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

2004－SLDM－117（35）
2004／12／2

順序回路のタイミング例外パス検出のための実用的方法

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研究院　〒 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　本稿では，大規模な順序回路のタイミング例外パスを検出するための実用的方法を提案する．まず，回路中のパスの数え上げを行わず，かつ，回路を大域的に見てフォールスパスの検出を行う方法として，マルチプレクサ (MUX) グラフという概念を導入し，MUX グラフの縮約と MUX グラフ上のパスの数え上げによりフォールスパス集合の集合を生成し圧縮する方法を提案する．また，フリップフロップ (FF) ペアベースのマルチサイクルパス解析において FF ペア間の一部のパスのみマルチサイクルであるようなパスも検出し，検出能力を向上させる方法を提案する．
キーワード　順序回路，タイミング例外パス，フォールスパス，マルチサイクルパス，タイミング解析

Practical Techniques for Automatic Detection of Timing Exception Paths in Sequential Circuits

Hiroyuki HIGUCHI† and Yusuke MATSUNAGA††

† CAD Laboratory, FUJITSU LABORATORIES LTD.
Kamikodanaka 4-1-1, Nakahara-ku, Kawasaki, 211-8588 Japan
†† Graduate School of Information Science and Electrical Engineering, Kyushu University
6-1 Kasuga Koen, Kasuga, Fukuoka, 565-0456 Japan
E-mail: †higuchi@labs.fujitsu.com, ††matsunaga@c.csce.kyushu-u.ac.jp

Abstract　In this paper, we propose two new practical methods for detecting timing exception paths in large sequential circuits. First, we propose a new false path detection and compaction method using multiplexer(MUX) graphs. MUX graphs is introduced to avoid path enumeration in large circuits and to globally look at large circuits. Second, we improve the detection ability of FF-pair-based multi-cycle path analysis by automatically and properly dividing groups of paths between FF pairs.
Key words　sequential circuit, timing exception path, false path, multi-cycle path, timing analysis

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でき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と複雑さが増大し，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，今CADツールによるサポートが最も望まれている部分の一つである．

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，本報告書では，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．まず，フォールスパス解析方法として，回路中でパス選択の役割を果たすマルチプレクサゲートに着目し，回路をマルチプレクサ(MUX)グラフというグラフに変換，縮退することにより，効率的にフォールスパスを検出する方法を提案する．この手法により，従来パスベースの解析で問題となっていたパス数爆発の問題や信号線間の含意関係に着目した解析で問題となっていた3つの信号線以上の関係に基づくフォールスパス検出の困難さの問題を避けることができ，高速，かつ回路を大域的に見た検出能力の高い実用的解析が行えるようになる．

次に，マルチサイクルパス解析において，FFペアベースの手法の検出能力を向上させる方法として，終点FFの値取込み条件によりFFペア間のパス集合を分割した後に分割したパス集合に対して解析を行う方法を提案する．本手法により，FFペア間のパスの一部のみがマルチサイクルパスである場合も検出できるようになる．また，パスベースの手法のようにパス一本一本処理するのでなく，パス集合を処理するため，パス数爆発の問題が起きることもない．

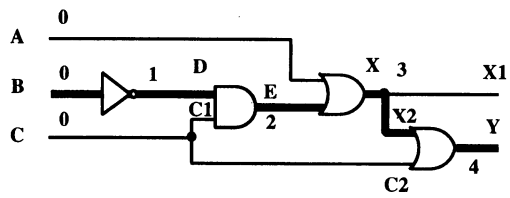

図1 フォールスパス

2. タイミング解析とタイミング例外パス

2.1 用語の定義

順序回路のパスとは，外部入力ピンまたはFFの出力ピンから始まり，ゲートの入力ピンと出力ピンが交互に現われ，最後に外部出力ピンまたはFFの入力ピン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．

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．

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とFFの間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．

2.2 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．図中の数字は，配線遅延を0，ゲート遅延を1としたときの外部入力からの遅延値である．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

(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．

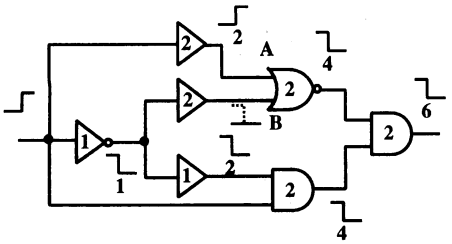

図 2　フローティングモードと固定遅延での計算 [10]

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. 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.

