設計自動化49－12
(1989.10.25)

バックトラック処理不要な組合せ回路テスト生成手法

A BACKTRACKLESS TEST GENERATION METHOD FOR COMBINATIONAL CIRCUITS

池田　光二　　畠山　一実　　林　照峯
Mitsuji Ikeda　　Kazumi Hatayama　　Terumine Hayashi

(株)日立製作所　日立研究所
Hitachi Research Laboratory, Hitachi, Ltd.

あらまし　実数型のシミュレーションを利用した組合せ回路のテストパターン生成手法を提案する．実数型のシミュレーションでは論理値の代わりに0から1までの実数を用いる．この手法はバックトラックの発生を抑止することができるという特徴をもっており，複雑なバックトラック処理を排除することができる．ところで，本手法と同様の目的をもつ手法としてはChengとAgrawalによって提案された手法（C－A法）がある．しかし，この手法には，ある種の回路構造に対してはテストパターンを誘導できないという問題点がある．われわれはC－A法の問題点を対策した改良C－A法も提案した．上記の各手法に対するISCAS'85ベンチマークデータを用いた評価実験により，本手法が最も高性能であることを確認した．

Abstract　This paper presents a test generation method using 'real number simulation'. Real number simulation is an extended logic simulation which uses real numbers between 0 and 1 instead of logic values 0 and 1. This method is appropriate to generate test patterns for VLSI logic circuits, because it does not require any backtracks. This paper also gives an improvement to threshold simulation approach. Experimental results show that the real number simulation is a promising approach to the test generation of VLSI logic circuits.

1．はじめに

近年の半導体技術の進歩に伴い，論理回路の大規模化・多様化が促進され，設計期間と設計工数の低減がますます重要となってきた．テストパターン設計においても，人手工数低減のためには，テストパターンを自動で生成するテスト設計自動化システムが必要である．この目的に沿って，従来からいくつかのテスト生成手法が提案されている[1)~3)]．

最近では，5kゲート以下の中規模の組合せ論理回路に対してはほぼ完全なテストパターン集合を高速に提供するアルゴリズムが提案されているが[4)]，これが10kゲートを超える組合せ回路に対して実用可能かどうかは不明である．一方，ランダムパターンと故障シミュレーションを利用して高速にテストパターン集合を生成するテスト方法もよく用いられている．この方法は，シミュレーション用ハードウェアや並列化による処理高速化は可能であるが，対象故障集合が小さくなると効率が低下するため，大規模な論理回路においては高い検出率を得るために非常に多くのランダムパターンをシミュレートしなければならない．そこで，対象故障集合が小さくなると，それらの故障に対するテストパターンを求めるために決定性アルゴリズムを併用する必要がある．このため，この方法は全体システムの処理高速化という面では必ずしも適当でない．

本研究では，特定の故障に対して，シミュレーションを利用して誘導的に(有向探索的に)テスト生成を行うというアプローチを検討した．シミュレーションを用いることの利点としては並列化が比較的容易に実現できることが挙げられる．現に，故障シミュレーションマシンを利用して，高速にテストパターンを生成する手法が提案されている[5)]．本論文では，論理値を0から1までの実数で表現し，乗算を基調とした演算を行う拡張論理シミュレーションと呼ぶシミュレーションを用いたテスト生成手法を提案する．本手法の特徴は，実数値を対象とするシミュレーションにより回路の状態の微妙な変化が表現できることを利用して，有向探

索的なテスト生成を行うことである．

以下では，まず本手法によるテスト生成方法について述べ，次に本手法と同様，実数型のシミュレーションを利用して有向探索的にテスト生成を行うChengとAgravalが提案した手法[6]（以降Ｃ－Ａ法と呼ぶ）の問題点およびそれを解決した改良型Ｃ－Ａ法について紹介し，最後に，ＩＳＣＡＳ'85 のベンチマーク回路を対象にして，本手法，Ｃ－Ａ法，改良Ｃ－Ａ法に対する性能評価結果を示す．

