

UNIVERSIDADE FEDERAL DE PELOTAS
INSTITUTO DE FÍSICA E MATEMÁTICA
DEPARTAMENTO DE INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO



Análise Automática da Propagação de *Single-Event Transients* Considerando Análise Temporal e Lógica

Matheus Porciuncula Braga

Pelotas, 2007

MATHEUS PORCIUNCULA BRAGA

**ANÁLISE AUTOMÁTICA DA PROPAGAÇÃO DE *SINGLE-EVENT TRANSIENTS*
CONSIDERANDO ANÁLISE TEMPORAL E LÓGICA**

Trabalho acadêmico apresentado ao Curso de Bacharelado em Ciência da Computação da Universidade Federal de Pelotas, como requisito parcial à obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Luciano Volcan Agostini

Co-Orientador: Prof. Dr. José Luís Almada Güntzel

PELOTAS, 2007

Dados de catalogação na fonte:
Ubirajara Buddin Cruz – CRB-10/901
Biblioteca de Ciência & Tecnologia - UFPel

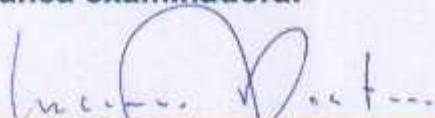
B813a Braga, Matheus Porciuncula

Análise automática da propagação de single-event transients considerando análise temporal e lógica / Matheus Porciuncula Braga ; orientador Luciano Volcan Agostini ; co-orientador José Luís Almada Güntzel. – Pelotas, 2007. – 96f. - Monografia (Conclusão de curso). Curso de Bacharelado em Ciência da Computação. Departamento de Informática. Instituto de Física e Matemática. Universidade Federal de Pelotas. Pelotas, 2007.

1.Informática. 2.Microeletrônica. 3.Efeitos da radiação em circuitos integrados. 4.Falhas transientes. 5.Algoritmos de ATPG. 6.Single-event transients (SETs). 7.Algoritmo PO-DEM. I.Agostini, Luciano Volcan. II.Güntzel, José Luís Almada. III.Título.

CDD: 004.2

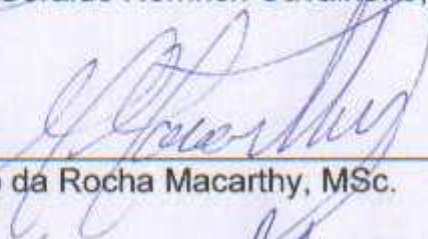
Banca examinadora:



Prof. Luciano Volcan Agostini, Dr. (Orientador)



Prof. Gerson Geraldo Homrich Cavaleiro, Dr.



Prof. Marcello da Rocha Macarthy, MSc.



Prof. Sandro Vilela da Silva, MSc.

*“Acima de tudo isto, porém,
esteja o amor, que é o vínculo da perfeição”*

Cl. 3:14

Agradecimentos

Em primeiro lugar quero agradecer aos meus pais (Valter e Gilda) por sempre serem meu guia e uma fonte de luz sempre iluminando meu caminho, mesmo em momentos em que nenhuma outra luz está brilhando. Sem eles, nada na minha vida seria possível. A eles meus sinceros e maiores agradecimentos pelo enorme amor compartilhado e pela educação que me deram. Agradeço também a eles por terem aberto mão de alguns de seus sonhos para que pudéssemos conviver mais e, assim, podendo compartilhar melhor o grande amor que sentem por mim. Além de terem me agüentado nos momentos de mau humor que muitas vezes aconteceram após um dia intenso e cansativo na faculdade :-). Pai e mãe, amo muito vocês.

Em segundo lugar quero agradecer ao meu irmão Fabiano que fez parte da minha educação sendo quase um segundo pai para mim. E, também, agradecer ao meu primo Ricardo por ter estado presente em boa parte da minha infância e adolescência sendo um outro irmão para mim.

Quero também agradecer aos corretores do processo seletivo do vestibular de verão da UFPel de 2002 que não gostaram das minhas respostas na segunda fase do vestibular fazendo com que eu reprovasse no vestibular daquele ano para o curso de Direito e, então, escolhendo Ciência da Computação no ano seguinte. Não poderia ter feito uma escolha melhor.

Um agradecimento em especial ao professor Güntzel que foi meu professor durante quase todos os semestres do curso e meu orientador durante três anos, sempre muito dedicado e um exemplo que deve ser seguido por todos. Um **Muito Obrigado** por ter sido paciente comigo e me ajudado nos momentos em que mais precisava. Também o agradeço por ser um ótimo professor de algoritmos, de arquiteturas, de microeletrônica, de profissionalismo, de respeito, de didática e, principalmente, de amizade.

A todos os meus colegas de trabalho do GACI: Helen, Zanetti, Meskita, Guilherme, João, Élvio, Fabiane e Rafael por tornarem o grupo muito mais que um ambiente de trabalho, mas também um ambiente agradável para se divertir. Como o Güntzel costuma dizer – o GACI não é um grupo, é uma família. E por ser uma família não posso deixar de citar os ex-gacistas: Thaísa, Carolina, Marcelo, André, Roger que também contribuíram para a consolidação da família GACI. Quero

agradecer em especial ao Guilherme por ter me ajudado incansavelmente na elaboração desse trabalho e por ser um amigo para todas as horas.

Ao professor Luciano Agostini pela sua dedicação como professor e amigo. E também por ter cedido a sua casa para os churrascos de fim de ano do GACI :P.

A todos os professores do curso que ainda não mencionei e com quem convivi, em especial ao professor Gil por seus sábios conselhos sempre pertinentes.

Agradeço à Helen que no início era apenas uma colega de aula, mas a partir do terceiro semestre tornou-se uma companheira permanente de trabalhos de aula. Porém, rapidamente, devido a sua amizade inesgotável e o seu grandioso coração tornou-se também uma grande e imprescindível amiga. À Helen, agradeço pelos puxões de orelha (:P) necessários e fundamentais para a conclusão desse trabalho (eu percebi que não consigo fazer um trabalho sem ela :P). Não posso deixar de agradecer aos pais da Helen (Delmar e Antônia) por sempre terem me tratado como um filho. Helen, agradeço-te do fundo do meu coração por ter demonstrado todas as virtudes que se deseja de uma grande amiga e companheira e por ter compartilhado ótimos momentos de alegria e felicidade e, dos quais, nunca esquecerei.

Ao Zanetti pela amizade e pelo companheirismo que sempre demonstrou com todos que o cercam; é impossível conhecê-lo e não ser seu amigo. Zanetti, muito obrigado pela tua grande amizade que tenho certeza que será para sempre.

Ao Meskita pela grande amizade e pelo senso de humor que fazem com que ninguém consiga estar triste ao seu redor.

A todos os meus colegas de aula que sempre fizeram da sala de aula um ambiente muito divertido e descontraído e em especial ao grupo de estudos para as provas do Gil. Quero agradecer em especial aos colegas e amigos que estiveram mais presentes nessa etapa da minha vida: Helen, Meskita, Zanetti, Virginia, Barros, Presidente e Foguinho.

Aos meus amigos que conheci na faculdade e em especial, Thaísa, Carol, Bea, Gustavo, Érico e Joel. Obrigado por tudo!

Aos meus amigos que colecionei durante toda a minha vida, entre eles: Gusmão, Dione, Fabiana (minha cunhada preferida), Michele, Christian, Daniel “Jedi”, Tatiana (minha irmã distante :P), Felipe, Luciano “Salsicha”, Suélen e toda a família Mussi :P. Muito obrigado pela força e por todo o carinho que me deram.

A todos, Muito Obrigado por tudo!

Resumo

A contínua redução das dimensões dos transistores fabricados em tecnologia CMOS estado-da-arte vem aumentando a probabilidade de ocorrência de falhas transientes induzidas por radiação, também conhecidas como *soft-errors*. Dentre as falhas transientes, as duas categorias mais comuns são os *Single-Event Upsets* (SEUs) e os *Single-Event Transients* (SETs). Um SEU ocorre quando uma partícula carregada colide com uma célula de memória, causando a inversão do valor armazenado, enquanto que um SET ocorre quando uma partícula carregada atinge uma região sensível da lógica combinacional. Neste último caso, um pulso de tensão pode ser gerado na saída da porta com falha. Esse pulso pode ser propagado pela lógica, podendo vir a ser armazenado pelos registradores das saídas do circuito, tornando-se então um *soft-error*. O teste de circuitos é a fase mais importante e custosa do projeto de circuitos integrados. A técnica mais conhecida de teste de circuitos é a cobertura de todos os vetores de entradas possíveis. Porém essa técnica demanda tempos de execução proibitivamente longos, visto que o número de vetores de entrada cresce exponencialmente com o número de entradas do circuito. O presente trabalho propõe uma técnica para a análise automática da propagação de pulsos gerados por SETs em circuitos combinacionais. Esta técnica é composta por uma análise do mascaramento por *latching-window* baseado em análise de *timing* topológica (TTA – *Topological Timing Analysis*), a qual visa reduzir o número de linhas nas quais se precisam injetar falhas transientes, e uma segunda etapa que realiza análise de mascaramento lógico mediante a aplicação do algoritmo de ATPG (*Automatic Test-Pattern Generation*) conhecido por PODEM (*Path-Oriented DEcision Making*), o qual realiza uma busca sistemática no espaço de entrada do circuito. Ao final, a taxa de falhas transientes (SER – *soft-error rate*) é calculada com base na proporção de falhas que conseguem atingir alguma das saídas primárias do circuito e ser capturadas por algum registrador de saída.

Palavras-Chave: Microeletrônica. Falhas Transientes. Efeitos da Radiação em Circuitos Integrados. *Single-Event Transients* (SETs). Algoritmos de ATPG. Algoritmo PODEM.

Abstract

The continuous shrinking of the transistor dimensions manufactured in state-of-the-art CMOS technology is increasing the probability of radiation-induced transient faults (also called soft-errors) to occur. Among the transient faults, the two more common categories are the Single-Event Upsets (SEUs) and the Single-Event Transients (SETs). An SEU occurs when a charged particle strikes with a memory cell, causing an inversion of the stored value, while an SET occurs when a charged particle strikes a sensitive region of the combinational logic. In the latter case, a voltage pulse may be generated at the output of the faulty gate. This pulse may be propagated down through the logic, and can be stored by the circuit outputs register, thus becoming a soft-error. Testing is the most important and expensive phase of integrated circuit design. The most naïve circuit testing technique is the coverage of all possible input vectors. However this technique requires prohibitively long execution times because the number of input vectors grows exponentially with the number of circuit's inputs. The present work proposes a technique for the automatic analysis of SET generated pulses in combinational circuits. This technique consists of a latching-window masking analysis based on the Topological Timing Analysis (TTA), that tries to reduce the number of lines where the transient fault injection is needed, and a second phase that accomplishes a logical masking analysis through the application of the PODEM (Path-Oriented Decision Making) ATPG (Automatic Test-Pattern Generation) algorithm, that makes a systematic search in the circuit's input space. Finally, the SER (soft-error rate) is computed based on the ratio between the faults that achieve a circuit's output and are captured by an output register and the total possible faults.

Keywords: Microelectronics. Transient Faults. Radiation Effects in Integrated Circuits. Single-Event Transients (SETs). ATPG Algorithms. PODEM Algorithm.

Lista de Figuras

Figura 1 – Colisão de uma partícula carregada em um circuito CMOS.....	16
Figura 2 – SEU em uma célula de memória SRAM	22
Figura 3 – Atenuação de um pulso transiente pela lógica combinacional	23
Figura 4 – Mascaramento lógico em uma porta NAND	24
Figura 5 – Fases da coleta de carga em uma região sensível	26
Figura 6 – Formas de onda típicas.....	27
Figura 7 – Modelagem do mecanismo de geração do pulso transiente	28
Figura 8 – Modelagens para a análise de propagação de um SET.....	28
Figura 9 – Propagação de um pulso transiente pela lógica combinacional.....	30
Figura 10 – Propagação do pulso transiente dentro da CB.....	32
Figura 11 – A relação entre a largura do pulso transiente de entrada e o de saída de uma CB.	33
Figura 12 – Algoritmo clássico de simulação de falha.....	35
Figura 13 – Algoritmo proposto	37
Figura 14 – Caracterização da geração do pulso através da simulação elétrica.....	40
Figura 15 – A modificação do BDD após um SET.....	41
Figura 16 – Propagação do pulso transiente e os BDDs dos nodos do circuito	41
Figura 17 – Particionamento do circuito	42
Figura 18 – Exemplo de circuito com reconvergência.....	44
Figura 19 – NAND com entrada em valor inicial controlante livre de falha.....	45
Figura 20 – NAND com entrada em valor inicial controlante com SET	46
Figura 21 – NAND sem entrada com valor inicial controlante	46
Figura 22 – Modelagem do pulso transiente	51
Figura 23 – Janela de amostragem de um registrador.....	52
Figura 24 – O circuito de um somador completo.....	54
Figura 25 – DAG do circuito da Fig. 24	54
Figura 26 – Procedimento <i>topological sort</i> em pseudocódigo	55
Figura 27 – Pseudocódigo do algoritmo <i>reverse topological sort</i>	57
Figura 28 – Estrutura de um nodo do grafo.....	58
Figura 29 – Resultados para os circuitos com poucas portas em relação à profundidade lógica	62
Figura 30 - Resultados para os circuitos com um grande número de portas em relação à profundidade lógica	62
Figura 31 – Resultados para o circuito alu4	63
Figura 32 – Resultados para o circuito C499	64
Figura 33 – Exemplo de sensibilização de um caminho.....	66
Figura 34 – Modelagem do pulso transiente para o Cálculo-D.....	70
Figura 35 – Cálculo da controlabilidade combinacional de portas NANDs e NORs..	72
Figura 36 – Cálculo da observabilidade combinacional de portas NANDs e NORs..	72
Figura 37 – Cálculo da controlabilidade probabilística de portas NANDs e NORs....	73
Figura 38 – Cálculo da observabilidade probabilística de portas NANDs e NORs....	73
Figura 39 – Fronteira- <i>D</i> contendo uma porta	75
Figura 40 – Exemplo de fronteira- <i>J</i>	75
Figura 41 – Propagação do sinal de erro até POs	76
Figura 42 – Algoritmo D	76
Figura 43 – Procedimento recursivo do algoritmo D	77
Figura 44 – Conflito na justificação de valores	78
Figura 45 – Circuito sem caminho-X	79

Figura 46 – Fluxograma em alto nível do algoritmo PODEM	80
Figura 47 – Algoritmo alto-nível para o algoritmo PODEM.....	81
Figura 48 – Procedimento recursivo do algoritmo PODEM.....	81
Figura 49 – Pseudocódigo do procedimento <i>objetivo</i> ()	82
Figura 50 – Algoritmo recursivo em pseudocódigo do procedimento <i>backtrace</i> ()....	83
Figura 51 – Estrutura de uma aresta do grafo.....	85
Figura 52 – Resultados comparativos entre a análise considerando somente mascaramento por <i>latching-window</i> (pré-processamento) e análise considerando mascaramento lógico para o circuito alu4	86
Figura 53 – Resultados comparativos entre a análise considerando somente mascaramento por <i>latching-window</i> (pré-processamento) e análise considerando mascaramento lógico para o circuito C432	87

Lista de Tabelas

Tabela 1 – Relação entre os atrasos de uma porta lógica e o intervalo de duração da forma de onda equivalente, na determinação do pulso de saída	47
Tabela 2 – Característica dos circuitos-exemplo e os tempos totais de execução ...	60
Tabela 3 – Extensão de valores lógicos para a notação D	69
Tabela 4 – Tabela-verdade da operação de negação no Cálculo-D	69
Tabela 5 – Tabela-verdade da operação E no Cálculo-D	69
Tabela 6 – Tabela-verdade da operação OU no Cálculo-D	69
Tabela 7 – SER dos circuitos utilizando o modo LW + LO de GILDA utilizando diferentes TIs.....	86
Tabela 8 – Comparação dos tempos de execução para os 3 modos de GILDA.....	87
Tabela 9 – Comparação entre o modo LO da ferramenta GILDA com a ferramenta ASPA em tempo de execução e SER resultante.....	88

Lista de Abreviaturas e Siglas

AGDA	<i>Automatic Gate Delay Analyzer</i>
ASPA	<i>Automatic SET Propagation Analyzer</i>
ATPG	<i>Automatic Test-Pattern Generation</i>
BDD	<i>Binary Decision Diagram</i>
CB	<i>Channel-connected Block</i>
CMOS	<i>Complementary Metal-Oxide Silicon</i>
DAG	<i>Direct Acyclic Graph</i>
DRAM	<i>Dynamic Random Access Memory</i>
ECAT	<i>Error Correction And Translation</i>
FIT	<i>Failure In Time</i>
GILDA	<i>GACI's Implementation of Latching-window and D propagation Analysis</i>
LET	<i>Linear Energy Transfer</i>
mds	<i>maximal delay from source</i>
mdsf	<i>maximal delay from source for a falling transition</i>
mdsr	<i>maximal delay from source for a rising transition</i>
mdt	<i>maximal delay to terminal</i>
mdtf	<i>maximal delay to terminal for a falling transition</i>
mdtr	<i>maximal delay to terminal for a rising transition</i>
NMOS	<i>N-Channel MOS</i>
PI	<i>Primary Input</i>
PMOS	<i>P-Channel MOS</i>
PO	<i>Primary Output</i>
PODEM	<i>Path-Oriented DEcision Making</i>
SDF	<i>Standard Delay Format</i>
SEE	<i>Single-Event Effect</i>
SER	<i>Soft-Error Rate</i>
SET	<i>Single-Event Transient</i>
SEU	<i>Single-Event Upset</i>
SRAM	<i>Static Random Access Memory</i>
STA	<i>Statistical Timing Analysis</i>
t_{hold}	<i>tempo de hold</i>
t_{setup}	<i>tempo de setup</i>

TBT-SET *Timed-Boolean Topological SET Propagation*

td_{HL} *topological delay high-low*

td_{LH} *topological delay low-high*

TTA *Topological Timing Analysis*

Sumário

1	INTRODUÇÃO	15
1.1	CONTRIBUIÇÃO DESTE TRABALHO	18
1.2	ORGANIZAÇÃO DA MONOGRAFIA	18
2	SINGLE-EVENT TRANSIENTS	20
2.1	SOFT SINGLE-EVENT EFFECTS (SEES).....	21
2.2	GERAÇÃO DO PULSO E OS MECANISMOS DE MODELAGEM DO PULSO	25
3	TÉCNICAS EXISTENTES PARA ANÁLISE DE PROPAGAÇÃO DE SETS	29
3.1	MÉTODO BASEADO EM SIMULAÇÃO NO NÍVEL DE CHAVES	31
3.2	MÉTODO BASEADO EM SIMULAÇÃO NO NÍVEL LÓGICO	34
3.3	MÉTODO BASEADO EM BDDs.....	38
3.4	MÉTODO BASEADO EM PROPAGAÇÃO TOPOLÓGICA BOOLEANO – TEMPORAL DE SETS.....	43
3.5	DA NECESSIDADE DE NOVOS MÉTODOS PARA A ANÁLISE DA PROPAGAÇÃO DO PULSO TRANSIENTE	48
4	ANÁLISE DE <i>TIMING</i> E A PROPAGAÇÃO DE PULSOS	50
4.1	ASPECTOS DE <i>TIMING</i> RELACIONADOS À PROPAGAÇÃO DO PULSO.....	50
4.2	USO DA ANÁLISE DE <i>TIMING</i> TOPOLÓGICA PARA AVALIAR OS ASPECTOS DE <i>TIMING</i> DA PROPAGAÇÃO DO PULSO TRANSIENTE	53
4.2.1	<i>Pré-processamento Baseado TTA</i>	56
4.2.2	<i>Resultados Obtidos</i>	59
5	ANÁLISE AUTOMÁTICA DO MASCARAMENTO LÓGICO	65
5.1	TRATANDO MASCARAMENTO LÓGICO	65
5.2	TÉCNICAS DE ATPG E SUA APLICAÇÃO PARA A ANÁLISE DO MASCARAMENTO LÓGICO.....	67
5.2.1	<i>Medidas de Testabilidade</i>	71
5.2.2	<i>Algoritmo D</i>	74
5.2.3	<i>Algoritmo PODEM</i>	78
5.3	RESULTADOS EXPERIMENTAIS	83
6	CONCLUSÕES	90
7	REFERÊNCIAS BIBLIOGRÁFICAS	93

1 Introdução

A contínua redução das dimensões dos transistores tem proporcionado um aumento sistemático na capacidade de integração dos circuitos fabricados em tecnologias CMOS (*Complementary Metal-Oxide Semiconductor*) estado-da-arte. Essa redução está permitindo que sejam fabricados circuitos com transistores de dimensões abaixo de 100nm (as chamadas tecnologias nanométricas).

Entretanto, está cada vez mais difícil garantir o pleno funcionamento de circuitos fabricados com tais tecnologias durante todo o período de vida útil devido ao aumento da vulnerabilidade desses circuitos a ruídos. Esse aumento adveio principalmente das reduções na tensão de alimentação e na tensão de limiar (*threshold*), necessárias para a evolução da tecnologia de fabricação CMOS.

Uma das grandes preocupações dos projetistas de circuitos integrados em tecnologias nanométricas reside no aumento da suscetibilidade a falhas transientes (COHEN et al., 1999). Conforme Baumann (2001), a maior fonte de falhas transientes em circuitos integrados é a radiação. No espaço sideral a atividade solar é a principal responsável pela radiação, a qual é constituída de partículas carregadas, como elétrons ou íons pesados, enquanto que ao nível do mar a principal fonte de falhas transientes são partículas alfa oriundas de impurezas presentes no encapsulamento (BAUMANN, 2005).

Até recentemente a análise do efeito da radiação em circuitos integrados era um problema apenas dos projetistas de circuitos desenvolvidos para aplicações espaciais, devido ao maior fluxo de partículas de alta energia no espaço sideral. Porém, devido ao avanço tecnológico, a energia necessária para causar uma falha transiente está se tornando cada vez menor, de modo que mesmo os circuitos projetados para operar na superfície da Terra começam a ser suscetíveis a tais falhas. Desta forma, os projetistas têm se preocupado cada vez mais com a análise

da suscetibilidade a falhas transientes e também com a proteção dos circuitos. Como consequência, a investigação de métodos de análise e de proteção de circuitos tem se intensificado recentemente.

Uma falha transiente acontece quando uma partícula carregada colide com uma região sensível de um transistor, isto é, a junção P-N do dreno de um transistor *off* (desligado) (BAUMANN, 2001). A partícula poderá ter energia suficiente para atingir a região ativa do transistor, formando uma trilha de ionização a qual poderá atingir o seu substrato (ou poço, dependendo do tipo de transistor e da sua tecnologia de fabricação), como mostrado na Fig. 1. Caso atinja o substrato (ou poço) do transistor, uma corrente elétrica irá se estabelecer (O'BRYAN et al., 2002). A corrente elétrica estabelecida poderá apresentar duração e/ou amplitude suficiente para carregar ou descarregar o nodo, gerando, então, um pulso transiente de tensão que poderá ser interpretado como uma mudança de nível lógico. Tal corrente elétrica normalmente é modelada como uma dupla exponencial cuja duração e amplitude são dependentes de fatores como a energia da partícula, as dimensões da área ativa afetada pela colisão e perfil de distribuição dos dopantes (MESSENGER, 1982).

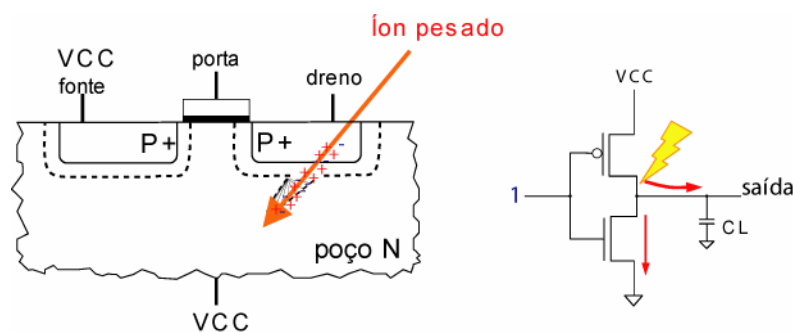


Figura 1 – Colisão de uma partícula carregada em um circuito CMOS

Uma partícula radioativa pode colidir com uma região sensível da lógica combinacional ou com uma região sensível da lógica seqüencial do circuito (ALEXANDRESCU; ANGHEL; NICOLAIDIS, 2002).

A colisão de uma partícula com uma célula de memória e a consequente mudança no valor por ela armazenado é referenciada por *Single-Event Upset* (SEU) (BAUMANN, 2001; NICOLAIDIS, 2005).

Quando uma partícula carregada colide com uma região sensível de um circuito combinacional, um pulso transiente pode ser gerado. Este fenômeno é

conhecido como *Single-Event Transient* (SET) (WIRTH et al., 2005; ALEXANDRESCU; ANGHEL; NICOLAIDIS, 2002). O pulso transiente gerado pode ser atenuado ou filtrado devido, principalmente, ao atraso da porta por onde ele passa e a capacitância a ela associada. Esse fenômeno é conhecido como **maskamento elétrico**. Além disso, o pulso pode ser bloqueado devido às características lógicas da porta. Tal fenômeno é chamado de **maskamento lógico**. Se o pulso transiente gerado não for mascarado lógica ou eletricamente pelo circuito (BAZE; BUCHNER, 1997; WIRTH et al., 2005), então o SET irá propagar-se pela lógica até alguma saída primária, podendo ser capturado por um elemento de memória e assim, ser erroneamente interpretado como um valor lógico correto.

No entanto, à medida que as dimensões dos transistores diminui, menor tende a ser o atraso das portas e ao mesmo tempo, maior tende a ser a duração dos pulsos transientes oriundos de SETs. Então, conclui-se que à medida que a tecnologia CMOS evolui, menor tende a ser o efeito do maskamento elétrico e maior tende a ser a importância do maskamento lógico (ZHANG; WANG; ORSHANSKY, 2006).

As células de memória tendem a ser mais suscetíveis que a lógica combinacional devido as suas características construtivas (basicamente, devido ao laço de realimentação existente). Portanto SEUs têm recebido mais atenção dos projetistas que SETs. Porém, é previsto que devido à constante redução das dimensões dos transistores, por volta do ano 2010 a taxa de erros provocados por SETs será tão grande quanto a taxa de erros em memórias não protegidas (SHIVAKUMAR et al., 2002). Com isso, surge a necessidade do desenvolvimento de técnicas para evitar que os SETs gerados possam ser capturados pelos elementos de memória, causando assim **falhas não-permanentes**. Para tanto, é necessário analisar a suscetibilidade do circuito a falhas não-permanentes através da análise da propagação de SETs.

