設 計 自 動 化 70－1
（1994．2．4）

インクリメンタルシミュレーションのゲートレベルへの適用

羽毛田 卓哉 † 新井 浩志 ‡ 深澤 良彰 †

†早稲田大学理工学部 ‡千葉工業大学

概要

シミュレーション結果の保存・再利用により、シミュレーション時間を短縮するアルゴリズムにインクリメンタルアルゴリズムがある。このアルゴリズムは機能シミュレーションを主な対象にしており、ゲートレベルへ適用した場合、保存するシミュレーション結果が膨大になるなどの問題がある。これらの問題に対処するためには、保存信号線の選択を行い、一部の信号線のみのシミュレーション結果を、ログ・イベントとして保存する必要がある。保存信号線選択において重要なことは、ログ・イベント量の削減と同時に、シミュレーション時間の増加をできるだけ抑えることである。本稿では、設計変更後の再シミュレーション時のイベント評価量削減の度合の期待値を評価関数として、保存信号線を選択するアルゴリズムを提案する。そして、保存信号線選択アルゴリズムとして本手法を用いた場合と、他のアルゴリズムを用いた場合での、シミュレーション時間を比較した結果について報告する。

An Application of Incremental Simulation to Gate Level Logic

Takuya Haketa † Hiroshi Arai ‡ Yoshiaki Fukazawa †

†School of Science and Engineering, Waseda University ‡Chiba Institute of Technology

Abstract

Incremental simulation algorithm is originaly developed to reduce the simulation time of the functional simulation by storing and re-using the simulation results. Appling this algorithm to gate-level logic simulation, simulation results which must be stored will be enormous. So, we need to use selective storage methods which select the signals and store the simulation results related only to those signals. The key to apply the selective storage method is how to balance the decrease of the number of the logged-events and the increase of simulation time. In this paper ,we present a new selective storage algolithm. Our algorithm is based on an expected value of the number of evaluations, reduced by storing each signal's simulation resulsts. In conclusion, we report the results of simulation times comparing this algolithm with other selective storage algolithms.

1 はじめに

ディジタルシステムの開発工程において、設計／検証サイクルが多くの時間を占める。設計者の生産性を高めるためには、この設計／検証サイクルを短縮することが重要になる。このサイクルを、シミュレーション結果の保存、再利用により短縮する１つの手法として、インクリメンタル機能シミュレーションアルゴリズム [1] が提案されている。

このアルゴリズムをゲートレベル論理シミュレーションに適用させることは、各ゲートを機能コンポーネントとみなすことにより可能である。しかし、これにより、回路内の信号線数が増加するため、保存しなければならないシミュレーション結果は膨大になる。そのために保存信号線の選択を行ない、一部の信号線のシミュレーション結果のみを保存する必要がある。保存信号線は、設計変更後のシミュレーションに備え、シミュレーション結果を保存する信号線である。本稿では、保存信号線の数を削減し、かつシミュレーション時間の増加を抑えることを目的とした保存信号線選択アルゴリズムを提案する。

以下、第２章で、インクリメンタルアルゴリズムの概要と保存信号線選択の必要性について述べる。また、３章で、本手法による保存信号線選択アルゴリズムについて、４章で、評価結果に基づき本手法の有効性を考察する。

2 インクリメンタルアルゴリズム

インクリメンタルアルゴリズムは、設計／検証サイクルにおいて、１度に修正される箇所はわずかである点に着目し、設計変更により影響を受ける信号線のみを再シミュレーションする手法である。このアルゴリズムの概念を図１に示す。

