フリップフロップ組合せの状態正当化による到達不能状態を用いた順序回路のテスト不能故障判定法

二関森人[†1]　細川利典[†2]　吉村正義[†3]
山崎紘史[†2]　新井雅之[†2]　四柳浩之[†4]　橋爪正樹[†4]

テストコストの削減やセキュリティ向上のため，スキャン設計を用いない非スキャン設計ベースのテストの要望がある．しかしながら，順序回路のテスト生成では，高い故障検出効率を得るために多大なテスト生成時間を要する．特にテスト不能故障判定時間が支配的である．そのため，テスト生成の前処理で，テスト不能故障を判定することが重要である．本論文では，SAT を用いて数個のフリップフロップ組合せの状態が到達不能状態か否かを判定し，その到達不能状態を用いたテスト不能故障判定法を提案する．また，既存の順序回路のテスト不能故障判定法と提案手法を組み合わせて，ISCAS’89 及び ITC’99 ベンチマーク回路においてテスト不能故障を判定し，評価する．

An Untestable Fault Identification Method for Sequential Circuits Using Unreachable States by Justification of State Cubes

MORITO NISEKI[†1]　TOSHINORI HOSOKAWA[†2]
MASAYOSHI YOSHIMURA[†3]　HIROSHI YAMAZAKI[†2]　MASAYUKI ARAI[†2]
HIROYUKI YOTSUYANAGI[†4]　MASAKI HASHIZUME[†4]

Non-scan based test generation is required for the purpose of resolving reduction of test cost and improvement of security. However, in the test generation of the sequential circuit, it consumes a lot of test generation time to obtain high fault efficiency. Especially, untestable fault identification time is dominant. Therefore, it is important to identify untestable faults in the pre-processing of the test generation. In this paper, an unreachable state identification method, which identifies whether states on a few flip-flops can be justified using SAT, and an untestable fault identification method using the unreachable states are proposed. Moreover, untestable faults are identified by applying the combination of conventional methods and our proposed method to ISCAS’89 and ITC’99 benchmark circuits, and the number of untestable faults is evaluated.

1. はじめに

近年，半導体微細化技術の進歩に伴って，設計される超大規模集積回路(Very Large Scale Integrated circuits：VLSI)の大規模化・複雑化が進んでいる．これに伴い，VLSI のテスト生成は益々困難な課題となってきている．特に順序回路のテスト生成は問題の解空間が大きく，現実的な時間で高い故障検出効率を達成することが困難である.そのため，テスト生成の高速化，容易化が求められている．

テスト生成を容易化する技術の一つとして，回路内のフリップフロップ(Flip-Flop：FF)を外部から制御，観測可能とするフルスキャン設計[1]が提案されている．フルスキャン設計を施すことにより，順序回路を疑似的に組合せ回路として扱うことが可能となり，組合せ回路のテスト生成技術が適用可能となる．組合せ回路に関しては，効率的なテスト生成アルゴリズムが提案されており[2]，大規模・複雑な回路に対しても，現実的な時間で高い故障検出効率を達成することができる．しかしながら，スキャン設計が施された順序回路では，面積，遅延時間，消費電力等のハードウェアオーバヘッドが増大するという問題がある．また，回路内のスキャン FF に値を設定，観測するためのシフト動作によるテスト実行時間の増加もテストコスト削減の観点から問題視されている．さらに，秘密情報を内部に保持する暗号回路に対してスキャン設計が適用されると，スキャンチェインを利用した攻撃が可能となり，秘密情報が外部に漏洩する危険性があると指摘されている[3]．上述の課題を解決するために，スキャン設計を施さない順序回路のテスト生成[4]や，機能動作時の状態のみを使用する擬似機能テスト[5]の適用が求められている．

しかしながら，スキャン設計を施していない順序回路のテスト生成では，内部状態を外部入力のみから設定し，故障の影響を外部出力のみで観測するため，テスト容易化設計なしでは高い故障検出効率を得ることが困難である．また，テスト不能故障に対して順序回路のテスト生成を実行

