# バーチャルタイムによる並列論理シミュレーション†

松本幸則†† 瀧 和男††

バーチャルタイム法による並列論理シミュレーションシステムを分散メモリ型並列マシン上に試作し，性能評価，ならびに他の方式との比較評価を行った．並列論理シミュレーションでは，時刻管理機構が重要な問題となる．並列処理に適した分散時刻管理機構として，コンサーバティブ法とバーチャルタイムの概念に基づく方法（バーチャルタイム法）がある．コンサーバティブ法は，デッドロック回避のオーバヘッドが問題である．バーチャルタイム法は，デッドロックの危険性がない半面，ロールバック処理が必要になる．われわれは，バーチャルタイム法による論理シミュレーションシステムを，分散メモリ型並列マシン「Multi-PSI」上に試作し，良好な速度向上，および性能を得た．さらに，ヌルメッセージを用いたコンサーバティブ法，および集中時刻管理機構によるシステムとの性能比較を行った結果，分散メモリマシン上の並列論理シミュレーションにおいては，バーチャルタイム法が最も有効な時刻管理機構であることを確認した．

## 1. はじめに

論理シミュレーションはLSI設計工程の一つであり，回路論理の検証，信号伝播遅延の検証を目的とする．論理シミュレーション工程は多大な計算時間を必要とすることから，高速シミュレータに対する要求は強い．また，高精度シミュレーション，およびマルチレベルシミュレーション[14]等の要求も高まっており，これらに柔軟に対応することも必要である．

シミュレーション専用ハードウェア[5]を用いることは，高速化には有効であるが，柔軟性をもつシミュレータの実現は容易ではない．ソフトウェアによる論理シミュレータを並列化することは高速かつ柔軟なシミュレータを実現する有望な方法であると考えられる．その一つとして，われわれはバーチャルタイムによる並列論理シミュレーションシステムを試作し，評価を行った．

並列論理シミュレーションは，並列イベントシミュレーションの問題として扱うことができる．並列イベントシミュレーションでは時刻の管理機構が重要である．時刻管理機構は，タイムホイルと呼ばれる集中時刻管理機構（タイムホイル法）と，分散時刻管理機構に大別できる[6]．タイムホイル法は，大域的な同期が必要であるため，多数のプロセッサを使用する場合，効率的な並列処理は難しいと考えられる．一方，代表的な分散時刻管理機構としては，コンサーバティブ法[9]およびバーチャルタイムの概念に基づく方法（バーチャルタイム法）[8]がある．コンサーバティブ法は，デッドロックの危険性があり，その回避のための処理が必要となる．また，バーチャルタイム法はデッドロックの危険性がない反面，ロールバック処理が必要である．

従来の並列論理シミュレーションの研究では，タイムホイル法[1),11]，およびコンサーバティブ法[7),16),17]，が取り上げられてきている．しかしながら，バーチャルタイム法の適用例は少なく[2),4]，ロールバック処理等オーバヘッドに関する計測結果は報告されていない．また，これら3種類の時刻管理機構を同一計算機上で比較した例もない．

バーチャルタイム法は，ロールバック処理のオーバヘッドを小さく抑えられる場合には，効率的な並列処理を実現すると考えられる．われわれはバーチャルタイム法の有効性検証を目的とし，この方法を改良した並列論理シミュレーションシステムを分散メモリ型並列マシン「Multi-PSI」[13]上に構築するとともに，性能評価，ロールバックオーバヘッド等の計測を行った．さらに，ヌルメッセージを用いたコンサーバティブ法，およびタイムホイル法による論理シミュレータも構築し，これらのシミュレーション性能を比較した*．

本稿では，以下第2章でバーチャルタイム法の概要を述べる．また，第3章で本並列論理シミュレーションシステムについて説明する．続いて，第4章で本システムの性能，速度向上，および各種オーバヘッドについての計測結果を報告する．最後に第5章でヌルメッセージを用いたコンサーバティブ法，およびタイム

† Parallel Logic Simulation Based on Virtual Time by YUKINORI MATSUMOTO and KAZUO TAKI (The Seventh Laboratory, Institute for New Generation Computer Technology).

†† (財)新世代コンピュータ技術開発機構第7研究室

* 本研究は第五世代コンピュータプロジェクトの一環として行われた．

ホイル法との比較結果を報告し，バーチャルタイム法の優位性を示す．

## 2. バーチャルタイムの概要

イベントシミュレーションは，複数のオブジェクトがメッセージ通信を行うことによって，次々と状態を変えていく形にモデル化できる．メッセージはイベント情報を持つとともに，イベントの発生時刻がスタンプされている（タイムスタンプ）．

Jefferson は，バーチャルタイムの概念と，その並列イベントシミュレーションへの応用を提案している[8]．バーチャルタイムの概念による方法（バーチャルタイム法）には，局所的な処理と大域的な処理がある．

### 2.1 局所的な処理

バーチャルタイム法では，メッセージは正しい順序，すなわちタイムスタンプ値の小さい順に，各オブジェクトに到着するという仮定に基づいて処理を進める．しかしながら，実際には誤った順序でメッセージが到着する場合が存在する．このような状況に備え，オブジェクトはメッセージおよび状態の履歴を保存する．

オブジェクトは，メッセージの到着順序の矛盾を発見したところで履歴を巻き戻し（ロールバック），処理のやりなおしをする．また，その時点で誤って送信したことが判明したメッセージに対しては，メッセージを取り消す役割を持つアンチメッセージを送信する．上記の処理によりシミュレーション結果の正当性が保証される．

