Usando o Algoritmo de Berlekamp-Massey em Testes de Circuitos Integrados

Cleonilson Protásio de Souza, Raimundo Carlos Silvério Freire e Francisco Marcos de Assis

***Resumo*— A principal contribuição deste trabalho é a introdução do Algoritmo de Berlekamp-Massey (BMA) na área de testes de circuitos integrados. Embora o BMA tenha sido inicialmente proposto para a localização de erros em sistemas de comunicações, de fato, esse provê uma solução geral para a síntese do mais curto registrador de deslocamento com realimentação linear (LFSR) capaz de gerar uma dada seqüência finita. Neste trabalho, mostra-se que com uma ligeira modificação no BMA, pode-se projetar um esquema de teste de circuitos integrados baseado totalmente em um único LFSR como gerador de testes no qual é capaz de obter cobertura de falha total e tendo tanto consumo de área de silício como tempo de teste reduzido.**

***Palavras-Chave*— Algoritmo de Berlekamp-Massey, Testes de Circuitos Integrados, Registrador de Deslocamento com Realimentação Linear.**

***Abstract*— The main contribution of this work is the introduction of Berlekamp-Massey algorithm (BMA) in the integrated circuit testing area. Although BMA had been initially used to error location in a communication system, it actually provides a general solution to synthesize the shortest Linear Feedback Shift Register (LFSR) capable of generating a given finite sequence in any field. In this work, it is shown that with a slight modification in the BMA, it is possible to design an integrated circuit test scheme based only on a single LFSR working as test generator. Such an LFSR is able to achieve complete fault coverage consuming reduced silicon area and test time.**

***Keywords*— Berlekamp-Massey Algorithm, Integrated Circuit Testing, Linear Feedback Shift Register.**

I. INTRODUÇÃO

O custo de se testar um circuito integrado (CI) é estimado em aproximadamente 25% do custo de produção [1] e com o avanço da tecnologia do processo de integração e com o aumento da complexidade no projeto dos CI's, o custo associado aos testes aumentarão consideravelmente [2]. Estudos indicam que o custo de teste tende a se igualar ao de produção no ano de 2015 [2].

Normalmente, testes de CI's são realizados utilizando-se **equipamentos de teste automático**. Tais equipamentos são usados na aplicação, através de pontas de sondagem, de vetores de testes no circuito sob teste (CUT[1]) e no recebimento das respostas aos testes com o objetivo de verificar se o circuito está falho ou não [3]. Um problema com esses equipamentos externos é o seu alto custo de aquisição, de operação, de treinamento, de manutenção e os custos relacionados com a compra e/ou desenvolvimento do *software* operacional [4].

Uma solução encontrada para diminuir esses custos foi o desenvolvimento de Circuitos Integrados Autotestáveis, chamados de BIST's[2] [5][6][7]. Na arquitetura básica de um BIST, a geração de vetores de testes e a avaliação das respostas do circuito são realizadas no próprio CI [7]. A arquitetura básica de um BIST pode ser vista na Figura 1 em que um gerador de testes (TPG[3]) é usado para aplicar uma seqüência de testes no CUT e as respostas a esses testes são aplicadas no Analisador de Resposta de Saída (ORA[4]) que verifica se o circuito está falho ou não. Algumas vantagens dos BIST's são [8]:

a) Operam na freqüência de operação do circuito (A maioria dos equipamentos automáticos de teste operam em baixa freqüência);

b) Atingem alta cobertura de falha, ou seja, detectam uma grande quantidade de falhas do circuito;

c) Provêem funcionalidades *on-line*. Dessa forma, BIST's podem ser usados tanto em testes na etapa de manufatura quanto em testes em campo (ou seja, em uso pelo consumidor final) [9];

d) E, principalmente, as arquiteturas BIST's são efetivas na redução de custos de testes [10].

Fig. 1. Arquitetura Básica de um Circuito Integrado Autotestável (BIST).

O desejável em uma arquitetura BIST é que a área de silício consumida pelo esquema de teste (gerador de testes e analisador de respostas), também chamada de sobreárea de *hardware*, e o tempo de teste sejam os menores possíveis e que o número de falhas detectado, denominado de cobertura de falha, seja ideal, ou seja, de 100%.

A princípio, em relação à cobertura de falha, o gerador de testes é o componente que mais influencia nesse parâmetro. Pois, de nada adianta otimizar o analisador de resposta se os testes oriundos do gerador não são capazes de possibilitar a detecção[5] de uma quantidade desejada de falhas (ou seja, têm baixa cobertura de falha) e/ou tem comprimento de teste proibitivo.

