システムＬＳＩ設計技術　95－6
（2000．2．9）

ディレイ故障を検出するBIST用テストパターン発生回路

浅川 毅*　金本 直之*　岩崎 一彦*　梶原 誠司**

*東京都立大学　大学院　工学研究科
〒192-0397　東京都八王子市南大沢1-2
**九州工業大学　情報工学部　電子情報工学科
〒820-8502　福岡県飯塚市川津680-4

あらまし　擬似ランダムパターンテストの欠点である長いテスト長，を補う一つの手法として，ディレイ故障に対し短いテスト長で高い故障検出効率を達成するBIST用テストパターン発生回路(TPG)を提案する．提案TPGは1個のLFSR，2個のシフトレジスタ，少量のROM，マルチプレクサによって構成され，ROMにはATPGの部分列を格納する．ATPGベクトルの部分列とLFSRで発生した擬似ランダムベクトルを混在させたテストパターンを発生する．ランダムパターン検出困難故障をATPGベクトルの部分列によって検出するので，短いテスト長で高い故障検出効率を達成できる．動作クロックごとにスキャンインを行う必要がないため，低速テスタを使用した実動作速度テストが可能である．提案手法をISCASベンチマーク回路に適用した場合のテスト長，故障検出効率，ハードウェア量に関して評価し，本手法の有効性を考察する．
キーワード　TPG，ディレイ故障，ランダムパターン検出困難故障，ATPGベクトル，実動作速度テスト，低速テスタ

BIST Pattern Generator for Detection of Delay Faults

Takeshi Asakawa*, Naoyuki Kanamoto*, Kazuhiko Iwasaki*, Seiji Kajihara**

***Graduate School of Engineering, Tokyo Metropolitan University**
Hachioji, Tokyo 192-0397 Japan
****Kyushu Institute of Technology**
Iizuka, Fukuoka 820-8502 Japan

Abstract　We propose a test pattern generator (TPG) for BIST that can effectively detect delay faults and has a short testing time. The proposed TPG consists of a linear feedback shift register (LFSR), two simple shift registers, and a small amount of ROM that stores a subset of test vectors generated by an Automatic Test Pattern Generator (ATPG). The TPG generates test patterns that are the result of mixing the ATPG vectors and pseudo-random vectors generated by the LFSR. It is possible to detect delay faults as effectively as with a full ATPG vector set because random-pattern resistant faults can be detected by applying the subset of ATPG vectors stored in the ROM. Experimental results show that the proposed method can reduce the testing time required to achieve a higher rate of fault detection than an LFSR-based method. An advantage of this method is that testing can be carried out at-speed even by a low speed tester because at-speed scan-in is not required.

key words　BIST, delay fault, random pattern resistant fault, ATPG vector, low speed tester

1. まえがき

VLSI の大規模化に伴い，テストに要する費用の削減および質の向上が一層求められている．BIST(Built-In Self-Test)は一つの有効なテスト手法となっている[1]-[3]．BIST ではテスト設備のコストダウンに加えて，実動作速度によるテストの品質向上が期待できる．BIST 用テストパターン発生回路(TPG: Test Pattern Generator)として，少ないハードウェアで高い故障検出効率を達成し，かつテスト時間ができるだけ短いことが望ましい．

ATPG(Automatic Test Pattern Generator)によって得られたすべてのテストパターンを ROM に格納した TPG を使用すると，短いテスト長で100%の故障検出効率を達成することが可能である．しかし，必要なハードウェアは大きい．LFSR(Linear Feed-back Shift Register)を用いて擬似ランダムパターンを発生する TPG では，ハードウェアは少ないがランダムパターン検出困難(rpr: random pattern resistant)故障が CUT(Circuit Under Test)に存在するために，100%の故障検出効率を達成するためには非常に長いテスト長を必要とする[4]．

縮退故障をテストする BIST では，テスト長を短くするために様々な手法が提案されている．例えば CUT に観測点を挿入し rpr 故障の数を減らしたテスト容易化設計手法[5], [6]が提案されている．また特定のビット位置の 1 出現確率に重みを付ける重み付き擬似ランダムパターン手法[7]，特定の論理値を確実なビット位置に固定するビットフィックス手法[8], [9], [10]，無効パターンのビット位置をわずかに変えるビットフリッピング手法[11]，LFSR やカウンタの初期値をリシードする手法[12], [13]が提案されている．ATPG によって得られたベクトル（以下 ATPG ベクトル）を利用する BIST 用 TPG も提案されている[14]．またランダムパターンでテスト容易な論理合成法も提案されている[15]．

