

XOR制約を用いたSAT問題のサンプリングと テストパターン生成への応用

松永 裕介^{1,a)}

概要：本稿では元の制約式に複数の変数の XOR で構成された制約式を追加することで SAT 問題のサンプリングを行う手法を紹介し、その応用として信号遷移回数を考慮した遷移故障向けテストパターンを生成するアルゴリズムの提案を行う。通常の SAT ソルバを用いた手法では故障を検出する論理的な制約を満たすテストパターンが唯一得られるだけで信号遷移回数のコントロールを行うことはできない。一方提案アルゴリズムではある程度ランダムにテストパターンを生成することができるためそのなかから信号遷移回数の少ないものを選ぶことによって生成されるテストパタンの質をコントロールすることができる。

XOR constraint based SAT model sampling and its application for test pattern generation

YUSUKE MATSUNAGA^{1,a)}

Abstract: This paper first introduces XOR constraint based SAT model sampling method, which adds randomly generated constraints to the original problem, then proposes a new test pattern generation algorithm considering signal transition activities as an application of the sampling method. While the existing SAT based test pattern generation algorithm does not control the signal transition activities of the generated patterns, the proposed one can handle them with randomly generating more than one patterns for a fault.

Keywords: test pattern generation, signal transition activity, SAT, random sampling

1. はじめに

製造した LSI に故障がないかどうかを検査するためテストパターンを生成する問題をテスト (パターン) 生成問題と呼ぶ。テスト生成を行う際には予め定義された故障モデルに基づいた故障を仮定し、それを (できるかぎり全て) 検出することを目的とする。故障モデルとして従来から広く用いられてきたものは信号線の値が 0 もしくは 1 に固定してしまうと仮定する縮退故障 (stuck-at fault) であるが、現在は縮退故障に加えてダイナミックな動作に関する遅延の異常を仮定する遷移故障 (transition fault)[1] も用いられるようになってきている。遷移故障とは LSI 中の回路の動作が遅くなると仮定する遅延故障の一種であり、検査のために実際の

動作速度で回路を動かす必要があるため消費電力に注意を払う必要がある。通常の動作時よりも消費電力が大きい場合、電源電圧の降下を引き起こし回路遅延の増大を招く恐れがある。また、電力消費に伴う発熱が LSI のパッケージなどの耐熱規格を超えると部品の破壊にもつながることになる。

既存のテスト生成手法で信号遷移回数を考慮したものも多く提案されているが、その殆どがアドホックなヒューリスティックであり一般性に乏しい。一方、テスト生成の手法としては近年、SAT(satisfiability) ソルバを用いた手法 [4] が広く用いられる様になっている。これは 2000 年代に入って Chaff[6], [8] や GRASP[5], MINISAT などの優れた SAT ソルバが開発されたため、大規模な回路のテスト生成に関しては SAT ソルバを用いたものが効率的であると考えられているためである。しかし、SAT ソルバは純粋に

¹ 九州大学大学院システム情報科学研究院
〒 819-0395 福岡市西区元岡 744

^{a)} matsunaga@ait.kyushu-u.ac.jp

与えられた CNF(conjunctive normal form:和積標準形) 式を充足する変数割り当てを求めるだけであり、なんの工夫も行わなければ生成されたテストパタンの質をコントロールすることはできない。

SAT ソルバの応用として充足される節の数を最大化する MAX-SAT や CNF 式の制約を満たす中で目的関数を最適化する PBO(Pseudo Boolean Optimization) などが提案されている。しかし、今回考えている問題は信号遷移回数を最小したいわけではなく、通常動作とおなじくらいの回数となるパターンを求めたいのでこれらの最適化系のアルゴリズムはうまくマッチしない。そこで、SAT 問題の解をランダムに生成するアルゴリズムを用いてこの問題を解くことを考える。今回用いたランダムサンプリング手法は XOR 制約に基づく手法で、もとの問題に対してランダムに生成された制約式を付加した問題を SAT ソルバで解くことでサンプリングを行うものである。この手法を信号遷移回数を考慮したテストパターン生成アルゴリズムへ応用することを考える。SAT 問題のサンプリング結果として得られるテストパターンは信号遷移回数などの尺度でもある程度ばらついていると考えられるのでそのなかから条件に合致したものを選ぶことで結果として生成されるテストパタンの質をコントロールすることが可能となっている。

