizando uma rotina que calcula o nome de um ponto a partir do seu número, em casos de distribuições regulares de pontos. Estas rotina pode ser feita sob encomenda do usuário, e ligada ao sistema.
- Chamada: Teclar F3 a partir do menu Nomeação de pontos. Digitar e transmitir o nome do padrão, seguido de F1 para confirmar ou de F2 para negar.

7.1.6.5.4 - Rotina NOM4

Semelhante a rotina NOM3.

7.1.6.5.5 - Rotina NOM5

Semelhante a rotina NOM3.

7.1.6.5.6 - Rotina NOM6

Semelhante a rotina NOM3.

7.1.6.6 - Arquivos

Vide seção 7.1.5.

7.1.6.7 - Gerar Arquivo .R

- Estado: Implementada. Temporária, para simulação.
- Acesso: Não necessita de senha.
- Função: Cria um arquivo do tipo ".R" a partir de um padrão base. Este arquivo pode ser utilizado para a programação por edição de outros padrões ou USTs, após possíveis modificações.
- Chamada: Teclar F7 a partir do menu Programação de Padrão, digitar o nome do padrão base, transmiti-lo e teclar F1 para confirmar, ou F2 para negar. A seguir

digitar o nome do arquivo ".R" que será gerado, transmiti-lo e teclar F1 para confirmar, ou F2 para negar.

7.1.7 - Programação de UST

- Estado: Implementada. Temporária, para simulação.
- Acesso: Necessita de senha.
- Função: Gerar um arquivo UST, que será incluído no diretório de USTs e poderá ser utilizado no teste de interconexões.
- Chamada: Teclar F7, a partir do menu Principal. A seguir, digitar a senha, transmití-la, e teclar F1 para confirmar a operação, ou F2 para negá-la.

7.1.7.1 - Preparar Parâmetros

- Estado: Implementada. Temporária, para simulação.
- Acesso: Não necessita de senha.
- Função: Ajustar os parâmetros que estarão associados à UST que será programada.
- Chamada: Teclar F1 a partir do menu Programação de UST. Será exibida a tela correspondente. A entrada dos parâmetros é feita de forma similar àquela discutida na seção 7.1.2, para a função Alterar Parâmetros.

7.1.7.2 - Programação por Edição

- Estado: Implementada. Temporária, para simulação.
- Acesso: Não necessita de senha.
- Função: A partir de um arquivo texto, editado pelo usuário, com extensão ".R", para descrever as redes, o sistema gera uma UST em disco.
- Chamada: Teclar F2 a partir do menu Programação de UST e a seguir F1 para confirmar que os parâmetros estão ajustados, ou F2 para cancelar a função. Digitar o nome da UST a ser programada, transmití-lo, e confirmar via F1, ou negar via F2. A seguir, digitar o nome do arquivo ".R", transmití-lo, e confirmar via F1, ou negar via F2.

7.1.7.3 - Arquivos

Vide seção 7.1.5.

7.1.8 - Auto-Diagnósticos / Manutenção

- Estado: Implementada.
- Acesso: Necessita de senha.
- Função: Realizar testes sobre o hardware do sistema de aquisição de dados, para verificar se o seu funcionamento está correto e, caso contrário, isolar os defeitos, na medida do possível.
- Chamada: Teclar F8 a partir do Menu Principal, digitar e transmitir a senha, confirmando via F1, ou negando via F2.

7.1.8.1 - Teste de Pontos

- Estado: Não implementada.
- Acesso: Não necessita de senha.
- Função: Detectar pontos de teste que possuem uma ou as duas chaves constituintes permanentemente abertas.

7.1.8.2 - Teste de Fugas

- Estado: Não implementada.
- Acesso: Não necessita de senha.
- Função: Detectar pontos de teste que possuem uma ou as duas chaves constituintes permanentemente fechadas.

7.1.8.3 - Teste de Bastidor

- Estado: Não implementada.
- Acesso: Não necessita de senha.
- Função: Realiza os testes de fugas e pontos sobre um bastidor, além de testar a placa "buffer de bastidor".

7.1.8.4 - Teste de Placa

- Estado: Não implementada.

- Acesso: Não necessita de senha.
- Função: Realiza os testes de fugas e pontos sobre uma placa de 64 pontos de teste, além de testar também a precisão com que esta efetua as medidas de resistência. Outros testes, mais complexos, deverão ser incluídos a nível de placa.

7.1.9 - Alterar Senha

- Estado: Implementada.
- Acesso: Necessita de senha.
- Função: Alterar a senha.
- Chamada: Teclar F9 a partir do menu Principal, digitar e transmitir a senha, seguida de F1 para confirmar ou de F2 para cancelar. A seguir digitar e transmitir a nova senha, seguida de F1 para confirmar ou de F2 para cancelar. O sistema exige a repetição deste último passo, por questões de segurança.

