設 計 自 動 化 81－13
(1996. 10. 18)

入力パタン並列故障シミュレーションの高速化手法

桃枝 孝一郎　　川崎 隆志　　大附 辰夫

早稲田大学理工学部電子通信学科
〒169 東京都新宿区大久保 3-4-1
E-mail: momoeda@ohtsuki.comm.waseda.ac.jp

あらまし

故障シミュレーションとは，与えられたテストパタン系列によって検出可能な故障を調べることである．従来，高速な故障シミュレーション手法として，入力パタン並列故障シミュレーション（PPSFP:Parallel-Pattern Single-Fault Propagation）法およびCPT（Critical Path Tracing）法を組み合わせて適用したものがある．しかしながら，PPSFP法はCPT法に比較し処理時間を多く必要とするため，故障シミュレーションの高速化には，PPSFP法の処理量の削減が不可欠とされている．本稿では，このような観点から故障シミュレーションの高速化手法を提案する．提案手法は，CPT法における故障伝搬情報を利用することにより，PPSFP法の処理量を削減する．提案手法を計算機上に実装し，ISCAS'85 ベンチマーク回路を用いた計算機実験により，提案手法の有効性を評価する．

キーワード　故障シミュレーション，テスト設計，テストパタン系列，故障検出率，PPSFP，CPT

A Fast Fault Simulation Algorithm Based on the Parallel-Pattern Single-Fault Propagation

Koichiro MOMOEDA　　Takashi KAWASAKI　　Tatsuo OHTSUKI

Dept. of Electronics and Communication Engineering
Waseda University
3-4-1 Okubo, Shinjuku, Tokyo 169, Japan
E-mail: momoeda@ohtsuki.comm.waseda.ac.jp

Abstract

Fault simulation is a method for examining the faults detected by a given set of test patterns. Combining PPSFP (the Parallel-Pattern Single-Fault Propagation) with the CPT (the Critical Path Tracing) is efficient for reducing computation time for fault simulation. Since PPSFP requires more computation time than CPT, however, reducing computation time for PPSFP is indispensable for faster fault simulation. From that viewpoint, this paper proposes a fast fault simulation algorithm. The algorithm reduces computation time for PPSFP by fault propagation informations given by CPT. Experimental results on ISCAS'85 benchmark circuits demonstrate its efficiency and effectiveness.

Key Words *fault simulation, test design, test patterns, fault coverage, PPSFP, CPT*

1 まえがき

故障シミュレーションとは，故障の存在を想定し，その時の論理回路の動作を調べるものである．故障シミュレーションは，テストパタン系列の故障検出率の評価，テストパタン系列の生成，故障辞書の作成などに用いられ，論理回路の故障検査に不可欠なものである．しかしながら，想定された多くの故障に対してシミュレーションしなければならないため，近年の回路規模の増大に伴う計算量，記憶量の増大が故障シミュレーションの深刻な問題となっている．

代表的な故障シミュレーション法として，同時故障シミュレーション法，並列故障シミュレーション法がある．前者は，正常回路と全ての故障回路を同時にシミュレーションし，正常回路と異なる論理動作する部分のみをトレースする．1テストパタンで検出可能な故障を1度の回路トレースで知ることができるが，必要とされる記憶量が多く，その記憶量の予測が困難という問題がある．一方後者は，計算機のワード単位のビット論理演算機能を利用したシミュレーション法で，(1) 回路の故障を1つに限定し，複数のテストパタンを各ビットに対応させる手法（入力並列故障シミュレーション法：PPSFP(Parallel-Pattern Single-Fault Propagation)），(2) テストパタンを1つに限定し，複数の故障を各ビットに対応させる手法（故障並列故障シミュレーション法）[1], [2] がある．並列故障シミュレーション法は，計算機ワードビット数に並列性が抑えられてしまうものの，同時故障シミュレーション法に比べ，簡素なアルゴリズムで実現でき，高速化，記憶量の予測が容易という利点がある．さらにPPSFP法では，テストパタンの質による差（多くの故障を検出できるか否か）が，シミュレーション時間に影響しにくいという利点を持ち，組合せ回路において特に有効とされている．だが，回路規模の増大に伴う計算量，必要記憶量の増大をいかに削減するかが依然として問題となっている．