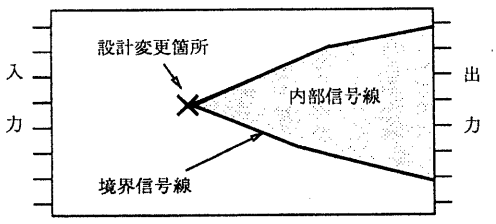

図 1: インクリメンタルアルゴリズムの概念

初回のシミュレーションでは、全ての信号線のシミュレーション結果をログ・イベントとして保存しておく。ログ・イベントは、シミュレーション中に各信号線で生じたイベントである。設計者により回路が変更されると、設計変更箇所の検出を行ない、接続関係に従って影響を受ける信号線を検出する。この信号線を内部信号線と呼ぶ。また、影響を受ける信号線と受けない信号線の境界となる信号線を検出する。この信号線を境界信号線と呼ぶ。再シミュレーションする際には、境界信号線のログ・イベントを、内部信号線への入力パターンとしてスケジューリングし、イベント駆動アルゴリズムを用いて内部信号線のみ再シミュレーションを行なう。この際、次回のシミュレーションに備え、各信号線のシミュレーション結果を、ログ・イベントとして全て保存する。

インクリメンタルアルゴリズムをゲートレベルに適用し、全信号線のシミュレーション結果を保存した場合、ログ・イベント量が膨大になるため、ディスクスペースを消費してしまうとともに、シミュレーション時の、ログ・イベントのメンテナンスによるオーバーヘッドが大きくなる。この問題を解決するためには、保存信号線の選択を行い、一部の信号線のシミュレーション結果のみを保存することにより、ログ・イベント量を削減する必要がある。

保存信号線の選択を行なった場合のインクリメンタルシミュレーションの手順は次のようになる。

1. 保存信号線を選択する
2. ある設計変更に対して、以下を繰り返す
   (a) 設計変更箇所を検出する
   (b) 内部信号線と、境界信号線を検出する
   (c) 境界信号線から入力側に向かって信号線をたどり、保存信号線を検出する
   (d) 検出された保存信号線のログ・イベントをスケジューリングし、シミュレーションを行なう
   (e) 選択されている保存信号線のシミュレーション結果をログ・イベントとして保存する

保存信号線の選択アルゴリズムとしては既に、２つの選択アルゴリズム [2] が提案されている。

1. レベル付けによる手法

   回路の各信号線にレベル付けを行ない、一定のレベル毎に信号線を保存信号線として選択する

2. クラスタ分割による手法

   回路を、相互接続の多いゲートのまとまりであるクラスタに分割し、そのクラスタの入力信号線を保存信号線として選択する

これらの手法は、比較的高速に保存信号線の選択ができるという利点を持つ反面、その信号線のシミュレーション結果を保存すると、設計変更後の再シミュレーション時にどれだけ有用であるか、いいかえれば、どれだけのイベント評価量を削減できるか、ということまでは考慮していない。特に、保存信号線数をより少なくする場合には、その信号線のシミュレーション結果を保存することによる有用性を考慮することは非常に重要であり、再シミュレーション時間を大きく左右する。次章では、これらの問題点を改善するための保存信号線選択手法について述べる。また、これら2つの手法のより詳細な問題点を、評価結果に基づき4章で考察する。

3 保存信号線選択手法

3.1 保存信号線選択時のイベント評価

我々は、保存信号線選択の効果を調べるにあたって、保存信号線の選択を行なった場合のインクリメンタルシミュレーションの概念を図2のように定義した。ここで、各信号線の意味は次の通りとする。

- 内部信号線： 設計変更により影響を受ける信号線
- 境界信号線： 設計変更により影響を受ける部分と受けない部分の境界となり、かつ保存信号線でない信号線
- 保存境界信号線：内部信号線をシミュレーションするために必要な、保存信号線
- 中間信号線：境界信号線と保存境界信号線の間にある信号線で、かつ保存信号線でない信号線
- 後方信号線：中間信号線で、かつその信号線の論理値が設計変更箇所に影響を与える信号線
- 外部信号線：再シミュレーションに関与しない信号線

ここで、図2の内部信号線を再シミュレーションすることを考える。図2では、全信号線が保存信号線である図1と異なり、一部の信号線のみ保存信号線となるため、境界信号線、保存境界信号線、中間信号線が存在する。中間信号線は、設計変更の影響は受ないが、内部信号線を再シミュレーションするためのログ・イベントを供給できないため、再シミュレーションされる。従って、保存境界信号線のログ・イベントをスケジューリングすることにより、内部信号線、中間信号線、境界信号線を再シミュレーションすること

