システムＬＳＩ設計技術　103－18
（2001. 11. 29）

論理回路のテストパターンに含まれるドントケアの判定法について

宮瀬 紘平† 梶原 誠司†‡

†九州工業大学情報工学部電子情報工学科
‡九州工業大学マイクロ化総合技術センタ
〒820-8502 福岡県飯塚市大字川津 680-4

E-mail: {miyase,kajihara}@aries30.cse.kyutech.ac.jp

あらまし 論理回路のテストパターンを構成する論理値には，逆の値に置き換え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（以下，Xと表す）と見なすことができる．本論文では，ATPGアルゴリズムの含意操作・正当化操作と同様の処理を用いて，与えられたテスト集合に含まれるドントケアをできるだけ多く見つける手法を提案する．ISCASベンチマーク回路に対する実験では，本手法は圧縮されていないテスト集合の約66%の入力がXとなり，また，圧縮されたテスト集合であっても平均47%のXとなることを示す．本手法は，SoCのテストにおける諸問題を解決するための種々の手法に応用できるが，本論文の最後に，本手法を用いたテストパターン変換の応用について議論する．
キーワード テスト生成，ドントケア入力，テスト圧縮，組合せ回路，縮退故障

On Identifying Don't-Care Inputs of Test Patterns for Logic Circuits

Kohei MIYASE† and Seiji KAJIHARA†‡

† Department of Computer Sciences and Electronics
‡ Center for Microelectronics Systems
Kyushu Institute of Technology
680-4, Kwazu, Iizuka, Fukuoka 820-8502
E-mail:{miyase,kajihara}@aries30.cse.kyutech.ac.jp

Abstract Given a test set for stuck-at faults, some of primary input values may be changed to opposite logic values without losing fault coverage. One can regard such input values as don't care(X). In this paper, we propose a method for identifying X inputs of test vectors in a given test set. While there are many combinations of X inputs in the test set generally, the proposed method finds one including X inputs as many as possible, by using fault simulation and procedures similar to implication and justification of ATPG algorithms. Experimental results for ISCAS benchmark circuits show that approximately 66% of inputs of un-compacted test sets could be X. Even for compacted test sets, the method found that approximately 47% of inputs are X. Finally, we discuss how logic values are reassigned to the identified X inputs where several applications exist to make test vectors more desirable for solving problems of SoC testing.

Key words test generation, don't-care input, test compaction, combinational circuit, stuck-at fault

1. はじめに

論理回路に対するテスト生成については，長い間研究されており，多くの結果が得られている[1]．テスト生成において重要なことは，短時間で単一縮退故障に対して高い故障検出率(または故障効率)のテスト集合を得ることであった．VLSI 技術の発展に伴い，テスト生成に対する要求も例えば次のように変わってきている．

・単一縮退故障だけでなく，ブリッジ故障や遅延故障等の他の故障も検出すること[2]．
・テストベクトル数を少なくすること[3]．
・テスト時の消費電力を削減すること[4]．

一般に，テスト集合は単一縮退故障の検出率を指標に生成され，一旦テスト集合が生成されると，他の要求を満たすようにテスト集合を修正することは容易ではない．ある対象故障について ATPG によりテストベクトルが生成された直後は不定値が残っているが，その不定値には，対象故障以外の未検出故障を検出するように，故障シミュレーション前に論理値が割り当てられる．したがって最終的なテストベクトル中に不定値は残っていない．

しかしながら，外部入力値には，逆の論理値に変更しても故障検出率が低下しない値も含まれることがある．そのような入力値は，ドントケア(以下，X と表す)と見なすことができる．本論文では，与えられたテスト集合中の X を判定する手法を提案する．一般に，X の選び方には複数の組合せが存在するが，提案手法では，できるだけ多くの X を含む組合せを発見する．X の検出には，故障シミュレーションと，ATPG アルゴリズムの含意操作・正当化操作と同様の処理を用いる．その際，得られたテスト集合が元のテスト集合を被覆するように含意操作と正当化操作には制限と拡張を加える．テスト集合に含まれる X の割合は元のテスト集合の大きさにより変わるが，ISCAS ベンチマーク回路における実験では，テスト圧縮技法を用いずに生成したテスト集合に対して，約 66%の入力が X となり，また，圧縮されたテスト集合[8]であっても約 47%の入力が X となることがわかった．X を含んだテスト集合が得られた後，X には任意の論理値を割り当てることができるが，本論文では，X をどのように再割当するかについて，いくつかの応用についても議論する．

