UNIVERSIDADE FEDERAL DA PARAÍBA – UFPB
CENTRO DE ENERGIAS ALTERNATIVAS E RENOVÁVEIS – CEAR
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA – PPGEE

LEONARDO ALVES DIAS

OTIMIZAÇÃO GENÉTICA DE SEQUÊNCIAS DE PADRÕES DE TESTE PARA CIRCUITOS VLSI

JOÃO PESSOA
2016

LEONARDO ALVES DIAS

OTIMIZAÇÃO GENÉTICA DE SEQUÊNCIAS DE PADRÕES DE TESTE PARA CIRCUITOS VLSI

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica - PPGEE, da Universidade Federal da Paraíba - UFPB, como requisito parcial para a obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Prof. Dr. Cleonilson Protásio de Souza

JOÃO PESSOA

2016

FICHA CATALOGRÁFICA

DIAS, Leonardo A.

OTIMIZAÇÃO GENÉTICA DE SEQUÊNCIAS DE PADRÕES DE TESTE PARA CIRCUITOS VLSI – João Pessoa, 2016.

Nº de páginas: 88

Área de concentração: Sistemas Eletroeletrônicos Energeticamente Eficientes.

Orientadores: Prof. Dr. Cleonilson Protásio de Souza.

Dissertação (Mestrado) – Universidade Federal da Paraíba - PPGEE

1. Teste de Circuitos Integrados; 2. Geradores de Padrões de Teste; 3. Consumo Energético 4. Algoritmo Genético 5. Algoritmo de Berlekamp-Massey.

UNIVERSIDADE FEDERAL DA PARAÍBA – UFPB

CENTRO DE ENERGIAS ALTERNATIVAS E RENOVÁVEIS – CEAR

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA – PPGEE

A Comissão Examinadora, abaixo assinada, aprova a Dissertação de Mestrado

OTIMIZAÇÃO GENÉTICA DE SEQUÊNCIAS DE PADRÕES DE TESTE PARA CIRCUITOS VLSI

Elaborada por

LEONARDO ALVES DIAS

como requisito final para obtenção do grau de

Mestre em Engenharia Elétrica.

COMISSÃO EXAMINADORA

_____

PROF. DR. CLEONILSON PROTÁSIO DE SOUZA - UFPB/CEAR/PPGEE

(Orientador - Presidente da Banca)

_____

PROF. DR. ALISSON VASCONCELOS DE BRITO - UFPB/CI/PPGI

(Membro Externo)

_____

PROF. DR. JUAN MOISÉS MAURÍCIO VILLANUEVA - UFPB/CEAR/PPGEE

(Membro Interno)

João Pessoa, 29 de Fevereiro de 2016.

A todos aqueles que possuem o dom da determinação, pois obstáculos são insignificantes diante de seus sonhos gigantes.

Dias, L. A.

AGRADECIMENTOS

Agradecer, uma palavra utilizada para expressar gratidão, porém em alguns casos impossível de expressar o real sentimento, pois não existem palavras capazes de mensurá-lo.

Quero agradecer primeiramente a Deus pela oportunidade e força de vontade para chegar até aqui.

A minha família que sempre me apoiou em todos os momentos, em principal a minha mãe que sempre acreditou em quão longe eu poderia chegar.

A meus amigos, que não é possível citar cada um aqui, mas destacar Luzenir Raimunda e Igor Almeida Brito por todo o apoio no decorrer da realização deste trabalho.

A todos os meus colegas do PPGEE que sempre estiveram comigo nessa jornada, em destaque aos que compuseram a turma 2014.1, Jeane Silva, Priscilla Kadja, João Raphaell, Thaunaí Segundo e Francisco Júnior, que marcaram história durante esses dois anos.

A todos os meus colegas do Laboratório de Microengenharia pelos momentos hilários de descontração que foram únicos, e por acompanhar de perto todo o processo, sempre estendendo a mão e dedicando um pouco do tempo para me ajudar nas dificuldades, em destaque a Pâmela Svetllana por todo o apoio moral.

Ao meu orientador, Cleonilson Protasio de Souza por toda paciência e dedicação até aqui.

A CAPES e ao CNPQ pelo apoio e suporte financeiro dado a esta pesquisa.

A coordenação do PPGEE e a UFPB por disponibilizar o ambiente adequado para realização dos nossos estudos.

A todos os professores que até hoje fizeram parte da minha vida acadêmica, por quem eu tenho uma profunda gratidão.

Enfim, a todos que direta ou indiretamente contribuíram para a realização desse trabalho, meus sinceros agradecimentos.

“A vida é feito andar de bicicleta: se parar você cai.
Vai em frente sem parar, que a parada é suicida,
porque a vida é muito curta e a estrada é comprida.
Você sobe e você desce na escada da vida e
às vezes parece que a batalha tá perdida
que você voltou pro ponto de partida.
Vai à luta, levanta, revida!
Vai em frente não se rende não se prende nesse medo de errar,
que é errando que se aprende que o caminho até parece complicado
e às vezes tão difícil que você se surpreende quando sente
de repete que era tudo muito simples – vai em frente
que você entende
Boa sorte, firme e forte, vai com a força da mente.
Vai sabendo que não há nenhum peso que você não aguente.
Vai na marra, vai na garra, vai em frente.
E se agarra no seu sonho com unhas e dentes.
Pra saber o que é possível é preciso que se tente conseguir o impossível,
então tente!
Sempre alimente a esperança de vencer.
Só duvide de quem duvida de você”
Gabriel Pensador

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIAÇÕES

AG	Algoritmo Genético
ATE	Equipamento de Teste Automático (*Automatic Test Equipment*)
ATPG	Gerador de Padrões de Teste Automático (*Automatic Test Pattern Generator*)
BIST	*Built-In Self-Test*
BMA	Algoritmo de Berlekamp-Massey
CI	Circuito Integrado
CUT	Circuito sob teste (*Circuit Under Test*)
DFT	Técnicas de testabilidade (*Design-for-Testability*)
FC	Cobertura de falhas
HDL	Linguagem de Descrição de Hardware (*Hardware Description Language*)
LCS	Simulação de Circuitos Lógicos (*Logic Circuit Simulation*)
LFSR	Registrador de Deslocamento com Realimentação Linear (*Linear Feedback Shift Register*)
ORA	Analisador de Respostas (*Output Response Analyzer*)
P_{TC}	Ponderação do número de transições
P_{FC}	Ponderação da cobertura de falhas
P_L	Ponderação do comprimento do polinômio/LFSR obtido pelo BMA
PRNG	Gerador de números aleatórios (*Pseudorandom Number Generator*)
RTL	Circuito em nível de silício (*Real-Transfer Level*)
TAP	Porta de controle de teste (*Test Access Port*)
TC	Número de transições
TPG	Gerador de Padrões de Teste (*Test Pattern Generator*)
VLSI	Integração em larga escala (*Very Large Scale Integration*)
WSA	Atividade de Chaveamento Ponderada (*Weighted Switching Activity*)

RESUMO

OTIMIZAÇÃO GENÉTICA DE SEQUÊNCIAS DE PADRÕES DE TESTE PARA CIRCUITOS VLSI

Um circuito integrado (CI) em modo de teste apresenta um maior consumo energético comparado ao modo de operação normal, devido ao aumento do número de transições nos nós do circuito decorrentes da aplicação de padrões de teste utilizados para estimular o CI durante a execução do teste resultando em uma alta dissipação de potência que pode danificar o CI, acarretando em maiores custos para as fabricantes. Assim, neste trabalho é proposto um algoritmo genético para otimização de sequências de padrões de teste visando o baixo consumo energético, durante a execução do teste, mantendo uma adequada cobertura de falhas. É proposto também o uso do algoritmo de Berlekamp-Massey para sintetizar um gerador integrado de padrões de teste com baixa sobreárea de *hardware* capaz de gerar as sequências otimizadas baseado em Registrador de Deslocamento com Realimentação Linear. A otimização das sequências é feita através da redução do número de transições nos nós cuja avaliação é feita por um programa de computador desenvolvido nesta pesquisa em C++. Por fim, simulações foram realizadas com o algoritmo genético para verificar o comportamento em relação a otimização do número de transições, da cobertura de falhas e da sobreárea de *hardware*.

Palavras-chave: Teste de circuitos integrados, geradores de padrões de teste, consumo energético, algoritmo genético, algoritmo de Berlekamp-Massey.

ABSTRACT

GENETIC OPTIMIZATION OF TEST PATTERN SEQUENCES FOR VLSI CIRCUITS

Integrated circuit (IC) on test mode has higher energy consumption compared than when they are on normal operating mode, due to the increasing number of transitions in the circuit nodes. These transitions increase is caused by test patterns used to stimulate the IC during test execution, resulting in a high power dissipation, which can damage the IC leading to higher costs for manufacturers. Hence, in this paper is proposed a genetic algorithm for optimization of test patterns sequences which achieve low power consumption during the execution of IC testing, maintaining an adequate fault coverage. It is also proposed the use of Berlekamp-Massey algorithm to synthesize a LFSR-based test pattern generator with low area overhead capable of generating those optimized sequences. The genetic optimization of test sequences is performed by reducing the number of transitions in the circuit nodes using a computer program developed in this research. Lastly, simulations were performed using genetic algorithm for evaluating the behavior towards optimization of the number of transitions, the fault coverage and area overhead.

Keywords: Integrated circuit testing, test pattern generators, energy consumption, genetic algorithm, Berlekamp-Massey algorithm

.

1 INTRODUÇÃO

1 INTRODUÇÃO

Circuitos Integrados (CIs) estão presentes em diferentes tecnologias tornando-se essenciais ao cotidiano. Tendo em vista sua importância, se faz necessário que esses dispositivos sejam projetados e fabricados com alta confiabilidade, pois a ocorrência de falhas pode acarretar em danos graves, tendo como exemplo a área militar ou médica (SOUZA, 2005).

A evolução da tecnologia proporcionou a confecção de transistores com dimensões cada vez menores, permitindo uma maior integração dos mesmos em uma menor área de silício (*Very Large Scale Integration* - VLSI), obtendo como vantagem o aumento da frequência de operação, e a integração de funcionalidades em um único CI. No entanto, isso acarretou no aumento da probabilidade de que defeitos venham a ocorrer durante o processo de manufatura. Assim, faz-se necessário o teste dos CIs para garantir que estejam conforme foram projetados e sua confiabilidade (WANG; WU; WEN, 2006).

Normalmente, o processo de testes é realizado aplicando sequências de padrões de teste nas entradas de um circuito e analisando as respostas obtidas na saída (WEST; HARRIS, 2009), como pode ser visto na Figura 1.1. No âmbito de CIs digitais, o processo de teste pode ser realizado através de um Equipamento de Teste Automático (ETA) ou por técnicas de testabilidade.

FIGURA 1.1 – ABORDAGEM BÁSICA DE TESTE DE CI.

FONTE: DO AUTOR.

O teste do CI, visa garantir a confiabilidade do mesmo, e tem como principais parâmetros: a cobertura de falhas, a sobreárea de *hardware*, o consumo energético e tempo de teste (SOUZA, 2005; WANG; WU; WEN, 2006).

A cobertura de falhas remete a quantidade de falhas que são detectadas durante a execução de um teste, em que o ideal é uma cobertura de 100%, ou seja, detectar todas as falhas existentes. Logo, são necessárias técnicas que proporcionem ou que se aproximem do valor ideal de cobertura. Isto é necessário para evitar que circuitos com falhas sejam considerados sem falhas e utilizados em produtos (WEST; HARRIS, 2009).

A sobreárea de *hardware*, é o aumento da área de silício para adição de técnicas de teste no próprio CI, permitindo em alguns casos o auto teste. Segundo Souza, 2005, a técnica de auto teste está presente em diferentes tipos de CIs e é amplamente empregada. Já o tempo de teste consiste no tempo necessário para realizar o teste do CI. É importante frisar que todos os CIs são testados, logo, busca-se sempre a redução do tempo de teste (WANG; WU; WEN, 2006).

Outro fator importante a ser considerado na execução do teste, é o consumo energético dos CIs. Um CI em modo de testes, ou seja, recebendo estímulos consecutivos, consome energia e consequentemente dissipa potência (NICOLICI; AL-HASHIMI, 2003). E, de acordo com Wang, Wu e Wen, 2006, a principal fonte de dissipação de potência em um circuito em modo de testes é a potência dinâmica consumida, que é resultante da transição dos nós (chaveamento decorrente da troca de *bits* de 0 para 1 e vice-versa), quando um CI está em modo de operação. Logo, no modo de testes há um aumento considerável no consumo de potência dinâmica em relação ao modo de operação normal, em razão da sequência de padrões de teste utilizada para estimular o CI. No entanto, existe um limite de dissipação de potência suportado em cada CI de acordo com as especificações definidas durante seu projeto, então uma alta dissipação de potência durante a execução do teste, se ultrapassar este limite, pode acarretar em danos ao CI, por sua vez, reduzindo o rendimento do processo de manufatura (WEST; HARRIS, 2009).

Busca-se através deste trabalho obter sequências de padrões de teste que não ultrapassem os limites de dissipação de potência suportados pelo CI, sem afetar a cobertura de falhas e com um curto tempo de execução de teste, e também desenvolver um gerador de padrões de teste que tenha uma baixa ocupação de sobreárea de *hardware*, com intuito de reduzir os custos de se testar CIs.

Como visto anteriormente na Figura 1.1, as sequências de padrões de teste, são utilizadas para estimular o circuito, e estas sequências são provenientes de

geradores de padrões de teste. Segundo Souza (2005), registradores de deslocamento com realimentação linear (*Linear Feedback Shift-Register* – LFSR) são uma das arquiteturas mais empregadas como gerador de teste em técnicas de testatiblidade devido ao baixo consumo de sobreárea. Portanto, os geradores de padrões de teste são responsáveis pela maior parte da potência dissipada durante a execução do teste, a qual é oriunda do alto chaveamento resultante da sequência de padrões (YEAP, 1998; NICOLICI; AL-HASHIMI, 2003).

Dessa forma, este trabalho tem como enfoque a baixa dissipação de potência através da redução do chaveamento durante a execução do teste utilizando algoritmo genético com intuito de reduzir os riscos de causar danos ao CI, e sintetizar LFSRs como geradores de padrões de teste a partir do algoritmo de Berlekamp-Massey para um baixo consumo de sobreárea.

O algoritmo genético foi desenvolvido por John Holland visando adotar os “mecanismos” da evolução natural, em sistemas computadorizados (MITCHELL, 1999) e aplicado em diversas áreas em que se busca a otimização de soluções. Na área de testes, o algoritmo genético se mostrou eficaz de forma empírica em diferentes objetivos, dentre estes, a redução da atividade de chaveamento ao se testar um circuito (ANITA; VANATHI, 2014), sendo adotado também neste trabalho com esse mesmo objetivo, de reduzir o chaveamento, obtendo até mais de 90% de redução nas transições.

O algoritmo de Berlekamp-Massey (BMA) teve sua primeira aplicação na área de testes por Souza (2005). Esses utilizaram o BMA para encontrar o polinômio de menor comprimento capaz de sintetizar um LFSR que gera uma determinada sequência binária. Assim, o BMA é utilizado neste trabalho para obter LFSRs de menor comprimento que gere a sequência de baixo consumo gerada pelo algoritmo genético com a menor sobreárea de *harware*, sem a necessidade de arquiteturas extras ou modificações acopladas ao LFSR.

Diante do exposto, propõe-se neste trabalho, a otimização de sequências de padrões de teste, com base em algoritmo genético, para a redução do consumo de energia e potência dissipada, durante a execução do teste de CIs, a fim de reduzir e/ou eliminar os riscos de danificá-los, sem comprometer a cobertura de falhas ou o tempo de execução de teste. Em adição, propõe-se também, sintetizar geradores de

teste, com base no algoritmo de Berlekamp-Massey, para gerar as sequências otimizadas em técnicas de testabilidade.

1.1 OBJETIVOS

O objetivo geral deste trabalho, é a otimização de sequências de padrões de teste para circuitos VLSI com foco no baixo consumo energético, mantendo uma alta cobertura de falhas.

Os objetivos específicos para atingir o objetivo geral desse trabalho, são:

- Computar a atividade de chaveamento do circuito sendo estimulado.
- Otimizar as sequências de padrões de teste reduzindo o número de transições através do algoritmo genético para redução da dissipação de potência.
- Utilizar o BMA para sintetizar geradores de padrões de teste com menor comprimento.
- Realizar simulações experimentais para analisar o comportamento do algoritmo genético e de Berlekamp-Massey, em relação ao objetivo pretendido com este trabalho.

1.2 ORGANIZAÇÃO DO TRABALHO

Este trabalho está organizado em 6 Capítulos. No Capítulo II, é descrito a fundamentação teórica que este trabalho tem como base. Introduzindo, desde testes de circuitos integrados, através de conceitos, técnicas e informações, a trabalhos publicados na área.

No Capítulo III, é descrito o simulador desenvolvido para computar a atividade de chaveamento de circuitos digitais.

No Capítulo IV, é descrito o algoritmo genético utilizado para otimizar as sequências de padrões de teste, enquanto no Capítulo V são expostos os resultados obtidos através de simulações.

Por fim, no Capítulo VI, é descrita a conclusão para com este trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

2 FUNDAMENTAÇÃO TEÓRICA

2.1 TESTES EM CIRCUITOS INTEGRADOS

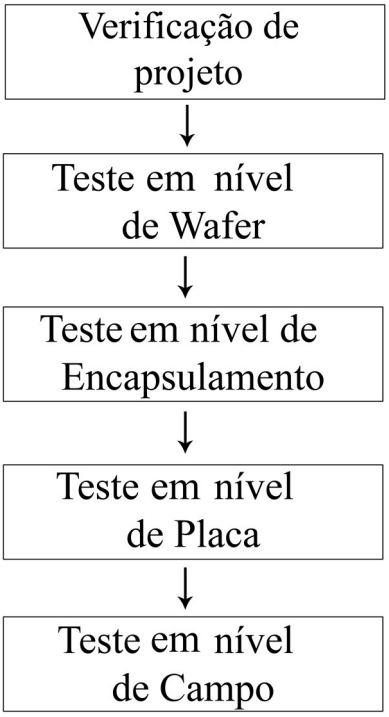
Durante o processo de manufatura os CIs estão susceptíveis a defeitos. A título de exemplo, em um CI com um grande número de transistores integrados, partículas terceiras, como um pequeno grão de poeira pode criar um curto entre duas interconexões, ou seja, um defeito em ponte, durante a etapa litográfica.

No âmbito de CIs digitais, o procedimento para teste de um CI pode ser descrito da seguinte maneira:

1. Aplica-se uma sequência de padrões de teste nas entradas do CI observando nas saídas a resposta para cada padrão. Uma sequência de padrões de teste é representada por uma sequência *bits*.
2. As respostas obtidas no ponto anterior são comparadas com as respostas para um CI sem falhas, em que, falha é a representação de um defeito a nível lógico. Se todas as respostas forem iguais o circuito é considerado bom/sem falhas (*Golden*/*fault-free circuit*), do contrário o circuito é considerado com falhas (*faulty-circuit*) (SOUZA, 2005).

Visando garantir que não haja defeitos nos CIs, o processo de testes é realizado em diferentes etapas no processo de manufatura: verificação de projeto, teste a nível de *wafer*, teste a nível de encapsulamento, teste a nível de placa, e teste a nível de campo (WEST; HARRIS, 2009; WANG; WU; WEN, 2006), mostradas na Figura 2.1.

FIGURA 2.1 – ETAPAS DO PROCESSO DE MANUFATURA EM QUE SE REALIZAM TESTES.



FONTE: DO AUTOR.