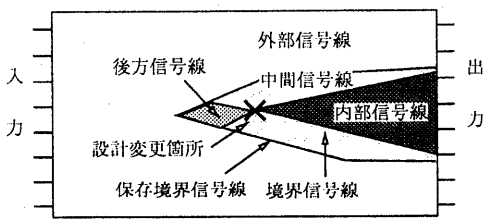

図 2: 保存信号線選択を行なった場合の概念

になる。ここで、保存信号線の数とシミュレーション時間には、

- 保存信号線の数が少ない→中間信号線が増加しシミュレーション時間は遅くなる
- 保存信号線の数が多い→中間信号線が減少しシミュレーション時間は速くなる

という関係があると考えられる。一方、保存しておくと再利用される可能性が高くシミュレーション時間を適宜短縮できる有用な信号線と、そうでない信号線が存在する。次章では、これらのことを考慮した保存信号線選択アルゴリズムについて述べる。

3.2 評価関数を用いた保存信号線選択

本手法の目的は、保存信号線の数を減らし、かつどんな箇所の設計変更においても、一様にシミュレーション時間の増加を抑えることである。

そのために、ある信号線を保存信号線として選択した場合、再シミュレーション時にどのくらいのイベント評価量を削減できるかを判断し、その期待値を評価関数として保存信号線を選択する。この評価関数の決定にあたって、その信号線のシミュレーション結果を必要とするような設計変更箇所を全て考慮する。そして、その信号線のシミュレーション結果を保存しない場合の再シミュレーション時の評価量の期待値と、保存した場合の期待値との差に従って決定する。この評価関数の決定には、以下の仮定をする。

- 設計変更が起きる確率は全ての信号線で等しく、設計変更は1箇所のゲート種の変更である
- 発生するイベント量は、全ての信号線で等しい
- 上の仮定に基づき、イベント評価量はシミュレーションされる信号線数である

このような仮定のもとに、イベント評価量の期待値は、全ての箇所での設計変更の可能性についての、設

計変更の起きる確率と、その箇所で設計変更が発生した場合再シミュレーション時に評価しなければならないイベント量との積の総和とした。この評価関数の値が大きいほど、再シミュレーション時のイベント評価量に差がでるため、有用な保存信号線になる。

このために、ある信号線 a のシミュレーション結果が必要になるような、設計変更の箇所を全て考える。ある回路で a に関連のある信号線は、それぞれ図3のような集合に分けることができる。

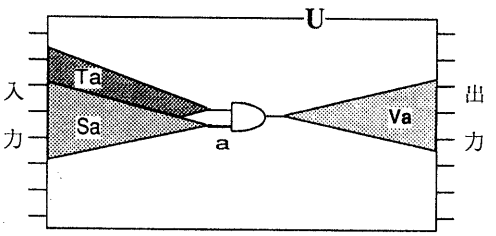

図 3: 信号線 a に関連する信号線の集合 (1)

各信号線の集合の意味は、以下の通りである。

- U 集合：U

  回路に含まれる信号線全体の集合

- S 集合：S_a

  信号線 a に対して、設計変更の影響を与える信号線の集合

- T 集合：T_a

  信号線 a を入力としているゲートの a 以外の全ての入力信号線の S 集合の和集合から、S_a に含まれる信号線を除いた信号線の集合

- V 集合：V_a

  信号線 a が設計変更された場合に、その影響を受ける信号線の集合

さらに、a に関連する信号線の集合として次の集合を定義し、図3の集合との関連をふまえて図4に示す。

- F 集合：F_a

  V_a の中で、信号線 a から出力側にたどった時に、保存信号線までの経路上に存在する信号線の集合 (その経路上の信号線は保存信号線ではない)

- B 集合：B_a

  S_a の中で、信号線 a から入力側にたどった時に、保存信号線までの経路上に存在する信号線と a 自身を含めた信号線の集合 (その経路上の信号線は保存信号線ではない) また、回路全体の入力信号線は含めない