### 2.2 大域的な処理

バーチャルタイム法ではロールバックに備えて履歴を保存するため，メモリ消費が重大な問題となる．このため，時々大域的なシミュレーション時刻（GVT）を求め，不要となった履歴領域を解放する．GVT は，ある時点での，全オブジェクトのシミュレーション時刻，およびオブジェクト間を通過中のメッセージのタイムスタンプ値のうちの，最小値以下のものである．GVT 以前にロールバックすることはないことから，GVT 以前の履歴領域は解放することができる．

## 3. 実験システム

### 3.1 実行環境の概要

本実験システムは，並列論理型言語 KL 1[15] で記述され，Multi-PSI[12),13)] 上に実装されたシステムである．

KL 1 は，並列オブジェクトモデルに基づくプログラムの記述に適した言語であり，言語処理系によってデータフロー同期* が実現される．プログラマは同期やプロセッサ間通信について，陽に記述する必要がないため，一般的な手続き型言語に比べ並列プログラム開発が容易である．

Multi-PSI は MIMD 型マシンで，要素プロセッサ (PE) 64 台が，2次元メッシュ状ネットワークで結合されている[12)]．要素プロセッサはマイクロ命令制御で 5 MIPS の性能であり，ネットワークの性能は 5 M バイト/秒である．各要素プロセッサは 16 M ワードのメモリを持ち，これらはすべて分散管理されている．したがって，他の要素プロセッサへのデータアクセスコストすなわち PE 間通信コストは大きいが，台数拡張性に富むアーキテクチャである．

### 3.2 論理シミュレーションの仕様

本システムでは，ゲートレベルで記述された回路を扱う．回路としては，組み合わせ回路，同期回路のみならず，非同期回路も扱う．64プロセッサを用いた場合，500 万ゲート規模の回路のシミュレーションが可能である**．

信号値は Hi, Lo, X（不定）の3値モデルとし，遅延は各ゲートに単位時間の整数倍を割り当てるようなノンユニット遅延モデルとする．また，スパイクについては，これを検出し，取り消しを行う機能を持つ．

本システムの目的は並列論理シミュレーションの実験であることから，最低限の一般性を持たせた単純な仕様にしているが，機能の拡張は容易である***．

### 3.3 実　　装

#### 3.3.1 全体構成

本システムは，前処理部とシミュレーション部の二つの部分から成る．

前処理部は，並列シミュレーションに備え負荷分散を決定する．ここでは，今回提案した縦割り指向戦略に基づき回路データを分割し，各 PE に静的に割り当てている．負荷分散は，PE 間メッセージ通信と共に，ロールバック頻度にも大きく影響を与えると考えられ

---

* 必要なデータがすべてそろった後に，処理が開始されるような同期機構

** 本システムでは，ユーザが使用できる約6M ワードのメモリ領域のうち，10～15% を履歴領域として使用し，残りをゲート情報の領域として用いる．これで GVT 更新オーバヘッドを全処理時間の 1% 程度に抑えられる．なお，履歴領域は必要時に動的に割り当てられ，その全体量は対象回路規模に依存しない．

*** 慣性遅延の取り扱いについても，GVT 更新処理を少し変更する程度で対応可能である．

るため，非常に重要である．

シミュレーション部では，バーチャルタイム法による並列シミュレーションを行う．各 PE には局所メッセージスケジューラを置く．スケジューリング戦略もロールバック頻度に大きく影響するため重要である*．また，ロールバックコスト低減のためアンチメッセージの削減方法を新たに提案，採用している．

### 3.3.2 局所メッセージスケジューラ

本システムでは，各ゲートが第2章で述べたオブジェクトに対応する．通常各 PE には，複数のオブジェクトおよび複数の未処理メッセージが存在する．この場合，各 PE 内でメッセージのスケジューリングを適切に行うことは，ロールバック頻度低減に有効となる．

スケジューリング戦略としては，いくつかの方法が提案されているが[3]，論理シミュレーションでは，一つのメッセージあたりの処理量が小さい（小粒度）ため，スケジューリング処理の軽いものが望ましいと考えられる．このことから，本システムでは，PE に到着している未処理メッセージのうち，最小タイムスタンプのものから処理を行う戦略（以後，最小時刻優先戦略と呼ぶ）を採用した．

スケジューラは各時刻に対応したスロットを持つ．各スロットには，同一のタイムスタンプ値を持つメッセージが登録される．スケジューラは登録中のメッセージのもつ最小のタイムスタンプ値をクロックとし，クロックに対応したメッセージを順次受信側オブジェクトに送る．

### 3.3.3 アンチメッセージの削減

Jefferson によるバーチャルタイム法では，メッセージ送受信における順序保存性を仮定しないため，取り消すべきメッセージすべてに対しアンチメッセージを送る（**図1**）．

本システムでは，信号線は，KL1 のストリームとして表現され，メッセージはストリーム上を流れるデータとして表現される．KL1 では，同一ストリーム上のデータの送信順が保存されるため，送信者と受信者の間ではメッセージの送信順序は保存される．

本システムでは，この環境のもとで有効な福井の方法[18]に改良を加えたアンチメッセージ削減方法を用いている．はじめに，元となった福井の方法を述べる．

アンチメッセージ削減方法1

取り消すべきメッセージのうち，最小のタイムスタンプ値を持つメッセージに対応したアンチメッセージ *AM* のみを送る．ここで *AM* のもつタイムスタンプ値を *TS*(*AM*) とする．この時，受信者は，同一信号線上で受信したメッセージのうち，*AM* 到着以前に受信し，かつ *TS*(*AM*) 以上のタイムスタンプ値を持つメッセージのみを取り消せばよい（**図2**）．

