

充足可能性問題の解法を用いた 縮退故障モデルのテスト生成の評価

日大生産工（学部）○山崎 紘史

日大生産工 細川 利典
九大 吉村 正義

1. はじめに

近年、半導体微細化技術の進歩に伴い、大規模集積回路（Large Scale Integration: LSI）が大規模化・複雑化し、テストパターンの生成時間が増加している。

従来の経路活性化法に基づくテスト生成アルゴリズム[1]では、LSIが大規模化・複雑化することにより、現実的な時間で100%の故障検出効率を得ることが困難になりつつある。この問題の解決の一つとして、テスト生成時に発生するバックトラック[1]の発生回数の削減と、バックトラック間の処理を高速化することの2点が挙げられる。バックトラックの発生回数削減に関しては過去に動的決定順序[2]や衝突による再帰学習[2]などテスト生成中の値衝突の発生を効率的に活用する方法が提案されている。

一方、近年では衝突項の学習[3]や非辞書式バックトラック[4]、ブール強制伝搬[4]の技術により、SAT-solverの処理速度が急速に進歩している。また回路の等価性検証やテスト生成技術など論理関数を処理する分野で、SAT（充足可能性問題）を利用した手法が提案されている[5][6][7]。

本論文ではSATを利用することで高速なテストパターン生成が実現できる故障の特徴を見つけ出す、前段階として最新のSAT-solver[8]に基づくテスト生成を実装し、テスト生成時間とバックトラック数をし、従来の経路活性化法に基づく方法と比較する評価する。

2. 経路活性化法を用いたテスト生成アルゴリズム

自動テストパターン生成（Automatic Test Pattern Generation: ATPG）ツールは、論理回路情報からLSIをテストするテストパターンを自動生成する。LSIのテストでは、品質の高いテストを行うために、高故障検出効

率のテストパターンを生成する必要がある。

しかしながら、LSIの大規模化・複雑化により、解探索空間の拡大によるバックトラックの多発が問題となっている。従来の経路活性化法を用いたATPGツールでは、現実的な時間で100%の故障検出効率を得るテストパターン生成をすることが困難となっている。

経路活性化法のバックトラックは含意操作等の値割当て時に矛盾が発生した場合に発生し、矛盾が生じた箇所に対して異なった値を再度割り当てる。このとき判定木において矛盾が生じたノードと、バックトラックを引き起こした原因となるノードが離れた位置関係にあったとしても、バックトラックを引き起こした原因に辿り着くまでバックトラックが行われる。そのためバックトラックを起こした原因をテスト生成中に解析しにくい。過去に起こした矛盾の原因をテスト生成中に活用しにくいという問題点がある。また経路活性化法では故障の伝搬経路を決定した後に信号線に値を割り当てる。そのため検出しづらい故障や、一部の冗長故障判定が難しく、これらの故障、冗長故障を判定しようとするテスト生成時間に多大な影響を与える[9]。

一方、近年のSAT-solverでは項衝突の学習[3]や非辞書式バックトラック[4]、ブール強制伝搬[4]により、経路活性化法の問題点を克服していることが報告されている。

3. 充足可能性問題を用いたテスト生成

充足可能性問題（Satisfiability problem: SAT）とは乗法標準形（Conjunctive Normal Form: CNF）が与えられたときに、全ての変数の値を1（真）または0（偽）に定めることで、全体の値を1（真）にできる割当てが存在するか否かを判定する問題である。CNFを1（真）にできる割当てが存在した場合、

Evaluation of on Test Generation for a
Stuck at Fault Model Using Satisfiability

Hiroshi YAMAZAKI, Toshinori HOSOKAWA, and Masayoshi YOSHIMURA

充足可能と言い、与えられた CNF が正当化可能であることを示す. CNF が 1 (真) にできる割当が存在しない場合、すなわち 0 (偽) である場合は与えられた CNF が正当化不可能であることを示す.