以下、2 節で遷移故障のテスト生成問題と SAT 問題への定式化について説明を行い、4 節で SAT 問題のサンプリング手法について述べ、5 節で提案手法の詳細について述べる。

2. 遷移故障とそのテスト方式

2.1 遷移故障モデルとその検出方式

遷移故障 [1] は LSI 中の回路の動作遅延が通常よりも遅くなる不具合を仮定する故障である。具体的には信号線の値が 0 から 1 に遷移する時間が遅くなる 'slow-to-rise' 故障と 1 から 0 に遷移する時間が遅くなる 'slow-to-fall' 故障の 2 種類がある。実際には信号線における値の変化が出力まで伝搬するかどうかはそのパス上のゲートの他の入力の値の到達時刻に依存するため厳密に遅延故障を検査すること、およびそのためのテストパターンを生成することは難しい。遷移故障モデルでは具体的な遅延時間を見捨てて変化前の論理値と変化後の論理値が異なっていたら信号遷移が伝搬したとみなす。そこで、遷移故障をテストするためには 2 つのテストパターンを続けて通常動作のクロック速度で入力する必要がある。そのためのテスト方式はいくつか提案されているが、本稿ではブロードサイド方式 [7] を用いるものとする。ブロードサイド方式ではスキャンチェーンを通じて最初のテストパターンをフリップフロップに設定する。その後、通常の動作モードで 1 クロック動かして結果のフリップフロップの値をスキャンチェーンを通じて読み出す。つまり、2 パターン目のフリップフロップの値は通常

の動作の結果与えられるものとなる。図 1 にブロードサイド方式の動作を時間展開した様子を示す。まず、スキャン

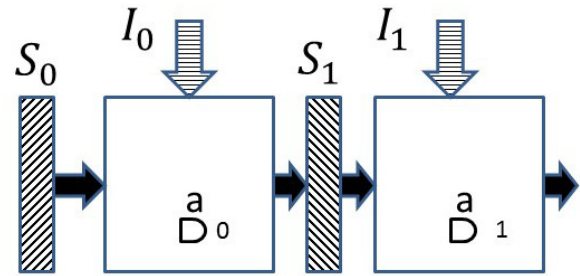


図 1 ブロードサイド方式

チェーンにより最初の状態 S_0 がフリップフロップにセットされる。外部入力には I_0 というベクタを与えるものとする。次に動作モードに切り替えて 1 クロック分の動作を行う。するとフリップフロップには次の状態 S_1 がセットされる。その際に外部入力には I_1 というベクタを与えるものとする。その結果の状態をスキャンチェーンによって読み出せば (S_0, I_0) と (S_1, I_1) を続けて印加した結果を得ることができる。このときに、 a における 'slow-to-rise' 故障を検出するのであれば、最初の時刻においては a の値は 0 でなければならない、2 時刻目においては 1 でなければならない。また、2 時刻目において a の値が正常値と異なっているとき (この場合は 0) に、その結果が外部出力かフリップフロップに伝搬しなければならない。実はこの 2 時刻目の条件は a における 'stuck-at-0' 故障の検出条件と全く同じものとなる。つまり、 a における 'slow-to-rise' 故障の検出条件は以下のようになる。

- 1 時刻目において $a = 0$ であること
- 2 時刻目において a の 'stuck-at-0' 故障が検出できること

このように遷移故障のテスト生成問題は縮退故障のテスト生成問題に似ているため、同様の手法もしくは拡張した手法が用いられることが多い。ただし、遷移故障の検査においては通常動作とおなじクロック速度で回路を動かす必要があるため、テスト動作時における消費電力に注意を払う必要がある。