Cleonilson Protásio de Souza é aluno de doutorado pelo Departamento de Engenharia Elétrica da Universidade Federal de Campina Grande, PB, Brasil. Raimundo Carlos Silvério Freire e Francisco Marcos de Assis são professores do Departamento de Engenharia Elétrica da Universidade Federal de Campina Grande, PB, Brasil, E-mails: protasio@dee.ufcg.edu.br, rcsfreire@dee.ufcg.edu.br e fmarcos@dee.ufcg.edu.br. Este trabalho é apoiado pelo CNPq e pela CAPES.

[1]Da expressão em inglês, Circuit Under Test.

[2]Da expressão em inglês, Built-In Self-Test.

[3]Da expressão em inglês, Test Pattern Generator.

[4]Da expressão em inglês, Output Response Analyzer.

[5]Um vetor de teste, ou simplesmente teste, detecta um falha se, quando esse teste for aplicado no circuito sob teste, a resposta de saída resultante difere da resposta do circuito sem falhas.

Em geral, dois tipos de geradores de testes são bastante usados em arquiteturas BIST's, a saber: os **geradores de testes pseudo-aleatórios** e os **geradores de testes determinísticos**.

Os geradores pseudo-aleatórios são, na maioria das vezes, projetados utilizando-se registrado de deslocamento com realimentação linear (LFSR[6]) por este ter uma estrutura compacta e, conseqüentemente, ocupar pouca sobreárea de *hardware*. Entretanto, devido à presença de **falhas de difícil detecção** por testes pseudo-aleatórios, faz-se necessário a geração de uma quantidade muita alta de vetores de testes fazendo com que o tempo de teste seja extremamente alto para atingir uma cobertura de falha adequada.

Os geradores determinísticos, por sua vez, geram vetores de testes previamente computados denominados de testes determinísticos [11] [12] [13]. A grande vantagem do gerador determinístico é que o tempo de teste é bastante reduzido, pois são necessários poucos testes para obter uma cobertura de falha desejada. A desvantagem é que a sobreárea de hardware, normalmente, é alta devido o fato da necessidade de gerar esses testes no próprio CI. Normalmente, esses são armazenados em memória.

Na tentativa de aproveitar as vantagens dos dois tipos de geradores de testes descritos, foi idealizado o **gerador de testes pseudo-determinísticos**, também denominado de **gerador de testes mistos**, o qual gera testes determinísticos para detectar as falhas de difícil detecção do circuito e testes pseudo-aleatórios para detectar as falhas restantes [14]. Em geral, estruturas adicionais são necessárias para **controle de operação**, para **armazenamento** e/ou para alguma **lógica combinacional** como podem ser vistos na Figura 2. Todas essas estruturas adicionais consomem sobreárea de *hardware*.

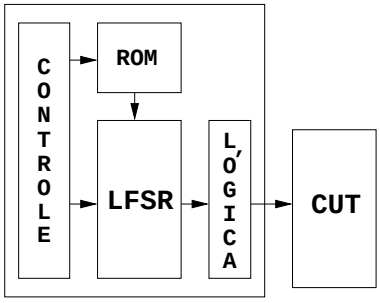

Fig. 2. Estruturas adicionais em gerador de testes mistos.

A principal contribuição deste trabalho é a introdução do Algoritmo de Berlekamp-Massey (BMA) na área de testes de circuitos integrados. O BMA sintetiza o mais curto LFSR capaz de gerar uma seqüência pré-definida de elementos em um dado alfabeto. O objetivo principal é descrever os procedimentos de desenvolvimento de um gerador de testes pseudo-determinísticos baseado totalmente em um único LFSR. Tal LFSR é sintetizado pelo Algoritmo de Berlekamp-Massey ligeiramente modificado. Tal modificação permite a adaptação do algoritmo aos requisitos de testes e, em adição, tenta otimizar o LFSR resultante.

O método de projeto proposto é orientado pelas falhas de difícil detecção[7] presentes no circuito. Assim, especificam-se testes determinísticos que detectam essas falhas difíceis e sintetiza-se, através do BMA modificado, um LFSR que gera esses testes. Desta forma, a princípio, o LFSR comportasse como um gerador determinísticos. Porém, o LFSR não necessariamente gera somente esses testes, mas, de fato, gera diversos outros testes "residuais", que podem ser utilizados no teste das falhas restantes.

Portanto, utilizando esse LFSR como gerador de testes, pode-se gerar tanto testes determinísticos quanto testes "residuais", conceitualmente testes pseudo-aleatórios, para o teste completo do circuito. Utilizando esse método, resultados experimentais que comprovam a eficiência do método são mostrados utilizando-se os circuitos de verificação de desempenho nos padrões *ISCAS'85* e *ISCAS'89*. Tais padrões são amplamente utilizados na área de testes de circuitos integrados.

