順序回路の時間展開を用いた前方順序的クロックゲーティングの自動挿入について

木村 晋二[1,a)]　後藤 智哉[1]　尤 云潔[1]　柳澤 政生[1]

概要：レジスタへのクロック供給を制御信号と論理ゲートで停止するクロックゲーティングは、LSI の動的電力を削減する手法として広く用いられている。近年、現在の値だけでなく、過去の値や未来の値を用いる順序的クロックゲーティングが注目されている。ここでは、過去の信号を制御信号として用いる前方順序的クロックゲーティングを自動挿入する手法を提案する。レジスタの現在の値と新しい値の EXOR を更新条件として候補を判定する手法を時間展開された回路に用いることで、過去の信号を含めて最適な制御信号を選ぶ。提案手法で、過去の信号候補と共に、これまで発見できなかった現在の信号も候補として発見でき、クロックの停止確率を最適化できた。

キーワード：ラッチベース CG、クロックイネーブル確率、制御信号の検出と選択、二分決定グラフ

Automatic Insertion of Forward Sequential Clock Gating Logic Using Time Expanded Circuits

Shinji Kimura[1,a)]　Tomoya Goto[1]　Yunjie You[1]　Masao Yanagisawa[1]

Abstract: Clock gating is to stop clocks to registers using a logic gate with a control signal and is widely used for reducing the dynamic power of LSI. Recently, sequential clock gating has been paid attention where not only current signals but also past and/or future signals are considered. This manuscript proposes an automatic insertion of forward sequential clock gating logic with past signals. The EXOR of a register's current and new values is used to check a candidate signal including past times. The proposed method can detect not only past signals but also current signals which cannot be detected for single time step circuits.

Keywords: Latch based CG, clock enable probability, control canddidate extraction and selection, BDD

1. はじめに

通常のディジタルシステムは同期式順序回路として設計されることが多い。同期式回路では、内部データを保持するための記憶素子であるフリップフロップがクロック信号の変化に応じて保持している値を変化させる。回路の大規模化とともにフリップフロップの個数も増加し、フリップフロップの消費電力削減が重要である。

クロックゲーティングは、フリップフロップへの不必要なクロック供給を停止し、フリップフロップの動的電力を削減する手法である。停止には、論理ゲートと制御信号を用いる。クロックゲーティングは、マルチプレクサの代替となるため、面積や遅延に良い影響を与えることも多い。

通常のフリップフロップへの新しい値の代入は、特定の状態など条件付であることが多いので、その条件をクロックゲーティングの制御信号とすることができる。商用の設計ツールでは、このようなナイーブな制御信号を用いたクロックゲーティングの生成を行うことができる。クロックゲーティングの電力削減効果は、制御信号に依存するので、最適な制御信号を求められれば、それを条件とする代入で最適なクロックゲーティングを実現できる。

[1] 早稲田大学
Waseda University, 3-4-1 Okubo, Shinjuku, Tokyo, 169-8555 JAPAN
[a)] shinji_kimura@waseda.jp

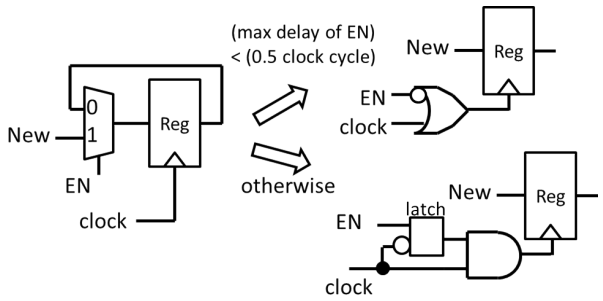

図 1 条件付き代入とクロックゲーティング.