2.2 SAT 問題としての定式化

このブロードサイド方式による遷移故障の検出問題を SAT 問題として定式化するのはほとんど縮退故障に対するテスト生成問題と同様である。まず、図 2 に示すようなテスト生成用の仮想的な合成回路を考える。ブロードサイド方式を用いているので正常回路は 2 時刻分展開されている。さらに 2 時刻目には故障回路も接続されており、2 時刻目の出力を正常回路と故障回路で比較することで故障の検出が行えるかを判定する。故障の検出条件は前述の通り

であるが、注意が必要なのは2時刻目の状態 S_1 は外部から与えるのではなく、回路内部で自動的に生成されるということである。そのため、外部から与えるテストパターンとしてはスキャンチェーンによって設定する S_0 と2つの時刻における外部入力 I_0, I_1 となる。以降、 (S_0, I_0, I_1) をテストパターンと呼ぶことにする。

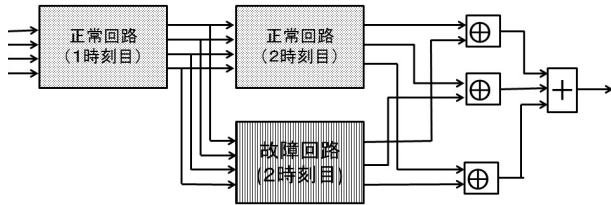


図2 テスト生成用の合成回路

最後に実用的な SAT ベースのテストパターンアルゴリズムについて注意が必要な点を挙げる。多くの場合、一つの故障の影響が伝搬する部分回路は回路全体に比べるととても小さい、また、その部分回路の値を決定するそのファンイン部分の同様である。そこで、SAT 問題として定式化する場合もこのように故障検出に関係する部分回路のみを取り出して CNF 式に変換している。SAT 問題を解いた結果として値が割り当てられるのはこの部分回路に属する外部入力（フリップフロップ）のみであり、それ以外の信号線の値はドントケアとなっている。また、SAT 問題としては値が確定していてもテストパターンとしては 0,1 どちらでも故障が検出できるものも存在する。このように SAT ベースで生成されるテストパターンは一般的には一部の入力のみ値が固定されており、残りはドントケアとなっていることが多い。

3. SAT 問題のランダムサンプリングを行うアルゴリズム

ここでは本手法の元となっている SAT 問題のランダムサンプリングアルゴリズムについて説明を行う。SAT 問題に対するランダムサンプリングはそもそも難しい問題^{*1}であり、現在提案されているものはほとんどが近似手法である。

4. XOR 制約を用いた SAT サンプリング

SAT 問題のサンプリング手法としてマルコフ連鎖モンテカルロ法を用いたランダムウォークベースの手法が知られている [3]。しかし、純粋なランダムウォークで SAT 解に到達する確率はとても低く、解に収束するように移動にバイアスをかけるとサンプリングの一様性が悪くなるという二律背反性がある。このランダムウォークベースの SAT サンプリングと異なるアプローチとして XOR 制約を用いたサンプリング手法が提案されている [2]。この手法は内

部で通常の SAT ソルバを用いるものでランダムウォークベースの手法に比べて高速な処理が期待できる。基本となるアイデアはブール空間を 2^l 個に分割して、その1つの区画に対して SAT 問題を解く、もしその区画に1つしか解が存在しなければ、その解をサンプリング結果として返すというものである。毎回、この 2^l 個の分割 (と1つの区画の選択) をランダムに行うことで全体としてランダムサンプリングを行っている。ブール空間を 2^l に分割しその中の1つの区画を選ぶという処理は実際には l 個の XOR 制約 $x_1 \oplus x_2 \oplus x_3 \dots$ を用いている。今、全部で n 個の変数のなかから k 個の変数 $x_1, x_2, \dots, x_k$ を選んだとする。すると 2^n のブール空間の各点は $x_1 \oplus x_2 \oplus \dots \oplus x_k$ を 0 と 1 のどちらかに評価するかで2つに分けられる。これを l 個の異なる変数集合に対して行うことで、全体で 2^l 個の区画に分割することができる。どの区画を選択するかは各 XOR 制約式が 0 になる区画か 1 になる区画かを選べば良い。そこで、ランダムに変数を k 個選び出し、さらに XOR のパリティ (結果が 0 か 1 か) を決めるという処理を l 回繰り返す、その結果生成された XOR 制約式を元の CNF に追加する。ただし、CNF は和積系論理式であるので XOR 制約式を和積系論理式に変換する際に人工的な中間変数を導入する必要がある。擬似コードを以下に示す。

```

input : CNF 式  $F$ 
input : 制約式の数  $l$ 
output:  $F$  に対する解

iterationSuccessful  $\leftarrow$  False;
while  $\neg$  iterationSuccessful do
     $Q_l \leftarrow \{l \text{ 個の XOR 制約} \}$ ;
     $F_l^q \leftarrow F \cup Q_l$ ;
    result  $\leftarrow$  SAT( $F_l^q$ );
    if result then
         $\sigma \leftarrow$  SAT( $F_l^q$ ) の解;
         $F' \leftarrow F_l^q \cup \{\sigma\}$ ;
        result'  $\leftarrow$  SAT( $F'$ );
        if  $\neg$  result then
            iterationSuccessful  $\leftarrow$  True;
            return  $\sigma$ ;
        end
    end
end
end