II. Algoritmo de Berlekamp-Massey

O ingrediente conceitual principal utilizado no projeto do gerador misto proposto é a aplicação do Algoritmo de Berlekamp-Massey (BMA). Embora o BMA tenha sido inicialmente proposto para a localização de erros em sistemas de comunicações [15], de fato, esse provê uma solução geral para a síntese do mais curto LFSR capaz de gerar uma dada seqüência finita $R = (R_1, R_2, \quad , R_n)$ em qualquer campo, seja finito ou infinito [16] [17].

Quando uma seqüência R é aplicada no BMA, tem-se como resultado os coeficientes do polinômio de realimentação, $\Lambda(x) = \Lambda_0 + \Lambda_1 x + \Lambda_2 x^2 + \quad + \Lambda_L x^L$, com grau L, do mais curto LFSR capaz de gerar R. Note que o problema solucionado pelo BMA é equivalente ao da obtenção de um filtro de resposta ao impulso infinito capaz de reproduzir (predizer) a seqüência R dos seus L valores iniciais. Dessa forma, carregando os L primeiros elementos de R no LFSR obtido, esse é capaz de gerar toda a seqüência R, como pode ser visto na Figura 3. O BMA é reproduzido na Figura 4.

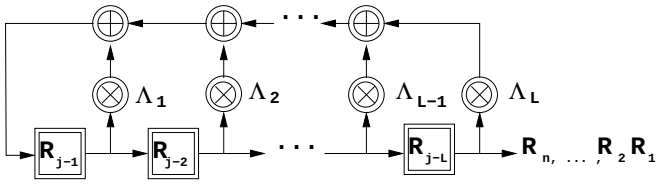

Fig. 3. Configuração do LFSR sintetizado via BMA.

Em termos gerais, o BMA opera da seguinte forma: após a devida inicialização e em cada iteração, é computada uma discrepância Δ a qual é definida com a diferença entre o valor "atual"e o valor "predito" do j-ésimo elemento de R, i.e., $\Delta = R_j - \hat{R}_j$ em que $1 \quad j \quad n$ e $\hat{R}_j = -\sum_{i=1}^{L} \Lambda_i R_{j+1} \ _i$. Dessa forma, em cada iteração j, se $\Delta = 0$, então o comprimento do LFSR não muda e o seu polinômio de realimentação fica igual a aquele definido na última mudança. Entretanto, se $\Delta \neq 0$, então o comprimento do LFSR pode mudar ou não de acordo

[6]Da expressão em inglês, Linear Feedback Shift Register.

[7]É importante salientar que existem em um circuito falhas de fácil detecção e as de difícil detecção, sendo que as primeiras apresentam-se em maior quantidade, normalmente, acima de 90% de todas as falhas.

ENTRADA: $R_1, R_2, \quad , R_n$
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
$\Delta = R_j - \hat{R}_j = R_j + \sum_{i=1}^{L} \Lambda_i R_{j+1 \ \ i}$
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
$\Lambda(x) \leftarrow \Lambda(x) - \Delta b \ \ ^1x^d B(x)$
$d \leftarrow d + 1$, vá para o passo 5.
Passo 4. Se $\Delta \neq 0$ e $2L \quad j$, então
$Temp(x) \leftarrow \Lambda(x)$
$\Lambda(x) \leftarrow \Lambda(x) - \Delta b \ \ ^1x^d B(x)$
$L \leftarrow j + 1 - L$
$B(x) \leftarrow Temp(x) \qquad b \leftarrow \Delta \qquad d \leftarrow 1$
Passo 5. $j \leftarrow j + 1$, retorne ao passo 1.

Fig. 4. Algoritmo de Berlekamp-Massey.

com as seguintes situações: se $2L > j$, então o comprimento não muda (observe que o polinômio de realimentação pode mudar). Caso contrário, é computado um novo LFSR capaz de gerar a seqüência até essa iteração.

III. Adaptando o BMA para operar com bits não especificados

Geralmente, em um vetor de teste, não é necessário especificar todos os seus bits como sendo 0 ou 1, ou seja, são bits irrelevantes (X's), pois há falhas que podem ser detectadas sem que todos os bits de entradas sejam especificados. Um exemplo pode ser o vetor $0XX10X$. Tais vetores são denominados de **hipercubos de testes**.

Considere o BMA operando em campo binário $(0, 1)$ e que a seqüência abaixo deve ser aplicada no BMA a fim de sintetizar um LFSR que a reproduza.

$(0\,1\,1\,X\,0\,X\,X\,1\,1\,X\,X\,X\,X\,X\,X\,0\,X\,1\,1\,X\,1\,X\,1\,X\,0\,0\,X\,1\,X\,X\,X)$

Como o BMA não opera com bits irrelevantes, podem-se escolher aleatoriamente os valores dos X's. Porém, uma questão surge: quais os valores dos X's que diminuem o comprimento do LFSR resultante?