2.3　マルチサイクルパス

あるパスを信号がソースからシンクまで 1 クロックで到達する必要がないときそのパスをマルチサイクルパスという. 特に最大 k クロックで信号がシンクまで到達すればよいパスを k サイクルパスという.

ある FF 対 (A, B) について，A をソース，B をシンクとする全てのパスがマルチサイクルパスであるとき (A, B) をマルチサイクル FF 対と呼ぶ. 特に，FF 対間のパスのうち少なくとも一つは k サイクルパスで，それ以外のパスは全て k サイクル以上のパスであるとき，その FF 対を k サイクル FF 対と呼ぶ.

2.4　タイミング解析のモードとパス活性化条件

本稿では，フォールスパス解析のモードとしてフローティングモードを仮定し，パス活性化条件として static sensitization を仮定する. 本節ではフローティングモードと static sensitization についてのみ説明する. その他のタイミング解析のモードとパス活性化条件については [9] などを参照されたい.

2.4.1　フローティングモード

フローティングモードは，信号遷移を考える連続する 2 クロックサイクルのうち一つ目のサイクルでの回路中の信号値が不定値，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 1 つめのサイクルでの回路中の信号値も考慮した信号遷移で解析する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 [10]. 図 2 はフローティングモードでの回路遅延の計算例である. 図 2 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

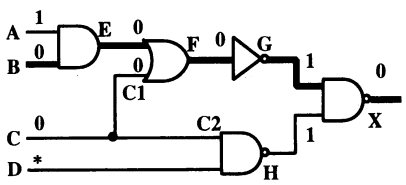

図 3　Static sensitization

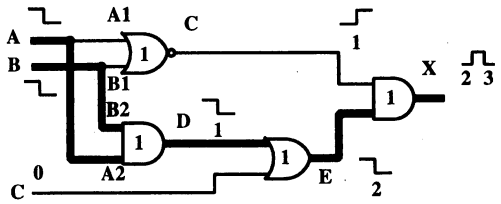

図 4　Static sensitization による false path が活性化できる例

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.

2.4.2　Static Sensitization

2.4.1 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 [11].

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 3 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 3 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 4 で例を示す. 太線のパス 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 4 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.

static sensitization の楽観性を回避するためには，遅延情報を用いてブロックする side-input の信号がブロックされる

フォールスパスよりも必ず速く到達することを確認するか，遅延情報を用いない場合には各ゲートの入力にあらかじめ優先度をつけてから static sensitization を行う Brand と Iyenger の方法を用いる必要がある．

2.5 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 [12]．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．

また，フォールスパスが本当にフォールスパスかどうかを遅延非依存に，あるいは遅延依存に再検証する際には，フォールスパス集合を指定する通過点での side-input の論理，あるいは遅延をみればよいため，コンパクトなフォールスパス記述はフォールスパスの検証も容易となる．フォールスパスをタイミング制約に含めることはそのパスのタイミングチェックをなくすことであるため，フォールスパス指定の誤りは許されない．そのため，コンパクトなフォールスパス指定はそのような点からも重要である．

3. MUX グラフに基づくフォールスパス制約の生成と圧縮

回路中のパス数は回路規模に対し最悪指数倍になる．フォールスパス解析の方法としてパスベースの方法が従来から数多く提案されているが，これらの方法を実設計回路に適用するのは計算コストの点で非現実的である．解析するパスをタイミング解析の結果得られるタイミングエラーパスに限定して適用することも考えられるが，最近の傾向としてタイミング収束の悪化により設計途中でのタイミングエラーパスの数は大幅に増大しており，タイミングエラーパスに限定したパスベース解析のアプローチでさえ現実的ではない．

