JANOR ARAUJO BASTOS

TESTE EM FUNCIONAMENTO DE UMA MATRIZ DE CHAVEAMENTO

Porto Alegre

2002

JANOR ARAUJO BASTOS

TESTE EM FUNCIONAMENTO DE UMA MATRIZ DE CHAVEAMENTO

ORIENTADOR: Prof. Dr. Marcelo Soares Lubaszewski

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica (PPGEE), da Universidade Federal do Rio Grande do Sul (UFRGS), como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica.

Área de concentração: Automação e Instrumentação Eletro-eletrônica

Porto Alegre

2002

JANOR ARAUJO BASTOS

TESTE EM FUNCIONAMENTO DE UMA MATRIZ DE CHAVEAMENTO

Esta dissertação foi julgada adequada para a obtenção do título de Mestre em Engenharia Elétrica e aprovada em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador:_____

Prof. Dr. Marcelo Soares Lubaszewski, UFRGS.

Doutor pelo Institut National Polytechique de Grenoble – França.

Banca Examinadora:

Prof. Dr. José Luiz Almada Güntzel.
Universidade Federal de Pelotas (UFPEL).

Prof. Dr. Altamiro Amadeu Susin.
Universidade federal do Rio Grande Do Sul (UFRGS).

Prof. Dr. Walter Fetter Lages.
Universidade Federal do Rio Grande do Sul (UFRGS).

Coordenador do PPGEE:_____

Prof. Dr.Carlos Eduardo Pereira.

Porto Alegre, dezembro, 2002.

Dedico este trabalho à minha esposa Valderez e à minha filha Clariss

AGRADECIMENTOS

Aos meus pais a quem devo a minha vida e o meu caráter.

À minha esposa Valderez e à minha filha Clarissa pelo incentivo, apoio, amor e compreensão pelos momentos que não estive presente.

Aos meus irmãos pelo carinho, compreensão, apoio e incentivo para que eu prosseguisse meus estudos.

Ao Prof. Dr. Marcelo Soares Lubaszewski pela orientação e incansável apoio e incentivo que sempre estiveram presentes me estimulando para que a realização deste trabalho se concretizasse.

Ao professor Anatólio Laschuck que com sua sabedoria, muito contribuiu para os meus primeiros passos no caminho da pesquisa.

A todos os professores do PPGEE, dos quais de maneira formal ou informal, recebi os ensinamentos tão necessários ao meu progresso como estudante do curso de mestrado.

Aos colegas Vanderli Cornélius, Cláudio Fernandez, Kemel Bensebâa, Eduardo Costa Motta, José Julio G. Fernandes e Fernando Paladino, por terem proporcionado um ambiente de amizade e companheirismo no qual convivemos durante este curso.

Aos bolsistas de iniciação científica Jáder Alexandre Kussler e Leandro José Cassol, que muito contribuíram nas simulações e implementação do protótipo.

Às secretárias Miriam A. O. Rosek, Janice Oliveira e Michele Cruz do PPGEE, Beatriz Ferraz e Ana Paula Freitas do PPGEMM pelo suporte e atenção que me dispensaram.

À Equitel S. A. pelo apoio financeiro.

Por fim, a todos que colaboraram direta ou indiretamente na elaboração deste trabalho, o meu reconhecimento.

RESUMO

Este trabalho se insere na área de teste de sistemas de hardware. O alvo principal é o estudo do comportamento de um circuito roteador de canais telefônicos, parte integrante de um sistema de comunicação mais complexo, na presença de falhas. Neste contexto, o teste em funcionamento do referido circuito roteador é considerado.

Na primeira parte deste trabalho são abordados aspectos do teste de circuitos e sistemas, do ponto de vista de sua aplicabilidade, tais como classificação, defeitos e modelos de falhas, simulação, geração de testes e projeto visando o teste.

Na segunda parte, relata-se os estudos realizados para implementar o teste em funcionamento do circuito roteador. Nesta etapa são abordados a arquitetura, o modelo de falhas e a metodologia utilizada, os ensaios de detecção de falhas e as técnicas de tolerância a falhas adotadas. O projeto do circuito de chaveamento é apresentado em uma versão utilizando componentes discretos e outra utilizando dispositivos programáveis.

Na conclusão deste trabalho são apresentados os resultados obtidos e as perspectivas para trabalhos futuros.

PALAVRAS-CHAVE:

Teste de sistemas de hardware, Teste em funcionamento, Circuito roteador, Testabilidade, Tolerância a falhas, Projeto visando o teste.

ABSTRACT

This work is inserted in the domain of hardware testing. Its main target is the investigation of the behaviour of a switching matrix, part of a more complex communication system, in the presence of faults. Within this context, the on-line testing of the circuit is considered.

In the first part of this work, general aspects of circuits and systems testing, from the applicability point of view, are addressed. The following aspects are discussed: classification, defects and fault models, fault simulation, test generation and design for test.

In the second part, the steps taken to implement the on-line testable switch matrix are described. At this point, the circuit under test and test circuitry, the fault model and test methodology used, the experiments for fault detection carried out and the techniques for fault tolerance adopted are widely discussed. The design of the switch matrix is presented in two versions: one based on discrete components and another based on programmable devices.

In the conclusions, the results obtained are summarized and future works are proposed.

KEYWORDS:

Hardware testing, on-line testing, switch matrix, testability, fault tolerant hardware, design for test.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ATPG - Automatic Test Pattern Generation

BIST - Built-In Self-Test

CAD - Computer Aided Design

CI - Circuito Integrado

CLB - Configurable Logic Block

FPGA - Field Programmable Gate Array

LFSR - Linear Feedback Shift Register

MOS - Metal Oxide Silicon

NMOS - Negative Metal Oxide Silicon

ORA - Output Response Analyser

SMD - Surface Mount Device

TPG -Test Pattern Generation

VHDL - Very Hight Speed Integrated Circuit (VHSIC) Hardware Description Language

1. INTRODUÇÃO

Na época em que os circuitos digitais de baixa complexidade funcional e estrutural eram implementados a partir de componentes discretos, tinha-se acesso amplo às linhas internas do circuito. Por este motivo, era razoável que houvesse pouca preocupação com o teste nas fases iniciais do projeto.

Atualmente, as telecomunicações e outros segmentos do mercado de sistemas eletrônicos exigem circuitos mais complexos, mais rápidos, mais densos, mais baratos e que sejam projetados com maior rapidez.

Com a utilização de circuitos integrados de alta densidade e placas de circuito-impresso baseadas em tecnologia SMD (*Surface Mount Device*) para a montagem dos componentes, o acesso aos pontos de teste ficou drasticamente reduzido. Por esta razão, passou a ser praticamente impossível o desenvolvimento de produtos de boa qualidade sem dispor de métodos e mecanismos adequados para o teste de circuitos e sistemas. Os testes funcionais convencionais não são suficientes para assegurar a qualidade requerida pelo mercado, é necessário que sejam utilizados métodos que procurem defeitos de fabricação e falhas que ocorram durante a vida útil do sistema. Testadores externos com características necessárias para obter coberturas de falhas aceitáveis têm custo muito elevado e, portanto, alternativas precisam ser buscadas para reduzir o custo do teste. Para agravar a situação, estima-se que o custo de detecção de falhas é multiplicado por um fator de dez quando se passa do nível de circuito ao nível de placa, da placa ao nível de sistema, e por fim, à aplicação do sistema no campo. Estes fatores levam a crer que mecanismos de teste devam ser integrados nos chips e que esses mecanismos possam ser reutilizados para testar circuitos no ambiente de operação do sistema.

Mecanismos que melhoram a testabilidade do circuito devem ser implementados segundo uma filosofia de projeto visando o teste. Por exemplo, estruturas que permitam a acessibilidade a pontos internos do circuito, tanto para aplicação de estímulos quanto para avaliação da resposta ao teste, podem ser previstas e adicionadas à lógica que implementa a função do circuito. Porém, como a integração de tais estruturas pode comprometer a área e o desempenho do circuito sob teste, deve-se analisar criteriosamente cada caso e buscar uma

solução que conduza ao melhor compromisso entre a qualidade e o custo. Nos dias atuais pode-se considerar que técnicas de projeto visando o teste passam a fazer parte do dia-a-dia dos projetistas.

É notório que muitos sistemas como os automotivos, de aviação, plantas nucleares, etc, requerem alta segurança de funcionamento e, portanto, não podem tolerar comportamentos errôneos que conduzam, por exemplo, à perda de vidas humanas. Nesses sistemas, além dos testes convencionais utilizados para a detecção de falhas durante o processo de fabricação, a detecção de falhas concorrente à aplicação torna-se uma condição indispensável. Mesmo em sistemas onde as falhas não oferecem perigo, existem situações em que a interrupção brusca de um processo é inadmissível, já que estas causariam danos de ordem econômica ou técnica de proporções inaceitáveis.

Este trabalho apresenta um estudo sobre testes de sistemas de hardware focando no teste em funcionamento de um circuito de chaveamento, parte integrante de um sistema de comunicação mais complexo projetado para a Siemens/Equitel do Brasil.

Inicialmente, abordam-se aspectos do teste de circuitos e sistemas, tais como classificação do ponto de vista da aplicabilidade, defeitos e modelos de falhas utilizados para simulação, geração de testes e projeto visando o teste.

Na seqüência, apresentam-se os estudos realizados no sentido de implementar o teste em funcionamento de uma matriz de roteamento de canais telefônicos. Aborda-se a arquitetura da matriz, o modelo de falhas e a metodologia de teste utilizados, os ensaios de detecção de falhas e as técnicas de tolerância a falhas adotadas. Neste contexto apresenta-se o projeto do circuito de chaveamento tolerante a falhas em duas versões, uma utilizando componentes discretos e outra utilizando dispositivos programáveis.

Finalmente conclui-se este trabalho discutindo-se os resultados obtidos e as perspectivas para trabalhos futuros.

2. MÉTODOS DE TESTE

2.1. CLASSIFICAÇÃO

Desde o projeto até a sua utilização no campo, um circuito passa por etapas de verificação que procuram identificar dispositivos falhos. Estas etapas são: a depuração do protótipo, o teste de produção e o teste de manutenção. A depuração preocupa-se em garantir que o protótipo atende às especificações do projeto. Os testes de produção visam depurar o processo de fabricação do circuito. O objetivo final do teste de produção é identificar as unidades defeituosas e separá-las das não defeituosas. Quando no campo, se o dispositivo estiver em operação, é necessário que esta seja interrompida para que os procedimentos de teste de manutenção sejam aplicados. No teste de manutenção é essencial que sejam previstos mecanismos para diagnosticar as partes defeituosas, a fim de que o problema possa ser corrigido. Todos estes testes são conhecidos como testes fora de funcionamento *(off-line)*.

Alguns circuitos inserem-se em sistemas que requerem alta segurança de funcionamento, cujas funções não podem ser interrompidas e também não podem apresentar um comportamento incorreto. Nestes casos torna-se essencial a detecção de falhas concorrente à aplicação. Este tipo de teste é chamado de teste em funcionamento *(on-line)*. O teste em funcionamento é usado para verificar a validade das operações e pode ser assegurado através de mecanismos de *software* e dados codificados, ou por dispositivos *self-checking*. Em circuitos *self-checking* a capacidade de detecção concorrente de falhas é conseguida através de blocos funcionais que produzem saídas codificadas e blocos verificadores que conferem se as saídas pertencem ao código de detecção de erros utilizado.

Retornando aos testes fora de funcionamento, podemos destacar duas classes distintas de métodos: o teste estrutural e o teste funcional. O teste destinado a verificar se a implementação física do circuito reflete seu esquemático é chamado *teste estrutural*, e o que verifica se o projeto está de acordo com as especificações no que diz respeito ao seu comportamento funcional, é chamado *teste funcional*.

Ao teste estrutural está associada uma medida quantitativa de detectabilidade, a cobertura de falhas. A cobertura de falhas vem a ser uma medida da qualidade de uma seqüência de teste, representada pela razão entre as falhas detectadas pela seqüência e as falhas pertencentes a um conjunto previamente definido de falhas.

O teste funcional por sua vez, visa identificar problemas de desempenho do circuito. Falhas de desempenho (atrasos de portas, de interconexões) podem ser detectadas se o teste estrutural puder ser realizado na velocidade de operação do sistema. Como, tipicamente, este não é o caso, o teste funcional, apesar de mais longo devido à redundância inerente de estímulos, precisa ser utilizado. Mais raramente, quando não se conhece a estrutura interna do circuito, o teste funcional termina substituindo o teste estrutural. Do ponto de vista do teste em funcionamento, somente o teste funcional tem sentido.

Outras duas grandes classes de teste são as dos testes de tensão e a dos testes de corrente. No teste de tensão, a resposta verificada é a amplitude do sinal de saída, por este motivo, a presença de falhas é indicada por variações na tensão resultando em valores diferentes dos previstos. Entretanto, muitas falhas, como transistores que sempre conduzem ou falhas de curto-circuito (*bridging*), resultam em elevação da corrente do circuito sob teste, sem no entanto alterar os valores de tensão (Maly, 1988). Se a corrente se torna maior que a esperada, estas falhas podem ser detectadas por sensores de corrente. Este método é conhecido como teste de corrente. Os testes fora e em funcionamento podem implementar tanto testes de tensão quanto de corrente.

2. 2 DEFEITOS E MODELOS DE FALHAS

Em um sentido mais geral, teste consiste em aplicar uma seqüência de sinais à entrada de um circuito, observar a seqüência de saídas e compará-la com a seqüência de saída esperada. Qualquer discrepância com relação à seqüência esperada constitui um *erro* e a causa desse erro é chamada *falha* (Breuer, 1976).

As falhas são causadas por defeitos físicos, resultantes do processo de fabricação do circuito ou de erros de projeto. Os defeitos podem ter várias origens. Podem ser inerentes ao substrato do silício no qual as estruturas integradas são fabricadas. Alguns podem ser provenientes de impurezas no material usado para produzir os *wafers*, outros provenientes de problemas ocorridos durante as várias etapas do processo de fabricação. Podem ser citados, por exemplo, a resistividade de contatos, que depende da quantidade de dopante; a presença de partículas de impurezas na sala limpa (*clean room*) ou nos materiais; o desalinhamento das máscaras, que pode causar alterações nas dimensões dos transistores, etc. Estes defeitos geralmente conduzem a falhas que atingem simultaneamente vários dispositivos. As falhas deste tipo são chamadas falhas

múltiplas. Até mesmo durante as fases de montagem em placas de circuito impresso e de teste, podem ser introduzidas falhas, como por exemplo, as falhas de interconexão. Outros defeitos ocorrem durante a vida útil do circuito. Estes normalmente são causados por mecanismos de transporte e fenômenos eletromecânicos, fatores térmicos, estresse mecânico (transporte, manipulação), interferências eletromagnéticas, etc. Estes defeitos em geral produzem falhas simples.

Falhas permanentes, como interconexões abertas ou em curto, portas que não estão chaveando, etc., podem surgir de defeitos oriundos do processo de fabricação ou do uso do circuito. As falhas transitórias estão diretamente ligadas a fenômenos intermitentes, como interferências eletromagnéticas ou radiações.

Muitos modelos de falhas que tentam representar corretamente defeitos físicos em circuitos têm sido apresentados (Wadsack, 1978; Galay, 1980; Courtois, 1981; Rajusman, 1992). Para os circuitos digitais, podem ser citados como exemplos de modelos de falhas:

1. *stuck-at fault*: fixo-em-zero ou fixo-em-um;
2. *stuck-on* ou *stuck-open:* transistores que sempre ou nunca conduzem;
3. *slow-to-rise e slow-to-fall:* portas com atraso de subida e descida;
4. *bridging fault*: curto circuito;
5. *open fault*: circuito aberto;
6. *falhas paramétricas*: variação de parâmetros elétricos;
7. *falhas funcionais*: instruções de microprocessadores que não executam como deveriam, por exemplo.

O modelo computacional mais utilizado para a representação de falhas é o modelo lógico *stuck-at*. Falha lógica é aquela que causa a troca da função lógica de um elemento do circuito ou sinal por alguma outra função. Uma falha lógica típica é algum sinal do circuito tornar-se fixo em um valor constante independente dos valores lógicos que contribuem para a determinação do valor lógico no nó. Se este valor for *um*, denomina-se a falha *stuck-at-one*, e se o valor for *zero, stuck-at-zero*. Convém observar que o nó em falha do tipo *stuck-at-one* ou *stuck-at-zero* não se comporta como um curto circuito ou contato elétrico com GND ou VCC. Faz mais sentido entender o seu comportamento se imaginarmos que o circuito foi interrompido no local da falha e o seu valor lógico, daquele ponto em diante, passa a ser fixo.

Utilizando como exemplo uma porta E, com sinais A e B em suas duas estradas, uma falha do tipo *stuck-at 1* na entrada A, faz a função lógica A.B da porta trocar para B.

O modelo mais utilizado é o *single stuck-at*, que assume que não pode ocorrer mais que uma falha em um dado momento. Sem esta consideração seria praticamente inviável gerar padrões de teste para circuitos com nível de complexidade médio.

2.3 SIMULAÇÃO DE FALHAS

Dentre as diversas ferramentas computacionais de auxílio à elaboração de testes encontram-se os simuladores de falhas. A simulação de falhas consiste em simular o circuito na presença de falhas do modelo e comparar o seu comportamento com o esperado para o circuito livre de falhas. O objetivo principal é verificar se estas falhas são detectadas pelos estímulos de teste utilizados. O processo de simulação de falhas deve obedecer os seguintes passos:

1. simulação do circuito sem falhas;
2. redução da lista de falhas, eliminando falhas com o mesmo comportamento lógico;
3. injeção de falhas na descrição do circuito;
4. simulação do circuito com falha;
5. comparação dos resultados da simulação com falha e da simulação sem falhas. No caso de discordância, remoção da falha da lista inicial de falhas.

Um ambiente típico de simulação de falhas é mostrado na figura 2.1.

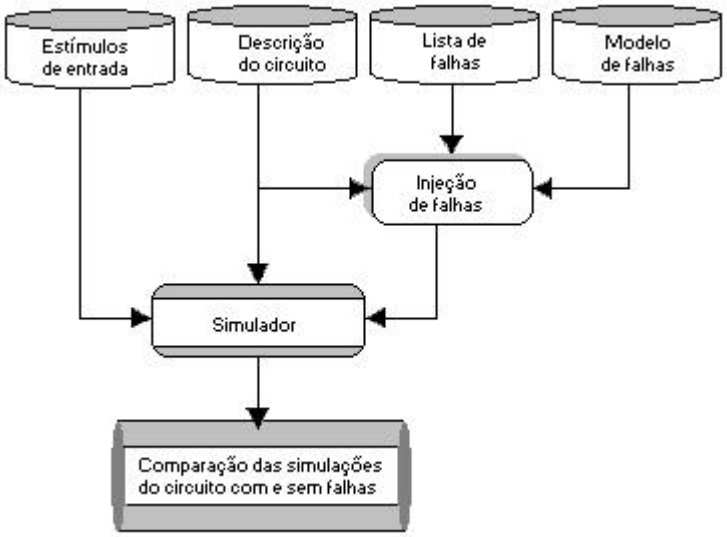

Figura 2.1 - Procedimento genérico de simulação de falhas.

