設計自動化　45—7
(1988. 12. 16)

# 高速化イベント駆動方式によるＲＴＬ機能シミュレーション

# High-speed Event Driven RTL Simulator

水野 雅信、宮阪 修二、井川 智、宮崎 守弘、五十嵐 祥晃
Masanobu MIZUNO, Syuji MIYASAKA, Satoshi IKAWA,
Morihiro MIYAZAKI, Yoshiaki IGARASHI

松下電器産業㈱ 開発研究所
Development Research Laboratory
Matsushita Electric Industrial Co., Ltd.

あらまし　機能記述におけるイベント駆動シミュレーションにおいて、イベント・ルックアヘッド方式により高速化を図った。従来の機能レベルのイベント駆動シミュレーション方式では、評価モデル（レジスタ転送記述を評価する単位）の評価結果に影響を及ぼさない不要イベントが多い。本方式は、①対象とする評価モデルを評価する前に、その評価モデルが評価される条件を”先読み”し、②その条件を基に機能記述文から不要なイベントの予測を行い、③不要イベントを削除するものである。
現在、あるサンプル評価結果では、従来のイベント駆動方式と比較して6倍程度の処理速度が得られている。

Abstract- A new "event look ahead algorithm" for high-speed register transfer level(RTL) simulation is described. It's based on the event-driven algorithm.
In RTL simulation, conventional event-driven simulators evaluate many functional models using events that don't affect the behavior of the logic circuit.
This algorithm is as fallows : looks ahead the evaluation conditions of a functional model before evaluating the model, forecasts those meaningless events from the conditions and the RTL notations, and suppresses those events.
This RTL simulator processed about 6 times faster than a conventional event-driven simulator in an example logic circuit.

## １．はじめに

本報告はLSI設計における論理設計より上位のシステム設計において、アーキテクチャを設計する機能設計のためのシミュレーションに関するものである。ここではレジスタ転送レベル（以下、RTLと呼ぶ）を対象としている。

これまで、RTLの機能記述言語、及びシミュレータに関する研究結果が、数多く報告されている［１－７］。RTL記述言語としては、並列転送とそのタイミング制御記述が基本の記述形式となってきている。一方、機能シミュレータとしては、cyclic_simulation方式［１］やコンパイル方式［５］、イベント駆動方式［２、３、７］などが報告されている。

しかしながら、シミュレータについて論じると、これらの方式は、遅延（立ち上がり、伝播等の遅延）の評価能力、データフローを解析できるデバッグ能力、高速処理能力の点で評価した場合、いくつかの問題がある。cyclic_simulation方式やコンパイル方式は同期回路を対象としており、非同期回路を扱えない。また、イベント駆動方式は処理速度の点で前述の方式に劣っている。

基本方式の選択にあたっては、個々の動作遅延を考慮したシミュレーションが可能であり、トレース等のデバック機能を容易に実現できる点で、イベント駆動方式を採用した。しかし、先にも述べたように、イベント駆動方式による場合、コンパイル方式と比較して、高速処理能力が課題となる。

本報告では、この課題をイベント・ルックア

ヘッドという方式で解決したので、その高速化法を述べる。

本稿では、第2章においてシミュレーションの入力となるRTL機能記述言語を規定し、第3章でイベント駆動方式によるRTLシミュレータについて述べる。第4章において本報告で提案するイベント・ルックアヘッド方式について詳述し、第5章においてこれを適用したシミュレータの性能評価結果について述べる。

なお、本報告の機能シミュレータは、混合シミュレータMULS[6]のサブシステムとして開発したものである。

## 2. RTL機能記述言語

### 2.1 言語クラス

機能シミュレーションを検討する場合、入力言語の記述能力に応じて処理速度の評価が大きく異なる。ここでは、本報告が対象とする言語のクラスを明らかにしておく。

本報告が対象とする言語は、イベント駆動、あるいはデータフローの動作モデルに基づき、機能を記述するクラスに属し、ハードウェアに明確に対応付けられ、同期／非同期／多相クロックの各種アーキテクチャを記述できる。