Existem diversas técnicas para análise da propagação de SETs, sendo a maioria delas baseadas em simulação. Entretanto, técnicas baseadas em simulação necessitam de uma exploração exaustiva do espaço de entrada para cada falha a ser testada. Ou seja, para um circuito com n entradas primárias e m portas lógicas, são necessárias 2^n simulações para cada uma das $2 \times m$ possíveis falhas, o que torna essa técnica impraticável para grandes circuitos. Então, é necessário desenvolver técnicas para a análise da suscetibilidade de circuitos a SETs que

reduzam esse proibitivo tempo de execução sem grandes perdas na precisão da análise.

1.1 Contribuição deste Trabalho

Dada a crescente probabilidade de ocorrência de um SET em circuitos combinacionais em tecnologia nanométrica, este trabalho investiga tal problema e propõe um método para a análise automática da propagação de um SET em circuitos combinacional.

Para tanto, este trabalho investiga as condições básicas necessárias para a propagação de um pulso transiente originado por um SET ao longo de um circuito combinacional. Também são levantadas as condições para que tal pulso seja capturado por elementos de memória. Com base nessa investigação, propõe-se uma técnica capaz de analisar a suscetibilidade de circuitos combinacionais a SETs. Tal técnica é capaz de levar em conta tanto o mascaramento por *latching-window* quanto o mascaramento lógico.

1.2 Organização da Monografia

Esta monografia está organizada da seguinte maneira. No capítulo 2 são expostos os conceitos sobre as falhas transientes em circuitos CMOS, abordando as principais fontes de falhas e as conseqüências da exposição dos circuitos CMOS estado-da-arte à radiação.

O capítulo 3 discorre sobre a análise da suscetibilidade dos circuitos a SETs, abordando as técnicas baseadas em simulação e as técnicas baseadas em propagação topológica. Após, são apresentadas 3 técnicas estado-da-arte.

O capítulo 4 aborda as questões referentes às informações de *timing* do pulso transiente resultante de um SET e sua influência no mascaramento por *latching-window*. Na seqüência, é proposta uma técnica de pré-processamento do circuito baseada em análise de *timing* topológica, a qual visa levar em conta o mascaramento por *latching-window* e, ao mesmo tempo, reduzir o número de SETs que devem ser considerados na análise do mascaramento lógico. Resultados sobre

a suscetibilidade dos circuitos e tempo de execução da técnica proposta são apresentados ao final desse capítulo.

O capítulo 5 ressalta a importância de se considerar o mascaramento lógico na análise da propagação de SETs. Para tanto, é proposta a adoção de técnicas de geração de teste. Desta forma são apresentados os conceitos fundamentais de ATPG (*Automatic Test-Pattern Generation*). Em particular, é apresentado o algoritmo PODEM (*Path-Oriented DEcision Making*), cuja utilização é proposta nesse trabalho para análise do mascaramento lógico. Ao final do capítulo são mostrados os resultados da técnica proposta quando aplicada aos circuitos do banco de *benchmarks* MCNC, tanto em termos de tempos de execução como em termos de valores de SER (*soft-error rate*) dos circuitos analisados. Os tempos de execução obtidos com o uso de PODEM são comparados com os tempos de execução obtidos pelo uso de técnica baseada em simulação (NEVES, 2006).

2 *Single-Event Transients*

O avanço da tecnologia de fabricação CMOS, proporcionado pela redução nas dimensões dos transistores, tem sido um dos principais fatores para o aumento na capacidade de processamento dos computadores. Esse avanço tem levado a um aumento de quase 100% na frequência de relógio, redução de 30% na capacitância dos nodos e redução de 30% na tensão de alimentação a cada novo nodo tecnológico (DHILLON; DIRIL; CHATTERJEE; SINGH, 2006).

Em função dessas reduções nas dimensões dos transistores e na tensão de alimentação dos dispositivos eletrônicos, cada vez mais os circuitos têm se tornado suscetíveis à radiação. Ademais, as reduções na tensão de alimentação permitiram que pequenas variações de tensão possam ser interpretadas como mudanças do estado lógico de um sinal, e portanto, está cada vez mais difícil garantir que um circuito fabricado em tecnologia estado-da-arte irá funcionar de forma correta.

Portanto, esse avanço tecnológico tem se transformado em uma das grandes preocupações dos projetistas de circuitos integrados em tecnologias nanométricas devido ao conseqüente aumento da suscetibilidade a falhas transientes (COHEN et al., 1999). Diversos são os efeitos da radiação em dispositivos fabricados em tecnologia CMOS. Tais efeitos variam em magnitude e extensão, indo desde distorção de dados até danos permanentes nos dispositivos (DODD; MASSENGILL, 2003).

Uma falha transiente pode decorrer da exposição do circuito à radiação, quando partículas carregadas colidem com o mesmo. Essas colisões podem causar distúrbios de carga suficiente para reverter o estado lógico de um sinal lógico do circuito combinacional ou o valor de um dado armazenado em uma célula de memória, registrador ou *flip-flop* (BAUMANN, 2005). Esse tipo de falha é considerado transiente pois, se um novo dado for escrito na célula de memória, o

dispositivo irá armazenar o valor correto. Essas falhas são também referenciadas como ***soft-errors***.

A contínua redução das dimensões dos transistores tem proporcionado um aumento sistemático na capacidade de integração dos circuitos integrados fabricados em tecnologias estado-da-arte. Essa redução permitiu que fosse possível a fabricação de circuitos com transistores de dimensões abaixo de 100nm (as chamadas tecnologias nanométricas).

Entretanto, está cada vez mais difícil garantir o pleno funcionamento de circuitos fabricados com tais tecnologias durante todo o período de vida útil devido ao aumento da vulnerabilidade desses circuitos a ruídos. Esse aumento adveio das reduções na tensão de alimentação e na tensão de limiar (*threshold*), necessárias para a evolução da tecnologia de fabricação CMOS (*Complementary Metal-Oxide Silicon*).

Dentre a gama de efeitos induzidos pela radiação, um dos que tem se tornado assunto de maior preocupação para aplicações comerciais são os *Single-Event Effects* (SEEs). Como o nome já diz, SEEs são falhas provocadas por um único evento de radiação. A nível do mar a preocupação está nos *Soft SEEs*, os quais podem ser divididos em duas categorias: *Single-Event Transients* (SETs) e *Single-Event Upsets* (SEUs). Já em ambientes espaciais a grande preocupação está nos *Hard SEEs*.

O foco deste trabalho são os SEEs, mais especificamente os SETs. Na seção que segue serão comentados com mais detalhes os SETs e os SEUs.

2.1 *Soft Single-Event Effects* (SEEs)

Como já foi dito anteriormente, um *soft SEE* pode ser do tipo *Single-Event Transient* (SET) ou *Single-Event Upset* (SEU). Tal distinção vem da região onde a partícula carregada colide. Uma partícula carregada pode colidir com uma região sensível da lógica combinacional ou com uma região sensível da lógica seqüencial do circuito (ALEXANDRESCU; ANGHEL; NICOLAIDIS, 2002). As regiões sensíveis dos transistores são as junções P-N dos drenos dos transistores desligados (em estado *off*) (BAUMANN, 2001). Tais falhas são consideradas transientes (*soft-errors*)

pois não são originadas de defeitos de fabricação e não danificam o circuito afetado (BUSHNELL; AGRAWALL, 2000; HEIJMAN; NIEUWLAND, 2006)

Quando a colisão se dá em elemento de memória, se a partícula colidiu com o dreno de um transistor PMOS (*P-Channel MOS*), a capacitância associada a este dreno é carregada, ligando assim o transistor por ele controlado. Caso a região afetada seja o dreno de um transistor NMOS (*N-Channel MOS*), o efeito é o mesmo exceto que a capacitância associada é descarregada. Em ambos os casos o efeito resultante é a inversão do valor lógico armazenado pela célula de memória, o que também é conhecido como *bit-flip*. Esse fenômeno é referenciado como *Single-Event Upset* (SEU) (BAUMANN, 2001; NICOLAIDIS, 2005). A Fig. 2 mostra um SEU e o conseqüente *bit-flip* em uma célula de memória SRAM causado pela colisão de uma partícula carregada no dreno de um transistor PMOS.

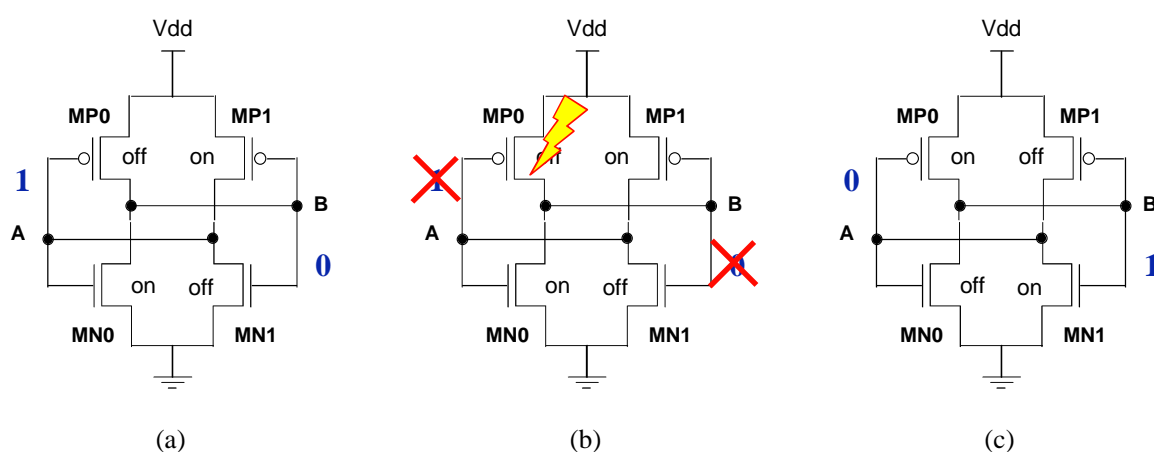


Figura 2 – SEU em uma célula de memória SRAM

Um *Single-Event Transient* (SET) ocorre quando uma partícula colide com uma região sensível de um circuito combinacional, gerando um pulso transiente (WIRTH et al., 2005; ALEXANDRESCU; ANGHEL; NICOLAIDIS, 2002). Se o pulso transiente não for mascarado lógica ou eletricamente pelo circuito (BAZE. BUCHNER, 1997; WIRTH et al., 2005), o SET irá se propagar até alguma saída primária do circuito, podendo eventualmente ser capturado por um elemento de memória e assim, originar um *soft-error*.

Com isso, é possível perceber que a probabilidade de um pulso propagar-se pela lógica combinacional até uma das saídas primárias tende a ser inversamente proporcional ao número médio de portas lógicas entre o ponto em que o pulso foi gerado e as saídas primárias. Além disso, quanto maior o número de portas lógicas

que o pulso precisa atravessar, maior é a probabilidade de ele ser completamente atenuado ou ser bloqueado pela lógica. Alguns trabalhos têm demonstrado que pulsos com duração da grandeza do atraso das portas lógicas tendem a ser atenuados ou até mesmo filtrados (OMANÑA et al., 2003; WIRTH et al., 2005).

Esse efeito de atenuação ou filtragem de um pulso devido, principalmente, ao atraso da porta e a capacitância a ela associada é conhecido como **maskamento elétrico**. Quando um pulso transiente propaga-se pela lógica combinacional, ele pode degradar em duração e amplitude. Simulações de circuitos mostraram que a degradação da amplitude ocorre quando uma entrada muda seu estado lógico antes da estabilização completa da porta em uma transição anterior. Quando isso ocorre, a saída da porta troca de valor na direção oposta antes de atingir a amplitude máxima, isto é, V_{DD} no caso de uma transição $0 \rightarrow 1$, ou GND no caso de uma transição $1 \rightarrow 0$. Além disso, o atraso das portas lógicas causa um aumento nos tempos de subida e descida do pulso transiente (WIRTH et al., 2005). Esse dois efeitos reduzem a duração de um pulso transiente na saída da porta, diminuindo a taxa de *soft-errors* causada por falhas transientes em circuitos combinacionais. A Fig. 3 mostra a atenuação de um pulso pela lógica combinacional. Cabe salientar que nessa figura foi desconsiderada a característica de inversão das portas CMOS a fim de se ter uma melhor visualização da atenuação.

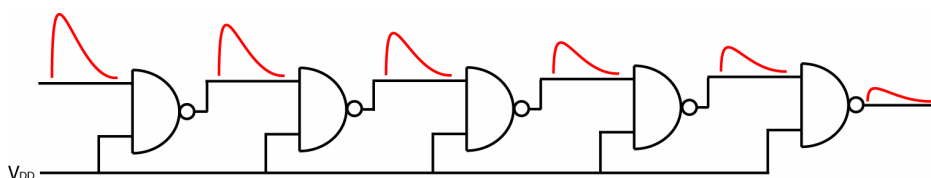


Figura 3 – Atenuação de um pulso transiente pela lógica combinacional

Os **maskamentos lógicos** estão associados ao conceito de controlabilidade de portas lógicas. Um pulso não irá se propagar se estiver ocorrendo em uma entrada de uma porta lógica que possua pelo menos uma outra entrada estável em seu **valor controlante**. O valor controlante de uma porta lógica é aquele valor que, quando aplicado a uma das entradas da porta determina o valor da saída da porta, independente do valor das demais entradas. A Fig. 4 mostra um exemplo de maskamento lógico. O maskamento lógico será tratado com mais profundidade no capítulo 5.

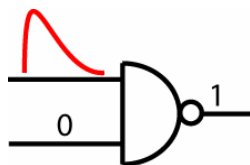


Figura 4 – Mascaramento lógico em uma porta NAND

Além desses dois tipos de mascaramento, há também o **mascaramento por *latching-window*** (CHA et al., 1996; SHIVAKUMAR et al., 2002; WIRTH et al., 2005). Esse mascaramento ocorre quando o pulso atinge o elemento de memória fora da janela de amostragem. A janela de amostragem representa o intervalo de tempo delimitado pelo t_{su} (tempo de *setup*) e t_h (tempo de *hold*). Esse fenômeno é explicado mais detalhadamente no capítulo 4.

Entretanto, à medida que as dimensões dos transistores diminuem, menor tende a ser o atraso das portas e ao mesmo tempo, maior tende a ser a duração dos pulsos transientes oriundos de SETs. Segundo Gadlage et al. (2004), para tecnologias nanométricas, se uma partícula carregada tem carga suficiente para criar um pulso transiente, então esse pulso não será atenuado pela lógica. Entretanto, estudos recentes feitos por Zhang, Wang e Orshansky (2006) mostraram que, para tecnologia CMOS de 100nm, a taxa de falhas não-permanentes oriundas de SETs pode ser sobrestimada em até 25 vezes se o efeito do mascaramento lógico for ignorado.

Para circuitos fabricados em tecnologia CMOS estado-da-arte, a ocorrência de falhas não-permanentes em memórias, causadas pela colisão de partículas carregadas, já são freqüentes (NIEUWLAND; JASAREVIC; JERIN, 2006). Com isso, diversos trabalhos têm focado as falhas originadas por SEUs e a proteção de memória.

Porém, com o avanço da tecnologia CMOS e o conseqüente aumento da quantidade de lógica combinacional nos circuitos, a contribuição dos SETs para o aumento da taxa de falhas está aumentando. Em 2010 a taxa de falhas transientes provocadas por SETs será igual à taxa de falhas em memórias não protegidas (SHIVAKUMAR et al., 2002). Assim, é necessário desenvolver técnicas para evitar que os SETs gerados nos blocos combinacionais venham a ser capturados por elementos de memória, causando *soft-errors*. Para tanto, é necessário analisar a suscetibilidade dos circuitos combinacionais a SETs.

Assim, técnicas para avaliar a suscetibilidade dos circuitos a falhas não-permanentes devem considerar a modelagem dos SETs a fim de poder tratá-los.

2.2 Geração do Pulso e os Mecanismos de Modelagem do Pulso

A modelagem da geração do pulso transiente é feita por meio da investigação do mecanismo básico da **coleta de carga**, que ocorre quando uma partícula carregada colide com uma junção p-n do silício. Esse mecanismo baseia-se na magnitude de distúrbio causado por uma partícula carregada, o qual depende da transferência de energia linear (*Linear Energy Transfer – LET*) da partícula. A transferência de energia é geralmente medido em ordem de mega-elétron volts (MeV) (BAUMANN, 2005). Além disso, cabe observar que no substrato de um circuito, um par elétron-lacuna é formado a cada 3.6 eV de energia dissipada pela partícula carregada.

Quando uma partícula colide com a região sensível de um circuito integrado, um caminho cilíndrico de pares elétron-lacuna com raio submicrônico é formado e uma grande concentração de portadores é formada na trilha deixada pela passagem da partícula carregada. Este efeito pode ser visto na Fig. 5a. Quando a trilha resultante atravessa ou atinge uma profundidade muito próxima da região de depleção, os portadores são rapidamente coletados pelo campo elétrico criando uma grande corrente/tensão transiente nesse nodo, e conseqüentemente, a região de depleção toma a forma de um funil (BAUMANN, 2005). Esse funil aumenta drasticamente a eficiência da coleção de carga por causa do aumento da profundidade da região de depleção dentro do substrato (ou poço), como pode ser visto na Fig. 5b. O tamanho do funil gerado é inversamente proporcional à densidade de dopagem do substrato. Essa fase de coleta e o funil formado duram apenas 1 ns e é seguida por uma fase onde a difusão começa a dominar o processo de coleta, como ilustrado na Fig. 5c.

Cargas adicionais são coletadas durante a difusão dos elétrons na região de depleção em uma larga escala de tempo (em ordem de centenas de nanosegundos) até todos os portadores excedentes serem coletados, recombinados ou difundidos para fora da junção. Quanto mais distante da junção ocorrer o evento, menor será a

quantidade de carga coletada e menor será a chance do evento gerar um *soft-error* (BAUMANN, 2005).

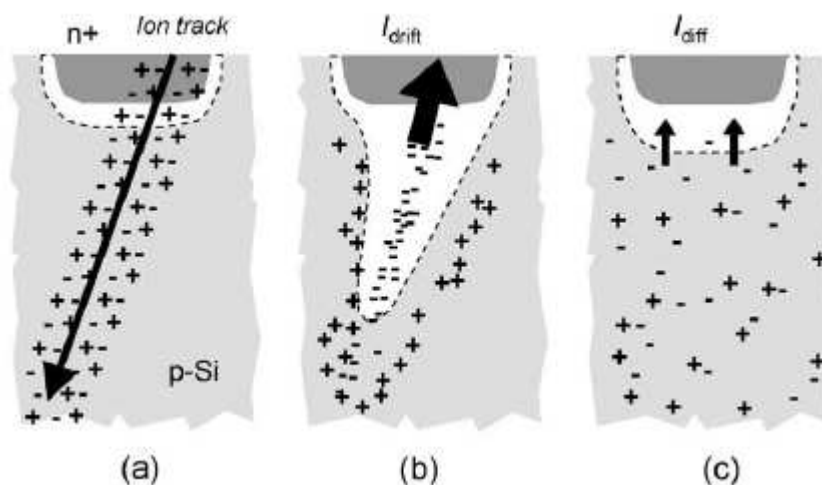


Figura 5 – Fases da coleta de carga em uma região sensível

Fonte: BAUMANN, 2005, p. 307.

A quantidade de **carga coletada** (Q_{coll}) depende de uma combinação complexa de fatores, dentre eles o tamanho do dispositivo, estrutura do substrato, tipo de dopagem, o tipo de partícula carregada, sua energia, sua trajetória, a posição inicial do evento no circuito e o estado do dispositivo. Além de Q_{coll} a sensibilidade dos dispositivos deve ser levada em conta também. Essa sensibilidade depende basicamente da carga mínima coletada em uma região sensível necessária para causar a inversão de um sinal lógico. Essa carga mínima é referenciada como **carga crítica** (Q_{crit}). A resposta do dispositivo à injeção de carga é dinâmica e depende da magnitude e das características temporais do pulso, e portanto, Q_{crit} depende dessas características, tornando sua modelagem um grande desafio (BAUMANN, 2005).

Em células de memória do tipo DRAM (*Dynamic Random Access Memory*), por exemplo, um *soft-error* ocorre quando a colisão de uma partícula carregada ocorrer tão próximo da região sensível que $Q_{coll} > Q_{crit}$. Caso contrário, nenhum *soft-error* irá ocorrer. No caso de células de memória SRAM e outros circuitos que contenham um laço de realimentação, há um termo adicional que influencia na velocidade com que o circuito pode reagir – velocidades de funcionamento mais baixas dão ao circuito mais tempo para que ele consiga recuperar o valor no nó corrompido, o que vem a contribuir para reduzir a probabilidade de uma falha

transiente. Este termo adicional tende a aumentar o valor efetivo de Q_{crit} . (BAUMANN, 2005).

A taxa de *soft-errors* em um circuito é referenciada por SER (*soft-error rate*) e é geralmente medida em falhas no tempo (FIT – *Failures In Time*). Uma FIT é equivalente a uma falha a cada 10^9 horas de funcionamento do dispositivo. Essas falhas transientes se tornaram um grande problema pois, caso não sejam tratadas, elas produzem uma taxa de falhas tão grande quanto todos os outros mecanismos combinados (BAUMANN, 2005).

Porém, para a análise da suscetibilidade dos circuitos a falhas transientes é necessária uma modelagem matemática do pulso transiente resultante. Para tanto, Messenger (1982) propôs uma modelagem baseada na disposição de carga na saída da porta lógica com uma fonte de corrente. Segundo Messenger, o mecanismo de modelagem do pulso transiente comporta-se de acordo com a dupla exponencial mostrada na equação (1) a seguir:

$$I(t) = I_0 \left(e^{-(t/t_\alpha)} - e^{-(t/t_\beta)} \right) \quad (1)$$

Na equação (1), I_0 é corrente máxima, t_α é a constante de tempo da junção e t_β é a constante de tempo para estabelecer o impacto inicial da partícula. A Fig. 6 mostra as formas de onda típicas para uma tecnologia estado-da-arte (100nm), considerando diversos valores de I_0 e t_β .

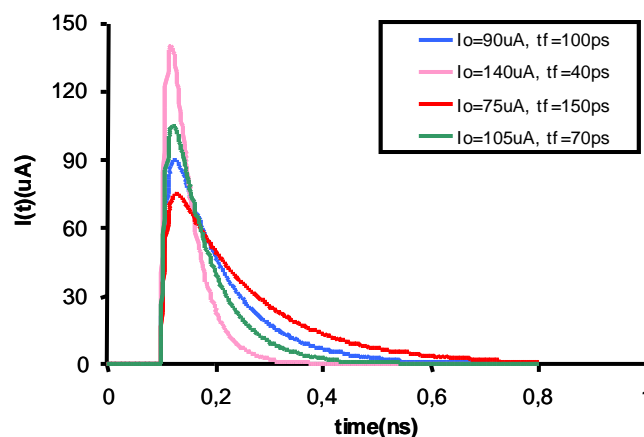


Figura 6 – Formas de onda típicas

A duração e amplitude desta corrente dependem de diversos fatores, tais como energia da partícula, dimensões da área ativa afetada e perfil da distribuição

dos dopantes (MESSENGER, 1982). Caso o pulso de corrente tenha duração ou amplitude suficiente para carregar ou descarregar o capacitor da saída do transistor, um pulso de tensão será gerado que, dependendo de sua amplitude, pode ser considerado como uma inversão do sinal lógico da saída da porta. A modelagem do pulso proposta por Messenger (1982) está ilustrada na Fig. 7.

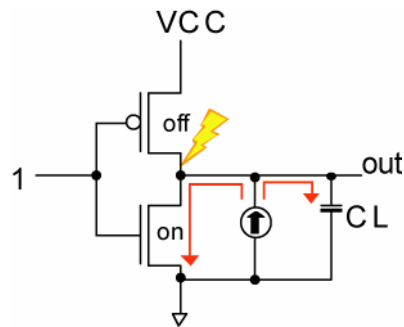


Figura 7 – Modelagem do mecanismo de geração do pulso transiente

Apesar de definir com bastante precisão a forma do pulso transiente oriundo do SET, a modelagem proposta por Messenger é de difícil uso para a análise da propagação de SETs. Por isso, diversos modelos e técnicas têm sido propostos na literatura (ALEXANDRESCU, ANGHEL, NICOLAIDS, 2004; OMAÑA et al, 2003; DAHLGREN, LIDÉN, 1995; WIRTH et al, 2005), conforme mostrados na Fig. 8. A Fig. 8a mostra a modelagem do pulso utilizando a equação (1). Entre as alternativas mais básicas de representação encontram-se a representação trapezoidal (Fig. 8b), triangular (Fig. 8c) e a desconsideração dos tempos de subida e descida do pulso (Fig. 8d).

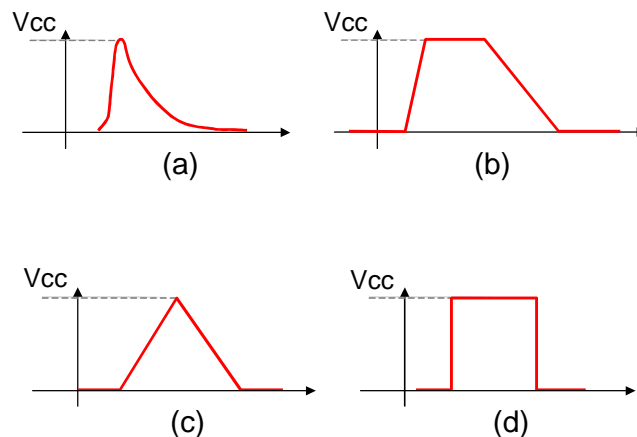


Figura 8 – Modelagens para a análise de propagação de um SET

3 Técnicas Existentes para Análise de Propagação de SETs

A análise da propagação de um SET por um circuito em estágios iniciais de desenvolvimento do projeto possibilita identificar que partes do circuito necessitam ser protegidas contra radiação. Essa análise do circuito antes da sua implementação permite que o circuito seja modificado, a fim de minimizar os efeitos de SETs e SEUs.

Tal análise consiste basicamente em determinar quais partes do circuito são mais sensíveis a SETs. A sensibilidade de uma porta lógica, por exemplo, pode ser estimada determinando-se a porcentagem dos pulsos oriundos por SETs nela ocorridos que consegue atingir alguma saída primária do circuito. Para isso é necessário analisar, de alguma forma, a propagação de um pulso transiente pela lógica do circuito.

Os principais métodos de análise de propagação são baseados em simulação ou em análise topológica.

Métodos baseados em simulação exercitam um modelo do circuito para o conjunto de estímulos fornecidos. Logo, tais métodos exigem a aplicação de vetores de entrada. O funcionamento geral consiste em analisar o circuito por completo (ou apenas a parte afetada) uma vez para cada passo da simulação, ou toda a vez que ocorre uma transição de sinal. Essa última técnica é conhecida por simulação guiada a eventos (*event-driven simulation*).

A simulação do circuito no nível elétrico é uma técnica que gera resultados bastante precisos. Ela pode ser feita através de um simulador de propósito geral, como SPICE (NAGEL, 1975). Tais simuladores representam o circuito como uma rede de elementos passivos e ativos (resistores, capacitores, indutores e fontes

controladas), montando sistemas de equações diferenciais lineares, os quais são resolvidos para cada passo da simulação.