2．拡張論理シミュレーションとテスト生成方法

2.1 拡張論理シミュレーション

特定の故障に対して，入力ベクトルＶがテストパターンになるかどうかは，正常回路と故障回路とで入力ベクトルＶに対して論理シミュレーションを行い，出力結果を比較すればよい．しかし，この方法では，入力ベクトルＶがテストパターンでないとき，入力ベクトルＶをどう変化すればテストパターンに近づくかを判断することができない．そこで，論理値の代わりに０から１までの実数値を用いたシミュレーションを導入することを検討した．実数値を用いたシミュレーションを実現するため，ＡＮＤ素子，ＯＲ素子，ＮＯＴ素子の各演算を以下に示すように規定する．なお，入力をx_i(１入力のときはx)，出力をyとする．

(1) ＡＮＤ素子

$$y = \prod_i x_i$$

(2) ＯＲ素子

$$y = 1 - \prod_i (1 - x_i)$$

(3) ＮＯＴ素子

$$y = 1 - x$$

上記の各演算は実数値を対象にしているが，入力値を０および１に限定すると通常の論理演算になるため，ここでは上記の演算を拡張論理演算と呼ぶ．また，拡張論理演算を用いたシミュレーションを拡張論理シミュレーションと呼ぶ．拡張論理シミュレーションでは，外部入力値として０または１を与えても単に通常の論理シミュレーションを実行するだけであり，外部入力値としては０および１に対応してそれぞれ，$0.0+\varepsilon$，$1.0-\varepsilon$ (ただし，$0<\varepsilon<0.5$) を与える必要がある．これにより，図１(a),(b),(c)に示した出力論理が０となるＡＮＤ素子の拡張論理シミュレーションによる出力値はそれぞれ，ε^3，$(1-\varepsilon)\varepsilon^2$，$(1-\varepsilon)^2\varepsilon$ となる．$\varepsilon<(1-\varepsilon)$より，

$$0 < \varepsilon^3 < (1-\varepsilon)\varepsilon^2 < (1-\varepsilon)^2\varepsilon < 1$$

という関係が成立する．図１(a),(b),(c) のＡＮＤ素子の出力論理を１にするために変更が必要な入力数はそれぞれ３，２，１であり，拡張論理シミュレーションが論理値より多くの情報を提供していることが判る．

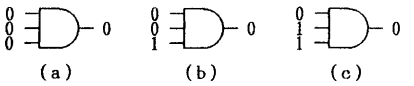


図１　出力論理が０となる３入力ＡＮＤ素子

なお，拡張論理シミュレーションの実数値は論理値と強い対応関係はないが，対応関係を強くする方法としては次節に示す演算結果の補正を行う方法がある．

2.2 拡張論理演算の補正

実数値を用いるために計算結果が曖昧になることがある．たとえば，16入力ＡＮＤ素子の入力値がすべて0.9（論理値は１）である場合，出力論理値は１だが，拡張論理演算による出力値は0.185 程度になってしまう．このことは，拡張論理演算による実数値が必ずしも論理値と対応していないことを示しており，拡張論理シミュレーションによるテスト生成の性能低下の可能性を意味する．

そこで，このような矛盾が起こらないようにするため，拡張論理演算結果の補正を考える．図２は補正関数の例を示している．図２において，αおよびβは，$0<\alpha<\beta<0.5$ を満足するものとする．図２の補正関数は次式で表される．

(1) 論理値が０のとき

$$y = x \qquad (0 \leqq x \leqq \alpha),$$

$$y = (\beta-\alpha)(x-\alpha)/(1-\alpha)+\alpha \qquad (\alpha \leqq x \leqq 1).$$

