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
(財) 新世代コンピュータ技術開発機構

概要

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取り扱うことができる。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な問題となる。並列処理に適した分散時刻管理機構を持つものとして、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という利点がある半面、ロールバック処理が必要になる。

我々は、分散メモリ型並列マシン「Multi-PSI」上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、速度向上、絶対性能の両面で良好な結果を得た。

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、集中時刻管理機構による並列論理シミュレーションの実験も行い、バーチャルタイム方式との性能比較を行った。その結果、並列論理シミュレーションの方法としては、バーチャルタイム法が最も有効な方法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設計工程の中で設計仕様の検証、とくに回路の論理の検証、信号伝播遅延の検証に重要な役割を果たしている。しかしながら膨大な時間が費やされてしまう点が大きな問題であり、高速シミュレータに対する要求は強い。また、高精度のシミュレーションに対する要求も非常に強くなってきている。

ハードウェアエンジンを用いることは、シミュレーションの高速化という視点からは良い手段であるが、高精度シミュレーションの要求に柔軟に対応することは困難である。ソフトウェアによる論理シミュレーションを並列化することは高速且つ高精度シミュレーションを実現する有望な方法であると考えられ、大きな期待が寄せられている。

イベント駆動による並列論理シミュレーションの時刻管理機構は、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、分散時刻管理機構の2つに大別できる [4]。タイムホイル法は、大域的な時刻の同期をとるため、多数のプロセッサを使用する場合、十分な並列性を抽出することが難しいと考えられている。分散時刻管理機構の代表的なものとしては、コンサーバティブ法 [2] 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) [1] 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るために、その回避のための処理が大きな問題となる。バーチャルタイム法はデッドロックの危険性がないという利点がある半面、ロールバック処理が必要であるという欠点をあわせ持つ。

我々は第五世代コンピュータプロジェクトの一環として、分散メモリ型の並列推論マシン実験機「Multi-PSI」[3] 上にバーチャルタイムによる並列論理シミュレーション実験システムを構築し、性能評価を行うとともに、ヌルメッセージを用いた場合のコンサーバティブ法及びタイムホイル法との比較を行った。

本論文では、以下第2節でバーチャルタイム法についての説明を行う。また、第3節で本並列論理シミュレーションシステムの概要を述べる。続いて、第4節で本システムの性能、速度向上についての計測結果を報告する。最後に第5節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

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なうことによって、次々と状態を変えていく形にモデル化できる。オブジェクトは状態オートマトンとして表現され、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)。

Parallel Logic Simulation based on Virtual Time
Yukinori MATSUMOTO, Kazuo TAKI
Institute for New Generation Computer Technology

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 [1]。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、各オブジェクトは、メッセージは正しい順序、すなわちタイムスタンプの小さい順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に対する処理と共に、メッセージおよび状態の履歴保存を行う。

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。さらに、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なるものを送信する。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) を求める必要がある。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、及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 実験システム概要

3.1 実行環境

本実験システムは、並行論理型言語 KL1[8] で記述され、Multi-PSI[3] 上に実装されたシステムである。Multi-PSI は MIMD 型マシンで、要素プロセッサ (PE) 64 台が 2 次元メッシュ状のネットワークで結合されている。メモリは全て分散管理されており、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 間通信コストは高いが、台数拡張性に富む。

3.2 仕様

本シミュレーションシステムでは、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。

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。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 構成

本システムは、前処理部とシミュレーション部の 2 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。

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。ここではロールバック頻度低減のため各 PE に局所メッセージスケジューラを置く。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処理も行う。

3.3.1 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 節で述べたオブジェクトに対応する。通常ゲート数は PE 数に比べはるかに多いため、各 PE 内でメッセージのスケジューリングを行うことは、ロールバック頻度低減に有効である。

本システムでのスケジューリング戦略は、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 (以後タイムスタンプソート戦略と呼ぶ) ものである。

スケジューラは各単位時間に対応したスロットを持つ。スロットはスタック構造を持ち同一タイムスタンプ値のメッセージは、到着順にスタックに登録 (プッシュ) 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ポップして受信側オブジェクトに送る。

3.3.2 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の仮定をおいていないため、送信側でロールバックが発生した場合、取り消すべきメッセージ全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