これまで、自動的に制御信号を探索する手法がいくつか提案されている ([1], [2], [3], [4], [5], [6])。とくに近年、現在の時刻の信号だけを考えてクロックゲーティングを行うのではなく、過去や未来の信号も考慮した順序的クロックゲーティングが注目されている。

文献 [1] では、フリップフロップの値のオブザーバビリティを用いて未来の信号を制御信号として用いる方法が提案された。ただし、未来の信号を現在に戻すための付加回路のオーバーヘッドが大きい。また、文献 [7], [8] では、順序的クロックゲーティングの分類とその制御の正当性の検証手法が提案されている。

現在の信号のみを考慮したクロックゲーティングにおいて、制御信号が満たすべき十分条件を用いた検出手法とヒューリスティックな選択手法が提案され [2]、それを改良した種々の手法が提案されている ([3], [9])。この手法は各フリップフロップの現在の値と新しい値の EXOR を判定条件とするもので、これと AND をとって unsatisfiable である信号の否定をクロック制御の候補とするものである。

本稿では、この判定手法を時間展開された回路に適用することで、過去の信号を制御信号として用いる前方順序的クロックゲーティングの候補判定手法を提案する。また、複数ある制御信号候補の中から最適なものを選ぶ問題の制約条件を論理式で表し、制約条件を満たす最小コストの解を選ぶ問題に帰着し、二分決定グラフを用いて解く手法を提案している。また、制約の緩和手法の提案も行っている。

提案手法は C 言語および二分決定グラフパッケージ上で実現され、ISCAS 89 順序回路ベンチマークで評価を行った。また、現時刻の回路のみからクロックゲーティングを行う手法に比べ、最小コストを小さくできることを示している。

2. 準備

2.1 クロックゲーティング

クロックゲーティングは不必要なクロックの変化を停止することで、エッジトリガ型のフリップフロップの動的電力を削減する手法である。フリップフロップのクロック入力の前に、クロックと制御信号を入力とする論理ゲートを置き、制御信号でクロックエッジの発生を制御することで値の変化を制御する。

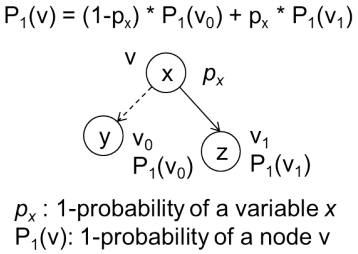

図 2 Probability Computation on BDD.

クロックゲーティングはレジスタ転送レベル設計における条件付代入 `if (EN) Reg <= New;` に対応している。この条件付代入の実現手法を図 1 に示す。組合せ回路的には、マルチプレクサ (Mux) を用いて、新しい値 New と現在の値を条件信号 EN に応じて選択することで実現する。一方、クロックゲーティングでは EN でクロック変化を制御することで New の取り込みを制御するので、Mux は不要で New を直接フリップフロップの入力に接続できる。これにより Mux 部が削減されるが、クロックゲーティング制御の回路を追加する必要がある。

クロックを制御する回路は EN の計算時間に応じて 2 通り構成できる。一つは EN の否定とクロックの OR 演算の結果をゲート後のクロックとするもので、EN の最大計算時間がクロックの半サイクル以下である場合に使用できる。もう一つは EN をクロックの否定でラッチした結果とクロックの AND 演算の結果をゲート後のクロックとするもので、 EN の計算時間に無関係に使用できる。ゲート後のクロック信号の 0 のパルス部 (ポジティブエッジの場合) は、EN の変化にかかわらず、元のクロックの波形と同じになる。

商用ツールは条件付き代入に対して、ユーザーのオプション指定に従い、記述された制御条件そのままでクロックゲーティングの制御回路を自動挿入することができる。ラッチベースクロックゲーティングを用いる場合は、制御回路の電力オーバーヘッドを考慮し、デフォールトでは 3 ビット以上のフリップフロップが同じ制御回路を共用できる場合にクロックゲーティング制御を用い、そうでない場合は Mux による実現とする。

