社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2003－ＳＬＤＭ－112 （９）
2003／11／28

ランレングス符号とピンポイントテストパターン変換によるテストデータ量削減

梶原 誠司† 土井 康稔† Lei Li‡ Krishnendu Chakrabarty‡

†九州工業大学 〒820-8502 福岡県飯塚市川津 680-4
‡デューク大学 Durham, NC 27708, U.S.A.

E-mail: †kajihara@cse.kyutech.ac.jp, †doi@aries30.cse.kyutech.ac.jp, ‡{ll, krish}@ee.duke.edu

あらまし 本論文では，ランレングス符号化によるテストデータ量削減効率の向上手法を提案する．提案手法は，与えられたテストパターンに対して，故障検出率を低下させることなく値を1から0に変換できるビットを特定し，そのビット値を変換することで，長く連続する0を多く持つテストパターンに変換するものである．値を変換可能なビットはドントケア判定により求めるが，このとき変換するビットによって削減できるテストデータ量が異なることに注目する．論理値が1の各ビットについて，値を0に変換したときのランレングス符号化によるテストデータ量削減ビット数を前もって計算することにより，データ量削減効果の高いビットをピンポイントに指定する．そのビットが優先的にドントケアとなるようにテストパターンを変換し，テストデータ量削減に最適なテストパターンを求めることができる．ベンチマーク回路に対する実験では，提案手法によりテストデータ量を平均で55%削減できた．

キーワード ランレングス符号，テストパターン変換，ドントケア

On Combining Pinpoint Test Set Relaxation and Run-Length Codes for Reducing Test Data Volume

Seiji KAJIHARA† Yasumi DOI† Lei LI‡ and Krishnendu CHAKRABARTY‡

†Kyushu Institute of Technology 680-4 Kawazu, Iizuka, Fukuoka, 820-8502 Japan
‡Duke University Durham, NC 27708, U.S.A.

E-mail: †kajihara@cse.kyutech.ac.jp, †doi@aries30.cse.kyutech.ac.jp, ‡{ll, krish}@ee.duke.edu

Abstract This paper presents a pinpoint test set relaxation method for test compression that maximally derives the capability of a run-length encoding technique. Before encoding a given set of test patterns, we selectively relax some specified bits of the test patterns. By changing a specified bit with value 1 to a don't-care, two consecutive runs of 0s in the test sequence can be concatenated into a longer run of 0, thereby facilitating run-length coding. This procedure retains the fault coverage of the test set. Since the increase in compression depends on the lengths of the two runs that are concatenated with each bit relaxation, by pre-computing the increase in compression, we pinpoint the bit positions with value 1, which when relaxed to don't-cares, will yield the most compression. In this way, the given test pattern set is appropriately modified as a preprocessing step before test compression. Experimental results for the ISCAS benchmark circuits show that the proposed method could reduce, on the average, test data volume by 55%.

Keyword run-length codes, test modification, don't-care

1. はじめに

近年のSoC(System-On-Chip)テストの複雑化により，SoCのテストに要するコストの割合が増加している．テストコストの削減を考える際に重要な要素となるのは，テストデータ量の削減とテスト印加時間の削減である．中でもテストデータ量の削減は直接的にテスト印加時間を削減するため，多くの手法が研究されている[1-15]．その中の一つに，高い故障検出率のテスト集合をできるだけ少ないベクトル数で実現するテストコンパクションがある[1,2,3]．フルスキャン順序回路

において、テスト集合が最小となるようなテスト生成技術が開発されてきた[2,3]. しかし，テストデータ量はテストベクトル数だけでなくスキャンフリップフロップ数にも比例しているため，フリップフロップ数が多くなるとテストコンパクション技術だけではテストデータ量の削減は不十分である.

そこで、テストコンプレッション技術を用いたテストデータ量削減手法が数多く提案されている．これらの手法は，テスト集合が持つ情報の一部をチップ上にハードウェア化することによってLSIテスタに保存すべきデータ量を削減する．テストコンプレッション技術は，チップ上に組み込まれるハードウェアの種類によって分類される．例えば，チップ上にLFSRに基づく回路が組み込まれる場合，reseedingなどは効果的なテストコンプレッション手法となる[4,5,6]. また，ランレングス符号化のような統計符号を用いたテストデータ量削減手法もある[7-11]. これらの手法においてチップ上に組み込まれるハードウェアは順序回路であるが，デコンプレッサとして組合せ回路を用いるテストコンプレッション技術も提案されている[12,13].