ディレイ故障の検出には,連続した 2 つのベクトル系列を必要とするため、rpr 故障の検出は，縮退故障の場合に比べ一層困難となる．ディレイ故障テストを実行する BIST として LFSR の初期値や多項式を入れ換える手法[16], [17]が提案されている．また実動作速度でディレイ故障テストをするための手法[18]が提案されている．これらの手法では故障検出に有効なベクトル系列テストパターン中にを増やすことができる．しかし，100%の故障検出効率を達成するには非常に長いテスト長を必要とする．

本論文では，短いテスト長で高い故障検出効率を達成する TPG を提案する．動作クロックごとにスキャンインを行う必要がないため，低速テスタを使用した実動作速度テストが可能である．提案 TPG は 1 個の LFSR，2 個のシフトレジスタ，少量の ROM，マルチプレクサによって構成され，ROM には ATPG の部分列を格納する．ATPG ベクトルの部分列と LFSR で発生した擬似ランダムベクトルを混在させたテストパターンを発生する．ランダムパターン検出困難故障を ATPG ベクトルの部分列によって検出するので，ATPG ベクトルをすべて使用する故障検出効率を達成できる．

以下，2 章で用語を定義する．3 章で提案手法による TPG 構成，ATPG ベクトルと擬似ランダムベクトルの混在テストパターン発生手順，ROM に格納する ATPG ベクトルの選択，および圧縮手順を述べる．次に 4 章で，提案手法を ISCAS ベンチマーク回路に適用した結果を述べ，故障検出効率，テスト長，ハードウェアの面より本手法の有効性を考察する．最後に 5 章で結論を述べる．

2. 準備

ディレイ故障にはトランジション故障，ゲート遅延故障，パス遅延故障が知られている．本論文では，組合せ回路のトランジション故障をテストの対象とし，テストの質の評価尺度として冗長故障を除く故障検出効率(fault efficiency)を用いる．

故障検出効率=検出された故障数/(生じうる故障数-冗長故障数)

である．

トランジション故障を検出するためには，連続した 2 つのテストベクトル系列(2 パターンベクトル系列) v_i, v_{i+1}を入力しなければならない．2 パターンテストではベクトルの並び方も重要な意味を持つ．テストベクトル v_i によって内部のある信号線を 0 または 1 に設定する．引き続く v_{i+1} によってその信号線を逆極性に遷移させると同時に原始出力(primary output)にその遷移を伝播させる．v_i または v_{i+1} の一方はランダムパターンで置き換えられる可能性がある．

CUT の入力数を m，対象トランジション故障を f_t (t=1, 2, ..., N)，対象トランジション故障 f_t を検出できるベクトル対の集合を $S(f_t)$と表す．ランダムな 2 パターンベクトルによって f_t が検出できる確率 $P(f_t)$は

$$P(f_t)=|S(f_t)| / (2^m \times 2^m)$$

と表すことができる．$P(f_t)$が小さいほどランダムベクトルでf_tを検出する確率は低くなる．$P(f_t)$が小さい故障f_tをランダムパターン検出困難トランジション(**rpr transition**)故障と呼ぶ．

2元のm次元ベクトル全体からなる集合をV_mと表す．V_mの要素からなる長さ$|V|$の系列を

$$V = v_0, v_1, v_2, \ldots, v_{|V|-1}$$

と表す．

回路Cを考える．ATPGツールを用いてトランジション故障に対するテストベクトルの系列を求める操作を**ATPG**(C)と表し，得られたn個のm次元ベクトルの系列をV_cと表す．$n = |V_c|$である．2パターンベクトル系列(v_0, v_1), (v_1, v_2), …, (v_{n-2}, v_{n-1})によってトランジション故障が検出される．V_cで達成される故障検出効率をATPG検出効率と定義する．

関数**SIM**(C, V)は，回路Cに対し，ベクトル系列Vを使用した故障シミュレーションを行い故障検出効率を求める．

関数**COVER**(C, V)は，回路Cに対し，ベクトル系列$V(= v_0, v_1, \ldots, v_{|V|-1})$を使用した故障シミュレーションを行いATPG検出効率を達成するまでに使用されたVの部分列$V'(= v_0, v_1, \ldots, v_{|V'|-1})$を求める．この関数は$V_c$に対し，より小さい$V'$を求めるために使用される．

関数**ADD1**(V)は，上記のベクトル系列Vに対してLFSRで生成した擬似ランダムベクトルを混在しベクトル系列V^*を生成するために使用される．この操作をパターン追加と呼び，以下の条件で行う．

(1) Vのベクトルv_i当りj個のベクトルを追加する．擬似ランダムベクトルを$j/2$個，v_iを$j/2$個追加する．V全体では$(j + 1) \times |V|$のパターンが出現する．