具体的には主に、以下の様な記述を可能とする言語である。

①資源（レジスタ／ラッチ／信号線）間の並列転送記述
②タイミング記述（立上り遅延、転送遅延、イベントエッジ）
③制御機能の明記（if-then/else構文、オートマトン等）

現在、このような記述能力を持つクラスに属する言語としてHSL-FX[2]の機能記述、VHDL[3]のデータフロー記述等、多数の言語が提案されている。

### 2.2 記述例

このクラスに属する言語による回路記述の例を図1に示す。図1の言語は、C-likeな文法と、上記の①、②、③の記述能力を持つ言語である。記述されている回路は画像処理において用いるランレングス符号のデコーダである。この回路は、"STAT"という状態変数で制御される順序回路である。図1の各転送記述文において、"at"に続く式はクロック信号を表している。

図1には、記述中の各転送記述文に対してその行の最後に文番号を付けている。

```
/**********************
   RUN LENGTH DECODER
  **********************/
MODULE_SECTION              /* 入力資源の宣言 */
     NAME  : RL_DECODER;
     INPUT : SEL, RUNIN [7 : 0] ;
     OUTPUT: RUNOUT [7 : 0] ;
     CLOCK : CLK;
     RESET : RST;
END_SECTION

DEFINE_SECTION              /* 内部資源の宣言 */      文番号

     REGISTER : REG1  [7 : 0], REG2 [7 : 0],
                COUNTER  [7 : 0] ;
     AUTOMATON: STAT, CLK, RST > s0;                  … 1
END_SECTION

BEHAVIOR_SECTION            /* RTL 機能記述 */
     at (CLK) REG1=IN(RUNIN);                         … 2
     if (SEL) RUNOUT=OUT(REG2);                       … 3
     else RUNOUT=OUT(REG1);                           … 4

switch (STAT)  {            /* オートマトン制御 */
case s0: at (CLK) COUNTER=IN(0b0::RUNIN  [6 : 0] ;   … 5
         if (RUNIN [7] ) STAT=s1;                     … 6
         else  { at (CLK) REG2=IN(RUNIN);             … 7
                 STAT=s0;  }                          … 8
         break;
case s1: switch (COUNTER)  {
         case 0: STAT=s3;                             … 9
                 break;
         case 2: at (CLK) REG2=IN(RUNIN);             … 10
                 STAT=s0;                             … 11
                 break;
        default: at (CLK) COUNTER=COUNTER-1;          … 12
                 STAT=s2;                             … 13
                 break; }
         break;
case s2: if (COUNTER==2)  { at (CLK) REG2=IN(RUNIN);  … 14
                            STAT=s0; }                … 15
         else at (CLK) COUNTER=COUNTER-1;             … 16
         break;
case s3: STAT=s3;                                     … 17
         break;
}
END_SECTION
```

図1. RTL機能記述例

## 3. イベント駆動方式機能シミュレータ

### 3.1 システム構成

本シミュレータは信号値の変化を求める評価の単位となる単位モデル（論理シミュレーションのゲートに相当し、RTL機能記述では各転送記述文に対応する）とこのモデル間のイベントの伝搬を示すネットリストに基づいてシミュレーションを行っている。システム構成を図2に示す。

(1) HDL コンパイラ

RTLの機能記述言語による機能記述（以下、RTL機能記述と呼ぶ）を構文解析し、シミュレーションデータとして上記の単位モデルを複数で構成した単位モデルリスト、ネットリストをはじめ、RTL機能記述中で使用される資源に関する資源情報（名前、レジスタ、バスなどのタイプ、ビット幅、立上がり遅延値など）を抽出する。