Porém as técnicas baseadas em simulação do circuito exigem uma exploração exaustiva do espaço de entrada, isto é, para analisar a propagação de um pulso originado por um SET ocorrendo em uma porta p qualquer é necessário que todos os vetores de entrada sejam exercidos. Considerando um circuito com n entradas primárias (PIs – *Primary Inputs*) e m portas lógicas, uma análise completa necessita da injeção de $2 \cdot m$ falhas (cada porta poderá originar um pulso de subida e um pulso de descida). Cada falha deve ser simulada para todas as 2^n combinações de valores nas PIs (ou seja, para todos os 2^n vetores de entrada). Por exemplo, para uma análise completa da propagação do pulso transiente do circuito ilustrado na Fig. 9, que possui 3 entradas primárias e 13 portas lógicas, são necessárias $2^3 \cdot 13 \cdot 2 = 208$ simulações do circuito, resultado em um significativo tempo de execução para essa análise.

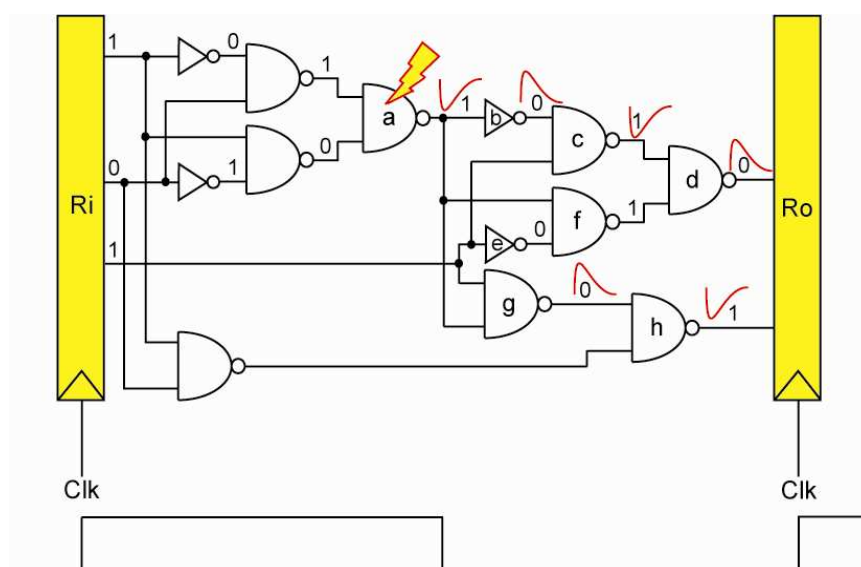


Figura 9 – Propagação de um pulso transiente pela lógica combinacional

Os métodos baseados em simulação exigem longos tempos de execução, pois necessitam dessa exploração exaustiva. Além disso, nesses métodos é necessário também que o circuito seja exaustivamente simulado sem a injeção de falha, pois essas técnicas determinam a suscetibilidade do circuito a partir da comparação entre o comportamento do circuito com falha e o comportamento do circuito livre de falha.

Por outro lado, os métodos baseados em análise topológica codificam o estado de cada linha do circuito para a falha, indicando se há ou não um pulso transiente. Essa característica facilita enormemente a análise dos resultados com o intuito de determinar os pontos mais sensíveis do circuito.

Tais métodos são bastante semelhantes à análise de *timing* topológica (TTA – *Topological Timing Analysis*) (GÜNTZEL, 2000), pois realizam travessia do circuito, enquanto processam as informações necessárias. A técnica de TTA foi incorporada a este trabalho como forma de pré-processamento do circuito e será explicada com maiores detalhes no próximo capítulo.

A seguir são descritos alguns métodos estado-da-arte para análise da propagação de um SET em lógica combinacional.

3.1 Método Baseado em Simulação no Nível de Chaves

Dahlgren e Lindén (1995) propuseram um método para a análise do pulso transiente em circuitos combinacionais baseado em simulação no nível de chaves. Esse método proposto apresenta uma modelagem para diversas grandezas, tais como as capacitâncias internas das portas e resistências dos caminhos ativados. Para tanto, o método define que uma porta lógica CMOS estática (simples ou complexa) pode ser representada por uma simples rede chamada de CB (*Channel-connected Block*).

Uma CB consiste de um conjunto de transistores e/ou resistores conectados, onde todas as entradas dos *gates* dos transistores são consideradas entradas da CB. Todos os nodos da CB que não são *gates* de transistores e estão conectados a outras CBs são considerados saídas da CB.

O processo de propagação de um pulso transiente é definido em dois passos distintos:

- 1) A modelagem do transiente e sua propagação dentro de uma CB;
- 2) A modelagem da propagação do transiente por diferentes CBs (propagação pelo circuito).

Para o primeiro passo, a partir de qualquer ponto de injeção de falha de uma CB, o pulso transiente deve atingir a saída da CB. A duração do pulso transiente

resultante da injeção de falha é, então, calculada. A Fig. 10 ilustra a propagação do pulso transiente pelo transistor M1. Como pode ser visto na tabela da Fig. 10, a duração do pulso transiente é fortemente dependente da topologia dos transistores na CB. Além disso, o pulso transiente pode não atingir a saída da CB devido às suas características elétricas. Os autores referenciam esse fenômeno por **mascamamento elétrico local**.

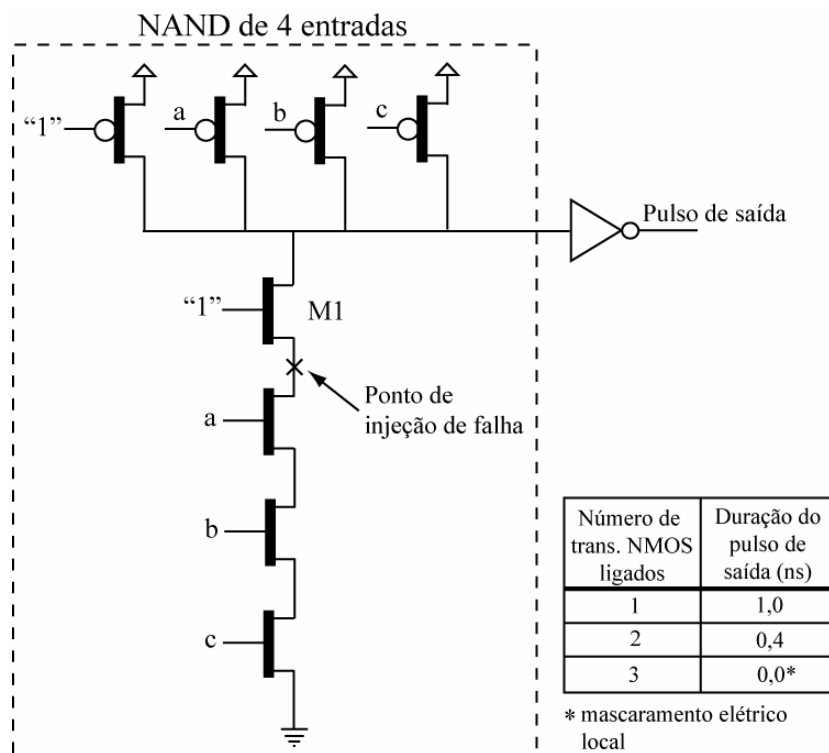


Figura 10 – Propagação do pulso transiente dentro da CB.

Fonte: DAHLGREN; LINDÉN, 1995, p. 210.

Após a modelagem do pulso transiente na saída de uma CB, o próximo passo do método proposto é a modelagem da propagação do pulso por outras CBs do circuito. Nesse caso o pulso atinge pelo menos o *gate* de um transistor NMOS e o *gate* de um PMOS. Devido à complementaridade das redes de transistores N e P, o pulso transiente irá ligar temporariamente um tipo de transistor e desligar o outro tipo. Então, se não houver mascaramento lógico na porta lógica, o pulso transiente irá fechar ou abrir o circuito da rede N (ou P).

A Fig. 11 mostra os resultados de simulações elétricas de uma porta NAND de 2 entradas. Tais resultados levam a uma classificação dos pulsos transientes de acordo com a relação entre a duração do pulso na saída da CB e um valor de limiar

(t_{PD}). Pulsos transientes com duração menor que o limiar são chamados de fracos. Tais pulsos se tiverem amplitude completa (V_{DD}), podem ser atenuados ou até mesmo filtrados pela porta. No caso do pulso ser filtrado, esse fenômeno é chamado de **mascaramento elétrico global**. Entretanto, pulsos com amplitude não completa e, portanto com longos tempos de subida e descida, podem ser amplificados pela CB e, conseqüentemente, aumentando suas durações.

De forma oposta, os pulsos transientes com grande duração são chamados de fortes, não são afetados durante sua propagação através do circuito. Como pode ser visto na Fig. 11, a duração de um pulso transiente forte na saída de uma CB é igual à duração do pulso transiente na entrada. Porém, devido às assimetrias dos tempos de subida e descida das portas lógicas, a duração do pulso transiente poderá aumentar.

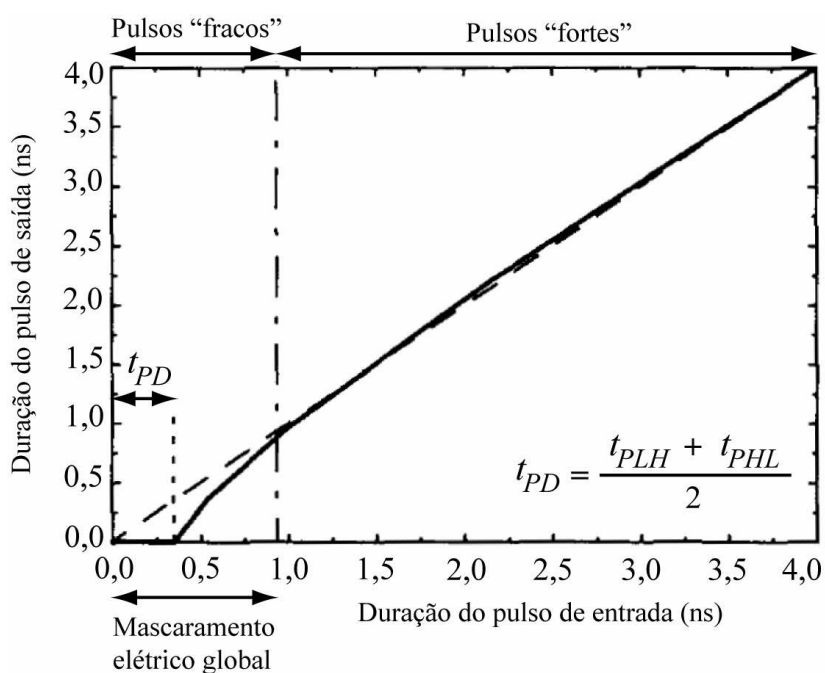


Figura 11 – A relação entre a largura do pulso transiente de entrada e o de saída de uma CB.

Fonte: DAHLGREN; LINDÉN, 1995, p. 211.

O comportamento da propagação de um pulso transiente dentro de uma CB e através de CBs conectados foi analisado a partir de simulações no nível elétrico. Os principais parâmetros para determinar se o pulso será mascarado ou propagado através do circuito são as capacitâncias de carga e a resistência das linhas dos caminhos conectados aos nodos destino, juntamente com a forma do pulso de

corrente introduzido. A duração do pulso transiente é estimada baseada em um modelo de rede RC linear similar ao proposto no trabalho de Cha et al. (1996).

Para realizar as simulações foi utilizado um simulador no nível de chaves proposto pelos mesmos autores em um trabalho anterior (DAHLGREN; LINDÉN, 1993). O algoritmo implementado por esse simulador é baseado em um modelo de avaliação local dos nodos, considerando a resistência equivalente de diversos resistores em série. Assim, cada transistor pode ter uma resistência arbitrária baseada na razão entre a sua largura e o seu comprimento. Embora a resistência de um transistor seja dependente da tensão de alimentação, a resistência constante na região linear é utilizada. O simulador também permite assinalar capacitâncias (lineares) aos diversos nodos do circuito, incluindo os drenos e fontes dos transistores.

A partir da injeção de mudanças temporárias do estado do nodo conectado à região sensível do transistor, o simulador consegue traçar os caminhos de propagação do pulso transiente.

A precisão do modelo proposto é avaliada por meio de comparação com simulações elétricas. Foram utilizados dois circuitos-teste próprios, um com 140 transistores e outro com 640 transistores. O método proposto demonstrou ser bastante preciso para esses exemplos, em comparação com simulação elétrica, sendo que os resultados ficaram dentro de uma margem de 10% de diferença em relação as simulação em SPICE. Segundo Dahlgren e Lindén (1995), a análise com o método proposto foi, em média, 1000 vezes mais rápida do que a análise baseada em simulação elétrica.

3.2 Método Baseado em Simulação no Nível Lógico

Alexandrescu, Anghel e Nicolaidis (2004) propuseram um método para análise da propagação do pulso transiente pela lógica combinacional baseado em simulação no nível lógico. Tal método avalia a probabilidade de um pulso transiente ser armazenado por um registrador de saída do circuito.

O pulso transiente originado pela ocorrência de um SET é caracterizado utilizando a dupla exponencial de Messenger (equação (1) descrita na seção 2.2) e modelado com um pulso quadrado (Fig. 8d da seção 2.2). Para esse método, a

forma exata do pulso não é importante, pois a avaliação é feita de forma estática, baseada no comportamento do circuito diante de cada falha transiente possível. Na realidade, as características do pulso mais relevantes para esse método são o tempo em que o evento ocorre dentro do ciclo de relógio e a duração do pulso transiente. Tais valores foram obtidos a partir de simulações elétricas (SPICE (NAGEL, 1975)).

O pulso transiente da saída da porta foi obtido a partir do **método de injeção de falha**. A injeção de falha consiste em simular o resultado da ocorrência de uma falha em uma porta lógica através de técnicas, tais como forçar um sinal a um específico valor. No caso de falhas transientes causadas por SETs, a injeção de falha consiste em forçar o sinal da saída da porta que sofreu a colisão da partícula para o valor oposto por um intervalo de tempo determinado pela duração do pulso transiente. Para tanto, é necessária a utilização de um simulador que permita que o usuário tenha acesso a dados internos do circuito a fim de poder forçar uma linha do circuito a um determinado valor lógico. Para isso o circuito é representado a nível de portas e é utilizado um simulador lógico guiado a eventos.

A Fig. 12 mostra o algoritmo clássico para a análise completa de um circuito utilizando o método de injeção de falha clássico. Um circuito referencial é utilizado para comparar os valores dos sinais e para restaurar o estado correto do circuito após a injeção da falha. Como pode ser visto na Fig. 12, o algoritmo clássico de injeção de falhas demanda uma simulação completa do circuito para cada falha, resultando em um tempo de simulação proibitivamente longo mesmo para circuitos de complexidade¹ moderada.

```

1  para cada falha faça
2    para cada vetor de entrada faça
3      aplicar o novo vetor de entrada e a falha;
4      analisar a falha;
5    fim_para
6  fim_para

```

Figura 12 – Algoritmo clássico de simulação de falha

Assim, usando-se o algoritmo clássico de injeção de falhas, a caracterização completa de um circuito sob a ocorrência de falhas transientes implica na injeção de

¹ Neste caso, entenda-se por complexidade a combinação de número de portas lógicas com número de entradas primárias.

falhas em todos os nodos do circuito para várias condições (momento da colisão e duração do pulso), exigindo a exploração completa do espaço de entrada mediante a aplicação de todos os 2^n vetores de entrada (com n sendo o número de entradas do circuito).

A fim de reduzir esse tempo de execução proibitivo, Alexandrescu, Anghel e Nicolaidis (2004) propõem:

- Um procedimento de particionamento o circuito e identificação os elementos de memória;
- Modificações no algoritmo clássico de injeção de falhas.

O objetivo desse procedimento é separar os blocos combinacionais dos registradores, a fim de proceder a injeção de falhas somente nos primeiros. A lógica combinacional é considerada conectada ou à saída do circuito ou à entrada de algum elemento de memória. Além disso, os elementos de memória são monitorados a fim de analisar se o pulso transiente atinge as suas entradas.

O algoritmo de injeção de falhas modificado é aplicado à lógica combinacional. Uma das modificações no algoritmo de injeção de falhas consiste em aplicar um vetor de entrada e a seguir, injetar de maneira seqüencial todas as falhas que devem ser analisadas (ver Fig. 13).

Além disso, o método analisa se um pulso transiente consegue atingir o registrador da saída do bloco combinacional no mesmo instante que o momento de ativação do elemento de memória, isto é, sua **janela de amostragem**. A janela de amostragem de um registrador é detalhadamente definida no próximo capítulo, mas por agora, é suficiente entender que se um pulso transiente atingir o registrador fora da sua janela de amostragem, o pulso não será capturado. Assim, alguns pulsos transientes não precisam ser considerados na simulação. O método proposto por Alexandrescu, Anghel e Nicolaidis explora tal fato para reduzir o esforço computacional na análise.

Quando o pulso atinge a entrada de um elemento de memória da saída primária, existe uma probabilidade do pulso ser armazenado provocando um *soft-error*. O momento da colisão da partícula, a duração do pulso transiente e os atrasos das portas lógicas do circuito são os fatores responsáveis para que isso ocorra. Portanto, é necessário ter informações de *timing* precisas sobre o circuito a ser testado.

As informações de atraso das portas são obtidas a partir de um arquivo SDF (*Standard Delay Format*), contendo três tipos de atrasos de cada porta: pior, nominal e o melhor, a fim de considerar questões de variabilidade nos processos de fabricação e ambientes de operação.

O algoritmo proposto nesse método está ilustrado na Fig. 13. Como pode ser visto, não é necessário avaliar um novo vetor de entrada para cada falha. O pulso transiente será simulado apenas dentro dos blocos combinacionais pois os registradores são desligados durante a injeção de falhas (Fig. 13). Após a simulação de todas as falhas, os elementos de memórias são reativados e a simulação prossegue com a aplicação de um novo vetor de entrada.

Como os elementos de memória estão desligados durante a injeção de falhas, é necessário monitorar as condições das entradas dos registradores a fim de analisar a possibilidade de um pulso transiente ser capturado pelo registrador. Para isso, se for detectada atividade na entrada do registrador, o pulso propaga-se pelo registrador e são extraídas informações a respeito do formato do pulso resultante.

Além disso, o método proposto leva em consideração pulsos com diferentes durações. Para tanto, em uma mesma rede são injetados diversos pulsos transientes com duração variando de poucas dezenas até centenas de picosegundos. Como o simulador utilizado é guiado a eventos, então as transições de subida e descida do pulso são propagadas separadamente e re combinadas na saída do circuito.

```

1 //fase de setup
2   particionar o circuito;
3   identificar os elementos de memória;
4   monitorar as entradas dos elementos de memória;
5 //fase de simulação
6   para cada vetor de entrada faça
7     aplicar o novo vetor de entrada;
8     desligar os elementos de memória;
9     para cada falha faça
10      aplicar a falha;
11      analisar a falha;
12     fim_para
13     ativar os elementos de memória;
14   fim_para

```

Figura 13 – Algoritmo proposto

A sensibilidade do circuito a uma falha transiente injetada em uma rede qualquer é medida em porcentagem. Se a falha não for armazenada por um elemento de memória, a falha é desconsiderada, não resultando em impacto no circuito. A sensibilidade do circuito nesse caso é de 0%.

Se a falha for armazenada, então a sensibilidade do circuito sob teste a essa falha é maior que 0%. A sensibilidade pode ser de 100% quando a falha é claramente observável na saída do elemento de memória. Por exemplo, quando o valor armazenado é 0, porém o valor correspondente correto é 1. Se o valor da saída do elemento de memória for indeterminado, a sensibilidade será menor que 100%.

Como já foi dito, o método foi implementado usando como base um simulador no nível lógico guiado a eventos. Além dos passos descritos detalhadamente nessa seção, a implementação do método possui um módulo de pós-processamento capaz de extrair informações estatísticas que conduzem à sensibilidade a SETs de cada porta do circuito. Foram gerados resultados para dois circuitos-exemplo, sendo um deles um somador *carry lookahead* de 8 bits com 98 portas lógicas e 9 registradores e o outro, o núcleo de um processador *Sparc* com 22.632 portas lógicas e 2.104 elementos de memória, sendo a maioria *flip-flops*. Os resultados em tempo de execução e precisão foram comparados com aqueles obtidos por meio do algoritmo clássico de simulação de falhas, usando o mesmo simulador no nível lógico e mesma tecnologia CMOS. A aceleração obtida foi de até 20 vezes, para 70 a 90 falhas injetadas (ALEXANDRESCU; ANGHEL; NICOLAIDIS, 2004).

3.3 Método Baseado em BDDs

A análise da suscetibilidade de um circuito a falhas transientes pode ser bastante eficiente através da representação de um pulso transiente através de um diagrama de decisão binária (BDD – *Binary Decision Diagram*) (ZHANG; WANG; ORSHANSKY, 2006).

BDDs foram propostos por Bryant (1986) para representar o resultado da aplicação de uma função booleana. Um BDD de uma função booleana é um grafo

acíclico direcionado e enraizado, com vértices terminais e não-terminais. Cada vértice não-terminal do BDD representa uma variável da função booleana e dele saem duas arestas, uma com o valor zero e outra com o valor um, para dois vértices. Assim, para um dado vetor de entradas de uma função booleana, o valor da função é determinado percorrendo-se o BDD desde a raiz até um dos nodos terminais. O caminho percorrido depende dos valores das variáveis da função.

A utilização dessas estruturas no apoio à análise da suscetibilidade de circuitos a falhas transientes foi proposta por Zhang, Wang e Orshansky (2006). A precisão desse método é garantida através da descrição precisa das características das portas usando uma pré-caracterização baseada em simulações SPICE de uma biblioteca de células. Além disso, o método também considera o mascaramento lógico de um pulso transiente.

Primeiramente a geração do pulso é modelada de acordo com a estrutura da rede de transistores da porta afetada pelo evento. Assim, o objetivo de caracterizar uma biblioteca de células é estimar para cada célula da biblioteca a forma do pulso transiente resultante na saída da porta devido à colisão da partícula em cada região sensível da célula. A Fig. 14 ilustra a caracterização de uma porta NAND de 2 entradas sob o efeito de um SET em cada nodo sensível da porta. As simulações foram feitas variando o valor da carga da partícula (q) e a capacitância de carga da saída da porta (C_{load}). O pulso de tensão produzido na saída da porta através da colisão de uma carga q específica é dependente dos valores do vetor de entrada da porta. No exemplo da Fig. 14, tanto o nodo n1 quanto o nodo n2 são sensíveis se o vetor de entrada for "10", enquanto que somente o nodo n1 é sensível se o vetor de entrada for "01" ou "00". O pulso de tensão produzido é, então, aproximado para uma forma trapezoidal, onde a sua duração é medida a partir dos tempos onde o pulso atinge metade da tensão de alimentação.

Após a geração do pulso transiente, ele é propagado até as saídas primárias do circuito. Durante essa propagação, as características elétricas do pulso transiente, tais como sua duração e sua magnitude, variam de acordo com as características das portas por onde o pulso propaga-se. Pulsos curtos tendem a ser atenuados, enquanto que os pulsos longos tendem a manter sua duração e amplitude após atravessar a lógica do circuito. Os pulsos podem também propagar-se por caminhos re-convergentes e, assim, mais de um pulso pode atingir as

entradas de uma porta. A interação entre dois pulsos atingindo simultaneamente uma porta lógica também é modelado.

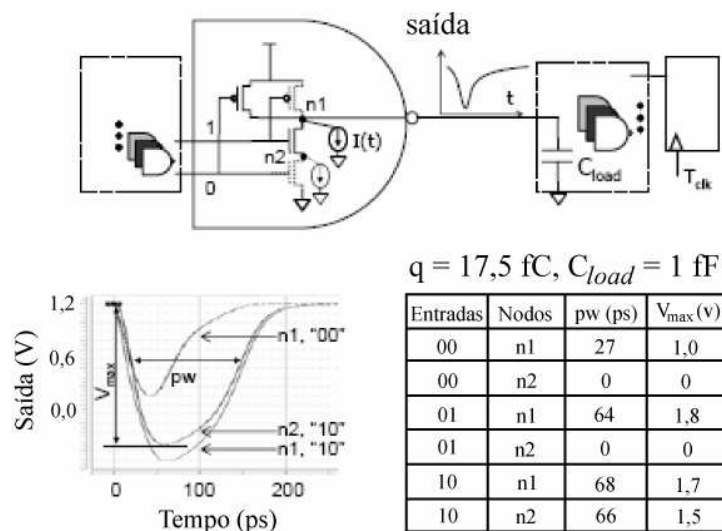


Figura 14 – Caracterização da geração do pulso através da simulação elétrica

Fonte: ZHANG; WANG; ORSHANSKY, 2006, p. 202.

O método proposto faz uma análise estática da propagação do pulso transiente usando BDDs para codificar e propagar o pulso transiente. Através da propagação da falha até uma saída primária, o algoritmo consegue estimar de forma precisa a probabilidade da ocorrência de erro na saída. O BDD que descreve a função booleana de um nodo livre de falhas é chamado de *static* BDD e é estruturado de acordo com o proposto por Bryant (1986). A colisão de uma partícula em um nodo, gerando um pulso transiente, pode ser modelada através da modificação de um *static* BDD. Tal variação na estrutura do BDD é conhecida como *event* BDD. No caso de um *event* BDD, os vértices terminais podem conter valores originais do *static* BDD e pulsos transientes. A Fig. 15 ilustra a modificação no BDD após um SET. No caso mostrado na Fig. 15, o pulso transiente foi modelado para o vetor de entrada "10" de uma porta NAND de 2 entradas.

A construção do *event* BDD da saída de uma operação entre 2 *event* BDDs nas entradas é feita de forma recursiva, da mesma forma que são construídos os *static* BDDs. A diferença está apenas na maneira em que os vértices terminais são processados. A propagação do pulso transiente através da lógica é feita através da construção de *event* BDDs para as portas lógicas do circuito pertencentes ao cone

lógico de propagação do pulso transiente em ordem topológica. Dessa forma, o método proposto pode ser também classificado como um método baseado em análise topológica.

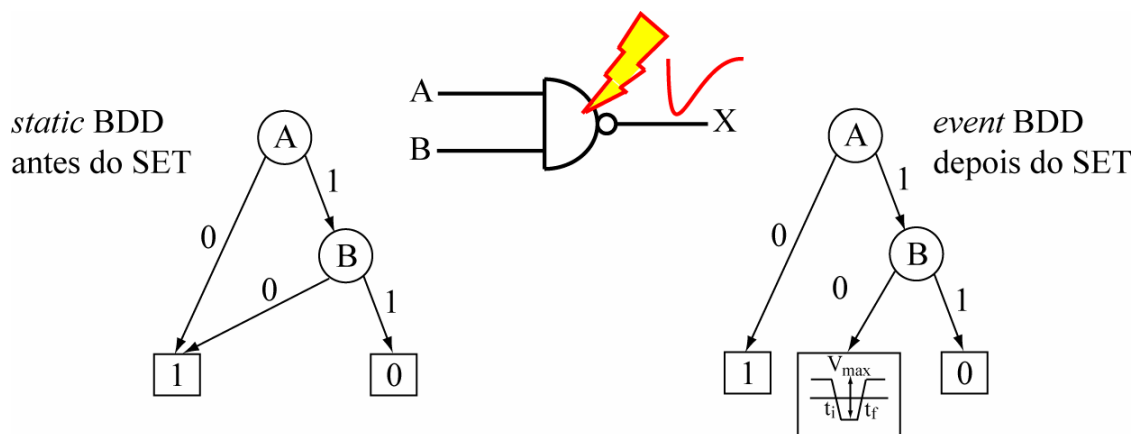


Figura 15 – A modificação do BDD após um SET

Fonte: ZHANG; WANG; ORSHANSKY, 2006, p. 202.