本稿では，PPSFP法の処理量を削減した故障シミュレーションの高速化手法を提案する．

本稿は次のように構成される．2章でPPSFP法を用いた従来手法を，文献 [5] の手法を中心に説明し，問題点を示す．3章では提案手法について述べる．4章では提案手法の計算機実験結果を示し，提案手法の有効性を評価する．5章で本稿をまとめ，今後の課題を検討する．

なお，本稿で扱う故障シミュレーションは次のとおりである．

シミュレーション対象回路 ゲートレベルの組合せ回路

入力 ランダムテストパタン，ネットリスト

出力 故障検出率

信号 2値論理

遅延 0遅延

故障モデル 単一縮退故障モデル

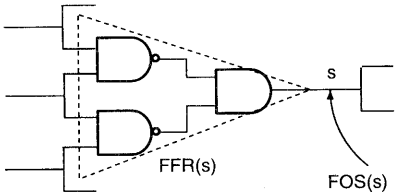

図 1: FFR と FOS

2 PPSFP 法を用いた従来手法

2.1 PPSFP 法の効率化

PPSFP法は，組合せ回路に対して特に高速であることが知られている．だが，想定される故障全てについてPPSFP法を適用することは，多大な処理時間を要する．そのため，PPSFP法の処理の効率化を図った手法がこれまでに提案されている [3],[4],[5].

従来手法の中で高速化を実現している文献 [5] の手法では，次のような処理が行われる．

1. 回路を FFR（Fanout Free Region: 分岐のない信号線で結ばれたゲートの集合，図1）に分割する．
2. 各 FFR 内の故障について，FOS（Fanout Stem: 分岐線の幹）において検出可能な故障を求め（FFR 内シミュレーション），検出可能な故障を FOS の故障として代表化する．
3. 代表化された FOS の故障についてのみ，PPSFP 法を適用する（FOS シミュレーション）．

次節以降，文献 [5] の手法における FFR 内シミュレーション，シミュレーションの実行順序について簡単に述べる．

2.2 FFR 内シミュレーション

FFR に対しては，CPT(Critical Path Tracing) 法 [6] が有効とされている．この理由は，並列法，同時法のような従来の故障シミュレーションアルゴリズムの計算量が $O(n^2)$(n : 信号線数) であるのに対し，FFR に CPT 法を適用した場合には，$O(n)$ で済むという点にある．

文献 [5] では，シミュレーション対象回路を FFR に分割し，分割された FFR 毎に CPT 法を適用する．

以降，CPT 法について説明する．まず，説明に必要な用語を定義する．

Critical, Critical 値： テストパタン t が，信号線 l の v 縮退故障を検出するとき，信号線 l はテストパタン t において Critical 値 v をもつといい，l は Critical であるという．

活性化： ゲート G の入力 i の論理値を反転したとき，ゲート G の出力値が変化するならば，入力 i は活性化されているという．

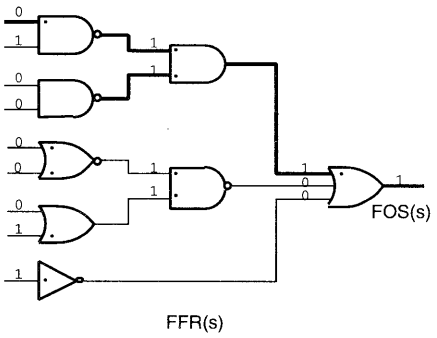

図 2: CPT 法

数値：正論理値，・：活性化入力，太線：Critical 経路

以上の定義を用いて，CPT 法の基本処理は下記の規則によって与えられる．

規則 1： ゲート G の出力線が Critical であるならば，ゲート G の活性化入力は Critical である．

CPT 法は，回路の各信号線の正常値を求めた上で，規則 1 を出力線側から繰り返し適用することにより，その入力テストパタンで検出可能な故障を列挙することができる．図 2に CPT 法の適用例を示す．図で検出される故障は，Critical 経路上の，論理値の反転値の縮退故障すべてである．

活性化の定義からも明らかなように，CPT 法は故障の伝搬経路が単一な場合のみ有効な手法である．従って，文献 [5] の手法では，未検出故障の存在する FFR 内に CPT 法を適用し，FOS で検出可能な故障を FOS の故障として代表化する．FOS の故障は，複数経路故障伝搬する可能性があるため，PPSFP 法を適用する．