本システムでは，ロールバックが発生したオブジェクトから，アンチメッセージと通常のメッセージが連続して送信される場合があることに着目し，次に示す改良を加えた．

アンチメッセージ削減方法2

送信側オブジェクトでロールバックが発生すると同時に，取り消すべき一連のメッセージの最小タイムスタンプ値以下のタイムスタンプを持つ新たなメッセージ $M_{new}$ が発生する場合，単に $M_{new}$ の送信を行うだけでアンチメッセージの送信は行わない．受信者は，同一信号線上で $M_{new}$ 以前に受信したメッセージのうち，$TS(M_{new})$ 以上のタイムスタンプ値を持つメッ

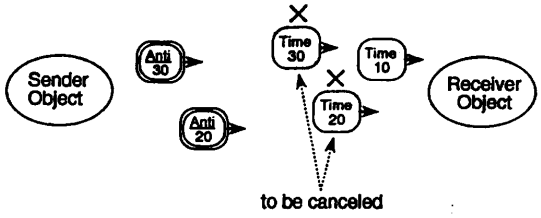


**図1** アンチメッセージ送信
Fig. 1 Cancelling invalid messages with antimessages.

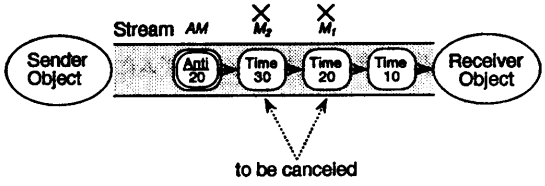


**図2** アンチメッセージ削減方法1
Fig. 2 Antimessage reduction method 1.

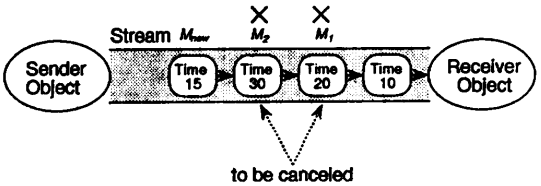


**図3** アンチメッセージ削減方法2
Fig. 3 Antimessage reduction method 2.

* 本システムでは，ロールバック頻度低減のため，負荷分散の工夫と，メッセージのスケジューリングを行っている．ロールバック頻度低減の方法としては，移動時刻窓 (Moving Time Window) も提案されているが[10]，窓の大きさの決分が難しい上に，その効果も小さい[2]ため，本システムでは採用していない．

セージについて取り消し処理をすれば良い（図3）.

### 3.3.4 負荷分散方法

並列論理シミュレーションは一つのメッセージあたりの処理量が小さい，すなわち小粒度であるためプロセッサ間通信のオーバヘッドが問題になる．また，バーチャルタイム法では，ロールバックの発生量も問題になる．この問題を分散メモリ型並列マシン上で実行する場合，1：負荷の均等分散，2：プロセッサ間通信の低減，3：十分な並列性の抽出，の3点が負荷分散の目標となる．

最適な負荷分散のためには，上記3点を満足するような評価関数を定義し，その値を最良にする負荷分散方法を求めなければならない．しかし，このような問題は一般にNP困難であるため，何らかのヒューリスティックスを用いることになる．

今回，上記目標の3点をある程度満足し，かつ計算時間がシミュレーション時間に比べ十分に小さい負荷分散方法として，縦割り指向戦略と名付けた戦略により回路を分割し，得られた部分回路を各プロセッサに静的に割り当てる方法を試みた．

縦割り指向戦略は，連結したゲートはできるだけ同一PEに割り当てるとともに，複数ファンアウト部の並列性を抽出することを意図している．この戦略では，回路入力端子から順にゲート接続関係をたどり，縦方向につながったゲートをグループにまとめることによって，回路をいくつかのクラスタに分割する．ここで，複数の出力先ゲートが存在する場合，そのうちの1ゲートのみをグループ中のゲート群に取り込み，他のゲートは，新たに別のグループ化処理の開始点とする．全ゲートのグループ化終了後，小さいクラスタについては，接続関係にある別クラスタに統合する．また，極端に長いクラスタはいくつかに切り分ける．最後に，生成されたクラスタをランダムに各PEに割り当てる．

縦割り指向戦略は，Agrawalの方法[1]に類似しているが，バーチャルタイム法での実行を前提として，より多くの並列性を抽出する手続きを加えたものである*.

* Agrawalの方法は，タイムホイル法による並列シミュレーションを前提にしているため，ゲート遅延情報を用いて，各単位時刻ごとに発生するイベント数を予測し，負荷の均等化を図っている．しかし，バーチャルタイム法は，異なる時刻のイベントを並列に処理できるため，縦割り戦略では，このような予測処理を行わない．また，縦割り戦略では，生成された極端に長いグループを切り分けることにより，パイプライン的な並列性をも抽出するようにしているが，Agrawalの方法ではこのような切り分けは行われていない．

## 4. 測定結果と考察

ISCAS '89のベンチマークから四つの順序回路についてシミュレーションを行い，性能，速度向上，および各種オーバヘッドを計測した．

対象回路のゲート数，信号線数および平均ファンイン $F_{in}$，ファンアウト $F_{out}$ を表1に記す．なお，Dフリップフロップはゲートに展開した．

今回の実験では，各ゲートには，すべて1単位時間の遅延値を与えた．また，クロックの周期は40単位時間とし，クロック線以外の入力端子には，クロックの立ち上がりに同期してランダムに信号値が変化するような入力信号列を与えた．