Na etapa de verificação do projeto, o teste é realizado através de sínteses computacionais para verificar a funcionalidade do circuito. Isto é feito utilizando linguagens de descrição de *hardware* (*Hardware Description Language* - HDL) como VHDL, Verilog, C ou C++. Os testes nas etapas restantes são realizados diretamente no CI (denominado teste em *Real-Transfer Level* – RTL) (WEST; HARRIS, 2009). Este trabalho possibilita o uso das sequências otimizadas em qualquer nível de realização do teste, podendo ser aplicada em nível RTL através do equipamento de teste automático ou técnicas de testabilidade, como também em nível funcional de verificação.

Os testes são realizados em diferentes níveis visando reduzir os custos necessários para se testar um CI, pois, esses custos aumentam em uma ordem de magnitude de dez vezes de um nível para o outro de acordo com a regra dos dez (WANG; WU; WEN, 2006). Dessa forma, se o teste a nível de *wafer* tem custo X, em nível de encapsulamento o custo é 10X, em nível de placa 100X, e a nível de campo o custo é de 1000X (WEST; HARRIS, 2009; WANG; WU; WEN, 2006).

No decorrer dos anos, o avanço tecnológico no desenvolvimento de transistores permitiu uma maior integração de funções em um mesmo CI decorrendo no desenvolvimento de sistemas de alto desempenho no chip (SoC) e de menor custo. Porém, junto a isso, deu-se também o aumento da complexidade e dos custos para realizar os testes (SOUZA, 2005; WANG; WU; WEN, 2006; KIRAN, et al., 2015). Um exemplo disto, é a dissipação de potência tolerada por um CI, pois em modo de testes a quantidade de potência dissipada é maior que em modo normal de operação.

Logo, se não houver um controle da potência dissipada na execução do teste, há grande riscos de o teste acarretar em danos ao CI, e consequentemente reduzir o rendimento (*yield*) da produção, dado de acordo com a Equação 2.1 (WEST; HARRIS, 2009; WANG; WU; WEN, 2006).

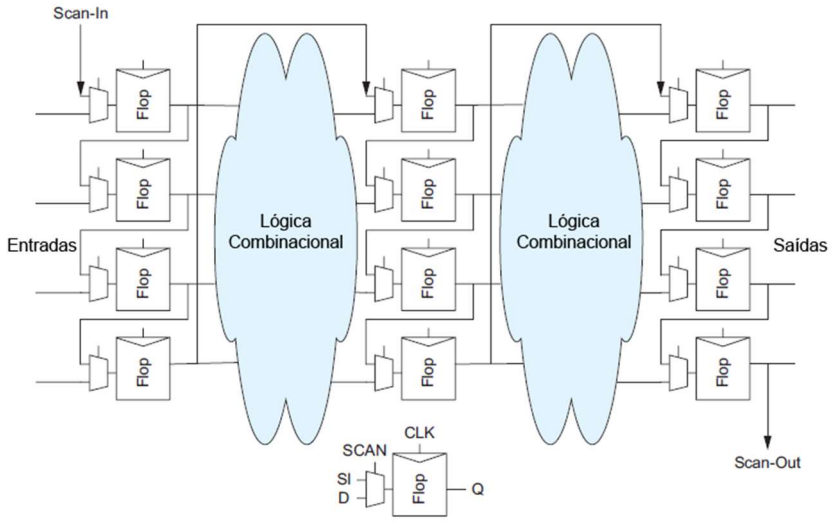
$$Rendimento\ (yield) = \frac{Número\ de\ CIs\ sem\ falhas}{Número\ total\ de\ CIs} \tag{2.1}$$

Os testes são realizados em um equipamento de teste automático (*Automatic Test Equipment* - ATE). Entretanto, como o ATE é um equipamento caro e que requer um alto investimento (SOUZA, 2005; MARGADE, 2015), técnicas foram criadas ainda na etapa de projeto do CI visando testabilidade (*Design-for-Testability* - DFT). Estas, permitem a controlabilidade e observabilidade do circuito a ser testado (WANG; WU; WEN, 2006). A controlabilidade consiste em definir o nó de um circuito para um valor específico desejado (0 ou 1), e a observabilidade em observar o valor atual de um nó. As técnicas de testabilidade mais empregadas são: *scan-test, boundary-scan test* e *built-in self-test* (WEST; HARRIS, 2009), abordadas a seguir.

Scan-test: nesta técnica, os *scan registers* são posicionados em cadeia, dando origem a um registrador de deslocamento de tamanho *N* nomeado *scan chain*. *Scan register* é o nome dado aos *flip-flops* precedidos por um multiplexador, que possibilita o uso normal do *flip-flop* quando o circuito está em modo normal de operação, ou o uso como registrador de deslocamento quando o circuito está em modo de teste, como mostrado na Figura 2.2. Note que os *scan registers* utilizados, são os já existentes no CI. A permuta entre os modos de operação é feita pela chave seletora *SCAN* presente nos multiplexadores. Em modo normal de operação, o valor da entrada D é inserido

no circuito, e em modo de teste o valor da entrada *SCAN-IN* é inserido na *scan-chain*, e após *Y* ciclos de *clock*, é inserido no circuito.

FIGURA 2.2 ARQUITETURA BÁSICA DO *SCAN-TEST*.

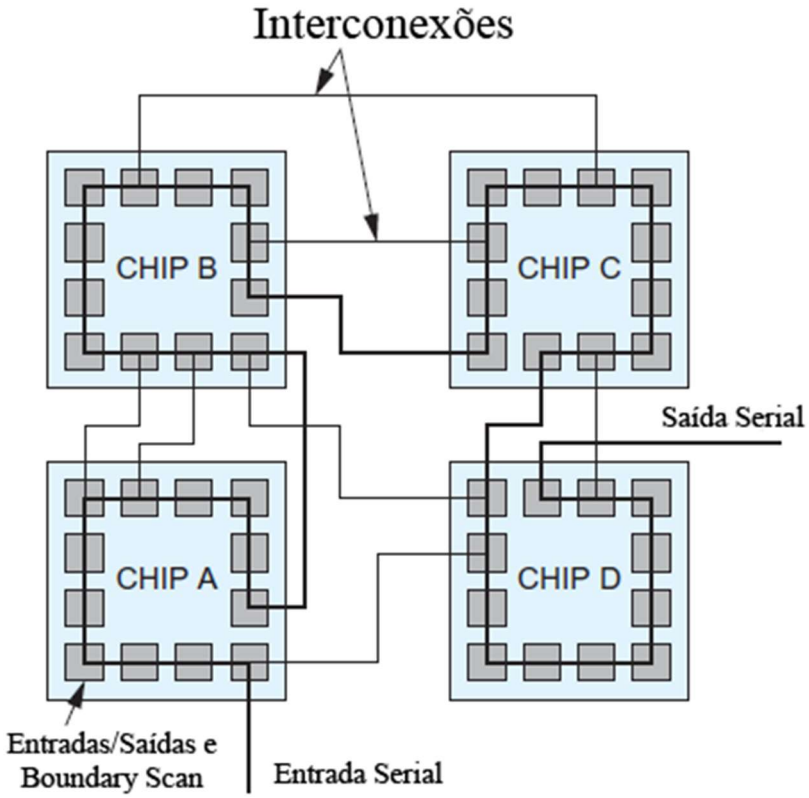


FONTE: (WESTE, 2009).

Dentre as vantagens do *scan-test* estão a alta controlabilidade e observabilidade, assim como a possibilidade de testar diversos circuitos diferentes ao formar uma única *scan chain* (a entrada *SCAN-IN* de um circuito conectada a saída *SCAN-OUT* do circuito anterior) entre esses. Entretanto, o tempo de execução do teste é alto por realizado de forma serial, assim como também há o aumento da sobreárea de *hardware* para adição de um pino extra (*SCAN-IN*), sendo considerados como as desvantagens dessa técnica.

Boundary-scan test: nesta técnica, *scan registers* são adicionados a cada pino de entrada e saída do CI e conectados entre si de forma serial formando um registrador de deslocamento, controlados por um elemento de controle (*Test Access Port* - TAP), como pode ser visto na Figura 2.3. Esta técnica é amplamente empregada no teste de placas de circuitos impressos por proporcionar uma alta controlabilidade e observabilidade dos pinos, e por permitir o teste de vários CIs na mesma placa.

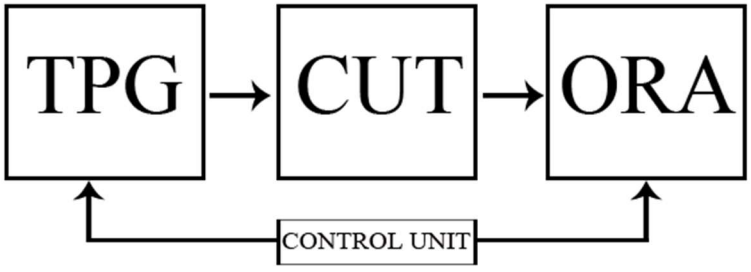
FIGURA 2.3 - ARQUITETURA BÁSICA *BOUNDARY-SCAN TEST*.



FONTE: (WESTE, 2009).

Built-in Self-test (BIST): nesta técnica, são inseridas funções de teste no próprio CI, permitindo seu auto teste. Os padrões de teste são gerados no próprio CI junto a análise das respostas. Um BIST, como ilustrado na Figura 2.4, é composto por um gerador de padrão de testes (TPG), uma unidade de controle (*Control Unit*) e um analisador de respostas (ORA), responsáveis por realizar o teste do circuito (CUT) (HUANG; CHOU; LI, 2010).

FIGURA 2.4 - ARQUITETURA BÁSCIA DE UM BIST.



FONTE: DO AUTOR.

Algumas vantagens oferecidas pelo BIST são: o teste na frequência de operação do circuito (*at-speed test),* a possibilidade de realizar teste *on-line*, e proporcionam a detecção de um grande número de falhas. Em contrapartida, há um aumento de sobreárea de *hardware* para inserir o BIST (SOUZA, 2005; HUANG; CHOU; LI, 2010; LIU et al., 2014).

As falhas de um CUT são expostas quando estes são estimulados por uma sequência de padrões de teste $S_T = \{T_1, T_2, T_3, ..., T_N\}$, em que cada padrão *Ti* pode ou não expor falhas. Para a técnica de BIST, os tipos de TPG amplamente empregados são os pseudoaleatórios e determinísticos (SOUZA, 2005; KIRTHI; SAMSON, 2014; SINGH; KUMAR; BASSI, 2014; HUSSAIN; PRIYA, 2013).

O TPG pseudoaleatório gera uma sequência pseudoaleatória a partir de um valor inicial, no qual todos os padrões pseudoaleatórios são gerados a partir dessa semente, enquanto o TPG determinístico gera uma sequência pré-estabelecida. Para cada padrão *Ti* de uma sequência é obtido uma resposta *Ri* para uma sequência de padrões resposta $S_R = \{R_1, R_2, R_3, ..., R_N\}$. Estes padrões são utilizados para avaliar se o circuito possui falhas ou não. Essa análise é feita comparando-os com as respostas esperadas para um circuito bom. Um método de análise de respostas bastante empregado é o de análise por assinatura, em que os padrões resposta *Ri* são comprimidos (redução do número de *bits*) a um padrão de comprimento consideravelmente reduzido proporcionando um menor uso de sobreárea de hardware (BUSHNELL; AGRAWAL, 2002).

É importante ressaltar que os padrões de teste, ao estimularem um CUT, acarretam numa dissipação de potência elevada devido à baixa correlação entre si (KIRAN; YELLAMPALLI, 2014; NAYANA; YELLAMPALLI; HARISH, 2014). Uma vez que um CI possui um limite de potência suportada (WEST; HARRIS, 2009), danos podem ser causados ao circuito durante a execução do teste, provocando o aumento dos custos de produção. Diante disto, a dissipação de potência deve ser mantida abaixo os limites suportados pelo CI durante a execução de todo o teste (PAPA; GARBOLINO, 2011). Assim, a fim de reduzir as chances de danificar um circuito durante o processo de teste é necessário gerar sequências de padrões de teste de alta correlação (KIRAN, et al., 2015; MARGADE, 2015; RONGHUI; XIAOWEI; YUNZHAN, 2003; ENMIN; SHENGDONG; WENKANG, 2007; NOURANI; TEHRANIPOOR, AHMED, 2008).

2.1.1 GERADORES DE TESTE

Os geradores de padrões de teste, como visto nas subseções anteriores, são responsáveis por gerar estímulos para verificar se o CI está livre de falhas. As falhas expostas durantes os testes, são definidas com base em um modelo de falhas (WANG; WU; WEN, 2006). Os modelos de falhas mais populares são:

Stuck-at faults: as falhas são modeladas fixando os nós do circuito em um valor. Denomina-se *stuck-at-0* o nó do circuito fixo no 0V (*ground*), e *stuck-at-1* o nó fixo no V_{CC} (tensão de alimentação).

Short-circuit e Open-circuit faults: as falhas são modeladas a nível de transistor. Denomina-se *short-circuit* um terminal de um transistor conectado a algum outro terminal do circuito, e *open-circuit* quando um terminal de um transistor não está conectado a nenhum ponto do circuito.

Bridging fault: as falhas são modeladas através da junção de duas ou mais interconexões.

Delay fault: as falhas são modeladas para verificar se o tempo de resposta para a propagação de uma dada entrada, será maior que o esperado.

As falhas, podem ser consideradas de fácil e difícil detecção de acordo com sua detectabilidade (quantidade de padrões de teste que expõe a falha). Essas, são de fácil detecção quando um grande número de padrões consegue detectar a falha, ou difícil detecção quando um pequeno número de padrões consegue detectar a falha.

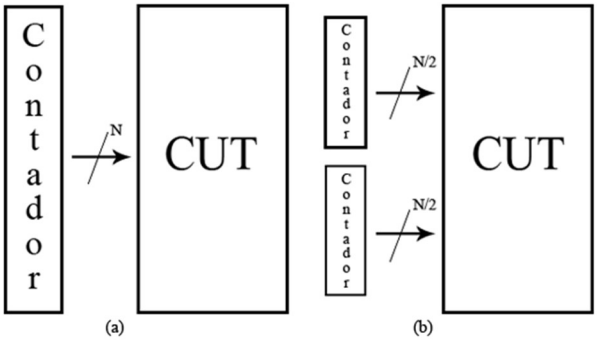
Visando atingir uma alta cobertura de falhas (*fault coverage*), dada de acordo com a Equação 2.2, e com o menor tempo necessário para execução do teste assim como uma sobreárea de *hardware*, diferentes tipos de TPGs foram projetados, descritos nas subseções a seguir.

$$Cobertura\ de\ Falha \quad \frac{N\acute{u}mero\ de\ falhas\ detectadas}{N\acute{u}mero\ total\ de\ fal \quad estimadas} \tag{2.2}$$

2.1.2 GERADORES EXAUSTIVOS

Na execução de testes exaustivos, o TPG gera uma sequência de 2^N padrões de teste, em que *N* é número de entradas do CUT, como mostra a Figura 2.5 (a). Estes TPGs proporcionam uma alta cobertura de falhas (100% garantido), ou seja, detectam um grande número de falhas. Contadores são amplamente empregados como TPG exaustivo. No entanto, para circuitos com um grande número de entradas, o tempo necessário para realizar o teste se torna excessivo, já que neste tipo de TPG é gerado a máxima sequência de vetores (SOUZA, 2005; WEST; HARRIS, 2009; WANG; WU; WEN, 2006). Uma solução para reduzir o tempo de teste é o TPG pseudoexaustivo, ilustrado na Figura 2.5 (b), que consiste em particionar o número de entradas conectadas ao TPG.

FIGURA 2.5 - CONTADOR EMPREGADO COMO TPG EXAUSTIVO.



FONTE: DO AUTOR

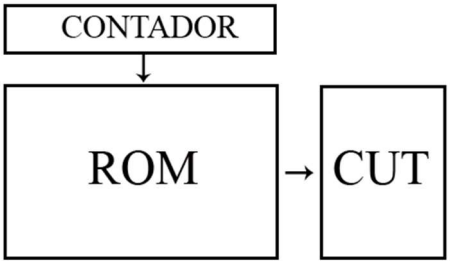
2.1.3 GERADORES DETERMINÍSTICOS

Nos geradores determinísticos, a sequência de padrões de teste é previamente gerada computacionalmente (ATPG), em que, para cada falha é gerado um padrão de teste (SOUZA, 2005). Um ATPG é uma ferramenta utilizada para encontrar a sequência de padrões de testes que expõem as falhas de um CUT. Alguns ATPGs grautitos são o TetraMAX® (Disponível em: e o Atalanta-ATPG. Segundo

Souza (2005) a vantagem dos TPGs determinísticos é a alta cobertura de falhas em um reduzido tempo de teste, pois são inseridos no CUT somente padrões que detectam falhas. Em contrapartida, esses geradores consomem uma grande sobreárea de *hardware* para armazenar a sequência (basicamente uma memória *ROM* e um contador para endereçamento, como pode ser visto na Figura 2.6).

A fim de solucionar o problema de sobreárea, diversas arquiteturas de TPGs determinísticos baseados em LFSR e em autômatos celulares (*Cellular Automata* - CA) foram propostas (SUDIREDDY; KAKADE; KAGARIS, 2008; CAO et al., 2013).

FIGURA 2.6 - ARQUITETURA BÁSICA DE UM TPG DETERMINÍSTICO.

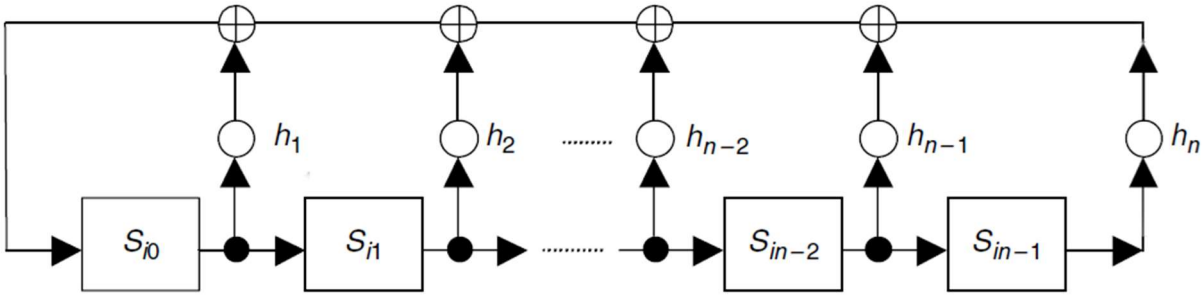


FONTE: DO AUTOR.

2.1.4 GERADORES PSEUDOALEATÓRIOS

Geradores pseudoaleatórios geram sequências de padrões de teste pseudoaleatórios. Normalmente esses geradores são baseados em LFSR, devido a estrutura simples, composta apenas por registradores (*D flip-flops*) e portas OU-EXCLUSIVO (*XOR*), ocupando pequeno espaço de sobreárea (SOUZA, 2005), e atingindo uma alta cobertura de falhas (NAIM; CHANDEL, 2014). Contudo, para expor falhas de difícil detecção, um alto número de vetores é necessário, consequentemente o tempo para realizar o teste é longo. Na Figura 2.7 é mostrado a arquitetura básica de um LFSR.

FIGURA 2.7 - ARQUITETURA BÁSICA DE UM LFSR.



FONTE: *VLSI TEST PRINCIPLES AND ARCHITECTURES*.