Para tentar resolver essa questão, é proposta uma modificação a fim de que o BMA seja capaz de operar com X's e que, em adição, otimize o comprimento do LFSR resultante.

A modificação sugerida é realizada no ***passo 1*** do algoritmo original e é baseada no fato que se $R_j = X$, então R_j é computado como $R_j = \sum_{i=1}^{L} \Lambda_i R_{j+1 \ \ i}$ e, no próximo passo, a discrepância Δ é garantida ser zero, pois, nesse caso, $\Delta = R_j - \hat{R}_j = \sum_{i=1}^{L} \Lambda_i R_{j+1 \ \ i} - \sum_{i=1}^{L} \Lambda_i R_{j+1 \ \ i} = 0$. Então, dessa forma, o comprimento L não é aumentado. A modificação proposta do BMA é vista na Figura 5.

INICIALIZAÇÃO:
$\vdots$
ITERAÇÃO:
Passo 1. Se $j = n$, pare. Caso contrário compute
$\hat{R}_j = \sum_{i=1}^{L} \Lambda_i R_{j+1 \ \ i}$
Se $R_j = X$, faça $R_j = \hat{R}_j$
$\Delta = R_j - \hat{R}_j$
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
$\Lambda(x) \leftarrow \Lambda(x) - \Delta b \ \ ^1x^d B(x)$
$d \leftarrow d + 1$, vá para o passo 6.
Passo 4. Se $\Delta \neq 0$ e $2L \quad j$, então
$Temp(x) \leftarrow \Lambda(x)$
$\Lambda(x) \leftarrow \Lambda(x) - \Delta b \ \ ^1x^d B(x)$
$L \leftarrow j + 1 - L$
$B(x) \leftarrow Temp(x) \qquad b \leftarrow \Delta \qquad d \leftarrow 1$
Passo 5. $j \leftarrow j + 1$ e retorne ao passo 1.

Fig. 5. BMA capaz de operar com bits irrelevantes.

Exemplo: Considere a seguinte seqüência com 31 elementos:

$(0\,1\,1\,X\,0\,X\,X\,1\,1\,X\,X\,X\,X\,X\,X\,0\,X\,1\,1\,X\,1\,X\,1\,X\,0\,0\,X\,1\,X\,X\,X)$

Aplicando essa seqüência no BMA modificado, é obtido o seguinte polinômio de realimentação: $\Lambda(x) = 1 + x^3 + x^5$ no qual é o polinômio do LFSR capaz de gerar a seqüência:

$(0\,1\,1\,0\,0\,1\,1\,1\,1\,1\,0\,0\,0\,1\,1\,0\,1\,1\,1\,0\,1\,0\,1\,0\,0\,0\,0\,1\,0\,0\,1)$

na qual os X's foram especificados de maneira a não incrementar o comprimento do LFSR quando o algoritmo está sendo executado.

IV. Procedimentos de Projeto do Gerador de Testes Proposto

Como visto na Seção I, o gerador de testes proposto é baseado somente em um único LFSR sintetizado pelo BMA. Dessa forma, esse LFSR deve ser capaz de gerar tanto testes determinísticos pré-computados, para detectar as falhas de difícil detecção do circuito, quanto outros vetores de testes para detectar as falhas restantes. O esquema do gerador misto proposto é visto na Figura 6.

O parâmetro principal de verificação de desempenho, em relação à sobreárea de *hardware*, é o comprimento L do LFSR sintetizado pelo BMA, pois esse parâmetro é proporcional à sobreárea de *hardware* consumida pelo esquema. Dessa forma, a tentativa de diminuir o comprimento L do LFSR é de grande valia para o esquema proposto.

A. Aplicando o BMA modificado

Suponha que em um circuito com q entradas, foram identificadas k falhas de difícil detecção (formando o conjunto $F = \{f_1, f_2, \quad , f_k\}$) e que k hipercubos de testes determinísticos, formando o conjunto $T = (T^1, T^2, \quad , T^k)$, foram especificados para detectar as falhas em F, sendo $T^i = (T^i_q, \quad , T^i_2, T^i_1)$ para $1 \quad i \quad k$.

Um fato conhecido é que operações aritméticas em campos finitos, tais como $GF(2^q)$, em que $q > 1$, são muitos mais

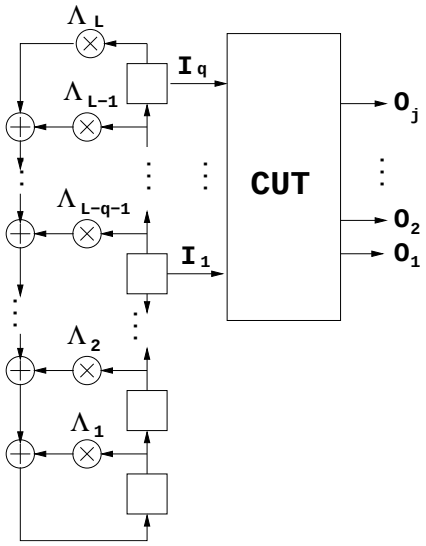

Fig. 6. Configuração do gerador de testes mistos proposto.

dispendiosas em termos de área de circuito do que operações em campo binário, $GF(2)$ [18] [19]. Então, para economizar sobreárea de *hardware*, executar-se-á o BMA em $GF(2)$.

