ISSN 2186-5647

2-65

BAST におけるテストデータ量削減のための インバータブロック構成法

日大生産工(院)　○田中　まりか　日大生産工(院)　山崎紘史　日大生産工　細川　利典
九大　吉村　正義　日大生産工　新井　雅之　読売理工　中尾教伸

1. はじめに

近年，半導体技術の急速な進歩に伴い，大規模集積回路(Large Scale Integrated circuits:LSI)の規模が増大している．一般的にLSI の論理部のテスト生成にはスキャン設計と自動テスト生成ツール（Automatic Test Pattern Generator : ATPG）[1]によるテスト生成が従来から LSI 設計者に広く受け入れられてきた．このテスト手法によって高い故障検出効率を得られるが，回路の高集積化に伴い，ATPG により生成されたテストパターン数が増加しており，それに比例してテスト実行時間やテストデータ量が増加の問題が発生する．テスト実行時間はテストコストに比例し，テストデータ量はテスタのメモリ容量内であることが望ましいため，テストデータ量を削減することが重要である．したがって，テストデータ圧縮技術の重要性が高まっている．

テストデータ量を削減するために用いられる主なテストデータ圧縮技術として，EDT(Embedded Deterministic)[2],XD-BIST(X-Tolerant Deterministic BIST)[3],BAST(BIST Aided Scan)[4]などが提案されている．本論文では，BAST 技術に着目し，テストデータ量の削減を目指す．BAST とは，組込み自己テスト（Built-In - Self - Test : BIST）[4,5,6,7,8,9]と ATPG を組み合せて，高い故障検出効率を維持しながら，テストデータ量を圧縮する技術である．BAST アーキテクチャにおいて，疑似ランダムパターン生成器 PRPG(Pseudo Random Pattern Generator)[10,11,12,13]から生成される疑似ランダムパターン中のあるビットを反転させ，ATPG で生成されるパターン(決定的パターン)に変換する．また反転されたビット位置を記憶するため，BAST コードと呼ばれるコードがテスタ(Automatic Test Equipment : ATE)にメモリに記憶される．その BAST コードにしたがって，PRPG で生成された疑似ランダムパターンの一部を反転させ，スキャンチェインを通じてテスト対象回路に入力される．BAST コードにおいてスキャンチェインをシフトさせる情報量は決定的パターン数によって一意に決定されるが，ビットを反転させる反転命令情報量は，従来，疑似ランダムパターンを反転させるビット数(反転ビット数)に依存するとされており，反転ビット数を削減する手法が考案されてきた．

BAST において，反転ビット数は，ATPG によって生成された決定的パターン集合，決定的パターンと疑似ランダムのマッチングによって決定する．文献[14]のドントケア抽出法[17]では，少なくともすべての故障を 1 回検出するために不必要なビット(ドントケアビット)を決定的パターン集合から特定する．ドントケアビットは 0 でも 1 でもどちらでもよいため，反転ビット数を削減することができる．マッチング法に関しては，文献[15]では決定的パターン集合と疑似ランダムパターン集合の先頭から 1 対 1 にマッチングを行う単純マッチング法が採用されており，文献[14]では総反転ビット数が少数になることを目的としたヒューリスティックアルゴリズムが提案され，さらに文献[16]では，このマッチングを最小コスト最大マッチング問題として定式化し，その問題の厳密解法として知られているハンガリアルゴリズム[18]を採用している．文献[16]において，文献[14]のヒューリスティックアルゴリズムと比較してハンガリアルゴリズムによる反転ビット数の削減率は小さいことが報告されている．

反転ビット数や反転してはいけないビット数(非反転ビット数)は各パターンまたは各スキャンチェインごとに異なる．本論文では，反転命令情報量を削減することによってテストデータ量を削減するためにこの性質に着目する．反転命令情報量を，反転ビット数のみに依存するのではなく，各スキャンチェインごとに反転ビット数と非反転ビット数を比較し，少ない方のビットに依存させることによって，反転命令情報量を削減することを目指す．そのために，BAST アーキテクチャのインバータブロックの構成を変更し，反転命令数削減志向インバータブロックを提案する．