(2) 擬似ランダムベクトルとv_iを交互に追加する．

図1にパターン追加例($j = 8$)を示す．v_iに注目すると，$j/2 = 4$個の擬似ランダムベクトルv_rと$j/2 = 4$個のv_iを交互に追加し，$j = 8$個のベクトルより成るベクトル系列$(v_r, v_i, v_r, v_i, v_r, v_i, v_r, v_i)$を追加する．同様に$v_{i+1}$に対してベクトル系列$(v_r, v_{i+1}, v_r, v_{i+1}, v_r, v_{i+1}, v_r, v_{i+1})$, v_{i+2}に対して$(v_r, v_{i+2}, v_r, v_{i+2}, v_r, v_{i+2}, v_r, v_{i+2})$を追加する．この結果，ベクトル系列$V^*$のベクトル数は$9 \times j$個となる．

図2に，C17ベンチマーク回路に対するパターン生成例を示す．もとになるATPGベクトル

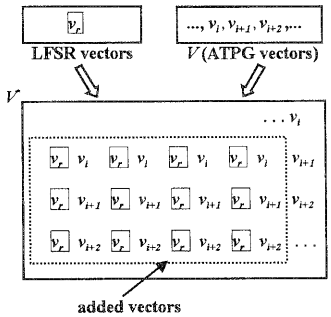


図1 関数ADD1(V)によるテストパターン生成例（$j = 8$）．

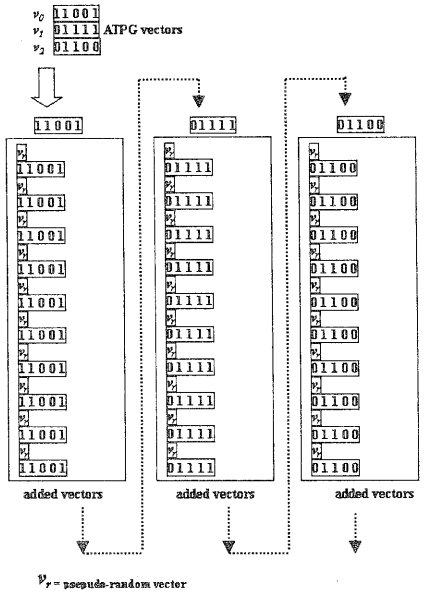


図2 関数ADD1(V)によるテストパターン生成例（C17, $j = 20$）．

(11001, 01111, 01100)それぞれに対して，$j = 20$個のパターン追加を行う．

関数**ADD2**(V, v_p)はベクトル系列Vよりベクトルv_pを取り除いた後，v_pの代わりにv_pに隣接していた2個のベクトルv_{p-1}, v_{p+1}に対して，ベクトル$(v_r, v_{p-1}, v_r, v_{p-1}, \ldots, v_{p-1})$と$(v_r, v_{p+1}, v_r, v_{p+1}, \ldots, v_{p+1})$を追加する．追加されるパターン長は$2 \times j$である．この関数はベクトル$v_p$が擬似ランダムベクトルで置き換えられる可能性を調べるために使用される．

関数**ADD^{-1}**(V)は関数**ADD1**(V)によって生成されたベクトル系列Vに対し，もとになるベク

トル系列を求める．例えば図 1 において

$$ADD^{-1}(V^*) = \ldots, v_i, v_{i+1}, v_{i+2}, \ldots$$

である．

3. 提案 TPG の構成とテストベクトル集合の発生

3.1 提案手法による TPG 構成

図 3 に提案手法による TPG 構成を示す．提案する TPG は ROM, LFSR, シフトレジスタ SR_A, SR_B, マルチプレクサ MUX により構成される．

ROM に格納されたベクトルはベクトル単位で交互に 2 個の SR にシフトインされる．シフトレジスタ SR_A, SR_B のベクトルは LFSR で生成される擬似ランダムベクトルと交互に CUT に印加される．

図 4 に提案 TPG によるテストパターン発生例を示す．CUT の入力数 $m = 2$, ベクトル v_0, v_1 を ROM に格納した場合を示す．$j = 8$ の場合を考える．ROM に格納したベクトル当り LFSR による擬似ランダムベクトル v_r が $j/2 (= 4)$ 回現れる．動作クロック周波数を基準とした場合，MUX の制御信号(SA, SB, SL)は 1/2 の周波数となる．SR, LFSR のクロック(ck_A, ck_B, ck_L)は 1/4 の周波数となる．

提案 TPG 手法では m ビット幅のテストベクトルをスキャンインする間に，j 回の追加パターンを発生するので，動作クロック周波数/シフトクロック周波数 $= j/m$ となる．

