ISSN 2186-5647

2-9

故障検出の必要十分条件を用いた複数目標故障テスト生成の高速化手法

日大生産工(院)　○澤田 太生　　日大生産工　細川 利典
京産大　吉村 正義　　日大生産工　新井 雅之

1. まえがき

超大規模集積回路（Very Large Scale Integrated circuits：VLSI）は，社会のあらゆる分野で利用されており，製造されたVLSIに故障が存在しないことを保証しなければならない．そのため，VLSIのテスト技術は非常に重要である．近年の半導体集積技術の発展により，VLSIは大規模化かつ複雑化しており[1]，テストパターン数の増加に伴うテスト実行時間およびテストコストの増大が深刻な課題となっている．テストコストはテストパターン数に比例するため，テスト品質を維持したままテストパターン数を削減するテスト圧縮技術の重要性が高まっている．

テスト圧縮技術は，既存のテスト集合に対して冗長なテストパターンを削減したり，テストキューブを併合したりする静的圧縮手法[2]-[4]と，テスト生成の過程で多くの故障を同時に検出するテストパターンを生成する動的圧縮手法[5]-[7]に大別される．動的圧縮手法の一つとして，複数の故障を同時にテスト生成の対象とする複数目標故障テスト生成（Multiple Target Test Generation: MTTG）と呼ばれる手法が提案されている[5]-[7]. MTTGは，故障間の共通する励起条件および伝搬条件を利用することで，よりコンパクトなテスト集合が得られる利点を持つ．しかしながら，従来のMTTGでは，故障箇所からの推移的ファンアウトコーン（Transitive Fan-Out Cone: TFOC）を対象として故障回路モデルを構築し，正常回路モデルとの比較によりテスト生成を行うため，大規模回路では故障間の依存関係を考慮した探索空間が膨大となり，テスト生成時間の増大が問題となっていた．

本論文では，この課題を解決するために，文献[11]で示された故障検出の必要十分条件抽出法を基礎とする．特に，ドントケア（X）判定[9]を抽出過程に統合的に導入し，不要な条件を早期に除去することで抽出処理を効率化する．抽出された必要十分条件をMTTGに適用することにより，高速なテスト生成を実現する手法を提案する．提案手法では，故障検出の成立条件を十分条件の概念[8]に基づいて定義し，X判定を用いて不要な条件を排除することで必要十分条件の抽出を効率的に行う．さらに，抽出した条件を論理合成によって論理圧縮し，冗長な項を削減した上で，圧縮済みの必要十分条件をMTTGに適用する．これにより，従来のTFOCベースのMTTGに比較して探索空間を大幅に縮小し，テスト生成時間の短縮と効率的な動的圧縮の実現を目指す．

以下，2章では本論文で使用する用語および基本概念について説明する．3章では故障検出条件生成アルゴリズムについて説明する．4章では必要十分条件を用いた複数目標故障テスト生成アルゴリズムについて説明する．5章ではベンチマーク回路を用いた実験結果を示す．6章では本論文のまとめおよび今後の課題について述べる．

2. 前提知識

本章では，本論文で使用する用語および基本概念について説明する．　2.1節では擬似ブール最適化（Pseudo-Boolean Optimization：PBO）ベースのテスト生成手法の概要について説明し，2.2節では故障検出条件の表現と簡単化について説明する．2.3節ではX判定について説明する．

2.1　PBOベースのテスト生成手法

PBOは，ブール変数を用いた線形制約式により論理回路の動作と最適化目標を同時に記述可能な枠組みであり[10]，大規模なテスト生成問題の表現に有効である．

PBOでは，ブール変数を$x_i \in \{0,1\}$とし，式(1)のような一般形で問題を定式化する．

$$maximaize(or\ minimize)F = \sum_{i=1}^{m} c_i x_i \qquad (1)$$

$$subject\ to \sum_{i=1}^{m} a_{ij} x_i \geq b_j, j = 1,2,\dots,k$$

ここで，係数a_{ij}, b_jおよび目的関数の重みc_iは整数定数であり，制約式は論理ゲートの機能や信号間の関係を表す．このような形式に変換することで，論理構造とテスト目的を統一的に扱うことが可能となる．