```

Algorithm 1: XORSample アルゴリズム

XOR 制約を作る際に q の確率で各変数を選ぶものとする。通常は $q = 0.5$ が用いられる。XOR 制約付きの論理式に対して SAT ソルバを適用し解が求まった場合、今の解を否定した形の節を追加した問題を再度 SAT ソルバに解かせる。もしこれが充足不能だった場合には、この区画内に

^{*1} P 空間完全である。

は σ という解がただ1つしか存在しないことがわかるのでそれをサンプリング結果として返す。そうでなければ別のXOR 制約を作り直して今までの処理を繰り返す^{*2}。

このアルゴリズムの最大の問題はいかにして l の値を決めるのか、ということである。この l を正確に求めるということはつまり、元の SAT 問題の解の個数を数えることであり、厳密には P 空間完全として知られた難しい問題である。実際にはそれほど神経質になる必要はなく大まかでよいので、上記のアルゴリズムで最初の SAT 問題が失敗する回数が多い場合には区画が細かすぎるので l の値を減らす、2 回めの SAT 問題で複数の解が求まる回数が多い場合には区画が粗すぎるので l の値を増やす、という動的な調整を行えばよい。

5. 提案手法

著者はこの XOR 制約に基づく SAT サンプリング手法をテスト生成へ応用した実験を行っている [9]。その結果、生成されるテストパタンの信号遷移回数がある程度ばらつくことが確認された。しかし、単純なやり方ではサンプリング回数に比例した計算時間を必要とするため実用的ではない。また、ランダムに生成された XOR 制約を付加した SAT 問題はかなりの確率で UNSAT(解が存在しない)であることがわかった。そこで今回はこれらの実験結果を踏まえて実用的なテストパターン生成のアルゴリズムの構築を行った。基本となるアイデアは文献 [2] と同様にランダムに生成された XOR 制約を付加して SAT 問題を解くことで SAT 解のランダムサンプリングを行うというものであるが、以下のような工夫を行っている。

- 通常のシミュレーションで信号遷移回数の平均値を求めておき、それに基づいて設定されたしきい値を超えた場合だけ、サンプリングを行う。
 - XOR 制約を付加する際にその式のパリティ (結果が 0 になるか 1 になるか) は固定しないで変数にしておく。SAT 問題を解く時にこのパリティを指定することで同じ制約で異なる問題を解くことができる。アルゴリズムの概略をアルゴリズム 2 に示す。
- オリジナルの XOR 制約サンプリングアルゴリズムと大きく異なっている部分は l 個の XOR 制約を付加したあとで 2^l 回異なるパリティ割り当てのもとで解を求めていることである。これには2つの利点がある。
- 現在の SAT ソルバの実装では同じ制約のもとで値の割り当てだけを変えて求解する場合、異なる制約のもとで求解するのに比べて大幅に実行時間が短い。
 - 元のテスト生成問題が SAT だった場合、 2^l のどれかの割り当てのなかには必ず SAT となる解が含まれる。このことにより実行時間を大幅に短縮できている。以下で

