UNIVERSIDADE DO VALE DO RIO DOS SINOS - UNISINOS
UNIDADE ACADÊMICA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA
NÍVEL MESTRADO PROFISSIONAL

SAMUEL LESSINGER

PROPOSTA DE UMA PLATAFORMA RECONFIGURÁVEL PARA TESTES DE MÓDULOS SDRAM DDR3

SÃO LEOPOLDO
2017

Samuel Lessinger

Proposta de uma plataforma reconfigurável para testes de módulos SDRAM DDR3

Dissertação apresentada como requisito parcial para obtenção do título de Mestre em Engenharia Elétrica, pelo Programa de Pós-Graduação em Engenharia Elétrica da Universidade do Vale do Rio dos Sinos - UNISINOS.

Orientador: Prof[a]. Dr[a]. Margrit Reni Krug

Coorientador: Prof. Dr. Eduardo Luis Rhod

São Leopoldo

2017

L639p Lessinger, Samuel
Proposta de uma plataforma reconfigurável para testes de módulos SDRAM DDR3 / por Samuel Lessinger. – 2017.
75 f. : il. ; 30 cm.

Dissertação (Mestrado)— Universidade do Vale do Rio dos Sinos, Programa de Pós-graduação em Engenharia Elétrica, São Leopoldo, RS, 2017.

Orientadora: Dra. Margrit Reni Krug.
Co-orientador: Dr. Eduardo Luis Rhod.

1. Teste eletrônico. 2. Memória SDRAM DDR3. 3. Hardware reconfigurável. 4. FPGA. 5. Teste de memória. I. Título.

CDU: 621.3.049.77

Catalogação na Publicação:
Bibliotecário Alessandro Dietrich - CRB 10/2338

Samuel Lessinger

Proposta de uma plataforma reconfigurável para testes de módulos SDRAM DDR3

Dissertação apresentada como requisito parcial para obtenção do título de Mestre em Engenharia Elétrica, pelo Programa de Pós-Graduação em Engenharia Elétrica da Universidade do Vale do Rio dos Sinos - UNISINOS.

Aprovado em 21 de Setembro de 2017.

BANCA EXAMINADORA:

Prof. Dr. Márcio Rosa da Silva – Unisinos
Avaliador

Profª. Drª. Érika Fernandes Cota – UFRGS
Avaliador Externo

Profª. Drª. Margrit Reni Krug (Orientador)
Prof. Dr. Eduardo Luis Rhod (Coorientador)

Visto e permitida a impressão
São Leopoldo

Prof. Dr. Eduardo Luis Rhod
Coordenador PPG em Engenharia Elétrica

A família.

AGRADECIMENTOS

Meus sinceros agradecimentos à todos que contribuíram para a realização deste trabalho.

Primeiramente aos meus pais, que por intermédio de muito trabalho, proveram apoio e educação.

Aos demais membros da minha família, pela paciência nos últimos meses.

A Professora Dr[a]. Margrit Reni Krug, pela orientação neste trabalho e ao Professor Dr. Eduardo Luis Rhod pela coorientação, contribuindo para minha formação profissional e acadêmica.

Ao agora Mestre Elcio Kondo, pelas infinitas dicas sobre o processo de teste.

Na pessoa da querida Bruna Konzen Severo, representando o corpo administrativo, pela dedicação ao programa de mestrado.

Aos colegas do laboratório, pela amizade e apoio.

Aos bolsistas Igor Tedeschi Franco e Israel Aires Cândido.

Ao Programa de Apoio ao Desenvolvimento Tecnológico da Indústria de Semicondutores (PADIS), pela concessão de bolsa de estudo.

Aos membros do itt CHIP.

Aos demais professores e colegas da UNISINOS.

"*Os cientistas estudam o mundo como ele é; os engenheiros criam um mundo como ele nunca havia sido*".

(Theodore von Karman)

RESUMO

O presente trabalho consiste em uma proposta de uma plataforma reconfigurável para testes de módulos de memória SDRAM DDR3. Testadores de módulos de memória consistem em sistemas de arquiteturas fechadas, nos quais o usuário possui pouca flexibilidade em sua utilização, transporte e são na maioria das vezes sistemas volumosos próprios para uso em bancadas. Neste cenário, uma plataforma portátil de baixo custo, que possibilite ao usuário descrever os algoritmos de teste torna-se interessante. A plataforma desenvolvida utiliza de *Field Programmable Gate Arrays* (FPGA) o que proporciona a característica de reconfiguração. Neste projeto foi proposta e validada uma estratégia de injeção de falhas do tipo *Stuck-At-Zero*, aliado a um sistema automático para coleta de vetores de teste e para a síntese em diferentes frequências de acesso aos módulos de memória. A etapa de validação do protótipo desenvolvido possibilitou reportar a captura de 131.751 falhas, graças ao framework criado para acompanhar a tarefa de injeção de falhas.

Palavras-chaves: Teste Eletrônico. Memória SDRAM DDR3. Hardware Reconfigurável. FPGA. Teste de Memória.

ABSTRACT

This work consists on a proposal of a DDR3 SDRAM memory module reconfigurable test platform. Memory module testers are usually closed architecture systems, in which the user has little flexibility in their use. In this scenario, a low-cost portable platform, which enables the user to describe his own test algorithm becomes interesting. This work explores the use of Field Programmable Gate Arrays (FPGAs) in order to construct a fully reconfigurable testing platform. In this work a Stuck-At-Zero fault injection strategy was proposed and validated. Results report the success in executing fault detection algorithms as well as the software framework developed for the fault injection campaign.

Key-words: Electronic Testing. DDR3 SDRAM Memory. Reconfigurable hardware. FPGA. Memory test.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ATE	Automatic Test Equipment (Equipamento Automático de Teste)
CAS	Column Address Strobe (Habilitação do Endereço de Coluna)
CI	Circuito Integrado
CPLD	Complex Programmable Logic Device (Dispositivo de Lógica Programável Complexo)
DDR	Double Data Rate (Dupla Taxa de Dados)
DIMM	Dual In-line Memory Module (Módulo de Memória em Linha Dupla)
DRAM	Dynamic Random Access Memory (Memória de Acesso Aleatório Dinâmica)
DUT	Device Under Test (Dispositivo Sobre Teste)
FPGA	Field Programmable Gate Array (Matrizes de Portas Programáveis em Campo)
HDL	Hardware Description Language (Linguagem de Descrição de Hardware)
LUT	Look-Up Table (Tabela de Consulta)
IP	Intellectual Property (Propriedade Intelectual)
JEDEC	Joint Electron Device Engineering Council (Conselho Conjunto de Engenharia de Dispositivos Eletrônicos)
RAM	Random Access Memory (Memória de Acesso Aleatório)
RAS	Row Address Strobe (Habilitação do Endereço de Linha)
SEM	Scanning Electron Microscopy (Microscópio Eletrônico de Varredura)
SDRAM	Synchronous Dynamic Random Access Memory (Memória de Acesso Aleatório Dinâmica Síncrona)
SODIMM	Dual In-line Small Outline Memory Module (Módulo de Memória em Linha Dupla de Pequeno Perfil)
SPLD	Simple Programmable Logic Device (Dispositivo de Lógica Programável Simples)
SRAM	Static Random Access Memory (Memória de Acesso Aleatório Estática)

SUMÁRIO

1 INTRODUÇÃO

O avanço tecnológico na área dos semicondutores permitiu o desenvolvimento de memórias de acesso aleatório, do inglês, *Random Access Memory* (RAM) de grandes capacidades de armazenamento de dados e elevadas velocidades de acesso (SAMSUNG, 2013), (WESTE; HARRIS, 2013), (CHEN, 2007). Memórias RAMs são indispensáveis em sistemas digitais computacionais, sendo reponsáveis por grande parte do desempenho do sistema (TOCCI; WIDMER; MOSS, 2007), (BONATTO, 2009). No sentido de permitir a portabilidade e utilização das memórias RAM nos mais diversos equipamentos, elas são montadas em módulos (JEDEC, 2013). No entando, as condições de uso ou de manufatura adversas podem levar os módulos a desenvolver falhas (BUSHNELL; AGRAWAL, 2000), (MARTIN, 1999), (HUANG et al., 2011).

O teste de sistemas eletrônicos permite verificar se um determinado dispositivo ou sistema apresenta correto funcionamento baseado nas especificações do mesmo (BUSHNELL; AGRAWAL, 2000), (GROUT, 2006). No que se refere a teste de módulos de memórias, tanto em nível de produção como posteriormente em situações de retrabalho, são realizados por sistemas comerciais de teste. Aplicam-se padrões pré-determinados chamados vetores de teste para verificar o funcionamento do módulo de memória e assim determinar a porcentagem de falhas detectadas em relação às existentes. Tal porcentagem é denominada cobertura de falhas (GROUT, 2006), (INNOVENTIONS, 2015), (CST, 2015) (LINDER et al., 2014). Os testadores comerciais possuem uma arquitetura fechada, o que dificulta ou até mesmo impossibilita a adição de testes extras ou a alteração dos algoritmos gerados de vetores de teste já implementados. A esta dificuldade somam-se as restrições existentes na modificação de um hardware ou software proprietário (INNOVENTIONS, 2015), (CST, 2015). Agregado a isso, o custo de um Equipamento Automático de Teste pode chegar a centenas de milhares de dólares (WANG; WU; WEN, 2006).

Neste sentido, uma plataforma reconfigurável torna-se interessante sob o ponto de vista não apenas comercial mas também acadêmico. Ela pode permitir que centros de pesquisa desenvolvam e validem algoritmos de teste, visto que o cenário dos dispositivos eletrônicos está em constante alteração, especialmente no que se refere ao aumento da complexidade de sistemas digitais, graças a adição cada vez maior de elementos em uma pastilha de silício (REIS, 2002), (GROUT, 2006). Também serve como motivação deste trabalho a promoção de tecnologia nacional para a indústria de manufatura eletrônica, além de permitir customização do teste ao produto em desenvolvimento.

O trabalho tem como objetivo principal a proposta de uma plataforma de teste de módulos de memória SDRAM DDR3 com parametrização dos padrões de testes aplicados. Essa característica habilita ao usuário ampliar a cobertura de falhas, uma vez que, ao descrever

os padrões aplicados, customiza-se o teste do dispositivo. A proposta refere-se a um sistema portátil de reduzidas dimensões para facilitar seu transporte, além de permitir conexão com um computador pessoal para definição dos algoritmos de teste aplicados através de interface USB.

Os objetivos gerais consistem no estudo de falhas em sistemas eletrônicos e algoritmos de teste de memória. Além disso, contribuição no desenvolvimento de subsídios, tando em nível de hardware como de software, para ampliação do *know-how* do Instituto de Tecnológico em Semicondutores (itt CHIP) da Universidade do Vale do Rio dos Sinos (UNSINOS).

No desenvolvimento deste projeto utilizou-se *Field Programmable Gate Arrays* (FPGAs), para acessar o módulo DDR3. O uso de FPGA torna a elaboração de sistemas digitais facilitada (CHEN, 2007) (GROUT, 2008), em especial pela possibilidade de reúso de núcleos de propriedade intelectual, do inglês, *Intellectual Property* (IP) (BONATTO, 2009).

No sentido de executar o teste, aplicou-se o modelo funcional de memória. Esse modelo abstrai os diversos mecanismos internos de uma memória, para um conjunto de blocos funcionais elementares (GOOR, 1993), de forma a identificar se a execução das funções, segue as especificações do projeto no que se refere a valores esperados nas saídas do circuito (KRUG, 2007).

Estudos anteriores, exploram aspectos correlatos desenvolvidos neste trabalho, mas com enfoques diversos. Em Oliveira et al. (2005) os autores propõem uma plataforma de baixo custo para teste de circuitos digitais. Já em Zhou, Cheng e Liu (2006) e Bonatto (2009), o desenvolvimento de controladores de memória DDR com uso de FPGAs é descrito comprovando-se a viabilidade de interface entre memórias DDR com estes dispositivos. O projeto de um Equipamento Automático de Teste baseado em FPGA é apresentado em (MOSTARDINI et al., 2009). Em Keezer et al. (2015) a elaboração de um cartão de teste de memórias que permite uma atualização em um Equipamento Automático de Teste é exposta. Estes trabalhos atestam o uso de FPGAs em aplicações de altas velocidades de transferências. Os sistemas descritos não exploram os aspectos específicos desejados por este trabalho, como o desenvolvimento de um sistema portátil.

Não é objetivo deste trabalho o desenvolvimento específico de um núcleo controlador de memórias DDR. Para tanto, foram empregadas alternativas como o uso de núcleos IP. Essa dissertação também não se destina a criar ou classificar algoritmos de teste em particular.

O texto desta dissertação está organizado da seguinte forma: no Capítulo 2, uma revisão bibliográfica contendo os principais conceitos para contextualização do estado da arte da área. Em seguida, no Capítulo 3, a metodologia utilizada para atingir o objetivo do projeto é discutida. No Capítulo 4 os resultados são apresentados e discutidos, enquanto, no Capítulo 5, as considerações finais do trabalho são expostas.

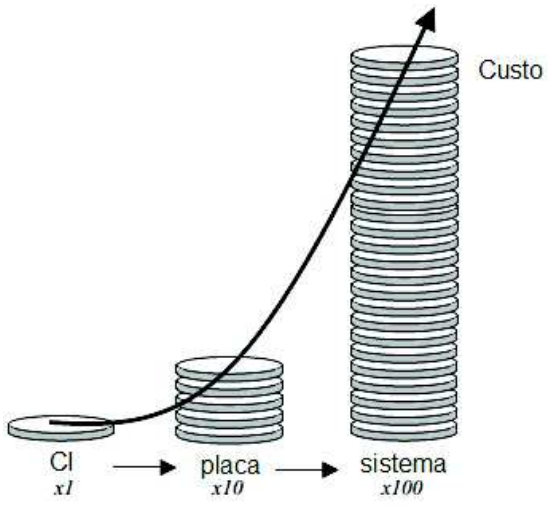
2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo é apresentada a fundamentação teórica necessária para compreensão dos objetivos desse trabalho. Neste sentido, uma descrição da teoria do teste de sistemas eletrônicos é desenvolvida, seguida pelo embasamento dos principais requisitos necessários para o entendimento do trabalho.

2.1 Teste em Sistemas Eletrônicos

O teste de sistemas eletrônicos permite verificar se um determinado dispositivo ou sistema apresenta correto funcionamento, baseado nas especificações do mesmo. A importância do teste é visualizada na Figura 1, na qual verifica-se que o custo de reparação incrementa por um fator 10 a cada avanço de etapa no processo de fabricação de um sistema que utiliza circuitos integrados (CI). Neste sentido, torna-se de fundamental prioridade que qualquer anomalia seja descoberta o mais rápido possível durante as fases de desenvolvimento de um produto (GROUT, 2006). Nesta complexidade somam-se os atuais níveis de integração de circuitos que adicionam milhões de elementos lógicos em uma pastilha, onde todos necessitam de correto funcionamento para a aplicação (BUSHNELL; AGRAWAL, 2000), (GROUT, 2006).

Figura 1 – Aumento do custo de reparação em diferentes etapas.



Fonte – Adaptado de (GROUT, 2006).

Desta forma, um conjunto de estímulos devem ser aplicados ao sistema de maneira a excitar diferentes partes do circuito, possibilitando-se a aferição do funcionamento. Estes estímulos recebem o nome de vetores de teste. A quantidade de vetores necessários para essa aferição será uma resultante das características do circuito ou sistema em estudo (BUSHNELL; AGRAWAL, 2000). Assim, torna-se necessário que o dispositivo em teste possua algumas

características que tornam o teste mais facilitado. No âmbito da literatura, essas características são denominadas de observabilidade e controlabilidade (GROUT, 2006).

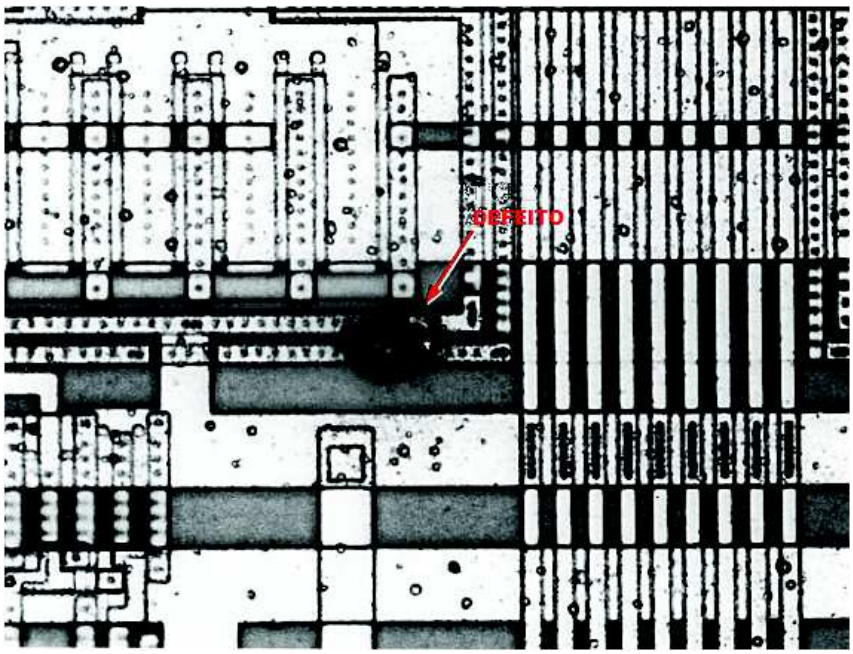
A controlabilidade avalia a facilidade com que um nível lógico desejado pode ser atribuído em um determinado ponto interno no circuito. A observabilidade indica a aptidão de avaliar esse nível lógico nas saídas primárias do sistema ou circuito (GROUT, 2006).

O teste pode ser classificado em teste funcional e teste estrutural. No teste estrutural busca-se descobrir a presença de algum defeito do processo de fabricação. O teste funcional abstrai a estrutura física e identifica se a execução das funções segue as especificações do projeto, no que se refere a valores esperados nas saídas do circuito (BUSHNELL; AGRAWAL, 2000).

2.2 Defeito, Falha e Erro em Sistemas Eletrônicos

Segundo Bushnell e Agrawal (2000), defeitos são imperfeições físicas existentes no circuito. Os defeitos possuem sua origem em problemas ocorridos, no processo de fabricação do circuito integrado. Durante o procedimento de produção, a presença de partículas em suspensão na sala limpa, impurezas no material dopante ou no *wafer*, além de desalinhamentos nas máscaras utilizadas, são condições indesejadas mas que podem levar à ocorrências de defeitos, conforme apresentado na Figura 2, onde uma partícula danifica um circuito integrado.

Figura 2 – Defeito em um circuito integrado causado por uma partícula.



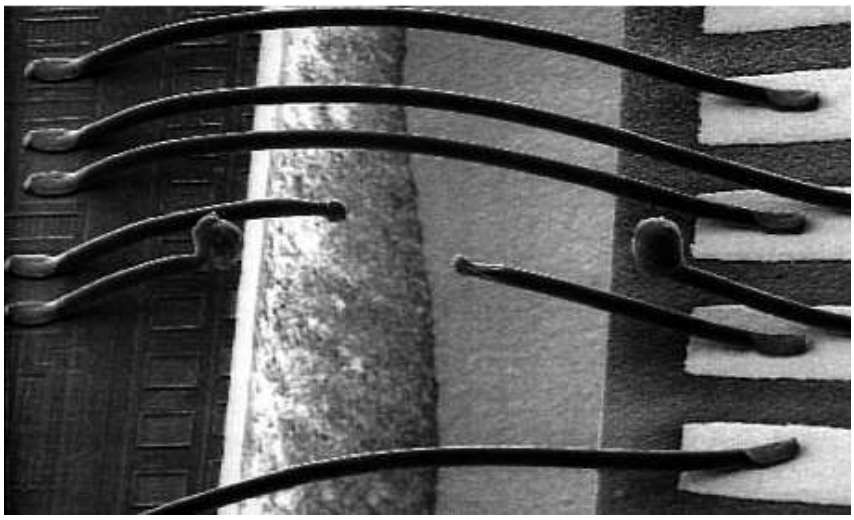
Fonte – Adaptado de (MARTIN, 1999)

A abstração de um defeito é conhecida como falha, segundo Moraes (2006 apud BUSHNELL; AGRAWAL, 2000), referindo-se a uma manifestação interna do sistema conforme citado por Krug (2007 apud BUSHNELL; AGRAWAL, 2000). A manifestação externa de um defeito é

denominada de erro, segundo Amory (2003 apud BUSHNELL; AGRAWAL, 2000). Segundo Krug (2007) entende-se o processo de teste como a verificação da existência de defeitos, cujas falhas são descobertas pela detecção de erros.

Segundo Bushnell e Agrawal (2000), erros em sistemas ocorrem quando a resposta do sistema é incorreta. A diferença entre o comportamento considerado correto e o incorreto indica a presença de um defeito. Defeitos podem ser classificadas como permanentes ou não permanentes. Defeitos permanentes são imperfeições físicas que existirão indefinidamente. Desta forma, podem ser modelados por modelos de falhas (BUSHNELL; AGRAWAL, 2000). Podem ser causados por ligações elétricas danificadas, como ilustrado na Figura 3, componentes contendo trincas em seu interior, como exibido na Figura 4, além de soldas que não proporcionam contato elétrico adequado, como mostra a Figura 5 (MARTIN, 1999). Condições inadequadas no processo de manufatura de placas de circuito impresso, podem causar defeitos permanentes em sistemas eletrônicos (HUANG et al., 2011). O defeito exibido na Figura 2 também ilustra um defeito permanente (MARTIN, 1999).

Figura 3 – Fios de conexão danificados internamente no circuito integrado.



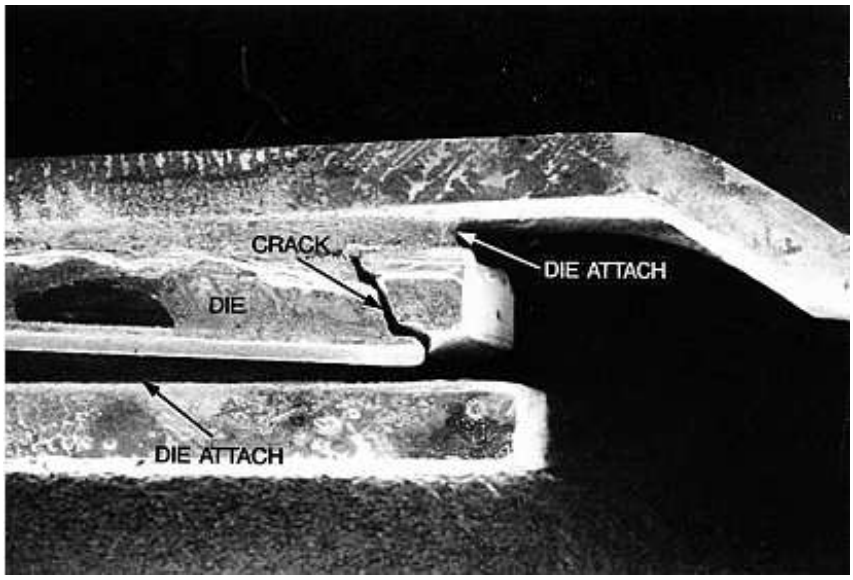
Fonte – (MARTIN, 1999)

Conforme visualizado na Figura 3, condições impróprias de operação como sobre correntes ou surtos de tensão nos terminais de um circuito integrado podem resultar em rompimento das ligações internas dos circuitos integrados (CIs). Os fios também podem ser danificados na etapa de encapsulamento da pastilha de silício (MARTIN, 1999).