A simulação de falhas pode ser também utilizada com os seguintes objetivos:

1. avaliação do teste, pela verificação da cobertura de falhas (percentual de falhas detectadas pelos estímulos de entrada aplicados);
2. a eliminação de falhas da lista original na geração automática de teste, verificando quais falhas do modelo são detectadas pelo estímulo de teste calculado; e,
3. o diagnóstico, pela construção do dicionário de falhas que identifica que falhas são detectadas por cada vetor de teste.

Existem várias técnicas de simulação de falhas digitais baseadas no modelo de falhas simples do tipo *stuck-at* (Seshu, 1965).

Na abordagem de simulação paralela (Seshu, 1965), o circuito sem falhas e um número limitado de circuitos com falhas são simulados em paralelo.Utilizando estruturas de dados, são armazenados simultaneamente todos os valores de sinais de um circuito com falhas e sem falha.

Na simulação de falhas dedutiva (Armstrong, 1972), somente o circuito sem falhas é simulado, enquanto o comportamento de um certo número de circuitos com falhas é deduzido. Uma lista de falhas, associada a um determinado nodo, determina todas as falhas, em um tempo de simulação, que fazem com que um nodo se comporte de forma diferente nos circuitos com falhas e sem falhas.

Na simulação de falhas concorrente (Ulrich, 1974), o circuito sem falhas é inteiramente simulado, enquanto que no circuito com falhas apenas a parte afetada pela falha injetada é simulada.

Pesquisas posteriores em termos de simulação de falhas têm focado principalmente aceleradores de *hardware* de propósito específico e algoritmos de simulação hierárquica de falhas (Abramovici, 1990).

2.4 GERAÇÃO DE TESTE

Após a escolha de um modelo apropriado e da simulação de falhas, a geração de vetores de teste é o passo natural para definir um procedimento eficiente de teste a ser aplicado a um circuito.

Quando esta geração de teste é feita através de um *software* de maneira automática tem-se a geração automática de padrões de teste, ATPG *(Automatic Test Pattern Generation*).

A geração de teste consiste, basicamente, em encontrar um conjunto de estímulos de entrada e de medidas de saídas, que garantam uma máxima cobertura de falhas. Se o objetivo da detecção de falhas é estendido para incluir o diagnóstico, o estímulo de teste deve ser capaz, adicionalmente, de distinguir falhas não equivalentes. Um ambiente típico de geração de vetores de teste é mostrado na figura 2.2.

Em circuitos digitais, a geração de vetores de teste pode ser feita através das abordagens exaustiva, pseudo-aleatória ou determinística (Abramovici, 1990). No teste exaustivo são consideradas a geração e aplicação de todos os vetores de entrada possíveis do circuito. Embora a geração exaustiva de teste seja trivial e assegure a maior cobertura de falhas possível, esta abordagem só pode ser aplicada a circuitos de baixa complexidade. Geradores especiais podem produzir vetores pseudo-aleatórios considerando diferentes vetores de inicialização (sementes) e diferentes tamanhos para as seqüências de teste. Este método produz, em geral, uma boa cobertura de falhas dentro de um tempo razoável de aplicação do teste. Não obstante, em muitos circuitos que necessitam de poucos vetores para conseguir uma alta cobertura de falhas, o uso do método pseudo-aleatório resultaria em seqüências de teste muito longas. Este é o caso de circuitos baseados em estruturas regulares.

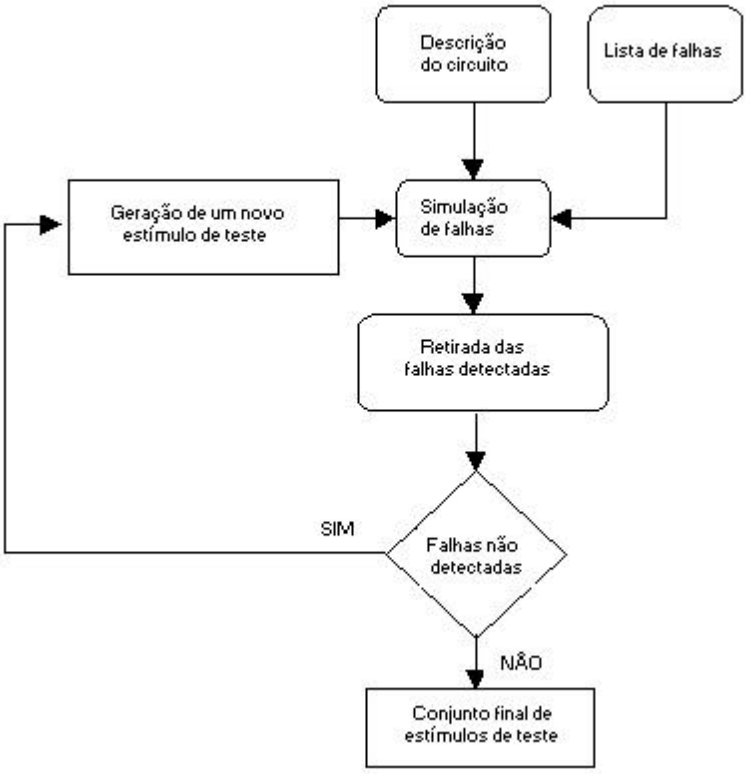

Figura 2.2 - Procedimento genérico de geração de teste.

Para resolver este problema, vetores podem ser pré-calculados a partir de ferramentas computacionais de geração de vetores de testes determinísticos e armazenados.

Dentre as várias abordagens para geração de teste, pode-se citar dois tipos de métodos que tratam a geração de vetores de teste de modo determinístico: os métodos algébricos e os métodos topológicos (Abramovici, 1990). Nos métodos algébricos, são computados vetores de teste a partir de expressões booleanas que descrevem o funcionamento do circuito sob teste. O método da diferença booleana é um método algébrico, a primeira vista bastante atraente pela sua simplicidade, concisão e formalismo. Entretanto, devido à complexidade computacional, a sua aplicabilidade se restringe a circuitos de pequno porte. Nos métodos topológicos, os vetores de teste são derivados a partir da estrutura do circuito. O processo de geração topológica de teste mais conhecido é o algoritmo D. O algoritmo D é um precursor dos ATPGs e foi proposto em (Roth, 1967). Desde então outros algoritmos e verificações surgiram. O algoritmo D, além de 0 e 1, utiliza os valores D e /D para distinguir o comportamento do circuito sem falhas do comportamento do circuito com falha. D e /D denotam 1 e 0 como comportamento correto e 0 e 1

como o comportamento do nodo defeituoso. Essencialmente, o algorítmo D começa propagando uma única falha *stuck-at* (denotada por D ou /D) para uma saída primária do circuito. A seguir, justifica as entradas do circuito de forma a produzir no nodo com falha o comportamento oposto ao previsto pela falha injetada. O algoritmo D foi desenvolvido visando primeiramente o teste de circuitos combinacionais, e mais tarde foi utilizado para gerar vetores para circuitos seqüenciais. Alguns exemplos de ferramentas para geração de testes bem estabelecidas são PODEM (Goel, 1981) e FAN (Fujwara, 1983), ambas baseadas em aprimoramentos do algorítmo D.

As últimas contribuições sobre o assunto têm se dedicado à geração de testes hierárquicos, geração de vetores a partir da descrição em alto nível do circuito (Murray, 1996), geração de estímulos de testes para circuitos seqüênciais (Marchok, 1995), e geração de teste que considera outros modelos de falhas como, por exemplo, falhas de atraso (Brackel, 1995), e outros modelos como teste de corrente (Nigh, 1989).

2.5 PROJETO VISANDO O TESTE

O crescente desenvolvimento da tecnologia exige o emprego de sistemas cada vez mais complexos. Este fato faz com que qualquer projeto que não tenha preocupação com teste tenha sérios problemas nas fases de prototipação, produção e manutenção. Por outro lado, os testes desses sistemas se tornam cada vez mais difíceis e caros, e o compromisso entre cobertura de falhas e tempo para geração de teste fica ameaçado, mesmo contando com ferramentas de geração de vetores de teste. Nos dias atuais é aceito que o re-projeto de partes do circuito pode facilitar a preparação, a aplicação e a avaliação do teste (*design-for-test*). O re-projeto pode ser mais ou menos complicado conforme se queira transferir mais ou menos funções do testador para o próprio circuito a testar. Basicamente, as técnicas de projeto visando a testabilidade transferem para dentro do *chip* mecanismos de acesso (*ponteiras eletrônicas*), as técnicas de auto-teste transferem o gerador e o avaliador de respostas de teste, e a auto-verificação *(self-checking)* transfere um avaliador de respostas capaz de detectar erros concorrentemente à aplicação do circuito. Estas três técnicas serão detalhadas nas seções que seguem.

2.5.1. PROJETO VISANDO A TESTABILIDADE

Uma abordagem para reduzir a dificuldade de geração de teste é considerar a testabilidade de um circuito tão cedo quanto possível e assim melhorar esta característica. Para se alcançar esse objetivo é necessário que haja um meio de conhecer o grau de dificuldade para gerar teste e as áreas do circuito pobres em testabilidade. Isto é chamado *Análise de Testabilidade*.

A análise de testabilidade requer uma *quantificação* para que a testabilidade de um circuito possa ser avaliada da maneira mais correta possível. A medida da testabilidade precisa indicar o grau de facilidade ou a dificuldade de geração e teste em um circuito, de forma que possa ser interpretado pelo projetista. As informações produzidas para a medida da testabilidade são usadas para modificar ou reprojetar circuitos. Se a computação das informações para a medida da testabilidade for rápida e prever as dificuldades de geração de teste, então será proveitosa na redução do custo do teste.

Muitos métodos para mensurar testabilidade em circuitos lógicos foram propostos (Stepheson; Grason, 1976; Keiner; West, 1977; Dejka, 1977; Dussaut, 1978; Wood, 1979; Kovijanic, 1979; Bennetts at al. 1981; Berg; Hess, 1981; Brglez, 1984). Estes métodos introduzem dois parâmetros em comum para estimar a testabilidade: a controlabilidade e a observabilidade.

C*ontrolabilidade* é definida como a medida da facilidade com que os circuitos lógicos podem ser controlados de suas entradas primárias.

Observabilidade é definida como a medida da facilidade com que a lógica interna de um circuito pode ser observada a partir de suas saídas primárias.

Uma abordagem direta para aumentar a controlabilidade e a observabilidade de um circuito é usar pontos de teste com a finalidade de permitir que sinais internos sejam controlados e observados. Uma boa medida de testabilidade indica a dificuldade de geração de teste no circuito. Esta análise de testabilidade pode identificar as áreas de pobre controlabilidade e pobre observabilidade e, conseqüentemente, o projetista pode saber onde somar entradas de controle e saídas observáveis para aumentar a testabilidade.

Usando um ponto de teste como uma entrada primária pode aumentar a controlabilidade de um circuito e usando um ponto de teste como uma saída primária pode aumentar a observabilidade do circuito. Em alguns casos, um único pino pode ser usado para ambas, uma entrada e uma saída. Várias técnicas para identificar como e onde adicionar estes

pontos de teste foram estudadas por (Hayes, 1974; Hayes; Friedman, 1974; Saluja; Reddy, Embora essas técnicas tenham alcançado um alto nível de testabilidade, elas requerem implementação com muitas portas e pinos extras.

Para circuitos seqüenciais, em geral, esses problemas de controlabilidade e observabilidade são tão difíceis que eles não podem ser resolvidos de maneira razoável, se for desejada uma alta cobertura de falhas. Porém, este problema pode ser resolvido completamente se for adotada uma abordagem tal que os circuitos sejam projetados de forma que seja fácil fixar o circuito em um estado desejado e fácil de observar seu estado interno. Estas propriedades referem-se à controlabilidade e observabilidade dos *flip-flops* ou *latches* em circuitos seqüenciais.

Várias técnicas de projeto têm sido utilizadas para dar a capacidade de controlar e observar o estado interno de um circuito. Uma representante destas técnicas é a abordagem de projeto *scan* usando *shift-register*. (Aqui, o termo *scan* refere-se à *habilidade de deslocamento para dentro ou para fora, de qualquer estado de um circuito*). A figura 2.3 mostra uma técnica de projeto *scan* proposta em (Williams; Angell, 1973).

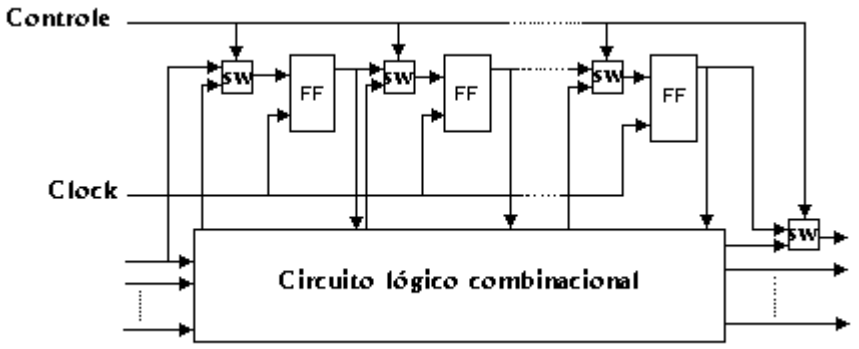

Figura 2.3 – Teste *scan* proposto por Williams e Angell, 1973.

Nesta figura, o circuito é projetado de forma que ele tem dois modos de operação: um modo função normal e um modo *shift-register*. Um sinal de controle pode chavear estes dois modos. No modo *shift-register*, todos os *flip-flops* do circuito são conectados em uma cadeia para se comportar como *shift-register*. Então, é possível deslocar um padrão de teste arbitrário para dentro dos *flip-flops*, e deslocar para fora o conteúdo dos *flip-flops*. O modo *shift-register* torna possível controlar e observar facilmente o estado interno. Usando este modo o problema de teste

para circuito seqüencial pode ser reduzido a um para circuitos combinacionais. O procedimento para testar tais circuitos é aproximadamente como segue:

1. Chavear no modo *shift-register* e carregar o estado inicial para um padrão de teste nos *flip-flops*.

2. Retornar ao modo de função normal e aplicar o padrão de teste à entrada.

3. Chavear no modo *shift-register* e trocar o estado final fixando o estado inicial para o próximo teste.

4. Como mostra esse procedimento, podemos projetar qualquer circuito seqüencial de forma que ele seja tratado como puramente combinacional, fazendo uso de técnicas de projeto *scan*.

2.5.2. AUTO-TESTE

Outra possibilidade para o projeto visando o teste é a concepção de circuitos auto-testáveis fora de funcionamento. Esta é uma alternativa para atender às exigências advindas dos avanços tecnológicos da integração de sistemas. Estes exigem testes mais eficientes e rápidos, e testadores que operem em altas freqüências, que ofereçam precisão, capacidade de memória, imunidade ao ruído, etc. Estes testadores, em geral, têm preço muito elevados. A técnica do auto-teste consiste em integrar algumas ou todas as funções do testador no *chip* propriamente dito, ou na placa sobre a qual os *chips* estão montados. Na produção de circuitos integrados em larga escala, o uso do auto-teste integrado (BIST - *Built-In Self-Test)* é aconselhável, a fim de reduzir o custo do teste de produção por *chip*.

Idealmente, uma arquitetura BIST é composta por três blocos de *hardware* além do circuito a ser testado. Estes blocos são: um gerador de padrões de teste (TPG), um analisador de resposta do teste (ORA) e um controlador de teste, como mostra a figura 2.4.

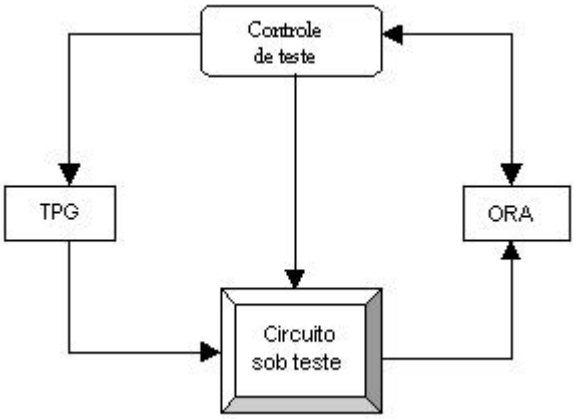

Figura 2.4. Arquitetura BIST

Nesta estrutura, o gerador de estímulos de teste (TPG) recebe um sinal do controlador de teste, ficando habilitado a produzir uma seqüência de valores que serão aplicados às entradas primárias do circuito a testar. Em se tratando de teste exaustivo ou pseudo-exaustivo, um contador pode ser utilizado como TPG. Quando o teste for determinístico, os padrões de teste podem estar armazenados em uma memória ROM, por exemplo. Qualquer implementação em hardware de um gerador de números pseudo-aleatórios também pode vir a ser utilizada como TPG.

As respostas aos estímulos de teste são coletadas pelo analisador de respostas de teste (ORA). O resultado é enviado ao controlador de teste que pode apenas sinalizar a presença de falhas ou até mesmo interagir com um controlador de mais alto nível. Os valores esperados podem ser previamente armazenados em uma ROM, para posteriormente serem comparados com as saídas do circuito sob teste. Este procedimento exige uma capacidade de armazenamento de dados considerável para guardar todas as saídas corretas associadas aos vetores de teste aplicados. Para minimizar este problema pode-se armazenar estas informações de forma compacta. A compactação ou compressão de resposta de teste é o processo de redução da resposta do circuito que gera o que se chama assinatura. Entre as técnicas de compressão mais comuns podem ser citadas: contagem de números de 1's; a contagem do número de transições de 1 para 0 ou de 0 para 1; a integração digital (*checksum*) e a divisão polinomial (Abramovici 1990).

A escolha do tipo de TPG e de ORA deve levar em consideração a cobertura de falhas desejada, o acréscimo de área tolerada, o número de pinos adicionais e a influência do BIST no desempenho do circuito. O acréscimo da área para a inclusão do BIST no *chip* tem como conseqüência um

provável aumento de peças com defeito na linha de produção, uma vez que a probabilidade de defeito de fabricação aumenta com a área final do circuito. Outra penalidade ocasionada pela integração do BIST é a inclusão de caminhos de atraso, o que pode diminuir o desempenho do circuito. Todas estas implicações devem ser recompensadas pela redução dos custos de teste e manutenção proporcionadas pelo BIST.

2.5.3. CIRCUITOS SELF-CHECKING

Circuitos digitais, onde a capacidade de detecção concorrente de falhas é conseguida através de blocos funcionais, que produzem saídas codificadas, e blocos verificadores, que conferem se as saídas estão de acordo com código de detecção de erros utilizado, são chamados circuitos *self-checking*. Os códigos de detecção de erros mais utilizados são os de paridade, o de Berger, o código dual (*double-rail*), a duplicação, etc.(Abramovici, 1990). Quando se utiliza um código corretor de erro e um bloco verificador capaz de restaurar a saída correta do bloco funcional, falhas podem ser toleradas pelo sistema resultante. Sistemas triplicados e votadores implementam este tipo de tolerância a falhas.

É de particular interesse para este trabalho o código de duplicação. A duplicação consiste em se replicar o circuito, ou sistema, total ou parcialmente. Utilizando os mesmos sinais de entrada, o bloco funcional e a sua cópia conduzem suas saídas a um circuito comparador que verifica possíveis diferenças, acusando a presença de falhas (figura 2.5). A duplicação pode ser parcial, na medida em que a réplica é programável e pode ser compartilhada, no tempo, com outros blocos funcionais que precisam ser testados. Este caso será avaliado mais detalhadamente ao longo deste trabalho.

