システムＬＳＩ設計技術　103－17
（2001. 11. 29）

# 大規模順序回路に対するマルチサイクルパス解析手法

樋口 博之

(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
Email: higuchi@flab.fujitsu.co.jp

あらまし

本稿では大規模順序回路に対して高速にマルチサイクルパスを検出する方法を提案する。本手法はフリップフロップ (FF) 間の全てのパスがマルチサイクルであるかどうかを検出するものであり、テストパタン生成 (ATPG) の技法を用いて効率よくマルチサイクルの判定を行う。まず含意操作によりできるだけ多くの「容易な」マルチサイクルパスを高速に同定し、その後、残りの FF 対についてシングルサイクルで動作する必要のある入力パタンが存在しないかどうかを ATPG を利用して判定する。実験結果より、FF 数が 1000 を越える回路に対しても数十秒程度で解析を行うことができ、本手法の実用性が示された。

キーワード

マルチサイクルパス、タイミング解析、順序回路、テストパタン生成、含意操作

# Multi-Cycle Path Analysis for Large Sequential Circuits

Hiroyuki Higuchi

Fujitsu Laboratories Ltd.
4-1-1, Kamikodanaka, Nakahara-ku, Kawasaki, 211-8588, Japan
Email: higuchi@flab.fujitsu.co.jp

**Abstract**

This paper proposes a fast multi-cycl path analysis method for large sequential circuits. It determines whether all the paths between every flip-flop pair are multi-cycle paths. ATPG techniques allow us to detect multi-cycle paths quickly by using information of circuit structure directly. Our method detect as many "easy" multi-cycle paths as possible by implication before applying ATPG. Experimental results show that our method can handle large circuits with more than one thouthand flip-flops in reasonable time.

**key words**

multi-cycle path, timing analysis, sequential circuit, ATPG, implication

# 1 はじめに

近年の集積回路システムの高性能化の要求に伴い、論理回路のタイミング検証やタイミング最適化の重要性がますます高まっている。正確なタイミング検証や最適化能力の高いタイミング最適化を行うには、回路中のどの部分、あるいはどのパスがタイミングクリティカルであるかをいかに正確に求めることができるかが鍵となる。

論理回路に対するタイミング制約はフリップフロップ (FF) のセットアップ制約とホールド制約および外部入出力と FF 間の最大・最小遅延制約により与えられるため、最も単純に得られるタイミングクリティカルパスはトポロジカルな最大遅延、最小遅延を与えるパスである。しかし、トポロジカルなパス遅延により求められたクリティカルパスは必ずしもクリティカルパスでない場合がある [1]。これは、回路の機能を考慮すると、決して活性化されないパスやパスの始点が活性化されてから終点が活性化されるまで複数クロックを要するパスが存在するからである。前者のようなパスをフォールスパス、後者のようなパスをマルチサイクルパスと呼ぶ。フォールスパスと分かった場合にはそのパスに対するタイミング制約がなくなり、マルチサイクルパスと分かった場合にはそのパスに対するタイミング制約がサイクル数分だけ緩くなるので、パス遅延のみによるタイミング検証は悲観的となる。すなわち、フォールスパスやマルチサイクルパスをできるだけ多く見つけることにより悲観的な度合いが少ない、より正確な検証が行える。

組合わせ回路からフォールスパスを検出する方法は従来から数多く提案されている [2, 3, 4, 5]。しかし、パス数はノード数に対し最悪指数爆発する可能性があるため、実際にはいくつかのパスのみを対象として解析する必要がある。また、実際の回路素子と配線の遅延の慣性やばらつきを考慮して解析しようとする場合には個々のパスの解析にも多くの時間を要する。一方、マルチサイクルパスを検出する方法は、1994 年にマイクロプロセッサのタイミング検証のためのものが提案されている [6]。ゲートレベルの論理回路に対しては、1998 年に二分決定グラフ (BDD) を用いた FSM の状態探索による手法 [7]、1999 年に充足可能性判定問題 (SAT) に帰着して解く手法 [8]、がそれぞれ提案されている。[6] は基本的にはフォールスパスの場合と同様に各パスをしらみつぶし的に解析する必要がある。一方、[7, 8] は 2 つの FF 間のパスをひとまとめにし、FF 間のパス全てがマルチサイクルパスであるかどうかを調べるものである。従って、パス数の指数爆発の問題を避けることができる。ある FF と FF、あるいは入力と出力、の間の全てのパスがフォールスであるという場合は現実的には少ないと思われるが、マルチサイクルの場合には実際に十分起りうる。しかし、これらの方法では、BDD のサイズの爆発や SAT に帰着することによるオーバヘッドの問題のため、FF 数が数百以上の大規模回路に対しては多くの時間を要した。

