

Felipe de Andrade Neves Lavratti

*Detecção de Defeitos do Tipo
Resistive-Open em SRAM com
o Uso de Lógica Comparadora
de Vizinhaça*

Porto Alegre

2012

Felipe de Andrade Neves Lavratti

*Detecção de Defeitos do Tipo Resistive-Open
em SRAM com o Uso de Lógica Comparadora
de Vizinhança*

Dissertação apresentada ao Programa de Pós-Graduação de Engenharia Elétrica, da Faculdade de Engenharia da Pontifícia Universidade Católica do Rio Grande do Sul, como requisito parcial à obtenção do título de Mestre em Engenharia Elétrica.

Orientador:

Prof. Dr. Fabian Luis Vargas

Co-orientador:

Prof.^a Dr.^a Leticia Maria Bolzani Pöhls

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Porto Alegre

2012

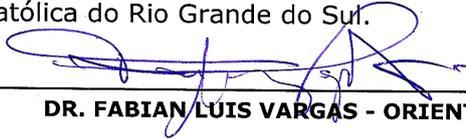


Pontifícia Universidade Católica do Rio Grande do Sul
FACULDADE DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

DETECÇÃO DE DEFEITOS DO TIPO RESISTIVE-OPEN EM SRAM COM O USO DE LÓGICA COMPARADORA DE VIZINHANÇA

CANDIDATO: FELIPE DE ANDRADE NEVES LAVRATTI

Esta Dissertação de Mestrado foi julgada para obtenção do título de MESTRE EM ENGENHARIA ELÉTRICA e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia Elétrica da Pontifícia Universidade Católica do Rio Grande do Sul.



DR. FABIAN LUIS VARGAS - ORIENTADOR



DRA. LETÍCIA MARIA BOLZANI POEHLS - CO-ORIENTADORA

BANCA EXAMINADORA



**DR. RENATO PEREZ RIBAS - DEPARTAMENTO DE INFORMÁTICA APLICADA -
INSTITUTO DE INFORMÁTICA - UFRGS**



DR. RUBEM DUTRA RIBEIRO FAGUNDES - PPGE - FENG - PUCRS

PUCRS

Campus Central
Av. Ipiranga, 6681 - Prédio 30 - Sala 103 - CEP: 90619-900
Telefone: (51) 3320.3540 - Fax: (51) 3320.3625
E-mail: engenharia.pg.eletrica@puhrs.br
www.puhrs.br/feng

RESUMO

O mundo de hoje é cada vez mais dependente dos avanços tecnológicos sendo os sistemas em chip (SoC, do inglês System-on-Chip) um dos principais alicerces desse avanço. Para tanto que a lei de Moore, que previu que a capacidade computacional dos SoCs dobraria a cada ano, já foi ultrapassada. Devido a essa forte demanda por crescimento novas tecnologias surgiram e junto novos modelos de falhas passaram a afetar a confiabilidade dos SoCs.

Os SoCs produzidos nas tecnologias mais avançadas (VDSM - Very Deep Sub-Micron), devido a sua alta integração de transistores em uma área pequena, passaram a apresentar um grande número de interconexões fazendo com que os defeitos do tipo *Resistive-Open*, que ocorrem nessas interconexões, se tornassem os maiores responsáveis por SoCs com defeitos escaparem os testes de manufaturas.

Ainda, segundo projeções da SIA Roadmap, a área consumida pela SRAM será em torno de 95% da área utilizada por um SoC. E sabendo que essas memórias possuem inúmeras interconexões, existe uma grande probabilidade de ocorrer defeitos do tipo *Resistive-Open* em seus circuitos.

Esses defeitos são capazes de causar falhas funcionais do tipo estáticas ou dinâmicas, de acordo com a sua intensidade. As falhas estáticas são sensibilizadas com apenas uma operação e as dinâmicas necessitam de duas ou mais operações para que sejam sensibilizadas.

Os testes de manufatura mais utilizados para aferir a saúde dos SoCs durante o processo de manufatura são hoje ineficientes frente aos defeitos do tipo *Resistive-Open*. O mais comum deles é o *March Test*, que efetua operações de escrita e leitura na memória com o objetivo de sensibilizar falhas e por fim detectá-las, entretanto é ineficiente para detectar as falhas do tipo dinâmicas porque é necessário efetuar mais operações que o tempo disponível permite para que essas falhas sejam sensibilizadas.

Outro teste utilizado durante a manufatura chama-se teste de corrente quiescente (teste de I_{ddq}), este monitora a corrente consumida do SoC como um todo durante a injeção de vetores nos sinais de entrada, o consumo de corrente do chip é comparado com limiares ou outro chip idêntico sob o mesmo teste para detectar defeitos, entretanto não é possível distinguir entre variações inseridas, nos sinais monitorados, pelos defeitos ou pelos *corners*, que são variações nas características dos transistores fruto do processo de manufatura.

E, por fim, o último teste que é apresentado é uma mistura dos dois testes anteriores, utiliza sensores de correntes e algoritmos de operações como em *March Test* onde que o defeito é detectado pelos sensores de corrente embutidos quando a corrente monitorada ultrapassa dado limiar, embora esse teste tenha condições de detectar defeitos que causam falhas dinâmicas e de não sofrerem influência dos *corners*, ele é ineficaz ao detectar defeitos

do tipo *Resistive-Open* que possam ocorrer em qualquer local, com qualquer tamanho de impedância em uma SRAM executando qualquer operação, porque os defeitos do tipo *Resistive-Open* ora aumentam o consumo de corrente e ora o diminui de acordo com essas três características citadas. Comparações por limiares não têm condições de contornar esta dificuldade.

Com tudo isso, o objetivo desta dissertação de mestrado é propor uma técnica de detecção de defeitos que seja capaz de vencer as três limitações dos testes convencionais de manufatura apontadas. Para a tarefa, sensores de corrente são utilizados associadamente com *March Test*, entretanto com o acréscimo de uma Lógica Comparadora de Vizinhança (LCV) que tomará para si a função de detectar defeitos, deixando os sensores apenas encarregados em transformar a corrente analógica em um sinal digital e que tem a capacidade de eliminar a necessidade do uso de limiares, junto com as demais limitações apontadas.

A LCV monitora o comportamento de uma vizinhança células e, comparando-os entre si, acusa aquela ou aquelas células que se comportarem diferentemente das suas vizinhas como defeituosas, desta maneira a referência de comportamento correto é obtida da própria vizinhança durante a execução do teste de manufatura, eliminando a necessidade de conhecimento prévio do tipo de distúrbio causado pelos defeitos do tipo *Resistive-Open*, trazendo facilidade na hora de projetar o sistema de detecção de defeitos e adicionado o poder de detectar qualquer defeito que gere alterações no sinal de corrente consumida das células da SRAM.

Neste contexto, o sensor de corrente tem apenas a função de gerar o sinal digital, que é de 1 bit para cada sinal monitorado (*Vdd* e *Gnd*) e modulado em largura de pulso (PWM), assim a LCV também tem sua complexidade diminuída, pois é constituída por apenas portas lógicas.

A LCV e os sensores de corrente são utilizados durante o teste de manufatura, as comparações que ocorrem na vizinhança são efetuadas paralelamente nas células da memória, então o teste de manufatura necessita efetuar operações de acesso para excitar semelhantemente todas as células que participam da mesma vizinhança. O *March Test* é um teste que efetua operações desta natureza e, portanto, é utilizado para controlar a execução do teste e recolher os dados proveniente da LCV, que contém o resultado da detecção efetuada em cada vizinhança.

A LCV, o sensor de corrente e o *March Test* juntos compõem a técnica de detecção de defeitos proposta nesta dissertação, e foram validados quanto as suas funções para comprovar que operam como projetados.

Por fim, a técnica proposta se mostrou capaz de detectar as 10 *milhões* de células defeituosas (com o defeito mais difícil de detectar que causa falha funcional dinâmica) em uma SRAM de 1Gbit, sem deixar passar nenhuma célula defeituosa pelo teste de manufatura, junto a isso, 294.890 células boas foram desperdiçadas, isto-é, foram dadas como defeituosas enquanto não tinham defeitos, o que representa apenas 0,029% de desperdício. Tudo isso, ao custo de área equivalente a área consumida por 56 células de memória, por coluna monitorada, e ao custo de um teste de manufatura que executa apenas 5 operações em cada linha da SRAM.

ABSTRACT

The world we live today is very dependent of the technology advance and the Systems-on-Chip (SoC) are one of the most important actors of this advance. As a consequence, the Moore's law has been outperformed due to this strong demand on the SoCs for growth, so that new silicon technologies has emerged along with new fault models that decreased the reliability of these devices.

SoCs built using Very Deep Sub-Micron technology have a great number of interconnections, increasing the occurrence of Resistive-Open defects that occur on these interconnections up to the point where Resistive-Open defects have become the most important responsible for defective SoCs escaping the manufacturing tests.

According to SIA Roadmap's projection, the area consumed by the SRAM on the SoC will be around 95% of the available area, knowing these memory have a great number of interconnections there is also a great probability of occurring Resistive-Open defects on the SRAM circuits which will compromise the overall SoC reliability.

When found on SRAMs cells, these defects are able to cause dynamic and static functional faults according to its size, where static faults are sensitized by performing only one operation at the SRAM cell, while dynamic are sensitized by two or more operations.

The most common manufacturing tests used to detect defective SoCs are today unable to detect dynamic faults caused by weak Resistive-Open defects. March test performs access on the memory with the intention of sensitizing the faults and detect them as consequence. Due to the higher number of operations necessary to sensitize dynamics faults, this test is not able to detect them properly.

Another test is the I_{ddq} test, which is able to detect the presence of defects by monitoring the overall current consumption of a SoC while it's being excited by a known vector of data on its inputs. The consumed current is compared to thresholds or to another similar device that is being excited on the same way. I_{ddq} test is not able to distinguish the variations on current caused by process variations or defects presence.

There is an other type of test using On-Chip Current Sensors (OCCS) with March tests that performs current monitoring on the circuits of the SoC and compare them with a threshold in order to set a flag when the monitored current gets higher or lower than a configured thresholds. Because the mentioned test uses threshold, it is not able to detect Resistive-Open defects that could happen in any node, with any size, in the SRAM cell performing any operation. In this scenario the current consumption could be higher or lower than the defectless current consumption of a cell, making impossible to detect defects using thresholds.

By all that, the objective of this dissertation is to propose a defect detection technique able to overcome the three mentioned limitations of preview explained tests. For that,

OCCS are along with March test, but a Neighborhood Comparator Logic (NCL) has been included with the objective to perform the detections itself, removing from the OCCS the mission of finding defects. Now the OCCS is only responsible in converting the monitored current consumption signal to a one bit PWM digital signal.

In this form, no threshold will be required because the NCL will obtain the reference of the correct current consumption (behavior reference) within the SRAM circuits, by comparing the neighboring cells and adopting the most common behavior as the reference one, so that it will detect those cells that behave differently from the reference as defective ones.

The neighborhood's cells are excited in a parallel form by the test processor, which performs a March test algorithm. The NCL, the OCCS and the March test, together, compose the proposed Resistive-Open detection technique, which has been validated on this work.

As result, the proposed technique has shown being able to detect all of the 10 million defective cells of a *1Gbit* SRAM containing the hardest defect to detect (small ones). No defective cell has escaped the simulated test and there was only 294,890 good cells being wasted, which represents 0.029% of the simulated SRAM cells. All of that, by costing only the equivalent to the area of 56 SRAM cells per monitored column and a manufacturing test that performs 5 operations per line of the SRAM.

LISTA DE ALGORITMOS

1	Algoritmo de teste de manufatura MarchSS	28
2	Pseudo-algoritmo do núcleo de comparação da LCV.	44
3	Algoritmo <i>march</i> para ser usado junto a técnica detectora de defeitos do tipo <i>Resistive-Open</i>	49

LISTA DE FIGURAS

1	SRAM simplificada.	22
2	Visão microscópica de um corte transversal de um IC revelando um defeito resistivo em uma interconexão entre o metal 5 e 6 (obtida em [1]).	24
3	Célula 6T com as posições referentes dos defeitos DF1 ao DF6.	25
4	Corrente de V_{dd} e G_{nd} para todos os <i>corners</i> e para DF4=50k Ω , demonstrando o ofuscamento das variações introduzidas pelo defeito quando considerada as variações introduzidas pelos <i>corners</i>	29
5	Corrente de V_{dd} e G_{nd} para a célula boa (área pintada) e para a célula com cada um dos seis defeitos considerados com tamanho de 50k Ω (DF1 a DF6 = 50k Ω).	31
6	<i>Hardware</i> da técnica de detecção de defeitos acoplada ao esquemático simplificado de uma SRAM.	36
7	Diagrama esquemático do sensor de corrente projetado com uma parte para monitorar I_{vdd} e outra para I_{gnd}	39
8	Diagrama esquemático do amplificador operacional projetado para amplificar I_{vdd} e I_{gnd} já convertidos para tensão.	40
9	Diagrama esquemático e representação das correntes do gerador de PWM para os sinais de V_{dd} . O circuito de G_{nd} é análogo.	41
10	Sinais de entrada e saída do sensor utilizando curvas de teste.	42
11	Lógica Comparadora de Vizinhança e os seus circuitos interiores.	43

12	Circuito lógico que implementa a LCV demonstrando o resultado da detecção quando o membro <i>M1</i> se comporta diferentemente dos demais entre o tempo 9 e 10.	45
13	Celula da SRAM projetada.	53
14	Valor armazenado (Bit) e dado de saída para todos os <i>corners</i> durante a rotina escreve 0, lê 0, escreve 1 e lê 1.	54
15	DF1-6 e suas falhas funcionais para diferentes tamanhos de defeito, obtidos via simulações com step de $1k\Omega$ e frequência de operação da memória de $200MHz$	55
16	Representação dos circuitos de detecção na vizinhança 0, onde os sinais (a), (b), (c), (d) e (e) são apresentados adiante para cada caso simulado.	57
17	Ilustração do processamento de sinais referente a todas as etapas de detecção de defeitos do tipo <i>Resistive-Open</i> da proposta para uma vizinhança onde apenas o membro de índice 6 é defeituoso.	61
18	Ilustração do processamento de sinais referente a todas as etapas de detecção de defeitos do tipo <i>Resistive-Open</i> da proposta para uma vizinhança onde os membros de índices 4 e 6 são defeituoso.	63
19	Ilustração do processamento de sinais referente a todas as etapas de detecção de defeitos do tipo <i>Resistive-Open</i> da proposta para uma vizinhança onde não há membro defeituoso.	65
20	Fluxograma das operações do algoritmo simulador de Monte Carlo, cujo código encontra-se no APÊNDICE A.	71

LISTA DE TABELAS

1	Saídas da LCV e o resultado da detecção.	47
2	Diferentes tamanhos de vizinhança e os custos e benefícios de cada uma. Sendo os custos compostos pelo número e tamanho das comparações e os benefícios pela confiabilidade da referência obtida normalizada.	48
3	Capacidade de detecção da técnica para as 6 posições e 2 tamanhos de defeitos.	67
4	Mapeamento entre a situação de detecção em uma vizinhança e os resulta- dos quanto às células da memória.	70
5	Desempenho de detecção para $P_{cdf} = 1\%$, $\#L_{def} = 6$ e $\#T_{def}$ variável. . . .	72
6	Desempenho de detecção para P_{cdf} variável, $\#L_{def} = 6$ e $\#T_{def} = 5$	73
7	Desempenho estimado da técnica em uma memória de 1Gbit considerando $P_{cdf} = 1\%$ e $\#T_{def} = 5$	73
8	Custo de área da técnica proposta.	74
9	Tempo de execução do teste de manufatura proposto comparado a testes existentes utilizando como referência a SRAM estudo de caso.	75
10	Desempenho da LCV para as variações <i>inter-die</i>	77

SUMÁRIO

1	Introdução	14
1.1	Objetivos do Trabalho	17
1.2	Métodos e Métricas	18
2	Fundamentos Teóricos	21
2.1	Experimentos de Monte Carlo	21
2.2	Memória Estática	22
2.3	Defeitos do Tipo <i>Resistive-Open</i>	24
2.4	Incidência de defeitos na SRAM	25
2.4.1	Modelamento na SRAM	25
2.4.2	Falhas funcionais	25
2.5	Testes de Manufatura para Detecção de Defeitos do Tipo <i>Resistive-Open</i>	27
2.5.1	<i>March Test</i>	27
2.5.2	<i>I_{ddq} test</i>	28
2.5.3	OCCS associado ao <i>March test</i>	30
2.6	Desafios para Detectar os Defeitos do tipo <i>Resistive-Open</i> em SRAM	31
3	Proposta	33
3.1	A técnica de Detecção de Defeitos	33

3.1.1	Estrutura de detecção	35
3.1.2	Sensor de Corrente	38
3.1.3	Lógica Comparadora de Vizinhaça	42
3.1.3.1	Núcleo de Comparação	43
3.1.3.2	Saída da LCV	46
3.1.3.3	Tamanho da Vizinhaça	46
3.1.4	Teste de Manufatura Proposto	48
4	Validação	51
4.1	Memória Estudo de Caso	51
4.1.1	Injeção de Defeitos na SRAM Estudo de Caso	54
4.2	Validação da Técnica Proposta	56
4.2.1	Condição 1	57
4.2.2	Condição 2	58
4.2.3	Condição 3	59
5	Resultados Experimentais	66
5.1	Capacidade de Detecção	66
5.2	Desempenho	67
5.2.1	Situações de Detecção	69
5.2.2	Simulador de Monte Carlo	70
5.2.3	Desempenho Estimado da Técnica	72
5.3	Custo	74

5.4	Considerações Quanto às Variações de Processo	75
5.4.1	Variações na SRAM	76
5.4.2	Variações no Sensor de Corrente	76
5.4.3	Variações na LCV	76
6	Conclusões	78
6.1	Trabalhos Futuros	80
	Referências	81
	Apêndice A – Código Fonte do <i>Software</i> simulador de Monte Carlo	84

1 INTRODUÇÃO

Hoje vivemos em um mundo tecnologicamente dependente, de maneira que se novas invenções pararem de aparecer as nossas economias frearão seu desenvolvimento por consequência, pois o processo de desenvolvimento econômico se dá pelas novas economias e novos mercados que são criados a partir de tecnologias disruptivas, como é afirmado no livro *The Innovator's Dilemma* [2]. Esse termo define tecnologias inovadoras de menor desempenho que geram produtos de menor preço total em relação aos já existentes, possibilitando que novos mercados surjam como consequência. A exemplo do mercado de discos rígidos, que no início eram comercializados apenas para computadores centrais (*mainframes*), então tecnologias disruptivas surgiram, gerando discos rígidos de menor preço de venda só que sem o desempenho necessário para participar do mercado de computadores centrais, mas que possibilitaram a criação de um novo mercado, o dos minicomputadores, pois este valorizava o tamanho físico mais que o desempenho funcional. E assim a tecnologia dessa área se desenvolveu, disrupção após disrupção novos mercados foram criados, primeiro foram os minicomputadores, então surgiram os computadores pessoais, após o portátil e enfim os dispositivos portáteis de mão, gerando um ciclo de desenvolvimento econômico que foi alimentado pela inovação tecnológica que só ocorreu devido a evolução dos SoCs.

Os discos rígidos, no início, tratavam-se de apenas peças mecânicas com um sistema elétrico de atuadores. O que possibilitou que tecnologias disruptivas surgissem foi o emprego de eletrônica discreta e posteriormente SoCs. Esse exemplo serve para ilustrar a importância dos SoCs para o desenvolvimento tecnológico global. E não surpreendente-

mente o mesmo avanço ocorreu, graças aos SoCs, em inúmeras outras áreas tecnológicas fundamentais para o desenvolvimento do planeta, como no ramo de transportes, no ramo de comunicação e em inúmeros outros ramos.

Devido a sua importância os SoCs são extremamente demandados por desenvolvimento, para tanto que a lei de Moore, que estima que a cada ano o desempenho computacional devesse dobrar, já foi ultrapassada há 10 anos [3]. Essa crescente demanda por desempenho implica que a tecnologia empregada em SoCs avance rapidamente, porém, esse avanço não ocorre sem efeitos colaterais.

Os SoCs são circuitos integrados implementados em silício desde a sua primeira versão, em 1968, para serem difundidos em 1971 quando a Intel lançou comercialmente o seu primeiro microprocessador denominado de Intel 4004. Desde então, a base da tecnologia dos circuitos integrados não mudou, pois a matéria prima dos computadores ainda hoje é o silício. Quando o Intel 4004 foi lançado ele utilizava um nodo tecnológico de $10\mu m$ (tamanho físico do gate do transistor) e contava com apenas 2300 transistores, enquanto o último lançamento da Intel até a data conta com 1.17 bilhões de transistores no nodo de $32nm$. Com todo esse avanço, a partir das tecnologias *Very Deep Submicron* (VDSM), o silício teve sua confiabilidade afetada, pois os defeitos introduzidos nos processos de manufaturas tornaram-se inevitáveis, aumentando a probabilidade de fabricação de SoCs com defeitos.

Os circuitos das tecnologias de VDSM possuem inúmeras interconexões. Em 1995 um chip possuía, em média, 380m de interconexões, para, em 2010, este número tornar-se 10km [4], fazendo com que os defeitos que ocorrem nessas interconexões sejam hoje os mais significativos, pois durante o processo de manufatura, onde os ICs (circuitos integrados) são construídos em camadas e os metais são inseridos por precipitação, e, assim, gerando estrutura tridimensionais; podem se formar más ligações metálicas entre uma camada e outra, fazendo com que essas conexões passem a se comportar como resistores, caracterizando a presença de um defeito do tipo *Resistive-Open*.

Os defeitos do tipo *Resistive-Open* foram analisados e apontados como a causa mais comum de gerar chips que escapam dos testes de manufatura [5]. De maneira geral, esses defeitos são definidos como um resistor em um nodo que idealmente deveria ter a sua impedância nula [6]. Ainda assim existe uma sub classe desses defeitos chamados de *Resistive-Open* fraco, que, baseados em uma distribuição de Poisson, são mais prováveis de ocorrer do que *Resistive-Open* de tamanho grande [7]. São responsáveis por gerar falhas dinâmicas que precisam de diversas operações para ser sensibilizadas [8] e, por isso, têm alta probabilidade de escapar dos testes de manufatura. E o que torna o problema mais alarmante é que os SoCs são largamente empregados em aplicações de missão crítica, onde falhas podem ser catastróficas.

Em um SoC existem três componentes elementares, por exemplo, o chip 4004 da Intel possuía apenas um processador (CPU), uma memória e um barramento de entrada e saída de dados encarregados pelo processamento, armazenamento e comunicação de dados, respectivamente. E em relação a distribuição desses componentes no chip estima-se que, em 2013, 95% da área de um SoC será gasta pela memória [4]. Entretanto, existem dois tipos de memórias utilizadas em SoCs: a estática (SRAM) e a dinâmica (DRAM). Dentre as duas, a SRAM é a mais encontrada por várias razões como ter maior desempenho, ser verdadeiramente de acesso aleatório, não precisar de *refresh*, ter menor consumo de energia e por tolerar maior ruído. Portanto a SRAM foi a memória escolhida para o estudo de caso deste trabalho.