テストコンプレッションでは，高い圧縮率を得るためにテストパターン中の未設定値が使われる．一方，テスト生成において未設定値を残す場合，逆順故障シミュレーション[16]や二重検出法[2]によるテストコンパクションでは未設定値が残るテストパターンでは効果が得られにくく，テスト集合を小さくすることができない．一般的に高い圧縮率は大きなテスト集合から得られるが，テストコンパクション技術によって圧縮されたテスト集合をテストコンプレッションによってさらに圧縮すると，結果的にテストデータ量をより削減することができる.

テストパターン中のすべてのビットに0か1が割り当てられている圧縮されたテスト集合が生成されたとき，このテストパターン中の値には，逆の論理値が割り当てられても故障検出率に影響を与えないものがある．そのような入力値はドントケア（X）として扱うことができる．テスト集合中のドントケア判定手法は[17]で提案されている．一般的にドントケアの組合せは多く存在するが，この手法ではできるだけ多くのXを含む組合せを見つけることができる.ISCASベンチマーク回路に対する実験では，高度な圧縮技術を用いて生成したテスト集合中にでさえ，およそ半分の入力値がXであることが報告されている．Xは，未設定値のように扱うことができるので，テストコンプレッションに利用することができる.

本論文では，Golomb 符号化や FDR 符号化のようなランレングス符号化でテストパターンを圧縮するとき，その効果を最大限に引き出すようなテストパターン変換手法を提案する．与えられたテストパターンを符号化する前に，テスト集合の故障検出率を下げることなく，[18]のドントケア判定手法を用いてテストパターン中のいくつかのビットの論理値を反転する．このとき論理値を反転するビットを，2つの連続する0が1つの長く連続する0に連結するように選択する．この連結による利得（符号化により削減されるビット数）は，前後に連続する0の数により決まるため，それぞれビットごとに異なる．本手法では，0を連結することによって削減するビット数を示す利得表をあらかじめ求め，この表を参照しながら論理値を変換するビットの候補を決定する．この利得表によって，利得の大きいビットを特定し，そのビット上にドントケア判定手法を用いてXを見つける．このような処理をテストパターンに対して繰り返すことによってテストデータ量削減効率を上げる．ISCAS ベンチマーク回路に対する実験では，提案手法の有効性を示す.

本論文は次のように構成される．2章では，提案手法に用いられる技術について説明する．3章では提案手法を説明する．4 章で実験結果を示した後，最後に5章で本論文をまとめる.

2. 準備

2.1 ランレングスに基づく符号化

テストパターンの符号化手法として,FDR (Frequency Directed Run-length)符号化や Golomb 符号化がある．これらの手法は，テストパターン中の連続する0を符号語に変換することによってテストデータ量を削減する．以降本論文では，符号化前のテストパターンと符号化後のテストパターンをそれぞれ T_D, T_E とする.

表1の(a)と(b)に FDR 符号とグループサイズが4の Golomb 符号の例を示す．図1(a)のようなテスト集合が与えられたとき，図1(b)のようにすべてのテストパターンを連結し，1つのテストストリームとして扱う．その後 TE のように符号語に変換される.

ランレングスに基づく符号化は，テストパターン中の0がより長く連続する個所が多いとき，その効果が上がる．従って，テスト集合中に多くの未設定値があれば,テストデータ量をより小さくすることができる.

2.2 ドントケア判定

一般に ATPG によって生成されたテストパターンには未設定値は残っていない．これは故障シミュレーションにより検出できる故障を増やすために，ランダムかあるいは静的圧縮や動的圧縮[19]によって値を割り当てるためである．しかし，テストベクトル中の値には，逆の論理値を割り当てても故障検出率に影響を与えないものがある.そのような入力値はドントケア(X)

表1: 符号語の割り当て

(a) FDR符号

Group	Run-length	Group prefix	Tail	Codeword
A_1	0	0	0	00
	1		1	01
A_2	2	10	00	1000
	3		01	1001
	4		10	1010
	5		11	1011
A_3	6	110	000	110000
	7		001	110001
	8		010	110010
	9		011	110011
	10		100	110100
	11		101	110101
	12		110	110110
	13		111	110111