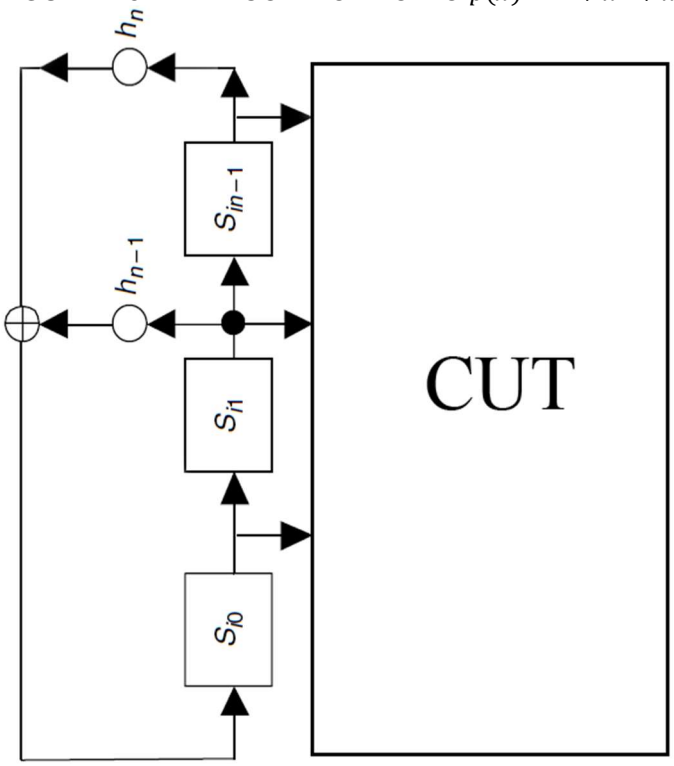
Como pode ser visto na figura acima, a realimentação é feita através da lógica XOR, em que, denomina-se *tap* a interconexão entre registrador e porta XOR. O deslocamento é realizado a partir de pulsos de *clock*, em que o estado atual de um registrador é o estado anterior de seu antecessor. De acordo com (WANG; WU; WEN, 2006), um LFSR pode ser descrito através de um polinômio característico, dado de acordo com a Equação 2.3 a seguir:

$$p(x) = 1 + h_1x^1 + h_2x^2 + \cdots + h_{n-2}x^{n-2} + h_{n-1}x^{n-1} + x^N \qquad (2.3)$$

Em que, se $hi = 1$, existe um *tap* entre o *flip-flop i* e uma lógica XOR e, $hi = 0$, não existe. As sequências de padrões de teste geradas em LFSR têm o comprimento dado de acordo com o polinômio característico. Se o polinômio aplicado ao LFSR for irredutível, a máxima sequência é gerada, denominando-se polinômio primitivo (WANG; WU; WEN, 2006). A sequência máxima gerada em um LFSR tem comprimento de até 2^N-1 padrões, pois o estado zero (todos os *bits* iguais a 0) não permite obter nenhum outro padrão a partir dele, por isto a sequência máxima não é de 2^N vetores. Algumas aplicações comuns de LFSRs são: geração de sequências pseudoaleatórias (*Pseudorandom number generator* - PRNG), comunicação serial-paralelo e vice-versa, codificação e decodificação.

Na Figura 2.8 é mostrado um LFSR com polinômio $p(x) = 1 + x^2 + x^3$ aplicado como TPG pseudoaleatório em um circuito qualquer com três entradas (N = 3) a ser testado.

FIGURA 2.8 - LFSR COM POLINOMIO $p(x) = 1 + x^2 + x^3$.



FONTE: DO AUTOR.

O primeiro padrão carregado nos registradores é denominado semente (*seed*) (PANDA; RAJPUT; SHUKLA, 2012) e todos os demais padrões são gerados a partir deste primeiro. Considere o padrão $T = (1, 0, 0)$ como semente aplicado ao LFSR da figura acima, e que 1 *bit* é deslocado a cada pulso de *clock*. Os padrões de teste gerados são mostrados Na Tabela 2.1.

Note que a sequência se repete após sete padrões serem gerados, assim o LFSR tem um período de sete, e após este período a sequência sempre se repetirá (WANG; WU; WEN, 2006), pois o mesmo gera sequências cíclicas caso não haja condições de parada. E, como foram gerados 2^3-1 vetores de teste antes da sequência se repetir, o polinômio $p(x) = 1 + x^2 + x^3$ é considerado um polinômio primitivo.

TABELA 2.1 - PADRÕES DE TESTES OBTIDO PARA T=(1,0,0)

.

#	S_{i0}	S_{i1}	S_{in-1}
1	1	0	0
2	0	1	0
3	1	0	1
4	1	1	0
5	1	1	1
6	0	1	1
7	0	0	1
-	1	0	0

2.2 DISSIPAÇÃO DE POTÊNCIA EM CIRCUITOS INTEGRADOS

A principal causa da alta dissipação de potência em um CI durante o teste é a baixa correlação entre padrões de testes. Estes padrões, quando gerados com TPGs pseudoaleatórios, é pretendido obter vetores de teste que se aproximem ao máximo de uma sequência verdadeiramente aleatória, pois, dessa forma é possível obter uma alta cobertura de falhas em um menor tempo de teste, enquanto com o TPG determinístico, o CUT é estimulado somente com os padrões detectores de falhas, por este motivo, os vetores são independentes um de outro, e proporcionam uma alta cobertura de falhas em um curto tempo de teste. No entanto, essa independência resulta na redução da correlação existente entre os vetores. De tal forma, ao estimular o CUT com um novo vetor, há uma alta probabilidade de que um grande número de nós chaveie, resultando no alto consumo de energia e altos picos de potência (NICOLICI; AL-HASHIMI, 2003). Outras causas para o aumento da dissipação de potência, é a execução do teste em paralelo, ou seja, estimular todo o CUT simultaneamente, como também, a adição de técnicas de DFT, dissipam

potência ao gerar os padrões de teste, por exemplo (KASUNDE; SHIVAKUMAR, KURIAN, 2013).

Para entender o processo de dissipação de potência em um CI e em como estima-la, alguns conceitos fundamentais são indispensáveis (WEST; HARRIS, 2009):

Primeiro, potência instantânea consumida é o produto da corrente e tensão no elemento em um determinado instante de tempo, e a energia total consumida é o consumo de potência em um intervalo de tempo. Assim, a potência média consumida em um intervalo de tempo *T*, é obtida através da a Equação 2.4.

$$P_{MÉDIA} = \frac{E}{T} = \frac{1}{T}\int_{0}^{T} P(t)dt \tag{2.4}$$

De acordo com (YEAP, 1998; NICOLICI; AL-HASHIMI, 2003), a potência dissipada em um CI é resultante da potência dinâmica e estática, como visto na Equação 2.5, no qual, a potência dinâmica é oriunda da atividade de chaveamento de um circuito, ou seja, quando ele está ativo e executando uma função, enquanto a potência estática é dissipada quando o circuito está em modo de espera, ou seja, ativo, mas não executa nenhuma função (*standby*).

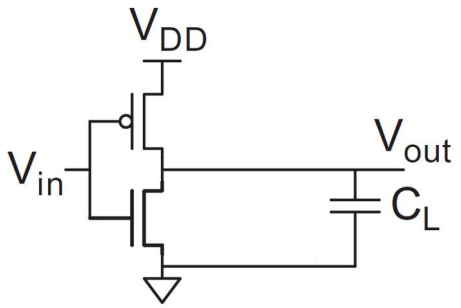
$$P_D = P_{DINÂMICA} + P_{ESTÁTICA} \tag{2.5}$$

De acordo com (WEST; HARRIS, 2009), a principal fonte da potência dinâmica é a carga e a descarga das capacitâncias de cargas somado a corrente de curto-circuito (*short-circuit current*) quando ambos os canais dos transistores pMOS e nMOS estão momentaneamente ativos, enquanto que a potência estática tem como principal fonte a corrente de fuga (*leakage*). As subseções a seguir abordam essas fontes de dissipação de potência.

2.2.1 POTÊNCIA DINÂMICA

Considere o circuito de um inversor CMOS, mostrado na Figura 2.9.

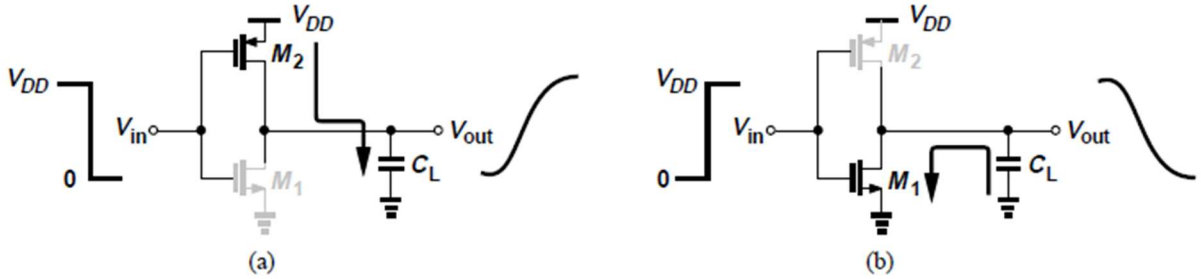
FIGURA 2.9 - CIRCUITO DE UM INVERSOR CMOS.



FONTE: CMOS VLSI DESIGN.

Ao aplicar uma transição de V_{DD} para 0 em *Vin*, a capacitância de carga (C_L) é carregada através do transistor pMOS, e descarregada no transistor nMOS ao aplicar uma transição de 0 para V_{DD} (RAZAVI, 2010), como é mostrado na Figura 2.10.

FIGURA 2.10 – CARGA E DESCARGA DA CAPACITÂNCIA DE CARGA NO INVERSOR.



FONTE: RAZAVI, 2010.

Quando a capacitância de carga C_L é carregada, a energia armazenada é obtida pela Equação 2.6, a seguir:

$$E_{C_L} = \frac{1}{2} C_L V_{DD}^2 \qquad (2.6)$$

Note, que só metade da energia fornecida (V_{DD}) é consumida ao carregar a capacitância de carga, isto ocorre porque a outra metade é dissipada no transistor M2. Dessa forma, a energia total consumida em uma transição, de acordo com a Equação 2.7 (RAZAVI, 2010), é:

$$E_{TOTAL} = \int_0^T I(t) V_{DD} dt$$

$$E_{TOTAL} = V_{DD} \int_0^T C_L \frac{dV_{DD}}{dt} dt$$

$$E_{TOTAL} = C_L V_{DD}^2 \quad (2.7)$$

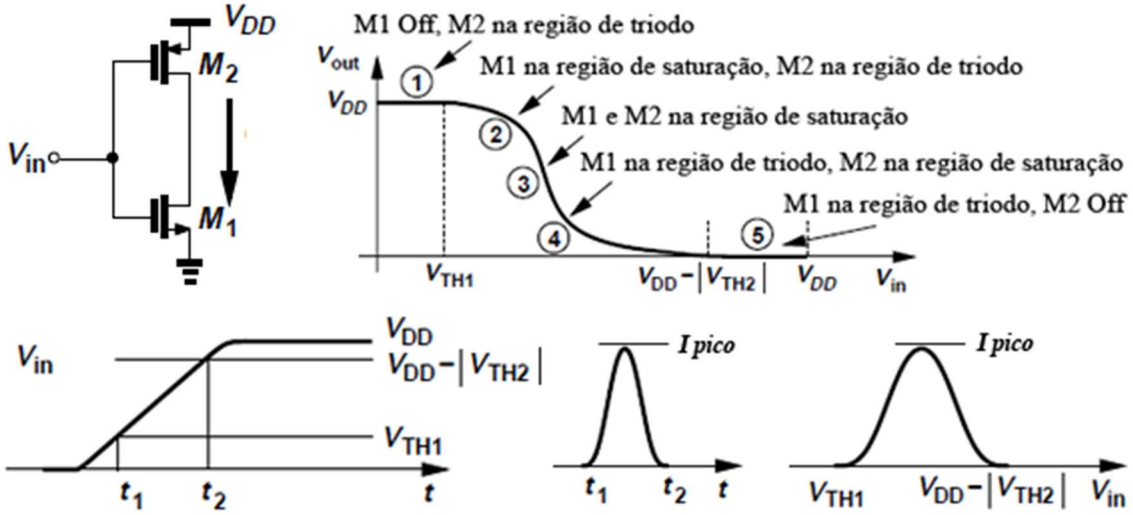
Quando o estímulo de entrada é uma transição de 0 para V_{DD}, a energia anteriormente armazenada na capacitância de carga é dissipada no transistor M1. Este fenômeno de carga e descarga da capacitância de carga caracteriza a potência de chaveamento, em que se dissipa potência ao ocorrer transição (chaveamento) nos nós do circuito (WEST; HARRIS, 2009). Em adição, devido os estímulos na entrada não serem abruptos (transições instantâneas), momentaneamente ambos os transistores M1 e M2 estarão na região de saturação, e neste intervalo de tempo, uma corrente de curto-circuito é conduzida, como mostrado na Figura 2.11. Esse fenômeno caracteriza a potência de curto-circuito, que também causa dissipação de potência quando os nós do circuito estão chaveando. Assim, a potência dinâmica pode ser obtida através da Equação 2.8, a seguir:

$$P_{DINÂMICA} = P_{CHAVEAMENTO} + P_{CURTO-CIRCU} \quad (2.8)$$

De acordo com West e Harris (2009) e Yeap (1998), considerando que o circuito do inversor, ao aplicar uma sequência qualquer de estímulos, permaneça chaveamento durante um período de tempo T_P, a uma frequência f, a capacitância de carga será carregada e descarregada $T_P f$ vezes, então a potência de chaveamento é obtida de acordo com a Equação 2.9.

$$P_{CHAVEAMENTO} = \frac{E}{T_P} = \frac{T_P f C_L V_{DD}^2}{T_P} = C_L V_{DD}^2 f \quad (2.9)$$

FIGURA 2.11 – CORRENTE DE CURTO-CIRCUITO.



FONTE: RAZAVI, 2006.

Contudo, caso alguns estímulos existentes na sequência se repitam consecutivamente, não haverá chaveamento quando inseridos, assim a frequência de chaveamento é obtida de acordo com a probabilidade de transição *α* do nó. Deste modo, a potência de chaveamento, é dada de acordo com a Equação 2.10.

$$P_{CHAVEAMENTO} = \alpha C_L V_{DD}^2 f \tag{2.10}$$

Note que durante a transição de 0 para V_{DD} em *Vin*, no intervalo de tempo t_1 a t_2 (ver Figura 2.11), quando *Vin* é ligeiramente maior que V_{TH1} (ponto 2), M1 começa a entrar na região de saturação permitindo a passagem de uma pequena quantidade de corrente do terminal V_{DD} para o terminal terra (*GND*). Ao se aproximar do ponto de gatilho do inversor (ponto 3), ambos os transistores entram em saturação e a passagem de corrente é máxima (*Ipico*), até *Vin* finalmente atingir o ponto 5 impedindo completamente a passagem de corrente. O valor de *Ipico* é obtido de acordo com a Equação 2.11.

$$Ipico = \frac{1}{2}\mu_n Cox \left(\frac{W}{L}\right)_1 \left(\frac{V_{DD}}{2} - V_{TH1}\right)^2 \left(1 + \lambda_1 \frac{V_{DD}}{2}\right) \tag{2.11}$$

2.2.2 POTÊNCIA ESTÁTICA

A corrente de fuga (*leakage*) é responsável pela potência estática dissipada. Circuitos com tecnologia CMOS tem como principais fontes dessa potência a corrente de fuga de s*ubthreshold,* do *gate*, e da corrente de polarização reversa de junção PN (WEST; HARRIS, 2009).

Segundo Razavi (2010), normalmente um transistor é ativado quando V_{GS} (tensão *source-gate*) é maior ou igual a V_{TH} (tensão de *threshold*), contudo, a condução é feita de forma gradual e não instantânea, portanto, o transistor conduz uma pequena corrente mesmo quando V_{GS} é menor que V_{TH} caracterizando a corrente de fuga de s*ubthreshold*. A corrente de fuga do *gate* é caracterizada pela espessura do dielétrico. Ocorre quando uma tensão V_G é aplicada no *gate* e cargas conseguem passam através do dielétrico. E a corrente de polarização reversa de junção é caracterizada ao conectar o *source* ou *drain* a uma carga diferente do substrato, e tem magnitude dependente diretamente da temperatura, da corrente de polarização e da área do dielétrico. Assim, a potência estática pode ser obtida de acordo com a Equação 2.12 (WEST; HARRIS, 2009) a seguir:

$$P_{ESTÁTICA} = V_{DD}\left(I_{SUB} + I_{GATE} + I_{JUNCTION}\right) \tag{2.12}$$

Entretanto, como em modo de testes o CUT é submetido a sequências de padrões de teste que o mantem em funcionamento constante, e a potência estática é geralmente desconsiderada em circuitos CMOS devido a pequena magnitude (RAZAVI, 2010) em relação a potência dinâmica. Então, dissipação de potência em CIs durante o teste é resultante da potência dinâmica (NICOLICI; AL-HASHIMI, 2003).

Assim, de acordo com as equações apresentadas até aqui nesta seção, fica claro a influência direta da atividade de chaveamento na dissipação de potência para um CI em modo de testes. De tal modo, ter um controle do chaveamento do circuito é importante para redução do consumo de potência. A seguir é descrito a forma de computar a atividade de chaveamento utilizada neste trabalho.

2.2.3 ATIVIDADE DE CHAVEAMENTO PONDERADA

Como descrito na subseção anterior, a principal fonte de dissipação de potência em um CI em modo de testes é a potência dinâmica, que é diretamente proporcional a atividade de chaveamento e corrente de curto-circuito (KASUNDE; SHIVAKUMAR, KURIAN, 2013). É importante lembrar, que diante da complexidade atual dos CIs, portas lógicas geralmente possuem mais de um elemento conectado em sua saída (*fan-out*), e que isto influência no consumo de potência, então o consumo de energia em um nó pode ser estimado de acordo com a Equação 2.13. (ALOUL; SAGAHYROON, 2006)

$$E_{CHAVEAMENTO} = \frac{1}{2} C_L V_{DD}^2 S_G \tag{2.13}$$

em que, C_L é a capacitância de carga, V_{DD} a tensão de alimentação, e S_G o número de chaveamentos no nó de saída de uma porta lógica *G*.

A capacitância pode ser aproximada pelo número de *fan-out* da porta, portanto S_G pode ser obtido de acordo com a Equação 2.14

$$S_G = F_i * SA \tag{2.14}$$

em que, *Fi* é o número de *fan-out* da porta lógica e *SA* a atividade de chaveamento do nó *i.*

Então, calculando a atividade de chaveamento ponderada (*Weighted Switching Activity* - WSA) de um nó, ou seja, o número de transições, é possível estimar o consumo de energia de um circuito (MANICH; FIGUERAS, 1997).

O WSA gerado em uma sequência de teste, de acordo com Kasunde; Shivakumar, Kurian, (2013); Aloul; Sagahyroon, (2006) e Manich; Figueras, (1997), pode ser obtido de acordo com a Equação 2.15.

$$WSA = \sum_{i}^{todas\ as\ portas} \left(F_i * \sum_{j=1}^{N} \left(g_i(T_t) \oplus g_i(T_{t-1}) \right) \right) \tag{2.15}$$

em que, T_t e T_{t-1} são vetores de teste aplicados no tempo t e t-1, consecutivamente na porta lógica *gi*, *N* o número de vetores de teste e *Fi* o *fan-out* da porta *gi*.

Dada a forma de obter o WSA, note que o termo WSA agora representa o número de transições de um nó levando em considerado o número de *fanout,* enquanto que o termo “número de transições” representa somente o número de vezes que o valor de um nó foi alterado.

É importante ressaltar, que boa parte da área de silício de um CI é ocupada com interconexões, portanto, atrasos de propagação (*propagation delays*) precisam ser considerados para se aproximar do valor real do WSA (ZHAO et al., 2010).

Alguns dos modelos de atraso são: modelo de atraso zero (*zero delay model*), modelo de atraso unitário (*unit delay model*) e o modelo de atraso variável (*variable delay model*) (MANICH; FIGUERAS, 1997):

Atraso zero: é considerado a propagação instantânea dos sinais, ou seja, em todas as portas lógicas, a saída responde a alterações na entrada no mesmo instante.

Atraso unitário: é considerado a mesmo valor de propagação dos sinais para todas as portas lógicas, ou seja, todas as portas possuem o mesmo tempo de atraso necessário para obter a saída para uma dada entrada.