Para ilustrar o método proposto, a Fig. 16 mostra os BDDs resultantes da ocorrência de um SET no nodo M. Dada uma quantidade de carga, um *event BDD* é gerado para cada vetor de entrada. O pulso transiente é mascarado pela lógica quando $B = 1$. O *event BDD* no nodo X é o mesmo do *static BDD*, pois o pulso no nodo X não atinge a tensão de chaveamento da porta devido à reconvergência no cone de propagação do pulso.

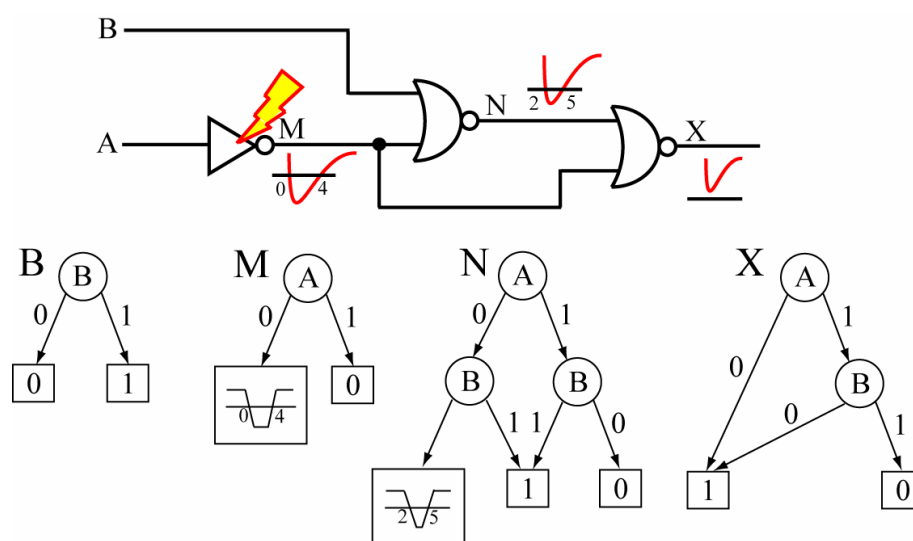


Figura 16 – Propagação do pulso transiente e os BDDs dos nodos do circuito

Fonte: ZHANG; WANG; ORSHANSKY, 2006, p. 202.

Porém, a codificação de funções lógicas utilizando BDDs possui complexidade exponencial, dependente do número de variáveis (BRYANT, 1986). Para realizar uma manipulação eficiente dos BDDs, uma etapa de particionamento do circuito é proposta. Esse particionamento permite uma aceleração significativa sem perda na precisão da estimativa.

O circuito é particionado em pequenos blocos como é mostrado na Fig. 17. Alguns nodos são designados para serem pseudo entradas primárias dos blocos gerados pelo particionamento, delimitando os blocos e a relação entre eles. Para estimar a probabilidade de armazenamento de um pulso transiente originado por um SET, um *event* BDD é gerado na saída da porta afetada e propagado até os nodos dos limites dos blocos. Quando o pulso atinge o nodo que separa dois blocos, o algoritmo passa a analisar o bloco seguinte, onde o *event* BDD é gerado nesse nodo. A probabilidade de armazenamento do pulso transiente é então aproximada a partir da soma das probabilidades do armazenamento dos pulsos transientes resultantes, ponderada pelas probabilidades de ocorrência de tais pulsos.

O tamanho dos blocos resultantes do particionamento são definidos de acordo com o máximo número de entradas primárias e pseudo entradas primárias de cada bloco. Um particionamento em grandes partes permite uma maior precisão nos resultados ao custo de um rápido aumento no seu tempo de execução. Essa relação entre precisão e velocidade de execução é tratada a partir do ajuste do tamanho das partições.

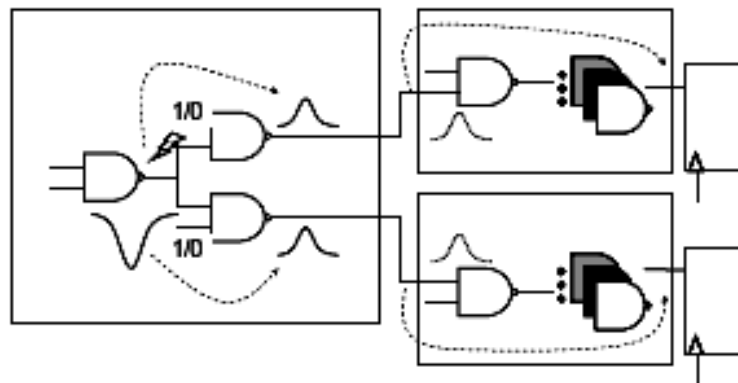


Figura 17 – Particionamento do circuito

Fonte: ZHANG; WANG; ORSHANSKY, 2006, p. 202.

Os resultados mostram que o método proposto atinge uma precisão tão boa quanto o método baseado em simulação SPICE. O erro médio dos circuitos simulados é de 12% e o método apresentou uma aceleração de 90.000 vezes. O método pode ainda gerar resultados mais exatos através de uma precisão maior na caracterização das células da biblioteca.

3.4 Método Baseado em Propagação Topológica Booleano – Temporal de SETs

O método TBT-SET (*Timed-Boolean Topological SET Propagation*), proposto por Neves (NEVES, 2006; NEVES et al., 2006) realiza uma análise automática da propagação de SETs em blocos combinacionais com o uso de cálculo lógico-temporal. Tal método parte de três assertivas básicas:

- No momento que ocorre um SET em uma porta lógica, o bloco combinacional a ser analisado encontra-se estável sob um dado vetor de entrada;
- Apenas uma partícula carregada atinge o circuito em alguma região sensível;
- O ciclo de relógio é suficientemente longo para permitir que um SET ocorrendo em qualquer uma das portas lógicas do circuito consiga propagar-se pela lógica até alguma saída primária.

A partir da estabilidade dos valores das saídas do circuito após a aplicação de um vetor de entrada v quando o SET ocorre em uma porta lógica p qualquer do circuito, o método TBT-SET realiza uma caracterização do circuito sob a ação do SET. Tal situação se inicia com a ocorrência do SET e termina quando o último pulso transiente atinge alguma saída primária. Em outras palavras, a propagação de todos os pulsos transientes oriundos do SET necessita ser determinada.

Outro fator importante decorrente dessa modelagem é que, caso ocorra um pulso transiente em uma linha l do circuito, então o pulso necessariamente será do tipo $l_v \rightarrow l'_v \rightarrow l_v$, onde l_v é o valor lógico de l sob o vetor de entrada v e l'_v é o valor lógico oposto de l_v . Esta propriedade resultante das assertivas feitas pelo método é a chave para a digitalização do fenômeno original, que é eminentemente analógico.

Além disso, os pulsos gerados a partir de um SET são modelados de forma que os tempos de início (**ti**) e de fim (**tf**) do pulso transiente correspondem aos pontos em que a dupla exponencial de Messenger (equação (1)) passa por $V_{DD}/2$. Tal modelagem é idêntica à modelagem mostrada na Fig. 8d (ver seção 2.2).

Para a análise da propagação de um SET ocorrido na saída de uma porta p , de um circuito, o método TBT-SET considera o instante de início do pulso como sendo o instante inicial do ciclo de relógio. A fase de pré-processamento do método inicia marcando as portas lógicas pertencentes ao cone lógico que começa na saída de p , a fim de restringir o processamento apenas para as portas que sofrem influência do SET em questão. Ainda na etapa de pré-processamento, os valores do vetor de entrada v são propagados pelo circuito até a estabilização dos valores das saídas primárias do circuito.

Em seguida, em todas as linhas de saída de p são assinalados pulsos transientes com **ti** = 0 e **tf** = **d**, onde **d** é a duração do pulso transiente na saída da porta onde ocorreu o SET. Após, o pulso inicial é propagado seguindo a ordem topológica do circuito, ou seja, uma porta somente será analisada quando todas as suas antecessoras (que lhe servem de *fanin*) tiverem sido analisadas. Apesar de se tratar de um método de análise de um único pulso transiente, a existência de caminhos reconvergentes poderá fazer com que mais de um pulso transiente propague-se concomitantemente pela lógica, conforme ilustrado na Fig. 18.

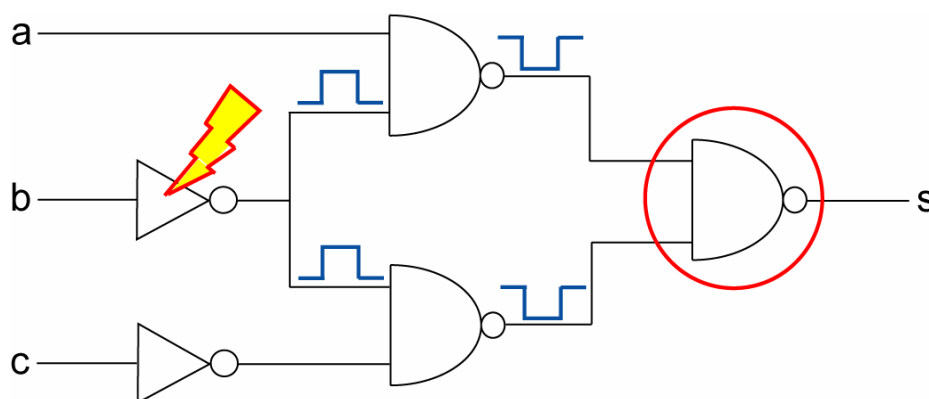


Figura 18 – Exemplo de circuito com reconvergência

A fim de considerar a possibilidade de mais de um pulso atingir as entradas de uma porta lógica, o TBT-SET usa um conjunto de regras lógico-temporais para computar a forma de onda resultante na saída da porta. Esse conjunto baseia-se nos conceitos de controlabilidade das portas lógicas (GOLDSTEIN, 1979;

ABRAMOVICI; BREUER; FRIEDMAN, 1990; BUSHNELL; AGRAWALL, 2000). Tal conjunto de regras, é necessário para a determinação da **forma de onda equivalente de entrada** para uma porta lógica p . Para tanto são consideradas três situações que podem ocorrer:

- 1) A porta p possui pelo menos uma entrada com o valor inicial controlante e livre de pulso transiente. Todas as entradas nessa situação definirão o valor da saída da porta, sem a influência das demais entradas. Nesse caso, ocorre um mascaramento lógico, e portanto, não há propagação do pulso transiente através da porta p . Essa situação está ilustrada na Fig. 19.
- 2) A porta p possui pelo menos uma entrada com valor inicial controlante, sendo que todas as entradas nessa situação apresentam em algum momento posterior um pulso transiente (Fig. 20). Durante o período em que todas as entradas que iniciaram com valor controlante estiverem com pulsos, seus valores serão não-controlantes, e portanto, nenhuma dessas entradas irá sozinho determinar o valor lógico da saída de p . A forma de onda equivalente é então definida como sendo a interseção entre os intervalos de duração dos pulsos existentes nas entradas que iniciaram com valor controlante, pois em qualquer tempo fora dessa interseção uma entrada com valor controlante irá definir sozinho o valor na saída da porta.
- 3) Nessa situação, ilustrada na Fig. 21, nenhuma entrada da porta p possui valor inicial controlante; a forma de onda equivalente de entrada é definida como uma união entre os intervalos de duração dos pulsos nas entradas de p , pois todos os pulsos possuem valor controlante, e assim, todos podem alterar o valor lógico da saída de p .

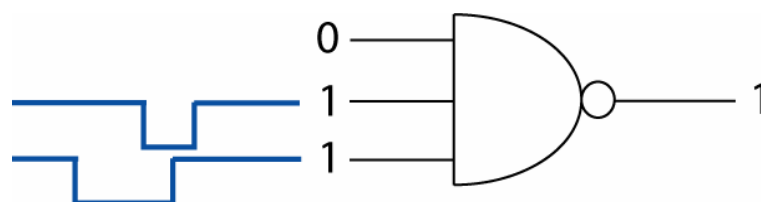


Figura 19 – NAND com entrada em valor inicial controlante livre de falha

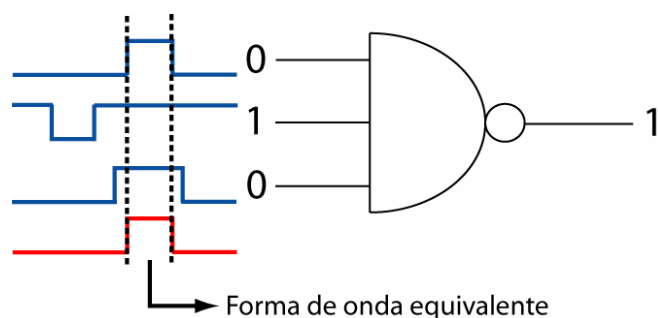


Figura 20 – NAND com entrada em valor inicial controlante com SET

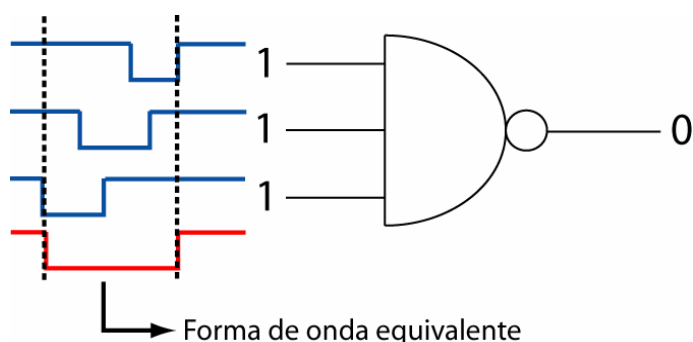


Figura 21 – NAND sem entrada com valor inicial controlante

O método TBT-SET realiza a análise da propagação de um pulso por uma porta lógica em duas etapas. A primeira etapa determina a forma de onda equivalente de entrada utilizando o conjunto de regras lógico-temporais a fim de possibilitar a consideração **dinâmica** de mais de um pulso transiente ocorrendo nas entradas da porta. Nessa etapa apenas a duração e a polaridade dos pulsos são considerados.

Na modelagem proposta pelo método TBT-SET, a forma de onda equivalente da entrada apresenta somente um pulso. Isso significa que conjuntos de pulsos em seqüência são aproximados por um pulso longo que contém todos os pulsos.

Uma vez determinada a forma de onda equivalente de entrada, a forma de onda da saída é obtida por uma função de propagação, a qual leva em conta a polaridade da porta, isto é, se é inversora ou não, e seus atrasos de propagação, segundo o modelo de par de atrasos (td_{HL} – *topological delay high-low* e td_{LH} – *topological delay low-high*). Assim, os tempos de início e fim ($t_{i_{out}}$ e $t_{f_{out}}$) de um pulso $0 \rightarrow 1 \rightarrow 0$, na saída de uma porta são calculados pelas duas equações a seguir:

$$t_{i_{out}} = t_i + td_{LH} \quad (2)$$

$$t_{f_{out}} = t_f + td_{HL} \quad (3)$$

Onde **ti** e **tf** são os tempos de início e de fim do pulso correspondente à forma de onda equivalente de entrada. Analogamente, os tempos de início e fim (**ti_{out}** e **tf_{out}**) de um pulso 1→0→1, na saída de uma porta são calculados pelas duas equações abaixo:

$$t_{i_{out}} = t_i + td_{HL} \quad (4)$$

$$t_{f_{out}} = t_f + td_{LH} \quad (5)$$

Conforme as equações 2, 3, 4 e 5, os atrasos da porta lógica sob análise serão responsáveis pela caracterização da duração do pulso transiente presente na sua saída (**d_{out}**), podendo essa duração ser maior ou menor que a duração do pulso da forma de onda equivalente da entrada e, em alguns casos, resultando em sua filtragem (mascaramento elétrico). A tab. 1 apresenta uma análise da relação entre o atraso de uma porta lógica e a duração de um pulso transiente em sua entrada, com a duração do pulso transiente em sua saída.

Tabela 1 – Relação entre os atrasos de uma porta lógica e o intervalo de duração da forma de onda equivalente, na determinação do pulso de saída

d_{out}	Atrasos	Relação atrasos x d	Conseqüência	Comentário
1→0→1 d - (td_{LH} - td_{HL})	td _{LH} - td _{HL} > 0	td _{LH} - td _{HL} ≥ d	d_{out} = 0	Pulso não propagado
		td _{LH} - td _{HL} < d	d_{out} < d	Pulso atenuado
	td _{LH} - td _{HL} = 0	--	d_{out} = d	Pulso propagado sem variação de duração
	td _{LH} - td _{HL} < 0	--	d_{out} > d	Pulso ampliado
0→1→0 d - (td_{HL} - td_{LH})	td _{HL} - td _{LH} > 0	td _{HL} - td _{LH} ≥ d	d_{out} = 0	Pulso não propagado
		td _{HL} - td _{LH} < d	d_{out} < d	Pulso atenuado
	td _{HL} - td _{LH} = 0	--	d_{out} = d	Pulso propagado sem variação de duração
	td _{HL} - td _{LH} < 0	--	d_{out} > d	Pulso ampliado

O método TBT-SET foi implementado em um protótipo de ferramenta (batizado ASPA – *Automatic SET Propagation Analyzer*) para permitir a geração de resultados práticos (NEVES et al., 2006). Esse protótipo tem como entradas o arquivo com a descrição do bloco combinacional a ser analisado, no formato SPICE e um arquivo contendo os atrasos das portas lógicas, sendo que para cada porta lógica deve haver um par de atrasos (td_{LH}, td_{HL}). O bloco combinacional é

representado por um grafo acíclico direcionado (DAG – *Direct Acyclic Graph*), no qual os nodos representam portas lógicas e as arestas representam as conexões. Os valores de atraso de cada porta lógica (td_{LH} , td_{HL}) são associados aos correspondentes nodos do circuito. Além disso, são adicionados nodos bobos (*dummy nodes*) com valores de atraso zero para representar as entradas e saídas primárias do circuito. A fim de facilitar a implementação do algoritmo de travessia do grafo em ambas as direções, são criados dois nodos – um nodo fonte (*source*) e um nodo terminal, para polarizar o grafo.

Os resultados apresentados por Neves (2006a) indicam que a precisão do método TBT-SET está fortemente relacionada com a precisão das estimativas de atraso das portas. Por outro lado, existem situações em que o método subestima a duração do pulso transiente na saída da porta, devido a possíveis discrepâncias na caracterização dos atrasos das portas. Entretanto, segundo Neves (2006a) o método não falhou em nenhum caso, tendo acertado os casos em que houve propagação do pulso até uma saída primária do circuito e os casos em que houve mascaramento lógico.

3.5 Da Necessidade de Novos Métodos para a Análise da Propagação do Pulso Transiente

Os métodos propostos para a análise da propagação do pulso transiente oriundo da ocorrência de SETs citados nas seções 3.1 a 3.4 nesse capítulo possuem algumas vantagens e desvantagens que devem ser comentadas com maior ênfase a fim de se compreender a necessidade da busca por novos métodos para essa análise.

Os métodos baseados em simulação citados acima (DAHLGREN; LINDÉN, 1995; ALEXANDRESCU; ANGHEL; NICOLAIDIS, 2004), assim como todos os métodos baseados em simulação, necessitam de uma exploração exaustiva do espaço de entrada, demandando um tempo de execução que aumenta de forma exponencial com o aumento do número de entradas do circuito.

As modificações no algoritmo clássico de simulação de falhas, proposta por Alexandrescu, Anghel e Nicolaidis (2004), tornaram o método bem mais rápido que uma simples simulação de falhas. Porém, o método ainda assim continua

dependente do vetor de entrada, exigindo a exploração exaustiva do espaço de entrada para a análise completa do circuito sob análise.

Zhang, Wang e Orshansky (2006) propuseram o método de análise da propagação do pulso transiente baseada em BDDs buscando uma análise mais rápida que os métodos baseadas em simulação, a um pequeno custo na precisão dos resultados. Porém, o tamanho dos BDDs utilizados é dependente do número de entradas do circuito. Ademais, o número de BDDs depende do número de conexões que o circuito possui.

A fim de controlar o tamanho dos BDDs, Zhang, Wang e Orshansky (2006) propuseram uma etapa de particionamento do circuito. Um grande particionamento, isto é, um grande número de pequenas partições leva a um tempo de execução menor, porém aumentando a perda na precisão dos resultados do método. Mesmo com o particionamento, os resultados apresentados pelos autores mostraram que o método exige um grande esforço computacional (alguns circuitos com BDDs em ordem de dezenas de milhares de nodos).

Os métodos baseados em propagação topológica foram propostos para eliminar a execução da simulação completa do circuito para se obter o seu comportamento correto. Esse “comportamento-de-ouro” é necessário para comparar com o comportamento obtido por meio da injeção de falha, a fim de se determinar se a falha gerou um *soft-error* ou não. Além disso, métodos baseados em propagação topológica tendem a ser mais rápidos do que os métodos baseados em simulação, pois utilizam modelos físicos e computacionais consideravelmente mais simples. Por outro lado, estes métodos não descartam a necessidade de se aplicar todos os 2^n possíveis vetores de entrada, resultando em uma aceleração apenas moderada, em comparação aos métodos baseados em simulação guiada a eventos.

4 Análise de *Timing* e a Propagação de Pulsos

No caso de um pulso transiente gerado por um SET não ser mascarado eletricamente, ele pode propagar-se pela lógica combinacional do circuito até uma das saídas primárias, podendo eventualmente ser armazenado por um elemento de memória (BAZE; BUCHNER, 1997).

Como visto no capítulo anterior, as técnicas existentes para a estimativa da taxa de falhas transientes (*soft-errors*) geradas por SETs são baseadas ou em simulação ou em análise topológica do circuito. Porém, técnicas baseadas em simulação demandam um tempo de execução que, para grandes circuitos, torna a análise impraticável.

Tendo em vista esse grande tempo de execução, é necessário realizar etapas de pré-processamento do circuito, a fim de reduzir o número de falhas a serem simuladas (ALEXANDRESCU; ANGHEL; NICOLAIDIS, 2004).

Assim sendo, propõe-se a execução de uma etapa de pré-processamento do circuito baseado nas informações de *timing* relacionadas à propagação de um pulso transiente pela lógica combinacional, a fim de encontrar os SETs mascarados por *latching-window* (ou seja, os SETs que chegam às saídas primárias fora da janela de amostragem do registrador de saída). A seção 4.1 apresenta os aspectos de *timing* da propagação do pulso transiente, enquanto que a seção 4.2 explica detalhadamente o pré-processamento proposto.

4.1 Aspectos de *Timing* Relacionados à Propagação do Pulso

Como foi mostrado no capítulo 2, após a colisão da partícula carregada com o circuito, um pulso de tensão pode ser gerado, podendo ser interpretado como uma mudança de sinal lógico no bloco combinacional. Este pulso pode ser modelado de

forma a simplificar a análise da sua propagação. Neste trabalho é proposta uma modelagem baseada na comparação do nível de tensão do pulso com a metade do valor da tensão de alimentação ($V_{DD}/2$) do circuito. Os tempos de subida e descida são desconsiderados e a duração d do pulso retangular, é determinada pelo tempo no qual a amplitude do pulso excede $V_{DD}/2$, como mostrado na Fig. 22.

Assim, qualquer pulso de uma determinada forma é equivalente a um pulso quadrado com duração igual a d (ALEXANDRESCU; ANGHEL; NICHOLAIDIS, 2004). Segundo Alexandrescu (2004) essa modelagem do pulso transiente não compromete na qualidade dos resultados obtidos pela simulação. Além disso, essa modelagem simplifica o processo de injeção de falhas e permite a simulação da falha sem necessidade de alteração da estrutura do circuito a ser testado.

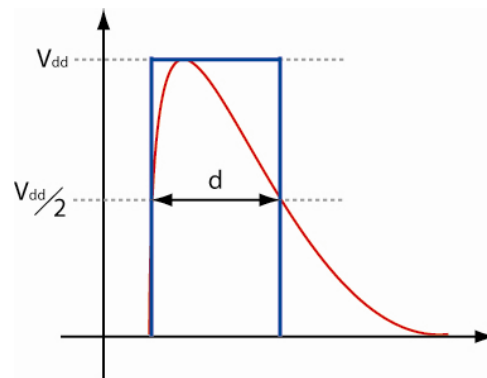


Figura 22 – Modelagem do pulso transiente

A seguir são explicados os aspectos de *timing* da propagação do pulso transiente. Para tanto, é necessário compreender os aspectos temporais do armazenamento de um sinal lógico por um elemento de memória, isto é, as condições necessárias para que um determinado sinal na entrada de um registrador possa ser armazenado. Essas condições são necessárias para evitar que um dado de entrada ou sinal de controle do registrador mude ao mesmo tempo em que a borda do sinal do relógio. Então, um sinal na entrada de um registrador só será capturado, com total garantia, se ele perdurar um intervalo de tempo delimitado por dois instantes de tempo característicos de cada registrador: t_{setup} (tempo de preparação ou de *setup*) e t_{hold} (tempo de manutenção ou de *hold*). Este intervalo de tempo é chamado de **janela de amostragem** (*latching-window*). O tempo de *setup* é o tempo antes da borda do sinal de relógio necessário para o sinal a ser capturado estar presente na entrada do registrador, enquanto que o tempo de *hold* é o instante

de tempo mínimo após a borda do relógio, necessário para o sinal ser capturado (BUSHNELL; AGRAWAL, 2000). A Fig. 23 ilustra os tempos t_{setup} e t_{hold} juntamente com a janela de amostragem. Nos casos ilustrados na Fig. 23, apenas o *dado1* possui garantia de ser armazenado pelo registrador, pois o sinal atinge a entrada do registrador antes de t_{setup} e tem duração que vai além de t_{hold} . Já os sinais *dado2* e *dado3* possuem uma probabilidade menor do que 1 de serem corretamente amostrados pelo registrador.