^{*2} 実際には回数や時間の制限を設けて最後は失敗を返すようにしておく。

```

input : 対象の回路  $C$ 
input : 対象の故障  $f$ 
input : 信号遷移回数の制限値  $a$ 
input : 試行回数の上限  $U$ 
output: テストパターン  $p$ 

 $p \leftarrow C$  上で  $f$  を検出するテストパターンを求める.;
if  $p$  の信号遷移回数が  $a$  よりも少ない then
    return  $p$ ;
end

while True do
    ランダムに  $l$  個の XOR 制約を作る.;
    個々の制約のパリティ条件を表す変数を  $b_i$  とする.;
    for  $v \leftarrow \{0, 1\}^l$  do
         $v$  に従って  $b_i$  に値を割り当てる.;
         $p \leftarrow$  現在の割り当てのもとでの SAT 解;
        if  $p$  が UNSAT then
            continue;
        end
        if  $p$  の信号遷移回数が  $a$  よりも少ない then
            return  $p$ ;
        end
        if 試行回数が  $U$  を超えた then
            break;
        end
    end
end

    今までの最も信号遷移回数の少ないパターンを返す.;

```

Algorithm 2: テスト生成のアルゴリズム

はこのアルゴリズムの詳細について述べる。

5.1 XOR 制約の生成

これは [9] でも用いたアイデアであるが、XOR 制約として選び出す対象の変数としてもとの回路の疑似外部入力 (1 時刻目のフリップフロップの出力と外部入力) のみを用いている。それ以外の変数はこれらの変数に従属しているため、単純にランダムに選んで XOR 制約を作った場合に制約を満たせない確率が高くなるためである。一つの制約のために選ばれた変数を $x_1, x_2, \dots, x_k$ とするとこの変数から作られる制約は以下のようになる。

$$x_1 \oplus x_2 \oplus \dots \oplus x_k \oplus b \quad (1)$$

通常は b は0か1の定数であるが、ここでは b も変数のままにしておいて求解の際に0か1の値を割り当てる様になっている。

5.2 信号遷移回数の平均値の計算

本来はテスト対象回路の機能検証用のテストベンチを用いてシミュレーションを行うのが適切であるが、今回用いたベンチマーク回路は回路構造以外の情報がないため、単純なランダムパターンシミュレーションを用いている。ただ

し、スキャンに入力されるベクタをランダムにするのではなく、外部入力のみをランダムに変化させ順序回路として動作させた場合の信号遷移回数を用いている。これはスキャンを用いて直接フリップフロップに値を設定する場合に、通常動作ではありえない状態を設定することがあり、これが不自然な信号遷移を引き起こすおそれがあると考えられたからである。このランダムシミュレーションの概要をアルゴリズム 3 に示す。

```

input : 対象の回路  $C$ 
input : ウォームアップ回数  $W$ 
input : 試行回数  $N$ 
output: 信号遷移回数の平均値  $sa$ 

 $I_0$  にランダムに生成された外部入力値を入れる.;
 $S_0$  にランダムに生成されたフリップフロップ値を入れる.;
 $I_0, S_0$  を初期値として  $C$  の論理シミュレーションを行う;

for  $W$  回 do
     $I$  にランダムに生成された外部入力値を入れる.;
    現在のフリップフロップの値と  $I$  を用いて  $C$  の論理シミュレーションを行う;
end

 $sa_{total} \leftarrow 0$ ;
for  $N$  回 do
     $I$  にランダムに生成された外部入力値を入れる.;
    現在のフリップフロップの値と  $I$  を用いて  $C$  の論理シミュレーションを行う;
    現在と直前の信号線の値から信号遷移回数を数える。それを  $sa$  とする;
     $sa_{total} += sa$ ;
end

return  $\frac{sa_{total}}{N}$ ;