A SRAM possui um crescente número de interconexões [9] dispostas de maneira densa, o que ocasiona numa considerável possibilidade de incidência de defeitos do tipo *Resistive-Open* nessas interconexões, e que, por fim, tende a causar falhas na memória do SoC. É de grande interesse neste trabalho analisar defeitos *Resistive-Open* em SRAM e propor uma técnica de detecção desses defeitos, pois qualquer incremento nesse assunto terá um impacto positivo em diversos pontos dos SoCs, como custos de manufatura, confiabilidade, desempenho, etc.

Entretanto, é importante destacar que a maioria dos testes de manufatura são desenhados para detectar falhas estáticas, que são falhas sensibilizadas com apenas uma operação [8]. E este trabalho tem interesse em desenvolver um sistema de detecção capaz de detectar defeitos independentemente de tamanho, e por consequência, evitar a presença de falhas tanto estáticas quanto dinâmicas quando ocasionados por defeitos do tipo *Resistive-Open*.

Este texto apresenta uma proposta elaborada na forma de uma metodologia implementada por um circuito analógico e digital com o objetivo de detectar defeitos do tipo *Resistive-Open*, que, por sua vez, foi validado e avaliado e os resultados encontram-se neste texto. Toda a proposta é desenvolvida fazendo o uso de simulador de circuitos integrados (HSPICE [10]), por onde os defeitos e suas respectivas consequências foram analisados detalhadamente, e, por onde o design pôde ser criado de maneira a solucionar qualquer incidência do defeito em questão através do monitoramento e comparação *on-chip* de sinais analógicos que sofram anomalias mediante a presença desses defeitos.

Neste documento, foi adotada a nomenclatura técnica em inglês assim como todas as siglas, para que não se crie divergências de tradução em relação aos termos comumente utilizados, com exceção dos termos criados para apresentar a técnica proposta, que foram mantidos em português. Contudo, para manter uma boa prática de escrita, a terminologia estrangeira está destacada em itálico, enquanto os nomes dos circuitos e elementos da proposta desta dissertação estão entre aspas em sua primeira menção, e, em itálico, quando forem palavras de língua inglesa.

1.1 Objetivos do Trabalho

O objetivo do trabalho é criar um sistema capaz de detectar e localizar a incidência de defeitos do tipo *Resistive-Open* em SRAMs, usado como teste de manufatura, para, posteriormente, recuperar o funcionamento ideal da memória em campo, permitindo injeção moderada de defeitos durante a manufatura, baixando o custo de produção e elevando a

confiabilidade do sistema onde ela é empregada. Este trabalho segue o seguinte cronograma de objetivos:

- Desenvolver uma SRAM estado da arte de complexidade mínima para servir de estudo de caso;
- estudar o impacto da ocorrência de defeitos do tipo *Resistive-Open* forte e fracos dentro da célula 6T da memória SRAM;
- propor um sistema de monitoramento de detecção de defeitos capaz de recuperar a confiabilidade de uma memória defeituosa em suas células;
- Validar e avaliar a eficiência da metodologia proposta considerando diferentes estudos de casos e cenários de memória com defeitos.

O presente trabalho visa à validação do funcionamento matemático da metodologia proposta. Então, questões que envolvam a robustez dos circuitos propostos que não sejam diretamente responsáveis em implementar a metodologia, acabam por não fazer parte do escopo. No trabalho tem-se o intuito de discutir e ao mesmo tempo de levantar os desafios associados a real implementação da proposta em um circuito integrado, já que se espera que em algum momento o que está aqui proposto seja realmente aplicado a SoCs.

1.2 Métodos e Métricas

A problemática deste trabalho envolve defeitos do tipo *Resistive-Open* que ocorram na célula da memória da SRAM. Para compreendê-la em volume considerável foram estudados diversos livros e artigos sobre defeitos resistivos e memória estática, e além disso, experimentos foram executados em uma memória estática especialmente desenvolvida para este trabalho, inserindo os defeitos resistivos e analisando a consequência no funcionamento dessa memória.

Após, os resultados desses experimentos foram confrontados contra os encontrados na

literatura, e por consequência, o modelamento dos defeitos e a memória desenvolvida foram dados como corretamente implementados por terem se comportado semelhantemente ao indicado por publicações científicas.

A memória, que foi desenvolvida para servir de estudo de caso, utilizou regras de dimensionamento datada de 1983 que ainda são válidas nas situações modernas, conferindo alta confiabilidade ao desenho de SRAM utilizado. Durante as apresentações da memória desenvolvida constam gráficos apresentando o seu funcionamento para demonstrar que ela opera como o esperado.

Os experimentos sobre os defeitos e o desenvolvimento da memória foram executados utilizando-se do simulador de circuitos HSPICE, assim como foi feito para o desenvolvimento da proposta desta dissertação. A proposta tem a forma de um circuito detector de defeitos, implementado em *netlist* de HSPICE, utilizando um método de construção iterativo baseado nos resultados das simulações rodadas pelo programa. Assim, parte a parte da estrutura de detecção de defeitos foi construída e testada até tomar a forma final, cuja é apresentada neste trabalho. Junto ao simulador, é utilizado uma biblioteca comercial da STMicroelectronics, que opera no nodo tecnológico $65nm$.

Adiante na dissertação, a proposta é validada junto ao HSPICE. Nesta parte do trabalho utilizou-se três condições de memória defeituosa para mostrar que a técnica proposta desempenha as suas funcionalidades como foi projetada.

Após a validação da proposta, o seu desempenho é estimado no capítulo Resultados Experimentais. Porém, o simulador de circuitos HSPICE não pôde ser utilizado para esta tarefa, porque para gerar os dados referente a capacidade de detecção da técnica foi utilizada uma abordagem de Monte Carlo que contou com bilhões de células de memórias sorteadas por simulação, e o HSPICE não tem condições computacional de executar esse volume de simulações em tempo hábil, nem mesmo tem condições de processar o volume de dados que seria gerado em tal cenário. Portanto um *software*, que simula funcionalmente a técnica proposta, foi desenvolvido especialmente para a tarefa. Executa os sorteios de

Monte Carlo para formar o uma situação de defeitos em um memória, e depois efetua as mesmas operações, que a técnica executa em *hardware* por portas lógicas sobre essa memória virtual. Em suma trata-se de um simulador da técnica proposta com abordagem de Monte Carlo junto a um gerador de memória defeituosa. O código-fonte deste programa encontra-se no APÊNDICE A ao final desta dissertação.

2 FUNDAMENTOS TEÓRICOS

A proposta deste trabalho consiste em um circuito detector de defeitos do tipo *Resistive-Open* que deve ser utilizado durante o teste de manufatura de SRAM e é avaliado utilizando experimentos de Monte Carlo. Assim sendo, este capítulo apresenta conceitos sobre Monte Carlo, o funcionamento da SRAM, defeitos do tipo *Resistive-Open*, como esses defeitos são modelados na SRAM e como são os testes de manufatura para detectá-los.

2.1 Experimentos de Monte Carlo

Experimentos de Monte Carlo são uma classe de algoritmo computacional que conta com sorteios repetitivos de variáveis aleatórias para computar os resultados dos experimentos. São largamente utilizados em situações que simulações reais são computacionalmente inviáveis, geralmente devido ao grande grau de liberdade dos sistemas simulados, que por sua vez dispendem de grande tempo de processamento para simulações determinísticas.

Neste trabalho foram utilizados experimentos de Monte Carlo para avaliar o desempenho de detecção da técnica proposta, tendo em vista que simular uma memória de um bilhão de células é computacionalmente inibidor, que especificamente para o caso, experimentos de Monte Carlo são fortemente indicados, pois a incidência de defeitos do tipo *Resistive-Open* na célula da SRAM apresenta um comportamento aleatório, independente e com distribuição gaussiana, cenário favorável a abordagem estatística.

Os resultados desses experimentos são histogramas que apresentam a contagem de cada situação monitorada. Para o uso neste trabalho, os experimentos geraram uma

contagem para cada possibilidade de detecção relacionada a técnica e com esses dados foi possível estabelecer um cenário em que a técnica pode ser utilizada com alto grau de confiabilidade. Tais dados encontram-se nos Resultados Experimentais, deste trabalho.

2.2 Memória Estática

Uma SRAM pode ser dividida superficialmente em 2 partes fundamentais: a matriz de células e os decodificadores. Explicações completas sobre a memória são encontradas no capítulo 6 do livro *Low-Power CMOS VLSI Circuit Design* [11]. A matriz de células é a parte encarregada em armazenar os dados, sendo os decodificadores de linha e coluna quem selecionam quais células dessa matriz serão acessadas em cada endereço.

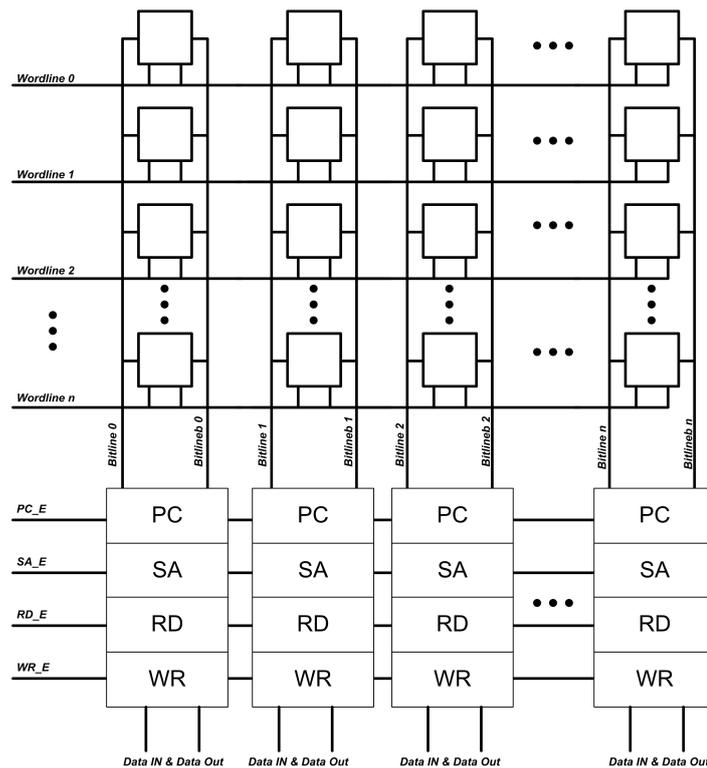


Figura 1 – SRAM simplificada.

A matriz de células da SRAM é composta por colunas e linhas, e, novamente de maneira simplificada, cada linha representa um endereço de memória onde as colunas, juntas, representam a largura do dado (8bits, 16bits, etc.).

Uma coluna é formada por todos os circuitos que utilizam o mesmo par de *bitlines*.

Durante uma operação apenas uma célula de cada coluna pode ser acessada para que a leitura ou escrita ocorra corretamente. Os *bitlines* são responsáveis por distribuir o dado para toda a coluna. Além das células, existem quatro circuitos conectados nos *bitlines*: o “Amplificador de Descarga” (SA, do inglês *Sense Amplifier*), o “circuito de pré carga” (PC - Pre-Charger) “*Pré-Charger*” (PC), o “circuito de leitura” (RD) e o “circuito de escrita” (WR), onde cada um tem o seu sinal de habilitação, “SA_E”, “PC_E”, “RD_E” e “WR_E” respectivamente (vide figura 1), que são compartilhados entre as colunas, exatamente como o sinal de habilitação das células de uma linha *i* (*wordline i*).

O circuito de pré carga tem a função de pré carregar os *bitlines*, que é o primeiro estágio de qualquer operação de leitura ou escrita¹.

O circuito amplificador de descarga tem a função de amplificar a diferença de tensão entre os *bitlines* a níveis lógicos. É sempre ativado na segunda etapa da operação, onde um dos *bitlines* é descarregado ou pela célula ou pelo circuito de escrita, e como ambos são circuitos fracos, o amplificador de descarga detecta qual dos *bitlines* está sendo drenado e atua para concluir a drenagem.

Já o circuito de escrita tem a função de drenar um dos *bitlines* na segunda etapa da operação de escrita, de acordo com o dado a ser escrito proveniente do meio externo da memória, enquanto o circuito de leitura tem a função de assimilar o dado dos *bitlines* na terceira etapa da operação de leitura e enviar o dado aos meios externos da SRAM.

E enfim, a célula, na leitura, é encarregada em descarregar um dos *bitlines* na segunda etapa, e, na escrita, é encarregada em assimilar o dado durante a terceira etapa.

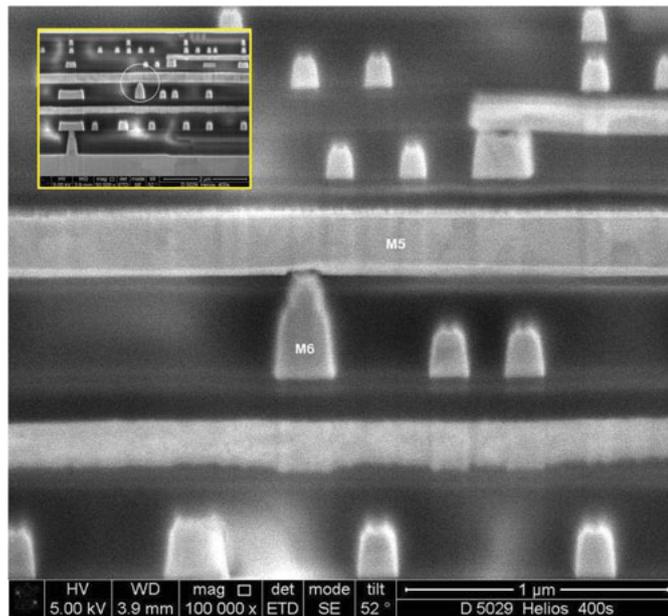


Figura 2 – Visão microscópica de um corte transversal de um IC revelando um defeito resistivo em uma interconexão entre o metal 5 e 6 (obtida em [1]).

2.3 Defeitos do Tipo *Resistive-Open*

Como pôde ser concluído nos tópicos introdutórios desta dissertação, o defeito do tipo *Resistive-Open* na prática é uma espécie de mal contato gerado durante a precipitação de metal sobre o metal de uma camada ou via já existente, em que os metais não se ligam da maneira em que foram projetados dentro de um circuito integrado, como ilustrado na figura 2. Com isso, do ponto de vista elétrico, esses defeitos comportam-se como resistores transformando um nodo que idealmente deveria ter sua impedância nula em um resistor.

Esses defeitos são modelados em simuladores através do aumento da impedância do nodo de interesse, que idealmente deveria ser nula. Para isso são utilizados resistores ideais.

¹Embora não seja eficiente, do ponto de vista de consumo dinâmico de energia, efetuar pré carga dos *bitlines* para a operação de escrita, assim a memória foi projetada com o intuito de controlar a complexidade da mesma. Existem autores que sugerem que a SRAM opere desta maneira [12] enquanto outros não [11].

2.4 Incidência de defeitos na SRAM

Este trabalho estuda o efeito dos defeitos do tipo *Resistive-Open* que ocorrem dentro das células da SRAM de maneira individual, e segue os dizeres abaixo.

2.4.1 Modelamento na SRAM

No esquema de conexões de uma célula de SRAM, observa-se dezoito nodos que podem ter o seu desempenho afetado devido a um defeito resistivo, então para que se simule todos os locais possíveis de incidir defeitos em uma célula, seria necessário explorar todos estes nodos. No entanto, existem seis posições de inserção de defeitos que, sozinhos, são capazes de sensibilizar todos os tipos possíveis de falhas na célula da memória [13], já que as células da SRAM possuem um desenho espelhado e posições diferentes ocasionam falhas semelhantes. Essas seis posições estão ilustradas na figura 3 e formam o modelamento dos defeitos do tipo *Resistive-Open* utilizado por este trabalho.

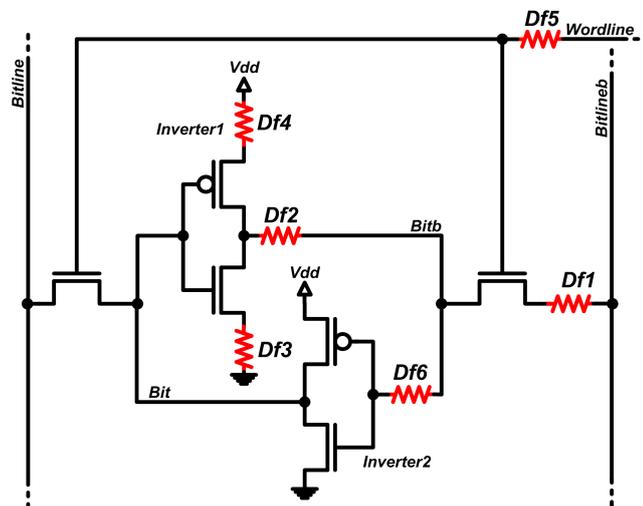


Figura 3 – Célula 6T com as posições referentes dos defeitos DF1 ao DF6.

2.4.2 Falhas funcionais

De acordo com o tamanho dos defeitos, a SRAM tende a apresentar falhas funcionais [14]. Elas são definidas segundo o tipo de falha que é observada do ponto de vista

funcional, o que envolve o dado armazenado e o dado de interesse durante as operações de acesso a memória. Existem centenas de tipos de falhas funcionais que podem ocorrer em memórias, dentre todas, as que ocorrem na SRAM devido a um defeito do tipo *Resistive-Open*, de ocorrência individual em uma célula individual, estão descritas abaixo, considerando a sua natureza, que pode ser dinâmica ou estática. Sendo que uma falha é considerada estática, quando precisa de apenas uma operação para sensibilizá-la, e considerada dinâmica, quando são necessárias duas ou mais operações.

- *Transition Fault* (TF): Quando uma célula falha ao executar a troca de dado durante uma escrita.
- *Read Destructive² Fault* (RDF): Quando uma célula tem o seu dado trocado durante uma leitura, e retorna o dado incorreto.
- *Deceptive Read Destructive Fault* (DRDF): Quando uma célula tem o seu dado trocado durante uma leitura, porém retorna o dado correto.
- *Incorrect Read Fault* (IR): Quando uma célula retorna o dado incorreto durante a leitura.
- *dynamic Read Destructive Fault* (dRDF): O mesmo que *Read Destructive Fault*, porém necessita de duas ou mais operações consecutivas para sensibilizar a falha.
- *dynamic Deceptive Read Destructive Fault* (dDRDF): O mesmo que *Deceptive Read Destructive Fault*, porém necessita de duas ou mais operações consecutivas para sensibilizar a falha.

Na ocorrência de defeitos do tipo *Resistive-Open* fraco as falhas funcionais ocasionadas são de natureza dinâmica [8], como dRDF e dDRDF. Esses defeitos fracos só irão sensibilizar falhas quando houver acessos consecutivos na célula defeituosa, fazendo com que os testes de manufatura de SRAM não detectem estes defeitos, já que pode ser necessário inúmeros acessos por célula para sensibilizá-las.

²Alguns autores utilizam a palavra *Disturb* [14] ao invés de *Destructive* [13].

2.5 Testes de Manufatura para Detecção de Defeitos do Tipo *Resistive-Open*

Neste tópico são apresentados três dos principais tipos de testes de manufaturas utilizados. *March test* são utilizados para detectar falhas e são eficientes quanto às estáticas, já *I_{ddq} test* tem a capacidade de detectar defeitos e um terceiro tipo de teste de manufatura consiste na união dos dois testes com o objetivo de utilizar as vantagens de ambos, porém existem implicações e limitações que são apresentadas nos próximos tópicos.

2.5.1 *March Test*

Especificamente em memórias existe um tipo de testes muito utilizado do tipo *Build-In Self Test* (BIST) e chamado de *March test*. São testes que executam algoritmos de operações de acessos que têm o intuito de sensibilizar falhas e possibilitar que sejam detectadas funcionalmente utilizando como princípio que ao se escrever um dado em um determinado endereço espera-se que ao ser lido o mesmo dado seja encontrado. Para que um teste dessa natureza seja efetuado em uma memória é necessário que exista um processador embutido que controlará a execução do teste (por isso é considerado BIST), e, que os decodificadores da SRAM permitam acesso a uma única célula, quando for necessário, pois a maioria dos *March tests* acessam células individuais. Como resultado o BIST gera uma assinatura que é enviada aos meios externos do SoC, para ser usada na reconfiguração da memória, em que colunas e/ou linhas defeituosas são substituídas por outras sobressalentes [15] quando técnicas de recuperação deste tipo estiverem presentes.

Diversos algoritmos de *March test* já foram propostos para detectar falhas em memórias, por exemplo, o mais comum é um algoritmo chamado de MarchSS [16], que executa 22 operações por célula (notação 22n) e promete detectar todas as falhas funcionais estáticas que ocorram em células individuais, exatamente o tipo de falhas geradas por defeitos do tipo *Resistive-Open* forte.

Tratando-se da detecção de falhas funcionais dinâmicas, existem outros algoritmos

Algoritmo 1 Algoritmo de teste de manufatura MarchSS

$$\{ \uparrow \downarrow (w0); \\ \uparrow (r0, r0, w0, r0, w1); \uparrow (r1, r1, w1, r1, w0); \\ \downarrow (r0, r0, w0, r0, w1); \downarrow (r1, r1, w1, r1, w0); \\ \uparrow (r0); \}$$

de *March test* desenvolvidos para esta função [17] [18], porém são apenas capazes de detectar falhas dinâmicas sensibilizáveis por duas operações consecutivas, já que não é possível utilizar *March tests* para detectar falhas dinâmicas de várias operações. Por exemplo, efetuando uma estimativa grosseira, o algoritmo MarchSS teria de ser modificado de maneira que cada acesso teria de ser repetido em torno 20 vezes consecutivas para detectar falhas dinâmicas de várias operações, elevando o número de acessos para 440 por célula durante o teste, o que vai ocupar mais tempo do que o disponível para a execução durante a manufatura, pois os testes geralmente empregados não passam de 30 acessos por célula. Então, todas as falhas dinâmicas não podem ser detectadas por este teste de manufatura.

2.5.2 I_{ddq} test

Existe outro tipo de teste de manufatura que é de interesse desta dissertação. Trata-se do I_{ddq} test, que consiste no monitoramento de corrente consumida para diagnosticar a saúde de um IC qualquer, não só de memórias. O monitoramento ocorre durante a excitação do circuito por um vetor definido, onde a sua saúde pode ser inferida de duas maneiras, uma comparando o consumo de corrente com limiares, e outra, comparando o consumo de corrente com o consumo de outro circuito idêntico que é excitado da mesma maneira. A física que permite que esta metodologia efetue detecções consiste em que a ocorrência de falhas ou defeitos, inevitavelmente, irá distorcer o consumo de corrente do circuito, como mostram as figuras 4 e 5 adiante no texto.