Atraso variável: é considerado que a propagação dos sinais é diferente para cada porta lógica, ou seja, cada porta tem um atraso de propagação próprio de se obter a saída para uma dada entrada. Geralmente, o valor de atraso de cada porta é aproximado pelo seu número de *fan-out*.

Logo, a alta correlação entre vetores de teste é fundamental para evitar picos de potência elevados. Assim, neste trabalho o algoritmo genético é utilizado para ter um controle do WSA durante a execução do teste para o baixo consumo energético e baixa dissipação de potência. Para uma redução da sobreárea de *hardware* em técnicas de testabilidade, o BMA é utilizado para sintetizar um LFSR convencional sem a necessidade de alterações e/ou adições em sua estrutura, e é descrito a seguir.

2.3 ALGORITMO DE BERLEKAMP-MASSEY E TESTE DE CI

O algoritmo de Berlekamp-Massey (BMA) é bastante implementado na decodificação de diversos códigos algébricos (CASEY, 2000). A primeira aplicação feita na área de testes foi por Souza (2005), devido a possibilidade de sintetizar um

LFSR capaz de gerar uma sequência desejada sem adição de nenhuma estrutura extra ao LFSR, evitando uma maior ocupação de sobreárea de *hardware*.

Através do BMA é possível obter o polinômio característico capaz de sintetizar um LFSR, a saber: $\Lambda(x) = \Lambda_0 + \Lambda_1 x^1 + \cdots + \Lambda_L x^L$ para um LFSR de comprimento *L*, que é capaz de gerar a sequência de entrada, a partir de $\Lambda_i$coeficientes. É mostrado na Figura 2.12 o pseudocódigo do BMA.

FIGURA 2.12 – PSEUDOCÓDIGO DO BMA.

ENTRADA: $T_1, T_2, \cdots, T_n$
SAÍDA: $\Lambda(x)$
INICIALIZAÇÃO:
$\Lambda(x) \leftarrow 1$: polinômio de realimentação do LFSR
$L \leftarrow 0$: comprimento do LFSR
$B(x) \leftarrow 1$: polinômio temporário
$d \leftarrow 1, b \leftarrow 1$: variáveis temporárias
$j \leftarrow 1$: contador
Δ : discrepância
ITERAÇÃO:
Passo 1. Se $j = n$, pare. Caso contrário, faça
$\Delta = T_j - \hat{T}_j = T_j + \sum_{i=1}^{L} \Lambda_i T_{j+1-i}$
Passo 2. Se $\Delta = 0$, então $d \leftarrow d + 1$, vá para o passo 5.
Passo 3. Se $\Delta \neq 0$ e $2L > j$, então
$\Lambda(x) \leftarrow \Lambda(x) - \Delta b^{-1} x^d B(x)$
$d \leftarrow d + 1$, vá para o passo 5.
Passo 4. Se $\Delta \neq 0$ e $2L \leq j$, então
$Temp(x) \leftarrow \Lambda(x)$
$\Lambda(x) \leftarrow \Lambda(x) - \Delta b^{-1} x^d B(x)$
$L \leftarrow j + 1 - L$
$B(x) \leftarrow Temp(x) \qquad b \leftarrow \Delta \qquad d \leftarrow 1$
Passo 5. $j \leftarrow j + 1$, retorne ao passo 1.

FONTE: SOUZA, C.P. (2005).

Considere uma sequência de vetores $S_T = \begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_k \end{bmatrix}$, em que $1 \leq T \leq k$. Para obter o menor LFSR, a entrada inserida no BMA precisa antes passar por um **processo de serialização**, no qual, a sequência S_T , após serializada, será:

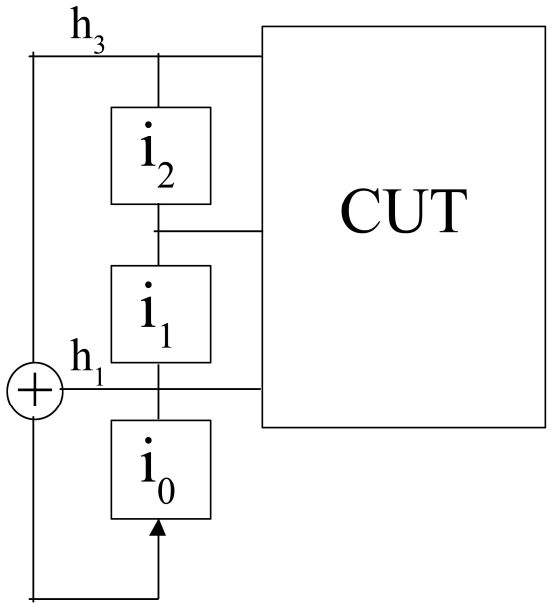
$$S_T = (T_1 T_2 T_3 \ldots T_k)$$

Segundo Souza, (2005), "*a cada iteração, o algoritmo é iniciado calculando a discrepância, ou seja, a diferença entre o valor 'atual' e o valor 'predito' do j-ésimo elemento de T. Logo* $\Delta = T_j - \hat{T}_j$ *em que* $1 \leq j \leq n$ *e* $\hat{T}_j = -\sum_{i=1}^{L} \Lambda_i \, T_{j+1-i}$"

O comprimento do LFSR, definido pelo grau do polinômio, permanece inalterado se a discrepância for igual a zero ($\Delta = 0$), do contrário o comprimento *L* pode ser alterado. Se $2L > j$ somente o polinômio é alterado e *L* permanece inalterado, no entanto, se $2L \leq j$ o comprimento do LFSR é alterado de modo a gerar a sequência até esta iteração.

A título de exemplo, considere $S_T = \begin{bmatrix} 100 \\ 010 \\ 101 \end{bmatrix}$, após o processo de serialização tem-se: $S_T = (001010101)$. Logo, o polinômio obtido será: $p(x) = 1 + h_1 x^1 + h_3 x^3$, e o respectivo LFSR é mostrado na Figura 2.13.

FIGURA 2.13 – LFSR SINTETIZADO A PARTIR DO BMA.



FONTE: DO AUTOR.

Na Tabela 2.2 é mostrada a sequência gerada pelo LFSR. Note que em meio a sequência estão os padrões da sequência S_T.

TABELA 2.2 – PADRÕES DE TESTES OBTIDOS DO LFSR SINTIZADO PELO BMA.

i0	**i1**	**i2**
0	**0**	**1**
1	0	0
1	1	0
1	1	1
0	1	1
1	**0**	**1**
0	**1**	**0**

2.4 REFERÊNCIAS BIBLIOGRÁFICAS

Diante dos fatores relevantes no processo de teste de CIs citados na seção 1.1, diversos trabalhos veem sendo propostos ao longo dos anos com intuito de evitar e/ou eliminar o risco de danos ao CI durante o processo sem perder sua eficiência.

O rendimento (*yield*) do processo de manufatura é dado de acordo com o número de CIs sem defeitos em relação ao número total de CIs fabricados. Dado que o processo de se testar o CI é uma etapa existente entre partes do processo de manufatura, é preciso garantir que um CI sem defeitos não seja danificado ao ser testado (WEST; HARRIS, 2009).

Diversas metodologias foram adotadas para atender os quesitos de relevância no processo de teste, como por exemplo, alguns trabalham buscam reduzir o consumo energético durante o teste através de métodos que reduzem a frequência de operação do CI durante a realização do teste. Técnicas que “particionam” o CI, ou seja, realizam o teste de partes (núcleos) do CI de forma independente, evitando o teste em paralelo do CI por completo também são utilizadas como solução para redução do consumo energético. Há também, o desenvolvimento de arquiteturas de teste que aumentam a correlação entre padrões de testes para reduzir o chaveamento interno do circuito (NAIM, CHANDEL, 2014).

Contudo, reduzir a frequência de realização do teste aumenta consideravelmente o tempo de teste, o mesmo acontece para testes “particionados”.

Logo, reduzir o chaveamento dos nós internos do circuito é o método mais adotado para obter uma redução da dissipação de potência sem afetar de forma abrupta os demais quesitos objetivados para com testes de CIs.

Kiran *et al.* (2015) propôs uma arquitetura de testes em que é substituído o uso de um LFSR como gerador de padrões, por um sintetizado utilizando os próprios elementos do CI. Entretanto, é necessário a adição de multiplexadores a cada *flip-flop* para ser utilizado como registrador de deslocamento. A arquitetura proposta obteve redução média do consumo de potência de 22,3%, contudo há um aumento da sobreárea de *hardware*, em que, em relação a este trabalho, busca-se reduzir.

Nourani et al. (2008), propôs um gerador de teste baseado em LFSR, no qual é inserido para cada entrada e saída de cada registrador de deslocamento do LFSR uma estrutura nomeada *R-Injection*. Esta estrutura é constituída por 3 portas lógicas *E* (*and*), uma porta lógica OU (*or*), e três entradas. O *R-injection* tem a função de inserir no circuito novos padrões intermediários entre dois padrões normalmente gerados no LFSR. A primeira entrada do *R-Injection* é conectada a entrada de um registrador, a segunda à saída deste mesmo registrador a qual a primeira entrada é conectada, e a terceira entrada é conectada a saída do último registrador do LFSR. Assim, a cada pulso de *clock*, se o valor do estado atual do registrador for igual ao valor do estado anterior, este valor é inserido na entrada do circuito, entretanto, se o estado atual for diferente do anterior, é inserido o estado atual do último registrador. Dessa forma, foi possível aumentar a correlação entre os padrões de teste. Em contrapartida, também há um aumento da sobreárea de *hardware* ao adicionar estruturas ao LFSR. A arquitetura proposta, resultou na redução média do consumo de potência em 77% e até 49% nos picos de potência.

Uma técnica similar a proposta por Nourani et al., é feita por Hussain e Priya, (2013) e Chethan e Lakkannavar (2013), mas foram utilizados multiplexados no lugar da estrutura *R-Injection*. Uma redução média do consumo de potência de até 25% é obtida, em contrapartida, como os anteriormente citados, também há o aumento da sobreárea de *hardware.*

Diversos outros trabalhos, apresentam propostas similares, mas todas requerem adição de estruturas adicionais a LFSRs, logo, há um aumento de sobreárea de *hardware* em todas (KIRTHI; SAMSON, 2014; CHANDET at al., 2010; WANG, 2007, QUIGLEY, 2009; HARISH, 2015; SUPRIYA; Rekha, 2013) como também o

aumento do consumo energético na própria arquitetura, o que se busca reduzir no trabalho aqui proposto através do BMA.

Alguns trabalhos também empregaram o algoritmo genético com intuito de reduzir a dissipação de potência durante a execução dos testes.

Enmin, Shengdong e Wenkang (2007), propuseram uma divisão das entradas do CI. No qual, as entradas são separadas em grupos e o algoritmo genético é utilizado para encontrar a melhor combinação de grupos que resultem na redução do número de transições. Uma redução média de 26% a 60% nos picos de transição e uma redução média no total das transições de até 90%. Entretanto, estruturas são adicionadas a um LFSR, que resultam no aumento do tempo de teste e sobreárea de *hardware*, como também a quantidade de transições totais no CI aumenta devido as decorrentes da própria arquitetura de teste.

Anita e Vanathi (2014) aplicaram o algoritmo genético para criar padrões de teste que atinjam a cobertura de uma quantidade de falhas por eles definidas. Em adição, os padrões que cubram as falhas definidas, gerados pelo algoritmo, são reordenados baseado em diferentes métodos buscando reduzir a potência dissipada durante o teste. Dentre os métodos utilizados, os autores consideram também o de redução através da redução do número de transições nos nós. Uma redução média de transições em até 59% foi obtida, no entanto não sugere nenhum gerador de padrão que gere as sequências obtidas.

3 SIMULADOR WSA

3 SIMULADOR WSA (WSA-T)

Foi desenvolvido um simulador, nomeado **WSA-T**, com base na biblioteca de simulação de circuitos lógicos, nomeada LCS (*Logic Circuit Simulation*). LCS é uma biblioteca desenvolvida em C++, disponível gratuitamente em: http://liblcs.sourceforge.net/.

O WSA-T sintetiza circuitos combinacionais e sequenciais descritos nos formatos ISCAS85, ISCAS89 e ITC99. Estes formatos foram implementados por serem amplamente adotados como *benchmark* na área de testes de CIs (KIRAN, et al., 2015; KIRTHI; SAMSON, 2014; HUSSAIN; PRIYA, 2013; NOURANI; TEHRANIPOOR, AHMED, 2008; CHANDET at al., 2010; CHETHAN; LAKKANNAVAR, 2013; WANG, 2007; QUIGLEY, 2009; HARISH, 2015; SUPRIYA; Rekha, 2013; ENMIN; SHENGDONG; WENKANG, 2007; ENMIN; WANG, 2009; ANITA; VANATHI, 2014), como também devido ao formato simples de descrição de circuitos, no qual um usuário pode descrever qualquer circuito desejado e sintetizar na ferramenta sem um conhecimento profundo no uso de HDLs. Nominado *netlist,* os formatos ISCAS e ITC descrevem de forma textual os circuitos, definindo elementos lógicos e interconexões, como mostrado na Figura 3.1.

FIGURA 3.1 – MODELO DE NETLISTS PARA (A) ISCAS85 C17, E (B) ISCAS89 E ITC99 C17.

```
 1     1gat inpt     1   0       >sa1
 2     2gat inpt     1   0       >sa1
 3     3gat inpt     2   0 >sa0 >sa1
 8     8fan from      3gat       >sa1
 9     9fan from      3gat       >sa1
 6     6gat inpt     1   0       >sa1
 7     7gat inpt     1   0       >sa1
10    10gat nand     1   2       >sa1
  1     8
11    11gat nand     2   2 >sa0 >sa1
  9     6
14    14fan from     11gat      >sa1
15    15fan from     11gat      >sa1
16    16gat nand     2   2 >sa0 >sa1
  2    14
20    20fan from     16gat      >sa1
21    21fan from     16gat      >sa1
19    19gat nand     1   2       >sa1
 15     7
22    22gat nand     0   2 >sa0 >sa1
 10    20
23    23gat nand     0   2 >sa0 >sa1
 21    19
```

(a)

```
INPUT(G1gat)
INPUT(G2gat)
INPUT(G3gat)
INPUT(G6gat)
INPUT(G7gat)
OUTPUT(G22gat)
OUTPUT(G23gat)

G10gat = nand(G1gat, G3gat)
G11gat = nand(G3gat, G6gat)
G16gat = nand(G2gat, G11gat)
G19gat = nand(G11gat, G7gat)
G22gat = nand(G10gat, G16gat)
G23gat = nand(G16gat, G19gat)
```

(b)

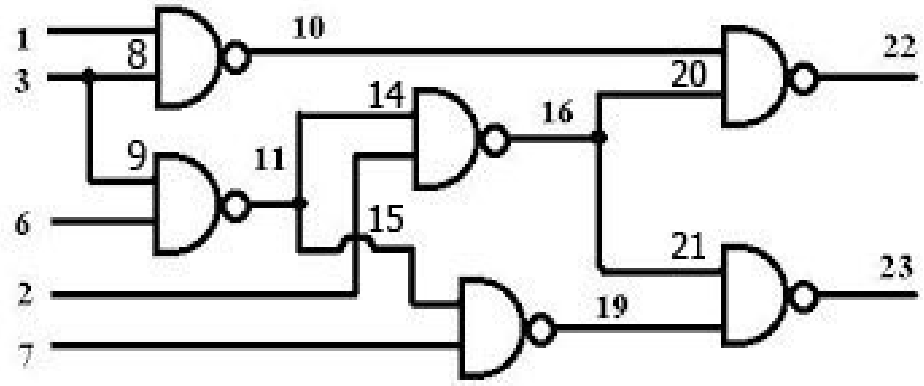
FONTE: ISCAS *NETLISTS*

Note que cada linha define um elemento do circuito. O ISCAS85 é descrito da seguinte forma: nó de saída, nome do elemento, tipo, quantidade de *fan-out,* quantidade de *fan-in,* falhas e nós de entrada, respectivamente. No qual, em relação ao tipo, tem-se: *inpts* para as entradas primárias do circuito, *from* para as ramificações (*fan-out*) ou a lógica, que para o c17 é a *nand*. Em relação as falhas, se existirem, são indicadas de acordo com o modelo de *stuck-at,* ou seja, *>sa0* indica a existência de um *stuck-at-0* e *>sa1* indica um *stuck-at-1*.

O ISCAS89 e ITC99 é descrito de uma forma mais compacta, a saber: entradas primárias nas primeiras linhas, seguidas pelas saídas primárias, respectivamente. As linhas seguintes definem as interconexões entre os elementos do circuito, descritas da seguinte forma: nó de saída, lógica, entradas, respectivamente.

O WSA-T funciona através da conversão dessas *netlists*, para o formato LCS de descrição de circuitos, que por sua vez, é executado. O fluxograma com o funcionamento do simulador é mostrado na Figura 3.3. O circuito ISCAS85 c17 mostrado na Figura 3.2, será utilizado para exemplificar o uso do simulador.

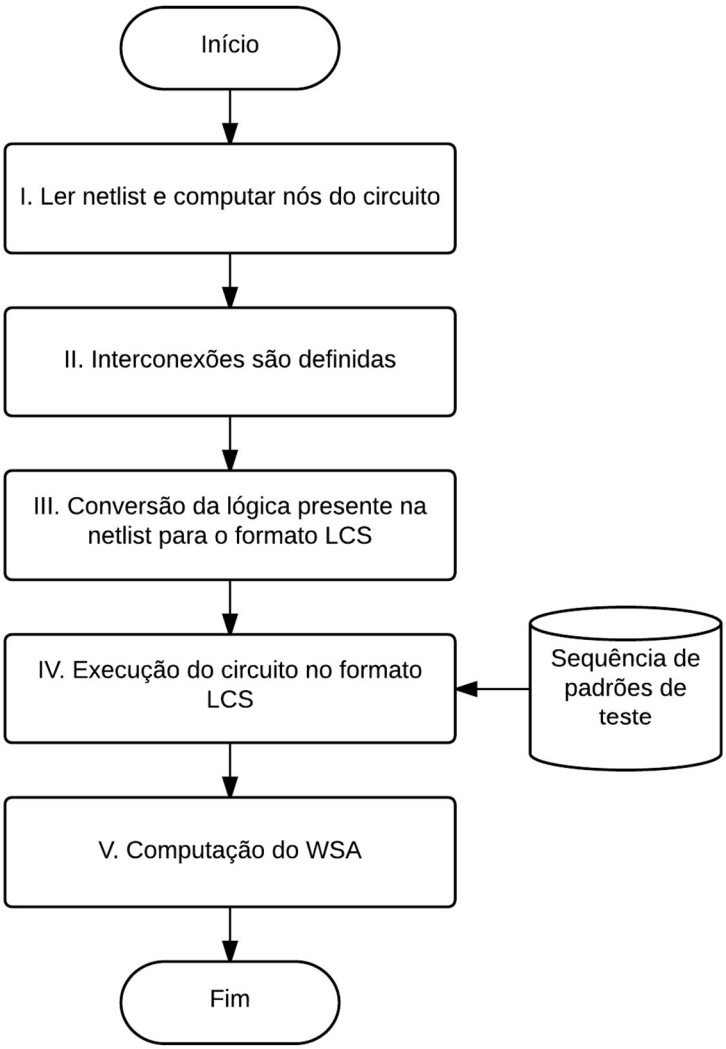
FIGURA 3.2 – CIRCUITO ISCAS85 C17.



FONTE: ISCAS NETLISTS.

O simulador é iniciado após informado a *netlist* com a descrição do circuito a ser sintetizado (neste exemplo, o c17) e a sequência de padrões de teste para estimular o circuito. Dado isto, um escaneamento é realizado na *netlist* para computar a quantidade de nós existentes no circuito.

FIGURA 3.3 – FLUXOGRAMA DO SIMULADOR WSA.

