並列論理シミュレーションにおけるロールバックの削減のための回路分割

世古 忠† 菊野 亨††

†奈良工業高等専門学校 ††大阪大学基礎工学部

本論文では, Jefferson の楽観的手法を用いた並列論理シミュレーションにおいて発生するロールバックの回数を削減するための新しい回路分割法について述べる. 提案する回路分割法では評価可能なゲートに注目してクラスタ (ゲートの集まり) を作り, 次にそれらをまとめてなるべく均等なサイズのクラスタに分割する. ベンチマーク回路を用いて行った適用実験の結果, 提案する回路分割は従来の回路分割と比べてロールバックの発生回数をかなり削減できることが分かった.

In this paper, we propose a new circuit partitioning method to reduce rollbacks in parallel logic simulation based on Jefferson's optimistic method. The proposed method constructs several clusters by searching evaluable gates(that is, gates which are currrently evaluable by the parallel logic simulation), and then merges too small clusters into a large cluster. The experimental evaluations using benchmark circuits show that the proposed method realizes a significant reduction of rollbacks compared with conventional methods.

1 はじめに

並列論理シミュレーションを Jefferson の楽観的手法 [2] を用いて行う場合に発生するロールバックは, 並列処理性能を低下させる大きな要因の1つと考えられ、いかにしてロールバックの発生回数を減らすかが重要な問題になっている.

これまでに松本と瀧は ICOT が開発した新しいアーキテクチャをもつ MIMD 型のマルチプロセッサシステム上で並列論理シミュレーションに関する多くの実験を行い、ロールバックに関する報告を行っている [4][5].

一方, 著者らはロールバック発生のメカニズムを解明するため, 木状回路を対象として、その発生要因についての分析を行ってきた [6]. その結果, ロールバックの発生数を削減するためには回路分割法が重要であることが分かった.

並列論理シミュレーションにおけるこれまでによく知られた回路分割法としては, タイムホイル法を前提とした Agrawal の方法 [1], 及び楽観的方法を前提とした松本と瀧による方法 [4] がある. 文献 [4] の部分回路の作成法は、いわゆる木探索の要領で縦方向に連結したゲートをクラスタ (1台のプロセッサで処理すべきゲートの集合) としてまとめる. それに対して本論文では評価可能なゲートに特に注目してクラスタを作ることを試みる. 提案手法の適用実験の結果, 一部の回路に対してロールバックの削減効果が認められた.

本論文で得られた主な成果は, 並列論理シミュレーションを議論するためのモデルの記述, ロールバック発生要因についての分析, ロー

A Circuit Partitioning Method for Reducing Rollbacks in Parallel Logic Simulation

†Tadashi SEKO, Nara National College of Technology.

††Tohru KIKUNO, Faculty of Engineering Science, Osaka University.

ルバックを削減するための新しい回路分割法の提案, である.

以下, 本論文の第2章ではモデルとロールバックについて説明し、従来の回路分割法と新しく提案する回路分割法について述べる. 第3章ではシミュレーションの手順, 及び, シミュレーション時間の構成要素について述べた後, ロールバックの発生要因を明らかにする. 第4章では, 適用実験の結果について述べる.

2 準備

2.1 モデル

並列論理シミュレーションはマルチプロセッサ上で行われると仮定する. シミュレーションシステムは1台の制御用のプロセッサとn(n≥2)台の計算用のプロセッサから構成される. 制御プロセッサは並列論理シミュレーションのために, (1) 負荷分散を目的とするプロセッサのスケジューリングと, (2) 同期を目的とするメッセージスケジューリングの2種類のスケジューリング機能をもつ.

一方, 各計算プロセッサには, 回路分割法に従って, あらかじめゲートが静的に割り当てられているものとする. 各計算プロセッサでは割り当てられたゲートの評価を実行する. また, 計算プロセッサではゲートの評価と同時に, イベント (ゲートへの入力値が変化したことを知らせるためのメッセージ) の送受信処理も行えると仮定する. 更に, 各プロセッサはイベントのスケジューリングのためのイベント管理表をもつと仮定する.

