PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL

FACULDADE DE ENGENHARIA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

RAÚL DARÍO CHIPANA QUISPE

TESTE DE SRAMs BASEADO NA INTEGRAÇÃO DE MARCH TESTE E SENSORES DE CORRENTE ON-CHIP

Porto Alegre

2010

TESTE DE SRAMs BASEADO NA INTEGRAÇÃO DE MARCH TESTE E SENSORES DE CORRENTE ON-CHIP

Dissertação apresentada como requisito para obtenção do grau de Mestre pelo Programa de Pós-Graduação da Faculdade em Engenharia Elétrica da Pontifícia Universidade Católica do Rio Grande do Sul.

Orientador: Prof. Dr. Fabian Luis Vargas

Porto Alegre

2010

TESTE DE SRAMs BASEADO NA INTEGRAÇÃO DE MARCH TESTE E SENSORES DE CORRENTE ON-CHIP

RAÚL DARÍO CHIPANA QUISPE

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

Fabian Luís Vargas, Dr.

Orientador

Rubem Dutra Ribeiro Fagundes, Dr.

Coordenador

Programa de Pós-Graduação em Engenharia Elétrica

Banca Examinadora:

Fabian Luís Vargas, Dr.

Presidente – PUCRS

Daniel Ferreira Coutinho, Dr.

PUCRS

Alfredo Arnaud Maceira, Dr.

UCU

Agradecimentos

Dedico meus sinceros agradecimentos:

- ✓ *Ao meu orientador Prof. Fabian Vargas pela oportunidade de trabalho no SiSC.*
- ✓ *Ao Dr. Letícia Bolzani pela colaboração durante o desenvolvimento deste trabalho.*
- ✓ *A os meus pais Dario e Ana pelo apoio e força ao longo da minha vida.*
- ✓ *A minha irmã Helena, Ana e Norma, que sempre me incentivaram e me apoiaram nas minhas escolhas.*
- ✓ *A minha namorada Raquel, pela compreensão, paciência e amor.*
- ✓ *A todos os integrantes do grupo SiSC, que contribuíram para a realização deste trabalho.*
- ✓ *A minha amiga Claudia, pela ajuda e colaboração desde os meus primeiros dias em Porto Alegre*.

Resumo

Atualmente é possível observar que a área dedicada a elementos de memória em sistemas embarcados (*Systems-on-Chip, SoC*) ocupa a maior porção dos circuitos integrados e com o avanço da tecnologia *Very Deep Sub-Micron* (VDSM), é possível integrar milhões de transistores em uma única área de silício. O fato desta elevada integração faz com que surjam novos tipos de defeitos durante a fabricação das memórias. Assim estes novos desafios exigem o desenvolvimento de novas metodologias de teste de SRAMs capazes não só de detectarem defeitos associados a modelos funcionais, e também associados a *resistive-open defects*.

Neste contexto, o desenvolvimento de novos e mais eficientes metodologias de teste de memória é extremamente importante para garantir tanto a qualidade do processo de fabricação como o seu correto funcionamento em campo. Assim, o objetivo deste trabalho é desenvolver uma metodologia de teste que combina um algoritmo simplificado de *March* com sensores *on-chip* que monitoram o consumo de corrente estática da memória.

A avaliação da viabilidade e eficiência da metodologia de teste proposta neste trabalho foi feita baseada em simulações elétricas de modelos de falhas aplicadas a um bloco de SRAM. Estas simulações foram desenvolvidas com HSPICE e CosmosScope em ambiente Synopsys.

A partir dos resultados obtidos, foi possível verificar a capacidade de detecção das falhas permanentes modeladas. A vantagem desta metodologia reside no desenvolvimento de um algoritmo híbrido de teste de memórias baseado fundamentalmente nos monitoramentos da tensão (através de elementos *March*) e da corrente estática (através de sensores de corrente *on-chip*). O resultado desta combinação é um novo algoritmo de teste de SRAMs menos complexo, isto é, capaz de detectar falhas em menor tempo de teste quando comparado com algoritmos existentes, ao passo que garante a mesma cobertura de falhas.

Abstract

Currently it's possible to observe that the area devoted to memory elements in embedded systems (Systems-on-Chip, SoC) occupies the largest portion of the integrated circuits and due to the advance in Very Deep Sub-Micron (VDSM) technology is possible to integrate millions of transistors on a single area. The high integration causes new types of defects not only during the fabrication, but also during the lifetime of memories. These new challenges require the development of new methodologies to test SRAMs able not only to detect faults associated with functional models in memories, but also associated with resistive-open defects.

In this context, the development of more efficient and effective methodologies is extremely important to ensure the quality of the manufacturing process and the field operation. Thus, the objective of this work is to develop an innovative test technique based simultaneously on the coupling of existing *March tests* with built-in current sensors to monitor static current dissipation.

The validation of the test methodology proposed in this work was based on electrical simulations of a SRAM, where resistors were placed into cells to induce abnormal current consumption. Simulations were performed in HSPICE and COSMOS under the Synopsys framework.

From the obtained results, we verify the detection capability of the proposed test strategy with respect to permanent faults generated in the SRAM. Clearly, the advantage of the proposed methodology was the reduced test complexity, i.e., the reduced test application time required to detect the target faults in comparison with existing algorithms, while maintaining the same fault coverage.

Lista de Acrônimos

ASIC - Application Specific Integrated Circuit

BL – Bit line

BLB – *Bit Line Bar* – complemento de BL

CI – Circuito Integrado

BICS – Built-in current sensor

CR – Cell Ratio

DRDF – Deceptive Read Destructive Fault

MOSFET – Metal-Oxide Semiconductor Field Effect Transistor

MTTF – Mean Time to Failure

MTBF – Mean Time Between Failure

MTTR – Mean Time to Repair

RDF – Read Destructive Fault

SB – Lado inverso da célula SRAM

SA – Sense Amplifier

SRAM – Static Random Access Memory

SIA – Semiconductor Industry Association

SoC – System-on-Chip

SEU – Single Event Upset

S – Lado não inverso da célula SRAM

TF – Transition Fault

VLSI – Very Large Scale Integration

VDSM – Very Deep Sub Micron

WL – Word Line

WE – Write Enable

Lista de Tabelas

Lista de Figuras

Sumário

1 Introdução

Elementos de memória são um dos componentes mais importantes de um sistema digital, visto que é comum encontrar em um simples circuito integrado (CI) dezenas de memórias de diferentes tipos e tamanhos. Também as memórias podem ser recursivamente embarcadas em núcleos [7]. Nos últimos anos, a área de silício dedicada à memória está constantemente crescendo quando consideramos *Systems-on-Chip* (SoCs). Os prognósticos da *Semiconductor Industry Association* (*SIA Roadmap*) para os próximos dez anos indicam que a densidade de memória em um *System-on-Chip* (SoC) será de cerca 94% [2]. Neste contexto, o desenvolvimento de novas metodologias de teste mais eficientes é extremamente importante e necessário.

Atualmente os testes de memórias correspondem a aproximadamente metade do custo de fabricação do um circuito integrado. Estes testes são executados com o intuito de detectar as falhas que tem maior probabilidade de ocorrer [5].

Desde meados da década de 80 vários estudos foram realizados com o intuito de identificarem e classificarem diferentes tipos de falhas que potencialmente podem afetar memórias SRAMs. Inicialmente, falhas estáticas e dinâmicas [8], falhas simples ou dependentes, falhas permanentes, transientes ou intermitentes.

Mas com o avanço de novas tecnologias do tipo *Very Deep Sub-Micron* (VDSM), os circuitos integrados podem ter milhões de transistores em uma pequena área de silício, por exemplo, o processador Intel Pentium 4, tem 55 milhões de transistores em 145mm^2. E com esta integração também surgem sérios problemas associados a *resistive-open defects* devido à presença de várias camadas de semicondutores e ao incremento exponencial do número de contatos entre estas camadas em uma mesma

pastilha de silício. Um *resistive-open defects* define-se como um defeito resistivo em trilhas de um circuito, separando dois nós que deveriam estar potencialmente conectados [23].

Neste contexto, os modelos de falhas tradicionalmente utilizados para teste de SRAMs são insuficientes para detectar algum tipo de defeito que possa ter ocorrido em CIs produzidos em tecnologias VDSM [23]. Neste sentido, é essencial e necessário o desenvolvimento de uma nova metodologia de teste capaz de detectar não somente falhas funcionais como *stuck-at* e *state coupling*, mas também falhas associadas a *resistive-open defects* como *transition, read destructive, deceptive read destructive* e *incorrect read fault.*

1.1 Motivação

As metodologias de teste mais conhecidas são baseadas no algoritmo *March Test* [5]. *March Test* é um método baseado em elementos que operam na memória, cada elemento é formado por operações básicas de escrita e leitura em cada célula. Na atualidade, existem vários algoritmos baseados em *March Test* [7][2][34] para verificar a integridade de memórias. A diferença entre cada um deles é a seqüência de execução e a complexidade dos elementos. A complexidade (n) é o número de operações básicas de escrita e leitura na memória. Neste contexto, a maioria dos algoritmos existentes apresenta complexidade maior, por exemplo, que *March G* [35] com 23n. Maior complexidade envolve mais tempo na execução do teste, a qual pode ser considerável dependendo da capacidade da memória. É neste contexto que se apresenta a motivação deste trabalho: desenvolver uma nova metodologia que reduza o tempo de teste de memórias SRAM através da redução da complexidade do algoritmo, ou seja, através da redução do número de acessos à memória, garantindo ao mesmo tempo, no mínimo, a mesma cobertura de falhas dos algoritmos existentes.

Dito isso, este trabalho, apresenta uma nova metodologia de teste de SRAMs baseada em algoritmo *March Test* e monitoramento de corrente estática do CI. O monitoramento de corrente é realizado utilizando-se *Built-In Current Sensor* (BICS)

[4][26][27]. Como resultado desta nova metodologia, o algoritmo baseado em *March Test* é reduzido para a complexidade de ordem 5n. Como conclusão a vantagem desta nova metodologia é que esta apresenta uma significante redução de 50% de tempo requerido para o teste de SRAMs comparado com o algoritmo modificado *March* C- apresentado em [23]. Com respeito ao tempo de acesso, a inserção do BICS degrada a operação de leitura em um 4%, enquanto que a operação de escritura não é afeitada com pela inclusão do BICS na memória.

1.2 Objetivos

O principal objetivo deste trabalho é propor uma nova metodologia de teste baseado em algoritmo March Test e monitoramento de corrente para direção de falhas permanentes em SRAMs. A metodologia de teste proposto busca garantir a detecção das falhas permanentes associadas a falhas funcionais e a detecção de falhas permanentes a *resistive-open defects* gerados no processo de fabricação de SRAMs.

Assim foram realizadas simulações elétricas utilizando Hspice da Synopsys com o intuito de demonstrar a eficiência e viabilidade da metodologia de teste proposta.

Para atingir a este objetivo, os seguintes pontos foram desenvolvidos ao longo deste trabalho:

1) Descrição em HSPICE: modelar uma célula SRAM de 6T em tecnologia de 350nm.
2) Inserção do *Built-In Current Sensor* (BICS) nas colunas de células da SRAM.
3) Modelagem das falhas funcionais e falhas associadas à *resistive-open defects* em células de 6T.
4) Verificação do comportamento da célula na presença das falhas-alvos e da capacidade de detecção da metodologia proposta
5) Avaliação da degradação do tempo de execução das operações de leitura e escritura da memória, em função da agregação do BICS.

1.3 Apresentação dos capítulos

A continuação deste trabalho se dará na seguinte forma:

Capítulo 1: neste capítulo, introduzir-se-á o tema que será desenvolvido juntamente com a motivação e os objetivos principais deste trabalho.

Capítulo 2: o capítulo dois contém conceitos clássicos da área de teste e tolerância à falhas.

Capítulo 3: neste capítulo será apresentada a estrutura básica da memória SRAM e o funcionamento de cada um dos blocos funcionais durante a execução das operações de escrita e leitura.

Capítulo 4: no capítulo quatro serão descritos os modelos de falhas adotados durante as simulações.

Capítulo 5: este capítulo contém as metodologias de teste de SRAMs baseados em *software* e no monitoramento de corrente proposto na literatura.

Capítulo 6: o capítulo seis consiste na metodologia de teste de SRAMs proposta nesta dissertação de mestrado.

Capítulo 7: neste capítulo serão descritos os resultados experimentais obtidos a partir das simulações elétricas realizadas como intuito de validar e avaliar a eficiência da metodologia de teste proposta.

Capítulo 8: neste capítulo serão apresentadas as conclusões.

2 Fundamentos teóricos

Nesta seção será apresentada uma série de conceitos relacionados à teoria de teste e de tolerância à falhas, tais como a definição de falha, erro e defeito bem como a classificação das falhas no que diz respeito a sua origem e ao tempo de ocorrência.

2.1 Falha, Erro ou Defeito

Um defeito (*failure*) é definido como um desvio da especificação. Define-se que um sistema está em estado errôneo, ou em erro, se o processamento posterior a partir deste estado pode levá-lo a um defeito. Finalmente define-se falha ou falta (*fault*) como a causa física ou algorítmica do erro.

Falhas são inevitáveis, pois componentes físicos envelhecem e/ou sofrem com interferências externas, sejam ambientais ou humanas, projetos de *software* e *hardware* são vítimas de sua alta complexidade e também da fragilidade humana em trabalhar com grande volume de detalhes ou ainda com a deficiência de especificações.

A Figura 2.1 apresenta uma simplificação, sugerida por Dhiraj K. Pradhan [12], e também adotada nesta dissertação para os conceitos de falha, erro e defeito. Nela falhas estão associadas ao universo físico, erros ao universo da informação e defeitos ao universo do usuário.

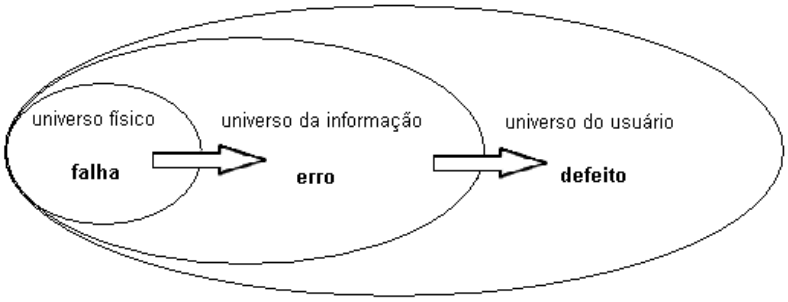

Figura 2.1 Relação entre falha, erro e defeito

Um exemplo para este modelo de três universos seria um *chip* de memória, que apresenta uma falha do tipo *stuck-at 0* em um de seus *bits* (falha no universo físico). Esta falha pode provocar uma interpretação errada da informação armazenada em uma estrutura de dados (erro no universo da informação). Como resultado deste erro, por exemplo, o sistema pode negar autorização de embarque para todos os passageiros de um vôo (defeito no universo do usuário). É interessante observar que uma falha não necessariamente leva a um erro (pois a porção da memória sob falha pode nunca ser usada) e um erro não necessariamente conduz a um defeito (no exemplo, a informação de vôo lotado poderia eventualmente ser obtida a partir de outros dados redundantes da estrutura).

2.1.1 Latência

Define-se latência de falha como o período de tempo desde a ocorrência da falha até a manifestação do erro provocado por esta falha. Seguindo esta linha de raciocínio, define-se como latência de erro o período de tempo desde a ocorrência do erro até a manifestação do defeito.

Baseando-se no modelo de três universos, o tempo total medido desde a ocorrência da falha até o aparecimento do defeito é a soma das latências de falha e de erro. A Figura 2.2 apresenta o conceito de latência de falha e latência de erro.

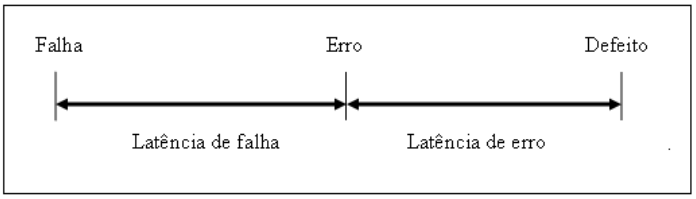

Figura 2.2 Conceito de latência de falha e latência de erro

2.1.2 Classificação de Falhas

Existem diversas classificações para falhas na literatura [12] [13] [14], entretanto estas são geralmente classificadas em falhas físicas (aquelas de que padecem os componentes) e falhas humanas (que compreendem falhas de projeto e falhas de interação). Assim, uma falha pode ocorrer tanto no âmbito de *hardware* quanto de *software*, sendo esta a causa de um erro. Componentes envelhecidos e interferências externas são exemplos de fatores que podem gerar uma falhas nos circuitos integrados e sistemas em geral.

Grande parte das causas relacionadas à ocorrência de uma falha são atribuídas a problemas de especificação, problemas de implementação, componentes defeituosos, imperfeições de manufatura, fadiga dos componentes físicos, além de distúrbios externos como radiação, interferência eletromagnética, variações ambientais (temperatura, pressão, umidade) e também problemas de operação.

Para definir uma falha, além do agente causador consideram-se também os seguintes itens:

Natureza: falha de *hardware*, falha de *software*, de projeto, de operação.

Duração ou persistência: permanente ou temporária (transitória).

Extensão: global ou local a um determinado módulo do circuito.

Valor: determinado ou indeterminado no tempo.

Assim, no que diz respeito a duração e persistência das falha, elas podem ser classificadas conforme as categorias descritas abaixo:

a) **Falhas Permanentes**: ocorrem no meio físico e são provocadas através de falhas no processo de fabricação e/ou pelo envelhecimento dos componentes do sistema; curtos circuitos e nós abertos e *stuck-at* são exemplos de falhas permanentes.

b) **Falhas Transientes**: ocorrem durante a vida útil dos componentes e são provocadas por adversidades e/ou fenômenos ambientais aleatórios onde o sistema está implementado; variações de tensão de alimentação e interferências eletromagnéticas são exemplos de falhas transientes.

c) **Falhas Intermitentes**: é caracterizada pela ocorrência temporária e cíclica do erro a partir de variações das condições externas e/ou ambientais do sistema; vibrações e variações da temperatura são exemplos de falhas intermitentes.

Existe uma crescente ocorrência de falhas provocadas por interação humana maliciosa, ou seja, por ações que visam propositalmente provocar danos aos sistemas. Essas falhas não são tratadas por técnicas de tolerância a falhas, mas sim por técnicas de segurança computacional (*security*). Entretanto deve-se considerar que um sistema tolerante a falhas deve também ser seguro a intrusões e ações maliciosas.

Falhas de *software* e também de projeto são consideradas atualmente o mais grave problema em computação crítica. Sistemas críticos são, tradicionalmente, construídos de forma a suportar e tolerar falhas físicas. Tendo em vista isto, é compreensível que falhas não tratadas e não previstas no projeto sejam as que mais danos causem aos sistemas, pois possuem um grande potencial de comprometer a sua confiabilidade e disponibilidade.

No capítulo 4 apresentaremos a classificação das falhas relacionadas a SRAMs e os modelos adotados durante as simulações elétricas realizadas para a validação e avaliação da metodologia de teste proposta nesta dissertação de mestrado.

2.2 Medidas relacionadas ao tempo médio de funcionamento

Dependabilidade é uma tradução literal do termo inglês *dependability*, que indica a qualidade e a confiança depositada no serviço fornecido por um dado sistema. Confiabilidade e disponibilidade são dois dos principais atributos da dependabilidade

As medidas para avaliação de dependabilidade mais usadas na prática são: taxa de defeitos, MTTF (*mean time to failure*), MTTR (*mean time to repair*), MTBF (*mean time between failure*). Estas medidas estão por sua vez relacionadas a outro parâmetro importante chamado confiabilidade [17]. A Tabela 2.1 apresenta uma definição informal dessas medidas.

Medidas	Significado
Taxa de defeito	Número esperado de defeitos em um dado periodo de tempo: é assumido um valor constante durante o tempo de vida útil do componente.
MTTF (mean time to failure)	Tempo esperado até a primeira ocurrência de defeito
MTTR (mean time to repair)	Tempo médio para reparo do sistema
MTBF (mean time between failure)	Tempo médio entre os defeitos do sistema

Tabela 2.1 Medidas de dependabilidade [12]

Estas medidas são determinadas estatisticamente, observando o comportamento dos componentes e dispositivos. Os fabricantes deveriam fornecer medidas de dependabilidade para os seus produtos, tanto para os componentes eletrônicos, como para os sistemas de computação mais complexos.

A taxa de defeitos de um componente e/ou dispositivo é mensurada em defeitos por unidade de tempo e é diretamente proporcional ao tempo de vida do componente e/ou dispositivo. Uma representação usual para a taxa de defeitos de componentes de

hardware é dada pela curva da banheira. A Figura 2.6 apresenta esta curva onde podemos distinguir três fases:

Mortalidade infantil: componentes fracos e mal fabricados;

Vida útil: taxa de defeitos constantes;

Envelhecimento: taxa de defeitos crescentes.

Os componentes de hardware só apresentam taxa de defeitos constante durante um período de tempo chamado de vida útil, que segue uma fase com taxa de defeitos decrescente chamada de mortalidade infantil. Para acelerar a fase de mortalidade infantil, os fabricantes recorrem a técnicas de *burn-in*, onde é efetuada a remoção de componentes fracos sendo estes substituídos por componentes que já sobreviveram à fase de mortalidade infantil através do processo de aceleração de operação.

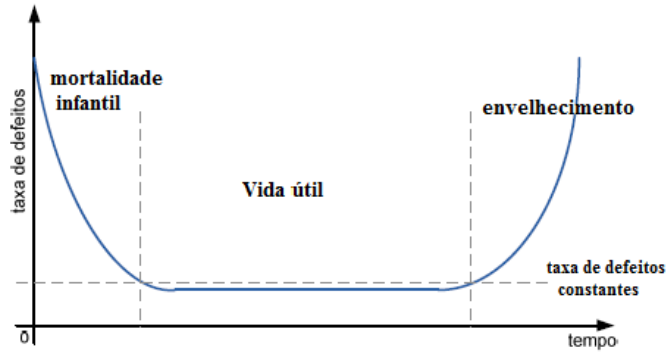

Figura 2.3 Curva da banheira

Para componentes de software é questionável se a curva da banheira pode ser aplicada. Entretanto, pode ser observado que os componentes de software também apresentam uma fase de mortalidade infantil ou taxa de erros acentuados no início da sua fase de ensaios, que decresce rapidamente até a sua entrada em operação. A partir desse momento, o software apresenta um taxa de erros constante até que, eventualmente, precise sofrer alguma correção ou sua plataforma de hardware torne-se

obsoleta. Nesse momento, a taxa de erros volta a crescer. Quando referido a software, foi mencionado taxa de erros (*bugs*) ao contrário de falhas, já que erro é o termo usualmente empregado quando se trata de programas.