Porém, um problema surge quando tenta-se isolar a variação de consumo inserida no consumo de corrente por defeitos fracos da variação natural inserida pelo processo

de fabricação em tecnologias VDSM. Por exemplo, estudos [19] fazem uma análise da detectabilidade dos defeitos do tipo *Resistive-Open*, frente as variações de processo de fabricação, utilizando *I_{ddq} test*, e, geram gráficos mostrando a distribuição da variação inserida pelos defeitos e a distribuição da variação inserida pelas variações do processo (*corners*) nos sinais monitorados pelo teste de manufatura. As conclusões são que apenas defeitos na ordem de grandeza de mega ohms são distinguíveis diante das variações geradas pelos *corners* na tecnologia utilizada por esse estudo. Conclusões semelhantes também foram obtidas ao comparar a magnitude da variação introduzida pelos *corners* com a variação introduzidas por um defeito do tipo *Resistive-Open* na SRAM que foi utilizada de estudo de caso neste trabalho (posteriormente descrita). Essa comparação está ilustrada na figura 4, que apresenta as correntes *I_{vdd}* e *I_{gnd}* de uma célula sendo acessada pelo algoritmo *March* (*w0,r0,w1,r1*).

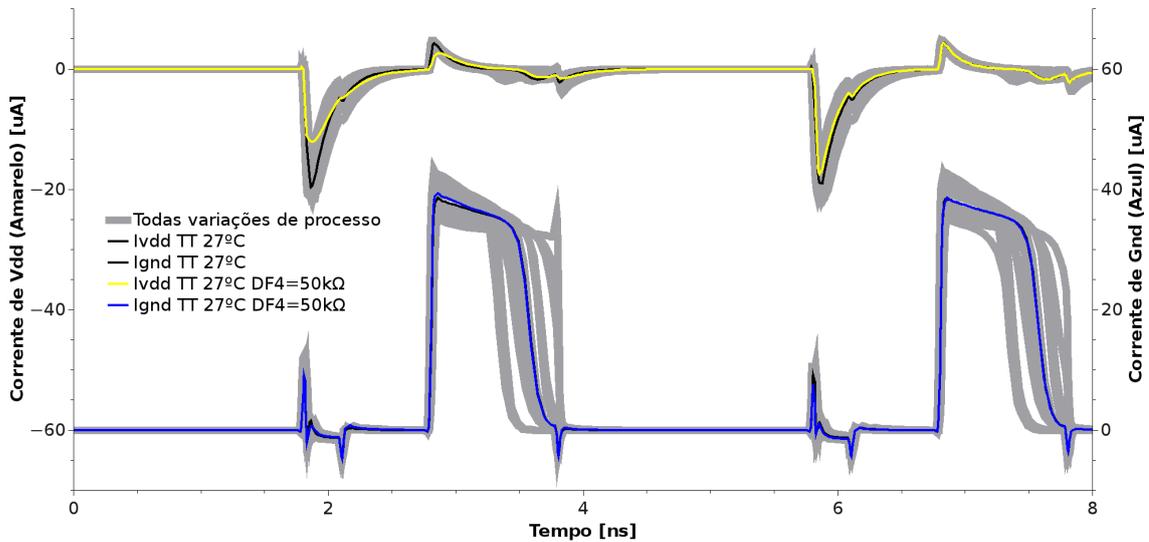


Figura 4 – Corrente de *V_{dd}* e *G_{nd}* para todos os *corners* e para *DF4=50kΩ*, demonstrando o ofuscamento das variações introduzidas pelo defeito quando considerada as variações introduzidas pelos *corners*.

As curvas aglomeradas em cinza são consequências das variações introduzidas pelos *corners*. Em preto, encontra-se o consumo para o *corner* TT 27°C de cada sinal apresentado. E, em amarelo e azul, estão o consumo *I_{vdd}* e *I_{gnd}* para uma célula com o *DF4* de tamanho de 50kΩ, ou seja, um defeito do tipo *Resistive-Open* fraco. Note que as curvas em amarelo e azul estão completamente compreendidas nas curvas cinzas, provando que

a presença desse defeito jamais será distinguida da variação introduzida pelos *corners* quando monitorado o consumo de corrente da maneira em que *I_{ddq} test*.

2.5.3 OCCS associado ao *March test*

Semelhante a *I_{ddq} test* pode-se efetuar o monitoramento de corrente consumida observando a corrente em silício, usando sensores de correntes embutidos, ou OCCSs, do inglês *On-Chip Current Sensors*. Ao utilizar esses sensores internos é possível monitorar regiões separadas do mesmo circuito integrado, e, também, possibilita o monitoramento *online*, não só na etapa de manufatura.

OCCSs são compostos geralmente por duas partes [20], o conversor de corrente em tensão e o amplificador e/ou comparador. O desafio do projeto desses sensores está no controle da queda de tensão inserida no sinal monitorado, no tempo de resposta dos sensores, no consumo (por tratarem-se de circuitos analógicos) e na influência que sofrem devido aos *corners*. Na prática, OCCSs são predominantemente empregados para a detecção de surtos de consumo (picos de amplitude significativa) causados por falhas ou grandes defeitos, portanto frequentemente utilizam-se limiares para se aferir a qualidade do sinal monitorado.

Recentemente, a possibilidade do emprego de OCCS para detectar a presença de defeitos do tipo *Resistive-Open* em SRAM foi investigada [21]. Os autores desta investigação demonstraram que os defeitos distorcem o consumo da célula, e, também propuseram que fosse utilizado um OCCS por coluna monitorada para disparar um sinal avisando que uma falha estática ocorrera sempre que a corrente consumida pela célula ultrapassasse dado valor, caracterizando o uso de limiares. A abordagem é ineficaz para os objetivos deste trabalho, onde a intenção é de detectar defeitos do tipo *Resistive-Open* de qualquer tamanho e posição de incidência, e assim, ora o consumo pode ser maior, e ora, menor do que o consumo de uma célula sem falhas, impossibilitando o uso de limiares fixos como os propostos na investigação citada. Entretanto é imune às variações *inter-die*, porque os

circuitos monitorados estão no mesmo *corner*, fazendo com que seja possível de detectar defeitos do tipo *resistive-open* fraco.

A exemplo das afirmações do último parágrafo, observa-se a ilustração da figura 5, que é proveniente da análise efetuada sobre a SRAM projetada e os defeitos em questão. O consumo de corrente para cada um dos defeitos modelados é exibido (em linhas vermelhas e pretas), provando que o consumo pode ser maior ou menor, do que o consumo de uma célula sem defeitos (pintados de amarelo e azul), de acordo com a operação executada na célula e a posição do defeito do tipo *Resistive-Open*. O que proíbe o uso de limiares para que seja possível detectar defeitos independentemente de posição, tamanho, e operação efetuada.

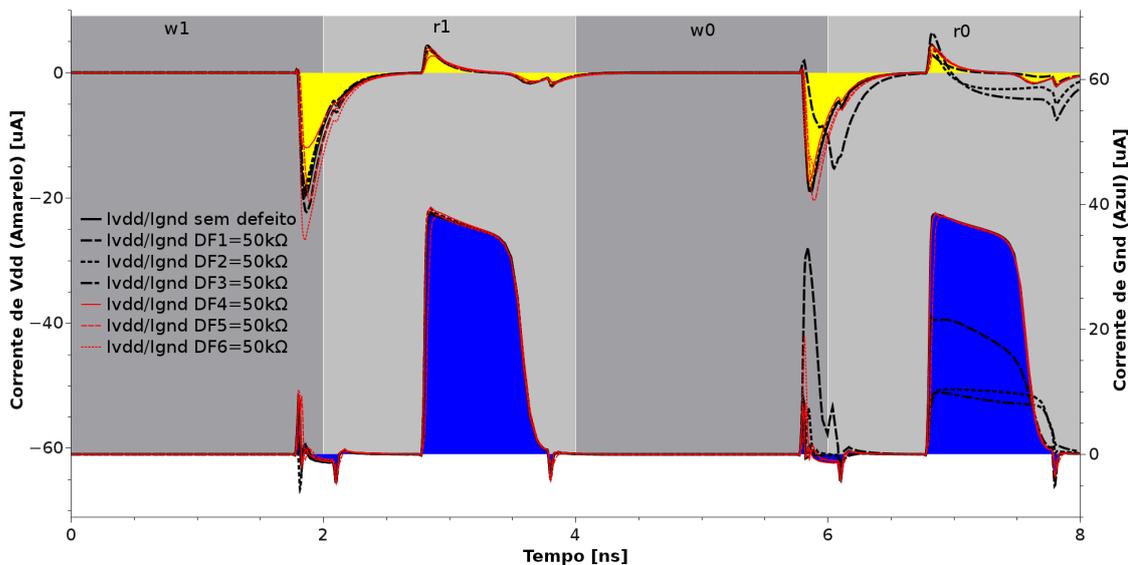


Figura 5 – Corrente de *Vdd* e *Gnd* para a célula boa (área pintada) e para a célula com cada um dos seis defeitos considerados com tamanho de $50k\Omega$ (DF1 a DF6 = $50k\Omega$).

2.6 Desafios para Detectar os Defeitos do tipo *Resistive-Open* em SRAM

Concluindo este capítulo, os três desafios observados para a detecção de defeitos do tipo *Resistive-Open* em memórias estáticas estão enumerados abaixo.

1. Testes de manufatura convencionais (*March test*) não são capazes de detectar falhas

dinâmicas que precisam mais de duas operações consecutivas para sensibilizá-las.

2. A variação introduzida pelos *corners* nas tecnologias VDSM é maior que a variação introduzida por grande parte dos defeitos do tipo *Resistive-Open* nos sinais analógicos de corrente, quando monitorado a nível do chip como um todo e sem o conhecimento prévio do *corner* em que o IC se encontra, podendo inviabilizar metodologias que monitoram o consumo de corrente para detectar esses defeitos.
3. Ainda em relação as metodologias de monitoramento de consumo, os defeitos em questão, de acordo com a sua posição, intensidade e a operação de acesso utilizada, podem aumentar ou diminuir o consumo de corrente, portanto, não é possível efetuar comparações que contam com limiares para aferir a qualidade das células da memória.

Assim sendo, este trabalho propõe uma metodologia de detecção, capaz de vencer os três desafios levantados, no próximo capítulo.

3 PROPOSTA

Esta dissertação de mestrado tem o objetivo de atribuir aos testes de manufatura a capacidade de detectar qualquer defeito do tipo *Resistive-Open*, sejam os que causam falhas estáticas (os fortes) ou os que causam falhas dinâmicas (os fracos), que ocorrem dentro da célula 6T da SRAM, independentemente de intensidade ou posição.

Para a função, a proposta foi elaborada considerando duas metodologias consagradas de detecção de defeitos e redundância de *hardware*, uma utiliza sensores de monitoramento de corrente [21], e, a outra, votadores [22] que elegem qual o sinal correto em um sistema com redundância de blocos lógicos.

Nesta proposta, os sensores de corrente têm o objetivo de monitorar o sinal de corrente (que é afetado na presença de defeitos) e gerar um sinal digital de 1 bit modulado em largura de pulso contendo informação a respeito da corrente consumida de uma coluna, assim é utilizado um sensor de corrente por coluna da SRAM. A partir daí, fazendo uso da enorme redundância de *hardware* que uma SRAM possui, um circuito lógico, semelhantemente a votadores, compara o sinal digital referente ao consumo de corrente de cada coluna, só que neste trabalho, diferentemente de votadores, o dado gerado por essa lógica contém a informação de qual coluna teve uma célula defeituosa acessada.

3.1 A técnica de Detecção de Defeitos

De maneira abrangente, o objetivo da técnica é efetuar aferições a respeito da presença de defeitos do tipo *Resistive-Open* que podem ocorrer em alguma célula da matriz da

memória através do monitoramento de consumo de corrente dessas células. Dividas em grupos de comparação, ou vizinhanças, cada célula, ou membro da vizinhança, tem o seu comportamento de consumo comparado com os membros da sua mesma vizinhança, sendo que, analogicamente falando, uma vizinhança é semelhante aos blocos funcionais da técnica de redundância de *hardware* e o circuito que efetua essas comparações é semelhante ao bloco votador.

Assim sendo, quando um ou mais membros da vizinhança se comportarem diferentemente dos seus vizinhos (em relação ao consumo de corrente), esses serão dados como defeituosos. A referência de comportamento correto (ou comportamento esperado) é obtida durante a comparação, sendo fruto do comportamento que ocorre em maior frequência na vizinhança, e portanto, deve ser utilizada apenas pela vizinhança que a gerou. Onde que uma vizinhança é definida por um grupo de colunas da memória em que as células acessadas durante o teste de manufatura têm o seu consumo monitorado e comparado entre si, com isso, uma memória pode ter inúmeras vizinhanças.

A metodologia caracteriza-se pelos seguintes pontos:

- Cada vizinhança é composta por um grupo de colunas (ou membros);
- O consumo de corrente das células acessadas de cada coluna é comparado na vizinhança;
- Cada vizinhança obtém o comportamento esperado (consumo da maioria dos membros) a partir dos seus membros;
- O comportamento esperado é fruto do comportamento ocorrido na maioria dos membros da vizinhança;
- Quando algum membro se comporta diferentemente do comportamento esperado, esse é considerado defeituoso.

Esta metodologia de detecção de defeitos traz grandes benefícios. Como exemplo, não há a necessidade de se conhecer o comportamento correto das células da memória

(consumo de corrente de uma célula sem defeitos) durante o projeto do circuito integrado, nem precisa fazer uso de limiares (que são limitados) para efetuar comparações e por efetuar comparações entre elementos de dentro do chip, torna os defeitos do tipo *Resistive-Open* fraco detectáveis diante das variações de processo, pois não haverá variações entre uma célula e outra, em um processo de manufatura sem variações *intra-die*.

Entretanto, o emprego desta metodologia não ocorre sem efeitos colaterais. Existe uma limitação em se efetuar comparações como as descritas. Se acontecer da maioria dos membros de uma vizinhança apresentar comportamento identicamente errôneos, a referência gerada por essa vizinhança também será errônea e os membros com o comportamento correto serão acusados defeituosos, enquanto os membros possuidores do comportamento errôneo serão acusados de se comportar adequadamente, algo que é extremamente indesejado durante os testes de manufatura. Esta é uma limitação da natureza da metodologia de detecção proposta, porém a ocorrência desta situação indesejada é dependente da distribuição probabilística de comportamentos dos membros monitorados. No capítulo Experimentos Teóricos essa limitação é exaustivamente analisada, concluindo que na prática não é limitante.

3.1.1 Estrutura de detecção

A implementação da estrutura encarregada em executar a metodologia de detecção exigiu que o barramento de alimentação (V_{dd} e Gnd) da SRAM fosse redesenhado. Os parágrafos a seguir explicam a estrutura de detecção proposta e devem ser lidos com o acompanhamento da figura 6, pois ela ilustra os circuitos explicados por eles. Tanto a figura quanto o texto mostra que as vizinhanças da estrutura têm tamanho de 4 membros, esta escolha será esclarecida adiante no capítulo.

A memória, os sensores de corrente (On-Chip Current Sensor, OCCS) e as Lógicas Comparadoras de Vizinhança (LCV) compõem todo o circuito utilizado pela técnica de detecção proposta neste trabalho. Na figura 6 a SRAM é representada apenas pelas suas

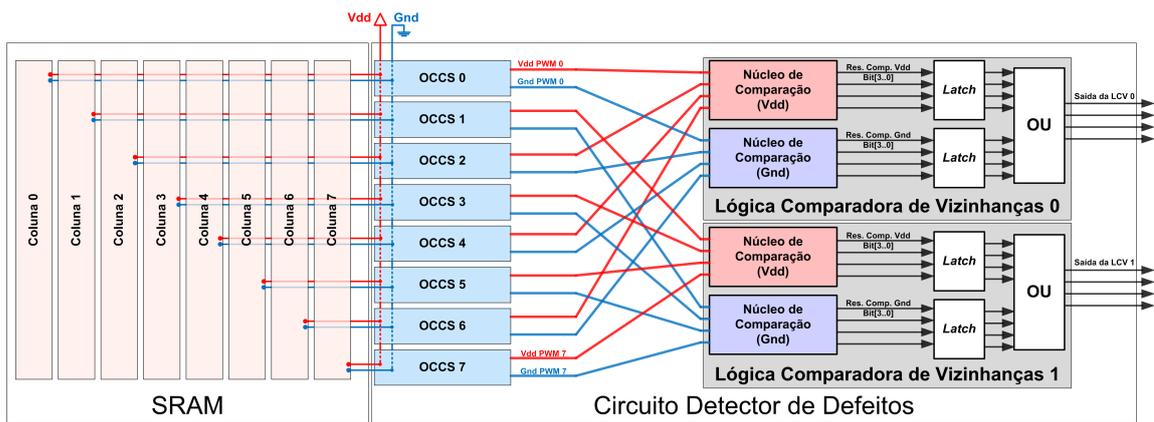


Figura 6 – *Hardware* da técnica de detecção de defeitos acoplada ao esquemático simplificado de uma SRAM.

colunas de células (de 0 a 7) contendo cada uma o seu respectivo sensor de corrente. A saída dos sensores são roteadas até as LCVs, caracterizando as duas vizinhanças utilizadas. E, então, cada LCV é ilustrada com seus circuitos internos, incluindo o núcleo de comparação para Vdd e Gnd e os circuitos que fazem a retenção (*latch*) e união (porta “ou”) dos dados provenientes da detecção efetuada em cada grupo de sinais de alimentação das células da memória.

A distribuição do barramento de alimentação da matriz de células foi alterada de maneira que as células de cada coluna passem a ter um sinal de Vdd e outro de Gnd próprios, enquanto em uma SRAM convencional o sinal de Vdd e Gnd são globais e compartilhado entre todos os circuitos existentes. Assim sendo, quando uma coluna é utilizada em uma operação, seja para leitura ou para escrita, sabendo que apenas uma célula é acessada, o consumo de corrente I_{vdd} e I_{gnd} medidos nessa coluna refletirá o consumo dinâmico da célula acessada acrescido do consumo de *leakage* das demais células em repouso nessa mesma coluna.

Para que as células recebam as tensões apropriadas do barramento de alimentação modificado, este, é conectado ao barramento global através dos sensores de corrente da coluna que também têm a função de medir e gerar um sinal digital de 1 bit para cada corrente de consumo monitorada. O sensor executa essa tarefa transformando o sinal analógico de corrente em um sinal digital modulado em largura de pulso (sinal PWM).

Ao saírem dos sensores, esses sinais PWM são roteados às LCVs. Neste trabalho foi adotado um esquema de distribuição de vizinhanças intercalados, pois existe uma classe de defeitos chamados de *resistive-shorts* que tendem a acoplar células fisicamente vizinhas. Com a distribuição intercalada, duas células acopladas por um defeito não cairão na mesma vizinhança, pois se duas células conterem defeitos numa vizinhança de 4, por exemplo, perde-se a referência de comportamento adequado. Embora este trabalho tenha considerado os *resistive-shorts* na distribuição das vizinhanças, essa classe de defeitos não faz parte do escopo desta dissertação e só será estudado em trabalhos futuros.

Finalmente, as LCVs recebem os sinais PWM para efetuarem a detecção. É utilizado uma LCV por vizinhança monitorada, sendo que foi adotado que cada vizinhança tem quatro membros (posteriormente explicado), e, sabendo que há 8 colunas na SRAM, existem duas vizinhanças, e logo, duas LCVs, na técnica de detecção proposta, como ilustrou a figura 6.

Durante o uso, todo o esquema proposto opera da seguinte maneira:

1. Colunas de índice par pertencem à vizinhança 0 (monitorada pela “LCV 0”) e de índice ímpar pertencem à vizinhança 1 (monitorada pela “LCV 1”);
2. Durante o acesso, as correntes I_{vdd} e I_{gnd} de uma coluna n são convertidas, pelos sensores, nos sinais “ Vdd PWM n ” e “ Gnd PWM n ” que representam a amplitude e a duração do pulso de consumo de I_{vdd} e I_{gnd} , respectivamente.
3. Os sinais que partem do Vdd PWM 0 ao Vdd PWM 7 e do Gnd PWM 0 ao Gnd PWM 7 são roteados para a LCV 0 e a LCV 1 formando as duas vizinhanças da memória estudo de caso;
4. Dentro de cada LCV ocorrem duas detecções, uma comparando os sinais de Vdd em um núcleo de comparação, outra comparando os sinais de Gnd no outro núcleo. O resultado desta comparação é armazenado pelos *latches* e depois unidos pela porta lógica “ou”, resultando nos sinais “Saída da LCV 0” e “Saída da LCV 1” que contém a

informação a respeito da detecção efetuada em cada vizinhança, ou seja, se alguma das células acessadas se comportou diferentemente das demais.

A estrutura apresentada é utilizada durante todo o trabalho sempre que houver experimentos utilizando o simulador HSPICE, a memória da figura é semelhante a memória estudo de caso, que também possui 8 colunas. Esta memória será apresentada adiante no texto do trabalho.

Na estrutura existem dois elementos chaves, o sensor de corrente e a LCV. Ambos são explicados a seguir.

3.1.2 Sensor de Corrente

O sensor de corrente (OCCS) proposto nesse capítulo tem o objetivo de suprir a LCV com os sinais digitais modulados em largura de pulso, assim sendo, o sensor não foi avaliado em relação ao seu desempenho, e sim, somente em relação a sua funcionalidade, garantindo que a conversão de corrente para PWM lógico ocorra como o esperado.

No projeto de um sensor de corrente há sérias dificuldades em controlar a resposta em frequência do sensor devido às variações do processo de fabricação. É muito importante que os sensores apresentem a mesma resposta, principalmente em sistemas onde os dados gerados por esses serão comparados entre si. Quando os sensores forem utilizados em tecnologias que apresentem variações *intra-die*, é fundamental que os sensores analógico apresentem alguma forma de compensação dessas variações [23]. Entretanto, a abordagem de desenvolvimento, nesta dissertação, não considera variações *intra-die* e nem esses sensores foram otimizados para tolerarem as variações *inter-die*, o que por fim abstrai grande complexidade durante o projeto do mesmo, embora sem comprometer o funcionamento da técnica proposta e nem a sua avaliação.

Outro desafio associado a esse tipo de sensor é o controle do distúrbio que a sua presença insere nos sinais monitorados, pois o sensor coloca um resistor em série no nodo em que se deseja medir a passagem de corrente, e por consequência, gera um distúrbio na

tensão do nodo monitorado. Por isso o sensor faz uso de um amplificador operacional, para que o distúrbio inserido seja o menor possível, e que, amplificado a níveis satisfatórios, seja facilmente manipulado pelos estágios seguintes.