PBOベースのテスト生成では，回路中の信号値や故障の伝搬をブール変数で表現し，各ゲートの機能を線形不等式として定式化する．さらに，故障の励起条件および外部出力への伝搬条件を制約式に含め，故障検出を満たす割当てを目的関数の最適化問題として解く．

複数の故障集合$F = \{f_1, f_2, \dots, f_n\}$に対しては，各故障の検出を示す変数$d_{f_i}$を導入し，目的関数を式(2)のように設定する．

$$F = d_{f_0} + d_{f_1} + \cdots + d_{f_n} \qquad (2)$$

この目的関数を最大化することで，同一テストパターンで検出可能な故障数を最大化することができる．ただし，本論文ではテスト生成高速化のために必ず検出しなければならない故障である1次故障d_{f_i}を一つ選

An Acceleration Method for Multiple Target Test Generation Using Necessary and Sufficient Conditions for Fault Detection

Tao SAWADA, Toshinori HOSOKAWA,
Masayoshi YOSHIMURA and Masayuki ARAI

択し，$d_{f_i} \geq 1$とし，目的関数には含めない．また，PBOソルバは制約充足と最適化探索を同時に扱うことができるため，MTTG を効率的に実行できる．

1.2 故障検出条件の表現と簡単化

故障を検出するためには，故障の励起条件と伝搬条件を同時に満たす入力割当てを求める必要がある．このとき，故障の影響が伝搬する範囲は限られており，故障箇所から外部出力までのTFOC内に存在する信号が対象となる．

特に，故障を挿入した際，その影響が伝搬する領域と，それ以外の正常動作領域との境界に位置するゲートを境界ゲート[11]と呼ぶ．境界ゲートは，故障値と正常値の両方に影響を受ける位置にあり，故障検出条件を表現する上で最小の論理単位となる．

大規模回路全体を対象に故障検出条件を構築する場合，膨大な数の信号やゲートが含まれ，式の規模が増大する．しかしながら，故障の励起および伝搬に直接関与するのは境界ゲートの出力信号線の値であり，この領域の信号値の組合せによって条件を記述することで，故障検出条件を大幅に簡略化できる．

この考え方に基づき，故障検出条件を必要条件と十分条件に分解して表現する手法が提案されている[8][11]．故障fの検出条件を表す論理式をψ_fとすると，次のように定義される．

・十分条件： 任意の割当て$\alpha \supseteq \beta$が常にψ_fを満たす場合，αの部分割当てβを故障fの十分条件と呼ぶ．これは，「βの条件を満たす入力であれば必ず故障が検出される」ことを意味する．

・必要条件： あるリテラルlについて式$(\psi_f \wedge \neg l)$が充足不能である場合，すなわち「lを偽にすると故障検出条件ψ_fが成立しなくなる」場合，lは故障検出に必須な条件である．このようなリテラルの集合を必要条件と呼ぶ．

定義より両者の関係は「必要条件 ⊇ 故障検出条件 ⊇ 十分条件」で表される．

さらに，境界ゲートを基点として十分条件を構成したものを拡張テストキューブと呼ぶ[11]．拡張テストキューブは，外部入力だけでなく，故障伝搬経路上の境界ゲート出力信号線に対する割当てを含む論理積である．

拡張テストキューブの集合$\{C_i | C_i = \wedge_j l_{ij}\}$を用いると，故障の検出条件は式(3)で表される．

$$\psi_g \wedge \bigvee_i \bigwedge_j l_{ij} \qquad (3)$$

ここで，ψ_gは正常回路の論理式であり，l_{ij}は境界ゲート出力信号線の値に対応するリテラルを表す．

2.3 X（ドントケア）判定

故障検出条件の抽出を効率的に抽出するためには信号値が「0」,「1」,「X」の3値を取る三値論理解析が有効である．ここで「X」は「0」と「1」のどちらでも故障検出に影響を及ぼさない値を表す．X判定とは，信号線がXであっても故障検出に影響を及ぼさない場合，その信号を必要十分条件の式から場外する手法である．すなわち，信号線sに対して$s = 0$および$s = 1$のいずれの場合でも，故障の影響が外部出力に伝搬するならば，その信号は故障検出に支配的でないとみなされる．この判定により，必要十分条件の導出過程で不要なリテラルを削減でき，論理式のサイズと抽出時間を同時に短縮できる．