Para tanto, T tem que ser submetido a um processo de serialização, como descrito a seguir. Considere $t = t_m, \quad , t_2, t_1$, em que $t_i \in (0,1)$ e $m = k \quad q$, como sendo a forma serializada de T. Dessa forma, tem-se:

$$t = (T_q^1, \quad , T_1^1; T_q^2, \quad , T_1^2; \quad ; T_q^k, \quad , T_1^k) \qquad (1)$$

Após a serialização, verificar-se que todos os testes em T ainda estão presentes em t.

Após isso, aplica-se t no BMA modificado a fim de obter o menor LFSR que gere t.

Obtido o LFSR, a operação do esquema consiste no fato que em cada q deslocamentos do LFSR (q *ciclos de relógio*) um teste determinístico presente em T é aplicado no circuito, como pode ser visto na Figura 7. Desta forma, no início da operação do LFSR, o teste T^1 é aplicado. Após q deslocamentos, T^2, é aplicado e assim por diante. Dessa forma, após $(k-1)q$ deslocamentos do LFSR, todos os testes em T são aplicados no circuito e, portanto, todas as falhas de difícil detecção identificadas são detectadas. Pode-se verificar que entre e após a geração dos testes determinísticos, outros vetores "residuais" são gerados pelo LFSR. Tais vetores são suficientes na detecção das falhas restantes do circuito, que são de fácil detecção.

Assim o LFSR sintetizado é capaz de gerar tantos os testes determinísticos em T, que detectam as falhas de difícil detecção, quanto outros testes residuais que detectam as falhas restantes. Sendo assim um gerador de teste misto.

V. Exemplo de Aplicação

Considere o circuito $c432$ (um decodificador de prioridade) da família de circuitos no padrão *ISCAS85*, o qual tem 36 entradas, 7 saídas e 524 falhas. Foram identificadas 12 falhas de difícil detecção que são detectadas, respectivamente, pelos hipercubos de testes determinísticos vistos na Figura 8.

```
111x00xxx0xxx0xxx0xxx0xxx0xxx0xxx0xx
x0x1x11x00xxx0xxx0xxx0xxx0xxx0xxx0xx
x0xxx0xxx0xxx0xxx0x1x11x00xxx0xxx0xx
x0xxx0x1x11x00xxx0xxx0xxx0xxx0xxx0xx
x0xxx0xxx0xxx0xxx0xxx0xxx0xxx0x1x110
x0xxx0xxx0xxx0xxx0xxx0xxx0x1x11x00xx
x0xxx0xxx0xxx0xxx0x1x11x00xxx0xxx0xx
x0xxx0xxx0xxx0xxx0xxx0xxx0xxx0x1x100
x0xxx0xxx0xxx0xxx0xxx0x1x11x00xxx0xx
x0xxx0xxx0x1x11x00xxx0xxx0xxx0xxx0xx
1101010x00xxx0xxx0xxx0xxx0xxx0xxx0xx
x0xxx0xxx0xxx0xxx0xxx0xxx0x1x10x00xx
```

Fig. 8. Hipercubos de testes que detectam as falhas de difícil detecção do circuito $c432$.

Aplicando esses hipercubos, após passarem pelo processo de serialização, no BMA modificado, é sintetizado um LFSR de comprimento igual a 79. Os vetores de testes gerados por esse LFSR são vistos na Figura 9. Observe que todos os testes determinísticos, provenientes dos hipercubos de testes, estão nessa seqüência (envolto por um retângulo). Como pode ser visto, outros vetores residuais são também gerados. Assim, as falhas de difícil detecção do circuito são detectadas pelos testes determinísticos e as falhas restantes pelos testes residuais. Nesse exemplo, foram necessários 435 deslocamentos do LFSR, ou seja, 435 testes para que todas as falhas do circuito fossem detectadas (cobertura de falha $= 100\%$).