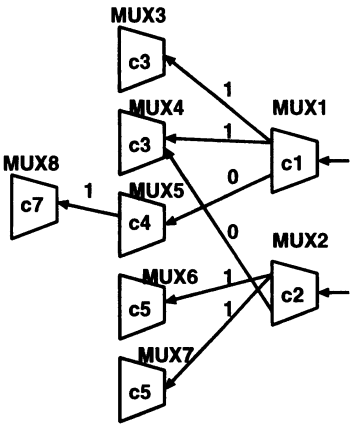

図 5 MUX グラフ

そのような状況を踏まえ，本稿では，回路中のマルチプレクサ (MUX) ゲートに着目したフォールスパス検出方法を提案する．回路中の MUX ゲートは回路中のパスを選択する働きを持つため，MUX ゲートのみに着目してフォールスパス解析をすることは実用的な時間で有効なフォールスパスを多く見つけることができると考えられる．

提案手法では，回路から MUX ゲートのみ考慮するため，MUX ゲートのみ抽出して作った MUX グラフという概念を導入する．解析のさらなる高速化のため MUX グラフの縮約化方法を提案する．さらに，縮約した MUX グラフ上でのフォールスパス生成および圧縮の手法を提案する．

3.1 MUX グラフの作成

本稿では MUX ゲートとして 2-to-1MUX ゲートのみ考える．データ入力が 2 より大きな MUX ゲートはあらかじめ 2-to-1MUX ゲートに分解されているものとする．

[定義 1] 組合わせ回路 C に対する MUX グラフ $G = (V, E, vlabel, elavel)$ は以下を満たすラベルつき非巡回有効グラフである．

（1） 節点 V は C 中の全ての MUX ゲートの集合である．

（2） 節点 v のラベル $vlabel(v)$ はもとの回路中の MUX ゲート v の制御入力のファンインゲートの出力に割り付けた論理変数である．

（3） ラベル $elavel(e) \in B = \{0,1\}$ を持つ枝 $e = (v_1, v_2) \in E$ は MUX ゲート v_1 における $elavel(e)$ のデータ入力ピンから推移的ファンインをたどり途中に他の MUX ゲートを通過せずに MUX ゲート v_2 に到達できることを示す． □

図 5 は MUX グラフの例である．MUX 節点中のラベルは $vlavel$ の変数名である．図の左側 (根側) が回路の入力側，右側 (葉側) が回路の出力側である．MUX グラフは回路からマルチプレクサのみを抽出したものと考えることができる．MUX グラフ上の根から葉への MUX パスは，もとの回路上では MUX

のデータピンを通過点指定したパス集合を表す．MUX パスは例えば図 5 では，(MUX1,1,MUX3,1) などである．葉でもその MUX のデータ入力 1 かデータ入力 0 かでパスを区別する必要があることに注意されたい．MUX グラフ上でのフォールス MUX パスを検出することで，もとの回路に対する通過点指定のフォールスパス集合が検出できる．通常，MUX グラフはもとの回路に対し大幅に規模が小さいため MUX グラフ上でパスベースの解析を行うことは現実的である．

3.2 MUX グラフの縮約

本手法では，MUX グラフ上でのフォールスパス解析を行うまえに MUX グラフを縮約し，できるだけ MUX グラフの規模を小さくしておく．MUX グラフの縮約の説明の準備としてまず二つの関数 $child1, child0$ を定義する．

［定義 2］ MUX グラフ $G=(V,E)$ において，$child1 : V \rightarrow 2^V$ は節点 $v \in V$ から v の 1 の子供の節点の集合を与える関数である．また，$child0 : V \rightarrow 2^V$ は節点 $v \in V$ から v の 0 の子供の節点の集合を与える関数である． □

MUX グラフにおける節点の等価性を定義する．

［定義 3］ MUX グラフ $G=(V,E,vlavel,elavel)$ の二つの節点 $v_1, v_2 \in V$ が以下の 3 条件を満たすとき v_1 と v_2 は等価であるという．

（1） $vlavel(v_1) = vlavel(v_2)$,
（2） $child1(v_1) = child1(v_2)$,
（3） $child0(v_1) = child0(v_2)$. □