### 4.1 測定結果

表2に各回路のシミュレーションにおける処理性能を，図4に速度向上のグラフを示す．

また，全実行時間に対する各種処理時間の割合を表3に示す．なお，この値は，64 PE使用時の全PEでの平均値を示している．

表4には，評価されたメッセージのうち，真のイベント（最終的に巻き戻されなかったもの）と最終的に巻き戻されたものの割合を示す．

表5には，64 PE使用時のロールバックの頻度 $f_r$ と平均的深さ（1ロールバックあたりの巻き戻し履歴数）$d_r$，PE間メッセージ通信の頻度 $f_c$ を示す．$f_r$，$d_r$，$f_c$ は以下のように定義する．

$$f_r = R/E$$

表1 対象回路
Table 1 Feature of target circuits.

| 回路 | s 1494 | s 5378 | s 9234 | s 13207 |
|---|---|---|---|---|
| ゲート数 | 683 | 3,853 | 6,965 | 11,965 |
| 信号線数 | 1,490 | 6,588 | 10,957 | 19,983 |
| $F_{in}$ | 2.15 | 1.70 | 1.57 | 1.66 |
| $F_{out}$ | 2.08 | 1.61 | 1.50 | 1.55 |

表2 性能（イベント/秒）
Table 2 Performance.

| PE数\回路 | s 1494 | s 5378 | s 9234 | s 13207 |
|---|---|---|---|---|
| 1 | 2,572 | 2,410 | 2,326 | 2,051 |
| 2 | 3,460 | 4,375 | 4,452 | 4,406 |
| 4 | 5,662 | 8,401 | 7,709 | 9,092 |
| 8 | 8,448 | 14,617 | 12,586 | 18,101 |
| 16 | 10,413 | 26,141 | 19,003 | 33,793 |
| 32 | 10,912 | 40,806 | 23,777 | 62,201 |
| 64 | 10,943 | 64,013 | 35,118 | 99,299 |

$d_r = H_r/R$

$f_c = M_c/M_{all}$

ここで，$R$：ロールバック発生回数，$E$：真のイベント数，$H_r$：巻き戻された履歴数，$M_c$：PE 間を移動したメッセージ数，$M_{all}$：全メッセージ数である．

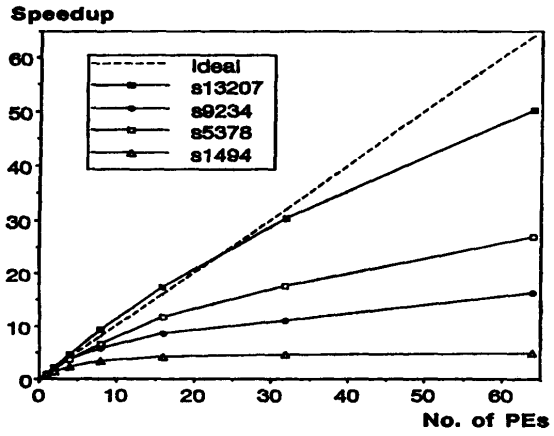


図 4 速度向上
Fig. 4 Speedup.

表 3 各種処理の占める割合 (%, 64 PE 使用時)
Table 3 Percentage of time for each process.

| 処理＼回路 | s 1494 | s 5378 | s 9234 | s 13207 |
|---|---|---|---|---|
| メッセージ評価等の処理(注) | 72.28 | 80.28 | 58.69 | 85.79 |
| ロールバック処理 | 5.32 | 2.50 | 1.61 | 1.53 |
| PE 間通信処理 | 13.13 | 8.24 | 4.38 | 2.12 |
| GVT 更新処理 | 1.21 | 0.48 | 0.62 | 0.64 |
| 履歴解放処理 | 0.43 | 2.41 | 1.57 | 3.86 |
| アイドル時間 | 7.63 | 6.09 | 33.13 | 6.06 |

(注) これは，真のイベント（巻き戻されなかったメッセージ），巻き戻されたメッセージ双方の評価とも含む．

表 4 評価メッセージの内訳 (%, 64 PE 使用時)
Table 4 Percentage of actual events and rolled back messages (64 PE).

| 回　路 | s 1494 | s 5378 | s 9234 | s 13207 |
|---|---|---|---|---|
| 真のイベント | 29.34 | 81.84 | 65.23 | 86.54 |
| 巻き戻されたメッセージ | 70.66 | 18.16 | 34.77 | 13.46 |

表 5 ロールバック頻度と深さ，PE 間メッセージ通信頻度 (64 PE 使用時)
Table 5 Frequency and depth of rollback, frequency of inter-PE communication.

| 回　路 | s 1494 | s 5378 | s 9234 | s 13207 |
|---|---|---|---|---|
| $f_r$ | 0.938 | 0.0703 | 0.0510 | 0.0227 |
| $d_r$ | 2.57 | 3.16 | 10.5 | 6.84 |
| $f_c$ | 0.402 | 0.121 | 0.0846 | 0.0222 |

**表 6** には，1回の GVT 更新処理時間と，回路 s 13207 の場合の GVT 更新回数を示す．なお，各 PE でのシミュレーション時刻は，ゲートごとではなく，メッセージスケジューラが管理しているため，1回の GVT 更新処理時間は対象回路規模に依存しない．