提案する TPG 手法は，実テストの面で以下の特長を有する．

(1) 短いテスト長で ATPG 検出効率の達成が可能である．

(2) スキャンインの周波数が動作クロックより低いため，低速テスタを使用した実動作速度テストが可能である．

3.2 ROM に格納するベクトル集合の選択

提案手法では，ROM に格納する ATPG ベクトルの部分列とパターン追加で付加されたベクトル系列により故障を検出する．従って ATPG 検出効率が保証される．ATPG ベクトル系列を

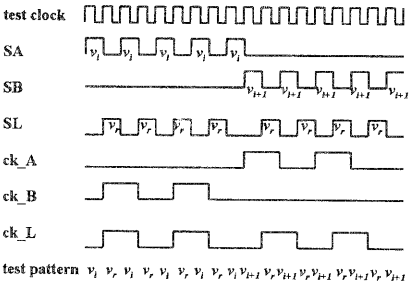
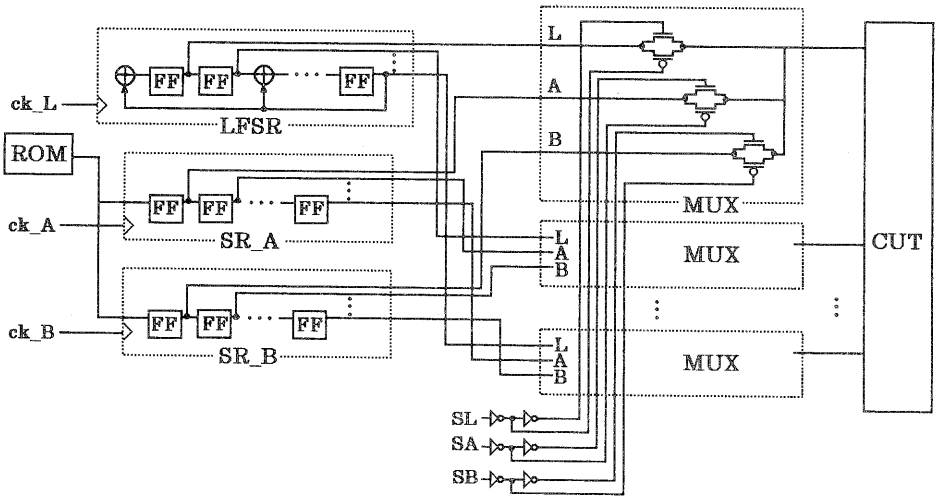


図 4 テストパターン発生例 ($j = 8$).



Ck_L: LFSR clock. Ck_A: SR_A shift-in clock. Ck_B: SR_B shift-in clock. SL, SA and SB: enable signals for inputs L, A and B of the MUX

図 3 提案する TPG 構成.

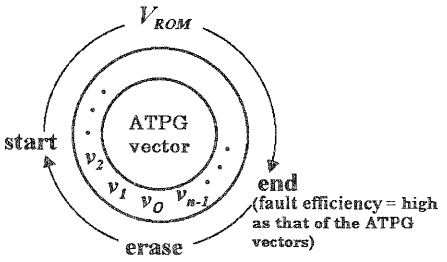


図 5 $O(n^2)$ 先頭位置選択法.

ROM に格納し ADD1 によってパターンを発生した場合，故障検出に関して必要でない ATPG ベクトルが存在する可能性がある．すなわち，これら必要ではない ATPG ベクトルを取り除き ROM 容量およびテスト長を減らすことが期待される．

以下に，ROM に格納するベクトル系列の選択および圧縮手順について述べる．

もとになる ATPG ベクトル系列 V_c に対して，ベクトルの並べ方の最適解は次のように求めることができる．V_c のすべての部分系列に対し，その順列を生成し，ADD1*(V)*によってテストベクトル系列を求める．ベクトルを並べ換えることによって ATPG 検出効率を達成できない並べ換えについては除外する．ATPG 検出効率を達成する並べ方のうち，最小ベクトル数である並べ方を選べば良い．V_c の部分集合は 2^n 個，それぞれの部分集合に対し，その順列は集合の大きさの階乗個存在するので，このアルゴリズムの複雑度は $O(2^n \times n!)$となる．n が数 10 以上に対し，現実的な時間で処理することは難しい．

そこで，より良い並べ方を現実的に求める手法として，$O(n^2)$先頭位置選択法と $O(n^2)$抜き取り法を提案する．

$O(n^2)$先頭位置選択法では，図 5 のように環状に配置した ATPG ベクトル系列より ROM に格納する先頭位置の選択を行う．ATPG ベクトルを環状に配置することにより，任意の先頭位置を選んだ場合に対する ATPG 検出効率は保証される．ATPG 検出効率を達成するのに使用されたベクトル系列(図中の V_{ROM})のみを ROM に格納する．すなわち ROM に格納するベクトル系列は圧縮される．