2.3 シミュレーションの実行順序

まず，必要用語を定義する．

Dominator： FOS(s) から主出力側へ分岐したすべての信号線が到達する信号線の中で，s に最も近い信号線を s の Dominator という（図 3）．

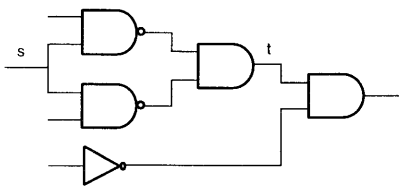

図 3: dominator

t：FOS(s) の Dominator

シミュレーション実行順序は次のようになる．

1. 故障の存在する FFR(s) の中で，最も主入力側の FFR について CPT 法を適用し，FFR 内の故障を FOS(s) の故障として代表化．
2. FOS(s) の故障に対し，PPSFP 法を適用．適用範囲は，1) FOS(s) が Dominator を持つ場合，Dominator まで適用．Dominator で FOS(s) の故障を検出可能な場合，FOS(s) の故障を Dominator の故障として代表化．2) Dominator を持たない場合，主出力線まで適用．
3. シミュレーション対象の故障がなくなったら終了．そうでなければ，1 へ．

各故障に対するシミュレーションは，故障検出が不可能と判明した時点で中止され，別の故障の処理に移る．

2.4 利点と問題点

文献 [5] の手法は，FFR(s) 内の故障を FOS(s) の故障に代表化することにより，PPSFP 法を適用する故障数を削減している．さらに，Dominator を利用することで，代表化された FOS の故障すべてに対して，主出力線まで PPSFP 法を適用する故障は，必ず主出力線までシミュレーションする必要はない．これらにより，PPSFP 法の処理量削減を実現している．

だが，Dominator を持つ FOS 数が少ない回路，あるいは FOS とその Dominator 間の領域が大きい回路の場合，Dominator の利用によるシミュレーションの高速化に対する効果は少ない．このような回路では，PPSFP 法を適用する領域は何度も重複することになり，故障シミュレーションの高速化のためには，この重複領域の PPSFP 法の処理量削減が必要である．

3 提案手法

3.1 基本方針

提案手法の基本方針は，PPSFP 法の処理量削減による故障シミュレーションの高速化である．具体的には，まず，PPSFP 法の前処理として，故障の存在する FFR に CPT 法を適用する．次に，CPT 法の際得られる各 FFR 毎の故障伝搬情報を，Dominator あるいは主出力線までの PPSFP 法適用の際に利用する．この処理により，シミュレーション重複領域での PPSFP 法の処理量削減を狙う．

3.2 基本アルゴリズム

アルゴリズムに必要となる用語を定義する．

Detectability D： $D(s,t)$ は 32 ビット値で，各ビットは，ビット位置に対応するテストパタンの故障検出可能性を表す．すなわち，$D(s,t)$ の各ビット値は，ビット位置に対応するテストパタンが信号線 s の縮退故障を信

表 1: Dominator FOS 数

circuit	# FOS	# Dominator FOS
c432	89	1
c499	59	0
c880	126	30
c1355	251	200
c1908	385	217
c2670	454	310
c3540	579	203
c5315	806	427
c6288	1456	0
c7552	1300	785

Dominator FOS: Dominator FOS を持つ FOS 数

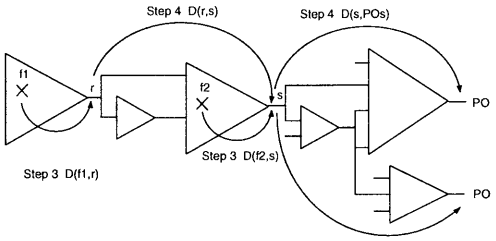

図 4: Step 3, Step 4

号線 t で検出するとき（信号線 s が信号線 t において Critical であるとき）1，検出しないとき 0 となる．

Dominator FOS：FOS(s) で主出力側へ分岐したすべての信号線が，到達する FOS の中で，最も FOS(s) に近い FOS を信号線 s の Dominator FOS という．Dominator FOS を持つ FOS 数は，ISCAS'85 ベンチマーク回路では表 1の通りである．