本提案手法を，文献[16]で提案されたマッチング法とドントケア抽出法[17]と組合せ，BAST コード量の削減を図り，ISCAS'89，ITC'99 ベンチマーク回路に適用し，BAST コード量を評価する．

2. BAST の基本概念とアーキテクチャ

BAST アーキテクチャを図 1 に示す．BAST は，疑似ランダムパターン発生器 PRPG[10,11,12,13]とテスト応答圧縮器 MISR(Multiple Input Signature Register)，特定のパターン中のビットを反転させるインバータブロックと不定状態をマスクする U マスクブロックから構成される．

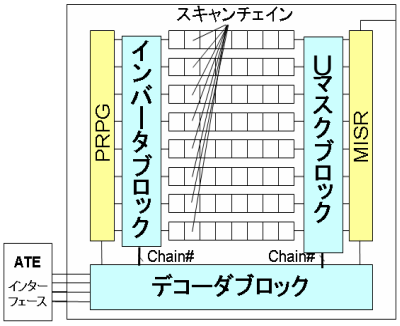

図 1.　BAST 基本概念図

本提案手法では，インバータブロックのアーキテクチャに着目するため，インバータブロックについては，2.1 節で詳細に説明する． デコーダブロックは,コード化された信号を受け取るインタフェースチャネルに接続され,インバータブロック中の対応する FF を制御する．インタフェースチャネルに入力されるコード化されたテストパターンを BAST コードと呼ぶ．

2.1 インバータブロック

図 2 にインバータブロックの詳細を示す．インバータブロックは，PRPG からスキャンチェンインへ入力する信号を ATE に保存している BAST コードにしたがって反転させる．インバータブロックの出力はスキャンチェンイン数と同一のビット幅を持つ．その論理は XOR ゲートから構成され，反転を制御するフリップフロップ(flip flop : FF)の信号にしたがい PRPG の出力信号を反転させる．

An inverter block construction method to reduce test data volume on BAST

Marika TANAKA, Hiroshi YAMAZAKI, Toshinori HOSOKAWA ,
Masayoshi YOSHIMURA, Masayuki ARAI, Michinobu NAKAO

FF の Set が 1 かつ Reset が 0 の時は，PRPG の出力信号を反転し，FF の Set が 0 かつ Reset が 1 の時は PRPG の出力信号がそのまま，スキャンチェインに印加される．

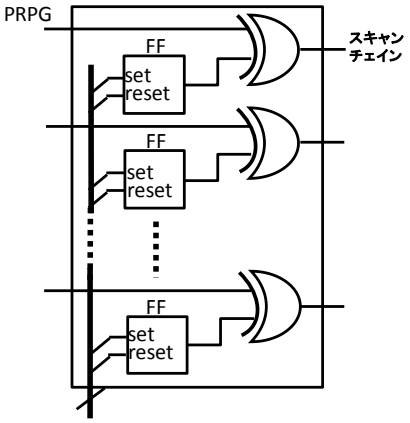

図 2.インバータブロック

2.2 BAST コード

BAST コードはインターフェースチャンネルに入力されるコード化されたテストパターンであり，デコーダブロックでデコードされる．BAST コードは LFSR を用いた PRPG によるランダムパターンと ATPG パターンをマッピングしたデータより生成される．BAST コード例を表 1 に示す．ビット反転命令･U マスク命令ともに，1 で動作，0 で動作せずにシフトするものとする．

表 1. BAST コードの例

TimeFrame	1	2	3
Invert flag	1100	0	10
U-Mask flag	0010	0	10
Chain Address2	1000	0	10
Chain Address1	0110	0	00
Chain Address0	0100	0	10