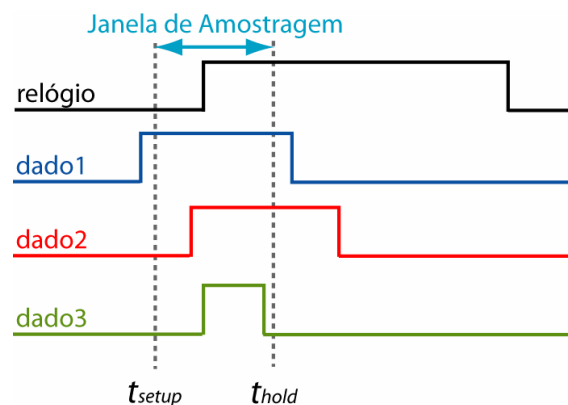


Figura 23 – Janela de amostragem de um registrador

Consideremos agora um circuito projetado corretamente, onde todos os aspectos de *timing* são respeitados. Vamos considerar também como sendo t_d o tempo de estabilização dos valores em uma rede r , *fanout* de uma porta p (i.e., r está conectada na saída de p), onde o SET ocorreu. Suponha que esse SET originou um pulso transiente no instante de tempo t_i com duração d . Além disso, considere também D_{max} como sendo o tempo máximo de propagação do pulso originado na rede r até uma PO (*Primary Output*) do circuito combinacional. De acordo com Alexandrescu (2004), existem duas condições que devem ser obedecidas para que um pulso transiente ocorrendo na rede r seja capturado por um registrador de saída:

- O pulso transiente deve ocorrer na rede r depois do tempo de estabilização de r . Isso ocorre pois, em caso contrário, o pulso transiente será absorvido pelo valor estável de r no instante de tempo t_d .
- O pulso transiente deve atingir a entrada do registrador de saída do circuito antes do tempo de *setup* do registrador.

As duas condições listadas anteriormente podem ser representadas pelas seguintes equações:

$$t_i + d > t_d \quad (6)$$

$$t_i + D_{\max} < t_{\text{setup}} \quad (7)$$

Através dessas equações, e considerando valores para o tempo de início e duração do pulso transiente, é possível identificar todas as redes onde um SET não se tornará uma falha transiente, pois não será capturado pelos registradores das saídas do circuito.

4.2 Uso da Análise de *Timing* Topológica para Avaliar os Aspectos de *Timing* da Propagação do Pulso Transiente

A verificação de *timing* determina se as restrições temporais impostas pelo projeto do circuito irão ser satisfeitas após sua fabricação. A partir dela é possível estimar o **atraso crítico** e a **freqüência máxima de operação** do circuito, no caso de circuitos seqüenciais síncronos. Esta verificação pode ser realizada através de simulação e/ou de **análise de *timing***. Por meio de simulação do circuito são obtidas estimativas mais precisas que através da análise de *timing*. Porém, a análise por meio de simulação requer a exploração total do espaço de entrada.

A análise de *timing* topológica (TTA – *Topological Timing Analysis*), também chamada de análise de *timing* estática (STA – *Statical Timing Analysis*), é uma técnica amplamente utilizada para estimar o atraso crítico de circuitos combinacionais, a partir do caminho mais longo do circuito (DEVADAS; KEUTZER; MALIK, 1993). Sua principal característica é ser uma técnica independente de vetores de entrada, isto é, não necessita de assinalamento de valores nas entradas primárias do circuito, sendo baseada apenas nas informações de atraso das portas lógicas e na topologia do circuito.

Para a utilização de TTA um bloco de circuito combinacional é representado por um Grafo Acíclico Direcionado (DAG – *Direct Acyclic Graph*) com peso nas arestas. Neste DAG os nodos estão associados às portas lógicas e as arestas representam as conexões do circuito. O atraso de cada porta do circuito pode ser armazenado no próprio nodo.

Além disso, são criados nodos “bobos” (*dummy nodes*), isto é, sem atraso, para as entradas e saídas primárias do circuito, a fim de auxiliar na execução do TTA. Em muitos casos, também são colocados mais dois nodos bobos chamados de nodo fonte (*source*), do qual partem arestas para as PIs do circuito, e o nodo terminal, onde chegam arestas que partem das POs do circuito. Assim, um caminho completo do circuito é formado pela lista de nodos (\mathbf{s} , \mathbf{a}_s , n_1 , \mathbf{a}_1 , n_2, \dots, n_n , \mathbf{a}_n , n_{n+1} , \mathbf{a}_t , \mathbf{t}), onde n_1 representa uma PI e n_{n+1} uma PO do circuito, \mathbf{s} e \mathbf{t} são os nodos fonte e terminal, respectivamente, e as arestas \mathbf{a}_s e \mathbf{a}_t são arestas “bobas”. As Figs. 24 e 25 ilustram o circuito de um somador completo e a sua representação utilizando um DAG, respectivamente.

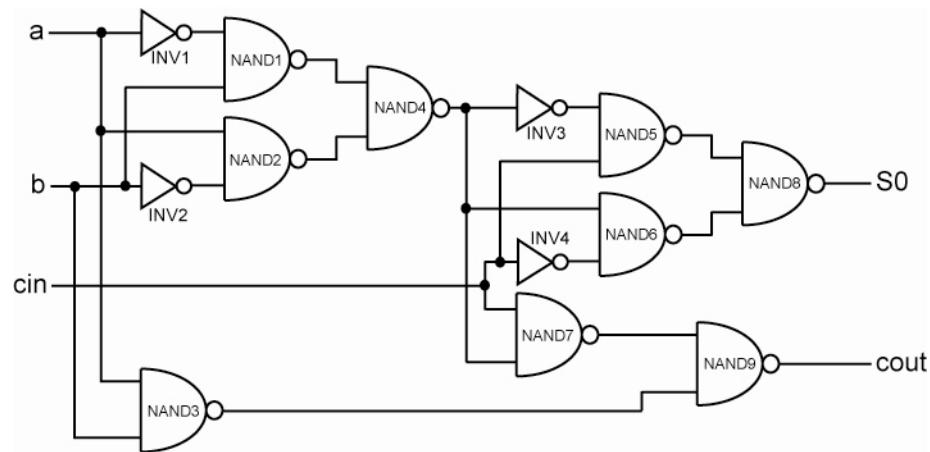


Figura 24 – O circuito de um somador completo

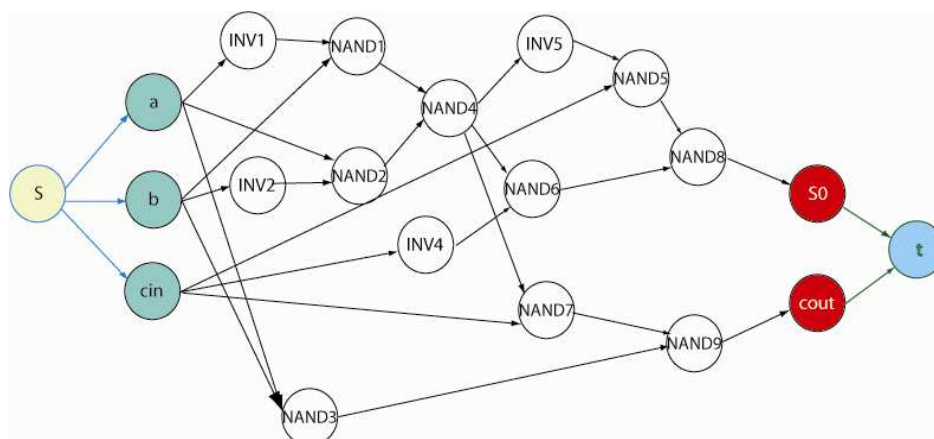


Figura 25 – DAG do circuito da Fig. 24

Após a montagem do DAG, o TTA encontra o maior caminho do circuito, o qual corresponde ao caminho que possui o maior atraso acumulado e assume esse

atraso como sendo o atraso crítico do circuito. Para isso, TTA utiliza o algoritmo *topological sort* (CORMEN; LEISERSON; RIVEST, 1990), o qual é bem conhecido por encontrar a melhor solução em tempo linear, em relação ao número de nodos do grafo.

O procedimento *topological sort* é ilustrado em pseudocódigo na Fig. 26. O processamento desse algoritmo se dá da seguinte forma: primeiramente são inseridos em uma fila de processamento todos os nodos sucessores do nodo fonte, isto é, todos os nodos que representam entradas primárias do circuito. A partir daí, um nodo de cada vez é retirado da fila e processado, até que todos os nodos do circuito sejam processados. Ao processar um nodo, o algoritmo examina cada um de seus nodos sucessores. Aqueles sucessores cujos antecessores já tiverem sido processados serão inseridos na fila de processamento.

O processamento de um nodo p , na versão do algoritmo *topological sort* implementada para ser usada em TTA consiste em encontrar o maior atraso de todos os caminhos possíveis que começam no nodo fonte e terminam no nodo p . Esse atraso é chamado de **mds** (*maximal delay from source*). Portanto, para cada nodo p do circuito é computado **mds(p)** através da equação que segue:

$$mds(p) = t_p + \max\{mds(a_i)\} \quad (8)$$

onde t_p é o atraso da porta p e $a_i \in \text{ante}(p)$ (lista dos nodos antecessores de p). Após todos os nodos serem processados, o nodo terminal terá como valor de **mds** o valor do atraso crítico do circuito.

```

                                Topological_sort(G)
1   insere na fila todos os nodos sucessores de  $s$ ;
2   enquanto fila não está vazia faça
3      $p$  = retira primeiro nodo da fila;
4     processa( $p$ );
5     para  $i \in$  sucessores de  $p$  faça
6       se todos os antecessores de  $i$  já foram processados então
7         insere na fila o nodo  $i$ ;
8     fim_se
9     fim_para
10  fim_enquanto

```

Figura 26 – Procedimento *topological sort* em pseudocódigo

Na próxima subseção será explicada a implementação do pré-processamento baseado em TTA para a análise do mascaramento por *latching*-

window. Também são mostrados os resultados obtidos através desse pré-processamento.

4.2.1 Pré-processamento Baseado TTA

Além das duas etapas (montagem do DAG e processamento do *topological sort*) citadas anteriormente, o pré-processamento proposto neste trabalho possui uma terceira etapa de processamento em sentido contrário ao processamento *topological sort*, a qual será explicada a seguir.

Neste trabalho são consideradas duas informações de atraso para cada porta: o atraso da porta para uma transição de subida na sua saída ($0 \rightarrow 1$) e o atraso da porta para uma transição de descida na sua saída ($1 \rightarrow 0$). Tais informações serão referenciadas como td_{LH} (*topological delay low-high*) e td_{HL} (*topological delay high-low*), respectivamente. Conseqüentemente, ao invés de se ter apenas uma informação de **mds**, é necessário que existam duas informações em cada nodo p : **mdsr**(p) (*maximal delay from source for a rising transition*) será o máximo atraso para uma transição de subida na saída de p , e **mdsf**(p) (*maximal delay from source for a falling transition*) será o máximo atraso para uma transição de descida.

Neste trabalho todos os circuitos alvo são mapeados para tecnologia CMOS. Todas as portas CMOS possuem a característica de serem portas inversoras, isto é, uma transição de subida na entrada provoca uma transição de descida na saída, caso ela se propague pela porta. Então, para calcular o **mdsr**(p) é necessária a informação de **mdsf** dos antecessores de p , enquanto que o valor de **mdsf**(p) é calculado a partir do valor de **mdsr** dos antecessores de p . Desta forma, no processamento realizado no algoritmo *topological sort* esses dois valores são calculados através das seguintes equações:

$$mdsf(p) = td_{HL} + \max\{mdsr(a_i)\} \quad (9)$$

$$mdsr(p) = td_{LH} + \max\{mdsf(b_i)\} \quad (10)$$

onde $a_i \in ante_{LH}(p)$ (lista dos nodos antecessores de p para uma transição de subida) e $b_i \in ante_{HL}(p)$ (lista dos nodos antecessores de p para uma transição de descida).

Após o cálculo dos **mdsr** e **mdsf** das portas é feita uma etapa de processamento de ordem inversa ao processamento da fase anterior, isto é, o grafo

é percorrido das saídas primárias do circuito para as entradas primárias, também utilizando o algoritmo *topological sort*. Este processamento foi chamado de **reverse topological sort** e está ilustrado na Fig. 27.

Nessa etapa o processamento de um nodo p calcula o seu **mdt** (*maximal delay to terminal*), isto é, o maior atraso dentre todos os caminhos possíveis entre o nodo p e o nodo terminal. Portanto, desse processamento resulta o valor do tempo máximo de propagação de um sinal desde um nodo qualquer do circuito até uma de suas saídas primárias.

```

                                reverse_topological_sort(G)
1   insere na fila todos os nodos antecessores de  $t$ ;
2   enquanto fila não está vazia faça
3      $p$  = retira primeiro nodo da fila;
4     processa( $p$ );
5     para  $i \in$  antecessores de  $p$  faça
6       se todos os sucessores de  $i$  já foram processados então
7         insere na fila o nodo  $i$ ;
8     fim_se
9     fim_para
10  fim_enquanto

```

Figura 27 – Pseudocódigo do algoritmo *reverse topological sort*

Semelhantemente ao caso de mds, mdt também é decomposta em duas variáveis distintas: **mdtr(p)** (*maximal delay to terminal for a rising transition*) que representa o atraso máximo para uma transição de subida em uma entrada de p e **mdtf(p)** (*maximal delay to terminal for a falling transition*) que representa o atraso máximo para uma transição de descida. O cálculo de mdtr e mdtf são realizados a partir das duas equações que seguem:

$$mdtf(p) = td_{HL} + \max\{mdtr(a_i)\} \quad (11)$$

$$mdtr(p) = td_{LH} + \max\{mdtf(b_i)\} \quad (12)$$

onde $a_i \in \text{suce}_{LH}(p)$ (lista dos nodos sucessores de p para uma transição de subida) e $b_i \in \text{suce}_{HL}(p)$ (lista dos nodos sucessores de p para uma transição de descida).

Para representar os nodos do grafo, é utilizada uma estrutura de dados chamada *GRAPH_NODE* conforme ilustrado pela Fig. 28.

Os campos da estrutura *GRAPH_NODE* representam as seguintes características do nodo:

- **name**: armazena o nome do nodo do grafo, o qual pode ser o nome de uma porta lógica, o nome de uma entrada primária, o nome de uma saída primária ou ainda “source” ou “terminal”, conforme o tipo de nodo.
- **nb_inputs**: contém o número de arestas que chegam no nodo. No caso de uma porta lógica, é o número de entradas da porta.
- **fanin_edges**: contém a lista das arestas que chegam no nodo.
- **nb_fanout**: representa o número de arestas que saem do nodo.
- **fanout_edges**: contém a lista das arestas que partem do nodo.
- **mds[2]**: armazena o valor de **mdsf** e **mdsr**, em **mds[0]** e **mds[1]**, respectivamente.
- **mdt[2]**: contém o valor de **mdtf** e **mdtr**, em **mdt[0]** e **mdt[1]**, respectivamente.
- **visit**: determina se um nodo já foi processado ou não em uma etapa de processamento do grafo.
- **status**: identifica se o nodo representa o nodo fonte, uma entrada primária, uma porta lógica, uma saída primária ou o nodo terminal.

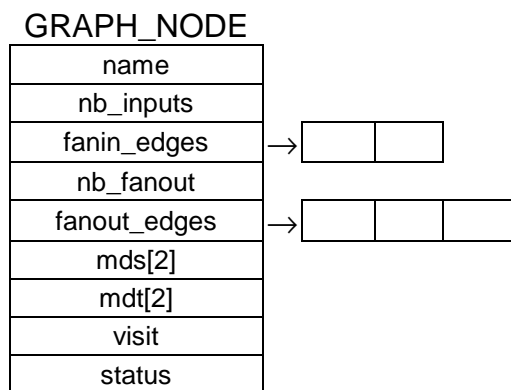


Figura 28 – Estrutura de um nodo do grafo

Para armazenar as informações de uma aresta do circuito, foi criada uma estrutura de dados chamada *EDGE_NODES*. Essa estrutura possui apenas dois campos:

- **fanin**: este campo aponta para o nodo da onde a aresta parte.
- **fanout**: aponta para o nodo onde a aresta chega.

4.2.2 Resultados Obtidos

Uma vez calculados os valores de **mds** e **mdt**, é possível calcular as equações (6) e (7) para identificar todas as portas do circuito onde um SET não consegue atingir os registradores das POs dentro da janela de amostragem. Para um nodo p qualquer, as informações de **mdsr(p)** e **mdsf(p)** coincidem respectivamente com as informações de t_{dr} e t_{df} , isto é, o tempo de estabilização de p para uma transição de subida e descida, respectivamente. Além disso, as informações de **mdtr(p)** e **mdtf(p)** representam o máximo atraso para um sinal propagar-se do nodo p até uma PO, ou seja, correspondem a D_{max} , considerando uma transição de subida ou descida em p , respectivamente.

Esse pré-processamento foi descrito e implementado em linguagem C. As características do pulso transiente **{ t_i , d }**, isto é, o tempo de início e a duração do pulso transiente, são fornecidos como dados de entrada, juntamente com o nome do arquivo que contém a descrição do circuito em formato SPICE. Para uma compreensiva análise dos circuitos combinacionais, a informação de tempo de início do pulso é informada como porcentagem do tempo mínimo de ciclo de relógio que pode ser aplicado ao circuito. A informação de duração do pulso é dada em picosegundos. Assume-se como tempo mínimo de ciclo de relógio o atraso crítico do circuito. Por exemplo, supondo D como o atraso crítico do circuito C , o par **{ $d=50$, $t_i=0,2$ }** significa que o circuito C será avaliado para um pulso transiente de duração de 50ps, começando no tempo $0,2 \times D$ (20% do atraso crítico) partindo da borda de início do relógio. Essa análise considera que um pulso com essas características pode ocorrer em qualquer porta lógica do circuito. Assim, a ferramenta provê uma lista com todas as portas do circuito de onde um SET pode propagar-se até uma PO.

O protótipo de ferramenta implementado foi executado para os circuitos do conjunto *benchmark* MCNC. Os circuitos foram todos mapeados apenas para portas NANDs e NORs de duas entradas e inversores. As informações de atraso de subida e descida de cada porta foram obtidos utilizando-se a ferramenta AGDA (*Automatic Gate Delay Analyzer*) (MATEUS, 2006), a qual realiza a caracterização das portas lógicas por meio de simulações elétricas com o simulador Spice Opus (SPICE,

2007), assumindo um modelo de tecnologia preditiva de 90nm de largura de canal do transistor (CAO et. al., 2000).

O pré-processamento dos circuitos do conjunto *benchmark* MCNC baseado TTA foi executado para esses circuitos utilizando cinco diferentes tempos de início de pulso t_i ($0 \times D$, $0,2 \times D$, $0,4 \times D$, $0,6 \times D$, $0,8 \times D$). A duração do pulso transiente simulado foi de 100ps. Essa duração do pulso transiente é justificada por estudos realizados por Dhillon et al. (2006), que mostraram que para tecnologias CMOS estado-da-arte, a duração do pulso transiente decorrente de um SET está dentro de intervalo de 1 – 100ps. Na tab. 2 são mostradas as características dos circuitos e o tempo de execução da ferramenta para esses circuitos. Os tempos de execução correspondem a rodar os 5 casos para cada circuito em um computador equipado com processador AMD Turion 2.0 de 64 bits com 1 GB de memória RAM.

Tabela 2 – Característica dos circuitos-exemplo e os tempos totais de execução

Circuito	Fanout médio	Fanout máximo	Profundidade Lógica (pl)	Número de portas (np)	Atraso crítico (ns)	Tempo total de execução (s)
C17	1,23	2	5	13	20,5	0,016
bw	1,68	4	13	279	42,2	0,451
C432	1,44	4	40	297	116,0	0,482
9sym	1,7	4	20	331	83,9	0,528
C880	1,5	4	36	512	126,2	1,138
C499	1,48	4	31	638	167,5	1,559
C1908	1,56	4	45	686	199,2	1,748
alu2	1,7	4	43	704	136,6	1,857
C1355	1,72	4	34	709	136,3	1,905
C2670	1,49	4	33	1128	94,4	4,763
C3540	1,66	4	58	1525	251,3	7,826
T481	1,72	4	26	2201	83,85	16,768
C6288	1,9	4	127	2624	498,7	27,669
C5315	1,62	4	56	2967	261,9	33,528
C7552	1,6	4	48	3459	219,9	48,137
alu4	1,73	4	25	5796	73,7	153,092

Como pode ser visto na tab. 2, para circuitos com menos de 1000 portas lógicas, o tempo de execução dos 5 casos foi menor ou igual a 2 segundos. Considerando o circuito C1355, por exemplo, com 709 portas lógicas, o tempo de execução de cada caso durou 0,4 segundos. Para os circuitos com mais de 1000 portas lógicas, o tempo de execução para todos os casos ficou entre 4,763

segundos para o circuito C2670 (que possui 1128 portas), e 153,092 segundos para o circuito alu4 (que possui 5796 portas). Porém, o resultado do circuito alu4 é uma exceção, pois esse circuito apresenta os maiores valores de *fanout* médio e um grande número de portas lógicas, já que para o circuito C7552 com um pouco menos que 3500 portas lógicas, o total do tempo de execução para todos os casos foi menor que 50 segundos, isto é, um pouco menos que 10 segundos para cada caso. Assim, pode-se concluir que a técnica proposta baseada em TTA é rápida o suficiente para ser usada como etapa de pré-processamento para análise de propagação de SETs.

Os gráficos das Figs. 29, 30, 31 e 32 mostram os resultados em termos de taxas de falhas transientes, ou SER (*soft-error rate*). Nestes gráficos, o eixo “y” representa o SER enquanto que o eixo “x” mostra o momento do início do pulso de 100ps, expresso como uma fração da duração do ciclo de relógio, “D”. O ciclo de relógio, por sua vez, foi estimado com base no atraso crítico topológico fornecido pelo próprio protótipo de ferramenta.

Os resultados obtidos referentes à estimativa de SER (*soft-error rate*) dos circuitos mostraram que para um pulso de duração de 100ps, quanto mais cedo for o momento de injeção do pulso, relativo ao início do ciclo de relógio, menor será o SER do circuito. Este resultado já era esperado, uma vez que pulsos que começam logo no início do período do relógio dispõem de mais tempo para se propagarem pela lógica, podendo então terem desaparecido quando da chegada da próxima borda ativa do sinal de relógio.

Particularmente, puderam-se perceber duas tendências distintas, quando se considera a razão entre o número de portas lógicas do circuito (np) e a sua profundidade lógica (pl). Essas tendências estão ilustradas nas Figs. 29 e 30. Examinando as figuras mais cautelosamente, nota-se que para momentos de injeção do pulso entre $0,2xD$ e $0,6xD$, o SER dos circuitos com np/pl baixo possuem SER significativamente menor que os circuitos com np/pl alto.

Como pode ser visto na Fig. 29, os circuitos que possuem um número pequeno de portas lógicas relativo à sua profundidade lógica, mostraram um crescimento quase que linear do número de portas lógicas suscetíveis em relação ao momento de injeção do pulso transiente. Porém, os circuitos ilustrados na Fig. 30,

os quais possuem valores altos na razão entre o número de portas lógicas e a sua profundidade lógica, mostraram um crescimento logarítmico.

Circuitos com baixa razão np/pl

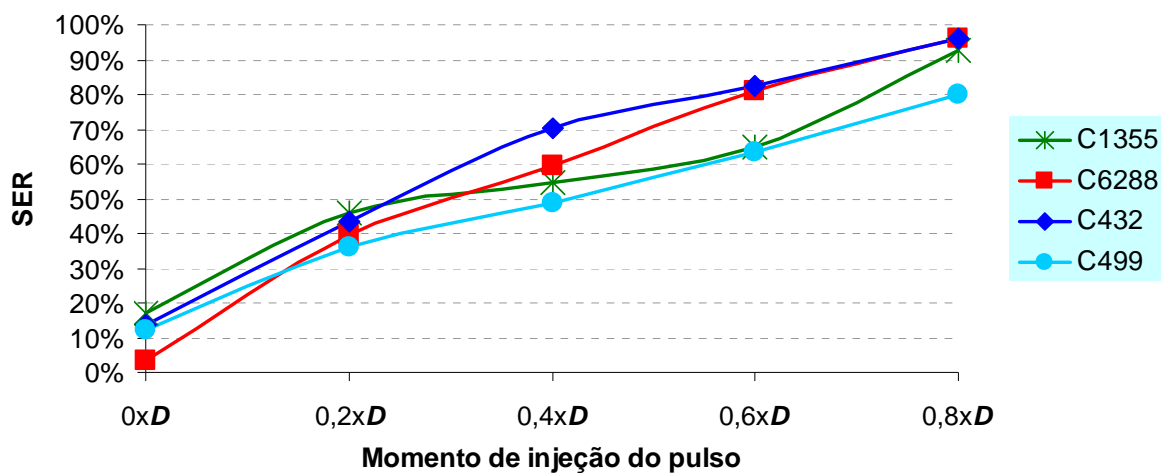


Figura 29 – Resultados para os circuitos com poucas portas em relação à profundidade lógica

Circuitos com alta razão np/pl

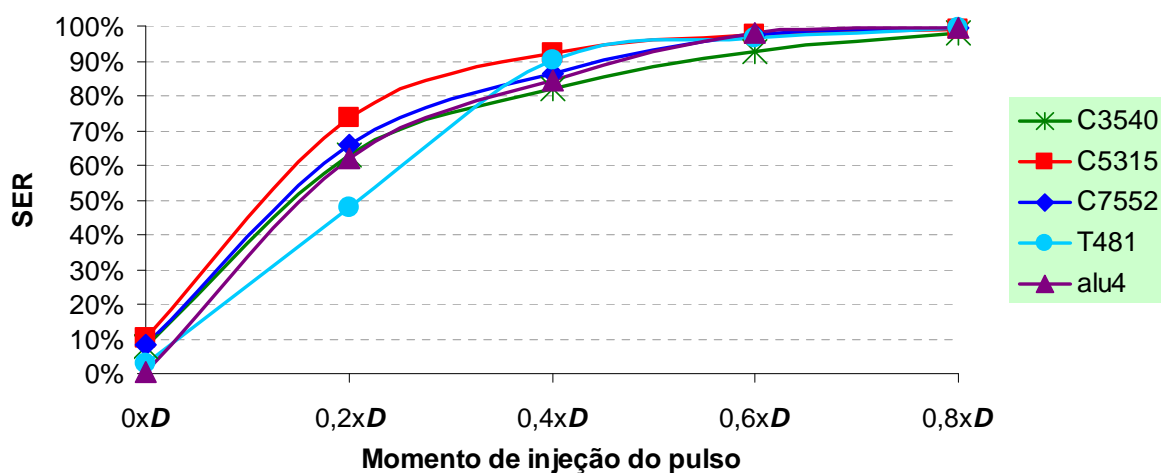


Figura 30 - Resultados para os circuitos com um grande número de portas em relação à profundidade lógica

Para demonstrar com maior ênfase os resultados de ambas as tendências, serão explicados em maiores detalhes os resultados de um circuito de cada

tendência. A Fig. 31 ilustra os resultados obtidos para o circuito alu4, o qual possui 5796 portas lógicas distribuídas em 25 níveis lógicos (portanto, com $np/pl = 231,84$). Note que para um pulso transiente de duração de 100ps, considerando o tempo de início do pulso igual a $0,4xD$, o SER do circuito é de 84,21%. Isto significa que, 15,79% das falhas não necessitam de simulação completa, uma vez não chegarão às saídas primárias fora da janela de amostragem (i.e., serão mascaradas por *latching-window*).