### 4.2 速度向上についての考察

64 PE 使用時の速度向上として，s 13207, s 5378 については良好な結果が得られたが，s 9234, s 1494 についてはやや不満足な結果となった．

速度向上に影響を与えるものとして 1：GVT 更新と履歴解放処理，2：問題の並列性，3：静的負荷分散（負荷の不均等），4：ロールバック処理，5：PE 間メッセージ通信，の5点が考えられる．以下，これらについて考察を行う．

#### 4.2.1 GVT 更新と履歴解放処理

表3および表6より，GVT 更新の処理時間は，64 PE 使用時でも非常に短く，GVT 更新処理がシステム性能に与える影響は小さいことが分かる．一方，GVT 更新後に行われる履歴解放処理の時間 $t_f$ は，s 1494 の場合を除き GVT 更新処理時間より長く，システム性能に与える影響は無視できないと考えられる．

本システムでは，状態履歴および出力値は各オブジェクト単位で，また，入力メッセージは各入力信号線単位で管理している．したがって $t_f$ は，オブジェクト数 $G$，ファンイン数 $F_{in}$，解放履歴数 $h_f$ と，次式に示す関係を持つ．

$$t_f = k_f h_f + (k_g + k_l F_{in})G + C_f \qquad (1)$$

ここで $k_f$, $k_g$, $k_l$ は係数，$C_f$ は定数分である．PE $i$ 番での全履歴解放処理時間 $T_f(i)$ は，シミュレーションの開始から終了までの期間，$n$ 回 GVT 更新されたとすると，式(1)から

$$T_f(i) = \sum_{j=1}^{n} t_f(j) = k_f E_i + n(k_g + k_l F_{in})G_i + nC_f \qquad (2)$$

表 6 GVT 更新処理時間
Table 6 Time of updating GVT.

| PE 数 | GVT 更新処理時間（1回あたり，ミリ秒） | GVT 更新回数 (s 13207) |
|---|---|---|
| 8 | 6.286 | 16 |
| 16 | 13.598 | 9 |
| 32 | 23.851 | 5 |
| 64 | 49.168 | 3 |

ここで，$E_i$ は PE $i$ 番での真のイベント数であり，$G_i$ は PE $i$ 番に割り当てられたオブジェクト数である．$E_i$ および $G_i$ は 1/(使用 PE 数) にほぼ比例すると考えられる．

一方，各 PE は等量のメモリを持つため，GVT 更新回数 $n$ も 1/(使用 PE 数) にほぼ比例すると考えられる．実際，表 6 も GVT 更新回数が 1/(使用 PE 数) にほぼ比例することを示している．結局，式(2)右辺 $n(k_g+k_lF_{in})G_i$ の項は 1/(使用 PE 数)$^2$ にほぼ比例する．

図 4 において，s 13207 の場合，2 PE～16 PE 使用時にはスーパリニアな速度向上が観測されているが，これは上記 $n(k_g+k_lF_{in})G_i$ の項の影響と考えられる．

#### 4.2.2 問題の並列性と静的負荷分散（負荷の不均等）

ここでは問題の持つ並列性についての考察を行う．以下，簡単のためメッセージの処理以外のコストはすべて無視できると仮定する．

シミュレーション問題 $Prb$ を PE $N$ 台への負荷分散 $Dst(N)$，スケジューリング戦略 $Sch$ の下で解く場合の計算時間を $T(Prb, Dst(N), Sch)$ とする．そして，問題自身の持つ並列度 $P_\infty$ を

$$P_\infty=\frac{T(Prb, Dst(1), Sch_{stf})}{T(Prb, Dst(\infty), oracle)}$$

と定義する．ここで $Dst(\infty)$ は各オブジェクトをすべて異なる PE に割り当てるものである．また，$Sch_{stf}$ は，最小時刻優先戦略とする．$Sch_{stf}$ は，1 PE でのシミュレーション時には，最適なスケジューリング戦略である．また $oracle$ は，常に最適なスケジューリングを与えるものとする．$P_\infty$ は，オブジェクト数以上の PE が存在する場合に得られる速度向上の上限となる．

現実には，PE 数は有限であり，しかもオブジェクト数より遙かに少ないことが一般的である．通常は，何らかの戦略に基づいて負荷分散を行う．

PE $N$ 台への負荷分散 $Dst(N)$，スケジューリング戦略 $Sch$ の下での問題の並列度 $P_N$ を，

$$P_N=\frac{T(Prb, Dst(1), Sch_{stf})}{T(Prb, Dst(N), Sch)}$$

と定義する．この値は，実際のシミュレーションで得られる速度向上の上限値の指標と考えられる．

各メッセージの処理コストはすべて等しいと仮定し，実際に行ったシミュレーションについて，それぞれの問題自身が持つ並列度 $P_\infty$ を実験的に求めた．また，縦割り指向戦略に基づく静的負荷分散方法を用いて 64 PE に負荷を分散し，スケジューリング戦略として最小時刻優先戦略を用いた場合の，各問題についての並列度 $P_{64}$ も実験的に求めた．さらに，1 回の GVT 更新にともなう履歴解放の処理時間は，どの PE でも等しいと仮定し．表 2 および式(2)を用いて履歴解放処理時間を除去した場合の 64 PE 使用時の速度向上 $S_{T_f=0}$ を計算した．表 7 に，$P_\infty$, $P_{64}$, $S_{T_f=0}$ の値を示す．