(b) Golomb符号

Group	Run-length	Group prefix	Tail	Codeword
A_1	0	0	00	000
	1		01	001
	2		10	010
	3		11	011
A_2	4	10	00	1000
	5		01	1001
	6		10	1010
	7		11	1011
A_3	8	110	00	11000
	9		01	11001
	10		10	11010
	11		11	11011

t_1 : 00100
t_2 : 10101
t_3 : 00011

TD : 001001010100011
TE : 100010000101100100

(a) 与えられたテスト集合　　(b) テストストリーム

図1: FDR符号化の例

m_1 : xffxf
m_2 : ffxxf
m_3 : xffxx

t_1' : x0100
t_2' : 101x1
t_3' : 000x1

(a) マスク集合　　(b) Xを含むテスト集合

図2: ドントケア判別

として扱うことができる．生成されたテスト集合のドントケア判定手法は[17]で提案されている．

判定したXには，符号化によりテストデータ量が削減するよう論理値を再割り当てできる．しかし，もし再割り当てした値が元の値と同じならばXの意味がなくなる．一般にXは一意に決まっていないため，多くの組合せが存在する．Xと判定されても意味がないビットがあらかじめ分かっている場合，それら以外のビットにXの数が増える．

そこで，テスト集合中のビット位置を特定した上で，そのビット上にドントケアを集中させるよう判定する手法が[18]で提案されている．この手法は，Xになることが望まれるビット位置を図2(a)に示す形式のマスクベクトルの集合で表現する．このマスク集合は，テスト集合の各入力値に対応しており，マスク集合中の'x'はXになることが望まれるビット位置を，'f'はそうでないビット位置を表す．図1(a)のテスト集合Tと図2(a)のマスク集合から図2(b)のようなテスト集合を得ることができる．

3. 提案手法

3.1 提案手法の概略

[18]のドントケア判定手法は，Xとなるビット位置を特定することができるため，テストコンプレッション技術の性能を最大限に引き出すことができる．ランレングス符号化は，テスト集合中の0の数が多い方がよいので，与えられたテスト集合中の1の部分が'x'で，0の部分が'f'であるようなマスク集合を用意する必要がある．図1(a)のテスト集合では，マスク集合は図3(a)のようになる．

ここで，元のテストデータTD_0から得られるXを含むテストデータとして，図3(b)のTD_1とTD_2のようなXのビット位置が異なる2種類を考える．TD_1のXに0を割り当てた後に表1(a)を用いてFDR符号化すると図4のTE_1のようになる．TD_0とマスク集合からTD_1をもとめ，Xに0を割り当てることによる利得は，$|TE_0|$ - $|TE_1|$=6ビットである．ここで$|TE_i|$はTE_iのビット数を示す．一方TD_2の場合，利得は$|TE_0|$ - $|TE_2|$=2ビットとなる．TD_1とTD_2のXの数は等しいが，利得は異なることが分かる．本手法では，ビット位置によってXにして0を割り当てることによる利得が異なることに注目する．

m_1 : ffxff
m_2 : xfxfx
m_3 : fffxx

TD_0 : 001001010100011
TD_1 : 00x0010101000x1
TD_2 : 0010010x01000x1

(a) マスク集合　　(b) テストストリーム

図3: テストパターン変換の例

TE_0 : 100010000101100100
TE_1 : 101101011010
TE_2 : 1000100010011010

図4: Xに0を割り当てた後にFDR符号化した例

3.2 利得表

論理値1のビットを反転することにより，2つの連続する0は1つの長く連続する0に連結される．連結により符号化後にどれだけビット数が少なくなるかは，図3, 4に示すように，連結するビットの前後に連続する0の長さで決まる．図3(b)のTD_1の場合，最初のXに0を割り当てることによって，0の長さが2の2つの部分が連結され，0の長さは5となる．FDR符号の場合，表1(a)の符号語より，0の長さが2のときの符号語と0の長さが5のときの符号語は，どちらも4ビットなので，利得は4ビットとなる．一方TD_2の場合，最初のXに0を割り当てると，0の長さが1の2つの部分が連結される．しかし，0の長さが1のときの符号語は2ビットであり，0の長さが3のときの符号語は4ビットなので，利得は0となる．