(2) シミュレーションプログラム

上記HDL コンパイラで抽出したデータを入力としてイベント駆動方式シミュレーションを行う。単位モデルの評価には組み込み関数のライブラリが呼び出される。

(3) 表示部

シミュレーション結果の表示をおこなう。

次にRTL 機能記述から抽出される単位モデルとネットリストについて述べる。

### 3.2 単位モデル

単位モデルの構成はこれを識別するためのＩＤ番号と、各転送記述文に対応した以下の各構成要素から成っている。

**[単位モデルの構成]**

＜タイミング式＞＜条件式＞
＜転送先資源＞＜転送式＞

＜タイミング式＞
レジスタなどへの信号転送時のクロック信号の論理式を表す。

＜条件式＞
enable信号やセレクト条件を表す論理式である。

＜転送先資源＞
信号値が転送される先の資源を表す。

＜転送式＞
転送される信号値を決定する信号名、定数と演算子からなる算術式あるいは論理式である。

また、以後、説明の簡略化のため、＜タイミング式＞、＜条件式＞、＜転送式＞に現れる資源を単位モデルに対するfanin と呼び、また＜転送先資源＞に現れる資源を単位モデルに対するfanoutと呼ぶことにする。

具体的には、転送記述文と各構成要素の対応関係は、次の転送記述文を例にとると以下のようになる。

```
at (!CLK) if ((SEL1)&(!SEL2))
                REG0 = REG1+ TERM2;
```

このとき、＜タイミング式＞は (!CLK) であり、＜条件式＞は ((SEL1)&(!SEL2)) となる。＜転送先資源＞は REG0 、＜転送式＞はREG1+TERM2 である。また、fanin はSEL1、SEL2、CLK 、REG1、TERM2 となり、fanoutはREG0となる

### 3.3 ネットリスト

RTL 機能シミュレーションにおけるネットリストは、fanin の資源と単位モデルとの間の関係を示すデータである。それは、複数のネット名とこの各々に対応するリストから構成される。

(1) ネット名

RTL 機能記述で用いられる各資源と1対1に対応する。

(2) リスト

単位モデルリスト中で、ネット名と同じ資源名が現れる単位モデルのＩＤ番号の並びである。

例として、図1のRTL 機能記述例から抽出されるネットリストを図3に示す。ここでは、ネ

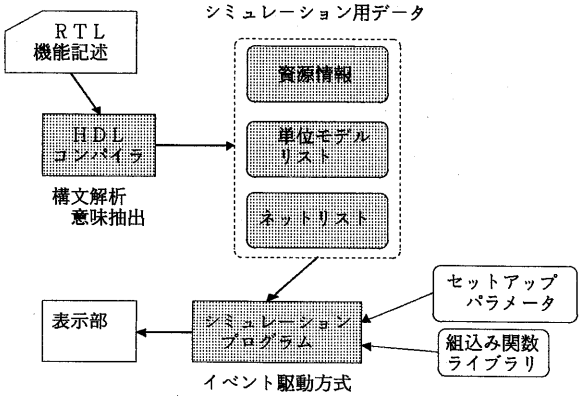


図2．システム構成

| ネット名（資源） | リスト（単位モデルのID番号） |
|---|---|
| SEL | 3，4 |
| RUNIN | 2，5，6，7，8，9，14 |
| RUNOUT | |
| CLK | 2，5，6，7，8，9，10，11，12，13，14，15，16，17 |
| RST | 1 |
| STAT | 5，6，7，8，9，10，11，12，13，14，15，16，17 |

（注　ID番号は、図1の文番号に対応する）

図3．ネットリスト例

ット名は対応する資源の名前を用いて示している。また、ＩＤ番号は図１の文番号である。例えば、ＳＥＬというネット名（資源）に対応するリストは、図１の文番号３、４に対応するそれぞれの単位モデルを意味するものである。

## ４．イベント・ルックアヘッド方式

### 4.1 従来方式の問題点

(1)イベント駆動方式

イベント駆動方式では、回路中で、信号値の変化（イベント）した資源をfaninとする単位モデルをネットリストより求め、この評価のみを行うことによりシミュレーションを進める。

このような従来のイベント駆動方式におけるイベントの流れと処理を図４に示す。これは次の各処理から構成される。

①イベントの発生した資源の信号値を更新する信号値更新。
②イベントとネットリストを用いて評価すべき単位モデルを選択する単位モデル選択。
③選択された単位モデルを評価する単位モデル評価。
④評価した単位モデルのfanoutに発生したイベントをDEQ(Delayed Event Queue)にスケジュールするfanoutイベントスケジュール。