Choques mecânicos como os causados por quedas do dispositivo ou até mesmo a presença de vibrações em excesso, em condições de uso ou no transporte dos circuitos integrados, podem danificar a pastilha de silício quebrando-a, conforme visualiza-se na Figura 4, desta forma, inutilizando o componente (MARTIN, 1999).

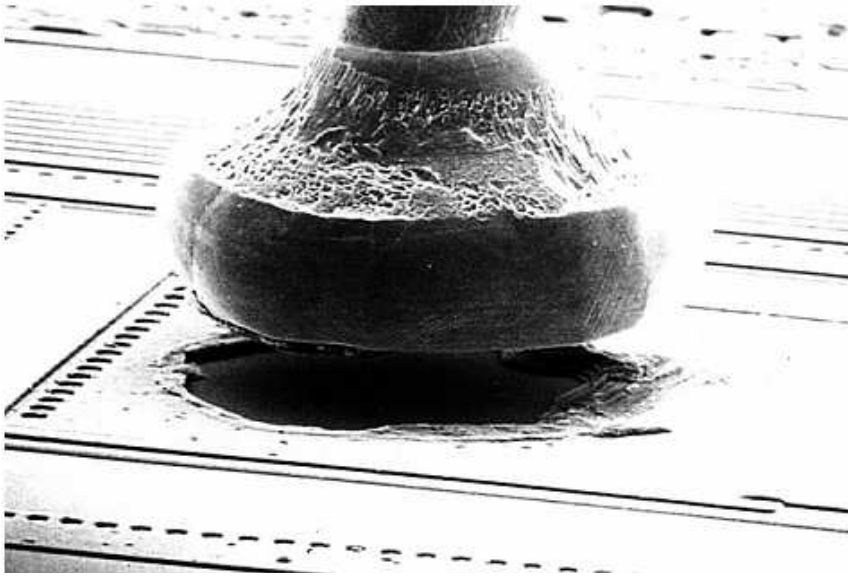
O processo de soldas de fios consiste em uma etapa crítica uma vez que existe necessidade de correta parametrização dos procedimentos envolvidos. Caso o método não seja corretamente

Figura 4 – Pastilha de silício apresentando uma trinca.



Fonte – (MARTIN, 1999)

Figura 5 – Defeito em solda de fios no interior do circuito integrado.



Fonte – (MARTIN, 1999)

realizado, pode existir perda de contato elétrico como o visualizado na Figura 5 (MARTIN, 1999).

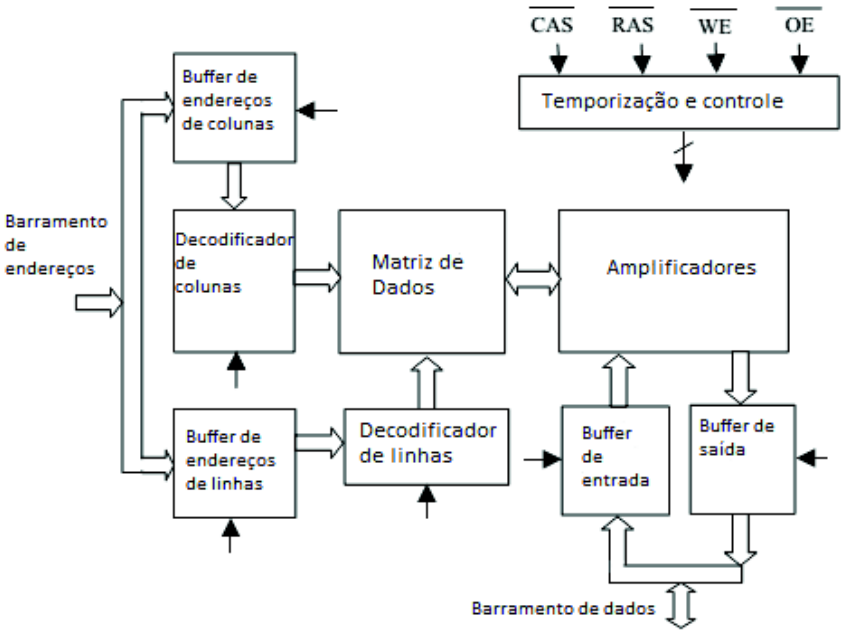
Segundo Bushnell e Agrawal (2000), defeitos não permanentes ocorrem de forma aleatória, desta maneira não possuem um modelo definido. Sua natureza não determinística indica que serão presentes apenas nos intervalos de tempo de seus mecanismos de ativação. Circuitos integrados são especialmente sensíveis a estes fenômenos. A seguir citam-se as causas mais comuns destes defeitos:

- Raios cósmicos;
- Umidade Relativa do ar;
- Temperatura do Ambiente;
- Pressão Atmosférica;
- Vibrações Mecânicas;
- Flutuações na Alimentação;
- Interferência Eletromagnética, e;
- Descargas Eletroestáticas.

2.3 Memórias DRAM

Memórias de Acesso Aleatório, do inglês, *Random Acess Memory* (RAM) do tipo Dinâmicas (DRAM) armazenam sua informação na forma de cargas elétricas em um capacitor. Desta forma, seu conteúdo deve ser periodicamente reescrito, a fim de que a célula não perca informação. Dispositivos DRAM são construídos em processos especializados para formar estruturas com grandes capacitores (WESTE; HARRIS, 2013). A estrutura interna de uma DRAM é exibida na Figura 6.

Figura 6 – Estrutura da DRAM



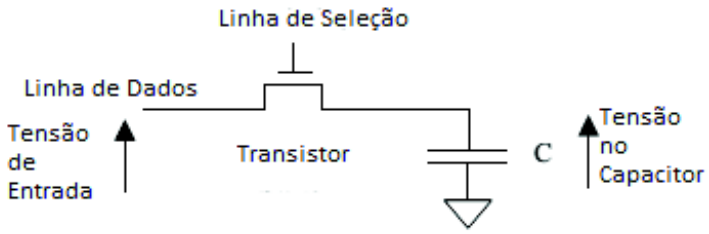
Fonte – Adaptado de (GROUT, 2006)

Os dados armazenados na DRAM são acessados por intermédio da decodificação do endereço de acesso que contém a posição de memória para a matriz DRAM, onde estão as células

que armazenam os bits dos dados. As entradas da memória representam os sinais de controle e de endereços. Os sinais de controle são responsáveis por informar a natureza da operação que é executada e os sinais de endereço indicam a localização da célula para executar a operação. A saída de dados são as linhas que transferem as informações das células para o exterior e vice-versa. É indispensável garantir-se que os tempos de acesso à memória sejam respeitados. Caso contrário, leituras incorretas serão realizadas (GROUT, 2006).

A Figura 7 ilustra a célula de uma RAM dinâmica. A informação é acessada ao se atribuir valor lógico alto à linha de seleção, conectando o capacitor na linha de dados (WESTE; HARRIS, 2013).

Figura 7 – Construção da Célula DRAM



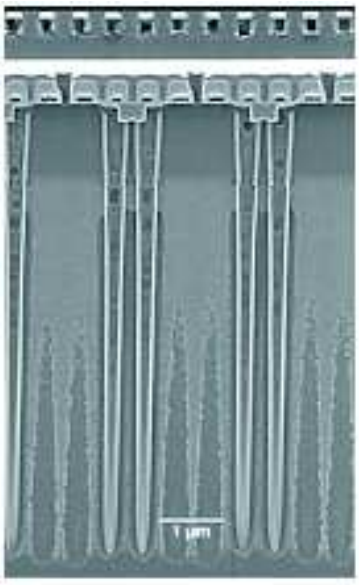
Fonte – Adaptado de (GROUT, 2006)

De modo a manter a variação de tensão em níveis toleráveis ao longo do tempo, é necessário que o valor da capacitância da célula seja corretamente dimensionado. Um valor típico está em torno de 30fF. No processo de fabricação especializado de DRAM, a maneira mais compacta para construir uma elevada capacitância é criar verticalmente o capacitor (WESTE; HARRIS, 2013). Na Figura 8, exibe-se um corte transversal em uma célula utilizando-se um microscópio eletrônico de varredura. Visualiza-se a estrutura tubular que origina o capacitor da célula. As paredes são cobertas com um óxido isolante e a estrutura é então preenchida com um condutor, que possui função de terminal, ligando o capacitor ao dreno do transistor, enquanto que o substrato fortemente dopado é o outro terminal (WESTE; HARRIS, 2013).

Grandes memórias DRAM são divididas em várias submatrizes (subarrays). A Figura 9 exibe um exemplo desta estrutura. O tamanho de uma submatriz representa um ponto comum entre a densidade de memória e o seu desempenho. Grandes submatrizes reduzem os decodificadores e amplificadores, alcançando assim uma melhor eficiência. No entanto, esse efeito reduz a velocidade por efeito das grandes capacitâncias (WESTE; HARRIS, 2013).

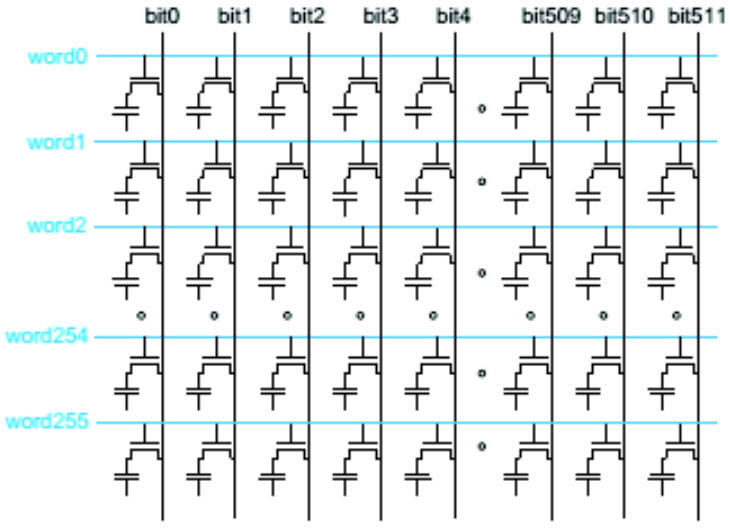
Durante um ciclo assíncrono de acesso à memória DRAM, o processador deve esperar pela memória. Gera-se com isso baixa eficiência na transferência de informações. Em uma interface síncrona, o controle das operações do sistema é realizado na borda do sinal de clock. Os dados de um determinado endereço estarão disponíveis depois de um certo número de ciclos de clock, deixando o processador livre. Na Figura 10, exibem-se as formas de onda de acesso de uma Memória Dinâmica de Acesso Aleatório Síncrona, do inglês, *Synchronous Dynamic*

Figura 8 – Imagem de varredura da Célula DRAM



Fonte – Adaptado de (WESTE; HARRIS, 2013)

Figura 9 – Construção de arrays de Células DRAM



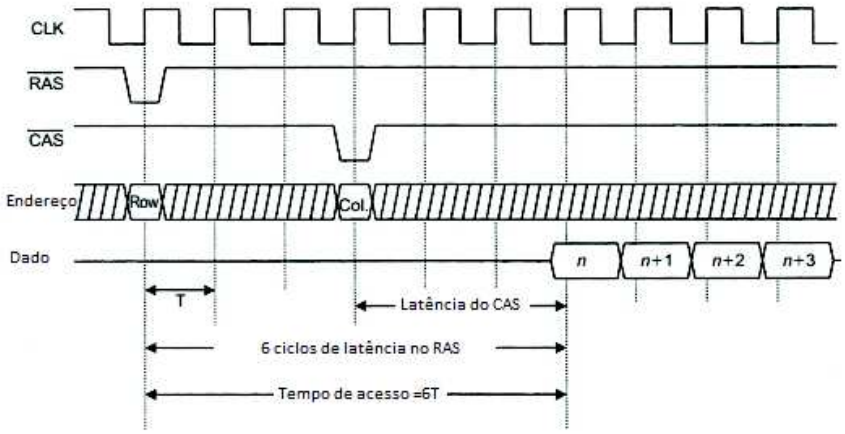
Fonte – Adaptado de (WESTE; HARRIS, 2013)

Random Access Memory (SDRAM), referidas deste ponto em diante apenas como SDRAM. Como resultado da interface síncrona, a velocidade de operação do sistema é aumentada (CHEN, 2007).

2.3.1 Memórias DDR

Segundo Bonatto (2009), memórias DRAM síncronas com transmissão de dados usando taxa simples, desenvolveram o conceito de agendamento de dados para sincronização com o barramento externo. Essa arquitetura separa o núcleo da memória e o circuito de sincronização.

Figura 10 – Forma de onda de acesso em uma SDRAM



Fonte – Adaptado de (CHEN, 2007)

Desta forma o núcleo da memória e a interface de dados trabalham com diferentes taxas de clock.

Manter o núcleo operando em uma frequência mais baixa e duplicar a taxa de transferência de dados representa uma significativa redução no custo da memória, pois incrementar a frequência do núcleo torna necessário o uso de processos de fabricação mais complexos (BONATTO, 2009).

Memórias de Dupla Taxa de Dados, do inglês, *Double Data Rate* (DDR), sendo referidas desde ponto em diante como memórias DDR, são padronizadas de acordo com as normas do conselho chamado JEDEC, do inglês, *Joint Electron Device Engineering Council*. A JEDEC consiste em um consórcio de empresas que, juntas, trabalham para a padronização de dispositivos semicondutores (JEDEC, 2015).

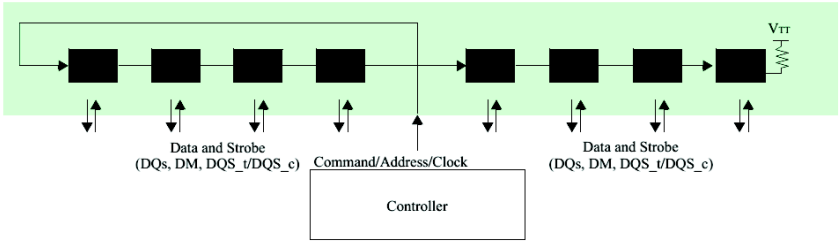
O padrão DDR3 é definido na norma JEDEC JESD79, sendo sua mais recente atualização desenvolvida em 2012. A norma define o padrão DDR3 SDRAM, onde lista recursos, funcionalidades, características de sinais AC e DC, encapsulamento dos circuitos integrados e sua configuração de pinos. Seu objetivo é definir um conjunto mínimo de requisitos (JEDEC, 2012).

Módulos de memória são descritos em normas complementares. De acordo com necessidades diversas, diferentes características de capacidade e dimensões físicas são requeridas, sendo essas então definidas em normas suplementares. Este trabalho enfoca em módulos de memórias DDR3 SDRAM no formato conhecido como DIMM, do inglês, Dual In-Line Memory Modules. Estes módulos são descritos na parte 4.20.19 e 4.20.20 da norma JEDEC JESD21 (JEDEC, 2013).

O diagrama interno de um módulo de memória DDR3 SDRAM é exibido na Figura 11 (JEDEC, 2013). A quantidade de circuitos integrados em um módulo é função da capacidade

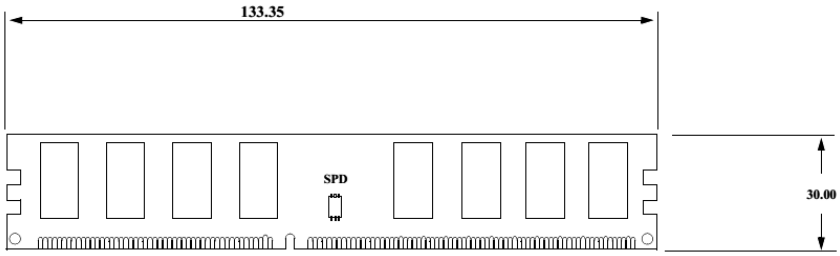
de armazenamento do mesmo podendo variar desde quatro circuitos integrados até dezesseis (JEDEC, 2012). Na Figura 12, as dimensões físicas padronizadas do módulo são exibidas junto com seu padrão físico.

Figura 11 – Diagrama interno do módulo DDR3.



Fonte – Adaptado de (JEDEC, 2013)

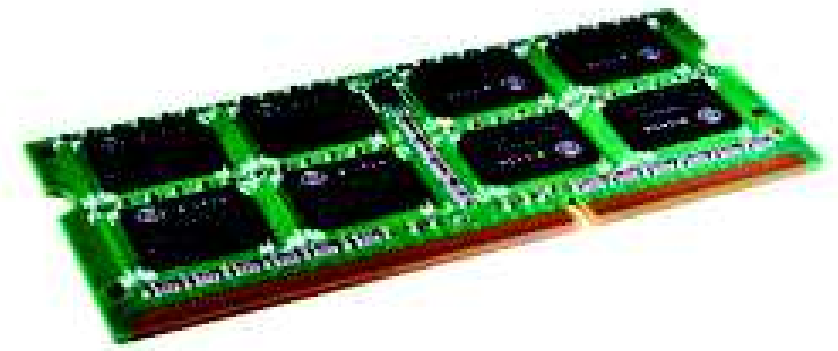
Figura 12 – Dimensões de módulo DDR3.



Fonte – Adaptado de (JEDEC, 2013)

O uso em dispositivos movéis como Notebooks gera a necessidade de criar módulos de dimensões reduzidas. Assim, o padrão conhecido como SO-DIMM, do inglês, *Dual In-Line Small Outline Memory Module*, foi desenvolvido. Este padrão é definido na parte 4.20.18 da norma JEDEC JESD21 (JEDEC, 2014). A Figura 13 exibe um exemplar de um módulo no formato SODIMM.

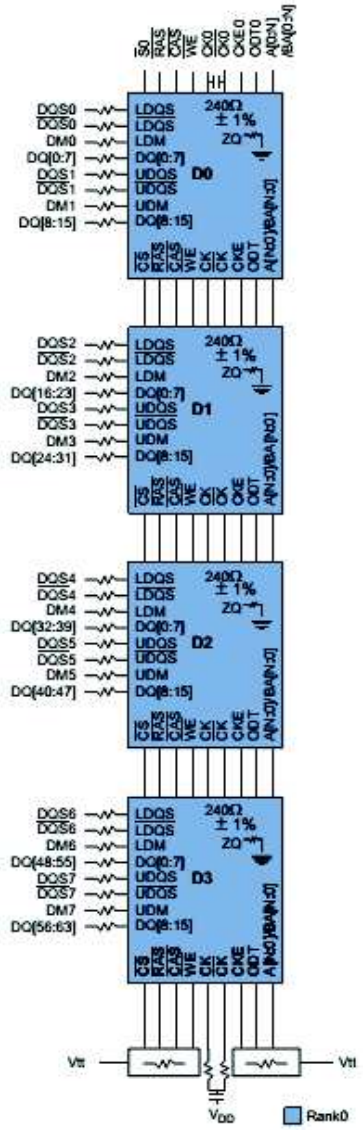
Figura 13 – Módulo DDR3 SODIMM.



Fonte – (TEIKON, 2015).

Conforme exibido na Figura 14, os módulos são formados pela conexão elétrica de vários circuitos integrados. O número de circuitos integrados presentes no módulo varia de acordo com sua capacidade e da quantidade de bits de dados que os mesmos possuem (SAMSUNG, 2013).

Figura 14 – Conexão elétrica dos CI's no módulo DDR3 SODIMM.



Fonte – Adaptado de (SAMSUNG, 2013).

2.4 Dispositivos Lógicos Programáveis

Dispositivos lógicos programáveis, do inglês, *Programmable Logic Device* (PLDs) são circuitos integrados que contêm células lógicas e interconexões programáveis. Estes dispositivos permitem ao projetista configurar sua lógica de interligações internas para desenvolver um circuito digital. A configuração do hardware é realizada por intermédio de uma Linguagem de Descrisão de Hardware, *Hardware Description Language* (HDL) (GROUT, 2008).

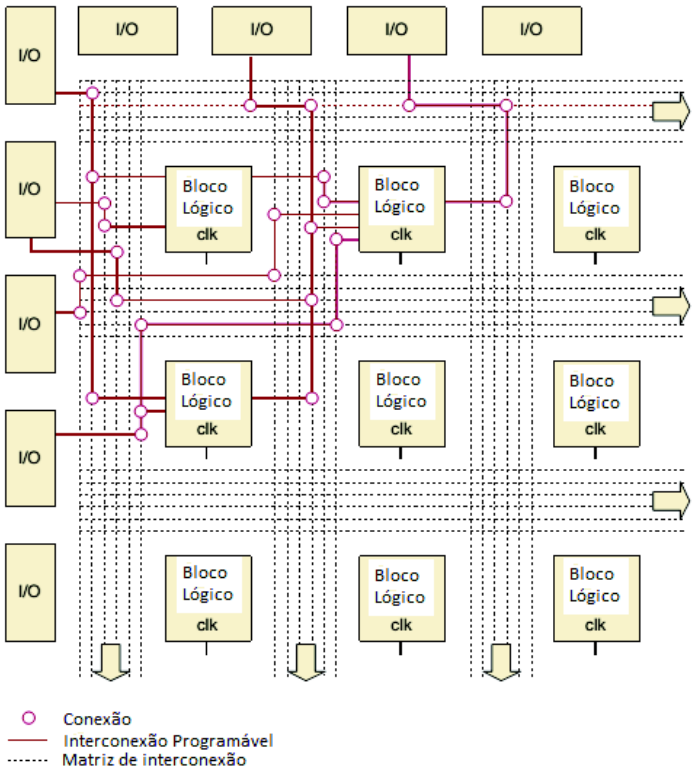
De modo geral, existem três grupos de PLDs (TOCCI; WIDMER; MOSS, 2007): Dispositivo de Lógica Programável Simples, do inglês, *Simple Programmable Logic Device* (SPLD), Dispositivo de Lógica Programável Complexo, do inglês, *Complex Programmable Logic Device* (CPLD), e Matrizes de Portas Programáveis em Campo, do inglês, *Field Programmable Gate Array* (FPGA).