3. 故障検出条件生成アルゴリズム

本章では，故障検出条件を抽出するアルゴリズムについて説明する．3.1節では故障検出の必要十分条件抽出アルゴリズムについて説明する．3.2節では故障検出条件の圧縮について説明する．

3.1 故障検出の必要十分条件抽出アルゴリズム

故障検出の必要十分条件を抽出する処理をAlgorithm 1 Extract_Conditionsに示す．本アルゴリズムは，入力として回路C，拡張テストキューブの抽出上限数n_EC，および故障集合Fを受け取り，各故障に対する故障検出条件集合$FDCset$を導出するものである．出力は故障検出条件集合$FDCset$，テスト不能故障集合$UFset$である．

式(3)に示したように，故障検出条件を表現するためには拡張テストキューブを求める必要がある．しかしながら，故障によっては拡張テストキューブの数が膨大となる場合があるため[11]，その抽出数をn_EC以下に制限する．

以下では，Algorithm 1の具体的な処理手順について説明する．

行1では，$FDCset$および$UFset$を空集合として初期化する．行2～21では，故障集合Fの各故障f_iに対して処理を行う．具体的には，各故障の故障検出条件を算出およびテスト不能故障の判定を行う．行3では，拡張テストキューブの抽出数ec_iを0に初期化し，f_iの故障検出条件FDC_iおよびテストパターンtをNULLに設定する．行4～15では，故障f_iに対して，拡張テストキューブがこれ以上得られなくなるか，その数がn_ECに達するまで抽出処理を繰返す．行5では，故障f_iを検出するテストパターンtを求める．また，同一の拡張テストキューブが再び抽出されることを防ぐため，故障検出条件FDC_iの否定を制約条件として追加する．行6～14で

Algorithm 1: Extract_Conditions
Input : Circuit C, Limit for number of Extended test Cube n_EC, Fault set F
Output : Fault Detection Condition Set $FDCset$, Untestable Fault set $UFset$

```
1.  FDCset = ∅ ; UFset = ∅ ;
2.  for each fi in F do
3.      eci = 0; FDCi = NULL; t = NULL;
4.      repeat
5.          t = PBO_STTG(C, fi, FDCi);
6.          if (t != NULL) then
7.              fdc = XID(C, fi, t);
8.              if (FDCi == NULL) then
9.                  FDCi = fdc;
10.             else
11.                 FDCi = FDCi ∨ fdc;
12.             end if
13.             eci++;
14          end if
15.     until (t == NULL || eci == n_EC);
16.     if (eci == 0) then
17.         UFset = UFset ∪ {fi};
18.     else if (eci < n_EC) then
19.         FDCset = FDCset ∪ {FDCi};
20.     end if
21. end for
22. return FDCset, UFset;
```

Algorithm 1. Extract_Conditions

は，テストパターンが得られた場合，X判定を実行し，対応する拡張テストキューブfdcを導出する．行8～12では，FDC_iがNULLである場合新たに得られた拡張テストキューブを設定する．そうでない場合，FDC_iとfdcを論理和演算することでFDC_iを更新する．行13では，拡張テストキューブの抽出数ec_iを1増加させる．行16～20では，故障f_iのテストパターンが一つも得られなかった場合，f_iをテスト不能故障と判断し，テスト不能故障集合$UFset$に追加する．一方，テストパターン数が0より大きくn_EC未満の場合，FDC_iは故障検出の必要十分条件を表すため，$FDCset$に追加する．行22では，故障検出条件集合$FDCset$とテスト不能故障集合$UFset$を出力する．

1.3　故障検出条件の圧縮

Algorithm 1で得られた故障検出条件集合は積和形論理式で記述され，論理合成により各故障につき一つの回路を生成する．各回路の外部入力は境界ゲートの出力信号線に対応し，外部出力は1ビットである．外部出力が「1」となる境界ゲート出力信号線の割当ては，当該故障が検出されることを意味する．反対に，外部出力が「0」となる割当ては，当該故障が検出不可能であることを意味する．

4. 必要十分条件を用いた複数目標故障テスト生成アルゴリズム

本章では，必要十分条件を用いた複数目標故障テスト生成アルゴリズムについて説明する．必要十分条件を用いた複数目標故障テスト生成処理をAlgorithm 2 FDC_MTTGに示す．本アルゴリズムは回路C，1次故障pf，2次故障集合SF,および故障検出条件集合$FDCset$を入力とし，複数目標故障に対するテストパターンtを出力するものである．