このように，表1の符号の割当に基づいて，変換する1の前後に連続する0の長さから利得をそれぞれ計算することができる．そこで，本手法では2つの0が連続する部分を連結して符号化することによって削減できるビット数を示す2次元の表をあらかじめ用意する．これを利得表と呼ぶ．FDR符号化の利得表を表2に示す．縦軸と横軸の数字は，それぞれ連結するビットの前後に連続する0の長さを示している．利得表を用いて，利得が大きいビットを特定してマスク集合を生成すれば，符号化の性能を最大限に引き出すようにテストパターンを変換することができる．

3.3 処理手順

テストパターン中のXの位置には組合せがあり，利得が小さなビットが多くXになると，利得が大きなビットのXは少なくなる．利得の小さいビットがXと判定されるのを避けるために，Xにしたいビットの利得の最小値をg_{min}とする．本手法では，利得表より論理値が1であるすべてのビットの利得を調べ，その利得がg_{min}以上のときそのビットを'x'，それ以外のときそのビットを'f'としてマスク集合を生成する．g_{min}の初期値は8として，Xが見つからなくなるまでX判定し，Xに0を割り当てる処理を繰り返す．Xが見つからなくなると，g_{min}の値を減らして処理を続ける．

処理手順の概略を以下に示す．

Step 0：与えられたテスト集合をT，g_{min}を8とする．

Step 1：テスト集合Tと利得表より，利得がg_{min}以上のビットを'x'，それ以外のビットを'f'としてマスク集合を生成する．

Step 2：マスク集合によって特定したTのビットに対してXを判定する．

Step 3：Xと判定されたビットの利得がg_{min}以上ならばXに0を割り当て，それ以外は，Xに1を再割り当てする．変換したテスト集合をT'とする．

表2: FDR符号に対する利得表

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	2	0	2	2	2	0	2	2	2	2	2	2	2	0	2	2
1	0	0	2	2	0	0	2	2	2	2	2	2	2	0	2	2
2	2	2	4	2	2	2	4	4	4	4	4	4	2	2	4	4
3	2	2	2	2	2	2	4	4	4	4	2	2	2	2	4	4
4	2	0	2	2	2	2	4	4	4	2	2	2	2	2	4	4
5	0	0	2	2	2	2	4	4	2	2	2	2	2	2	4	4
6	2	2	4	4	4	4	6	4	4	4	4	4	4	4	6	6
7	2	2	4	4	4	4	4	4	4	4	4	4	4	4	6	6
8	2	2	4	4	4	2	4	4	4	4	4	4	4	4	6	6
9	2	2	4	4	2	2	4	4	4	4	4	4	4	4	6	6
10	2	2	4	2	2	2	4	4	4	4	4	4	4	4	6	6
11	2	2	2	2	2	2	4	4	4	4	4	4	4	4	6	6
12	2	0	2	2	2	2	4	4	4	4	4	4	4	4	6	6
13	0	0	2	2	2	2	4	4	4	4	4	4	4	4	6	6
14	2	2	4	4	4	4	6	6	6	6	6	6	6	6	8	6
15	2	2	4	4	4	4	6	6	6	6	6	6	6	6	6	6

Step 4：もし$T=T'$ならば，g_{min}に$g_{min}-2$を代入する．$g_{min}<0$ならば処理を終了する．

Step 5：TにT'を代入してStep 1の処理を行う．

X判定はテスト集合とマスク集合を用いて実行されるが，Step 3ではすべてのXに0を割り当てるわけではない．これはビットの論理値を変換すると，隣接する1のビットの利得が変わってしまうためである．2つのビット変化による利得が，必ずしも個々のビット変化による利得の合計と等しくなるわけではない．本手法は処理時間を抑えるために，個々のビット変化による利得が大きいビットを変換している．従って，テストパターン変換の最適解ではない．

図5に処理手順の例を示す．テスト集合TD_0が与えられたとする．まず2行目に示すように各ビットの利得を調べる．利得が4より大きいビットがないので，$g_{min}=4$の場合から説明する．TD_0に対するマスク集合は*mask set_0*のように生成される．TD_0と*mask set_0*

TD_0 : 001001010100011
gain : 004002000200022
mask set_0 : ffxfffffffffff

TD_0' : 00X001010100011
TD_1 : 000001010100011
gain : 000000000200022
mask set_1 : fffffffffxfffxx

TD_1' : 000001010X000X1
TD_1 : 000001010000011
gain : 000000000000002
.....
.....