O sensor projetado nesta proposta é pouco diferente dos convencionais, possui um conversor de corrente para tensão e um amplificador como mencionado, e foi acrescido de um gerador de PWM que transforma o sinal de corrente, já amplificado e convertido para tensão, em um pulso digital PWM. E, como o uso do sensor é para monitorar a corrente I_{vdd} e I_{gnd} das colunas da SRAM, o sensor passa a ter dois circuitos, cada um competente em monitorar um dos sinais de corrente, como mostra o esquemático da figura 7. Cada uma dessas três partes é explicada a seguir.

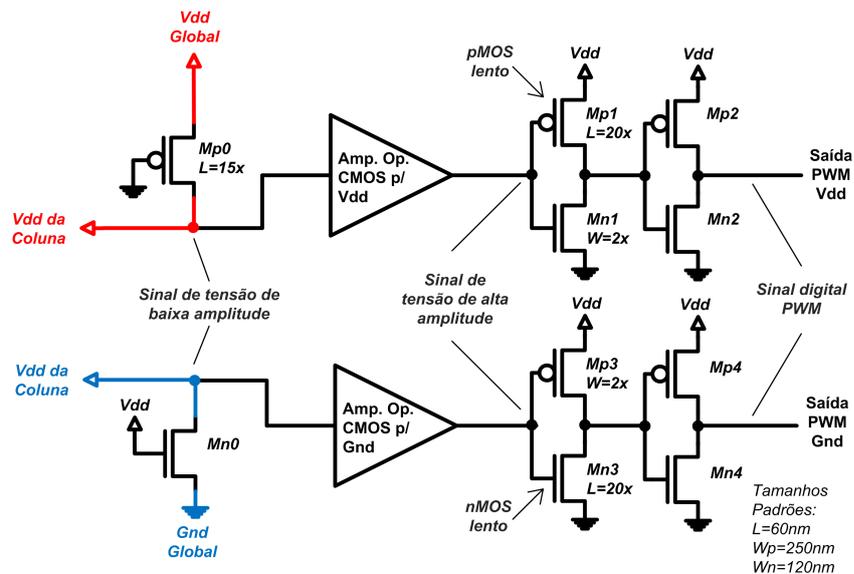


Figura 7 – Diagrama esquemático do sensor de corrente projetado com uma parte para monitorar I_{vdd} e outra para I_{gnd} .

Em suma, a corrente é convertida em tensão por um transistor que funciona como resistor com taxa de conversão de para Gnd e Vdd de $3,686\text{mV}/\mu\text{A}$ e $14,867\text{mV}/\mu\text{A}$ respectivamente. Depois este sinal de baixa amplitude é amplificado por um amplificador operacional convencional de ganho aproximado de 8800 para Vdd e 9500 para Gnd , e, finalmente, transformado em PWM pelos inversores no final do circuito do sensor.

O amplificador para Vdd e Gnd têm estrutura análoga, são proveniente de recomendações e esquemáticos encontrados em artigos [24] [25]. A figura 8 apresenta o esquemático

utilizado para os amplificadores operacionais.

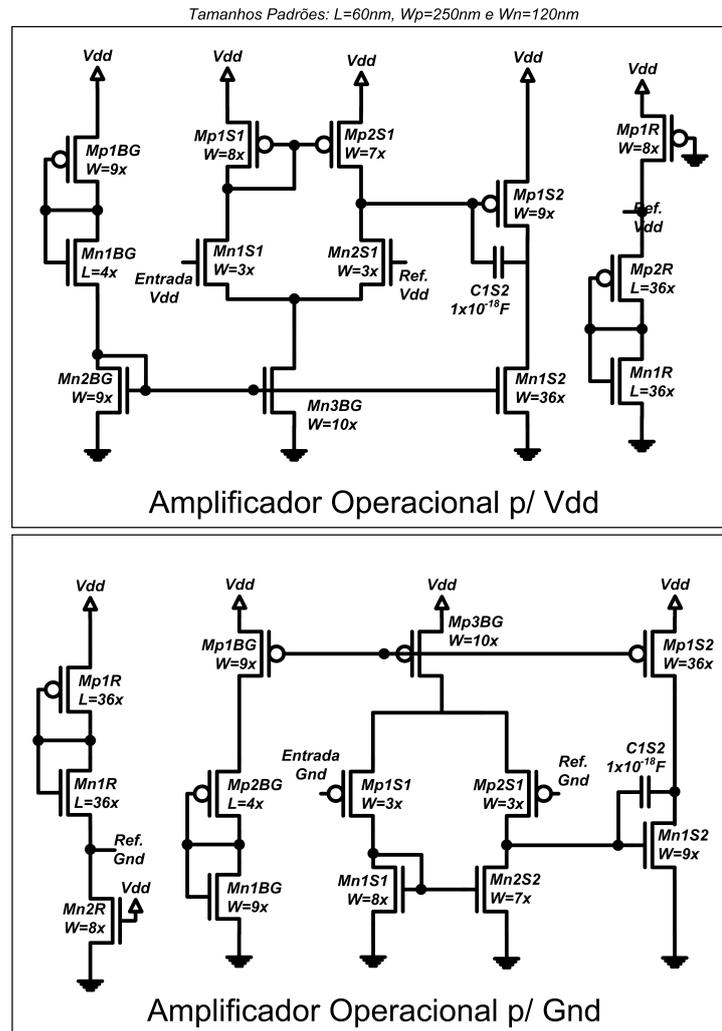


Figura 8 – Diagrama esquemático do amplificador operacional projetado para amplificar I_{vdd} e I_{gnd} já convertidos para tensão.

Esses amplificadores operacionais são compostos por quatro partes: primeiro os transistores com sufixo $S1$ (como $Mp1S1$) formam o primeiro estágio do amplificador, que tem o objetivo de amplificar a tensão diferencial da entrada do amplificador, depois o segundo estágio, formado pelos transistores com sufixo $S2$, tem a função de aumentar a excursão do sinal de saída para toda a magnitude de V_{dd} (*rail to rail*), a terceira parte é o gerador de tensão de bias (sufixo BG), que tem o objetivo de polarizar os transistores para mantê-los na região linear de condução de corrente, e por fim, a quarta parte, o gerador do sinal de referência (sufixo R) providencia a tensão que junto do sinal de entrada formam o par diferencial de entrada do primeiro estágio do amplificador.

A terceira parte do sensor de corrente proposto diz respeito ao gerador de PWM. Este gerador é composto por um inversor desequilibrado e outro normal, em série. A figura 9 ilustra como os inversores operam para gerar o sinal PWM. O inversor desequilibrado, na presença de um sinal de entrada, irá inverter a sua saída rapidamente porém quando o sinal em sua entrada cessar, esse inversor demorará para tornar a sua saída de volta ao estado original, porque possui um dos seus transistores lento, criando um sinal que carrega rápido e se descarrega lentamente. No final, o próximo inversor gera um sinal modulado em largura de pulso a partir do sinal gerado pelo inversor desequilibrado.

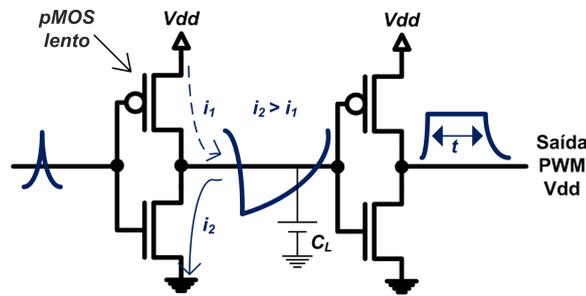


Figura 9 – Diagrama esquemático e representação das correntes do gerador de PWM para os sinais de V_{dd} . O circuito de Gnd é análogo.

Nas figuras 7 e 8 observa-se o tamanho da largura e comprimento do poço dos transistores utilizados, as medidas apresentadas são multiplicadores dos tamanhos padrões para L , W_p e W_n . Durante o projeto, essas medidas foram calibradas seguindo as recomendações da literatura e também na base de tentativa e erro até que se obtivesse cada parte do sensor funcionando adequadamente. Mais detalhamento a respeito desta etapa do desenvolvimento do trabalho não serão abordados.

Juntos, essas três partes do sensor são capazes de transformar um sinal de corrente em um sinal digital modulado em largura de pulso.

Para avaliar o funcionamento do sensor de maneira global, foi observado a sua saída quando utilizado entradas de duas correntes de teste diferentes (vide figura 10). Essas duas correntes foram geradas por células durante uma operação de escrita seguida de uma de leitura, em que uma delas possuía defeitos enquanto a outra não. Ao final, o sensor retorna dois sinais digitais de largura de pulso diferentes, pois entre o tempo $4,2ns$ e

5,5ns o primeiro sinal encontra-se no nível lógico baixo, enquanto o outro no nível lógico alto, diferença na qual ocorreu porque os consumos de corrente das células foram, de fato, diferentes. Na verdade, o sinal 1 pertence a uma célula boa enquanto o sinal 2 a uma célula com um defeito do tipo *Resistive-Open* de $50k\Omega$, fraco o suficiente para não gerar falha alguma, porém suficientemente grande para inserir um aumento na corrente consumida e por sua vez aumentar o tempo de duração do sinal PWM que representa o seu comportamento.

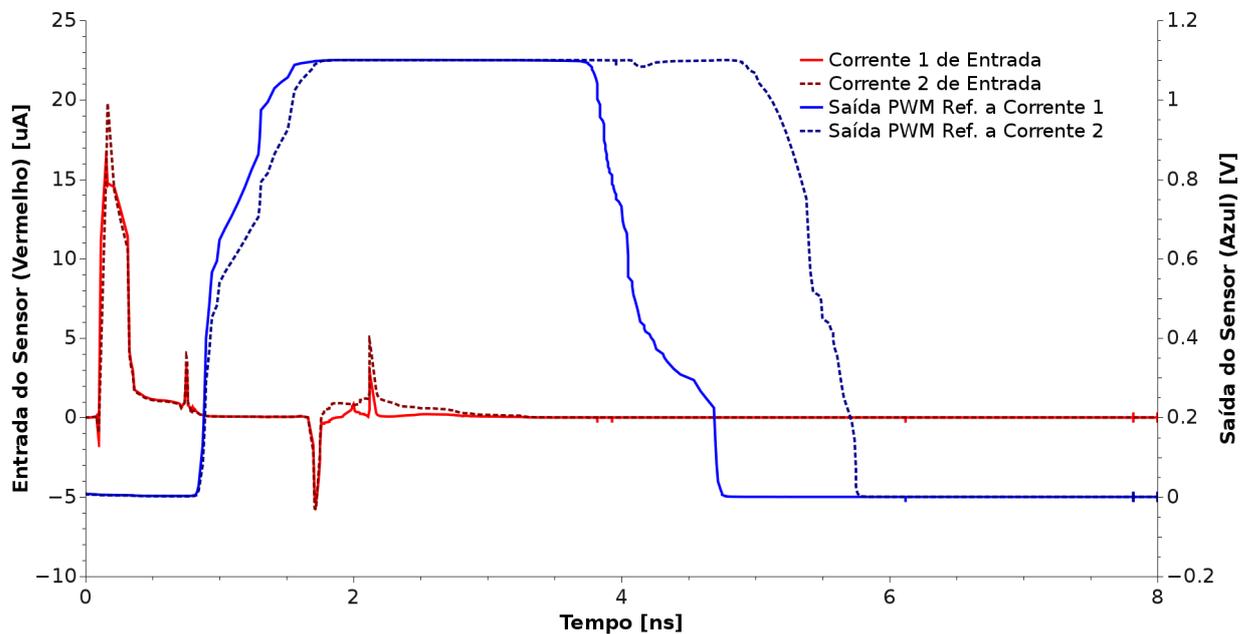


Figura 10 – Sinais de entrada e saída do sensor utilizando curvas de teste.

3.1.3 Lógica Comparadora de Vizinhança

A Lógica Comparadora de Vizinhança é o circuito digital responsável em comparar o comportamento dos membros de uma vizinhança entre si e apontar aquele ou aqueles que se comportarem diferentemente dos demais. Trata-se da principal parte da técnica proposta e apresenta um circuito capaz de implementar as regras da metodologia proposta com um custo computacional muito pequeno, é, portanto, justamente apropriado para o uso em testes de manufatura. A figura 11 ilustra todos os elementos da LCV.

A LCV recebe dos sensores de corrente o comportamento instantâneo de cada célula

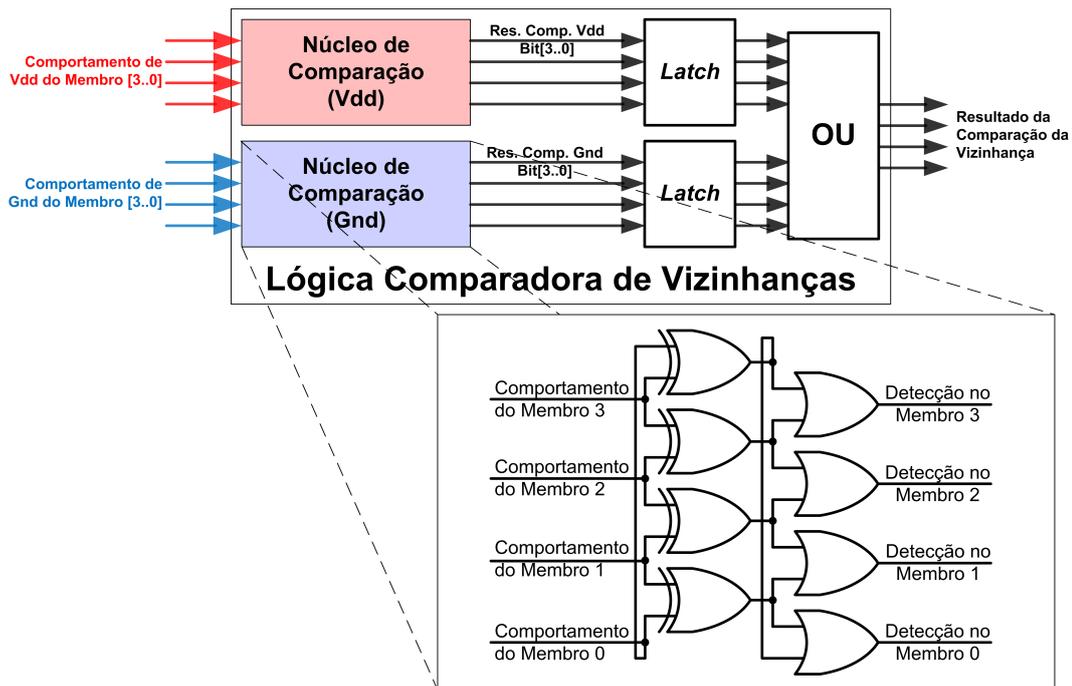


Figura 11 – Lógica Comparadora de Vizinhança e os seus circuitos interiores.

membro da vizinhança monitorada referente a I_{vdd} e a I_{gnd} , e com isso localiza o membro defeituoso de uma vizinhança ou aponta que existem dois ou mais membros defeituosos na vizinhança, porém desta vez sem localizá-los. Essas operações são efetuadas pelo Núcleo de Comparação.

3.1.3.1 Núcleo de Comparação

O comportamento de cada membro é comparado através de dois núcleos de comparação implementados, um para comparar os sinais de V_{dd} e o outro para os de G_{nd} . Cada um, usa portas lógicas que obedecem a rotina descrita pelo algoritmo 2 (vide figura 11), de maneira cíclica, em que o ultimo torna-se vizinho imediato do primeiro membro da vizinhança.

O algoritmo, assim como o circuito digital, funcionam da seguinte maneira: Primeiro ocorre a comparação com as células imediatamente vizinhas na LCV, esta comparação é executada pelas portas lógicas “ou exclusivo”, do núcleo de comparação, que retornam o valor 0 enquanto as suas entradas forem idênticas. E, depois, portas “ou” fazem a

Algoritmo 2 Pseudo-algoritmo do núcleo de comparação da LCV.

```

para cada Membro  $N$  da vizinhança
  se
    (Comportamento do membro  $N \neq$  Comportamento do membro  $N + 1$ );
    ou
    (comportamento do membro  $N \neq$  Comportamento do membro  $N - 1$ );
  então
    Membro  $N$  pode ser defeituoso, bit  $N$  do resultado da LCV = 1;
  senão
    Membro  $N$ ,  $N + 1$  e  $N - 1$  não são defeituosos, bit  $N$  do resultado da LCV = 0;
fim do loop

```

propagação preferencial do nível lógico 1. Assim, quando encontra-se algum valor igual a 1 na saída (“bit $N = 1$ ”, do algoritmo 2) interpreta-se que o membro pode ser defeituoso, pois este dado foi gerado por dois membros da vizinhança terem comportamentos diferentes entre si. Se a comparação da lógica retornar 0 (“bit $N = 0$ ”), então sabe-se que o respectivo membro teve o seu comportamento idêntico aos seus dois vizinhos imediatos, totalizando 3 comportamentos idênticos em um total de 4 membros, o que representa a maioria. Assim, na ocorrência de 0 na saída, garante-se que 3 membros têm o comportamento esperado e resta apenas, então, que o outro membro seja defeituoso, caso a sua saída seja diferente de 0. Essas comparações dão embasamento para os dados contidos na tabela 1 apresentada adiante.

A exemplo, a figura 12 demonstra a operação ao longo do tempo do núcleo de comparação da LCV com as suas entradas e saídas ilustradas. Observando a figura, percebe-se que enquanto os sinais de entrada foram idênticos, a saída da lógica foi nula, o que significa que nenhum membro apresentou comportamento diferente em relação aos demais membros, visto que os sinais de entrada apresentados na ilustração são idênticos, até o momento em que o comportamento do Membro 1 ($M1$) se difere dos demais (tempo marcado pelo fundo em cinza na ilustração), a lógica apresentou a saída binária “0111”, demonstrando as afirmações do parágrafo anterior, em que apenas o membro $M3$ gerou ‘0’ na saída da lógica, pois só ele teve seu comportamento idêntico ao dos seus vizinhos imediatos, assim, assume-se que $M2$, $M3$ e $M0$ não possuem defeitos, enquanto $M1$ é as-

sumido possuidor de defeitos, pois foi o responsável por gerar três bits da saída da lógica com valor '1'.

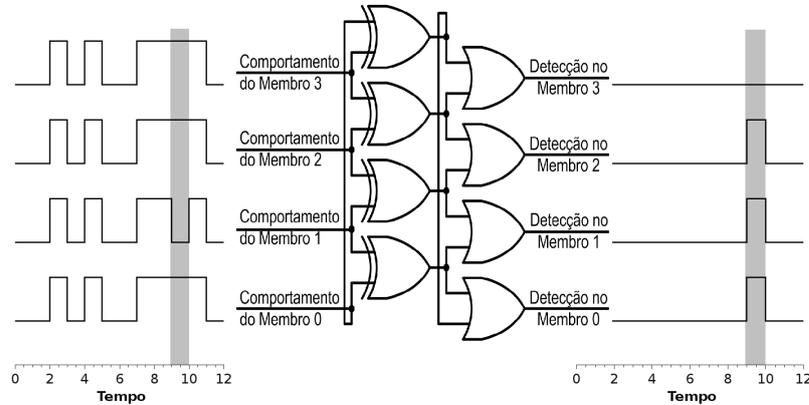


Figura 12 – Circuito lógico que implementa a LCV demonstrando o resultado da detecção quando o membro *M1* se comporta diferentemente dos demais entre o tempo 9 e 10.

Para abordar a sistemática de detecção de outra maneira, na lista a seguir é apresentado cinco possibilidades de comportamento dos membros comparados pelo núcleo de comparação e como a detecção ocorre:

1. Nenhum membro defeituoso: Nesta situação todos os membros tem exatamente o mesmo comportamento e a saída gerada é “0000”, como mostra a figura 12 nos tempos não marcados em cinza. Caracterizando “não existem membros defeituosos”.
2. Apenas um membro defeituoso: Nesta situação os três membros que se comportam corretamente geram a referência de comportamento correto, ou comportamento esperado, por se comportarem identicamente. Então, o comportamento do membro restante, por ser diferente, define três bits da saída do núcleo igual a '1'. Como mostra a figura 12 nos tempos hachurados em cinza. Caracterizando “Membro *M1* com defeito”.
3. Dois ou mais membros defeituosos: Agora, por não haver três membros com comportamento idêntico a referência de comportamento não será encontrada e a saída do núcleo de comparação será “1111”. Representando uma detecção sem localização,

podendo qualquer um dos membros da vizinhança ser defeituoso, e por isso o dado é interpretado como “Dois ou mais membros defeituosos”. Quando ocorre esta situação todas as células da vizinhança devem ser descartadas, assim nenhuma célula defeituosa escapa o teste de manufatura. Isso gera um desperdício de uma ou duas células sem defeitos.

4. Três membros identicamente defeituosos: Nesta situação os membros defeituosos apresentam comportamentos idênticos fazendo com que o núcleo de comparação tome esse como comportamento de referência. Isso gerará uma detecção e localização errônea, pois o membro que não participou da geração da referência, o sem defeitos, será acusado defeituoso, enquanto os que possuíam defeitos idênticos serão acusados sem defeitos pelo núcleo de comparação. As três células defeituosas escapariam o teste de manufatura nesta situação, e a célula sem defeitos seria desperdiçada.
5. Quatro membros identicamente defeituosos: Ocorre uma detecção errônea, onde os quatro membros identicamente defeituosos são considerados sem defeitos. As quatro células defeituosas escapariam o teste de manufatura nesta condição.

3.1.3.2 Saída da LCV

Dentro da LCV, os valores gerados pelo núcleo de detecção são capturados pelos *latches* e unidos pela porta “ou”, gerando apenas um sinal para identificar membros defeituosos em uma vizinhança. Na tabela 1 encontram-se todas as saídas possíveis da LCV e como essas saídas devem ser interpretadas pelo processador de BIST (encarregado em controlar o teste de manufatura).

3.1.3.3 Tamanho da Vizinhança

Quanto ao tamanho de vizinhança, foi adotado vizinhanças de quatro membros, pois este tamanho resulta nos melhores resultados em termos de custo/benefício. Esta parte

Tabela 1 – Saídas da LCV e o resultado da detecção.

Saída da LCV ($M_3M_2M_1M_0$)	Membros defeituosos ($M\#$)
0000	Nenhum
1110	Apenas M_2
1101	Apenas M_3
1011	Apenas M_0
0111	Apenas M_1
1111	Dois ou mais.

do trabalho apresenta as justificativas para esta escolha.

Para mostrar que este é o tamanho ideal de vizinhança, adote a hipótese de termos vizinhanças de três membros. Assim o algoritmo ou lógica teriam que ser modificados de maneira a obter a referência de comportamento correto de apenas dois membros, já que três membros representariam a totalidade e não a maioria, como ocorre nas vizinhanças de quatro membros. Assim fazendo, as chances de se obter uma referência errônea aumenta, pois agora só dois membros contribuem para geração desta referência de comportamento esperado, e quanto menos membros contribuírem, maior a probabilidade deste ser errado, caso membros apresentem comportamentos igualmente errôneos. Por exemplo, numa vizinhança de três membros necessitaria-se que dois membros apresentem comportamento igualmente errôneos para gerar uma referência errada, enquanto numa vizinhança de quatro membros precisaria-se que isso ocorresse em três membros, enquanto na de sete membros seria necessário que ocorresse em quatro membros, e assim por diante. Então, neste ponto de vista, quanto maior a vizinhança, mais robusta é a referência obtida.