2.2 二分決定グラフと 1 確率の計算

二分決定グラフ (Binary Decision Diagram, BDD) は論理関数を表す有向非巡回グラフであり [10], [11]、二分決定グラフ上で論理演算を実行できる。BDD の各節点は論理関数を表す。BDD は論理の 0 と 1 の定数関数を表す二つの定数節点と、変数でラベル付された変数節点を持つ。定数節点はそこから出る枝を持たない。一方変数節点は、ラベルの変数が 0 となる場合の関数を表す節点への 0 枝と、1 となる場合の関数を表す節点への 1 枝の二つの出て行く

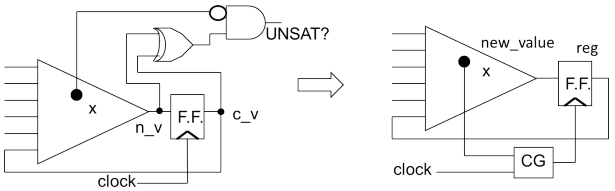

図 3 Clock Gating Candidate Detection [2].

枝を持つ。

論理関数 $f(x_0, x_1, ..., x_{n-1})$ を表す BDD の構造は、

$$f(x_0, x_1, ..., x_{n-1}) = (\overline{x_0} \wedge f(0, x_1, ..., x_{n-1})) \vee (x_0 \wedge f(1, x_1, ..., x_{n-1}))$$

で表されるシャノン展開に対応している。ここで $\overline{x_0}$ は x_0 の否定を、$\wedge$ は論理積を、$\vee$ は論理和を表す。$f(x_0, x_1, ..., x_{n-1})$ を表す節点は、変数 x_0 をラベルとして持ち、x_0 が 0 の場合の $f(0, x_1, ..., x_{n-1})$ を表す節点への 0 枝と、x_0 が 1 の場合の $f(1, x_1, ..., x_{n-1})$ を表す節点への 1 枝を持つ。

BDD はゲートレベルの回路あるいは論理式表現から構成することができる。まず外部入力に対する 1 変数関数の BDD を構成し、それらの BDD 間の論理演算を BDD 上で行うことによる。

ある論理関数の BDD を構成すると、その論理関数が 1 となる確率を計算することができる。計算方法を図 2 に示す。1 定数節点の 1 確率は 1.0、0 定数節点は 0.0 とし、変数節点はその変数が 1 となる確率に 1 枝の節点の 1 確率を乗じたものと、0 となる確率に 0 枝の節点の 1 確率を乗じたものの和となる。

制約条件を満たす論理関数の BDD に対し、各変数にその変数が 1 となる場合のコストが割り当てられているとすると、BDD をグラフと見て、1 定数節点へ行く最小コストのパスを求めることができる。具体的には、0 定数節点のコストを無限大、定数 1 節点のコストを 0 とし、変数節点のコストを、0 枝に接続されている節点のコストと、1 枝に接続されている節点のコストに変数のコストを加えたコストの小さい方とする。変数節点では、どちらの枝を選んだかを記憶しておく。ルートから各節点での選択結果をたどることで、最小コストのパスが求められる。

3. 時間展開回路での制御信号候補抽出

3.1 クロックゲーティング制御信号候補の満たすべき条件

Hurst は 文献 [2] において、フリップフロップのクロックゲーティングの制御信号が満たすべき条件を示し、各フリップフロップの制御信号候補を求め、その中から制御回路のコストと制御信号の共有を考慮しつつヒューリスティックに制御信号を選ぶ手法を提案している。あるフリップフロップのクロックゲーティングの制御信号が満たすべき条件は、フリップフロップの現在の値と次のクロックでの値 (すなわち現時刻でのフリップフロップの入力、今後新しい値と呼ぶ) が異なる場合、制御信号は必ず 1 でないといけないというものである。