基本アルゴリズムは下記のようになる．

Step 1. FFR 分割，シミュレーション順の決定．

Step 2. 32 テストパタンを発生し，正論理値シミュレーションの実行．

Step 3. 故障の存在する各 FFR(s) について，CPT 法を適用．各 FOS(s) の故障に代表化．FOS(s) が FOS(r) の Dominator FOS の場合，FOS(r) の故障も FOS(s) の故障に代表化．

Step 4. 代表化された FOS の故障について，Dominator FOS あるいは主出力線まで PPSFP 法を適用．

Step 5. 検出可能故障を求め，故障リスト等の更新．

Step 6. 全故障が検出された，あるいは所定のテストパタン数に達した場合は終了．そうでなければ，Step 2 へ．

Step 3,Step 4 を簡略化した図を 4に示す．提案手法では，PPSFP 法の処理量削減のため，Step 4 に処理の効率化を試みる．次節以降，Step 3，Step 4，Step 5 について詳説する．

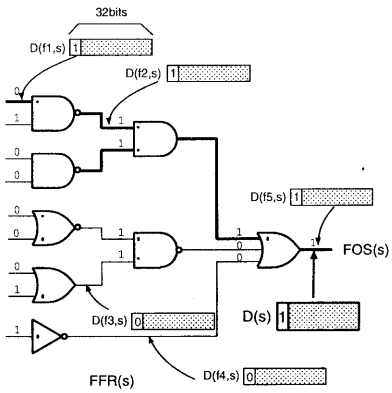

図 5: CPT 法による $D(s)$ の算出

数値は正論理値

3.3 Step 3. FFR 内シミュレーション

Step 3 では，未検出故障 f の存在する FFR(s) に対し，次の処理を行う．

1. CPT 法を適用し，$D(f,s)$ を算出．
2. FFR 内の故障を FOS(s) の故障に代表化．

$D(f,s)$ の算出

未検出故障 f の存在する FFR(s) に対し CPT 法を 32 テストパタン同時に適用し，f の FOS(s) での Detectability $D(f,s)$ を算出する．CPT 法による $D(f,s)$ のビット値格納の様子を図 5に示す．図では，32 ビットの内の 1 ビットについてのみビット値を表記している．

$D(f,s)$ の各ビットは，ビット位置に対応するテストパタンが信号線 s に対して故障信号線 f が Critical であるとき 1，そうでないとき 0 を格納する．このとき，CPT 法適用の際トレースした信号線 l は，いずれも FOS(s) での Detectability $D(l,s)$ を保持している．

FOS(s) の故障へ代表化

FFR 内の未検出故障 f の中で FOS(s) で検出可能な故障を，FOS(s) の故障として代表化する．代表化した故障を $D(s)$ と表す．$D(s)$ は，前節の $D(f,s)$ を利用し，次式で算出される．

$$D(s) = \sum_{i=1}^{N} D(f_i, s)$$

（f_i:FFR(s) 内の未検出故障，N：f_i数）

$D(s)$ の各ビット値は，ビット位置に対応するテストパタンにより，FOS(s) で検出可能な未検出故障が FFR(s) 内に存在する時 1，存在しない場合 0 となる．

3.4 Step 4. FOS シミュレーション (PPSFP 法)

Step 4 では，次の条件を少なくとも 1 つ満たすような FOS(s) を主入力側から順次求め，PPSFP 法を適用する．

1. FFR(s) 内に FOS(s) で検出可能な故障 f が存在．
2. FOS(s) が FOS(r) の Dominator FOS であり，且つ FOS(r) の故障が FOS(s) において検出可能．

これらの条件は次式のビット毎和演算により判定される．第 1 項が条件 1，第 2 項が条件 2 である．第 1 項は前節の式と同様である．

$$D'(s) = \sum_{i=1}^{N} D(f_i, s) + \sum_{i=1}^{M} D(r_i, s)$$

(r_i : FOS(s) を Dominator FOS とする FOS，M : r_i数)

$D'(s)$ が非零のとき，上記の条件を少なくとも 1 つ満たす．$D'(s)$ の 1 となるビットに対応するテストパタンは，信号線 s において検出可能な故障が存在することを表す．