2.4.1 FPGA

FPGAs são formados por blocos simples e independentes de lógica que são interligados criando sistemas complexos. Muitos dispositivos utilizam a abordagem de uma tabela de consulta, *Look-Up Table* (LUT) para implementar as funções lógicas. A LUT armazena a tabela verdade da saída desejada para cada combinação de sinal na entrada do bloco (TOCCI; WIDMER; MOSS, 2007).

O roteamento de sinais dentro do FPGA tende a ser bastante complexo, de modo que os atrasos na propagação dos sinais dependem do mapeamento utilizado pelo software de síntese. Os blocos de lógica não estão diretamente associados aos pinos de I/O, desta forma os pinos são conectados a um bloco de entrada ou saída, que está ligado a blocos de interconexões programáveis. Uma representação geral da estrutura de um FPGA é exibida na Figura 15 (TOCCI; WIDMER; MOSS, 2007).

Figura 15 – Estrutura Interna do FPGA.



Fonte – Adaptado de (TOCCI; WIDMER; MOSS, 2007)

Segundo Chen (2007), com FPGAs a depuração e prototipagem de um sistema é rapidamente desenvolvida graças às Linguagens de Descrição de Hardware. A velocidade com que um FPGA executa um processo é muito mais rápida do que a de um processador. Elementos adicionais de hardware podem ser adicionados em um FPGA, como memórias de acesso aleatório e blocos dedicados de funções. Diversos fabricantes oferecem diferentes arquiteturas de FPGAs com otimizações e customizações para variadas aplicações (INTEL, 2017), (MICROSEMI, 2017), (XILINX, 2017).

FPGAs são amplamente encontrados na prototipação e produção de sistemas que necessitam de computação intensa, como no processamento de imagens e em visão computacional, além de permitir soluções de alto desempenho com customizações para a aplicação em foco, aliando-se a possibilidade de reconfiguração em tempo de execução (SRINIVASA et al., 2017). Estas características habilitam seu uso em diversas áreas, como a de defesa, aeroespacial, automobilistica, comunicações de alta velocidade (INTEL, 2017), (MICROSEMI, 2017), estendendo-se até na elaboração de Redes Neurais Artificiais (COTTENS, 2016). Atualmente estão disponíveis em tecnologias de até 16nm (XILINX, 2017).

2.5 Modelos de Falhas em Memória RAM

Dada a complexidade e quantidade dos elementos internos de uma memória, uma análise física da mesma não é possível, sendo assim, faz-se necessário comparar o comportamento lógico de uma memória com defeito, contra uma funcional. Isso implica em uma modelagem das falhas físicas como falhas lógicas. A modelagem de falhas torna a metodologia de ensaio independente da tecnologia e do processo de fabricação, uma vez que falhas físicas dependem da tecnologia empregada (BUSHNELL; AGRAWAL, 2000).

Conforme Bushnell e Agrawal (2000), os modelos de falhas mais comuns em memórias são:

- *Stuck-At (SAF)*: Quando o valor da célula é fixo em um nível lógico, seja um ou zero, não alterando de estado em nenhum momento;
- *Transition Faults (TF)*: Falha de transição, quando a célula falha em uma transição de lógica, seja de nível alto para baixo, ou vice-versa;
- *Coupling Faults (CF)*: Falha de acoplamento, quando a escrita em uma célula provoca alteração do conteúdo de uma célula vizinha;
- *Inversion Coupling Faults*: Caso especial de falha de acoplamento, quando a transição em uma célula inverte o estado da célula vizinha;
- *Adress Fault (AF)*: Falha de endereçamento, quando o bloco de decodificação de endereços apresenta falhas e assim um endereço incorreto é acessado ou até vários simultaneamente;

- *Neighborhood Pattern Sensitive Coupling Faults*: Quando uma célula é afetada pela alteração do estado de células vizinhas seguindo um padrão, não apenas uma célula, mas um conjunto delas ao seu redor.

2.6 Teste de Memória RAM

Segundo Grout (2006), o teste de uma memória RAM baseia-se em operações de escrita e leitura de valores em um determinado endereço da memória. A leitura do conteúdo da posição de memória, é comparada com o valor anteriormente escrito na mesma posição. Caso os valores sejam os mesmos, o endereço em teste é considerado funcional. Se os valores divergirem, a posição de memória é considerada danificada. A ordem em que os endereços de memória serão acessados e os valores lidos ou escritos, será definida pelo algoritmo de teste em uso. A cobertura de falhas para cada algoritmo, bem como seu tempo de execução, deverão ser considerados.

2.6.1 Algoritmos de Teste de Memória RAM

Cabe destacar que esse trabalho não teve como objetivo criar ou classificar algoritmos de teste em particular, mas acredita-se que uma breve introdução seja importante para a contextualização do trabalho, bem como para evidenciar a importância do estudo e desenvolvimento destes para a otimização do processo de teste.

Entende-se como cobertura de falhas de um algoritmo de teste, a indicação da porcentagem de falhas detectáveis pelo algoritmo em relação ao número total de falhas existentes (LINDER et al., 2014) e que, graças ao uso de ferramentas de simulação, pode ser determinado (AMORY, 2003). Neste sentido, um algoritmo é dito eficaz se sua cobertura é considerada alta (LINDER et al., 2014). Porém, uma cobertura de 100% não garante um sistema livre de falhas (AMORY, 2003).

Algoritmos de teste de memória são baseados na modelagem das falhas que se deseja encontrar (BUSHNELL; AGRAWAL, 2000), (GROUT, 2006). Nos parágrafos abaixo descrevem-se alguns dos testes mais citados na literatura (WANG; WU; WEN, 2006), (GOOR, 2004), (GOOR, 1993), (LINDER et al., 2014).

O algoritmo *Memory Scan* (MSCAN) consiste no padrão mais simples e é chamado de "Algoritmo Zero-One". Nele as posições de memórias são preenchidas com *0's* e depois verificadas. Na sequência, as células são gravadas e verificadas com *1's*. O procedimento é repetido em todos os endereços. De maneira similar o teste *Checker Pattern* povoa a memória com uma sequência alternada de 0s e 1s, por exemplo, 01010101 no endereço atual, seguido por 10101010 no próximo. Em seguida as células são lidas para verificação. As posições são novamente preenchidas com um padrão complementar ao anterior, seguido pelo processo de leitura para a verificação (GROUT, 2006). Estes consistem em padrões clássicos que possibilitam apenas detecção de falhas do tipo *Stuck-At* (LINDER et al., 2014).

No algoritmo *Galloping Patterns* (GALPAT) uma célula base é alternadamente lida e comparada com o restante da memória (WANG; WU; WEN, 2006). Inicia-se o conteúdo da memória em zero, verificando-se a partir do endereço mais baixo. O dado da posição de memória é lida, esperando-se por um valor igual a zero. Na sequência, o valor do endereço atual é preenchido com '1's em todos os bits da palavra. Neste processo, a partir do endereço atual, as posições de memória inferiores são verificadas para a presença de '1's, e as posições superiores verificadas para a presença de '0's, pois ainda não foram checadas. Segundo Wang, Wu e Wen (2006), GALPAT consiste em um algoritmo lento, não sendo usado em testes de produção, apesar de possuir boa cobertura para falhas de transição e falhas de acoplamento.

Os testes de memória mais utilizados na atualidade realizam os algoritmos assim chamados de *March* (LINDER et al., 2014). Neles as operações são realizadas em uma célula antes de prosseguir para a seguinte. Algoritmos March foram desenvolvidos como alternativa de teste de memórias no final dos anos 70 (NAIR, 1979). Já naquela época os métodos de teste consumiam tempos relativamente altos, lembrando que as memórias em questão na época eram da ordem de 16k (NAIR; THATTE; ABRAHAM, 1978), (SUK; REDDY, 1981).

Os testes desenvolvidos possuem complexidade linear, acelerando o tempo de teste. São definidos como um conjunto de operações elementares atribuídos a uma célula sob teste. São desta forma descritos por uma simbologia que é detalhada na Tabela 1 (BUSHNELL; AGRAWAL, 2000). Assim um programa de teste March nada mais é do que um conjunto de instruções em notação de teste formando um algoritmo específico desenvolvido para excitar uma falha em particular.

Tabela 1 – Notação utiliza em algoritmos MARCH

Notação	**Descrição**
r	Operação de leitura
w	Operação de escrita
r0	Operação de leitura, com expectativa de ler valor lógico 0
r1	Operação de leitura, com expectativa de ler valor lógico 1
w0	Operação de escrita, com expectativa de escrever valor lógico 0
w1	Operação de escrita, com expectativa de escrever valor lógico 1
$\uparrow$	Atribui valor lógico 1 na célula, altera seu estado de baixo para alto
$\downarrow$	Atribui valor lógico 0 na célula, altera seu estado de alto para baixo
$\updownarrow$	Complementa o valor da célula
$\Uparrow$	Incrementa endereço
$\Downarrow$	Decrementa endereço
$\Updownarrow$	Endereços acessados em ordem crescente ou decrescente
$\longrightarrow$	Escrita de nível baixo, em célula contendo nível baixo
$\rightarrow$	Escrita de nível alto, em célula contendo nível alto
$\Rightarrow$	Escreve valor x em célula contendo valor x
D	Realiza um atraso de tempo

Fonte – Adaptado de (BUSHNELL; AGRAWAL, 2000).

Movido pela necessidade de melhorar a cobertura de falhas obtida pelos algoritmos passados, bem como atender as demandas de processos de fabricação modernos, vários tipos de algoritmos MARCH foram desenvolvidos. Atualmente em torno de 30 deles são referenciados em diferentes publicações (GOOR, 1993), (GOOR, 2004), (HAMDIOUI et al., 2006), (LINDER et al., 2014). Alguns se tornaram mais gerais enquanto outros destinam-se a buscas por falhas específicas (LINDER et al., 2014). Na Tabela 2, são descritos alguns algoritmos clássicos segundo a literatura.

Tabela 2 – Descrição de algoritmos MARCH clássicos

Algoritmo	**Notação MARCH**
MATS	{⇕(w0);⇕(r0,w1);⇕(r1)}
MATS+	{⇕(w0);⇑(r0,w1);⇓(r1,w0)}
MATS++	{⇕(w0);⇑(r0,w1);⇓(r1,w0,r0)}
MARCH B	{⇕(w0);⇑(r0,w1,r1,w0,r0,w1);⇑(r1,w0,w1);⇓(r1,w0,w1,w0);⇓(r0,w1,w0)}
MARCH C-	{⇕(w0);⇑(r0,w1);⇑(r1,w0);⇓(r0,w1);⇓(r1,w0);⇕(r0)}
MARCH X	{⇕(w0);⇑(r0,w1);⇓(r1,w0);⇕(r0)}
MARCH Y	{⇕(w0);⇑(r0,w1,r1);⇓(r1,w0,r0);⇕(r0)}
MARCH G	{⇕(w0);⇑(r0,w1,r1,w0,r0,w1);⇑(r1,w0,w1);⇓(r1,w0,w1,w0);⇓(r0,w1,w0); **D**;⇑(r0,w1,r1);**D**;⇑(r1,w0,r0)}

Fonte – Adaptado de (LINDER et al., 2014).

A fim de entendimento da notação utilizada, detalha-se o algoritmo *MATS+*, sendo um dos códigos mais básicos. Com base em sua descrição apresentada na Tabela 2, verifica-se a presença de três elementos March, exibidos com mais detalhes na Tabela 3. Conforme a Tabela 3 o elemento M0 inicia o conteúdo da memória escrevendo zeros em todas as células com ordem de endereçamento indiferente. O elemento M1 realiza em ordem crescente de acesso, a leitura da posição de memória esperando-se ler o valor lógico zero. Imediatamente após é escrito, na mesma posição, o valor lógico um. Somente no término da escrita na posição atual, o endereço é incrementado de uma unidade. Repete-se esse processo até verificar todos os endereços. O elemento M2 acessa a memória em ordem decrescente de endereçamento, lendo-se a célula com expectativa de encontrar valor lógico um. Imediatamente após a leitura, o valor lógico zero é escrito na mesma posição. Somente após essa operação o endereço de acesso é decrementado em uma unidade, verificando-se sequencialmente toda a memória (BUSHNELL; AGRAWAL, 2000).

Tabela 3 – Descrição do algoritmo MARCH MATS+

Elemento	**Notação**	**Descrição**
M0	⇕(w0)	Alheio a ordem de endereçamento, escreve zero em todas as células.
M1	⇑(r0,w1)	Em ordem crescente de endereçamento, lê zero e escreve um.
M2	⇓(r1,w0)	Em ordem decrescente de endereçamento, lê um e escreve zero.

Fonte – Adaptado de (BUSHNELL; AGRAWAL, 2000).

O algoritmo March realiza as operações descritas em cada posição de memória antes de alterar o endereçamento de acesso. Assim as operações de escrita e leitura necessárias ao propósito do teste são desenvolvidas no endereço atual de maneira que, somente ao final da operação, o endereço é modificado de acordo com o padrão do algoritmo. A Tabela 4 permite visualizar as falhas detectáveis por cada algoritmo anteriormente citado. Ilustra-se assim que cada algoritmo apresenta diferentes caracteristícas de cobertura de falhas (BUSHNELL; AGRAWAL, 2000).

Tabela 4 – Falhas capturadas pelos Algoritmos.

Algoritmo	**SAF**	**AF**	**TF**	**CF**
MATS	Todas	Algumas	Nenhuma	Nenhuma
MATS+	Todas	Todas	Nenhuma	Nenhuma
MATS++	Todas	Todas	Todas	Nenhuma
MARCH B	Todas	Todas	Todas	Algumas
MARCH C-	Todas	Todas	Todas	Todas
MARCH X	Todas	Todas	Todas	Algumas
MARCH Y	Todas	Todas	Todas	Algumas
MARCH G	Todas	Todas	Todas	Algumas

Fonte – Adaptado de (BUSHNELL; AGRAWAL, 2000).

2.7 Testadores Comerciais

Um Equipamento Automático de Teste, do inglês *Automatic Test Equipment* (ATE), consiste em um equipamento controlado por computador utilizado no teste de produção de circuitos integrados, tanto a nível de *wafer* como em componentes encapsulados e também no teste de placas de circuito impresso (PCI) (WANG; WU; WEN, 2006). Segundo Bushnell e Agrawal (2000) são equipamentos de elevado custo, ultrapassando centenas de milhares de dólares, uma vez que utilizam uma eletrônica de alta precisão e velocidade. Na Figura 16, um exemplo de ATE é ilustrado.

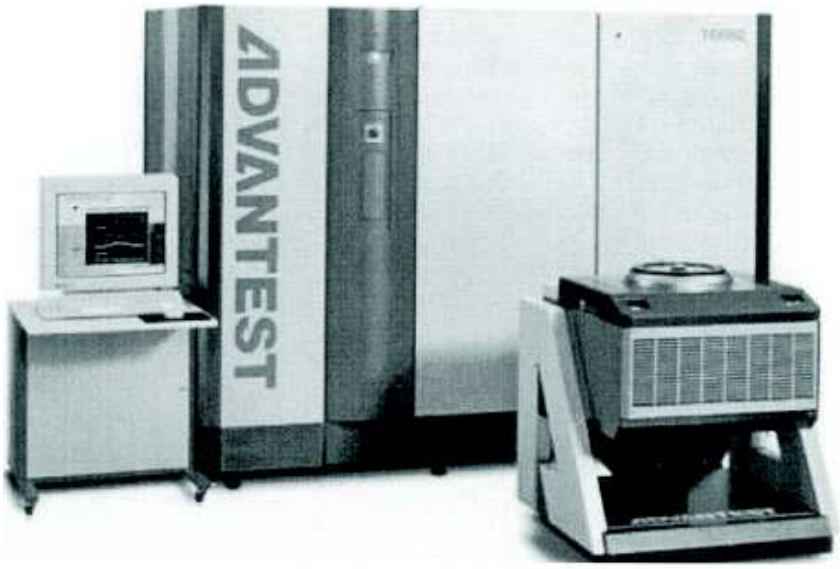
Conforme Wang, Wu e Wen (2006), um ATE deve permitir modularização, uma vez que as necessidades de teste mudam de acordo com o circuito integrado testado, sendo também importante possuir capacidade de paralelismo na execução dos testes e possibilitar o aumento de produção e redução dos custos do teste.

2.7.1 Testadores de Módulos de Memória RAM

Alguns exemplos de testadores comerciais de módulos de memória DDR RAM são destacados. A empresa INNOVENTIONS® Inc, com sede na cidade de Houston, nos Estados Unidos da América (EUA), desenvolve o testador RAMCHECK® LX, exibido na Figura 17.

Segundo Innoventions (2015), o equipamento possui capacidade de teste de módulos de memória DDR1,DDR2 e DRR3. Uma interface USB permite configurações e um display no

Figura 16 – Uma ATE modelo Advantest T6682.



Fonte – Adaptado de (BUSHNELL; AGRAWAL, 2000)

Figura 17 – Testador Comercial RAMCHECK LX, INNOVENTIONS® Inc.

Fonte – (INNOVENTIONS, 2015).

equipamento permite visualizar o *status* do teste. O testador também identifica informações do fabricante do módulo, possui capacidade de alterar a tensão de alimentação do módulo durante o teste e pode operar em modo autônomo, sem auxilio de um computador. O fabricante não informa os testes executados, nem se o usuário pode desenvolver algoritmos próprios.

A empresa CST® Inc, desenvolve o testador Eureka2®, exibido na Figura 18. O Instituto Tecnológico em Semicondutores (itt CHIP), com sede no campus da Universidade do Vale do Rio dos Sinos (UNISINOS) possui em seu laboratório de teste um exemplar deste

testador.

Figura 18 – Testador Comercial Eureka2, da CST Inc.

Fonte – (CST, 2015).

Segundo CST (2009), o testador possui capacidade de teste de módulos de memória DDR1, DDR2 e DRR3. Uma conexão USB é necessária para comunicação entre o testador e um microcomputador. O equipamento executa cerca de 25 testes com implementação de diferentes algoritmos, incluindo algoritmos March, teste de varredura e medição de parâmetros DC do módulo, possibilitando alteração da alimentação do módulo. Os testes são comprados em bibliotecas e não é possível editá-los. Um software de interface de usuário possibilita que o mesmo selecione os testes necessários e os execute na ordem desejada (CST, 2015).

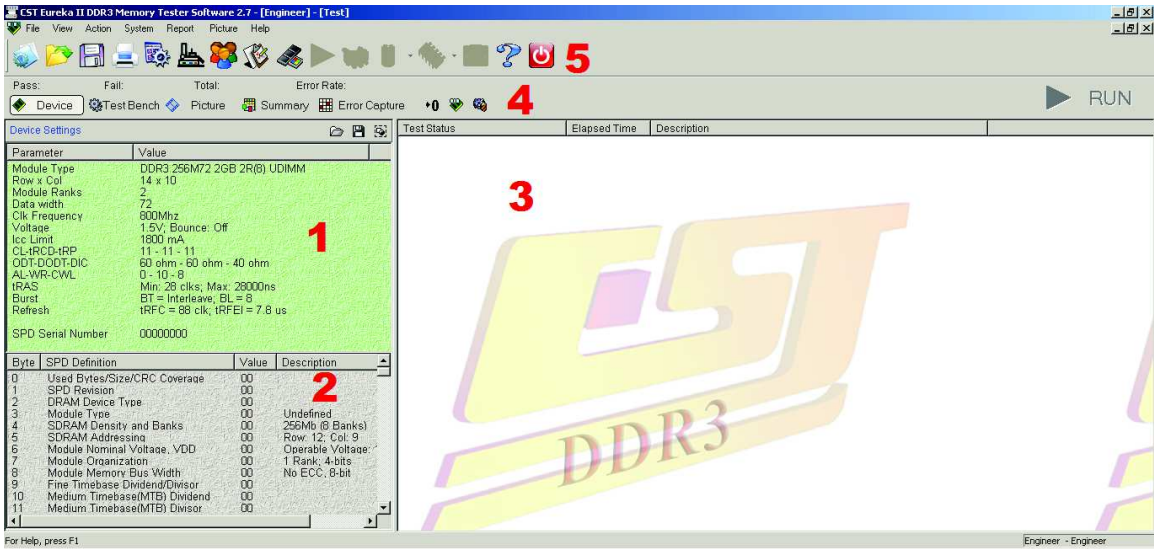
A Figura 19 exibe uma tela principal do aplicativo de interface com o Eureka2. Verifica-se na área indicada com o número ***1***, o ambiente em que o módulo a ser testado será identificado, sendo que suas características serão repassadas para o programa nessa área. Nesta sessão indica-se a frequência de operação do módulo, organização dos dados, tensão de alimentação e tempos de acesso do dispositivo (CST, 2009).

Ainda visível na Figura 19, na área com a indicação do número ***2***, exibem-se as informações lidas da memória não volátil que existe no módulo, chamada de *SPD Data*. Nela são salvos dados de identificação do fabricante do módulo bem como características de frequência e tensão de operação, além de registro de lote de produção. São informações fixas que identificam o módulo (CST, 2009).

Na área ***3*** da Figura 19 os testes realizados ao módulo estão listados. No item ***4*** um menu com funções de configuração permite navegar por funcionalidades do testador. Na área ***5***, ícones com utilidades do aplicativo estão disponíveis (CST, 2009).