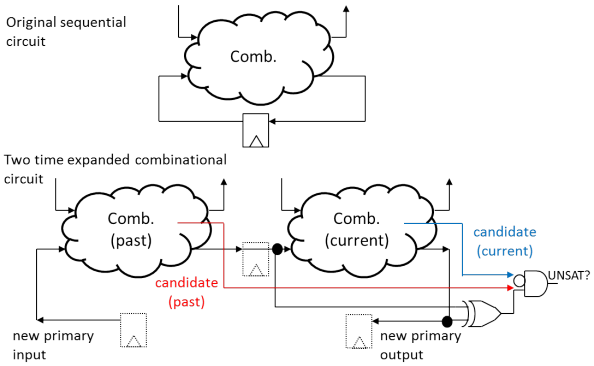

図 4 Candiadate Detection in a Time Expanded Circuit

候補となる条件を言い換えると、フリップフロップの現在の出力値と現在の入力値の EXOR に対し、信号の否定との AND をとったものが常に 0 であることと表せる。この条件を回路で表すと図 3 に示すようになる。文献 [2] では、判定の時点では信号そのままを用いて、制御信号として用いる時に否定して用いたが、ここではイネーブルの意味で一貫性を取るため、判定の時点で否定して unsatisfiability を判定し、成り立てばそのままクロックゲーティングの制御信号として用いる。一般に、この条件を満たす制御信号の候補は回路中に複数見つかる。なお、判定回路は、候補を判定する手法を表したもので、この回路を実際に使用することはない。

3.2 時間展開回路での候補判定

回路の時間展開は、順序回路の組合せ回路部を複数コピーし、フリップフロップの対応部分をコピーした回路間で接続したものであり、順序回路を組合せ回路として解析するために用いられる。時間展開で、論理ゲート、外部入力、外部出力は展開した時間分だけ増加する。ここでは、この時間展開された回路を用い、過去の時刻を含めてクロックゲーティングの制御候補を求める手法を提案する。

以下では簡単のために、図 4 に示す 2 時刻分の展開回路を用いる。元の順序回路の組合せ回路部 (Comb.) は 2 つコピーされる。下図の左側が 1 時刻前 (past)、右側が現在の時刻の回路 (current) に対応する。1 時刻前の組合せ回路部のフリップフロップへの出力と、現時刻の組合せ回路部のフリップフロップからの入力を接続する。1 時刻前の組合せ回路部のフリップフロップからの入力は外部入力とする。現時刻の組合せ回路部のフリップフロップへの出力は外部出力とする。組合せ回路部の外部入力と外部出力はそのまま新しい回路でも外部入力と外部出力となる。

展開された回路での制御信号候補の探索は、現時刻だけの回路に対するものと同様であるが、条件判定での対象の

表 1 フリップフロップと制御信号候補の関係変数表

Sig. (Prob) Circuit	c_1 (p_1)	c_2 (p_2)	c_3 (p_3)	c_4 (p_4)	c_5 (p_5)	— (1.0)
r_1	x_{11}	x_{12}	—	—	x_{15}	z_1
r_2	x_{21}	—	—	—	—	z_2
r_3	—	x_{32}	x_{33}	x_{34}	x_{35}	z_3
—	y_1	y_2	y_3	y_4	y_5	—

フリップフロップの現在の値は 1 時刻前の組合せ回路の出力で、新しい値は現時刻の組合せ回路の出力となる。判定では、これらの EXOR に対し、現在および過去の組合せ回路の内部信号の否定との AND をとり、unsatisfiable となればその信号が制御信号の候補となる。

現時刻の信号を制御信号として用いる場合はそのままで使用できるが、過去の信号を制御信号として用いる場合は、現時刻まで信号を伝搬する必要がある。1 時刻前であれば、フリップフロップを 1 つ用いて信号を現時刻まで伝えることができる。一般に k 時刻前であれば k 個のフリップフロップを用いたシフトレジスタが必要となる。