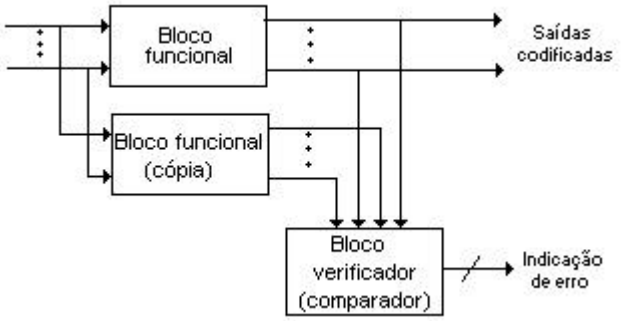

Figura 2.5. Circuito *self-checking* baseado no código de duplicação.

Freqüentemente, os circuitos *self-checking* devem atender ao princípio *totally self-checking*, no qual é considerado que a primeira saída errônea do bloco funcional deve resultar em uma indicação de erro na saída do bloco verificador. As propriedades básicas necessárias para se atender a este princípio são independentes da forma de implementação do circuito. Estas propriedades foram inicialmente definidas em (Carter, 1968) e posteriormente refinadas em (Anderson 1971; Smith, 1978; Nicolaidis, 1988). Em geral, a eficiência dos circuitos *totally self-checking* é baseada na hipótese de que as falhas ocorrem uma de cada vez e que, entre a ocorrência de duas falhas quaisquer, há um período de tempo suficiente para que o bloco funcional e o bloco verificador recebam todos os elementos de seus respectivos espaços de código de entrada.

3. TESTE EM FUNCIONAMENTO DE UMA MATRIZ DE ROTEAMENTO

Este trabalho foca a implementação do teste em funcionamento de uma matriz de roteamento de sinais, destinada a fazer parte de um enlace ótico para canais telefônicos. Nas seções que seguem descrevem-se o circuito sob teste, os experimentos de detecção de falhas realizados e a técnica proposta para a tolerância a falhas.

3.1 ARQUITETURA

A matriz de roteamento é um circuito composto por quatro células idênticas projetada para ser implementada em tecnologia NMOS 5µm com transistores de carga tipo depleção, desenvolvida pelo Instituto de Física da Universidade Federal do Rio Grande do Sul. Cada célula é dotada de dois terminais de entrada, dois terminais de saída e um terminal para o sinal de controle, sendo que a interligação adequada destas quatro células produz uma matriz com quatro entradas e quatro saídas. A figura 3.1 mostra o diagrama em blocos da matriz de roteamento.

Os sinais de controle, responsáveis pelo roteamento, podem assumir dois valores distintos (0 ou 1). Ao serem realizadas combinações com os sinais nos quatro terminais de controle, torna-se possível que um sinal ao chegar em qualquer uma das quatro entradas possa ser conduzido a qualquer uma das quatro saídas.

A estrutura interna de cada um dos quatro células da matriz é composta por 12 transistores NMOS interligados de forma a realizar a função acima descrita. O diagrama esquemático de uma das células que compõem a matriz pode ser visto na figura 3.2**.** Baseado no fato de que as quatro células possuem a mesma estrutura interna, as simulações foram feitas em apenas um das quatro células do circuito,utilizando o *HSPICE* versão 6.3, sendo os resultados das simulações feitas com essa célula considerados válidos para as demais células da matriz.

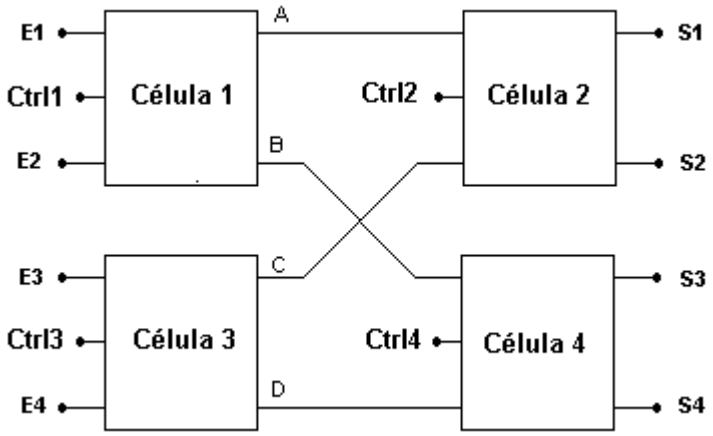

Figura 3.1. Diagrama em blocos da matriz de roteamento.

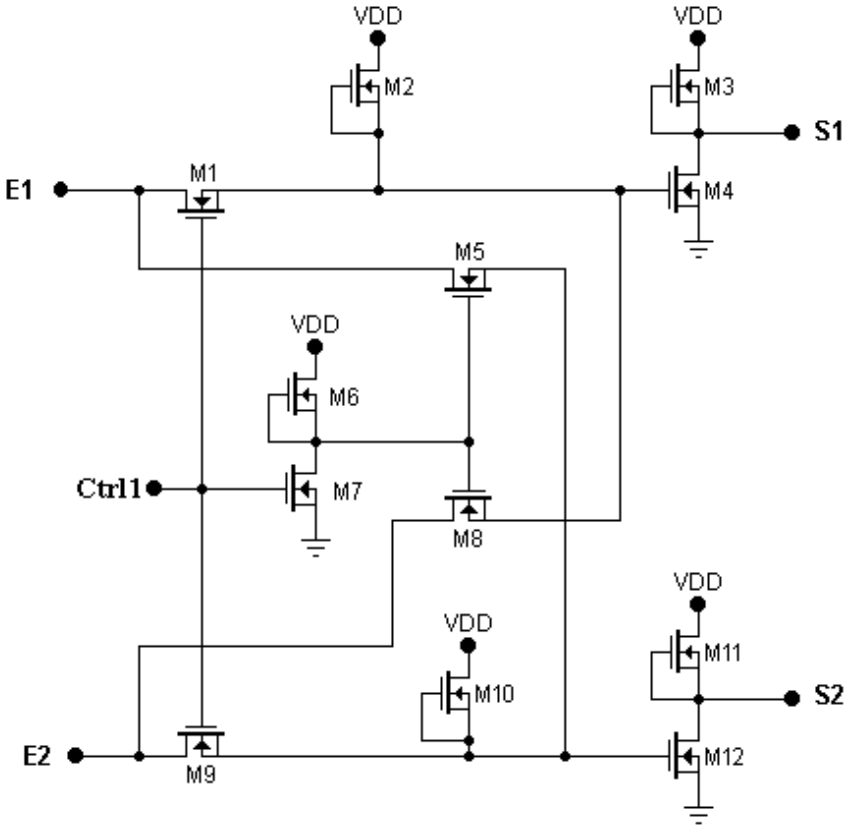

Figura 3.2. Diagrama esquemático de uma das células do circuito.

3.2 MODELO DE FALHAS E METODOLOGIA DE TESTE

O modelo mais adequado para circuitos descritos em nível de transistor é aquele composto por: curto-circuito entre dois terminais quaisquer do transistor NMOS e circuito aberto em todos os terminais (figura 3.3). Este modelo tem se mostrado mais representativo que o modelo *stuck-on/stuck-open* dos defeitos físicos que podem ocorrer em um circuito integrado (Kolarík, 1994). Para a simulação de falhas do tipo curto-circuito costuma-se usar um resistor de baixo valor (1 Ω) e para falhas do tipo circuito aberto um resistor de valor elevado (100 MΩ).

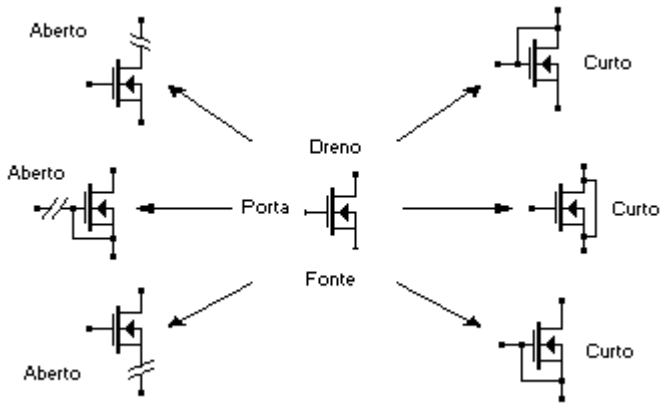

Figura 3.3. Modelo de falhas utilizado.

A metodologia de teste utilizada para a investigação do comportamento e de métodos de reconfiguração da matriz na presença de falhas é baseada na seguinte seqüência de passos:

1. Simulação de falhas e ensaio de detecção sobre o menor módulo replicável;
2. Generalização das técnicas de detecção às combinações do menor módulo;
3. Definição de estratégia de diagnóstico automático;
4. Definição de técnicas de isolamento e substituição do módulo defeituoso.

Estes passos são discutidos na seqüência.

3.3 ENSAIOS DE DETECÇÃO DE FALHAS

Para efeito de simulação de falhas, inicialmente aplicou-se na entrada do circuito sem falhas um sinal com características conhecidas para se ter um padrão de saída. Posteriormente, simulou-se uma falha por vez em cada transistor NMOS. Para cada falha simulada aplicou-se o sinal na entrada do circuito e comparou-se a saída com a do circuito sem falhas, a fim de verificar a existência de diferença entre os sinais de saída. As figuras 3.4 e 3.5 mostram as saídas do circuito sem falhas e com um curto-circuito entre o dreno e a fonte do transistor M5 da célula 1 (M5 -1), com o controle em 5 volts.

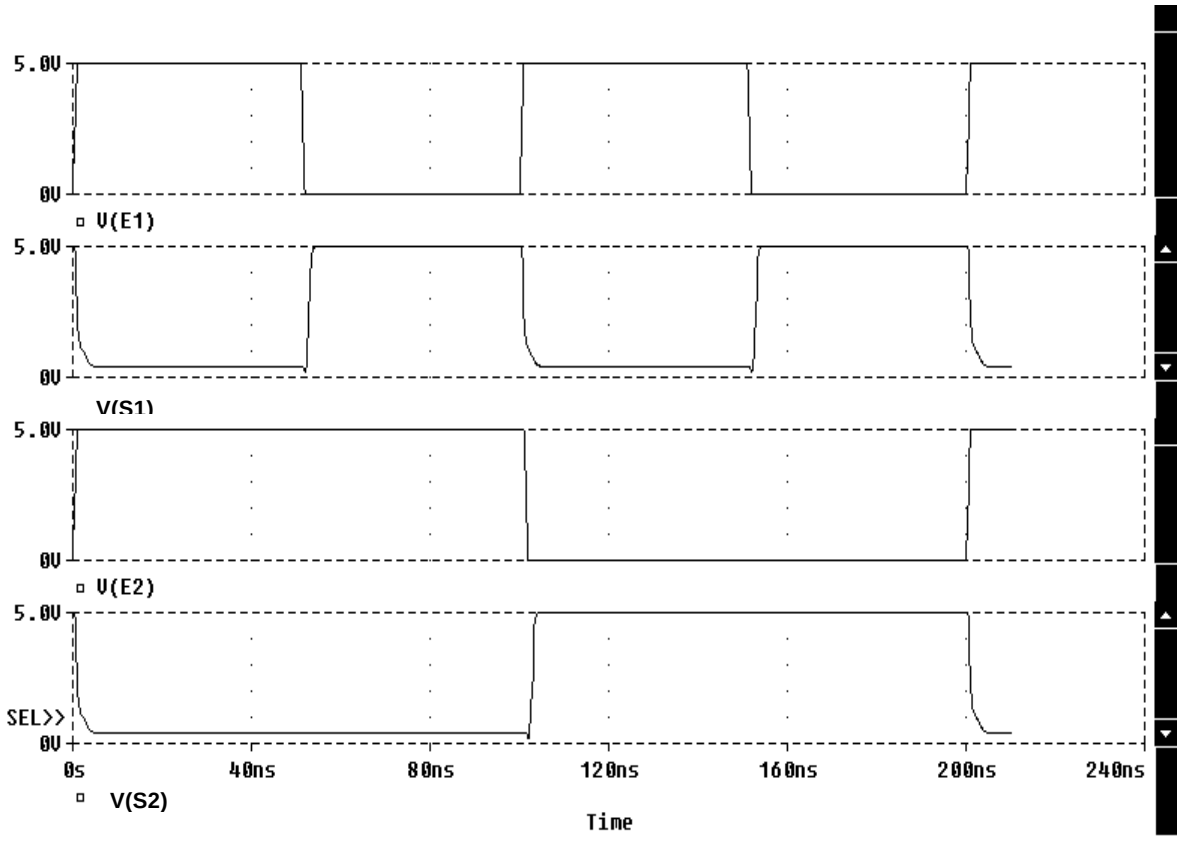

Figura 3.4. Simulação do circuito sem falhas

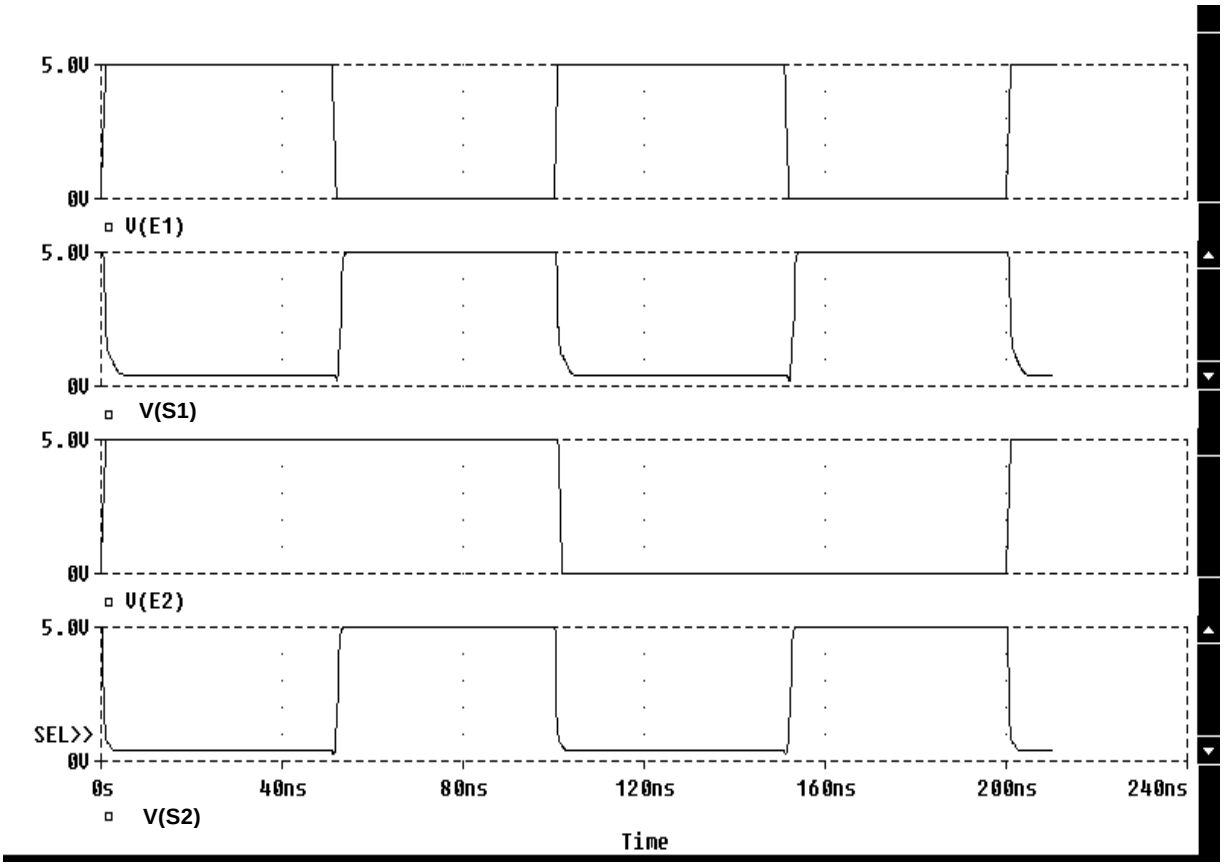

Figura 3.5. Simulação da célula 1 na presença de um curto-circuito entre dreno e fonte do transistor M5-1 (controle em 5 volts).

Os resultados obtidos nas simulações de circuitos abertos e curtos-circuitos são apresentados nas tabelas 3.1 e 3.2.

Os resultados obtidos com as simulações de circuitos abertos e curtos-circuitos mostraram que a grande maioria das falhas foram detectadas, isto é, produziram alterações na tensão de saída. Entretanto, para um pequeno número de falhas, essas alterações são praticamente imperceptíveis ou inexistem. Este fato conduziu a uma observação do comportamento da corrente na presença das falhas que não foram detectadas pela observação da tensão na saída. Foi constatado, nestes casos, que há uma alteração significativa na corrente. Numa comparação entre as figuras 3.6 e 3.7, pode ser observado que a intensidade da corrente no circuito sem falhas é

praticamente nula, e no circuito com um curto-circuito entre a porta e a fonte do transistor M7-1 este valor se eleva para 5 A.

Tabela 3.1. Simulação de circuitos abertos.

		CONTROLE EM 5V		CONTROLE EM 0V	
		DETECTA	NÃO DETECTA	DETECTA	NÃO DETECTA
M1 -1	D	X			X
	P	X			X
	F	X			X
M2 -1	D		X		X
	P	NS	NS	NS	NS
	F		X		X
M3 -1	D	X		X	
	P	NS	NS	NS	NS
	F	X		X	
M4 -1	D	X		X	
	P	X		X	
	F	X		X	
M5 -1	D		X	X	
	P		X	X	
	F	X			X
M6 -1	D		X	X	
	P	NS	NS	NS	NS
	F		X		X
M7 -1	D	X			X
	P	X			X
	F	X			X
M8 -1	D		X	X	
	P		X	X	
	F	X		X	
M9 -1	D	X			X
	P	X			X
	F	X			X
M10 -1	D		X		X
	P	NS	NS	NS	NS
	F		X		X
M11 -1	D		X		X
	P	NS	NS	NS	NS
	F		X		X
M12 -1	D	X		X	
	P	X		X	
	F	X		X	

D = Dreno P = Porta

F = Fonte NS = Não simulado (falha = função especificada)

Tabela 3.2. Simulação de curtos-circuitos.