(2) 論理値が１のとき

$$y = (\beta-\alpha)x/(1-\alpha)+1-\beta \qquad (0 \leqq x \leqq 1-\alpha),$$

$$y = x \qquad (1-\alpha \leqq x \leqq 1).$$

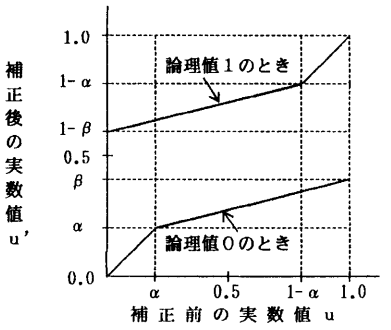


図２　補正関数

この補正により，論理値が１のときには実数値が 0.5 以上，論理値が０のときには実数値が 0.5以下となり，論理値と実数値の対応関係の矛盾が回避できる．

なお，上記の補正は，並列化の妨げとなる判定文を用いないで加減乗除および絶対値演算のみで実現することができる．論理値をv，補正前の実数値をuとす

ると，補正後の実数値u'は，次式で導出される．

$$u' = |v-\{(1+\beta-2\alpha)\Delta v+(1-\beta)\times(\alpha-|\Delta v-\alpha|)\}/2(1-\alpha)|.$$

ただし，$\Delta v = |v-u|$である．

2.3　テスト生成方法

開発手法では，特定の故障を検出するテストパターンを求めるため，2.1節(または2.2節)で定義した拡張論理シミュレーションを用いる．

論理回路Cにおいて故障fを考えた場合の出力ピンyに対する入力ベクトルIのコストKを以下のように定義する．なお，Iの各要素を0.0+εまたは1.0−εに対応させたベクトルをI^とする．

[定義]　故障のない論理回路Cに入力ベクトルI^を与えて拡張論理シミュレーションを行ったときの出力ピンyの値をv，故障fをもつ論理回路Cに入力ベクトルI^を与えて拡張論理シミュレーションを行ったときの出力ピンyの値をuとする．このとき，出力ピンyに対する入力ベクトルIのコストKを次式で定義する．

$$K = \frac{1}{|v-u|}.$$

ただし，論理回路，故障，出力ピン，入力ベクトルのうち明らかなものがあれば，それらを省略することがある．

論理回路（入力数n，出力数m）と故障が与えられたとき，拡張論理シミュレーションを用いたテスト生成は次のように行う．

[特定の故障に対するテスト生成手続き]

S1. 変化入力ピン番号i，および隣接ベクトル番号pをそれぞれi＝0，p＝1に初期設定する．

S2. 故障fに対して初期入力ベクトルIを1つ選び，拡張論理シミュレーションを行って，すべての外部出力ピンに対してコストKj（1≦j≦m)を求める．また，同時に通常の論理シミュレーションを行い，Iがテストパターンかどうか判定する．Iがテストパターンならば処理終了．

S3. i＝(i mod n)＋1とする．

S4. 入力ベクトルIの第i番目の外部入力ピンの論理値を反転させたベクトルI'に対して拡張論理シミュレーションを行って，すべての外部出力ピンにおけるコストKj'(1≦j≦m)を求める．また，同時に通常の論理シミュレーションを行い，I'がテストパターンかどうか判定する．I'がテストパターンならば処理終了．

S5. ΣKj>ΣKj'ならば，I'をI，ΣKj'をΣKj，p＝1としてS3へ．ΣKj<ΣKj'ならば，p＝p＋1とする．ここでp<nならばS3へ．p≧nならば，故障fのテスト生成失敗として処理終了．