以上のようなイベント駆動方式により、RTL機能シミュレーションを行う場合、処理速度の遅さが問題となる。この処理時間コストの大部分は上記②に示した単位モデルの選択と③の単位モデルの評価の時間コストで占められる［７］。この理由として、以下が予想される。

①ゲートレベル等と比較して、単位モデルが複雑であり、また、多くの faninを持つためネットリストが大きくなる。このため、単位モデルの選択においてリスト検索に時間がかかる。

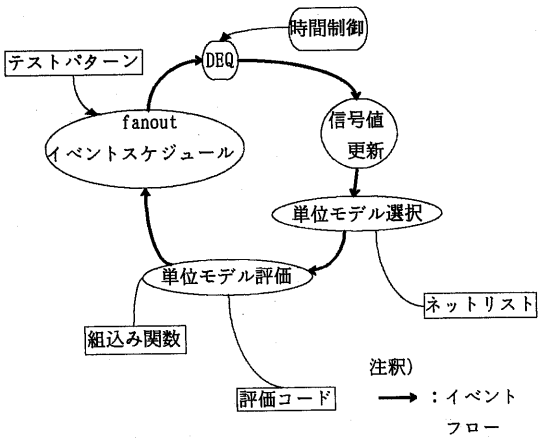


図４．従来のイベント駆動方式のイベントフロー

②多くの単位モデルにfanin となるグローバルな資源（クロック信号、制御信号、バスなど）が存在し、常に多くの単位モデルが評価されている。

従って、イベント駆動方式においてシミュレーションの高速化を図るためには以上のような処理の時間のコストを削減しなければならない。

(2)従来の高速化法

イベント駆動方式によるシミュレーションの高速化を図る方法として、sensitize/desensitize方式［８］、意味記述法［４］、clock suppression 方式［９］などが提案されている。これまで、ゲートレベルの論理シミュレータに適用した例が報告されている［８、10、11］。

①意味記述法，sensitize/desensitize方式

これは、制御信号などの信号の意味に着目し、セレクト信号等の制御信号がONにならない限り制御される部分回路の出力が変化しないことを利用するものである。この方式では、あらかじめ部分回路の制御信号を指定しておき、その制御信号がOFF である限り、その部分回路の入力にイベントが起きても部分回路をシミュレーションせず、このイベントの履歴のみが残される。制御信号がONになると、イベントの履歴がある場合のみ部分回路のシミュレーションを行う。

②clock suppression 方式

これは、クロック信号より発生するイベントの多くは不要であることを利用するものである。この方式では、クロック信号の立ち上がりエッジから次の立ち上がりエッジの間ではイベントが発生してもシミュレーションをしない。クロック信号の立ち上がりエッジにおいて、クロック信号以外の信号にイベントがあった場合にのみシミュレーションを行う。

しかしながら、①の方式を機能レベルに適用した場合、オートマトンのような同一の条件式を持つ単位モデルでは重複した評価を行ってしまう問題がある。また、②の方式では多相クロックや制御信号に適用することが難しい。

### 4.2 イベント・ルックアヘッド方式の考え方

従来方式の問題を解決し、シミュレーションの高速化を図るには次のことが必要である。

①評価される単位モデルの数を必要最小限に抑えること。

②評価すべき単位モデルを選択する処理の時間コストを削減すること。

この２点を効率良く実行するため、本報告のイベント・ルックアヘッド方式は次に示す考え方を採用した。

①図４に示した従来のイベント駆動方式のイベントフローにおいて、信号値更新後のfanoutイベントを用いて単位モデル評価の出力となるfanoutイベントの発生の有無を予測する”先読み”を行う。

②”先読み”は、クロック信号と制御信号によって行い、新たなfanoutイベントを生じる場合にのみ、その信号値更新後のfanoutイベントを選択する。以降の処理は、選択され、必要最少限のイベントのみにより行う。