MUX の縮約は以下の定理に基づく．

［定理 1］ MUX グラフ $G=V,E,vlabel,elabel$ において，v_1 と v_2 が等価であるとする．v_1 を通るパスがフォールスパスならば v_1 を v_2 で置き換えたパスもフォールスパスである． □

この定理から MUX グラフで等価節点があればその一方を削除することでグラフを縮約できることが分かる．

［系 1］ MUX グラフ $G=V,E,vlabel,elavel$ において，子供の節点がない二つの葉節点 $v_1, v_2 \in V$ が同じラベルを持てばその一方を削除してグラフを縮約できる． □

図 5 を縮約したグラフを図 6 に示す．

3.3 MUX グラフからのフォールスパスの生成

MUX グラフの各パスについて，そのパスを選択する MUX ゲートの制御信号値を求め，それらの値が同時に成立するような組合わせ回路 C の入力が存在するか C 上の解析で調べ，存在しなければフォールスパスと判定する．生成した MUX グラフでの MUX フォールスパスの集合は縮約したグラフに対するものであるため，生成したパスを等価な節点で置き換えたパスも MUX フォールスパスの集合に追加する．このようにして生成した MUX フォールスパスの集合中の各パスがもとの組合わせ回路中で MUX のゲート入力ピンを通過点として指定された

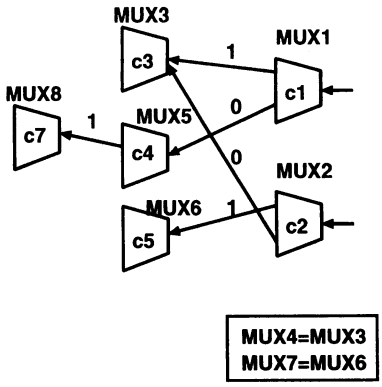

図 6 縮約した MUX グラフ

フォールスパスの集合となる．

MUX フォールスパスを生成する際に，その MUX フォールスパスが通過点集合として既に生成した MUX フォールスパスの通過点集合を包含していれば新たにフォールスパス集合に含めないようにすれば，フォールスパス集合としてのコンパクションが行える．また，ある MUX フォールスパスを指定する通過点の集合を最小化することでその MUX フォールスパスの記述の最小化が行える．

4. パス集合ベースのマルチサイクルパス解析

本章では，FF ペアベースのマルチサイクルパス解析手法 [2], [13] を改良し，FF ペア間の一部のパスのみマルチサイクルであるようなパスも検出する方法を提案する．本手法では，FF ペア間のパス集合をマルチサイクルで値を取り込むパス集合とそれ以外のパス集合に分け，マルチサイクルパスの検出に対しては前者のパス集合内の全てのパスがマルチサイクルかどうかを一度に解析する．そのため，FF ペア間のパスが全てマルチサイクルパスでない場合でもマルチサイクルパスが検出でき，かつパスごとの解析ではなくパス集合のまま解析するため計算時間の増大も抑えることができる．また，出力するマルチサイクルパスの記述もパスごとの記述でなく，パス集合として記述することで，記述量の削減も行える．

図 7 の回路を例に説明する．この回路は MULT=0 のとき ADD(加算) を 1 サイクルで行い，MULT=1 のとき MULT(乗算) を 2 サイクルで行うものである．今 FF ペア (FF_1, FF_4) を考える．回路中に，終点 FF の FF_4 がマルチサイクルで値を取込む条件である $FF_4.EN(1)=0, FF_4.EN(2)=1$ を割り当てる．カッコ中の数字は時刻を表す．この割り当てをもとに含意操作を行う．図 7 で信号線上の数値はその信号線での値を示す．カッコ内は時刻を表す．例えば，FF7.Q にある 1(2) は時刻 2 で FF7.Q=1 であることを示す．MUX2.C が FF1 から FF4 へのパスを選択する信号であるので，MUX2.C(2)=1 をマルチサイクル値取込条件の値割り当て C とする．この値は MULT(0)=1 に対応し，MULT を行うときの条件が正しく抽出されていることが分かる．MUX2.C(2)=1 を割り当てた状態