```

Algorithm 3: 信号遷移回数の平均値の計算

ウォームアップ回数分だけ空でシミュレーションをしている理由はランダムに生成された初期フリップフロップ値が不適切な場合を取り除くためである。簡単な試行ではこのように順序回路としてシミュレーションする場合と毎回ランダムにスキャンベクタを作って 1 クロックだけブロードサイド方式でシミュレーションする場合で信号遷移回数が大きく異なることがわかった。本質的にブロードサイド方式のテストを用いると信号遷移回数が多くなりやすい傾向があるものと思われる。

6. 実験結果

提案手法の効果を調べるために評価実験を行った。ベンチマーク回路は ISCAS89 および ITC99 を用いている。仕様計算機は Intel Core i7-2600 (3.40GHz)(メモリ 16GB), OS は FreeBSD 11.0-RELEASE-p10, 使用コンパイラは clang++-3.4 である。プログラムはシングルスレッドであり、複数のコアは同時には使用していない。

今回はウォームアップ回数 $W = 100$, 試行回数 $N =$

10000 でランダムシミュレーションを行い、信号遷移回数の平均値を求めている。さらにその平均値を 1.2 倍したものを信号遷移回数の制限値と定めた。この制限値以下のテストパターンを求めるという実験を行っている。表 1 に結果を示す。

提案手法の設定は以下のとおりである。

- 追加する XOR 制約の数 l は 5 に固定。
- 試行回数の上限 $U = 20$ 。

これらの数にそれほど意味は無いのでパラメータのチューニングは今後の検討・研究課題である。

表中の各列の記号の意味は以下のとおりである。

- A:** 最初に求められたテストパタンの信号遷移回数が制限値を超えていたものの数。
- B:** 最終的なテストパタンの内、その信号遷移回数が制限値を超えていたものの数。
- C:** サンプルング中の SAT 問題の内、SAT になったものの割合。
- D:** サンプルング中で解を求めた平均回数。数が多ければなかなか信号遷移回数の少ないパターンを見つけることができなかったことを意味する。
- E:** 提案手法の計算時間。単位は秒。
- F:** 同じ実験をサンプルング抜きで行った場合の計算時間。単位は秒。つまり、最初のテストパターンを求めるだけ。

A と B の関係からベンチマーク回路は大まかに以下の 3 通りに区分できる。

- もともと違反するテストパタンの少ないもの。s385417, s38584, b13, b18 等
- 提案手法によって違反するテストパターンを大幅に減らすことのできたもの。s1196, s1238, s1423, s15850, s35932, b10, b11, b12, b14, b15, b17 等
- 多くのパターンが違反状態で、提案手法によってもそれが解決しなかったもの。s1488, s1494 等

そもそも制限値がランダムシミュレーションから求めた平均値を用いており、制限値以下の信号遷移回数を持つ解が存在するかどうか不明なので、この分類にどれだけ意味があるかは不明だが、少なくとも多くの回路に対して提案手法が効果的であることが示されている。

次に実行時間であるが、一回だけ SAT ソルバを起動する既存手法に比べて数倍程度の時間に収まっており実用的には問題ないと言える。これは今回の手法の工夫が効率的であったと言える。

C と D に関しては内部に立ち入った情報である。まず C はもともとが SAT である問題に対して XOR 制約を追加したときに SAT であるものの割合である。これは回路によらず 0.04 から 0.10 の間の値を取っており、予想以上に値が低かった。理想的な場合を考えると SAT 問題の解空間に対しても一様に 2^l 個の区画に分割すれば、この確率は 1