図5: ピンポイントテストパターン変換の例

からテスト集合 TD_0'が得られたと仮定すると，TD_0'のXに0が割り当てられ，その結果テスト集合は TD_1 となる．次に $g_{min}=2$ のとき，TD_1 の各ビットの利得から *mask set_1* が生成される．TD_1'のようにXが2つ判定されたと仮定すると，左のXには0を割り当てるが右のXには0を割り当てない．これは，左のXに0を割り当てることによって，右のXの利得が g_{min} より小さい0となるためである．この後も処理は続くがここでは説明を省略する．

4. 実験結果

提案手法をPC(PentiumⅣ 1.80GHz, 256MB)上にC言語で実装し，ISCAS'89ベンチマーク回路に対して実験を行った．最初のテスト集合として，コンパクトテスト集合[2]を用い，利得表として，510×510の行列を用意した．表3にFDR符号化と $m=4$ でのGolomb符号化に対する実験結果を示す．2列目と3列目は，符号化される各テスト集合のテストパターン数とテストパターンのビット数を示す．*TD* は元のテストデータ量，"FDR"と"Golomb"はそれぞれFDR符号化とGolomb符号化によって圧縮したときのテストデータ量を示している．両符号化法ともテストデータ量を削減できるが，FDR符号化の方がより削減することができた．その理由としては，ランレングスが0や1などの短いときのコードワード長が，Golomb符号よりFDR符号の方が短いことが考えられる．また，表4にFDR符号化に対する実験結果の詳細を示した．"#1s"と"#1to0s"は元のテストパターン中の論理値1のビット数と提案手法によって1から0に変換したビット数を示す．また，"#iterations"は処理手順において各 g_{min} での繰り返し回数を示す．最後の"time"の欄に処理時間を示す．与えられたテストパターンの1のうち，約75%を0に変換した．$g_{min}=2$ のとき最も多く変換しているため今後の課題として，$g_{min}=2$ での処理の新しい基準を考える必要がある．処理時間に関しては，繰り返し回数は20を超えているが，2000秒程度であった．

表3: FDR符号化とGolomb符号化に対する実験結果

circuit	#tests	#inputs	TD [bits]	FDR [bits]	Golomb [bits]
s5378	100	214	21400	10490	12454
s9234	111	247	27417	15454	18096
s13207	235	700	164500	22840	52890
s15850	97	611	59267	18466	26077
s35932	12	1763	21156	18072	26609
s38417	87	1664	144768	64078	77711
s38584	114	1464	166896	59262	74991

表4: FDR符号に対する実験結果の詳細

circuit	#1s [bits]	#1to0s [bits]	#iterations g_{min}=8	g_{min}=6	g_{min}=4	g_{min}=2	g_{min}=0	time [sec]
s5378	11333	8489	1	2	3	29	3	31
s9234	15154	10727	1	2	4	17	4	90
s13207	72141	67555	3	4	4	133	4	1184
s15850	28748	24212	1	2	4	49	4	412
s35932	13457	5595	2	2	3	17	7	501
s38417	94695	78147	1	2	4	77	8	2254
s38584	67733	54357	2	3	6	42	4	1216

表5: 実験結果の比較

circuit	ours	nonpinpoint	EFDR [9]	VIHC [14]	RESPIN++ [15]	XORnet [12]
s5378	10490	10572	11419	11516	17332	N/A
s9234	15454	15846	21250	17736	17198	N/A
s13207	22840	24736	29992	27737	26004	25344
s15850	18466	19906	24643	30271	32226	22784
s35932	18072	17550	5554	9458	N/A	7128
s38417	64078	66020	64962	74938	89132	89356
s38584	59262	61960	73853	85674	63232	38976

表記法については，

テスト集合中の0を1に変換し，連続する1を符号化するべきと思われる．

5. まとめ

本論文では，FDR符号化のようなランレングス符号化の効果を最大限に引き出すテストパターン変換手法を提案した．与えられたテストパターンを符号化する前に，X判定手法を用いて故障検出率を低下させることなく，連続する0を連結するようにテストパターン変換を行った．このとき利得表を用いて連続する0を連結したときの利得を参照し，より大きな利得が得られるビットを特定した上でX判定を行った．ISCASベンチマーク回路に対する実験結果では，提案手法によりテストデータ量を平均55%削減できた．

文　献

[1] I. Pomeranz, L. N. Reddy and S. M. Reddy, "COMPACTEST: A Method to Generate Compact test Sets for Combinational Circuits," IEEE Trans. CAD, pp. 1040-1049, Jul. 1993.