Na Figura 20 exibe-se a tela capturada com a indicação de um programa de teste bem

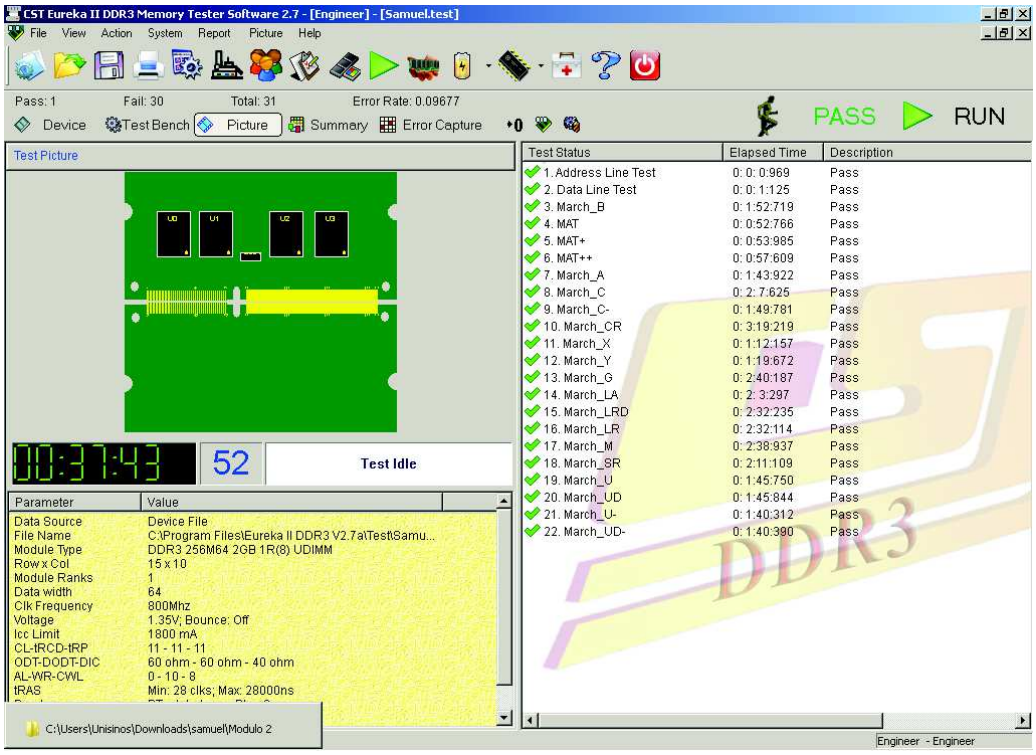
Figura 19 – Programa do Eureka2 da CST Inc.



Fonte – Elaborado pelo Autor.

sucedido. Observa-se a marcação em verde indicando que não ocorreram falhas no procedimento de teste. De acordo com as características do módulo, uma imagem representativa do mesmo pode ser criada.

Figura 20 – Detalhe da lista de teste que será executada.

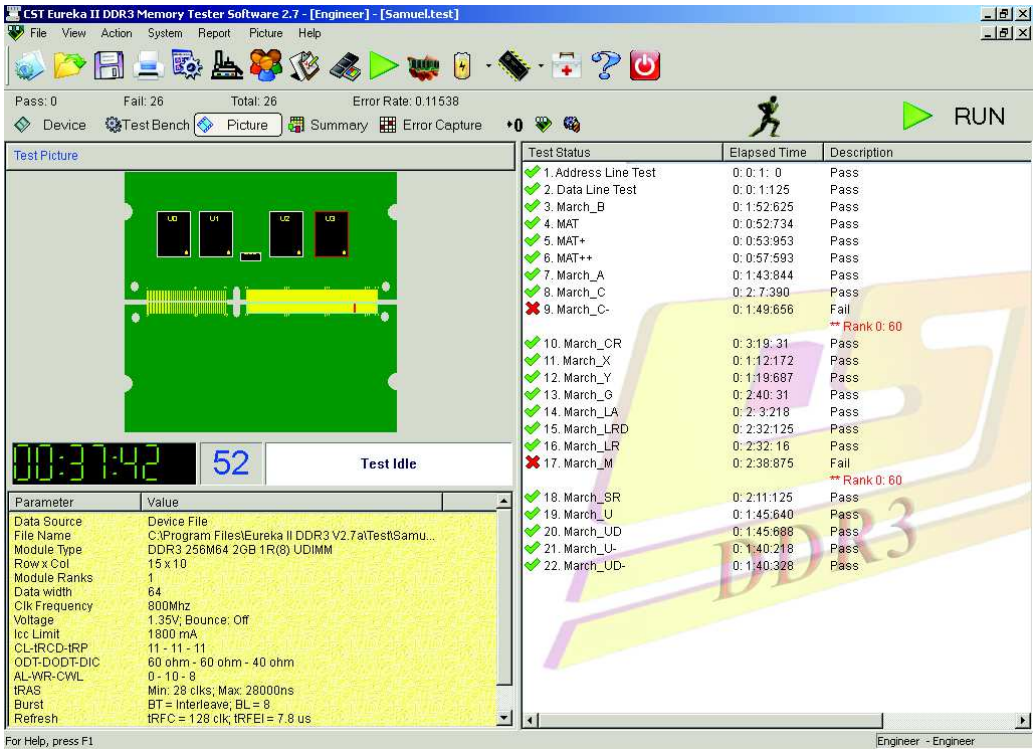


Fonte – Elaborado pelo Autor.

Na Figura 21 exibe-se a tela capturada com a indicação de um programa de teste que

apresentou falhas. Observa-se a marcação em vermelho indicando que ocorreram falhas no procedimento de teste para os algoritmos MARCH C- e MARCH M, no exemplo da figura 21. A imagem representativa do módulo também indica em vermelho a posição da linha de dado que apresentou leitura incorreta. Na Figura 22 exibe-se a tela capturada com a indicação em detalhes da linha de dado que apresentou falhas.

Figura 21 – Indicação de falha no algoritmo de teste.

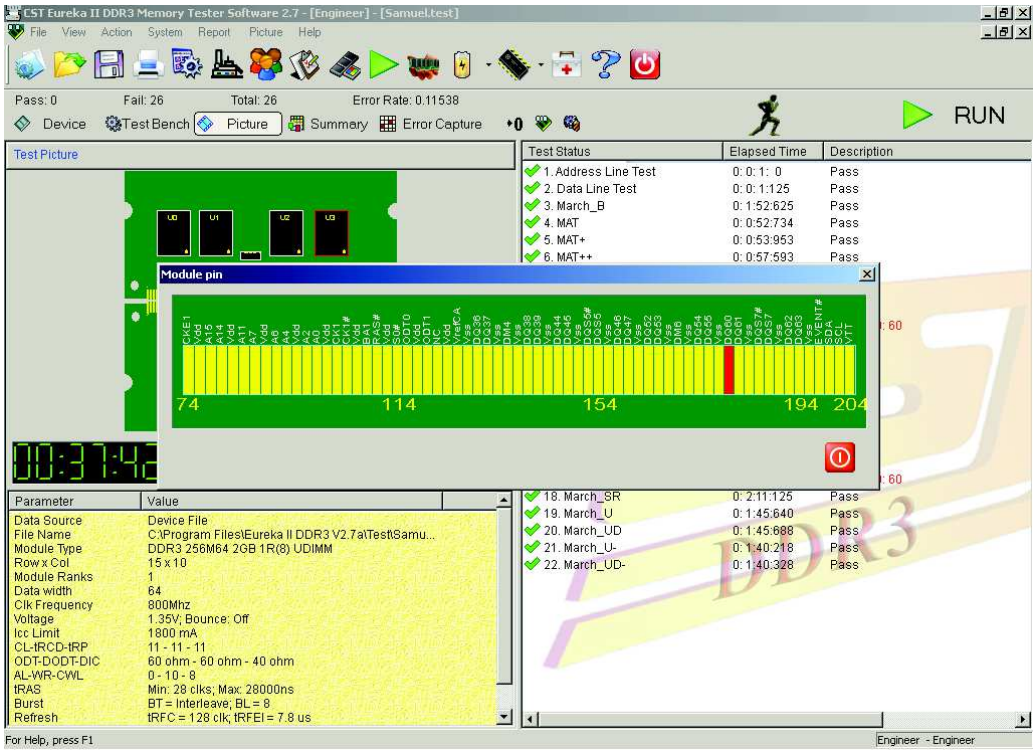


Fonte – Elaborado pelo Autor.

Na Figura 23 exibe-se a tela capturada com a indicação em detalhes do pino do circuito integrado de memória que apresentou falhas. Pode-se verificar pelas imagens das Figuras 21, 22 e 23 que não existe uma indicação sobre o endereço em que a falha ocorre, desta forma não é possível determinar se esta resultou de um defeito de fabricação na placa de circuito impresso do módulo ou se refere a um defeito no próprio circuito integrado. O equipamento Eureka2® apenas informa o pino em que foi observada a falha de leitura.

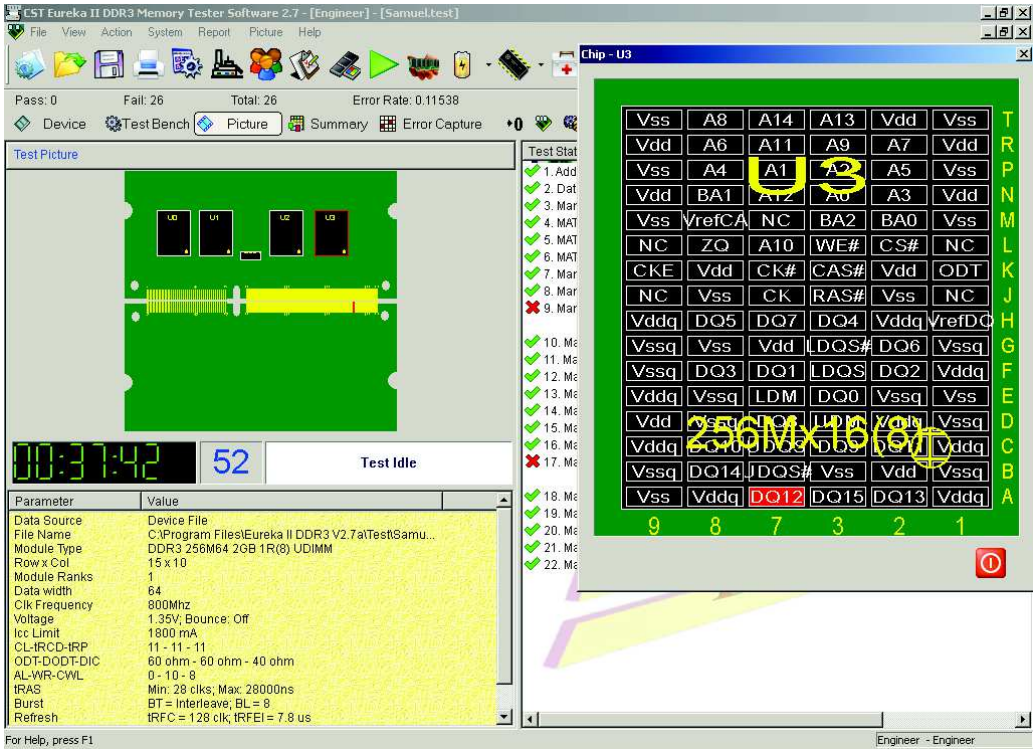
Uma função relevante em teste de circuitos integrados consiste na criação do diagrama assim chamado de *Shmoo* que representa uma maneira de caracterizar um cicuito integrado frente a variações. Podem-se avaliar mudanças em condições ambientais, como temperatura ou pressão de operação, além de alterações nas circunstâncias de funcionamento, como desvios na frequência de trabalho ou tensões de alimentação do dispositivo (BAKER; BEERS, 1997). A função *Shmoo* desenvolvida pelo Eureka2 é ilustrada na Figura 24, onde pode-se alterar tensão de alimentação e frequência de operação do módulo.

Figura 22 – Detalhe do pino do módulo que falhou



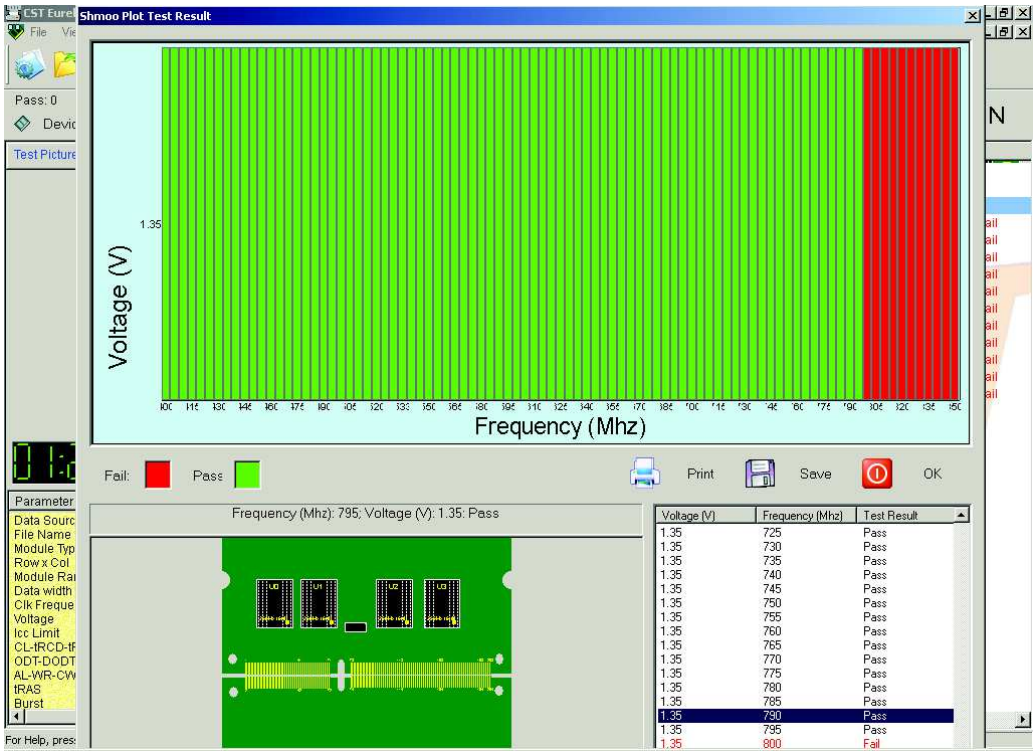
Fonte – Elaborado pelo Autor.

Figura 23 – Detalhe da linha de dados que falhou.



Fonte – Elaborado pelo Autor.

Figura 24 – Função *Shmoo* existente.

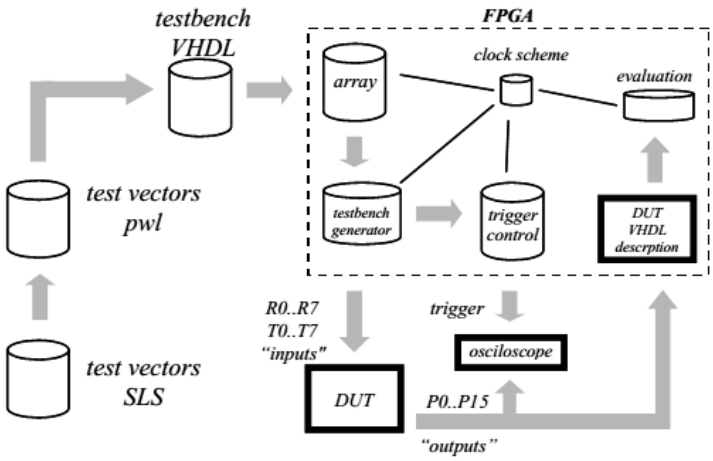


Fonte – Elaborado pelo Autor.

2.8 Trabalhos Correlatos

Oliveira et al. (2005) propôs um sistema de teste de baixo custo que utiliza um FPGA para teste de circuitos digitais. O sistema proposto é exibido na Figura 25.

Figura 25 – Metodologia empregada por Oliveira et al. (2005).



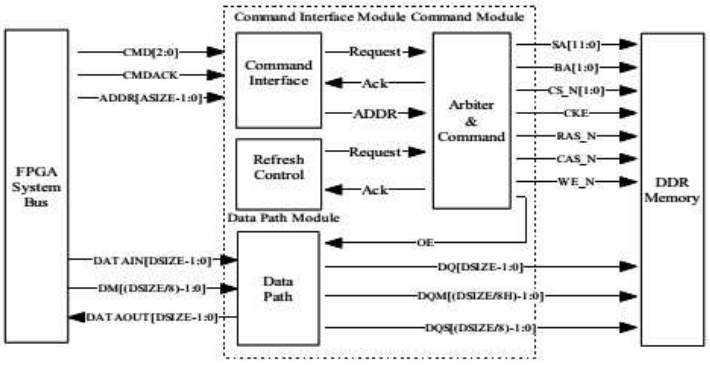
Fonte – Adaptado de (OLIVEIRA et al., 2005)

No trabalho de Oliveira et al. (2005), um vetor de teste é sintetizado e gravado no FPGA, que os aplica ao dispositivo em teste. O FPGA também executa a leitura dos padrões de saída para

comparação e gera sinais de sincronismo utilizados no osciloscópio com função de analisador lógico. O osciloscópio possibilita visualizar as formas de onda, no entanto acrescenta mais um equipamento ao testador. O trabalho ilustra como um FPGA pode ser utilizado para implementar um testador.

Zhou, Cheng e Liu (2006) sintetizam um controlador de memória DDR no interior de um FPGA para permitir aquisição de dados analógicos em altas velocidades. O trabalho de Zhou, Cheng e Liu (2006) possui relevância no âmbito de exemplificar uma aplicação onde grandes taxas de informação são necessárias, pois o dispositivo FPGA, além de gerenciar conversores AD com aquisições de 250 Milhões de Amostras por Segundo, *Million Samples Per Second* (MSPS), acessa a memória DDR RAM, controlando a mesma. Um diagrama de blocos do controlador é apresentado na Figura 26.

Figura 26 – Controlador desenvolvido por Zhou, Cheng e Liu (2006).

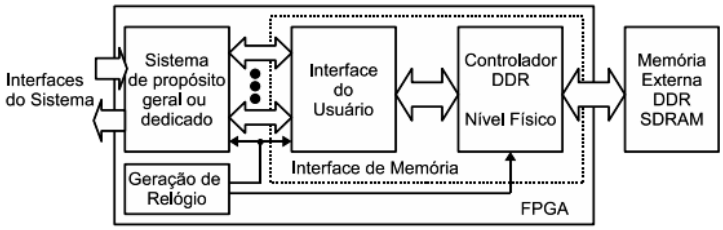


Fonte – Zhou, Cheng e Liu (2006)

Bonatto (2009) desenvolveu um controlador de memória DDR para Sistemas-Em-Chip. A solução de Bonatto (2009) é apresentada na Figura 27. Naquele trabalho o autor utilizou um kit de desenvolvimento do modelo Digilent XUPV2P que serve de base para a implementação. Nesta plataforma, um FPGA da familia Virtex-II Pro® acessa um módulo de memória DDR SDRAM. No trabalho de Bonatto (2009) o objetivo era o desenvolvimento de um controlador de memória DDR em um formato que possa ser reutilizado para Sistemas-Em-Chip. Para tanto, Bonatto (2009) implementou um núcleo como propriedade intelectual (*Intellectual Property* - IP). A implementação de IP possibilita o reúso de módulos de hardware e com isso possibilita a economia de tempo em desenvolvimento de sistemas digitais de grande porte. O uso de IP também garante o sigilo do hardware implementado e constitui em um produto, pois o IP desenvolvido pode ser comercializado, gerando receitas (BONATTO, 2009).

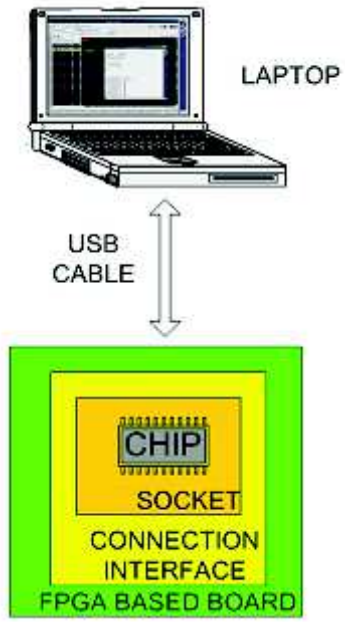
Mostardini et al. (2009) utilizou um FPGA para construção de um ATE de baixo custo. Uma visão geral da solução apresentada por Mostardini et al. (2009) é apresentada na Figura 28. Como pode ser observado na Figura 28, o sistema é composto de um computador que realiza a transferência dos padrões de teste por meio de uma interface USB para o FPGA, que os aplica ao componente em teste (MOSTARDINI et al., 2009).

Figura 27 – Controlador desenvolvido por Bonatto (2009).



Fonte – Bonatto (2009)

Figura 28 – Sistema desenvolvido por Mostardini et al. (2009).



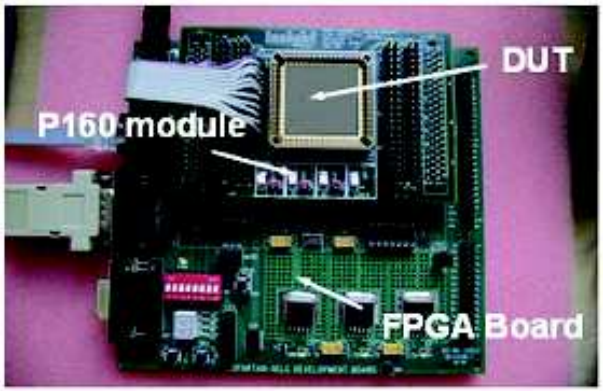
Fonte – Mostardini et al. (2009)

Segundo Mostardini et al. (2009), um testador baseado em FPGA oferece uma maneira de baixo custo para realizar uma validação de padrões de teste, desenvolvimento de programas de teste, depuração ou análise de falhas, proporcionando alta flexibilidade e portabilidade além de reduzido tempo de implementação. O protótipo desenvolvido por Mostardini et al. (2009) é exibido na Figura 29.

Mostardini et al. (2009), utilizou uma placa de desenvolvimento DS-KIT-3SMB1500-EURO da empresa Memec InsightTM®, com um FPGA Xilinx® Spartan 3, modelo XC3S1500-4FG676C. Uma placa auxiliar foi conectada ao kit, assim permitindo prototipagem e teste de dispositivos externos, além de simplificar as conexões físicas. O FGPA implementa 4 máquinas de estados para controlar o fluxo dos testes, além de uma interface serial e três bancos de memória. A máquina de estados 1 é responsável pelo recebimento dos padrões que serão aplicados no Dispositivo Em Teste, do inglês, *Device Under Test* (DUT).

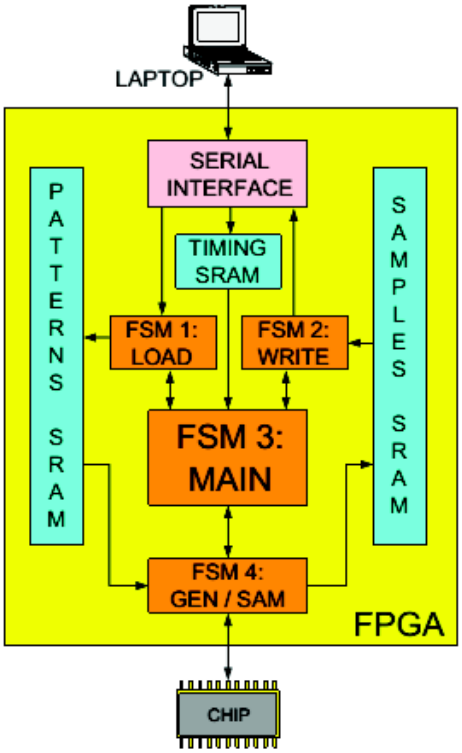
Os dados são recebidos pelo módulo que implementa uma interface serial. Os padrões de

Figura 29 – Protótipo desenvolvido por Mostardini et al. (2009).