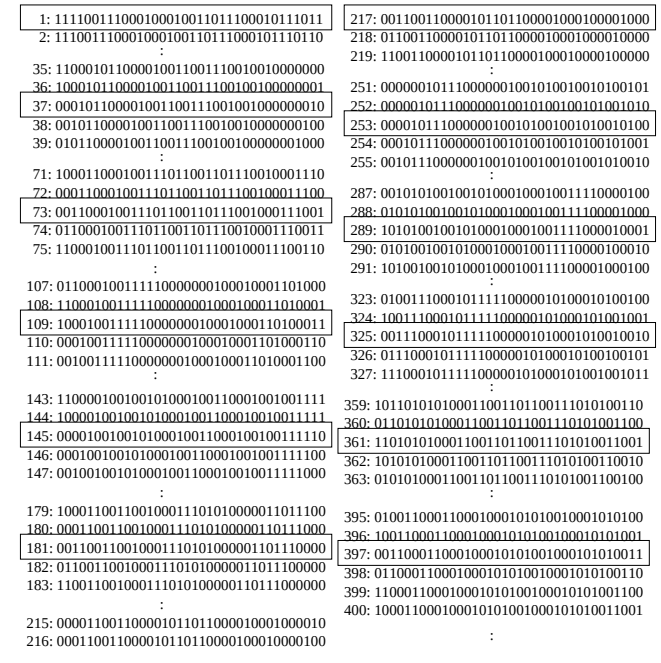

Fig. 9. Testes gerados pelo gerador misto baseado em LFSR para o teste do circuito $c432$.

VI. Resultados Experimentais

De maneira a validar a efetividade do esquema proposto, simulações experimentais foram realizadas usando os circuitos de verificação de desempenho nos padrões *ISCAS85* e a parte combinacional dos circuitos nos padrões *ISCAS89*.

Os resultados foram obtidos por meio de um simulador desenvolvido nesta pesquisa, chamado de **LFSRMAKER**, desenvolvido em linguagem *C++*. Foram também utilizados,

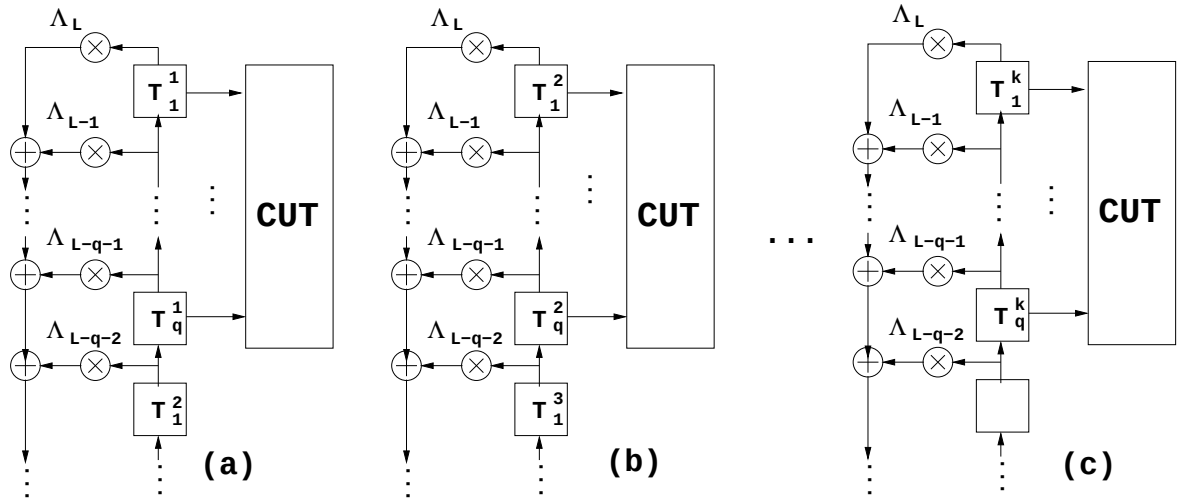

Fig. 7. (a)Inicialmente, T^1 é aplicado no CUT. (b) Após q deslocamentos do LFSR, T^2 é aplicado no CUT. (c) Finalmente, após $(k \quad 1)q$ deslocamentos, T^k é aplicado no CUT. Durante e após isso, outros vetores de testes são gerados e aplicados no CUT.

os simuladores ATALANTA [20] para a especificação de testes determinísticos e o FSIM [21] como simulador de falhas. Os procedimentos experimentais foram os seguintes:

1) Identificação das falhas de difícil detecção do circuito. O algoritmo visto na Figura 10 foi usado nessa identificação. Esse algoritmo utiliza o método de Monte Carlo clássico, que consiste em realizar vários experimentos a partir de amostras aleatórias a fim de inferir os valores de certos parâmetros. Dessa forma, faz-se vários experimentos (cada experimento consiste em se aplicar de uma seqüência pseudo-aleatória de testes no circuito e anotar as falhas não-detectadas por essa seqüência) e, no final, faz-se uma lista das falhas não-detectadas. As que mais aparecerem nessa lista são consideradas como falhas de difícil detecção.

```
INÍCIO
    ENQUANTO NÃO fim, FAÇA
        INÍCIO
            Gere uma seqüência de testes pseudo-aleatórios.
            Aplique essa seqüência no circuito.
            Salve em uma lista as falhas não-detectadas pela seqüência.
        FIM
    Identifique como FDD's, as falhas mais presentes na lista.
FIM
```

Fig. 10. Algoritmo usado na identificadas de falhas de difícil detecção (FDD's).