†1 日本大学大学院 生産工学研究科
Graduate School of Industrial Technology, Nihon University.
†2 日本大学 生産工学部
College of Industrial Technology, Nihon University
†3 京都産業大学 コンピュータ理工学部
College of Faculty of Computer Science and Engineering, Kyoto Sangyo University
†4 徳島大学大学院 社会産業理工学研究部
Graduate School of Technology, Industrial and Social Sciences, Tokushima University

すると，テスト不能故障判定に多大な時間を必要とする．そのため，スキャン設計を施さずに高い故障検出効率を得るためのレジスタ転送レベルでのテスト容易化設計手法[6]とテスト生成法[6,7]や，テスト生成時間を削減するために，テスト不能故障をあらかじめ判定する手法[8-12]が提案されている．

テスト不能故障の一部を高速に判定する手法として，回路構造に着目してテスト不能故障を判定する FIRE アルゴリズム[8](以下，FIRE と略す)と拡張 FIRE アルゴリズム[9](以下，拡張 FIRE と略す)が提案されている．また，順序回路の機能動作において遷移しえない状態(到達不能状態)を判定し, 判定された到達不能状態を用いてテスト不能故障判定を行うアルゴリズムが提案されている[11,12]．特に, 文献[12]では FIRE と到達不能状態を用いたテスト不能故障判定を組み合せることにより，多くのテスト不能故障を判定できることが報告されている．

しかしながら，文献[11,12]の手法では，実行時間の制約や，扱うことのできる変数の数が少ないといった点から，大規模な回路に対して部分回路に分割して判定処理を行う必要がある．そのため，回路全体を対象とした判定が行われておらず，到達不能状態の一部を判定できない可能性があり，これが課題となっている．この課題を解決した手法として，文献[13]では，回路全体を対象とした到達不能状態判定法が提案されている．しかしながら, 文献[13]では，判定した到達不能状態を逐次的に制約として付与することができず，判定された到達不能状態からのみ到達可能な状態を判定できていない可能性がある．そのため，提案手法では扱うことのできる変数の数が多く，逐次的に制約付与が可能で，高速化の進む充足可能性問題(satisfiability problem：SAT)を用いる．

本論文では，文献[16]で提案したフリップフロップ組合せの状態正当化に着目した到達不能状態判定法，得られた到達不能状態を禁止状態制約とした順序回路のテスト不能故障判定法と FIRE 及び拡張 FIRE と組み合わせてテスト不能故障の判定数を評価する．

2. 従来のテスト不能故障判定法と到達不能状態判定法

本章ではテスト不能故障判定法[8,11]と到達不能状態判定法[13]の従来手法を説明する．2.1 節に回路構造に着目したテスト不能故障判定法である FIRE[8]について説明する．2.2 節に FF の値正当化を用いた到達不能状態判定法[13]と到達不能状態を用いたテスト不能故障判定法[11]について説明する．

2.1 回路構造に着目したテスト不能故障判定法

文献[8]で提案されている FIRE は，回路構造に着目し，ある信号線に 0, 1 両方の値を割当てる必要のある故障をテスト不能故障と判定する手法である．信号線が 0, 1 両方の値を同時に有することは不可能であり，相反する割当てを必要とする故障はテスト不能故障と判定することができる．

図 1 に FIRE によるテスト不能故障判定の例を示す. 図 1 の回路において信号線 a に 0,1 を割当てる必要のある故障を考える．また，図 1 中の set[a,0]は信号線 a に 0 を割当てなければ励起・伝搬できない故障集合，set[a,1]は信号線 a に 1 を割当てなければ励起･伝搬できない故障集合とする．また，a/1 は信号線 a の 1 縮退故障，a/0 は信号線 a の 0 縮退故障を表している．

まず信号線 a に 0 を割当てなければ励起できない故障集合を求める．このとき図 1(a)に示すように信号線 a に 1 を割当て，含意操作を行う．このとき，一意に値が割当てられる信号線に対し，割当てられた値と同値の縮退故障が発生したと仮定すると，それらの信号線の縮退故障は信号線 a に 0 を割当てなければ励起できない．そのため，信号線 a に 1 を割当て，含意操作を行うことで，信号線 a に 0 を割当てなければ励起できない故障集合が得られる．図 1(a)の例では信号線 a の 1 縮退故障，信号線 a_c の 1 縮退故障，信号線 f の 1 縮退故障が，信号線 a に 0 を割当てなければ励起できない故障となる．