チェインアドレスは，反転・U マスク命令がどのスキャンチェインに適応されるかを示している．タイムフレームは，スキャンチェインの一列(スキャンスライス)の処理が行う時刻を表わしている．2 進数で表現したものである．表 2 のタイムフレーム 1 では，左から 1 列目の命令は，ビット反転命令をチェインアドレス’100’（スキャンチェイン 4）で行う．タイムフレーム 1 の左から 2 列目ではビット反転の命令をチェインアドレス’011’（スキャンチェイン 3)で行う．タイムフレーム左から 3 列目では U マスク命令をチェインアドレス’010’（スキャンチェイン 2)の出力で行う．タイムフレームの左から 4 列目では，何の命令もない．何も命令のない状態ではチェインアドレスは’000’とする．この 4 列目の状態では，次のタイムフレームへのシフトの命令とする．

3. BAST におけるテストパターン生成

3.1. BAST パターン

BAST におけるテストパターン生成について述べる．ATPG によりテスト対象回路(CUT)に対する決定的パターン集合 TD を生成し，同時に LSI 内部に埋め込まれた PRPG が生成するものと同一の疑似ランダムパターン集合 TR を作成する．

表 2 に BAST パターン生成規則を示す．(tr_i,bit_k)は，ある PRPG パターン(tr_i)のあるビット(bit_k)を指し，(td_i,bit_k)は，ある ATPG パターン(td_i)のあるビット(bit_k)を指す．関数 B(p,bit_k)では，あるパターン(p)の，あるビット(bit_k)の値を返す．この規則は式(1)に示すことができる．疑似ランダムパターンの値と決定的パターンの値が一致する場合，疑似ランダムパターンの値を保持する．また，疑似ランダムパターンの値と決定的パターンの値が衝突する場合，疑似ランダムパターンの値を反転する．決定的パターンの値がドントケア(X)である場合，疑似ランダムパターンの値を保持する．表 2 に示す生成規則$\cap_B$を用いると，BAST パターンの k 番目の値(tb_k)は式(2)で示すことができる．

表 2. BAST パターン生成規則$\cap_B$

$B(tr_i, bit)$ \ $B(td_i, bit)$	0	1
0	0	0
1	1	1
X	0	1

$$\mathrm{B}(\mathrm{p}, bit_k) = \begin{cases} 0: \textit{あるパターン}p\textit{の}k\textit{番目のビットが}0 \\ 1: \textit{あるパターン}p\textit{の}k\textit{番目のビットが}1 \\ X: \textit{あるパターン}p\textit{の}k\textit{番目のビットが}X \end{cases}$$

･･･式(1)

$$tb_k = \mathrm{B}(tr_i, bit_k) \cap B(td_i, bit_k) \cdots \text{式(2)}$$

3.2 BAST パターン生成例

図 3 は，スキャンチェイン 4 本，スキャンチェイン長 3 での，決定的パターンと疑似ランダムパターンのマッチング結果の例を表す．式(2)より生成されたテストパターンが，BAST パターンとなる．スキャンチェイン 1 では，時刻 1 でビット反転を行い，時刻 2 と時刻 3 ではビット反転を行わない．このとき，スキャンチェイン 1 の反転ビット数は 1 となり，非反転ビット数はビット反転を必要としないケアビット数なので 1 となる．スキャンチェイン 0 では反転ビット数は 0，非反転ビット数は 1 となっているが，スキャンチェイン 2 では反転ビット数は 3，非反転ビット数は 0 となっている．このように，反転ビット数の方が非反転ビット数より多いスキャンチェインや，非反転ビット数の方が反転ビット数より多いスキャンチェインが存在する．

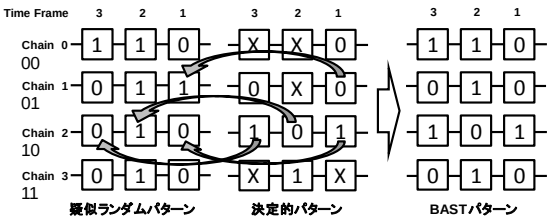

図 3.BAST パターン生成例

4. 反転ビット数と非反転ビット数

本章では，ISCAS’89 ベンチマーク回路と ITC’99 ベンチマーク回路において，決定的パターンを生成し，全故障検出ドントケア抽出を行い，文献[16]で提案されたマッチング法を用いてマッチングを行った場合，各 BAST パターンごとに，反転ビット数が非反転ビット数より多いスキャンチェインや，非反転ビット数が反転ビット数より多いスキャンチェインが，どの程度存在するか実験を行った．表 3 に実験結果を示す．この結果から，反転ビット数が非反転ビット数より多いスキャンチェインが約 2 割