2) A partir das falhas de difícil detecção identificadas, gerar um conjunto T de hipercubos de testes determinísticos, usando o ATALANTA, que detectem essas falhas.

3) Aplicar T aos procedimentos para a sintetização do LFSR proposto neste trabalho.

4) A partir do LFSR sintetizado, base do gerador misto proposto, determinar a sobreárea de *hardware* consumida e o comprimento de teste que proporcione 100% de cobertura de falha.

Para o cálculo da sobreárea de *hardware*, foi adotado o método da **equivalência de portas** [24] que utiliza uma unidade de medida chamada de GE[9]. Nesse método, a cada 4 transistores utilizados em um bloco lógico, considera-se que esse consome 1 GE. Assim, tem-se que, uma porta *NAND* ou uma porta *NOR* consome 1 GE, pois podem ser construídos usados somente 4 transistores cada.

Como a estrutura de um LFSR só consome portas *XOR* e *flip-flop*'s, faz-se necessário determinar a equivalência de portas dessas estruturas. Verificou-se que cada porta *XOR* consume 1.5 GE, pois essa porta pode ser construída usando um *MUX*$(2 \rightarrow 1)$ (que consome 1 GE) e uma porta *NOT* (que consome 0.5 GE). De acordo com [22] cada *flip-flop* consome 3.5 GE.

Os resultados experimentais obtidos foram comparados com os resultados obtidos pelos esquemas propostos por Kavousianos [22] e por Chiusano [23]. A comparação consistiu em verificar o comprimento de teste requerido de cada esquema para atingir 100% de cobertura de falha. O comprimento de teste para cada circuito são vistos nas Colunas 4, 6 e 8 da Tabela I. Em relação a sobreárea de *hardware*, os resultados podem ser vistos na Coluna 3, 5 e 7. Entre esses resultados, o símbolo (-) indica que o autor não forneceu a informação requerida.

Da Tabela I, pode ser observado que, para a maioria dos circuitos, o método proposto proporciona comprimento de testes menores que os resultados apresentados em [23]. E, por outro lado, também para a maioria dos circuitos, consome menos sobreárea de *hardware* que os resultados apresentados em [22].

VII. Conclusões

Tendo como idéia principal o uso do algoritmo de Berlekamp-Massey, amplamente aplicado em sistemas de comunicações, um método de desenvolvimento de um gerador de testes mistos aplicado em BIST foi apresentado. O gerador proposto é baseado totalmente em um único e simples LFSR sintetizado pelo algoritmo de Berlekamp-Massey ligeiramente

[9]Da expressão em inglês, Gate Equivalent.

TABELA I

COMPARAÇÕES ENTRE AS SOBREÁREA DE *hardware* (GE) E COMPRIMENTO DE TESTE (l_{teste}) OBTIDOS PELO MÉTODO PROPOSTO E OS MÉTODOS EM [22] E [23] PARA ATINGIR 100% DE COBERTURA DE FALHA.

Circuito	# Entradas	# Saídas	# Falhas	Método proposto		[22]		[23]	
				GE	l_{teste}	GE	l_{teste}	GE	l_{teste}
c880	60	26	942	91.5	314	159.5	248	0	1829
c1355	41	32	1574	143.5	1421	281	465	0	1334
c1908	33	25	1879	92	2883	147	1397	0	3759
c3540	50	22	3428	202	2724	-	-	0	4505
s420	35	17	430	348	2336	141.5	630	> 97	10843
s641	54	42	463	189.4	1996	106.5	248	> 94.5	2430
s1196	32	32	1242	114	14321	-	-	> 40	18776
s713	54	42	581	51.5	1877	-	-	> 108	2759

modificado para fins de adaptação e de otimização. Esse LFSR é capaz de gerar testes determinísticos, que detectam as falhas de difícil detecção do circuito, e testes residuais que detectam as falhas restantes. De acordo com os resultados experimentais obtidos, o método mostrou-se efetivo na redução da sobreárea de *hardware* e do comprimento de teste, além de permitir uma operação de controle bastante simples na geração dos testes.

REFERÊNCIAS

[1] Curry, R.J. Evaluation of economics of testing in a manufacturing environment. *In Proceedings of IEEE AUTOTESTCON*, p. 481 – 489, 2002.

[2] Gonciari, P.; Al-Hashimi, B. e Nicolici, N. Test cost reduction through compression. *Electronics Systems and Software*, v. 1, p. 37–41, June/July 2003.