2.3 Conceitos associados à tolerância a falhas

Tolerância a falhas é a habilidade de um circuito ou sistema continuar a execução correta das suas tarefas, mesmo diante da ocorrência de falhas em seu *hardware* ou em *software* evitando assim prejuízos físicos e materiais [13].

Além disso**,** Confiabilidade é a capacidade de atender as especificações do projeto dentro de condições definidas durante certo período de funcionamento e estar operacional no início desse período. Finalmente, disponibilidade é a probabilidade de o sistema estar operacional quando a utilização deste for necessária.

3 Introdução à SRAMs

Neste capítulo será apresentada a estrutura interna de uma SRAM bem como as diferentes arquiteturas possíveis para uma célula que compõe uma SRAM. Além disto, esta seção apresenta a descrição funcional das operações de escrita e leitura em uma célula 6T bem como os circuitos relacionados a execução das mesmas.

3.1 Introdução

Nas últimas décadas temos sido testemunhas do rápido desenvolvimento da tecnologia CMOS, paralelamente, a capacidade de armazenamento seguiu o mesmo desenvolvimento. Começando com 1Kb de capacidade de memória DRAM, desenvolvida pela Intel na década de 70, até hoje em dia onde há uma capacidade maior que 1Gb do mesmo tipo [10].

A chegada da memória virtual em computadores pessoais contribuiu com a estrutura hierárquica de vários tipos de memória que vão desde capacidade pequena, rápida mas de custo elevado, como a memória cache. Até memórias de alta capacidade, lenta e de baixo custo como o *hard disk*. A figura 3.1 mostra a hierarquia de memórias de um computador pessoal em forma de pirâmide. A hierarquia tem como prioridade a velocidade e o custo por bit. O aumento da diferença no tempo de acesso entre o Processador e a memória DRAM torna necessário a introdução de vários níveis de memória cache nos processadores de última geração. Nos computadores pessoais tais níveis são representados como memórias cache SRAM de nível L1 e L2. Por ter um maior V_{th} e um menor corrente *leakage*, o uso de memórias DRAM se limita sistemas em operação de mediana velocidade, mas de alta capacidade.

Por outro lado, a memória SRAM teve êxito acelerando o rendimento nos microprocessadores e outros dispositivos que usam processos regularmente rápidos e não requerem tarefas adicionais.

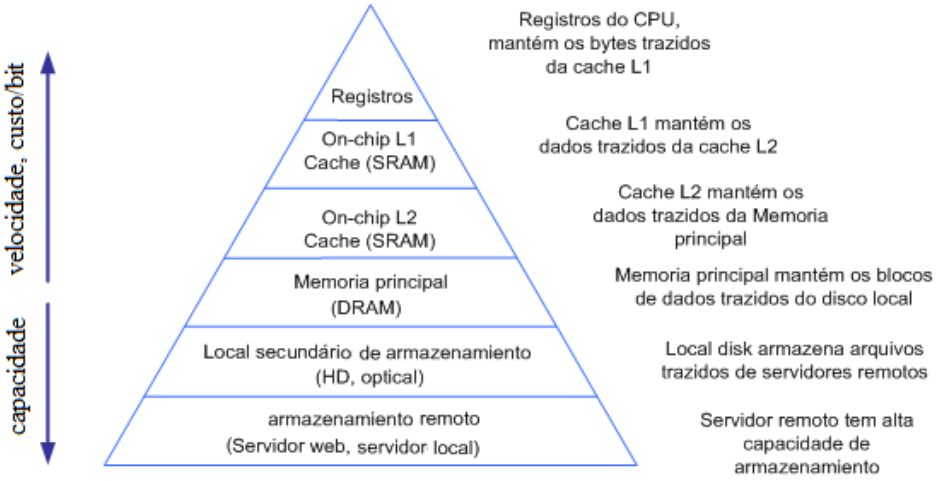

Figura 3.1 Hierarquia da memória de um computador pessoal [10]

Uma SRAM típica utilizará seis transistores MOSFET para armazenar cada *bit*. Adicionalmente, podemos encontrar outros tipos de SRAM, que utilizam oito, dez, ou mais transistores por bit [18]. Isto é utilizado para desenvolver mais de uma porta de leitura o escrita em determinados tipos de memória de vídeo e na memória cache (8T) dos processadores *Nehalem* de Intel.

3.1.1 Lei de Moore

Desde os primeiros dias do semicondutor tem-se desejado minimizar as dimensões do transistor, desenvolver sua capacidade e conseguir a redução do peso de todo o sistema. Todos estes objetivos podem ser alcançados integrando mais componentes em uma mesma área. Um dos motivos na integridade de alta escala dos

semicondutores é a redução no custo do sistema e o incremento do rendimento. Gordon Moore em 1965 mostrou que um dos maiores benefícios de integração e escalabilidade tecnológica e a redução do custo em função de um sistema. O custo da fabricação está decrescendo em cada nova geração de tecnológica, enquanto a integração de componentes em uma mesma área está crescendo.

Em 19 de abril de 1965, *Electronics Magazine* publicou um documento elaborado por Gordon Moore no qual ele antecipava que a complexidade dos circuitos integrados se duplicaria cada ano com uma redução de custo comensurável. Ver figura 3.2. A predição da Lei de Moore tornou possível a proliferação da tecnologia em todo o mundo, e hoje se converteu no motor da rápida mudança tecnológica. Moore atualizou sua predição em 1975 para sinalar que o número de transistores em um chip se duplica a cada dois anos e isto segue se cumprindo hoje [1].

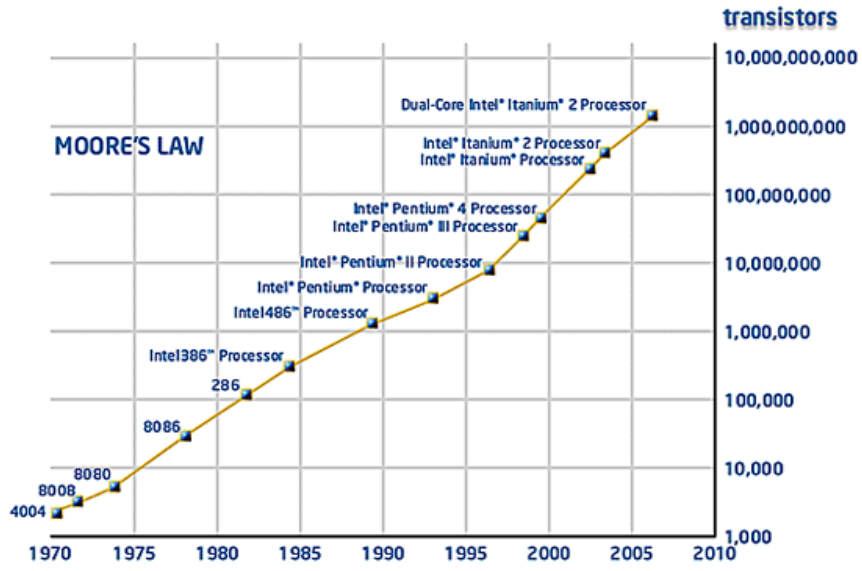

Figura 3.2 Evolução da Lei de Moore em processadores Intel. [1].

Com o avanço das novas tecnologias, o aumento da área de memória chega a medidas críticas dos transistores, o efeito da alta densidade pode causar defeitos de curto circuito às pontes entre transistores e danificar as células. A probabilidade de ocorrer um defeito em um bloco de memória é maior enquanto aumenta o número de

células em uma mesma área. Como conseqüência, as metodologias de teste das memórias estão sendo mais complexas com cada nova tecnologia.

Enquanto o custo de fabricação pode estar decrescendo devido a alta integração na mesma área, o custo dos testes aumenta. O custo dos testes pode alcançar a metade do custo do produto e é diretamente proporcional ao tempo de teste [10].

3.2 Estrutura de blocos de memória SRAM

Uma importante área dos sistemas modernos SoCs é ocupada pela memória SRAM. Por exemplo, a memória cache do microprocessador *Itanium* de Intel (2006) baseada em SRAM ocupa mais de 90% de 1.72 bilhões de transistores. Do mesmo modo está ocorrendo um aumento de SRAM em *Application-Specific Integrated Circuit* (ASIC).

A arquitetura da memória SRAM é importante por diversas razões: o desenho da célula assegura a estabilidade e robustez durante a operação de escrita e leitura.

A estrutura básica de uma memória SRAM está ilustrada na figura 3.3. O bloco de memória SRAM consiste em um bloco de NxM, onde N é o número de filas e M é o número de bits. Se o núcleo é organizado em bloco de páginas, se agrega uma variável Z para selecionar cada uma das páginas. Por exemplo, a figura 3.3 mostra uma matriz SRAM de quatro páginas.

A memória pode ser orientada por *bit* ou por *word*. Quando a memória está orientada por *bit* cada endereço de acesso está associado a somente um *bit.* Por outro lado, se a memória é orientada por *word* cada endereço é associado a uma w*ord.* Cada *word* é de *n bit* onde *n* pode ser 8, 16, 32, 64. Finalmente, cada célula é organizada em colunas as quais compartilham o mesmo *sense amplifier.* A maioria das SRAMs atuais inclui seu próprio gerador de *clock.*

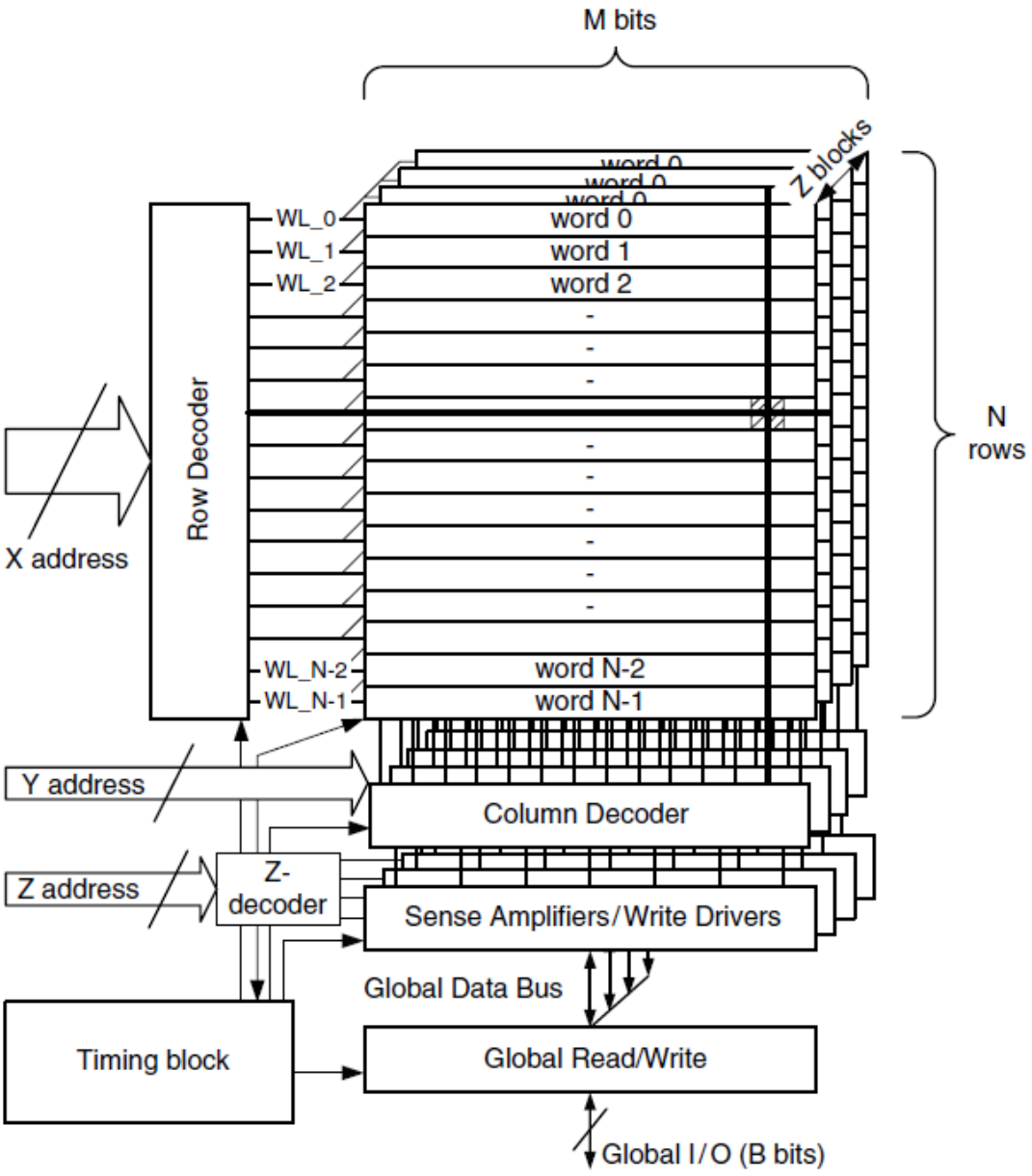

Figura 3.3 Estrutura de blocos de uma memória SRAM [10]

3.3 Célula de uma SRAM

Um dos componentes mais importantes da memória é núcleo SRAM onde se armazena a informação binária. Tipicamente o núcleo está composto de dois inversores cruzados formando um *latch* com transistores de acesso. Os transistores de acesso permitem ler, escrever e isolar a célula para que o dado armazenado não seja corrompido.

Três tipos de células SRAM são mais conhecidos até o momento:

- Célula 4T SRAM com carga resistiva
- Célula 6T SRAM
- Célula 4T SRAM sem carga

Cada desenho tem características próprias de robustez, área, velocidade, consumo de corrente e rendimento.

Uma célula de menor tamanho permite aumentar o número de *bit* em uma mesma área, desta maneira também reduz o custo por *bit*. Reduzindo a área indiretamente também melhora a velocidade e o consumo de energia devido à redução da capacitância em distintos nodos. Com células de menor dimensão se obtêm menor capacitância nas linhas de *bit line* (BL) e *word line* (WL) as quais ajudam a melhorar a velocidade de acesso.

A dimensão do transistor não pode ser reduzida indefinidamente sem comprometer outros parâmetros. Por exemplo, uma célula extremamente pequena pode ter a sua estabilidade comprometida.

3.3.1 Células 4T SRAM com carga resistiva

A célula de quatro transistores com resistência de poli-silício (figura 3.4) é uma tecnologia pré-CMOS. Utilizando somente dois transistores no núcleo da célula, temos menor ganho na região de transmissão. A principal vantagem da célula de quatro transistores com carga de Poli-silicio (*polysilicon resistor load*) é a redução de 30% de área em comparação à célula de seis transistores. Devido à mobilidade dos elétrons (*mobilidade efetiva*) ($\mu_n/\mu_p = 1.5 - 3$). Normalmente todos os transistores da célula de 4T são do tipo NMOS. Os resistores servem para compensar a polarização dos transistores *driver*s. Os valores dos resistores devem ser o mais alto possível para conseguir um *noise margin* baixo (NM_L) para reduzir o consumo estático. Por outro lado, uma resistência demasiadamente alta pode aumentar o *delay* e também aumenta a área [21].

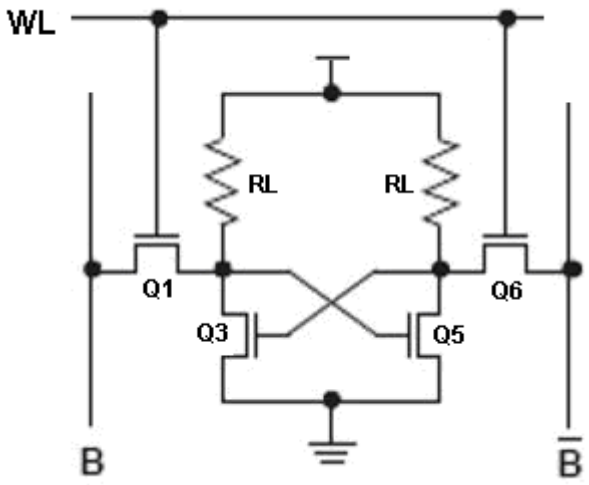

Figura 3.4 Célula SRAM 4T [22]

O limite superior da resistência de carga é fixado pela corrente de *pull-up.* A corrente *pull-up* deve ser no mínimo o dobro da corrente de *pull-down* [10]. O limite inferior do resistor é fixo pela imunidade ao ruído e pelo consumo de energia requerido.

O resistor de poli-silício usado na célula não pode ser tão escapável como a célula a base de transistores. Além disso, é necessário um processo extra na fabricação para obter a alta resistividade do poli-silício, o qual não faz parte de um processo padrão de

tecnologia CMOS. Por outro lado, possui uma baixa tolerância a falhas *soft error* a qual se une a um *static noise margin* (SNM) inadequado. Devido a isso não é possível utilizá-lo em sistemas embarcados (SoCs) os quais são integramente implementados com tecnologia CMOS.

3.3.2 Células 6T SRAM

A célula SRAM de seis transistores (6T) está implementada como um *latch* RS para armazenar o valor de bit. Os quatro transistores internos, Q2, Q3, Q4 e Q5 ilustrados na figura 3.5 formam dois inversores opostos. Q1 e Q6 são os transistores de acesso, que habilitam o núcleo para as operacionais de escrita e leitura. A célula 6T SRAM é a mais popular devido a sua robustez, baixo consumo de energia e alimentação de baixa voltagem. Nas simulações elétricas que serão apresentadas nos próximos capítulos, este tipo de célula será utilizado.

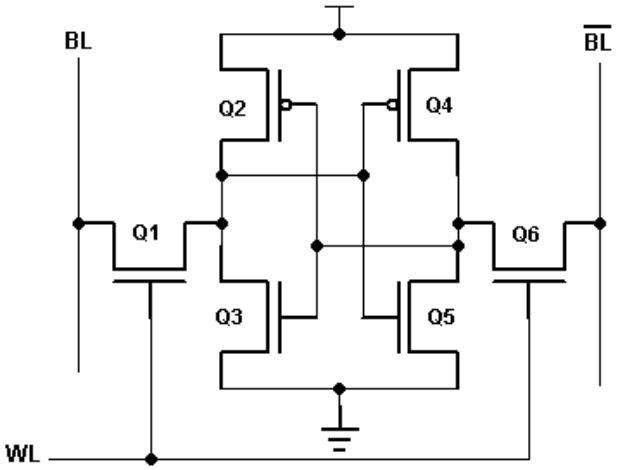

Figura 3.5 Célula 6T SRAM

3.3.3 Célula 4T SRAM sem carga

Uma célula de quatro transistores sem carga utiliza dimensões mínimas em todos os seus transistores. Para a retenção dos dados não é necessário atualizar o valor do *bit.* O dado se mantém com a corrente de *leakage* dos transistores PMOS Q3 e Q4 ilustrados na figura 3.6. Para manter o dado a corrente *leakage* dos transistores PMOS deve ser muito maior que a corrente de *leakage* dos transistores NMOS. Esta condição é normalmente conhecida por utilizar *dual-V_{TH}* com $V_{TH\text{-}p} < V_{TH\text{-}n}$. Em mais detalhes, para que a célula mantenha o dado estável, a corrente de *leakage* do transistor PMOS deve ser cerca de 100 vezes maior do que corrente dos transistores *drivers* NMOS. Todos os transistores em 4T SRAM podem possuir as dimensões mínimas devido a proporção associada a mobilidade dos transistores NMOS e PMOS (μ_n/μ_p), que por su vez está entre dois e três. Os transistores NMOS *drivers* (Q1 e Q2) devem ser de 1,5 a 2,5 maiores que maiores que os transistores de acesso Q3 e Q4 (figura 3.6).

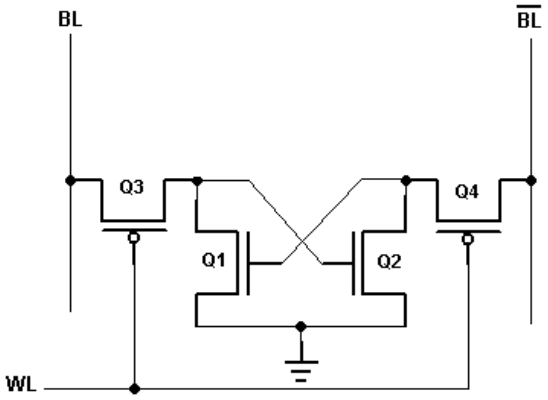

Figura 3.6 Célula 4T SRAM sem carga resistiva

A área de uma célula 4T SRAM é de 50 a 65% da área convencional de uma célula 6T SRAM. Apesar desta vantagem, as dimensões dos transistores podem comprometer a estabilidade da célula. Outra desvantagem da célula 4T está associada as operações de escrita. Em mais detalhes, quando se escreve em una célula, um dos *bit lines* deve mudar de *Vdd* e a outra a 0, devido que a célula continua sendo dois inversores em contraposição. Durante este tempo, um dos transistores PMOS presente

em outra célula da mesma coluna não pode deixar passar a corrente de *leakage* para o interior da célula para que o nó seja mantido em 1. A célula 4T SRAM sem carga tem sido alvo de estudos até o momento [19] [20].

3.4 Operação de leitura de célula 6T SRAM

Antes de iniciar a operação de leitura, o estado dos *bit lines* (BL e BLB) são pré-carregados a *Vdd*. A operação de leitura inicia habilitando o *word line* (WL) e conectando as pré-cargas de BL e BLB com os nodos internos da célula.

A figura 3.7 mostra uma operação de leitura quando a célula armazena um 0. A voltagem de BLB (complemento de BL) permanece em Vdd. A voltagem de BL é descarregada através dos transistores NMOS Q1 e Q3 conectados em serie. Estes dois últimos transistores formam um divisor de tensão e está conectado à entrada do inversor composto pelos transistores Q4 e Q5 ilustrados na figura 3.5. As dimensões de Q1 e Q3 devem manter uma voltagem (0 + ΔV) a qual deve ser menor que V_{th} do inversor formado por Q4 e Q5 mais uma margem de segurança. A equação 1

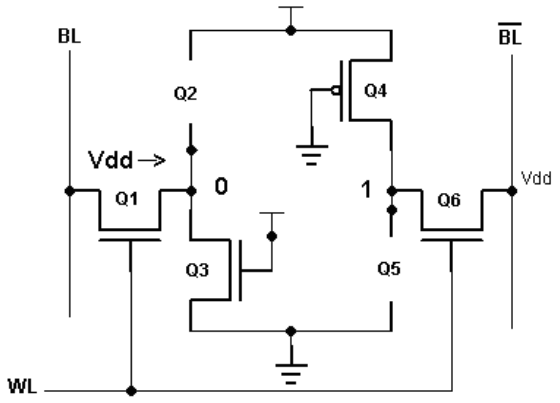

Figura 3.7 Célula de 6T CMOS SRAM durante uma operação de leitura

$$\Delta V = \frac{V_{DSATn} + CR(V_{DD} - V_{THn}) - \sqrt{V_{DSATn}^2(1+CR) + CR^2(V_{DD} - V_{THn})^2}}{CR} \quad \text{(eq. 1)}$$

De forma simplificada, o máximo valor de um 0 lógico durante uma leitura está expressado como 0 + ΔV, onde ΔV está definido na equação 1. A proporção da célula (*CR*) se define como a razão das dimensões dos transistores Q3 e Q1. Esta proporção se calcula como: $CR = (W_3/L_3)/(W_1/L_1)$. De maneira similar, o *CR* deve ser igual para os transistores Q5 e Q6. Geralmente para assegurar uma operação de leitura sem perder o dado e uma adequada margem de ruído, o *CR* deve ser maior que um e dependendo da aplicação da célula, pode variar de 1 a 2.5. Um *CR* maior assegura uma maior corrente I_{read} e maior estabilidade para a célula.