		CONTROLE EM 5V		CONTROLE EM 0V	
		DETECTA	NÃO DETECTA	DETECTA	NÃO DETECTA
	DF		X	X	
M1 -1	DP		X		X
	FP	X		X	
	DF	X		X	
M2 -1	DP	X		X	
	FP	NS	NS	NS	NS
	DF	X		X	
M3 -1	DP	X		X	
	FP	NS	NS	NS	NS
	DF	X		X	
M4 -1	DP	X		X	
	FP	X		X	
	DF	X		X	
M5 -1	DP	X		X	
	FP	X		X	
	DF	X			X
M6 -1	DP	X			X
	FP	NS	NS	NS	NS
	DF		X	X	
M7 -1	DP	X		X	
	FP		X		X
	DF	X			X
M8 -1	DP	X		X	
	FP	X		X	
	DF	X		X	
M9 -1	DP		X		X
	FP	X		X	
	DF	X		X	
M10 -1	DP	X		X	
	FP	NS	NS	NS	NS
	DF	X		X	
M11-1	DP	X		X	
	SP	NS	NS	NS	NS
	DF	X		X	
M12-1	DP	X		X	
	FP	X		X	

D = Dreno P = Porta

F = Fonte NS = Não simulado (falha = função especificada)

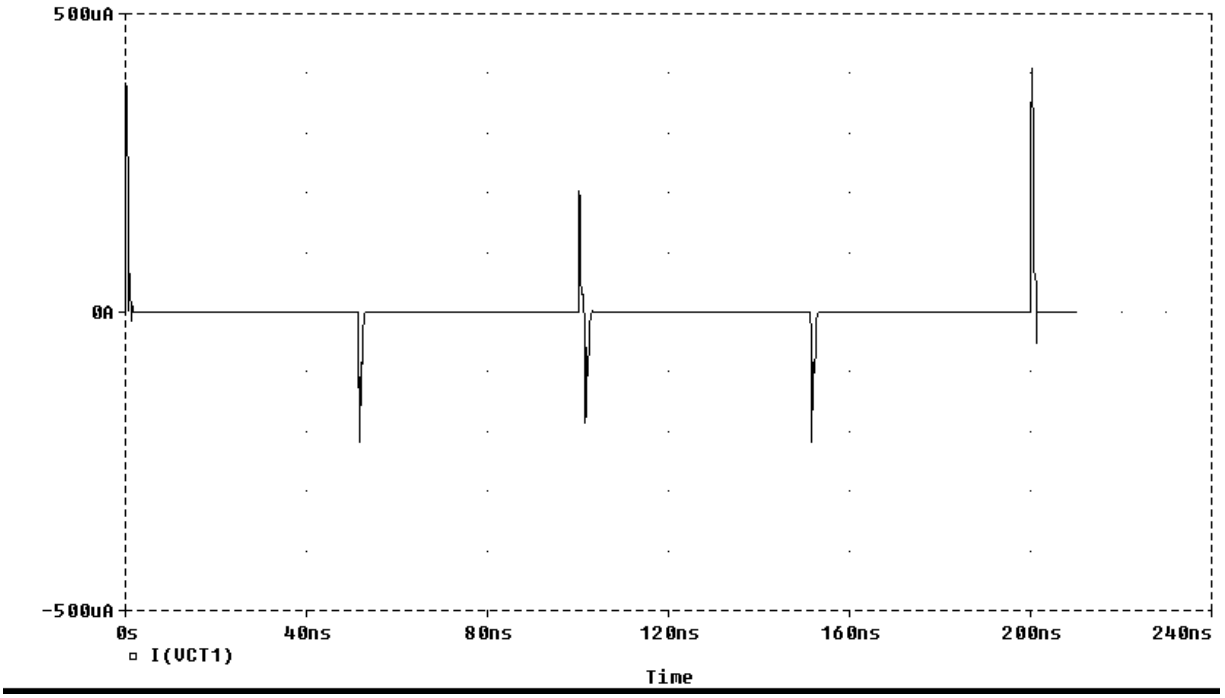

Figura 3.6. Corrente na fonte de controle (CTRL1) sem falha.

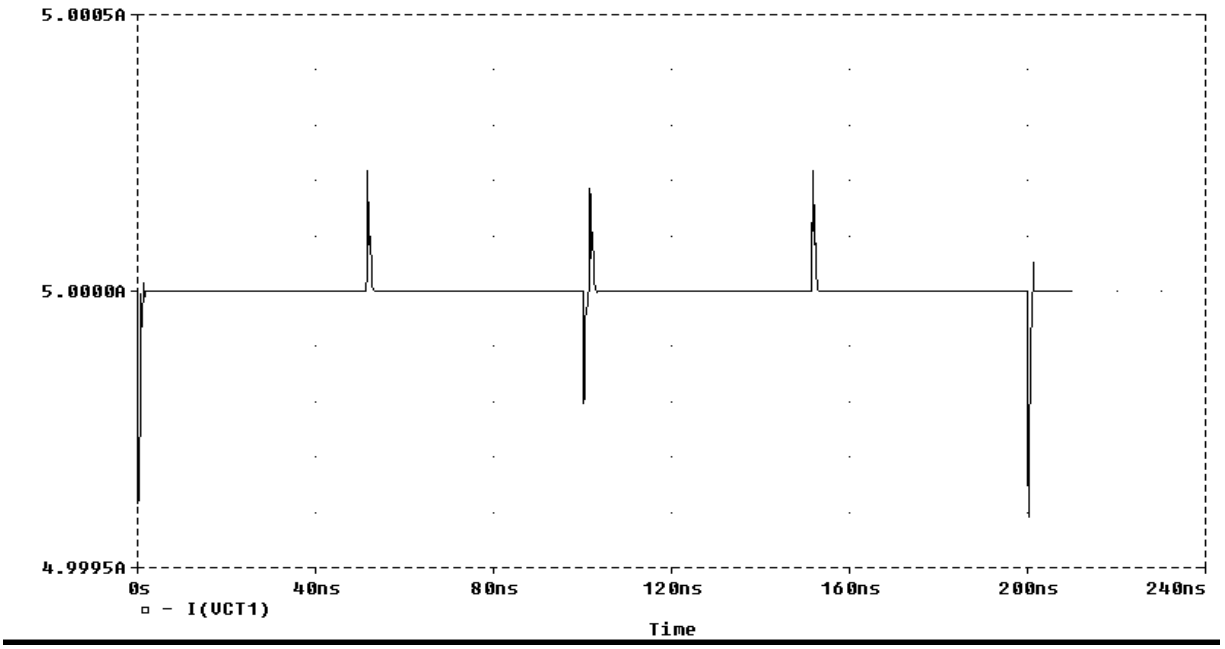

Figura 3.7. Corrente na fonte de controle (CTRL1) com um curto-circuito entre porta e fonte do transistor M7-1.

3.4 TOLERÂNCIA A FALHAS

Com o conhecimento do comportamento do circuito na presença de falhas foi possível desenvolver uma estratégia de teste capaz de proporcionar ampla cobertura de falhas, levando a uma arquitetura reconfigurável capaz de tornar o circuito de chaveamento tolerante a falhas.

Fazendo uso da abordagem da duplicação, foi possível pensar em um sistema no qual, uma vez detectada uma falha, a parte do circuito que a provocou possa ser substituída por outra equivalente, devolvendo ao sistema o desempenho das suas funções normais. O mecanismo em *hardware* que permitiu detectar as falhas segundo a abordagem da duplicação é mostrado na figura 3.8.

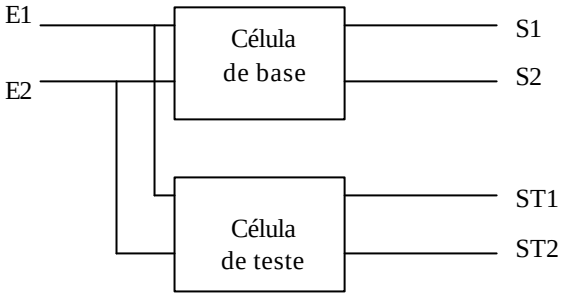

Figura 3.8. Mecanismo para detecção de falhas por duplicação.

Esgotadas todas as possibilidades de simulação e detecção de falhas em nível de transistor e com o auxílio do mecanismo apresentado na figura 3.8, a atenção volta-se para a investigação de falhas de interação.

São consideradas falhas de interação as falhas provocadas por curto-circuito entre diferentes elementos da mesma célula ou entre células diferentes. Para esse tipo de falha o procedimento de simulação adotado foi o seguinte:

Uma falha de interação foi simulada em uma das células de base. Na entrada desta célula de base e na entrada da célula teste, injeta-se, simultaneamente, um mesmo sinal com características conhecidas. Observando-se todas as saídas das quatro células foi identificado em que parte do circuito a falha se manifestava. Essa operação foi repetida para todas as situações possíveis nas quatro células do circuito de roteamento. Os resultados dessas simulações estão impressos na tabela 3.3. Analisando estas tabelas podemos observar que as falhas que não puderam ser detectadas por variação de tensão foram detectadas por variação de corrente, dando uma cobertura de 100%.

Tabela 3.3. Detecção por variação de tensão e de corrente.

	POR TENSÃO		**POR CORRENTE**	
Curto Entre :	**DETECTA**	**NÃO DETECTA**	**DETECTA**	**NÃO DETECTA**
E1 e E2		X	X	
E3 e E4		X	X	
A e B	X			
C e D	X			
A e C	X			
B e C	X			
B e D	X			
S1 e S2	X			
S3 e S4	X			

4. PROJETO DA MATRIZ TOLERANTE A FALHAS

Com o objetivo de manter o roteador em operação, mesmo na presença de alguma falha, foi projetado um circuito com a finalidade de fazer automaticamente a detecção e a correção dos erros. Para isso foi acrescentado um sistema de multiplexação, um comparador e um circuito de controle ao mecanismo apresentado na figura 3.8. O circuito resultante está apresentado na figura 4.1.

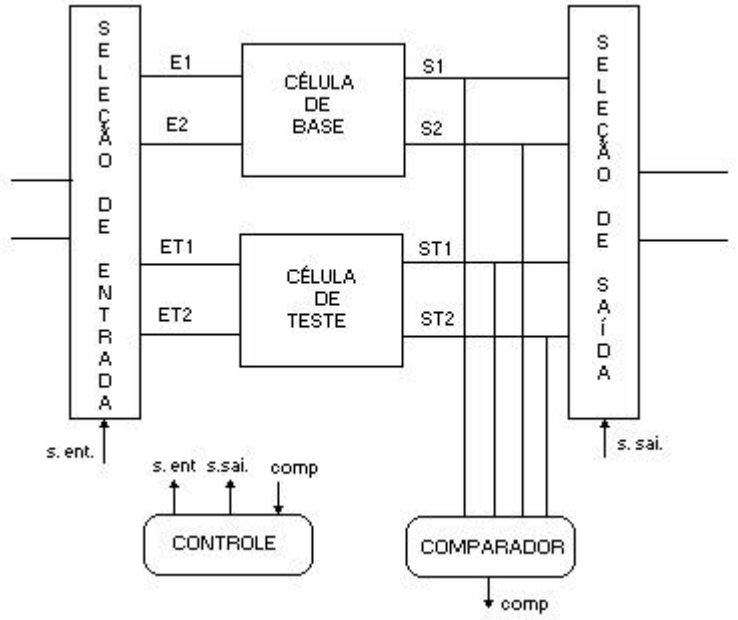

Figura 4.1. Mecanismo para detecção e correção de erro.

Como pode ser observado, este mecanismo consiste em se colocar uma célula de teste, idêntica às que compõem o circuito do roteador, em paralelo com cada uma das células de base do circuito para que, submetidas às mesmas entradas, seja feita a comparação entre os sinais em suas saídas. O comparador recebe em suas entradas os sinais provenientes das saídas da célula de base e da célula de teste, compara as saídas e envia o resultado para o circuito de controle.

Se não houver discrepância entre as duas saídas, será gerado um sinal para indicar esta situação e a célula de teste será colocada em paralelo com outra célula de base do circuito

para que novamente sejam verificados os sinais em suas saídas. Este processo continua se repetindo indefinidamente se o comparador não detectar nenhuma diferença entre os sinais nas saídas das células de base e da célula de teste.

Caso o comparador detecte alguma discrepância entre as saídas das células, emite um sinal de erro para indicar que a célula sob teste está com defeito. Neste caso, a célula defeituosa deve ser desconectada, para ser substituída pela célula de teste, até que uma outra célula padrão seja colocada no lugar da célula danificada para que a célula de teste retome suas funções de supervisionar o funcionamento do circuito.

4.1 IMPLEMENTAÇÃO UTILIZANDO COMPONENTES DISCRETOS

Com o objetivo de validar a estratégia de tolerância a falhas aqui proposta, a matriz de roteamento e o respectivo mecanismo de teste foram implementados em placa de circuito impresso tomando como base os integrados 4007, cuja estrutura interna é composta por transistores MOS, e o 4066 que tem a função de chave analógica.

O CI 4066, quando recebe um sinal de nível alto (1) em um de seus terminais**,** funciona como um curto-circuito (chave fechada) e, se o sinal for de nível baixo (0), funcionará como circuito aberto (chave aberta). A figura 4.2 mostra o esquemático de uma célula de base implementada. A figura 4.3 mostra o circuito completo da matriz de roteamento, incluindo o circuito para substituição de células, implementados com os integrados acima citados.

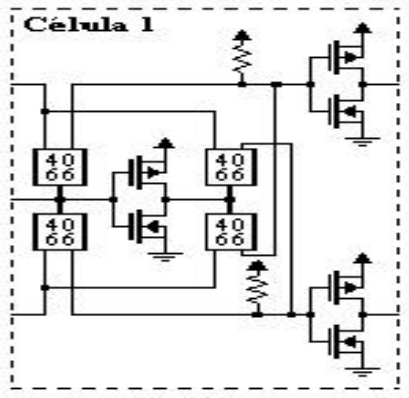

Figura 4.2. Esquemático de uma das células de base após modificação.

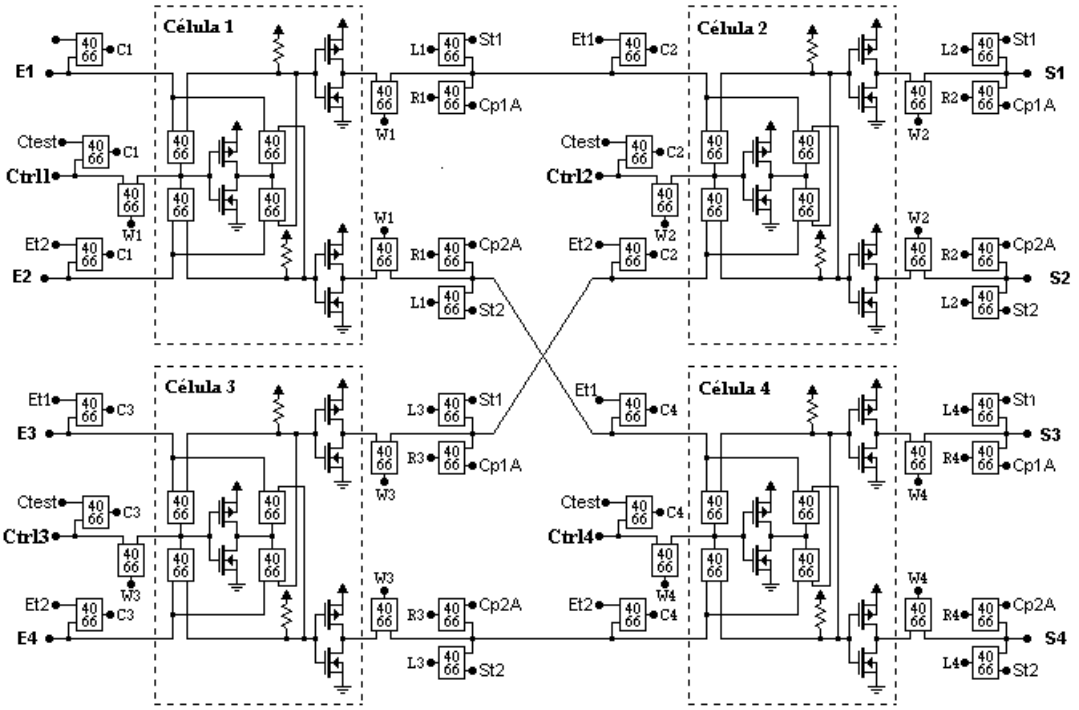

Figura 4.3. Esquemático completo da matriz com o circuito para substituição de célula.

Os primeiros testes sobre a matriz discreta foram realizados acionando manualmente as chaves analógicas contidas nos CIs 4066. Esta implementação permitiu que as falhas fossem testadas em um circuito real, comprovando assim, os resultados obtidos anteriormente através de simulação. Na seqüência, partiu-se para a automatização da varredura, detecção, sinalização e substituição da célula defeituosa. Abaixo apresenta-se o algoritmo para realização destas funções.

Para elaboração deste algoritmo foram consideradas as seguintes convenções:

C – chave que conecta a entrada de cada célula do roteador à entrada da célula de teste;

L – chave que conecta a célula de teste no lugar da célula defeituosa;

R – chave que conecta a saída da célula de base à entrada do comparador;

W – chave que conecta a célula de base ao circuito roteador.

$$\left.\begin{array}{l} C = 1 \\ L = 1 \\ R = 1 \\ W = 1 \end{array}\right\}$$ chaves fechadas

$$\left.\begin{array}{l} C = 0 \\ L = 0 \\ R = 0 \\ W = 0 \end{array}\right\}$$ chaves abertas

S = 1 Saída sem defeito

S = 0 Saída com defeito.

No momento em que o circuito é ligado, a célula de teste estará com suas entradas conectadas em paralelo com as entradas de uma das células do roteador. As saídas de ambas estarão ligadas às entradas do comparador.

Supondo que a célula de base com a qual a célula de teste está em paralelo seja a célula 1, teremos as seguintes situações:

A combinação das chaves será:

C1 = 1 L1 = 0 R1 = 1 W1 = 1
C2 = 0 L2 = 0 R3 = 0 W2 = 1
C3 = 0 L3 = 0 R3 = 0 W 3= 1
C4 = 0 L4 = 0 R4 = 0 W4 = 1

Se a saída do comparador indicar alguma falha (S = 0), a supervisão é interrompida, é gerado um sinal para indicar esta situação e a célula 1 é substituída com a seguinte combinação de abertura e fechamento das chaves:

C1 = 1 L1 = 1 R1 = 0 W1 = 0
C2 = 0 L2 = 0 R3 = 0 W2 = 1
C3 = 0 L3 = 0 R3 = 0 W 3= 1
C4 = 0 L4 = 0 R4 = 0 W4 = 1

Se a saída não apresentar falha (S = 1), a célula de teste passa a supervisionar a célula 2 e a combinação de chaves será:

C1 = 0 L1 = 0 R1 = 0 W1 = 1
C2 = 1 L2 = 0 R3 = 1 W2 = 1
C3 = 0 L3 = 0 R3 = 0 W 3= 1
C4 = 0 L4 = 0 R4 = 0 W4 = 1

Se a saída do comparador indicar alguma falha (S = 0), a supervisão é interrompida, é gerado um sinal para indicar esta situação e a célula 2 é substituída com a seguinte combinação de chaves:

C1 = 0 L1 = 0 R1 = 0 W1 = 1
C2 = 1 L2 = 1 R3 = 0 W2 = 0
C3 = 0 L3 = 0 R3 = 0 W 3= 1
C4 = 0 L4 = 0 R4 = 0 W4 = 1

Se a saída não apresentar falha (S = 1), a célula de teste passa a supervisionar a célula 3 e a combinação de chaves será:

C1 = 0 L1 = 0 R1 = 0 W1 = 1
C2 = 0 L2 = 0 R3 = 0 W2 = 1
C3 = 1 L3 = 0 R3 = 1 W 3= 1
C4 = 0 L4 = 0 R4 = 0 W4 = 1

Se a saída do comparador indicar alguma falha (S = 0), a supervisão é interrompida, é gerado um sinal para indicar esta situação e a célula 3 é substituída com a seguinte combinação de chaves:

C1 = 0 L1 = 0 R1 = 0 W1 = 1
C2 = 0 L2 = 0 R3 = 0 W2 = 1
C3 = 1 L3 = 1 R3 = 0 W 3= 0
C4 = 0 L4 = 0 R4 = 0 W4 = 1

Se a saída não apresentar falha (S = 1), a célula de teste passa a supervisionar a célula 4 e a combinação de chaves será:

C1 = 0 L1 = 0 R1 = 0 W1 = 1
C2 = 0 L2 = 0 R3 = 0 W2 = 1
C3 = 0 L3 = 0 R3 = 0 W 3= 1
C4 = 1 L4 = 0 R4 = 1 W4 = 1

Se a saída do comparador indicar alguma falha (S = 0), a supervisão é interrompida, é gerado um sinal para indicar esta situação e a célula 4 é substituída com a seguinte combinação de chaves:

C1 = 0 L1 = 0 R1 = 0 W1 = 1
C2 = 0 L2 = 0 R3 = 0 W2 = 1
C3 = 0 L3 = 0 R3 = 0 W 3= 1
C4 = 1 L4 = 1 R4 = 0 W4 = 0

Finalmente, se a saída não apresentar falha (S = 1), a célula de teste volta a supervisionar a célula 1, reiniciando o processo, e a combinação de abertura e fechamento de chaves será:

C1 = 1 L1 = 0 R1 = 1 W1 = 1
C2 = 0 L2 = 0 R3 = 0 W2 = 1
C3 = 0 L3 = 0 R3 = 0 W 3= 1
C4 = 0 L4 = 0 R4 = 0 W4 = 1

A implementação em hardware deste algoritmo foi realizada utilizando dispositivos programáveis conforme descrito na próxima seção.