またマルチサイクルパスを ATPG を用いて検出する方法としては遅延故障の ATPG の枠組みで行う方法が提案されている [9]。シングルサイクルで活性化できるパタンが存在しない場合、そのパスはマルチサイクルパスと判定できる。しかし、この方法ではパス毎に解析を行う必要がある。

本稿では、従来法で多くの時間を要するような大規模回路に対して高速にマルチサイクルパスを検出する方法を提案する。本手法は [7, 8] と同様に FF 間の全てのパスがマルチサイクルであるかどうかを検出するものであり、テストパタン生成 (ATPG) の技法を用いて高速にマルチサイクルの判定を行う。ATPG の技法を利用することで回路構造を直接利用することが可能になる。

以下、2 章では本稿で扱うマルチサイクルパスとその検出問題の定義を行う。3 章では提案するマルチサイクルパス解析手法の説明を行う。4 章では静的ハザードの影響を考察する。5 章では提案手法の性能評価を行う。6 章ではまとめと今後の課題について述べる。

# 2 マルチサイクルパス検出問題

## 2.1 マルチサイクルパス

論理回路のパスとは外部入力あるいは FF の出力を始点とし、途中に FF を経由せずに、外部出力あるいは FF の入力に至る経路をいう。パスの始点をソース、パスの終端をシンクと呼ぶ。

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。

## 2.2 マルチサイクルパスの例

マルチサイクルパスの例として図 1 の論理回路を考える。$FF_1, FF_2, FF_3, FF_4$は D-FF である。$IN, OUT$ はそれぞれ外部入力、外部出力である。一般的には、$FF_1$と $FF_2$はそれぞれ複数ビットからなるレジスタ、すなわち記憶素子の集合であってもよい。同様に、$IN, OUT$ もそれぞれ複数ビット、すなわち、複数本の外部入力線、外部出力線であってもよい。$FF_3$と $FF_4$は、初期状態を $(FF_3, FF_4) = (0, 0)$

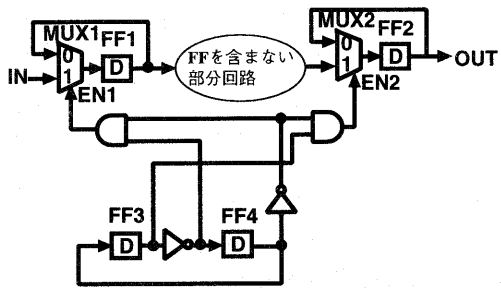


図 1: 例題の回路

とすると、$(0,0) \rightarrow (0,1) \rightarrow (1,1) \rightarrow (1,0) \rightarrow (0,0)$ と繰り返し変化する周期 4 のカウンタをなす。$FF_1$ は $(FF_3, FF_4) = (0,0)$ の次のクロックでのみデータを取り込む。一方、$FF_2$は、$FF_1$で取り込まれたデータを楕円部分で処理し、$(FF_3, FF_4) = (1,0)$ の次のクロックでのみデータを取り込む。カウンタ $(FF_3, FF_4)$ は (0,0) から (1,0) となるのに 3 クロック必要であるので、3 サイクル FF 対であると分かる。

### 2.3 マルチサイクル FF 対の条件

FF 対 $(A, B)$ がマルチサイクル FF 対である必要十分条件は、

「$(A, B)$ がマルチサイクル FF 対」
⇔ 「A での信号変化が次クロックでは B に到達しなくてよい」
⇔ 「(1)A の信号変化が次時刻で B に到達しない、又は、
(2)B に到達するが外部出力から観測不能」