表 7 において，s 5378, s 9234, s 13207 の $P_\infty$ が PE 数 64 に比べ十分大きいため，$P_{64}$ の値としては 64 が期待される．また，s 1494 については，$P_{64}$ の上限値は 55.55 である．しかしながら，静的な負荷分散，スケジューリング戦略により並列性が制限されている．ここで注目したいことは，s 5378, s 9234, および s 13207 については $P_{64}$ と $S_{T_f=0}$ の差が小さい点である．このことは，負荷分散，スケジューリング戦略によって決まる並列性が，実際の速度向上を決定する大きな要因であることを示している．

なお，s 1494 では $P_{64}$ と $S_{T_f=0}$ の差が大きいが，これは表 4 に示すように，評価したメッセージの多くが巻き戻されているためである．

#### 4.2.3 ロールバック処理

ロールバック処理は，バーチャルタイム法の処理速度を低下させる最大の原因と考えられがちである．しかし，発生したロールバックがすべて処理速度を低下させるわけではない．なぜなら，ロールバックはシミュレーション時刻が将来に進みすぎている PE にのみ発生するのに対し，システムの処理速度は最も遅れている PE が決定するためである．

また，表 3 から，ロールバック処理時間が処理時間全体に占める割合は，最も大きい s 1494 の場合でも，高々 5% 程度であったことが分かる．したがって，いずれの回路においてもロールバック処理時間の影響は小さいと考えられる．

#### 4.2.4 PE 間通信

表 5 および表 3 より，s 13207, s 9234 の場合，PE 間通信頻度が低く抑えられており，実際に PE 間通信

表 7 並列度
Table 7 Parallelism.

| 回 路 | s 1494 | s 5378 | s 9234 | s 13207 |
|---|---|---|---|---|
| $P_\infty$ | 55.55 | 295.7 | 316.0 | 608.9 |
| $P_{64}$ | 18.88 | 35.52 | 17.95 | 43.24 |
| $S_{T_f=0}$ | 3.96 | 23.62 | 12.51 | 35.66 |

が全体の処理時間に占める割合も小さいことが分かる．一方，s1494 では PE 間通信頻度が高いため，他の回路に比べ，メッセージ通信に費やされた時間の割合が大きい．一般に，PE 数一定の環境では，オブジェクト数が少ないほど，また，一つのオブジェクトのファンイン，ファンアウト数が多いほど PE 間通信頻度も大きくなると考えて良い．表1から分かるように，s1494 は，他の3回路に比べ，ゲートオブジェクト数が少なく，平均ファンイン，ファンアウト数が大きい．s1494 の場合に PE 間通信が多いのはこれらの理由によるものであると考えられる．

以上の考察から，速度向上を決定する主な要因は，負荷分散およびスケジューリング戦略によって決まる並列性であり，ロールバック処理の影響は小さいことが分かった．また，s1494 のように平均ファンイン，ファンアウト数の大きい場合を除いて，PE 間メッセージ通信処理の影響も小さいことが分かった．今回の実験では，バーチャルタイム法は，問題の持つ並列性を効率的に引き出していると考えられる．

### 4.3 アンチメッセージ削減の効果

今回の実験では，ロールバック発生時に送信したアンチメッセージ数とともに，福井の方法を用いた場合に送信すべきアンチメッセージ数，および Jefferson の方法を用いた場合に送信すべきアンチメッセージ数についてもそれぞれ計測した．

**表8** に結果を示す．表では，64 PE 使用時に，Jefferson の方法を用いた場合に発生するアンチメッセージ数に対して，福井の方法（削減方法1），および本システムの方法（削減方法2）が削減したアンチメッセージ数の割合を百分率で示している．この表から，本論文で提案したアンチメッセージ削減方法が有効であることが分かる．

## 5. 時刻管理機構の比較

### 5.1 コンサーバティブ法

#### 5.1.1 機　　構

コンサーバティブ法は，バーチャルタイム法と同様に分散時刻管理機構の一つである[9]．

**表8** アンチメッセージ削減率（%，64 PE 使用時）
Table 8 Percentage of reduced anti-messages.

| 方法＼回路 | s1494 | s5378 | s9234 | s13207 |
|---|---|---|---|---|
| 削減方法1 | 27.69 | 39.74 | 84.12 | 67.15 |
| 削減方法2 | 38.14 | 49.83 | 89.23 | 69.98 |

コンサーバティブ法でも，各オブジェクトがメッセージ通信を行うことによってシミュレーションが進行するモデルを考える．この方法では，メッセージ通信において，同一信号線上メッセージの送受信順序が保存されている環境を前提とする．各オブジェクトは，自身のすべての入力信号線上に最低一つのメッセージが受信されるまで待ち合わせを行う．その後，最小のタイムスタンプ値を持つメッセージに対して処理を行う．

#### 5.1.2 ヌルメッセージ削減機構

コンサーバティブ法における最大の問題点は，回路がループ構造を持つ場合等にデッドロックが発生することである．ヌルメッセージを用いる方法はデッドロックを発生させない一つの方法であるが，ヌルメッセージが多量に生成されてしまう点が問題となる．ヌルメッセージを削減するために幾つかの方法が考案されている[7),9),16),17)]．