Fonte – Mostardini et al. (2009).

Figura 30 – Blocos de Hardware no interior do FPGA por Mostardini et al. (2009).

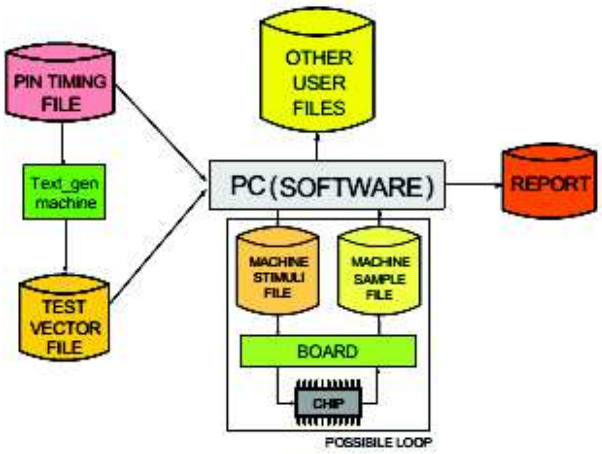


Fonte – Mostardini et al. (2009)

teste são salvos em uma memória SRAM sintetizada no FPGA. Um vetor contendo informações sobre a temporização dos padrões também é recebido via comunicação serial. No momento de aplicar os testes, a máquina de estados 4 resgata a informação de temporização e padrões necessário. Ela realiza o teste do dispositivo e captura as saídas, sendo escritas em uma SRAM dedicada. A máquina de estados 2 é responsável pela transmissão dos resultados ao computador. O controle das operações é implementado pela máquina de estados 3 (MOSTARDINI et al., 2009).

Segundo Mostardini et al. (2009), os padrões são divididos em várias partes e em seguida são aplicados em ciclos consecutivos. Quando o último padrão é aplicado, encerrando o último ciclo, o software gera um arquivo de relatório. Na Figura 31, o processo como um todo é exibido. Mostardini et al. (2009) não detalha o funcionamento das funções realizadas pelo software no computador.

Figura 31 – Sistema desenvolvido por Mostardini et al. (2009).



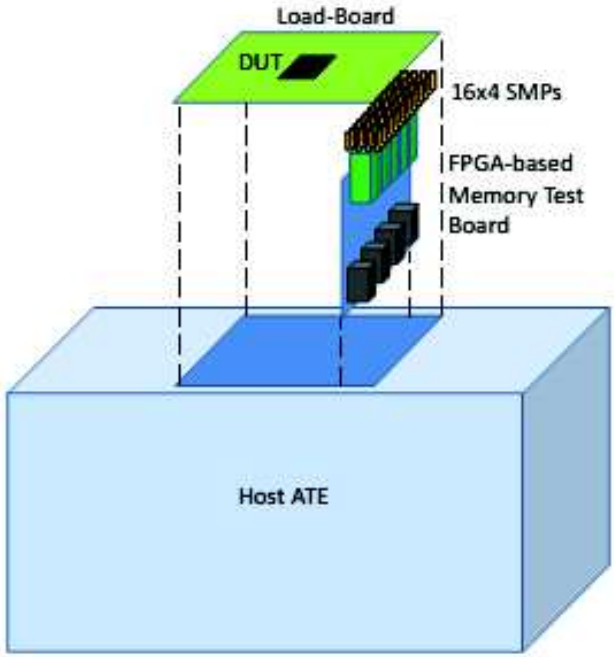
Fonte – Mostardini et al. (2009)

Keezer et al. (2015) criou um módulo de extensão para um ATE. O módulo desenvolvido utiliza um FPGA para realizar uma atualização em um ATE de baixo custo. O módulo é acoplado em um ATE já existente e com isso possibilita o aumento da vida útil do equipamento, além de permitir o teste de memórias de alta velocidade. A solução desenvolvida é ilustrada na Figura 32. O módulo aproveita o cabeçote de conexões de um ATE de baixo custo, desta forma a placa de teste de memória é inserida como um módulo de encaixe no sistema. As interfaces com o DUT são fornecidas utilizando-se conexões apropriadas. Desta forma, sinais de uso geral e de energia, além de sinais de sincronização são fornecidos pelo hospedeiro da ATE. Para tornar a extensão possível, espera-se que o cartão de memória seja capaz de conduzir o teste de forma autônoma. O arranjo pode ser estendido para permitir paralelismo no teste (KEEZER et al., 2015). Na Figura 33, a placa desenvolvida por Keezer et al. (2015) é apresentada. Na confecção da placa, utiliza-se um FPGA da família Xilinc Kintex-7®. Uma taxa máxima de transferência de dados de 5Gbps é utilizada em uma placa de cerca de 20 layers. Os autores não fazem menção sobre qual tipo de memória é possível testar com a placa.

2.9 Considerações do Capítulo

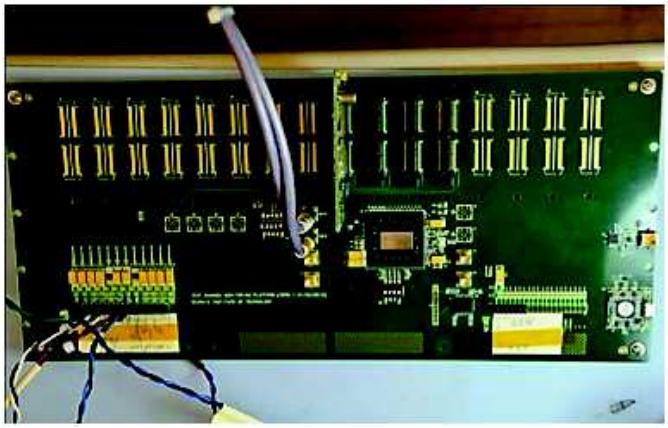
Conforme o exposto neste capítulo, viu-se uma introdução geral dos princípios do teste de sistemas eletrônicos e em especial o teste de memórias RAM.

Figura 32 – Sistema desenvolvido por Keezer et al. (2015).



Fonte – Keezer et al. (2015)

Figura 33 – Placa de teste desenvolvida por Keezer et al. (2015).



Fonte – Keezer et al. (2015)

Os trabalhos estudados indicam ser possível o interfaceamento de um FPGA em módulos de memória DDR3 SDRAM. A natureza flexível e paralela do FPGA torna possível seu uso em sistemas digitais de grande complexidade, como demonstrado na revisão. O desenvolvimento de diferentes plataformas de teste é também exposto. Estes trabalhos, exploram aspectos relacionados aos desenvolvidos nesta dissertação, mas com enfoques diversos. Em Oliveira et al. (2005) os autores propõem uma plataforma de baixo custo para teste de circuitos digitais. Já em Zhou, Cheng e Liu (2006) e Bonatto (2009), o desenvolvimento de controladores de memória DDR com uso de FPGAs é descrito e assim comprova a viabilidade de interface entre memórias DDR com estes dispositivos. O projeto de um Equipamento Automático de Teste baseado em FPGA

é apresentado em (MOSTARDINI et al., 2009). Em Keezer et al. (2015) a elaboração de um cartão de teste de memórias, que permite uma atualização em um Equipamento Automático de Teste é exposta. Isso atesta o uso de FPGAs em aplicações de altas velocidades de transferências. Os sistemas descritos não exploram os aspectos específicos desejados por este trabalho, como o desenvolvimento de um sistema portátil. Uma relação dos trabalhos correlatos, com os objetivos desta dissertação é apresentada na Tabela 5.

Tabela 5 – Relação dos trabalhos com essa dissertação.

Autores	**Trabalho realizado**	**Relação**
(OLIVEIRA et al., 2005)	Uma plataforma de teste	Motivação ao uso de FPGA
(ZHOU; CHENG; LIU, 2006)	Um controlador DDR	Conexão com memórias DDR
(BONATTO, 2009)	Um controlador DDR	Uso de IP
(MOSTARDINI et al., 2009)	Uma plataforma de teste	Definição na Metodologia
(KEEZER et al., 2015)	Um cartão de teste	Motivação ao uso de FPGA

Fonte – Elaborado pelo Autor.

3 METODOLOGIA

Este capítulo apresenta as ferramentas de software, kit de desenvolvimento e a metodologia, ou seja, a descrição dos passos seguidos para desenvolver este trabalho. Com base na fundamentação teórica realizada no Capítulo 2, obtém-se elementos para seu entendimento. A seguir o esquema geral da metodologia é exibido, seguido das etapas para sua realização e validação.

3.1 Esquema Geral da Metodologia

A Figura 34 apresenta o esquema geral para a solução do problema exposto. Observa-se que os módulos são inseridos na plataforma, um após o outro de maneira sequencial, sempre testados individualmente. Após a inserção, os algoritmos de teste são executados e os vetores de saída são capturados em um processo contínuo, enquanto existirem padrões de teste selecionados para execução. Esse processo é realizado para todos os módulos que se deseja testar. Nas próximas seções, descrevem-se as etapas de implementação da solução.

3.2 O Sistema Proposto

O sistema proposto é exibido na Figura 35, sendo composto de um FPGA responsável pela aplicação dos padrões de teste no módulo de memória sob teste, um módulo de memória DDR3 SDRAM no formato SODIMM e um computador é utilizado para captura das saídas do teste através de uma comunicação serial. Apresenta-se o sistema proposto na Figura 35 e detalha-se o sistema nos itens a seguir.

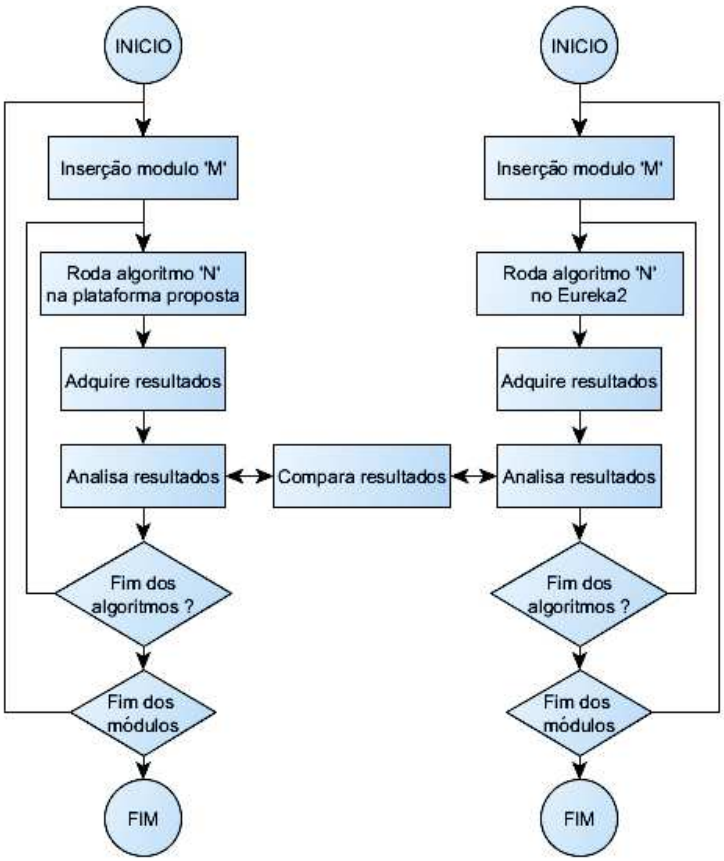
No desenvolvimento deste trabalho, utilizou-se FPGAs para acessar o módulo DDR3. O uso de FPGA torna a elaboração de sistemas digitais facilitada, graças a possibilidade de uso de núcleos de IP (BONATTO, 2009).

3.3 Concepção do Sistema

Na Figura 36 representa-se a concepção do sistema proposto. A comunicação com um microcomputador externo, referido como um *host*, é realizada por uma interface de comunicação necessária para a transferência de dados e adequação do padrão de sinalização elétrica, presente na porta de comunicação USB com o FPGA utilizado. A escolha por uma comunicação USB deve-se ao fato de a mesma ser amplamente difundida nos microcomputadores atuais.

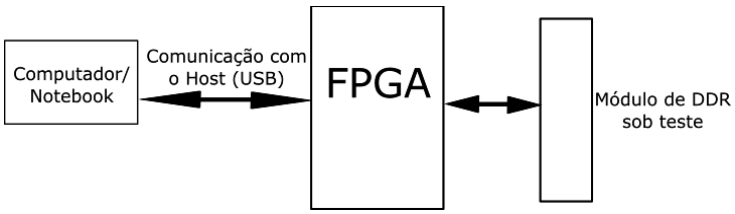
Conforme apresentado na Figura 36, a interface de comunicação entre o *host* e o FPGA é o meio de reportar erros de leitura encontrados pelos algoritmos de teste. Cria-se no FPGA

Figura 34 – Esquema Geral.



Fonte – Elaborado pelo Autor.

Figura 35 – Sistema Proposto

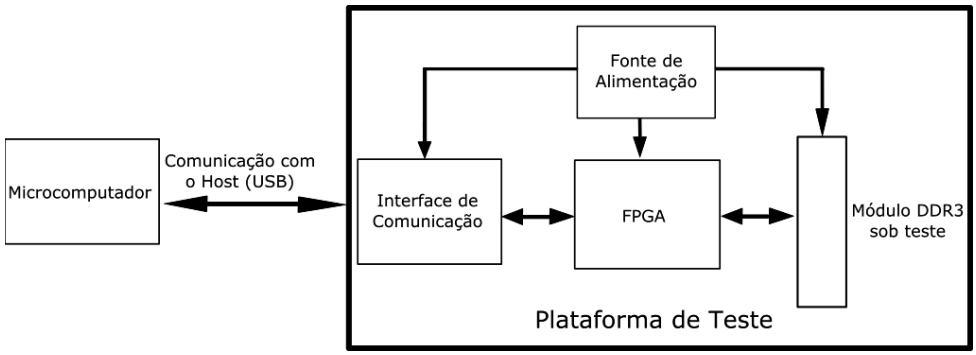


Fonte – Elaborado pelo Autor.

uma estrutura para ministrar os testes de memória, além de um núcleo controlador de memória DDR3 e demais mecanismos periféricos necessários ao propósito. O módulo DDR3 SDRAM é conectado diretamente ao FPGA por intermédio de portas destinadas a essa função. Um sistema provê alimentação elétrica adequada à operação dos diferentes elementos.

A representação do fluxo de informação no interior do FPGA é ilustrada na Figura 37. Observa-se o bloco do controlador DDR responsável pelo controle do módulo que realiza as operações físicas de escrita e leitura na memória em teste. O bloco do controlador DDR comunica-se com o bloco de controle de teste, o qual é o responsável pela execução dos algoritmos de teste implementados. O terceiro núcleo é o responsável pela lógica de comunicação de todo o

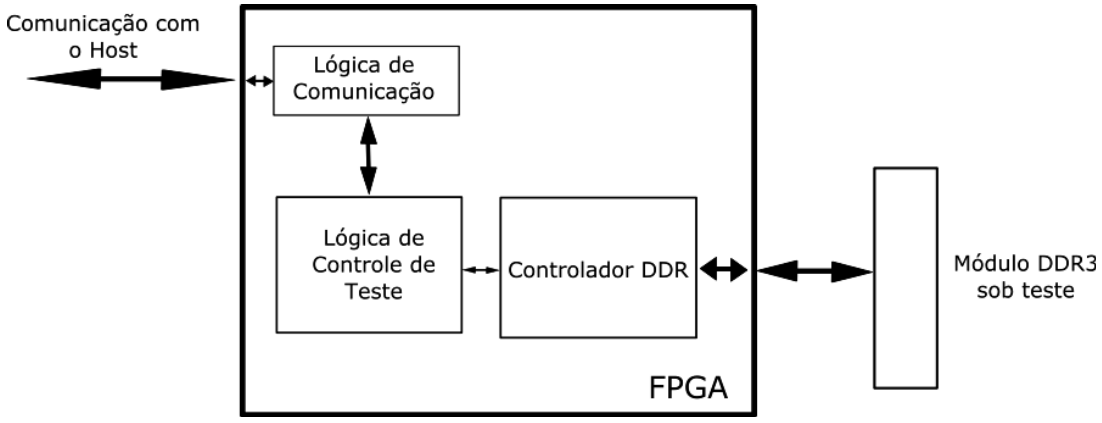
Figura 36 – Diagrama de Blocos da Solução



Fonte – Elaborado pelo Autor.

sistema, gerencia o protocolo de transferência com o computador, sendo encarregado pelo fluxo das informações de resultados.

Figura 37 – Representação do fluxo de informação no FPGA.



Fonte – Elaborado pelo Autor.

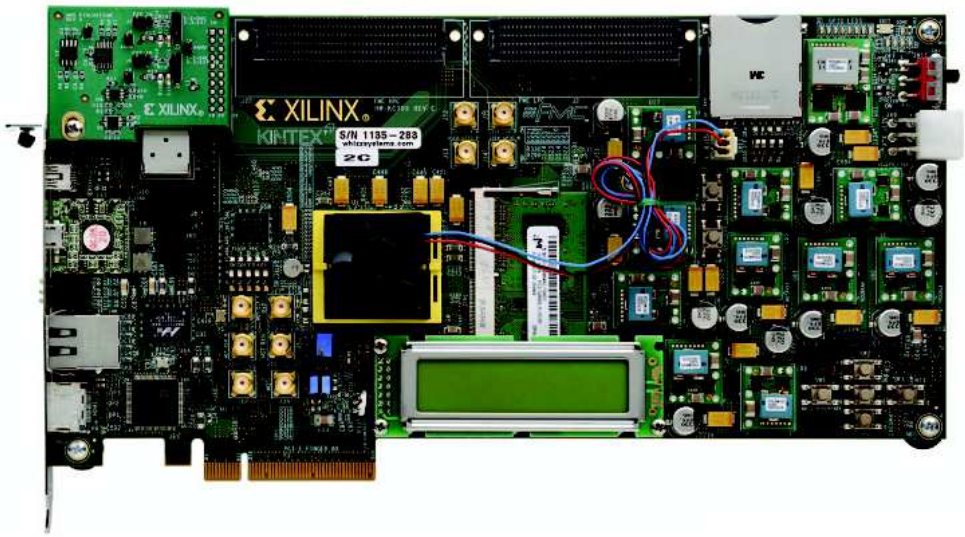
Um ambiente de interface no computador é utilizado para capturar os resultados dos testes. A ferramenta VIVADO® da empresa Xilinx foi utilizada para a síntese da arquitetura e a ferramenta Xilinx SDK foi empregada no desenvolvimento dos algoritmos de teste implementados.

3.4 Desenvolvimento do Sistema

O sistema proposto teve sua validação conceitual com o kit de desenvolvimento Kintex-7 FPGA KC705 Evaluation Kit, apresentado na Figura 38. A escolha por esse kit em particular justifica-se pelo fato de estar disponível uma interface para módulos de memória DDR3 SDRAM no formato SODIMM. Alia-se também a existência de um exemplar do mesmo no Laboratório de Prototipação Digital e Sistemas Embarcados (LaPSE), pertencente ao Instituto Tecnológico em Semicondutores (itt CHIP) da Universidade do Vale do Rio dos Sinos (UNISINOS). Neste sentido, pode-se utilizar esse kit para validação da metodologia sugerida, bem como construir

a codificação de algoritmos de teste com o propósito de permitir a validação funcional de memórias.

Figura 38 – Plataforma de Desenvolvimento KC705.



Fonte – Adaptado de (XILINX, 2014).

Segundo a documentação do kit Xilinx (2016), a plataforma de desenvolvimento Kintex-7 FPGA KC705 Evaluation Kit possui uma série de periféricos que a tornam capaz de ser base de projeto para muitos sistemas digitais, incluindo recursos de multimídia. Para essa dissertação, os recursos relevantes para a implementação são:

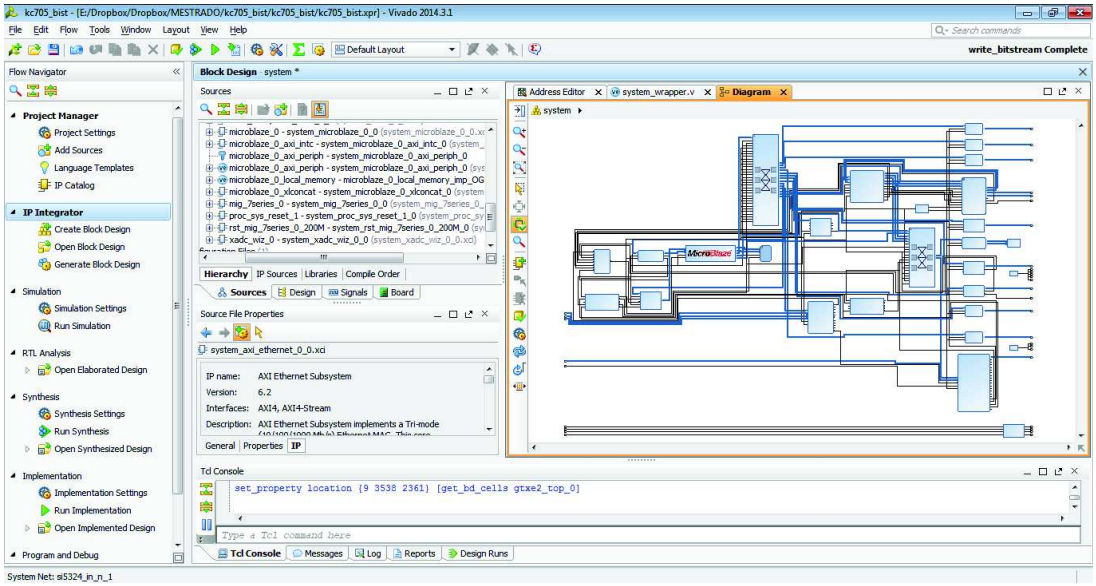
- Xilinx Kintex-7 XC7K325T-2FFG900C FPGA;
- Porta de Comunicação USB;
- Conexão de módulos DDR3 SODIMM em frequência de 800MHz;
- Sistema gerador de clock programável;
- Teclas e um display LCD para interface com usuário;

O sistema é composto por um microprocessador que realiza as operações dos algoritmos de teste, como por exemplo, o algoritmo March e, com isto, permite o teste da memória que está sendo avaliada.

A UNISINOS possui parceria com a empresa Xilinx por intermédio do Xilinx University Program (XUP). Por conta disso diversos núcleos de IP estão acessíveis a alunos e professores da universidade para desenvolvimento de pesquisas. Uma vez que o objetivo desta dissertação não é o desenvolvimento específico de um controlador de memória, a utilização de um núcleo controlador pronto mostrou-se vantajosa. Foi utilizada a ferramenta de desenvolvimento Xilinx

Vivado®, em sua versão 2014.3 disponível juntamente com a documentação do kit KC705, para síntese da arquitetura. A Figura 39 ilustra o ambiente.