Entretanto, quando maior a vizinhança, muito maior será o consumo de área da LCV, pois veja bem, em uma vizinhança de 3 membros, faz-se comparações de dois a dois membros em um total de três. Em outras palavras, o número das comparações que precisam ser efetuadas se dá pela operação de combinação. Então em uma vizinhança de três membros combinados dois a dois, tem-se três comparações, já em uma vizinhança de tamanho 4, tem-se 4 comparações 3 a 3. Para cada tamanho de vizinhança a tabela 2 mostra o número de comparações necessárias e quantos membros participam em cada

comparação além dos demais dados representados.

Tabela 2 – Diferentes tamanhos de vizinhança e os custos e benefícios de cada uma. Sendo os custos compostos pelo número e tamanho das comparações e os benefícios pela confiabilidade da referência obtida normalizada.

Tamanho da Vizinhança	Número de Comparações	Tamanho das Comparações	Confiabilidade da Referência	Capacidade de Localização
3	3	2 a 2	0.67	1 membro em 3
4	4	3 a 3	1	1 membro em 4
5	10	3 a 3	1	2 membros em 5
6	20	4 a 4	1	2 membros em 6
7	35	4 a 4	1.33	3 membros em 7

Baseado nos dados apresentados pela tabela foi escolhido que cada vizinhança deve ter o tamanho de 4 membros, pois é o que oferece melhor custo/benefício. De outra forma, para se obter um incremento na confiabilidade dessa referência seria necessário usar uma vizinhança de seis membros e pagar o alto custo de se executar 20 comparações de 4 a 4. A vizinhança de 4 membros se mostrou a que tem maior confiabilidade da referência obtida pelo menor custo de comparação, e por isso que a técnica proposta foi elaborada com este tamanho de vizinhança. Entretanto este tamanho tem o poder de apenas localizar 1 defeito por vizinhança, embora pareça uma limitação grave, o capítulo Resultados Experimentais demonstra que a localização de apenas 1 membro por vizinhança possui um excelente desempenho.

A estrutura de detecção de defeitos, o sensor de corrente e a LCV foram apresentados, pertinente a técnica de detecção de defeitos resta apenas compreender como ocorre o teste de manufatura.

3.1.4 Teste de Manufatura Proposto

Um teste de manufatura deve ser empregado para efetuar detecções dos defeitos utilizando a estrutura de detecção proposta, como tal, será o responsável em controlar o acesso às células da memória junto com a operação da estrutura de detecção e será o encarregado em ler os dados gerados pelas LCVs.

O teste proposto neste trabalho é muito semelhante ao *March test*. Um algoritmo de acessos é utilizado para excitar identicamente as células monitoradas aplicando uma mesma operação, porém, diferentemente do *March test*, o defeito é detectado pela estrutura de detecção e não pelo algoritmo do teste em si.

Esse algoritmo *March* necessita efetuar todas as quatro operações que uma célula poderia executar durante o uso na sua aplicação para garantir que qualquer defeito do tipo *Resistive-Open* seja excitado, e, assim, distorcer o consumo de corrente durante o teste fazendo com que a detecção ocorra. Operações que são: escrita de 0 para 1 ($0w1$) e de 1 para 0 ($1w0$) e também leitura de 0 ($r0$) e leitura de 1 ($r1$), ou seja, o BIST deve executar $(w1)(w0,r0,w1,r1)$ em cada célula da linha da SRAM. Ao início da execução deste algoritmo em cada endereço, os *latches* das LCVs precisam ser reiniciados, esta operação é representada arbitrariamente pela notação rL , concluindo, então, o *March test* que deve ser executado pelo BIST, que associado a estrutura apresentada forma a técnica de detecção inovadora e que está representado no algoritmo 3, porém, diferentemente do *MarchSS* que acessa uma célula por vez, este acessa todas as células de uma linha paralelamente, sendo, ainda, mais rápido que os algoritmos convencionais, além de possuir apenas 5 operações por linha. Este ganho em desempenho que a técnica apresenta ocorre graças ao uso de um sistema detector de defeitos acoplado na memória.

Algoritmo 3 Algoritmo *march* para ser usado junto a técnica detectora de defeitos do tipo *Resistive-Open*.

$$\{\uparrow(w1); \downarrow(rL,w0,r0,w1,r1); \}$$

Por fim, a cada endereço acessado, as células da SRAM são avaliada pela técnica segundo o seguinte procedimento:

1. Primeiro, todos os endereços são acessados com a operação $(w1)$ para conferir a condição inicial esperada às células;
2. Após, começam as detecções em cada linha. Primeiro o BIST zera os *latches* e depois executa o bloco de algoritmo *March* $(w0,r0,w1,r1)$;

3. Ao final dos acessos da linha, os dados gerados pelas LCVs são lidos e processados pelo BIST. Esses dados contêm a informação a respeito das células acessadas que são defeituosas;
4. Depois que todas as linhas forem acessadas, o BIST terá a informação de todas as células da memória que contenham defeitos *Resistive-Open*. Então se terá a opção de efetuar a substituição de linhas e/ou colunas para recuperar a robustez da memória testada, caso técnicas do gênero sejam empregadas.

E assim terminam as explicações sobre a técnica detectora de defeitos proposta, restando apenas validá-la e analisá-la.

4 VALIDAÇÃO

Com tudo que já foi apresentado, este capítulo visa comprovar o funcionamento da técnica proposta, ou seja, comprovar que é apta a detectar defeitos através do seu sensor de corrente, que gera um dado digital contendo informações a respeito do consumo de corrente das células acessadas, e da LCV, que processa estes dados para descobrir qual célula possui um defeito.

Para a tarefa foi necessário desenvolver uma memória, referenciada no trabalho como memória estudo de caso, e então acoplada à estrutura de detecção, recebeu células com defeitos para demonstrar como ocorre o processamento dos sinais na técnica proposta.

A seguir a memória estudo de caso é apresentada seguida da validação da técnica.

4.1 Memória Estudo de Caso

A matriz de células desta memória tem apenas 8 colunas e 1024 linhas, formando uma memória de $8kBit$ ou $1kByte$. E quanto aos decodificadores de linha e coluna, este trabalho não se limitou em projetá-los, porque testá-los frente aos defeitos não faz parte do escopo desta dissertação, então, apenas a matriz de células e os circuitos necessários para operá-la foram projetados (cujos são os mesmos representados na figura 1 do capítulo 2), enquanto os sinais provenientes dos decodificadores foram gerados manualmente (*wordline* 0 ao *wordline* n).

O circuito de pré carga foi dimensionado para garantir a completa carga dos *bitlines* no tempo de $0,7ns$, referente ao tempo disponível da primeira etapa de uma operação

de escrita ou leitura da SRAM, o amplificador de descarga descarrega um dos *bitlines* em $1ns$, referente ao tempo da segunda etapa, e a célula e o circuito de leitura foram projetados para assimilar o dado em $0,3ns$, referentes ao tempo disponível da última etapa das operações. Somando os tempos das três etapas obtém-se o período de operação da memória de $2ns$ e a frequência de $500MHz$.

O projeto da célula 6T foi baseado no livro *Analysis and design of digital integrated circuits* [12, Pág. 374], onde é sugerida uma metodologia de dimensionamento dos transistores da célula trabalhando com a proporção dos transistores dos inversores em relação aos transistores nMOS de acesso. Baseado em cálculos matemáticos, o autor recomenda a proporção de $R_P = R_C = 1.5$, ou seja, os transistores dos inversores possuindo o dobro da largura dos transistores de acesso, pois $R_P = \frac{W_P}{W_a}$ e $R_C = \frac{W_N}{W_a}$, onde W_P , W_N e W_a são respectivamente a largura dos transistores de *pull-up* (pMOS), *pull-down* (nMOS) e dos de acesso (nMOS).

Já para manter a área e o consumo controlado foi escolhido W_a mínimo¹, e então R_C e R_P foram modificados até se obter uma célula funcional com o *Static Noise Margin* (SNM) desejado, resultando nas relações de $R_P = 1,83$ e $R_C = 1,67$ para que os transistores da célula garantam a correta operação em qualquer variação de processo a $500MHz$ e com o SNM de leitura de $200mV$. A figura 13 apresenta a célula projetada com os transistores já dimensionados.

Static Noise Margin (SNM) refere-se a uma unidade de medição de robustez das células, e, em termos práticos, define o valor que a célula tolera de ruído durante a leitura, escrita ou repouso. Sendo que o momento mais sensível da operação da célula é durante a leitura, escolheu-se manter o SNM de leitura em torno de $200mV$ baseando-se nos valores encontrados na literatura, como na memória desenvolvida por Ohbayashi, S. et al. [26]. Já para medir o SNM da célula foi utilizado um modelo implementado em simulador [27], onde a célula é fixada na posição de leitura e excitada com fontes de ruídos

¹A largura mínima do poço na tecnologia utilizada é $120nm$, quando o tamanho não for identificado nas ilustrações assume-se tamanhos padrões de $W_P = 250nm$, $W_N = 120nm$ e $L = 65nm$.

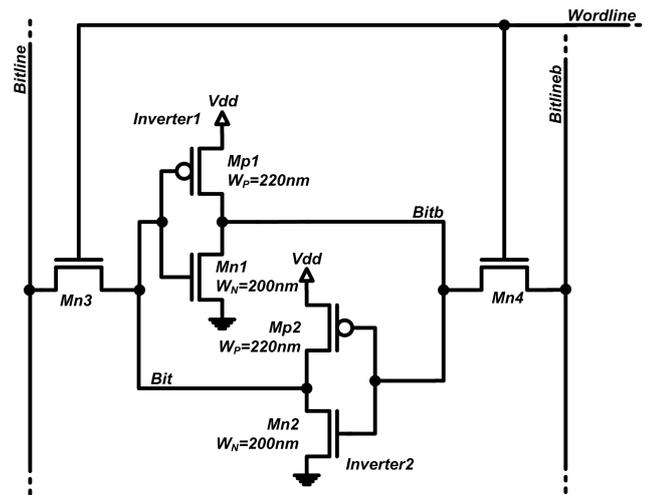


Figura 13 – Celula da SRAM projetada.

dependentes das saídas dos inversores, que modela em circuito o cálculo matricial para encontrar o SNM de uma célula. Porém, mais detalhes não fazem parte do escopo desta dissertação.

Ao final, a SRAM projetada foi testada, considerando todas as variações *inter-die* do processo de manufatura, para garantir que tolerasse as variações naturalmente inseridas pelo processo simulado e pela temperatura. O sinal do dado armazenado em uma célula e o sinal do dado de saída estão ilustrados na figura 14, onde em cor fraca estão sobrepostas curvas para cada *corner* (FF², SS, SF, FS e TT a -40°C, 27°C e 100°C cada) e destacado em preto os sinais referente a variação típica em temperatura ambiente (TT a 27°C).

O objetivo dessa figura é demonstrar que a memória projetada opera em qualquer um dos *corners*, assim sendo, o comportamento correto pode ser observado para cada operação, pois nas operações de escrita o dado armazenado na célula foi alterado corretamente, e, nas de leitura, o dado correto foi retornado.

Após o dimensionamento da célula, foi medido que o seu *leakage* é de $0,5nA$. Quando a técnica de detecção monitorar a corrente de uma coluna, ela estará na prática monitorando a corrente da célula de interesse acrescida da corrente de 1023 células em repouso, nas quais totalizam $0,51\mu A$, portanto, a corrente total de *leakage* é insignificante diante do consumo

²F=Fast, S=Slow e T=Typical, onde FS significa que os transistores pMOS estão no *corner fast* enquanto os nMOS estão no *slow*.

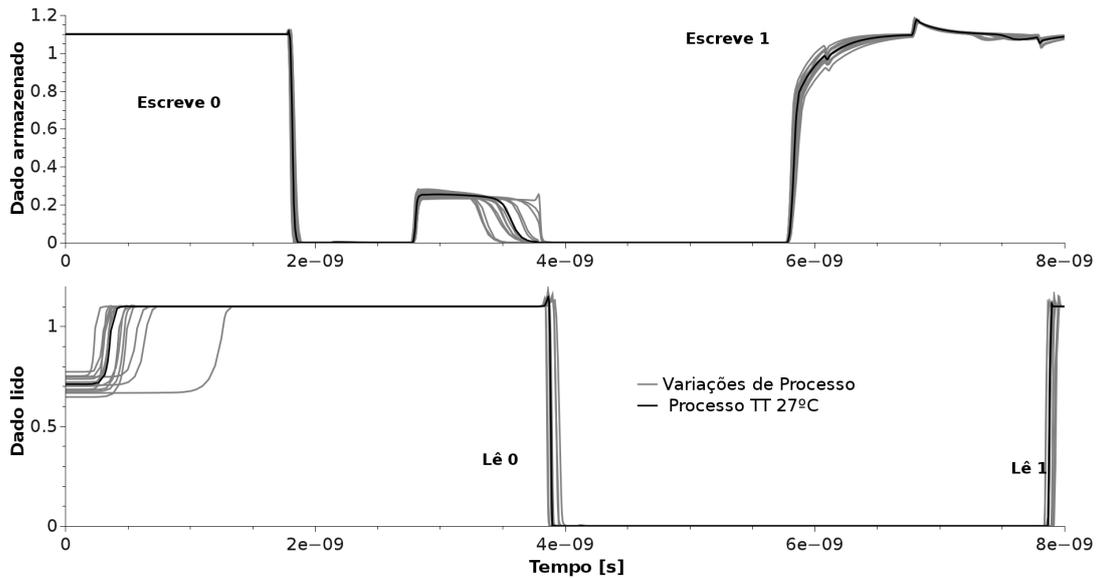


Figura 14 – Valor armazenado (Bit) e dado de saída para todos os *corners* durante a rotina escreve 0, lê 0, escreve 1 e lê 1.

dinâmico de uma célula que chega a $20\mu A$ e $40\mu A$ para V_{dd} e G_{nd} respectivamente (vide figura 5 no capítulo 2).

4.1.1 Injeção de Defeitos na SRAM Estudo de Caso

Uma análise foi efetuada considerando a presença individual dos defeitos DF1 ao DF6 (da figura 3 no capítulo 2) nas células da memória SRAM projetada, utilizando o simulador HSPICE para simular e gerar os dados a respeito da operação de todo o circuito. A figura 15 apresenta a relação entre falha funcional e tamanho de defeito para cada um dos seis defeitos. E, observando essa figura, percebe-se que o DF2 e o DF3, enquanto suficientemente fracos, causaram falhas dinâmicas que necessitaram de 4 operações consecutivas para ser sensibilizadas³. Enquanto defeitos de tamanho maior, passaram a causar apenas falhas funcionais do tipo estáticas.

Entretanto a figura não apresenta falhas para o DF4, porque, embora tenha degradado a robustez da memória, esse defeito não causou nenhuma falha funcional. Contudo, em outros trabalhos [13], o DF4 causou falhas funcionais e inclusive de natureza dinâmica,

³Falhas dinâmicas de maiores operações não foram sensibilizadas devido ao passo de $1k\Omega$ utilizado nas simulações para gerar esses gráficos da figura.

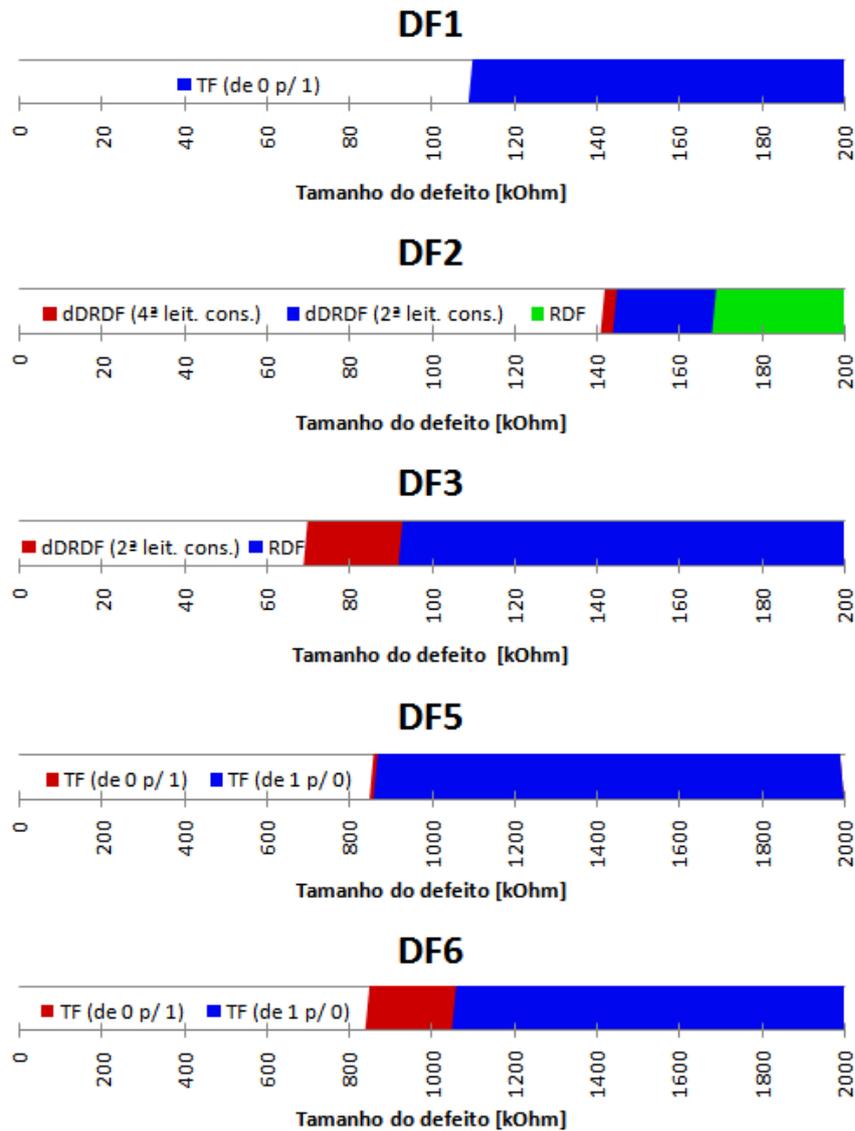


Figura 15 – DF1-6 e suas falhas funcionais para diferentes tamanhos de defeito, obtidos via simulações com step de $1k\Omega$ e frequência de operação da memória de $200MHz$.

enquanto que, na memória projetada isso não ocorreu, pois o seu dimensionamento não permite que uma inversão ocorra devido ao DF4 por causa da relação R_c dos transistores da célula. Tal divergência é natural, já que o tipo de falha inserida por cada defeito é dependente das condições dinâmicas da célula de memória, ou seja, é dependente do dimensionamento e da tecnologia em que ela foi desenvolvida.

4.2 Validação da Técnica Proposta

A validação da técnica de detecção de defeitos do tipo *resistive-open* utilizou a memória estudo de caso contendo três condições diferentes de defeitos diferentes que estão explicados na lista após o parágrafo, entretanto apenas uma linha da memória foi acessada, pois não há a necessidade de percorrer toda a memória em busca de defeitos para validar a técnica, e sim, apenas uma vizinhança de um endereço é o suficiente. Assim, para todas condições, apenas a vizinhança 0 foi considerada, cuja é formada pelas colunas 0, 2, 4 e 6 da memória e contém a seguinte configuração de membros defeituosos em cada uma das três condições simulados:

- Condição 1: O quarto membro da vizinhança, possuidor dos sinais “ $I_{vdd} 6$ ” e “ $I_{gnd} 6$ ”, recebeu o defeito DF3 de tamanho $122,11k\Omega$, capaz de causar uma falha funcional do tipo dRDF na oitava leitura consecutiva, enquanto os demais membros não possuem defeitos.
- Condição 2: Além do quarto membro com o defeito citado na condição anterior, o terceiro teve o DF3 de tamanho $100k\Omega$ em seus circuitos, enquanto o primeiro e o segundo membro continuam sem defeitos;
- Condição 3: Sem membros defeituosos.

O simulador HSPICE foi utilizado para executar as três simulações da lista, executando a operação $(1w0,rL,r0,w1,r1)$ a uma frequência de $500MHz$, utilizando a biblioteca do processo de manufatura da STMicroelectronics do nodo tecnológico $65nm$ no *corner* TT $27^{\circ}C$.

Nos próximos parágrafos desta sessão, todos os sinais envolvidos na detecção de defeitos gerados pelo HSPICE são ilustrados e explicados detalhadamente, entretanto, antes é necessário compreender a figura 16, onde a LCV0 é representada junto com os sensores e todos os barramentos de dados, cada barramento tem o seu conteúdo posteriormente

ilustrado nas figuras 17, 18 e 19, para a condição 1, 2 e 3 respectivamente. Em todas as ilustrações a legenda (a), (b), (c), (d) e (e) é seguida.

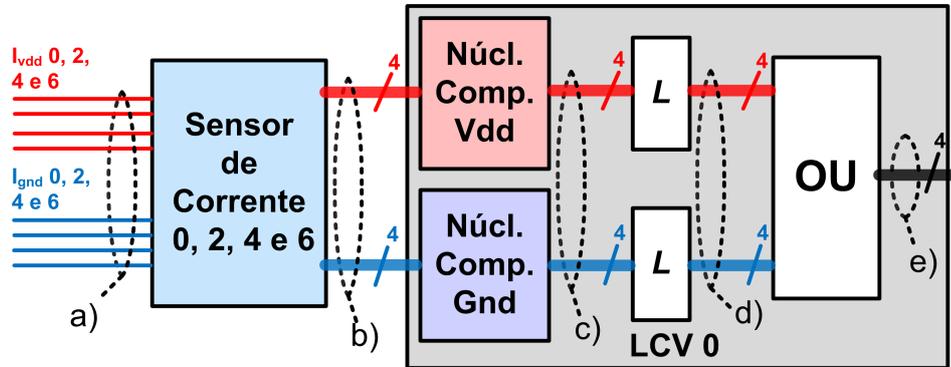


Figura 16 – Representação dos circuitos de detecção na vizinhança 0, onde os sinais (a), (b), (c), (d) e (e) são apresentados adiante para cada caso simulado.

4.2.1 Condição 1

A lista a seguir, apresenta explicações a respeito dos sinais encontrados na figura 17, que é proveniente da simulação executada considerando a condição 1.