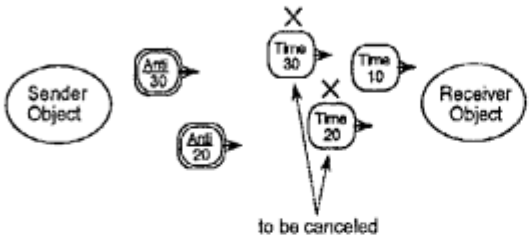

図 1: アンチメッセージ送信

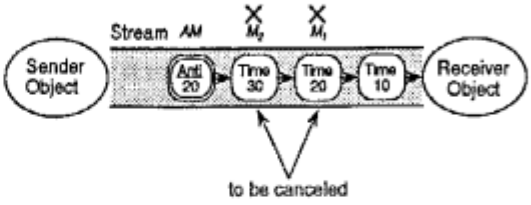

図 2: アンチメッセージ削減 (a)

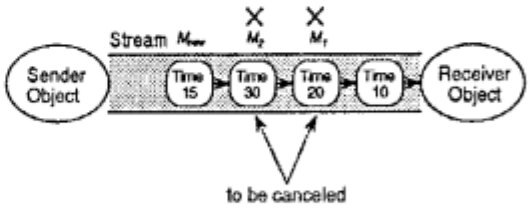

図 3: アンチメッセージ削減 (b)

受信者の間ではメッセージの送信順序は保存される。この環境では、以下のようにアンチメッセージを削減することができる [6]。

本システムでは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、且つ $TS(AM)$ 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、更に次のような場合にもアンチメッセージの省略を行っている。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セージについて取り消し処理をすれば良い (図 3)。

3.3.3 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、即ち小粒度であるためプロセッサ間通信量が問題になる。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。

一般に、論理回路では、ゲートの複数ファンアウト部に多くの並列性を見出すことができる。縦割り指向戦略は、縦方向につながったゲートをグルーピングすることで回路を幾つかのクラスタに分割するものである。縦割り指向戦略は、連結したゲートはできるだけ同一クラスタとすると同時に、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。

縦割り指向戦略によって生成されたクラスタのうち、大きさの小さいものについてはそれがつながっている別のクラスタにマージする。また、極端に長いクラスタは幾つかに切り分ける。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。

4 測定結果と考察

ISCAS'89 のベンチマークから 4 つの順序回路について、縦割り指向戦略に基づく負荷分散方法を用いて並列論理シミュレーションを行ない、性能、速度向上等を計測した。

対象回路のゲート数、信号線数及び平均ファンイン $\overline{Fanin}$、ファンアウト $\overline{Fanout}$ 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、全て 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。

表 1: 対象回路

回路	s1494	s5378	s9234	s13207
ゲート数	683	3,853	6,965	11,965
信号線数	1,490	6,588	10,957	19,983
$\overline{Fanin}$	2.15	1.70	1.57	1.66
$\overline{Fanout}$	2.08	1.61	1.50	1.55

表 2: 性能 (イベント / 秒)

PE 数 \ 回路	s1494	s5378	s9234	s13207
1	1,460	1,410	1,313	1,246
2	2,240	2,624	2,642	2,570
4	3,613	4,929	4,715	5,424
8	5,702	8,731	7,605	10,753
16	7,498	16,024	11,294	20,385
32	8,857	25,210	13,958	36,930
64	8,980	40,034	21,133	59,974

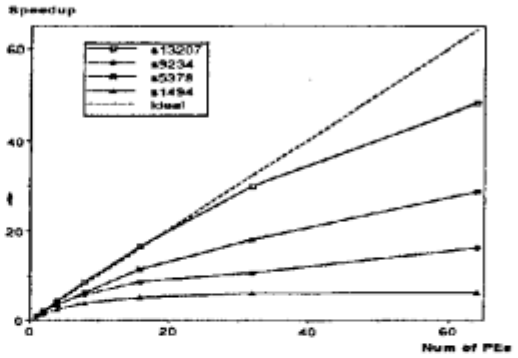

図 4: 速度向上

表 3: ロールバック頻度と深さ (64PE 使用時)

回路	s1494	s5378	s9234	s13207
f_r	0.6090	0.0696	0.0562	0.0227
d_r	2.700	3.691	11.107	7.310