No processo de leitura o *bit line* (BL) é descarregado a $V_{BL} = V_{DD} - \Delta V$ por meio do transistor Q1, o nível de V_{BL} é suficiente para que o *sense amplifier* (SA) detecte uma diferença entre os *bit lines* (BL e BLB) e entregue o valor lido.

3.5 Operação de escrita de célula 6T SRAM

Na operação de escrita um dos *bit lines* (neste caso BLB) é conduzido do estado de pré-carga (Vdd) para 0. Se o transistor Q4 e Q6 são apropriadamente dimensionados, então a célula muda de estado como resultado de operação de escrita. O dado que desejamos escrever deve ser colocado no BL. Se desejamos escrever 0 em uma célula que contém 1, a descarga se produz mediante o transistor de acesso Q1. Se desejamos escrever 1 na mesma célula, não é produzida nenhuma descarga porque a voltagem nos extremos do transistor Q1 é a mesma. Uma medida das células SRAM sobre a capacidade de escrita é definida como *write margin. Write margin* é definido como a voltagem mínima requerida da *bit line* para que a célula mude de estado [10].

O valor da margem de escrita está em função do desenho da célula, o tamanho do bloco da memória e a variação de processo.

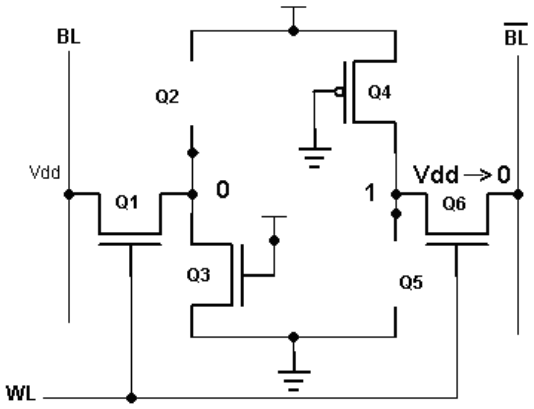

Figura 3.8 Célula de 6T CMOS SRAM durante uma operação de escrita

A função dos transistores *pull-up* Q4 e Q2 da figura 3.8 é somente a de manter o valor lógico 1 em alto nível e impedir a descarga durante a retenção do dado, quando os transistores de acesso estiverem inativos.

Geralmente para minimizar a área da célula e aumentar a densidade de área, as medidas de *pull-up* e dos transistores de acesso são selecionadas para serem mínimas e aproximadamente iguais.

3.6 Sense Amplifier

Sense Amplifier (SA) é um componente importante para a operação de leitura. O desenho SA define o rendimento de medida dos *bit lines*, da velocidade de leitura e do consumo de energia. O SA é um circuito separado do bloco de memória. A função principal do *sense amplifier* é amplificar a pequena diferença de voltagem entre os *bit lines* devido à operação de leitura e entregar o dado lido. Dependendo do desenho o SA o tempo de operação de leitura se reduz acelerando o processo de descarga dos *bit lines*. O SA permite que as dimensiones da célula sejam menores porque cada célula não necessita descarregar completamente.

As restrições para um SA são definidas pela amplitude de sinal, pelo ganho e pela tolerância às condições ambientais. A área do SA pode variar dependendo da arquitetura das colunas do bloco de memória. Um SA pode ser compartilhado por outras colunas sendo estas multiplexadas.

Geralmente os parâmetros do *sense amplifier* incluem:

Ganho $A = V_{out} / V_{in}$

Sensibilidade $S = V_{in\_min}$ - sinal mínimo a ser detectado

Offset V_{offset} y I_{offest} – diferença entre saída e entrada em modo comum.

Rejeição em modo comum CMRR - razão de amplificação entre modo diferença e modo comum.

Tempo de elevação t_{rise}, tempo de descida t_{fall} – 10% a 90% do sinal transiente.

Delay do SA t_{sense} – tempo medido desde 50% de V_{BL} até que a saída do SA chegue a 50% de V_{out}

O SA freqüentemente utiliza componentes desconsiderando o tamanho de *length (L)* e *width(W).* Porque é desenhado para um bloco de memória particular, com ganho e restrições de tempo e *layout.*

A diferença de tensão entre os níveis dos *bit lines* é um fator que determina o tempo total de leitura da célula e também determina a velocidade da memória. O SA rejeita o ruído em modo comum introduzido em ambas as linhas dos *bit lines.* Ruídos como, picos na corrente, acoplamento capacitivo entre os *bit lines* e entre o *word line.* O ruído em modo comum é atenuado pelo CMRR e o sinal verdadeiro é amplificado.

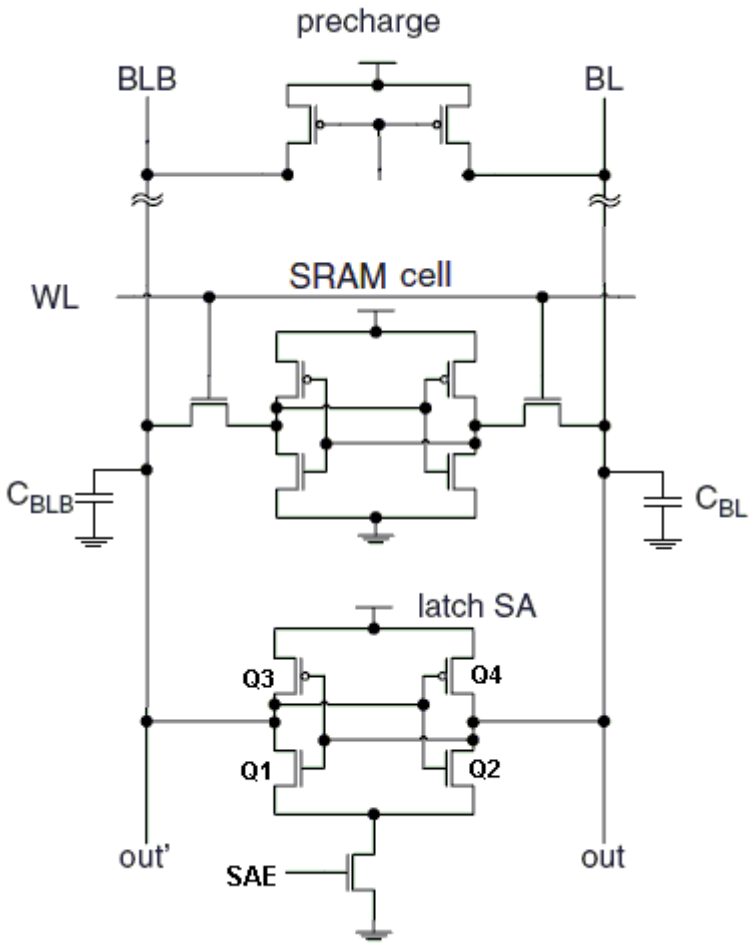

Figura 3.9 *Sense Amplifier* do tipo latch

O processo pré-leitura inicia carregando as entradas ao SA para que estejam no mesmo nível $V_{DD} \approx V_{BL} = V_{BLB}$, logo, os níveis de pré-carga são armazenados nos capacitores dos *bit lines* C_{BL} e C_{BLB} (figura 3.9). Então, para iniciar a operação de leitura o *word line* (WL) é ativado por meio do decodificador de endereços e então se inicia a descarga de um dos *bit lines* (dependendo do valor armazenado na célula), a descarga gera uma diferença de voltagem entre os *bit lines* na entrada do *sense amplifier*. Quando a diferença alcança o valor de sensibilidade *S* do SA, se ativa o habilitador do SA (SAE) para que o sinal possa ser amplificado e finalmente o valor de saída é o dado armazenado na célula. Para completar a operação de leitura, os sinais de WL e SAE são desativado.

A figura 3.9 ilustra um *sense amplifier* do tipo *latch*. Este tipo de SA é composto por um par de inversores opostos, similar ao núcleo da célula 6T SRAM [10].

3.7 Circuito de escrita

O circuito de escrita se encarrega de ingressar o dado na coluna do bloco SRAM, posteriormente o decodificador de linha se encarrega de ativar o WL para escrever na célula. A função do circuito é descarregar rapidamente um dos *bit lines* do nível de pré-carga ao 0. Normalmente, o circuito de escrita é ativado pelo sinal de *write enable* (WE). A ordem cujos WL e WE são ativados é irrelevante para uma operação de escrita. A figura 3.10 mostra três circuitos de escrita típicos.

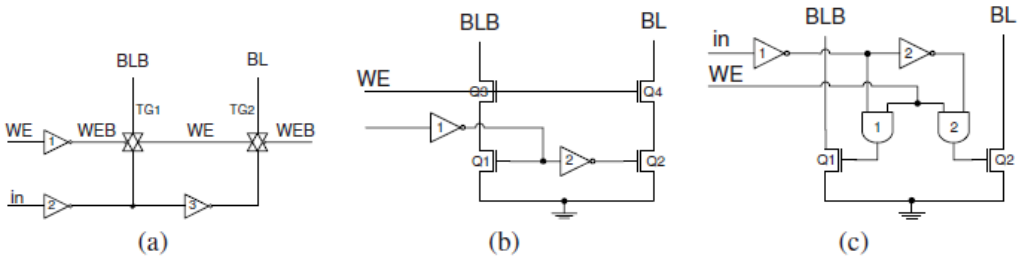

Figura 3.10 Circuitos de escrita

No circuito da figura 3.10a o dado ingressa pelo terminal *in* do inversor 2 e o complemento do dado pelo inversor 3 para o BLB e BL respectivamente. Para que os dados passem a ser armazenados, o WE deve ativar os transistores TG1 e TG2 mediante o inversor **1**. Se desejamos armazenar **0** na memória o valor de *in* deve ser **0** para que a saída do inversor 3 também esteja em **0** e para quando o WE for ativado o transistor TG2 permita a descarga do BL. A figura 3.10b usa transistores de passo (Q1, Q2, Q3, Q4) para formar caminhos de descarga de um dos *bit lines.* No circuito b, o complemento do dado ativa um dos transistores Q1 ou Q2 para produzir a descarga de BLB ou BL, dependendo do dado que está sendo escrito.

Outro circuito de escrita está ilustrado na figura 3.10c. Quando ativamos o WE as duas portas AND permanecem esperando pelo dado *in*. Dependendo do valor de *in,* uma das portas AND terá na saída **1** o qual, mediante o transistor Q1 ou Q2, gera a descarga em um dos *bit lines* o levando a **0**.

Em resumo, o ciclo de escrita tem inicio com a aplicação do valor a escrever nos *bus* de dados. Se desejamos escrever **0**, ajustaremos. E para escrever **1** realizamos o processo inverso, ou seja, BL a **1** e BLB a **0**. Uma vez feito isso, se ativa o bus WL e o WE, e o dado permanece armazenado. Em qualquer um dos circuitos, a descarga dos *bit lines* deve ocorrer apesar dos C_{BL} serem altos. Somente um circuito de escrita é necessário para uma coluna de bloco de memória SRAM.

4 Modelo de falhas em SRAMs

Este capítulo apresenta um conjunto de modelos de falhas específicos para SRAMs. Em mais detalhes, este capítulo descreve os modelos funcionais associados a SRAMs bem com os modelos de falhas desenvolvidos para representar *resistive-open defects*.

4.1 Introdução

Avanços relacionados aos circuitos integrados em larga escala (VLSI) fez com que a tarefa de testar uma memória se tornasse uma tarefa extremamente complexa e isto intensificou a pesquisa de novas e mais eficientes metodologias de teste capazes de detectarem diferentes tipos de falhas. Entende-se que um modelo de falhas é uma representação sistemática e precisa de um comportamento inadequado, destinado a ser simulado e testado. Sendo uma representação de defeitos físicos, uma falha representa funcionalmente uma imperfeição física na funcionalidade do componente. Uma falha também é vista como um desvio da função normal, ou seja, quando um sistema deixa de proporcionar o serviço que deveria prestar. E estas podem ser causadas por múltiplos defeitos. [10]

Durante a modelagem de falhas, assume-se a mesma probabilidade de ocorrência para todas as falhas. Contudo, a probabilidade de falha é em função da probabilidade do defeito que a causa. Os defeitos são modelados como alterações locais no bloco de memória SRAM e estas alterações podem ser a nível de transistor ou a nível de células.

O comportamento elétrico de cada defeito é analisado e classificado em um modelo de falha.

As falhas transientes em SRAM muitas vezes podem ser causadas por partículas de alta energia, descargas eletrostáticas (ESD Event) [11] acoplamento capacitivo e *ripple* na linha de alimentação.

4.2 Modelos de falhas

O modelo funcional de uma SRAM pode ser descrito de acordo com o bloco funcional onde a falha ocorre. Assim, a partir de cada bloco é possível descrever o tipo de falha baseado no comportamento ou na função que o mesmo desempenha [32]. A figura 4.1 mostra o diagrama de blocos da SRAM adotada durante as simulações elétricas realizadas neste trabalho.

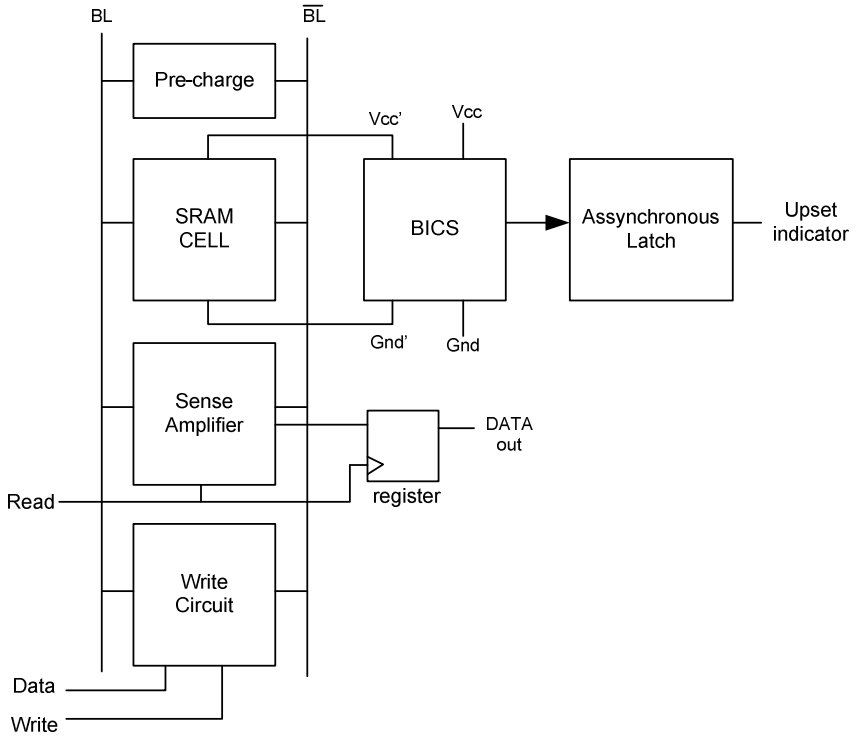

Figura 4.1 Diagrama de bloco da memória adotada.

O diagrama de blocos esta conformado por o circuito de pré-carga, a célula de memória SRAM, o BICS, o *assynchronous latch*, *sense amplifier*, circuito de escrita e circuito de leitura. Neste caso estudaremos somente modelos de falhas no bloco da memória. Os modelos de falhas adotados são de tipos funcionais e *resistive-open defects*.

4.2.1 Falhas funcionais

Os blocos funcionais relacionados a uma SRAM podem ser alvos de diferentes tipos de falhas, que por sua vez podem ser representadas através de modelos funcionais. Esses modelos são desenvolvidos a partir de defeitos físicos que podem ser o resultado de conexões abertas, curto circuito entre vias, falta de conexão ou contato extra. Em mais detalhes, este modelos são desenvolvidos com o intuito de representar os possíveis defeitos físicos que podem ocorrer em uma SRAM. Além disto, eles também podem ser associados a falhas que envolvem uma única célula ou um grupo de células. Segundo Dekker [32] os modelos de falhas podem ser divididos em três blocos dependendo da localização: bloco da memória, decodificador de endereço e a lógica de escrita e leitura. Assim, os principais modelos de falhas funcionais presentes na literatura são definidos a seguir:

Stuck-at fault (SAF): Uma célula é desenhada para poder armazenar 0 ou 1, e poder ser lida muitas vezes sem destruir o dado. Se por alguma razão a célula permanece em 0 ou 1 sem poder escrever o complemento, isto significa que a falha é do tipo SA0 ou SA1 [5].

Stuck-open (SOF): Um SOF significa que a célula não pode ser acessada para nenhuma operação. Talvez por causa de una linha aberta no WL [5].

Coupling fault (CF): Este modelo de falha envolve duas células, uma célula vitima e uma célula agressora. CF podem ser divididos em *inversor coupling fault* (CFin), *idempotent coupling faults* (CFid) e *state coupling fault* (CFst).

A falha CFin se manifesta quando duas células estão unidas. Neste modelo existe uma célula agressora e uma célula vítima. Assim, durante uma transição do conteúdo da célula agressora, a célula vitima inverte o valor de seu conteúdo, ou seja, se escrevemos 0 na célula agressora (1w0) isto faz com que o conteúdo armazenado na célula vitima seja invertido.

Um CFid se apresenta quando a célula agressora realiza uma transição e como conseqüência o conteúdo da célula vítima é alterado para um determinado valor que pode ser 0 ou 1.

Um CFst ocorre quando o conteúdo da célula vítima é alterado a partir de um determinado estado armazenado na célula agressora. Diferentemente de CFid, isso é causado pelo estado lógico da célula agressora.

4.2.2 Falhas associadas a *Resistive-Open defects*

Atualmente, r*esistive-open defects* representam um dos problemas mais significativos em tecnologias VDSM devido à presença de muitos *layers* interconectados e a inclusão de muitas conexões entre cada *layer* [23]. Um r*esistive-open defects* é definido como um defeito resistivo entre dois nodos de um circuito que seriam conectados [24]. R*esistive-open defects* dependem da temperatura e da voltagem de alimentação (V_{cc}), assim, os resultados de testes são diferentes de acordo com V_{cc} e com a temperatura de prova. Além disso, este tipo de defeitos induz a falhas dependentes de tempo como falhas de *delay.*

Resistive-open defects são modelos de acordo com a figura 4.2. Em mais detalhes, este tipo de defeito é modelo a partir da inserção de uma serie de resistores posicionados no interior de uma célula SRAM.

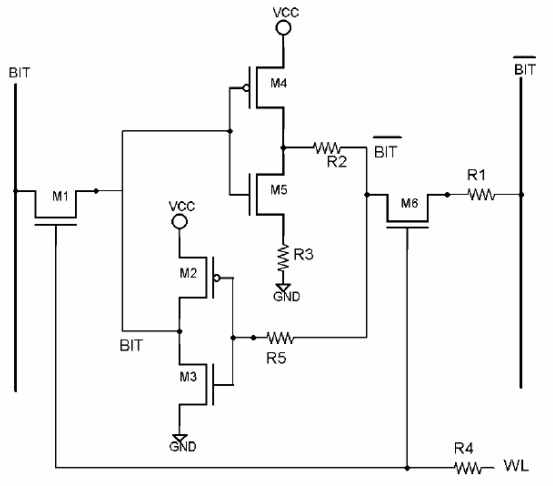

Figura 4.2 *Resistive-open defects* inserido na célula 6T SRAM

Assim, *resistive-open defects* são associados a seguintes defeitos:

Transition fault (TF): A célula deve ser capaz de sobrescrever um 0 em uma célula que possui 1, ou vice-versa. Se, por exemplo, uma célula falha ao realizar uma transição de 0 para 1 ou vice-versa, indica que a célula apresenta um TF. Um TF pode se apresentar como um SAF se a célula assume o estado no qual esta não pode mais mudar ao complemento do dado inicial. Contudo, diferentemente de SAF, uma célula com TF pode regressar ao estado anterior devido a outra falha [10].

Read Destructive Fault (RDF): Dizemos que una célula possui RDF se uma operação de leitura é capaz de mudar o estado da célula e entregar um valor incorreta na saída. Por exemplo, se a célula contem 1 e se realiza uma leitura, esta operação muda o estado da célula deixando 0 nesta. Como resultado, o *sense amplifier* entrega um 0 na saída.

Deceptive read Destructive faul (DRDF): Dizemos que uma célula possui DRDF se a operação de leitura realizada na célula entrega um valor correto na saída do *sense amplifier* e a célula muda de estado. Como o exemplo anterior, se uma célula possui 1 e realizamos a leitura, esta operação muda o estado da célula para 0, mas diferentemente a RDF, o DRDF entrega um 1 enquanto a célula possui 0.

Incorrect Read Fault (IRF): Dizemos que uma célula tem um IRF se uma operação de leitura realizada na célula entrega um valor incorreto na saída, e o valor correto ainda esta armazenada na célula.

A tabela 4.1 mostra a relação entre cada resistor utilizado para modelar um determinado defeito com seu respectivo modelo de falha.

Resistor	Fault Model
R1	TF
R2	RDF/DRDF
R3	RDF/DRDF
R4	IRF/TF
R5	TF

Tabela 4.1 Resistor relacionado ao modelo de falhas.

Neste caso, os resistores estão distribuídos para gerar uma falha no sentido de uma transição, por exemplo, se o R1 está no lado BLB este gera uma TF de 0 para 1. Mas se R1 esta no lado de BL então R1 gera um TF de 1 para 0. Esse comportamento é repetido para cada um dos modelos de falhas acima descritos devido à simetria da célula.