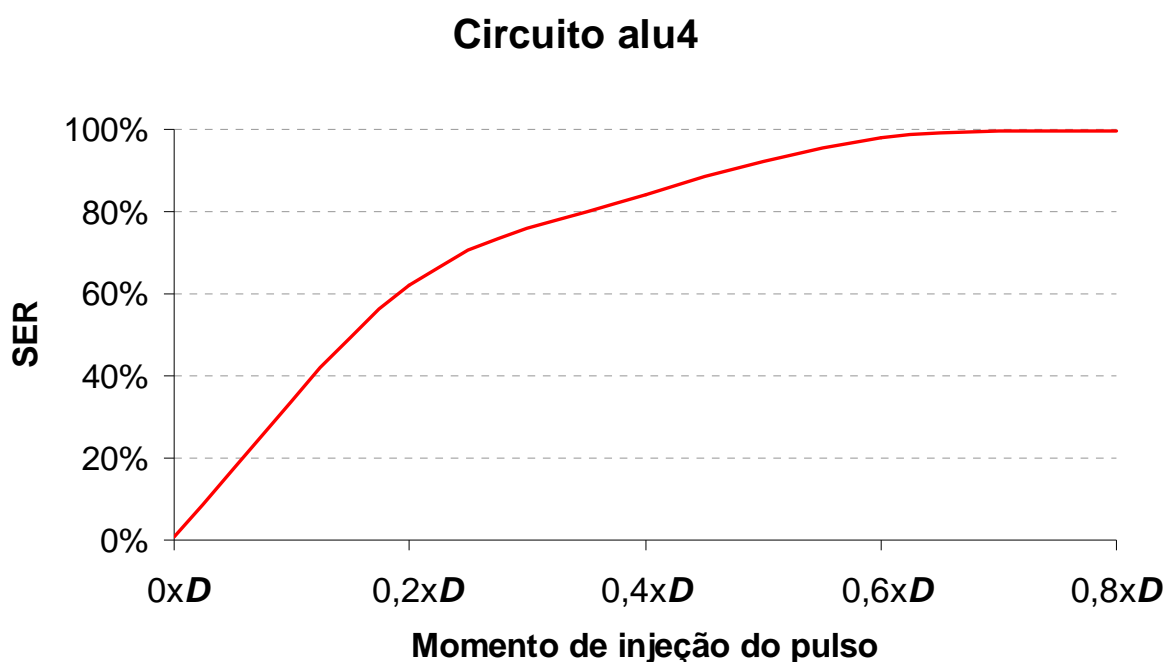


Figura 31 – Resultados para o circuito alu4

A Fig. 32 mostra os resultados referentes ao circuito C499 pertencente à segunda tendência observada nos resultados (com $np/pl = 20,58$). Analisando os resultados para os valores de t_i variando entre $0,2xD$ e $0,6xD$, o caso em que o SER do circuito é maior, onde t_i é igual a $0,6xD$, o número de falhas que devem ser consideradas é de 63,32%, resultando em um número de 36,68% de falhas que não necessitam ser testadas. Este é um resultado que faz sentido, uma vez que quanto maior for o número de níveis lógicos, mais chance do pulso chegar atrasado nas saídas primárias, e logo, fora da janela de amostragem.

Por fim, é importante ressaltar que se o número de portas lógicas suscetíveis for de 70% para um dado pulso de duração d e momento de injeção do pulso t_i ,

significa que somente 70% das falhas precisam ser testadas, pois os outros 30% não atingem os elementos de memória das POs dentro da janela de amostragem. Uma tal redução no número de testes implica em uma **aceleração de 42,9%** no tempo de execução da simulação de falhas.

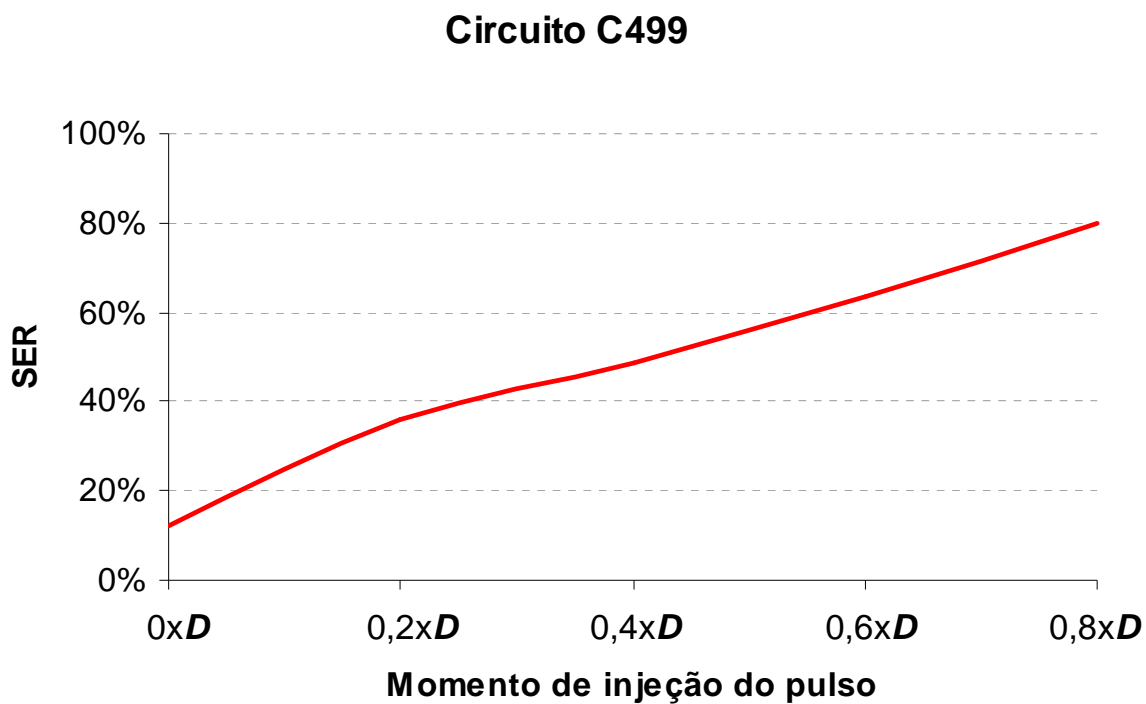


Figura 32 – Resultados para o circuito C499

5 Análise Automática do Mascaramento Lógico

Neste capítulo é abordada a análise da propagação de um pulso transiente em um combinacional sob o ponto de vista da funcionalidade lógica. Para isso, foram realizados diversos estudos sobre as questões de mascaramento lógico e sobre técnicas para analisar esse mascaramento. Ao final deste capítulo são mostrados os resultados obtidos com o protótipo da ferramenta desenvolvida.

5.1 Tratando Mascaramento Lógico

Para o entendimento do conceito de mascaramento lógico é necessário compreender os conceitos de controlabilidade das portas lógicas (GOLDSTEIN, 1979; ABRAMOVICI; BREUER; FRIEDMAN, 1990; BUSHNELL; AGRAWALL, 2000). Os conceitos de controlabilidade das portas aqui apresentados tratam apenas de portas lógicas simples, isto é, as quais aplicam uma função booleana básica. Definiremos agora os conceitos de **valores controlantes/não-controlantes** e **valores controlados/não-controlados** das portas lógicas. Para tanto, supomos uma porta lógica p como sendo uma porta simples (AND, OR, NAND, NOR).

O **valor controlante** de p é definido como sendo o valor lógico que sozinho determina o valor lógico da saída de p . Associado ao conceito de valor controlante está o de valor lógico controlado de uma porta lógica. O **valor controlado** de uma porta lógica é o valor da saída da porta, resultante da aplicação do valor controlante da porta em pelo menos uma de suas entradas. Por exemplo, se p for uma porta que aplica uma operação do tipo E (AND/NAND), o valor controlante de p será o valor 0, mas se p for uma porta do tipo AND o valor controlado é 0 e se p for uma porta do tipo NAND o valor controlado é 1.

O **valor não-controlante** de uma porta p é o valor lógico que sozinho não determina o valor da saída da porta. O **valor não-controlado** da porta p é o valor resultante na saída de p , oriundo da aplicação do valor não-controlante de p em todas as suas entradas. Por exemplo, o valor não-controlante de portas ANDs e NANDs é 1, porém o valor não-controlado de portas do tipo AND é 1 e o valor não-controlado de uma porta do tipo NAND é 0.

Com relação ao mascaramento lógico, dizemos que uma falha é mascarada (ou bloqueada) pela lógica do circuito se ela não atinge as saídas de uma determinada porta p devido aos valores das suas entradas que estão livres de falhas. Ou seja, para que uma falha consiga propagar-se pela lógica do circuito, sem haver bloqueios lógicos no seu caminho, é necessário que sejam respeitadas as seguintes condições:

- p é uma porta lógica pertencente ao caminho da falha até uma das saídas primárias;
- Todas as entradas de p que estão livres de falha, estão estáveis no valor não-controlante de p .

As condições citadas acima mostram que, o tratamento do mascaramento lógico de falhas transientes em circuitos combinacionais pode ser modelado como um **problema de sensibilização estática de caminhos** (DEVADAS, 1993). Um caminho dito **estaticamente sensibilizável** é aquele que não bloqueia a propagação de um pulso de tensão devido à lógica do circuito. A Fig. 33 mostra um caminho estaticamente sensibilizável para a propagação de um pulso transiente.

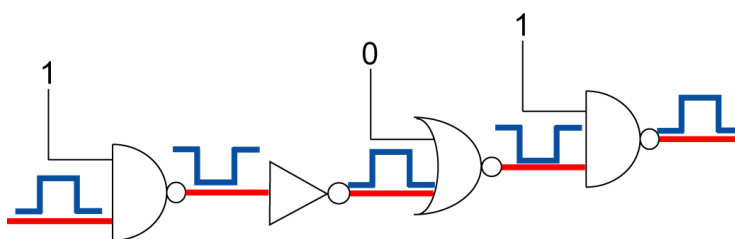


Figura 33 – Exemplo de sensibilização de um caminho

Tendo em vista esse problema, é necessário um método para analisar a possibilidade da ocorrência desses valores não-controlantes nas entradas livres de falhas das portas que pertencem ao caminho da falha até pelo menos uma das saídas primárias do circuito.

5.2 Técnicas de ATPG e sua Aplicação para a Análise do Mascaramento Lógico

Geração automática de padrões de teste (*Automatic Test-Pattern Generation - ATPG*) é o processo de geração de vetores de entrada para testar um circuito, o qual é geralmente descrito em nível lógico. A esse esquemático se dá o nome de *netlist*.

Algoritmos de ATPG possuem diversas funções, além da geração de padrões de teste. É possível também encontrar redundância ou sub-circuitos desnecessários. Com ATPG também é possível provar se um circuito implementa a mesma função lógica que outro circuito. (MA; DEVADAS; SANGIOVANNI-VINCENTELLI, 1989)

A maioria dos métodos para geração de testes para circuitos combinacionais utiliza o modelo de **falha de colagem simples** (*single stack-at faults*) (ABRAMOVICI; BREUER; FRIEDMAN, 1990; BUSHNELL; AGRAWALL, 2000; WANG; WU; WEN, 2006). Esse modelo trata de uma falha que afeta uma **linha** de um circuito combinacional, incluindo entradas primárias, saídas primárias e entradas ou saída de portas internas. Nesse modelo, uma linha l qualquer do circuito é dita fixa sempre em um valor lógico v (0 ou 1). Tal falha é representada como $l\ s-a-v$ (*l stack at v*). Portanto, existem 2 tipos básicos de falhas nesse modelo: se uma linha l está fixa no valor lógico 0, a falha é $l\ s-a-0$. Caso contrário, a falha é $l\ s-a-1$.

Para ser detectada uma falha $l\ s-a-v$, é necessário assinalar em l um valor lógico \bar{v} . Após esse assinalamento, o procedimento para a geração do padrão de teste para essa falha consiste de duas etapas: **ativação da falha** e **propagação do erro** até alguma das saídas primárias do circuito.

A ativação da falha consiste em encontrar um conjunto de valores que aplicados às entradas primárias do circuito e propagados pela lógica do circuito, fazem aparecer em l o valor lógico \bar{v} . Essa etapa é necessária para garantir um comportamento diferente entre o circuito bom e o com falha.

Esse processo descrito acima é chamado de **justificação de linha**. Usando uma definição genérica, justificação de linha é um procedimento recursivo no qual a saída de uma porta é justificada a partir do correto assinalamento de valores nas suas entradas e assim por diante, até o assinalamento de valores nas entradas primárias do circuito (GÜNTZEL, 2000). Após a linha l ser justificada, o algoritmo

APTG passa para a etapa de propagação do erro, isto é, verificar se o valor lógico \bar{v} atinge pelo menos uma das saídas primárias do circuito.

Tendo em vista que a propagação do erro consiste em sensibilizar pelo menos um dos caminhos que começam na linha l e terminam em uma das saídas primárias do circuito, é necessário, portanto, justificar os valores não-controlantes nas entradas livres de falhas das portas pertencentes ao caminho. Com isso, pode-se dizer que ambos os processos de ativação da falha e propagação do erro são aplicações diretas do processo de justificação de linha (GÜNTZEL, 2000).

Porém, para a etapa de propagação do erro é necessário definir a notação de sinal de erro. Essa notação foi introduzida por Roth (ROTH, 1966) e permite comparar o valor em um circuito bom (livre de falhas) com o valor em um circuito com falha. Essa notação, conhecida como **notação D**, origina-se da composição de dois valores v/v_f , que representam o valor lógico em um circuito bom e em um circuito com falha, respectivamente.

Dadas duas instâncias de um circuito C , sendo N uma instância livre de falhas e N_f uma instância com falha. Considere que ambas as instâncias estão com os mesmos valores nas suas entradas primárias. Se uma dada linha do circuito apresenta o valor lógico 1 na instância N e o valor lógico 0 na instância N_f , então a composição dos valores 1/0 representa a ocorrência de uma fonte de erro no circuito.

A composição 1/0 é representada pela variável **D**. Da mesma forma, se na instância N o valor lógico de uma dada linha for 0 e na instância N_f for 1, então se tem a composição 0/1, com os valores inversos aos valores representados por **D**. Portanto, essa nova composição é conhecida como \bar{D} . Assim, o espaço de valores booleanos é estendido, conforme pode ser visto na tab. 3. Além dos valores mostrados na tab. 3, utiliza-se o símbolo **X** para representar um valor indeterminado.

Para aplicar as operações lógicas a variáveis compostas (v/v_f) , as propriedades da álgebra booleana são mantidas. Porém, as operações devem ser efetuadas separadamente para os valores v e v_f , como por exemplo, na expressão $D + 1/1 = 1/0 + 1/1 = 1+1/0+1 = 1/1 = 1$. Essa aplicação da álgebra booleana ao conjunto de valores $\{0, 1, D, \bar{D}, X\}$ é conhecida como **Cálculo D** (*D-Calculus*). Na

tab. 4, é apresentada a tabela-verdade da operação de negação aplicada ao Cálculo-D.

Tabela 3 – Extensão de valores lógicos para a notação D

	v/v_f
0	0/0
1	1/1
D	1/0
\overline{D}	0/1

Tabela 4 – Tabela-verdade da operação de negação no Cálculo-D

E	S
0	1
1	0
D	\overline{D}
\overline{D}	D
X	X

Nas tabs. 5 e 6 apresentam respectivamente as tabelas-verdade das operações de E e OU aplicadas aos valores compostos do Cálculo-D.

Tabela 5 – Tabela-verdade da operação E no Cálculo-D

E	0	1	D	\overline{D}	X
0	0	0	0	0	0
1	0	1	D	\overline{D}	X
D	0	D	D	0	X
\overline{D}	0	\overline{D}	0	\overline{D}	X
X	0	X	X	X	X

Tabela 6 – Tabela-verdade da operação OU no Cálculo-D

OU	0	1	D	\overline{D}	X
0	0	1	D	\overline{D}	X
1	1	1	1	1	1
D	D	1	D	1	X
\overline{D}	\overline{D}	1	1	\overline{D}	X
X	X	1	X	X	X

Neste trabalho os circuitos tratados são formados por portas lógicas CMOS simples, isto é, NANDs, NORs e inversores. Para as portas NAND e NOR, os

valores resultantes no Cálculo-D são o inverso dos valores resultantes das operações E e OU, respectivamente.

Este trabalho aborda a análise da propagação do pulso transiente resultante da ocorrência de uma falha transiente (vista no capítulo 2). É necessário, então, modelar o pulso transiente para a implementação do Cálculo-D. A ocorrência de um pulso de descida ($1 \rightarrow 0 \rightarrow 1$) em uma dada linha l do circuito representa a ocorrência do valor 1 no circuito livre de falha e a ocorrência do valor 0, temporariamente, na linha l quando o circuito sofre a falha.

A situação citada acima é muito semelhante à representada pelo valor D ($1/0$). Portanto, para a elaboração deste trabalho, o pulso transiente de descida é modelado como sendo o valor D , enquanto que o pulso de subida ($0 \rightarrow 1 \rightarrow 0$) é representado pelo valor \bar{D} . A Fig. 34 mostra essa modelagem.

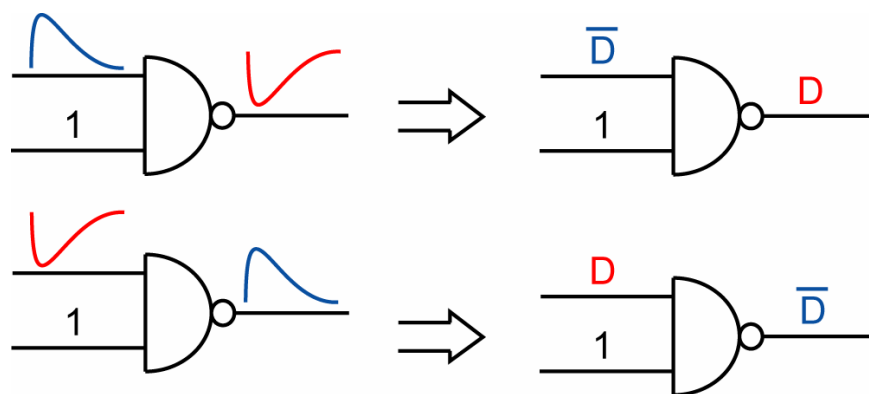


Figura 34 – Modelagem do pulso transiente para o Cálculo-D

Em circuitos combinacionais que não apresentam caminhos reconvergentes, isto é, existe apenas um caminho entre uma dada linha l do circuito e alguma PO, qualquer seqüência de decisões efetuadas referente a justificação de linhas resulta em uma solução válida para a geração do teste.

Porém, em circuitos combinacionais que apresentam caminhos reconvergentes, a justificação de uma linha pode interferir na justificação de alguma outra linha, gerando uma inconsistência. Uma inconsistência surge com a necessidade de se assinalar dois valores diferentes a uma mesma linha do circuito.

Como uma seqüência de assinalamentos pode resultar em uma inconsistência, o algoritmo ATPG deve permitir uma exploração do espaço de possíveis soluções, a fim de recuperar os valores atribuídos anteriores a uma decisão, caso essa decisão resulte em uma inconsistência. Para isso, essa classe

de algoritmos possui um **processo de retrocesso**, também conhecido por *backtracking*.

A seguir são explicadas as medidas de testabilidade necessárias para a compreensão de técnicas que as usem, como por exemplo, o algoritmo **PODEM** (*Path-Oriented DEcision Making*) que será explicado em detalhes na subseção 5.2.3.

5.2.1 Medidas de Testabilidade

As medidas de testabilidade têm por função orientar as tomadas de decisão de forma a acelerar o processamento de algoritmos de ATPG, como por exemplo, o algoritmo PODEM (veja seção 5.2.3). Elas medem o custo ou esforço para testar um determinado circuito lógico. Essas medidas, originalmente propostas por Goldstein (1979) baseiam-se nos conceitos de controlabilidade e observabilidade.

A controlabilidade de uma linha do circuito representa a capacidade de controlar um determinado valor lógico nessa linha. Ou seja, a controlabilidade mede a dificuldade em justificar uma linha do circuito a partir dos valores aplicados às PIs, enquanto que a observabilidade representa a dificuldade em propagar um valor lógico da linha até alguma PO. A seguir são apresentadas duas medidas de controlabilidade e observabilidade utilizadas para circuitos combinacionais.

A. Controlabilidade e Observabilidade Combinacionais

Para o cálculo da controlabilidade combinacional, primeiramente todas as entradas primárias do circuito são inicializadas com o valor de controlabilidade igual a 1, $CC^0[e] = 1$ (Controlabilidade combinacional para 0) e $CC^1[e] = 1$ (Controlabilidade combinacional para 1).

A partir desse cálculo, para as demais linhas do circuito os valores de controlabilidade são calculados a partir das equações da Fig. 35. As equações apresentadas são referentes às portas NANDs e NORs. Para as portas inversoras a $CC^0[s] = CC^1[e]$ e vice-versa, onde e representa a entrada da porta e s a sua saída.

O cálculo da controlabilidade das linhas internas do circuito obedece à ordem de processamento do algoritmo *Topological Sort* (descrito no capítulo 4). Como mostrado nas equações da Fig. 35, a distância em número de portas entre uma linha qualquer do circuito e uma PI é levada em consideração. Essa distância é conhecida como **profundidade lógica**, e está representada nas equações pelo valor 1 somado no final delas. Para o cálculo da controlabilidade do valor não-controlado,

soma-se o valor da controlabilidade relativa aos valores não-controlantes de todas as entradas. Enquanto que para o cálculo da controlabilidade do valor controlado é necessário obter o valor de menor controlabilidade relativa ao valor controlante dentre todas as entradas da porta.

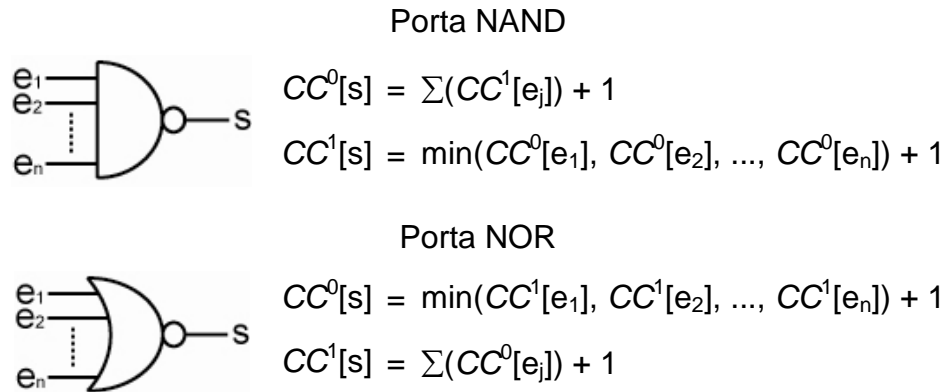


Figura 35 – Cálculo da controlabilidade combinacional de portas NANDs e NORs

A observabilidade combinacional é calculada de maneira similar controlabilidade, porém, partindo das saídas primárias para as entradas primárias do circuito. Então, inicialmente a todas as saídas primárias do circuito é atribuído o valor de observabilidade igual a 1, $OC[s] = 1$ (Observabilidade combinacional). Na Fig. 36 estão as equações para o cálculo da observabilidade de portas NANDs e NORs. As equações da Fig. 36 mostram que para a observabilidade a profundidade lógica também é levada em conta.

A observabilidade combinacional de uma entrada de uma porta é calculada pela soma da observabilidade da saída, com a controlabilidade relativa ao valor não-controlante de todas as outras entradas da porta e com o fator 1.

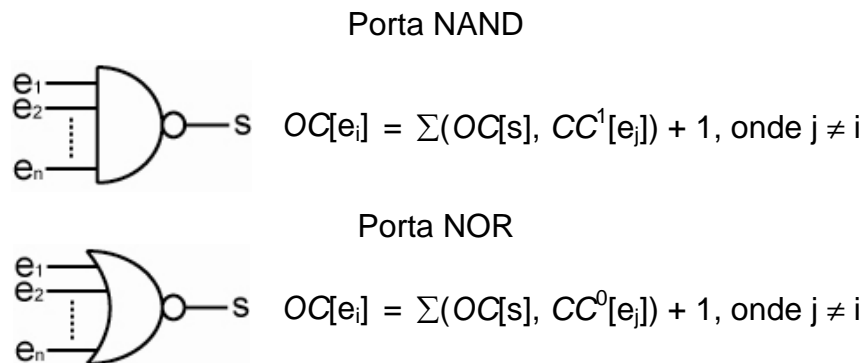


Figura 36 – Cálculo da observabilidade combinacional de portas NANDs e NORs

B. Controlabilidade e Observabilidade Probabilísticos

Os cálculos da controlabilidade e da observabilidade probabilística (BRGLEZ, POWNALL; HUM, 1984) obedecem às mesmas ordens de processamento da controlabilidade e observabilidade combinacional descritas no item anterior.

Para calcular as controlabilidades probabilísticas todas as PIs do circuito são inicializadas com o valor 0.5. A Fig. 37 mostra as equações para o cálculo da controlabilidade probabilística de portas NAND e NOR. A controlabilidade probabilística relativa ao valor não controlado da saída de uma porta é igual ao produto da controlabilidade probabilística relativa ao valor não controlante de todas as entradas da porta. A controlabilidade probabilística relativa ao valor controlado é o complemento da controlabilidade probabilística relativa ao valor não controlado.

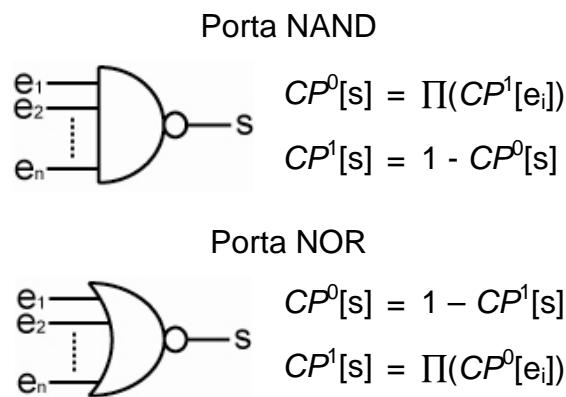


Figura 37 – Cálculo da controlabilidade probabilística de portas NANDs e NORs

Para calcular a observabilidade probabilística, inicialmente as saídas primárias do circuito são inicializadas com o valor 0.5. A observabilidade probabilística das demais linhas do circuito é calculada de acordo com as equações da Fig. 38.