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示す。また、図 4に速度向上のグラフを示す。

表 3には、64PE 使用時のロールバックの頻度 f_r と平均的深さ d_r を示す。f_r, d_r は、以下の式で定義する。

$$f_r = R/E$$

$$d_r = H_r/R$$

ここで、R: ロールバック発生回数、E: 真のイベント数 (最終的に巻き戻されなかったメッセージ数)、H_r: 巻き戻された履歴数である。

表 4には、64PE 使用時の PE 間メッセージ通信の頻度 f_c を示す。f_c は、以下の式で定義する。

$$f_c = M_c/M_{all}$$

ここで 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。

さらに、バーチャルタイム法における処理プリミティブのうち、以下に示すものについて処理時間を計測した。

GVT 更新処理

使用プロセッサ数を変えて、GVT 更新に必要な処理時間を計測した。表 5にその結果を示す。なお、各プロセッサでのシミュレーション時刻はメッセージスケジューラが管理しているため、GVT 更新処理時間は対象回路に依存しない。

履歴解放処理

一回の GVT 更新後の履歴解放処理時間 t_f は、ゲートオブジェクト数 G と解放履歴数 h_f に比例する。

$$t_f = k_f h_f + k_g G + C_f \qquad (1)$$

ここで C_f は定数分である。計測の結果、$k_f = 0.027$ (ミリ秒)、$k_g = 0.302$ (ミリ秒)、$C_f = 0.036$ (ミリ秒) であった。

ロールバック処理

一回のロールバック発生時の履歴巻き戻し処理時間 t_r は、巻き戻された履歴数 h_r に比例する。

$$t_r = k_r h_r + C_r \qquad (2)$$

ここで C_r は定数分である。計測の結果、$k_r = 0.027$ (ミリ秒)、$C_r = 0.367$ (ミリ秒) であった。

PE 間メッセージ通信

本システムでの一つのメッセージ (20 バイト) の PE 間通信にかかる処理時間は 0.256 ミリ秒であった。

4.2 考察

64PE 使用時の速度向上として、s13207, s5378 については良好な結果が得られたが、s9234, s1494 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

表 4: PE 間メッセージ通信頻度 (64PE 使用時)

回路	s1494	s5378	s9234	s13207
f_c	0.4016	0.1206	0.08458	0.02219

表 5: GVT 更新処理時間

PE 数	GVT 収集処理時間 (1 回あたり、ミリ秒)
8	6.286
16	13.598
32	23.851
64	49.168

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 5より、GVT 更新 1 回あたりの処理時間は 64 PE 使用時でも約 49 ミリ秒と非常に短いことがわかる。GVT 更新頻度は低いため、GVT 更新処理はシステム全体の性能にはほとんど影響しないと考えて良い。

さて、PE i 番での履歴解放に関わる処理時間 $T_f(i)$ 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

$$T_f(i) = \sum_{i=1}^{n} t_f(i) = k_f E_i + n k_g G_i + n C_f \quad (3)$$

ここで、E_i は PEi での真のイベント数であり、G_i は PEi に割り当てられたゲート数である。E_i および、G_i は使用 PE 数にほぼ反比例すると考えられる。

GVT 更新の目的はメモリ消費を少なくすることである。各 PE は等量のメモリを持っているとすると、GVT 更新回数 n も使用 PE 数にほぼ反比例すると考えられる。結局、式 (3) 右辺 nk_GG の項は使用 PE 数の 2 乗にほぼ反比例する。いま、各 PE に均等にゲートが分配され、均等にイベントが発生したとする。s13207 での真のイベント数は 2,341,374 であり、2PE 使用時の GVT 更新回数は 62 回であった。以上から、2PE 使用時の $k_f E_i$ の値はほぼ 31.6 秒、nk_GG_i の値は 115.6 秒 、nC_{t_f} の値は 0.002 秒と見積もれる。したがって $T_f(i)$ に対する nk_GG の項の影響は無視できない。

図 4において、2PE~16PE 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、これは使用 PE 数が増えるに連れ、各 PE での履歴解放に要する処理時間が短くなるためと考えられる。

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全て無視できるような場合を考える。

論理シミュレーションの問題 Prb は、回路構造 c、入力信号列 v、及びシミュレーション期間 t によって決定される。

$$Prb = Prb(c, v, t)$$

このとき、問題 Prb を PE N 台への負荷分散方法 $Dst(N)$、スケジューリング戦略 Sch を用いて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 $P_{[Prb]}$ を

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

と定義する。ここでの負荷分散方法 $Dst(\infty)$ は各ゲートオブジェクトを全て異なる PE に割り当てるものである。また、スケジューリング戦略 Sch_{ts} は、タイムスタンプソート戦略とする。タイムスタンプソート戦略は、全てのゲートオブジェクトが一つの PE に存在する場合、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。$P_{[Prb]}$ は、ゲート数以上の 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、しかもゲートオブジェクト数より遥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回路を分割し、部分回路を PE に割り当てる負荷分散方法をとる。