4.3 Conclusões

Neste capitulo foram apresentados dois modelos de falhas para memórias SRAM, os quais serão adotados nos seguintes capítulos para validar a metodologia de teste proposto. Os dois métodos apresentados são baseados no uso de resistores no interior da célula, com a diferença que os modelos de falhas funcionais utilizam resistores com baixa impedância, ou seja, para provocar os defeitos na célula os resistores tendem a 0Ω. E os modelos de falhas associados à *resistive-open defects* utiliza resistores que tendem a circuito aberto.

5 Teste de SRAMs

Devido ao crescimento da área da memória e ao desenvolvimento da tecnologia VDSM, os testes das memórias estão se tornando cada vez mais complexos. Assim, uma memória com maior número de células SRAMs requer maior tempo para o teste, o que se traduz em maior custo da produção. Um teste eficiente e econômico deverá fornecer uma maior capacidade de detecção em menor tempo.

Nesta seção serão apresentadas as duas metodologias de teste veremos duas metodologias previu a nossa metodologia proposta.

5.1 Metodologias baseadas em *software*

A metodologia de teste de SRAM tradicional é baseada em *software.* March Teste é o algoritmo de teste mais popular devido a ser personalizável.

5.1.1 March Teste

March teste consiste em uma metodologia de teste de SRAMs extremamente eficiente e capaz de garantir a detecção de um vasto número de modelos de falhas.

O March teste é definido como uma seqüência de elementos March. Cada elemento tem um número de operações seqüenciais os que são aplicados em uma célula de cada vez seguindo uma determinado ordem pré-definida. Essa ordem está associada aos endereços da memória e pode ser crescente (do endereço 0 para o endereço *n-1*) ou decrescente (do endereço *n-1* para o endereço 0). A complexidade do algoritmo é definida pelo número (*n*) de operações totais. O símbolo atribuído para a ordem crescente é ⇑ e para a ordem decrescente é ⇓. O símbolo ⇕ indica que a ordem de

execução é irrelevante podendo ser crescente ou decrescente. A tabela 5.1 mostra alguns exemplos conhecidos de March Test.

March Tests and their complexity		Equivalent Known March Test
$\{\Uparrow w_1 \Downarrow r_1 w_0 \Downarrow r_0\}$	$4n$	MATS ($4n$)
$\{\Uparrow w_1 \Uparrow r_1 w_0 \Downarrow r_0 w_1\}$	$5n$	MATS+ ($5n$)
$\{\Uparrow w_0 \Uparrow r_0 w_1 \Downarrow r_1 w_0 \Uparrow r_0\}$	$6n$	MATS++ ($6n$)
$\{\Uparrow w_0 \Downarrow w_1 \Downarrow r_1 w_0 \Uparrow r_0 w_1\}$	$6n$	MarchX ($6n$)
$\{\Uparrow w_1 \Uparrow r_1 w_0 \Uparrow r_0 w_1 \Downarrow r_1 w_0 \Downarrow r_0 w_1 \Downarrow r_1\}$	$10n$	March C- ($10n$)
$\{\Uparrow w_0 \Uparrow r_0 w_1 w_0 \Downarrow r_{01}\}$	$5n$	Not Found

Tabela 5.1 Exemplos de March Test [36].

5.2 Metodologias baseadas no monitoramento de corrente

Em condições normais de operação, os transistores CMOS consomem maior corrente só no momento da computação. Isso significa que, a corrente estática é muito menor em comparação à corrente de comutação. No entanto, a corrente estática pode se incrementar consideravelmente devido à presença de defeitos físicos.

Assim, um incremento na corrente estática pode ser utilizado como um indicador de defeitos nos circuitos CMOS bem como em células de SRAMs..

5.2.1 Built-in current sensor em SRAMs

Na década passada o circuito *Built-in current sensor* (BICS) foi proposto em [4] [25] [26] [27] [28], para detectar falhas em SRAMs geradas a partir da radiação. Cada BICS monitora a corrente de alimentação das células e detecta o consumo anormal de

corrente produzido nas células SRAM como resultado de um *single-event upset* (SEU). De modo geral, o BICS é conectado em cada uma das colunas de células da memória e monitora o seu respectivo consumo de corrente. Assim, se o conteúdo armazenado em uma determinada célula é alterado (*bit-flip*), o BICS em combinação com um verificador de paridade pode localizar a célula que sofreu *upset* e isto consequentemente permite a correção do conteúdo da célula seja corrigido..

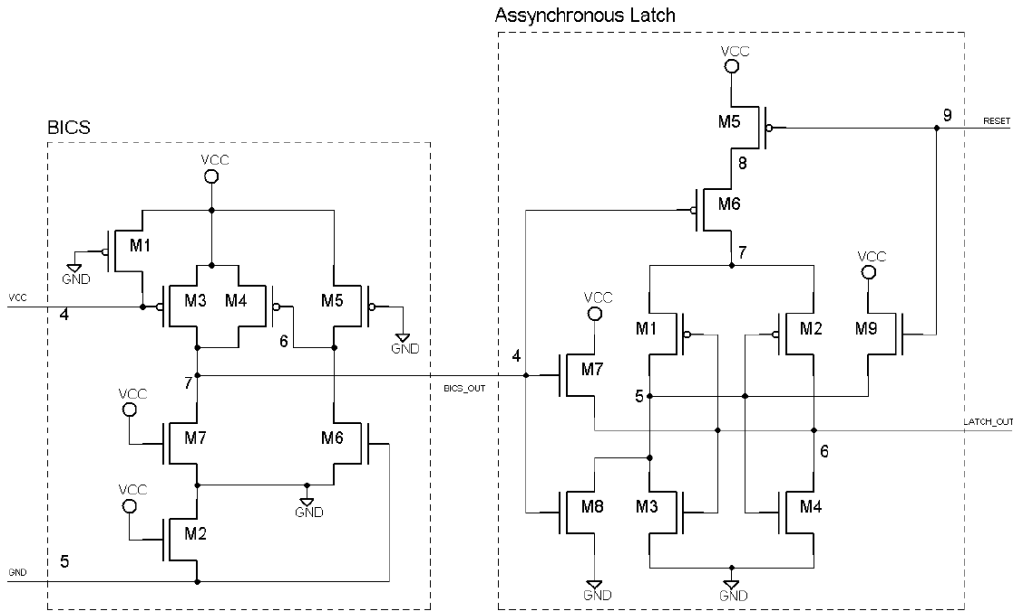

Figura 5.1 Buil-In Current Sensor (BICS)

O circuito BICS está associado a um *latch* assíncrono (*Assynchronous Latch*) conforme ilustrado na figura 5.1. Geralmente, o consumo anormal de corrente é produzido a partir de uma mudança do estado da célula como resultado de um SEU, por exemplo. É importante salientar que a amplitude e duração do pulso deve ser suficiente para que o *latch* possa capturar-lo e assim indicar um *upset* na saída.

Se acontecer um maior consumo de corrente na célula, o nó V_{cc}' da figura 5.1 sofre uma queda de tensão proporcional à corrente consumida. Dependendo da queda de tensão, o V_{th} do transistor M3 do *sensing cell* pode colocar o nó 7 em um nível próximo a V_{cc} e ativar o transistor M7 do bloco de *assynchronous latch,* como resultado o *latch* indica uma falha.

Na ausência de excesso de corrente, os níveis de V_{cc}' e *Gnd'* são próximos a V_{cc} e Gnd, então o transistor M3 do *sensing cell* fica inativo devido que o *gate* do transistor M3 é V_{cc} e o BICS adota um nível 0 na saída mediante o transistor M7.

As dimensões dos transistores do BICS são redimensionadas para regular a sua sensibilidade acordo com o incremento da corrente. Em termos gerais, para aumentar a sensibilidade do BICS é necessário aumentar a impedância entre os terminais de fonte e dreno dos transistores. Uma impedância maior no nó 7 em relação ao *Gnd* favorece um maior nível de tensão quando o transistor M3 é ativo. A mesma situação pode ser observada quando consideramos o M1, uma maior impedância em relação a V_{cc} favorece a condução do transistor M3. Como resultado o BICS mantém um estado favorável para detectar falhas na célula.

5.3 Conclusões

Neste capitulo, foram apresentadas duas metodologias de teste de SRAMs. Uma metodologia baseada em software como é o algoritmo *March Teste* e uma técnica baseada em monitoramento de corrente baseado no uso de *Built-in current sensor* (BICS). No seguinte capitulo, formaram parte da nova metodologia proposta nesta dissertação de mestrado.

6 Metodologia proposta: March+BICS

A metodologia de teste de SRAMs proposta nesta dissertação de mestrado baseia-se no uso de BICS em combinação com um algoritmo de teste otimizado baseado em elementos *March*. A metodologia proposta tem como objetivo garantir a detecção de falhas permanentes associadas a modelos de falhas funcionais bem como a modelos de falhas associados a *resisitive-open defects* gerados durante o processo de manufatura ou durante a vida útil de SRAMs. É importante salientar que é a primeira vez que a utilização do BICS é proposta com o intuito de garantir a detecção de falhas permanentes, visto que todos os trabalhos apresentados na literatura até hoje visam a detecção de falhas transientes.

Basicamente o BICS foi conectado sobre os terminais de alimentação da célula, mostrado na figura 6.1. Ou seja, mediante o BICS se fornece corrente à célula. Qualquer consumo anormal de corrente será facilmente detectado por o BICS. Na pratica, o BICS monitora cada coluna de um bloco de memória, isso significa que se temos um bloco de M endereços de *n* bit cada um, necessitamos *n* BICS para cobrir a memória.

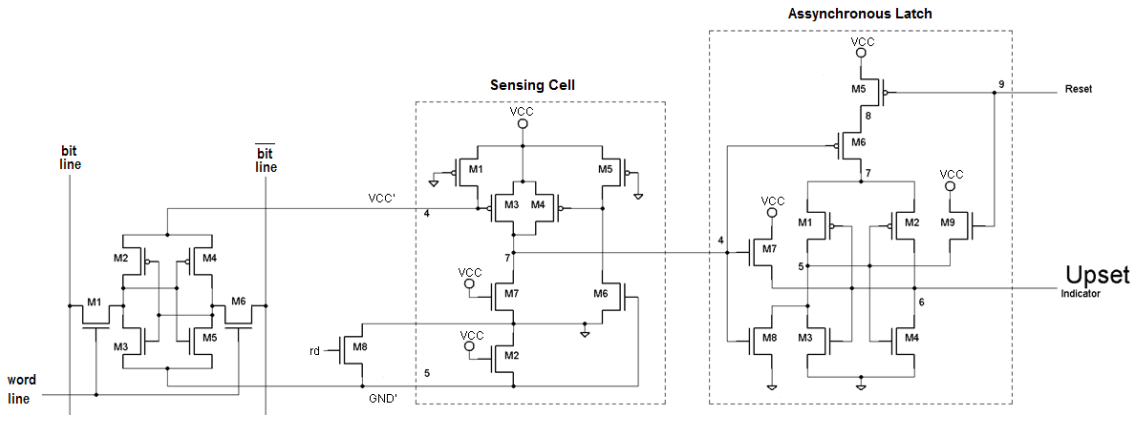

Figura 6.1 BICS conectado a uma célula SRAM.

Finalmente, a metodologia proposta baseia-se na execução de um algoritmo simplificado baseado em elementos *March*. Assim, o algoritmo otimizado para o teste de SRAMs é apresentado a seguir:

$$\Updownarrow (w1) \Uparrow (w0,r0) \Uparrow (w1,r1)$$

Observando o algoritmo acima é possível verificar que o mesmo é composto por três elementos *March* e a sua complexidade é de 5n. Cada elemento tem uma seqüência de operações aplicadas a cada endereço antes de passar ao endereço seguinte. Cada endereço pode ser orientado por *bit* ou por *word* de n *bits.* Uma operação consiste de escrita de 0 (w0), leitura da célula esperando 0 (r0), escrita de 1 (w1) e leitura da célula esperando 1 (r1). A ordem de execução do algoritmo pode ser crescente, de endereço 0 para *n*-1 representado com o símbolo $\Uparrow$, ou também pode ser decrescente, de endereço *n*-1 para 0 representado por o símbolo $\Downarrow$. O símbolo $\Updownarrow$ indica que a ordem de execução pode ser crescente ou decrescente. As cinco operações de leitura e escrita feitas nas células são realizadas com o objetivo de excitar as células para reproduzir uma falha e esta ser detectado por o BICS.

Para a validação desta metodologia foi inserindo células com defeito no interior do bloco de memórias e executando o algoritmo visto acima.

No capítulo 7 veremos os resultados das simulações de testes utilizando o metodologia proposto em células com falhas permanentes.

Comparado com o algoritmo encontrado no artigo [23], o algoritmo proposto apresenta um significante redução de tempo requerido para o teste do bloco de memória SRAM. A redução de tempo é devido à redução de número de elementos, para ilustrar melhor a situação mostramos o algoritmo proposto em [23]:

$$\Updownarrow (w0) \Uparrow (r0,w1) \Uparrow (r1,w0) \Downarrow (r0,w1) \Downarrow (r1,w0) \Updownarrow (r0)$$

Podemos observar que a algoritmo proposto utilizando BICS reduz o numero de operações de 10 para 5 elementos e conseqüentemente reduz o tempo de teste um 50%.

Como conclusão a vantagem desta nova metodologia é que esta apresenta uma significante redução de 50% de tempo requerido para o teste de SRAMs comparado com o algoritmo modificado *March* C- apresentado em [23].

7 Resultados experimentais

Neste capitulo serão apresentadas as simulações elétricas relacionadas a cada modelo de falha permanente descrito no capítulo 4.

7.1 Introdução

As simulações elétricas visam validar e demonstrar a capacidade de detecção da metodologia proposta no que diz respeito a falhas funcionais e a falhas associadas aos *resistive-open defects.* Convém salientar que a modelagem das falhas alvo foi feita a partir da inserção de resistores em uma célula SRAM. Assim, cada resistor está associado a uma única falha associada a um determinado modelo. A célula com defeito é inserida no bloco de memória, onde o BICS monitora o consumo de corrente.

A estrutura da memória utilizada para as simulações relacionadas a *resistive-open defects* é composta de um *sense amplifier*, um circuito de escrita, um circuito de pré-carga e um *flip-flop* D*,* conforme a figura 7.1. Todas as simulações do bloco de memória SRAM foram feitas no HSPICE da Synopsys [29] [30]. Para a visualização dos resultados foi utilizado CosmosScope também da Synopsys [31].

A partir da descrição HSPICE da estrutura, cada uma das falhas alvo é excitada através da execução de um determinado conjunto de operações de leitura e escrita de o algoritmo proposto. O algoritmo tem ordem ascendente ou descendente dependendo do tipo de falha a detectar.

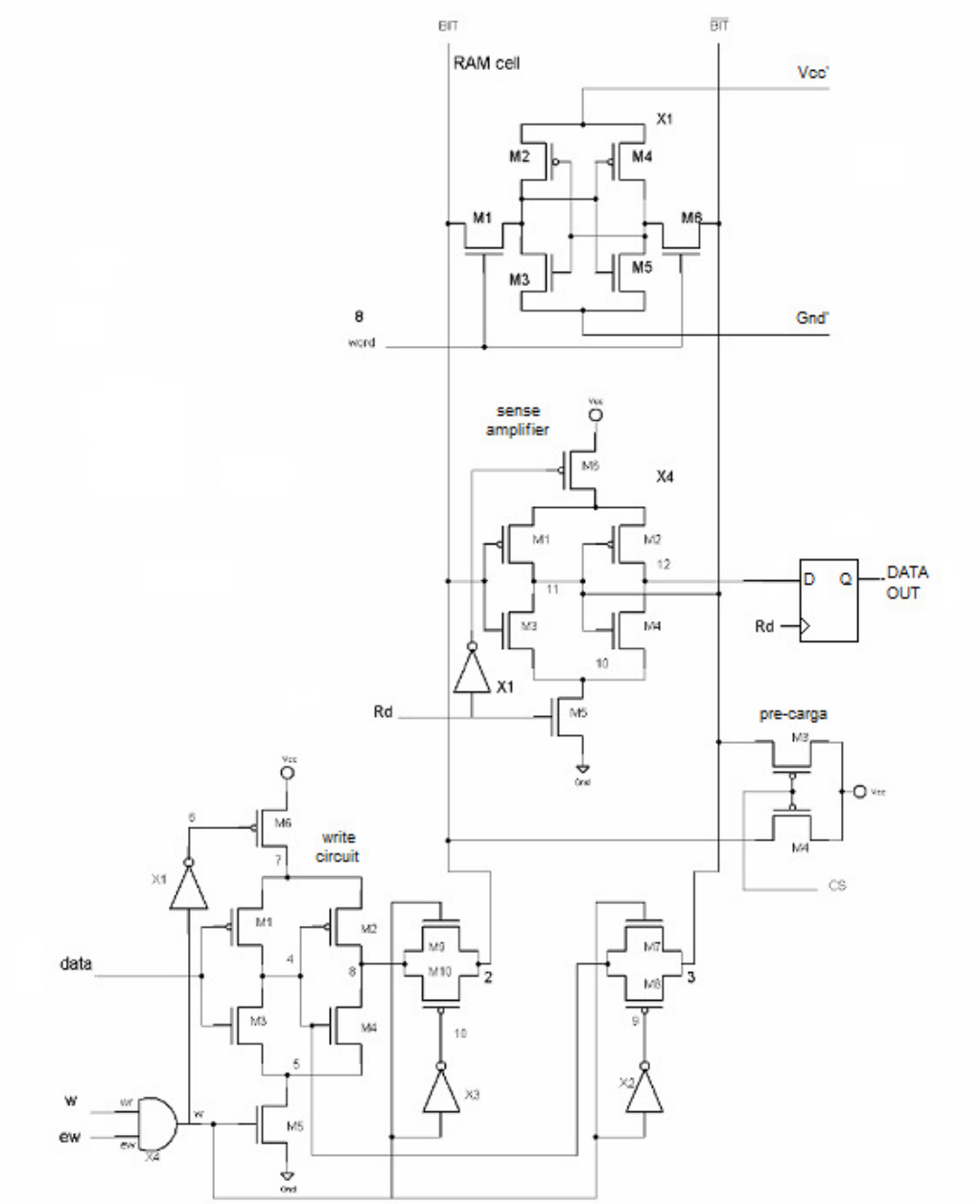

Figura 7.1 Circuito de memória SRAM adotado.

7.2 Falhas funcionais

Para o desenvolvimento das simulações de falhas funcionais, foi necessário gerar um bloco de memórias de 8x8 *bits* figura 7.2. O bloco de memória é constituído de 8 palavras de 8 *bits* cada uma, (b0, b1,…b7). Cada coluna tem um BICS totalizando 8

BICS. É importante salientar que no inicio foram realizados algumas simulações com um bloco de memória contendo apenas células sem qualquer tipo de defeito, com o objetivo de validar a memória gerada e garantir que a mesma funcione corretamente. Logo depois, foram inseridas as células com defeito e realizadas várias operações de escrita em todo o bloco de memória com o objetivo de verificar se as mesmas são capazes de alterar o consumo de corrente da célula defeituosa e conseqüentemente excitar o BICS.

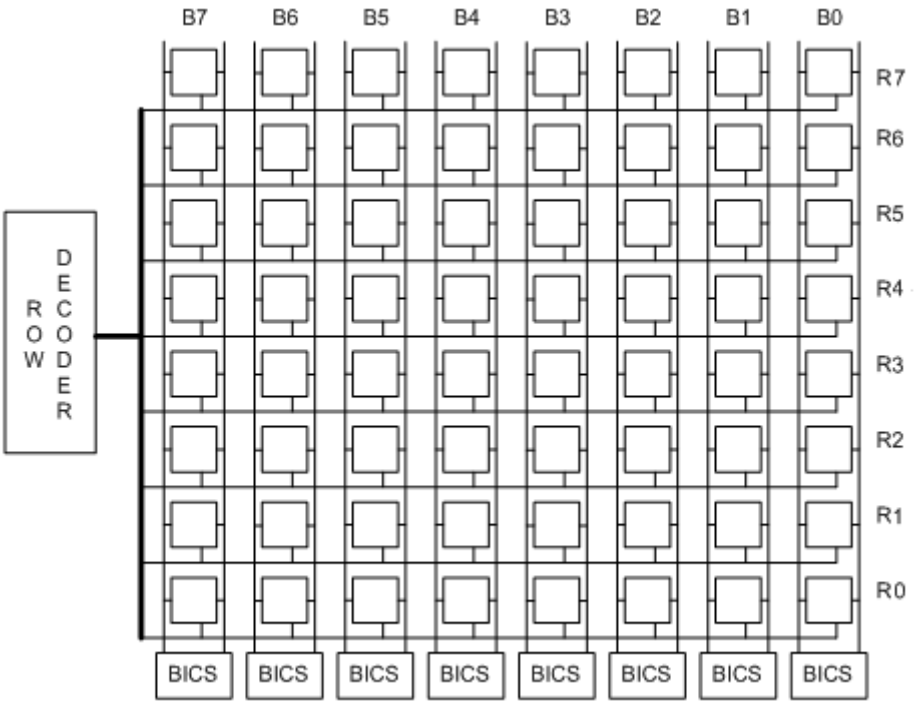

Figura 7.2 Bloco de memória SRAM de 8x8 bits.

Em resumo, quando se realiza uma operação de escrita em uma célula com defeito, dependendo do defeito, esta produz um consumo de corrente anormal, o qual é detectado pelo BICS.

Assim, foram simuladas várias seqüências distintas de operações de escrita com o intuito de promover a detecção de *Stuck-at faults* e *Coupling faults*. É importante salientar que alterar a ordem de escrita dos valores 0 e 1 permite a detecção de diferentes falhas tais como *Stuck-at 0* e *Stuck-at 1*. Os defeitos que levam a tais falhas

são causados por impedâncias inadequadas dos transistores que compõe o núcleo. Para modelar estes defeitos, são inseridos resistências em determinados lugares do núcleo em diferentes posições de acordo com a falha alvo. Os valores de cada resistência foram determinados experimentalmente, ou seja, foram considerados vários valores ate obter o valor apropriado para produzir uma falha permanente.

Para criar os defeitos no núcleo por meio de resistências, foi considerado que um circuito aberto é equivalente a uma resistência muito alta e que um curto circuito é equivalente a uma resistência muito baixa. Então, para simular um defeito de curto circuito ou de uma impedância pequena, iniciamos com um valor de resistência muito alto, que não gere nenhum defeito, e começamos a diminuir o valor até encontrar os defeitos na célula. Assim, o valor obtido é utilizado durante a simulação.

7.2.1 Simulação de *stuck-at Fault*

A modelagem de *stuck-at fault* é dividida em duas simulações, simulação de *stuck-at 0* e simulação de *stuck-at 1*.