図 6 に $O(n^2)$先頭位置選択手順を示す．行番号は説明のために付した．

まず，CUT C に対して，ATPG ツールを使用し，ATPG 検出効率を達成するベクトル系列を求め V_c とする(行番号 1)．次に ROM に格納するベクトル系列 V_{ROM}, V, V_cを V_cに初期化する(行番号 2 から 4)．Vと Vcは一時的に使用するベクトル系列である．

ループ **for** 中では手順 6. から 12. を n 回繰り返す．関数 ADD1*(V)*により，ベクトル系列 V からパターン追加されたテストベクトル系列 V^*を生成する(行番号 6)．故障シミュレーションにより ATPG 検出効率の達成に必要なテストベクトル系列 V^*の部分列 V'を求める(行番号 7)．関数 $ADD^{-1}(V')$により，パターン追加によって V'を生成するもとの系列を求め，V とする(行番号 8)．もし $|V|$が$|V_{ROM}|$に対して圧縮される場合は，V_{ROM}を Vに更新する(行番号 9)．V_c' の最後に位置するベクトルを先頭に並び換え先頭位置を更新する(行番号 10 から 12)．上記の手順(行番号 5 から 12)で求めたベクトル系列 V_{ROM} の中でベクトル長が最小のものを ROM に格納するベクトル系列とする(行番号 13)．

回路 C を固定してアルゴリズムの複雑度を考える．ADD1*(V)*でのベクトルの生成操作および COVER*(C, V*)*での故障シミュレーションはそれぞれ*(j+1)* × n 回繰り返されるので複雑度は $O(n)$である．$ADD^{-1}(V')$はインデックスの計算で求められるのでその複雑度は $O(1)$である．行番号 5 から 12 は n 回繰り返されるので，この手順の複雑度は $O(n^2)$である．

次に，先頭位置選択法で求めたベクトル系列 V_{ROM} に対しさらに圧縮を行う $O(n^2)$抜き取り法を示す．

図 7 に $O(n^2)$抜き取り手順を示す．行番号は説明のため付した．

$O(n^2)$先頭位置選択法で求めたベクトル系列 V_{ROM} の各ベクトルについて以下の手順で不必要

```
reduction_r   O(n²) (C)
{
1.    V_c = ATPG(C);
2.    V_ROM = V_c;
3.    V = V_c;
4.    V_c' = V_c;
5.    for (i = 0; i < n; i++){
6.          V* = ADD1(V);
7.          V' = COVER(C, V');
8.          V = ADD⁻¹(V');
9.          if (|V| < |V_ROM|) V_ROM = V;
10.         V = V_c';
11.         V = v_n-1, v_0, v_1, ..., v_n-2 ;
12.         V_c' = V;
      }
13.   return V_ROM;
}
```

図 6 $O(n^2)$ 先頭位置選択手順.

```
reduction_p   O(n) (C, V_c, V_ROM)
{
1.    V_c' = V_c;
2.    V_ROM' = V_ROM ;
3.    for (i = 0; i < |V_ROM|; i++){
4.          V = V_ROM;
5.          v_p = v_i;
6.          V=ADD2(v_p);
7.          if (SIM(C,V) == 100%){
8.                V = V_ROM' - v_p;
9.                V = ADD1(V);
10.                 if (SIM(C,V) == 100%){
11.                 V_ROM' = V_ROM' - v_p;
                          }
            }
      }
12.   V_ROM = V_ROM';
13.   return V_ROM ;
}
```

図7 $O(n^2)$ 抜き取り手順.

であるものを求め V_{ROM} より抜き取る．一時的に使用するベクトル系列 V_c'と V_{ROM}'のそれぞれを V_c と V_{ROM} に初期化する（行番号 1, 2）．ここで，V_c は ATPG ツールで求めたベクトル系列，V_{ROM} は先頭位置選択法で圧縮した ROM に格納するベクトル系列である．V_{ROM} の i 番目のベクトルを評価対象ベクトル v_p とする(行番号 4, 5)．ADD2(V, v_p)により V_c より v_p を取り除き，(V, v_p)の代わりに(V, v_{p-1})と(V, v_{p+1})をもにする長さ $2 \times j$ のパターン追加を行いベクトル系列 V を生成する(行番号 6)．V が ATPG 検出効率を達成した場合は，ベクトル v_p が不必要であると仮定し，V_{ROM}'より v_p を一時的に取り除いたベクトル系列 V に対して ADD1(V)によるパターン追加を行う(行番号 9)．V が 100%の故障検出効率を達成した場合は，V_{ROM}'より v_p を取り除く(行番号 10, 11)．以上の手順(行番号 3 から 11)を繰り返し求めたベクトル系列 V_{ROM}'を ROM に格納するベクトル系列とする(行番号 12, 13)．