[3] Zhang, S.; Choi, M.; Park, N. e Lombardi, F.;. Probabilistic balancing of fault coverage and test cost in combined built-in self-test/automated test equipment testing environment. *Proceedings of 19th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT 2004)*, p. 48 – 56, October 2004.

[4] Ungar, L.Y. e Ambler, T. Economics of built-in self-test. *IEEE Design & Test of Computers*, v. 18, n. 5, p. 70–79, Sept-Oct 2001.

[5] Jas, A.; Krishna, C.V. e Touba, N.A. Weighted pseudorandom hybrid BIST. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, p. 1277 – 1283, Dec 2004.

[6] Kalligeros, E.; Kavousianos, X. e Nikolos, D. Multiphase BIST: a new reseeding technique for high test-data compression. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, p. 1429 – 1446, Oct 2004.

[7] Youhua Shi e Zhe Zhang. Multiple test set generation method for LFSR-based BIST. *Proceedings of the Design Automation Conference (ASP-DAC)*, p. 863 – 868, Jan 2003.

[8] Dufaza, C. Theoretical properties of LFSRs for built-in self-test. *INTEGRATION, the VLSI journal*, p. 17–35, 1998.

[9] Abramovici, M; Breuer, M. A. e Friedmann, A. D. *Digital System Testing and Testable Design*. [S.l.]: IEEE Press/ W.H.Freeman, 1990.

[10] Hiraide, T.; Boateng, K. O.; Konishi, H.; Itaya, K.; Emori, M.; Yamanaka, H. e Mochiyama, T. BIST-aided scan test - a new method for test cost reduction. *Proceedings of 21st VLSI Test Symposium*, p. 359 – 364, April - May 2003.

[11] Schulz, M., Trischler, E. e Sarfert, T. M. Socrates: a Highly Effi cient Automatic Test Pattern Generation System. *IEEE Transation on Computer-Aided Design*, v. 7, n. 1, p. 126–137, January 1988.

[12] Hellebrand, S., Reeb, B., Tarnick S. e Wunderlich, H. J. Pattern Generation for a Deterministic BIST Scheme. *ACM/IEEE International Conference on CAD95, ICCAD-95*, San Jose, Ca, p. 1–7, November 1995.

[13] Pomeranz, I., Reddy, L. N. e Reddy, S. M. Compactest: a Method to Generate Compact test Sets for Combinational Circuits. *IEEE International Test Conference*, p. 194–203, 1991.

[14] Karkala, M., Touba, N. A. e Wunderlich, H. J. Special ATPG to Correlate Test Patterns for Low-Overhead Mixed-Mode BIST. *7th. Asian Test Symposium*, v. 32, p. 462–469, December 1998.

[15] Blahut, R. E. *Theory and Practice of Error Control Codes*. Owego, NY: Addison-Wesley Publishing Company, Inc., 1983.

[16] Massey, J. L. Shift-Register Synthesis and BCH Decoding. *IEEE Transactions on Information Theory*, v. 15, n. 1, p. 122–127, January 1969.

[17] Gustavson, F. G. Analysis of the Berlekamp-Massey linear feedback shift-register synthesis algorithm. *IBM J. Res. Dev.*, p. 204–212, 1976.

[18] Assis, F. M. e Pedreira, C. E. An architecture for computing Zech's logarithms in GF(2^m). *IEEE Transactions on Computers*, v. 49, n. 5, p. 519–524, 2000.

[19] Mastrovito, E. D. *VLSI Architecture for Computations in Galois Fields*. [S.l.]: Linkoping studies in science and technology. Dissertation, 1991.

[20] H.K. Lee e D.S. Ha. Atalanta: an effi cient ATPG for combinational circuits. *Technical Report, 93-12, Dep't of Electrical Eng., Virginia Polytechnic Institute and State University, Blacksburg, Virginia*, 1993.

[21] H. K. Lee e D. S. Ha. An effi cient forward fault simulation algorithm based on the parallel pattern single fault propagation. *In Proceedings of International Test Conference*, p. 946–955, October 1991.

[22] Kavousianos, X.; Bakalis, D.; Nikolos, D. e Tragoudas, S. A new built-in TPG method for circuits with random pattern resistant faults. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, v. 21, n. 7, p. 859–866, Jul 2002.

[23] Chiusano, S.; Prinetto, P. e Wunderlich, H. J. Non-intrusive BIST for systems-on-a-chip. *Proceedings of the International Test Conference*, p. 644– 651, 2000.

[24] Huang, L. R. ; Jou, J.Y. e Kuo, S.Y. Gauss-elimination-based generation of multiple seed-polynomial pairs for LFSR. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, v. 16, n. 9, p. 1015–1024, September 1997.