

FIRE アルゴリズムを用いた順序回路のテスト不可能故障判定の評価

日大生産工(学部) 〇二関 森人 日大生産工 山崎 紘史
日大生産工 細川 利典 京都産大 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、超大規模集積回路(Very Large Scale Integrated circuit: VLSI)が大規模化・複雑化し、テストパターン生成時間の増加が問題となっており、テストパターン生成の高速化、容易化が求められている。

テストパターン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)を外から制御、観測可能とするスキャン設計[1, 2]が提案されている。しかしながら、スキャン設計が施された回路は容易に内部状態を外部に出力することが可能なため、セキュリティを考慮したテスト容易化設計の回路に対してはスキャン設計を行うことにより、機密情報の流出などに繋がる可能性が高くなる。そのため、内部状態を外部から直接的に制御、観測を出来ない条件の元でテストパターン生成を行う非スキャンベーステスト[3]が提案されている。

スキャンベースのテストパターン生成では FF の出力信号線を疑似外部入力(Pseudo Primary Input: PPI)、FF の入力信号線を疑似外部出力(Pseudo Primary Output: PPO)として組合せ回路と同様のテストパターン生成が可能である。しかしながら、非スキャンベースのテストパターン生成では、FF の出力信号線に対して値割当てが発生した場合、一時刻前の回路(FF の入力信号線)に対して正当化処理を適用する必要がある。そのため順序回路のテストパターン生成アルゴリズム[4]を用いる必要がある。

順序回路のテストパターン生成では、内部状態を外入力のみで設定するため、スキャン設計なしでは高い故障検出率を得ることが困難である。また、テスト不可能故障に対してテストパターン生成を行おうと、テスト不可能故障判定に多大な時間を必要とする。そのため、テストパターン生成時間を削減するために、テスト不可能故障をあらかじめ判定する様々な手法が提案されている[5-8]。

本論文では、従来のテスト不可能故障を判定する手法として、回路構造に着目し含意操作による不可

能状態からテスト不可能故障を判定する FIRE アルゴリズム[5]と拡張 FIRE アルゴリズム[6]を実装し、その評価を行う。

本論文の構成は以下の通りである。第 2 章では、今回評価を行うテスト不可能故障判定法である FIRE 及び拡張 FIRE アルゴリズムで用いる直接含意操作[2][3]について述べる。第 3 章では FIRE 及び拡張 FIRE アルゴリズムを高速化するために用いる静的学習[3]について述べる。第 4 章では FIRE アルゴリズムについて述べる。第 5 章では拡張 FIRE アルゴリズムについて述べる。第 6 章では実験結果を示し、第 7 章では本論文についてまとめる。

2. 直接含意操作

直接含意操作とは、ゲートの入出力信号線の接続関係から値を求めることである。また、直接含意操作は前方含意操作と後方含意操作に分けられる。図 1 に 2 入力 AND ゲートに対する直接含意操作の例を示す。図 1(a)は前方含意操作の例である。AND ゲートは一方の入力に 0 が割当てられたとき出力が 0 となるため、信号線 a の値が 0 のとき出力信号線 c の値は 0 となる。このように信号線の入力側から出力側に値を決定する処理が前方含意操作である。図 1(b)は後方含意操作の例である。AND ゲートの出力が 1 となるためには全ての入力が 1 でなくてはならないため、信号線 c の値が 1 のときは信号線 a, b の値は 1 となる。このように出力側から入力側に値を決定する処理が後方含意操作である。

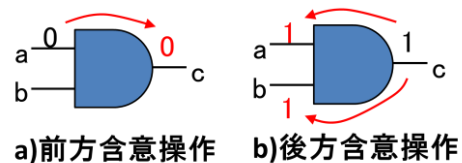


図 1. 直接含意操作の例

An Evaluation of Untestable Fault Identification for Sequential Circuits Using FIRE Algorithm