4.2. IMPLEMENTAÇÃO UTILIZANDO DISPOSITIVOS PROGRAMÁVEIS

4.2.1. PROTOTIPAÇÃO DO CONTROLE

Como os resultados alcançados com o protótipo utilizando componentes discretos foram satisfatórios, passou-se a desenvolver um circuito para que a detecção de falhas e a substituição de células fossem executadas automaticamente. O algoritmo apresentado no item 4.1 foi evidentemente o foco. O circuito que faz a detecção de falhas e a substituição da célula

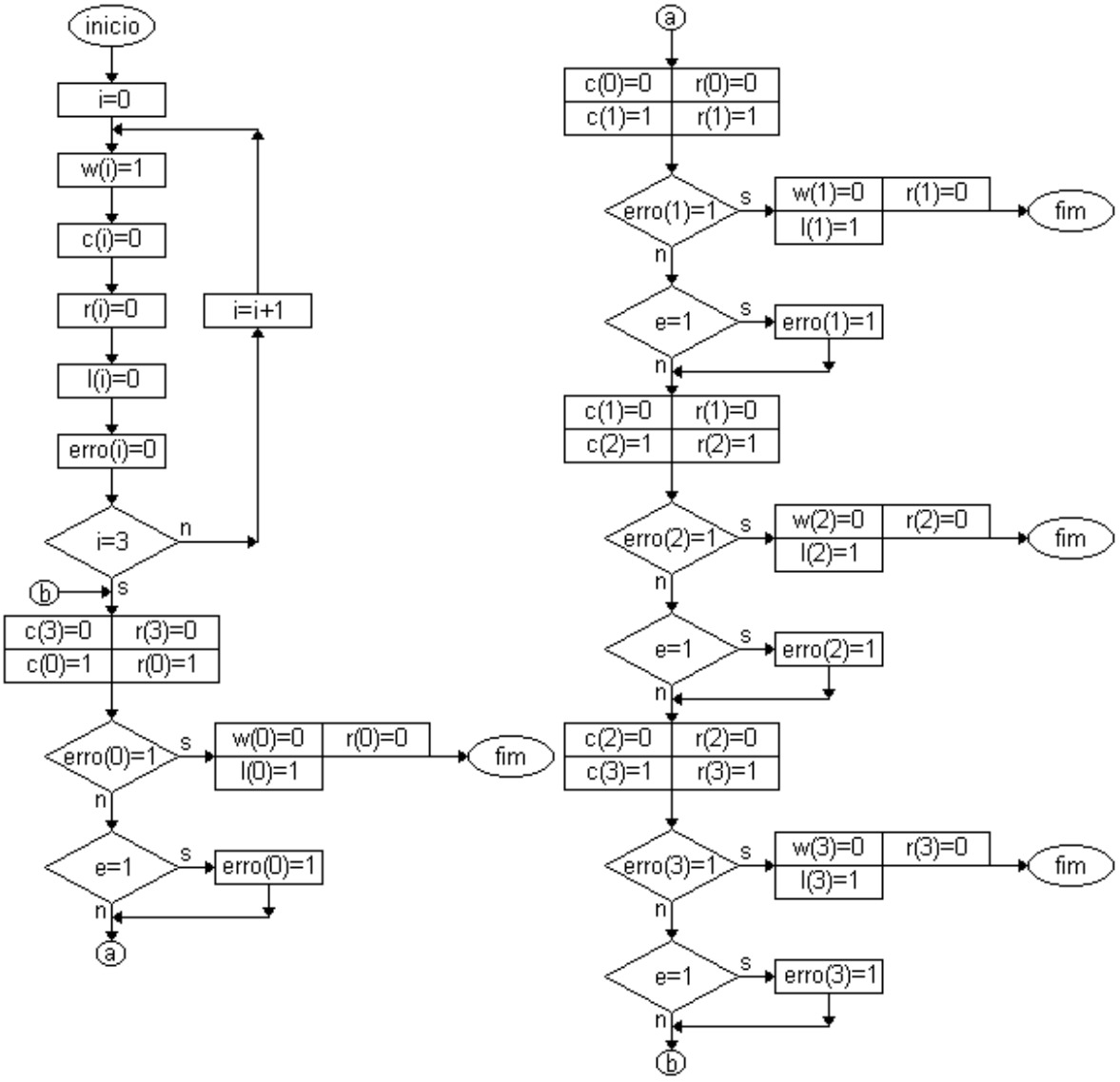

Figura 4.4a. Fluxograma da primeira versão descrita em VHDL.

defeituosa foi denominado *circuito de controle*. Um circuito dessa natureza não é de fácil implementação em uma placa com componentes discretos devido, principalmente, ao grande número de conexões que devem ser realizadas. Por isso buscou-se uma solução em dispositivos lógicos programáveis. Com este propósito foram desenvolvidas duas versões para a lógica de controle de teste do roteador. Baseadas em dois fluxogramas diferentes, as duas versões foram

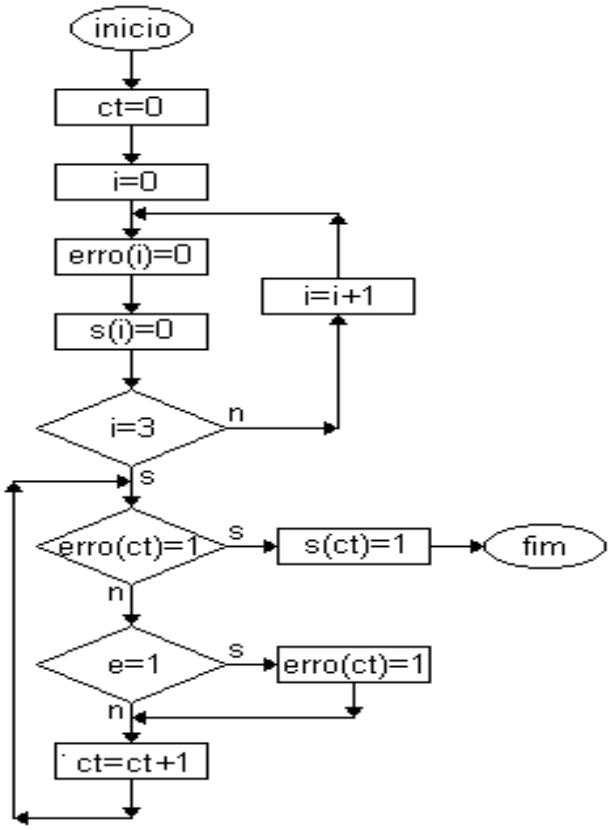

Figura 4.4b. Fluxograma da segunda versão descrita em VHDL.

descritas em código VHDL e simuladas de forma a auxiliar na escolha do sistema digital com melhor desempenho. Os fluxogramas das duas versões desenvolvidas estão apresentados nas figuras 4.4a e 4.4b. A versão implementada em FPGA segue o fluxograma **a**, pois esta, apesar de mais lenta, não chega a comprometer a operação do sistema e ocupa um número menor de blocos lógicos.

O sistema digital proposto é composto de quatro entradas de dados (e1, e2, e3 e e4), quatro saídas (s1, s2, s3 e s4), quatro entradas de controle (c1, c2, c3 e c4), responsáveis pelo

roteamento dos sinais, além de entradas e saídas necessárias ao controle de teste, conforme a figura 4.5.

O processo de teste deve ser realizado *on-line*, ou seja, a célula de teste, comandada pelo controle, varre todo o circuito em busca de falhas durante o funcionamento da matriz roteadora. A célula de teste com entradas ‘et1’, ‘et2’ e ‘ct’, utiliza as entradas de cada uma das células de base do roteador no momento de seu teste, gerando saídas que correspondem às saídas de uma célula operando corretamente (‘st1’ e ‘st2’). A comparação entre as saídas geradas pela célula de base (‘sca’ e ‘scb’) e as saídas geradas pela célula de teste (‘st1’ e ‘st2’) resulta em um sinal de erro (‘e’), indicando a presença ou ausência de falha na célula testada.

O controle gerencia a varredura de todas as células com o objetivo de encontrar eventuais falhas. Uma falha é detectada através do sinal ‘e’ (erro), resultante da comparação entre os sinais de saída da célula de base e da célula de teste. No momento em que a falha é detectada, o controle seta o registrador de erro correspondente à célula falha, permitindo desta forma uma varredura adicional do sistema, antes da substituição da célula de base defeituosa pela célula de teste. Este procedimento é importante de forma a evitar que a substituição seja feita em circunstâncias que a célula de teste estaria defeituosa.

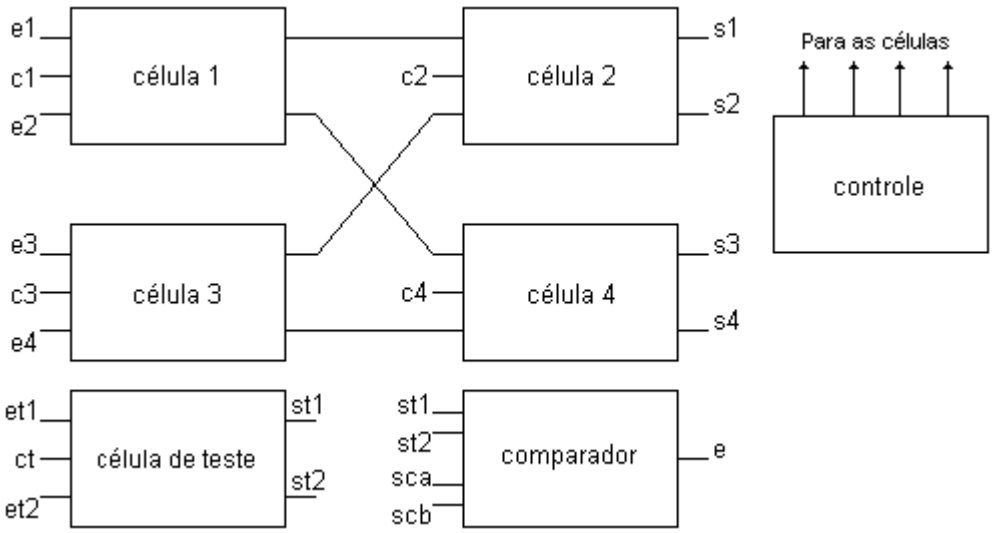

Figura 4.5. Arquitetura do sistema digital.

4.2.2. PROTOTIPAÇÃO DE TODO O SISTEMA

A matriz de roteamento, em sua totalidade, foi também descrita em VHDL e simulada utilizando a ferramenta de CAD *MaxPlus II* da Altera.

O protótipo implementado em placa de circuito impresso utiliza chaves analógicas, procurando emular o funcionamento de um circuito NMOS. Em VHDL, o funcionamento de chaves analógicas pode ser aproximado em âmbito digital por portas tipo *three-state*. Entretanto, os FPGAs disponíveis possuem estas portas dispostas apenas na periferia dos *chips*, inviabilizando o seu uso na solução do problema proposto. Portanto, a topologia sofreu alguns ajustes de forma a ser implementada com o uso de multiplexadores sem perda quanto ao funcionamento lógico.

A regularidade de algumas estruturas do roteador permitiu a descrição do sistema com o uso de duas sistemáticas. Parte do projeto foi descrito sob forma estrutural e outra sob forma comportamental. A descrição estrutural consiste em descrever as interconexões entre componentes do sistema, enquanto que a descrição comportamental consiste em descrever de forma mais abstrata o comportamento lógico das partes.

A célula padrão de roteamento é implementada com o uso de 4 multiplexadores 2:1, sendo controlada basicamente pelos sinais ‘c1’ (externo via usuário) e ‘c2’ (interno via controle), conforme figura 4.6. O controle ‘c1’ modifica a associação das saídas ‘s1’ e ‘s2’ com as entradas ‘e1’ e ‘e2’, de forma a rotear convenientemente as informações provenientes destas entradas. As saídas ‘sc1’ e ‘sc2’ são observadas pelo controle no momento em que a célula está sendo testada. Estas saídas são comparadas com ‘st1’ e ‘st2’, de forma a indicar a veracidade das informações roteadas pela célula sob teste. Caso apresente falha, o controle ‘c2’ é selecionado, roteando as saídas da célula de teste (‘st1’ e ‘st2’) para as saídas da célula com problemas, corrigindo o erro.

A célula de teste utiliza as entradas de cada uma das células de base do roteador no momento de seu teste, gerando saídas que correspondem às saídas de uma célula operando corretamente (‘st1’ e ‘st2’). A seleção destas entradas se processa através do uso de dois multiplexadores 4:1 controlados pelo sinal ‘ct’ provindo do controle, conforme pode ser visto

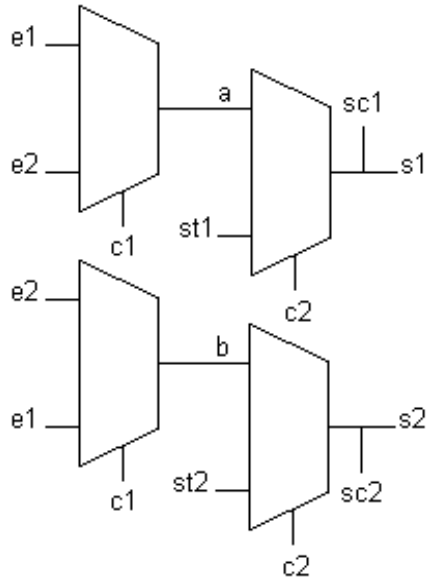

Figura 4.6. Célula de base.

na figura 4.7. Da mesma forma, sob o comando de ‘ct’, são selecionadas as saídas da célula sob teste, gerando os sinais ‘sca’ e ‘scb’ que serão comparados juntamente com ‘st1’ e ‘st2’ no comparador. Em caso de falha, o controle mantém o estado do sinal ‘ct’ e direciona os sinais ‘st1’ e ‘st2’ para a célula defeituosa, substituindo-a.

Os sinais de cada célula de base observados pela célula de teste encontram-se esquematizados abaixo, de forma a permitir uma melhor compreensão da topologia empregada:

1. célula 1 (‘ct’=0): ‘e1’, ‘e2’, ‘c1’, ‘sc1’ e ‘sc2’;
2. célula 2 (‘ct’=1): ‘e3’, ‘e4’, ‘c2’, ‘sc3’ e ‘sc4’;
3. célula 3 (‘ct’=2): ‘e5’, ‘e6’, ‘c3’, ‘sc5’ e ‘sc6’; e
4. célula 4 (‘ct’=3): ‘e7’, ‘e8’, ‘c4’, ‘sc7’ e ‘sc8’.

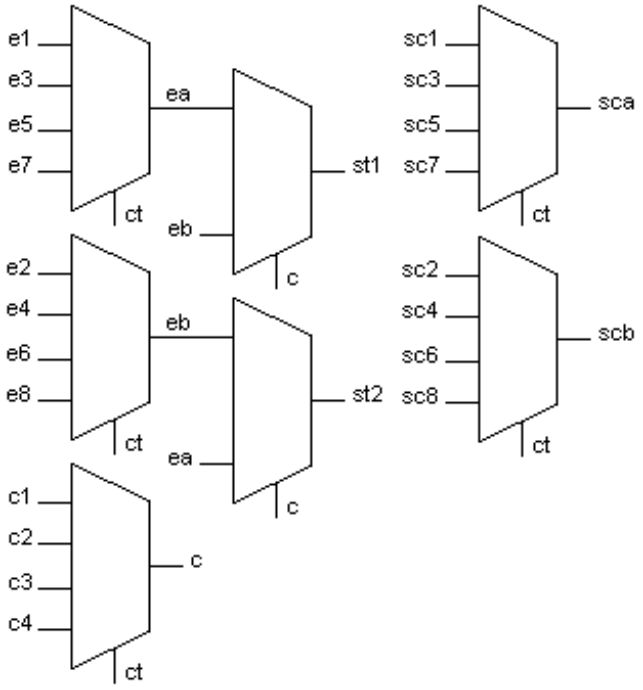

Figura 4.7. Célula de teste.

O controle do sistema é responsável pelo gerenciamento da operação de teste da matriz de chaveamento. A arquitetura necessária para a realização deste processo utiliza os seguintes componentes:

1. Contador ‘ct’: contador que indica ao controle a célula que está sendo testada.
2. Sinal ‘e’: sinal que fornece o estado de erro da célula sob teste, resultado da comparação entre as saídas da célula de base e da célula de teste.
3. Registrador ‘erro’: registrador que armazena a ocorrência de falhas nas células testadas.
4. Registrador ‘s’: registrador que indica que célula deve ser substituída.

A máquina de estados, responsável pelo controle da operação de teste da matriz de chaveamento, é composta por 4 estados distintos, estando estruturada da seguinte maneira:

Estado 0 - o sistema verifica a existência de falhas anteriormente detectadas na célula sob teste, decidindo sobre sua substituição;

Estado 1 - o sistema realiza o teste da célula;

Estado 2 - o sistema procede a substituição da célula;

Fim - o sistema encerra o processo de teste.