以下では，Algorithm 2の具体的な処理手順について説明する．

行1では，目標故障を検出するための制約ψ_Fを1に初期化し，テストパターンtをNULLで設定する．行2では回路Cをもとに正常回路の論理制約ψ_gを生成する．

Algorithm 2: FDC_MTTG

Input : Circuit C, primary fault pf, Secondary Fault set SF
Fault Detection Condition Set $FDCset$
Output : test pattern t

1. ψ_F = 1; t = NULL;
2. ψ_g = GoodCircuitModel(C);
3. **for each** fi **in** ($SF \cup \{pf\}$)**do**
4. ψ_{fi} = LookupFDC($FDCset$, fi);
5. **if** (ψ_{fi} == NULL) **then**
6. ψ_{fi} = FaultCircuitModel(C, fi);
7. **end if**
8. ψ_F = $\psi_F \wedge \psi_{fi}$
9. **end for**
10. ψ_D = Detection(pf);
11. ψ_{MD} = MaximizeDetection(SF);
12. t = Solver($\psi_g, \psi_F, \psi_D, \psi_{MD}$);
13. **return** t;

Algorithm 2. FDC_MTTG

行3～9では，目標故障集合の各故障に対して処理を行う．具体的には，各故障f_iに対して論理圧縮後の故障検出回路が存在する場合は，当該回路の論理制約を生成し，そうでない場合，故障回路の論理制約，故障伝搬制約を生成する．行4では，$FDCset$に故障f_iの検出回路が存在する場合，当該回路の論理制約を生成する．そうでない場合はψ_{fi}をNULLで設定する．行5~7では，ψ_{fi}がNULLである場合，故障回路の論理制約，故障伝搬制約を生成する．行8では，新たに得られた当該故障に対応する制約ψ_{fi}をψ_Fに追加する．行10では，1次故障pfを必ず検出する制約ψ_Dを生成する．行11では，2次故障集合SFの検出数を最大化するための最適化関数ψ_{MD}を生成する．故障検出回路の論理制約を用いた故障の場合，外部出力が故障検出を示す変数である．そうでない場合，故障信号線における故障伝搬変数(D-variable)[10]が故障検出を示す変数である．行12および13では，行2～11で得られた制約$\psi_g, \psi_F, \psi_D,$およびψ_{MD}をPBOソルバに入力し，テストパターンtを生成し，出力する．

5. 実験結果

本章では，必要十分条件を用いた複数目標故障テスト生成プログラムで得られた実験結果について説明する．本実験はC言語で実装し，ISCAS'89ベンチマーク回路に対して 11th Gen Intel® Core™ i7-11700 を搭載したコンピュータを用いて行った．論理圧縮はDesign Compilerを使用し，X判定は内製ツールを使用した．PBOソルバは Clasp [12]のバージョン3.3.9を使用した．Claspのテスト生成一回当たりの制限時間は100秒に設定し，目標故障数は300に設定した．対象故障モデルは単一縮退故障モデルである．比較手法では，文献[10]で説明されている制約式を適用し，MTTGを行った．目標故障の選択は両手法ともゲート段数の昇順に選択した．

表1に，提案手法および比較手法に基づく複数目標故障テスト生成（MTTG）の結果を示す．対象とした回路は ISCAS'89 ベンチマークの s5378_C, s9234_C, および s15850_C の3種類である．表中の“tp”は生成されたテストパターン数を示し，“optimum[%]” は最適解が得られたテストパターン数の割合を表す．また，“MTTG time[s]” は制約式の作成からテストパターン集合を生成するまでの全処理時間を，“CLASP time[s]” は PBOソルバ Clasp の総処理時間を示している．

表 1. 実験結果

method		s5378_C	s9234_C	s15850_C
FDC + FC	tp	100	114	115
	optimum[%]	100.00	99.12	99.13
	MTTG time[s]	132.43	606.14	282.70
	CLASP time[s]	117.20	585.68	248.75
	sufficient[%]	9.22	12.59	9.54
FC	tp	103	116	115
	optimum[%]	99.03	100.00	99.13
	MTTG time[s]	286.64	500.95	575.85
	CLASP time[s]	286.08	492.40	544.61