程度存在することがわかる．本論文では，反転ビット数が非反転ビット数より多いスキャンチェインに着目し，反転命令回数の削減を目指す．

表 3. 反転ビット数・非反転ビット数が多いスキャンチェインの割合

回路名	SC8本			SC16本		
	反転＞非反転	反転＜非反転	反転＝非反転	反転＞非反転	反転＜非反転	反転＝非反転
s13207	13.4%	64.8%	21.8%	14.2%	46.5%	39.3%
s15850	83.9%	14.1%	2.0%	20.9%	52.9%	26.1%
s35932	30.1%	63.6%	6.3%	30.4%	64.2%	5.4%
s38417	20.5%	72.9%	6.6%	25.4%	61.0%	13.6%
s38584	20.1%	72.7%	7.2%	25.7%	62.8%	11.5%
b14	13.7%	50.5%	35.9%	8.9%	36.9%	54.1%
b15	8.5%	46.9%	44.7%	7.8%	34.5%	57.7%
b17	10.0%	55.5%	34.5%	8.5%	38.3%	53.2%
b20	16.6%	60.6%	22.8%	34.5%	40.7%	24.9%
b21	17.3%	64.3%	18.3%	37.0%	40.3%	22.7%
b22	17.8%	68.7%	13.5%	16.0%	52.7%	31.3%
Average	22.9%	57.7%	19.4%	20.9%	48.3%	30.9%

5. 反転命令数削減指向インバータブロックによるビット反転/ビット非反転制御

5.1. 反転命令数削減指向インバータブロック

図 4 に，反転命令数削減指向インバータブロックを示す．従来のインバータブロックに，XOR ゲートと FF を 1 つずつ追加する．FF_Bは，反転ビット指定 FF となっており，FF_Pは，反転パターン指定 FF である．これら 2 つの FF の XOR 演算の結果が 1 の場合は PRPG の出力信号を反転し，XOR 演算の結果が 0 の場合は PRPG の出力信号がそのまま，スキャンチェインに印加される．

パターン反転機能は，各スキャンチェインごとに，反転ビット数が非反転ビット数より 2 以上多い場合に，PRPG の出力信号を反転させてスキャンチェインに印加するときに用いる．反転パターン指定 FF では，パターン反転機能を用いない場合は，Set を 0，Reset を 1 とし，パターン反転機能を用いる場合は，Set を 1，Reset を 0 とする．

反転ビット指定 FF では，反転パターン指定 FF の Set が 0，Reset が 1 のとき，ビット反転を行う場合は Set を 1，Reset を 0 とする．ビット反転を行わない場合は，Set を 0，Reset を 1 とする．また，反転パターン指定 FF の Set が 1，Reset が 0 の場合にビット反転を行う場合は Set を 0，Reset を 1 とする．ビット反転を行わない場合は Set を 1 とし，Reset を 0 とする．

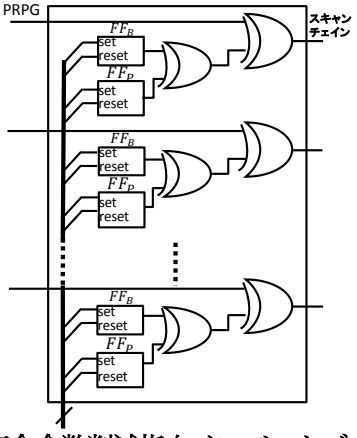

図 4.反転命令数削減指向インバータブロック

このように設定することで，反転パターン指定 FF の set が 1，reset が 0 になっていると，ビット反転を行う場合，反転命令を出さずにビット反転を行うことが可能となる．反転パターン指定 FF の値を各スキャンチェインごとに切り替えることによって，BAST コードとなる反転命令の数が，各スキャンチェインごとの反転ビット数と非反転ビット数の少ないビット数の合計となる．そのため，反転命令数が削減され，BAST コード量が削減される．