4. クロックゲーティング候補の選択

4.1 候補選択問題

前節で示した手法により、各フリップフロップのクロックゲーティング制御信号候補が得られる。ここではその中で最適な制御信号を 1 つ選ぶ問題を扱う。なお、クロックゲーティングを行わないという選択もあり得る。

制御候補については、その信号が 1 になる確率がわかっているものとする。制御候補は Enable として用いるので、1 の場合にクロックが入れられる。そこで、全フリップフロップについてクロックが入れられる確率を最小化するコスト最小化問題を扱う。クロックが入れられるとその分の電力が消費されるので、この確率は電力を表しているとみることもできる。停止確率は、制御候補の論理関数や、ランダムパターンシミュレーションでを用いて求めることができるので、事前に計算しておくとする。

まずクロックゲーティングを行わない場合はそのフリップフロップのコストは 1.0 である。1 確率が p_i である信号 c_i でクロックゲーティングを行う場合のコストは p_i である。ただし、制御回路のコストがかかる。なお、制御回路のコストは信号 c_i について 1 回だけで、複数のフリップフロップで同じ制御信号を用いても同じ追加コストで済む。

制御回路の追加コストとして、制御信号の遅延にロバストであるラッチベースの制御回路の電力比を用いる。先に述べたように、確率と電力は比例しているので、確率と同じように扱うことができる。実際の追加コストは、フリップフロップの消費電力を 1.0 として、ラッチベース制御回路の消費電力が何倍になっているかを実際に求めておく。本稿では、あるセルライブラリでの実測値の 0.8 を用いる。

上記のクロック印加確率に対応するコスト定義の下で、全フリップフロップのコストおよび全制御回路のコストの総和を最小化する手法を示す。

4.2 候補選択の条件

候補選択の条件の第一は、各フリップフロップが一つしか候補信号を選べないという条件である。クロックゲーティングを行わない場合もあり得るので、それも含めて排他条件を設定する。条件の第二は、ある制御信号が選ばれた場合に、その制御信号に対するコストを加算することに対応するものである。

ここではこれらを論理式として表すために、各行にフリップフロップを、各列に制御信号候補を並べた二次元表を考え、その交点に対して論理変数 x_{ij} を置く。$x_{ij} = 1$ となるのは、i 番目のフリップフロップが j 番目の制御信号を用いる場合である。フリップフロップと候補の関係により、x_{ij} が 0 にしかなれない ($x_{ij} \equiv 0$) こともあり、この場合は変数自体を導入する必要がない。また、クロックゲーティングを行わない場合を表すために z_i を導入し、付加回路のコストのために y_j を導入する。

各変数 x_{ij} には、制御信号の 1 確率 p_j がコストとして割り当てられる。z_i はクロックゲーティングしない場合に対応し、コストは 1.0 である。y_j は制御回路のコストに対応し、コストは 0.8 である。

表 1 にフリップフロップが 3 個で制御候補が 5 個の例を示す。交点の “—” は $x_{ij} \equiv 0$ を表す。

各フリップフロップに対する制御信号の排他条件は、x_{ij} および z_i のどれか一つしか 1 にならない、と表せる。論理式としては全ての変数の AND で、一つの変数だけがそのまま、他はすべて否定としたものの、OR となる。例えば r_1 に対する制御信号の排他条件は

$$(x_{11} \wedge \overline{x_{12}} \wedge \overline{x_{15}} \wedge \overline{z_1}) \vee (\overline{x_{11}} \wedge x_{12} \wedge \overline{x_{15}} \wedge \overline{z_1}) \vee (\overline{x_{11}} \wedge \overline{x_{12}} \wedge x_{15} \wedge \overline{z_1}) \vee (\overline{x_{11}} \wedge \overline{x_{12}} \wedge \overline{x_{15}} \wedge z_1)$$