SAT は CNF を入力として問題を解決するため、SAT を用いてテストパターンの生成を行うには、論理回路を CNF に変換する必要がある. 3.1 で論理回路の CNF 変換について述べ、3.2 で SAT を用いたテスト生成について述べる. 3.3 で SAT を用いた ATPG の全体のアルゴリズムを示す. 3.4 でアルゴリズムを改善した SAT を用いた ATPG の全体アルゴリズムを示す.

3.1. 論理回路の CNF 変換

表 1. 論理ゲートの CNF 変換規則

ゲートタイプ	入力	出力	CNF
AND	X Y Z	Z	$(\neg Z+X) \cdot (\neg Z+Y) \cdot (\neg X+\neg Y+Z)$
OR	X Y Z	Z	$(Z+\neg X) \cdot (Z+\neg Y) \cdot (X+Y+\neg Z)$
EXOR	X Y Z	Z	$(\neg X+Y+Z) \cdot (X+\neg Y+Z) \cdot (\neg X+\neg Y+Z) \cdot (X+Y+\neg Z)$
NOT	X Y	Y	$(X+Y) \cdot (\neg X+\neg Y)$
Fan-OUT	X Y Z	Y Z	$(\neg X+Y) \cdot (X+\neg Y) \cdot (\neg X+Z) \cdot (X+\neg Z)$

表 1 に各論理ゲートの CNF 変換規則を示す. 各論理ゲートは表 1 の規則に従い CNF に変換される. それぞれの括弧でくくられた論理和式を項といい、項を論理積した式が各ゲートの CNF である. 例として 2 入力 (X, Y) 1 出力 (Z) の OR ゲートを考える. 全体の値が真になるのは (X, Y, Z) = (0, 0, 0), (0, 1, 1), (1, 0, 1), (1, 1, 1) の 4 通りだけである. これにより、CNF が実際の論理ゲートの動作を表現していることがわかる. 回路全体の CNF は各論理ゲートの CNF を論理積することで表現できる.

最新の SAT-solver[8] では CNF に含まれる変数に値を割当てて処理で、値割当て頻度の高い状態変数から割当てて、変数選択ヒューリスティック[3]や、値割当てに矛盾が生じた場合、矛盾を解析・学習[4]する処理が組み込まれている.

正当化が不可能な場合は、項のどれかひとつでも偽であった時点で、その割当ての組合せでは正当化が不可能であることがわかる. したがって、経路活性化法に基づくアルゴリズムに比べ、効率的に早期に矛盾を発見でき、高速な正当化可能判定処理が可能である.

3.2. SAT を用いたテスト生成

初めにテスト対象となる回路に対して、正常動作する回路と、ある信号線の一つが縮退故障[1]している故障回路を用意する. そして、正常回路と故障回路の入力に共通の外部入力を接続する.

外部出力に対しては、故障の影響を励起・伝搬させるためには片方の外部出力を真、もう一方を偽にさせる必要があるため、それぞれの外部出力を EXOR ゲートの入力に接続する. 故障の検出は少なくとも一つの外部出力へ故障の影響が伝搬できればよい. 各外部出力を接続した EXOR ゲートの出力を OR ゲートの入力に接続し、この出力が常に 1 (真) になる情報を付加する. 故障情報は故障信号線に対して、外部入力側に正常値、外部出力側に故障値を設定することで示す.

図 1 に SAT を用いたテスト生成モデルの例を示す. 図 1 の G0 から G6 部分が正常回路である. また、G0' から G6' が故障回路であり、AND ゲート G4' の出力信号線に 0 縮退故障があるとする. このため AND ゲート G4' の出力が正常値 1 になるように設定し、OR ゲート G6' は、G4' からの入力が故障値 0 になるように設定する.

a, b, c, d, e は外部入力であり、これらは正常回路と故障回路の入力に接続している. 正常回路の出力は m と n, 故障回路の出力は m' と n' である. これらの出力 m と m', n と n' をそれぞれ EXOR ゲートの入力に接続し、EXOR ゲートの出力が x と y である. EXOR ゲートの出力である x と y を OR ゲートの入力に接続し、その OR ゲート出力が z であり、故障の励起・伝搬のため z に 1 を設定する.