Simulação de *stuck-at* 0:

Uma falha de *stuck-at-0* (ST0) se manifesta quando não é possível escrever outro valor além de 0. Para modelar este defeito na célula foi utilizado uma resistência entre o lado não inverso (S) e *virtual ground* (Gnd'), como mostra a figura. 7.3. Nesta simulação a freqüência de operação foi 50Mhz, V_{cc} = 3.3vcd a temperatura 27ºC e em tecnologia 350nm.

O valor da resistência foi determinado experimentalmente a partir de um valor mais alto, diminuindo este até que não seja possível armazenar um 1.

O que se consegue com isto é reduzir a impedância equivalente do nó S em relação ao *virtual ground* (Gnd'). Assim a corrente que circula por este nó gera uma baixa queda de tensão que por sua vez ativa o transistor M4 mantendo um 1 lógico no lado inverso (SB). Desta forma, o resistor R1 faz com que a célula armazene um 0

permanente. Finalmente encontramos o valor de R1 igual a 8.9KΩ o qual causa *Stuck-at 0* na célula.

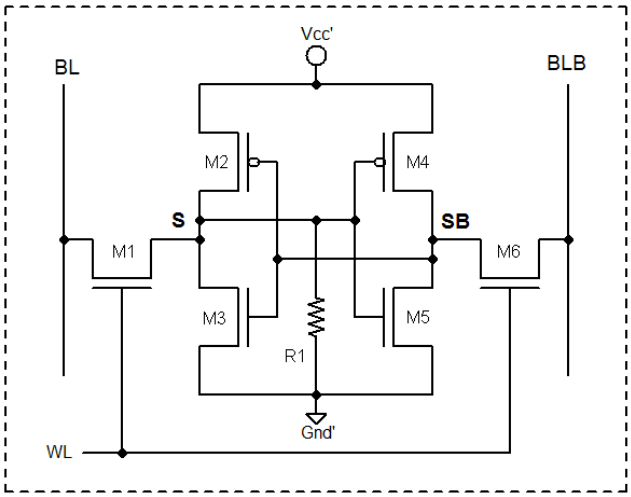

Figura 7.3 Modelagem de *stuck-at 0*

O algoritmo utilizado para detectar a falha ST0 consiste em ⇑w0⇑w1, escrevemos 0 em todo o bloco de memória em ordem ascendente, logo após escrevemos 1 em todo o bloco de memória no mesmo sentido. O BICS associado à célula com defeito indicará um erro por meio de sinal de UPSET. Neste caso, a célula com defeito é a célula no endereço três (R3) e na coluna um (B1). No momento que se intenta escrever um 1 a célula incrementa o consumo da corrente pela presença de R1, o qual mantém ativo o transistor M4. O consumo adicional é mediante o transistor M4, devido a que o nó SB permanece em 1 apesar que o BLB é 0 enquanto acontece a operação de escrita.

Na figura 7.4 mostra a detecção de *Stuck-at 0*. Na figura 7.4c podemos observar 8 operações de escrita no bloco, logo depois, operações de leitura dos 8 *bits* em paralelo. Pode-se observar que tem uma única célula contendo 0 na coluna um (B1) mediante o sinal (b1_out). Na figura 7.4a mostra o momento da detecção por parte do BICS mediante o sinal (bics_1). Neste momento o se faz a escrita de 1 na célula com falha ST0. Na figura 7.4b mostra o *assynchronous latch* no momento que captura o sinal do BICS mediante o sinal de saída (upset_1).

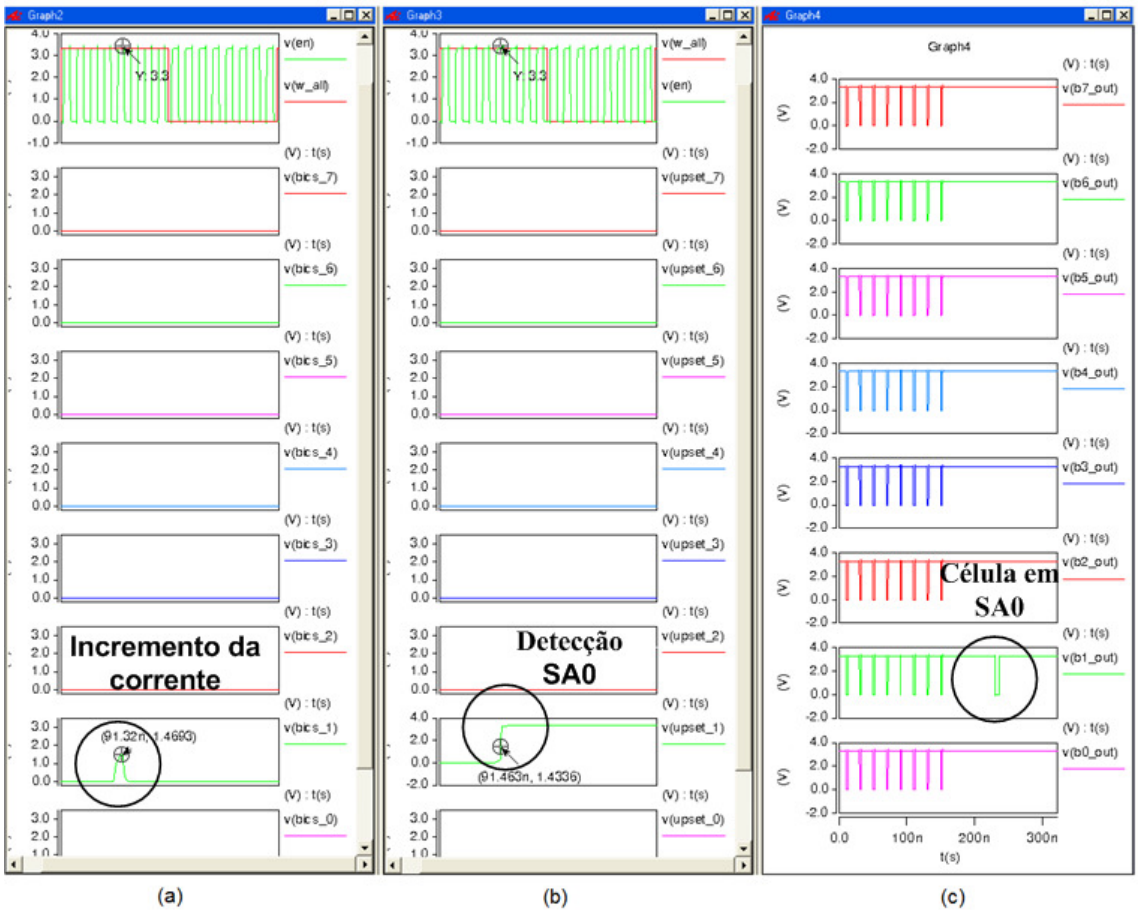

Figura 7.4 Detecção de *Stuck-at 0* no bloco de memória SRAM

A detecção de ST0 é segura para defeitos com impedância menor que 8.9KΩ.

Simulação de *stuck-at* 1:

Para esta simulação se utiliza um resistor de 8KΩ (R1 da figura 7.5) posicionada paralelamente em relação ao transistor M5, o valor desta resistência foi encontrado de maneira similar ao caso anterior. Sob estas circunstâncias, a resistência provoca uma baixa impedância no lado inverso da célula (SB) respectivo a *virtual ground* (Gnd'). Assim a corrente que circula por este nó gera uma baixa queda de tensão que por sua vez ativa o transistor M2 mantendo 1 lógico no lado não inverso (S). A freqüência de operação é 50MHz, V_{cc} = 3.3vcd a temperatura 27ºC em tecnologia 350nm.

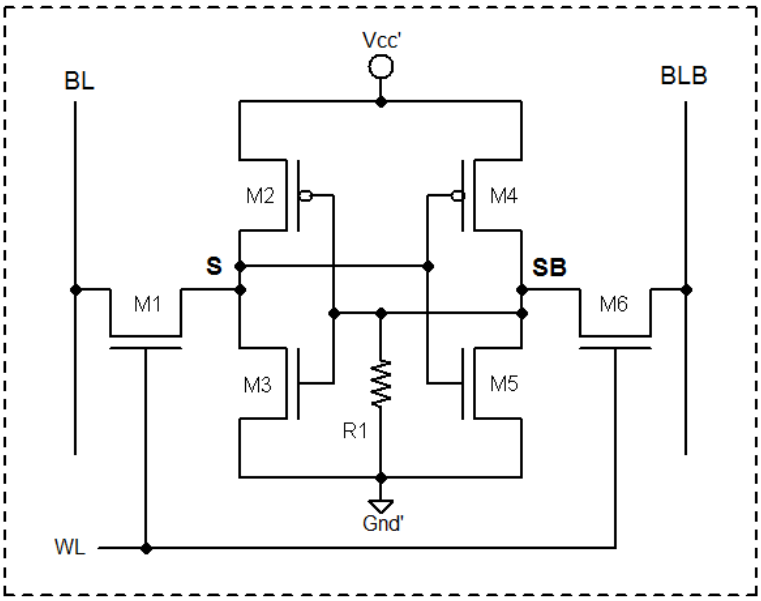

Figura 7.5 Modelagem de *stuck-at 1*

Após obter o modelo da célula, se insere o bloco de memória e se realiza as seguintes operações: ⇑w1⇑w0. Este algoritmo inicia escrevendo 1 em todas as células em ordem ascendente. Como a célula está permanentemente em 1, esta primeira operação não afeta o BICS em nada. A detecção da falha se produz quando se escreve 0 na célula, e isto provoca um maior consumo de corrente. Recordando que é o BICS que energiza as colunas de células por meio dos terminais V_{cc}' e *Gnd'*.

A detecção de *Stuck-at 1* é mostrado na figura 7.6. No sinal (bics_2) da figura (a) mostra no momento quando o BICS faz a detecção da falha ST1 em uma das células da coluna 2. O sinal (en) representa WL ativando as oito células em cada endereços para escrita ou leitura. E o sinal (w_all) representa seqüência de escrita, quando ele tem nível alto realiza-se as operações de escrita nos oito endereços, se tem nível baixo, realiza-se a operação de leitura nos oito endereços. As características da simulação são as mesmas que a simulação anterior.

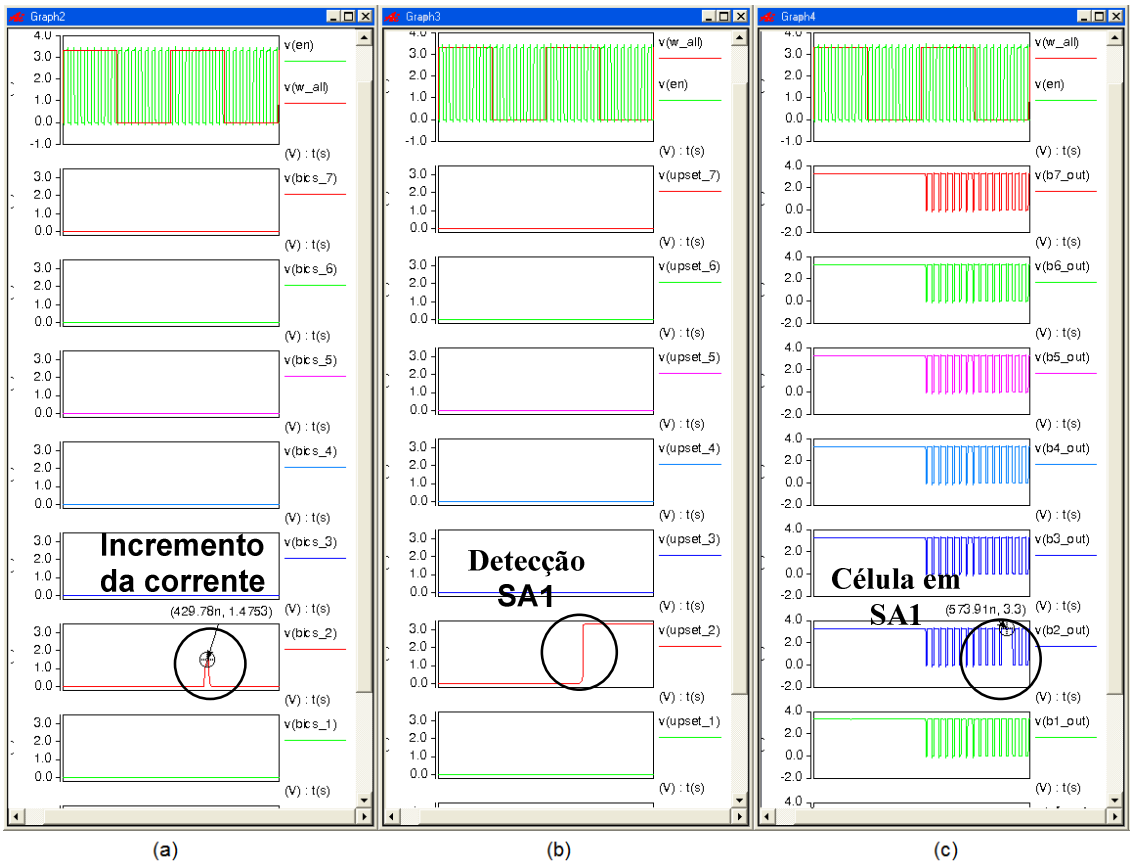

Figura 7.6 Detecção de *stuck-at 1* no bloco de memória SRAM

7.2.2 Simulação de state coupling fault

As falhas do tipo *State-coupling* (CFs) envolvem no mínimo duas células, e estas células podem estar acopladas de formas diferentes [33]. Para nossa simulação, as células estarão unidas através de um resistor R1 conforme a figura 7.7. Do mesmo modo que nas simulações anteriores, o valor da resistência foi calculado experimentalmente a partir de um valor extremadamente alto (MΩ) até encontrar um valor suficientemente baixo para provocar uma conexão entre as células. Nesta oportunidade contamos com um resistor de 1.5KΩ entre as células (3,5) e (5,6). A célula (3,5) é o quinto *bit* do endereço três, e a célula (5,6) é o sexto *bit* do endereço cinco. O resistor é localizado no lado no inverso (S) de cada célula. Este defeito no

bloco de memória faz com que o valor armazenado de uma determinada célula se replique em outra.

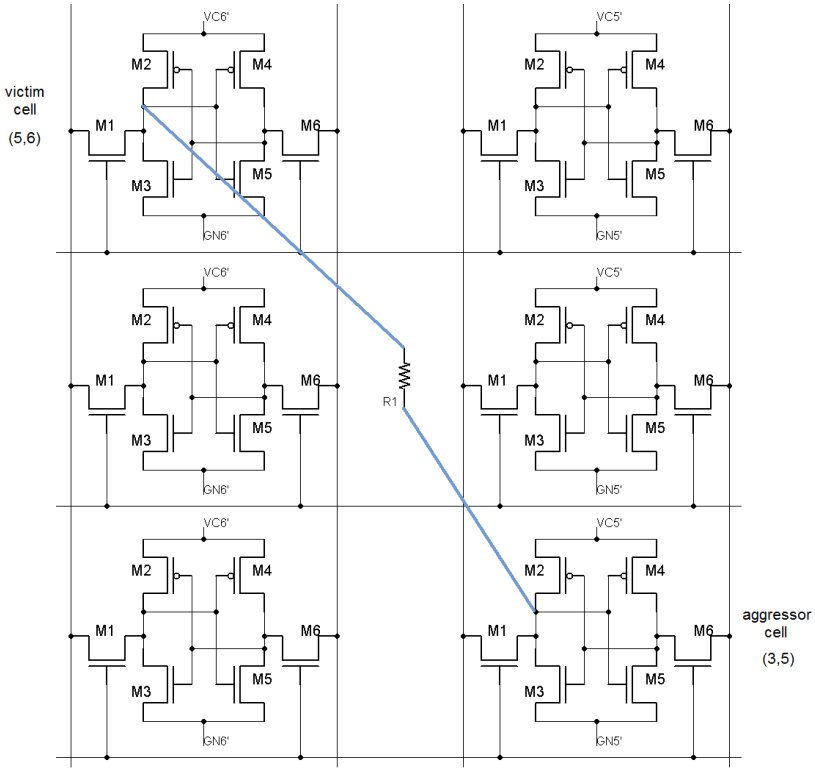

Figura 7.7 Modelagem de *coupling fault* com resistor de 1.5K

O algoritmo utilizado para detectar este tipo de falhas é ⇑w1⇑w0. As operações serão realizadas por bytes, ou seja, as escritas se realizam em oito *bits* simultaneamente. O primeiro elemento (⇑w1) é executado para manter todas as células em 1 como estado inicial, a segunda operação (⇑w0) escrevendo 0 em ordem ascendente a cada célula. Em relação ao algoritmo, a primeira célula a ser escrita com 0 é a (3,5) denominada célula agressora, duas períodos depois a célula vitima (5,6). Quando se realiza a escrita de 0 na célula agressora, este força a célula (5,6) a mudar de estado também para 0. O BICS da coluna 6 (upset_6 da figura 7.8) indica uma falha quando a célula (5,6) muda de estado involuntariamente.

Na figura 7.8 mostra a detecção de *state coupling fault*. O sinal 35 e 56 representam os lados S de cada célula, os sinais bit_b junto a eles representam o lado inverso SB das células.

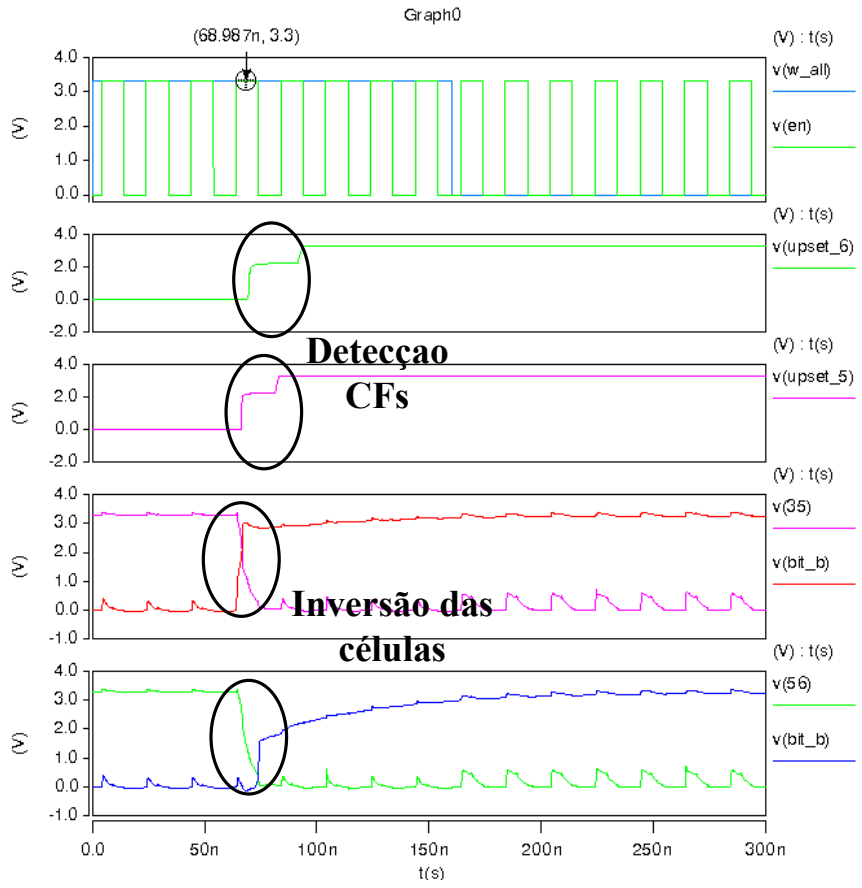

Figura 7.8 Detecção de *coupling fault* no bloco de memória SRAM

A detecção de *state coupling fault* em as duas colunas é devido à fuga de corrente do lado S da célula (5.6) para o lado S da célula (3,5). Na célula (5.6) ocorre uma fuga da corrente que provoca uma queda de tensão em Vcc' da coluna 6, como resultado o BICS da coluna 6 indica uma falha. Na célula (3,5) ocorre um incremento de corrente, e neste caso, a tensão incrementa em Gnd' superando o Vth do transistor M6 do BICS da coluna 5. Em conseqüência, indica também um erro.

7.3 Falhas associadas à resistive-open defetcs

Nesta secção veremos as simulações das falhas permanentes associadas a *resistive-open defects* [23]. O modelo *resistive-open defetcs* nos permite a simulação de *Transition Fault* (TF), *Read Destructive Fault* (RDF) e *Deceptive* read *destructive fault*. A metodologia a seguir é igual a modelagem anterior, encontrar experimentalmente um valor resistivo que permita sensibilizar a falha. O modelo adotado utilizando cinco resistores distribuídos no interior da célula. Cada resistor sensibiliza uma falha e uma falha pode ser sensibilizada por mais de um resistor. A secção 4.2.2 detalha a posição dos resistores. Note que as seguinte operações w0, r0, w1, r1 representam escrita de 0, leitura de 0, escrita de 1, leitura de 1 respectivamente.

7.3.1 Simulação de Transition Fault

A modelagem de *transition fault* (TF) é realizada utilizando um resistor R1 ao lado de um dos transistores de acesso M1 ou M6. A localização do resistor R1 depende do tipo de falha TF que se pretende simular, neste caso simulamos um TF de 0 para 1. A figura 7.9 mostra o circuito da célula adotado. O resistor R1 gera uma impedância no lado inverso SB da célula, o qual impede a descarga para BLB. No nó SB acontece uma distribuição de correntes, a corrente que não sai para BLB vai para Gnd' mediante o transistor M5, o qual conduz porque o nível V_{th} foi superado. Isto só acontece quando a célula contem um 0 e fazemos uma escrita de 1 (w1).

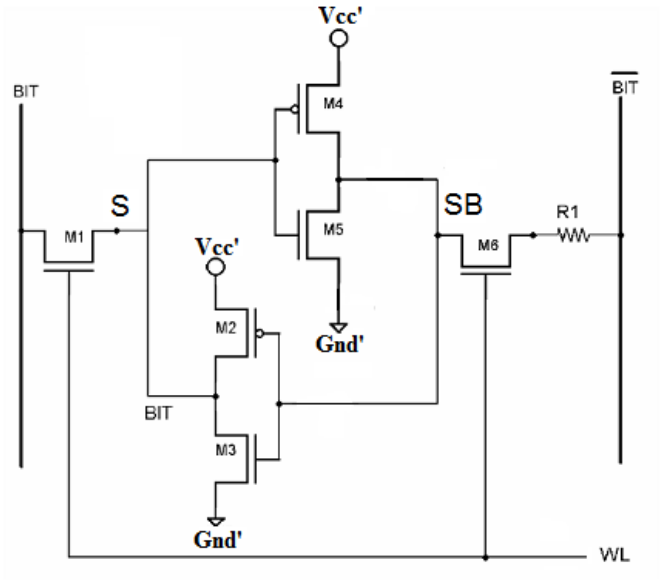

Figura 7.9 Modelagem de TF com R1

O TF foi simulado em três temperaturas, -40ºC, 27ºC e 100ºC. Inicialmente o experimento é realizado sem o BICS para verificar o efeito do resistor R1 sobre a célula, ou seja, para sensibilizar a falha. A figura 7.10 mostra os valores mínimos de R1 relacionado ao consumo de corrente da célula sem BICS e com BICS. Tal como menciona Dilillo em [23] TF pode ser modelado com os resistores R4 e R5 da figura 4.2.