同様に, 信号線 a に 0 を割当て, 含意操作を行うことで, 信号線 a に 1 を割当てなければ励起できない故障集合を得ることができる．このときの含意操作の結果を図 1(b)に示す．この結果から信号線 a の 0 縮退故障，信号線 a_c の 0 縮退故障, 信号線 c の 1 縮退故障, 信号線 e の 0 縮退故障, 信号線 f の 0 縮退故障，信号線 g の 0 縮退故障が，信号線 a に 1 を割当てなければ励起できない故障となる．また，図 1(b)の含意操作結果と同じ値割当ての図 1(c)において，菱形で囲った信号線 a_c，c，e，f に着目すると，各ゲートに対する入力の制御値となっていることがわかる．そのため，図 1(c)の菱形に対する丸で囲った値割当ての行われていない信号線 b，d，e，f に 0 縮退故障，1 縮退故障があったとき，信号線 a に 0 が割当てられた場合，故障影響が伝搬不能なため，丸で囲われている信号線の 0 縮退故障，1 縮退故障は信号線 a に 1 を割当てなければ伝搬できない故障となる．

最後に 2 つの故障集合 set[a,0]と set[a,1]の積集合からテスト不能故障集合を得ることができる．図 1 の例では信号線 f の 1 縮退故障がテスト不能故障であると判定できる. 図 1 の例では信号線 a に着目していたが, FIRE では回路中のすべての信号線に対しテスト不能故障を判定する処理を行う．

また，文献[9]で提案されている拡張 FIRE は，FIRE を基に発展させた手法であり, FIRE よりも多くのテスト不能故障を判定可能である．本来の FIRE は順序回路の組合せ回路部分に対してテスト不能故障判定を行っているが，拡張 FIRE では, 順序回路を k 時間展開した回路モデルを用いて

テスト不能故障判定を行っている．また，FIRE ではテスト不能故障を同一信号線の矛盾する割当てを用いて判定を行っているが，拡張 FIRE では，判定対象として回路中に存在する多入力論理ゲートも判定対象に追加しており，
多入力論理ゲートの入出力信号線の矛盾する組合せを必須とする故障集合からテスト不能故障を判定する手法が提案されている．

2.2 FF の値正当化を用いた到達不能状態判定法と到達不能状態を用いたテスト不能故障判定法

文献[13]は，回路内に存在するファンアウトが到達不能状態を引き起こす原因であることを示し，ある FF に割当てた値に対する正当化を行い，求められた正当化結果を用いて到達不能状態を判定する手法を提案している．また，文献[11]で提案されている到達不能状態を用いたテスト不能故障判定法は，判定された到達不能状態を用いた場合のみ検出可能な故障をテスト不能故障として判定する手法である．

図 2 に文献[13]の手法による到達不能状態判定の例を示す. 図 2 の回路において各 FF の入力信号線に 0,1 を割当て，値の正当化を行い, 正当化結果から到達不能状態を求める．