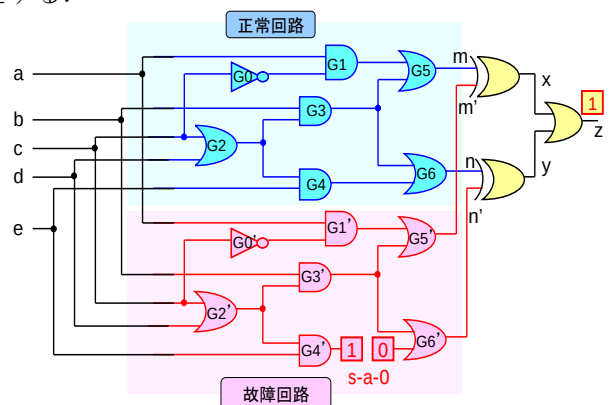


図 1. SAT を用いたテスト生成モデルの回路例

これらの条件を付加した CNF を作成し、SAT-solver を解くことで充足可能であればテスト生成が可能、充足不可能であれば、その故障に対するテスト生成が不可能、すなわち冗長故障であると判定できる。

3.3. SAT を用いたテスト生成の全体アルゴリズム

図 2 に SAT を用いた ATPG の全体アルゴリズムを示す。

(Step1)

テスト生成対象回路データを読み込む。

(Step2)

Step1 で読み込んだデータから、代表故障 [1] リストを作成する。

(Step3)

Step1 で読み込んだ回路データから CNF 式を生成する。

(Step4)

代表故障リスト内の全故障に対してテスト生成を行ったかを判断する。生成し終わった場合は終了し、それ以外のときは Step4 へ進む。

(Step5)

Step4 で選択した故障信号線部分の CNF 式を生成する。

(Step6)

Step3, Step5 で生成した CNF 式からテスト生成可能な CNF 式を生成する。

(Step7)

Step6 で生成した CNF を SAT-solver に読み込ませ、充足可能か否かを判定する。

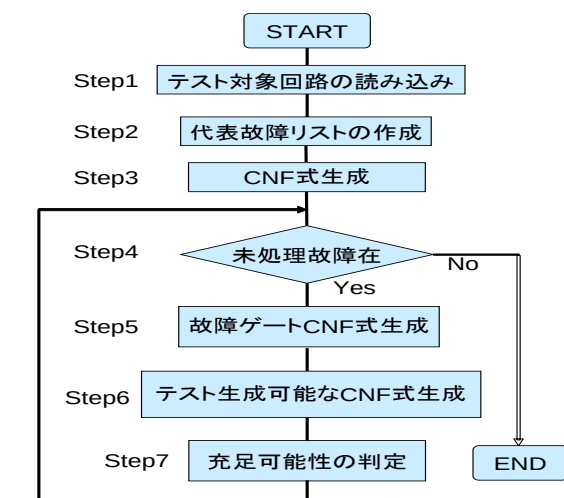


図 2. SAT を用いたテスト生成の全体アルゴリズム

4. 実験結果

表 2. 実験結果

回路名	SATを用いたATPG		FANⅢ		SPOP	
	テスト生成時間(秒)	バックトラック数	テスト生成時間(秒)	バックトラック数	テスト生成時間(秒)	バックトラック数
c880	5.82	5182	1.58	0	1.63	0
c1908	40.75	31066	43.69	106	45.03	1732
c2670	81.07	14449	138.65	56698	27.22	595
c5315	315.46	54041	156.24	1101	237.75	3922

本論文では、実験結果として 3.2 で述べた、充足可能性問題を用いたテスト生成法を図 3 のアルゴリズムで実装し、経路活性化法を用いたテスト生成である FANⅢ[10] と SPOP[11] でテスト生成時間とバックトラック数を比較した。実験環境は、CPU は Pentium4、メモリは 2 GB、OS は RedHat、使用した SAT-solver は MiniSat-C v1.14.1[8] である。実験対象として ISCAS'85 ベンチマーク回路の c880, c1908, c2670, c5315 を用いた。バックトラック制限値は無制限である。故障シミュレーションは使用していない。表 2 に実験結果を示す。