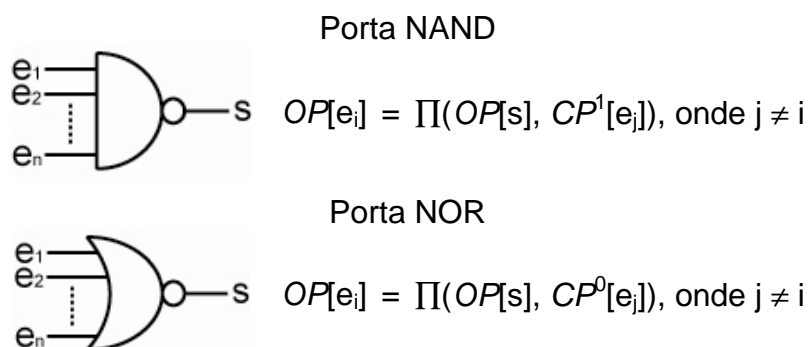


Figura 38 – Cálculo da observabilidade probabilística de portas NANDs e NORs

A seguir são apresentados dois algoritmos baseados em ATPG. Primeiramente é abordado o **algoritmo D**, o algoritmo clássico baseado em ATPG. Na seção seguinte é apresentado o algoritmo PODEM, algoritmo base para elaboração deste trabalho.

5.2.2 Algoritmo D

O algoritmo D foi proposto a fim de tratar a geração de vetores de entrada em circuitos combinacionais (ROTH, 1966). Como o nome da técnica propriamente diz, o algoritmo D tenta propagar um valor lógico D ou \bar{D} da linha onde o erro se originou até uma das saídas primárias do circuito. A principal característica do algoritmo D é admitir a propagação de erro por caminhos reconvergentes. Esta característica é referenciada por **sensibilização de múltiplos caminhos** (SCHENEIDER, 1967).

Para discutir com detalhes o mecanismo de funcionamento do algoritmo D, é necessário conceituar dois conjuntos importantes para os algoritmos de ATPG, chamados de **fronteira-D** e **fronteira-J**. A **fronteira-D** consiste do conjunto de todas as portas que possuem um sinal de erro (D ou \bar{D}) em pelo menos uma de suas entradas, porém não possui um valor definido na sua saída. Para que isso ocorra, uma ou mais entradas da porta também possuem uma indeterminação no seu valor. Representa-se essa indeterminação de uma linha por X (“*don't care*”).

Por exemplo, no início do algoritmo D, com uma falha f a ser detectada, existe apenas um D (ou \bar{D}) no circuito, isto é, a própria falha. Todos os outros sinais ainda estão com valor X . Nesse caso, a **fronteira-D** é composta por todas as portas sucessoras da linha com a falha f .

Como já foi citado anteriormente, após a ativação da falha é necessário propagar o sinal de erro através da justificação de valores não-controlantes nas entradas livres de falhas das portas pertencentes a pelo menos um dos caminhos da fonte de falha até uma saída primária do circuito. Para essa etapa o algoritmo seleciona uma das portas da **fronteira-D** e assinala valores lógicos nas suas entradas com valores X , de forma a aparecer D ou \bar{D} na sua saída. Se a **fronteira-**

D estiver vazia, significa que para os valores assinalados nas PIs, a falha não pode ser detectada. A Fig. 39 mostra um exemplo de fronteira- D .

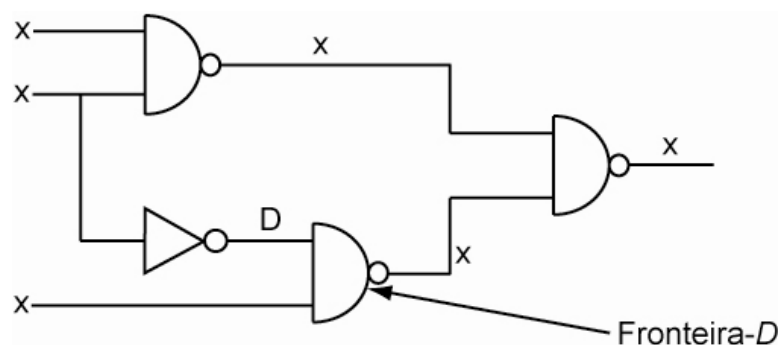


Figura 39 – Fronteira- D contendo uma porta

A fronteira- J é o conjunto de todas as portas do circuito cujos valores das suas saídas são um dos cinco valores do Cálculo- D , mas não foram justificados pelos valores lógicos das suas entradas. Ela é utilizada para controlar os problemas de justificação de linhas não solucionados. Deste modo, para uma falha ser detectada, todas as portas na fronteira- J devem ser justificadas; de outra forma, uma ou mais portas na fronteira- J poderão causar um conflito, onde suas saídas não podem ser justificadas para o valor desejado. A Fig. 40 mostra um exemplo da fronteira- J .

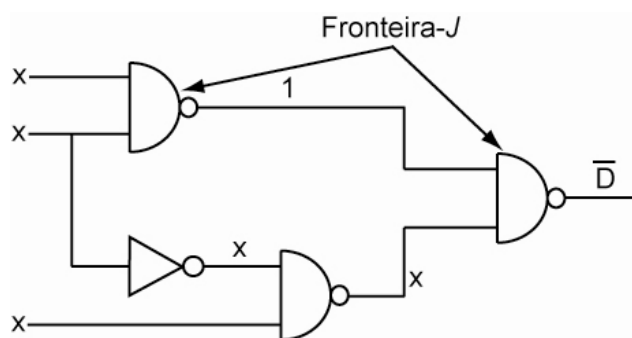


Figura 40 – Exemplo de fronteira- J

Tendo em vista esses dois conceitos, pode-se começar a explicação do algoritmo D . O algoritmo D inicia tentando propagar o sinal de erro (D ou \bar{D}) da linha l , fonte de falha, até uma das POs. Esse processo de propagação consiste em assinalar os valores não-controlantes nas entradas livres de falhas de todas as portas de um caminho necessário para propagar o sinal de erro. À medida que o sinal de erro é propagado para uma das POs, a fronteira- D se reduz a apenas as

POs atingidas pelo erro. Além disso, os valores não-controlantes assinalados nas entradas livres de falhas das portas pertencentes a esse caminho, formam a *fronteira-J* pois ainda não foram justificados pela implicação dos valores de suas entradas. A Fig. 41 ilustra esse procedimento descrito acima.

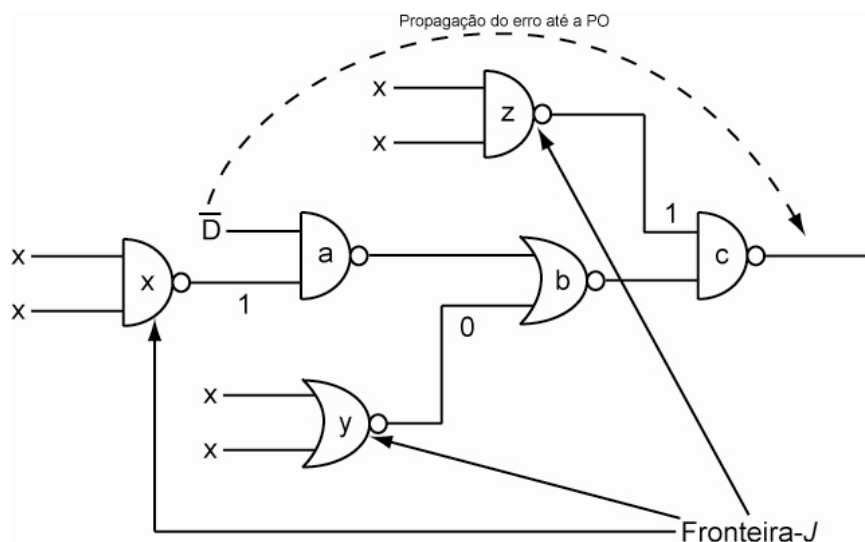


Figura 41 – Propagação do sinal de erro até POs

Assim que o sinal de erro atinge uma saída primária, todas as portas que estão na *fronteira-J* devem ter os valores lógicos de suas saídas justificados. Para isso, o algoritmo avança na *fronteira-J* em direção às PIs, colocando as portas antecessoras na *fronteira-J* de forma a justificar as portas antes não justificadas.

Tanto na propagação do sinal de erro quanto na justificação dos valores das saídas das portas que fazem parte da *fronteira-J*, sempre que um conflito ocorre, um *backtracking* é chamado. O procedimento completo do algoritmo D é representado por dois algoritmos em pseudocódigo mostrados nas Figs. 42 e 43.

```

                                Algoritmo-D(C,f)
1  inicializa todas as linhas com valor X;
2  assinala ( $D$  ou  $\bar{D}$ ) em uma linha  $l$  e a insere na fronteira-D;
3  fronteira-J =  $\emptyset$ ;
4  resultado = Alg-D-Recursivo(C);
5  se resultado == SUCESSO então
6      imprime os valores das PIs;
7  senão
8      imprime falha na linha  $l$  não é testável;
9  fim_se

```

Figura 42 – Algoritmo D

```

                                Alg-D-Recursivo(C)
1  se existe um conflito ou fronteira-D está vazia então
2    retorne FALHA;
3  fim_se
4  se nenhum sinal de erro atingiu uma PO então
5    enquanto existe uma porta na fronteira-D que não foi tratada faça
6      g = uma porta da fronteira-D que não foi tratada;
7      assinala todas as entradas de g no valor não-controlante;
8      adiciona todas as entradas de g na fronteira-J;
9      resultado = Alg-D-Recursivo(C);
10     se resultado == SUCESSO então
11       retorne SUCESSO;
12     fim_se
13   fim_enquanto
14   retorne FALHA;
15 fim_se
16 se fronteira-J está vazia então
17   retorne SUCESSO;
18 fim_se
19 g = uma porta da fronteira-J;
20 enquanto g não foi justificada faça
21   j = uma entrada não assinalada de g;
22   assinala j = 1 e insere na fronteira-J;
23   resultado = Alg-D-Recursivo(C);
24   se resultado == SUCESSO então
25     retorne SUCESSO;
26   senão
27     assinala j = 0;
28   fim_se
29 fim_enquanto
30 retorne FALHA;

```

Figura 43 – Procedimento recursivo do algoritmo D

O procedimento para o algoritmo D não incorpora qualquer tipo de inteligência no processo de decisão. Assim sendo, às vezes é possível determinar que alguns valores assinalados não são justificáveis no estado atual do circuito porém, devido a essa falta de inteligência, o algoritmo poderá assinalar esses valores necessitando de futuros *backtrackings*. Na Fig. 44 é possível perceber que para justificar a saída da porta $a = 0$ é necessário que seja assinalado o valor lógico 1 em todas as suas entradas. Porém, uma das entradas também é entrada da porta b , que para justificara sua saída $b = 1$, necessita do assinalamento do valor lógico 0 em todas as suas entradas, ocorrendo assim um conflito.

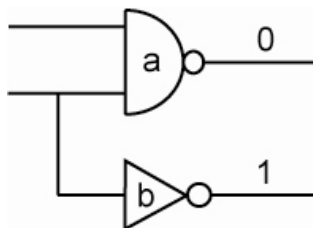


Figura 44 – Conflito na justificção de valores

5.2.3 Algoritmo PODEM

As memórias DRAM incluídas nos computadores *mainframes* que surgiram em meados dos anos 70, necessitavam de circuitos do tipo ECAT (*Error Correction And Translation*) para a detecção e correção de erros. Tais circuitos são caracterizados por apresentarem caminhos reconvergentes e portas do tipo XOR. Em estudos relatados por Goel (1981), o algoritmo D demonstrou-se ineficiente para tais tipos de circuitos. Para lidar com tais circuitos, Goel (1981) propôs o algoritmo PODEM (*Path-Oriented DEcision Making*).

Como foi visto na subseção 5.2.2, no algoritmo D o espaço de decisão abrange todas as linhas do circuito. Porém, o resultado final do algoritmo, e de qualquer algoritmo ATPG, é um vetor de soluções nas entradas primárias do circuito. Geralmente, o número de entradas do circuito é expressivamente menor que o total de portas do circuito (WANG; WU; WEN, 2006).

Então, baseado nessa relação entre número de PIs e número linhas do circuito, o algoritmo PODEM toma decisões somente nas PIs. Como no algoritmo D, a *fronteira-D* é mantida, mas como as decisões são feitas somente nas PIs, a *fronteira-J* não é necessária. No algoritmo PODEM, a cada passo do procedimento de busca ATPG, o algoritmo verifica se a falha foi ativada.

Quando a falha é ativada, o PODEM verifica se existe um caminho composto apenas de linhas com valor indeterminado (com valor X), começando em uma das portas que estão na *fronteira-D* e terminando em uma PO. Esse caminho composto somente por valores X é conhecido como **caminho-X**. Se não existe um caminho-X, então significa que todos os sinais de erro na *fronteira-D* estão sendo bloqueados, como mostra a Fig. 45, onde todos os caminhos possíveis entre o sinal de erro e uma PO estão bloqueados pela lógica.

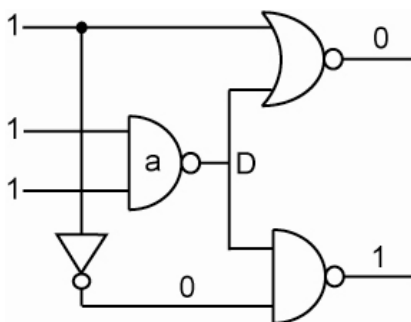


Figura 45 – Circuito sem caminho-X

Caso exista pelo menos um caminho-X, o algoritmo PODEM selecionará o melhor caminho para propagar o sinal de erro, baseado na medida de observabilidade das portas que estão na fronteira-*D*. As linhas mais observáveis são selecionadas primeiro. A Fig. 46 mostra o fluxograma em alto-nível do algoritmo PODEM.

Para as etapas de ativação da falha e propagação do erro, o algoritmo PODEM possui três procedimentos básicos: *backtrace*, implicação e *backtracking*. A etapa de *backtrace* consiste em assinalar um valor lógico em uma PI do circuito a fim de justificar uma linha / desejada do circuito. Para tanto, o algoritmo assinala valores lógicos nas linhas do caminho partindo da linha / até uma PI.

Após uma PI ser assinalada, o algoritmo passa para a etapa de implicação. Nessa etapa, o algoritmo implica os valores lógicos nas saídas das portas do circuito, de acordo com os valores lógicos em suas entradas. A implicação pode resultar em três diferentes resultados:

- Caso o objetivo seja satisfeito, isto é, a linha / está justificada, o algoritmo busca um novo objetivo a ser satisfeito.
- Caso o objetivo não seja satisfeito, uma nova etapa de *backtrace* é necessária para assinalar valores em outras PIs.
- Se a implicação resultar em um conflito, a etapa de *backtracking* desfaz o último assinalamento feito em uma PI.

Como todas as decisões são tomadas apenas para as PIs, todos os sinais internos do circuito são determinados a partir de implicações dos valores assinalados às PIs. Com isso, os únicos conflitos possíveis no algoritmo PODEM ocorrem no caso da impossibilidade de ativação da falha e caso a fronteira-*D* se torne vazia. Em ambos os casos, uma decisão anterior deve ser retomada

(*backtracking*). A etapa de *backtracking* desfaz o último assinalamento feito em uma PI e assinala o valor inverso nessa PI. Após uma etapa de *backtracking*, uma etapa de implicação é feita. Nas Figs. 47 e 48 estão os algoritmos básicos em pseudocódigo para implementação do algoritmo PODEM.

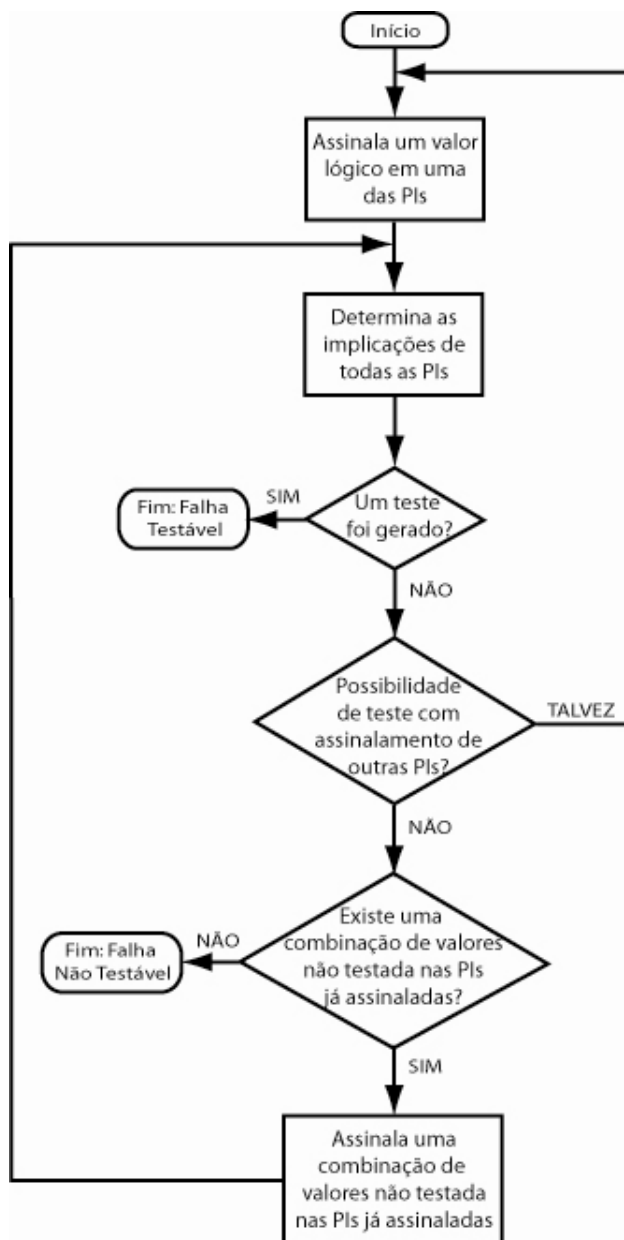


Figura 46 – Fluxograma em alto nível do algoritmo PODEM

De acordo com o algoritmo da Fig. 48, o PODEM começa escolhendo um objetivo a ser atingido, isto é, uma linha do circuito a ser justificada. A partir da linha o algoritmo traça um melhor caminho até uma PI do circuito através do procedimento

backtrace(). Para definir qual é o melhor caminho, o algoritmo baseia-se nas medidas de controlabilidade das linhas. Assim, o algoritmo assinala valores às PIs do circuito, uma de cada vez.

```

                                PODEM(C, f)
1  inicializa todas as linhas com o valor X;
2  fronteira-D =  $\emptyset$ ;
3  resultado = PODEM-Recursivo(C);
4  se resultado == SUCESSO então
5      imprime os valores das PIs;
6  senão
7      imprime falha f não testável;
8  fim_se

```

Figura 47 – Algoritmo alto-nível para o algoritmo PODEM

```

                                PODEM-Recursivo(C, f)
1  se o sinal de erro é observado em uma PO então
2      retorne SUCESSO;
3  fim_se
4  (g, v) = objetivo(C);
5  (pi, u) = backtrace(g, v);
6  implicacao(pi, u);
7  resultado = PODEM-Recursivo(C);
8  se resultado == SUCESSO então
9      retorne SUCESSO;
10 fim_se
11 /*backtracking */
12 implicacao(pi,  $\bar{u}$ );
13 resultado = PODEM-Recursivo(C);
14 se resultado == SUCESSO então
15     retorne SUCESSO;
16 fim_se
17 /* A última decisão gerou um conflito, o valor da PI é inicializado */
18 implicacao(pi, X);
19 retorne FALHA;

```

Figura 48 – Procedimento recursivo do algoritmo PODEM

Se em alguma decisão a falha não foi possível de ser ativada ou propagada, o mecanismo de *backtracking* inverte o valor da última decisão tomada. Caso essa reversão continue gerando um conflito, então o algoritmo recursivo desfaz a última decisão tomada e o mecanismo de *backtracking* é chamado para uma decisão anterior, até que não seja possível fazer mais inversões e, conseqüentemente, a falha é considerada não testável.

Na Fig. 48 mostra que o algoritmo *PODEM-Recursivo*() possui três importantes funções: *objetivo*(), *backtrace*() e *implicacao*(). A seguir, essas três funções são explicadas mais detalhadamente.

A função *objetivo*() retorna o próximo objetivo que o algoritmo deve tentar justificar, ou seja, retorna a linha a ser justificada e o valor a ser justificado nessa linha. Antes de a falha ser ativada, o objetivo a ser retornado é simplesmente a linha com a falha e o valor oposto ao valor da falha. Após a falha ser ativada, o procedimento *objetivo*() seleciona o sinal de erro com melhor chance de se propagar dentre as linhas que estão na fronteira-*D*. A escolha do melhor caminho até uma PO é feita baseada nos valores de observabilidade das linhas que estão na fronteira-*D*. A Fig. 49 mostra a função *objetivo*() em pseudocódigo.

```

                                Objetivo(C)
1  se a falha não foi ativada então
2    retorne (g,  $\bar{v}$ );
3  fim_se
4  d = a linha mais observável da fronteira-D;
5  g = a entrada menos controlável de d que esteja com valor X;
6  v = valor não-controlante de d;
7  retorne (g, v);

```

Figura 49 – Pseudocódigo do procedimento *objetivo*()

A função *backtrace*() encontra o melhor caminho entre a linha que se deseja justificar e uma PI do circuito. Após sua execução, ela retorna a PI e o valor a ser assinalado. Então, o *backtrace*() nunca passa por uma linha já justificada no circuito. Para determinar o melhor caminho até uma das PIs, a função *backtrace*() utiliza uma heurística baseada nos valores de controlabilidade das linhas do circuito. Para justificar um valor controlado na saída de uma porta lógica é necessário assinalar o valor controlante em apenas uma das entradas da porta. Então, é selecionada a entrada da porta mais fácil de ser controlada, pois o seu valor controla o valor da saída da porta. Para justificar um valor não-controlado na saída de uma porta é selecionada a entrada da porta mais difícil de ser controlada para o valor não-controlante, pois é necessário que se assinale o valor não-controlante em todas as entradas da porta. Esta heurística descrita acima é chamada de **heurística easy/hard**. Na Fig. 50 é mostrado o algoritmo *backtrace*() em pseudocódigo.

```

                                backtrace( $g, v$ )
1  se  $g$  é uma PI então
2    retorne ( $g, v$ );
3  fim_se
4  se  $v$  é o valor controlado de  $g$  então
5    seleciona a entrada  $a$  com valor  $X$  mais controlável;
6     $u$  = valor controlante de  $g$ ;
7  fim_se
8  se  $v$  é o valor não controlado de  $g$  então
9    seleciona a entrada  $a$  com valor  $X$  menos controlável;
10    $u$  = valor não-controlante de  $g$ ;
11 fim_se
12 backtrace( $a, u$ );

```

Figura 50 – Algoritmo recursivo em pseudocódigo do procedimento *backtrace*()

Finalmente a função *implicacao*() é simplesmente uma rotina que realiza uma simulação lógica, ou seja, as portas lógicas do circuitos são processadas de acordo com os valores de suas entradas. Essa etapa de implicação é uma característica particular do algoritmo PODEM entre os demais algoritmos de ATPG pois a etapa de implicação se dá apenas no sentido para as saídas primárias do circuito, em função dos valores assinalados nas entradas primárias.

Conforme os resultados apresentados por Goel (1981), o algoritmo PODEM mostrou um tempo de execução menor que o algoritmo-*D*, principalmente para circuitos com maior número de portas lógicas.

5.3 Resultados Experimentais

A fim de avaliar o desempenho do método proposto por este trabalho, um protótipo de ferramenta foi descrito e implementado em linguagem C. O protótipo, batizado de GILDA (*GACI's Implementation of Latching-window and D propagation Analysis*), possui três modos de execução:

- **LW**: executa somente o pré-processamento baseado em TTA, deste modo analisando apenas o mascaramento por *latching-window*;
- **LO**: executa apenas a análise do circuito baseada no algoritmo PODEM, analisando, desta forma, apenas o mascaramento lógico;

- **ALL (LW + LO):** Execução do pré-processamento baseado em TTA e, em seguida, da análise da propagação baseada no algoritmo PODEM. Logo, este modo realiza tanto a análise do mascaramento por *latching-window* quanto a análise do mascaramento lógico.

O protótipo de ferramenta recebe como dados de entrada o circuito a ser analisado, descrito em formato SPICE, e o modo de execução que deve ser realizado. Caso seja escolhido os modos **LW** ou **LW+LO** devem ser informados também as características $\{t_i, d\}$ do pulso a ser injetado.

A implementação da análise do mascaramento lógico obedece uma seqüência de etapas muito semelhante às etapas realizadas pelo pré-processamento:

- 1) Leitura do arquivo de descrição do circuito (*netlist*);
- 2) Criação do DAG, utilizando o procedimento descrito no capítulo 4;
- 3) Aplicação do algoritmo PODEM.

As estruturas de dados descritas no capítulo 4 tiveram que ser modificadas para a execução do algoritmo PODEM. Foi inserido na estrutura *GRAPH_NODE* um dado extra que contém a função lógica realizada pela porta lógica. Tal função é obtida a partir da análise da topologia dos transistores da porta lógica.

A estrutura *EDGE_NODE* foi modificada a fim de poder conter as informações das medidas de testabilidade de cada linha do circuito e os valores lógicos de cada linha. Para a análise do mascaramento lógico baseada no algoritmo PODEM são utilizadas as medidas de testabilidade probabilísticas (BRGLEZ; POWNALL; HUM, 1984). A Fig. 51 mostra a estrutura *EDGE_NODE* utilizada.

Os campos da estrutura *EDGE_NODE* representam as seguintes características da linha do circuito:

- **fanin:** aponta para o nodo de entrada na aresta (linha).
- **fanout:** aponta para o nodo que recebe a aresta (linha) como entrada.
- **PC[2]:** armazena o valor da controlabilidade probabilística da linha do circuito, onde **PC[0]** e **PC[1]** representam a controlabilidade probabilística para 0 e a controlabilidade probabilística para 1, respectivamente.
- **Logic_Status:** contém o valor lógico da linha do circuito.

EDGE_NODE
fanin
fanout
PC[2]
PO[2]
Logic_Status

Figura 51 – Estrutura de uma aresta do grafo

A tab. 7 mostra os resultados da taxa de falhas (SER – *soft-error rate*) dos circuitos do conjunto *benchmark* MCNC, para a execução do protótipo de ferramenta executando pré-processamento e, em seguida, a análise do mascaramento lógico. Os circuitos foram todos mapeados apenas para portas NANDs e NORs de duas entradas e inversores. Para a execução do pré-processamento foram injetados pulsos com duração de 100ps e com 10 diferentes **tis** (momentos de injeção do pulso) $\{0xD, 0,1xD, 0,2xD, 0,3xD, 0,4xD, 0,5xD, 0,6xD, 0,7xD, 0,8xD, 0,9xD\}$ onde **D** representa o atraso crítico topológico do circuito. Como pode ser visto na tab. 7, quanto maior o atraso do momento de injeção do pulso transiente com relação ao início do ciclo de relógio, maior é o SER do circuito. Isso é justificado pelos resultados obtidos apenas com o pré-processamento que mostraram essa mesma tendência.

Para analisar a importância de considerar o mascaramento lógico na análise da propagação de pulsos transientes oriundos de SETs foram feitas comparações entre o SER do circuito obtido pelo pré-processamento e o SER obtido pelo pré-processamento mais a análise do mascaramento lógico. As Figs. 52 e 53 mostram os resultados para os circuitos alu4 e c432 respectivamente. É possível perceber que, para o caso do circuito alu4, a consideração do mascaramento lógico na análise da propagação do pulso transiente gera uma redução do SER do circuito de até 30% do SER obtido pela análise somente do mascaramento por *latching-window*. Além disso, como pode ser visto nos gráficos das Figs. 52 e 53 essa redução independe do momento de injeção do pulso.