である[1]。

条件 (2) の「B に到達するが外部出力から観測不能」という条件は、1 時刻前に A が信号変化するという条件下で B をソースとする全ての FF 対がマルチサイクル FF 対であること意味する。この条件も含めてマルチサイクルパス解析を行うと B をソースとするマルチサイクルパスの一部を前倒しで利用することになるため、本稿では、条件 (1) の「A での信号変化が次のクロックで B に到達しない」という条件のみを用いた解析を行う。さらに、A での信号変化が起った次のクロックで B に信号変化がなければ、A での信号変化が B に到達していないことが言えるので[2]、以下の関係が成り立つ。

「$(A, B)$ がマルチサイクル FF 対」
⇐ 「A の信号変化が起った次時刻で B の値が不変」 (3)

本稿ではこの十分条件を利用してマルチサイクル FF 対の解析を行う。このマルチサイクル条件は [8] と同一である。例として図 1 の回路を考えると、$(FF_3, FF_4)$ =(0,0) の次のサイクルのみで $FF_1$の信号が変化し、$(FF_3, FF_4) = (0,1)$ のサイクルで $FF_2$ の信号変化が $MUX2$ でブロックされシンクの値は不変なので条件 (3) を満たしていることが分かる。

### 2.4 マルチサイクル FF 対検出問題

本稿では以下のようなマルチサイクルパス検出問題の解法を考える。

入力: 単一のクロックに同期した論理回路
出力: 論理回路中のマルチサイクル FF 対全て

## 3 マルチサイクルパス解析手法

本章では、前章で定義したマルチサイクル FF 対検出問題を高速に解くアルゴリズムを提案する。

### 3.1 アルゴリズムの流れ

マルチサイクル FF 対検出問題を解くにはソースの信号変化前のサイクルと変化後のサイクルの関係を調べる必要がある。そのため、与えられた論理回路を 2 クロック分時間展開した組み合わせ回路を用いる。解析は基本的に各 FF 対について ATPG で用いられる含意操作とバックトラックつき探索により条件 3 が成り立つ値割り当てが存在しないことを調べる。

本手法によるマルチサイクル FF 対の検出は以下の流れで行う。ただし、時刻 $t$ でのノード $A$ の出力値を $A(t)$ などと表す。またノード $A$ に値 $v$ を代入するとはノード $A$ の出力の値を $v$ にすることを示す。

1) 全ての FF 対についてその間にパスが存在するかどうか調べ、存在するもの全てを候補とする。
2) 乱数パタンシミュレーションにより 2.3 節の条件 3 を満足しない FF 対を候補から除く。
3) 組み合わせ回路部分を 2 時刻分時間展開する。時刻を t=0,1,2 とする[3]
4) 候補の FF 対 $(A, B)$ 各々に対して以下を行う。
4.1) $(A(0), B(1))$ がとりうる任意の値の組合わせ $(v_a, v_b) = \{(0,0), (0,1), (1,0), (1,1)\}$ に対して以下の 4.1.1)~4.1.5) を行う。
4.1.1) $A(0) \leftarrow v_a, A(1) \leftarrow \overline{v_a}, B(1) \leftarrow v_b$
4.1.2) 含意操作を行う。
4.1.3) $B(2) = B(1)$ であるか、含意操作中に矛盾が生じれば 4.1) に戻る。
4.1.4) 時間展開した回路上で $B(2) \neq B(1)$ となる入力パタンを探索する。

[1] B がマルチサイクルで A の値を受けるためには A で信号変化した次のクロックで A は値を保持しておくという条件が必要であると思われるかもしれない。しかしこの条件は不要である。なぜならば、A で値が保持されず変化が生じた場合、最初の信号変化後の値はその次のクロックで B に到達しなければ永遠に B に到達しないサイクル数無限大のマルチサイクルパス、すなわち、フォールスパスであると言えるからである。

[2] 逆は必ずしも真ではない。例えば、A 以外の FF の信号変化が B に到達する場合。

[3] 時刻 2 の値をとりうるのは FF の出力のみ。

4.1.5) 入力パタンが存在すれば $(A, B)$ をシングルサイクルと判定して 4) に戻る。

4.2) FF 対 $(A, B)$ をマルチサイクルと判定する。

本アルゴリズムはマルチサイクル FF 対を全て求めるものであるが、ステップ 3 での時間展開する時刻数を増やすことで $k$ サイクル $(k = 3, 4, \ldots)$ 以上の FF 対を検出するよう容易に拡張できる。