Morito NISEKI, Hiroshi YAMAZAKI, Toshinori HOSOKAWA and Masayoshi YOSHIMURA

3. 静的学習

静的学習とは直接含意の結果から固定値、接含意を求める処理で、繰り返し実行することでより多くの間接含意を得られる。静的学習の処理では回路内の各信号線について、その信号線の値を0に設定した含意操作と1に設定した含意操作をそれぞれ実行する。本論文では静的学習で2つの学習規則と対偶規則、固定値学習を用いる。3章では3.1節で学習規則を説明し、3.2節で対偶規則について説明する。3.3節では固定値学習について説明する。

3.1. 学習規則

学習規則とは、静的学習を行うことにより、直接含意の結果から間接含意を保持する条件である本論文では、学習規則Aまたは学習規則Bのどちらか一方を満たす直接含意が存在する場合、間接含意を保持する。なお、それぞれの間接含意の保持については対偶規則を適用させている。

[学習規則A] 信号線 s の値を v ($v \in \{0, 1\}$) とする含意操作により、ある2入力以上のゲートの出力信号線 t が値 w ($w \in \{0, 1\}$) をとり、かつ、 w がそのゲートの非制御値であるとき、間接含意として “ $t = \bar{w}$ ならば $s = \bar{v}$ ” を保存する。ここで、非制御値とは、AND ゲートの出力に対する値である1のように、ゲート入力値がすべて決定されたときにゲート出力値が決定される値のことである。

[学習規則B] 信号線 s の値を v ($v \in \{0, 1\}$) とする含意操作において、後方への含意操作により、ある2入力以上のゲート入力信号線 t の値が w ($w \in \{0, 1\}$) をとり、かつ、 w がそのゲートの制御値であるとき、間接含意として “ $t = \bar{w}$ ならば $s = \bar{v}$ ” を保存する。

3.2. 対偶規則

対偶規則とはある命題が真のとき、対偶も真であるという規則である。図2に対偶規則の例を示す。はじめに、図2のように信号線 b の値を0とし、前方含意を行うと信号線 f の値が0となる。これにより “ $b=0$ ならば $f=0$ ” という命題が真になる。対偶規則を適用すると “ $f=1$ ならば $b=1$ ” という命題が成り立ち、間接含意を得ることができる。

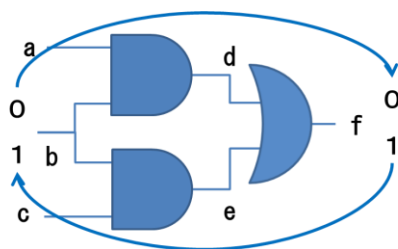


図2. 対偶規則の例

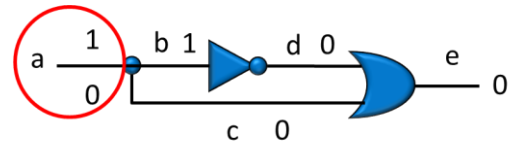


図3. 固定値学習の例

3.3. 固定値学習

固定値学習とは含意操作中に値の矛盾が発生した場合、はじめに信号線に割当てた値と逆の値を固定の値として決定するものである。図3に固定値学習の例を示す。信号線 e の値を0とし後方含意を行うと、信号線 d , c , a の値は0となるが、信号線 d は NOT ゲートのため信号線 b , a の値が1となり、信号線 a において値が矛盾することがわかる。これにより信号線 e に0を割当てることは不可能なため、信号線 e の値は1に固定される。

4. FIRE アルゴリズム

文献[5]で提案されている FIRE アルゴリズムは、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可能故障と判定する手法である。ここでは、ある信号線 x に0を割当てる必要がある故障集合を $\text{set}[x, 0]$, 1を割当てる必要がある故障集合を $\text{set}[x, 1]$ と表す。また拡張 FIRE アルゴリズムにおいても同様に示す。

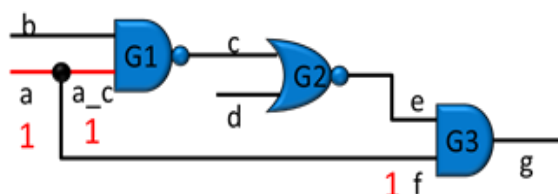
図4に FIRE によるテスト不可能故障判定の例を示す。図4の回路において信号線 a に値0と1の両方を割当てる必要のある故障を考える。まず信号線 a に0を割当てる必要のある故障集合($\text{set}[a, 0]$)を求める。このとき図4(a)に示すように a に1を割当てて含意操作を行う。 $a=1$ より一意に値が決定する信号線は、一意に決定する値を故障値とする縮退故障の検出が不可能である。そのため、これらの故障は $a=1$ が原因で検出不可能な故障であり、 $a=0$ の割当てが故障検出に必要な故障である。図4(a)において、信号線 a の1縮退故障(以下、 $a/1$ と略)、 $a_c/1$, $f/1$ が、 $a=0$ が故障検出に必要な故障集合である。そのため、 $\text{set}[a, 0]$ は $a/1$, $a_c/1$, $f/1$ と求まる。また、図4(b)に示すように、 $a=1$ を必要とする故障も信号線 a に0を割当てて含意操作を行い、故障集合 $\text{set}[a, 1]$ を得る。このとき $\text{set}[a, 1]$ より、各ゲートの制御値となっている値がないかを調べると、信号線 a_c の値に0が割当てられていることから、信号線 a_c の値がゲート $G1$ の制御値となっていることが分かる。同様に、信号線 c の値にゲート $G2$ の制御値、信号線 e , f の値にゲート $G3$ の制御値が割当てられていることが分かる。このことから、各ゲートにおいて制御値が割当てられていない入力信号線に縮退故障があった場合、制御値によって伝搬不可能となる。そのため、 $a=1$ を必要とする故障集合($\text{set}[a, 1]$)は図4(b)に描かれた信号線値だけでなく、 $b/0$, $b/1$, $d/0$, $d/1$, $e/1$, $f/1$ も含まれる。そのため、 $\text{set}[a, 1]$ は $a/0$, $a_c/0$,

b/0, b/1, c/1, d/0, d/1, e/0, e/1, f/0, f/1, g/0 と求まる．そして $\text{set}[a, 0]$ と $\text{set}[a, 1]$ の積集合から，図 4(c) に示すように f/1 がテスト不可能故障であると判定できる．

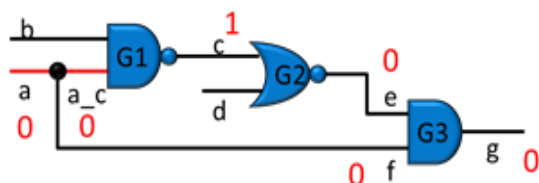
5. 拡張 FIRE アルゴリズム

第 4 章で述べた FIRE アルゴリズムは各信号線における 0, 1 を必要とする故障をテスト不可能故障と判定したのに対し，文献[6]の拡張 FIRE アルゴリズムでは論理ゲートの不可能状態からテスト不可能故障を判定する．論理ゲートの不可能状態の例として，2 入力 AND ゲートの不可能状態の例を図 5 に示す．

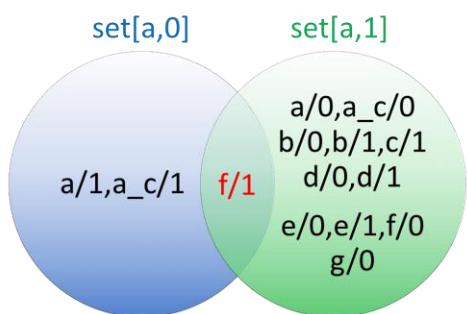
図 5 に示したように，論理ゲートの不可能状態とは多入力ゲートに対し起こるものである．不可能状態の種類としては，図 5 の不可能状態:1 と不可能状態:2 のようにゲートのいずれかの入力値が制御値であり，ゲートの出力値がゲートのすべての入力が非制御値のときに決定する値となっている状態と，図 5 の不可能状態:3 のようにすべての入力が非制御値であり，出力値を入力が制御値のときに決定する値となっている状態の 2 種類がある．



(a) $a=1$ による信号線割当て



(b) $a=0$ による信号線割当て



(c) テスト不可能故障

図 4. FIRE によるテスト不可能故障判定

しかしながら，不可能状態:1 のときに信号線 $a=0$ と信号線 $c=1$ から含意操作を行うと， $a=0 \Rightarrow c=0$ ， $c=1 \Rightarrow a=1$ といった関係が出てくる．このことから，この不可能状態によって得られるテスト不可能故障集合は $(\text{set}[a,0] \cap \text{set}[a,1]) \cup (\text{set}[c,0] \cap \text{set}[c,1])$ となり，これは FIRE アルゴリズムによって判定されていることが分かる．これにより，拡張 FIRE アルゴリズムに用いる不可能状態は図 5 の不可能状態:3 のようなすべての入力が非制御値であり，出力値を入力が制御値のときに決定する値となっている状態となる．

表 1 を用いて拡張 FIRE アルゴリズムの例を示す．表 1 は図 6 の G1 における各入出力信号線の 0 割当てを必要とする故障集合 $(\text{set}[a, 0], \text{set}[b, 0], \text{set}[c, 0])$ ，1 割当てを必要とする故障集合 $(\text{set}[a, 1], \text{set}[b, 1], \text{set}[c, 1])$ をまとめたものである．

表 1 に示した故障集合に対し FIRE アルゴリズムを用いた場合， $\text{set}[a, 0]$ と $\text{set}[a, 1]$ の積集合と， $\text{set}[b, 0]$ と $\text{set}[b, 1]$ の積集合と， $\text{set}[c, 0]$ と $\text{set}[c, 1]$ の積集合は，空集合となるためテスト不可能故障を判定することができない．しかしながら，拡張 FIRE アルゴリズムを用いた場合，すべての入力の非制御値と入力値が制御値により決定する出力から求めるため，G1 の場合は $\text{set}[a, 1], \text{set}[b, 1], \text{set}[c, 0]$ の 3 つの故障集合の積集合を求めることで，テスト不可能故障を判定することができる．今回の例では表 1 に示すように，この 3 つの積集合から h の 1 縮退故障がテスト不可能故障であると判定できる．

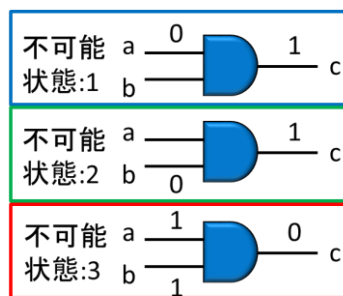


図 5. 2 入力 AND ゲートの不可能状態例

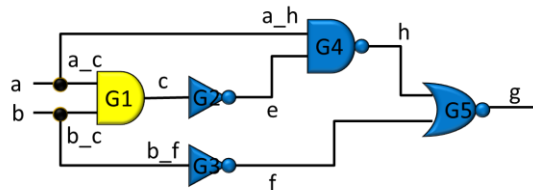


図 6. 拡張 FIRE アルゴリズム例題回路

表 1. AND ゲートに対する拡張 FIRE アルゴリズムによるテスト不可能故障判定

set[a,0]	a/1, a_c/1, a_h/1
set[a,1]	a/0, a_c/0, a_h/0, c/0, e/0, e/1, f/0, f/1, g/0, h/1
set[b,0]	b/1, b_c/1, b_f/1
set[b,1]	b/0, b_c, b_f/0, c/0, e/1, f/1, g/0, h/0, h/1
set[c,0]	a/1, b/1, a_c/1, a_h/1, a_h/0, b_f/1, b_c/1, c/1, e/0, f/0, f/1, g/0, h/0, h/1
set[c,1]	c/0, e/1

6. 実験結果

本論文ではテスト不可能故障の判定法として、FIRE アルゴリズム、拡張 FIRE アルゴリズムの評価を行った。実験対象回路は ISCAS'89 ベンチマーク回路に対して実験を行った。

表 2 に実験結果を示す。左から回路名、時間展開数、本論文で実装した FIRE アルゴリズムによるテスト不可能故障数、本論文で実装した拡張 FIRE アルゴリズムによるテスト不可能故障数、文献[6]による FIRE アルゴリズムでのテスト不可能故障数、文献[6]による拡張 FIRE アルゴリズムでのテスト不可能故障数である。表 2 から、ほとんどの回路において、文献[6]で示されているテスト不可能故障数と同数のテスト不可能故障が判定された。また、s526, s1238, s1423 等の回路では、文献[6]と本論文で判定されたテスト不可能故障数が異なる。これは、文献[6]と本論文で判定された静的学習の結果が異なるためと考えられる。また、s9234, s13207, s15850 においては、文献[6]において比較できる実験結果が記載されていないため、本論文の結果のみを記載している。

7. おわりに

本論文ではテスト不可能故障判定アルゴリズムである FIRE アルゴリズムと拡張 FIRE アルゴリズムを説明し、上記アルゴリズムを実装した。また、実装した FIRE アルゴリズムと拡張 FIRE アルゴリズムに対して実験を行い、文献[6]とのテスト不可能故障数についての評価を行った。今後の課題として、静的学習の精度を高めること、無効状態を用いたテスト不可能故障判定などのその他のテスト不可能故障判定アルゴリズムとの組合せを検討し、更に多くのテスト不可能故障の判定を行うことが挙げられる。

「参考文献」

[1]. H. Fujiwara “Logic Testing and Design for Testability”, The MIT Press, 1985

[2]. M. Abramovici, M. A. Breuer and A. D. Friedman “Digital systems testing and testable design”, IEEE Press, 1995

[3]. Dong Xiang, Yi Xu and H. Fujiwara “Non-Scan Design for Testability for Synchronous Sequential Circuits Based on Conflict Analysis”, Test Conference, 2000. Proceedings. International

[4]. Michael H. Schulz, Erwin Trischler, Thomas M. Sappert, “SOCRATES : a highly efficient automatic test pattern generation system , ” IEEE Transactions on Computer-Aided Design, Vol. 7, No1, pp. 126-137, Jan 1988.

[5]. M. A. Iyer, D. Long, and M. Abramovici, “FIRE : a fault independent combinational redundancy identification algorithm, ” IEEE Trans, VLSI systems, pp. 295-301, June 1996.

[6]. Michael S. Hsial, “Maximizing Impossibilities for Untestable Fault Identification, ” IEEE Press, pp. 949-953, Mar 2002.

[7]. Vishwani D. Agrawal and Srimat T. Chakradhar , “Combinational ATPG theorems for identifying untestable faults in sequential circuits, ” IEEE Trans, Vol. 14, pp. 1155-1160, Sep 1995.

[8]. Xiao Liu and Michael S. Hsiao , “Constrained ATPG for Broadside Transition Testing, ” IEEE Press, pp. 175-182, Nov 2003.

表 2. FIRE 及び拡張 FIRE アルゴリズムによるテスト不可能故障判定数

回路名	時間展開数	テスト不可能故障数			
		本論文 FIRE	文献[6] FIRE	本論文 拡張FIRE	文献[6] 拡張FIRE
s298	5	3	3	6	6
s344	3	3	3	4	4
s400	2	8	8	10	10
s526	5	6	2	11	11
s713	2	32	32	38	38
s1238	2	6	9	18	21
s1423	2	10	9	14	14
s5378	3	769	796	867	884
s9234	2	2987		3247	
s13207	3	4672		5278	
s15850	3	4231		4576	
s38417	3	335	328	470	466