[2] S. Kajihara, I. Pomeranz, K. Kinoshita and S. M. Reddy, "Cost-Effective Generation of Minimal Test Sets for Stuckat Faults in Combinational Logic Circuits," IEEE Trans. CAD, pp.1496-1504, Dec. 1995.

[3] I. Hamzaoglu and J. H. Patel, "Test Set Compaction Algorithms for Combinational Circuits," Proc. Int'l Test Conf, pp. 283-289, Oct. 1998

[4] 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. Comp., pp. 223-233, Feb. 1995.

[5] S. Hellebrand, H.-G. Liang, H. -J. Wunderlich, "A mixed-mode BIST scheme based on reseeding of folding counters," Int'l Test Conf., pp.778-784, Oct. 2000.

[6] C. V. Krishna, A. Jas, and N. A. Touba, "Test Vector Encoding Using Partial LFSR Reseeding," International Test Conf., pp. 885-893, Oct. 2001.

[7] A. Chandra and K. Chakrabarty, "Test Data Compression for System-on-a-Chip Using Golomb Codes," VLSI Test Symposium, pp. 113-120, April 2000.

[8] A. Chandra and K. Chakrabarty, "Frequency-directed Runlength (FDR) Codes with Application to System-on-a-chip Test Data Compression," Proc. VLSI Test Symp., pp. 42-47, April 2001.

[9] A. El-Maleh, R. Al-Abaji, "Extended Frequency-directed Runlength Codes with Improved Application to System-on-a-Chip Test Data Compression," Int'l Conf. Electronics, Circuits and Systems, pp. 449-452, 2002.

[10] A. Jas, J. Ghosh-Dastidar, and N. A. Tuba, "Scan Vector Compression/Decompression Using Statistical Coding," VLSI Test Symposium, pp. 114-120, April 1999.

[11] H. Ichihara, K. Kinoshita, I. Pomeranz, S. M. Reddy, "Test Transformation to Improve Compaction by Statistical Encoding," International Conf. on VLSI Design, pp.294-299, Jan. 2000.

[12] I. Bayraktaroglu, and A. Orailoglu, "Test Volume and Application Time Reduction Through Scan Chain Concealment," Design Automation Conference, pp.151-155, June 2001.

[13] S. M. Reddy, K. Miyase, S. Kajihara and I. Pomeranz, "On Test Data Volume Reduction for Multiple Scan Chain Designs," VLSI Test Symposium, pp. 103-108, April 2002.

[14] P. T. Gonciari, B. Al-Hashimi, N. Nicolici, "Improving compression ratio, area overhead, and test application time for system-on-a-chip test data compression/decompression," Design Automation and Test in Europe Conf., pp. 604-611, 2002.

[15] L. Schafer, R. Dorsch, H. -J. Wunderlich, "RESPIN++ -Deterministic Embedded Test," European Test Workshop, pp.37-44, May 2002.

[16] M. Schulz, E. Trischler, and T. Sarfert, "SOCRATES: A Highly Efficient Automatic Test Pattern Generation System," IEEE Trans. on CAD., pp. 126-137, Jan. 1988.

[17] S. Kajihara, and K. Miyase, "On Identifying Don't Care Inputs of Test Patterns for Combinational Circuits," Proc. Int'l Conf. on Computer Aided Design, pp. 364-369, Nov. 2001.

[18] K. Miyase, S. Kajihara, I. Pomeranz and S. M. Reddy, "Don't care identification on specific bits of test patterns," 2002 International Conference on Computer Design, pp. 194- 199, Sep. 2002.

[19] P. Goel, and B. C. Rosales, "Test Generation and Dynamic Compaction of Tests," Digest of Papers 1979 Test Conf., pp. 189-192, Oct. 1979.

[20] A. Jas and N. A. Touba, "Using an embedded processor for efficient deterministic testing of system-on-a-chip," *Int.'l Conf. on Computer Design*, pp.418-423, 1999.