2. 準備

2.1 問題設定

本論文では，組合せ回路，またはフルスキャン順序回路の単一縮退故障について生成されたテスト集合を扱う．テストベクトルの外部入力値がすべて 0 か 1 に特定されたテスト集合 T が与えられたとき，次の特性をもった，X を含むテスト集合 T'を導出することを考える：

(1) T'は T を被覆する．
(2) T'と T の縮退故障検出率は等しい．
(3) T'はできるだけ多くの X を含む．

以下に簡単な例を示す．図 1 の回路に対して表 1(a)のテスト集合 T が生成されたとき，表 1(b)のテスト集合は T'の一つの解である．テストベクトル t_1 は，故障 a/0, b/0, そして c/1 を検出する．ここで s/v は，信号線 s の v 縮退故障を意味する．a/0 は t_1 以外で検出できないため，t_1 によって必ず検出されなければならない．一方 t_3 でも検出可能な c/1 は，必ずしも t_1 によって検出される必要はない．このため t_1 の入力 c の信号値 0 は X にできる．同様に t_4 における入力 a の信号値 0 も X になり，結果として，表 1(b)のテスト集合 T'が求められる．

以下本論文では，与えられたテスト集合と得られたテスト集合をそれぞれ T と T'で表す．同様に，T に含まれるテストベクトルと T'に含まれるテストベクトルを，それぞれ t_i と t_i'で表す．ただし t_i'は，t_i から得られたものとする．

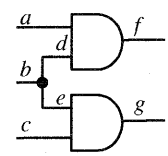


図1　回路例

表1(a)
テスト集合T

	a b c
t_1	1 1 0
t_2	1 0 1
t_3	0 1 0
t_4	0 1 1

表1(b)
テスト集合T'

	a b c
t_1	1 1 x
t_2	1 0 1
t_3	0 1 0
t_4	x 1 1

2.2 必須故障

次に本論文で用いる用語を定義する．与えられた回路に対して生成されたテスト集合を T とする．T において，故障 f がテストベクト $t \in T$ によって検出できるが，T の t 以外のテストベクトルでは検出できないとき，f を t の必須故障[7]という．t が必須故障を持たないとき，t を冗長ベクトルという．T が冗長ベクトルを含まないとき，T を極小テスト集合という．必須故障と極小テスト集合は二重検出法[8]によって簡単に求めることができる．

2.3 提案手法の概略

与えられたテスト集合 T からテスト集合 T' を得るまでの基本処理手順を説明する．2.1 で述べた T' の制約を満たすためには，$t_i \in T$ が検出する必須故障は，$t_i' \in T'$ によって必ず検出されなければならない．一方，t_i で検出される必須故障以外の故障は，必ずしも t_i' で検出される必要はない．それゆえ，最初に各 t_i の必須故障を検出するような t_i の入力値を決定した後，決まった入力値では未検出の故障を検出するように，さらに t_i' の他の値を決定する．最終的に値が決まらなかった論理値が X とみなされる．

T から T' を得るための処理手順を図 2 に示す．まず Step1 で，t_i の必須故障集合を求める．次に Step2 で，それらの必須故障を検出するための t_i の外部入力値を計算し，仮の t_i' とする．Step2 で求めた t_i' はまだ一時的なものであり，最終的なものではない．Step3 で t_i' に対する故障シミュレーションを行う．ここで用いる故障シミュレータは X を含んだテストベクトルを扱うものである．Step1 から Step3 の処理により，初期の T' が得られる．

この T' では未検出故障があるので，全ての故障が検出されるように，t_i' の X のいくつかを元のテストベクトル t_i の値に戻す．Step4 で，t_i で検出できる未検出故障を抽出し，Step5 でそれらを検出するための入力値を計算する．これで t_i が最終的に決定する．

```
Basic Procedure X-search(C, T)
        Circuit C; Test set T;
{
    for each test pattern t_i in T {
        F=collect_essential_fault(t_i);        /* step 1 */
        t_i' = find_value(F);                  /* step 2 */
        fault_simulation(t_i');                /* step 3 */
    }
    for each test pattern t_i in T {
        G=collect_undetected_fault(t_i);       /* step 4 */
        t_i' += find_value(G);                 /* step 5 */
        fault_simulation(t_i');                /* step 6 */
    }
        return T' composed of t_i';
}
```