表 2 より c880, c1908 などの経路活性化法に基づくテスト生成でバックトラック数が少ないものに関しては、SAT を用いた ATPG のほうが 2 倍以上テスト生成時間を要している。これは経路活性化法に基づくテスト生成法では故障の伝搬経路を決定しているため含意操作[1]で信号線の値が一意に決定する。そのためテスト生成が容易な故障に関してはバックトラック数が少ない。一方 SAT を用いたテスト生成では与えられた CNF 式の全ての変数に 0 または 1 の割り当てを行うためテスト生成時間が経路活性化法より要したと考えられる。

一方 c1908 や c2670 のようなバックトラック数の多い回路に関しては SAT によるテスト生成のほうが、経路活性化法に基づくテスト生成法よりテスト生成時間がみ短く、FANⅢと比べると約 30% のテスト生成時間の削減ができていることがわかる。これは経路活性化法では検出しづらい故障や、冗長故障に関してはバックトラック数が多くなりテスト生成時間が増大する。そのため c1908, c2670 は経路活性化法では検出困難な故障と冗長故障が多いと考えられる。また SAT を用いたテスト生成法は高速な冗長故障判定

が出来る[9]ため、経路活性化法よりもテスト生成時間が高速化されたと推測できる。

5. おわりに

実験結果から経路活性化法に基づくテスト生成が多くバックトラックをするような故障に対しては SAT を用いたテスト生成のほうが高速になる可能性が考えられる。またバックトラック数が少ない検出が容易な故障に関しては経路活性化法のほうが高速であると考えられる。これらの結果から SAT を用いたテスト生成にも経路活性化法の技法である一意活性化などの手法を取り入れることでバックトラック数の削減が可能と考察できる。また検出が容易であると思われる故障に関しては前処理としてランダムパターンシミュレーションや故障シミュレーションを取り入れることによりテスト生成時間の大幅な削減が可能と考えられる。

今後の課題として、故障シミュレーション、ランダムパターンシミュレーションの実装。また ITC'99 ベンチマーク回路などの大規模回路による手法の評価が挙げられる。

参考文献

- 1) 藤原秀雄, “デジタルシステムの設計とテスト”, 工学図書株式会社 (2004), pp262
- 2) Chen Wang, Sundhakar M.Reddy “Conflict Driven Techniques for Improving Deterministic Test Pattern Generation”, IEEE/ACM international conference on Computer-aided design (2002), pp87-93
- 3) Matthew W. Moskewicz, “Chaff: Engineering an Efficient SAT Solver”, 38th annual Design Automation Conference (2001), pp530-535
- 4) Joao P. Marques-Silva, “GRASP: A Search Algorithm for Propositional Satisfiability”, IEEE TRANSACTIONS ON COMPUTERS Volume 48 (1999), pp506-521
- 5) Tracy Larrabee, “Test Pattern Generation Using Boolean Satisfiability”, IEEE TRANSACTION ON COMPUTER-AIDEDDESIGN, VOL.11, No.1 (1992)
- 6) Paul Stephan, Robert K.Brayton, Alberto.Sangiovanni-Vincentelli, “Combinational Test Generation Using Satisfiability”, IEEE.TRANSACTION.ON.COMPUTER-AIDEDDESIGN, VOL.15, No.9(1996)
- 7) Paul Tafertshofer, Andreas Ganz, “SAT Based ATPG Using Fast Justification and Propagation in the Implication Graph”, IEEE/ACM international conference on Computer-aided design(1999), pp139-146
- 8) Niklas Een, Niklas Sorensson “An Extensible SAT-solver”(2003)
- 9) 秋山祐介, “充足可能性問題の解法を用いた ATPG 困難故障の故障分類判定の高速化に関する研究”, 日本大学大学院生産工学研究科 2008 年度修士研究(2009)
- 10) H.Fujiwara and T.Shimono, “On the Acceleration of Test Generation Algorithms”, IEEE Transactions on Computers Vol32 (1983), pp1137-1144
- 11) M.Henftling, H.Wittmann and K.Antreich, A Single-Path-Oriented- Fault-Effect Propagation in Digital Circuits Considering Multiple-Path Sensitization, Proceedings of ICCAD(1995), pp304-309.