図3を用いて本手法によるテストパターン生成の例

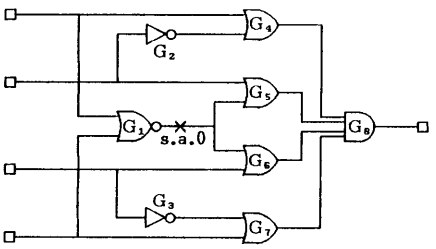


図3　テスト生成の例題回路

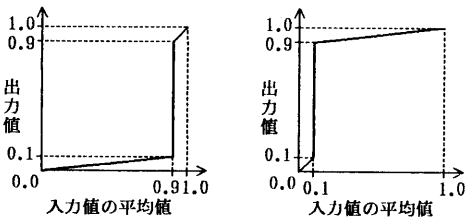
を示す．図3において，NOR素子G_1の出力には0縮退故障がある．まず，初期入力ベクトルIとして，(1，0，1，0)を与える．外部入力値としては，0および1に対応してそれぞれ，0.01および0.99を与える．通常の論理シミュレーションにより，Iはテストパターンでないことが判るため，Iに対するコストを求め，5181を得る．次に，第1入力ピンの論理値を変えたベクトルI_1＝(0，0，1，0)に対して同様の操作を行い，コスト52.3を得る．I_1の方がコストが小さいため，次にI_1の第2入力ピンの論理値を変えたベクトルI_2＝(0，1，1，0)に対して同様の操作を行う．I_2もテストパターンではなく，また，コストは125000となるため，I_2を探索過程から除外する．引き続いてI_1の第3入力ピンの論理値を変えたベクトルI_3＝(0，0，0，0)について同様の操作を行い，I_3をテストパターンとして処理を終了する．

なお，演算を補正した場合も上記と同様にテストパターンを求めることができる．

3．C－A法とその改良

3.1　C－A法[6)]

本手法と同じ目的をもつ手法としてC－A法がある．この方法では，テストパターンを有向探索的に求めるため，しきい値シミュレーションと呼ぶ実数値を用いた変形の故障シミュレーションを行っている．しきい値シミュレーションでは，AND素子およびOR素子に対して，入力値の平均値をそれぞれAND素子用しきい値関数およびOR素子用しきい値関数（図4参照）



(a)　AND素子用　　(b)　OR素子用

図4　しきい値関数

と呼ぶ関数で写像した値を出力値とする．ＮＯＴ素子は入力値を1.0 から引いた値を出力値とする．図４のしきい値関数は論理値が１の場合は0.9 以上，論理値が０の場合は0.1 以下に対応するように定義された単調増加関数である．ただし，素子の入力数が多くなると対応関係が成立しないことがあるため，回路に含まれる素子の入力数をあらかじめ求めておき，それに応じてしきい値関数を定義している．図４のしきい値関数は素子の入力数の最大値が８以下の回路に対するものである．これを用いると，前述の図１のＡＮＤ素子(a),(b),(c)の出力値はそれぞれ，0.0，0.037，0.074となる．

しきい値シミュレーションを用いたテスト生成手続きは本手法とほぼ同様である．ただし，しきい値シミュレーションで用いる実数値は論理値と対応しており，論理値が異なれば0.8 以上の差が生じることから，通常の論理シミュレーションを用いないでテストパターンかどうかの判定ができる．なお，本手法におけるコストの定義において，シミュレーション方法を代えることによりＣ－Ａ法に関するコストを定義する．

3.2　しきい値シミュレーションの改良

簡単な回路でもＣ－Ａ法ではテストパターンをうまく導出できない場合がある．図５は，３入力ＡＮＤ素子１つからなる３入力１出力の回路である．図５にお

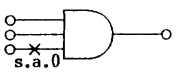


図５　ＣＡ法で検出困難な故障例

いて，ＡＮＤ素子の第３入力の０縮退故障を検出するテストパターンを，入力ベクトル(0,0,1) から開始して求める手順を考える．入力ベクトル(0,0,1) および隣接ベクトル(1,0,1)，(0,1,1)，(0,0,0) のコストはそれぞれ，27.0，27.0，27.0，∞となる．これより，最初の入力ベクトルよりコストの減少した隣接ベクトルが存在しないためテストパターンを導出しないままテスト生成処理を終了する．しかし，実際はテストパターン(1,1,1) が存在するためテスト生成が失敗したことになる．この原因はしきい値関数にある．すなわち，隣接ベクトル(1,0,1) は初期ベクトル(0,0,1) よりテストパターン(1,1,1) に接近したことになるからコストが減少するようにしきい値関数を定義するべきである．このように，しきい値関数は故障情報がどの程度伝播しているかを表現するには不十分である．そこで，故障情報の伝播状況をより強く表現するため，しきい値関数を直線ではなく曲線で定義することを考案した．ＡＮＤ素子は論理値が１に近づくほど，またＯＲ素子は論理値が０に近づくほど故障情報が伝播され易くなる．したがって，ＡＮＤ素子用しきい値関数は下に凸な単調増加関数，ＯＲ素子用しきい値関数は上に凸な単調増加関数に改良することにより上記の目的は達成できる．改良しきい値関数の１つの例を次式に示す．