図 2　基本処理手順

3. 故障を検出するための論理値

本手法は各テストベクトルで検出すべき故障を求めた後，検出に必要な入力値を計算する．本章では，入力値をどのように計算するかを述べる．

3.1 故障伝搬経路の選択

テストベクトルが故障を検出するとき，故障は顕在化され，かつ，少なくとも 1 つの外部出力へ故障の影響が伝搬している．そこで，故障検出に必要な値の計算は，まず，故障信号線と故障伝搬経路を活性化するための内部信号値を決定する．次に，それらの値を満たす外部入力値を ATPG アルゴリズムで用いられる含意操作と正当化操作を用いて決定する．

故障伝搬経路の選択において，経路が 2 つ以上存在する場合，1 つの経路が故障を伝搬すれば十分である．本手法では，故障伝搬経路は 1 つだけ選択する．また，経路の選択は，より多くの故障が伝搬している外部出力を選択する．なお，故障の伝搬経路と故障伝搬に必要な内部信号値は，故障シミュレーションの結果を参照して求める．

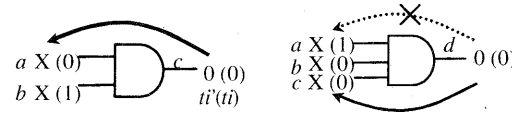
3.2 限定含意操作と限定正当化操作

故障を顕在化し故障伝搬経路を活性化する外部入力値の決定は，ATPG アルゴリズムで用いられる含意操作と正当化操作に類似した処理を使用する．ATPG での処理と異なる点は，本手法は，元の論理値と矛盾しないように取る値に制限を持つことである．そこで本手法で使用する含意操作と正当化操作を限定含意操作，限定正当化操作と呼ぶこととする．

図 3(a)に限定含意操作の例を示す．括弧の外の値は t_i の論理値を表し，括弧の中の値は t_i の論理値を表す．t_i' におけるゲートの入力 a，b は未割当状態で，ゲートの入力 c に新しく 0 が割り当てられたとする．限定含意操作では，c=0 に対して，b は t_i での値が 1 より t_i' で 0 にできないため，a=0 が求められる．通常の ATPG では，含意操作により a, b の値は決まらず，正当化操作でも a=0 でなく b=0 としてもよい．

次に図 3(b)に限定正当化操作の例を示す．t_i' におけるゲート入力 a，b，c が未割当の状態で，ゲート出力 0 を正当化する場合を考える．限定正当化操作では，t_i における a の論理値が 1 のため，b=0 か c=0 のみ選択でき，a=0 は選択できない．

外部出力へ故障を伝搬する経路を選んだ後，未正当化信号線がなくなるまで，限定含意操作と限定正当化操作を適用することで，故障を検出する内部信号値を満たす外部入力は求まる．値が割り当てられていない外部入力が，t_i' の X となる．



(a)限定含意操作　　(b)限定正当化操作

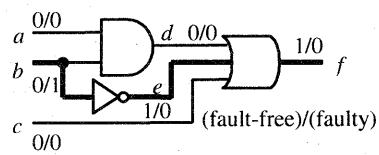
図3　限定含意操作と限定正当化操作

3.3 拡張含意操作

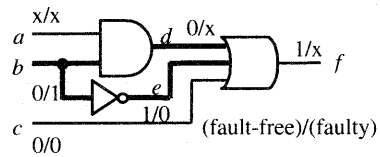
限定含意操作と限定正当化操作は 3 値論理(0,1,X)

に基づく操作である．故障値を無視しているために，求めた T'には，検出を見逃す故障の存在する場合がある．図4に例を示す．テストベクトル(a,b,c)=(0,0,0)は信号線 b の1縮退故障を検出する．故障顕在化のための b=0，故障伝搬経路 b-e-f を活性化する d=0，c=0 が故障検出に必要な値である．b と c は外部入力で，d=0 は b=0 によって決定されるので，それ以上の割当は不要である．その結果，a の値はXとなる．しかしながら，図4(b)に示すように，a=X では信号線 b の1縮退故障の検出は保証されず，実際に a=1 では検出できない．