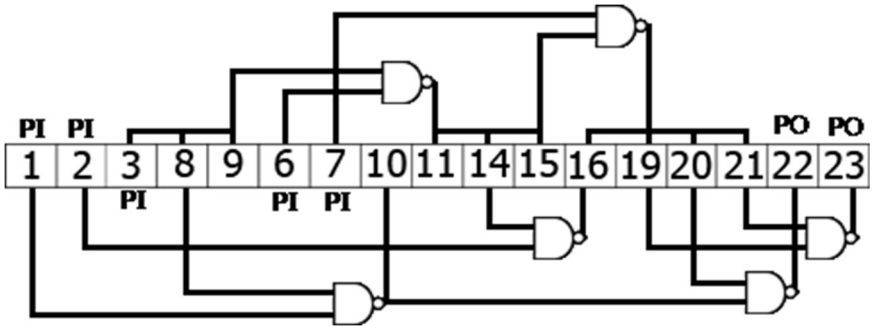


FONTE: DO AUTOR.

Após computados todos os nós (I), um vetor é criado para armazenar a informação.

O segundo passo, é criar as interconexões para cada nó existente na *netlist* (II). A seguir, na Figura 3.4, é mostrado o c17 aplicado sobre o vetor de interconexões gerado com os nós descritos na *netlist*. Note que a nomenclatura utilizada é a mesma da *netlist*, no qual PI e PO representam os entradas primeiras e saídas primárias, respectivamente.

FIGURA 3.4 – C17 APÓS OS NÓS TEREM SIDO COMPUTADOS.



FONTE: DO AUTOR.

É então, realizado um novo escaneamento na *netlist* convertendo-a para o formato LCS (III). Na Figura 3.5 é mostrado o c17 em formato LCS

FIGURA 3.5 – C17 NO FORMATO LCS.

```
lcs::InOutBus<1> nodes[24]; //All circuit Nodes
lcs::InOutBus<1> inptnode[6];   //Input circuit Nodes
lcs::InOutBus<1> outnode[3];    //Output circuit Nodes

lcs::Buffer<1> _1gat(nodes[1], inptnode[1]);    //Primary Input
lcs::Buffer<1> _2gat(nodes[2], inptnode[2]);    //Primary Input
lcs::Buffer<1> _3gat(nodes[3], inptnode[3]);    //Primary Input
lcs::Buffer<1> _8fan(nodes[8], nodes[3]);   //Fanout From
lcs::Buffer<1> _9fan(nodes[9], nodes[3]);   //Fanout From
lcs::Buffer<1> _6gat(nodes[6], inptnode[4]);    //Primary Input
lcs::Buffer<1> _7gat(nodes[7], inptnode[5]);    //Primary Input
lcs::Nand<2> _10gat(nodes[10], (nodes[1], nodes[8]));   //Nand Gate
lcs::Nand<2> _11gat(nodes[11], (nodes[9], nodes[6]));   //Nand Gate
lcs::Buffer<1> _14fan(nodes[14], nodes[11]);    //Fanout From
lcs::Buffer<1> _15fan(nodes[15], nodes[11]);    //Fanout From
lcs::Nand<2> _16gat(nodes[16], (nodes[2], nodes[14]));  //Nand Gate
lcs::Buffer<1> _20fan(nodes[20], nodes[16]);    //Fanout From
lcs::Buffer<1> _21fan(nodes[21], nodes[16]);    //Fanout From
lcs::Nand<2> _19gat(nodes[19], (nodes[15], nodes[7]));  //Nand Gate
lcs::Nand<2> _22gat(nodes[22], (nodes[10], nodes[20])); //Nand Gate
lcs::Buffer<1> _22gat_(outnode[1], nodes[22]);  //Primary Output
lcs::Nand<2> _23gat(nodes[23], (nodes[21], nodes[19])); //Nand Gate
lcs::Buffer<1> _23gat_(outnode[2], nodes[23]);  //Primary Output
```

FONTE: DO AUTOR.

As interconexões são definidas na classe *InOutBus* em forma de vetores. Uma lista com todas as classes disponíveis na biblioteca pode ser encontrada em: http://liblcs.sourceforge.net/classes.html. Note que é definido um vetor *nodes* para todos os nós do circuito, um vetor *inptnode* para as entradas primárias e um vetor *outnode* para as saídas primárias. Após definidos os vetores que representam as interconexões, o circuito c17 é descrito utilizando apenas *buffers* e portas E-NEGADA (*nand)*. No C++, os vetores são inicializados a partir do endereço zero, então é adicionado uma posição de endereço a mais em cada vetor de interconexão para utilizar a mesma nomenclatura existente na *netlist,* por esta razão, o vetor *nodes*, por exemplo, possui 24 posições.

A sintaxe de construção lógica em LCS é feita especificando-se a lógica desejada e suas respectivas saídas e entradas. É possível também, definir um atraso de propagação se desejado e atribuir uma nomenclatura ao bloco lógico. Assim, a sintaxe lógica para construir o c17 é:

Buffer<atraso, busses de entrada> nomenclatura(bus de saída, busses de entrada)
Nand<atraso, busses de entrada> nomenclatura(bus de saída, busses de entrada)

Na conversão mostrada na Figura 3.5 o modelo de atraso zero foi adotado, então foram especificados somente a quantidade de *busses* de entrada, no qual, é definido de acordo com a quantidade de interconexões na entrada de cada boco lógico (1 para *buffers* e 2 para as portas *nand*).

Por fim, o arquivo LCS é executado (IV), inserindo na entrada do circuito todos os padrões de teste existente na sequência especificada no início da simulação e o WSA é computado (V).

Ao término da simulação é gerado um resumo com o valor de WSA total do circuito, e a quantidade de transições em cada nó, permitindo assim analisar qual nó dissipa mais potência por ter maior quantidade de chaveamento. Em adição é criado um arquivo (.vcd) com as formas de onda geradas no decorrer da simulação.

Suponha que o c17 mostrado na Figura 3.5, seja executado com uma sequência de 6 padrões $S_T = (11000, 00111, 11011, 01110, 10101, 1000)$ inseridos para estimular o circuito. O resumo obtido com os valores de WSA é mostrado na Figura 3.6.

FIGURA 3.6 – RESUMO DA FERRAMENTA WSA-T.

```
Total WSA of c17_Internal nodes: 32

Node WSA Fanout
10 5 1
11 5 2
16 4 2
19 3 1
22 4 1
23 2 1
```

FONTE: DO AUTOR

Então, de acordo com o resumo, o maior número de transição ocorreu nos nós 10 e 11, entretanto os nós 11 e 16 dissipam mais potência, pois, para o nó 11 o total de chaveamentos é 2 * 5 = 10 e para o nó 16 é 2 * 4 = 8, em que 2 é o número de *fan-out.* As formas de onda geradas no decorrer da simulação, são mostradas na Figura 3.7.

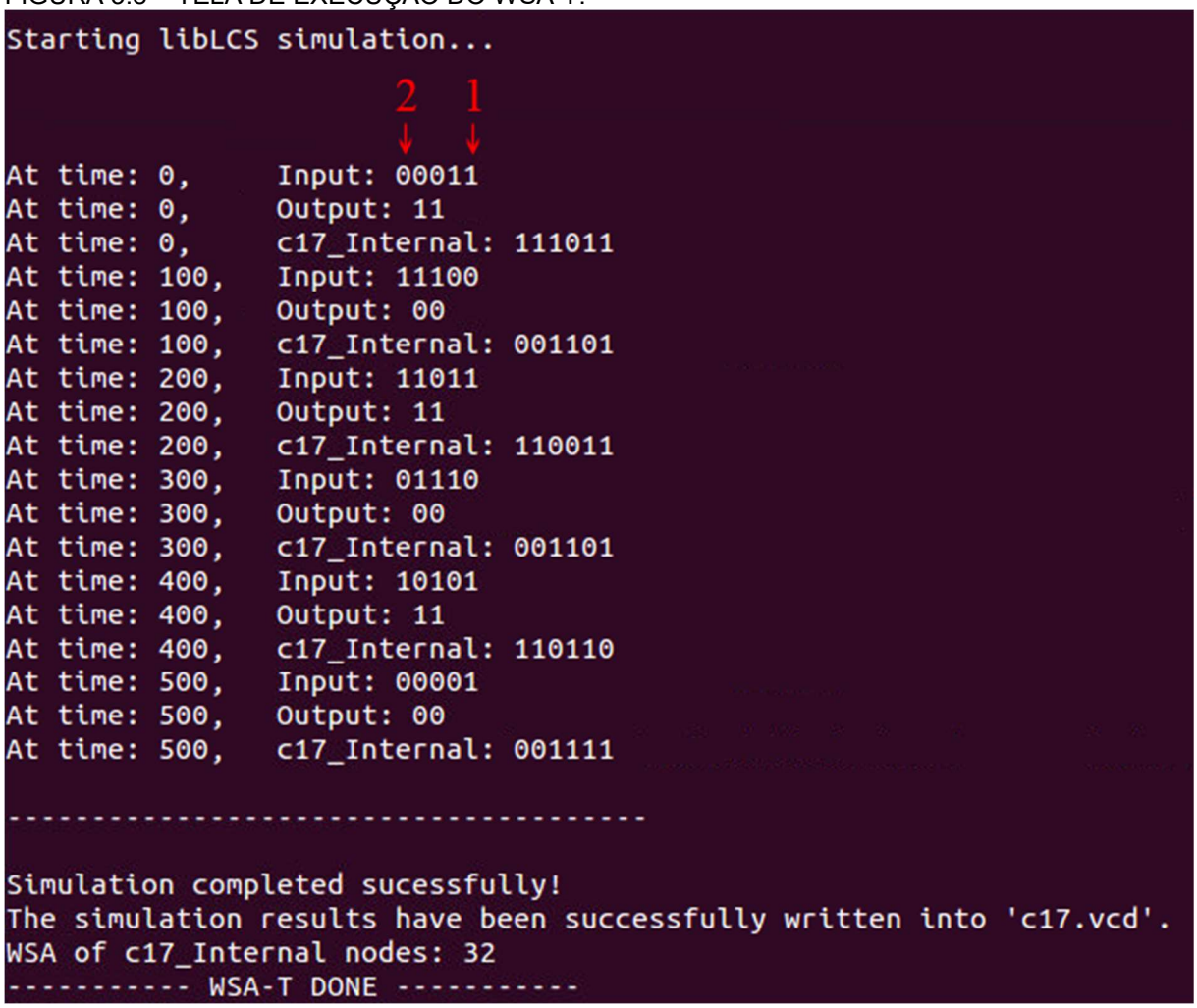
FIGURA 3.7 – ARQUIVO .VCD COM RESULTADOS EM FORMAS DE ONDA.

FONTE: DO AUTOR.

O vetor *c17.tps[5:0]* contém os valores dos nós de saída de cada porta *nand* do circuito, e estes valores são chamados *_*Internal*, na tela de exibição apresentada

no decorrer da simulação. Na Figura 3.8 é mostrado a tela exibida ao usuário durante a simulação com todas as informações referente a execução do simulador.

FIGURA 3.8 – TELA DE EXECUÇÃO DO WSA-T.



FONTE: DO AUTOR.

A verificação do simulador foi realizada através de simulações comparativas com o Atalanta ATPG e simulador de falhas FSIM. Em ambos os simuladores e no WSA-T, foram realizadas simulações para um mesmo circuito, estimulado com os mesmos vetores de teste, e o resultado obtido em ambos os foram iguais.

Na Figura 3.9 é mostrado o resumo da simulação realizada no FSIM utilizando o mesmo padrão $S_T = (11000, 00111, 11011, 01110, 10101, 1000)$. No resumo cada padrão de teste é mostrado seguido da saída obtida e da quantidade de falhas detectáveis pelo padrão.

Os números 1 e 2 destacados em vermelho nas Figuras 3.8 e 3.9 foram adicionadas para indicar que os padrões de teste são os mesmos em ambos os simuladores, porém a exibição na tela do WSA-T aparece de forma invertida, mas que isso não altera a resposta obtida.

FIGURA 3.9 – RESUMO DA SIMULAÇÃO GERADA NO FSIM.

```
* Log file for the circuit c17.bench.
* Number of faults detected by each test pattern:
                 1  2
test    1: 11000 11    6 faults detected
test    2: 00111 00    7 faults detected
test    3: 11011 11    2 faults detected
test    4: 01110 00    1 faults detected
test    5: 10101 11    3 faults detected
test    6: 10000 00    3 faults detected

End of fault simulation.
```

FONTE: FSIM.

4 OTIMIZAÇÃO BASEADA EM ALGORITMO GENÉTICO

4 ALGORITMO GENÉTICO E TESTE DE CI

Como visto na seção 1.1, o algoritmo genético foi desenvolvido visando adotar os “mecanismos” da evolução natural, em sistemas. O procedimento usualmente é iniciado de forma aleatória, gerando um conjunto de soluções a um determinado problema, em que, cada solução é comumente nomeada de **cromossomo** ou **indivíduo**, e **genoma** ou **população** o conjunto de soluções. Baseado nas operações de evolução natural, o algoritmo genético submete os indivíduos a **operação de seleção** e, os selecionados, a **operações de evolução: cruzamento e mutação** dando origem a novos indivíduos. Os indivíduos são selecionados com base na aptidão atribuída a cada um, em que, espera-se que os melhores indivíduos (com maior aptidão) sejam selecionados, e que as operações de evolução, resultem em novos indivíduos (prole) ainda melhores (com aptidão ainda melhor). O processo se repete até que uma condição de parada seja atendida (SOUZA, 2005).

No âmbito de testes de CIs, algoritmos genéticos são bastante implementados na etapa de verificação das especificações de projeto, otimizando a geração de sequências de padrões de testes (SHARMA; SABHARWAL; SIBAL, 2013), assim como no desenvolvido de TPGs (ENMIN; SHENGDONG; WENKANG, 2007; BANU; POORNIMA, 2012; ENMIN; SHENGDONG; YAN, 2011). O procedimento básico do algoritmo é descrito a seguir (MITCHELL, 1999):

1. *Início*: uma população é gerada (normalmente de forma aleatória) com *K* indivíduos. Geralmente, os indivíduos são representados por *bit strings.*
2. *Avaliação e cálculo da aptidão*: a aptidão *f(x)* é calculada para cada indivíduo *x* presente na população de acordo com uma função objetivo.
3. Enquanto uma condição de parada não for atendida, repita os passos a seguir até que *n* proles sejam criadas:
   a. *Seleção*: indivíduos são submetidos a operação de seleção, e selecionados com base na aptidão *f(x).* Os selecionados são intitulados como pais, e um mesmo indivíduo pode ser selecionado mais de uma vez.
   b. *Cruzamento*: cruze um par de indivíduos pais selecionados, a partir de uma probabilidade P_C (probabilidade de cruzamento), em uma ou mais posições

definidas aleatoriamente gerando indivíduos filhos. Em caso de exceder a probabilidade de cruzamento, gere filhos exatamente iguais aos pais.

c. *Mutação*: modifique os indivíduos filhos, a partir de uma probabilidade P_M (probabilidade de mutação), em uma ou mais posições definidas aleatoriamente.

4. *Atualização*: substitua a população atual com a nova população (prole).
5. Vá para o passo 2.

Cada iteração é denominada **geração**, no qual, a cada geração se tem uma nova população de cromossomos/indivíduos, sendo estes, constituídos por **genes**. Genes são as partes que constituem o indivíduo, representados por um *bit* ou um conjunto de *bits* (por exemplo, um *nibble*) se representados por *bit strings*. Através de uma **função objetivo** é determinada a aptidão de cada cromossomo, ou seja, o quão "bom" este indivíduo é para solucionar o problema**.** A função objetivo é determinada de acordo com o objetivo pretendido para o problema. A seleção é realizada com intuito de escolher os indivíduos de melhor aptidão, a partir de um método. Alguns métodos de seleção de indivíduo são: método de seleção por giro de roleta, método de seleção por torneio, entre outros. Operações de cruzamento e mutação são realizadas com intuito de criar indivíduos com aptidão ainda melhor que seus antecessores, a partir de técnicas, a saber: cruzamento de um ponto (1PX), cruzamento multipontos (MPX), cruzamento segmentado (SX), mutação *flip,* mutação por troca, entre outros (LUCAS, 2002). As subseções a seguir abordam esses métodos e operadores.

4.1.1 MÉTODOS DE SELEÇÃO

Seleção por giro de roleta: cada indivíduo da população recebe uma "fatia" da roleta de acordo com a aptidão obtida. A roleta é girada *K* vezes, em que *K* é o número de indivíduos na população. A cada giro um indivíduo é selecionado de acordo com os seguintes passos:

1. Obtenha a soma *S* da aptidão de todos os indivíduos da roleta.
2. Repita *K* vezes:
   2.1. Gere um valor aleatório *R* entre 0 e *S.*

2.2. Faça iterações na população somando as aptidões até que esta soma seja maior ou igual a *R,* e selecione o indivíduo correspondente a aptidão que resulte nesta soma.

Seleção por torneio: os indivíduos são ordenados de acordo com a aptidão, então dois indivíduos são escolhidos aleatoriamente. Um valor aleatório *R* entre 0 e 1 é obtido. Se *R* < *X*, em que *X* é um parâmetro, selecione o indivíduo de maior aptidão, do contrário, selecione o de menor aptidão.

4.1.2 OPERADORES GENÉTICOS

Cruzamento de um ponto (1PX): uma posição de corte entre 0 e l, em que l é o comprimento do indivíduo, é escolhida de forma aleatória, logo 0 < *R* < *l*, no qual o filho é criado com os genes das posições 0 a *R* de um dos pais, e os genes das posições *R*+1 a l do outro pai.

Cruzamento multipontos (MPX): um número *n* de posições de corte são geradas aleatoriamente, em que *n* < l, e os filhos são gerados permutando os genes de cada posição *n* dos pais.

Cruzamento segmentado (SX): funciona de maneira semelhante ao de multipontos, porém a cada execução as posições de corte são geradas novamente.

*Mutação flip***:** uma posição *R* e um valor *Z* do alfabeto aplicado (neste caso o alfabeto em análise é o binário) são selecionados aleatoriamente, e o gene existente na posição *R* é trocado por *Z*.

Mutação por troca: são selecionados *n* pares de genes, e os valores dos genes são permutados entre os pares.

4.1.3 PAREAMENTO

A operação de cruzamento é dada misturando os genes dos indivíduos pais para formar os filhos, e a escolha dos pares para reproduzir pode ser realizada

segundo (MITCHELL, 1999; LUCAS, 2002), por diversos métodos. Alguns dos métodos são descritos a seguir:

Escolha aleatória: os indivíduos selecionados são agrupados em pares de forma aleatória e cruzamento realizado entre os pares.

Criação em linha: um indivíduo de alta aptidão é cruzado com uma subpopulação de indivíduos.

Autofecundação: o indivíduo é cruzado consigo mesmo.

Endogamia: indivíduos muito parecidos (“parentes”) são cruzados.

Note que AGs podem ser aplicados a uma grande diversidade de problemas, logo, não é possível citar qual o melhor método ou tipo de operação a ser realizada, portanto, a solução adotada deve ser a que proporcionar melhor resultado ao problema em questão.