- Em (a), constam os sinais de correntes proveniente das células. Essas são as correntes que servem de entrada para os OCCSs. As correntes I_{vdd} e I_{gnd} do quarto membro estão tracejadas em amarelo e azul respectivamente. Observando-as percebe-se que devido a presença do defeito o consumo foi diferente do consumo dos demais membros da vizinhança, apresentados em preto. Também na ilustração, setas apontam os locais em que essas correntes se diferiram;
- Já em (b), tem-se o sinal digital gerado pelos sensores de correntes que representam o comportamento observado nas correntes da ilustração (a). Como esperado, os sinais “ PWM_{vdd} 6” e “ PWM_{gnd} 6”, referentes ao comportamento de Vdd e Gnd do membro defeituoso da vizinhança, se mostraram diferentes dos demais nos locais apontados pelas setas. Agora esses dois sinais de 4 bits cada deve ser processado pelos núcleos de comparação da LCV0, que deve acusar o quarto membro como defeituoso;

- Em (c), após os núcleos de comparação executarem esse processamento, tem-se, em um determinado momento o valor “1101” para os sinais de V_{dd} e G_{nd} , informando que o quarto membro se comportou diferentemente dos demais (vide tabela 1, item 3.1.3.2). Por fim, os sinais desta ilustração são enviados aos *latches*;
- E em (d), os sinais de saída desses *latches* são apresentados, que mostra o dado “1101” sendo armazenado;
- Finalmente em (e), o resultado da detecção da LCV0 é apresentado após passar por uma porta “ou” que uniu os sinais de (d). E assim encerrando a detecção de defeitos nessa vizinhança com sucesso, pois retornou o valor “1101”, informando que “Apenas o membro M4 é defeituoso”. No caso da memória estudo de caso, trata-se da célula acessada na coluna de índice 6, exatamente a que a célula cessada possuía defeito.

4.2.2 Condição 2

Nesta condição, o processo de detecção do membro defeituoso ocorre exatamente como na lista recém apresentada da condição anterior, entretanto, para esta, diferenças são encontradas nos sinais processados, pois agora existem dois membros defeituosos. Segue abaixo a figura 18 contendo todos os sinais gerados pelo simulador durante a detecção e a lista explicando como ocorreu a detecção.

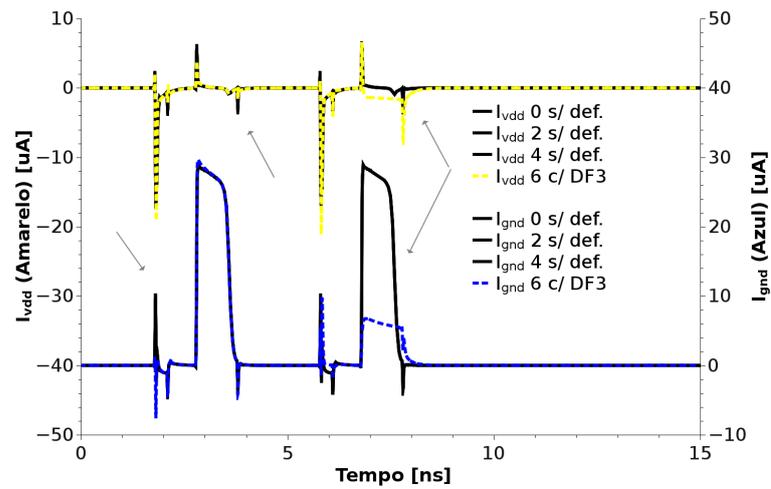
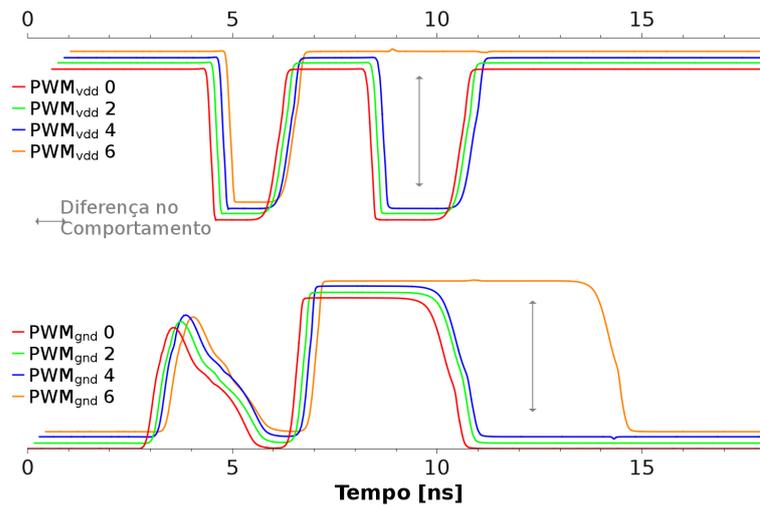
- Em (a), as correntes do segundo membro defeituoso adicionado nesta condição foram exibidas em cinza e em traço não contínuo. Observa-se que existem 3 comportamentos de consumo de corrente no total dessa vizinhança: as curvas em preto apresentam um comportamento, as em cinza outro e o terceiro em azul e amarelo. Então, segundo o algoritmo de comparação utilizado, espera-se obter a saída “1111” no final da comparação, informando que existem dois ou mais membros defeituosos nesta vizinhança.
- Em (b), dois dos sinais PWM se diferem dos de comportamento correto, o azul e o laranja passam a se diferenciar dos sinais vermelho e verde;

- Para então em (c), essas diferenças ser detectadas pelo núcleo de comparação, gerando detecções sucessivas, apontando que “Apenas o membro $M2$ é defeituoso” em $5ns$ e $14ns$, e que “Existem dois ou mais membros defeituosos” entre $9ns$ e $13,5ns$;
- Detecções nas quais, passando pelos *latches*, são fixadas em “1111”, omitindo a localização do defeito no membro $M2$, ou de índice 4, para ambos os sinais monitorados (Vdd e Gnd).
- E, finalmente em (e), o resultado da comparação é o esperado, ou seja, '1111'.

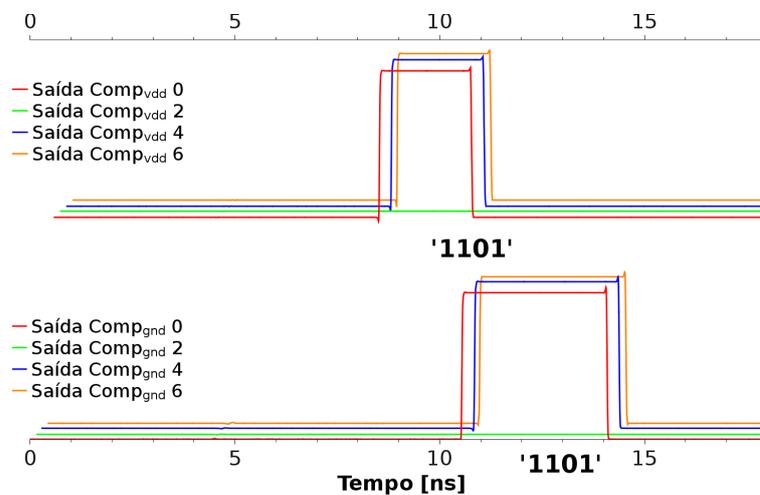
4.2.3 Condição 3

Já na terceira condição, onde nenhum membro da vizinhança possui defeitos, espera-se que o resultado da comparação seja “0000”, ou “Nenhum membro defeituoso”, e para que isto ocorra, as correntes devem ser semelhantes assim como os sinais PWM consequentes, fazendo com que o resultado da comparação, no núcleo na LCV, seja sempre “0000” ao longo do tempo. Como mostra a figura 19, demonstrando que na ausência de células defeituosas, a técnica proposta atua corretamente.

A partir dos três casos apresentados, observou-se toda a evolução e processamento dos sinais envolvidos na detecção de uma vizinhança para as três situações possíveis, e que o resultado da detecção foi exatamente o esperado: nenhuma detecção deve ocorrer quando não houver membros defeituosos, detecção e localização deve ocorrer quando existir apenas um membro com defeito, e detecção sem localização deve ocorrer se existir mais de um membro defeituoso na vizinhança. Entretanto a condição de falha da metodologia não é apresentada, pois do ponto de vista da detecção não há diferença nos sinais de membros sem defeitos para membros identicamente defeituosos.

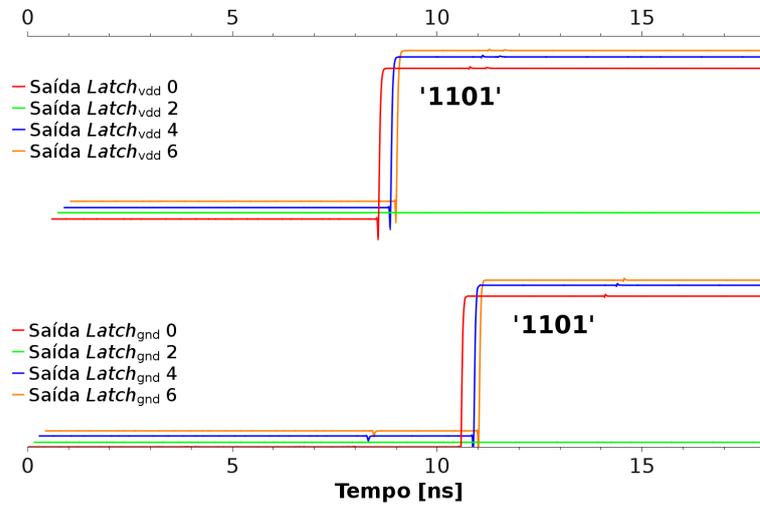
(a) Correntes I_{vdd} e I_{gnd} das células da vizinhança

(b) Sinal PWM gerado pelos sensores de corrente, para cada corrente de entrada.

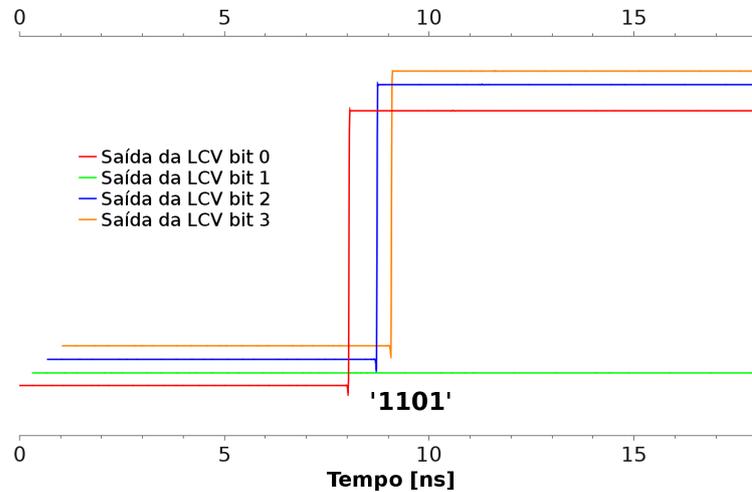


(c) Resultado da comparação efetuada pelo núcleo da LCV.

(Continua na próxima página)

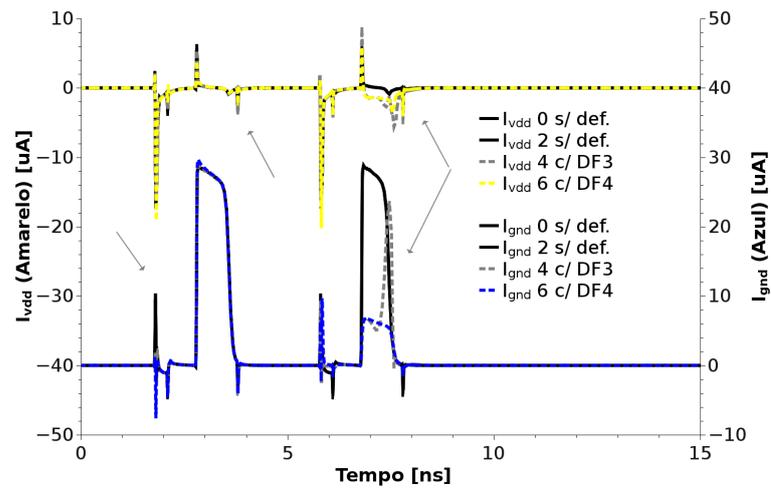
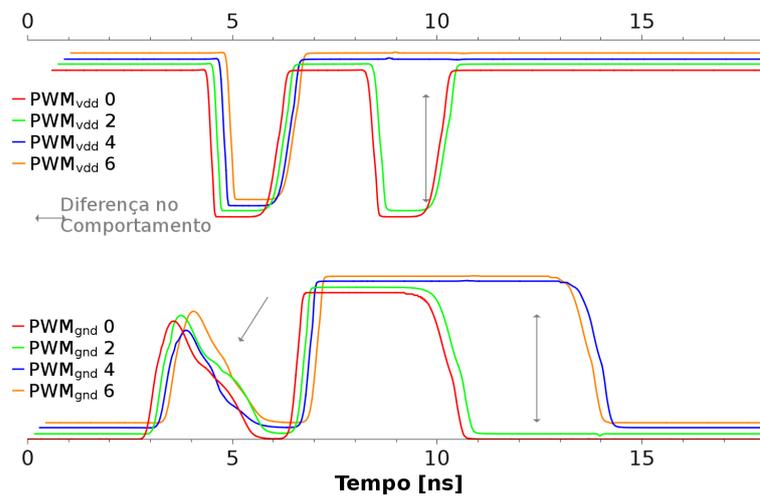


(d) Resultado da comparação após passar pelos *Latches*.

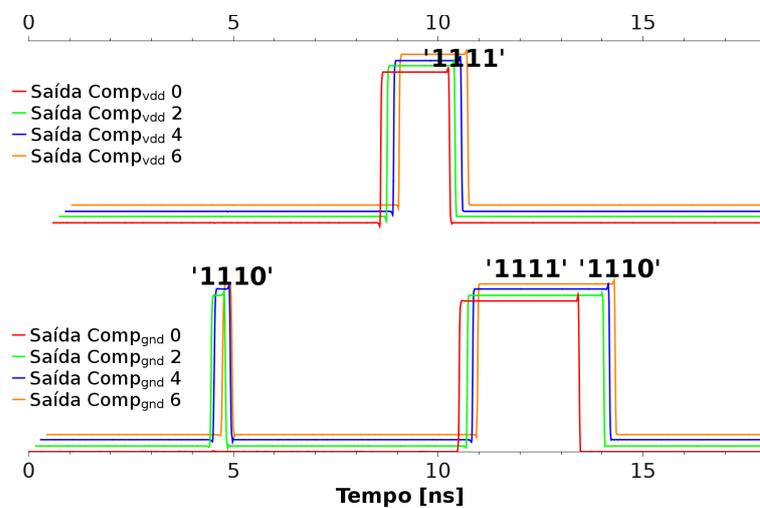


(e) Saída final da LCV contendo o resultado da detecção.

Figura 17 – Ilustração do processamento de sinais referente a todas as etapas de detecção de defeitos do tipo *Resistive-Open* da proposta para uma vizinhança onde apenas o membro de índice 6 é defeituoso.

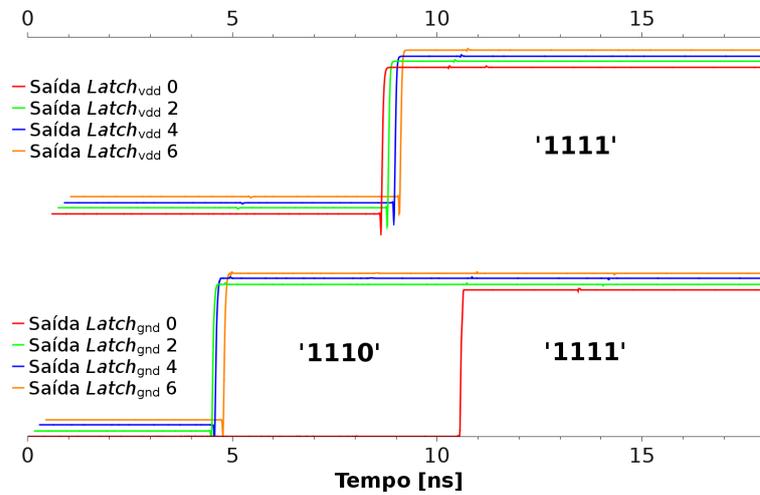
(a) Correntes I_{vdd} e I_{gnd} das células da vizinhança

(b) Sinal PWM gerado pelos sensores de corrente, para cada corrente de entrada.

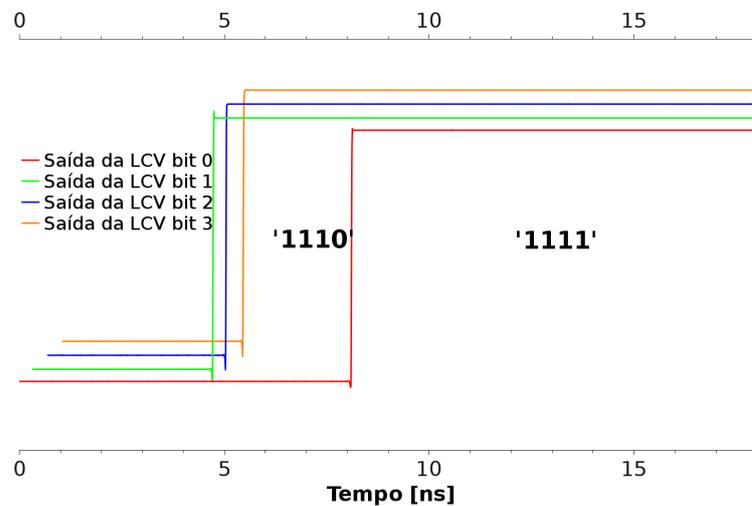


(c) Resultado da comparação efetuada pelo núcleo da LCV.

(Continua na próxima página)

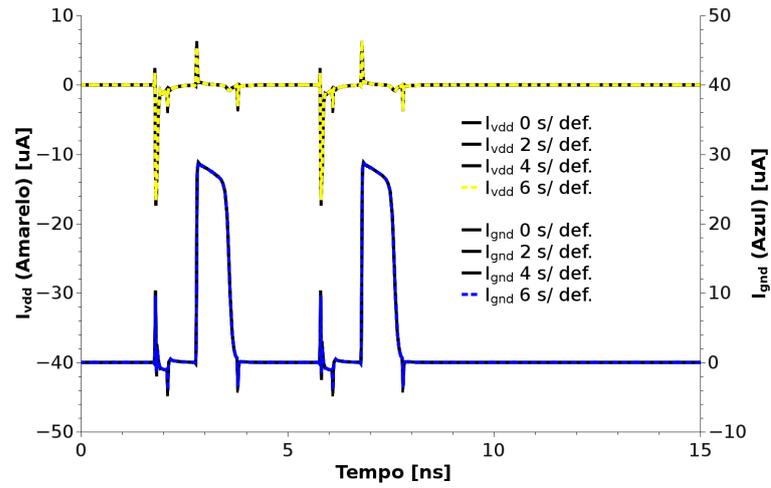


(d) Resultado da comparação após passar pelos *Latches*.

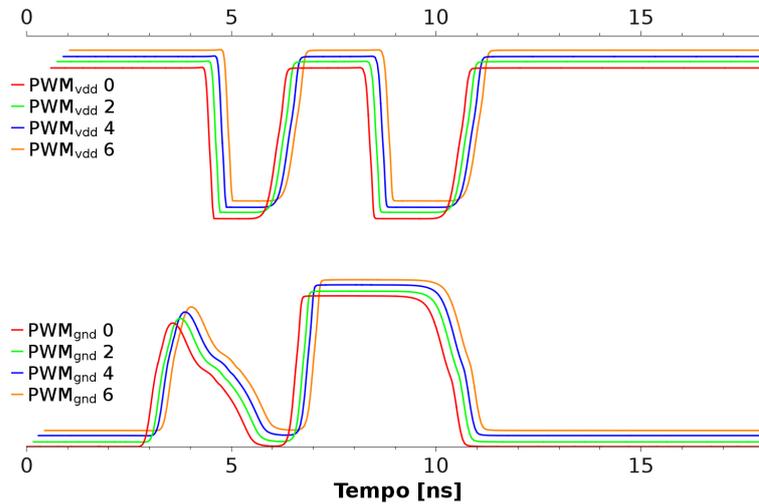


(e) Saída final da LCV contendo o resultado da detecção.

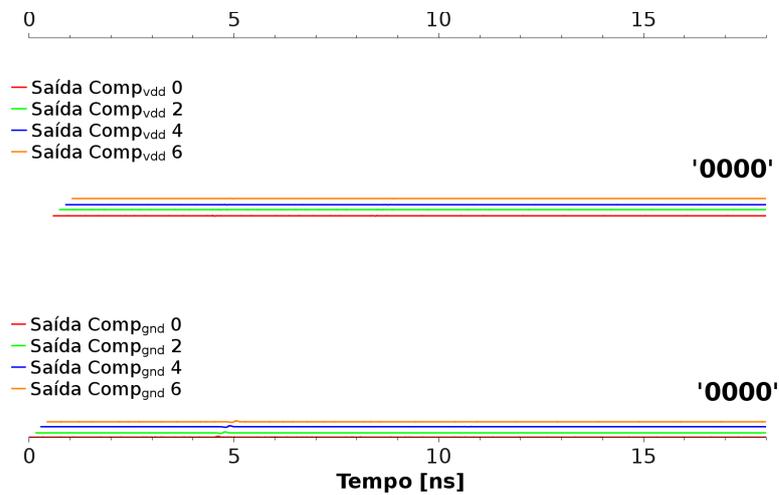
Figura 18 – Ilustração do processamento de sinais referente a todas as etapas de detecção de defeitos do tipo *Resistive-Open* da proposta para uma vizinhança onde os membros de índices 4 e 6 são defeituoso.



(a) Correntes I_{vdd} e I_{gnd} das células da vizinhança

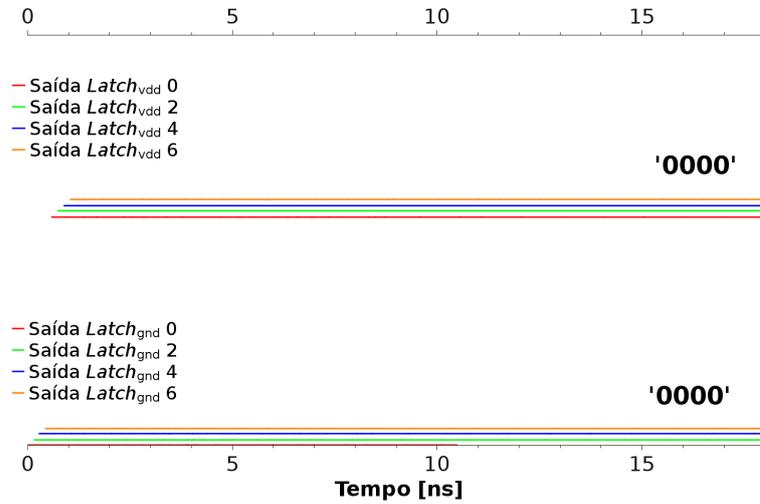


(b) Sinal PWM gerado pelos sensores de corrente, para cada corrente de entrada.