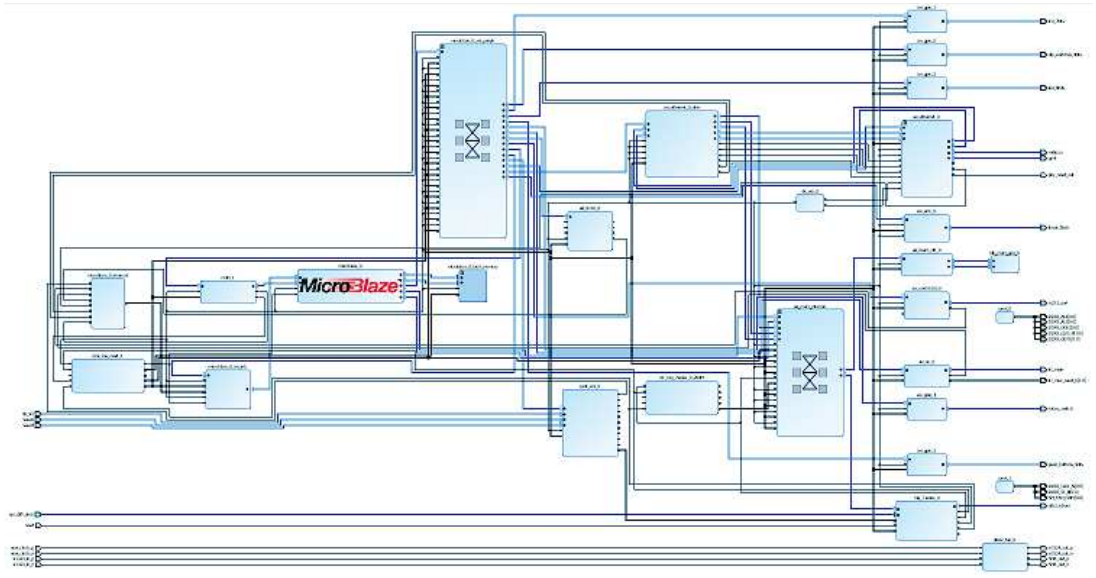
Figura 39 – Ferramenta Xilinx Vivado®.



Fonte – Elaborado pelo Autor.

Segundo a documentação da ferramenta Vivado® Xilinx (2014b), o ambiente permite a integração de núcleos IP disponíveis, o que agiliza a prototipação de sistemas digitais complexos. Nesse sentido utilizou-se nesta dissertação o IP do microprocessador MicroBlaze® com a configuração padrão para o kit KC705, utilizando-se o controlador de memória MIG. Na Figura 40 destaca-se a estrutura de processador utilizada.

Figura 40 – Arquitetura MicroBlaze®.

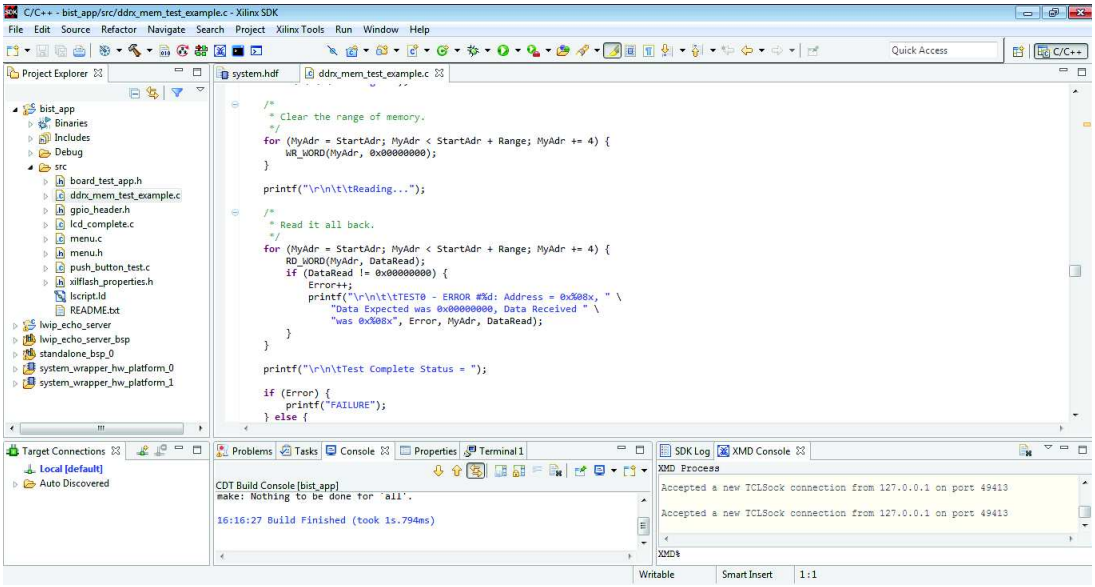


Fonte – Elaborado pelo Autor.

Conforme ilustrado na Figura 40, o sistema criado no ambiente Vivado® consiste na interligação de núcleos IP desenvolvidos em linguagem de descrição de hardware, cujas funcionalidades foram verificadas e validadas pelo desenvolvedor. Neste contexto, cada bloco da Figura 40 representa um núcleo IP e as linhas são as conexões lógicas entre eles.

O sistema exibido na Figura 40 deve ser sintetizado de modo que sua implementação física seja realiza. Com isso, pode-se fazer a exportação dos elementos de hardware para um ambiente de programação. A programação dos algoritmos e o uso dos periféricos da arquitetura é possível utilizando-se a ferramenta Xilinx SDK® que é apresentada na Figura 41. O uso de uma linguagem de programação permite fácil modificação dos algoritmos.

Figura 41 – Ferramenta Xilinx SDK®.



Fonte – Elaborado pelo Autor.

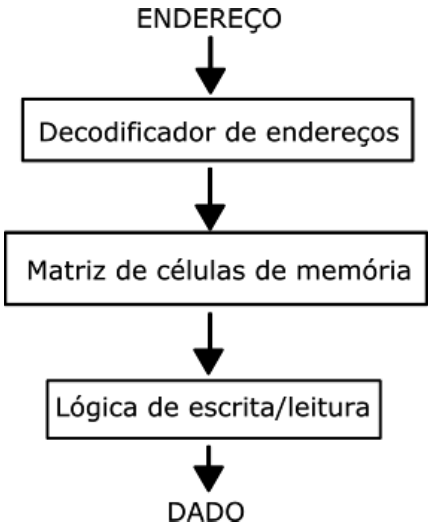
No aplicativo Xilinx SDK®, a linguagem de programação C é utilizada para implementação dos algoritmos de teste. Conforme apresentado no capítulo 2, os algoritmos de teste de memória foram selecionados mediante suas características de cobertura a falhas. Os algoritmos empregados nos testes realizados neste trabalho são: MAT, MAT+, MAT++, MARCH X, MARCH Y, MARCH B, MARCH C- e MARCH G, uma vez que também estão disponíveis no sistema de teste Eureka2®.

3.5 Validação do Sistema

A validação do sistema foi realizada a partir da comparação dos resultados dos testes dos módulos, anteriormente testados pelo equipamento Eureka2®, com a plataforma aqui desenvolvida. O modelo funcional de memória é utilizado para executar o teste (GOOR, 1993). A Figura 42 ilustra o modelo utilizado nesta dissertação. Segundo Goor (1993), esse modelo abstrai os diversos mecanismos internos de uma memória para um conjunto de blocos funcionais

elementares. Neste sentido, o tipo de memória, a topologia de circuitos utilizada e a tecnologia empregada em sua construção são abstraídos no momento do teste. Este modelo é utilizado no teste funcional em produção.

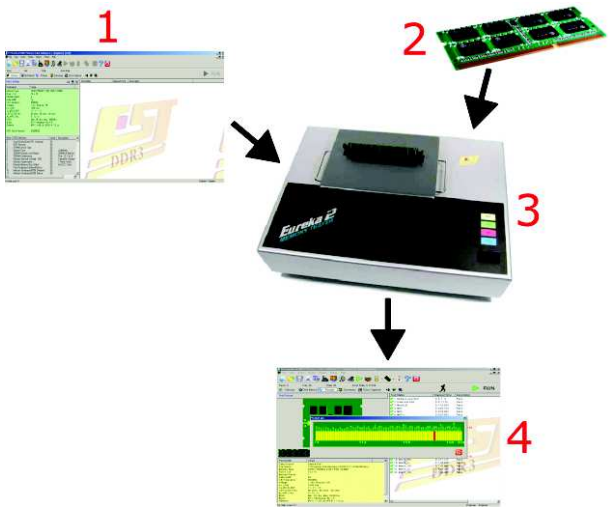
Figura 42 – Modelo funcional reduzido para memórias.



Fonte – Elaborado pelo Autor.

Na Figura 43, ilustra-se o fluxo das etapas de validação dos módulos testados. Os testes executados foram apresentados na seção anterior. Conforme apresentado na figura na imagem marcada com o número *1*, os testes são selecionados no aplicativo do testador, na imagem marcada com o número *2*. A seguir os módulos disponíveis para teste são inseridos. Na sequência, na imagem sinalizada com o número *3*, o Eureka2® aplica os vetores de teste, gerando os resultados ilustrados no número *4*.

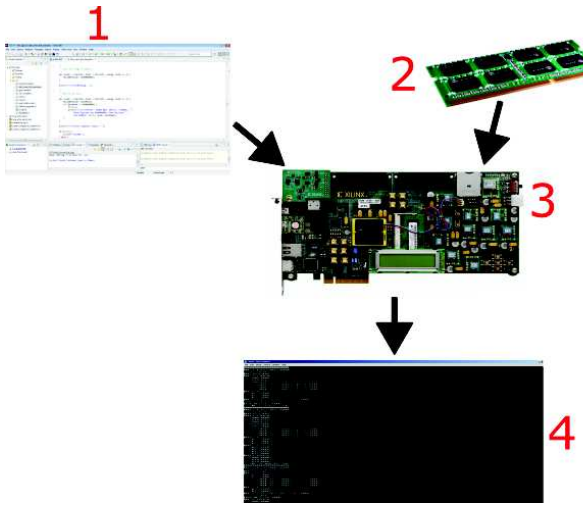
Figura 43 – Procedimento realizado com o testador Eureka2®.



Fonte – Elaborado pelo Autor.

O processo de validação dos módulos na plataforma proposta é ilustrado na Figura 44, seguindo-se praticamente os mesmos passos que a validação no Eureka2®. Verifica-se o fluxo das etapas com a indicação dos números. Os algoritmos de teste são os mesmos executados no Eureka2®. Conforme visualizado na figura através da imagem marcada com o número *1*, os testes são implementados no ambiente Xilinx SDK®. Na imagem marcada com o número *2* os módulos já testados no Eureka2® são inseridos. Na imagem sinalizada com o número *3*, a plataforma aplica os vetores de teste, gerando os resultados ilustrados no número *4*, por fim exibidos na tela de um aplicativo de terminal.

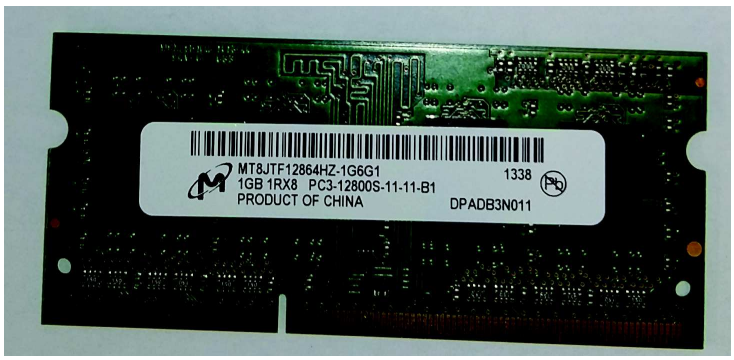
Figura 44 – Procedimento realizado na plataforma desenvolvida.



Fonte – Elaborado pelo Autor.

O módulo disponível no kit KC705 é do modelo MT8JTF12864HZ, fabricado pela empresa MICROM®. Consiste em um módulo de memória SODIMM com 1GB de capacidade, velocidade de trabalho em 800MHz, na configuração de 128Mx64 (MICRON, 2010). Utiliza-se esse módulo para validação da plataforma, realizando-se simulações de falhas em suas linhas de dados. A Figura 45 apresenta o módulo.

Figura 45 – Módulo de memória DDR3 SODIMM testado.



Fonte – Elaborado pelo Autor.

3.6 Considerações do Capítulo

Neste capítulo descreveram-se os passos para o desenvolvimento da plataforma de teste proposta. No que se trata de ferramentas de prototipação os ambientes Xilinx Vivado® e SDK® permitem foco na construção do trabalho pelo uso de IP's. Destaca-se nesta seção a metodologia utilizada para validar a plataforma proposta através da comparação com o equipamento comercial que realiza a mesma tarefa. No capítulo 4, os resultados obtidos nos testes são expostos.

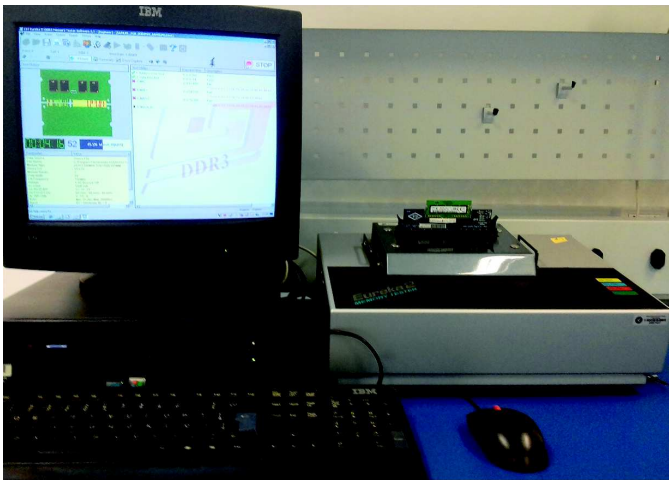
4 RESULTADOS

Neste capítulo apresentam-se os resultados obtidos a partir da metodologia exposta no capítulo 3, utilizada para desenvolver esta dissertação. Uma seção de considerações e análises sobre o capítulo discute os resultados levantados no trabalho.

4.1 Caracterização do Módulo pelo Testador Eureka2®

Conforme capítulo anterior, objetivando-se verificar o funcionamento do módulo utilizado para teste, uma bateria de testes com o testador Eureka2® foi realizada. Com isso, pode-se descobrir se o módulo é funcional, baseados nos resultados dos algoritmos de teste, expostos no capítulo anterior. Visualiza-se na Figura 46 o sistema Eureka2® em funcionamento, onde reportam-se as falhas detectadas.

Figura 46 – Experimento realizado no testador Eureka2®.

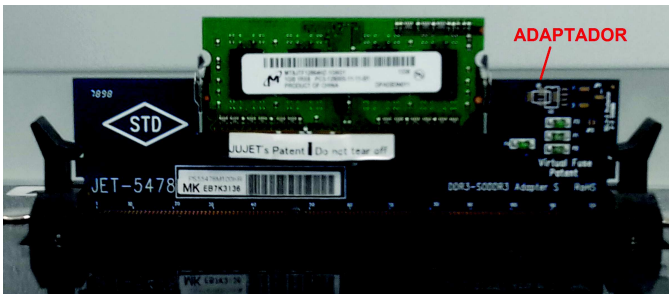


Fonte – Elaborado pelo Autor.

De acordo com a Figura 46, e como reportado no capítulo anterior, o módulo é inserido no testador, que executa a sequência de teste selecionada pelo usuário. No entanto é necessário o uso de um adaptador do padrão DIMM, existente no cabeçote do Eureka2®, para o padrão SODIMM, de maneira a permitir o teste. O adaptador é visualizado na Figura 47.

A saída dos resultados do testador é apresentada em uma tela que informa sobre a existência de falhas de execução no processo, conforme ilustra a Figura 48. Neste sentido, apenas informações qualitativas sobre a memória em teste são apresentadas, de forma a indicar se existiu detecção de falhas ou se o algoritmo foi executado de modo livre de falhas. A notação utilizada indica a palavra em inglês *Pass*, para indicar sucesso, seguido de uma sinalização na cor verde e a palavra em inglês *Fail* para indicar falha na execução, juntamente com a sinalização em vermelho. A informação sobre endereço não é disponibilizada pelo testador Eureka2®, apenas

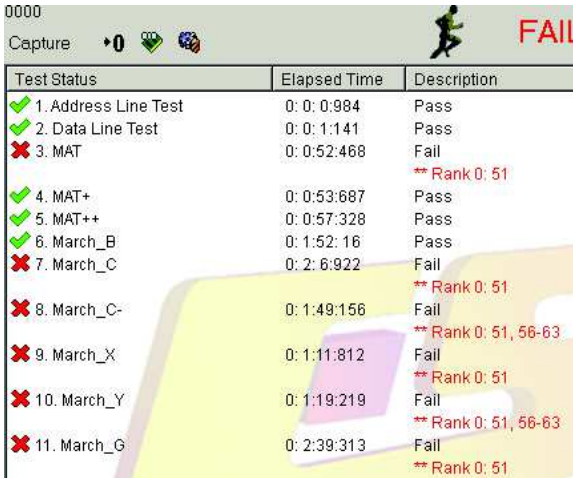
Figura 47 – Adaptador necessário para inserção do módulo.



Fonte – Elaborado pelo Autor.

as linhas de dados que apresentam erros na execução são indicadas, desta forma dificulta-se uma avaliação quantitativa da cobertura de falhas, limitando-se a indicação da existência de erros de leitura e aos pinos de saída afetados.

Figura 48 – Apresentação dos resultados no sistema Eureka2®.



Fonte – Elaborado pelo Autor.

Nas Tabelas 6, 7 e 8, apresentam-se os resultados dos testes no testador Eureka2® para o módulo em estudo. A fim de determinar o comportamento do módulo, o teste foi realizado em diferentes frequências de operação, mantendo-se fixa a tensão de alimentação do módulo em 1,5V sendo a mesma disponível na plataforma desenvolvida. A fim de contornar variáveis aleatórias relacionadas ao ambiente, o módulo foi submetido a uma série de 5 testes, em cada uma das frequências indicadas nas tabelas. Desta forma permitiu-se o levantamento de influencias externas para uma melhor confiabilidade nos resultados levantados, uma vez que, manteve-se o ambiente da sala climatizado em temperatura de 20 ^{o}C, permanecendo-se assim por todos os testes.

Por intermédio das informações levantadas nos testes com o testador Eureka2® e expostas nas Tabelas 6, 7 e 8, verifica-se que o módulo disponível apresenta falhas no funcionamento em frequências elevadas de trabalho. Os algoritmos foram selecionados com

Tabela 6 – Testes do módulo MT8JTF12864HZ em diferentes frequências (MHz).

Algoritmo	400 serie					450 serie					500 serie				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
MATS	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
MATS+	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
MATS++	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
MARCH B	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
MARCH C-	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
MARCH X	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
MARCH Y	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
MARCH G	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o

Fonte – Elaborado pelo Autor.

Tabela 7 – Testes do módulo MT8JTF12864HZ em diferentes frequências (MHz).

Algoritmo	550 serie					600 serie					650 serie				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
MATS	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
MATS+	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
MATS++	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
MARCH B	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
MARCH C-	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
MARCH X	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
MARCH Y	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
MARCH G	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o

Fonte – Elaborado pelo Autor.

base na fundamentação apresentada, de maneira a criar uma bateria de testes capazes de detectar as falhas mais comuns em memórias. Visto isso, a presença de defeitos físicos será detectada pelos testes, assim pode-se verificar que o não funcionamento em determinadas frequências de operação deve-se ao não cumprimento dos tempos de propagação dos sinais de acesso a memória, uma vez que, apenas em elevadas frequências foi verificado presença de falhas. Para manter a simplicidade na exibição das informações, a indicação *Fail* foi substituída pelo caractere *'x'* e informa que ao menos uma linha de dados na saída apresentou valor incorreto durante a etapa de execução dos algoritmos. De maneira complementar a indicação *Pass* foi substituído pelo caractere *'o'* e indica que não foram verificadas inconsistências.

Tabela 8 – Testes do módulo MT8JTF12864HZ em diferentes frequências (MHz).

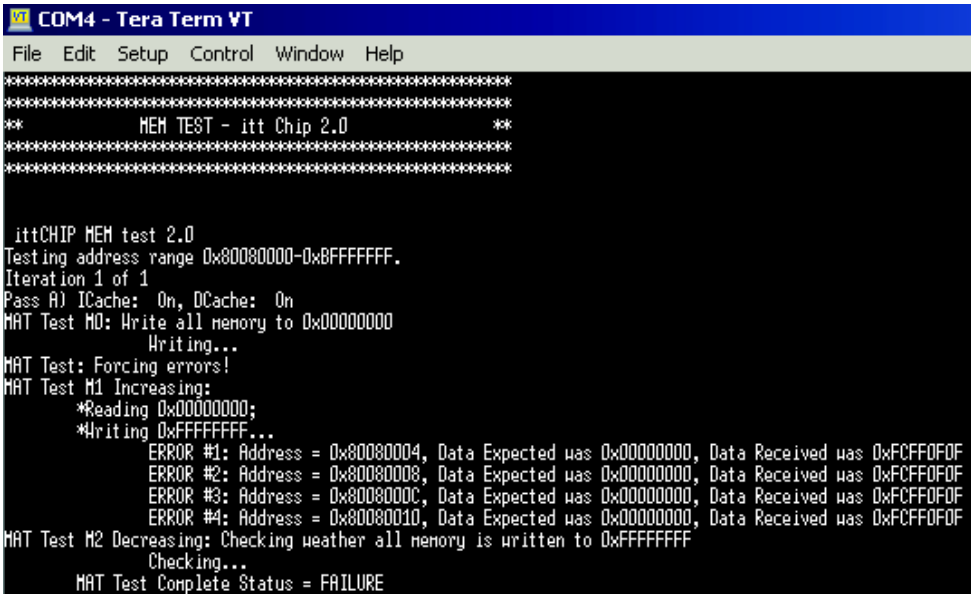
Algoritmo	700 serie					750 serie					800 serie				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
MATS	o	o	o	o	o	o	x	x	x	x	x	x	x	x	x
MATS+	o	o	o	o	o	o	x	x	x	x	x	x	x	x	x
MATS++	o	o	o	o	o	o	x	x	x	x	x	x	x	x	x
MARCH B	o	o	o	o	o	o	x	x	x	x	x	x	x	x	x
MARCH C-	o	o	o	o	o	o	x	x	x	x	x	x	x	x	x
MARCH X	o	o	o	o	o	o	x	x	x	x	x	x	x	x	x
MARCH Y	o	o	o	o	o	o	x	x	x	x	x	x	x	x	x
MARCH G	o	o	o	o	o	o	x	x	x	x	x	x	x	x	x

Fonte – Elaborado pelo Autor.