上式により得られる $D'(s)$ を用い，信号線 s から出力側へ PPSFP 法を適用する．その適用範囲は，FOS(s) が信号線 t なる Dominator FOS を持つ場合は t まで，Dominator FOS を持たない場合は主出力線 POs までである．この処理により，$D(s,t)$ あるいは $D(s,POs)$ を求める．

提案手法では，CPT 法で得られる FFR 内の Detectability を故障伝搬情報として利用し，シミュレーション重複領域における PPSFP 法の処理量削減を狙う．最良の場合，FFR の入出力のみのトレースで PPSFP 法を実現可能である．具体的な処理は，PPSFP 法を適用する各 FFR における故障伝搬に応じて，次の 2 つ場合に分けられる．なお，故障信号線を a，PPSFP を適用する FFR を FFR(b) とする．

Case 1 FFR(b) の 1 入力線にのみ a の故障が伝搬する

Case 2 FFR(b) の複数入力線に a の故障が伝搬する

Case 1

この場合 FFR(b) は，信号線 a が Critical となる FFR の入力線を 1 本のみ持つ．この場合の処理は次のようになる．この入力線からゲート毎に PPSFP 法を適用する．その際，Step 3 で FFR(b) に適用した CPT 法でトレースした信号線 l に到達した場合，$D(l,b)$ を利用して $D(a,b)$ を算出する．最良の場合，FFR(b) の入出力のみで $D(a,b)$ を算出することが可能となる．

具体例として，図 6のような部分回路を考える．FOS(s) の Dominator FOS は FOS(v) である．従って，FOS(s) で検出可能な故障が存在する場合，$D(s,v)$ を PPSFP 法により求める必要がある．ここで，FOS(t) に $s2$ に CPT 法が適用されている場合，信号線 t での信号線 s の Detectability $D(s,t)$ は，

$$D(s,t) = D(s) \times D(s2,t)$$

(ビット毎積演算) で算出され，FFR(t) の入出力のみで FOS(t) までの PPSFP 法を実行したことになる．算出された $D(s,t)$ が非零ならば，さらに出力側へ PPSFP 法が適用され，零ならばこの経路に関する PPSFP 法は終了する．図の場合，FFR(u) に対しても同様の処理が可能である．

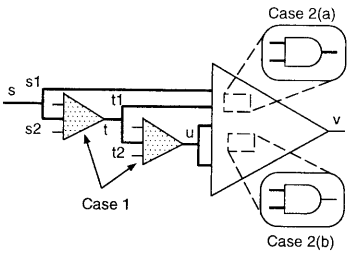

図 6: PPSFP 法処理量の削減

太線は信号線 s の Critical (故障伝搬) 経路

Case 2

この場合 FFR(b) は，信号線 a が Critical となる FFR(b) の入力線が複数存在する．この場合は，従来手法と同じく，これらの入力線から出力側へゲート毎に PPSFP 法を適用する．その際，Critical 経路の合流 (図 6 Case 2(a))，あるいは Critical 経路の減少 (図 6 Case 2(b)) により，FFR(b) 内の Critical 経路数が 1 本になった時，Case 1 と同様の処理を施すことができる．

3.5 Step 5. 検出可能故障の特定

FOS シミュレーションにより，$D(s,POs)$ あるいは $D(s,t)$ (t は s の Dominator FOS) が非零のものについて，FFR(s) 内の未検出故障 f の検出可能性 $D(f,POs)$ を調べる．これは，次のようなビット毎の積演算を主出力側の FOS から適用することにより知ることができる．

- Dominator FOS を持つ FOS(s) の場合

$$D(f,POs) = D(f,s) \times D(s,t) \times D(t,POs)$$

- Dominator FOS を持たない FOS(s) の場合

$$D(f,POs) = D(f,s) \times D(s,POs)$$

算出された $D(f,POs)$ が非零のとき，1 となるビットに対応するテストパタンが故障 f を検出可能なテストパタンとして求められる．

4 計算機実験結果

提案手法を SUN Sparc Station5 上に C 言語を用いて実装し (使用コンパイラは gcc)，故障検出率の算出時間を測定した結果について報告する．

実験では，ネットリストを入力として用い，テストパタンはランダムに発生させた．提案手法を ISCAS'85 ベンチマーク回路に適用し，従来手法の中で高速な手法である文