Na simulação preliminar, apresentada na figura 4.9, verifica-se que inicialmente um sinal de *reset* é aplicado ao sistema de forma a inicializar os contadores e registradores internos, além de levar a máquina de controle ao estado inicial (‘ea’=0). Durante o processamento do teste, o contador ‘ct’ é incrementado de forma a varrer todo o circuito em busca de falhas. O sistema, ao detectar uma condição de erro na transição entre os estados S0 (‘ea’=0) e S1 (‘ea’=1) através do nível alto do sinal ‘e’, atualiza o registrador ‘erro’, mas não procede à imediata substituição da célula falha. Uma varredura adicional é processada, com o
objetivo de informar ao sistema o estado de funcionamento das demais células antes de substituir a célula falha. Ao término da varredura adicional, o sistema utiliza o registrador ‘s’ (‘ea’=2) para substituir a célula defeituosa. É importante perceber que, no momento da substituição da célula, o sistema mantém o estado do contador ‘ct’, de forma a permitir que as

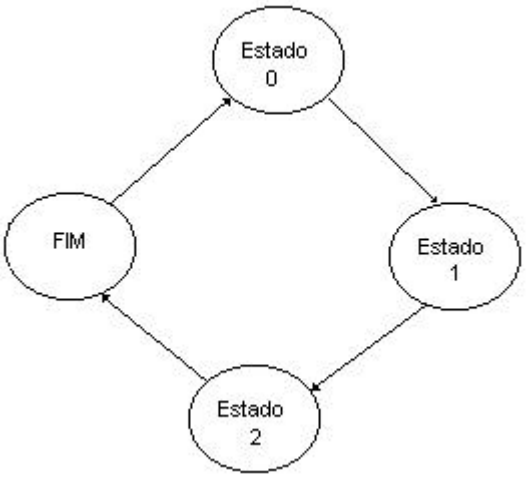

Figura 4.8. Máquina de estados do controle da operação de teste.

entradas da célula falha sejam conduzidas para a célula de teste, sendo roteadas novamente para as saídas da célula falha. Finalmente, o sistema alcança o estado final de processamento, ou seja, o estado FIM (‘ea’=3).

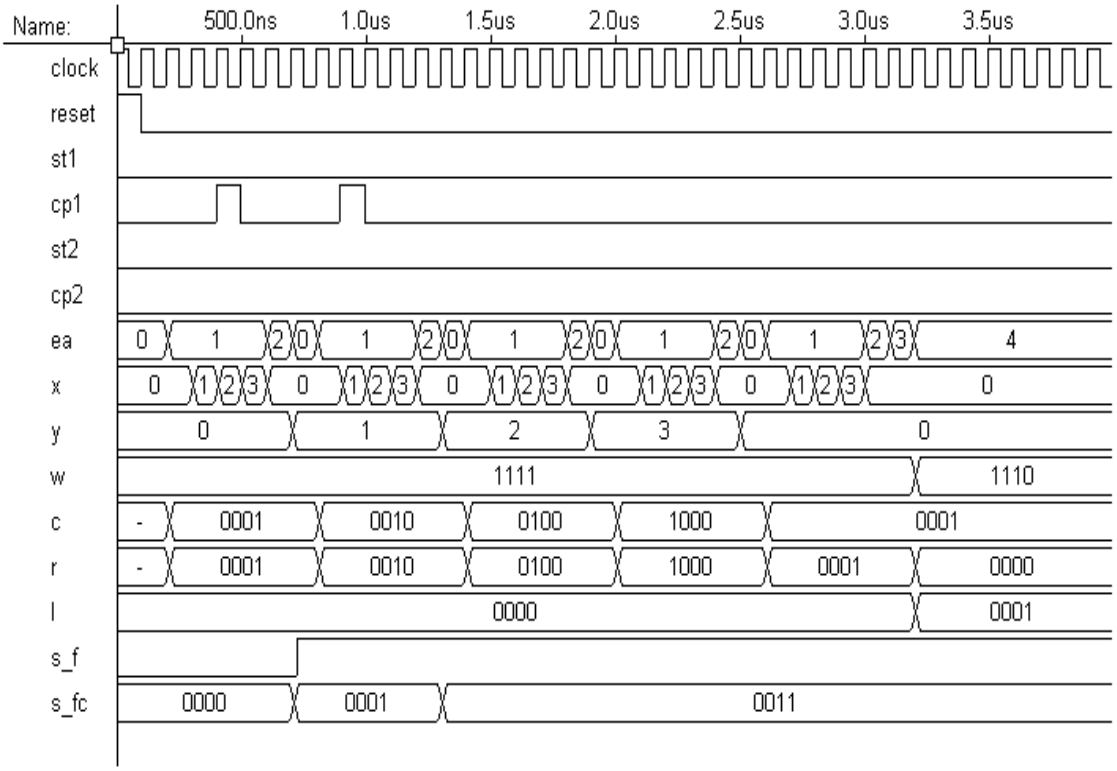

Figura 4.9. Simulação do controle.

A matriz de chaveamento fica então constituída por 4 células de base, uma célula de teste, um comparador e um circuito de controle, conforme anteriormente descrito. O circuito roteador, descrito sob forma estrutural, consiste na integração de todos estes componentes de forma a gerar o circuito completo. Através de simulações do sistema total, pode-se verificar a correção da implementação da estratégia de teste.

Observando a simetria da figura 4.10 verifica-se o perfeito funcionamento da matriz de chaveamento implementada em VHDL, uma vez que os sinais ‘falha’ e ‘cs’, responsáveis, respectivamente, pelas sinalizações de falha e célula substituída, não foram alterados pelo sistema, indicando um funcionamento livre de falhas. Da mesma forma, através da inspeção das saídas obtidas, nota-se que os dados constantes em ‘s1’, ‘s2’, ‘s3’ e ‘s4’ são provenientes de ‘e1’, ‘e3’, ‘e2’ e ‘e4’ respectivamente, uma vez que todos os sinais de controle (‘c1’, ‘c2’, ‘c3’ e ‘c4’) estão em nível baixo, caracterizando um correto roteamento dos sinais de entrada. É importante notar que todas as saídas apresentam defasagem de uma amostra com relação à entrada. Esta é uma

condição imposta pela implementação do *flip-flop* de sincronização, no caminho de todas as entradas externas.

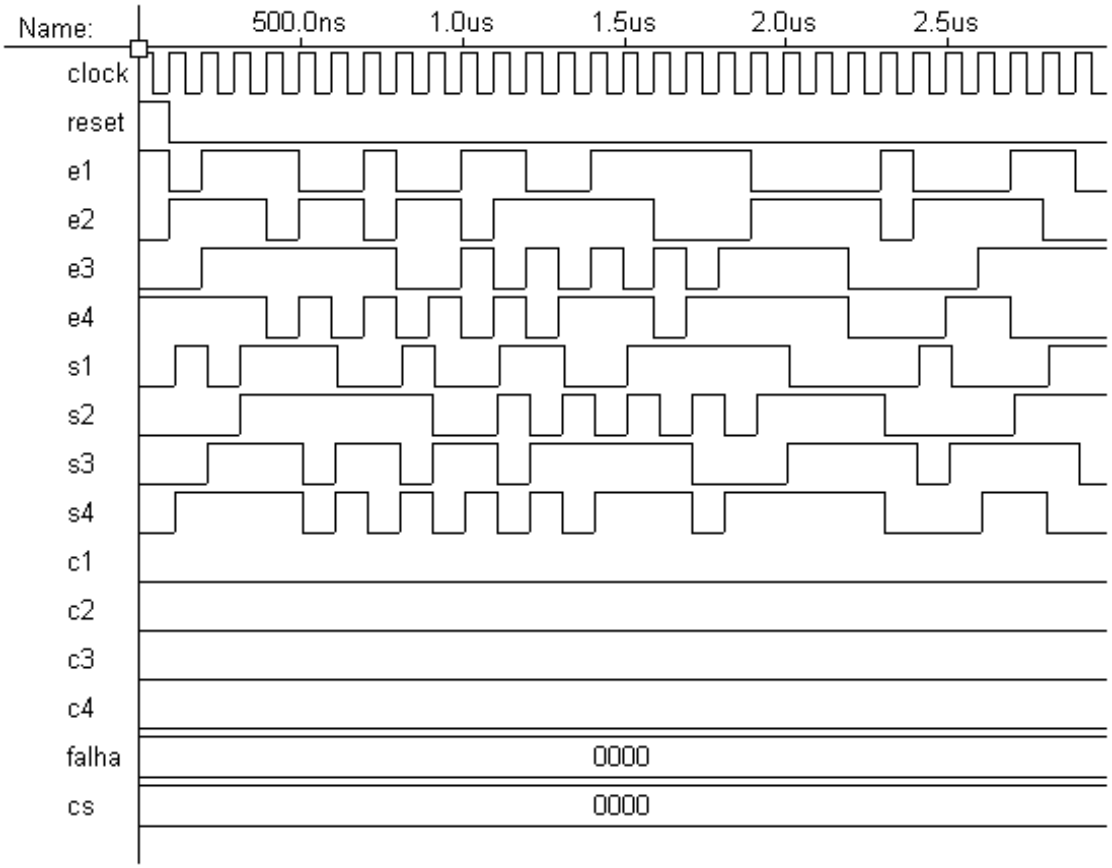

Figura 4.10. Simulação do circuito roteador livre de falhas

4.2.3. INJEÇÃO E SIMULAÇÃO DE FALHAS

Com o objetivo de avaliar a eficiência da estratégia de teste empregada, foram inseridas algumas portas lógicas na descrição do roteador, de forma a emular defeitos no circuito. As falhas foram adicionadas às células de base, em algumas entradas e em algumas conexões internas ao circuito roteador (entre as células de base), como pode ser observado nas figuras 4.11 e 4.12. As falhas são do tipo curto-circuito (implementadas com multiplexador) e *stuck-at* (implementadas com portas lógicas tipo *and* e *or*), sendo controladas por entradas específicas (cf1 a cf10).

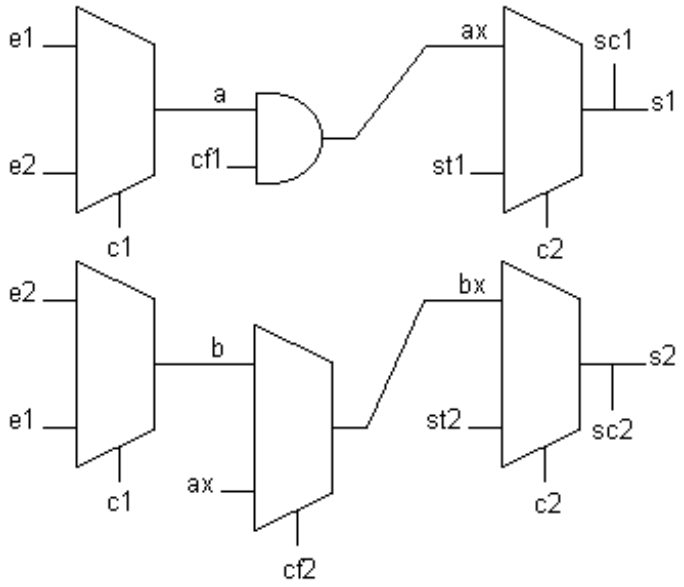

Figura 4.11. Célula padrão com falha do tipo *stuck-at*.

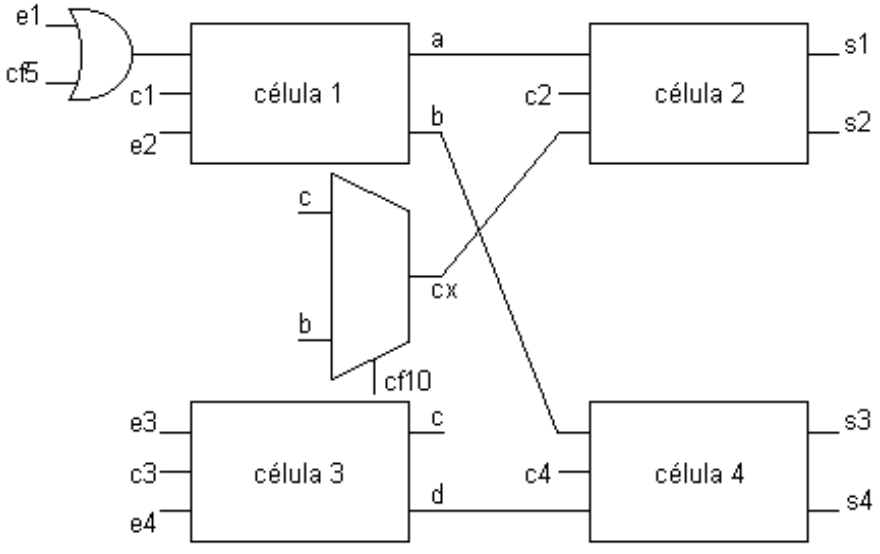

Figura 4.12. Falha do tipo *curto-circuito* no circuito roteador.

Através de uma porta lógica *and* foi implementada uma falha *stuck-at* 0 no nó 'a' de cada uma das células. Quando o sinal cf1 assume valor lógico 0, então a falha é injetada no nó (figura 4.11).

Através de portas lógicas *or* foram implementadas falhas *stuck-at 1* nos nós 'e1' das diversas células. O sinal cf5 ficou responsável pela ativação da falha, conforme abaixo:

Cf5:	e1
0	Operação normal
1	*stuck-at* 1

Utilizando um multiplexador foi implementada uma falha de curto-circuito entre os nós 'a' e 'b' de cada uma das células. Quando o sinal cf2 assume o valor lógico 1, então o curto-circuito é ativado (figura 4.12).

Utilizando um multiplexador foi implementado um curto-circuito entre os nós 'b' e 'c' do circuito roteador. O sinal cf10 ficou encarregado da ativação da falha, conforme abaixo:

Cf10:	Entrada
0	Operação normal
1	Falha

As falhas internas às células são detectadas com sucesso pelo método de teste adotado, conforme pode ser visto na simulação apresentada na figura 4.13. A falha do tipo *stuck-at* 0 presente na célula 1 (cf1 em 0) apresenta-se de forma clara na saída 's1', desaparecendo seu efeito no momento da substituição da célula. A detecção de alguma falha no sistema se dá no momento em que o registrador 'falha' é utilizado para armazenar os dados referentes às células falhas. Já a substituição de alguma célula se dá no momento em que o registrador 'cs' é utilizado.

Como era de se esperar a detecção das falhas na matriz de chaveamento é altamente dependente dos sinais em suas entradas no momento do teste. Estes sinais devem ser capazes de

evidenciar as falhas. Por isto, a falha do tipo *stuck-at* 0 na célula 1 (cf1 em 0), mostrada na simulação da figura 4.14, não é detectada, apesar de ser óbvia ao se comparar os sinais ‘e1’ e ‘s1’. Este problema pode ser resolvido aumentando-se o tempo de teste de cada célula, de forma a dar o tempo para a aplicação fornecer um conjunto mais amplo de valores de entradas funcionais.

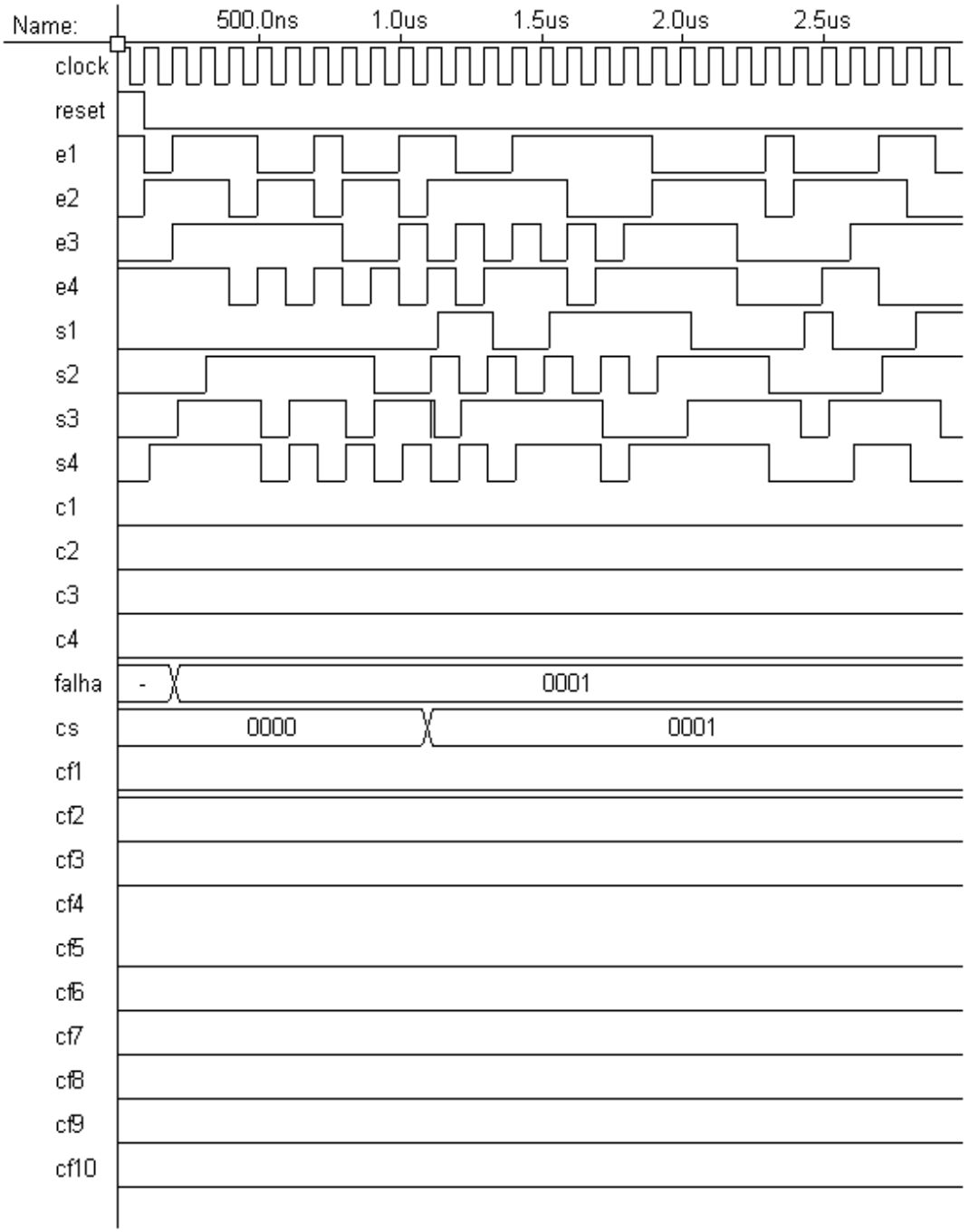

Figura 4.13. Simulação de uma falha *stuck-at* 0 na célula 1 (cf1 = 0).