これは，限定含意操作と限定正当化操作が故障時の値を考慮してないため生じる．図4の例では，d=0 が故障時の値に必要なのに対して，正常時の b=0 によって d=0 が含意されるが，この含意では，故障時 d=1 は得られない．そこで，見逃された故障を確実に検出するため，新たに拡張含意操作を提案する．



(a) ドントケアに変換前



(b) ドントケアに変換後

図4　故障を見逃す例

拡張含意操作では，ゲート入力信号線の少なくとも1つに故障の影響が伝搬しているとき，ゲート入力信号線の値を全て t_i と同じ値に決定する．図5に拡張含意操作の例を示す．ゲートの入力の t_i における値が(a, b, c)=(0, 1, 0)であり，a と b に故障が伝搬していたとする．ゲート入力 a, b, c が t_i'に対して未割当の状態で，ゲート出力 d の値が0に割り当てられた場合を考える．このとき拡張含意操作は，d=0 から a=0, b=1, c=0 のすべてを含意する．もし，限定含意操作が適用されるなら，どの入力値も決定できず，限定正当化操作によって a=0 もしくは c=0 のどちらか一つを決定する．拡張含意操作では，すべての入力値を決定するため，必要のない値までも決定するかもしれない．図5の例では，d=0 にするためには，c=0 のみ，または a=0 と b=1 で十分である．したがって，拡張含意操作の適用はできるだけ避けた方がよく，限定含意操作と限定正当化操作で見逃した故障に対してのみ適用する．また，拡張含意操作では前方への含意操作は行わない．なぜなら拡張含意操作を行った場合でも前方への含意操作により故障が検出されなくなる場合があるからである．

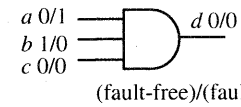


図5　拡張含意操作適用例

図2の処理手順に図6の Step7 から Step9 の処理を加えることで，故障の見逃しのないテスト集合 T' が得られる．拡張含意操作は Step8 でのみ使用する．

```
for each test pattern t_i in T {
        H=collect_undetected_fault(t_i);   /* step 7 */
        t_i' += find_value_extended(H);    /* step 8 */
        fault_simulation(t_i');            /* step 9 */
}
return T' composed of t_i';
```

図6　追加する処理

4. 実験結果

提案手法を PC 上（Pentium3 700MHz, 384M）に C 言語で実装し，ISCAS のベンチマーク回路に対して実験した．それぞれの回路に，圧縮されていないテスト集合と圧縮されたテスト集合[8]の2種類のテスト集合を用意した．表2と表3に，それぞれ圧縮されていないテスト集合に対する実験結果と圧縮されたテスト集合に対する実験結果を示す．

最初の3つの欄は，回路名，擬似外部入力も含む入力数，与えられたテスト集合のテストベクトル数を表す．次の3つの欄は，見つかった X の平均値，Xを最も多く含むテストベクトルにおけるXの割合，X が最も少ないテストベクトルの X の割合を表す．次の"#faults"と"#flt-ext"の欄は，対象故障数と図2の基本処理手順で検出できなかった故障数を表す．"#flt-ext"で数えられた故障には 3.3 で述べた拡張含意操作を含む処理を適用し，その検出が保障される．最後の欄は実行時間で，単位は秒である．