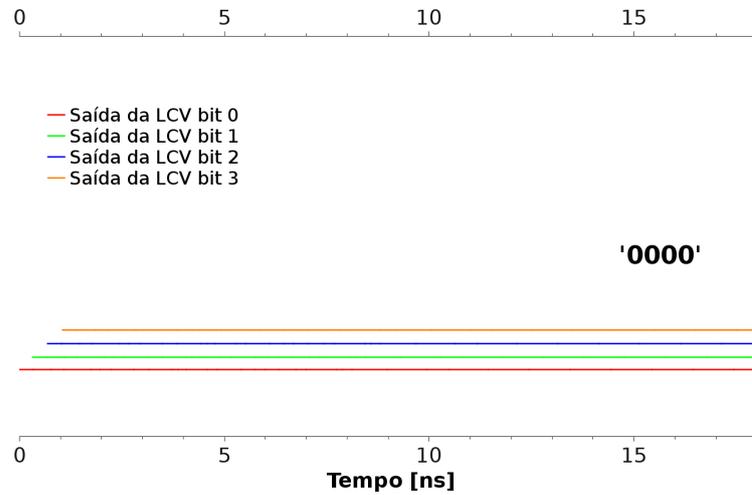


(c) Resultado da comparação efetuada pelo núcleo da LCV.

(Continua na próxima página)



(d) Resultado da comparação após passar pelos *Latches*.



(e) Saída final da LCV contendo o resultado da detecção.

Figura 19 – Ilustração do processamento de sinais referente a todas as etapas de detecção de defeitos do tipo *Resistive-Open* da proposta para uma vizinhança onde não há membro defeituoso.

5 *RESULTADOS EXPERIMENTAIS*

Os resultados desta dissertação consistem na análise experimental da técnica proposta. A técnica foi analisada frente a sua capacidade de detectar defeitos do tipo *Resistive-Open* fracos e fortes, seu desempenho global considerando a limitação intrínseca da metodologia de detecção e os custos associados ao uso da técnica.

5.1 Capacidade de Detecção

Para avaliar a capacidade de detecção da técnica, foi utilizada a mesma memória estudo de caso anteriormente apresentada. Nestes experimentos, apenas a célula da última coluna recebeu defeitos. Apenas um endereço foi acessado, a operação efetuada também foi a mesma: $(1w0,rL,r0,w1,r1)$ e o simulador HSPICE, com a mesma biblioteca de $65nm$, foi utilizado novamente.

No experimentos deste tópico, os defeitos introduzidos na última coluna (de índice 7) da SRAM foram os seis defeitos modelados (DF1-DF6), um de cada vez, com impedância de $50K\Omega$ e $10M\Omega$ cada, pois o objetivo deste experimento é testar se a técnica é capaz de detectar os defetos, sendo fraco ou fortes, nas seis posições consideradas.

Como os defeitos foram inseridos apenas na coluna de índice 7, ou seja, o quarto membro da LCV1, espera-se que ao final de cada detecção o dado “1101” seja retornado pela LCV1, e o dado “0000” pela LCV0. Os resultados encontram-se na tabela 3, em que a última coluna mostra o momento em que a LCV estabilizou com o resultado da detecção.

Assim sendo, a tabela demonstra que a técnica proposta tem capacidade de detectar

Tabela 3 – Capacidade de detecção da técnica para as 6 posições e 2 tamanhos de defeitos.

Posição	Defeito		Saída da:		Momento da Detecção
	Tamanho		LCV0	LCV1	
DF1	50kΩ		'0000'	'1101'	5,14ns
DF1	10MΩ		'0000'	'1101'	3,05ns
DF2	50kΩ		'0000'	'1101'	8,04ns
DF2	10MΩ		'0000'	'1101'	4,71ns
DF3	50kΩ		'0000'	'1101'	8,06ns
DF3	10MΩ		'0000'	'1101'	8,06ns
DF4	50kΩ		'0000'	'1101'	4,88ns
DF4	10MΩ		'0000'	'1101'	4,15ns
DF5	50kΩ		'0000'	'1101'	4,75ns
DF5	10MΩ		'0000'	'1101'	3,05ns
DF6	50kΩ		'0000'	'1101'	4,69ns
DF7	10MΩ		'0000'	'1101'	3,06ns

todos os 6 defeitos considerados, sejam eles de tamanho fraco ou forte.

5.2 Desempenho

O objetivo desta etapa do trabalho é demonstrar que a limitação da metodologia tende a não afetar o desempenho da técnica proposta em situações reais. Entende-se por situações reais, memórias produzidas cujo processo possui uma probabilidade realista de produzir células defeituosas, e que possam ser inseridos defeitos em qualquer posição da célula com qualquer tamanho, entretanto, “possibilidade realista” e “qualquer tamanho” são valores difíceis de estimar, ao contrário de “qualquer posição”, que para este trabalho são considerada apenas seis, já que sozinhas sensibilizam todas as falhas possíveis de ocorrer na célula da SRAM. Para contornar esse obstáculo em relação a essas duas variáveis, é feita uma análise de tendência, onde diversos valores são tomados para ambas, assim, fica evidente como é a influência delas no desempenho da técnica proposta e torna possível que seja estabelecido um critério, no qual define em quais processos de fabricação a técnica

proposta pode ser utilizada.

Para relembrar, a limitação referida da técnica é definida pelos seguintes itens:

1. A LCV só é capaz de detectar e localizar um defeito quando ele ocorre em uma célula apenas da vizinhança.
2. Quando não houver três membros com comportamento idêntico a LCV não consegue detectar quais são os membros com defeitos e os sem defeitos (na ocorrência desta situação é necessário adotar que todos os membros da vizinhança são faltosos, caso contrário células com defeitos escapariam do teste de manufatura).
3. Quando houver três ou mais membros com defeitos idênticos a LCV fará uma falsa detecção, acusando as células sem defeitos como defeituosas, e o mais grave, acusando células defeituosas como sem defeitos.

Essas limitações são fruto da metodologia de detecção utilizada, cuja obtém da própria vizinhança a referência de comportamento correto e que precisa que a maioria dos membros se comportem semelhantemente para que essa referência seja adotada.

Foi desenvolvido um simulador especialmente para a função que tem o objetivo de analisar o comportamento da técnica nesses cenários realistas. O simulador implementa todas as operações digitais referentes a técnica de detecção de defeitos e também monta uma vizinhança de células virtual que é populada com células com ou sem defeitos a partir de sorteios aleatórios, caracterizando uma abordagem de Monte Carlo.

Antes de apresentar a análise efetuada sobre o desempenho da técnica é importante apresentar quais são as situações de detecções possíveis da LCV e suas consequências e como foi programado o simulador que foi utilizado neste tópico, e só após, o desempenho estimado da técnica é apresentado.

5.2.1 Situações de Detecção

Existem três situações de detecção que podem ocorrer na LCV, que são: Referência Correta, Referência Não Encontrada e Referência Falsa. A palavra referência está relacionada ao comportamento que é adotado de referência quando encontrado em uma vizinhança em que a maioria dos seus membros se comportaram identicamente, na lista abaixo os três termos são explicados em relação às situações de membros com ou sem defeitos que ocorram em uma vizinhança:

- (1)¹ Referência Correta: Ocorre quando (a) três ou (b) quatro membros da vizinhança não possuem defeitos;
- (2) Referência Não Encontrada: Quando existem (a) dois, (b) três ou (c) quatro membros com defeitos diferentes (seja em tamanho ou posição) na vizinhança;
- (3) Falsa Referência: Quando ocorre (a) três ou (b) quatro membros com defeitos idênticos na vizinhança.

Durante a execução do teste de manufatura utilizando a técnica, podem ocorrer três resultados quanto às células da memória: Célula com defeito sendo detectada e localizada, célula sem defeitos sendo dada como defeituosa e célula com defeito escapando o teste de manufatura. Utilizando os rótulos (a), (b) e (c) presentes no texto, a tabela 4 apresenta o mapeamento entre a situação de detecção em uma vizinhança e os resultados quanto às células da memória.

Após apresentar os dados dessa tabela fica evidente que existem situações em que células vão escapar do teste de manufatura, assim células boas serão desperdiçadas, entretanto a probabilidade de que esses dois fatos realmente ocorram está completamente relacionada ao cenário realista de produção, ou seja, relacionado as variáveis probabilidade de Célula Defeituosa (P_{cdf}), o Número de Locais ($\#L_{def}$) em que é possível ocorrer

¹Os rótulos (1), (2), (3), (a), (b) e (c) são os mesmos utilizados no log gerado pelo simulador, vide código no APÊNDICE A.

Tabela 4 – Mapeamento entre a situação de detecção em uma vizinhança e os resultados quanto às células da memória.

	(1)		(2)			(3)	
	Referência Correta		Referência Não encontrada			Referência Falsa	
	(a)	(b)	(a)	(b)	(c)	(a)	(b)
Célula c/ def. detec. e loc.	1	0	2	3	4	0	0
Célula s/ def. data defeituosa	0	0	2	1	0	1	0
Célula c/ def. escapando o teste	0	0	0	0	0	3	4

defeitos na célula da SRAM e a Quantidade de Tamanhos ($\#T_{def}$) que um defeito pode assumir.

O simulador programado faz a contagem para cada uma das sete situações (colunas) da tabela 4 e utiliza como dados de entrada valores para P_{cdf} , $\#L_{def}$ e $\#T_{def}$. Com essa contagem é possível calcular a quantidade de células detectadas, escapadas e desperdiçadas em uma memória.

5.2.2 Simulador de Monte Carlo

Como a técnica proposta apresenta a característica de deixar que células defeituosas escapem os testes de manufatura, e que esse escape está relacionado a variáveis que envolvem o processo de manufatura e o projeto da memória (P_{cdf} , $\#L_{def}$ e $\#T_{def}$), fica importante que uma abordagem estatística seja utilizada com o objetivo de mensurar a eficácia da técnica proposta, ou seja, medir a quantidade de células que são detectadas e escapadas no teste de manufatura, assim como, compreender a influência das variáveis mencionadas no sucesso da detecção de células de SRAM defeituosas quando utilizado um teste de manufatura cujo conte com a técnica proposta.

Essa abordagem estatística foi executada com o uso de simulador de Monte Carlo de confecção própria. A simulação de Monte Carlo trata-se do sorteio consecutivo e exaustivo de situações de células defeituosas sendo analisadas pela técnica proposta e o resultado da detecção anotado, para cada sorteio. O fluxograma ilustrado da figura 20 apresenta o

procedimento executado pelo simulador confeccionado para avaliar o poder de detecção da técnica proposta.

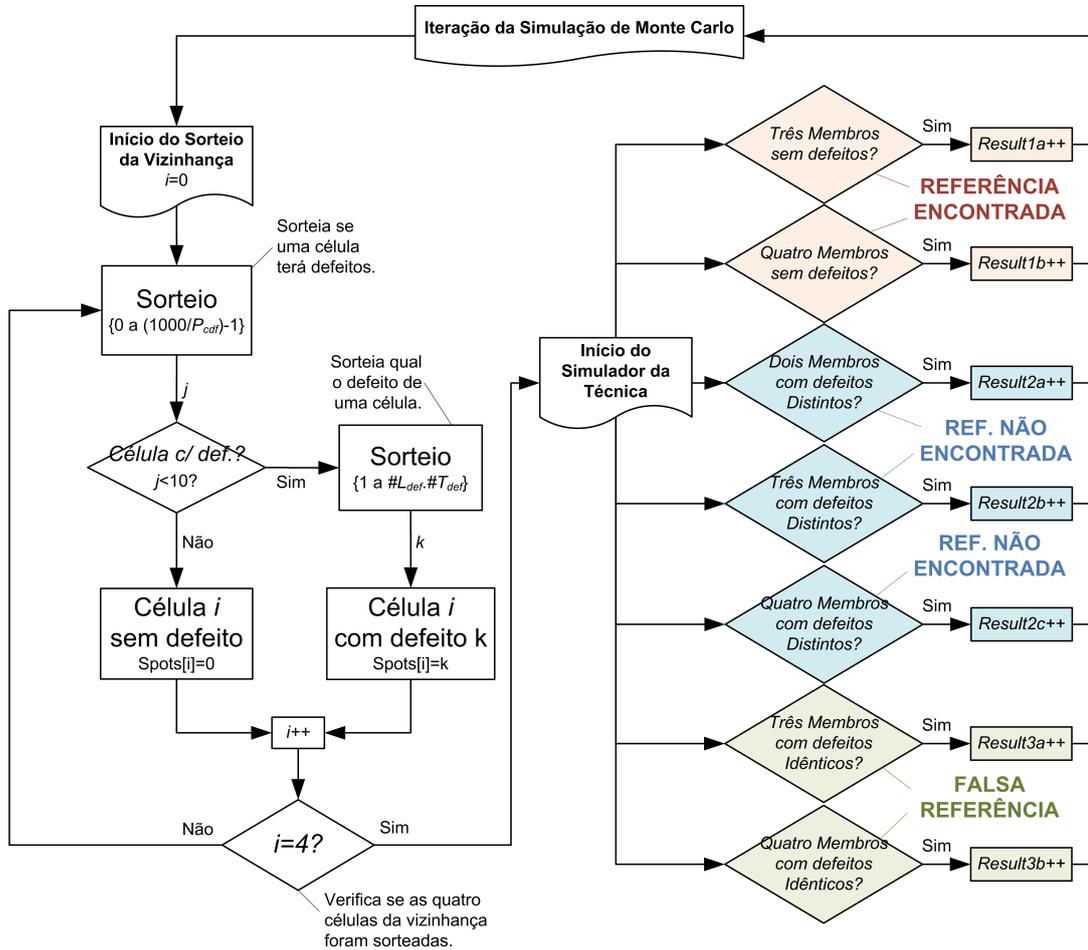


Figura 20 – Fluxograma das operações do algoritmo simulador de Monte Carlo, cujo código encontra-se no APÊNDICE A.

O primeiro passo do algoritmo consiste em sortear a condição de defeitos para cada uma das quatro células da vizinhança. Primeiramente, é sorteado se a célula terá defeito utilizando P_{cdf} , e então, se o sorteio resultar em uma célula com defeito, é sorteado qual a posição e tamanho desse defeito utilizando $\#L_{def} \times \#T_{def}$. Com isso, a vizinhança estará configurada e basta aplicar o algoritmo que simula a técnica proposta para observar como ocorre a detecção, o que define o segundo passo do algoritmo, que irá contar a quantidade de vezes que houve detecção e localização, detecção sem localização e falsa detecção e/ou localização. E assim, através do primeiro e do segundo passo, o algoritmo desempenha as iterações de Monte Carlo gerando os percentuais para cada contagem registrada.

5.2.3 Desempenho Estimado da Técnica

Com o objetivo de utilizar o simulador para avaliar o desempenho da técnica é necessário configurar as três variáveis de entrada (P_{cdf} , $\#L_{def}$ e $\#T_{def}$). A variável P_{cdf} controlará a quantidade de células defeituosas na memória, trata-se de um dado completamente dependente de tecnologia e do design da memória a ser produzida e é difícil de estimar, já a variável $\#L_{def}$ será considerada de valor 6, representando as seis posições DF1-DF6 capazes de sensibilizar todas as falhas possíveis em uma célula 6T da SRAM [13], e a variável restante, $\#T_{def}$, na prática pode ser infinita, tornando ainda mais difícil o ajuste das variáveis. Para contornar o obstáculo da falta de conhecimento do que seria um dimensionamento realista dessas duas variáveis, serão apresentados dados com o objetivo de traçar uma tendência de como varia o desempenho de detecção da técnica devido ao crescimento, ou decréscimo, das variáveis P_{cdf} e $\#T_{def}$, e por fim, conhecer as condições para que a técnica opere com o desempenho desejado.

Tabela 5 – Desempenho de detecção para $P_{cdf} = 1\%$, $\#L_{def} = 6$ e $\#T_{def}$ variável.

	Referência Correta	Referência não encontrada	Referência Falsa
$\#T_{def} = 1$	99,940658%	0,059327%	0,000014%
$\#T_{def} = 2$	99,940905%	0,059094%	0,000001%
$\#T_{def} = 5$	99,940830%	0,059170%	0,000000%
$\#T_{def} = 10$	99,940674%	0,059326%	0,000000%
$\#T_{def} = 15$	99,940806%	0,059194%	0,000000%
$\#T_{def} = 30$	99,940692%	0,059308%	0,000000%

O desempenho ideal de uma técnica de detecção de defeitos é aquele que nenhuma célula defeituosa escapa do teste de manufatura, e o mínimo de células sem defeitos são dadas como defeituosas, ou seja, desperdiçadas. As tabelas 5 e 6 apresentam o percentual da contagem, proveniente do simulador de Monte Carlo. Observando-as conclui-se que o desempenho da técnica aumenta junto com o aumento da variável $\#T_{def}$ e queda da variável P_{cdf} . De maneira correta, pode-se aferir que a técnica terá o desempenho desejado,

Tabela 6 – Desempenho de detecção para P_{cdf} variável, $\#L_{def} = 6$ e $\#T_{def} = 5$.

	Referência Correta	Referência Não encontrada	Referência Falsa
$P_{cdf} = 10\%$	94,769357%	5,230196%	0,000447%
$P_{cdf} = 5\%$	98,598532%	1,401407%	0,000059%
$P_{cdf} = 2\%$	99,766312%	0,233685%	0,000003%
$P_{cdf} = 1\%$	99,940830%	0,059170%	0,000000%

onde nenhuma célula defeituosa escaparia dos testes de manufatura (Falsa Referência nula), para $P_{cdf} < 1$ e $\#T_{def} > 5$. Portanto, sempre que for utilizada em qualquer processo de fabricação que obedeça a essas duas regras, a técnica irá detectar todos as células defeituosas e apresentar um desperdício menor que $0,029\%^2$.

Para ilustrar o desempenho da técnica de detecção em uma memória, adote uma SRAM com $1Gbit$ de tamanho produzida em um processo de fabricação que atende as exigências $P_{cdf} = 1\%$ e $\#T_{def} = 5$. A tabela 7 apresenta a contagem de células com defeitos que escaparam do teste e as células sem defeitos que foram desperdiçadas (dada como defeituosas). Essa tabela foi confeccionada utilizando o simulador de Monte Carlo e os dados da tabela 4.

Tabela 7 – Desempenho estimado da técnica em uma memória de $1Gbit$ considerando $P_{cdf} = 1\%$ e $\#T_{def} = 5$

Total de células defeituosas	Celulas boas desperdiçadas*	Células c/ def. não detectadas
10.000.000	294.890	0

*. Na ocorrência de detecção de defeitos sem localização todas as células da vizinhança foram dadas como defeituosas.

²O percentual de desperdício é em torno do percentual de “Referência Não Encontrada” dividido por dois.

5.3 Custo

Em termos de custos, a técnica foi avaliada considerando o consumo de área e o tempo de execução do teste de manufatura.

Para fazer o levantamento do custo de área, calculou-se a área de todos os transistores utilizados pelos sensores e lógicas comparadoras e dividiu-se pela área de um transistor normalizado, assim, estimando o custo de *overhead* associado ao uso da técnica proposta. Note que estes cálculos não incluem a área consumida pelo roteamento.

O consumo de área por coluna monitorada da técnica proposta foi o equivalente de 132 transistores nMOS e 156 transistores pMOS normalizados³ ou $3,2904(\mu m)^2$, o que equivale à área de 56 células da SRAM projetada para estudo de caso neste trabalho. Para fins comparativos, uma técnica [28] que utiliza BIST e faz recuperação de células da SRAM, substituindo linhas e colunas por sobressalentes, insere em torno de 1 a 5% de *overhead* dependendo do tamanho da memória. Sabendo que o objetivo da técnica proposta é que seja usada junto a essas técnicas de recuperação e teste, os custos de área da técnica proposta devem somar-se aos das técnicas de recuperação da memória. A tabela 8 apresenta o custo estimado de área, devido ao uso isolado da técnica proposta, para diferentes tamanhos de memórias, e mostra que o mesmo encontra-se dentro dos limites admitidos, baseado nos tamanhos recém mencionados de outros tipos de técnicas.

Tabela 8 – Custo de área da técnica proposta.

Tamanho da SRAM	Disposição <i>linha x coluna</i>	Custo $(\mu m)^2$	Overhead (%)
<i>8kbit</i>	1024 x 8	26,3	5,46
<i>64kbit</i>	2048 x 32	105,3	2,73
<i>512kbit</i>	8192 x 64	26,954k	0,68

Já quanto ao tempo de execução do teste de manufatura proposto, a técnica se mostrou vantajosa primeiramente porque não é o algoritmo *March* quem faz a detecção, e sim o circuito de detecção proposto (sensores e LCV), e segundo, porque os acessos não ocorrem

³ $L = 60nm$, $W_P = 250nm$ e $W_N = 120nm$

bit a bit, e sim, linha a linha, minimizando drasticamente o tempo gasto nos testes de manufatura. A tabela 9 mostra um comparativo entre os testes mais comuns encontrados na literatura e o teste proposto nesta dissertação.

Tabela 9 – Tempo de execução do teste de manufatura proposto comparado a testes existentes utilizando como referência a SRAM estudo de caso.

Teste <i>March</i>	Tamanho do Teste	Número de Acessos	Duração Estimada μs
MarchSS [29]	$22n$	180224	360,5
MarchSD [29]	$20n$	163840	327,7
MarchRAW [29]	$26n$	212992	426,0
Proposta	$5n^*$	5120	41,0

*Acessos linha a linha

5.4 Considerações Quanto às Variações de Processo

Existem dois tipos de incidência de variações de processos que podem ser encontradas durante a manufatura de circuitos integrados, as variações *inter-die* e as variações *intra-die*, que como o nome já informa, variações entre um *die* e outro são chamados de *inter-die*, enquanto as variações que ocorrem dentro de um mesmo chip são chamadas de *intra-die*. Ambas as variações são consideradas variáveis gaussianas aleatórias e independentes, dessa maneira nenhum padrão de variação pode ser estabelecido, obrigando uma análise estatística (para *intra-die*) e/ou de cobertura total (para *inter-die*).

Em processos mais antigos (130nm e maiores) as variações *intra-die* não são consideráveis, e, para projetar ICs nessas tecnologias, examina-se o circuito adotando que todos os elementos do chip encontram-se em um mesmo *corner*, como é feito neste trabalho. Já, conforme a integração dos semicondutores aumenta, as variações *intra-die* deixam de ser insignificativas para tornarem-se completamente influentes na integridade dos circuitos integrados, obrigando que os engenheiros passem a considerar que dois transistores vizinhos possam se encontrar em *corners* completamente opostos, o que agregará maior complexidade no projeto dos ICs.

A biblioteca de $65nm$ utilizada neste trabalho fornece apenas informações a respeito das variações *inter-die*, o que, por fim, simplifica as simulações deste trabalho. A discussão do impacto que as variações *intra-die* ocasionariam na metodologia e nos circuitos propostos nesta dissertação são apontados nesta sessão.