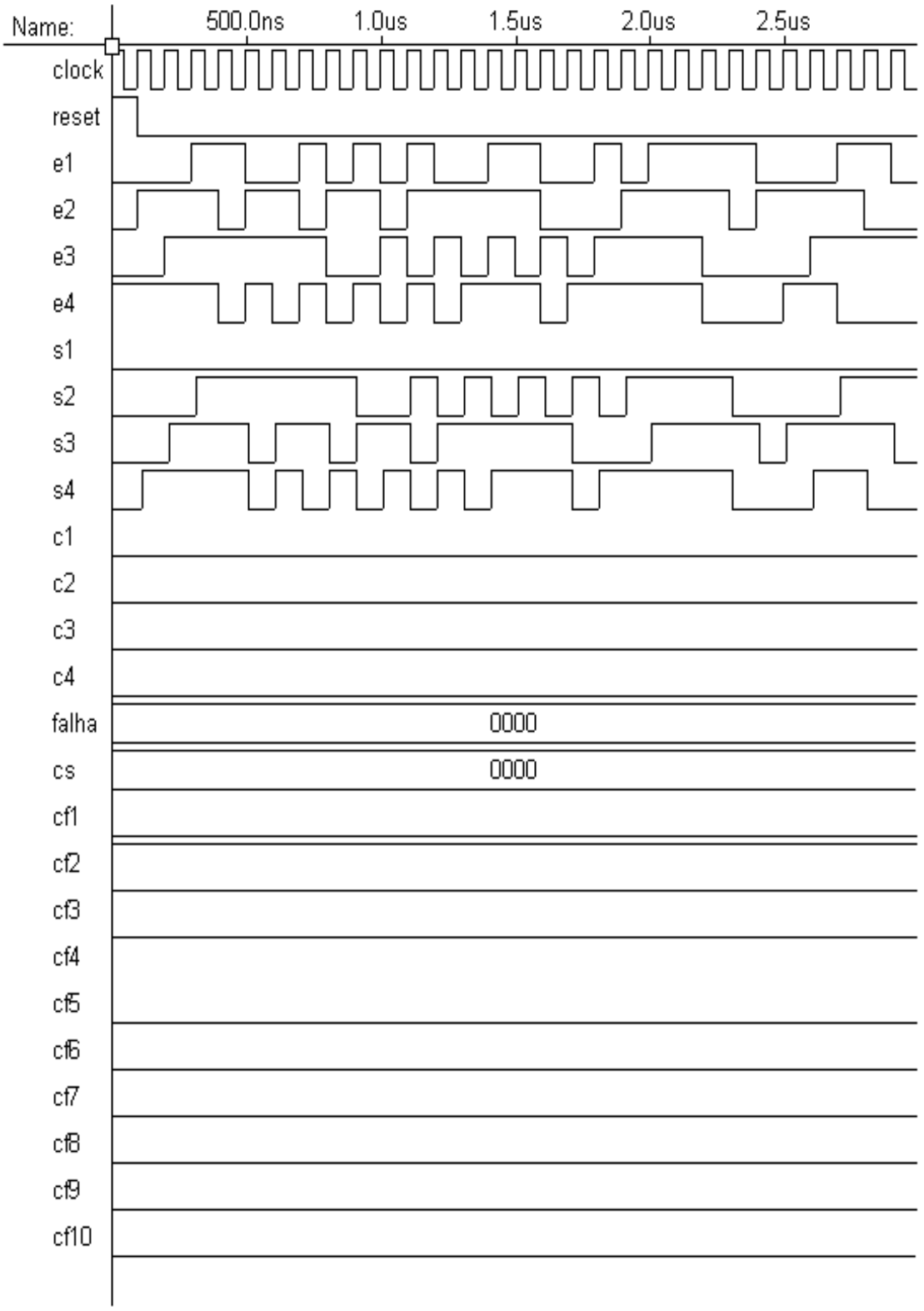

Figura 4.14. Simulação de uma falha *stuck-at* 0 na célula 1 (cf1=0) com falha não detectada.

As falhas nas entradas primárias do circuito roteador não são detectadas pelo método de teste adotado. A figura 4.15 apresenta a simulação do sistema com falha do tipo *stuck-at* 1 em ‘e1’ (cf5 em 1), mostrando a incapacidade do sistema de detectar tal falha, apesar da presença da falha ser evidente ao se comparar os sinais ‘e1’ e ‘s1’. Isto ocorre devido ao fato de que tanto a célula 1 quanto a célula de teste utilizaram as mesmas entradas (falhas ou não) para a geração de suas respectivas saídas. A solução deste problema exige a utilização de técnicas adicionais de teste.

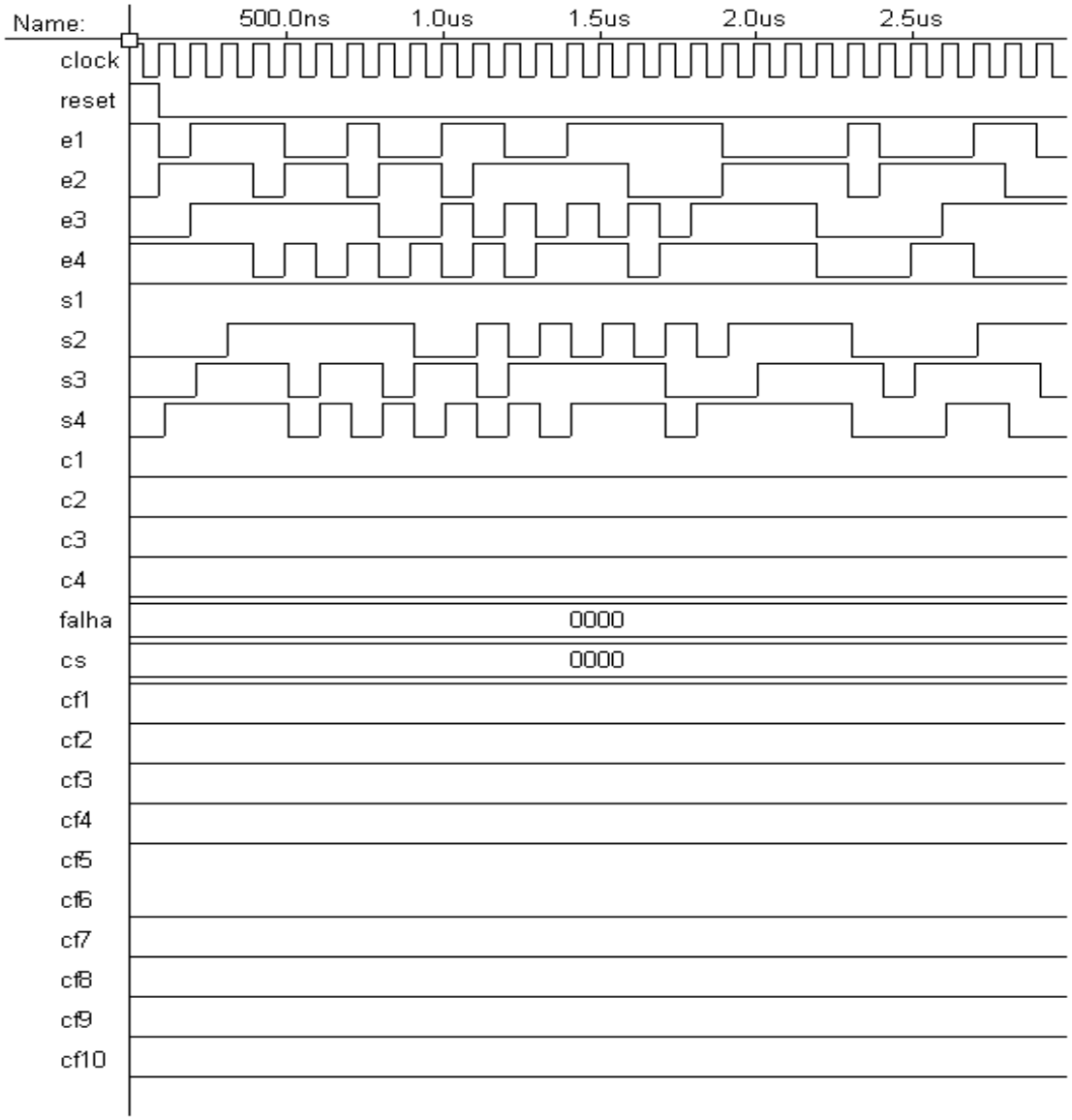

Figura 4.15. Simulação de uma falha *stuck-at* 1, na entrada e1 do circuito roteador (cf5=1), não detectada pelo sistema.

As falhas nas interconexões do circuito roteador (entre células) são detectadas pela sistemática de teste adotada, apesar de serem diagnosticadas de forma errônea. A figura 4.16 apresenta a simulação do sistema com uma falha de curto-circuito entre os nós ‘b’ e ‘c’ (cf10 em 1, figura 4.12), explicitada de forma clara nas saídas ‘s2’ e ‘s3’. O sistema detecta a falha mas apresenta um diagnóstico errôneo, explicitado através da substituição de uma célula funcionando adequadamente. Este problema se explica pelo fato de o sistema comparar as saídas da célula sob teste com as saídas da célula de teste, ou seja, o sistema procura sempre atribuir a causa de uma falha a uma célula. Entretanto, o problema mais grave consiste na incapacidade do sistema de decidir se a falha é contornável, se o pleno funcionamento do sistema pode ser recuperado. Este problema poderia ser resolvido com a utilização de duas células de teste. Após a substituição de uma célula do circuito roteador, a outra célula de teste se encarregaria de verificar a consistência do diagnóstico e, claro, do funcionamento do sistema.

Um problema grave que pode ocorrer é que uma falha em uma célula encubra falhas em outras células. Um exemplo disso seria o caso em que as células 1 e 2 possuíssem falhas do tipo *stuck-at* 0 (cf1 e cf2 em 0), evidenciadas na saída ‘s1’, conforme simulação apresentada na figura 4.17. A falha na célula 1 induziria a não detecção da falha na célula 2, mesmo com a varredura adicional, já que a saída falha da célula 1 é encarada pela célula 2 como entrada válida. No momento em que a célula 1 for substituída, a falha na célula 2 torna-se evidente mas o controle considera esta célula válida, incorrendo em um erro grave: a existência de falha sem seu reconhecimento. Novamente, este problema pode ser evitado com a utilização de duas células de teste.

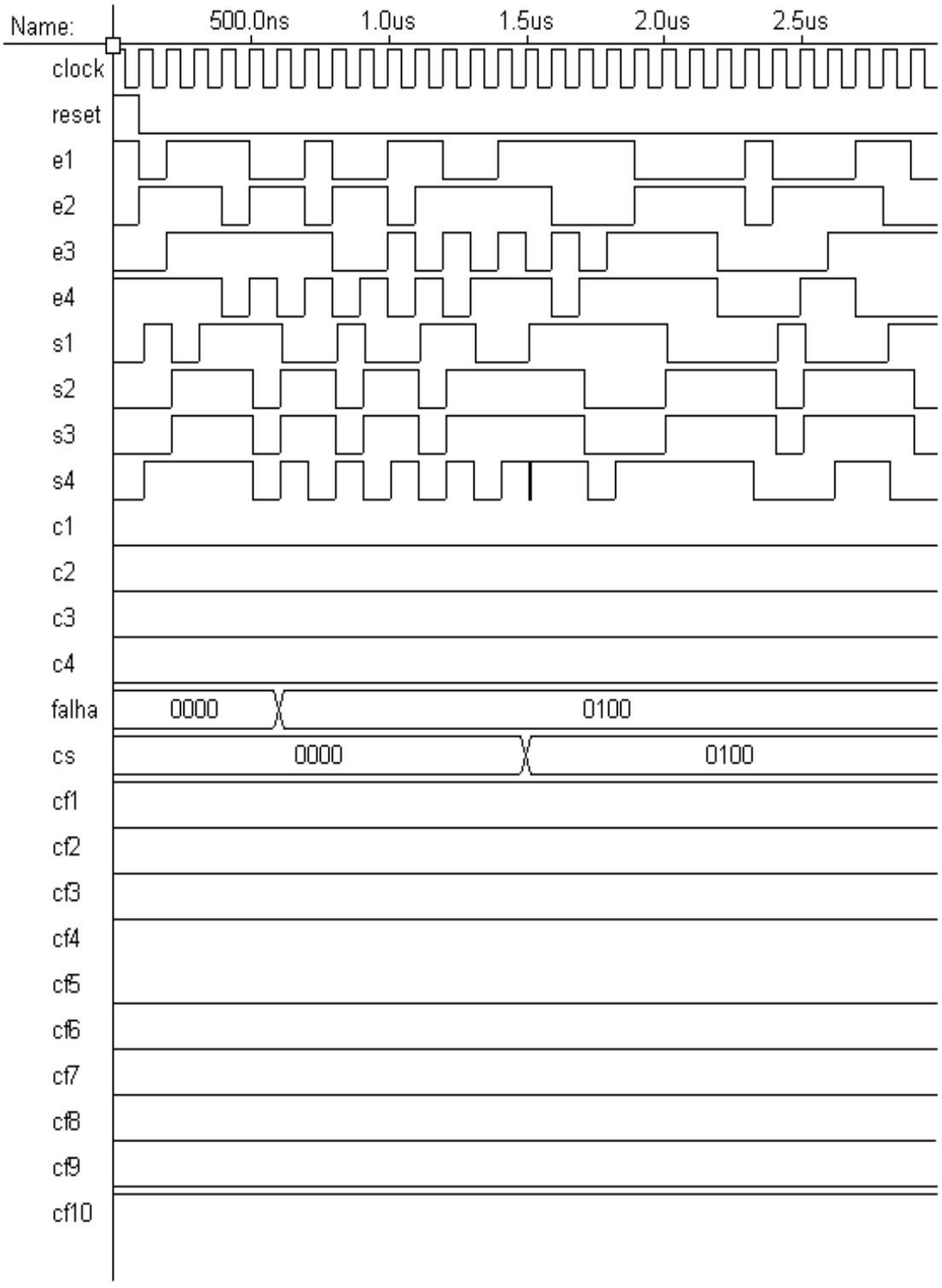

Figura 4.16. Simulação do sistema com uma falha de curto-circuito entre os nós ‘b’ e ‘c’ (cf10 em 1).

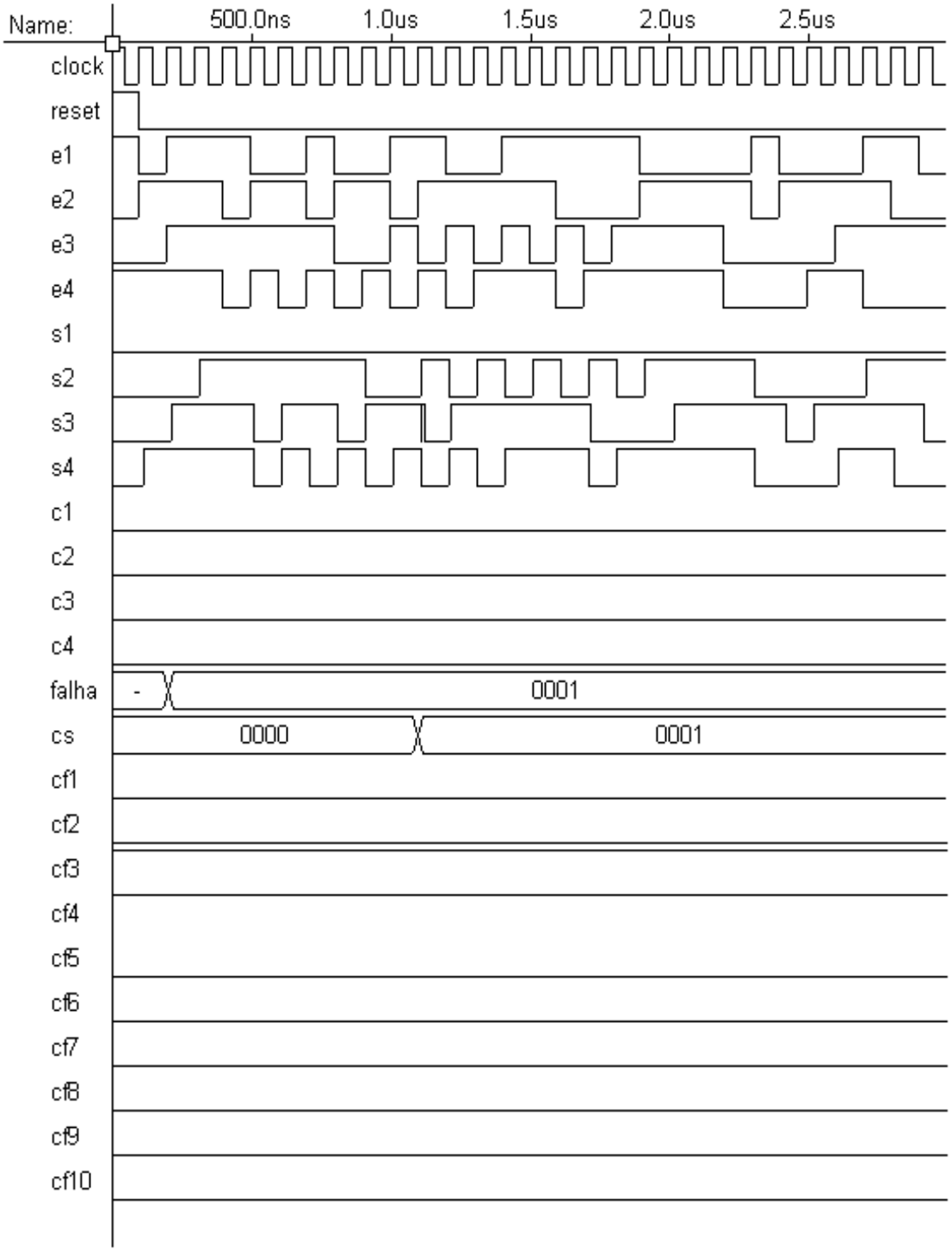

Figura 4.17. Simulação de falhas do tipo *stuck-at* 0, nas células 1 e 2 (cf1=0 e cf2=0), não detectadas pelo sistema.

Os resultados obtidos na prática com a sistemática de teste adotada foram animadores, apesar de apresentar os problemas já discutidos anteriormente. Importante notar que existe grande diferença quanto ao limite máximo de funcionamento do *clock* obtido via simulação (em torno de 50Mhz) e via bancada de testes (em torno de 2MHz). O sistema implementado apresenta sensibilidade quanto ao *clock*, que em freqüências elevadas (em torno de 2MHz) torna-se bastante instável, acusando falhas inexistentes. Contudo, espera-se que a futura integração do sistema de teste com a matriz de chaveamento em silício torne o sistema mais estável e operacional com uma freqüência de *clock* superior.

A área ocupada pelo sistema de controle é 4 vezes superior à área ocupada pela matriz de chaveamento. O aumento da área causado pelo acréscimo de uma célula ao circuito seria compensado pela melhoria na cobertura de falhas, uma vez que o circuito de controle, que é responsável pela ocupação da maior área permaneceria, praticamente o mesmo.

5. CONCLUSÃO

Neste trabalho foi realizado um estudo sobre o comportamento de uma matriz de roteamento de canais telefônicos na presença de falhas. Inicialmente foram feitas considerações a respeito do teste de sistemas de hardware, no que diz respeito a sua utilização para aumentar a confiabilidade, e a necessidade de uma abordagem desde o início do projeto com o objetivo de facilitar a sua aplicabilidade.

Alguns aspectos do teste de circuitos e sistemas tais como a metodologia, classificação para os tipos mais utilizados, defeitos mais comuns que ocorrem em circuitos e as falhas por eles causadas, assim como os modelos para essas falhas, simulação de falhas e geração de teste, foram abordados no capítulo 2. O projeto visando o teste, encarado como uma solução para melhorar a testabilidade dos circuitos e sistemas, também foi tratado no mesmo capítulo.

O capítulo 3 tratou do teste em funcionamento do roteador de canais telefônicos, levando em consideração sua arquitetura, a metodologia de teste utilizada para a detecção de falhas, bem como um estudo de como torná-lo tolerante a falhas.

No capítulo 4 foi abordado o projeto de um circuito roteador tolerante a falhas, primeiramente com componentes discretos e num segundo momento utilizando dispositivos programáveis. Neste capítulo, são apresentados os resultados obtidos através de simulações.