圧縮されていないテスト集合に対しては，本手法は平均約 66%の X を検出できた．圧縮されたテスト集合ではXの割合が少なくなるが，それでも，平均

表2 圧縮されていないテスト集合に対する実験結果

circuits	#Pis	#tests	%X-ave	%X-max	%X-min	#faults	#flt-ext	time(sec)
c432	36	54	49.0	100	13.9	520	3	0.03
c499	41	94	20.5	100	0	750	41	0.05
c880	60	78	64.3	100	11.7	942	7	0.07
c1355	41	129	25.3	100	0	1566	0	0.16
c1908	33	150	33.7	100	0.0	1862	17	0.32
c2670	233	142	84.0	100	27.9	2630	17	0.69
c3540	50	207	59.2	100	12.0	3291	42	1.71
c5315	178	186	80.7	100	25.8	5291	34	2.04
c6288	32	38	13.5	100	0	7710	16	1.91
c7552	207	290	76.2	100	16.4	7419	90	5.01
s1238	32	195	62.0	100	28.1	1286	1	0.16
s1423	91	93	76.5	100	20.9	1501	29	0.12
s1494	14	166	38.1	100	7.1	1494	1	0.16
s5378	214	333	88.7	100	19.2	4563	11	2.89
s9234	247	480	88.3	100	23.9	6475	58	7.74
s13207	700	586	96.0	100	33.0	9664	158	14.84
s15850	611	500	94.0	100	24.7	11336	77	15.71
s35932	1763	76	84.8	99.9	0.6	35110	16	18.44
s38417	1664	1243	96.7	100	24.2	31015	271	108.53
s38584	1464	854	96.3	100	8.3	34797	187	75.02

表3 圧縮されたテスト集合に対する実験結果

circuits	#Pis	#tests	%X-ave	%X-max	%X-min	#faults	#flt-ext	time(sec)
c432	36	28	46.9	66.7	8.3	520	0	0.02
c499	41	52	0.6	14.6	0.0	750	34	0.03
c880	60	21	31.7	63.3	11.7	942	0	0.04
c1355	41	84	0.0	2.4	0.0	1566	2	0.13
c1908	33	106	15.8	63.6	3.0	1862	22	0.28
c2670	233	45	70.1	86.3	20.6	2630	3	0.33
c3540	50	93	49.3	76.0	26.0	3291	58	0.98
c5315	178	186	59.5	89.3	23.0	5291	3	0.81
c6288	32	14	0	0	0	7710	0	1.18
c7552	207	75	52.7	82.1	10.6	7419	64	1.90
s1238	32	125	55.0	65.6	25.0	1286	0	0.10
s1423	91	24	41.1	71.4	17.6	1501	7	0.08
s1494	14	100	25.5	57.1	0.0	1494	0	0.11
s5378	214	100	71.0	88.3	8.9	4563	5	1.13
s9234	247	111	67.2	87.5	36.0	6475	35	2.45
s13207	700	235	91.6	98.4	16.4	9664	149	7.34
s15850	611	97	76.1	96.6	24.2	11336	60	4.87
s35932	1763	12	34.4	83.5	0.0	35110	345	9.03
s38417	1664	87	73.4	89.4	20.1	31015	93	13.73
s38584	1464	114	79.7	95.6	14.3	34797	88	15.78

47%のXを検出できた．Xの割合は回路とそのテスト集合の大きさに依存するが，一般的に，外部入力の数が多いほど，テスト集合中のXの割合も大きいといえる．また，与えられたテスト集合が極小でない場合，テストベクトルの値が全てXになることがある．拡張含意操作によって検出される故障は，回路の全故障数に対して約0.6%であった．本手法の計算時間は，与えられたテスト集合の大きさと回路の規模に顕著に依存する．なぜなら提案手法は，テストベクトルを1つづつに処理するからである．

5. SoCテストの諸問題への応用

本章では，提案手法により見つかったXへの値の再割り当ての例を紹介する．

5.1 テスト圧縮

最も典型的な応用はテスト圧縮である．例えば，表1(b)のテスト集合T'は，t_1'とt_4'の併合により圧縮できる．これは静的圧縮[6]で，テスト集合が十分に圧縮されていないときに有効である．また，いくつかのテストベクトルが冗長になるようTに対して動的圧縮[9]を適用することも可能である．

フルスキャン回路のテスト時間の削減も，可能である．もしスキャンアウト出力に近い擬似外部入力にXが存在すれば，スキャンインの操作は省略できる．スキャンイン入力に近い擬似外部出力のXも同様である．スキャンシフトの省略は，テスト時間の削減[10],[11]に貢献する．

その他の応用として，決定性BISTに対するテスト圧縮である．ROMにテスト集合を格納する場合，テスト集合を符号化[12]することによってデータ量を少なくできる．Xを含むテスト集合に対してはより効率的な符号化が期待できる．

5.2 テスト時の消費電力削減

テスト時の消費電力は，通常動作時と比べて100%から200%高いことが知られている[13]．そのため，静的な状態では問題のない回路でも，いくつかのテストベクトルに対しては，回路中に故障があるかのような応答をすることがある．LSIの消費電力はスイッチング動作に比例して増加するので，Xにスイッチングを少なくする値を割り当てることによりテスト時消費電力を削減できる[14]．