文献[13]では，対象回路の FF の入出力線を擬似外部出力・入力とした組合せ回路を用いて到達不能状態判定を行っている．

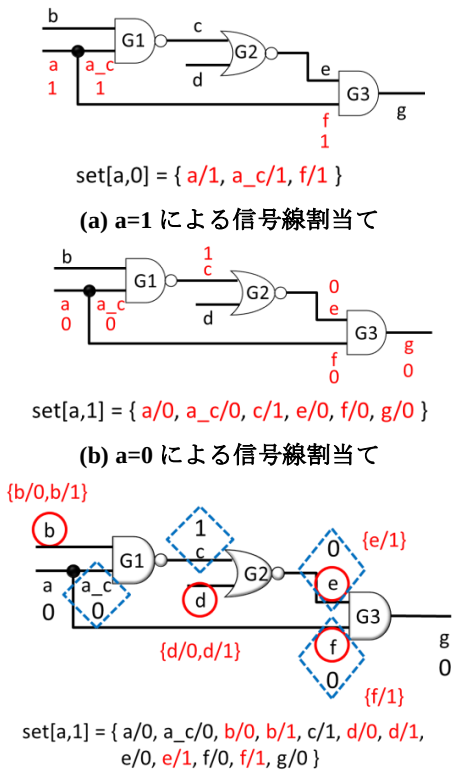


図 1. FIRE によるテスト不能故障判定例

図 2 に，FF0 の入力信号線である f0 に 1，FF1 の入力信号線である f1 に 1 を割当て，正当化を行った例を示す．

まず，f0 に 1 を割当て，正当化を行った結果，ファンアウトステムである信号線 in0 に 0，信号線 in1 に 0，信号線 in2 に 1 が割当てられた．

同様に，f1 に 1 を割当て，正当化を行った結果，ファンアウトステムである信号線 in0 に 0，信号線 in1 に 1，信号線 in2 に 0 が割当てられた．

この結果から FF0 に 1 を割当てたときの正当化結果と FF1 に 1 を割当てたときの正当化結果を比較すると，ファンアウトステムである信号線 in1 と信号線 in2 で値が相反していることが分かる．これより，FF0 と FF1 を同時に 1 に正当化することができないことがわかる．そのため, FF0 と FF1 に 1 が割当てられている状態は到達不能状態であることがわかる．

表 1 に文献[10]の手法による到達不能状態を用いたテスト不能故障判定例を示す. 表 1 は FF0 の出力信号線 F0 に 1 を割当てる必要がある故障集合(set[F0,1])，FF1 の出力信号線 F1 に 1 を割当てる必要がある故障集合(set[F1,1])をまとめたものである．

set[F0,1]と set[F1,1]の 2 つの故障集合に対し積集合をとることで，FF0 と FF1 ともに 1 が割当てられているときのみ検出可能な故障集合を得ることができる．しかしながら，前述の通り FF0 と FF1 をともに 1 に割当てることはできないため，FF0 と FF1 ともに 1 が割当てられているときのみ検出可能な故障集合に含まれている故障はテスト不能故障と判定することができる．表 1 の例では，信号線 g13_o の 0 縮退故障がテスト不能故障であると判定できる．

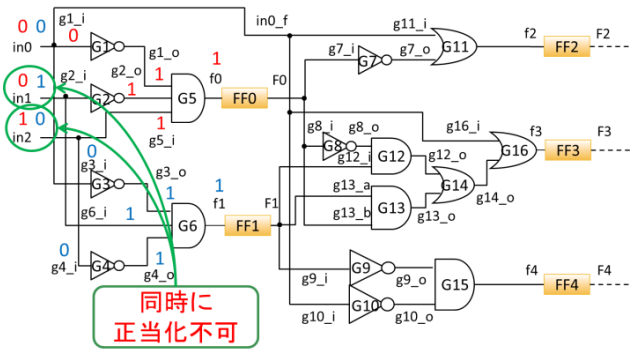


図 2. FF の値正当化による到達不能状態判定例

表 1. 到達不能状態を用いたテスト不能故障判定

set[F0,1]	F0/0, g7_i/0, g7_o/1, f2/1, F2/1, g8_i/0, g8_o/1, g13_b/0, **g13_o/0**
set[F1,1]	F1/0, g12_i/0, g12_o/0, g13_a/0, **g13_o/0**, g9_i/0, g9_o/1, g14_o/0

3. フリップフロップ組合せの状態正当化に基づくテスト不能故障判定法

文献[11,12]の到達不能状態判定法では，実行時間の制約や，扱うことのできる変数の数が少ないといった点から，大規模な回路に対して部分回路に分割して判定処理を行う必要がある．そのため，回路全体を対象とした判定が行われておらず，到達不能状態の一部を判定できない可能性があり，これが課題となっている．この課題を解決した手法として，文献[13]で回路全体を対象とした到達不能状態判定法が提案されている．しかしながら，文献[13]では，判定した到達不能状態を逐次的に制約として付与することができず，判定された到達不能状態からのみ到達可能な状態を判定できていない可能性がある．