7.1.10 - Reinicializar o Equipamento

- Estado: Implementada.
- Acesso: Não necessita de senha.
- Função: Reinicializa todas as variáveis do sistema, sem recarregar o programa.
- Chamada: Teclar, simultaneamente, CTRL e I a partir do menu Principal.

7.1.11 - Desligar o Equipamento

- Estado: Implementada.
- Acesso: Não necessita de senha.
- Função: Desliga o equipamento.
- Chamada: Teclar, simultaneamente, CTRL e D a partir do menu Principal.

8 - CONCLUSÕES

Em âmbito internacional, o Analisador de Interconexões é um equipamento de testes cuja tecnologia parece ter sido dominada. Algumas empresas que comercializam o equipamento devem ter encontrado algoritmos e arquiteturas ótimos no aspecto da velocidade do teste, colocando no mercado AIs rápidos, e além disso tecnica e economicamente viáveis.

Aumentar a velocidade de um AI só se justifica se a redução do tempo do teste de interconexões puder ser percebida como uma economia significativa no tempo global de teste, que também inclui tempos de deslocamento e posicionamento mecânicos, além de outros. As necessidades das empresas usuárias devem ser estudadas, de forma a cumprir os requisitos de velocidade e ao mesmo tempo minimizar o custo do equipamento.

As empresas que dominam a tecnologia de Analisadores de Interconexões não costumam publicar informações sobre seus algoritmos e arquiteturas. Por este motivo, nesta dissertação o mestrando baseou-se em informações e criação pessoais. A bibliografia obtida limita-se a considerar aspectos como vantagens econômicas do uso de AIs, inserção do equipamento no ambiente de teste, danos que pode causar, entre outros.

Nesta dissertação, procurou-se idealizar algoritmos e arquiteturas com velocidade e custo similares aos dos equipamentos internacionais. Acredita-se que tais objetivos foram alcançados, embora um estudo mais profundo fosse necessário para provar este resultado intuitivo. Uma comparação justa necessitaria, por exemplo, do levantamento de custos de componentes no mercado externo. Uma comparação realística, entretanto, necessitaria levar em conta as condições locais.

Uma condição local importante, ao se

industrializar um AI, é a inexistência de camas-de-prego adequadas com tecnologia nacional. Conforme o contato do autor com indústrias deste tipo de equipamento, uma das aplicações que mais parece crescer nos últimos anos é o teste de placas nuas de circuito impresso. Este segmento do mercado torna a cama-de-pregos um dispositivo tão importante como a própria parte eletrônica do AI. Enquanto este problema não for resolvido a contento, dificilmente um AI nacional conseguirá atingir este mercado.

Através desta dissertação, imagina-se que foi dado um passo fundamental no projeto inteiramente nacional de um Analisador de Interconexões, competitivo no mercado internacional, o que pode ser considerado um avanço no setor nacional de Equipamentos Automáticos de Teste.

O prosseguimento deste trabalho depende de investimentos adicionais, principalmente na prototipação do hardware, e na montagem de uma estrutura de produção. O software pode ser evoluído a partir do simulador. O hardware já está detalhado até o nível lógico. Os esquemas do hardware, bem como o código fonte do simulador, não fazem parte do texto desta dissertação. O projeto da parte essencial do AI, isto é, a arquitetura e os algoritmos, em alto nível de abstração, está praticamente completo.

BIBLIOGRAFIA

/CAB SD/ CABLESCAN, INC. Electronic Assembly Aid - Model ET-1024. Instruction Manual. S.D.

/CIR SD/ CIRCUIT LINE. PCA 32K-64K Bare Board Tester. S.D.

/COO 76/ COOLBEAR,B. & LOVITT,G. The Development of an Automatic Wiring Analyzer System for Testing Telephone Switching Rack Backplanes. The Radio and Electronic Engineer. 46(7):323-31, July 1976.

/ERI 79/ ERICKSON, D. Cable/Harness Testers. Electronic Packaging and Production. 19(9):90-101, Sept. 1979.

/GLA 84/ GLAWCY, D. R. Developments in Test Fixturing. Test & Measurement World. 4(12):27-45, Dec. 1984.

/HOL 86/ HOLLEY, C. D. Hipot and Ground Continuity Testing. Test & Measurement World. 6(12):22-30, Dec. 1986.

/HRO 87/ HROUNDAS, G. Eliminate Bare Board-PCB Test Damage Whith Strict Parameter Control. Test & Measurement World. 7(12):17-28, Dec. 1987.

/MAY 84/ MAYER, J. Semi- and Full Automation Enhance Continuity Test. Test & Measurement World. 4(12):27-45, Dec. 1984.

/TES SD/ TESTRONICS. Model 102A High Voltage Interconnect Tester. S.D.

/CON 86/ CONTINUITY Tester Sampler. Test & Measurement World. 6(12):33-4, Dec. 1986.

/WIT 80/ WITOMSKI, D. The Benefits of Using Modern Cable Assembly and Testing Techniques. Electronic Packaging and Production. 20(12):165-70, Dec. 1980.