回路 C を固定してアルゴリズムの複雑度を考える．ADD2(V, v_p)は $2 \times j$ 回のパターン追加を行う．SIM(C,V)は，行番号 8 では $n + 2 \times j$ 回，行番号 11 では評価対象ベクトルが抜き取れる可能性がある場合のみ実行されるのでコンスタントである．行番号 3 から 11 は最大 n 回繰り返されるので，この手順の複雑度は $O(n^2)$である．

4. 実験および検討

ここでは，提案手法を ISCAS'85 およびフルスキャン ISCAS'89 ベンチマーク回路に対して適用した実験結果を示す．

4.1 実験方法

3 章で述べた構成の TPG について 100%の故障検出効率の達成に必要な ROM 容量とテスト長とを求めた．冗長故障は故障の対象から除外した．ROM に格納するベクトル系列は ATPG ツールによって生成した後，$O(n^2)$先頭位置選択法と $O(n^2)$抜き取り法により圧縮した．ATPG ツールは，縮退故障用 ATPG ツール[19]をトランジション故障用に拡張した．

ATPG ベクトル 1 個当りの追加パターン数 j に関し，$j = 4 \times m$ と $j = 8 \times m$ の場合をいくつかの CUT を用いて比較した（m は CUT の入力数）．

その結果，$j = 8 \times m$ の場合は $j = 4 \times m$ の場合と比べてテスト長が約 2 倍になるが格納するベクトルは圧縮されなかった．そこで，$j = 4 \times m$ として評価を行った．

提案している TPG のハードウェア量の評価に関し，我々は文献[7], [10]と同様に，TPG 回路をゲート換算し評価を行った．アドレスデコーダ等含む ROM 部分は量産デバイス情報をもとに 1 ゲート当り 22.6 ビットとした．

4.2 ATPG ベクトル系列の圧縮効果

$O(n^2)$先頭位置選択法と $O(n^2)$抜き取り法による ATPG ベクトル系列の圧縮結果を表 1 に示す．表中の“# test vectors”は 100%の故障検出効率を達成するために必要なテストベクトル数，“reduction(%)”は ATPG ベクトルで生成されたテストベクトル数を基準としたときの圧縮率を示す．red.1 は $O(n^2)$先頭位置選択法，red.2 は $O(n^2)$先頭位置選択法と $O(n^2)$抜き取り法の両方を適用した結果を示す．

例えば C1355 では，ATPG で生成された 100%の故障検出効率を達成するテストベクトル 203 個に対し，$O(n^2)$先頭位置選択法によりベクトル数は 164 個(80.8%)に圧縮される．$O(n^2)$先頭位置選択法に加えて $O(n^2)$抜き取り法を行った場合，ベクトル数は 126 個(62.1%)に圧縮される．

$O(n^2)$先頭位置選択法による ATPG ベクトル集合の圧縮率は平均で 83.3%，加えて $O(n^2)$抜き取り法を行うと 72.8%であった．BIST における ROM 容量および外部テスタを利用した場合の使用メモリ量を圧縮できることが分かる．

4.3 原始 LFSR による TPG との比較

表 2 に提案 TPG 方式を故障検出効率，テスト長，ハードウェアの面より原始 LFSR と比較した結果を示す．対象故障数は冗長故障を除いたトランジション故障数を示す．提案 TPG に関しては，100%の故障検出効率を達成するテスト長，ROM 容量(bit)，ハードウェア比率を示す．ハードウェア比率は原始 LFSR との比率（提案 TPG

のハードウェア量 / 原始 LFSR による TPG のハードウェア量）を示す．原始 LFSR 方式に関しては 90%の故障検出効率を達成した時点でのテスト長と故障検出効率を示す．評価した範囲で最も高い故障検出効率を達成したテスト長と故障検出効率は "maximum F.E. we computed" の欄に示す．

例えば C880 では，提案 TPG 方式はテスト長 7017 で 100%の故障検出効率を達成し，1740(= 29 × 60)ビットの ROM 容量を必要とする．ハードウェア比率は 3.2 である．これに対して原始 LFSR 方式では，テスト長 951 で故障検出効率は 90.1%，テスト長 262144 で故障検出効率は 94.5%である．