(1)ＡＮＤ素子用改良しきい値関数（図６(a)参照）

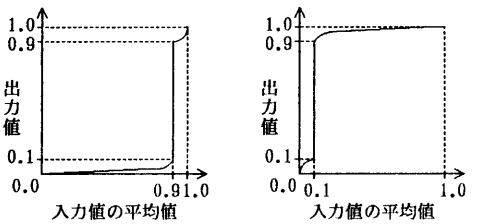
$y = x^2 / 8.1$　　$(0 \leqq x < 0.9)$,

$y = (x - 0.9)^2 / 0.1 + 0.9$　　$(0.9 \leqq x < 1)$.

(2)ＯＲ素子用改良しきい値関数（図６(b)参照）

$y = \sqrt{x / 10}$　　$(0 \leqq x < 0.1)$,

$y = \sqrt{(x - 0.1) / 90} + 0.9$　　$(0.1 < x \leqq 1)$.



(a)　ＡＮＤ素子用　　(b)　ＯＲ素子用

図６　改良しきい値関数

一方，本手法で用いる拡張論理シミュレーションでは上記の問題が生じないことを示す．前述した図５のＡＮＤ素子の第３入力ピンの故障／非故障の差をΔVで表し，他の入力ピンの値が(0,0)の場合と(0,1)の場合を比較する．(0,0) の場合はＡＮＤ素子の出力の故障／非故障の差は$\varepsilon^2 \Delta V$であり，一方，(0,1)の場合は$(1-\varepsilon)\varepsilon \Delta V$である．$\varepsilon^2 \Delta V < (1-\varepsilon)\varepsilon \Delta V$であるため，(0,1) の方が故障／非故障の差が大きくなり，コストが減少する．

４．性能評価結果

本手法（演算補正あり，演算補正なし），Ｃ－Ａ法，改良Ｃ－Ａ法のそれぞれに対して性能評価を行った．本手法における外部入力値としては，論理値０に対して0.01，論理値１に対して0.99を与えた．さらに，演算補正ありの場合の補正関数のパラメータα，βをそれぞれ，0.1，0.2に設定した．また，評価用データとしては，ＩＳＣＡＳ'85 のベンチマークデータ10品種を用いた．なお，実験の公正さを期すため，同一のテスト生成手順を用いた†．以下にテスト生成手順を示す．

---

†文献[６]に記されているテスト生成手順は回路内部の全故障を扱う上で効率の良い手法であるが，ここでは特定故障のテスト生成能力を判定する意味で適切と思われる方法を用いた．

［テスト生成手順］

S1．テスト生成対象の回路（入力数ｎ，出力数ｍ）に対して故障リストを作成する．

S2．故障リストより，未検出未試行故障を１つ取り出す（ｆとする）．未検出未試行故障がなければ対象回路のテスト生成終了．

S3．S2で選択した故障ｆに対して2.3 節で示した特定故障のテスト生成を行う．このとき，初期入力ベクトルはランダムパターンで与える．Ｃ－Ａ法および改良Ｃ－Ａ法に関しては通常の論理シミュレーションを行わず，全出力ピンのうち１つのコストが1.25以下になればテスト生成成功とする．