5.2 反転命令数削減指向インバータブロック用 BAST コード

反転命令数削減指向インバータブロックを用いるにあたって，BAST コードも 2.2 節で述べた BAST コードを変更させる必要がある．表 4 に提案手法用 BAST コードの例を示す．本手法を実現するにあたって，初めのタイムフレームで，パターン反転を行うスキャンチェインアドレスを指定する．従来，ビット反転命令フラグ，U マスク命令フラグ共に 1 指定する場合，そのスキャンチェインに対してビット反転命令と U マスク命令の両方を行っていたが，本手法では，パターン反転を行うスキャンチェインアドレスを指定する．表 4 のタイムフレーム 1 では，左から 1 列目の命令は，パターン反転命令をチェインアドレス’100’（スキャンチェイン 4）で行い，左から 2 列目の命令は，パターン反転命令をチェインアドレス’011’（スキャンチェイン 3）で行う．その他の命令は，2.2 節で述べた命令と同じ働きをする．

表 4. 提案手法 BAST コード

TimeFrame	1	2	3	4
Invert flag	11	010	0	10
U-Mask flag	11	100	0	10
Chain Address2	10	010	0	10
Chain Address1	01	100	0	00
Chain Address0	01	010	0	10

5.3 反転パターン指定 FF の値決定例

表 5 に示した反転パターン指定 FF の値決定例と図 3 の例を用いて説明する．疑似ランダムパターンと決定的パターンのマッチングを行い，反転ビット数と非反転ビット数を求める．このとき，スキャンチェインアドレス 10 のみ，反転ビット数>(非反転ビット数+1)となっているため，パターン反転機能を用いる．そのため，ビット反転命令フラグと U マスク命令フラグを 1 に設定することによって，反転パターン指定 FF の set を 1，reset を 0 に設定する．このように設定することによって，疑似ランダムパターンの値は反転してスキャンチェインに印加される．よって，反転命令を出さずにビット反転を行えるので，反転ビット数を変えずに反転命令数を削減することができる．

表 5.反転パターン指定 FF の値決定例

アドレス	反転ビット数	非反転ビット数	反転パターン指定FF	
			set	reset
00	0	1	0	1
01	1	1	0	1
10	3	0	1	0
11	0	1	0	1

この例の場合，提案手法を用いない場合，反転命令数は各スキャンチェインの反転ビット数の合計となるので 4 となるが，本提案手法を用いた場合の反転命令数は各スキャンチェインの反転ビットと非反転ビットを比較し，少ない方のビット数の合計となるため，1 となる．

表 6.ビット反転命令数とパターン反転命令数

回路名	従来法				提案手法											
	SC8本	SC16本	SC32本	SC64本	SC8本			SC16本			SC32本			SC64本		
	反転命令数	反転命令数	反転命令数	反転命令数	ビット反転指定回数	パターン反転指定回数	合計	ビット反転指定回数	パターン反転指定回数	合計	ビット反転指定回数	パターン反転指定回数	合計	ビット反転指定回数	パターン反転指定回数	合計
s13207	5,440	5,475	5,508	7,024	4,638	222	4,860	4,536	245	4,781	4,002	671	4,673	4,091	693	4,784
s15850	6,635	6,508	6,515	8,096	6,041	127	6,168	5,595	133	5,728	4,994	606	5,600	4,835	753	5,588
s35932	7,790	7,739	7,775	7,993	7,304	22	7,326	7,100	22	7,122	6,840	22	6,862	6,200	416	6,616
s38417	16,869	16,870	16,895	18,832	15,504	137	15,641	15,057	141	15,198	14,668	141	14,809	12,993	1,435	14,428
s38584	16,380	16,143	16,161	18,359	14,886	148	15,034	14,317	152	14,469	13,877	152	14,029	12,188	1,557	13,745
b14	24,515	22,778	22,938	30,660	21,660	611	22,271	18,869	1,490	20,359	17,527	2,196	19,723	15,971	3,162	19,133
b15	13,253	13,003	12,946	17,305	11,986	330	12,316	11,077	774	11,851	10,466	1,054	11,520	9,795	1,437	11,232
b17	58,810	56,971	57,214	77,305	54,221	962	55,183	51,898	982	52,880	50,882	1,070	51,952	47,153	3,906	51,059
b20	48,448	45,855	45,919	55,679	43,942	771	44,713	40,832	800	41,632	37,380	3,223	40,603	34,811	4,662	39,473
b21	53,239	50,368	50,611	60,847	48,220	795	49,015	44,919	821	45,740	41,164	3,649	44,813	38,122	5,175	43,297
b22	63,537	60,551	60,560	76,575	58,124	711	58,835	54,729	730	55,459	50,546	3,514	54,060	47,629	4,988	52,617