### 3.2 アルゴリズムを適用した例

図 1 の論理回路に前節のアルゴリズムを適用する。

まずステップ 1 でパスが存在する FF 対を求めると、全 FF 対 16 個のうち $\{(FF_1, FF_1), (FF_1, FF_2), (FF_2, FF_2), (FF_3, FF_1), (FF_3, FF_2), (FF_3, FF_4), (FF_4, FF_1), (FF_4, FF_2), (FF_4, FF_3)\}$ の 9 個が候補として残る。ステップ 2 で乱数パタンシミュレーションを行うと候補が $\{(FF_1, FF_1), (FF_1, FF_2), (FF_2, FF_2), (FF_3, FF_2), (FF_4, FF_1)\}$ に絞られる。ここでは乱数パタンシミュレーションの詳細は省略する。ステップ 3 で組み合わせ回路部分を 2 時刻分時間展開する。

次にステップ 4 で FF 対として $(FF_1, FF_2)$ を選んだ場合を考える。ステップ 4.1 で値割り当てとして $(v_a, v_b) = (0, 0)$ をまず選ぶ。ステップ 4.1.1 でソースの $FF_1$ が時刻 1 で信号が変化する場合を調べるため $FF_1(1)$ に $\overline{FF_1(0)} = 1$ を割り当てる。ステップ 4.1.2 で含意操作を行い、ステップ 4.1.1 の値割当てから一意的に決められる値を割り当てていく。

含意操作を行った後の値割り当ては図 2 のとおりである。図中、丸印の付いた値は含意操作前から割り当てられている値、それ以外の値は含意操作で得られた値を示す。図 2 より、$FF_2(2) = 0$ が得られているので、$FF_2$ は $FF_1$ が立ち上がりの信号変化が起った次のクロックでどのような入力と回路の状態のもとでも値が変化しないことが分かる。すなわち、2.3 節の条件 3 が成り立ち、$(v_a, v_b) = (0, 0)$ の場合には $(FF_1, FF_2)$ はマルチサイクルと判定できる。従ってステップ 4.1 に戻り、以下同様に $(0, 1), (1, 0), (1, 1)$ の場合を順に調べる。含意操作の結果、もしマルチサイクルパス条件が満たされず、かつ、矛盾も生じ無い場合には、ステップ 4.1.4 で ATPG を用い $FF_2$ で値が変化する入力パタンが存在しないかどうかを調べ、存在した場合その FF 対をシングルサイクルと判定する。全ての 5 つの候補の FF 対についてマルチサイクルパスかどうかを判定すると図 1 の回路については全ての候補がマルチサイクルパスであると判定できる。

### 3.3 アルゴリズムの各ステップの説明

#### 3.3.1 乱数シミュレーション

乱数シミュレーションは 1 ワードパラレルのシミュレーションを行う。1 ワードの乱数を各外部入力と FF 出力に対して割り当て 2 クロック分シミュレーションする。残っている FF 対 $(A, B)$ のそれぞれについて "$A(0) \neq A(1) \wedge B(1) \neq B(2)$" の関係が成り立つビットがあるかを調べ、あればその FF 対をシングルサイクルとして候補から除く。上の論理演算はビット演算によりワード長分並列に行える。この 2 クロック分の乱数シミュレーションを与えられたパタン数だけ繰り返し行う。

#### 3.3.2 含意操作 (implication)

割り当てられた値を伝搬させることにより一意的に決められる信号値を可能な限り割り当てていく。途中で値割り当てに矛盾が生じた場合には、含意操作前に割り当てられた値の組合わせが実際にはあり得ないことを示している。その場合、そのような組合わせに対してはマルチサイクルかどうかを判断する必要がないこと意味する。

含意操作は、遠く離れた信号線間の含意関係を記憶しておく学習機能を追加することにより、さらに多くの含意関係を求めることができる [10]。

#### 3.3.3 入力パタンの探索

ATPG のパタン生成と同様にして行う。ATPG のアルゴリズムは D アルゴリズムを用いている。本問題の特徴は ATPG を適用する際にあらかじめいくつかの値割当てが行われているという点と、入力パタンが無いことを調べにいくという点にある。従って、外部入力までバックトレースして新たな値を割り当てる PODEM アルゴリズムより、回路中のノードにもどんどん値を割り当てる D アルゴリズムの方が矛盾を早く見つけることができると考えた。ATPG と同様にバックトラック数の制約を設けることにより処理時間と探索能力の高さのトレードオフを外部から制御できる。