と表せる。全体の排他条件は各フリップフロップの排他条件の AND である。

つぎに追加コストに関する条件は、x_{ij} が 1 であるならば、y_j も 1 である $x_{ij} \rightarrow y_i$ を、すべての i に対して AND したもので表せる。例えば c_1 については、
$(x_{11} \rightarrow y_1) \wedge (x_{21} \rightarrow y_1)$ であり、$(\overline{x_{11}} \wedge \overline{x_{21}}) \vee y_1$ と表せる。全体の追加回路コスト条件は、各制御信号のコスト条件の AND である。

これらの排他条件と追加回路コスト条件を AND したものが制約条件となり、条件を満たす論理変数への値割当が実現可能な解となる。

4.3 条件の緩和と最小コスト選択問題

制約条件を 1 とする論理変数への値割当に対し、1 である変数のコストを加えたものがその解のコストとなる。x_{ij}

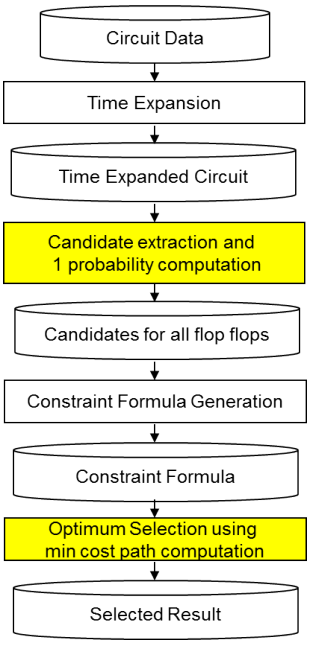

図 5 Flow of Processing.

が 1 となる場合は、制御信号に対する p_j をコストとして加え、かつ y_j が 1 になるので 0.8 を加える。

制御条件を満たす論理式から、二分決定グラフを構成することができ、二分決定グラフ上で 1 定数節点にゆく最小コストパスを求めることができる。ただし、二分決定グラフの処理に各変数のコスト属性を扱う処理を追加する必要がある。アルゴリズムは 2 節に示した通りである。

しかし、元の排他条件のままでは二分決定グラフが大きくなることがわかった。そこで、最小コスト問題であることに着目し、排他条件を緩和することとした。具体的には排他条件のかわりに行方向の変数のどれかが 1 になることを表す OR を条件とした。例えば r_1 については $x_{11} \vee x_{12} \vee x_{16} \vee z_1$ と表せる。この緩和により、論理関数は非常に簡単となり、二分決定グラフの節点数も押さえられる。

緩和により、複数の論理変数が同時に 1 となる場合も含まれるが、これは複数の制御変数でクロックゲーティングを行うことに対応し、複数の制御信号のコストおよび追加回路のコストが加算されるので、最小コストパスを選ぶ段階で排除できる。

さらに、制約条件について、制御信号候補の最大削減の期待値を用いた簡単化を適用する。ある制約信号候補 c_j の 1 確率を p_j とし、それを使う可能性のあるフリップフロップの個数を k_j とすると、削減後のコストは $k_j \times p_j + 0.8$ となる。もしもこれが k_j よりも大きければ c_j が制御信号として選ばれることはなく、c_j に関する列を除いて制約条件および最小コスト問題を考えることができる。

5. 実現と実験

5.1 実現方法

提案手法は二分決定グラフのパッケージを用いて C で実現された。二分決定グラフのパッケージは、以下の 3 点の拡張を行った。第一は二分決定グラフで表された論理関数の 1 確率の計算方法の実現である。第二は、変数に 1 確率を割り当てることである。第三は、1 定数節点までの最小コストパスの計算である。

処理の流れを図 5 に示す。まず与えられた順序回路のデータに対して時間展開を行う。時間展開後の回路に対して、二分決定グラフを構成し、提案手法で各フリップフロップのクロックゲーティング制御信号候補を求める。同時に各候補の 1 確率も計算する。候補の情報から条件緩和および簡単化を行った後の制約条件式を生成する。制約式から二分決定グラフを構成し、1 定数節点への最小コストパスを計算する。最小コストパス上の変数への 1 割り当てから各フリップフロップの制御信号が得られる。