5.4.1 Variações na SRAM

A técnica proposta é imune a variações *inter-die* que ocorram dentro da SRAM já que estas variações não criam diferença no comportamento das células quando comparadas entre si, entretanto, se houverem variações *intra-die*, a técnica teria de ser redesenhada em alguns aspectos porque a corrente monitorada das células não seria mais idêntica para células sem defeitos, e sim, passariam a apresentar alguma diferença, já que células vizinhas se encontrariam em *corners* distintos.

5.4.2 Variações no Sensor de Corrente

Os sensores de correntes são circuitos que operam de maneira analógica e são extremamente sensíveis a variações no processo de fabricação, entretanto os sensores deste trabalho não foram testados quanto a variações *intra-die* nem *inter-die*, pois é muito elevada a complexidade de projetar sensores que operam corretamente em qualquer situação de *corner*.

Para que o sensor deste trabalho seja imune a variações de processo, é necessário monitorar a corrente de “*sub-threshold leakage*” de cada sensor e então recalibrá-los com algum sistema de reconfiguração individual, como é recomendado pela literatura [23].

5.4.3 Variações na LCV

A Logica Comparadora de Vizinhança é composta por um circuito puramente digital, e nesses circuitos as variações de processo afetam apenas o tempo de propagação dos sinais. Para testar a LCV frente às variações de processo *inter-die* foi utilizado o simulador

HSPICE onde os sinais de entrada da LCV foram gerados manualmente e a sua saída observada, a tabela 10 contém o resultado.

Tabela 10 – Desempenho da LCV para as variações *inter-die*.

	Saída Correta?	Tempo de Propagação
FF -10 ^o C	Sim	0.16ns
FF 140 ^o C	Sim	0.16ns
SS -10 ^o C	Sim	0.28ns
SS 140 ^o C	Sim	0.28ns
FS -10 ^o C	Sim	0.22ns
FS 140 ^o C	Sim	0.22ns
SF -10 ^o C	Sim	0.22ns
SF 140 ^o C	Sim	0.22ns

Em circuitos digitais as variações *intra-die* não irão afetar o tempo de propagação com maior intensidade do que as *inter-die*, portanto a tabela 10 apresenta as maiores variações que poderiam ser introduzida pelos *corners* nos circuitos da LCV.

6 CONCLUSÕES

Esta dissertação de mestrado, que teve como principal objetivo propor uma técnica de detecção de defeitos do tipo *Resistive-Open* que ocorrem dentro da célula 6T da SRAM, dissertou sobre a SRAM, os defeitos, e como eles são inseridos e detectados nessa memória. Depois apresentou a técnica de detecção proposta e os resultados da análise efetuada a respeito da sua funcionalidade e desempenho.

A memória estática é responsável por grande parte da área de um SoC e como é possuidora de grande volume de interconexões tornou-se um grande responsável pela fabricação de SoC com defeitos, porque os defeitos do tipo *Resistive-Open* ocorrem nessas interconexões e afetam negativamente a operação das células da memória terminando por comprometer a operação do SoC como um todo.

Os defeitos do tipo *Resistive-Open* são modelados por um resistor inserido no lugar de um nodo que idealmente deveria ter sua impedância nula. Na célula foram apresentados seis locais em que os resistores devem ser inseridos para sensibilizar todas as falhas possíveis relacionadas a esse tipo de defeito. As falhas funcionais consequentes desses defeitos são divididas em duas classes, as estáticas e as dinâmicas, a primeira é causada por defeitos fortes, de grande impedância, e a segunda, por defeitos de pequena impedância, e este trabalho também mostrou que a presença de defeito, inclusive de intensidade fraca, afeta o consumo de corrente da célula da SRAM. Durante o desenvolvimento da memória estudo de caso, os defeitos resistivos foram inseridos nesses seis locais recomendados e as falhas funcionais observadas formularam a figura 3, cuja demonstra que o modelamento utilizado é apropriado e confirma a relação entre tamanho do defeito e o tipo de falha

ocasionada alegada anteriormente neste parágrafo.

Também foram apresentados os testes de manufatura mais comuns para detecção desses defeitos na memória, são eles: *March test*, *I_{ddq}test* e a mistura dos dois, fazendo uso de *On-Chip Current Sensors* (OCCS). O primeiro teste de manufatura é dirigido a detecção de falhas, e se mostrou ineficiente em detectar falhas dinâmicas porque seriam necessárias muitas operações por célula para sensibilizá-las. Já o segundo, foi construído para detectar defeitos e se mostrou limitado em detectar a presença de defeitos diante das variações do processo de fabricação, porque as variações de processo ofuscam a presença de defeitos quando monitorada a corrente de fora do chip. E o terceiro, por utilizar limiares, se mostrou incapaz em lidar com o fato, de que os defeitos do tipo *Resistive-Open*, ora aumentam o consumo de corrente, e ora, diminuem, de acordo com a posição, tamanho e local de ocorrência na SRAM.

Com as limitações das técnicas convencionais apontadas, foi proposta uma nova técnica de detecção de defeitos do tipo *Resistive-Open* com o intuito de contornar essas limitações. A técnica proposta mostrou que sensores de corrente embutidos são capazes de anular as variações de processo que ocorrem entre as pastilhas de circuito integrado, e que limiares podem ser evitados com o uso da Lógica Comparadora de Vizinhanças (LCV) que foi apresentada neste trabalho. A LCV opera similarmente a um votador proveniente das técnicas de redundância de *hardware* consagradas. Como resultado, a técnica proposta mostrou-se apta em detectar todos os defeitos do tipo *Resistive-Open* considerados, sejam eles de tamanho fraco ou forte.

Também, o desempenho da técnica foi avaliado, concluindo que ela é capaz de detectar todas as células defeituosas de uma memória de *1Gbit*, que possui 10 milhões de células defeituosas, contanto que o processo de manufatura insira defeitos a uma probabilidade menor, ou igual, a 1%, e que insira 5, ou mais, tamanhos de defeitos. Tudo isso a um custo de área equivalente a área consumida por 56 células da memória estudo de caso, por coluna monitorada, e tempo de execução de *March test* de 5 operações por linha.

6.1 Trabalhos Futuros

Tem-se como objetivo de trabalhos futuros validar a arquitetura proposta considerando todas as variações de processo possíveis. Primeiramente o sensor de corrente teria de ser ajustado a tolerar tanto as variações *inter-die* quanto as variações *intra-die*, depois a Lógica Comparadora de Vizinhanças teria de receber alguma forma de tolerância para que as células da SRAM possam apresentar variações entre si.

Outro objetivo como trabalho futuro é utilizar esta mesma SRAM estudo de caso e a técnica de detecção de defeitos para detectar demais problemas que distorcem o consumo de corrente das células junto com os defeitos do tipo *resistive-open*, seriam esses problemas: demais defeitos resistivos, *Single Event Upset* ou envelhecimento por NBTI.

Ainda assim, como a técnica proposta faz uso da redundância de *hardware* presente nos sistemas, ela poderia ser empregada para detectar defeitos ou falhas em outros elementos dos SoCs, como no processador, por exemplo, que geralmente possui redundância de núcleos ou *pipelines*.

REFERÊNCIAS

- [1] C. Renfrew and B. Swanson, “Ensuring high-quality ICs,” Jan. 2012.
- [2] C. M. Christensen, *The innovator’s dilemma: When new technologies cause great firms to fail*. Boston: Harvard Business School Press, 1997.
- [3] J. Koomey, “Outperforming Moore’s Law,” *Spectrum, IEEE*, vol. 47, p. 68, march 2010.
- [4] SIA, “International Technology Roadmap for Semiconductors (ITRS),” Jan. 2012.
- [5] W. Needham, C. Prunty, and E. H. Yeoh, “High volume microprocessor test escapes, an analysis of defects our tests are missing,” in *Test Conference, 1998. Proceedings., International*, pp. 25 –34, oct 1998.
- [6] J. Li, C.-W. Tseng, and E. McCluskey, “Testing for resistive opens and stuck opens,” in *Test Conference, 2001. Proceedings. International*, pp. 1049 –1058, 2001.
- [7] P. Dubey, A. Garg, and S. Mahajan, “Study of Read Recovery Dynamic Faults in 6T SRAMS and Method to Improve Test Time,” *Journal of Electronic Testing*, vol. 26, pp. 659–666, 2010. 10.1007/s10836-010-5176-5.
- [8] S. Hamdioui, R. Wadsworth, J. Delos Reyes, and A. van de Goor, “Importance of dynamic faults for new SRAM technologies,” in *European Test Workshop, 2003. Proceedings. The Eighth IEEE*, pp. 29 – 34, may 2003.
- [9] L. Dilillo, P. Girard, S. Pravossoudovitch, A. Virazel, S. Borri, and M. Hage-Hassan, “Resistive-open defects in embedded-SRAM core cells: analysis and march test solution,” in *Test Symposium, 2004. 13th Asian*, pp. 266 – 271, nov. 2004.
- [10] SYNOPSYS, “HSPICE,” Jan. 2012.
- [11] K. Roy and S. Prasad, *Low-power CMOS VLSI circuit design*. "A Wiley-Interscience publication.", Wiley, 2000.
- [12] D. Hodges and H. Jackson, *Analysis and design of digital integrated circuits*. McGraw-Hill series in electrical engineering, McGraw-Hill, 1983.
- [13] L. Dilillo, P. Girard, S. Pravossoudovitch, A. Virazel, and M. Bastian, “Resistive-open defect injection in SRAM core-cell: analysis and comparison between 0.13 μm and 90nm technologies,” in *Design Automation Conference, 2005. Proceedings. 42nd*, pp. 857 – 862, june 2005.
- [14] A. van de Goor and Z. Al-Ars, “Functional memory faults: a formal notation and a taxonomy,” in *VLSI Test Symposium, 2000. Proceedings. 18th IEEE*, pp. 281 –289, 2000.

- [15] J.-F. Li, T.-W. Tseng, and C.-S. Hou, "Reliability-Enhancement and Self-Repair Schemes for SRAMs With Static and Dynamic Faults," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 18, pp. 1361–1366, sept. 2010.
- [16] S. Hamdioui, A. van de Goor, and M. Rodgers, "March SS: a test for all static simple RAM faults," in *Memory Technology, Design and Testing, 2002. (MTDT 2002). Proceedings of the 2002 IEEE International Workshop on*, pp. 95–100, 2002.
- [17] G. Harutunyan, V. Vardanian, and Y. Zorian, "Minimal March Tests for Dynamic Faults in Random Access Memories," in *Test Symposium, 2006. ETS '06. Eleventh IEEE European*, pp. 43–48, may 2006.
- [18] S. Hamdioui, Z. Al-Ars, A. J. van de Goor, and M. Rodgers, "Dynamic Faults in Random-Access-Memories: Concept, Fault Models and Tests," *Journal of Electronic Testing*, vol. 19, pp. 195–205, 2003. 10.1023/A:1022802010738.
- [19] M. Fawaz, N. Kobrosli, A. Chehab, and A. Kayssi, "Testing techniques for resistive-open defects in future CMOS technologies," in *Circuits and Systems (APCCAS), 2010 IEEE Asia Pacific Conference on*, pp. 256–259, dec. 2010.
- [20] J.-C. Lo, "Analysis of a BICS-only concurrent error detection method," *Computers, IEEE Transactions on*, vol. 51, pp. 241–253, mar 2002.
- [21] R. Chipana, L. Bolzani, F. Vargas, J. Semião, J. Rodríguez-Andina, I. Teixeira, and P. Teixeira, "Investigating the Use of BICS to detect resistive-open defects in SRAMs," in *On-Line Testing Symposium (IOLTS), 2010 IEEE 16th International*, pp. 200–201, july 2010.
- [22] R. B. Broen, "New Voters for Redundant Systems," *Journal of Dynamic Systems, Measurement, and Control*, vol. 97, no. 1, pp. 41–45, 1975.
- [23] C. Kim, K. Roy, S. Hsu, R. Krishnamurthy, and S. Borkar, "An on-die CMOS leakage current sensor for measuring process variation in sub-90nm generations," in *Integrated Circuit Design and Technology, 2005. ICICDT 2005. 2005 International Conference on*, pp. 221–222, may 2005.
- [24] M. Taherzadeh-Sani and A. Hamoui, "A 1-V Process-Insensitive Current-Scalable Two-Stage Opamp With Enhanced DC Gain and Settling Behavior in 65-nm Digital CMOS," *Solid-State Circuits, IEEE Journal of*, vol. 46, pp. 660–668, march 2011.
- [25] K. Khare, N. Khare, and P. Sethiya, "Analysis of low voltage rail-to-rail CMOS operational amplifier design," in *Electronic Design, 2008. ICED 2008. International Conference on*, pp. 1–4, dec. 2008.
- [26] S. Ohbayashi, M. Yabuuchi, K. Nii, Y. Tsukamoto, S. Imaoka, Y. Oda, M. Igarashi, M. Takeuchi, H. Kawashima, H. Makino, Y. Yamaguchi, K. Tsukamoto, M. Inuishi, K. Ishibashi, and H. Shinohara, "A 65 nm SoC Embedded 6T-SRAM Design for Manufacturing with Read and Write Cell Stabilizing Circuits," in *VLSI Circuits, 2006. Digest of Technical Papers. 2006 Symposium on*, pp. 17–18, 0-0 2006.
- [27] E. Seevinck, F. List, and J. Lohstroh, "Static-noise margin analysis of MOS SRAM cells," *Solid-State Circuits, IEEE Journal of*, vol. 22, pp. 748–754, oct 1987.

- [28] J.-F. Li, T.-W. Tseng, and C.-S. Hou, “Reliability-Enhancement and Self-Repair Schemes for SRAMs With Static and Dynamic Faults,” *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 18, pp. 1361 –1366, sept. 2010.
- [29] S. K. Thakur, R. A. Parekhji, and A. N. Chandorkar, “On-chip Test and Repair of Memories for Static and Dynamic Faults,” in *Test Conference, 2006. ITC '06. IEEE International*, pp. 1 –10, oct. 2006.

APÊNDICE A - CÓDIGO FONTE DO SOFTWARE SIMULADOR DE MONTE CARLO

```
1 #include <stdio.h>
2 #include <termios.h>
3 #include <unistd.h>
4 #include <fcntl.h>
5 #include <stdlib.h>
6 #include <time.h>
7
8 /* Função para mudar a configuração do terminal */
9 void changemode(int dir)
10 {
11     static struct termios oldt, newt;
12
13     if ( dir == 1 )
14     {
15         tcgetattr( STDIN_FILENO, &oldt);
16         newt = oldt;
17         newt.c_lflag &= ~( ICANON | ECHO );
18         tcsetattr( STDIN_FILENO, TCSANOW, &newt);
19     }
20     else
21         tcsetattr( STDIN_FILENO, TCSANOW, &oldt);
22 }
23
24 /* Detector de aperto no teclado */
25 int kbhit (void)
26 {
27     struct timeval tv;
28     fd_set rdfs;
29
30     tv.tv_sec = 0;
31     tv.tv_usec = 0;
32
33     FD_ZERO(&rdfs);
34     FD_SET (STDIN_FILENO, &rdfs);
35
```

```

36     select(STDIN_FILENO+1, &rdfs, NULL, NULL, &tv);
37     return FD_ISSET(STDIN_FILENO, &rdfs);
38 }
39
40 /* Rotina da Simulação */
41 #define _PLUS1 (((Pdef==100)|| (ndef%2==1)) ? 0 : 1)
42
43 int main (int argc, char **argv) {
44
45     if( (argc<3) || strstr(argv[1], "-h") || strstr(argv[1], "--help")) {
46         printf("Usage:\n %s <Pdef> <#Tdef>\n", argv[0]);
47         exit(-1);
48     }
49
50     unsigned int i,j,k;
51
52     /* Variáveis de Configuração da Simulação */
53     unsigned int ndef=6*atoi(argv[2]); // Número de defeitos possíveis de acontecer
54     unsigned int Pdef=atoi(argv[1]); // Probabilidade da célula ter defeito em %
55
56     /* Variáveis internas */
57     unsigned int drawTotal=0, drawZero=0;
58     unsigned int total=0;
59     /* Variáveis de Log */
60     unsigned int result1=0, result1a=0, result1b=0;
61     unsigned int result2=0, result2a=0, result2b=0, result2c=0;
62     unsigned int result3=0, result3a=0, result3b=0;
63     /* Variável que armazena o estado das células sorteadas */
64     unsigned int spots[4];
65
66     changemode(1); // Ativa modo NÃO BLOCANTE no terminal
67     srand ( time(NULL) ); //Inicializa função Srand
68
69     printf("Press 'c' to stop, or any key to show current log\n\n");
70
71     /* Loop da simulação */
72     while(1) {
73
74         /*
75          *
76          * Sorteio dos defeitos de uma Vizinhança
77          *
78          */
79         /* Para cada membro da Vizinhança */
80         for(i=0;i<4;i++) {
81             /* Sorteia se o membro vai ter defeito */
82             j = rand() % (1000/Pdef); //
83             drawTotal++;
84             if (j<10) {
85                 /* Membro com defeito, sorteia qual defeito */
86                 k = (rand()%ndef) + _PLUS1;

```

```

87     else {
88         /* Membro sem defeito */
89         k=0;
90         drawZero++;
91     }
92     spots[i]=k;
93 }
94
95
96 /*
97  *
98  * Simulador da Técnica de Detecção de Defeitos
99  *
100 */
101 /* Teste 1: Membros da Vizinhança produzem referência errônea. */
102 /* Condição: Quatro Membros com defeito idêntico */
103 if((spots[0] == spots[1]) && (spots[0] == spots[2]) &&
104     (spots[0] == spots[3]) && (spots[0] != 0) )
105     result3b++;
106
107 /* Condição: Três Membros com defeito idêntico. */
108 else if((spots[0] == spots[1] && spots[0] == spots[2] && spots[0] != 0) ||
109         (spots[0] == spots[1] && spots[0] == spots[3] && spots[0] != 0) ||
110         (spots[1] == spots[2] && spots[1] == spots[3] && spots[1] != 0) ||
111         (spots[0] == spots[2] && spots[0] == spots[3] && spots[0] != 0)
112         )
113     result3a++;
114
115 /* Teste 2: Membros da Vizinhança produzem uma referência correta */
116 /* Condição: Quatro Membros sem defeitos (=0) na vizinhança */
117 else if(spots[0] == spots[1] && spots[1] == spots[2] &&
118         spots[2] == spots[3] && spots[0] == 0 )
119     result1b++;
120
121 /* Condição: Três Membros sem defeitos (=0) na vizinhança */
122 else if((spots[0] == spots[1] && spots[1] == spots[2] && spots[0] == 0) ||
123         (spots[0] == spots[1] && spots[1] == spots[3] && spots[0] == 0) ||
124         (spots[1] == spots[2] && spots[2] == spots[3] && spots[1] == 0) ||
125         (spots[0] == spots[2] && spots[2] == spots[3] && spots[0] == 0)
126         )
127     result1a++;
128
129 /* Teste 3: Vizinhança não produz referência */
130 /* Subteste 3.a: Nenhum membro sem defeito */
131 else if((spots[0] != 0 && spots[1] != 0 &&
132         spots[2] != 0 && spots[3] != 0) )
133     result2c++;
134
135 /* Subteste 3.b: Apenas um membro sem defeito */
136 else if((spots[0] == 0 && spots[1] != 0 && spots[2] != 0 && spots[3] != 0) ||
137         (spots[0] != 0 && spots[1] == 0 && spots[2] != 0 && spots[3] != 0) ||

```

```

138     (spots[0] != 0 && spots[1] != 0 && spots[2] == 0 && spots[3] != 0) ||
139     (spots[0] != 0 && spots[1] != 0 && spots[2] != 0 && spots[3] == 0) )
140     result2b++;
141
142     /* Subteste 3.c: Dois membros sem defeito */
143     else if((spots[0] == 0 && spots[1] == 0 && spots[2] != 0 && spots[3] != 0) ||
144     (spots[0] == 0 && spots[1] != 0 && spots[2] == 0 && spots[3] != 0) ||
145     (spots[0] == 0 && spots[1] != 0 && spots[2] != 0 && spots[3] == 0) ||
146     (spots[0] != 0 && spots[1] == 0 && spots[2] == 0 && spots[3] != 0) ||
147     (spots[0] != 0 && spots[1] == 0 && spots[2] != 0 && spots[3] == 0) ||
148     (spots[0] != 0 && spots[1] != 0 && spots[2] == 0 && spots[3] == 0) )
149     result2a++;
150
151     total++;
152
153     /* Calcula os totais para cada teste */
154     result1 = result1a + result1b;
155     result2 = result2a + result2b + result2c;
156     result3 = result3a + result3b;
157
158     /* Verificação de overflow da simulação */
159     if(total == 0xFFFFFFFF) { printf("Total OF"); break;}
160     if(drawTotal == 0xFFFFFFFF) { printf("drawTotal OF"); break;}
161
162     /* Termina a simulação e/ou imprime resultados ao teclar do teclado.*/
163     if(kbhit())
164     {
165         printf("##### Simulation #####\n"
166             "PDEF=%d NDEF=%d\n\n",Pdef, ndef);
167
168         printf("Cells: Drow 0: %10d Total: %10d / %f%%\n",
169             drawZero, drawTotal,
170             (double)(100*(double)drawZero/(double)drawTotal));
171
172         printf("Neighb: Total: %d / 0h%X\n\n",total, total);
173
174
175         printf("(1) True Reference: %10d / %f%%\n",
176             result1, (double)(100*(double)result1/(double)total));
177         printf(" (a) 3 good: %10d / %f%%\n",
178             result1a, (double)(100*(double)result1a/(double)total));
179         printf(" (b) 4 good: %10d / %f%%\n\n",
180             result1b, (double)(100*(double)result1b/(double)total));
181
182
183         printf("(2) No Reference: %10d / %f%%\n",
184             result2, (double)(100*(double)result2/(double)total));
185         printf(" (a) 2 defects: %10d / %f%%\n",
186             result2a, (double)(100*(double)result2a/(double)total));
187         printf(" (b) 3 defects: %10d / %f%%\n",
188             result2b, (double)(100*(double)result2b/(double)total));

```

```
189     printf(" (c) 4 defects: %10d / %f%%\n\n",
190           result2c, (double)(100*(double)result2c/(double)total));
191
192
193     printf("(3) False Reference: %10d / %f%%\n",
194           result3, (double)(100*(double)result3/(double)total));
195     printf(" (a) 3 defects: %10d / %f%%\n",
196           result3a, (double)(100*(double)result3a/(double)total));
197     printf(" (b) 4 defects: %10d / %f%%\n\n",
198           result3b, (double)(100*(double)result3b/(double)total));
199
200     printf(" 100%% test: %f%%\n",
201           ((100*(double)(result1a + result1b + result2a + result2b +
202             result2c + result3a + result3b))/(double)total));
203
204
205     if (getchar()=='c') break;
206     else {
207         printf("Still running... Press 'c' to stop, "
208             "or any key to show current log\n\n");
209     }
210 }
211 }
212 changemode(0); // Liga o modo BLOCANTE do terminal
213 exit(0);
214 }
```