表 1 実験結果

回路名	故障数	検出故障数	A	B	C	D	E	F
s1196	1242	1241	315	15	0.04	3.01	0.52	0.22
s1238	1355	1285	260	19	0.04	3.38	0.49	0.24
s1423	1515	1418	489	13	0.08	2.06	0.93	0.42
s1488	1486	1353	1331	1242	0.07	20.24	4.77	0.33
s1494	1506	1362	1348	1260	0.07	20.14	4.85	0.33
s5378	4603	4253	515	326	0.10	15.24	3.67	1.53
s9234	6927	5844	1924	1116	0.06	14.53	30.73	5.30
s13207	9815	8199	2115	1077	0.08	12.96	29.68	5.57
s15850	11725	8677	953	21	0.08	1.79	17.76	11.50
s35932	39094	35110	4378	326	0.04	3.85	27.30	12.67
s38417	31180	30668	2	0	0.04	0.00	32.52	24.26
s38584	36303	32858	2	2	0.07	21.00	30.96	21.54
b10	517	436	183	49	0.08	7.80	0.23	0.05
b11	1740	1210	881	265	0.07	10.48	3.35	0.61
b12	2878	2575	120	8	0.06	3.38	0.84	0.65
b13	852	671	34	7	0.05	6.03	0.09	0.05
b14	22802	21685	16437	2458	0.10	7.67	448.55	94.17
b15	21988	19737	9993	2633	0.10	8.95	411.61	101.51
b17	76625	67109	19399	4773	0.09	7.71	1127.83	374.86
b18	264091	218786	17	0	0.06	0.11	3249.88	2560.73

に近くなるはずである。この値が低いということは実際には SAT 問題の解空間が偏っているためにごく一部の区画に固まっているということであり、ランダムに選んだ変数の XOR 制約という分割の仕方があまり適切でないことを示唆している。とはいえこれに代わるよいやり方があるとは思えない。

D は解が求まった場合にその信号遷移回数が制限を超えたものの平均値である。こちらは回路ごとに値はおおきくばらついている。明らかに D の値が大きい回路は最終的に制限値を超えたテストパタンの数が多くなっている。これはこの手法で生成しているテストパターンがそこそランダムサンプリングとなっていることの説明になっている。

7. おわりに

本稿では XOR 制約を用いた SAT サンプリングの応用として信号遷移回数を考慮したテストパターン生成の基本アルゴリズムを提案した。現状では回路によって効果のない場合などがあり、改善の余地はあるが、SAT ベースのテストパターン生成アルゴリズムの一般的な機能拡張としてこの XOR 制約を用いた SAT サンプリングは効果的であることを示したと言える。

今後はより一様なサンプリングをするためのヒューリスティックの考案などが研究課題となると思われる。また、信号遷移回数などの制約のもとでテストパターン数を最小化するなどの高度な応用の考えられる。

参考文献

- [1] Cheng, K.-T.: Transition fault testing for sequential circuits, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 12, No. 12, pp. 1971–1983 (online), DOI: 10.1109/43.251160 (1993).
- [2] Gomes, C. P., Sabharwal, A. and Selman, B.: Near-uniform sampling of combinatorial spaces using XOR constraints, *NIPS*, pp. 481–488 (2006).
- [3] Kautz, H. A., Selman, B. and Jiang, Y.: A general stochastic approach to solving problems with hard and soft constraints., *Satisfiability Problem: Theory and Applications*, Vol. 35, pp. 573–586 (1996).
- [4] Larrabee, T.: Test pattern generation using Boolean satisfiability, *IEEE Transactions on Computer-Aided Design of Integrated Circuits*, Vol. 11, No. 1, pp. 4–15 (1992).
- [5] Marques-Sliva, J. and Sakallah, K.: GRASP: A Search Algorithm for Propositional Satisfiability, *IEEE Transactions on Computer-Aided Design of Integrated Circuits*, Vol. 48, No. 5, pp. 506–521 (1999).
- [6] Moskewicz, M., Madigan, C., Zhao, Y., Zhang, L. and Malik, S.: Chaff: Engineering an Efficient SAT Solver, *Proceedings of the 38th Design Automation Conference* (2001).
- [7] Savir, J. and Patil, S.: Broad-side delay test, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 13, No. 8, pp. 1057–1064 (online), DOI: 10.1109/43.298042 (1994).
- [8] Zhang, L. and Malik, S.: Efficient Conflict Driven Learning in a Boolean Satisfiability Solver, *Proceedings of IEEE International Conference on Computer Aided Design (ICCAD 2001)* (2001).
- [9] 松永裕介: 信号遷移回数を考慮したテストパターン生成のための SAT 問題のサンプリング手法について, 信学技法 IEICE-VLD2017-24, pp. 107–112 (2017).