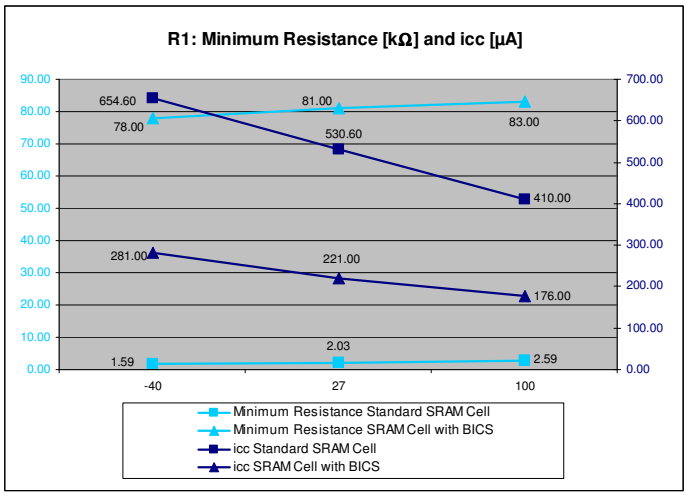

Figura 7.10 Mínima R1 e I_{cc} da célula

A figura 7.11 e 7.12 mostra os resultados obtidos durante a simulação com os resistores R4 e R5 respectivamente. As figuras indicam os valores mínimos dos resistores para causar um TF em uma célula com BICS e sem BICS a distintas temperaturas. As gráficas também contêm informação de consumo de corrente da célula no momento da falha.

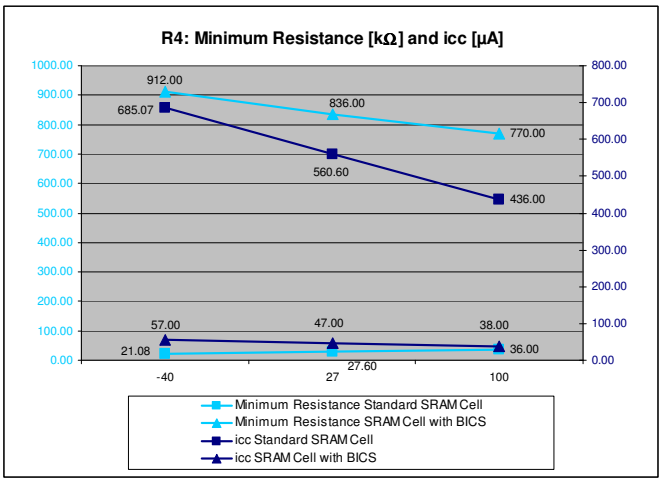

Figura 7.11 Mínima R4 e I_{cc} da célula

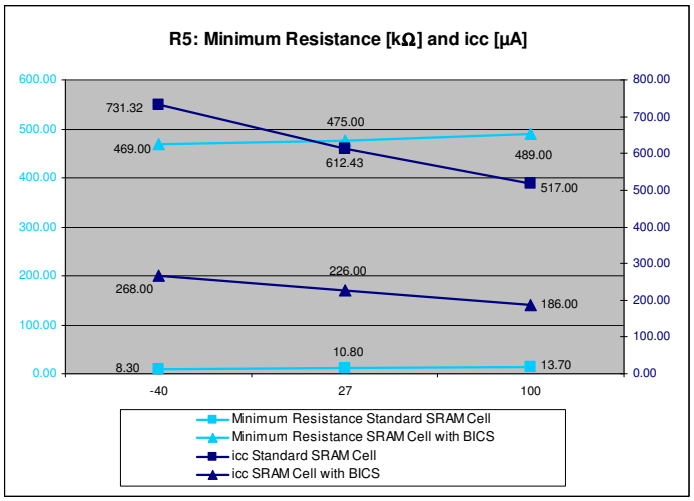

Figura 7.12 Mínima R5 e I_{cc} da célula

Nas três figuras anteriores podemos observar que em geral o valor mínimo dos resistores R1, R4 e R5 são diretamente proporcionais à temperatura. Também é possível entender que para menor temperatura a célula é mais sensível à falhas associadas à *resistive-open defects.*

A figura 7.13 mostra os resultados da simulação de TF com maior detalhe. Detalhes como o conteúdo do lado não inverso S da célula é representado por o sinal (bit), as operações march test, a detaccao do BICS, o sinal do *latch* e a saída do registro de leitura representado como (data_out).

Na figura 7.13 podemos observar o comportamento relacionado a *transition fault* quando realizamos o algoritmo w0, r0, w1, r1. Note que a célula inicia com 1 e logo da w0 a célula já não pode sair de nível 0. O momento da detecção de TF por o BICS é quando realizamos w1 e a célula tenta mudar para 1 sem êxito. Em outras palavras a célula fica com 0 tal como se observa quando acontece r1

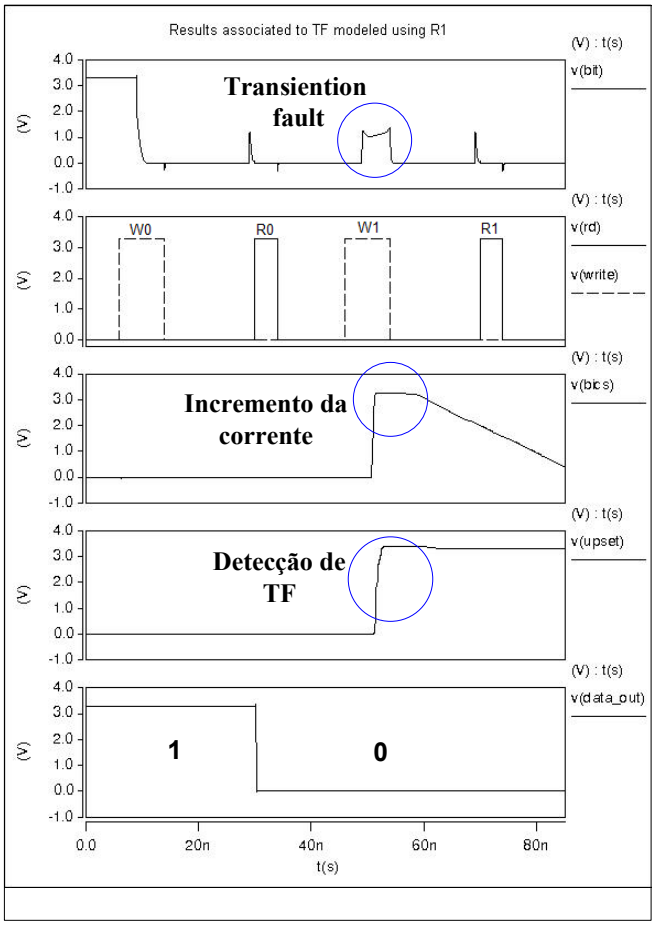

Figura 7.13 Simulação de TF utilizando R1 em 27ºC

7.3.2 Simulação de Read Destructive Fault

Read Destructive Fault (RDF) é uma falha das células que se manifesta mudando o estado da célula quando se realiza uma operação de leitura. A falha pode se apresentar em leituras de 0 ou leituras de 1. Na simulação de RDF utilizamos o resistor R2 [23] localizado na saída do inversor conformado por os transistores M4 e M5, como mostra a figura 7.14.

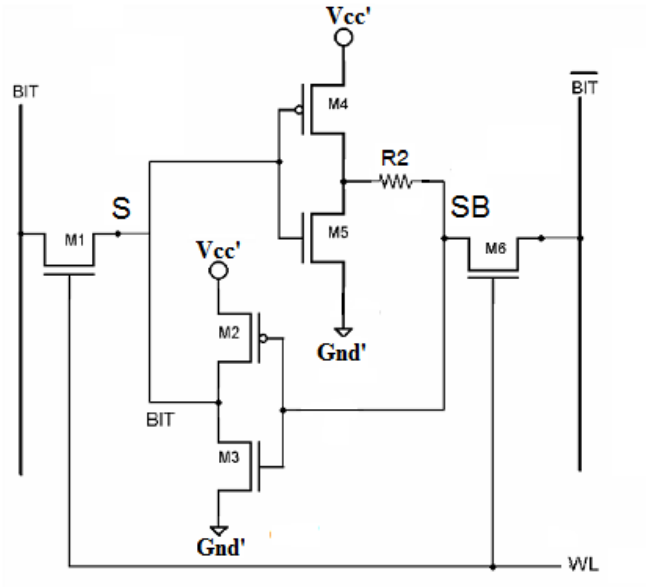

Figura 7.14 Modelagem de RDF com R2

A inserção de o resistor R2 provoca um RDF devido à impedância resultante no nó SB. A simulação de RDF é de 1 para 0 devido à localização de R2. Se assumimos que o lado inverso SB da célula contem 0 e fazemos uma operação de leitura, o resistor R2 impede a descarga de corrente de BLB (nível de pré-carga) para *Gnd'* mediante o transistor M5. Nesse momento M5 é ativo porque o lado S esta em 1. A corrente é desviada para a entrada do inversor conformado por os transistores M2 e M3 da figura 7.14, como o nível da pré-carga é aproximado a V_{cc} resulta que o transistor M3 conduz e a célula muda de estado a causa de uma leitura. É importante indicar que o resistor R2 também pode provocar *Deceptive Read Destructive Fault* (DRDF) [23]. Os resultados as simulações de RDF utilizando R2 é mostrado na figura 7.15.

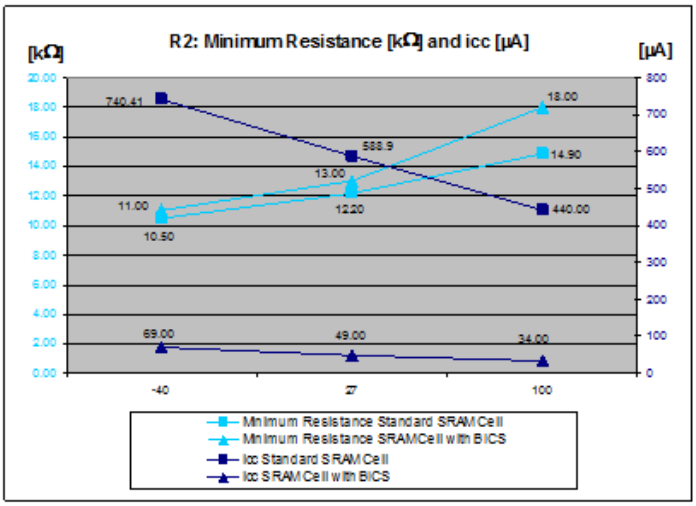

Figura 7.15 Mínima R2 e I_{cc} da célula

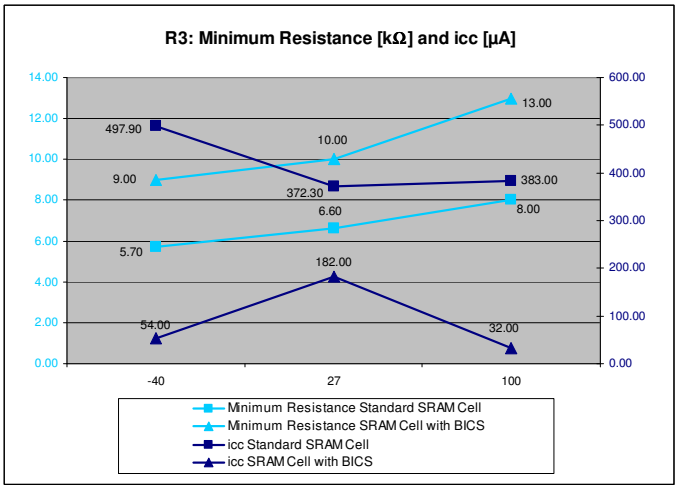

Figura 7.16 Mínima R3 e I_{cc} da célula

Também, RDF pode ser modelado com o resistor R3 tal como é mostrado na figura 7.16. As simulações do valor mínimo de R2 e R3 também foram a três temperaturas, -40ºC, 27ºC, 100ºC. O comportamento do valor mínimo de R2 e R3 com respeito à temperatura é diretamente proporcional o qual implica menor consumo de corrente no momento da detecção. Na figura 7.17 podemos observar a detecção de RDF quando realizamos r1 como parte do algoritmo de detecção. A célula inicia contendo 1, logo realizamos o algoritmo proposto w0, r0, w1, r1. Com a última leitura r1 acontece a falha e o BICS indica um erro.

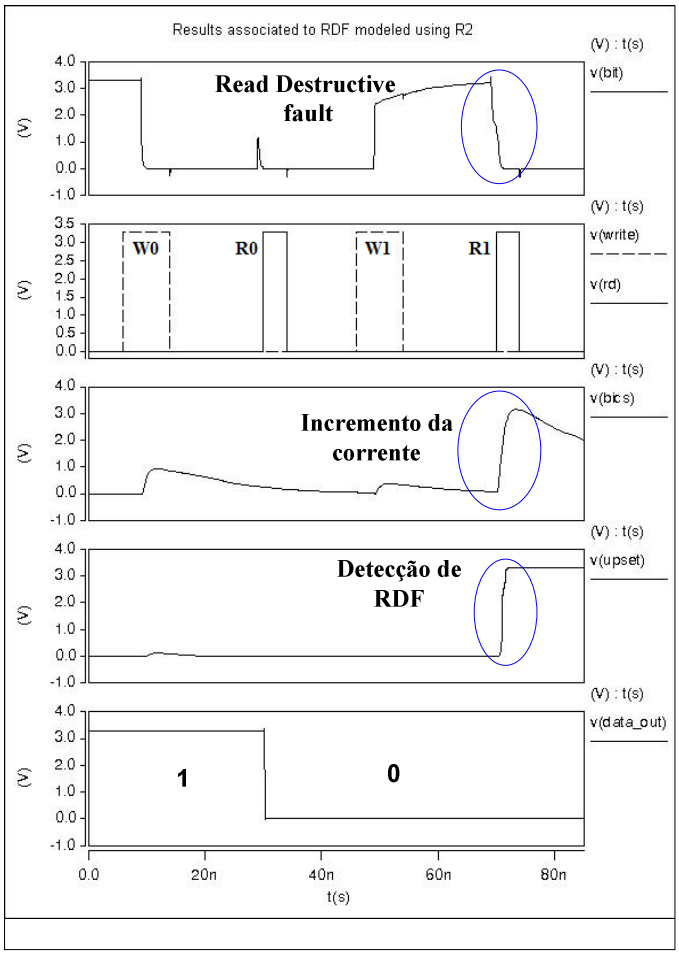

Figura 7.17 Simulação de RDF com R2 em 27ºC

7.3.3 Simulação de Deceptive Read Destructive Fault

A simulação de *Deceptive Read Destructive Fault* (DRDF) também pode se realizar com os resistores R2 e R3, lembremos que a diferencia entre RDF e DRDF é o valor da saída da célula. No caso de DRDF o conteúdo da célula é mascarado pela saída. Para modelar a falha DRDF foi necessário reduzir o tempo de leitura em 80%, resultando em 1nseg. A figura 7.18 mostra os sinais obtidos durante as simulações de DRDF. Podemos verificar a capacidade de detecção do BICS durante a leitura R1. Durante o acesso à célula o conteúdo muda, mas o valor de saída (data_out) ainda é correto.

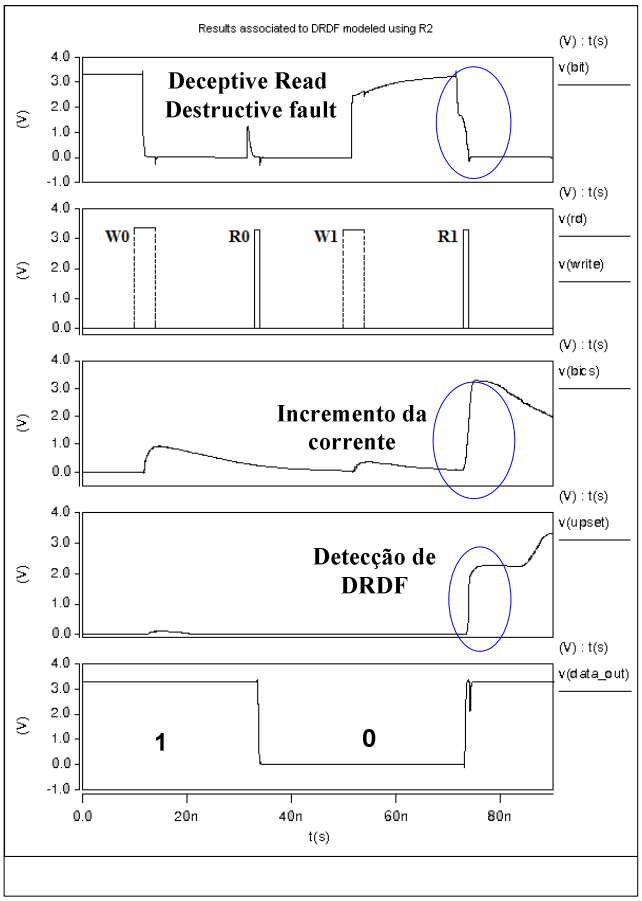

Figura 7.18 Simulação de DRDF com R2 em 27ºC

7.3.4 Incorrect Read Fault

A respeito de *Incorrect Read Fault* (IRF), o BICS não é capaz de detectar este o tipo de falha. Segundo o modelo de falhas associadas de *resistive-open defetcs,* utilizando o resistor R4 da figura 4.2 pode se modelar IRF. Mas o resistor R4 localizado no *gate* dos transistores de acesso deixa a célula isolada. Pode se entender que a falha não gera uma inversão na célula e por tanto não tem um consumo anormal de corrente. O BICS não é capaz de detectar falhas que não provoquem uma corrente adicional nas linhas de alimentação virtual (*Vcc'* e *Gnd'*). Não entanto, a nossa metodologia é ainda capaz de realizar a detecção através do algoritmo visto no capítulo 6. O fato de fazer uma escrita de um dado x antes de fazer a leitura (wx, rx) nos antecipa a conhecer o

resultado da leitura. Se fazemos uma escrita de 0 (w0) nós esperamos uma leitura com resultado 0 (r0). Se o resultado é 1, a célula tem IRF e o algoritmo indica um erro.

7.4 Degradação da memória devido ao BICS

A inserção do BICS no bloco de memória introduz um *delay* na operação da memória. Resultados experimentais indicam que a operação mais afetada é no processo de leitura. A figura 7.19 mostra o *delay* cerca de 4% na operação de leitura. O *delay* é causado por a inserção de transistores no caminho da corrente que alimenta a célula. Neste caso os transistores relacionados são M1 e M2 do BICS (figura 4.2), porque eles têm maior impedância. Os transistores M1 e M2 causam o *delay*, mas também são os principais responsáveis pela direção das falhas. Como resultado o nível de corrente da célula é menor e a degradação da velocidade não é considerável.

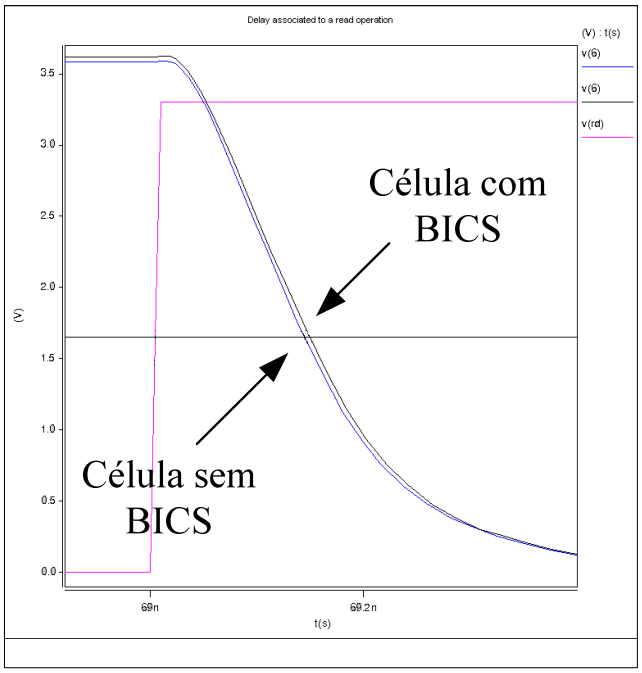

Figura 7.19 *Delay* devido ao BICS em uma leitura

7.5 Conclusões

Dos valores de resistores mínimos encontrados nas figuras 7.10, 7.11, 7.12, 7.15 e 7.16, resistores que provocam falha associadas a *resistive-open defects*. Podemos observar que a tendência quanto à temperatura é igual para todos. O valor mínimo de resistor é diretamente proporcional à temperatura. Isso significa que a célula SRAM é mais sensível quando é submetida a baixas temperaturas. Também podemos observar que o valor do resistor necessário para modelar as falhas em uma célula com BICS é muito maior que uma célula sem BICS. Isso significa que a inserção do BICS incrementa a tolerância às variações de processo.

Por outro lado, os resultados experimentais em 350nm demonstram que a metodologia proposta neste trabalho de dissertação é capaz de detectar falhas funcionais como SAF, CFs e falhas permanentes associadas a *resistive-open defects* como TF, RDF e DRDF.

8 Experimentos em 65nm

Neste capítulo veremos o alcance da metodologia proposta utilizando tecnologia em 65nm de STMicroelectronics. Lembremos que as simulações em os capítulos anteriores foram realizadas em 350nm. As simulações são realizadas adotando o modelo de falhas associadas a *resistive-open defects.* O circuito adotado é mostrado na figura 7.1 descrito também em HSPICE. As simulações elétricas foram realizadas com os seguintes parâmetros:

Tecnologia de CI: 65nm CMOS.

Process corner: típica,

Voltagem de fonte: 1.1 volt,

Temperatura: -40ºC, 27ºC e 100ºC,

Durante a primeira fase dos experimentos, se realiza a simulação de operações de leitura e escrita na célula sem nenhum tipo de defeito, com o objetivo de estandardizar as correntes em cada operação. Na tabela 8.1 podem ser observados os valores de corrente de cada operação nas três temperaturas e pode-se concluir que as correntes de operação associada à célula SRAM são fortemente dependentes da temperatura.