2.2 ロールバック

ここでは図1を用いてロールバックの直観的な説明をする. 今, 2つのゲート G, G' はそれぞれプロセッサ P_1, P_2 に割り当てられていると仮定する. また, ゲート G における局所時刻 (t_L で表す) は既に15になっていると仮定する. 更に, イベント $[n_2, 8, 0]$ がプロセッサ P_2 からプロセッサ P_1 へ送信されるものとする. ここで, $[n_2, 8, 0]$ は, 信号線 n_2 の信号値が時刻8で0に変化することを表す.

このとき, ゲート G でイベント $[n_2, 8, 0]$ を評価するためには, ゲート G の局所時刻を8以前の時刻まで巻き戻す必要がある. 更に, それに伴って8から15の間の時刻をもつ全てのイベントを再計算する必要がある. 局所時刻の巻き戻しに関連するこうした一連の操作をロールバックという.

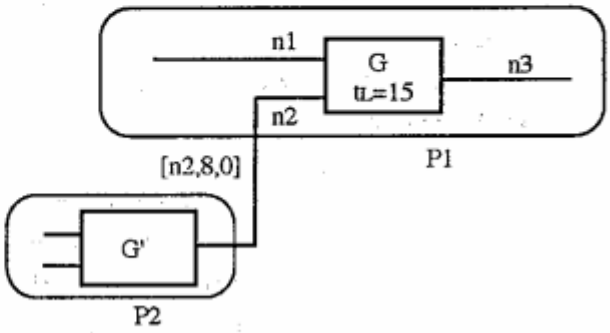


図1 ロールバックの説明図

2.3 従来の回路分割

並列論理シミュレーションのための回路分割においては、負荷の均等配分、通信回数の削減、並列性の抽出などがその目的とされる. これらをかなり満足する発見的手法が松本と瀧 [4] によって提案されている. その回路分割法では, まず回路をいわゆる木探索の要領で探索し、幾つかの部分回路 (クラスタ) に分割する. 次にそれらのクラスタをまとめてなるべく均等なサイズの回路に分割する. 最後に, こうして求まったクラスタをランダムにプロセッサに割り当てる.

この方法を用いてクラスタを作成する場合, 接続先ゲートを多くもつ回路に対しては, 最初の段階でゲート数の少ないクラスタが多く作られるため, クラスタをまとめる段階で, 適切な選択を実行することが難しくなる.

[例 1] 図 2(a) は従来の分割法により,3 台のプロセッサ P_1, P_2, P_3にゲート割り当てを行なった例を示す. 同図では、はじめに, ゲート 1 から探索を開始して, 次の 5 個のクラスタ C_1={1,4,6,9,12},C_2={13,15},C_3={14}, C_4={7,10}, C_5={8,11} を作る. 次に, ゲート 2 を開始点として C_6={2,5} を作り, 最後にゲート 3 を開始点として C_7={3} ができる. これらのクラスタを例えば $C_1 \cup C_4$, $C_2 \cup C_6$, $C_3 \cup C_5 \cup C_7$とまとめて, 3 台のプロセッサに割り当てると図 2 (a) の割り当てが求まる.

2.4 新しい回路分割

ここでは評価可能なゲートに特に注目して, クラスタを作ることを提案する. 提案するクラスタ生成アルゴリズムの概要を以下に示す. ここでは, あるゲートのすべての入力が, 既にクラスタに属するゲートの出力と接続されているとき, そのゲートを評価可能ゲートと呼ぶ.

クラスタ生成アルゴリズム

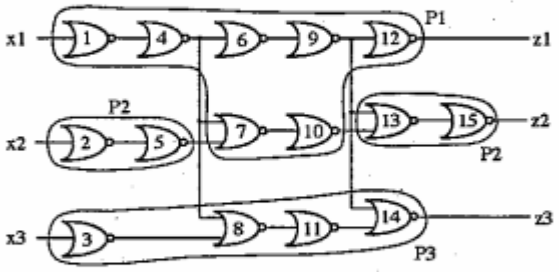
Step1: すべての入力端子が外部入力に接続しているゲートを 1 つのクラスタとする.