4.2 OTIMIZAÇÃO DE SEQUÊNCIAS DE PADRÕES DE TESTE BASEADA EM ALGORITMO GENÉTICO

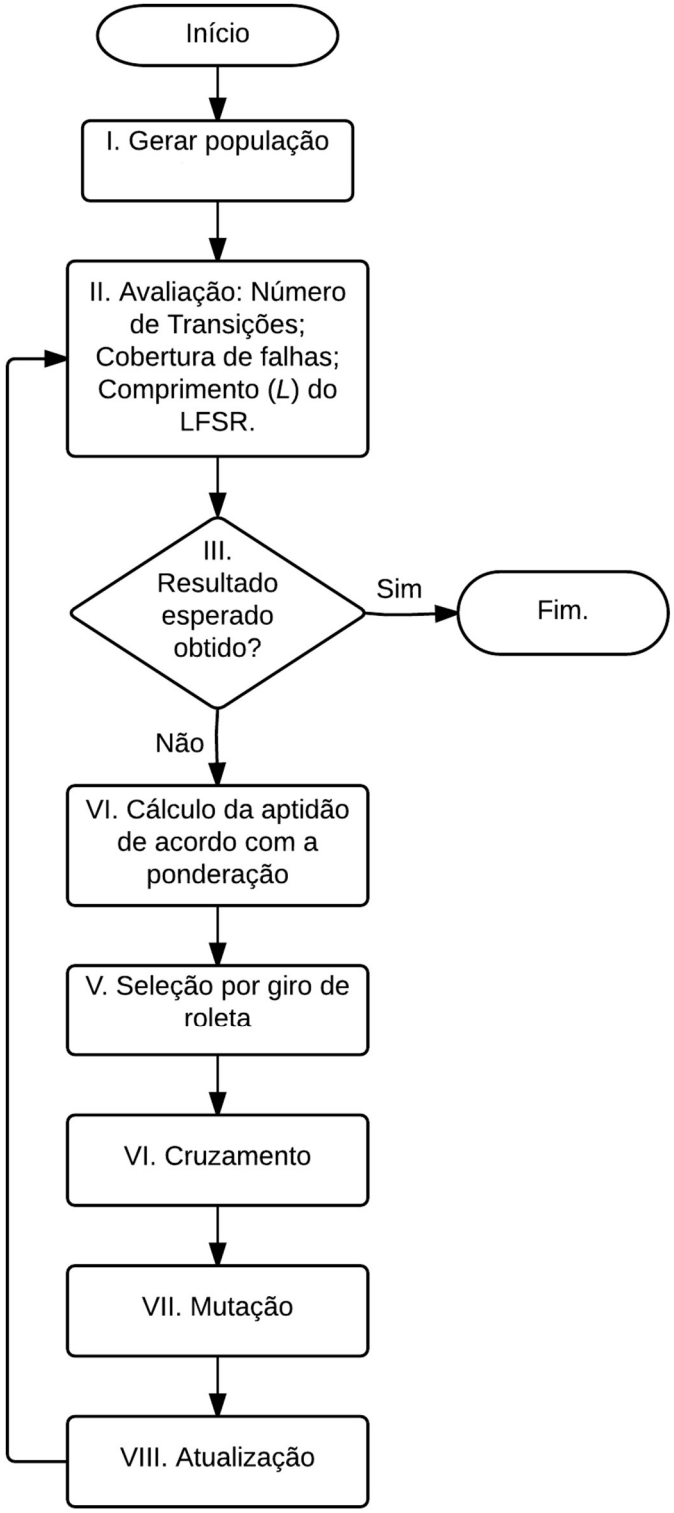
Neste trabalho o algoritmo genético foi aplicado para otimizar as sequências de padrões de teste. A otimização pretendida é a redução do WSA resultante destas, através da redução das transições (0→1 e vice-versa) entre dois padrões de teste consecutivos, sem reduzir de forma abrupta a cobertura de falhas, e mantendo o tempo de testes da sequência inicial.

As sequências de padrões otimizadas podem ser aplicadas em teste de verificação lógica computacional, como também implementadas em ATEs, ou em arquiteturas de testes que utilizem TPGs determinísticos para sequências pré-definidas.

Em adição, como anteriormente mencionado é proposto a aplicação do BMA para sintetizar o comprimento L de um LFSR capaz de gerar essas sequências obtidas com o algoritmo genético, otimizando a pequena ocupação de sobreárea proporcionando a implementação direta em técnicas de testabilidade.

O processo do algoritmo genético implementado neste trabalho é descrito no fluxograma mostrado na Figura 4.1.

FIGURA 4.1 – FLUXOGRAMA DO ALGORITMO GENÉTICO.



FONTE: DO AUTOR.

A seguir, são descritos cada etapa do fluxograma considerando o circuito ISCAS85 c17 mostrado na Figura 3.1.

I. Gerar população

O algoritmo genético é iniciado com a criação de uma população aleatoriamente, em que cada indivíduo é representado por uma sequência de padrões de teste. Os parâmetros de entrada para inicializar o processo são: o comprimento *N* de cada padrão de teste (*N* é a quantidade de entradas do CUT), a quantidade máxima l de padrões que se deseja em cada sequência e quantas sequências *K* devem ser geradas. Os indivíduos da população podem ser sequências de padrões de teste pseudoaleatórias, determinísticos, ou de qualquer outro tipo.

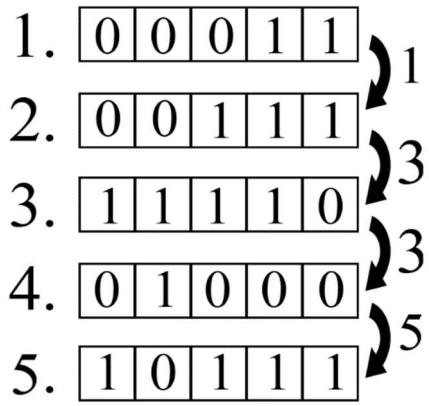
A título de exemplo suponha que a população inicial são sequências geradas por LFSRs, e que cada sequência *K* pode ter de 1 a l padrões de teste, cada um com comprimento *N = 5* (número de entradas do c17).

II. Avaliação

Após criados, cada indivíduo é avaliado. Os parâmetros avaliados em cada indivíduo são: a quantidade de transições, chame-a de TC, a cobertura de falhas, chame-a de FC, e o comprimento (grau do polinômio) do LFSR, chame-o L, capaz de gerar os padrões do indivíduo.

Suponha uma população $S = (S_1, S_2, \dots, S_T)$, em que $1 \leq T \leq K$, gerada para o c17 (*N* = 5), e que um dos indivíduos S_T é composto por 5 padrões de teste (l = *5*). Na Figura 4.2 é mostrado o indivíduo S_T definido.

FIGURA 4.2 – INDIVIDUO S_T GERADO ALEATORIAMENTE.

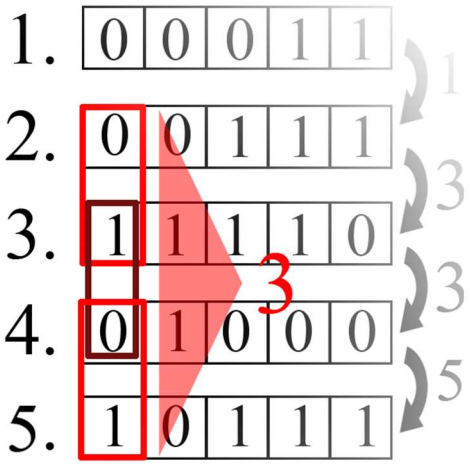


FONTE: DO AUTOR.

Como descrito nas seções anteriores, para as aplicações deste trabalho, um padrão de teste é uma sequência de *bits.* Logo, o número de transições é contabilizado somando a quantidade de mudanças que ocorre em cada posição da sequência, como pode ser visto Figura 4.3. Note que há um total de 3 transições na primeira posição.

Assim, o número total de transições existente neste indivíduo é TC = 12, e o quinto vetor é responsável pelo maior número/pico de transições desta sequência, com TC = 5.

FIGURA 4.3 – NÚMERO DE TRANSIÇÕES NA POSIÇÃO 1 DOS PADRÕES EXISTENTES NO INDIVÍDUO.



FONTE: DO AUTOR.

O comprimento L do LFSR, ou o grau do polinômio capaz de gerar o indivíduo é obtido através do BMA descrito na subseção 2.3 do capítulo 2 deste trabalho.

A cobertura de falhas é obtida através da ferramenta de simulação de falhas FSIM e HOPE. O FSIM foi utilizado para simular falhas em circuitos combinacionais e o HOPE para simular falhas em circuitos sequenciais. Na Figura 4.4 é mostrado o resumo gerado no FSIM com a cobertura de falhas atingida pelo indivíduo da Figura 4.2. Note que o indivíduo, atinge uma cobertura de 86,36%.

FIGURA 4.4 – COBERTURA DE FALHAS ATINGIDA PELO INDIVÍDUO OBTIDA NO FSIM.

```
* Log file for the circuit c17.bench.
* Number of faults detected by each test pattern:

test    1: 00011 01    8 faults detected
test    2: 00111 00    5 faults detected
test    3: 11110 10    3 faults detected
test    4: 01000 11    3 faults detected
test    5: 10111 10    0 faults detected

End of fault simulation.

        ******************************************************
        *                                                    *
        *            Welcome to fsim (version 1.2)           *
        *                                                    *
        *               Dong S. Ha (ha@vt.edu)               *
        *            Web: http://www.ee.vt.edu/ha            *
        *  Virginia Polytechnic Institute & State University *
        *                                                    *
        ******************************************************

******       SUMMARY OF FAULT SIMULATION RESULTS      ******
1. Circuit structure
   Name of the circuit                    : c17
   Number of primary inputs               : 5
   Number of primary outputs              : 2
   Number of gates                        : 6
   Level of the circuit                   : 3

2. Simulation parameters
   Simulation mode                        : file (c17.test)

3. Simulation results
   Number of test patterns applied        : 5
   Fault coverage                         : 86.364 %
   Number of collapsed faults             : 22
   Number of detected faults              : 19
   Number of undetected faults            : 3
```

FONTE: FSIM.

III. Condição de parada

Uma vez que os parâmetros obtidos atendam os objetivos buscados ou a condição de parada seja satisfeita, a simulação é encerrada, do contrário, um valor de aptidão é atribuído a cada indivíduo de acordo com os parâmetros avaliados (número de transições, cobertura de falhas, comprimento do LFSR).

IV. Cálculo da aptidão

A aptidão é atribuída a cada indivíduo de acordo com a função objetivo, normalizada entre 0 e 1, mostrada na Equação 4.1 de acordo com o número de transições (TC), a cobertura de falhas (FC) e o comprimento do LFSR (L).

$$Aptidão_T = P_{TC} * \left(\frac{TC_{TOTAL} - TC_T}{TC_{TOTAL}}\right) + P_{FC} * \left(\frac{FC_T}{100}\right) + P_L * \left(\frac{L_{TOTAL} - L_T}{L_{TOTAL}}\right) \quad (4.1)$$

em que, P_{TC}, P_{FC} e P_L são as ponderações desejadas para cada parâmetro, TC_{TOTAL} e L_{TOTAL} a soma do *TC* e *L*, respectivamente, de todos os indivíduos existentes na população, e TC_T, FC_T e L_T são os parâmetros avaliados em cada indivíduo *T* no passo II.

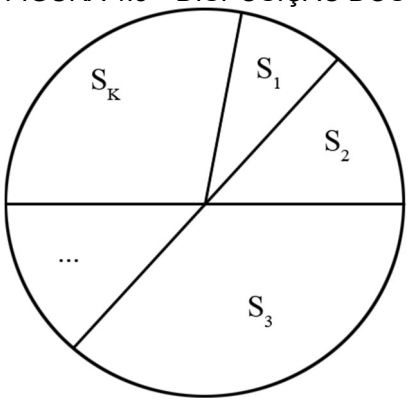
É importante ressaltar que a forma em que são atribuídos os valores de TC e *L* na função objetivo, é para que os indivíduos com o menor número de TC e L tenham uma maior aptidão.

V. Seleção por giro de roleta

Após atribuída a aptidão, cada indivíduo passa por um processo de seleção. O processo adotado neste trabalho é o método de seleção por giro de roleta em razão desse proporcionar uma boa diversidade da população (MITCHELL, 1999), reduzindo as chances de o AG convergir para o ponto ótimo local.

Os indivíduos da população recebem uma fatia da roleta de acordo com a aptidão atribuída pela função objetivo, como mostrado na Figura 4.5. Note que S_3 possui a maior fatia, então o mesmo é o indivíduo de maior aptidão. Os indivíduos são escolhidos de acordo com os procedimentos descritos na subseção 4.1.1 deste Capítulo. Para facilitar a implementação da roleta e adaptar ao procedimento da subseção 4.1.1, os indivíduos na roleta são organizados de forma crescente de acordo com os valores de aptidão.

FIGURA 4.5 – DISPOSIÇÃO DOS INDIVÍDUOS NO MÉTODO DE SELEÇÃO POR GIRO DE ROLETA.



FONTE: DO AUTOR.

Para cada indivíduo existente na população é feito um giro na roleta, então são realizados *K* giros. Logo, um mesmo indivíduo pode ser selecionado mais de uma vez para reproduzir. Espera-se que os melhores indivíduos sejam selecionados, devido a maior probabilidade (maior fatia da roleta) de serem escolhidos, porém indivíduos de aptidões inferiores também possuem chances de serem selecionados, o que de fato acontece, mantendo a diversidade da nova geração.

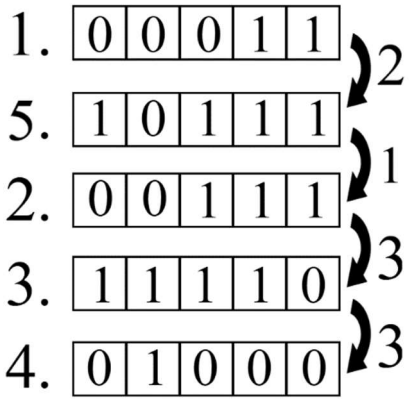
VI. Cruzamento

Indivíduos após selecionados como pais, são submetidos a reprodução, ou seja, a operação de cruzamento e mutação. Busca-se com a operação de cruzamento gerar indivíduos filhos a partir dos genes dos indivíduos pais. Neste trabalho foram desenvolvidos dois tipos de operações de cruzamento intituladas **cruzamento por clonagem** e **cruzamento por rompimento**. Em ambos, o cruzamento é realizado de forma similar ao cruzamento segmentado e o método de pareamento de autofertilização é aplicado.

Deu-se o nome cruzamento por clonagem a operação de cruzamento básica do algoritmo genético com intuito de facilitar a análise em relação ao outro tipo de cruzamento implementado. Tendo em vista isso, o cruzamento por clonagem opera da seguinte forma: para cada indivíduo selecionado é gerado um valor aleatório *R*. Se $R \leq P_C$, em que P_C é a probabilidade de cruzamento, os padrões de teste existentes no indivíduo que tenham o maior valor de *TC* são removidos e, para cada padrão removido, é gerado aleatoriamente uma posição R_P, em que $1 \leq P \leq N$ (número de entradas do CUT). E, os vetores removidos são reinseridos nas posições R_P geradas.

Suponha que o indivíduo selecionado seja composto pelos vetores mostrados na Figura 4.2, que $R < P_C$ e R_P = 2. O quinto vetor resulta em um maior TC, então o mesmo é removido e reinserido na posição 2. Logo, o indivíduo filho gerado, mostrado na Figura 4.6, teve uma redução do número total de transições, de TC =12 para TC = 9, e o maior número/pico de transições reduzido de TC = 5 para TC = 3.

FIGURA 4.6 - FILHO GERADO PELO MÉTODO DE CRUZAMENTO POR CLONAGEM.

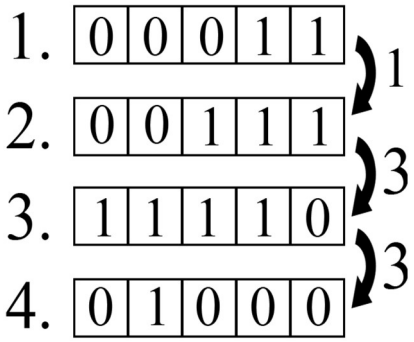


FONTE: DO AUTOR.

Em contrapartida, se $R > p_C$ o indivíduo filho será uma cópia do pai. Em suma, o objetivo do cruzamento por clonagem é reorganizar a ordem dos padrões no indivíduo buscando a ordem que tenha o menor número de transições. A vantagem dessa operação é a possibilidade de reduzir o WSA a partir da redução do TC sem alterar a cobertura de falhas atingida pelas as sequências iniciais.

O cruzamento por rompimento opera de forma semelhante ao anterior se $R \leq P_C$, ou seja, os padrões de maior valor de TC são removidos e depois reinseridos em posições aleatórias R_P, em que $1 \leq P \leq N$. Porém, se $R > P_C$, os vetores de maior transição são removidos e não são reinseridos. Logo, o indivíduo filho resultante do indivíduo com os padrões de teste mostrados na Figura 4.2 é mostrado na Figura 4.7, no qual foi obtido uma redução do número total de transições de TC= 12 para TC = 7, e o maior número/pico de transição existente foi reduzido de TC = 5 para TC = 3

FIGURA 4.7 – FILHO GERADO PELO MÉTODO DE CRUZAMENTO POR ROMPIMENTO.



FONTE: DO AUTOR.

É importante destacar que nessa operação de cruzamento, a cobertura de falhas pode ser alterada. A vantagem dessa operação em relação à anterior, é que o comprimento do LFSR sintetizado pelo BMA para gerar a sequência, geralmente, é curto, já que as sequências tendem a ser mais curtas em relação as sequências da população inicial.

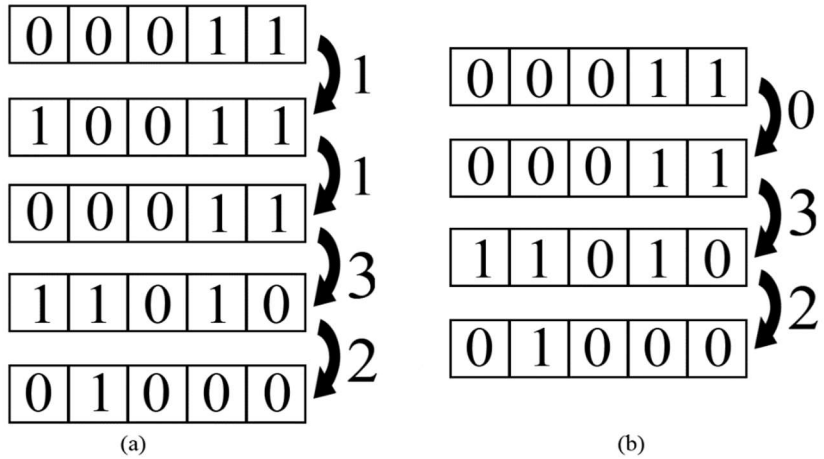
VII. Mutação

A mutação, é aplicada aos indivíduos filhos gerados na operação de cruzamento. A operação de mutação implementada neste trabalho é a mutação *flip*.

A operação de mutação implementada dar-se-á selecionando aleatoriamente uma posição *R*. Se $R \leq P_M$, em que P_M é a probabilidade de mutação, todos os padrões dos indivíduos filhos, em uma posição aleatória R_P, em que $1 \leq P \leq N$, tem o *bit* trocado por um valor *Y* também obtido aleatoriamente. Desse modo, é possível garantir que em um dos pinos do CUT não haverá transições.

Na Figura 4.8 (a) e (b) são mostrados os resultados da operação de mutação aplicada nos filhos gerados anteriormente no cruzamento por clonagem e por rompimento, respectivamente, considerando os seguintes parâmetros: $R < P_M$, $R_P = 3$ e $Y = 0$. Logo, para o c17, o pino de entrada 3 permanece em um valor fixo reduzindo o número total de transições de TC = 9 para TC = 7 em (a) e de TC = 7 para TC = 5 em (b).

FIGURA 4.8 – INDIVÍDUOS GERADOS APÓS MUTAÇÃO PARA (A) FILHO GERADO NO CRUZAMENTO POR CLONAGEM, E (B) FILHO GERADO NO CRUZAMENTO POR ROMPIMENTO.