③イベント選択の判定は、イベント選択表を用いて、一括に行う。

### 4.3 イベント選択表

ここでは、4.1 ②に述べた問題点を解決し、イベント最少化法を効率良く行うイベント選択表について述べる。

(1)構成

イベント選択表は、複数のエントリから構成される。各エントリは、以下の構成要素からなる。

**[エントリの構成要素]**

・＜タイミング式＞
・＜条件式＞
・ネットリストのネット名の並び

(2)生成方法

イベント選択表は、HDLコンパイラの出力する単位モデルリストから生成される。以下、この手順を述べる。

**[イベント選択表の生成手順]**

①単位モデルリストの各単位モデルの＜タイミング式＞と＜条件式＞そのものと、＜条件式＞と＜転送式＞に現れる資源名（ネット名）を抽出する。これらをエントリの構成要素とする。

②このエントリのうち、同一の＜タイミング式＞と＜条件式＞を持つエントリを統合する。統合された後のネット名の並びは、もとのエントリの持つすべてのネット名から構成される。

ここでは、上記の手順②により、4.5に述べるイベント最少化法において＜タイミング式＞と＜条件式＞が重複して評価されることを防いでいる。

＜RUN LENGTH DECODERの例（図1）＞

| ＜タイミング式＞ | ＜条件式＞ | イベントが発生するネット |
|---|---|---|
| ―― | ―― | SEL, RST, REG1, REG2 |
| CLK | ―― | RUNIN |
| CLK | STAT==s0 | RUNIN, STAT |
| CLK | STAT==s1 | RUNIN, COUNTER, STAT |
| CLK | STAT==s2 | RUNIN, COUNTER, STAT |
| CLK | STAT==s3 | STAT |

図5．イベント選択表の例

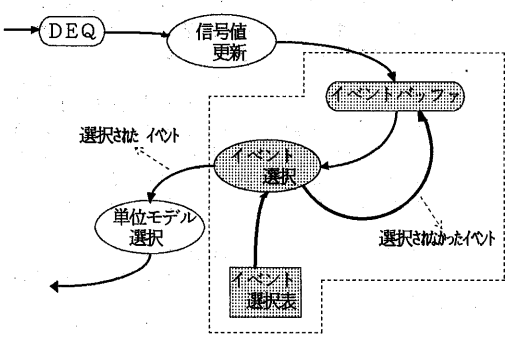


図6．イベントルックアヘッド方式

具体的には、図1の記述例に対して生成されるイベント選択表の内容を図5に示す。図5では、表の各行がイベント選択表の1個のエントリを表している。図5において、＜タイミング式＞あるいは＜条件式＞が空欄のエントリがあるのは、単位モデルに対応する転送記述文において、クロック信号CLKあるいはオートマトンで制御されないものがあるためである。

### 4.4 イベントの先読み

この先読みの処理は、図4のイベントフローにおいて信号値更新と単位モデル選択の処理の間に行われる。この処理におけるイベントフローを図6に示す。先読みの処理は、信号値更新後のイベントの履歴を保持するイベントバッファとこのイベントの履歴から有効なイベントの選択を判定するイベント選択表によって行う。

### 4.5 イベント最少化法

上記の先読みにおいて、必要なイベントのみを選択して不要なイベントを削除し、以降の処理で扱うイベント数を最少化するイベント最少化法についてのべる。

イベント選択の判定は、単位モデルの＜タイミング式＞、＜条件式＞を評価することによって行う。各単位モデルの＜タイミング式＞、＜条件式＞は、あらかじめ、図6のイベント選択表に記憶される。

次に、イベント最少化法のイベント選択手順を図6のイベントフローに従って説明する。

**[イベント最少化法のイベント選択手順]**

①信号値更新後のfanoutイベントをすべてイベントバッファへ記憶する。

②イベント選択表に記憶される各単位モデルの＜タイミング式＞と＜条件式＞の論理積を順次評価する。

③各単位モデルに対して、②の評価結果が真となる場合は④以降の処理を行い、結果が偽となる場合は処理を行わず、②に戻る。

④その単位モデル内の＜条件式＞と＜転送式＞のfaninとなる資源のイベントがイベントバッファに記憶されているか否かを判定する。

⑤イベントバッファにイベントが記憶されている場合は⑥の処理を行い、記憶されていない場合は②に戻る。
⑥イベントをイベントバッファから読み出し、イベントバッファの記憶を消去する。
⑦イベント選択表に記憶される各単位モデルの評価がすべて終了するまで、②から⑥までの処理を繰り返す。終了時点で、次の単位モデル選択に処理が移る。