5.2 実験結果

提案手法は ISCAS 89 順序回路ベンチマークの内の 9 つの回路について適用された。実験は 2 時刻展開で行った。実験環境は Intel i5 2.20 GHz CPU で 1.5 GB 主記憶の計算機で Linux 2.6.27 を用いて行った。実行時間は s641 と s713 を除いて 1 CPU 秒以下であった。s641 と s713 では、制約条件を満たす最小コストパスの計算に 370 秒程度必要であった。

表 2 に、制御信号候補の抽出結果および最適化の結果を示す。各列は回路名、フリップフロップ数、制御信号候補数、最小コストを示す。表中の STCS (Single Time Step Circuit) は単一時刻だけを考えた回路を、TTSC (Two Time Step expanded Circuit) は 2 時刻分の時間展開回路を表す。

まず制御信号候補数について述べる。STSC では現時刻の信号のみが候補として得られるが、TTSC では現時刻だけでなく過去の時刻の信号も候補として得られるので、それらを分けて示している。表からわかるように、過去の信号がかなりの数得られている。また、TTSC における現時刻の候補の数は、STSC の候補数に比べて多くなっている。これは、2 時刻展開回路を考えることで、どこからも到達できない状態を除いて解析が行われ、結果としてより多くの信号が抽出できたと考えられる。

2 時刻展開の回路では、左側が過去の回路で、その状態変数は外部入力で任意の状態を含むが、右側の現在の回路では、状態変数は 1 時刻前の回路で計算されたものである。

つぎに、表 2 の最適な制御信号の選択結果の評価について述べる。STSC での最小コスト、TTSC での最小コ

ストが示されている。これらのコストには、クロックゲーティングの付加回路のコストも含まれている。s27 を除き、TTSC の方が良い結果となっている。

TTSC での最小コストの “+1.0” は過去の信号を現時刻へ移すためのフリップフロップのコストを示している。s208 や s444 で、過去の信号が使用が認められる。

なお、過去の信号をフリップフロップで現時刻に移した場合は、現時刻での論理的な処理がないので、半サイクル以内で制御信号が決定されるという条件を満たし、ラッチベースの制御回路を用いる必要がなく、付加回路のコストを今の 0.8 から 0.2 〜 0.3 に下げることができる。

提案手法では、過去の信号で良い結果が得られるだけでなく、s510 や s820 では現在の信号のみを用いて、現在の回路だけを用いた手法より良い結果が得られている。これは、到達可能でない状態を排除することで、信号の論理関数および 1 確率が変化したことによるものと考えられる。

6. おわりに

本稿では、回路の時間展開と EXOR ベースの判定条件を用いた前方的順序的クロックゲーティングの自動化手法について述べた。順序回路には数時刻分の時間展開が適用され、大きな組合せ回路としてクロックゲーティングの制御信号候補の検出が行われる。候補の検出は通常のフリップフロップの現在の値と新しい値の EXOR との AND で行うが、時間展開後の回路により、現時刻の回路だけでなく過去の値も制御信号の候補として検出される。これらの候補の中からクロックの印加確率に基づき最適な制御信号を各フリップフロップに対して選ぶ手法、および制約条件の簡単化手法を提案した。

回路の時間展開により、単に過去の信号を候補として検出できるだけでなく、他から到達できない状態を排除する効果により、各信号の論理関数および制御信号としてのクロック印加確率が変化し、現時刻の回路だけを用いた手法では検出できなかった現時刻の信号が検出できるようになった他、選択結果もより良いものとなった。