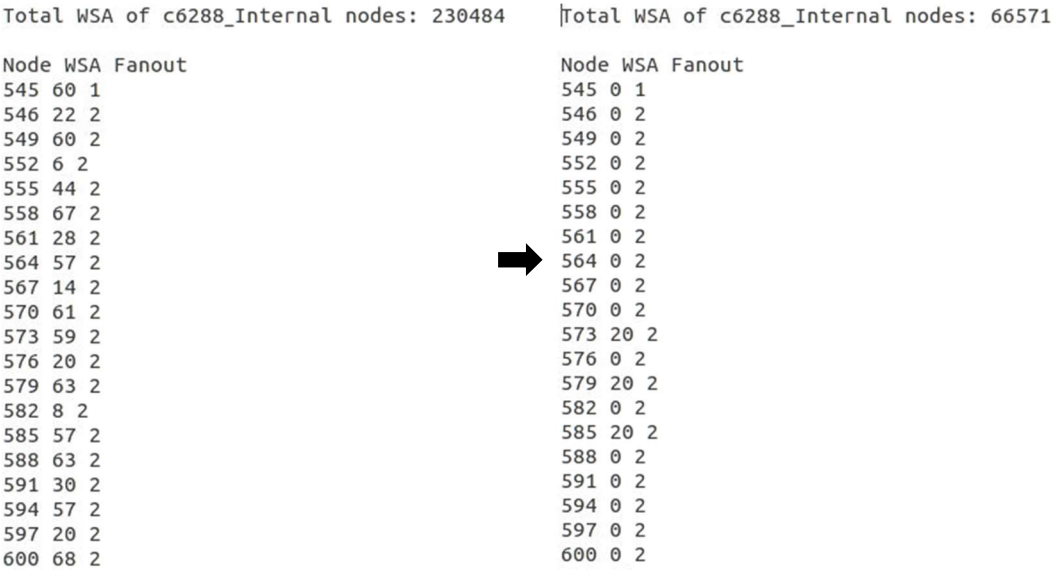
FONTE: DO AUTOR.

VIII. Atualização

Por fim, a prole gerada nas operações de reprodução (cruzamento e mutação), substituem a população atual completamente, e o processo é iniciado novamente a partir do ponto II até que a condição de parada seja satisfeita. É importante salientar que, de acordo com o tipo de cruzamento, e da probabilidade de cruzamento, como também de mutação, muitos indivíduos pais são copiados, ou seja, a população não é alterada por completo a cada geração.

Na Figura 4.9 é mostrado o resultado obtido para o ISCAS85 c6288, obtido através do WSA-T. A sequência inicial foi aplicada no algoritmo genético com o processo de cruzamento por clonagem. Note que foi obtido uma redução considerável do WSA, de 230.484 para 66.571.

FIGURA 4.9 – RESULTADOS OBTIDOS PARA O ISCAS85 C6288 ATRAVÉS DO WSA-T.



FONTE: DO AUTOR.

5 RESULTADOS E DISCUSSÕES

5 RESULTADOS E DISCUSSÕES

A validação da a otimização de sequências de padrões foi realizada através de simulações. O simulador WSA-T, o algoritmo genético e o algoritmo de Berlekamp-Massey foram implementados na linguagem de programação C++11, sendo desenvolvidos no ambiente de desenvolvimento gráfico *Code::Blocks® 13*, sob o sistema operacional *LINUX*.

As simulações foram realizadas em circuitos ISCAS85, por serem amplamente empregados na área de testes, como anteriormente citado. Na Tabela 5.1, são mostradas algumas informações dos circuitos em que os experimentos foram realizados.

TABELA 5.1 – DADOS SOBRE CIRCUITOS BENCHMARK.

Padrão	Circuito	Número de entradas	Número de saídas	Número de portas
ISCAS85	c432	36	7	160
	c499	41	32	170
	c1908	33	25	855
	c3540	50	22	1647
	c6288	32	32	2384

O procedimento seguido para realizar os testes simulacionais de verificação para otimização de sequências foi:

1. Definir os parâmetros de inicialização:
   a. O comprimento dos padrões (*N*).
   b. O número máximo de padrões que podem existir em um indivíduo (l).
   c. A quantidade de indivíduos a serem gerados (*K*).
2. Definir os parâmetros genéticos:
   a. Ponderações P_{TC}, P_{FC} e P_{L}, que representam as ponderações para número de transições, cobertura de falhas e comprimento do LFSR, respectivamente, da função objetivo.
   b. Tipo de cruzamento a ser utilizado.
   c. Número de gerações.

De acordo com Souza (2005), em simulações realizadas com algoritmo genético, o melhor indivíduo, que representa a solução ao problema, pode estar presente em qualquer geração e não somente na última.

Nas seções a seguir serão apresentados os resultados obtidos nas simulações. Estas foram realizadas aplicando diferentes probabilidades de cruzamento e mutação para uma mesma população e com diferentes ponderações na função objetivo para observar o comportamento do algoritmo genético. Para isto, a mesma *seed* foi utilizada para que a cada simulação realizada, a mesma população e valores aleatórios gerados, fossem replicados.

As probabilidades de cruzamento adotadas foram: 0,2; 0,3; 0,4; 0,5; 0,6; 0,7, 0;8; 0,9; e para mutação foram: 0,01 e 0,05. Para o cruzamento por clonagem, os baixos valores de probabilidade (até 0,6) foram utilizados com intuito de evitar que o algoritmo genético convergisse em um ponto ótimo local ou que somente indivíduos ruins fossem proliferados, e os valores mais altos para evitar percas bruscas na cobertura de falhas para o cruzamento por rompimento. É importante ressaltar que as populações inicias são de indivíduos pseudoaleatórios, gerados por LFSRs.

Duas condições de parada foram adotadas, em todas as simulações, a saber: 1. Após 500 gerações, ou 2. Indivíduos com apenas um gene. A condição 2 é aplicada somente para o cruzamento por rompimento, devido a sua característica de remoção de padrões de teste que resultem em um número de transições alto. No qual, a cada geração esses padrões podem ser removidos em um ou mais indivíduos. Assim, é possível que antes de atingir o limite máximo de gerações definido como condição de parada, a maior parte dos indivíduos esteja com um único padrão de teste.

É importante lembrar que para atingir os objetivos para com este trabalho, busca-se reduzir o número de transições e o comprimento do polinômio obtido no BMA sem alterar a cobertura de falhas.

Em relação aos resultados apresentados a seguir, P_{TC} representa o valor de ponderação para o número de transições, P_{FC} representa o valor de ponderação para a cobertura de falhas e P_L representa o valor de ponderação para o comprimento L do LFSR obtido pelo algoritmo de Berkelamp-Massey, e que, os valores dos números de transição (TC) representam quantas vezes um nó chaveou, e o comprimento (L) obtido no BMA representa quantos registradores de deslocamento possui o LFSR.

5.1 OTIMIZAÇÃO DO CONSUMO ENERGÉTICO

Como visto nas seções anteriores, reduzir as transições entre padrões de teste consecutivos reduz o chaveamento o qual o CUT é submetido, por sua vez, o consumo energético. Para esta otimização a função objetivo é ponderada somente para o número de transições, ou seja, $P_{TC} = 1, P_{FC} = 0, P_L = 0$. O intuito para com esta otimização é assegurar um alto valor de redução do número de transições. Na Tabela 5.2 são mostrados os resultados simulacionais obtidos para o ISCAS85 c6288.

TABELA 5.2 – OTMIZAÇÃO DO CONSUMO ENERGÉTICO PARA O C6288.

c6288	TC (#chaveamentos)		BMA L (#registradores)		FC (%)	
	Máximo	Mínimo	Máximo	Mínimo	Máximo	Mínimo
Valores iniciais do indivíduo	1754	107	1024	160	99,56	81,66
Valores obtidos com cruzamento por clonagem após a condição de parada ser atendida	57	41	161	159	66,66	62,61
Valores finais obtidos com cruzamento por rompimento após a condição de parada ser atendida	11	0	34	16	52,16	35,15
Melhor valor obtido dentre todas as gerações no cruzamento por clonagem	125		162		79,89	
Melhor valor obtido dentre todas as gerações no cruzamento por rompimento	0		16		35,65	

Com intuito de obter a máxima otimização obtida, os valores máximos foram utilizados nas análises. Os resultados para o cruzamento por clonagem são mostrados nas Equações 5.1, 5.2 e 5.3 para o número de transições (TC), comprimento do LFSR (L) e a cobertura de falhas (FC), respectivamente.

$$TC_{RESULTANTE} = \left(\frac{TC_{INICIAL} - TC_{FINAL}}{TC_{INICIAL}}\right) * 100 = \left(\frac{1754 - 125}{1754}\right) * 100 = 92{,}87\% \qquad (5.1)$$

$$L_{RESULTANTE} = \left(\frac{L_{INICIAL} - L_{FINAL}}{L_{INICIAL}}\right) * 100 = \left(\frac{1024 - 162}{1024}\right) * 100 = 84{,}17\% \quad (5.2)$$

$$FC_{RESULTANTE} = \left(\frac{FC_{INICIAL} - FC_{FINAL}}{FC_{INICIAL}}\right) * 100 = \left(\frac{99{,}56 - 79{,}89}{99{,}56}\right) * 100 = 19{,}76\% \quad (5.3)$$

Logo, com este tipo de cruzamento e uma função objetivo ponderada somente para a redução do número de transições, é possível reduzir consideravelmente o número de transições, para estas simulações em até 92,87%. Algo importante a ressaltar é que mesmo sendo desconsiderado na função objetivo, fica claro que alterar a ordem da sequência de entrada influência diretamente no comprimento do LFSR obtido pelo BMA, obtendo nestas simulações uma redução de 84,17%. Contudo, houve uma redução da cobertura de falhas de 19,76%, o que é indesejado. Isto ocorre devido a cobertura de falhas ser desconsiderada na função objetivo, logo, é possível que os indivíduos com a melhor cobertura de falhas sejam aqueles com um alto valor de transições, portanto, não são selecionados.

Os resultados para o cruzamento por rompimento são mostrados nas Equações 5.4, 5.5 e 5.6 para o número de transições (TC), comprimento do LFSR (L) e a cobertura de falhas (FC), respectivamente

$$TC_{RESULTANTE} = \left(\frac{TC_{INICIAL} - TC_{FINAL}}{TC_{INICIAL}}\right) * 100 = \left(\frac{1754 - 0}{1754}\right) * 100 = 100\% \quad (5.4)$$

$$L_{RESULTANTE} = \left(\frac{L_{INICIAL} - L_{FINAL}}{L_{INICIAL}}\right) * 100 = \left(\frac{1024 - 16}{1024}\right) * 100 = 98{,}43\% \quad (5.5)$$

$$FC_{RESULTANTE} = \left(\frac{FC_{INICIAL} - FC_{FINAL}}{FC_{INICIAL}}\right) * 100 = \left(\frac{99{,}56 - 35{,}65}{99{,}56}\right) * 100 = 64{,}19\% \quad (5.6)$$

Como o processo de cruzamento por rompimento consiste em remover padrões com alto número de transições, é possível que os indivíduos após gerações possuam somente um padrão restante. Dessa forma, como não há mais de um padrão existente na sequência, o número de transições é zero, e o WSA consequentemente será baixo. Portanto, foi possível reduzi-lo em 100% em relação ao valor inicial. Em relação ao comprimento L, houve uma redução de 98,43%, mesmo sendo desconsiderado na função objetivo, mostrando que o tamanho da sequência de entrada influência diretamente no comprimento do LFSR gerado pelo BMA.

Entretanto, a perda na cobertura de falhas é significante, de 64,19%. Isto ocorre devido a redução do número de padrões, principalmente se os restantes forem padrões de baixa detectabilidade.

Para o cruzamento com rompimento, foi possível obter um LFSR muito curto (polinômio de grau 16 para um circuito de 32 entradas), isto se dá, devido à redução no número de padrões do indivíduo, tornando mais curta a sequência a ser gerada pelo LFSR sintetizado. Enquanto no cruzamento por clonagem, como a quantidade de padrões é mantido, é possível atingir uma cobertura de falhas muito maior que no cruzamento por rompimento.

Logo, com essa ponderação é possível obter bons resultados para uma menor dissipação de potência, porém, o resultado não é bom em relação aos demais parâmetros.

5.2 OTIMIZAÇÃO DA COBERTURA DE FALHAS

Com o intuito de manter a cobertura de falhas, simulações foram realizadas com a função objetivo ponderada para a mesma. Logo, $P_{TC} = 0, P_{FC} = 1, P_L = 0$. Na Tabela 5.3 são mostrados os valores obtidos c3540.

Com intuito de verificar a máxima otimização obtida, os valores máximos foram utilizados nas análises. Os resultados para o cruzamento por clonagem são mostrados nas Equações 5.7, 5.8 e 5.9 para o número de transições (TC), comprimento do LFSR (L) e a cobertura de falhas (FC), respectivamente.

$$TC_{RESULTANTE} = \left(\frac{TC_{INICIAL} - TC_{FINAL}}{TC_{INICIAL}}\right) * 100 = \left(\frac{2709 - 2192}{2709}\right) * 100 = 19{,}08\% \quad (5.7)$$

$$L_{RESULTANTE} = \left(\frac{L_{INICIAL} - L_{FINAL}}{L_{INICIAL}}\right) * 100 = \left(\frac{2500 - 2424}{2500}\right) * 100 = 3{,}04\% \quad (5.8)$$

$$FC_{RESULTANTE} = \left(\frac{FC_{INICIAL} - FC_{FINAL}}{FC_{INICIAL}}\right) * 100 = \left(\frac{81{,}88 - 81{,}94}{81{,}88}\right) * 100 = -0{,}07\% \quad (5.9)$$

TABELA 5.3 – OTIMIZAÇÃO DO FC PARA C3540.

c3540	TC (#chaveamentos)		BMA L (#registradores)		FC (%)	
	Máximo	Mínimo	Máximo	Mínimo	Máximo	Mínimo
Valores iniciais do indivíduo	2709	147	2500	175	81,88	41,48
Valores obtidos com cruzamento por clonagem após a condição de parada ser atendida	2286	2161	2503	2497	80,83	79,90
Valores finais obtidos com cruzamento por rompimento após a condição de parada ser atendida	21	0	49	25	23,72	12,31
Melhor valor obtido dentre todas as gerações no cruzamento por clonagem	2192		2424		81,94	
Melhor valor obtido dentre todas as gerações no cruzamento por rompimento	1962		2250		83,43	

Logo, é possível manter a cobertura de falhas, ou seja, 0% de perdas em relação aos valores iniciais. O valor de *-0,07%* significa que não houve perca na cobertura, mas que houveram ganhos, ou seja, que o valor final de cobertura de falhas obtido é maior que o inicial, porém, este valor é considerado como margem de erro do simulador, pois a sequência pode ser reordenada, e não pode ser alterada, então a cobertura de falhas não muda. Em contrapartida, houve uma baixa redução do número de transições, com menos de 20% de redução. Isto mostra que quanto mais padrões

maior a cobertura de falhas, principalmente quando os padrões são de natureza pseudoaleatória. A redução obtida no comprimento do LFSR também foi muito baixa, não atingindo 4%.

Os resultados para o cruzamento por rompimento são mostrados nas Equações 5.10, 5.11 e 5.12 para o número de transições (TC), comprimento do LFSR (L) e a cobertura de falhas (FC), respectivamente

$$TC_{RESULTANTE} = \frac{TC_{INICIAL} - TC_{FINAL}}{TC_{INICIAL}} * 100 = \left(\frac{2709 - 1962}{2709}\right) * 100 = 27{,}57\% \quad (5.10)$$

$$L_{RESULTANTE} = \left(\frac{L_{INICIAL} - L_{FINAL}}{L_{INICIAL}}\right) * 100 = \left(\frac{2500 - 2250}{2500}\right) * 100 = 10\% \quad (5.11)$$

$$FC_{RESULTANTE} = \left(\frac{FC_{INICIAL} - FC_{FINAL}}{FC_{INICIAL}}\right) * 100 = \left(\frac{81{,}88 - 83{,}43}{81{,}88}\right) * 100 = -1{,}89\% \quad (5.12)$$

Nota-se que, com a otimização voltada somente para a cobertura de falhas é possível evitar perdas. No entanto, para manter essa alta cobertura de falhas, a redução do número de transições é baixa, nestas simulações chegando a somente 27,57%. Logo, a redução do comprimento L também é baixa, chegando a somente 10% em relação aos valores iniciais.

Conclui-se que, se a função objetivo for ponderada para cobertura de falhas é possível mantê-la, porém as melhorias nos demais parâmetros são muito pequenas.

5.3 OTIMIZAÇÃO DO COMPRIMENTO DO LFSR OBTIDO PELO BMA

Simulações foram realizadas buscando sintetizar o menor LFSR gerado pelo BMA com intuito de gerar as sequências otimizadas em técnicas de testabilidade. Para esta otimização a seguinte ponderação foi adotada: $P_{TC} = 0, P_{FC} = 0, P_L = 1$. O intuito para com esta otimização é assegurar polinômios com grau que se aproximem ao número de entradas do CI. Na Tabela 5.4 são mostrados os valores obtidos c432.

TABELA 5.4 – OTIMIZAÇÃO DO BMA PARA C432.

c432	TC (#chaveamentos)		BMA L (#registradores)		FC (%)	
	Máximo	Mínimo	Máximo	Mínimo	Máximo	Mínimo
Valores iniciais do indivíduo	1869	56	1296	35	93,89	30,53
Valores obtidos com cruzamento por clonagem após a condição de parada ser atendida	64	44	91	89	38,55	36,64
Valores finais obtidos com cruzamento por rompimento após a condição de parada ser atendida	27	0	55	17	18,32	7,25
Melhor valor obtido dentre todas as gerações no cruzamento por clonagem	100		36		34,54	
Melhor valor obtido dentre todas as gerações no cruzamento por rompimento	0		17		10,69	

Com intuito de encontrar a máxima otimização obtida, os valores máximos foram utilizados nas análises. Os resultados para o cruzamento por clonagem são mostrados nas Equações 5.13, 5.14 e 5.15 para o número de transições (TC), comprimento do LFSR (L) e a cobertura de falhas (FC), respectivamente.

$$TC_{RESULTANTE} = \left(\frac{TC_{INICIAL} - TC_{FINAL}}{TC_{INICIAL}}\right) * 100 = \left(\frac{1869 - 100}{1869}\right) * 100 = 94{,}64\% \qquad (5.13)$$

$$L_{RESULTANTE} = \left(\frac{L_{INICIAL} - L_{FINAL}}{L_{INICIAL}}\right) * 100 = \left(\frac{1296 - 36}{1296}\right) * 100 = 97{,}22\% \quad (5.14)$$

$$FC_{RESULTANTE} = \left(\frac{FC_{INICIAL} - FC_{FINAL}}{FC_{INICIAL}}\right) * 100 = \left(\frac{93{,}89 - 34{,}54}{93{,}89}\right) * 100 = 63{,}21\% \quad (5.15)$$

Logo, é possível obter boa redução em relação ao comprimento do LFSR com os valores iniciais. Como pode ser visto, foi possível obter 97,22% de redução no L, no qual, este teve o número de registradores igual ao número de entradas do circuito. Em adição foi possível reduzir o número de transições em quase 95%, entretanto, a cobertura de falhas foi reduzida em 63,21%.

Os resultados para o cruzamento por rompimento são mostrados nas Equações 5.16, 5.17 e 5.18 para o número de transições (TC), comprimento do LFSR (L) e a cobertura de falhas (FC), respectivamente

$$TC_{RESULTANTE} = \left(\frac{TC_{INICIAL} - TC_{FINAL}}{TC_{INICIAL}}\right) * 100 = \left(\frac{1869 - 0}{1869}\right) * 100 = 100\% \quad (5.16)$$

$$L_{RESULTANTE} = \left(\frac{L_{INICIAL} - L_{FINAL}}{L_{INICIAL}}\right) * 100 = \left(\frac{1296 - 17}{1296}\right) * 100 = 98{,}69\% \quad (5.17)$$

$$FC_{RESULTANTE} = \left(\frac{FC_{INICIAL} - FC_{FINAL}}{FC_{INICIAL}}\right) * 100 = \left(\frac{93{,}89 - 10{,}69}{93{,}89}\right) * 100 = 88{,}61\% \quad (5.18)$$

Nota-se que, com a otimização voltada somente para o comprimento do LFSR (grau do polinômio) foi possível sintetizar um LFSR curto, no qual uma redução de 98,69% foi obtida. Porém, fica claro que para isto, é preciso que as sequências de padrões de teste tenham uma pequena quantidade de padrões, resultando também em uma redução considerável no número de transições (100%), porém na cobertura de falhas houve perda de até 88,61%.