**図5** に一つの例を示す．この例は，同一信号線上でヌルメッセージを受信した直後に，さらに別のメッセージを受信した場合を示している．この場合，ヌルメッセージを即座に消去できる．

#### 5.1.3 計測結果および考察

上述したヌルメッセージ削減機構をもつコンサーバティブ法による論理シミュレーションを行い，性能，ヌルメッセージ数を計測した．

対象回路は s13207 であり，負荷分散およびスケジューリング戦略は第4章と同様のものである．**表9** に，各々の計測結果を示す．

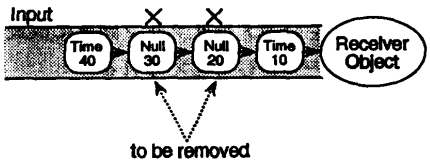


**図5** ヌルメッセージ削減機構
Fig. 5 Null message reduction mechanism.

**表9** コンサーバティブ法（回路 s13207）
Table 9 Performance using a conservative method.

| PE数 | 全メッセージ数 | ヌルメッセージ数 | 性　能（イベント/秒） |
|---|---|---|---|
| 2 | 94,783,392 | 92,442,018 | 91 |
| 4 | 98,768,309 | 96,426,935 | 164 |
| 8 | 101,437,325 | 99,095,951 | 291 |
| 16 | 102,163,983 | 99,822,609 | 503 |
| 32 | 102,589,874 | 100,248,500 | 953 |
| 64 | 104,182,696 | 101,841,322 | 1,684 |

ヌルメッセージ数

表9から，生成されたメッセージのうち，約 97% までがヌルメッセージであったことが分かる．

ヌルメッセージ削減機構のない場合，メッセージ受信ごとに必ずファンアウト数分だけメッセージを送信する．一般に論理回路でのゲートの平均ファンアウト数は1より大きいと考えられる．したがって，ヌルメッセージも含めた全メッセージ数は指数関数的に増大することが予想される．

実際には，タイムスタンプ値，および各ゲートでの遅延値が離散的であるためにメッセージ数の上限が存在し，(シミュレーション時間長)×(全信号線数)で与えられる．実験では期間 10,000 単位時間のシミュレーションを行ったため，メッセージ数上限は 199,830,000 となる．これに対し，実際のメッセージ数は 64 PE 使用時で 104,182,696 である．ヌルメッセージ削減機構を用いても，上限値の約半分のメッセージ発生があったことになる．

速度向上と絶対性能

速度向上の面では良好であるが，絶対性能は非常に悪い．1 PE 使用時のヌルメッセージを含めたメッセージ処理性能の測定結果は 2,006 メッセージ/秒であり，この値はバーチャルタイム法とほぼ等しい．それにもかかわらず，コンサーバティブ法の絶対性能が悪いのはヌルメッセージ数があまりに多いためである．ヌルメッセージを用いたコンサーバティブ法は，低コストでヌルメッセージを十分に削減できる方法がない限り，論理シミュレーションには不適切と考えられる．共有メモリマシンを前提としたヌルメッセージの削減方法[7),16)]を分散メモリマシン向きに改良することが，解決策の一つとして考えられる．

### 5.2 タイムホイル法

#### 5.2.1 機　　構

タイムホイル法は，従来の逐次的なシミュレーションで最も一般的なものである．

タイムホイルは既に述べた，最小時刻優先戦略によるスケジューラとほぼ同じものと考えて良い．ただし，タイムホイルでは，クロックは時刻の増大方向にのみ進む点が異なる．これは，時刻を集中管理しているため，タイムホイルのクロックよりも若い時刻のタイムスタンプ値を持ったメッセージが到着することがないためである．

#### 5.2.2 並列化方法

タイムホイル法の並列化方法を簡単に述べる．各 PE には一つのタイムホイルと，分割された部分回路が割り当てられる．タイムホイルは各々の部分回路に対応したメッセージを管理する．そして，すべてのタイムホイルは同期してクロックを進める．

表 10 タイムホイル法 (回路 s 13207)
Table 10 Performance using a time-wheel mechanism.

| PE 数 | 性能 (イベント/秒) |
|---|---|
| 1 | 3,236 |
| 2 | 5,696 |
| 4 | 11,374 |
| 8 | 17,916 |
| 16 | 25,394 |
| 32 | 21,656 |
| 64 | 11,957 |

#### 5.2.3 計測結果および考察

実験の対象回路として s 13207 を用いた．負荷分散はやはり第4章の方法を用いた．計測結果を表 10 に示す．

使用 PE 数が8以下の場合，バーチャルタイム法の場合よりも高い性能を示した．そして，16 PE 使用時で 25,394 イベント/秒という最高値を示している．しかしながら，32 PE 以上ではかえって性能が低下している．この理由としては，1：タイムホイルのクロックを同期して進めるためのオーバヘッドが大きいこと，2：使用 PE すべてを効率的に稼働させるだけの十分な並列性がないこと，の2点が考えられる．

タイムホイル法は，使用 PE 数が少なく，かつシミュレーション時間を粗く離散化する場合のみ，良好な性能が期待できる．

## 6. おわりに

バーチャルタイム法による論理シミュレーションシステムを構築し，性能評価を行った．その結果，問題に十分な並列性があるものについては，64 PE (要素プロセッサ) 使用時に約 99 K イベント/秒という性能および約 48 倍の速度向上を得た．これらの値は，本システムがノンユニット遅延モデルに基づくソフトウェアシミュレータとして高性能なものであることを示している．

一方，十分な速度向上および性能を得ることができなかったものについては，問題自身に十分な並列性がない，あるいは，静的負荷分散およびスケジューリング戦略により，並列性が制限されていることが主な原因であるとの分析結果を得た．