5.3 欠陥検出率の向上

テスト集合の欠陥検出率は，テスト集合がそれぞれの縮退故障を1回以上検出するように生成されると向上することが知られている[2],[15]．提案手法によるXを使用することにより，テスト集合の欠陥検出率を向上させることができる．

本論文では，縮退故障のテストにおけるXを見つける手法を述べたでが，パス遅延故障のテスト集合のXも識別できる．電力供給ノイズはパス遅延に影響を及ぼすことが知られているが[16]，Xを用いて，テストパターンを遅延が最悪なものに変換できる．

6. まとめ

本論文では，テスト集合中に含まれるXを識別する手法を提案した．提案手法は，故障シミュレーションとATPGアルゴリズムの含意操作・正当化操作に類似した処理を使用した．ISCASベンチマーク回路に対しての実験では，提案手法は圧縮されていないテスト集合の約66%の入力が，また，圧縮されたテスト集合の，約47%の入力がXであることを示した．最後に，Xの再割当法について議論し，いくつかの応用を紹介した．

文　献

[1] M.L.Bushnell, V.D.Agrawal, Essentials of Electronic Testing for Digital, Memory & Mixed-Signal VLSI Circuits, Kluwer Academic Publishers, 2000.

[2] S. C. Ma, P. Franco, and E. J. McCluskey, "An Experimental Chip to Evaluate Test Techniques Experiment Results," 1995 International Test Conf., pp. 663-672, Oct. 1995.

[3] I. Pomeranz, L. N. Reddy, and S. M. Reddy, "COMPACTEST: A Method To Generate Compact Test Sets for Combinational Circuits," 1991 International Test Conf., pp. 194-203, Oct. 1991.

[4] S. Wang, S. K. Gupta, "ATPG for Heat Dissipation Minimization during Test Application," IEEE Trans. Computer, Vol. 47, No. 2, pp. 256-262, Feb. 1998.

[5] 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.

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

[7] J. -S. Chang, and C. –S. Lin, "Test Set Compaction for Combinational Circuits, " IEEE Trans. on Computer-Aided Design, pp.1370-1378, November 1995.

[8] S. Kajihara, I. Pomeranz, K. Kinoshita and S. M. Reddy, "Cost Effective Generation of Minimal Test Sets for Stuck at Faults in Combinational Logic Circuits", IEEE Trans. Computer-Aided Design of Integated Circuits and Systems, Vol. 14, No. 12, pp. 1496-1504, Dec. 1995.

[9] L. N. Reddy, I. Pomeranz, and S. M. Reddy, "ROTCO: A Reverse Order Test Compaction Technique," 1992 IEEE EURO-ASIC Conference, pp. 189-194, Sept. 1992.

[10] S. Y. Lee and K. K. Saluja, " An Algorithm to Reduce Test Application Time in Full Scan Designs" Proc. Int'l Conf. on CAD, pp. 17-20, Nov. 1992.

[11] Y. Higami, S. Kajihara, and K. Kinoshita, "Reduced Scan Shift: A New Testing Method for Sequential Circuits," IEEE International Test Conference, pp. 624-630, Oct. 1994.

[12] V. Iyenger, K. Chakrabarty, and B. T. Murray, "Built-in Self Testing of Sequential Circuits Using Precomputed Test Sets," 16th VLSI Test Symposium, pp. 418-423, 1998.

[13] Y. Zorian, "A Distributed BIST Control Scheme for Complex VLSI Devices," 11th VLSI Test Symposium, pp. 4-9, 1993.

[14] R. Sankaralingam, R. R. Oruganti, N. A. Touba, "Static Compaction Techniques to Control Scan Vector Power Dissipation," 18th VLSI Test Symposium, pp. 35-40, 2000.

[15] S. M. Reddy, I. Pomeranz, and S. Kajihara, "Compact Test Sets for High Defect Coverage," IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, Vol. 16, No.8, pp.923-930, Aug. 1997.

[16] A. Krstic, Y. –M. Jiang, K. –T. Cheng, "Delay Testing Considering Power Supply Noise Effects," International Test Conf., pp. 181-190, Sept. 1999.