いずれの回路においても完全故障検出効率を達成している．

提案手法は，必要十分条件を適用した FDC と，必要十分条件が構築できなかった故障に対して比較手法と同様の FC 制約を適用した組合せ手法（FDC＋FC）である．一方，比較手法（FC）は，すべての故障に対して FC 制約のみを適用している．なお，“sufficient[%]” は Algorithm 1 において必要十分条件が得られなかった故障の割合を示しており，これらの故障に対しては FC 制約を適用している．実験の結果，MTTG time および CLASP time において s5378_Cおよびs15850_Cで提案手法が比較手法よりも短縮されている．s15850_C において，提案手法の CLASP time は 248.75[s] であるのに対し，比較手法では 544.61[s] と約2.2倍の差が確認された．これは，提案手法が必要十分条件を導入することで，ソルバの探索空間を削減し，処理効率を向上させた結果であると考えられる．また，テストパターン数に関してすべての回路で同等またはより小さいテスト集合が生成された．

6. むすび

本論文では，故障検出の必要十分条件を用いた複数目標故障テスト生成を提案した．拡張テストキューブからX判定および論理合成を用いることで故障検出制約をコンパクトにした．ベンチマーク回路を用いた実験の結果一部の回路で，テスト生成時間を大幅に削減し，テストパターン数の削減も見られた．しかしながら，s9234_Cのようにテスト生成時間が増加するケースも観察された．今後の課題としては，さらなる故障検出制約の縮小が挙げられる．

参考文献

1) 藤原秀雄,”ディジタルシステムの設計とテスト”, 工学図書株式会社,pp.146, May. 2004.
2) L. N. Reddy, I. Pomeranz, and S. M. Reddy, "ROTCO: A reverse order test compaction technique," in Proc. Euro ASIC Conf., Paris, France, pp. 189-194,1992.
3) S. Kajihara, I. Pomeranz, K. Kinoshita, "Cost-Effective Generation of Minimal Test Sets for Stuck-at Faults in Combinational Logic Cireuits, " IEEE TCAD Vol.14 Issue12, pp.1496-1504, Dec.1995.
4) K. Miyase, S. Kajihara, and S. M. Reddy, "A Method of Static Test Compaction Based on Don't Care Identification, "Proc. First IEEE International Workshop on Electronic Design, Test and Applications, Vol.43, No.5, pp. 392-395, May 2002.
5) J .- S. Chang and C .- S. Lin, "Test set compaction for combinational circuits," IEEE Trans. on Computer-Aided Design, Nov. 1995. Vol. 14, No. 11, pp.1370- 1378.
6) S. Eggersglüß, R. Krenz-Bååth, A. Glowatz, F. Hapke, and R. Drechsler, "A new SAT-based ATPG for generating highly compacted test sets," in Proc. of Symp. on Design and Diagnosis of Electronic Circuits and Systems, 2012, pp. 230-235.
7) G .- J. Tromp, "Minimal test sets for combinational circuits," in Proceedings of the Intl. Test Conf., 1991, pp. 204-209.
8) 松永裕介:大規模回路向けテストパタン集合最小化手法の高速化について,信学技法 IEICE-VLD2015-2, pp. 25-30(2015).
9) I. Miyase and M. Kajihara, “XID：identification of don’t cares in test patterns for combinational circuits,” IEEE Trans. Comput.−Aided Des. Integr. Circuits Syst., vol. 23, no. 2, pp. 321−326, Feb. 2004. DOI：10. 1109/TCAD. 2003. 822103.
10) Eggersglüß, Stephan, Kenneth Schmitz, Rene Krenz-Baath and Rolf Drechsler. "On Optimization-Based ATPG and Its Application for Highly Compacted Test Sets.", Proc. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems vol. 35, pp.2104-2117, Dec. 2016.
11) 松永裕介:拡張テストキューブを用いたテストパタン圧縮手法, DA シンポジウム 2024,pp.5B-4(2024).
12) M. Geber, B. Kaufmann, A. Neumann, and T. Schaub, "Conflict-Driven Answer Set Solving, "IJCAI, pp.386-392, Hyderabad, India, Jan. 2007.