Operation	Temperature -40ºC	Temperature 27ºC	Temperature 100ºC
	I_{cc} [µA]	I_{cc} [µA]	I_{cc} [µA]
SRAM without resistive-open defects			
0→W1	28	26	26
R1	3	3	4
1→W0	28	23	26
R0	5	4	4

Tabela 8.1Medidas da célula sem defeito

Segundo os modelos de falhas associados à *resistive-open defects* utilizando os resistores R1 e R2 da figura 4.2 podemos modelar *Transition Fault* (TF), *Read Destructive Fault* (RDF) e *Deceptive Read Destructive Fault* (DRDF). Os valores mínimos dos resistores R1 e R2 são mostrados na figura 8.1. as simulações mostram os resultados em uma célula com BICS e sem BICS. Lembremos que o resistor R1 modela o TF e o resistor R2 modela RDF e DRDF.

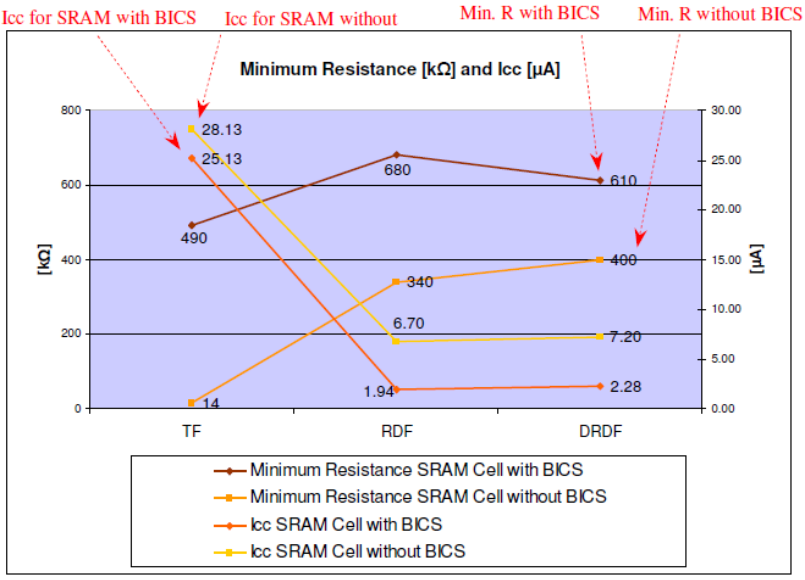

Figura 8.1 Resistores mínimos e I_{cc} para TF, RDF, DRDF

Em concordância com a metodologia proposta no capítulo 6, as simulações em tecnologia de 65nm, as células com defeitos são excitadas e testadas com o algoritmo otimizado *March Test* de 3 elementos e complexidade de 5n.

Na simulação de *Transition Fault* foi utilizando R1= 490kΩ e em 27ºC. o resultado da simulação pode-se observar na figura 8.2. Nesta simulação o BICS detecta antes da ocorrência da falha, devido a que a operação de escrita w0 faz com que a corrente seja maior e gera uma queda de tensão na linha V_{cc}' que ativa o BICS.

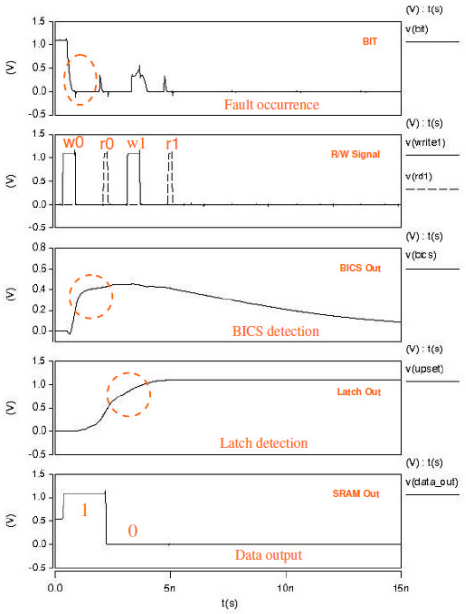

Figura 8.2 Simulação de TF com R1 em 27ºC

Respeito a detecção de *Deceptive Read Destructive Fault* (DRDF) a simulação foi com R2 = 610kΩ na temperatura de 27ºC. A figura 8.3 mostra os resultados obtidos no momento da detecção. Neste caso, a detecção se realiza de maneira similar a os resultados obtidos em tecnologia de 350nm.

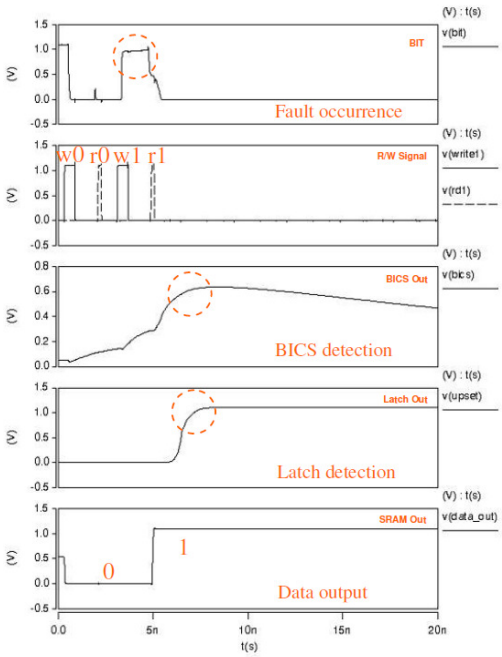

Figura 8.3 Simulação de DRDF com R2 em 27ºC

Na figura 8.4 mostra os resultado das simulações respeito a *Read Destructive Fault* (RDF). Neste caso o valor do resistor R2 é 680kΩ. Lembremos que o R2 é capaz de modelar RDF e DRDF, com diferencia do valor. A detecção de RDF se realiza de maneira similar a os resultados obtidos em tecnologia de 350nm e em 27ºC.

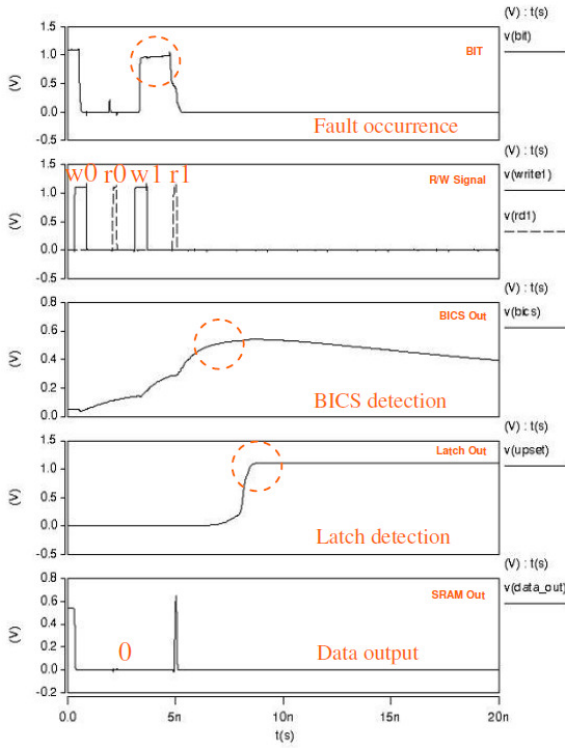

Figura 8.4 Simulação de RDF com R2 em 27ºC

8.1 Conclusões

De acordo com os experimentos realizados em tecnologia de 65nm, podemos dizer que é necessário realizar ainda mais simulações para validar a metodologia proposta, devido a que a detecção das falhas não é coberta para temperaturas superiores a 27ºC.

9 Conclusões

Os resultados experimentais demonstram que a solução proposta neste trabalho baseada na utilização do BICS em conjunto com um algoritmo otimizado representa uma solução extremamente interessante. As simulações elétricas apresentadas no capítulo 7 demonstram a validade e a eficácia da metodologia proposta, em a detecção de falhas funcionais e de falhas associadas a *resistive-open defects.*

A solução proposta do BICS em conjunto com elementos de *March Test*, reduz em 50% de tempo requerido para realizar o teste de memória SRAM comparado com um modificado March C- apresentado em [23]. Isso é possível porque o algoritmo proposto neste trabalho, visto no capitulo 6, é conformado por um número reduzido de elementos *March*.

A inserção do BICS no bloco de memória SRAM, incrementa a tolerância nos processos de variação, o que resulta em um processo melhorado. Isso é devido a que o valor mínimo de resistência associada a *resistive-open defects* para uma memória com BICS é mais alta comparado a uma memória sem BICS, considerando que a falha tem o mesmo comportamento.

Em relação ao tempo de acesso, a inserção do BICS degrada a operação de leitura em só 4%, a operação de escritura não é afeitada com o uso do BICS. A degradação no tempo de escrita devido ao BICS é considerada não excessiva.

Respeito dos experimentos em 65nm, podemos concluir que é preciso realizar mais simulações para chegar a um resultado mais claro. É necessário realizar simulações de maiores temperaturas e simulações a varias freqüências.

10 Referências

[1] Intel C. http://www.intel.com/cd/corporate/pressroom/EMEA/SPA/221594, último acesso 11/03/2010.

[2] L. Dilillo, P. Girard, S. Pravossoudovitch, A. Virazel, "Efficient March Test Procedure for Dynamic Read Destructive Fault Detection in SRAM Memories", Journal of electronic testing: theory and applications 21, pp. 551-561, Springer Science 2005.

[3] J. Fabula, J. Moore, A. Ware, "Understanding neutron single-event phenomena in FPGAs", Designing for Radiation Tolerance, OpenSystem Publishing 2007.

[4] F. Vargas, M. Nicolaidis, "Seu-Tolerance Sram Desing Based On Current Monitoring", 24th International Symposium on Fault-Tolerance Computing, FTCS-24 Austin, USA, Jun 1994.

[5] A.J. Vand de Goor, "Using March Tests to test SRAMs", IEEE Design & Test of Computers. 1993.

[6] S. Hamdioui, A.J. Vand de Goor, M. Rodgers, "March SS: A Tests for All Static Simple RAM Faults", IEEE International Workshop on Memory Technology, Design and Testing, pp: 95-100, 2002.

[7] A. Benso, A. Bosio, S. Di Carlo, G. Di Natale, P. Prinetto, "A Unique March Test Algorithm for Wide Spread of Realistic Memory Faults in SRAMs", Design and Diagnostics of Electronic Circuits and Systems, 2006.

[8] Z. Al-Ars, A.J. Van de Goor, "Static and Dynamic Behavior of Memory Cell Array Spot Defects in Embedded DRAMs", 2003 IEEE Transactions on Computers.

[9] A. J. van de Goor, B. Smit, "Automatic Verification of March Tests", IEEE International Workshop on Memory Testing, pp: 131-136, 1993.

[10] A. Pavlov, M. Sachdev. CMOS SRAM Circuit Design and Parametric Test in Nano-Scaled Technologies. Springer, ISNB 978-1-4020-8362-4, 2008.

[11] Credence Technologies, *Measure of ESD Events*, 1999.

[12] D. Pradhan, Fault-Tolerant Computer System Design, Prentice Hall, 1996.

[13] M. L. Moraes, *Validação De Uma Técnica Para O Aumento Da Robustez De Soc's A Flutuações De Tensão No Barramento De Alimentação*. Faculdade de Engenharia, Programa de Pós-Graduação em Engenharia Elétrica. Porto Alegre: Pontifícia Universidade Católica do Rio Grande do Sul - PUCRS, 2008. Dissertação de Mestrado.

[14] L. F. Cristofoli, *Validação De Uma Técnica Para O Aumento Da Robustez De Soc's A Flutuações De Tensão No Barramento De Alimentação*. Faculdade de Engenharia, Programa de Pós-Graduação em Engenharia Elétrica. Porto Alegre : Pontifícia Universidade Católica do Rio Grande do Sul - PUCRS, 2008. Dissertação de Mestrado.

[15] A. Benso, A. Bosio, S. Di Carlo, G. Di Natale, P. Prinetto, "March AB, March AB1: New March Tests for Unlinked Dynamic Memory Faults". International Test Conference, pp: 841, 2005.

[16] A. Bosio, G. Di Natale, "March Test BDN: A new March Test for Dynamic Faults", IEEE International Conference on Automation, Quality and Testing, Robotics, pp: 85-89, 2008.

[17] A. Avižienis, J. C. Laprie, B. Randell, C. Landwehr, "Basic Concepts and Taxonomy of Dependable and Secure Computing", IEEE Transactions On Dependable and Secure Computing, Vol. 1, no. 1, pp: 11-33, January 2004.

[18] H. Noguchi, Y. Iguchi, H. Fujiwara, S. Okumura, Y. Morita, K. Nii, H. Kawaguchi, M. Yoshimoto. "A 10T Non-precharge Two-Port SRAM Reducing Readout Power for Video Processing", The Institute of Electronics, Information and Communication Engineers, April 2008.

[19] J. Yang, L. Chen, "A New Loadless 4-Transistor SRAM Cell with a 0.18 μm CMOS Technology", Electrical and Computer Engineering, Canadian Conference 2007.

[20] K. Noda. "A Loadless CMOS Four-Transistor SRAM Cell in a 0.18μm Logic Technology", IEEE Transactions on Electron Devices, Vol. 48, no. 12, December 2001.

[21] J. Rabaey, A. Chandrakasan, B. Nicoloc. *Digital Integrated Circuits: A Design Prospective*, Second Edition, Prentice Hall. 2003.

[22] Electronics Engineering Herald, *SRAM memory interface to microcontroller in embedded systems*, http://www.eeherald.com/section/design-guide/esmod15.html, ultimo acesso 01/03/2010.

[23] L. Dilillo, P. Girard, S. Pravossoudovitch, A. Virazel, S. Borri, M. Hage-Hassan, "Resistive-Open Defects in Embedded-SRAM core cells: Analysis and March Test Solution", IEEE 13th Asian Test Symposium, 2004.

[24] R. Chipana, L. Bolzani, F. Vargas, "Bics-Based March Test for Resistive-Open defect Detection in SRAMs", 11th IEEE Latin-American Test Workshop, 2010.

[25] T. Calin, F. Vargas, M. Nicolaidis, R. Velazco, "A Low-Cost, Highly Reliable SEU-Tolerant SRAM: Prototype and Test Results", IEEE Transactions on Nuclear Science, Vol. 42, no. 6, December 1995".

[26] C. Argyrides, F. Vargas, M. Moraes, D. Pradhan, "Embedding Current Monitoring in H-Tree RAM Architecture for Multiple SEU Tolerance and Reliability Improvement", 14th IEEE International On-Line Testing Symposium, 2008.

[27] M. Nicolaidis, F. Vargas, B. Courtois, "Design of Built-In Current Sensors for Concurrent Checking in Radiation Environments", IEEE Transactions on Nuclear Science, December 1993.

[28] B. Gill, M. Nicolaidis, F. Wolff, C. Papachristou, S. Garverick, "An efficient BICS design for SEUs detection and correction in semiconductor memories". Design,

Automation and Test in Europe Conference and Exhibition, Vol 1, pp. 592-597, 2005.

[29] HSPICE Simulation and Analisys User Guide. Synopsys, In. Release U-2003.03-PA, March 2003.

[30] HSPICE Quick Reference Guide. Synopsys, Inc. Release W-2004.09.

[31] CosmosScope Reference Manual. Synopsys, Inc. Version Z-2007.03, March 2007.

[32] R. Dekker, F. Beenker, L. Thijssen. "A Realistic Fault Model and Test Algorithms for Static Random Access Memory". IEEE Transaction on Computer-Aided Design, Vol 9, pp: 567-572, June 1990.

[33] J. Li, T. Tseng, C. Wey. "An Efficient Transparent Test Scheme for embedded Word-Oriented Memories". Design, Automation and Test in Europe Conference and Exhibition, Vol. 1, pp: 574-579, 2005.

[34] L. Dilillo, P. Girard, S. Pravossoudovitch, A. Virazel, S. Borri. "March iC-: An Improved Version of March C- for ADOFs Detection", 22nd IEEE VLSI Test Symposium, pp.129-134, April 2004.

[35] W. Wang, K. Lee, J. Wang, "An On-Chip March Pattern Generator for Testing Embedded Memory Cores", Very Large Scale Integration (VLSI) Systems, IEEE transactions on, Vol. 9, no. 5, pp: 730-735, October 2001.

[36] A. Benso, S. Di Carlo, G. Di Natale, P. Prinetto, "An Optimal Algorithm for the Automatic Generation of March Tests", Design, Automation and Test in Europe Conference and Exhibition, pp: 938-943, 2002.

[37] T. Calin, F. L. Vargas, M. Nicolaidis, "Upset-tolerant CMOS SRAM using current monitoring: prototype and test experiments", IEEE International Test Conference, pp. 45-53, 1995.

Anexos

BICS-Based March Test for Resistive-Open Defect Detection in SRAMs

R. Chipana, L. Bolzani, F. Vargas
Electrical Engineering Dept.
Catholic University – PUCRS
Porto Alegre, Brazil
vargas@computer.org

***Abstract* — Nowadays, embedded Static Random Memories (SRAMs) can occupy a significant portion of the chip area and contain hundreds of millions of transistors. Due to technology scaling, functional fault models traditionally applied in SRAMs' testing have become insufficient to correctly reproduce the effects caused by some defects generated during the manufacturing process. In this paper, we investigate the possibility to use Built-In Current Sensors (BICSs) in combination with an optimized March algorithm to detect static faults associated to resistive-open defects. Experimental results obtained throughout electrical simulations validate the proposed technique demonstrating its viability and effectiveness.**

Keywords: SRAM; Resistive-Open Defects; BICS.

I. INTRODUCTION

Advances in Very Deep Sub-Micron (VDSM) technology have made possible the integration of millions of transistors into a small area, allowing the increase of circuits' density. However, the rapidly increasing need to store more information results in the fact that the Static Random Access Memory (SRAM) can occupy great part of the System-on-Chip (SoC) silicon area. This is confirmed by the SIA Roadmap which forecasts a memory density approaching 94% of the SoC area in about 10 years [1]. Consequently, memory has become the main responsible of the overall SoC area. Moreover, technology scaling has led to the development of new fault models which differ from the traditional functional ones presented in [2], such as stuck-at, transition and coupling faults. In detail, the functional fault models have become insufficient to model the effects produced by some specific defects that can be generated during the manufacturing process [5].

Nowadays, resistive-open defects have become one of the most significant problems in VDSM technologies due to the presence of many interconnection layers and an ever growing number of connections between each layer [5]. The importance of resistive-open defects is analyzed and pointed out as the most common cause of test escapes in deep-submicron technology in [4]. In general terms, a resistive-open defect is defined as a defect resistor between two circuit nodes that should be connected [3]. As described in [3], resistive-open defects are voltage and temperature dependent, which means the test results are different according to the applied values of *Vdd* and to the test temperature, respectively. Moreover, this type of defect causes timing dependent defect behavior, which means the test results may depend on the test timing. In other words, resistive-open defects generally cause timing-dependent faults. As a consequence, a two-pattern sequence is usually necessary to sensitize the fault.

In this context, the development of new test solutions able to provide detection of specific fault models, such as the ones associated to resistive-open defects, has become increasingly essential. According to [5], resistive-open defects in embedded-SRAMs can be modeled according to the following fault models: (1) Transition Fault, (2) Read Destructive Fault, (3) dynamic Read Destructive Fault, (4) Deceptive Read Destructive Fault and (5) Incorrect Read Fault. In [5] a modified March C- algorithm, able to exhaustively detect all faults induced by resistive-open defects, has been presented.

In this paper, we propose to investigate the possibility to adopt Built-In Current Sensors (BICSs) in combination with an optimized algorithm based on March elements to detect static faults associated to resistive-open defects. BICSs have been presented in the past as very interesting solutions in different research issues [8-13]. However, it is important to highlight that the novelty of the proposed technique is related to the use of BICSs to detect permanent faults associated to resistive-open defects generated during SRAM manufacturing.

The evaluation of the proposed approach has been divided in three distinct phases. During the first one, we modeled the static faults associated to resistive-open defects. In detail, we inserted the resistors necessary to model each specific static fault in an SRAM with and without BICS in order to find the minimum resistance necessary to generate the respective faulty behavior. In a second phase, we verified the BICS' fault detection capability. Finally, we developed the optimized March algorithm able to assure the detection of the target faults. The experimental results demonstrate that the BICSs adoption represents a very interesting solution, since it significantly reduces the time required to perform the SRAM testing in comparison with the technique presented in [5]. This can be attributed to the fact that the BICS insertion reduces the number of March elements present in the algorithm. Indeed, the electrical simulations demonstrate the effectiveness of the proposed solution in detecting static faults associated to resistive-open defects.

It is important to note that the experimental results presented in this paper have been obtained using a 350 nm technology. However, the idea was to initially investigate the

possibility to use BICSs to detect resistive-open defects therefore providing valid preliminary results. Moreover, we would like to point out that at the moment we are migrating all the experiments shown in this paper to a 65 nm technology library.

This paper has been organized as follows: In Section II, we detail the fault models adopted and describe the basic structure of the BICS. Section III summarizes the experimental results. Finally, in Section IV we draw the conclusions and point out future works.

II. BACKGROUND

The technique proposed in this paper aims to provide the detection of permanent faults produced during the SRAMs' manufacturing process. In this section we will describe the fault model adopted as well as the BICS's structure.

A. Fault Model Adopted

The standard six-transistor CMOS SRAM cell is composed of four transistors that form two cross-coupled CMOS inverters and two NMOS transistors that provide read and write access to the cell. In this paper we address the detection of static faults associated to resistive-open defects classified in the following fault models according to [5]:

- *Transition Fault* (TF): A cell is said to have a TF, if it fails to undergo a transition (from 0 to 1 or from 1 to 0) when it is written.
- *Read Destructive Fault* (RDF): A cell is said to have an RDF, if a read operation performed on the cell changes the data in the cell and returns an incorrect value to the output.
- *Deceptive Read Destructive* Fault (DRDF): A cell is said to have a DRDF, if a read operation performed on the cell returns the correct logic value, but changes the contents of the cell.
- *Incorrect Read Fault* (IRF): A cell is said to have an IRF, if a read operation performed on the cell returns an incorrect logic value, and the correct value is still stored in the cell.

Fig. 1 shows the scheme of a standard six-transistor SRAM cell, where five different resistive-open defects have been injected to model the static faults previously described.

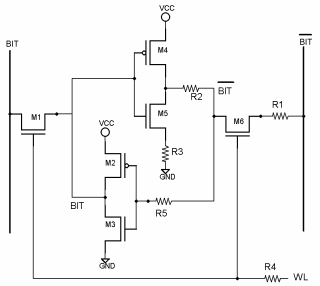

Figure 1. Resistive-open defects injected in a 6-transistor SRAM cell

In detail, the resistors R1, R2, R3, R4 and R5 model respectively the following fault models: TF, RDF/DRDF, RDF/DRDF, IRF/FT and TF. Due to the symmetry of the structure, these five locations allow an exhaustive analysis of the targeted faults within the cell structure.