以上の手順によって、選択されたイベントのみで以降の単位モデル選択、単位モデル評価が行われる。

**4.6 従来方式との比較**

イベント駆動方式に対する従来の高速化方式と本方式を比較する。ここでは、図7の例を用いて、本方式と、意味記述法及びclock suppression方式との比較を選択されるイベントの数で示す。

図7は、図に示す一定の転送遅延を持つ単位モデル（転送記述文）の fanin、fanoutのタイミング波形に対して発生するイベントを示したものである。この波形の変化点がイベントの発生点を表し、 faninのイベントについては点線で示している。次に、図中の記号を説明する。

(1)評価に必要なイベント：◎
点線で示す faninのイベントに対して、どのような方式であっても単位モデル(1)を評価するために選択しなければならない faninのイベントである。

(2)意味記述法：○
意味記述法によって選択される faninのイベントである。なお、図7の単位モデルでのSEL信号が制御信号として設定されている。この例では、図7の①、⑤、⑥のように制御信号SELが OFFの場合にイベントが選択されない。

(3)clock suppression 方式のイベント：△
clock suppression 方式によって選択される fanin のイベントである。この例では、クロック信号CLK の立上がりエッジにない図7の②、④、⑧のようなイベントは選択されない。

(4)イベント・ルックアヘッド方式：●
イベント・ルックアヘッド方式で選択される fanin のイベントである。

(5)各方式の比較
意味記述法とclock suppression 方式のどちらの方式も互いに異なる場合に不要なイベントを選択していることがわかる。
これに対して、イベント・ルックアヘッド方式では、図中の必要最少限のイベントのみを選択している。従来のどちらの方式に比較してもより高い効果があることがわかる。
なお、上記の方式を採用していない従来のイベント駆動方式では全てのfanin イベントを選択して単位モデルの評価を行っている。

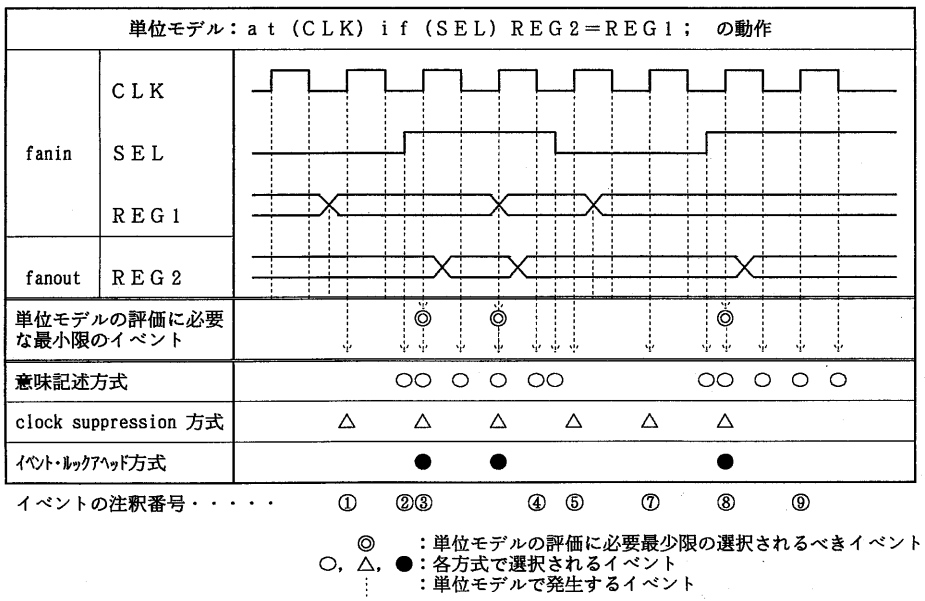


図7．各方式のイベント選択の比較例

## 5．実験結果

RTL機能記述のサンプルにより、従来のイベント駆動方式（以下、従来方式と呼ぶ）とイベント・ルックアヘッド方式によるシミュレーションの処理時間を比較した。

(1)実験方法

本報告では、順序回路を評価対象とした。これは本方式が、同期回路、オートマトンにより記述される順序回路に高い効果があることによる。

実験に用いる順序回路として、図1に示したランレングス符号のデコーダを用いた。処理時間は、CPUタイムを計測した。なお、本報告の機能シミュレータは、4MIPS、16MBメインメモリのUNIXマシン上で稼働している。