## 4 静的ハザードに関する考察

本稿で提案する手法により得られるマルチサイクル FF 対は必ずしも任意の素子遅延および回路遅延モデルのもとで 2.3 節の条件 3 が成り立つことを保証していない。本手法では静的ハザードを考慮していないからである。シンクに静的ハザードがおこり、かつ、定常状態への遷移時刻がマルチサイクル FF 対と同定されたパスに依存する場合、そのパスのタイミング制約をマルチサイクルに緩めると実際のパス遅延によっては静的ハザードの非定常値をシンクが取り込む可能性がある。従って、本手法で得られたマルチサイクル制約を利用する場合 FF の値が正常値であるというチェックも必要である。それを行わない場合には、マルチサイクル FF 対について、遅延独立なパス活性化条件の上界を与える static cosensitization 条件 [4] を用いてパス活性化の可能性があるかをチェックするか、遅延独立なパス活性化条件の下界を与え

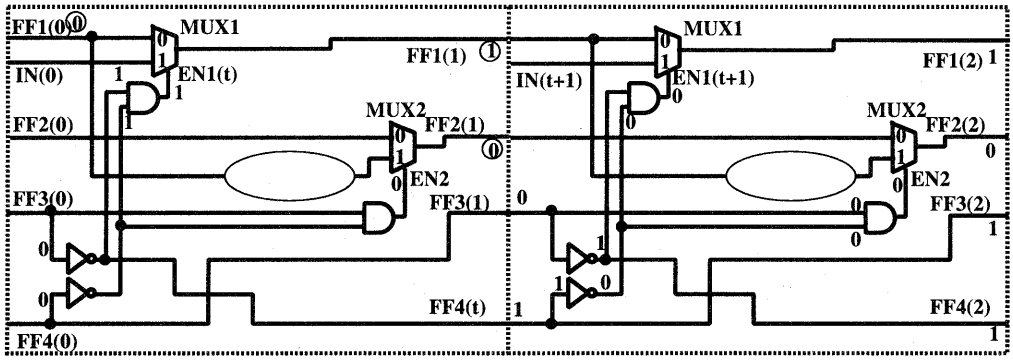


図 2: 含意操作の結果

る static sensitization によりそのパスをブロックするパスを見つけブロックするパスがマルチサイクル指定されないことを保証するか、のどちらかの対応が必要となる。

## 5 実験結果

前章で述べたマルチサイクル FF 対検出アルゴリズムを C++を用いて実装し、性能評価を行った。

性能評価では、ISCAS89 ベンチマーク回路に対するマルチサイクルパス検出の実験を行い、従来法 [8] との比較を行った。本実験では、乱数シミュレーションについては、FF 対候補数に変化がなくなってから 32 × 10 パタン後に終了するようにした。また、入力パタン探索のバックトラック数を最大 50 に制限した。ただし、“s9234.1”,“s9234”,“s13207.1”,“prolog” については最大 50 ではバックトラック制限をオーバーしてマルチサイクルパスかどうか判定できない FF 対が存在したため、バックトラック数制約をそれぞれ 600,150,150,51 とし、さらに “s9234.1”,“s9234” については学習機能をオンにした。本手法による実験を行った計算機は Pentium III(864MHz、主メモリ 512MB)+FreeBSD4.4-RELEASE で、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。その結果を表 1 に示す。

表 1 において、「In」は回路の外部入力数、「FF」は FF 数、「FF 対」はパスが存在する FF 対の数である。「本手法」は本稿で提案した手法による結果、「従来法」は [8] による結果である。また、「MC 対」はマルチサイクル FF 対の数、「時間 (秒)」は回路の読込みから結果の出力までの CPU 時間を秒で示したものである。ただし、従来法の結果は [8] の結果をそのまま引用したものであり、使用計算機は DEC AlphaServer8400(CPU alpha21164A 617MHz × 10、主メモリ 8GB) である。表中「-」は [8] の結果の表に記載されていなかった回路であることを示す。一番下の行の「総和」は各行の総和である。ただし、[8] については、結果の得られている回路についてのみの和である。