Assim, é possível obter boas reduções no comprimento do LFSR juntamente do número de transições, contudo a perdas abruptas de 63% a 88% na cobertura de falhas.

O algoritmo genético se mostrou bastante eficaz ao trabalhar com cada parâmetro separadamente, porém, simulações foram realizadas com ponderações equivalentes entre os parâmetros para analisar o seu comportamento, como descrito a seguir.

5.4 OTIMIZAÇÃO PONDERADA

Para esta otimização, ponderações equivalentes foram adotadas com intuito de suprir os objetivos deste trabalho, a saber: $P_{TC} = 0{,}33, P_{FC} = 0{,}33, P_L = 0{,}33$. O intuito desta otimização é buscar um balanceamento (*trade-off*) entre todos os parâmetros, ou seja, alta redução do número de transições, maior cobertura de falhas e menor comprimento do LFSR. Na Tabela 5.5 são mostrados os valores obtidos c499.

TABELA 5.5 – OTIMIZAÇÃO PONDERADA PARA C499.

c499	TC (#chaveamentos)		BMA L (#registradores)		FC (%)	
	Máximo	Mínimo	Máximo	Mínimo	Máximo	Mínimo
Valores iniciais do indivíduo	2219	0	1640	21	94,06	23,75
Valores obtidos com cruzamento por clonagem após a condição de parada ser atendida	962	851	1786	1782	84,96	78,23
Valores finais obtidos com cruzamento por rompimento após a condição de parada ser atendida	24	0	42	21	36.41	20,58
Melhor valor obtido dentre todas as gerações no cruzamento por clonagem	1429		1784		94,33	
Melhor valor obtido dentre todas as gerações no cruzamento por rompimento	914		1148		91,69	

Nota-se que, com a otimização ponderada é possível obter um balanceamento entre a cobertura de falhas e o número de transições, porém o comprimento do LFSR (grau do polinômio) é muito alto. O valor de otimização com cruzamento por clonagem é dado de acordo com as Equações 5.19, 5.20 e 5.21, para o número de transições (TC), comprimento do LFSR (L) e a cobertura de falhas (FC), respectivamente. Os valores máximos foram utilizados para encontrar a maior otimização. Assim, tem-se:

$$TC_{RESULTANTE} = \left(\frac{TC_{INICIAL} - TC_{FINAL}}{TC_{INICIAL}}\right) * 100 = \left(\frac{2219 - 1429}{2219}\right) * 100 = 35{,}60\% \quad (5.19)$$

$$L_{RESULTANTE} = \left(\frac{L_{INICIAL} - L_{FINAL}}{L_{INICIAL}}\right) * 100 = \left(\frac{1640 - 1784}{1640}\right) * 100 = -8{,}78\% \quad (5.20)$$

$$FC_{RESULTANTE} = \left(\frac{FC_{INICIAL} - FC_{FINAL}}{FC_{INICIAL}}\right) * 100 = \left(\frac{94{,}06 - 94{,}33}{94{,}06}\right) * 100 = -0{,}29\% \quad (5.21)$$

Logo, houve uma redução de 35,60% do número de transições com 0% de perdas na cobertura de falhas (o “ganho” de 0,29%, representado pelo *-0,29%,* também é considerado margem de erro do simulador). Porém, o comprimento do LFSR aumentou em 8,78% em relação ao comprimento inicial que já era elevado.

O valor de otimização com cruzamento por rompimento é obtido de acordo com as Equações 5.22, 5.23 e 5.24, para o número de transições (TC), comprimento do LFSR (L) e a cobertura de falhas (FC), também considerando os melhores valores obtidos dentre as gerações. Os valores máximos foram novamente adotados para encontrar o máximo valor de diferença.

$$TC_{RESULTANTE} = \left(\frac{TC_{INICIAL} - TC_{FINAL}}{TC_{INICIAL}}\right) * 100 = \left(\frac{2219 - 914}{2219}\right) * 100 = 58{,}81\% \quad (5.22)$$

$$L_{RESULTANTE} = \left(\frac{L_{INICIAL} - L_{FINAL}}{L_{INICIAL}}\right) * 100 = \left(\frac{1640 - 1148}{1640}\right) * 100 = 30\% \quad (5.22)$$

$$FC_{RESULTANTE} = \left(\frac{FC_{INICIAL} - FC_{FINAL}}{FC_{INICIAL}}\right) * 100 = \left(\frac{94{,}06 - 91{,}69}{94{,}06}\right) * 100 = 2{,}52\% \quad (5.24)$$

Assim, é possível obter reduções no número de transições em até 58,81%, sem perdas abruptas na cobertura de falhas, chegando somente a 2,5%. Porém o comprimento LFSR teve reduções muito baixas em seu comprimento.

5.5 COMPARAÇÃO DE RESULTADOS

Anita e Vanathi, (2014) também buscam reduzir o WSA aumentando a correlação da sequência de padrões de teste assim como este trabalho, utilizando algoritmo genético. Diante disto, testes simulacionais foram realizados e comparados. Como anteriormente descrito na seção 2.1, propuseram a aplicação do algoritmo genético para obter uma alta cobertura de falhas com uma redução no consumo de potência através de diferentes métodos de redução do número de transições. Na tabela 5.6, são mostrados os resultados comparativos em relação a redução de transição e a cobertura de falhas obtidas com a proposta deste trabalho.

É importante ressaltar que os autores definiram quais falhas queriam cobrir, diante disto, a cobertura de falhas foi de 100%. Após atingirem a cobertura de falhas de 100% eles reduziam o número de transições para obter valores menores de WSA.

Como os autores não oferecem nenhuma técnica de testabilidade, as simulações foram realizadas desconsiderando o BMA da função objetivo com foco na redução do número de transições e manter a cobertura de falhas. Dessa forma, a ponderação adotada foi: $P_{TC} = 0{,}5, P_{FC} = 0{,}5, P_L = 0$.

TABELA 5.6 – COMPARAÇÃO DE RESULTADOS.

Circuito	**Anita e Vanathi (2014)**		**Trabalho proposto**	
	TC (#chaveamentos)	**FC (%)**	**TC (#chaveamentos)**	**FC (%)**
C1908	2895	100	731	80,36
C3540	3516	100	5677	90,11
C6288	12218	100	700	99,46
Valor Médio	6209,67	100	2369,33	89,98

A diferença entre os resultados é dada nas Equações 5.25 e 5.26 para o número de transições (TC) e cobertura de falhas (FC), respectivamente.

$$TC_{RESULTANTE} = \left(\frac{6209{,}67 - 2369{,}33}{6209{,}67}\right) * 100 = 61{,}84\% \quad (5.25)$$

$$FC_{RESULTANTE} = \left(\frac{100 - 89{,}98}{100}\right) * 100 = 10{,}02\% \quad (5.26)$$

Assim, foi possível obter uma redução considerável de 61,84% do número de transições em relação ao trabalho que está sendo comparado. No entanto, houve uma redução de 10% na cobertura de falhas também em relação ao mesmo.

5.6 DISCUSSÕES

Diante dos resultados obtidos com simulações realizadas para diferentes ponderações, e das simulações comparativas com o trabalho de outro autor, pode-se concluir que, para aumentar a cobertura de falhas é preciso aumentar também a quantidade de padrões de teste quando gerados de forma pseudoaleatória, resultando no aumento do tempo de teste. Contudo, a otimização proposta neste trabalho se mostrou satisfatória quando ponderada para o objetivo desejado, principalmente na redução do WSA através do aumento da correlação entre padrões presentes nas sequências de teste (redução do número de transições). Logo, de acordo com a Equação 2.13, quanto menor o número de chaveamentos (S_G) menor é o consumo de energia durante ao teste.

6 CONCLUSÕES

6 CONCLUSÕES

Neste trabalho foi proposto a aplicação do algoritmo genético como solução aos problemas citados, através da otimização de sequências de padrões de teste, de qualquer tipo. A aplicação foi feita com o objetivo principal de buscar a solução ótima para o processo de teste, a saber: reduzir o número de transições de uma sequência de padrões de teste e manter a cobertura de falhas da mesma ou não haver perdas abruptas. Foi buscado ainda nesse trabalho, sintetizar TPGs pseudoaleatórios com pequena sobreárea de *hardware.* Para isto, o algoritmo de Berlekamp-Massey foi utilizado, visando encontrar o LFSR mais curto para gerar as sequências otimizadas pelo algoritmo genético.

Simulações foram realizadas, ponderando a otimização para cada objetivo pretendido buscado no processo de testes. Análises com diferentes ponderações foram feitas, mostrando que o algoritmo é eficaz em conseguir reduzir o WSA através da redução do número de transições entre padrões de teste, ou seja, do aumento da correlação, e manter a cobertura de falhas da sequência, em que foi possível obter até 92,8% na redução do número de transições sem alterar a cobertura de falhas. No entanto, os resultados obtidos no BMA não foram favoráveis para implementação de um TPG pseudoaleatório somente a base de LFSR, porém, essas sequências otimizadas podem ser geradas por outros meios, como por exemplo o ATE e TPGs determinísticos. Logo, podem ser aplicadas em todos os níveis de testes no processo de manufatura de CIs.

É importante ressaltar que, por não aumentar o comprimento das sequências de teste, o simulador mantém o tempo de execução original da sequência, conseguindo em alguns casos, reduzir o mesmo ao fazer uso do cruzamento por rompimento, em que, ao reduzir o tamanho da sequência de padrões de teste se reduz também o tempo para realizar o mesmo. Assim, conclui-se que os objetivos definidos para este trabalho, foram satisfeitos.

REFERÊNCIAS

Abdallatif S. Abu-Issa, Steven F. **Bit-Swapping and Scan-Chain Ordering: A Novel Technique for Peak- and Average-Power Reduction in Scan-Based BIST.** *IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems,* vol. 28, n. 5, 2009.

ALOUL, F. A., SAGAHYROON, A. **Estimation of the Weighted Maximum Switching Activity in Combionational CMOS Circuits.** *IEEE International Symposium on Circuits and Systems 2006,* 4 pp., 2006.

ANITA, J. P., VANATHI, Dr. P. T. **Genetic Algorithm Based Test Pattern Generation for Multiple Stuck-at Faults and Test Power Reduction in VLSI Circuits.** *IEEE ICECS,* pp. 1-6, 2014.

BANU, F., POORNIRAMA, N. **BIST Using Genetic Algorithm for Error Detection and Correction.** *IEEE ICAESM,* pp. 588-592, 2012.

BUSHNELL, M. L., AGRAWAL, V. D. **Essentials of Electronic Testing.** Kluwer Academics Publishers: 2002.

RAZAVI, Behzad, **Fundamentos de Microeletrônica**. Editora LTC, 1 ed., 2010.

CAO, B. et. al. **CA Optimization Based on Simulation annealing in BIST.** *IEEE International Conference on Optoeletronics and Microeletronics,* pp. 138-141, 2013

CASEY, Erin. **Berlekamp-Massey Algorithm.** Disponível em: < http://www.math.umn.edu/~garrett/students/reu/MB_algorithm.pdf> Acesso em: 23 de fev. de 2016.

CHAND, R. et. al. **Fault Diagnosis for Using TPG Low Power Dissipation and High Fault Coverage.** *IEEE ICCIC,* pp. 5-10, 2010.

CHETAN, J., LAKKANNAVAR, M. **Design of Low Power Test Pattern Generator Using Low Transition LFSR for High Fault Coverage Analysis.** *I. J. Information Engineering and Electronic Business,* vol. 2, pp. 15-21, 2013.

ENMIN, T., SHENGDONG, S., WENKANG, S. **Power Reduction in BIST Design Based on Genetic Algorithm and Vector-Inserted TPG.** *IEEE ICEMI,* pp. 533-537, 2007.

ENMIN, T. SHENGDONG, S., YAN, Z. **Weighted Test Generator in Built-in Self-test Design Based on Genetic Algorithm and Cellular Automata.** *IEEE, ICEMI,* vol. 2, pp. 134-138, 2011.

FUJIWARA, H. **A Neural Netlist of 10 Combionational Benchmark Circuits**. in Proc. *IEEE International Symposium Circuits and Systems,* IEEE Press. Piscataway, N. J., pp. 695-698, 1985

GIRARD, P. et al. **Power-Aware Testing and Test Strategies for Low Power Devices.** Springer: 2010.

HUAN, Y-J., CHOU, C-W., LI, J-F. **A Low-Cost Built-In Self-Test Scheme for na Array of Memories.** *IEEE European Test Symposium*, 2010.

HUSSAIN, S., PRIYA, K. P. **Test Pattern Generator (TPG) for Low Logic Built-In Self-Test (BIST).** IEEE *Internacional Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering,* vol. 2, n. 4, 2013.

JUNMING, L., ZHENGHUI, L. **Effects of Delay Models on Maximum Power Estimation of VLSI Circuits.** in Proc. *International Conference on ASIC,* pp. 179-182, 2001.

KAJIHARA, S., et. al. **Combinationally Irredundant ISCAS89 Benchmark Circuits.** *IEEE International Symposium on Circuits and Systems,* vol. 4, pp. 632-364, 1996.

KASUNDE, P., SHIVAKUMAR, Dr. K. B., KURIAN, Dr. M. Z. **Improved Design of Low Power TPG Using LP-LFSR.** *IEEE International Journal of Computer & Organization Trends.* Vol. 3, pp.102-106, 2013.

KIRAN, R., YELLAMPALLI, S., HARISH, G. **Low Hardware Cost STUMPS BIST.** *IEEE Int. Symposium on VLSI Design and Test*, 2015.

KIRAN, R., et. al. **Low Power and Hardware Cost Stumps BIST.** *IEEE Int. Symposium on VLSI Design and Test*. pp. 1-4, 2015

KIRAN, R., YELLAMPALLI, S., HARISH, G. **Modified Low Power STUMPS Architecture.** IEEE ICECCE, 2014.

KIRAN, R., YELLAMPALLI, S. **Low Hardware Cost STUMPS BIST.** *In. Proc. I4C,* 2014.

KIRTHI, V, SAMSON, Mamatha. **Design of BIST with Low Power Test Generator.** *Journal of VLSI and signal processing*. pp. 30-39, vol. 4, 2014.

LIU, T. et. al., **A Built-In Self-Test Technique for Load Inductance and Lossless Current Sensing of DC-DC Converters.** *IEEE VLSI Test Symposium*, 2014.

LUCAS, D. **Algorítmos Genéticos:** uma Introdução. Disponível em <http://www.inf.ufrgs.br/~alvares/INF01048IA/ApostilaAlgoritmosGeneticos.pdf>. Acesso em: 23 de fev. de 2016.

MANICH, S., FIGUERAS, J. **Maximizing the Weighted Switching Activity in Combinational CMOS Circuits under the Variable Delay Model**, in Proc. *IEEE European Design and Test Conference,* pp. 597-602, 1997.

MARGADE, P. **Low Power Testing using Re-configurable Johnson Counter and Scalable SIC Counter.** IEEE ICCSP, 2015.

MITCHELL, M. **An Introduction to Genetic Algorithms.** 5 ed. 1999.

NAIM, L., CHANDEL, T. A. **Design of Low Transistion Pseudo-Random Pattern Generator for BIST Applications.** *IEEE International Journal of Computer Applications,* vol. 87, n. 15, 2014.

NICOLICI, Nicola, AL-HASHIMI, Bashir M. **Power-Constrained Testing of VLSI Circuits**. Estados Unidos da América: Kluwer Publishers, 2003.

NOURANI, M., TEHRANIPOOR, M., AHMED, N. **Low-Transition Test Pattern Generation for BIST-Based Applications.** *IEEE Transac. On Computers,* vol. 57, n. 3, 2008.

PAPA, G., GARBOLINO, T. **Optimal On-Line Built-In Self-Test Structure for System-Reliability Improvement.** *IEEE Congresso on Evolutionary Computation (CEC),* pp. 222-229, 2011.

PACHECO, M. A. **Algoritmos Genéticos: Princípios e Aplicações.** Disponível em: <www.ica.ele.puc-rio.br>. Acesso em: 23 de fev. de 2016.

PANDA, A. K., RAJPUT, P., SHUKLA, B. **FPGA Implementation of 8, 16 and 32 Bit LFSR with Maximum Length Feedback Polynomial using VHDL.** *IEEE International Conference on Communication Systems and Netword Technologies,* pp. 769-773, 2011

RAM, B.V., HARISH, G., YELLAMPALLI, S. **Stepped Segmented LFSR for Low Test Power BIST.** *IEEE ICSTM,* pp. 424-427, 2015.

RONGHUI, H., XIOWEI, L., YUNZHAN, G. **A Low Power BIST TPG Design.** *In Proc. 5th International Conference on ASIC.* volt. 2, pp. 1136-1139, 2003.

SINGH, A., KUMAR, P.M., BASSI, M. **Strategies and Tecniques for Optimizing Power in a BIST: A Review.** *IEEE. Int. Journal of Computer Applications.* Vol. 86, n. 4, 2014.

SOUZA, C. P. **Uma Arquitetura Autotestável para Circuitos Digitais Baseada no Algoritmo de Berlekamp-Massey e em Sistemas Imunológicos Artificiais.** 2005. 122 f. Teste de Doutorado em Engenharia Elétrica – Universidade Federal de Campina Grande, Campina Grande, 2005.

SUDIREDDY, S., KAKADE, J., KAGARIS, D. **Deterministic Built-In TPG with Segmented FSMs.** *IEEE International On-Line Testing Symposium,* pp. 261-266, 2008.

SUPRYIA, K., REKHA, B. **Implementation of Low Power Test Pattern Generator Using LFSR.** *International Journal of Science and Research,* vol. 2, pp.165-170 2013.

SHARMA, C., SABHARWAL, S., SIBAL, R. **A Survey on Software Testing Techniques using Genetic Algorithm.** *International Journal of Computer Science Issues,* vol. 10, n. 1, pp. 381-393, 2013.

TAN, E., WANG, L. **A Built-in Self-test Design with Low Power Comsumption Based on Genetic Algorithm.** *International Conference on Electronic Measurement & Instruments.* pp. 527-529, 2009.

TetraMAX Automatic Test Pattern Generation. Disponível em: < http://www.synopsys.com/Tools/Implementation/RTLSynthesis/Test/Pages/TetraMAXATPG.aspx>. Acesso em: 23 de fev. 2016.

Virginia Tech VLSI for Telecommunications. **CAD Tools for Testing**. Disponível em: < http://www.vtvt.ece.vt.edu/vlsidesign/cadtools.php>. Acesso em: 23 de fev. de 2016.

WANG, S. **A BIST TPG for Low Power Dissipation and High Fault Coverage.** *IEEE Transactions on VLSI Systems,* vol. 15, n. 7, 2007.

WANG. L-T., WU, C-W., WEN, X. **VLSI Test Principles and Architectures**. Morgan Kaufmann Publishers: 2006.

WESTE, Niel H. E., HARRIS, D. M. **CMOS VLSI Design:** A Circuits and Systems Perspective. 4 ed. Pearson: 2009.

YEAP, Gary K. **Practical Low Power Digital VLSI Design.** Springer: 1998.

YORIYAZ, H. Método de Monte Carlo: princípios e aplicações em Física Médica. **Revista Brasileira de Física Médica.** São Paulo, v. 3, n. 1, 2009.

ZHAO, W. et. al. **Power-Safe Application of Transition Delay Faults Patterns Considering Current Limit during Wafer Test.** *IEEE Asian Test Symposium,* pp. 301-306, 2010.