- W 集合：W_a

  F_a に含まれる全ての信号線の T 集合の和集合のから、F_a, a, B_a, T_a を除いたもの (a, B_a は T 集合の和集合の中には、最初から当然含まれていないはずである)

- X 集合: X_a

  F_a に含まれる全ての信号線の B 集合の和集合 (評価関数に直接は関係しない)

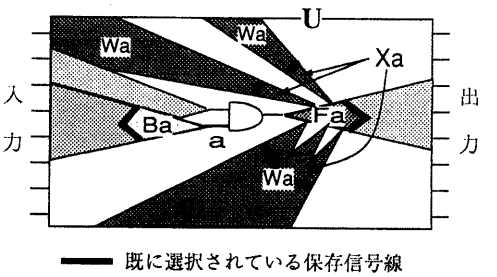

図 4: 信号線 a に関連する信号線の集合 (2)

ここで、設計変更が T_a, F_a, W_a で発生した場合には、信号線 a のシミュレーション結果を保存してあるか否かで再シミュレーション時の評価量が変化する。その他の集合では、a 自身が影響を受けるか、または、a のログ・イベントを必要としないため評価量は変化しない。そこで、以下では、T_a, F_a, W_a での設計変更における評価量の期待値を考える。これら3つの集合で設計変更が起きた場合、信号線 a が保存信号線であるかどうかで表1のような信号線になることがわかる。

表 1: 設計変更箇所に応じた分類

	保存信号線の場合	保存信号線でない場合
Ｔａで設計変更	保存境界信号線	境界信号線
Ｆａで設計変更	保存境界信号線	後方信号線
Ｗａで設計変更	保存境界信号線	中間信号線

3.3 保存信号線選択のための評価関数

次に、実際に評価関数を求める。先に述べた通り、評価関数は、評価関数値を求めるある信号線 a に関して、a のシミュレーション結果が保存されていない場合の、再シミュレーション時のイベント評価量の期待

値と、保存した場合のイベント評価量の期待値の差である。この期待値の差を考えると、まず、T_a で設計変更が起きた場合の評価量は、

- a のシミュレーション結果が保存されていない場合→ T_a で必要な評価信号線数 $+|V_a|+|X_a|+|B_a|$
- a のシミュレーション結果が保存されている場合→ T_a で必要な評価信号線数 $+|V_a|+|X_a|$

であり、a のシミュレーション結果が保存されている場合といない場合の評価量の差は $|B_a|$ だけになる。ただし、上記の2つの場合それぞれに、さらに' V_a に含まれる信号線から F_a に含まれる信号線を取り除いた残りの信号線の B 集合の和集合の要素数' が加わるが、論旨を簡潔にするために省略している。また、$|B_a|$ は、B_a に含まれる信号線数を表す。同じように考えると、F_a, W_a における設計変更においても、評価量の差は $|B_a|$ だけになる。従って、a のシミュレーション結果が保存されている場合の評価量を基準として0にすると、各集合での設計変更の確率と、その時のある信号線 a のシミュレーション結果が保存されていない場合の相対的な評価量は、表2のようになる。

表 2: 設計変更箇所と相対評価量

変更箇所	確率	保存されていない場合の相対評価量
T_a	$\frac{\lvert T_a\rvert}{\lvert U\rvert}$	$\lvert B_a\rvert$
F_a	$\frac{\lvert F_a\rvert}{\lvert U\rvert}$	$\lvert B_a\rvert$
W_a	$\frac{\lvert W_a\rvert}{\lvert U\rvert}$	$\lvert B_a\rvert$