表 1 より、本手法が全てのベンチマーク回路に対して極めて高速にマルチサイクル FF 対が検出できていることが分かる。FF 数が 1000 以上の回路に対しても数十秒で解析が終了しており、十分実用に耐える性能であると考える。また、全ての回路で全ての FF 対についてマルチサイクルかどうかの判定ができた。回路中に存在するマルチサイクル FF 対の数はパスの存在する FF 対の約 1 割程度で、マルチサイクル FF 対が回路中にある程度は存在していることが分かる。

また、従来法と比較すると、実験で用いた計算機の違いがあるので直接比較はできないが、処理時間の点で本手法が大幅に有利であることが分かる。従来法ではソースとシンクが同一 FF である FF 対は解析を行っていないと思われるが、本手法ではそのような FF 対に対しても解析を行っている。ソースとシンクが同一 FF でない FF 対に関して本手法により検出されたマルチサイクルパス数は従来法の数と同一であった。

表 2 は、本アルゴリズムの各ステップでマルチサイクルおよびシングルサイクルと同定された FF 対の数、その割合、及び処理時間の総和である。「Sim.」は乱数シミュレーションを表す。表より乱数シミュレーション後に残った FF 対の約半数が含意操作により高速にマルチサイクルかどうかの判別が行えていることが分かる。

## 6 おわりに

本稿では大規模な同期式順序回路に適用可能な高速なマルチサイクル FF 対検出アルゴリズムを提案した。実験結果より FF 数が 1000 以上の回路に対して数十秒で検出可能であることが分かり、実用に耐

表 1: マルチサイクルパス検出の実験結果

| 回路 | In | FF | FF 対 | 本手法 MC 対 | 本手法 時間 (秒) | [8] 時間 (秒) |
|---|---|---|---|---|---|---|
| s208.1 | 10 | 8 | 36 | 28 | 0.3 | - |
| s208 | 11 | 8 | 36 | 0 | 0.3 | 0.5 |
| s298 | 3 | 14 | 70 | 3 | 0.3 | 0.9 |
| s344 | 9 | 15 | 89 | 1 | 0.3 | 1.3 |
| s349 | 9 | 15 | 89 | 1 | 0.3 | 1.3 |
| s382 | 3 | 21 | 146 | 13 | 0.3 | 2.0 |
| s386 | 7 | 6 | 36 | 4 | 0.3 | 0.9 |
| s400 | 3 | 21 | 146 | 13 | 0.3 | - |
| s420.1 | 18 | 16 | 136 | 120 | 0.4 | - |
| s420 | 19 | 16 | 88 | 0 | 0.3 | 1.1 |
| s444 | 3 | 21 | 146 | 13 | 0.6 | 2.6 |
| s499 | 1 | 22 | 484 | 379 | 0.5 | - |
| s510 | 19 | 6 | 36 | 3 | 0.3 | 1.4 |
| s526 | 3 | 21 | 144 | 4 | 0.3 | 2.5 |
| s526n | 3 | 21 | 144 | 4 | 0.3 | 2.5 |
| s635 | 2 | 32 | 528 | 0 | 0.5 | - |
| s641 | 35 | 19 | 115 | 1 | 0.4 | 1.9 |
| s713 | 35 | 19 | 115 | 1 | 0.3 | 2.6 |
| s820 | 18 | 5 | 25 | 0 | 0.3 | 1.7 |
| s832 | 18 | 5 | 25 | 0 | 0.3 | 1.6 |
| s838.1 | 34 | 32 | 528 | 496 | 1.1 | - |
| s838 | 35 | 32 | 192 | 0 | 0.3 | 3.0 |
| s938 | 34 | 32 | 538 | 496 | 1.0 | - |
| s953 | 16 | 29 | 156 | 23 | 0.4 | 6.2 |
| s967 | 16 | 29 | 156 | 24 | 0.4 | - |
| s991 | 65 | 19 | 71 | 20 | 0.3 | - |
| s1196 | 14 | 18 | 20 | 0 | 0.3 | 2.2 |
| s1238 | 14 | 18 | 20 | 0 | 0.4 | 2.4 |
| s1423 | 17 | 74 | 1765 | 47 | 0.4 | 60.4 |
| s1488 | 8 | 6 | 36 | 0 | 0.4 | 2.0 |
| s1494 | 8 | 6 | 36 | 0 | 0.3 | 2.0 |
| prolog | 36 | 136 | 578 | 19 | 0.6 | - |
| s1269 | 18 | 37 | 288 | 3 | 0.3 | - |
| s1512 | 29 | 57 | 513 | 1 | 0.4 | - |
| s3271 | 26 | 116 | 899 | 0 | 0.4 | - |
| s3330 | 40 | 132 | 549 | 0 | 0.5 | - |
| s3384 | 43 | 183 | 1831 | 0 | 0.4 | - |
| s4863 | 49 | 104 | 628 | 0 | 1.1 | - |
| s5378 | 35 | 179 | 1200 | 55 | 0.8 | - |
| s6669 | 83 | 239 | 2179 | 0 | 0.6 | - |
| s9234.1 | 36 | 211 | 2681 | 28 | 12.0 | - |
| s9234 | 19 | 228 | 2830 | 159 | 11.5 | - |
| s13207.1 | 62 | 638 | 3411 | 580 | 18.2 | - |
| s13207 | 31 | 669 | 3716 | 937 | 14.6 | - |
| s15850.1 | 77 | 534 | 11873 | 320 | 8.3 | - |
| s15850 | 14 | 597 | 15363 | 3756 | 10.4 | - |
| s35932 | 35 | 1728 | 4763 | 0 | 2.3 | - |
| s38417 | 28 | 1636 | 33852 | 240 | 39.3 | - |
| s38584.1 | 38 | 1426 | 16372 | 17 | 10.1 | - |
| s38584 | 12 | 1452 | 17978 | 1622 | 14.8 | - |
| 総和 | | 10908 | 125504 | 9435 | 158.8 | 103.0 |