As conclusões a que se pode chegar no final deste trabalho são as seguintes:

A metodologia de teste empregada para testar o circuito de roteamento de canais telefônicos mostrou-se muito promissora, apesar das suas restrições em termos de desempenho do sistema final. Existe um limite máximo de funcionamento do *clock* obtido via simulação que fica em torno de 50Mhz e na bancada, que fica em torno de 2Mhz. Em freqüências elevadas o sistema fica instável acusando falhas inexistentes.

Espera-se que este problema possa ser resolvido com a integração do sistema completo em silício.

As falhas internas às células podem ser detectadas com sucesso. No entanto, esta detecção é altamente dependente dos sinais em suas entradas no momento do teste. Por esta razão algumas falhas podem não vir a ser imediatamente detectadas quando o sistema estiver em funcionamento.

Este problema pode ser contornado aumentando o tempo de teste de cada célula de forma a fornecer um conjunto mais amplo de entradas funcionais.

As falhas nas entradas primárias do circuito não são detectadas. Isto se deve ao fato de que tanto a célula de base quanto a célula de teste utilizam as mesmas entradas para a geração de suas saídas. Já as falhas nas interconexões das células do circuito roteador são detectadas embora possam ser diagnosticadas de forma errônea.

Este problema poderia ser resolvido com a utilização de duas células de teste, uma para detectar a célula falha e outra para verificar a consistência do diagnóstico.

Finalizando, pode-se dizer que a validade deste trabalho está relacionada à idéia de sua aplicabilidade com o objetivo de aumentar a confiabilidade de circuitos e sistemas. Tanto a idéia é viável que trabalhos recentes (Gericota, 2002) a exploram para o teste concorrente de uma FPGA da família *Virtex da Xilinx*.

Os problemas mencionados acima, relativos à implementação, à redundância de células e ao tempo de teste, permanecem para trabalhos futuros.

REFERÊNCIAS

ABRAMOVICI, M.; BREUER, M. A.; FRIEDMAN, A. D**. Digital Systems Testing and Testable Design**. New York: IEEE Press, 1990.

ANDERSON, D. A. **Design of Self-Checking Digital Networks Using Coding Techniques.** University of Illinois, Urbana CSL, 1971. Technical Report 527.

ARMSTROMG, D. B. A Deductive Method of Simulating Faults in Logic Circuits. **IEEE Transactions on Computers,** v. 21, p. 464-471, 1972.

BRACKEL, G.V.; GLÄSSER, U.; KERKHOFF, H.G.; VIERHAUS, H.T. Gate Delay Fault Test Generation for Non-Scan Circuits. In: EUROPEAN DESIGN TEST CONFERENCE, 1995, **Proceedings...** p. 308-312.

BENNETTS, R. G.; MAUNDER, C.M.; ROBINSON, G. D. CAMELOT: A Computer-Aided Measure for Logic Testability. **IEEE Proceedings**, 128, n. 5, 1981, p. 177-189.

BERG, W. C.; HESS, R. D. COMET: A Testability Analysis and Design Modification Package. In: IEEE TEST CONFERENCE, 1982, **Proceedings ...** p. 364-378.

BREUER, M. A.; FRIEDMAN, A.D. *Diagnosis and Reliable Design of Digital Systems.* Computer Science Press, 1976.

BRGLEZ, F. On Testability Analysis of Combinational Networks, In: INTERNATIONAL SYMPOSIUM CIRCUITS AND SYSTEMS, 1984. **Proceedings...** p. 221-225.

CARTER, W. C.; SCHNEIDER, P. R. Design of Dynamically Checked Computers, In: IFIP CONGRESS,1968. **Proceedings**.

COURTOIS, B. Failure Mechanisms, Fault Hypotheses and Analytical Testing of LSI-NMOS (HMOS) Circuits, In: VLSI CONFERENCE, 1981.**Proceedings...** p. 341-350.

DEJKA, W.J. Mensure of Testability in Devices and System Design, In: *20th* MIDWEST SYMPOSIUM CIRCUITS AND SYSTEMS, 1977. **Proceedings...** p. 39-52.

DUSSAUT, J.A. A Testability Measure, In: IEEE SEMICONDUCTOR TEST CONFERENCE, 1978. **Proceedings...**, p. 113-116.

FUJIWARA, H.; SHIMONO, T. On the Acceleration of Test Generation Algorithms. **IEEE Transactions on Computers,** v. 32, p. 1137-1144, 1983.

FUJIWARA, H. **Logic Testing Design for Testability**. Cambridge, Ma, The MIT Press, 1985.

GALAY, J.; CROUZET, Y.; VERGINIAULT, M. Physical Versus Logical Fault Models MOS-LSI Circuits: Impact of their Testability. **IEEE Transaction on Computers,** v. 29, n. 6, p. 527-531, 1980.

GERICOTA, M. G.; ALVES, G. R. Dynamic Replication: The Core of a Truly Non-Intrusive SRAM-based FPGA Structural Concurrent Test Methodology, In: 3RTH IEEE LATIN AMERICAN TEST WORKSHOP, 2002, Montevideu, Urugay. **Proceedings...** p. 70-75.

GOEL, P. An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits. **IEEE Transactions on Computers**, v. 30, p. 215-222,1981.

HAYES, J. P. On Modifying Logic Network to Improve their Diagnosability. **IEEE Transactions on Computer,** C-23, n.1, p. 56-62, 1974.

HAYS, J. P.; FRIEDMAN, A. D. Test Point Placement to Simplify Fault Detection. **IEEE Transactions on Computer,** C-23, n. 7, p. 727-735, 1974.

KEINER, W.; WEST, R. Testability Measures. **IEEE Autotestcon**, **Proceedings...** p. 49-55, 1977.

KOLARÍK, V**. Techniques Avancées de Test de Circuits Analogiques et Mixtes Analogiques/numériques.** 1994.. Tese de Doutorado, (Microéletronique). Institut Polytechnique de Grenoble.

KOVIJANIC, P. G. Testability Analysis, In: IEEE TEST CONFERENCE, 1979. **Proceedings...** p. 310-316.

LUBASZEWSKI, M.; COUTOIS, E. B. Reliable Fail-Safe Systems. IEEE ASIAN SYMPOSIUM, PEQUIM, 1993,China. **Proceedings**.

MALY, W.; NIGH, P. Bilt-In Current Testing - Feasibility Study, In: INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, 1988.**Proceedings...** p. 340-343.

MARCHOK, T.E.; EL-MALEH, A.; MALY, W; RAJISKI, J. Complexity of Sequential ATPG, In: EUROPEAN DESIGN AND TEST CONFERENCE, 1995. **Proceedings...** p. 252-261.

MURRAY, B.T.; HAYES, J.P. Testing ICs: Getting the *Core* of the Problem. **IEEE Computer**, p. 32-38, nov. 1996.

NICOLAIDIS, M.; COURTOIS, B. Strongly Code Disjoint Checkers**. IEEE Transactions on Computers**, v. 37, n. 6, p. 751-756, 1988.

NIGH, P.; MALY, W. Test Generation for Current Testing, In: INTERNATIONAL TEST CONFERENCE, 1989. **Proceedings...** p. 194-200.

RAJSUMAN, R. **Digital Hardware Testing:** Transistor-level Fault Modelling and Testing, Artech House, 1992.

ROTH, J. P.; BOURICIUS, W. G.; SCHNEIDER, P. R. Programmed Algorithms to Compute Test to Detect and Distinguish Failures in Logic Circuits. **IEEE Transactions on Electronic Computers,** v.16, n. 10, p. 567-580, 1967.

SALUJA, K. K.; REDDY, S. M. On Minimally Testable Logic Networks. **IEEE Transactions on Computer,** v. 23, n.1, p. 552-554, 1974.

SESHU, S. On an Improved Diagnosis Program. **IEEE Transactions on Electronic Computers,** v.12, n. 2, p. 76-79, 1965.

SMITH, J. E.;METZE, G. Strongly Fault Secure Logic Networks. **IEEE Transactions on Computers,** v. 27, n. 6, p. 491-499, 1978.

STEPHESON, J. E.; GRASON, J. A Testability Measure for Register Transfer Level Digital Circuits, In: 6TH SYMPOSIUM FAULT-TOLERANT COMPUTING, 1976. **Proceeding...**, p. 101-107.

ULRICH, E. G.; BAKER, T.G. Concurrent Simulation of Nearly Identical Digital Networks. **IEEE Computer**, v.7, n.4, p.39 – 44, 1974.

WADSACK, R. L. Fault modeling and Logic Simulation of CMOS and MOS Integrated Circuits. **The Bell System Technical Journal**, 1978.

WILLIAMS, M. J. Y.; ANGELL, J. B. Enhancing Testability of Large Scale Integrated Circuits Via Test Points and Additional Logic. **IEEE Transactions on Computer**, 22, n.1, p. 46-60, 1973.

WOOD, C.T. The Quantitative Measure of testability, In: IEEE AUTOTESTCON, 1979. **Proceedings...** p. 286-291,.

ANEXO – Descrição do sistema em código VHDL

```
-- mux 2:1

ENTITY mux2 IS
PORT(a,b,c: IN bit;
     d: OUT bit);
END mux2;

ARCHITECTURE comportamento OF mux2 IS
BEGIN
d<=a WHEN c='0' ELSE
   b;
END comportamento;

-----------------------------------------

-- mux 4:1

ENTITY mux4 IS
PORT(a,b,c,d: IN bit;
     e: IN integer RANGE 0 TO 3;
     f: OUT bit);
END mux4;

ARCHITECTURE comportamento OF mux4 IS
BEGIN
f<=a WHEN e=0 ELSE
   b WHEN e=1 ELSE
   c WHEN e=2 ELSE
   d;
END comportamento;

-----------------------------------------



-- comparador

ENTITY comparador IS
PORT(st1,st2,sca,scb: IN bit;
     e: OUT bit);
END comparador;

ARCHITECTURE comportamento OF comparador IS
BEGIN
e<='1' WHEN ((st1/=sca) OR (st2/=scb)) ELSE
   '0';
END comportamento;
```

```
----------------------------------------

-- controle
-- clock,reset: sinais do sistema
-- e: sinal de falha provindo de comparador
-- ct: indicador de qual celula esta sob teste
-- erro: memoria para efeito de teste
-- s: sinal de substituicao da celula correspondente

ENTITY controle IS
PORT(clock,reset,e: IN bit;
     ct: OUT integer RANGE 0 TO 3;
     erro,s: OUT bit_vector (3 DOWNTO 0));
END controle;

ARCHITECTURE comportamento OF controle IS
TYPE estado IS (S0,S1,S2,FIM);
SIGNAL ea,pe: estado;
SIGNAL ce,ex: bit;
SIGNAL ctx: integer RANGE 0 TO 3;
SIGNAL errox: bit_vector (3 DOWNTO 0);
BEGIN

PROCESS(clock)
BEGIN
IF (clock'event AND clock='1') THEN ea<=pe;
END IF;
END PROCESS;

PROCESS(clock)
BEGIN
IF (clock'event AND clock='1') THEN
 IF reset='1' THEN ctx<=0;
  ELSIF ce='1' THEN ctx<=ctx+1;
 END IF;
END IF;
END PROCESS;
ct<=ctx;

ex<=errox(0) WHEN ctx=0 ELSE
    errox(1) WHEN ctx=1 ELSE
    errox(2) WHEN ctx=2 ELSE
    errox(3);

PROCESS(ea)
BEGIN
IF reset='1' THEN pe<=S0;
 ELSE CASE ea IS
```

```
      WHEN S0=> ce<='0';
              IF ex='1' THEN pe<=S2; ELSE pe<=S1;
              END IF;
      WHEN S1=> ce<='1';
              pe<=S0;
      WHEN S2=> ce<='0';
              pe<=FIM;
      WHEN FIM=> ce<='0';
               pe<=FIM;
     END CASE;
END IF;
END PROCESS;

PROCESS(clock)
BEGIN
IF reset='1' THEN
 FOR i IN 0 TO 3 LOOP
  errox(i)<='0'; s(i)<='0';
 END LOOP;
 ELSIF (clock'event AND clock='1') THEN
  CASE ea IS
   WHEN S0=> IF e='1' THEN
             IF ctx=0 THEN errox(0)<='1';
              ELSIF ctx=1 THEN errox(1)<='1';
               ELSIF ctx=2 THEN errox(2)<='1';
                ELSE errox(3)<='1';
             END IF;
            END IF;
   WHEN S2=> IF ctx=0 THEN s(0)<='1';
             ELSIF ctx=1 THEN s(1)<='1';
              ELSIF ctx=2 THEN s(2)<='1';
               ELSE s(3)<='1';
            END IF;
   WHEN OTHERS => --
  END CASE;
END IF;
erro<=errox;
END PROCESS;

END comportamento;

-----------------------------------------

-- celula
-- e1,e2: entradas da celula
-- s1,s2: saidas da celula
-- c1: controle da celula
-- c2: controle para substituicao da celula
-- st1,st2: saidas da celula teste
```

```
-- sc1,sc2: saidas da celula para efeito de teste
-- cf1: controle falha 1 - stuck-at 0 (0-falha 1-normal) / no a
-- cf2: controle falha 2 - curto-circuito (0-normal 1-falha) / no a
                                        (strong driver)

ENTITY celula IS
PORT(e1,e2,c1,c2,st1,st2,cf1,cf2: IN bit;
    s1,s2,sc1,sc2: OUT bit);
END celula;

ARCHITECTURE comportamento OF celula IS
COMPONENT mux2
 PORT(a,b,c: IN bit; d: OUT bit);
END COMPONENT;
SIGNAL a,b,ax,bx: bit;
BEGIN
u1: mux2 PORT MAP(e1,e2,c1,a);
u2: mux2 PORT MAP(e2,e1,c1,b);
u3: mux2 PORT MAP(ax,st1,c2,s1);
u4: mux2 PORT MAP(bx,st2,c2,s2);
u5: mux2 PORT MAP(b,ax,cf2,bx); -- falha curto-circuito
ax<=a AND cf1; -- falha stuck-at 0
sc1<=ax;
sc2<=bx;
END comportamento;

-----------------------------------------

-- celulat
-- e1,e2,e3,e4,e5,e6,e7,e8,c1,c2,c3,c4: entradas das celulas escolhidas
                                        por ct - teste
-- sc1,sc2,sc3,sc4,sc5,sc6,sc7,sc8: saidas das celulas escolhidas por
                                        ct - teste
-- ct: indicador de qual celula esta sob teste
-- st1,st2: saidas da celula-teste para teste e substituicao de celula falha
-- sca,scb: saidas das celulas sob teste

ENTITY celulat IS
PORT(e1,e2,e3,e4,e5,e6,e7,e8,c1,c2,c3,c4,sc1,sc2,sc3,sc4,sc5,sc6,sc7,sc8:
                                        IN bit;
    ct: IN integer RANGE 0 TO 3;
    st1,st2,sca,scb: OUT bit);
END celulat;

ARCHITECTURE comportamento OF celulat IS
COMPONENT mux2
 PORT(a,b,c: IN bit; d: OUT bit);
END COMPONENT;
COMPONENT mux4
```

```
 PORT(a,b,c,d: IN bit; e: IN integer RANGE 0 TO 3; f: OUT bit);
END COMPONENT;
SIGNAL ea,eb,c: bit;
BEGIN
u1: mux4 PORT MAP(e1,e3,e5,e7,ct,ea);
u2: mux4 PORT MAP(c1,c2,c3,c4,ct,c);
u3: mux4 PORT MAP(e2,e4,e6,e8,ct,eb);
u4: mux4 PORT MAP(sc1,sc3,sc5,sc7,ct,sca);
u5: mux4 PORT MAP(sc2,sc4,sc6,sc8,ct,scb);
u6: mux2 PORT MAP(ea,eb,c,st1);
u7: mux2 PORT MAP(eb,ea,c,st2);
END comportamento;

----------------------------------------

-- roteador
-- clock,reset: sinais de controle do sistema
-- e1,e2,e3,e4,c1,c2,c3,c4: entradas do roteador
-- s1,s2,s3,s4: saidas do roteador
-- falha: celulas com falha
-- cs: celula substituida
-- cf1,cf2,cf3,cf4: controle de falha (stuck-at 0) das celulas
-- cf5: controle de falha (stuck-at 1) do roteador (e1)
-- cf6,cf7,cf8,cf9: controle de falha (curto-circuito) das celulas
-- cf10: controle de falha (curto-circuito) do roteador - no b
                                    (strong driver)

ENTITY roteador IS
PORT(clock,reset,e1,e2,e3,e4,c1,c2,c3,c4,cf1,cf2,cf3,cf4,cf5,cf6,cf7,cf8,
                                          cf9,cf10: IN bit;
    s1,s2,s3,s4: OUT bit;
    ex: OUT bit; -- temporario
    ctx: OUT integer RANGE 0 TO 3; -- temporario
    falha,cs: OUT bit_vector (3 DOWNTO 0));
END roteador;

ARCHITECTURE comportamento OF roteador IS
COMPONENT mux2
 PORT(a,b,c: IN          bit; d: OUT bit);
END COMPONENT;
COMPONENT celula
 PORT(e1,e2,c1,c2,st1,st2,cf1,cf2: IN bit; s1,s2,sc1,sc2: OUT bit);
END COMPONENT;
COMPONENT celulat
 PORT(e1,e2,e3,e4,e5,e6,e7,e8,c1,c2,c3,c4,sc1,sc2,sc3,sc4,sc5,sc6,sc7,sc8:
                                                IN bit;
     ct: IN integer RANGE 0 TO 3; st1,st2,sca,scb: OUT bit);
END COMPONENT;
COMPONENT comparador
```

```
 PORT(st1,st2,sca,scb: IN bit; e: OUT bit);
END COMPONENT;
COMPONENT controle
 PORT(clock,reset,e: IN bit; ct: OUT integer RANGE 0 TO 3;
     erro,s: OUT bit_vector (3 DOWNTO 0));
END COMPONENT;
SIGNAL a,b,c,cx,d,e,e1x,st1,st2,sc1,sc2,sc3,sc4,sc5,sc6,sc7,sc8,sca,scb: bit;
SIGNAL erro,s: bit_vector (3 DOWNTO 0);
SIGNAL ct: integer RANGE 0 TO 3;
BEGIN
u1: celula PORT MAP(e1x,e2,c1,s(0),st1,st2,cf1,cf6,a,b,sc1,sc2);
u2: celula PORT MAP(a,cx,c2,s(1),st1,st2,cf2,cf7,s1,s2,sc3,sc4);
u3: celula PORT MAP(e3,e4,c3,s(2),st1,st2,cf3,cf8,c,d,sc5,sc6);
u4: celula PORT MAP(b,d,c4,s(3),st1,st2,cf4,cf9,s3,s4,sc7,sc8);
u5: celulat PORT MAP(e1x,e2,a,cx,e3,e4,b,d,c1,c2,c3,c4,sc1,sc2,sc3,sc4,sc5,
                                  sc6,sc7,sc8,ct,st1,st2,sca,scb);
u6: comparador PORT MAP(st1,st2,sca,scb,e);
u7: controle PORT MAP(clock,reset,e,ct,erro,s);
u8: mux2 PORT MAP(c,b,cf10,cx); -- falha curto-circuito
e1x<=e1 OR cf5; -- falha suck-at
falha<=erro;
cs<=s;
ctx<=ct; -- temporario
ex<=e; -- temporario
END comportamento;
```