一方，近年では衝突項の学習[14]や非辞書式バックトラック[15]，ブール制約伝搬[15]の技術により，SAT ソルバーの処理速度が急速に進歩している．また，文献[11,12]で到達不能状態判定の際に用いている二分決定グラフ(Binary Decision Diagram：BDD)では数百変数の論理式を扱うのが限度であるが，SAT では数百万変数からなる論理式を扱うことができる．さらに，充足可能性判定を行う CNF 式に対して制約節を追加することで，逐次的に制約を付与することが可能である．そのため，SAT を用いることで大規模な回路において回路全体に対する判定を行うことができ，制約を逐次的に付与しながら，高速に充足可能性判定を行うことができる．

本章では文献[16]で提案したフリップフロップ組合せの状態正当化に着目した到達不能状態判定法と，到達不能状態を禁止状態制約とした順序回路のテスト不能故障判定法を説明する．

3.1 フリップフロップ組合せの状態正当化による到達不能状態判定法

本節で説明するアルゴリズムでは，FF ペア(トリプル)と k 時間展開モデルを用いている．FF ペア(トリプル)とは，順序回路内に存在する 2 個(3 個)の FF の組合せのことである．また，k 時間展開モデルとは順序回路の組合せ回路部の接続関係を k 時間展開した回路である．順序回路は時間展開モデルで表現することによって，組合せ回路として扱うことができる．

以下に，SAT を用いた到達不能状態判定法のアルゴリズムについて説明する．

(Step1)

回路内に存在する FF ごとに影響外部入力{論理関数内に含まれている(擬似)外部入力}を調べる．

(Step2)

FF ペア(トリプル)の作成を行い，FF ペア(トリプル)同士の共通の影響外部入力が存在するか調べ，共通の影響外部入力が存在していない FF ペア(トリプル)を FF ペア(トリプル)集合から削除する．

(Step3)

対象となる k 時間展開モデルの CNF 式を論理ゲートの CNF 変換規則に従い生成する．

(Step4)

Step3 で生成した k 時間展開モデルの CNF 式に共通の影響外部入力が存在している FF ペア(トリプル)のそれぞれの状態を制約として与え，充足可能性判定を行う．

充足不能性判定後，充足不能となった FF ペア(トリプル)の状態を用いて，CNF 式に対して禁止状態制約を追加する．

(Step5)

充足不能となった FF ペア(FF トリプル)の状態数の比較を行う．新たに充足不能となった FF ペア(トリプル)の状態が存在した場合，再度全ての FF ペア(FF トリプル)の状態に対して充足可能性判定を行う．

(Step6)

充足不能と判定された FF ペア(トリプル)の状態を用いて到達不能状態の識別を行う．

図 3 を用いて本手法の到達不能状態判定例を説明する．

まず，FF ペアの生成を行う．例では FF ペアとして生成されたペアである(G10,G11)が選択されている．なおこの例では FF ペア共通の影響外部入力数の探索を省略している．

次に選択した FF ペアがとりうる状態の一つを選択し，選択した状態を正当化できるかどうか判定する．例では，FF ペア(G10,G11)の状態(1,1)を正当化する際に矛盾が発生しているため，FF ペア(G10,G11)の状態(1,1)が正当化不能であることを示している．また，正当化不能と判定された FF ペアとその状態を用いて禁止状態制約を付与する．例では，FF ペア(G10,G11)の状態(1,1)が正当化不能だったため，(G10,G11)と接続関係にある(G5,G6)に対して状態(1,1)となるような割当てを禁止する制約を付与している．

最後に正当化不能と判定された FF ペアとその状態を用いて到達不能状態を識別する．例では，FF ペア(G10,G11)が状態(1,1)だったとき正当化不能と判定されたため，(G10,G11)=(1,1)という割当てとなっている状態が到達不能状態と識別することができる．