PE N 台への負荷分散方法 $Dst(N)$、スケジューリング戦略 Sch を用いた場合の問題の並列度 $P_{[Prb,Dst(N),Sch]}$ を、

$$P_{[Prb,Dst(N),Sch]} = \frac{T(Prb, Dst(1), Sch_{ts})}{T(Prb, Dst(N), Sch)}$$

と定義する。

各メッセージの処理コストは全て等しいと仮定し、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 $P_{[Prb]}$ 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 64PE に負荷を分散し、スケジューリング戦略としてタイムスタンプソート戦略を用いた場合の、各問題についての並列度 $P_{[Prb,Dst(64),Sch]}$ も実験的に求めた。

さらに、一回の 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

表 6: 並列度

回路	s1494	s5378	s9234	s13207
$P_{[Prb]}$	55.65	298.6	487.7	609.3
$P_{[Prb,Dst(64),Sch]}$	18.88	35.52	17.95	43.24
$S_{[T_f=0]}$	5.83	25.46	13.47	38.17

び式 (3) を用いて、履歴解放処理時間を除去した場合の 64PE 使用時の速度向上 $S_{[T_f=0]}$ を計算した。表 6 に、$P_{[Prb]}$, $P_{[Prb,Dst,Sch]}$, $S_{[T_f=0]}$ の値を示す。

表 6から、s1494 に関しては、問題の持つ並列度 $P_{[Prb]}$ 自体が非常に小さいことが分かる。また、s5378, s9234, および s13207 については、負荷分散及びスケジューリング戦略による並列度が速度向上を制限していることがわかる。とくに、s9234 については、負荷分散方法及びスケジューリング戦略によって極端に並列度が抑えられてしまっている。

表 6は、全てのメッセージ処理のコストが等しいと仮定していているため、必ずしも現実と完全に一致するものではない。しかし、問題自身の並列性、および負荷分散戦略、スケジューリング戦略による問題の並列度低下が、速度向上を抑える最大の原因であることが予測できる。とくに、負荷分散戦略については、改良の余地が大きいと考えている。

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全て処理速度を低下させているわけではない。一般にロールバックは、シミュレーション時刻が将来に進み過ぎている PE にのみ発生する。

極端な例として、シミュレーション時刻が常に遅れている PE が存在した場合を考えてみる。この場合、ロールバックは先行した PE においてのみ発生し、決して全体の処理を遅らせることはない。

s13207 の場合、平均ロールバック処理時間は表 3 および、式 (2) から、0.469 ミリ秒となる。一方、履歴解放処理時間を除去した場合の、1 メッセージあたりの処理時間は 0.618 ミリ秒である。ロールバック発生頻度 f_r は 0.0227 と非常に低いことから、s13207 の場合、ロールバック処理は全体の処理にほとんど影響を与えないと考えられる。s1494 については、平均ロールバック処理時間 : 0.440 ミリ秒、1 メッセージあたりの処理時間 : 0.647 ミリ秒であり、ロールバック発生頻度 f_r が 0.6090 と高いため、ロールバック処理はある程度全体の処理に影響を与えていると思われる。しかし、実際にシミュレーションの進行を遅らせているものがどの程度かを正確に計測することは非常に難しい。

4.2.4 PE 間通信

表 4より、s13207, s9234 の場合、PE 間通信頻度が比較的低く抑えられているが、s1494 ではかなり PE 間通信頻度が高くなっていることがわかる。一般に、プロセッサ数が一定の環境では、ゲート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からわかるように、s1494 は比較的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 間通信が多くなるのはこれらの理由によるものであると考えられる。

PE 間通信がシミュレーション性能に与える影響については、s13207, s9234 の場合、比較的小さいと考えて良かろう。しかし、s1494 では、シミュレーション性能を低下させる要因の一つであると考えられる。

以上の考察から、速度向上が悪いものについては、負荷分散およびスケジューリング戦略による並列性低下がその主な原因であり、s1494 の場合を除いて、ロールバック処理、および PE 間メッセージ通信コストの影響は比較的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、ほとんどの場合、問題の持つ並列性を比較的効率的に引き出していると考えられる。

5 時刻管理機構の比較

本節では、絶対性能の観点から、他の時刻管理機構とバーチャルタイム法との比較を行い、各時刻管理機構の有効性の検討を行う。

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 [2]。

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考える。各オブジェクトは、自身の全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。そして、各入力信号線上にメッセージが到着後、最小のタイ