4.2 Desenvolvimento da Interface com o Usuário

Na Figura 49, ilustra-se a saída de dados no aplicativo de terminal utilizado para interface com a plataforma desenvolvida, na qual verifica-se que, apesar da baixa interatividade com o sistema criado, informações mais detalhadas são fornecidas como o endereço em que os erros de leitura são encontrados.

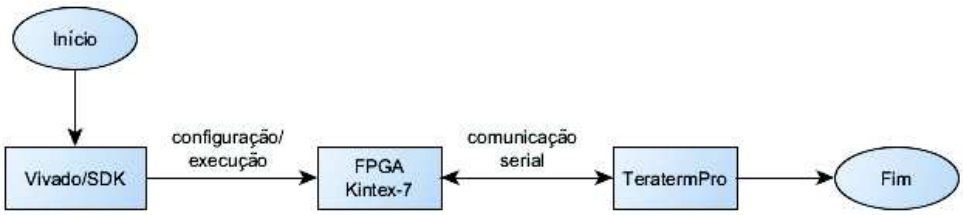
Figura 49 – Primeira ferramenta de interface utilizada.



Fonte – Elaborado pelo Autor.

O fluxo de solução da primeira proposta com interface em ambiente terminal, anteriormente exibido na Figura 49, é descrito na Figura 50. Verifica-se que a utilização das ferramentas Vivado® e SDK® ocorre de forma manual, atuando-se diretamente na plataforma desenvolvida que transmite vetores de saída para o aplicativo terminal.

Figura 50 – Fluxo da primeira ferramenta.

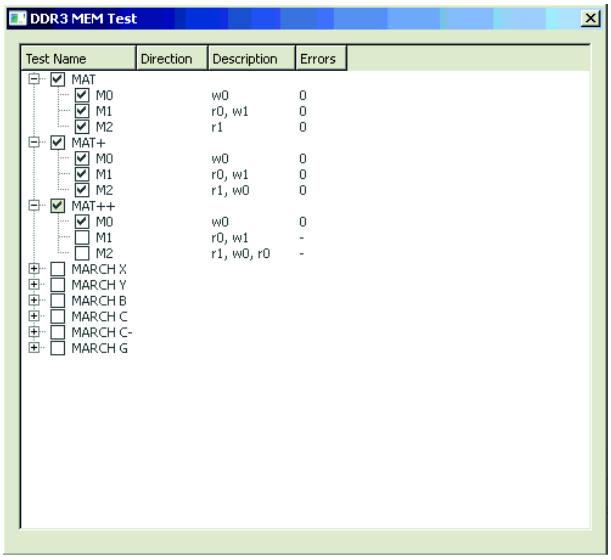


Fonte – Elaborado pelo Autor.

Neste sentido, ao longo do desenvolvimento desta dissertação, verificou-se que esse processo de coleta de resultados por intermédio do aplicativo terminal era bastante ineficiente, demandando grande tempo. Viu-se com isso a necessidade de elaboração de uma interface com o usuário, de modo a facilitar a operação do sistema proposto.

Neste contexto, uma interface mais intuitiva foi desenvolvida para facilitar a função de captura e análise dos dados recebidos. A Linguagem de Programação Python foi utilizada para construir uma aplicação de captura das informações. Na Figura 51, exibe-se a solução.

Figura 51 – Segunda ferramenta de interface desenvolvida.



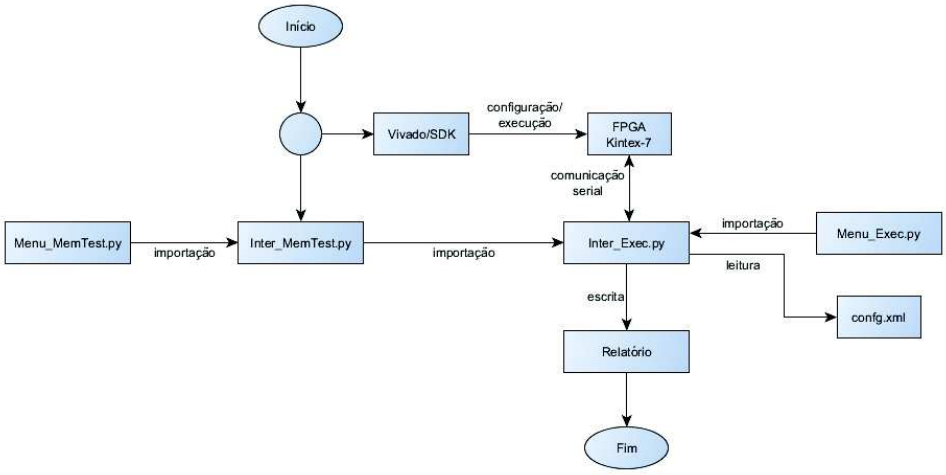
Fonte – Elaborado pelo Autor.

Com a ferramenta apresentada na Figura 51, pode-se coletar os dados de maneira mais eficiente, uma vez que a solução permite a criação de *logs* de informações ao final da execução. Essa solução mostrou-se eficaz na automatização do processo de aquisição.

O fluxo desta segunda ferramenta, anteriormente exibida na Figura 51, é visualizado na Figura 52. Observa-se na Figura 52 que as ferramentas Vivado® e SDK® ainda são manualmente acessadas, mas rotinas em linguagem de programação Python foram criadas e assim possibilitam a automação do processo de aquisição de resultados. Neste sentido, o arquivo

Inter_Exec.py evoca as rotinas auxiliares de criação de menus e controle sobre a porta serial, ainda sendo responsável pela criação do relatório de eventos.

Figura 52 – Fluxo da segunda Ferramenta.



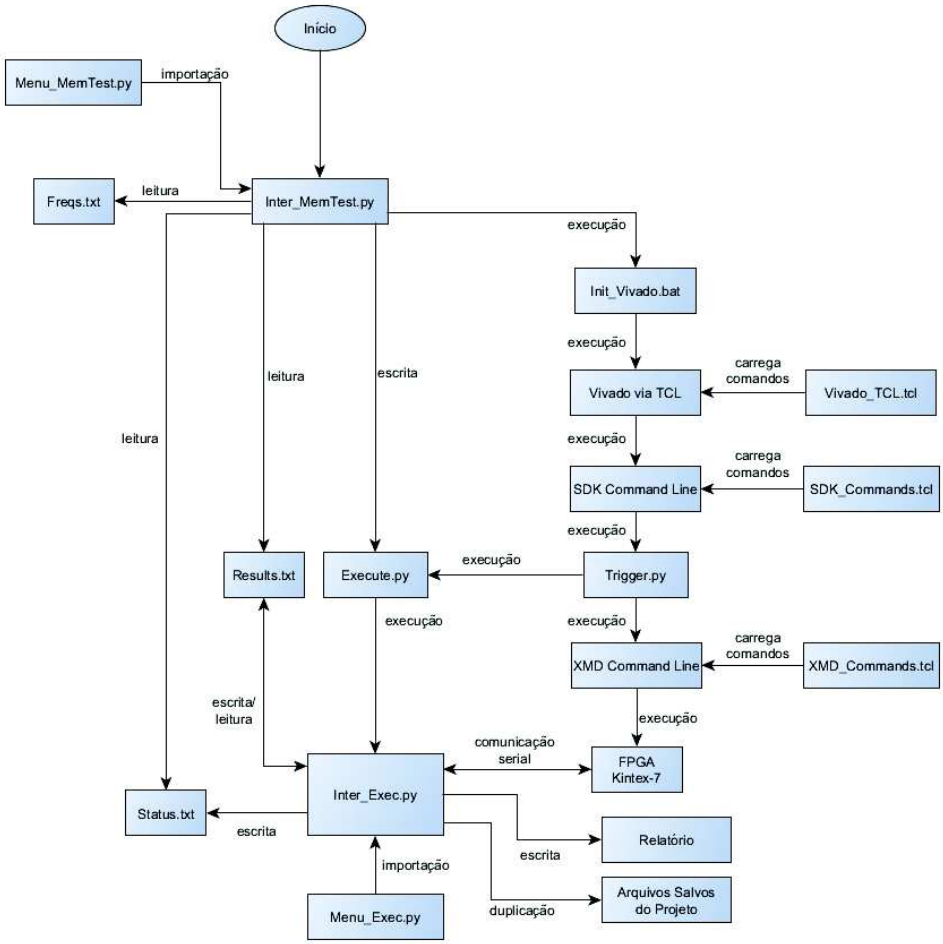
Fonte – Elaborado pelo Autor.

A fim de permitir que a plataforma desenvolvida obtivesse similar grau de funcionalidade apresentado pelo testador comercial, a função de varredura em frequência foi adicionada como opção de teste. Neste sentido, também se avaliou os módulos na plataforma com diferentes velocidades de acesso à memória. As mesmas frequências testadas no testador Eureka2® foram utilizadas.

Neste sentido, uma terceira ferramenta foi desenvolvida utilizando-se de *scripts* em Linguagem de Programação Tcl, com o auxílio da aplicação desenvolvida em Python, para realizar a síntese da arquitetura em diferentes frequências de acesso à memória. Desta forma, a ferramenta Vivado® foi automatizada com uso de *scripts* para facilitar a alteração da síntese e assim aplicar os algoritmos em diferentes frequências de maneira a reproduzir a funcionalidade do testador Eureka2®. O diagrama em blocos desta ferramenta é exibida na Figura 53.

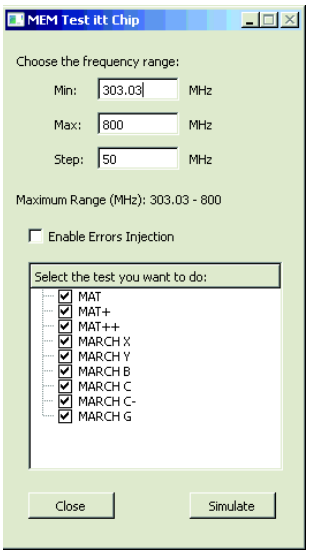
Conforme a Figura 53, observa-se que a rotina presente no arquivo *Inter_MemTest.py* evoca as demais bibliotecas presentes, incluindo os arquivos de *scripts* Tcl. Neste processo, os parâmetros informados pelo usuário são repassados as ferramentas de prototipação, que sintetizam a arquitetura. Neste sentido, o diagrama em blocos visualizado na Figura 53 torna-se uma ferramenta de interação com o ambiente de síntese Vivado® e o aplicativo de programação SDK® que por intermédio da utilização da linguagem Python e Tcl, permite criar a interface exibida na Figura 54. Criou-se assim um sistema automatizado de síntese em diferentes frequências de execução de algoritmos de teste de memória e coleta dos vetores de saída.

Figura 53 – Terceira ferramenta em diagrama de blocos.



Fonte – Elaborado pelo Autor.

Figura 54 – Terceira ferramenta, interface para a alteração da frequência.



Fonte – Elaborado pelo Autor.

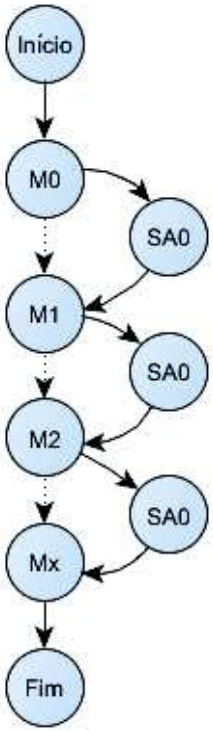
4.3 Resultados Apresentados pela Plataforma

Seguindo-se as etapas descritas na metodologia apresentada, o módulo já testado no equipamento Eureka2® foi inserido na plataforma desenvolvida, onde foram executados os mesmos algoritmos de teste, utilizando-se a tensão de alimentação de 1,5V como no processo de teste no Eureka2®, uma vez que esta é a única tensão disponível no kit FPGA. Nos subitens seguintes, apresentam-se os resultados obtidos pelo sistema de teste proposto, em diferentes cenários.

4.3.1 Simulação de Falhas tipo Stuck-At-Zero

Com intuito de validar a capacidade dos algoritmos de detectarem falhas, realizou-se a simulação de falhas do tipo *Stuck-At-Zero* entre a execução dos algoritmos, como indica a Figura 55. Ao final de cada elemento do algoritmo de teste, um conjunto de 16 falhas foram inseridas, de maneira que cada uma das falhas possui um endereço fixo de injeção e um bit específico de alteração. O resultado da captura de falhas é apresentado na Tabela 9.

Figura 55 – Simulação de falhas tipo Stuck-At-Zero.



Fonte – Elaborado pelo Autor.

Com as informações apresentadas na tabela 9, não se verifica a limitação de funcionamento do módulo em uso nas frequências de acesso à memória de 750MHz e 800Mhz, reportadas na qualificação anterior do módulo, realizado pelo testador Eureka2®. No que tange a cobertura de falhas, observa-se que todas as falhas injetadas foram detectas.

Tabela 9 – Captura de falhas, na plataforma, em diferentes frequências [MHz].

Algoritmo	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**
MATS	16	16	16	16	16	16	16	16	16
MATS+	16	16	16	16	16	16	16	16	16
MATS++	16	16	16	16	16	16	16	16	16
MARCH B	16	16	16	16	16	16	16	16	16
MARCH C-	16	16	16	16	16	16	16	16	16
MARCH X	16	16	16	16	16	16	16	16	16
MARCH Y	16	16	16	16	16	16	16	16	16
MARCH G	16	16	16	16	16	16	16	16	16

Fonte – Elaborado pelo Autor.

4.3.2 Técnica de Injeção Física de Falhas tipo Stuck-At-Zero

No sentido de propor a injeção externa ao algoritmo de teste, de falhas do tipo *Stuck-At-Zero*, foi proposta uma técnica utilizando-se um resistor de *pull-down* que ao ser conectado na linha de dados do módulo fornece um caminho de baixa impedância ao sinal de escrita, assim forçando a escrita de um nível lógico baixo nas células da memória, mesmo que o algoritmo mantivesse nível lógico alto. A Figura 56 representa a técnica utilizada, com cuidado de realizar a injeção da falha após a rede resistiva de compensação das linhas de dados, garantindo-se assim a segurança do procedimento para os pinos de conexão com o FPGA.

4.3.3 Validação da Técnica de Injeção Física de Falhas tipo Stuck-At-Zero

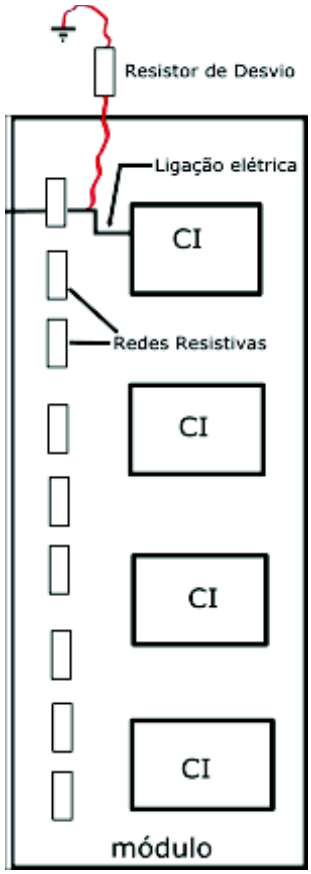
A validação da técnica proposta de injeção física de falhas do tipo *Stuck-At-Zero* foi realiza utilizando-se novamente do testador Eureka2®. A Figura 57 exibe o procedimento no qual o módulo é inserido no testador que executa os algoritmos anteriormente citados. Durante a execução dos testes, os terminais do resistor de desvio são conectados à linha de dados, de maneira a inserir as falhas no módulo.

Na Tabela 10, exibe-se o resultado da captura de falhas injetadas no módulo pela técnica de inserção de resistor de *pull-down*. Verifica-se que os algoritmos reportaram falhas, exceto o algoritmo MARCH X, que propositalmente foi deixado de fora da inserção de falhas, para poder-se verificar que as falhas reportadas decorriam da técnica utilizada. Restrições de prazo limitaram essa validação a apenas a frequência de 700 MHz, uma vez que, consiste em uma frequência em que é possível realizar os testes tanto no sistema Eureka2® como na plataforma proposta.

4.3.4 Técnica de Injeção Física de Falhas Tipo Stuck-At-Zero Aplicada na Plataforma

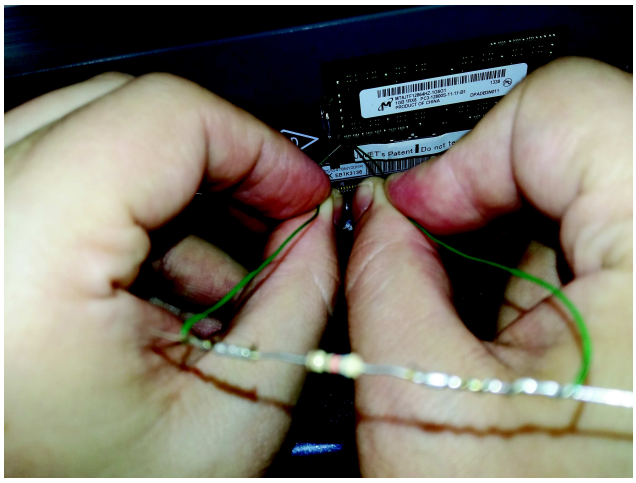
Utilizando-se o procedimento anteriormente realizado e validado no testador Eureka2®, aplica-se a técnica na plataforma proposta, a fim de permitir a validação funcional da mesma. A

Figura 56 – Injeção de falhas tipo *Stuck-At-Zero* no módulo.



Fonte – Elaborado pelo Autor.

Figura 57 – Validação da injeção de falhas tipo *Stuck-At-Zero* no módulo.



Fonte – Elaborado pelo Autor.

Figura 58 exibe a reprodução do procedimento de injeção de falhas físicas externas ao sistema de teste para o módulo em uso, desta vez inserido na plataforma desenvolvida.

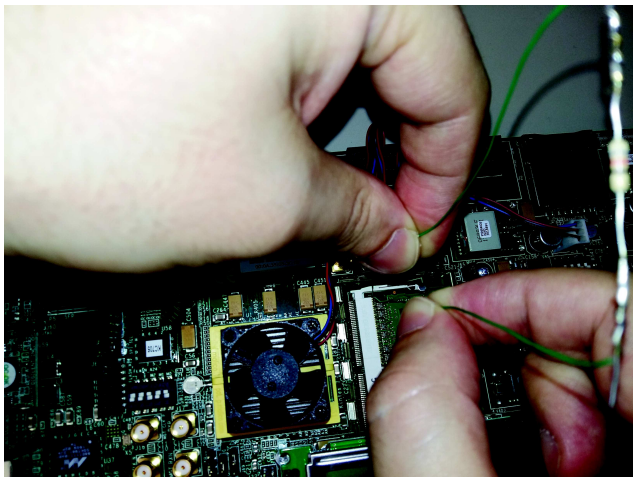
Com o sistema em funcionamento, executando-se os algoritmos anteriormente citados, também aplicados pelo testador Eureka2® ao módulo, o resistor de desvio foi conectado a

Tabela 10 – Captura de falhas da injeção.

Algoritmo	**700**
MATS	Fail
MATS+	Fail
MATS++	Fail
MARCH B	Fail
MARCH C-	Fail
MARCH X	Pass
MARCH Y	Fail
MARCH G	Fail

Fonte – Elaborado pelo Autor.

Figura 58 – Injeção de falhas tipo *Stuck-At-Zero* na plataforma.



Fonte – Elaborado pelo Autor.

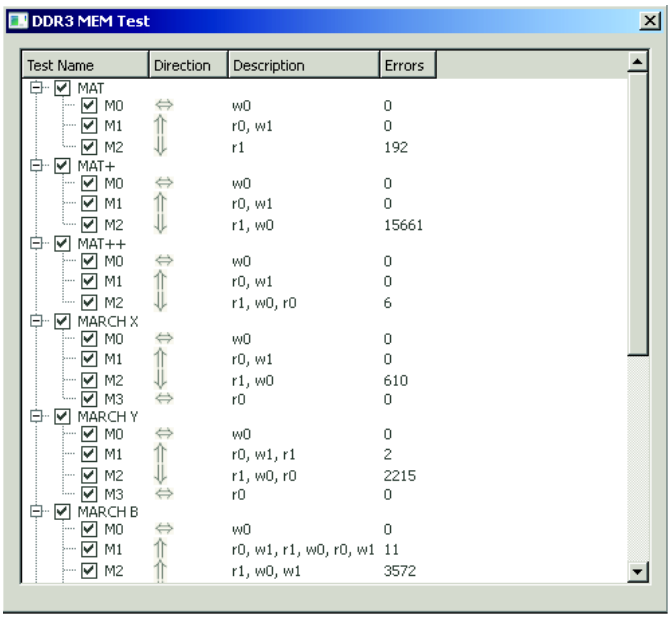
linha de dados, assim forçando um nível lógico baixo na mesma, que é inserido na matriz da memória de forma indiferente aos algoritmos em execução. Apresentam-se nas Figuras 59 e 60, os resultados da captura de falhas realizada pelo sistema proposto em funcionamento.

Observa-se que as falhas são reportadas nos elementos march que realizam a leitura com expectativa de encontrar valor lógico um. Visto que a injeção força a escrita de um valor lógico zero, existe inconsistência das informações, sendo reportada. Um *log* de dados é criado, reportando as falhas para análise, sendo exibido na Figura 61. Um total de 131.751 falhas foram reportadas pelo sistema. A falta de controle no processo de injeção de falhas aliada ao ruído elétrico causado pelo contato dos terminais do resistor justificam a aleatoriedade da captura.

4.3.5 Considerações de Desempenho

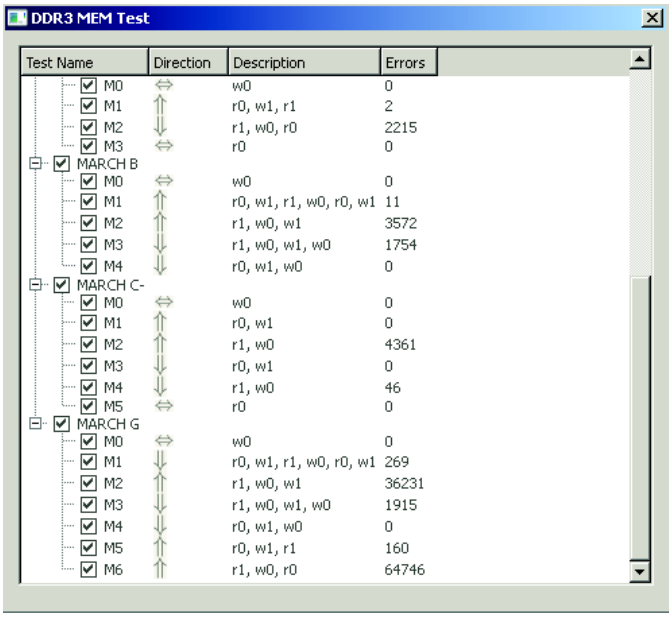
Com relação ao tempo médio de teste nos dois sistemas utilizados, pode-se visualizar na Tabela 11 um comparativo entre os mesmos. Verifica-se diferença dos valores, no caso da plataforma proposta, questões da arquitetura utilizada restringem o desempenho do MicroBlaze®,

Figura 59 – Captura de falhas pela plataforma.