6.実験結果

本章では，ISCAS'89 ベンチマーク回路と ITC'99 ベンチマーク回路において，決定的パターンを生成し，全故障検出ドントケア抽出を行った後に，本論文で提案した反転命令数削減指向インバータブロックを用いてマッチングを行い，その結果をまとめた．また，反転命令数削減指向インバータブロックを用いた場合の回路面積オーバーヘッドの解析を行った．決定的パターン集合は，Synopsys 社の TetraMAX で生成されたものを用いた．

表 6 では，従来法(全故障検出ドントケア抽出を行った後に文献[16]で提案されたマッチング法を用いた場合)での反転命令数と，本手法を用いた場合のビット反転指定回数とパターン反転指定回数とその合計(反転命令数)を示している．表 7 では，従来法と比較した場合の BAST コード量の削減率を示している．表 8 では，従来法と本手法で，BAST 回路を付加していない場合に対し，どの程度面積オーバーヘッドが増えるかを示している．少ない面積オーバーヘッドで，BAST コード量削減に成功した．

表 7.BAST コード量削減率

回路名	SC8本	SC16本	SC32本	SC64本
s13207	2.08%	4.15%	7.41%	22.32%
s15850	2.75%	6.64%	10.01%	26.47%
s35932	3.67%	6.06%	10.14%	15.99%
s38417	2.80%	5.50%	8.82%	19.82%
s38584	3.05%	5.56%	9.21%	21.11%
b14	4.24%	6.47%	10.63%	33.20%
b15	2.20%	4.12%	6.90%	28.70%
b17	1.37%	2.55%	4.81%	25.39%
b20	3.55%	5.69%	8.78%	25.55%
b21	3.82%	5.85%	8.87%	25.55%
b22	3.47%	5.27%	8.27%	27.99%

表 8.面積増加量

回路名	従来BAST				提案手法BAST			
	SC8本	SC16本	SC32本	SC64本	SC8本	SC16本	SC32本	SC64本
s13207	1.77%	3.48%	6.88%	13.67%	2.54%	5.01%	9.96%	19.83%
s15850	1.57%	3.08%	6.09%	12.10%	2.25%	4.44%	8.81%	17.55%
s35932	0.77%	1.52%	3.00%	5.97%	1.11%	2.19%	4.35%	8.66%
s38417	0.67%	1.31%	2.59%	5.15%	0.96%	1.89%	3.75%	7.47%
s38584	0.67%	1.31%	2.60%	5.16%	0.96%	1.89%	3.76%	7.48%
b14	1.93%	3.79%	7.51%	14.93%	2.77%	5.47%	10.87%	21.65%
b15	1.93%	3.78%	7.48%	14.86%	2.76%	5.45%	10.82%	21.55%
b17	0.55%	1.08%	2.14%	4.24%	0.79%	1.56%	3.09%	6.15%
b20	0.97%	1.91%	3.79%	7.52%	1.40%	2.76%	5.48%	10.91%
b21	0.95%	1.86%	3.69%	7.32%	1.36%	2.68%	5.33%	10.62%
b22	0.65%	1.28%	2.54%	5.05%	0.94%	1.85%	3.68%	7.32%

7.おわりに

本論文では，テストデータ量を削減するために，BAST アーキテクチャにおけるインバータブロック構成法を用いた疑似ランダムパターンのビット反転数削減手法の提案とその評価を行った．実験結果から，本提案手法を適用すると，反転命令数を削減することができ，また，スキャンチェイン本数を増やすほど高い効果が得られることがわかった．

今後の課題として，反転命令数だけではなく，シフト命令数を削減する手法の提案とテストデータ量の削減率の評価が挙げられる．

参考文献

[1] H. Fujiwara, Logic Testing and Design for Testability, The MIT Press, 1985.