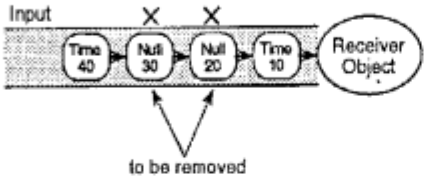

図 5: ヌルメッセージ削減機構

表 7: コンサーバティブ法 (回路 s13207)

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

ムスタンプ値を持つメッセージに対して処理を行う。その後、再びメッセージの待ち合わせを行う。

5.1.2 ヌルメッセージ削減機構

コンサーバティブ法で最も大きな問題となるのが、デッドロックの発生である。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有効な方法であるが、現実にはヌルメッセージが非常に多く発生し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 [2]。

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、更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、負荷分散方法およびスケジューリング戦略は第 4節と同様のものである。表7に、各々の計測結果を示す。

ヌルメッセージ数

表7から、発生したメッセージのほとんどがヌルメッセージであったことがわかる。

ヌルメッセージ削減機構のない場合、一つのメッセージ受信毎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、及び各ゲートでの遅延値が離散的であるために上限が存在する。このメッセージ数の上限は (シミュレーション時間長) × (全信号線数) で与えられる。

s13207 の信号線数は 19,983 である。また実験では期間 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 64PE 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

速度向上の面からみれば良好な値を得ることができている。しかしながら、絶対性能は非常に悪い。

PE1 台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を凌ぐ。それにもかかわらず、コンサーバティブ法によるシミュレーション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。

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、タイムスタンプソート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

タイムホイル方式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 8: タイムホイル法 (回路 s13207)

PE 数	性能 (イベント / 秒)
1	2,261
2	4,210
4	8,307
8	13,756
16	19,748
32	18,493
64	11,320

5.2.3 計測結果および考察

実験の対象回路として s13207 を用いた。負荷分散方法はやはり第 4節の方法を用いた。

計測結果を表 8に示す。

1PE 使用時の処理性能は 2,261 イベント / 秒であり、バーチャルタイム法に比べて高速である。これは、履歴保存処理及びロールバックへの対処が不要になるためである。

複数 PE を使用した場合の性能については、8PE 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ている。そして、16PE 使用時で 19,748 イベント / 秒という最高値を示している。しかしながら、32PE 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 全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 の二つが考えられる。

タイムホイル法は、使用 PE 数が少なく、且つシミュレーション時間を粗く離散化する場合に限れば、比較的良好な性能を示すと考えられる。

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、64PE 使用時に約 60k 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、十分な速度向上及び性能を得ることができなかったものについては、問題に十分な並列性がない、或いは、静的負荷分散により、十分な並列性を引き出せないことが主な原因であると考えられる。

さらに、ヌルメッセージを用いたコンサーバティブ法、及び、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、バーチャルタイムとの性能比較を行った。コンサーバティブ法は、速度向上は良いものの、ヌルメッセージの発生量が非常に大きく、絶対性能の面からはバーチャルタイムに遥かに劣るものであった。また、タイムホイル法は、使用 PE 数が少ない場合にはバーチャルタイム方式を凌ぐ性能を示したが、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、バーチャルタイム法は、並列論理シミュレーションの方法としては最も現実的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。

参考文献

[1] D.R.Jefferson, "Virtual Time", ACM Transactions on Programming Languages and Systems,Vol.7,N0.3, 1985, pp. 404-425

[2] J.Misra, "Distributed Discrete-Event Simulation", ACM Computing Surveys,Vol.18,No.1, 1986, pp. 39-64

[3] K. Taki, "The parallel software research and development tool: Multi-PSI system", Programming of Future Generaiton Computers, North-Holland, 1988, pp. 411-426

[4] L. Soule', T. Blank, "Parallel Logic Simulation on General Purpose Machines", Proceedings of 25th Design Automation Conference, 1988, pp. 166-170

[5] 下郡慎太郎, 鹿毛哲郎, "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, CAS88-110, 1989, pp. 23-30

[6] 福井 眞吾, "バ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, 1989, pp. 1547-1554

[7] R.M.Fujimoto, "Parallel Discrete Event Simulation", Communications of the ACM, Vol.33, No.10, 1990, pp. 30-53

[8] K. Ueda, T. Chikayama, "Design of the Kernel Language for the Parallel Inference Machine", The Computer Journal, Vol.33, No.6, 1990, pp. 494-500