原始 LFSR 手法の TPG が 100%の故障検出効率を達成するのが困難であるのに対し，提案 TPG 方式では短いテスト長での達成が可能であることがわかる．

4.4 低速テスタによる実動作速度テスト

図 7 に提案 TPG と外部テスタを組合せたテスト手法を示す．ベクトル系列 V_{ROM} は外部テスタのメモリに格納され，スキャンクロック ck_A, ck_B は外部テスタで発生する．このシステムでは CUT の内部クロックよりも遅いスキャンクロックを使用することができる．すなわちトランジション故障に対して，CUT より低速な外部テスタを使用した実動作テストが可能である．また，外部テスタの代わりに FPGA と ROM（または不揮発性メモリ）を用いた安価なテスト回路を実現できる可能性がある．

また分割シフト法[12]を用いるとスキャンインのためのピンが増加するものの，テスト時間は大幅に短縮できる可能性がある．

5. むすび

本論文では，ATPG ベクトルを利用することにより短いテスト長でトランジション故障に対して ATPG 検出効率を達成する TPG を提案した．提案 TPG は 1 個の LFSR，2 個のシフトレジスタ，少量の ROM，マルチプレクサによって構成され，ROM には ATPG ベクトルの部分集合を格納する．ATPG ベクトルの部分集合と LFSR で発生した擬似ランダムベクトルを混在させたテストパターンを発生する．また，ROM に格納するベクトル系列を圧縮する $O(n^2)$先頭位置選択法と $O(n^2)$抜き取り法を示した．提案手法をベンチマークに適用した実験結果を原始 LFSR による TPG のそれと比較した．これらの結果は，提案 TPG 手法は原始 LFSR に比べ 3〜4 倍程度のハードウェア量が必要であるもののトランジション故障に対して短いテスト長で ATPG 検出効率を達成できることを示した．また提案手法ではスキャンインの周波数が実動作速度テストクロックより低くできるため，低速テスタを使用した実動作速度テストが可能である．従って提案手法は BIST 設計および実テストの面で有効な一手法であると言える．

表 1 ROM に格納するベクトルの圧縮．

CUT	fault efficiency = 100%				
	# test vectors			reduction(%)	
	ATPG	red.1	red.2	red.1	red.2
C880	43	34	29	79.1	67.4
C1355	203	164	126	80.8	62.1
C1908	219	198	170	90.4	77.6
C2640	115	96	88	83.5	76.5
C3540	233	215	185	92.3	79.4
C5315	97	67	51	69.1	52.6
C6288	58	15	15	25.9	25.9
C7552	147	142	131	96.6	89.1
S298	43	36	29	83.7	67.4
S349	34	22	18	64.7	52.9
S382	45	37	32	82.2	71.1
S400	46	39	30	84.8	65.2
S444	45	40	36	88.9	80.0
S526	86	72	67	83.7	77.9
S641	48	36	33	75.0	68.8
S820	180	162	151	90.0	83.9
S832	182	150	139	82.4	76.4
S953	141	122	110	86.5	78.0
S1196	221	213	195	96.4	88.3
S1238	227	226	213	99.6	93.8
S1423	58	50	44	86.2	75.9
S1488	206	206	169	100.0	82.0
S1494	210	199	171	94.8	81.4

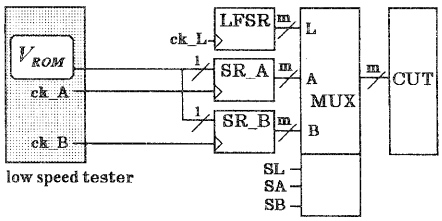


図 8 低速テスタを使用した実動作速度テスト．

文献

[1] S. Runyon, "Testing big chip becomes an internal affair," IEEE Spectrum, Vol. 36, No. 4, pp. 49-55, Apr. 1999.

[2] Y. Zorian, "Testing the monster chip," IEEE Spectrum, Vol. 36, No. 7, pp. 54-60, July 1999.

[3] Y. Zorian, E. J. Marinissen, and S. Dey, "Testing embedded-core-based system chips," IEEE Comput., Vol. 32, no. 6, pp. 52-60, June 1999.

[4] M. Lempel, S. K. Gupta, and M. A. Breuer, "Test embedding with discrete logarithms," VLSI Test Symp.,

表 2 原始 LFSR との比較.