従って、評価関数 h_a は以下の様に求めることができる。

$$h_a = \underbrace{(\frac{|T_a|}{|U|}\times|B_a| + \frac{|F_a|}{|U|}\times|B_a| + \frac{|W_a|}{|U|}\times|B_a|)}_{(1)} - \underbrace{0}_{(2)}$$

$$= \frac{|B_a|(|T_a|+|F_a|+|W_a|)}{|U|} \quad ---\ (3)$$

$$= \frac{|B_a|(|T_a+F_a+W_a|)}{|U|} \quad ---\ (4)$$

$$= \frac{|B_a|(|T_a+F_a+\bigcup_{i\in F_a}T_i|)}{|U|} \quad ---\ (5)$$

上式において、(1) は a のシミュレーション結果が保存されていない場合の再シミュレーション時のイベント評価量の期待値を表し、(2) は保存されている場合の評価量の期待値を表す。そして、評価量削減の度合の期待値として (3) が導出される。また、(3) から (4) への変換は、各集合が同じ要素を含まないために変換が可能である。最後に、(4) から (5) への変換で、W 集合を T 集合の和集合に変換した。しかし、そのために、変換された和集合に、T_a の要素が含まれる可能性があるため変換された和集合が、W 集合の定義に反する可能性がある。しかし、結局、T_a と変換した和集合との論理和をとるので、変換された集合の中に T_a の要素が含まれたとしても評価関数値は結果的に変わらない。これら3つのどの式を用いて評価関数値を求めてもよい。次に、本手法における保存信号線の選択手順を示し、導出された評価関数を用いた保存信号線の選択例、その時の計算過程を示す。

3.4 保存信号線の選択手順と選択例

本手法では、3.3節で求めた評価関数値が大きい順に保存信号線を選択する。しかし、1つの信号線を選択すると、選択された信号線の、F 集合の要素の信号線の B 集合が変化し、B 集合の要素の信号線の F 集合が変化する。変化した信号線について、評価関数値を求め直す必要がある。ただし、どの信号線が選択されても各信号線の W 集合は変化しない。従って、本手法において評価関数式として (3) 式を用いた場合の保存信号線の選択手順は次のようになる。

1. 各信号線の T, F, B, W 集合を求める
2. 各信号線の評価関数値を求める
3. 保存信号線の選択数が基準に達するまで以下を繰り返す
   (a) 評価関数値が最も大きい信号線を選択
   (b) F 集合、B 集合が変化する信号線の評価関数値を求め直す

次に、ISCAS85 のベンチマーク回路の1つである c17 回路 (図5参照) を用いて、1本の信号線を選択する場合の選択例と計算過程を示す。ただし、評価関数には、(3) 式を用いる。

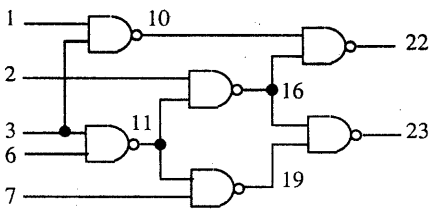

図 5: 回路 -c17

1. 各信号線の T, F, B, W 集合を求める

(a) T 集合

$$T_{10} = \{2,6,11,16\} \quad T_{11} = \{2,7\}$$
$$T_{16} = \{1,7,10,19\} \quad T_{19} = \{2,16\}$$

(b) F 集合、B 集合

$$F_{10} = \{22\} \qquad B_{10} = \{10\}$$
$$F_{11} = \{16,19,22,23\} \quad B_{11} = \{11\}$$
$$F_{16} = \{22,23\} \qquad B_{16} = \{11,16\}$$
$$F_{19} = \{23\} \qquad B_{19} = \{11,19\}$$

(c) W 集合

$$W_{10} = \phi$$
$$W_{11} = \{1,10\}$$
$$W_{16} = \phi$$
$$W_{19} = \phi$$

2. 各信号線の評価関数の値を求める

(a) 評価関数値の計算

$$h_{10} = \frac{|B_{10}|(|T_{10}+F_{10}+W_{10}|)}{|U|} = \frac{1(|\{2,6,11,16,22\}|)}{11} = \frac{5}{11}$$
$$h_{11} = \frac{1(|\{1,2,7,10,16,19,22,23\}|)}{11} = \frac{8}{11}$$
$$h_{16} = \frac{2(|\{1,7,10,19,22,23\}|)}{11} = \frac{12}{11}$$
$$h_{19} = \frac{2(|\{2,16,23\}|)}{11} = \frac{6}{11}$$