表 2: 各段階で同定した FF 対の数・処理時間の総和

| | Sim. | 含意操作 | ATPG |
|---|---|---|---|
| シングルサイクル | 107974 (86%) | 261 (0.2%) | 122 (0.1%) |
| マルチサイクル | 0 (0%) | 7712 (6.1%) | 9435 (7.5%) |
| 処理時間 (秒) | 29.4 | 29.5 | 76.6 |

えうる性能であるといえる。

今後、生成されたマルチサイクル指定が、タイミングクロージャー問題などに有効に利用できるかどうかを確かめていきたい。また、現在は単一同期式回路を対象としているが、独立でないクロックが複数存在している回路に対する拡張も考える予定である。

## 参考文献


[1] V. Hrapcenko. Depth and Delay in a Network. *Soviet Math. Dokl.*, 19(4), 1978.

[2] D. Brand and V. S. Iyengar. Timing Analysis using Functional Relationships. In *Proceedings of IEEE International Conference on CAD-86*, pages 126–129, November 1986.

[3] 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.

[4] S. Devadas, K. Keutzer, and S. Malik. Computation of Floating Mode Delay in Combinational Circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits*, 12(12):1913–1923, December 1993.

[5] 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.

[6] A. P. Gupta and D. P. Siewiorek. Automated Multi-Cycle Symbolic Timing Verification of Microprocessor-based Designs. In *Proceedings of the 31th ACM/IEEE Design Automation Conference*, pages 113–119, 1994.

[7] K. Nakamura, S. Kimura, and K. Watanabe. Timing Verification of Sequential Logic Circuits based on Controlled Multi-Clock Path Analysis. *IEICE Trans. on Fundamentals*, E81-A(12):2515–2520, December 1998.

[8] 中村 一博, 丸岡 新治, 木村 晋二, 渡邊 勝正. 充足可能性判定手法に基づいたマルチクロックパス解析. In 信学技報 *VLD99-82,ICD99-211,FTS99-60*, pages 55–62, November 1999.

[9] W.-C. Lai, A. Krstic, and K.-T. Cheng. Functionally Testable Path Delay Faults on a Microprocessor. *IEEE Design & Test of Computers*, oct 2000.

[10] M. Schulz, E. Trishler, and T. Sarfert. Socrates: A Highly Efficient Automatic Test Pattern Generation System. In *Proceedings of International Test Conference*, pages 1016–1026, 1987.