CUT			proposed (fault efficiency = 100%)			LFSR			
						F. E. ≒90%		maximum F.E. we computed	
	#m	#faults	#test cycle	ROM (bit)	#area(proposed) / #area(LFSR)	#test cycle	fault efficiency(%)	#test cycle	fault efficiency(%)
C880	60	1277	7017	1740	3.2	951	90.1	262144	94.5
C1355	41	1980	20915	5166	3.6	721	90.0	131072	96.6
C1908	33	2477	22759	5610	3.9	2247	90.1	262144	97.3
C2670	233	3461	82105	20504	3.4	N. A.	N. A.	65536	85.6
C3540	50	4513	37369	9250	4.0	1266	90.0	131072	93.3
C5315	178	7324	26364	9078	3.2	191	90.0	65536	96.7
C6288	32	9580	1949	480	2.9	20	90.2	65536	99.8
C7552	207	10004	108600	27117	3.7	N. A.	N. A.	65536	96.1
S298	17	430	2029	493	3.0	N. A.	N. A.	131072	77.9
S349	24	456	1672	432	3.0	230	90.1	524288	91.4
S382	24	541	3042	768	3.0	778	90.2	524288	90.2
S400	24	568	2844	720	3.0	711	90.1	524288	90.5
S444	24	609	3434	864	3.1	N. A.	N. A.	524288	83.4
S526	24	797	6469	1608	3.3	N. A.	N. A.	524288	80.8
S641	54	624	7009	1782	3.1	1564	90.1	131072	93.9
S820	23	1307	14193	3473	3.8	N. A.	N. A.	524288	79.2
S832	23	1324	12982	3197	3.7	N. A.	N. A.	524288	78.6
S953	45	1413	19840	4950	3.5	N. A.	N. A.	131072	88.6
S1196	32	1705	25156	6240	4.1	N. A.	N. A.	131072	84.6
S1238	32	1796	27532	6816	4.2	N. A.	N. A.	131072	82.9
S1423	141	2000	25037	6204	3.1	N. A.	N. A.	262144	63.8
S1488	14	2199	9639	2366	3.9	—	—	16384	79.5
S1494	14	2218	9761	2394	3.9	—	—	16384	79.4

pp. 74-80, Apr. 1994.

[5] H. H. S. Gundlach and K. D. Muller-Glaser, "On automatic testpoint insertion in sequential circuits," Int'l Test Conf., pp. 1072-1079, Sept. 1990.

[6] M. Nakao, S. Kobayashi, K. Hatayama, K. Iijima, and S. Terada, "Low-overhead test point insertion for scan-based BIST," Int'l Test Conf., pp. 348-357, Sept. 1999.

[7] J. Savir, "On chip weighted random patterns," Asian Test Symp., pp. 344-352, Nov. 1997.

[8] M. F. AlShaibi and C. R. Kime, "MFBIST: A BIST method for random pattern resistant circuits," Int'l Test Conf., pp. 176-185, Oct. 1996.

[9] G. Kiefer and H. J. Wunderlich, "Deterministic BIST with multiple scan chains," Int'l Test Conf., pp. 1057-1064, Oct. 1998.

[10] C. Fagot, O. Gascuel, P. Girard, and C. Landrault, "A ring architecture strategy for BIST test pattern generation," Asian Test Symp., pp. 418-423, Dec. 1998.

[11] G. Kiefe and H.J. Wunderlich, "Using BIST control for pattern generation," Int'l Test Conf., pp. 347-355, Nov. 1997.

[12] S. Hellebrand, J. Rajski, S. Tarnick, S. Venkataraman, and B. Courtois, "Built-in test for circuits with scan based on reseeding of multiple-polynomial linear feedback shift registers," IEEE Trans. Comput., Vol. 44, No. 2, pp. 223-233, Feb. 1995.

[13] K. Chakrabarty, B. T. Murray, and V. Tyengar, "Built-in test pattern generation for high-performance circuits using twisted-ring counters," VLSI Test Symp., pp. 22-27, Apr. 1999.

[14] T. Asakawa and K. Iwasaki, "On using ATPG vectors for BIST TPG," The first IEEE Asia Pacific Conf. on ASICs, pp. 359-362, Aug. 1999.

[15] N. A. Touba and E. J. McCluskey, "RP-SYN : Synthesis of random pattern testable circuits with test point insertion," IEEE Trans. Comput., Vol. 18, No. 8, pp. 1202-1213, Aug. 1999.

[16] X. Li and P. Y. S. Cheung, "Exploiting BIST approach for two-pattern testing," Asian Test Symp., pp. 424-429, Dec. 1998.

[17] K. Fukuya, S. Yamazaki and M. Sato, "Evaluations of various TPG circuits for use in two-pattern testing," Asian Test Symp., pp. 242-247, Nov. 1994.

[18] A. Kristic, K. T. Cheng, and S. T. Chakradhar, "Testing high speed VLSI devices using slower testers," VLSI Test Symp., pp. 16-21, Apr. 1999.

[19] S. Kajihara, I. Pomeranz, K. Kinoshita, and S. M. Reddy, "On compacting test sets by addition and removal of test vectors," VLSI Test Symp., pp. 202-207, Apr. 1994.