B. Built-In Current Sensors for SRAMs

In the past, BICSs have been proposed for Built-In Current (BIC) testing of static CMOS circuits [11][12]. In general terms, BIC testing involves the monitoring of power bus currents in a VLSI circuit in order to detect malfunction-causing defects. Thus, BICSs detect the vast majority of defects that can occur during manufacturing by monitoring abnormal quiescent currents. In other words, BICSs have been proposed to detect physical defects during the circuits' production in order to identify and facilitate rejection of defective parts. Later, the use of BICSs has been proposed to provide on-line concurrent detection of radiation-induced leakage currents in circuits [8]. In more details, each BICS monitors the power-bus static current of these circuits to detect excessive current consumption. This excessive current consumption is compared to a predefined reference value in order to detect radiation-induced multiple parametric failures and system power supply breakdowns. Moreover, BICSs have been presented as a very interesting solution to provide transient fault detection for SRAMs in radiation-exposed environments [9][10]. The technique presented in [9][10] is based on the idea to monitor the SRAM power-bus by using BICS circuits in order to detect abnormal current dissipation. This abnormal current is the result of a Single-Event Upset (SEU) in the memory and is generated during the inversion of the state of the memory cell being upset. Thus, the current checking is performed on the SRAM columns and it is combined with a single-parity bit per RAM word to perform error correction. Fig. 2 shows the current sensor's schematic composed of a sensing cell followed by an asynchronous latch.

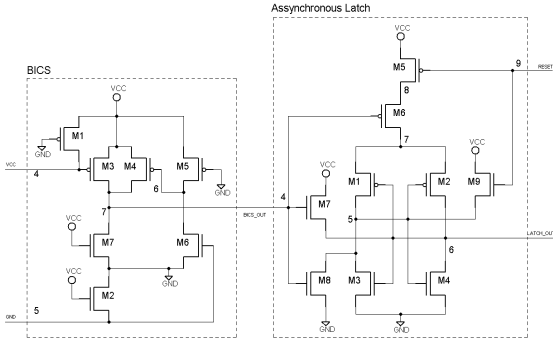

Figure 2. Built-In Current Sensor (BICS)

Finally, Fig. 3 depicts the general structure of the current-monitored memory cell array presented in [9][10]. In detail, BICSs are placed on the memory's vertical power lines in order to indicate the bit position on which a failure has occurred. Thus, when a SEU occurs in the corresponding memory column, an internal latch is set. A parity check makes the localization of the affected word possible and thus allows error correction.

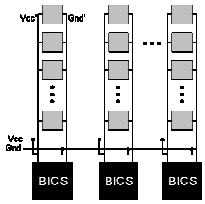

Figure 3. Memory cell array with BICS

Finally, it is important to note that BICSs have been recently proposed to improve the reliability of H-tree RAM memories [13].

III. EXPERIMENTAL RESULTS

In this section, we present the results obtained during electrical simulations performed with Synopsys HSPICE. To do so, we modeled the resistive-open defects following the scheme presented in Fig. 1 in two different versions of SRAM circuits defined as follow: (1) Unhardened SRAM: without BICS and (2) Hardened SRAM: with BICS. The goal of these simulations was twofold: (a) understand how the BICS insertion affects the cell behavior, and (b) investigate the BICS fault detection capability with respect to the injected defects. To conclude this section, we developed an optimized algorithm based on March elements.

A simplified memory circuit composed of the following structures has been adopted in order to reduce the simulation time: (1) one six-transistor SRAM cell, (2) sense amplifier, (3) write driver and (4) address decoder. Fig. 4 depicts the memory circuit adopted during the electrical simulations.

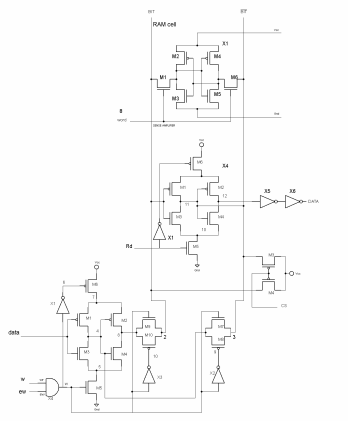

Figure 4. SRAM memory circuit adopted

Additionally, the electrical simulations have been performed adopting the following parameters:

- Process corner: typical
- Supply voltage: 3.3V
- Temperature: -40ºC, 27ºC and 100ºC.

A. *Fault Modeling*

The static faults associated to resistive-open defects have been modeled using according to the scheme presented in Fig. 1. The fault modeling has been divided in two different parts. During the first one, we injected the resistive-open defects in an unhardened SRAM cell. Finally, we performed electrical simulations in a hardened memory containing BICS. To facilitate the analysis of the fault modeling with respect to the two different versions of the memory circuit, figures 5 to 10 show the relation between the minimum resistance values needed to model the effects produced by the resistive-open defects represented by R1, R2, R3, R4 and R5 and the I_{cc} associated to each resistor respectively, considering the three different targeted temperatures. Note that *w0*, *w1*, *r0* and *r1* represent respectively the following operations: write 0, write 1, read 0 and read 1.

Fig. 5 shows the minimum resistance for modeling TF, from 0 to 1, using R1 as well as the I_{cc} related to the respective faulty behavior according to the temperature adopted during simulation. The obtained results demonstrate that the resistance value is directly proportional to the temperature, which means that with increasing temperature the resistance value necessary to generate the faulty behavior has to be higher. As consequence, the SRAM cell is more sensitive when operating in low temperatures. Moreover, it is possible to observe that the resistance value necessary to model the fault in the hardened version of the memory circuit is higher than in the unhardened one.

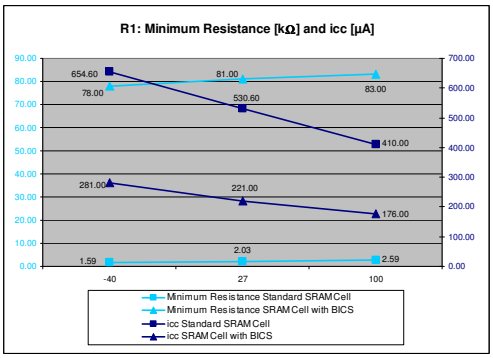

Figure 5. Minimum resistance and *Icc* value for R1

Fig. 6 shows the results obtained using R2 to model the faulty behavior associated to RDF or DRDF from 1 to 0.

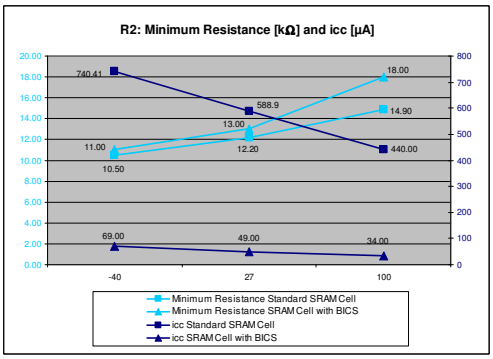

Figure 6. Minimum resistance and *Icc* value for R2

In other words, the fault is observed when the cell stores the value 1 and we perform a read operation. Thus, the cell content

will switch from 1 to 0 while performing the read, when considering RDF and after performing the read when considering DRDF. It is important to highlight that according to [5], the defect induced by R2 always generates a RDF and in certain cases a DRDF. In Fig. 7, R3 models the same faults as associated to R2.

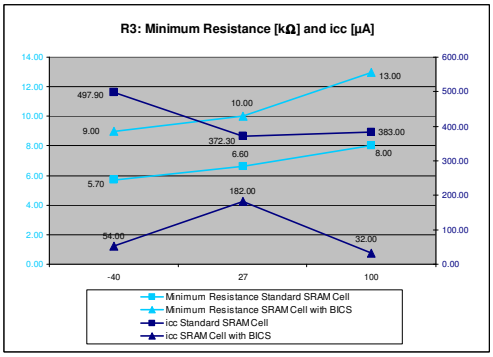

Figure 7. Minimum resistance and *Icc* value for R3

Fig. 8 shows the results obtained when R4 has been applied into the SRAM cell. This resistor models two different faults: TF and IRF. Both faulty behaviors are related to a transition from 0 to 1. Finally, Fig. 9 depicts the results associated to R5, where a TF from 0 to 1 has been modeled.

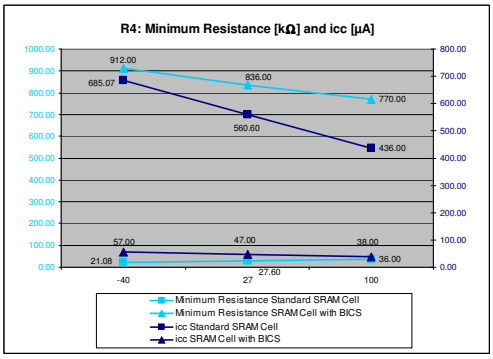

Figure 8. Minimum resistance and *Icc* value for R4

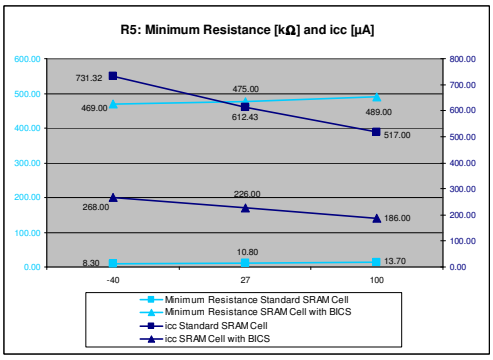

Figure 9. Minimum resistance and *Icc* value for R5

B. *Considerations About How the BICS Insertion Affects the Cell Behavior*

Inserting transistors on the current path (BICS transistors M1 and M2, Fig. 2) could decrease the I_{cc} (as the case for M1) and I_{gnd} (for M2). For faults that result in the inversion of the cell, the limitation of these currents is very positive in the sense that if these currents are limited by the resistances of M1 and M2, a smaller current is generated to charge/discharge the *Bit* and *Bit_b* nodes of the defective memory cell and thus, preventing the cell from inverting. As consequence, the resistance of the fault associated to R1, R2, R3 R4 and R5 must be larger in order to provoke cell misbehavior. Fig. 10 summarizes the resistance values of R1 to R5 and their respective currents (I_{cc}) for the SRAM with and without BICS, at room temperature. As conclusion, the addition of the BICS increases the cell resistance to resistive-open defects that tend to cause bit-flips. From this fact, it can be concluded that the BICS insertion increases the SRAM tolerance to process variations, which in turn results in a process yield improvement.

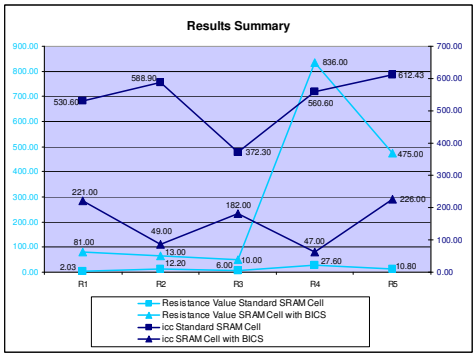

Figure 10. Resistance and *Icc* values related to SRAM with and without BICS.

C. *BICS Fault Detection Capability*

The BICS fault detection capability with respect to each previously mentioned fault model has been investigated at room temperature. Fig. 11 illustrates the simulation related to TF modeled using R1. In detail, Fig. 11 shows the waveforms associated to the *Bit Line*, the type of operation performed, the BICS detection, the latch output and the data output. Observing this figure, it is possible to see the faulty behavior related to a TF. In other words, we can observe that the cell initially stores value 1 and the first operation performed is related to a *w0*. Afterwards, we execute an *r0* followed by a *w1* that sensitizes the TF. Thus, when the cell is storing the value 0 and we try to write 1 into the cell, we can observe that the cell has been prevented from undergoing a transition from 0 to 1, which characterizes a TF behavior. In the sequence, it is possible to observe the moment when the BICS detects the TF and the respective latch output.

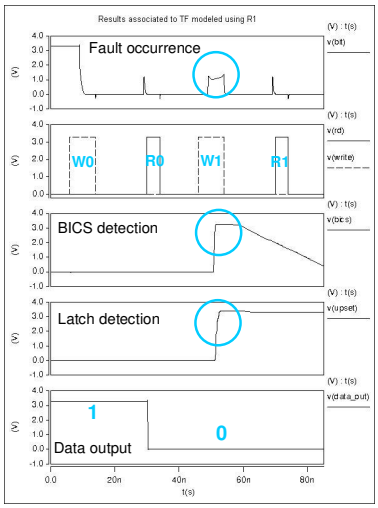

Figure 11. Simulating TF using R1

Fig. 12 shows the waveforms resulted from the simulation associated to RDF using R2. The RDF is observed when we perform a read operation. In detail, the content of the cell is changed from 1 to 0 when we try to read the cell storing the value 1. One should observe that the cell content is changed to 0 before providing the output value.

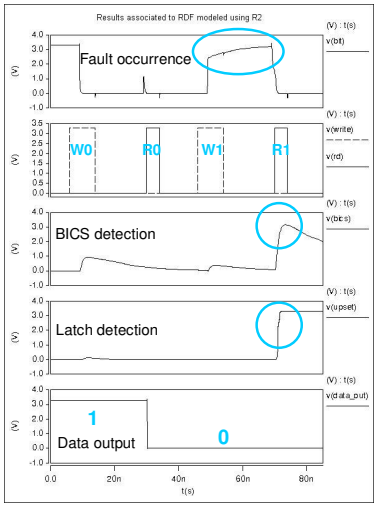

Figure 12. Simulating RDF using R2

Fig. 13 shows the waveforms obtained by simulating a DRDF in the above mentioned configuration. We can verify the BICS fault detection capability with respect to a DRDF during an *r1*. During memory access, the cell content is changed after the read operation, but anyhow the cell output is still correct.

Regarding IRF, the BICS was not able to detect this type of fault. We attribute this inability to the fact that this fault does not produce the inversion of the cell and so, no I_{cc} (resp. I_{gnd}) currents are generated on the virtual power supply lines. Since the BICS is not able to detect faults that do not provoke current variations on the virtual power lines, IRF are uncovered. However, the BICS inability does not compromise the proposed approach because the IRF is completely detected by the algorithm to be presented in Sub-Section *E*.

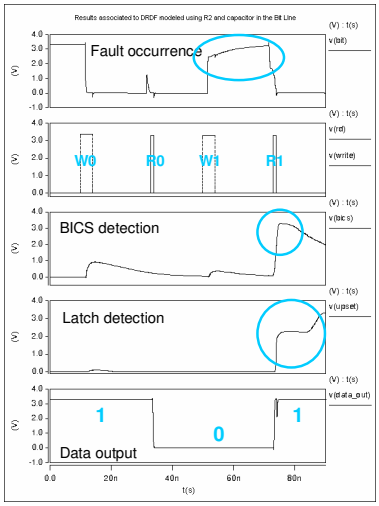

Figure 13. Simulating DRDF using R2 and a capacitor in the Bit Line

D. *Considerations on Memory Speed*

To conclude the experiments, we performed a simulation to estimate the delay associated to the BICS insertion. The obtained result is shown in Fig. 14, where it is possible to observe that the BICS introduces a delay of about 4% during read operations.

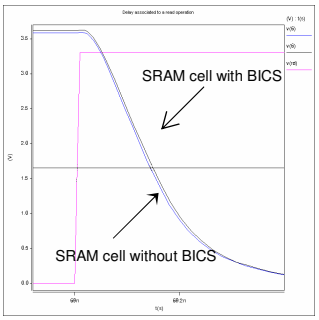

Figure 14. Delay introduced by BICS insertion: SRAM read operations

Generally speaking, inserting transistors on the current path decreases the memory speed. Fortunately, in opposition to the standard BICS applications, where the BICS has to drive the current produced by many gates at a time, in the present application only one cell per column is read or written at a time. Thus, the current level to be driven by the BICS is low and the speed degradation is not excessive.

For read operations both sides of the cell are connected to the pre-charged *Bit Lines*. The voltage level of the *Bit Line* connected to side with 0 logic value will be decreased to a value between V_{cc} and *Gnd*, while the voltage level of the second *Bit Line* is not affected. Thus, the critical current path connects the first *Bit Line* to the *Gnd* and passes through the n-

transistor controlled by the *Word Line*, the *n*-transistor of the cell inverter and the transistor inserted on the path by the BICS (transistor M2, Fig. 2). As observed in Fig. 14, this degradation due to the insertion of M2 is on the order of 4%. Furthermore, this degradation can be eliminated by slightly increasing the width of the two first transistors.

Considering write operations, the critical case occurs when the cell has to change its state. We have to consider two phases for this case. During the first one, the Bit Line imposes the new values (0,1 or 1,0) to the two sides of the cell and brings the cell state close to the new equilibrium state (0,1 or 1,0). During this phase the currents which flow from *Bit Lines* to V_{cc} and *Gnd* resist to the state transition. Thus, increasing the resistance of these paths increases the speed of this phase. During the second phase, the voltage values of the two sides of the cell are close to the new equilibrium state. In this case, the above currents help the cell to reach the new equilibrium state. Thus, the resistance's increase involved by the BICS will decrease the speed of this phase. The two above mentioned phenomena annihilate each other and there is *no significant speed degradation for write operations*. Furthermore, since write operations are faster than read ones, slight speed degradation for write operations are not considered a drawback [9].

E. The Proposed Algorithm

To assure the sensitizing of the targeted faults, we developed an optimized algorithm based on March elements. The obtained results shown in the previous section demonstrate clearly that the use of BICS represents a very effective solution to achieve fault detection of TF, RDF and DRDF. Thus, the optimized algorithm to test SRAMs with BICS is the following:

$$\Updownarrow (w1) \Uparrow (w0,r0) \Uparrow (w1,r1)$$

In more detail, the proposed algorithm is composed of three March elements, each one consisting of a sequence of operations applied to each cell in the memory before proceeding to the next cell. An operation consists of writing a 0 into a cell (*w0*), writing a 1 into a cell (*w1*), reading a cell with expected value 0 (*r0*) and reading a cell with expected value 1 (*r1*). The address to the next cell is determined by the address order that can be an increasing address order, from address 0 to *n-1* denoted by the $\Uparrow$ symbol, a decreasing address order, from *n-1* to 0 denoted by the $\Downarrow$ symbol or irrelevant, denoted by the symbol $\Updownarrow$.

Compared to well known algorithms found in the literature [5], the presented BICS-based algorithm significantly decreases the time required to test the memory, since the number of March elements is dramatically reduced. To illustrate this situation, below it is indicated the algorithm proposed by [5]:

$$\Updownarrow (w0) \Uparrow (r0,w1) \Uparrow (r1,w0) \Downarrow (r0,w1) \Downarrow (r1,w0) \Updownarrow (r0)$$

With respect to [5], the BICS-based optimized algorithm reduces the number of operations from 10 to 5 and consequently reduces the test time by 50%.

IV. FINAL CONSIDERATIONS

This paper presented a preliminary study that investigates the possibility to adopt BICS in combination with an optimized algorithm to detect static faults associated to resistive-open defects in SRAMs. The obtained results demonstrate the benefits from using the proposed approach. First, the BICS insertion increases the SRAM tolerance to process variations, since the effects produced by the same physical defect are distinct from a SRAM with BICS to a SRAM without BICS. In other words, resistive-open defects must present higher resistance values in SRAMs with BICS than in SRAMs without BICS in order to produce the same faulty behavior. Second, the proposed solution assures detection of the target faults associated to resistive-open defects. Third, the solution presented herein decreases by 50% the time required to perform SRAM testing in comparison with the March C- algorithm presented in [5]. This time optimization is possible because the proposed algorithm is composed of a reduced number of March elements. With regards to memory access time, the BICS insertion degrades read operations in the order of 4%, whereas write operations are not affected.

REFERENCES

[1] Semiconductor Industry Association (SIA), "International Technology Roadmap for Semiconductors (ITRS)", 2003 Edition.

[2] A D. J. Van de Goor, Using March Tests to Test SRAMs, IEEE Design & Test of Computers, 1993.

[3] J. C.-Mi li, Chao-Wen Tseng, E. J. McCluskey, "Testing for Resistive Opens and Stuck Opens", IEEE International Test Conference, 2001.

[4] V. Needham et al., "High Volume Microprocessor Test Escapes-Analysis of Defects Our Tests are Missing", IEEE International Test Conference, 1998.

[5] L. Dilillo, P. Girard, S. Pravossoudovitch, A. Virazel, S. Borri, M. Hage-Hassan, "Resistive-Open Defects in Embedded-SRAM core cells: Analysis and March Test Solution", IEEE 13th Asian Test Symposium, 2004.

[6] D. Dilillo, P. Girard, S. Pravossoudovitch, A. Virazel, S. Bom, M. Hage-Hassan, "Dynamic Read Destructive Fault in Embedded-SRAM: Analysis and march Test Solution, IEEE European Test Symposium, 2004.

[7] A. Rubio, J. Figueras, J. Segura, "Quiescent current sensor circuits in digital VLSI CMOS testing", Electronic Letters, Vol. 26, pp. 1204-1206, 1990.

[8] F. Vargas, M. Nicolaidis, B. Courtois, "Quiescent Current Monitoring to Improve the Reliability of Electronic Systems in Space Radiation Environments", IEEE International Conference on Computer Design, pp. 596-600, 1993.

[9] F. Vargas, M. Nicolaidis, "SEU-Tolerant SRAM Design Based on Current Monitoring", 24th International Symposium Fault-Tolerant Computing, FTCS-24, Austin, USA, Jun. 1994.

[10] T. Calin, F. Vargas, M. Nicolaidis, R. Velazco, "A Low-Cost, Highly Reliable SEU-Tolerant SRAM: Prototype and Test Results, IEEE Transactions on Nuclear Science, Vol. 42, N. 6, December 1995". OR T. Calin, F. L. Vargas, M. Nicolaidis, "Upset-tolerant CMOS SRAM using current monitoring: prototype and test experiments", IEEE International Test Conference, pp. 45-53, 1995.

[11] W. Maly, P. Nigh, "Built-In Current Testing – Feasibility and Study", International Conference on Computer-Aided Design, November 1988.

[12] P. Nigh, W. Maly, "Test Generation for Current Testing", IEEE Design & Test, 1990.

[13] C. Argyrides, F. Vargas, M. Moraes, D. Pradhan, "Embedding Current Monitoring in H-Tree RAM Architecture for Multiple SEU Tolerance and Reliability Improvement", 14th IEEE International On-Line Testing Symposium, 2008.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo