社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2003－ＳＬＤＭ－112　(19)
2003／11／28

信号線間の含意関係に着目したフォールスパス検出手法

樋口　博之†　　松永　裕介††

† (株) 富士通研究所 CAD 研究部　〒211-8588 川崎市中原区上小田中 4-1-1
九州大学大学院システム情報科学府情報工学専攻　〒816-8580 福岡県春日市春日公園 6-1
†† 九州大学大学院システム情報科学研究院情報工学部門　〒816-8580 福岡県春日市春日公園 6-1
E-mail: †higuchi@labs.fujitsu.com, ††matsunaga@c.csce.kyushu-u.ac.jp

あらまし　含意関係などの信号線間の関係に着目して大規模な組み合わせ回路からフォールスパスを効率的に抽出する方法を提案する。本手法は、信号線間の関係から導かれたあり得ない値割り当てが間接的にフォールスパスの集合を表現するという考えに基づいて、パスの数え上げを行わずに、かつ遅延独立に、フォールスパスの検出を行う。
キーワード　フォールスパス、タイミング解析、組合せ回路、含意関係、含意操作、タイミング最適化

A False Path Detection Algorithm Based on Identification of Implication Relations

Hiroyuki HIGUCHI† and Yusuke MATSUNAGA††

† CAD Laboratory, Fujitsu Laboratories Ltd.
Kamikodanaka 4-1-1, Nakahara-ku, Kawasaki, 211-8588 Japan
Department of Computer Science and Communication Enginnering,
Graduate School of Information Science and Electrical Engineering, Kyushu University
†† Department of Computer Science and Communication Engineering,
Graduate School of Information Science and Electrical Engineering, Kyushu University
6-1 Kasuga Koen, Kasuga, Fukuoka, 816-8580 Japan
E-mail: †higuchi@labs.fujitsu.com, ††matsunaga@c.csce.kyushu-u.ac.jp

Abstract　In this paper we propose a false path detection method based on identification of relations between signals in combinational circuits. The method focuses on the idea that relations between signals implicitly represents sets of false paths and generates false paths without path enumeration in an delay-independent manner.
Key words　false path, timing analysis, combinational circuit, implication, timing optimization

1. はじめに

半導体集積回路の設計から遅延を見積り、正常に動作する範囲内であるかどうかを確かめるタイミング解析は、製造した回路が正しく動作することを保障するために必要不可欠な工程である。タイミング解析を行うには設計回路のネットリストだけでなく、回路のクロックの指定、モードの指定、遅延制約の指定、およびフォールスパスやマルチサイクルパスなどのタイミング例外パスの指定などを記述したタイミング制約の情報が必要である。設計回路のネットリストについては、論理合成ツールやレイアウトツールなどの CAD ツールの進歩によりある程度自動で生成が行えるようになっている。しかし、もう一方のタイミング制約の情報については、依然人手により作成する場合がほとんどである。従来は、タイミング制約の記述量がそれほど多くなく人手でも問題なく作成が行えていたが、近年以下のような点から人手による作成が困難になってきている。

- 回路の低消費電力化によりクロック数や回路のモード数が増え、タイミング制約が複雑化してきた。
- 回路の高速化のために、より正確に、すなわち、より多くのタイミング例外パス、いわゆるフォールスパスとマルチサイクルパス、を指定する必要性が増してきた。
- 回路規模の増大により、タイミング例外パスの長さも増大し、人手によるタイミング例外パスの確認が困難になってきた。

さらに、近年、回路の高速化の要求が高まり、論理合成とレイアウトを融合した物理設計でのタイミング最適化ツールが広く利用されるようになった [1]。これらのツールでタイミング最適化を行うには正確なタイミング制約情報が必要であり、でき

るだけ早期にタイミング制約の作成を収束させることが従来以上に望まれている。しかし、上述のように人手によるタイミング制約の作成はますます困難になっているため、集積回路の設計において以下のような問題が生じはじめている。

- タイミング制約が不完全なままタイミング最適化を行い、最適化に長い時間を要する。
- 長時間のタイミング最適化後にタイミング制約が修正され、それまでのタイミング最適化が無駄になる場合が少なくない。
- 長時間のタイミング最適化後にタイミングエラーパスを見直した結果そのパスがフォールスパスであることが判明する場合がある。

これらの問題はいずれもタイミング最適化中心の現在の典型的な設計フローに人手のみによるタイミング制約の生成が適合できていないことが原因であると考えられる。中でも、フォールスパス解析やマルチサイクルパス解析[2]などのタイミング例外パスの解析は人手では多くの手間がかかり、かつ間違いも混入しやすく、最もCADツールによるサポートが最も必要な部分の一つである。

フォールスパスの自動解析の研究は1980年代頃から始まり[3]、その後その理論的研究が盛んに行われてきた[4]~[7]。しかし、それらの研究は主に、より正確な解析を目指して行われ、現在、これらの方法を利用したフォールスパス解析は実際の設計ではあまり利用されていない。その原因の一つは遅延の考慮とパス数の数え上げでの組合わせ爆発により処理時間が長くなる点にある。また、自動でフォールスパスを生成した場合、処理しきれないほど多くのフォールスパスが生成されることが多く、フォールスパス制約を生成する際に、利用しやすさという点が考慮されていないという問題点もあった。

そこで、本稿では、タイミング重視の設計フローの流れの中でタイミング制約生成をある程度自動的に行えるようにすることを目指して、より実用的なフォールスパス解析手法を提案する。本手法は、回路中のあり得ない値割り当ての組合せが非明示的にフォールスパスの集合を表現するという考えに基づき、パスの数え上げや遅延の考慮を行わずに高速にフォールスパスを検出する。また、回路中のあり得ない値割り当ての組合せの要素数は、生成されるフォールスパス集合を指定する通過点の数に対応するため、値割り当ての組合せの数の少ないものから処理することで通過点指定数の少ないフォールスパスから順に生成することが可能となる。フォールスパスを利用するタイミング解析の計算量はフォールスパスの通過点指定数と通過点が指定する部分パスの長さに大きく依存するため、本手法で生成したフォールスパスはそれを利用するCADツールでの計算量の軽減につながる。

2. タイミング解析とフォールスパス

2.1 用語の定義

組合わせ回路のパスとは、外部入力から始まり、信号線とゲートが交互に現われ、最後に外部出力で終わる系列をいう。あるパス P に対して、P 上の各ゲートの入力のうち、P 上の入力を**on-input**といい、それ以外の入力を**side-input**という。

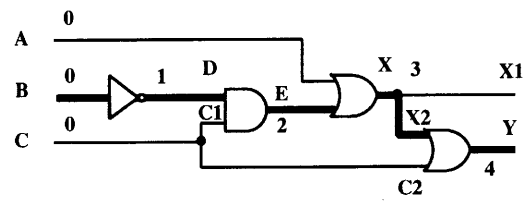


図1 フォールスパス

ゲートの出力を一意に決めるゲート入力値を**controlling value**という。controlling valueでない値を**non-controlling value**という。controlling valueにより決まるゲート出力の値を**controlled value**という。例えば、AND(OR)ゲートの場合はcontrolling valueは0(1)、non-controlling valueは1(0)、controlled valueは0(1)である。XORゲートはこれらの値を持たない。

回路の遅延と同じ遅延のパス、すなわち回路の遅延を与えるパスを**クリティカルパス**という。

2.2 タイミング解析

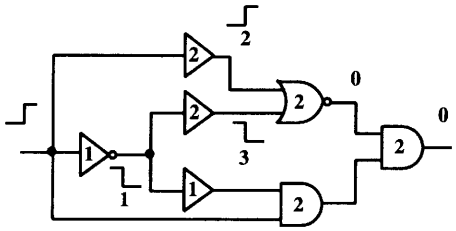
回路の遅延を計算して、フリップフロップ(FF)のセットアップ制約やホールド制約あるいは外部入出力からFFまでの遅延制約を満たすかどうか調べることを**タイミング解析**と呼ぶ。タイミング解析には、入力パタンを与えてそれに対する回路遅延を計算する**動的タイミング解析**と入力パタンなしに回路遅延の最悪値を見積もる**静的タイミング解析**の2つがある。以下、静的タイミング解析のことを単にタイミング解析と呼ぶ。

2.3 フォールスパス

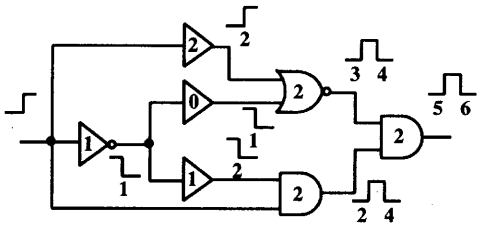
回路中の全てのパスが回路遅延に影響を与えるとは限らない[8]。パスの遅延の最大値を回路の遅延とするトポロジカルタイミング解析の問題点はその点を考慮できないことである。回路の遅延に影響を与えないパスのことを**フォールスパス(false path)**とよぶ。例えば、図1の回路では、B,D,E,X,X2,Yを通る太線で示されたパス P がフォールスパスである。なぜなら、$C=0$ なら side-input C_1 が controlling value となり P をブロックし、$C=1$ なら side-input C_2 が controlling value となり P をブロックするので、いずれにしても P がブロックされるからである(注1)。したがってこの回路の最大遅延はトポロジカルタイミング解析で得られる遅延値4ではなく3となる。

フォールスパスでないパス、すなわち、回路の遅延に影響を与えうるパスのことを**トゥルーパス(true path)**または**活性化可能パス(sensitizable path)**とよぶ。あるゲートの入力での信号遷移がそのゲートの出力に伝搬する条件をそのゲートの**活性化条件(sensitization condition)**という。あるパスが活性化可能であるとは、パス上の全てのゲートの活性化条件が同時に満たされるような場合が存在するということである。

(注1):厳密には、この条件だけでは後述するstatically sensitizableでないというだけで、必ずしもフォールスパスであるとは限らない。この例の場合、与えられたゲート遅延の条件からブロックするside-inputがon-inputより必ず早く到達するということも分かっているのでフォールスパスであると判定できる。



(a) 元の回路



(b) 真ん中のバッファの遅延を2から0に

図 2　トランジッションモードの問題点

逆に、フォールスパスとは、外部入力での信号の変化がどこかのゲートで必ず止まり、外部出力まで伝搬することが決してないパスである。

2.4　解析のモードと遅延モデル

2.4.1　トランジッションモードとその問題点

パスの活性化条件を考える場合、信号の遷移が伝搬するかどうかを調べるので、2つの連続する入力ベクトルに対する活性化条件を考えるのが自然である。この解析モードを**トランジッションモード**という。

トランジッションモードに基づくタイミング解析は問題があることが知られている。例えば、図2(a)の回路の遅延をトランジッションモードのもとで計算する。図2(a)中の各ゲート内の数字はそのゲートの遅延を表す。配線遅延は0とする。図2(a)では回路の遅延は0である。ここで、図2(b)のように真ん中のバッファの遅延が2から0にした場合を考えると、回路の遅延が0から6に増加する。すなわち、トランジッションモードかつ固定遅延モデル下では回路中の素子の遅延が減るとき回路の遅延が増えることは決してないという**モノトーンスピードアップ条件**が満たされない。

図2での問題点は、上限値のみ考慮しているため、NORゲートの下の方の入力での信号遷移が時刻2より早くなると出力で静的ハザードが生じるという現象を見逃しまうところにある。この点を解決する方法の一つは、固定遅延モデルでの遅延dのかわりに、区間$[0,d]$の任意の遅延をとりうるとする**モノトーンスピードアップ遅延モデル**[9](**XBD0 モデル (extended bounded delay-0 model)** [4] とも呼ばれる)を用いることである。しかし、トランジッションモードでこの遅延モデルを用いると計算時間が非現実的になる。

2.4.2　フローティングモード

トランジッションモードの問題点を解決するため、**単一ベクトル条件**で活性化条件を考える**フローティングモード**が用いられるようになった。

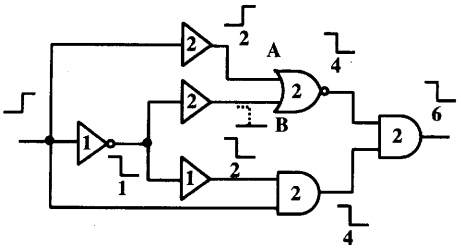


図 3　フローティングモードと固定遅延での計算[9]

フローティングモードは、2ベクトル条件の一つ目のベクトルが不定値、すなわち任意の値を取りうるものとして解析するモードである。このモードはトランジッションモードより悲観的であるが、固定遅延モデルとモノトーンスピードアップ遅延モデルが等価になるという好ましい性質をもつことが知られている。例えば図2(a)の回路の遅延をフローティングモードで計算すると、図3のようになる。図3のNORゲートの入力Bの点線の波形が実線の波形になっているように、NORの入力Aのcontrolling valueへの遷移が出力に伝搬するようにBの一つめのベクトルでの値もnon-controlling valueにすることができる。これはフローティングモードの、一つめのベクトルの値は任意の値にすることができるという性質を利用したものである。それに対して、モノトーンスピードアップ遅延モデルでは、あるゲートの遅延を任意の小ささにスピードアップすることで図2(a)のNORの出力の遷移が消えてしまうのを避けている。以下、タイミング解析のモードはフローティングモードを仮定する。

2.5　遅延非依存の活性化条件

遅延独立な活性化条件とは、遅延情報を利用しないでゲートの論理のみから判断する活性化条件である。代表的な遅延独立な活性化条件を以下に示す。

2.5.1　Static Sensitization

2.4.2節の議論から分かるように、side-inputがnon-controlling valueでさえあれば、フローティングモードの下ではその一つ前のベクトルでの値もnon-controlling valueとすることによって、必ず活性化可能である。**static sensitization**はこの考えに基づく活性化条件である[10]。

ある入力ベクトルvに対してパスπの全てのside-inputがnon-controlling valueになるとき、πはvにより**static sensitizable**であるという。例えば、図4では太線のパスB-E-F-G-Xの全てのside-input A, C1, Hがnon-controlling valueであるのでstatic sensitizableである。図4で「*」は不定値、すなわち0か1かは決まっていない、または、どちらでもよいことを示す。

static sensitizationはside-inputがcontrolling valueであればそのゲートで信号遷移がブロックされるという直感的に理解しやすい条件である。しかし、static sensitizationは楽観的な解析になる場合があることが知られている。図5で例を示す。太

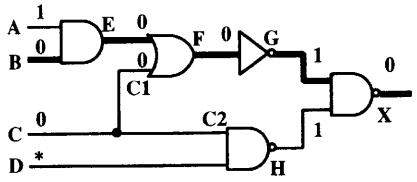


図 4　Static sensitization

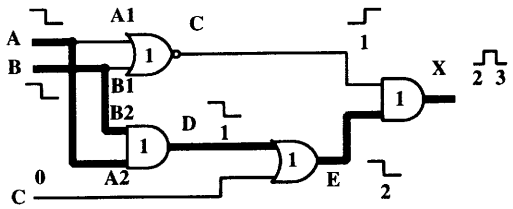


図 5　Static sensitization による false path が活性化できる例

線のパス A-A2-D-E-X, B-B2-D-E-X を statically sensitizable にするには、AND ゲート X での side-input が 1 でなければならない。しかし、C を 1 にすると $A=B=0$ となるため、パス A-A2-D-E-X に対しては side-input B2 が controlling value になり、パス B-B2-D-E-X に対しては side-input A2 が controlling value になるためどちらも statically sensitizable でない。すなわち、static sensitization ではフォールスパスとなる。しかし、実際には図 5 にあるように、A,B とも 1→0 に遷移する場合回路の遅延は 3 になり太線のパスは活性化可能であることがわかる。この点を解決する方法として次に述べる Brand と Iyenger の条件が提案された。

2.5.2　Brand と Iyenger の条件

前節で述べたように static sensitization の楽観性は、パス上のあるゲート入力で controlling value が複数ある場合、controlling value を持つ入力それぞれが他の controlling value を持つ入力にブロックされると考え、結局 controlling value を持つ入力全てが活性化不可能としてしまうことが原因である (例えば図 5 の AND ゲート D)。実際には、そのゲートの出力では信号遷移が伝搬するのでそれらのうち少なくとも一つは活性化されるはずである。そこで Brand と Iyenger の方法では各ゲートの入力に優先順序をつけておき、controlling value を持つゲート入力のうちその優先順序の一番高い入力が活性化するとして static sensitization の楽観性をなくすように改良した [3]。例えば図 5 の例では例えば AND ゲート D の入力を A2,B2 の順に優先順序をつけるとすると、Brand と Iyenger の方法ではパス A-A2-D-E-X が活性化可能ということになる。

Brand と Iyenger の方法によるタイミング解析は悲観的なタイミング解析を保障する。その悲観性の度合いは優先順序のつけかたに依存する。優先順序は信号遷移の到着順と一致すれば正確な遅延を得るが、異なる場合はその分だけ悲観的な遅延が求まる。この他に、悲観的な解析を行える活性化条件としては、**static co-sensitization** がある [6]。この方法では、ゲート入力で controlling value が複数ある場合、全ての入力を活性化可能とするため、Brand と Iyenger の条件の方が精度が高い。

2.6　フォールスパスの表現

人手であるいは自動で検出したフォールスパスはタイミング制約情報としてタイミング解析の入力となる。パス数は最悪回路規模の指数倍になりうるため、一本ずつフォールスパスを指定する明示的表現も指数爆発の可能性がある。そのため、フォールスパスの集合をまとめて表現する非明示的表現がいくつか提案されている。一つは**通過点指定**と呼ばれるもので、フォールスパスが通過すべき回路中の点の集合を与え、その集合内の全ての点を通過するパスが全てフォールスパスであるという方法である。もう一つは**部分グラフ指定**と呼ばれるもので、始点と終点を指定した回路の部分グラフを与え、そのグラフで始点から終点へのパスのうちの少なくとも一つを部分パスとするパスが全てフォールスパスであるという方法である [11]。本稿では、現在フォールスパスの指定方法として広く用いられている通過点指定による方法のみ考慮する。

フォールスパスの数はタイミング解析の計算量に影響を与える。フォールスパスが存在する場合、遅延計算の途中で最大遅延のパスがフォールスパスかどうか確定していないパスであった場合それよりも遅延の小さいパスの遅延も保持しておく必要がある。したがって、最悪フォールスパス数だけ遅延を保持して計算を進める必要がある。すなわち、タイミング解析も回路規模に対して指数爆発する可能性がある。フォールスパスが通過点指定により非明示的に表現された場合、最悪各通過点指定の始点から終点の間でフォールスパス集合の数だけ遅延を保持して計算をするめることになる。したがって、できるだけコンパクトにフォールスパスを表現することによりタイミング解析の計算量への影響を削減することができる。

3.　信号線の関係に着目したフォールスパス解析

3.1　本手法で仮定する活性化条件

本稿で提案するフォールスパス解析では、実用的観点から、活性化条件として Brand と Iyenger の条件を用いる。ただし、Brand と Iyenger の条件は static sensitization に優先順序を導入しただけであるので、以下の議論では基本的には static sensitization で考えるが、優先順序を与えれば Brand と Iyenger の条件での解析となる。

3.2　例

例えば、図 1 を考える。この回路のパスは計 7 本存在する。したがって、パス毎に活性化条件が成立かどうかを調べてフォールスパスかどうかを確かめる場合、7 本のパスそれぞれについてパスの始点から終点まで各ゲートで活性化条件が成立するかどうかを解析する必要がある。その結果パス $\{C, C1, E, X, X2, Y\}$ がフォールスパスであると分かる。

しかし、フォールスパス $\{C, C1, E, X, X2, Y\}$ が活性化できない原因を考えると、$C1$ と $C2$ が同時にゲート E とゲート Y を controlling value にできないことにある。さらにそれは $C1$ と $C2$ はともに C からのファンアウトであり、$C1 \neq C2$ となる値割り当てがあり得ないことが原因である。このようにフォールスパスは回路中のあり得ない値割り当てに対応する。逆に言

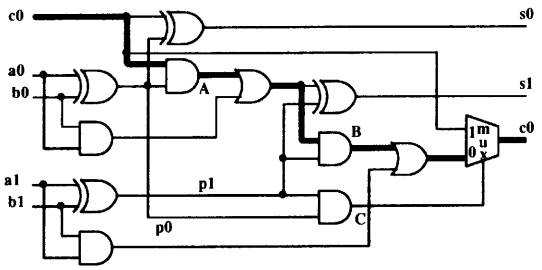


図 6　3 つの信号線間の関係から生成されるフォールスパス

うと、回路中のあり得ない値割り当ては非明示的にフォールスパスの集合を表している。

本稿で提案する手法はこのような考え方に基づき、まずファンアウト情報からあり得ない値割り当てを抽出し、それに対応するフォールスパスの集合を生成するという方法でフォールスパスを検出する。したがって、パス毎の解析は不要となる上、パスの始点から終点までの解析も不要で、フォールスパスの原因となる部分のみの解析でフォールスパスが検出できる。

回路中の 2 つの信号線 A, B が等価な場合、すなわち $A = B$ が常に成り立つ場合も、ファンアウトの場合と同様に、等価点同士で値の異なるような値割り当てが存在しないとしてフォールスパスの検出が行える。

さらに、回路中の 2 つの信号線 A, B が含意関係にある場合、すなわち $A = v_1 \rightarrow B = v_2$ が常に成り立つ場合には、$(A, B) = (v_1, \overline{v_2})$ となる値割り当てが存在しないとしてフォールスパスの検出が行える。2 つの信号線間の含意関係は ATPG の静的学習により回路から抽出することができる。

3 つ以上の信号線間の関係を抽出することは一般的には難しい。しかし、局所的な関係は比較的容易に求められる場合もある。例えば、図 6 の回路を考える。この回路は 2 ビットの桁上げバイパス回路であり、太線のパスがフォールスパスである。太線のパスがフォールスパスであることは次のような考察から分かる。太線のパスを活性化させるにはマルチプレクサの制御入力は 0 でなければならない。AND ゲート C の出力が 0 でなければならないということは $p1$ か $p2$ のいずれかが 0 でなければならない。すなわち、太線のパス上の AND ゲート A か B のいずれかが活性化できなくなる。したがって太線のパスはフォールスパスである。この考察から、太線のフォールスパスは

$$C = 0 \rightarrow \text{“}p0 = 0 \text{ または } p1 = 0'' \qquad (1)$$

という 3 信号線間の関係から得られることが分かる。この関係は、AND ゲート C の機能から容易に抽出可能である。したがって、各ゲートの機能からこのような関係を抽出すれば、2 つの信号線間の関係の場合と同様に，それらに対応するフォールスパスを求めることができる。

3.3　アルゴリズムの説明

本節では提案するフォールスパス解析のアルゴリズムについて述べる。前節の例のようなフォールスパス解析を行うアルゴリズムの処理の流れは以下の (1)~(10) ようになる。

1)　乱数シミュレーションにより回路中の定数ノードと等価ノードの候補を絞る。

2)　二分決定グラフ (BDD) による関数計算で定数ノード候補の検証を行う。

3)　検証された定数ノードに対応する、1 つの通過点により指定されるフォールスパス集合を求める。

4)　各ノードからのファンアウトの任意の組 (a, b) に対応するフォールスパスが存在するかどうかを以下の手順で調べる。

4.1)　あり得ない値割り当てでのみ活性化可能かどうか調べる。具体的には、a, b を入力とするノード N_a, N_b について、N_a と N_b の non-controlling value が異なるかどうか調べる。異なればあり得ない値割り当てでのみ活性化可能ということになりフォールスパスの可能性がある。その場合のみ以下を行う。

4.2)　N_a(あるいは N_b) の出力から N_b(あるいは N_a) の $b(a)$ 以外の入力に至る部分パスが存在するかを調べる。

4.3)　以上の 2 つのチェックをパスした場合 N_a の a 以外の入力と N_b の b 以外の入力を 2 つの通過点とするパス集合をフォールスパスとする。

5)　等価検証の技術を利用して等価ノード候補の検証を行う。

6)　検証された等価ノードの組 (A, B) に対応するフォールスパスの集合をファンアウトの場合と同様にして求める。この場合も求まるフォールスパスの集合は 2 つの通過点により指定されるものとなる。

7)　静的学習を行い 2 信号線間の含意関係を抽出する。

8)　2 信号線間の含意関係に対応するフォールスパスの集合を等価ノードの場合と同様にして求める。ただし、含意関係 "$a = v_a \rightarrow b = v_b''$ に対するあり得ない値割り当ては $a = v_a, b = \overline{v_b}$ なので、non-controlling value のチェックは v_a が N_a の non-controlling value かつ $\overline{v_b}$ が N_b の non-controlling かどうかを調べることで行う。この場合条件が成立すればあり得ない値割り当てでのみ同時に活性化可能ということでフォールスパスの可能性がある。2 信号線間の含意関係の場合も求まるフォールスパスの集合は 2 つの通過点により指定されるものとなる。

9)　前節の図 6 のように、2 信号線間の含意関係から求まる 3 信号線以上の間の関係を抽出する。

10)　抽出した関係に対応するフォールスパスを今までと同様に non-controlling value のチェックと部分パスのチェックにより行う。　□

4.　実験結果

前章で述べたフォールスパス検出アルゴリズムを C++を用いて実装し、論理回路からフォールスパスを検出する実験を行った。

実験では ISCAS85 組合わせ回路ベンチマークと ISCAS89 順序回路ベンチマークを用いた (ただし、小規模なものは省略した)。順序回路ベンチマークは組合わせ回路部分のフォールスパスを検出した。本実験では乱数シミュレーションについては、定数候補数と等価候補分割数に変化がなくなってから 32 × 10

表 1　フォールスパス検出の実験結果

回路	ゲート数	フォールスパス集合数 定数	fanout	含意	CPU (秒)
c432	160	0	27	0	0.0
c499	202	0	4	0	0.0
c880	383	0	0	0	0.0
c1355	546	0	4	96	0.1
c1908	880	0	6	18	0.1
c2670	1193	13	6	23	0.1
c3540	1669	1	0	98	2.9
c5315	2307	1	1	12	0.2
c6288	2406	0	30	17	14.2
c7552	3512	4	0	74	0.6
s1196	388	0	3	10	0.1
s1238	428	0	0	5	0.1
s1269	437	0	0	4	0.0
s1423	490	0	8	2	0.0
s1488	550	0	0	0	0.1
s1494	558	0	0	0	0.1
s1512	413	0	0	0	0.1
prolog	978	61	1	12	0.2
s3271	1035	4	4	47	0.1
s3330	815	0	0	0	0.1
s3384	1070	0	0	3	0.1
s4863	1600	0	0	0	0.2
s5378	1004	24	0	102	0.3
s6669	2071	0	0	24	0.2
s9234.1	2027	24	6	688	1.0
s9234	2027	68	6	673	0.9
s13207.1	2573	98	2	433	3.2
s13207	2573	0	2	528	1.9
s15850.1	3448	82	7	432	2.1
s15850	3448	60	13	399	1.9
s35932	12204	0	1	0	14.4
s38417	8709	65	33	226	6.3
s38584.1	11448	356	13	1282	24.5
s38584	. 11448	331	9	1200	12.9

パタン後に終了するようにした。活性化条件は、実験結果が優先度順により影響を受けないよう Brand と Iyenger の条件ではなく、static sensitization を用いた。また、実験ではファンアウトノード以外の等価ノードの抽出や 3 信号線以上の関係の抽出は行っていない。本実験を行った計算機は SPARC64 IV(702MHz、主メモリ 8GB)、OS は Solaris 8、使用コンパイラは gcc version 2.95.3、コンパイラオプションは"-O2" である。

実験結果を表 1 に示す。表 1 において、「定数」は定数縮退ノードから得られる指定通過点数 1 のフォールスパス集合の数、「fanout」はファンアウトから得られる指定通過点数 2 のフォールスパス集合の数、「含意」は静的学習により得られた 2 信号線間の含意関係から得られる指定通過点数 2 のフォールスパス集合の数、フォールスパス数は得られたフォールスパス集合により非明示的に表現されているフォールスパスの総数、「CPU(秒)」は CPU 時間を秒で表したものである。

表 1 より 1 万ゲート程度の回路でも数十秒程度で解析が終了しており高速にフォールスパスが生成できていることが分かる。また、1 万ゲート程度の回路では 1000 個程度のフォールスパス集合が求まっている。これは少ない信号線の間の関係が原因でフォールスになるパスが多いことを示している。パス毎にフォールスパス解析を行う場合には、このようなパス全体を解析する必要がない場合でもパス全体を解析し、計算時間が多くなる可能性が高いと考えられる。また、生成されたフォールスパス集合の指定通過点数は高々2 であり、コンパクトなフォールスパス表現が可能となっている。

5. おわりに

本稿では含意関係などの信号線間の関係に着目して大規模な組み合わせ回路からフォールスパスを効率的に抽出する方法を提案した。実験結果より 1 万ゲート程度の回路に対して数十秒で解析が行え実用に耐えうる性能であるといえる。また、2 信号線間の含意関係など少ない信号線間の関係でフォールスになっているパスが多いことも分かった。

今後、生成されたフォールスパスをタイミング解析に入力し、実際にタイミング解析の処理時間への影響を確かめていきたい。また、実設計にも適用し、指定通過点数の少ないフォールスパスがどの程度あるか、クリティカルなパスのうちどの程度がフォールスパスかなどについても調べていく予定である。

文　献

[1] 田代尚美 and 今野正. Performance driven layout システムの概要. In *DA シンポジウム 2002*, pages 7–12, 2002.

[2] H. Higuchi. An implication-based method to detect multicycle paths in large sequential circuits. In *Proceedings of the 39th ACM/IEEE Design Automation Conference*, pages 164–169, June 2002.

[3] D. Brand and V. S. Iyengar. Timing analysis using functional analysis. *IEEE Transactions on Computers*, 37(10):1309–1314, October 1988.

[4] P. C. McGeer, A. Saldanha, R. K. Brayton, and A. Sangiovanni-Vincentelli. Delay models and exact timing analysis. In T. Sasao, editor, *Logic Synthesis and Optimization*, pages 167–189. Kluwer Academic Publishers, 1993.

[5] H.-C. Chen and D. H.-C Du. Path sensitization in critical path problem. *IEEE Transactions on Computer-Aided Design of Integrated Circuits*, 12(2):196–207, February 1993.

[6] S. Devadas, K. Keutzer, and S. Malik. Computation of floating mode delay in combinational circuits: Theory and algorithms. *IEEE Transactions on Computer-Aided Design of Integrated Circuits*, 12(12):1913–1923, December 1993.

[7] Y. Kukimoto and R. K. Brayton. Exact required time analysis via false path detection. In *34th ACM/IEEE Design Automation Conference*, pages 220–225, June 1997.

[8] V. Hrapcenko. Depth and delay in a network. *Soviet Math. Dokl.*, 19(4), 1978.

[9] S. Devadas, A. Ghosh, and K. Keutzer. *Logic Synthesis*. McGraw-Hill, Inc., 1994.

[10] J. Benkoski, E. V. Meersch, L. J. M. Claesen, and H. De Man. Timing verification using statically sensitizable paths. *IEEE Transactions on Computer-Aided Design of Integrated Circuits*, 9(10):1073–1084, October 1990.

[11] K.. P. Belkhale and A. J. Suess. Timing analysis with known false sub graphs. In *Proceedings of International Conference on CAD-95*, pages 736–740, November 1995.