(b) 評価関数値が最も大きな信号線を選択
信号線 16 を選択する

4 評価

本手法の評価システムの構成を図6に示す。本システムは、大きく分けて、保存信号線選択プログラムとインクリメンタルシミュレータから構成される。まず、シミュレーション対象回路の初期ネットリストを、保存信号線選択プログラムに入力し、どの信号線のシミュレーション結果を保存するかを決定する。1回目のシミュレーション時には、通常のシミュレーションを行ない、保存信号線として選択された信号線のシミュレーション結果を保存する。その後のシミュレーション時には、設計変更前のネットリストと変更後のネットリストを比較し、設計変更箇所の検出を行ない、変更情報を出力する。次に変更情報をインクリメンタルシミュレータに与え、内部信号線の検出、保存境界信号線の検出を行なう。そして保存境界信号線からログ・イベントを初期イベントとしてスケジューリングを行ない、シミュレーションを行う。その際、保存信号線として選択されている信号線のシミュレーション結果を保持しておき、シミュレーション終了時に、ログ・イベントとしてファイルに保存し、設計変更後のシミュレーション時に再利用する。

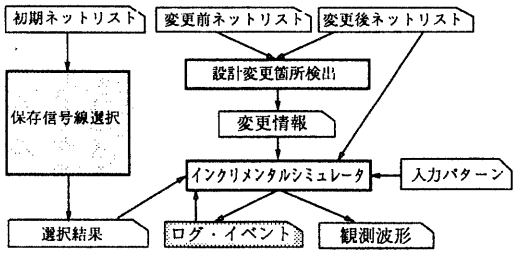

図 6: システム構成

評価回路には、ISCAS85 のベンチマーク回路を用い、また、評価は全て SUN SPARCstation2 上で行った。

このシステムを用いて、まず、保存信号線選択部分に、本手法による保存信号線選択アルゴリズム、及び、レベル付け、クラスタ分割による手法を用いた場合の、シミュレーション時間を示す。各手法の保存信号線数を回路の全信号線数の約 20% でほぼ同様にする、入力パターンは、約 6000 タイムフレームのランダムなパターン、シミュレーション時間は、任意の 2-3 箇所の設計変更箇所における平均実 CPU 時間とする。表 3 に、各手法を用いた場合のシミュレーション時間と保存信号線選択時間を示す。

表 3: シミュレーション時間／選択時間 (sec)

回路	信号線数	本手法	レベリング	クラスタ分割
c880	443	24.4/6.2	30.3/0.4	32.5/9.2
c3540	1719	39.0/52.1	56.0/1.1	63.7/73.0
c5315	2485	19.2/74.4	22.6/1.7	24.5/121.0
c7554	3720	49.5/98.6	71.2/1.9	90.3/190.2

表3より、どの回路においても本手法の方がシミュ

レーション時間が短く、効率的に保存信号線が選択されていることがわかる。また、回路規模が大きくなるほど、シミュレーション時間の差が広がり、より、本手法が有効になることが分かった。ただし、c5315 については、選択手法による差はあまりでなかった。これは、入力パターンが良くなかったために回路全体のイベント発生率が低かった、または回路自体のイベント発生率が低かったためと考えられる。 ここで、他の2つのアルゴリズムにおける問題点を考える。まず、レベル付けの手法であるが、レベル毎に選択したとしても、本手法でいう各信号線の F, B 集合の数は各信号線で一定でない。従って、必ずしも一様にシミュレーション時間を短縮できるとは限らない。また、クラスタ分割による手法は、確かに接続関係の密接な信号線、ゲートをクラスタにすることにより、クラスタ内の設計変更において、そのクラスタの入力の保存信号線は有効に利用される。しかし、クラスタ間の接続関係までは考慮していない。これら2つの手法はともに、その信号線のシミュレーション結果を保存することによりどれだけ有用であるかということは考慮していない。