[2] J. Rajski, J. Tyszer, M. Kassab, N. Mukherjee, R. Thompson, K.H. Tsai, A. Hertwig, N. Tamarapalli, G. Mrugalski, G. Eide, and J. Qian, "Embedded Deterministic Test for Low Test Manufacturing Test," in Proc. ITC, pp. 301-310, 2002.

[3] P. Wohl, J.A. Waicukauski, S. Patel, and M.B Amin, "X-tolerant compression and application of Scan-ATPG Patterns in a BIST Architecture," in Proc. ITC, pp. 727-736, 2003.

[4] T. Hiraide, K.O. Boateng, H. Konishi, K. Itaya, M. Emori, H.Yamanaka, and T. Mochiyama, "BIST-aided scan test-A new method for test cost reduction," Proc. VTS, pp. 359-364, 2003..

[5] P. H. Bardell, W. H. McAanney and J. Savir, Built-In Pseudo-Random Testing of Digital Circuits, Chapter 8. John Wiley & Sons, New York, 1987.

[6] R. Chandramouli, S. Pateras, "Testing Systems on a Chip," IEEE Spectrum, Vol. 33, No. 11, pp. 42-47, November 1996.

[7] Y. Zorian, E. J. Marinissen, S. Dey, "Testing Embedded Core-based System chips," in Proc. IEEE International Test Conference, Washington, DC, pp. 130-143, 1998.

[8] J. Aerts, E. J. Marinissen, "Scan Chain Design for Test Time Reduction in Core-based ICs" in Proc. IEEE International Test Conference, Washington, DC, pp. 448-457, 1998.

[9] B. Nadeau-Dostie, Design for At-Speed Test, Diagnosis and Measurement, Boston, Dordrecht, Kluwer Academic Publishers, London, 2000.

[10] H. Bardell, W. H. McAnney, "Parallel Pseudo-random Sequences for Built-in Test," in Proc. IEEE International Test Conference (ITC), 1984, pp. 302-308, 1984.

[11] H. Bardell, W. H. McAnney, "Pseudo-Random Arrays for Built-In Tests," IEEE Transactions on Computers, Vol. C-35, No. 7, 1986, pp. 653-658.

[12] H. Bardell, W. H. McAnney, J. Savir, Built-In Test for VLSI, Wiley-Intersience, New York, 1987.

[13] G.. Kiefer, H.-J. Wunderlich, "Using BIST Control for Pattern Generation," in Proc. IEEE Int. Test Conf. Washington, DC, pp. 347-355, November 1997.

[14] M. Arai, S. Fukumoto, K. Iwasaki and T. Hiraide, "Test Data Compression using TPG Reconstruction for BIST-aided Scan Test," in Digest of Papers of the IEEE 6th Workshop on RTL and High Level Testing, pp. 12-17, 2005.

[15] Masayuki Arai, Satoshi Fukumoto and Kazuhiko Iwasaki, "Test Data Compression of 100x for Scan-Based BIST," IEEE INTERNATION TEST CONFERENCE 2006

[16] LingLing Wan, Motohiro Wakazono, Toshinori Hosokawa, and Masayoshi Yoshimura, A Bit flipping Reduction Method for Pseudo-random Patterns Using Don't Care Identification on BAST Architecture, 9th Workshop on RTL and High Level Testing, 2008.11.

[17] K. Miyase and Seiji Kajihara, "XID: Don't Care Identification of Test Patterns Circuits and Systems," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 23, No. 2, pp. 321-326, 2004.

[18] A. Dolan, R. J. Wilson and J. Aldous, Networks and Algorithm: An Introductory Approach, John Wiley & Sons, 1994.