S4．テストパターンが生成されれば，そのテストパターンを用いて同時検出・故障ドロップを行う．

S5．S2へ．

ＩＳＣＡＳ'85 のベンチマークデータには，１素子の入力数が９の回路が含まれている．第３章で説明したように図４（図６も同様）のしきい値関数は入力数が８以下の素子からなる回路を対象としたものであり，本来はしきい値関数を変更しなければならない．しかし，入力数が９の場合に論理値と実数値の対応関係が成立しない可能性はほぼ０に等しいため，図４および図６で示したしきい値関数を用いることにした．

表１に性能評価結果を示す．なお，表１の未検出故障には冗長故障を含んでいない．表１より，改良Ｃ－Ａ法はＣ－Ａ法と比べて，Ｃ1355以外のすべての回路で未検出故障の低減を達成したことが判る．この結果より，3.2 節の検討内容はほぼ裏付けられた．また，本手法に関しては，演算補正したものとしないものとを比較すると演算補正しないものの方が故障検出能力が高かった．このことは，シミュレーションの結果が必ずしも論理値と直接対応する必要がないことを意味している．表１より，演算補正のない本手法が上記４つのテスト生成手法のなかで，最も検出能力が高い手法であるといえる．

５．おわりに

シミュレーションを用いた有向探索的なテスト生成方法を開発し，性能評価実験により本手法の検出能力の有効性を確認した．また，Ｃ－Ａ法の問題点を分析することにより，テストパターンを有向探索する効果的なシミュレーション方法の方向付けをした．今後は，処理速度の向上を図るため，並列化技法等の検討を進める予定である．

参考文献

1) Roth, J.P., Bouricius, W.G., and Schneider, P.T. : Programmed Algorithms to Compute Tests to Detect and Distinguish between Failures in Logic Circuits , IEEE Trans. Electron. Comput., EC-16, 5, pp.567-579(1967).

2) Goel, P.: An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits , IEEE Trans. Comput., C-30, 3, pp.215-222(1981).

3) Fujiwara, H. and Shimono, T. : On the Acceleration of Test Generation Algorithms, IEEE Trans. Comput., C-32, 12, pp.1137-1144 (1983).

4) Schulz, M.H., Trischler, E. and Sarfert, T.M.: SOCRATES: A Highly Efficient Automatic Test Pattern Generation System, IEEE Trans. CAD, Vol. 7, No. 1, pp. 126-137 (1988).

5) Hirose, F., Takayama, K. and Kawato, N. : A Method to Generate Tests for Combinational Logic Circuits Using an Ultrahigh-Speed Logic Simulator, Proc. Int. Test Conf. 1988, pp. 102-107 (1988).

6) Cheng, K-T. and Agrawal, V.D.: A Simulation -Based Directed-Search Method for Test Generation, Proc. Int. Conf. Comp. Des. (ICCD'87), pp. 48-51(1987).

表１　テスト生成評価結果

		本手法（補正なし）		本手法（補正あり）		C-A法		改良C-A法	
回路名	仮定故障数†	未検出故障数	処理時間	未検出故障数	処理時間	未検出故障数	処理時間	未検出故障数	処理時間
C432	520	0	0.59	1	0.67	19	0.68	3	0.69
C499	750	3	0.90	11	0.95	11	0.84	10	1.03
C880	942	0	1.18	0	1.53	29	2.06	6	1.91
C1355	1566	2	3.49	50	4.16	13	3.61	36	3.38
C1908	1870	4	5.60	6	5.22	117	5.98	41	5.51
C2670	2630	13	44.15	16	73.81	230	76.69	14	45.00
C3540	3291	10	21.25	19	29.10	99	24.48	33	23.50
C5315	5291	12	58.22	23	78.41	78	58.88	54	60.85
C6288	7710	4	15.94	6	17.72	10	12.40	0	12.74
C7552	7419	44	171.71	49	254.54	337	274.06	148	232.11
平　均	3199	9.2	32.30	18.1	46.61	94.3	45.97	34.5	38.67

† 仮定故障は検出可能なもののみ