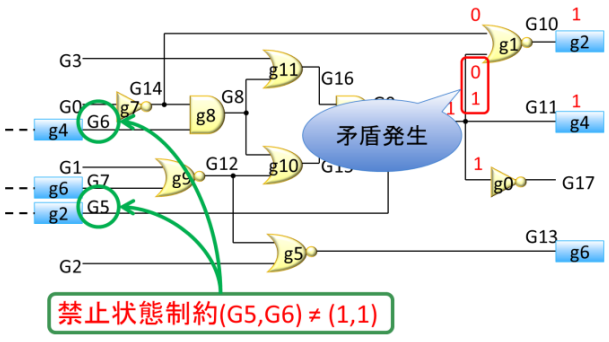


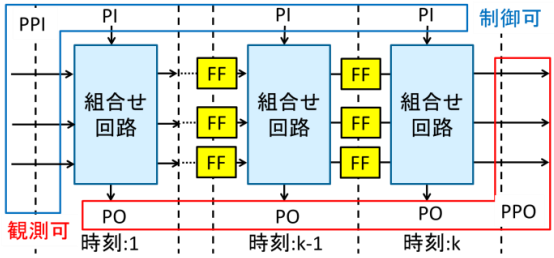
図 3. 到達不能状態判定例

例として対象回路の FF 数が 100 個だった場合，正当化不能になった FF ペアとその状態を用いることで2^{98}の状態が到達不能状態であることがわかる．

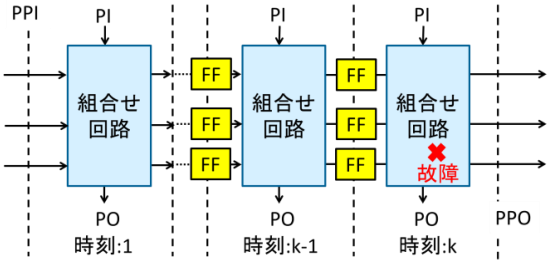
本手法の到達不能状態判定では，FF ペア(トリプル)を用いることで，充足不能となった FF ペア(トリプル)の状態というわずかな情報で多数の到達不能状態を表すことが可能である．また，判定された到達不能状態を禁止状態制約として付与することで，既知の到達不能状態のみから到達可能な状態を新たに到達不能状態と判定することが可能となる．さらに，FF ペア(トリプル)同士の共通の影響外部入力が存在するか調べることで，充足不能と判定される可能性がない FF ペア(トリプル)を制限することが可能である．

3.2 到達不能状態を禁止状態制約とした順序回路のテスト不能故障判定法

提案手法のテスト不能故障判定法では，対象回路モデルとして，テスト不能故障判定のための k 時間展開モデル[10]を用いている．また，対象故障モデルを単一故障モデル[10]とした．



(a) テスト不能故障判定のための k 時間展開モデル



(b) 単一故障モデル

図 4. 対象回路モデルと対象故障モデル

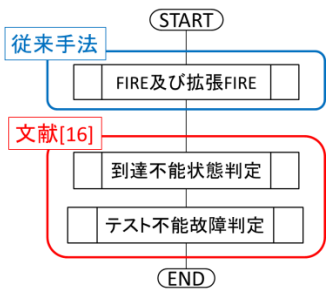


図 5. 本手法のテスト不能故障判定フロー

図 4(a)にテスト不能故障判定のための k 時間展開モデルについて，図 4(b)に単一縮退モデルについて示す．テスト不能故障判定のための k 時間展開モデルは，順序回路に対して k 時間展開を行い，各時刻の外部入力および 1 時刻目の疑似外部入力を制御可能とし，各時刻の外部出力および k 時刻目の疑似外部出力を観測可能としたモデルである．また，単一故障モデルはテスト不能故障判定のための k 時間展開モデルの k 時刻目にのみ故障を仮定し，それ以外の時刻は正常回路とする故障モデルである．単一故障モデルは縮退故障モデルを仮定しており，この故障モデルにおいてテスト不能な故障は，順序回路においてもテスト不能と判定することができる[10].