Fonte – Elaborado pelo Autor.

Figura 60 – Captura de falhas pela plataforma, demais algoritmos.



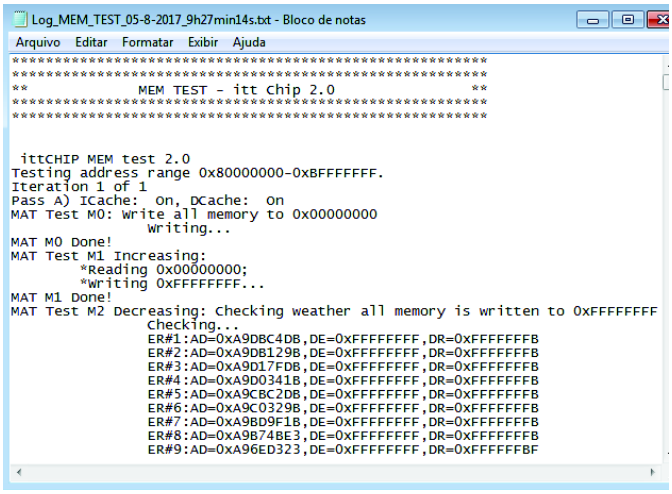
Fonte – Elaborado pelo Autor.

como sua frequência máxima de trabalho aliada ao fato do mesmo ser responsável pelo controle da comunicação utilizada.

4.3.6 Medidas de Confirmação da Frequência de Acesso ao Módulo

Com objetivo de confirmar o acesso à memória em diferentes frequências, utilizou-se o osciloscópio modelo MSO-X 9130A do fabricante Keysight Technologies®, disponível no

Figura 61 – Log de dados da captura de falhas pela plataforma.



Fonte – Elaborado pelo Autor.

Tabela 11 – Tempo médio de teste do módulo em diferentes sistemas.

Sistema	Tempo	Escala
Eureka2®	00:06:32	horas
Plataforma	02:26:57	horas

Fonte – Elaborado pelo Autor.

Laboratório de Teste do itt CHIP, para medir as diferentes frequências de acesso à memória, no sentido de comprovar se a frequência programada foi realmente sintetizada. A Figura 62, ilustra o acesso à memória em 800Mhz e a Figura 63, o acesso à memória em 400MHz. Estas medidas são de caráter informativo, com objetivo de validar as diferentes frequências de acesso, não se deseja realizar análises relativas à integridade do sinal.

Figura 62 – Acesso à memória em 800MHz.

Fonte – Elaborado pelo Autor.

Figura 63 – Acesso à memória em 400MHz.

Fonte – Elaborado pelo Autor.

4.3.7 Restrições Encontradas na Solução

Durante a construção do sistema proposto verificou-se que o IP controlador de memória possui a necessidade de calibração inicial nos pinos de conexão do FPGA com o módulo de memória. A Figura 64 exibe estes sinais de calibração enviados do FPGA à memória durante a inicialização do núcleo IP do controlador de memória. Esse fato limita o emprego do controlador MIG utilizado neste trabalho, ao interfaceamento de módulos com respectivas linhas de dados, endereços e controles funcionais.

Figura 64 – Sinais de calibração dos pinos do FPGA.

Fonte – Elaborado pelo Autor.

4.4 Considerações do Capítulo

Neste capítulo descreveram-se os resultados obtidos na plataforma de teste proposta. Pode-se destacar que o kit KC705 demostrou-se satisfatório para a função idealizada. O processo

de levantamento das características do módulo e sua comparação com os resultados obtidos foi descrito, além de medidas de frequência comprovando-se o acesso à memória. Esse trabalho limitou-se ao teste de apenas um módulo, uma vez que grande parte do esforço foi dirigido às etapas de desenvolvimento.

Resultados adicionais desta dissertação incluem a construção de uma interface de automação para síntese da arquitetura e aquisição dos vetores de saída do módulo em teste, permitindo-se expandir as funcionalidades da plataforma desenvolvida frente ao testador comercial.

A forma de validação para obtenção de resultados foi exposta, apoiando-se na metodologia proposta por essa dissertação. No capítulo 5 são apresentadas as conclusões e análise de pontos em aberto deste trabalho.

5 CONSIDERAÇÕES FINAIS

Essa dissertação consiste em uma contribuição ao desenvolvimento de sistemas de teste, para o qual procurou-se propor uma plataforma para teste de módulos de memória SDRAM DDR3 SODIMM além de uma metodologia para sua validação.

Na revisão da literatura encontram-se elementos que permitem compreender a importância do teste em sistemas eletrônicos, bem como entender sua dificuldade no atual cenário de aumento da complexidade de circuitos integrados, além da fundamental importância da etapa de teste na qualidade final dos produtos, uma vez que o custo de reparação ao nível de sistema é incrementado na ordem das dezenas. Soma-se a isso, a apresentação de uma revisão sobre os mecanismos de falhas em dispositivos eletrônicos e também de algoritmos de teste tradicionais aplicados em memórias.

No que tange aos testes realizados nessa dissertação, não se evidencia a presença de limitações funcionais no módulo de memória disponível para os experimentos, uma vez que, não se encontrou defeitos físicos no módulo em uso. A não presença de defeitos físicos não inviabiliza a utilização da plataforma proposta, uma vez que a mesma é capaz de desenvolver o teste funcional dos módulos.

Uma técnica de injeção física de falhas tipo *Stuck-At-Zero* foi proposta e validada por intermédio de experimentos no testador Eureka2® em uma frequência de acesso à memória que pode ser reproduzida na plataforma desenvolvida, neste sentido a injeção de erros reportou 131.751 falhas detectáveis pelos algoritmos, indicando a funcionalidade dos mesmos. Questões de ordem prática, como o respeito a limites de tensão e correntes nos pinos de conexão do FPGA com o módulo de memória restringiram essa injeção a falhas ao tipo *Stuck-At-Zero*. No entanto a condição de falha foi devidamente reportada. Restrições no prazo limite, forçaram a utilizar-se apenas a frequência de 700MHz, uma vez que não houve detecção de falhas em nenhuma etapa de teste, tando com o testador Eureka2® como com a plataforma, a fim de permitir a comparação de resultados.

Em relação ao desempenho da plataforma, verifica-se que existe grande diferença na ordem de grandeza dos tempos de testes, quando comparados ao testador comercial Eureka2®. Neste sentido, pode-se explicar essa discrepância na característica de restrição de velocidade máxima permitida pelo IP MicroBlaze® na ordem de 330MHz para o FPGA em uso no kit KC705, aliada ao fato de que nessa arquitetura o MicroBlaze® absorveu muitas tarefas, em especial o controle dos padrões de teste e a comunicação com o *host*.

A utilização de núcleos de IP mostrou-se de grande relevância par a rápida prototipação da plataforma, ao possibilitar uso de elementos de uso comercial. No entanto, nas etapas de varredura em frequência, verificou-se um pequeno inconveniente, a necessidade de nova síntese

a cada alteração na frequência de acesso ao módulo de memória. Soma-se a isso a necessidade constante de calibração das linhas de conexão com o módulo, característica que é responsável por adicionar uma limitação em relação ao tipo de defeito existente no módulo inserido, uma vez que, seus pinos de dados devem ser funcionais.

Limitações atuais da plataforma desenvolvida também incluem a incapacidade de teste do módulo utilizando-se diferentes tensões de alimentação, o que permitiria expandir a capacidade de análise de funcionamento de módulos em teste. O atual sistema de geração de clock incluído no kit também limita a quantidade e valores de frequências que podem ser utilizados. Soma-se a isso o fato de que esse sistema de clock possui utilização geral, no sentido de que desvios na frequência podem ter contribuído para os resultados obtidos.

A ferramenta desenvolvida, anteriormente exibida na Figura 54 tornou-se importante mecanismo de aceleração na coleta de dados, sendo parte importante dos resultados e contribuições desta dissertação ao itt CHIP. Também é resultado deste trabalho o conhecimento desenvolvido ao longo dessa dissertação e absorvido pelo grupo de teste do instituto.

Naturalmente trabalhos que exploram aspectos práticos, defrontam-se com questões relacionadas à impossibilidade de previsão de todos os acontecimentos que decorrem de sua execução. Neste sentido, dificuldades devidas à utilização do adaptador para módulos SODIMM podem ter contribuído para a inserção de falhas fantasmas no módulo, apesar do adaptador ser considerado adequado. Adicionam-se com o mesmo linhas de transmissão de sinais, criando-se impedâncias extras nas linhas de conexão com o módulo que podem permitir a reflexão dos sinais, além de contatos elétricos adicionais. Desta forma, uma contribuição imaterial desta dissertação é o levantamento da interferência do adaptador SODIMM em frequências de acesso ao módulo entre 750MHz e 800MHz, uma vez que nestas frequências não foram reportadas falhas no sistema desenvolvido. Assim, cuidados ambientais devem ser tomados, como a climatização da sala bem como o respeito ao tempo de estabilização térmica do testador Eureka2®. Neste sentido, em trabalhos futuros poderão ser explorados aspectos de integridade de sinal, inclusive estendendo-se a prototipação de um adaptador compatível com o Eureka2®.

Trabalhos futuros também podem explorar outros aspectos não desenvolvidos neste trabalho, como técnicas de codificação segura, no sentido de possibilitar a integridade funcional dos códigos. O desenvolvimento de um controlador de memória DDR3 dedicado a função de teste, ganha um especial destaque, uma vez que pode contribuir na redução do tempo de teste, graças a otimizações que permitam a mudanças na frequência de acesso ao módulo. Ao utilizar a concepção de IP, tal controlador pode permitir reúso de suas funções em outras aplicações. Outro aspecto seria o de um mecanismo utilizando-se de chaves estáticas controladas, permitindo-se uma injeção de falhas com maior controle no sistema.

Abre-se caminho para o uso e desenvolvimento de tecnologias voltadas ao teste de sistemas eletrônicos, evidenciando-se a amplitude da área do teste, uma vez que utiliza-se de diversos campos de conhecimento para a realização de seus objetivos.

REFERÊNCIAS BIBLIOGRÁFICAS

AMORY, A. M. *Integração e Avaliação de Técnicas de Teste Baseado em Software no Fluxo de Projeto de SOCs*. 133 p. Dissertação (Dissertação de Mestrado em Ciências da Computação) — Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre, 2003. Citado 2 vezes nas páginas 19 e 29.

BAKER, K.; BEERS, J. V. Shmoo plotting: the black art of ic testing. *IEEE Design Test of Computers*, v. 14, n. 3, p. 90–97, Jul 1997. ISSN 0740-7475. Citado na página 36.

BONATTO, A. C. *Núcleos de Interface de Memória DDR SDRAM para Sistemas-Em-Chip*. 122 p. Dissertação (Dissertação de Mestrado em Engenharia Elétrica) — Universidade Federal do Rio Grande do Sul, Porto Alegre, 2009. Citado 10 vezes nas páginas 9, 15, 16, 23, 24, 39, 40, 43, 44 e 45.

BUSHNELL, M. L.; AGRAWAL, V. D. *Essentials of Electronic Testing - For Digital Memory and Mixed-Signal VLSI Circuits*. [S.l.: s.n.], 2000. ISBN 0792386868. Citado 11 vezes nas páginas 15, 17, 18, 19, 20, 28, 29, 30, 31, 32 e 33.

CHEN, W.-K. *The VLSI Handbook, Second Edition (Electrical Engineering Handbook Series)*. [S.l.]: CRC Press, 2007. ISBN 9780849341991. Citado 5 vezes nas páginas 15, 16, 23, 24 e 28.

COTTENS, P. E. P. d. A. *Development Of An Artificial Neural Network Architecture Using Programmable Logic*. 61 p. Dissertação (Dissertação de Mestrado. Programa de Pós-Graduação em Engenharia Elétrica) — Universidade do Vale do Rio dos Sinos - UNISINOS, São Leopoldo, 2016. Citado na página 28.

CST. *Eureka-2 DDR3 User Manual – Revision 1*. 2336 LU FIELD ROAD, DALLAS, TEXAS 75229 USA, 2009. Https//:www.simmtester.com. Citado na página 34.

CST. *Eureka2 DDR3 1333Mhz*. 2015. Disponível em: http://www.simmtester.com/page/products/productSpecs/Eureka2_DDR3Acesso em: 15/11/2015. Citado 2 vezes nas páginas 15 e 34.

GOOR, A. J. V. D. Using march tests to test srams. *IEEE Design Test of Computers*, v. 10, n. 1, p. 8–14, March 1993. ISSN 0740-7475. Citado 4 vezes nas páginas 16, 29, 31 e 50.

GOOR, A. J. V. D. An industrial evaluation of dram tests. *IEEE Design Test of Computers*, v. 21, n. 5, p. 430–440, Sept 2004. ISSN 0740-7475. Citado 2 vezes nas páginas 29 e 31.

GROUT, I. *Digital Systems Design with FPGAs and CPLDs*. [S.l.]: Elsevier Ltd, 2008. ISBN 978-0-7506-8397-5. Citado 2 vezes nas páginas 16 e 26.

GROUT, I. A. *Integrated Circuit Test Engineering Modern Techniques*. [S.l.]: Springer-Verlag London, 2006. ISBN 9781846280238. Citado 6 vezes nas páginas 15, 17, 18, 21, 22 e 29.

HAMDIOUI, S. et al. Memory test experiment: industrial results and data. *IEE Proceedings - Computers and Digital Techniques*, v. 153, n. 1, p. 1–8, Jan 2006. ISSN 1350-2387. Citado na página 31.

HUANG, C.-Y. et al. *Material Characterization and Failure Analysis for Microelectronics Assembly Processes, Wide Spectra of Quality Control*. 2011. Dr. Isin Akyar (Ed.), InTech, DOI: 10.5772/23532. Disponível em: https://www.intechopen.com/books/wide-spectra-of-quality-control/material-characterization-and-failure-analysis-for-microelectronics-assembly-processes Acesso em: 26/04/2017. Citado 2 vezes nas páginas 15 e 19.

INNOVENTIONS. *RAMCHECK LX Specifications*. [S.l.]: Innoventions, 2015. Disponível em:http://www.innoventions.com/pdf_files/ramcheck_lx_specifications.pdf Acesso em: 15/11/2015. Citado 3 vezes nas páginas 15, 32 e 33.

INTEL. *FPGA Devices*. [S.l.]: Intel Corporation, 2017. Disponível em: https://www.intel.com/content/www/us/en/fpga/devices.html Acesso em: 20/06/2017. Citado na página 28.

JEDEC. *JESD79-3F DDR3 SDRAM Standard*. [S.l.], 2012. Disponível em: http://www.jedec.org/standards-documents/docs/jesd-79-3d Acesso em: 04/01/2016. Citado 2 vezes nas páginas 24 e 25.

JEDEC. *4.20.19 - 240-Pin PC3-6400/PC3-8500/PC3-10600/PC3-12800/PC3-14900/PC3-17000 DDR3 SDRAM Unbuffered DIMM Design Specification*. [S.l.], 2013. Disponível em: https://www.jedec.org/sites/default/files/docs/4_20_19R22A.pdf Acesso em: 04/01/2016. Citado 3 vezes nas páginas 15, 24 e 25.

JEDEC. *204-Pin DDR3 SDRAM Unbuffered SO-DIMM Design Specification*. [S.l.], 2014. Disponível em: https://www.jedec.org/system/files/docs/4_20_18R24.pdf Acesso em: 10/02/2017. Citado na página 25.

JEDEC. *JM21R JEDEC Manual of Organization and Procedure*. [S.l.], 2015. Disponível em: http://www.jedec.org/sites/default/files/docs/JM21R.pdf Acesso em: 05/01/2016. Citado na página 24.

KEEZER, D. et al. An fpga-based ate extension module for low-cost multi-ghz memory test. *Test Symposium (ETS), 2015 20th IEEE European*, p. 1–6, May 2015. Citado 5 vezes nas páginas 9, 16, 42, 43 e 44.

KRUG, M. R. *Aumento da Testabilidade do Harware com Auxilio de Técnicas de Teste de Software*. 102 p. Tese (Tese de Doutorado em Ciências da Computação) — Universidade Federal do Rio Grande do Sul, Porto Alegre, 2007. Citado 3 vezes nas páginas 16, 18 e 19.

LINDER, M. et al. An analysis of industrial sram test results. a comprehensive study on effectiveness and classification of march test algorithms. *IEEE Design Test*, v. 31, n. 3, p. 42–53, June 2014. ISSN 2168-2356. Citado 4 vezes nas páginas 15, 29, 30 e 31.

MARTIN, P. L. *Electronic Failure Analysis Handbook*. [S.l.]: McGraw-Hill, 1999. ISBN 978-0071626347. Citado 4 vezes nas páginas 15, 18, 19 e 20.

MICRON. *1GB, 2GB, 4GB (x64, SR) 204-Pin DDR3 SODIMM*. [S.l.], 2010. Disponível em: https://www.micron.com/parts/modules/ddr3-sdram/mt8jtf51264hz-1g6 Acesso em:04/06/2017. Citado na página 52.

MICROSEMI. *PolarFire FPGA Product Brochure*. [S.l.]: Microsemi Corporation, 2017. Disponível em: https://www.microsemi.com/ Acesso em: 20/06/2017. Citado na página 28.

MORAES, M. S. *STEP:planejamento, geração e seleção de auto-teste on-line para processadores embarcados*. 144 p. Dissertação (Dissertação de Mestrado em Ciências da Computação) — Universidade Federal do Rio Grande do Sul, Porto Alegre, 2006. Citado na página 18.

MOSTARDINI, L. et al. FPGA-based low-cost automatic test equipment for digital integrated circuits. *2009 IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*, n. September, p. 32–37, 2009. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5343031>. Citado 7 vezes nas páginas 9, 16, 39, 40, 41, 42 e 44.

NAIR, R. Comments on "an optimal algorithm for testing stuck-at faults in random access memories". *IEEE Transactions on Computers*, C-28, n. 3, p. 258–261, March 1979. ISSN 0018-9340. Citado na página 30.

NAIR, R.; THATTE, S. M.; ABRAHAM, J. A. Efficient algorithms for testing semiconductor random-access memories. *IEEE Transactions on Computers*, C-27, n. 6, p. 572–576, June 1978. ISSN 0018-9340. Citado na página 30.

OLIVEIRA, L. L. D. et al. A Low-Price Platform to Test Digital Integrated Circuits Using FPGA. *Circuits and Systems, 2005. 48th Midwest Symposium on*, p. 1127–1130, 2005. Citado 5 vezes nas páginas 9, 16, 38, 43 e 44.

REIS, R. A. d. L. *Concepção de Circuitos Integrados*. 2. ed. Porto Alegre: Instituto de Informática da UFRGS:Sagra Luzzatto, 2002. (Série Livros Didáticos, número 7). ISBN 85-241-0625-5. Citado na página 15.

SAMSUNG. *204pin Unbuffered SODIMM based on 4Gb Q-die - Rev. 1.21*. [S.l.], 2013. Disponível em: http://www.samsung.com/semiconductor/products/dram/pc-dram/ddr3-sodimm/M471B5674QH0?ia=694 Acesso em:02/04/2017. Citado 2 vezes nas páginas 15 e 26.

SRINIVASA, S. R. et al. Improving fpga design with monolithic 3d integration using high dense inter-stack via. In: *2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. [S.l.: s.n.], 2017. p. 128–133. Citado na página 28.

SUK, D. S.; REDDY, S. M. A march test for functional faults in semiconductor random access memories. *IEEE Transactions on Computers*, C-30, n. 12, p. 982–985, Dec 1981. ISSN 0018-9340. Citado na página 30.

TEIKON. *Catálogos de Produtos*. [S.l.], 2015. Disponível em: http://www.teikon.com.br/site_ptbr/images/site/catalogos/produtos_teikon.pdf. Acesso em: 1/12/2015. Citado na página 25.

TOCCI, R. J.; WIDMER, N. S.; MOSS, G. L. *Digital Systems Principles and Applications*. [S.l.]: Pearson Education, Inc, 2007. ISBN 0-13-173969-7. Citado 2 vezes nas páginas 15 e 27.

WANG, L. T.; WU, C. W.; WEN, X. *VLSI Test Principles and Architectures - Design for Testability*. [S.l.]: Elsevier Inc, 2006. ISSN 0717-6163. ISBN 9780874216561. Citado 4 vezes nas páginas 15, 29, 30 e 32.

WESTE, N. H. E.; HARRIS, D. M. *CMOS VLSI Design*. [S.l.]: Addison-Wesley, 2013. v. 53. ISSN 1098-6596. ISBN 9788578110796. Citado 4 vezes nas páginas 15, 21, 22 e 23.

XILINX. *Kintex-7 FPGA KC705 Evaluation Kit - Getting Started Guide - UG883*. [S.l.], 2014. Disponível em: www.xilinx.com Acesso em: 14/12/2016. Citado na página 48.

XILINX. *Vivado Design Suite Tutorial- Embedded Processor Hardware Design - UG940*. [S.l.], 2014b. Disponível em: www.xilinx.com Acesso em: 15/01/2017. Citado na página 49.

XILINX. *KC705 Evaluation Board for the Kintex-7 FPGA - User Guide - UG810*. [S.l.], 2016. Disponível em: www.xilinx.com Acesso em: 14/12/2016. Citado na página 48.

XILINX. *Delivering a Generation Ahead at 20nm and 16nm*. [S.l.]: Xilinx Inc, 2017. Disponível em: https://www.xilinx.com/about/generation-ahead-16nm.html Acesso em: 1/07/2017. Citado na página 28.

ZHOU, Z.; CHENG, S.; LIU, Q. Application of DDR Controller for High-speed Data Acquisition Board. In: *First International Conference on Innovative Computing, Information and Control - Volume I (ICICIC'06)*. IEEE, 2006. v. 2, p. 611–614. ISBN 0-7695-2616-0. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1692061>. Citado 5 vezes nas páginas 9, 16, 39, 43 e 44.