さらに，ヌルメッセージを用いたコンサーバティブ法，およびタイムホイル法による並列論理シミュレーションの実験も行い，パーチャルタイム法との性能比較を行った．コンサーバティブ法は，多量のヌルメッセージ発生により，パーチャルタイム法に大きく劣る性能を示した．また，タイムホイル法は，使用PE数が少ない場合にはパーチャルタイム法をうわまわる性能を示したが，使用PE数が増加するにつれ，性能が低下した．以上の比較から，パーチャルタイム法は，分散メモリマシン上での並列論理シミュレーションの時刻管理機構として最も有効な方法であることが確認できた．

今後は，静的負荷分散による並列性抽出が不十分な場合に対処するため，動的負荷分散導入の検討を予定している．また，実際のLSI設計データを用いたシミュレーションを行い，性能評価をする予定である．最終的には，本システムを，Multi-PSIの約20倍の総合性能を持つ並列推論マシンPIM上に移行する予定である．

**謝辞** 本研究を行うにあたり，多くの有益な助言をいただいたICOT第7研究室研究員諸氏，およびICOT PIC-WG委員諸氏に深謝する．

## 参考文献

1) Agrawal, P.: Concurrency and Communication on Hardware Simulators, *IEEE Trans. Computer-Aided Design*, Vol. CAD-5, No. 4, pp. 617-623 (1986).
2) Briner, J. V. et al.: Parallel Mixed-level Simulation Using Virtual Time, *CAD Accelerators*, Ambler, A. P. et al. (eds.), pp. 273-285, North-Holland (1991).
3) Burdorf, V. and Marti, J.: Non-Preemptive Time Warp Scheduling Algorithm, *ACM Operating System Review*, Vol. 24, No. 2, pp. 7-18 (1990).
4) Chung, M. J. and Chung, Y.: Data Parallel Simualtion Using Time-Warp on the Connection Machine, *26th ACM/IEEE Design Automation Conference*, pp. 98-103 (1989).
5) Denneau, M. M.: The Yorktown Simulation Engine, *19th ACM/IEEE Design Automation Conference*, pp. 55-59 (1982).
6) Fujimoto, R. M.: Parallel Discrete Event Simulation, *Comm. ACM*, Vol. 33, No. 10, pp. 30-53 (1990).
7) Soule, L. and Gupta, A.: Analysis of Parallelism and Deadlock in Distributed-Time Logic Simulation, Stanford University Technical Report, CSL-TR-89-378 (1989).
8) Jefferson, D. R.: Virtual Time, *ACM Trans. Prog. Lang. Syst.*, Vol. 7, No. 3, pp. 404-425 (1985).
9) Misra, J.: Distributed Discrete-Event Simulation, *ACM Comput. Surv.*, Vol. 18, No. 1, pp. 39-64 (1986).
10) Sokol, L. M. et al.: MTW: A Strategy for Scheduling Discrete Simulation Events for Concurrent Execution, *SCS Multiconference on Distributed Simulation*, pp. 34-42 (1988).
11) Soulé, L. and Blank, T.: Parallel Logic Simulation on General Purpose Machines, *25th ACM/IEEE Design Automation Conference*, pp. 166-170 (1988).
12) Takeda, Y. et al.: A Load Balancing Mechanism for Large Scale Multiprocessor Systems and Its Implementaion, *The International Conference on Fifth Generation Computer Systems*, pp. 978-986 (1988).
13) Taki, K.: The Parallel Software Research and Development Tool: Multi-PSI System, *Programming of Future Generaiton Computers*, Fuchi, K. and Nivat, M. (eds.), pp. 411-426, North-Holland (1988).
14) Tham, K. et al.: Functional Design Verification by Multi-Level Simulation, *21st ACM/IEEE Design Automation Conference*, pp. 473-478 (1984).
15) Ueda, K. and Chikayama, T.: Design of the Kernel Language for the Parallel Inference Machine, *Comput. J.*, Vol. 33, No. 6, pp. 494-500 (1990).
16) 工藤ほか：共有メモリを想定した並列論理シミュレータ，電子情報通信学会研究会報告，CPSY 91-23, pp. 151-158 (1991).
17) 下郡，鹿毛：メッセージドリブンによる並列論理シミュレーション，電子情報通信学会研究会報告，CAS 88-110, pp. 23-30 (1989).
18) 福井：パーチャルタイムアルゴリズムの改良，情報処理学会論文誌，Vol. 30, No. 12, pp. 1547-1554 (1989).

(平成3年8月1日受付)
(平成3年12月9日採録)

**松本　幸則**

昭和37年生．昭和60年京都大学工学部電子工学科卒業．同年三洋電機(株)入社．筑波研究所にてFA用画像処理検査装置の開発に従事．平成元年(財)新世代コンピュータ技術開発機構に出向．以来並列応用ソフトウェアの研究開発に従事．電子情報通信学会会員．

**瀧　和男**（正会員）

昭和27年生．昭和51年神戸大学工学部電子工学科卒業．昭和54年同大学院修士課程システム工学修了．工学博士．同年(株)日立製作所入社．同社大みか工場にて制御用計算機システムの設計に従事．昭和57年(財)新世代コンピュータ技術開発機構に出向．以来逐次型および並列型推論マシンと並列応用プログラムの研究開発に従事．現在第1研究室室長．並列マシンのアーキテクチャ，並列プログラミング，LSI CADなどに興味を持つ．電子情報通信学会，IEEE，ソフトウェア科学会各会員．