表 2: 故障検出率算出時間（入力テストパタン数：224，20,000）

回路	テストパタン数 224			テストパタン数 20,000		
	故障検出率 [%]	文献 [5][s]	提案手法 [s]	故障検出率 [%]	文献 [5][s]	提案手法 [s]
c432	97.33	0.15	0.08	99.23	0.27	0.22
c499	95.78	0.17	0.10	98.95	0.35	0.30
c880	94.16	0.30	0.18	100.00 †	0.32	0.28
c1355	91.04	0.47	0.35	92.49	0.92	0.75
c1908	80.95	0.70	0.58	99.52	1.23	1.08
c2670	77.50	0.88	0.62	84.64	3.75	4.40
c3540	88.01	1.43	1.15	95.97	3.37	3.10
c5315	96.21	1.75	1.25	98.80	3.68	3.08
c6288	99.56	2.98	2.90	99.56	4.45	4.20
c7552	90.45	2.53	1.85	94.72	7.42	6.87

†入力テストパタン数：10,016

献 [5] の手法と比較した．結果を表 2に示す．入力テストパタン数は 224，20,000 である．

表より，提案手法は文献 [5] の手法に比較して，故障検出率算出時間をテストパタン数 224 の場合，平均 28%，テストパタン数 20,000 の場合，平均 10% 短縮していることがわかる．

実験では，故障検出率の算出時間の短縮を目的としているため，あるテストパタンで検出可能と判明した故障は，以降のテストパタンのシミュレーション対象から除外するという高速化が図られる．これを故障のドロップという．各 32 テストパタン毎の検出故障数は，最初の 32 テストパタンで最も多く，テストが進むにつれ急速に減少する．従って，提案手法での CPT 法による FFR 内の故障伝搬情報量も，テストが進むにつれ減少することになる．表 2 において，20,000 テストパタンの場合のシミュレーション時間が，224 テストパタンの場合に比べ高速化の割合が低いのは，故障伝搬情報量の減少によるものと考えられる．

以上の結果より，提案手法は，従来手法に比べ高速に故障検出率を算出することが確かめられた．特に，テスト初期の高速化において有効であることが確かめられた．

5 むすび

本稿では，故障シミュレーション法の高速化手法を提案した．提案手法は，CPT 法により得られる FFR 毎の故障伝搬情報を利用することにより，PPSFP 法の処理量を削減することができる．計算機実験の結果，ISCAS'85 ベンチマーク回路に対し，提案手法は従来手法よりもテストパタン系列の故障検出率算出時間を平均 28%（テストパタン数：224）短縮できることを確認した．

提案手法は，故障検出率の算出だけでなく，故障診断等，回路の正確な故障検査を要する場合，その特徴を生かすことができる．従って，今後の課題としては，提案手法の順序回路への拡張とともに，提案手法を利用した故障解析，テストコンパクション等への検討を行う．

謝辞

本研究を進めるに当たり，貴重な御指導，御助言を賜りました本学佐藤政生助教授に深謝致します．

参考文献

[1] T. M. Niermann, W. T. Cheng, and J. H. Patel, "PROOFS: A fast, memory efficient sequential circuit fault simulator," *IEEE Trans. on Computer-Aided Design*, vol.11, no.2, pp. 198–207, Feb. 1992.

[2] H. -K. Lee, and D. -S. Ha, "New methods of improving parallel fault simulation in synchronous sequential circuits," in *Proc. ICCAD-93*, pp.10–17, 1993.

[3] F. Maamari and J. Rajski, "A method of fault simulation based on stem regions," *IEEE Trans. on Computer-Aided Design*, vol.9, no.2, pp. 212–220, Feb. 1990.

[4] K. J. Antreich and M. H. Schulz, "Accelerated fault simulation and fault grading in combinational circuits," *IEEE Trans. on Computer-Aided Design*, vol.CAD-6, no.5, pp. 704–712, Sep. 1987.

[5] H. -K. Lee and D. -S. Ha, "An efficient, forward fault simulation algorithm based on the parallel pattern single fault propagation," in *Proc. IEEE International Test Conference*, pp. 946–955, 1991.

[6] M. Abramovici, P. R. Menon, and D. T. Miller, "Critical path tracing - An alternative to fault simulation,"in *Proc. 20th DAC*, pp. 214–220, 1983.