今回の提案手法は、二分決定グラフのパッケージを用いて、C 言語で実現され、ISCAS 89 ベンチマーク回路 9 個に適用し、有用性を示している。論理関数処理やクロックの印加確率計算に二分決定グラフを用いているため、適用回路規模に制限がある。しかし、時間展開で到達できない状態を排除する効果は、SAT を用いた論理関数処理やランダムパターンシミュレーションを用いた 1 確率の近似計算などでも有用である。今後は、これらの異なるベースでの手法の開発や、後方順序的クロックゲーティングの自動化についても研究を行う。

謝辞 早稲田大学戸川望教授、吉村猛教授、渡邊孝博教授、史又華准教授はじめ、戸川研究室、史研究室、柳澤研究室、木村研究室の皆様に対し、日頃からの御討論を感謝します。本研究は一部、早稲田大学特定課題研究費および NEC の委託研究費による。

表 2 クロックゲーティング制御信号候補および最適化結果

名前	F.F. 数	制御信号候補数			最小コスト	
		STSC 候補数	TTSC		STSC	TTSC
			現時刻候補数	過去の候補数		
s27	3	3	6	1	3.0	3.0
s208	8	79	138	159	8.0	6.6+1.0
s444	21	64	186	305	20.5	15.5+1.0
s510	6	195	209	145	6.0	5.8
s526	21	153	445	244	20.5	15.5+1.0
s641	19	110	121	89	19.0	15.8+1.0
s713	19	114	122	91	19.0	15.8+1.0
s820	5	52	53	0	5.0	4.3
s832	5	52	53	0	5.0	4.3

参考文献

[1] Pietro Babighian, Luca Benini, and Enrico Macii, “Scalable Algorithm for RTL Insertion of Gated Clocks Based on ODCs Computation,” *IEEE Trans. on CAD*, Vol. 24, No. 1, pp. 29–42, Jan. 2005.

[2] A. Hurst, “Automatic Synthesis of Clock Gating Logic with Controlled Netlist Petrurbation,” In *Proc. of DAC*, pp. 654–657, June 2008.

[3] Man Xin, Takashi Horiyama, and Shinji Kimura, “Power Optimization of Sequential Circuits Using Switching Activity Based Clock Gating,” *IEICE Trans. on Fundamentals*, Vol. 93, No. 12, pp. 2472–2480, Dec. 2010.

[4] L. Li, W. Wang, and K. Choi, “Automatic Register Transfer Level CAD Tool Design for Advanced Clock Gating and Low Power Schemes,” In *Proc. of International SoC Design Conference (ISOCC)*, pp. 21–24, Nov. 2012.

[5] Shih-Hung Weng, Yu-Min Kuo, and Shih-Chieh Chung, “Timing Optimization in Sequential Circuit by Exploiting Clock-Gating Logic,” *ACM TODAES*, Vol. 17, No. 2, pp. 1–15, April 2012.

[6] S. Ahuja and S. Shukla, “MCBCG: Model Checking Based Sequential Clock-Gating,” In *Proc. of IEEE International High Level Design Validation and Test Workshop*, pp. 20–25, Nov. 2009.

[7] H. Savoj, A. Mishchenko, and R. Brayton, “Sequential Equivalence Checking for Clock-Gated Circuits,” *IEEE Trans. on CAD*, Vol. 32, No. 2, pp. 305–317, Feb. 2014.

[8] Y. Y. Dai, K. Y. Khoo, and R. K. Brayton, “Sequential Equivalence Checking of Clock-gated Circuits,” In *Proc. of Design Automation Conference (DAC)*, pp. 1–6, June 2015.

[9] Ting-Hao Lin and Chung-Yang Huang, “Using SAT-based Craig Interpolation to Enlarge A Clock Gating Functions,” In *Proc. of DAC*, pp. 621–626, June 2011.

[10] Shin ichi Minato, *Binary Decision Diagrams and Applications for VLSI CAD*. Kluwer Academic Publishers, 1996.

[11] R. E. Bryant, “Graph-Based Algorithms for Boolean Function Manipulation,” *IEEE Trans. on Comput.*, Vol. C-35, No. 8, pp. 677–691, August 1986.