一方、選択時間に着目すると、レベル付けによる選択が最も速く、本手法においては、回路規模が大きくなるにつれ選択時間がかなり増加しており、改善の余地があるといえる。これは、評価関数値の計算における集合演算に時間がかかるためと考えられる。この問題を改善するためには、プログラムの高速化、また、ある程度精度を落しても、高速に計算できる評価関数にすることが必要になる。しかし、保存信号線は各回路につき1回しか行なわれない。従って選択時間が長いことが特別な問題にはならない。

次に、選択信号線数の割合に対するシミュレーション時間の評価を行なった。評価回路に c3540 を、また保存信号線選択アルゴリズムとして本手法とレベル付けによる手法を用いている。回路レベルにおける中央付近の任意の設計変更箇所を1箇所選び再シミュレーションした。すなわち、設計変更箇所は同一にし、保存信号線数により再シミュレーション時間に差がでるかを2つの手法を用いて検討している。評価結果を表4に、グラフ化したものを図7に示す。ただし、選択時間は本手法におけるものである。

表 4: 選択割合と選択時間／シミュレーション時間

選択数 (%)	0%	10%	30%	50%	80%	100%
選択時間 (sec)	0.0	43.4	57.3	64.5	74.2	0.0
本手法 (sec)	59.8	53.6	41.0	46.1	60.4	71.0
レベル付け (sec)	59.8	79.2	63.0	47.0	61.2	71.0

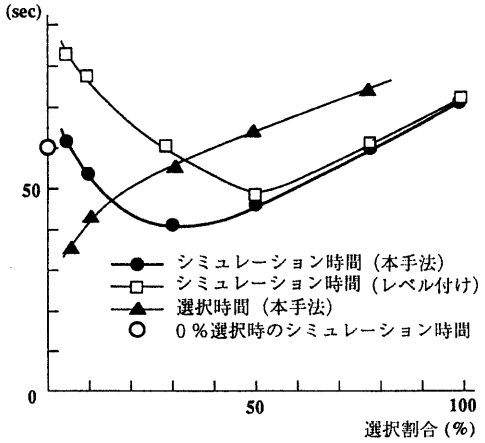

図 7: 選択割合と選択時間／シミュレーション時間

必ずしも全信号線のシミュレーション結果を保存すれば、再シミュレーション時間が短縮されるわけではないことが分かった。これは、シミュレーションの準備段階において、保存されている膨大なシミュレーション結果を読み込む、また、シミュレーション中に随時シミュレーション結果をメモリに記憶する、また膨大なシミュレーション結果をディスク上に吐き出すなどの操作に時間を費やすためと考えられる。簡潔にいえば、シミュレーション結果のメンテナンス、保存によるオーバーヘッドである。

また、図7には示さないが、設計変更箇所により、図7のシミュレーション時間の曲線の頂点が上下に移動する。設計変更箇所により、シミュレーション時間が変化することは、既に論文 [1] で示されている。従って、0% 選択時のシミュレーション時間とこの曲線を単純に比較することはできない。しかし、回路中央付近の設計変更箇所であれば、平均的な曲線になると考えられる。従って、一般的にいえることは、0% 選択の時のいわゆる慣例的なイベント駆動シミュレーションよりも、数 % の選択で、逆にシミュレーション時間が増加するのは、設計変更箇所検出、保存境界信号線検出などのシミュレーションの準備段階の操作によるものである。

また、保存信号線数を増やすと選択時間も増加してしまう。従って、本手法の場合、回路の 20-40% くらいで、最もシミュレーション時間を短縮でき、かつ選択時間も抑えられる。

また、本手法とレベル付けによる手法で、シミュレーション時間を比較すると、本手法において、最も速くシミュレーションが終了したのは、約 30% の信

号線を選択した時で、一方、レベル付けによる手法では、約50%であったが、本手法において50%選択した場合のシミュレーション時間とほぼ同じあった。そして、保存信号線数の割合にかかわらず、常に本手法の方がシミュレーション時間が短く、本手法の有用性が示された。