図 5 に本手法のテスト不能故障判定の流れを示す．また，本手法の到達不能状態を禁止状態制約とした順序回路のテスト不能故障判定の流れは以下の通りである．

(Step1)

テスト不能故障判定のための k 時間展開モデルの CNF 式を論理ゲートの CNF 変換規則に従い生成する．

(Step2)

Step1 で生成された CNF 式に前述の手法で判定した到達不能状態を禁止状態制約として付与する．

(Step3)

与えられた代表故障集合に対し，Step2 で生成された CNF 式と SAT ソルバーを用いてテスト生成を行う．このとき，充足不能と判定された故障はテスト不能故障と判定する．

4. 実験結果

フリップフロップ組合せの状態正当化による到達不能状態判定法，到達不能状態を禁止状態制約とした順序回路のテスト不能故障判定法を C 言語で実装した．

実験環境は Windows 8.1，メモリ 8GB，プロセッサ : Intel Core i7-4790，3.6GHz．対象回路は ISCAS’89 及び ITC’99 ベンチマーク回路である．

FF 数が 21 以下の回路に対しては 2 時間展開モデルで FF ペアと FF トリプルを探索し，到達不能状態判定を行った．FF 数が 21 を超える回路に対しては 2 時間展開モデルで FF ペアを探索し，到達不能状態判定及びテスト不能故障判定を行った．また，FIRE 及び拡張 FIRE は時間展開数を 5 に設定し，テスト不能故障判定を行った．

表 2 に FF 数 21 以下の回路に対する到達不能状態判定数を示す．表 2 は左から回路名，FF 数，全内部状態数，文献[12]の手法によって判定された到達不能状態数，文献[16]で提案したフリップフロップ組合せの状態正当化に着目した到達不能状態判定法によって判定された到達不能状態数，文献[16]で提案した到達不能状態判定法の実行時間である．この結果から，s208 と s420 の回路では文献[12]の手法と同等の判定結果が得られたが，s298，s444，s526，s1196，s1238

の回路では文献[12]の手法よりも少ない判定結果となっている．これは文献[12]の手法で判定された到達不能状態に FF ペアと FF トリプルといった FF の組合せだけでは判定できない到達不能状態が存在しているためだと考えられる．

表 3 に文献[9]と本手法のテスト不能故障判定法により判定されたテスト不能故障数を示す．表は左から回路名，代表故障数，文献[9]で提案されている拡張 FIRE によって判定されたテスト不能故障数，本手法で実装した到達不能状態を禁止状態制約とした順序回路のテスト不能故障判定法と本手法で実装した FIRE 及び拡張 FIRE を組合せることによって判定されたテスト不能故障数，文献[9]の実行時間，本手法の実行時間，判定数増加率である．この結果から，ISCAS’89 回路に対し，文献[9]と比較して最大 67.9%，平均 41.9%，多くのテスト不能故障を判定できた．これは回路構造だけでなく，順序回路の通常の機能動作において遷移しえない状態である到達不能状態に着目してテスト不能故障判定を行ったためだと考えられる．

5. むすび

本論文では, 到達不能状態判定のアルゴリズムを提案し，到達不能状態を用いたテスト不能故障判定を行った．実験結果では，いくつかの回路で従来手法と同等の到達不能状態を判定できた．また，文献[9]と比較し，最大 67.9%，平均 41.9%，多くのテスト不能故障を判定できた．

今後の課題として，より多くの到達不能状態を判定するために FF の組合せの数を増加させること，テスト不能故障判定のための k 時間展開モデルの時間展開数を増加させること，FIRE 及び拡張 FIRE を実行する際の間接含意量を増加させること，より多くの到達不能状態及びテスト不能故障を判定できるようなアルゴリズムの提案が挙げられる．