(2)実験結果

表1は、実験結果を示している。各評価項目の比は、本方式の処理時間に対する従来方式の処理時間の比である。評価モデル数は、シミュレーション中で評価された単位モデルの総数を表す。これは、本方式によるイベント削減の効果を直接表す値である。

評価モデル数比でみると本方式の効果は17.7倍となった。一方、シミュレーション処理時間比でみると約6倍であった。

なお、処理時間は、従来方式、本方式ともシミュレーション時間にほぼ比例していた。

(3)実験結果の検討

単位モデルの評価数比と処理時間比に結果として差が生じた。この理由は次のようなことが考えられる。

①単位モデル評価の時間コストは改善されたが、全体の処理時間に占めるイベント処理の時間コストの割合が大きかった。ただ、これはシミュレーションの入力パターンに影響するものである。

②各単位モデルの評価に要する時間コストが一定ではない。

したがって、評価数比が処理時間比に直接反映されるものではない。

<イベントルックアヘッド方式の効果>

| 評価項目 | 実験結果 |
|---|---|
| 評価モデル数比* | 17.7 |
| ｼﾐｭﾚｰｼｮﾝ処理時間比* | 3.7 |

＊：　従来方式／本方式

表1．実験結果

## 6．おわりに

本報告では、イベント駆動方式によるRTL機能シミュレーションの高速化を図るイベント・ルックアヘッド方式を提案した。そして、本方式が、従来のイベント駆動方式あるいは従来の高速化方式に対してよりも、より高い効果が得られることを示した。

サンプル評価の結果では、従来のイベント駆動方式に比較して6倍程度の処理速度が得られており、本方式の有効性を確認している。

今後の課題として、各種アーキテクチャに対しRTL機能記述を行い、シミュレータの能力を評価・検討することが挙げられる。

## 参考文献


[1] J.H.Aylor, R.Waxman, C.Scarrat, "VHDL-Feature Description and Analysis", IEEE Design & Test, April, 1988, pp.17-27, 1986.

[2] 星野，唐津，中島，" HSL-FX と言語標準化 ", 信学技報, Vol.87, No.126, pp.1-8, 1987.

[3] M. Shahdad, " An Overview of VHDL Language and Technology", 23rd DAC, pp.320-326, 1986.

[4] 稲垣，樋田，酒井，矢島，"LSI を含む階層的論理システムのための論理シミュレータ SIM/D", 情報処理論文誌, Vol.21, No.4, pp.332-339, 1980.

[5] C.Hansen, "Hardware Logic Simulation by Compilation", 25th DAC, pp.712-715, 1988.

[6] 水野，井川，宮崎，" 混合シミュレータ MULSのイベントバッファを用いた実現手法 ", 情報処理第36回全国大会, pp.1927-1928, 1988.

[7] L.Soulé, T.Blank, " Statistics for Parallelism and Abstraction Level in Digital Simulation", 24th DAC, pp.558-591, 1987.

[8] A.Miczo, D.Mohapatra, S.Perkins, K. Kaufman, K.Huang, " The Effects of Modeling on simulator Performance", IEEE Design & Test, Dec., 1987, pp.46-54, 1987.

[9] E.Ulrich, M.Kearney, J.Tellier, S.Demba, "Design Verification for Very Large Digital Networks Based on Concurrent Simulation and Clock Suppresion", ICCAD'83, pp.277-280, 1983.

[10] Y.Takamine, S.Miyamoto, S.Nagashima, " Clock Event Suppression Algorithm of VELVET And Its Apprication to S-820 Development", 25th DAC, pp.716-719, 1988.

[11] M.Loughzail, M.Gôté, M.Aboulhamid, E.Cerny, " Experience with the VHDL Environment", 25th DAC, pp.28-33, 1988.