最後に、本手法とレベル付けによる手法を用いた場合の、設計変更箇所によるシミュレーション時間の比較を示す。設計変更箇所は、それぞれ、変更箇所1が回路の入力より、変更箇所2が回路レベルにおける中央付近、変更箇所3が出力よりの設計変更箇所である。また、評価回路にはc3540を用い、保存信号線数は回路の全信号線数の約20%とした。評価結果を表5に示す。

表5: 設計変更箇所とシミュレーション時間 (sec)

設計変更箇所	0% 選択時	本手法	レベル付け
変更箇所1(入)	59.8	63.1	89.7
変更箇所2(中)	59.8	44.6	64.7
変更箇所3(出)	59.8	29.0	43.2

表5より、2つの手法について共通にいえることは、設計変更箇所が回路の入力よりで、より設計変更の影響が広がる場合、シミュレーション時間は長くなり、シミュレーション結果の保存を行なわない0%選択時に相当するイベント駆動シミュレータを使用した場合のシミュレーション時間とほとんど変わらないことである。また、設計変更箇所が出力よりで、あまり設計変更の影響が広がらない場合、シミュレーション時間は短縮されている。これは、論文[1]に示された通りである。しかし、2つの手法の比較を行なうと、どの設計変更箇所においても本手法の方がシミュレーション時間は短かった。従って、本手法の目的どおり、どの箇所の設計変更においても、一様にシミュレーション時間の増加を抑えることができた。

5 おわりに

本稿では、インクリメンタルアルゴリズムをゲートレベルへ適用する上で、ログ・イベント量を削減し、かつシミュレーション時間の増加を抑えるための保存信号線選択アルゴリズムを提案した。評価で示されたように、本手法は、多数の回路でシミュレーション時間の増加を抑えることができた。また、必ずしも全信号線のシミュレーション結果を保存することがインクリメンタルシミュレーションの高速化にはつながらず、保存信号線数をある程度少なくし、かつ有用な保存信号線選択アルゴリズムを用いることにより、高速化できることが分かった。ゲートレベルシミュレーションへ本手法を適用することにより、保存信号線数の削減と同時にシミュレーション時間を増加を抑える、さらにインクリメンタルシミュレーションの高速化が可能になる。

一方、本手法の問題点は、回路規模が大きくなるにつれて、選択時間が長くなることである。今後の課題として、評価関数を高速に計算できるものにする、プログラムの高速化などが挙げられる。しかし、本手法は、保存信号線選択が各回路で1回行われるのに対し、設計／検証は繰り返し行われることを前提としている。また、1回の選択時間が長くなっても、設計／検証作業の中でシミュレーション時間ができるだけ短縮された方が、生産性の向上につながると考えている。従って、本システムが有効であるかどうかは、選択時間のオーバーヘッドに対し、どれだけ少ないログ・イベント量で、どれだけシミュレーション時間が短縮され、どれだけ設計／検証が繰り返さたかのトレード・オフになるといえる。

また、現在、保存信号線選択を用いたインクリメンタルアルゴリズムとデマンド駆動シミュレーションアルゴリズム[4]を併用することの有用性を検討しており、今後本システムに取り入れる予定である。

参考文献

[1] Sun Young Hwang, Tom Blank, Kiyoung Choi. *Incremental Functional Simulation of Digital Circuits.* Stanford University. IEEE Design Automation Conference(1987)

[2] Kiyoung Choi, Sun Young Hwang, Tom Blank. *Fast Functional Simulation: An Incremental Approach.* IEEE Trans On Computer-Aided Design(1988)

[3] Kiyoung Choi, Sun Young Hwang,and Tom Blank. *Incremental-in-Time Algorithm for Digital Simulation.* Stanford University. IEEE Design Automation Conference(1988)

[4] Steven.P.Smith, M.Ray Mercer, Bishop Brock. *Demand Driven Simulation: BACKSIM.* IEEE Design Automation Conference(1987)

[5] K.Subramanian, M.R.Zargham. *Distributed and Parallel Demand Driven Logic Simulation.* IEEE Design Automation Conference(1990)