参考文献

1) H. Fujiwara, *Logic Testing and Design for Testability*, The MIT Press, 1985.
2) C. Wang, S. Reddy, I. Pomeranz, X. Lin, and J. Rajski, "Conflict driven techniques for improving deterministic test pattern generation," IEEE International Conference on Computer-Aided Design, pp. 87-93, Nov. 2002.
3) B. Yang, K. Wu, and R. Karri, "Scan Based Side Channel Attack on Dedicated Hardware Implementations of Data Encryption Standard," IEEE International Test Conference, pp. 339-344, Oct. 2004.
4) D. Xiang, Y. Xu, and H. Fujiwara, “Non scan Design for Testability for Synchronous Sequential Circuits Based on Conflict Resolution,” IEEE Trans. on Computers, Vol. 52, No. 8, pp. 1063-1075, Aug. 2003.
5) Y. C. Lin, F. Lu, and K. T. Cheng, "Pseudo Functional Testing,” IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, Vol. 25, No. 8, pp. 1535-1546, Aug. 2006.
6) K. Sugiki, T. Hosokawa, and M. Yoshimura, “A Test Generation Method for Data path Circuits Using Functional Time Expansion Models,” IEEE Workshop on RTL and High Level Testing, pp. 39-44, Nov. 2008.
7) T. Masuda, J. Nishimaki, T. Hosokawa, and H. Fujiwara, “A Test Generation Method for Data Paths Using Easily Testable Functional Time Expansion Models and Controller Augmentation,” IEEE Asian Test Symposium, pp. 37-42, Nov. 2015.
8) M. A. Iyer, and M. Abramvici, “FIRE: a Fault Independent Combinational Redundancy Identification Algorithm,” IEEE Trans. on VLSI systems, Vol. 4, No. 2, pp. 295-301, June 1996.
9) M. Syal, and M. S. Hsiao, “New Techniques for Untestable Fault Identification in Sequential Circuits,” IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, Vol. 25, No. 6, pp. 1117-1131, June 2006.
10) V. D. Agrawal, and S. T. Chakradhar, “Combinational ATPG Theorems for Identifying Untestable Faults in Sequential Circuits,” IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems , Vol. 14, No. 9, pp. 1155-1160, Sep. 1995.
11) H. Yotsuyanagi, and K. Kinoshita, “Undetectable Fault Removal of Sequential Circuits Based on Unreachable States,” IEEE VLSI Test Symposium, pp. 176-181, Apr. 1998.
12) D. E. Long, M. A. Iyer, and M. Abramovici, “FILL and FUNI : Algorithms to Identify Illegal States and Sequentially Untestable Faults,” ACM Trans. on Design Automation of Electronic Systems, Vol. 5, No. 3, pp. 631-657, July 2000.
13) F. Yuan, and Q. Xu, “On Systematic Illegal State Identification for Pseudo-Functional Testing,” ACM/IEEE Design Automation Conference, pp. 702-707, July 2009.
14) J. P. Marques-Silva, and K. A. Sakallah, “GRASP: A Search Algorithm for Propositional Satisfiability,” IEEE Trans. on Computers, Vol. 48, No. 5, pp. 506-521, May 1999.
15) M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik, “Chaff: Engineering an Efficient SAT Solver,” IEEE Design Automation Conference, pp. 530-535, June 2001.
16) 二関森人，細川利典，吉村正義，新井雅之，四柳浩之，橋爪正樹, “到達不能状態を用いた SAT ベース順序回路のテスト不能故障判定法,” 信学技報, vol. 116, no. 466, DC2016-79, pp. 29-34, 2017.

表 2. 到達不能状態判定結果

回路名	FF数	全内部状態数	到達不能状態判定数		提案手法実行時間 (sec)
			文献[13]	提案手法	
s208	8	256	239	239	1.6
s298	14	16,384	16,166	14,968	38.3
s420	16	65,536	65,519	65,519	187.2
s444	21	2,097,152	2,088,287	2,032,232	541.0
s526	21	2,097,152	2,088,284	1,503,232	404.6
s641	19	524,288	517,827	517,632	284.3
s713	19	524,288	517,827	517,632	307.3
s1196	18	262,144	259,528	254,334	147.0
s1238	18	262,144	259,528	254,334	153.0

表 3. テスト不能故障判定結果

回路名	代表故障数	テスト不能故障判定数		実行時間(sec)		判定数増加率 (%)
		文献[10]	本手法	文献[10]	本手法	
s9234	6,927	433	727	157.3	515.1	67.9
s13207	9,815	822	1,345	201.8	43537.6	63.6
s38417	31,180	491	604	570.3	10931.3	23.0
s38584	36,303	1,916	2,166	2014.1	244372.5	13.1
b14	22,802	N/A	157	N/A	5478.3	N/A
b15	21,988	N/A	906	N/A	9265.1	N/A
b20	45,459	N/A	320	N/A	42336.1	N/A
b21	46,154	N/A	379	N/A	51766.7	N/A
b22	67,536	N/A	344	N/A	102880.2	N/A