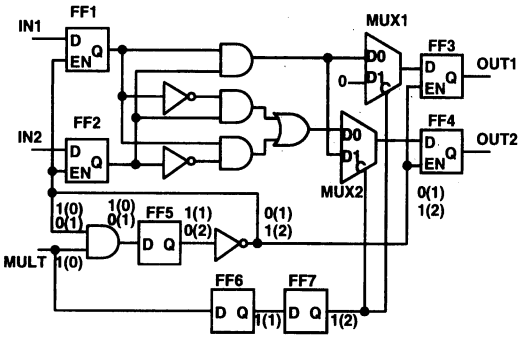

図 7　マルチサイクルパス解析のためのパス集合の分割

表 1　フォールスパス検出の実験結果

		MUX ベース		原因ベース [9]	
回路	セル数	FP 集合数	時間 (秒)	FP 集合数	時間 (秒)
回路 A	246K	737,157	8,879	14,844	75
回路 B	290K	457,109	1,451	7,937	120

で FF ペアベースのマルチサイクルパス解析を行い，この状態で FF ペア (FF_1, FF_4) がマルチサイクルパスであると分かる．すなわち，(FF_1, FF_4) 間で MUX2.D1 を通るパスは全てマルチサイクルパスであると分かる．マルチサイクルパス解析は具体的には例えば [2] などの FF ペアベースの解析で行える．

5. 実験結果

3. 章で述べた MUX グラフを用いたフォールスパス検出アルゴリズムを C++言語で実装し，実設計からフォールスパスを検出する実験を行った．本実験を行った計算機は SPARC64 V(1100MHz，主メモリ 16GB)，OS は Solaris 9，使用コンパイラは g++ version 2.95.3，コンパイラオプションは"-O2" である．実験では，static sensitization を満たすかどうかの判定を含意操作のみで行っている．すなわち，実際にはフォールスパスである場合でも検出できない場合がある．また，通過点指定の包含関係チェックによる記述の圧縮は行っていない．

実験結果を表 1 に示す．表中,「セル数」は回路中のセルの数,「MUX ベース」の欄は本手法によるフォールスパス解析の結果,「原因ベース」の欄は従来法の [9] によるフォールスパス解析の結果である.「FP 集合数」は検出した通過点記述のフォールスパス集合数,「時間 (秒)」は処理時間を秒で示したものである．原因ベースの手法は同一ファンアウトや 2 信号線間の含意関係など回路中のあり得ない値割り当てから対応するフォールスパスを求める方法である．

表 1 より，MUX グラフを用いたフォールスパス解析は多くのフォールスパスを検出していることが分かる．処理時間も 20 万セル以上の大規模回路に対して 2〜3 時間であり，実用的な時間で解析が行えている．原因ベースのフォールスパス解析 [9] と比較すると，検出できるフォールスパス数が大幅に増加していることが分かる．これは原因ベースの解析が同一ファンアウトや 2 信号線間の含意関係など (含意関係では learning は用いているが) 比較的ローカルな情報を用いているのに対し，本手法では MUX グラフという大域的な情報で，かつ，パス選択に直接関係する情報を抽出しているためであると考えられる．

6. おわりに

本稿では，大規模な順序回路のタイミング例外パスを検出するための実用的方法を提案として，まず，マルチプレクサ (MUX) グラフを導入して大規模回路から実用的な時間内にフォールスパス集合の集合を生成し圧縮する方法を提案した．また，フリップフロップ (FF) ペアベースのマルチサイクルパス解析において FF ペア間の一部のパスのみマルチサイクルであるようなパスも検出し，検出能力を向上させる方法を提案した．MUX グラフを用いたフォールスパス検出の実験結果より，実設計に対しても効率的に多くのフォールスパスが検出できることを示した．

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] 樋口 博之. 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. In 情処研報 *SLDM-112*, pages 121–126, November 2003.

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

[11] 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.

[12] 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.

[13] K. Nakamura, K. Takagi, S. Kimura, and K. Watanabe. Waiting false path analysis of sequential logic circuits for performance optimization. In *Proceedings of IEEE/ACM International Conference on CAD-98*, pages 392–395, November 1998.