Tabela 7 – SER dos circuitos utilizando o modo LW + LO de GILDA utilizando diferentes TIs

Circuito			Resultados de SER com pulsos em diferentes TIs (em %)									
Nome	# portas	# entradas	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%
C17	13	5	42,31	42,31	42,31	42,31	42,31	65,38	65,38	65,38	65,38	65,38
bw	279	5	6,63	24,19	24,19	43,01	43,01	56,45	60,57	60,57	75,63	75,63
C432	297	36	11,11	28,79	30,13	41,25	46,97	49,33	54,38	59,43	62,29	63,3
9sym	331	9	9,67	36,56	54,08	60,27	66,16	67,67	67,82	68,43	69,64	69,79
C880	512	60	19,53	37,3	44,82	51,07	56,35	62,3	65,33	66,7	68,46	70,31
C499	638	41	11,91	23,35	32,68	36,52	42,95	48,35	54,62	65,28	67,95	80,88
C1908	696	33	8,24	22,01	31,2	37,24	42,35	46,5	55,39	65,31	70,63	75,36
alu2	704	10	5,68	37,43	45,38	49,15	55,89	58,88	63,07	66,55	67,47	67,76
C1355	709	41	16,01	29,62	33,22	35,05	38,43	40,69	45,06	58,04	61,42	63,96
C2670	1128	157	17,51	23,8	31,69	47,07	54,3	63,52	67,91	70,57	71,76	73,01
C3540	1525	50	7,7	33,48	47,21	54,36	58,92	62,56	65,87	67,44	69,44	70,03
T481	2201	16	3,04	15,92	35,98	54,57	64,56	65,61	69,92	71,85	72,56	72,76
C6288	2624	32	3,22	19,28	26,71	32,89	38,64	44,34	50,72	54,92	58,96	60,77
C5315	2967	178	10,09	42,69	54,5	62,99	67,66	71,13	71,92	72,28	72,87	73,09
C7552	3459	206	8,17	31,05	48,53	56,85	61,78	67,04	69,89	71,02	71,36	71,55
alu4	5796	14	0,71	23,37	44,16	58,59	58,61	65,5	68,57	68,57	69,44	69,64

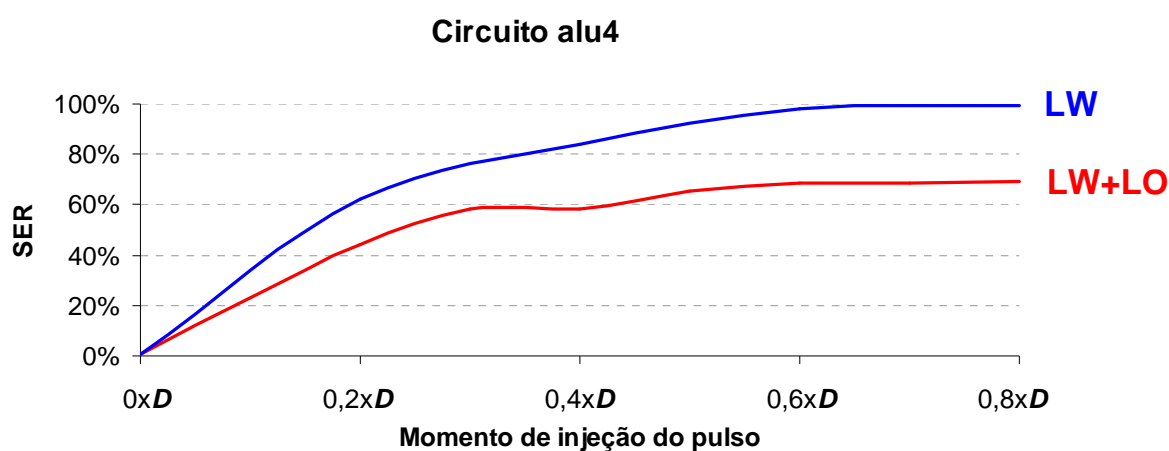


Figura 52 – Resultados comparativos entre a análise considerando somente mascaramento por *latching-window* (pré-processamento) e análise considerando mascaramento lógico para o circuito alu4

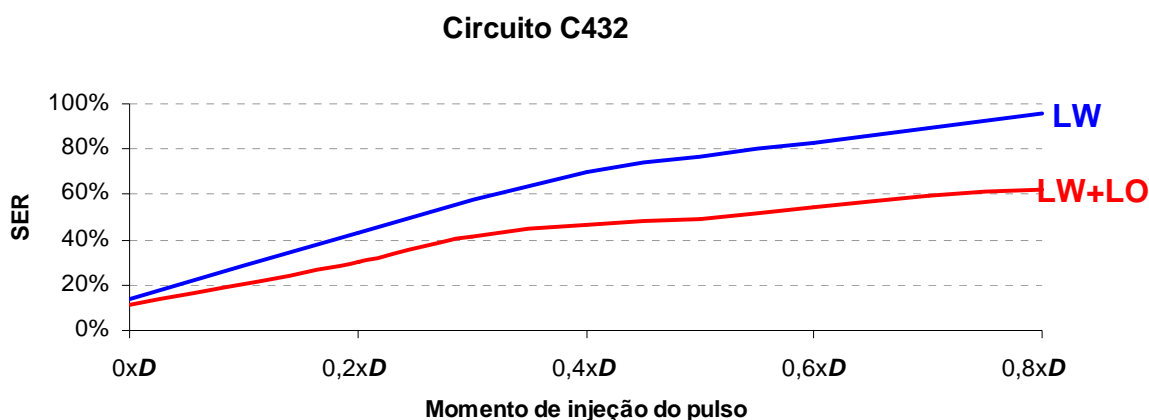


Figura 53 – Resultados comparativos entre a análise considerando somente mascaramento por *latching-window* (pré-processamento) e análise considerando mascaramento lógico para o circuito C432

A tab. 8 mostra os resultados de tempo de execução para os circuitos utilizando os três modos de execução do protótipo de ferramenta GILDA. Para a execução dos modos **LW** e **LW+LO**, foram utilizadas as características $\{t_i = 0xD, d = 100ps\}$. Os tempos de execução são para o protótipo de ferramenta GILDA rodado em um computador equipado com processador AMD Turion 2.0GHz de 64 bits com 1 GB de memória RAM.

Tabela 8 – Comparação dos tempos de execução para os 3 modos de GILDA

Circuito	Fanout médio	Fanout máximo	Prof. Lógica	Número de portas	Atraso Crítico (ns)	Tempo de execução de LW(s)	Tempo de execução de LO(s)	Tempo de execução de LW+LO(s)
C17	1,23	2	5	13	20,5	0,003	< 0,001	0,015
bw	1,68	4	13	279	42,2	0,09	0,234	0,093
C432	1,44	4	40	297	116,0	0,096	0,406	0,156
9sym	1,7	4	20	331	83,9	0,105	0,812	0,203
C880	1,5	4	36	512	126,2	0,228	0,859	0,39
C499	1,48	4	31	638	167,5	0,312	1,578	0,515
C1908	1,56	4	45	686	199,2	0,35	1,296	0,453
alu2	1,7	4	43	704	136,6	0,371	3,421	0,609
C1355	1,72	4	34	709	136,3	0,381	0,859	0,484
C2670	1,49	4	33	1128	94,4	0,953	5,171	2,015
C3540	1,66	4	58	1525	251,3	1,565	9,546	2,468
T481	1,72	4	26	2201	83,85	3,354	89,593	11,468
C6288	1,9	4	127	2624	498,7	5,534	37,89	7,296
C5315	1,62	4	56	2967	261,9	6,706	28,171	10,703
C7552	1,6	4	48	3459	219,9	9,627	62,687	16,89
alu4	1,73	4	25	5796	73,7	30,618	1287,593	50,953

Analisando os resultados da tab. 8 pode-se perceber que somente para o circuito C17 que análise somente do mascaramento lógico foi mais rápida que a análise do pré-processamento. Isto se deve ao tamanho do circuito ser muito pequeno (possui apenas 13 portas lógicas). Por outro lado, para circuitos grandes, como por exemplo o circuito alu4, a redução do tempo de execução foi de mais de **15 vezes**. Cabe salientar que esses resultados são para uma caracterização apenas do pulso transiente com duração de 100ps. Para uma caracterização mais completa do circuito é necessário analisar a propagação de pulsos com diversos tempos de início do pulso e diversas durações.

Além disso, para o caso do circuito alu4, onde o tempo de execução foi o maior, a ferramenta GILDA levou pouco mais de 50 segundos para a análise da propagação de SETs. Assim, o método proposto por este trabalho realiza uma análise bastante rápida do circuito proporcionando uma boa precisão nos resultados.

Para uma melhor análise dos resultados obtidos pelo método proposto, o tempo de execução e o SER obtido por GILDA foram comparados com o tempo de execução e o SER obtido pela ferramenta ASPA (NEVES, 2006) (descrita na seção 3.4). Os resultados comparativos podem ser vistos na tab. 9. As características do pulso transiente utilizadas foram $\{t_i = 0xD, d = 100ps\}$.

Tabela 9 – Comparação entre o modo LO da ferramenta GILDA com a ferramenta ASPA em tempo de execução e SER resultante

Circuito			ASPA		GILDA	
Nome	Total de portas	Total de Pls	SER (em %)	Tempo de execução (s)	SER (em %)	Tempo de execução (s)
C17	13	5	61,54	0,052	76,92	< 0,001
bw	279	5	78,9	1,116	78,32	0,234
C432	297	36		> 5 horas	65,15	0,406
9sym	331	9	75,23	1,324	70,24	0,812
C880	512	60		> 5 horas	72,66	0,859
C499	638	41		> 5 horas	84,09	1,578
C1908	696	33		> 5 horas	75,95	1,296
alu2	704	10	74,86	2,816	69,32	3,421
C1355	709	41		> 5 horas	65,87	0,859
C2670	1128	157		> 5 horas	73,36	5,171
C3540	1525	50		> 5 horas	70,56	9,546
T481	2201	16		> 5 horas	72,94	89,593
C6288	2624	32		> 5 horas	61,26	37,89
C5315	2967	178		> 5 horas	73,64	28,171
C7552	3459	206		> 5 horas	71,73	62,687
alu4	5796	14		> 5 horas	69,82	1287,593

Como pode ser visto na tab. 9, a ferramenta ASPA não conseguiu realizar a análise da maioria dos circuitos em menos de 5 horas de execução. Como a ferramenta ASPA realiza a análise da propagação de SETs baseada em análise topológica, então o seu tempo de execução cresce exponencialmente com o aumento do número de entradas do circuito. Essa característica é perceptível através dos resultados obtidos, visto que ASPA conseguiu analisar em menos de 3 segundos o circuito alu2 que possui 704 portas lógicas e 10 entradas primárias, mas não conseguiu analisar em menos de 5 horas o circuito C432 que possui 297 portas lógicas e 36 entradas primárias. Os resultados apresentados mostram que o método proposto é eficiente para a análise da propagação de SETs em circuitos combinacionais.

Com relação ao SER obtido por ASPA e por GILDA, é possível perceber uma semelhança nos resultados. A maior diferença foi no circuito C17 que possui 13 portas, onde a diferença foi de 15%. Essa semelhança deve-se ao fato de tanto a ferramenta ASPA quanto o protótipo de ferramenta GILDA, utilizando o modo LO, realizarem apenas a análise do mascaramento lógico do pulso transiente. Porém, as diferenças apresentadas resultam do fato de a ferramenta ASPA realizar uma análise **dinâmica** da propagação do pulso transiente, enquanto que o método proposto por este trabalho realiza uma análise **estática** da propagação. Assim, alguns pulsos que em uma análise estática são considerados bloqueados pela lógica, em uma análise dinâmica esse bloqueio pode não acontecer, ou vice-versa.

6 Conclusões

Este trabalho propôs uma análise da propagação do pulso transiente oriundo da colisão de partículas carregadas, fenômeno esse referenciado por SET (*single-event transient*), considerando o mascaramento *latching-window* e o mascaramento lógico. Para tanto, foi feita uma análise investigativa do efeito da radiação em circuitos combinacionais desenvolvidos em tecnologias CMOS nanométricas. Também foram estudadas diversas técnicas estado-da-arte para análise da propagação do pulso transiente oriundo da ocorrência de SETs. Ademais, os aspectos mais relevantes sobre análise de *timing* e sobre algoritmos de ATPG (*Automatic Test-Pattern Generation*) foram apresentados.

O método proposto por este trabalho consiste em duas etapas:

- 1) Realizar um pré-processamento considerando as condições temporais da propagação de um pulso transiente pela lógica combinacional;
- 2) Analisar os possíveis bloqueios sofridos pelos pulsos transientes devido à lógica do circuito combinacional.

A primeira etapa realiza uma análise do circuito considerando as características temporais do pulso transiente e as condições temporais para a captura de um sinal lógico por um elemento de memória. Tal pré-processamento proposto é baseado em análise de *timing* topológica (TTA – *Topological Timing Analysis*), a qual originalmente foi proposta para o cálculo do atraso crítico topológico de um circuito. O pré-processamento foi descrito no capítulo 4 e os resultados a respeito de SER (*soft-error rate*) dos circuitos sob teste foram apresentados. Os circuitos-alvo deste trabalho foram os circuitos do conjunto *benchmark* MCNC, mapeados para a tecnologia CMOS com o uso de NAND, NOR e inversores. Os resultados práticos mostraram que o pré-processamento apresentou

baixos tempos de execução (pouco mais de **30 segundos** para o circuito alu4 que possui **5.796 portas lógicas**) para os circuitos-alvo e, portanto, justificando a sua utilidade como uma técnica de pré-processamento para uma análise da propagação de SETs.

A segunda etapa do método proposto consiste na análise da propagação do pulso transiente considerando as condições lógicas para essa propagação. Para isso foram estudados os algoritmos de ATPG para implementação dessa análise. Tais algoritmos têm por objetivo encontrar um conjunto de valores (cubo ou vetor de entrada) que aplicado às entradas do circuito conseguem testar uma determinada falha do circuito. Dentre os diversos algoritmos de ATPG, foi escolhido o algoritmo PODEM (*Path-Oriented DEcision Making*) que realiza o processamento do circuito tomando decisões apenas nas entradas primárias do circuito. Os conceitos fundamentais dos algoritmos de ATPG e a descrição detalhada do algoritmo PODEM são apresentados no capítulo 5.

Para avaliação do método proposto foi implementado um protótipo de ferramenta batizado de GILDA (*GACI's Implementation of Latching-window and D propagation Analysis*), conforme descrito no capítulo 5. Os resultados apresentados mostraram que o método proposto é capaz de realizar a análise da propagação do pulso transiente em circuitos com um número significativo de entradas e de portas (por exemplo, circuito alu4 que possui 5796 portas lógicas) em um tempo razoável (pouco mais de 50 segundos).

Os resultados apresentados mostraram que o mascaramento lógico deve ser levado em consideração na análise da propagação do pulso transiente pois, caso contrário, a análise pode sobrestimar em 30% o SER do circuito, desconsiderando as características temporais do pulso transiente. Além disso, a análise dos circuitos *benchmark* MCNC considerando **apenas o mascaramento lógico** resultou em um SER médio entre os circuitos de **cerca de 72%**. Isso significa que, **independentemente** das características do pulso transiente oriundo da ocorrência de um SET, **28%** das possíveis falhas do circuito não atingirão os registradores das saídas primárias do circuito e, então, não devem ser levadas em conta na análise da suscetibilidade do circuito. Essa redução implica em uma **aceleração de 38,9%** na análise completa do circuito.

Este trabalho contribui para um tema bastante atual que tem se tornando um dos principais assuntos de pesquisas na área, uma vez que os circuitos mais

suscetíveis a falhas transientes são os fabricados em tecnologia estado-da-arte. O método proposto por este trabalho explora diversas áreas estudadas ao longo do curso e assuntos abordados durante os três anos de iniciação científica junto ao GACI (Grupo de Arquiteturas e Circuitos Integrados) – UFPel. Este trabalho também proporcionou um avanço nos estudos realizados no GACI – UFPel sobre falhas transientes geradas por SETs.

Como trabalhos futuros pretende-se comparar os resultados acerca de tempo de execução e SER estimado do método proposto com um simulador de nível lógico comercial, tal como o ModelSim da Mentor Graphics (MENTOR, 2007).

Pretende-se também implementar um protótipo de ferramenta que use solvabilidade (SAT) para a análise do mascaramento lógico. Um estudo prévio sobre SAT já está em andamento.

Como trabalho futuro pretende-se analisar os circuitos sintetizados automaticamente, buscando detectar o efeito dos parâmetros de síntese no SER dos circuitos e encontrar partes do circuitos mais suscetíveis a falhas a fim de projetar circuitos mais robustos a falhas transientes.

7 Referências Bibliográficas

ABRAMOVICI, M.; BREUER, M.; FRIEDMAN, A. **Digital Systems Testing and Testable Design**. Piscataway, NJ: IEEE Press, 1990. 652p.

ALEXANDRESCU, D.; ANGHEL, L.; NICOLAIDIS M. New Methods for Evaluating the Impact of Single-Event Transients in VDSM ICs. In: IEEE INTERNATIONAL SYMPOSIUM ON DEFECT AND FAULT TOLERANCE IN VLSI SYSTEMS, 17. **Proceedings...** Vancouver, Canada, 2002, p. 99-107.

ALEXANDRESCU, D.; ANGHEL, L.; NICOLAIDIS M. Simulating Single Event Transients in VDSM ICs for Ground Level Radiation. **Journal of Electronic Testing: theory and applications**, v. 20, p. 413-421, 2004.

BAUMANN, Robert C. Soft Errors in Advanced Semiconductor Devices – Part I: The Three Radiation Sources. **IEEE Transactions on Device and Materials Reliability**. v. 1, n. 1, p. 17-22, março de 2001.

BAUMANN, Robert C. Radiation-Induced Soft Errors in Advanced Semiconductor Technologies. **IEEE Transactions on Device and Materials Reliability**. v. 5, n. 3, p. 305-316, setembro de 2005.

BAZE, M.; BUCHNER, S. Attenuation of Single Event Induced Pulses in CMOS Combinational Logic. **IEEE Transactions on Nuclear Science**. v. 44, n. 6, dezembro de 1997.

BRGLEZ, F.; POWNALL, P.; HUM, P. Applications of Testability Analysis: From ATPG to Critical Path Tracing. IEEE INTERNATIONAL TEST CONFERENCE, 1984. Philadelphia, outubro de 1984. **Proceedings...** Los Alamitos: IEEE Computer Society, 1984. p. 705-712.

BRYANT, R. Graph-based Algorithms for Boolean Function Manipulation. **IEEE Transactions on Computers**. v. 35, n. 8, p. 677-691, agosto de 1986.

BUSHNELL, M. L.; AGRAWAL, V. D. **Essentials of Electronic Testing for Digital, Memory, and Mixed-Signal VLSI Circuits**. Kluwer Academic Publishers, 2000. 690p.

CAO, Y.; SATO, T.; ORSHANSKY, M.; SYLVESTER, D.; HU, C. New Paradigm of Predictive MOSFET and Interconnect Modeling for Early Circuit Simulation. In: IEEE CUSTOM INTEGRATED CIRCUITS CONFERENCE, 2000. **Proceedings...** p. 201–204, maio de 2000.

CHA, H et al. A Gate-Level Simulation Environment for Alpha-Particle-Induced Transient Faults. **IEEE Transactions on Computers**. v.45, n.11, novembro de 1996.

COHEN, N.; SRIRAM, T. S.; LELAND, N.; MOYER, D.; BUTLER, S.; FLATLEY, R. Soft error considerations for deep-submicron CMOS circuit applications. In: INTERNATIONAL ELECTRON DEVICES MEETING, 1999. **IEDM Technical Digest**, p. 315-318, 1999.

CORMEM, T. H.; LEISERSON, C. E.; RIVEST, R. L. **Introduction to Algorithms** MIT, Cambridge, 1990.

DAHLGREN, P.; LIDÉN, P. Efficient Modeling of Switch-Level Networks Containing Undetermined Logic Node States. In: IEEE/ACM INTERNATIONAL CONFERENCE ON CAD, 1993. **Proceedings...** p. 746-752, 1993.

DAHLGREN, P.; LIDÉN, P. A Switch-Level Algorithm for Simulation of Transients in Combination Logic. In: IEEE FAULT TOLERANT COMPUTING SYMPOSIUM, 1995. **FTCS-25 Digest of Papers**, p. 207-216, 1995.

DEVADAS, S.; KEUTZER, K.; MALIK, S. Computation of Floating Mode Delay in Combinational Circuits: Theory and Algorithms. **IEEE Transactions on Computer-Aided Design**, v.12, n.12, p.1913-1923, dezembro de 1993.

DHILLON, Y. S.; DIRIL, A. U.; CHATTERJEE, A.; SINGH, A. D. Analysis and Optimization of Nanometer CMOS Circuits for Soft-Error Tolerance. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, v.14, n.5, p.514-524, maio de 2006.

DODD, P. E.; MASSENGILL, L. W. Basic Mechanisms and Modeling of Single-Event Upset in Digital Microelectronics. **IEEE Transactions on Nuclear Science**, v.50, n.3, p.583–602, junho de 2003.

GADLAGE, M. G. et al. Single Event Transient Pulsewidths in Digital Microcircuits. **IEEE Transactions on Nuclear Science**, v. 51, n. 6, p. 3285-3290, dezembro de 2004.

GOEL, P. An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits. **IEEE Transactions on Computers**, Piscataway, NJ, v. C-30, n. 3, p. 215-222, março de 1981.

GOLDSTEIN, Lawrence H. Controllability/Observability Analysis of Digital Circuits. **IEEE Transactions on Circuits and Systems**, v. CAS-26, n. 9, p. 685-693, setembro de 1979.

GÜNTZEL, José Luís A. **Functional Timing Analysis of VLSI Circuits Containing Complex Gates**. Tese de Doutorado em Ciência da Computação. Porto Alegre: Instituto de Informática da Universidade Federal do Rio Grande do Sul, dezembro de 2000.

HEIJMEN, T.; NIEUWLAND, A. Soft-Error Testing of Deep-Submicron Integrated Circuits. In: 11TH IEEE EUROPEAN TEST SYMPOSIUM. Southampton, UK, maio de 2006. **Proceedings...** Los Alamitos: IEEE Computer Society, 2006. p.247-252.

MA, H.-K. T.; DEVADAS, S.; SANGIOVANNI-VINCENTELLI, A.; WEI, R. "Logic Verification Algorithms and their Parallel Implementation," **IEEE Transactions on Computer-Aided Design**, v. 8, n. 2, p. 181–189, fevereiro de 1989.

MATEUS, Gustavo. **Caracterização Automática de Atrasos de Portas Lógicas em Circuitos Combinacionais CMOS**. Trabalho de Conclusão de Curso. Pelotas: UFPel, outubro de 2006.

MENTOR Graphics Corporation, "Technology Support for Altera Devices", Disponível em: <<http://www.mentor.com>>, Acesso em Jan. 2007.

MESSENGER, C. Collection of Charge on Junction Nodes from Ion Tracks. **IEEE Transactions on Nuclear Science**. v. NS-29, n. 6, p. 2024-2031, dezembro de 1982.

NAGEL, W. **SPICE2, A Computer Program to Simulate Semiconductor Circuits**. Berkeley, California: University of California, Department of Electrical Engineering and Computer Sciences, 1975. 63p. (UCB/ERL M75/520).

NEVES, Carolina. **Análise Automática da Propagação Single Events Transients em Circuitos Combinacionais CMOS**. Trabalho de Conclusão de Curso. Pelotas: UFPel, abril de 2006.

NEVES, C.; RIBEIRO, I.; HENES, E.; WIRTH, G.; KASTENSMIDT, F. L.; GÜNTZEL, J. L.; Avoiding Circuit Simulation in Single Event Transient Propagation Analysis in Combinational Circuits. In: 11TH IEEE EUROPEAN TEST SYMPOSIUM. Southampton, UK, maio de 2006. **Digest of Papers...** Southampton: Univ. of Southampton, 2006. p. 46-51.

NICOLAIDIS, M. Design for Soft Error Mitigation. **IEEE Transactions on Devices and Materials Reliability**. v. 5, n. 3, p. 405-418, setembro de 2005.

NIEUWLAND, A.; JASAREVIC, S.; JERIN, G. Combinational Logic Soft Error Analysis and Protection. In: 12TH IEEE INTERNATIONAL ON-LINE TESTING SYMPOSIUM. Lake of Como, Italy, julho de 2006. **Proceedings...** Los Alamitos, IEEE Computer Society, 2006. p. 99-104.

O'BRYAN, Martha V. et al. Current Single Event Effects and Radiation Damage Results for Candidate Spacecraft Electronics. In: IEEE RADIATION EFFECTS DATA WORKSHOP, 2002. **IEEE Radiation Effects Data Workshop Proceedings**. p. 82-105, 2002.

OMAÑA, M. et al. A Model for Transient Fault Propagation in Combinatorial Logic. In: 9th IEEE INTERNATIONAL ON-LINE TESTING SYMPOSIUM, 2003. **Proceedings..** Los Alamitos, 2003, p. 111-115.

ROTH, J. P. Diagnosis of Automata Failures: A Calculus and a Method. **IBM Journal of Research and Development**, Armonk, v.10, n. 4, p. 278-291, julho de 1966.

SCHENEIDER, R. R. On the Necessity to Examine D-Chains in Diagnostic Test Generation. **IBM Journal of Research and Development**, v.11, n.1, p.114, janeiro de 1967.

SHIVAKUMAR, P. et al. Modeling the Effect of Technology Trends on the Soft Error Rate of Combinational Logic. In: INTERNATIONAL CONFERENCE ON DEPENDABLE SYSTEMS AND NETWORKS, 2002. **Proceedings...** p.389-398.

SPICE Opus: SPICE with integrated OPTimization UtilitieS. Available at <<http://fides.fe.uni-lj.si/spice/>>, 2007.

WANG, Laung-Terng; WU, Cheng-Wen; WEN, X. **VLSI Test Principles and Architectures: Design for Testability**. Elsevier, 2006. 777p.

WIRTH, G., VIEIRA, M., HENES NETO, E., KASTENSMIDT, Fernanda L. Single-Event Transients in Combinational Circuits. In: 16th INTERNATIONAL SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN, 2005. **Proceedings...** ACM: Association for Computing Machinery, New York, 2005. p. 121-126.

ZHANG, B.; WANG, W-S.; ORSHANSKY, M. FASER: Fast Analysis of Soft Error Susceptibility for Cell-Base Designs. In: 7TH INTERNATIONAL SYMPOSIUM ON QUALITY ELECTRONIC DESIGN. San Jose, março de 2006. **Proceedings...** Los Alamitos: IEEE Computer Society, 2006. p.100-106.