Step2: クラスタ (すなわち, 今の場合はゲート) を 1 つ選択して, クラスタリングの開始点とする.

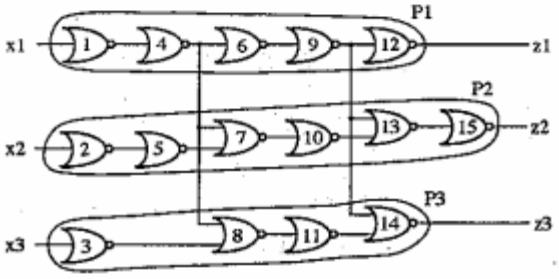
Step3: 評価可能ゲートがあればそれをクラスタに取り込む. なければ、他のクラスタを 1 つ選択して操作を続ける.

Step4: Step3 の終了時点で, まだどのクラスタにも属していないゲートがあれば、そのゲートを開始点として, 操作を続ける. すべてのゲートがいずれかのクラスタに属するようになれば終了する.

[例 2] 図 2(b) は新しい分割法を適用してゲートを割り当てた例を示す. 同図では, はじめに, ゲート 1 から探索を開始してクラスタ C_1'={1, 4, 6, 9, 12} を作る. 次にゲート 2 を開始点として, クラスタ C_2'={2, 5, 7, 10, 13, 15} を作る. 最後にゲート 3 を開始点として, クラスタ C_3'={3, 8, 11, 14} ができる. これらのクラスタをプロセッサ P_1, P_2, P_3に割り当てた例を図 2(b) に示す.



(a) 従来法



(b) 提案する方法

図 2 回路分割

3 並列シミュレーション

3.1 シミュレーションの手順

ここでは Jefferson の楽観的手法に従った並列論理シミュレーションを採用する. 入力端子における信号値が確定する前に, つまり見込み (つまり, 以前の時刻における信号値を代用すること) でゲート評価を行なう.

次の Step1-Step6 に各計算プロセッサ P_iでのイベント評価の手順を述べる. 回路分割の仕方によって, Step3 で選択する (従って, Step4 及び Step5 で計算すべき) ゲート, 及び, 新しく生成されるイベントが変わる. 最終的には,

Step3 におけるロールバック判定の結果に影響が及ぼされる.

Step1:(シミュレーション時刻の更新) …プロセッサ P_iのイベント管理表が空でなければ, P_iの物理時刻を1単位時刻だけ進める.

Step2:(イベントの選択) ・・・ イベント管理表から最小時刻のイベント $[n, t_s, x]$ を取り出す.

Step3:(ロールバックの判定) ・・・ P_iに割当てられるゲートの集合から, ネット n を入力端子とするものを1つ選ぶ. それを G とし, その局所時刻を t_Lとする. もし $t_s < t_L$ならば, ロールバック処理を行いその処理の終了後 Step1 へ行く. そうでなければゲート G の局所時刻を t_sに更新し, 次は Step4 へ行く.

Step4:(イベントの評価) …イベント $[n, t_s, x]$ の評価を行う. その結果, ゲート G の出力の値が変化すれば, 新しいイベントを生成する. Step4 で値が変化したネットを n'とする.

Step5:(新しいイベントの処理) ・・・ ネット n'を入力端子とするゲートを G'とする. もし G'がプロセッサ P_iに割当てられていれば, 新しいイベントをイベント管理表に登録する. 一方, G'が他のプロセッサ P_j(j≠i) に割当てられていれば, イベントを P_jに転送する.

Step6:(イベントの受信) …Step1 から Step5 の実行中に, 他の複数のプロセッサ P_k(k≠i) から新しいイベントを受信しておれば, これらのイベントをイベント管理表 E_i に登録する. 次は Step 1へ行く.

3.2 シミュレーション時間

並列論理シミュレーションにおけるシミュレーション時間を構成する要素には次の4つがある [6].

- τ_{eval}… イベント1個当たりの評価時間
- τ_{oh}… ロールバック1回当たりのオーバヘッド時間
- τ_{idle}… プロセッサの遊休時間
- τ_{com}… プロセッサ間でのイベントの送受信に要する時間

4. では以上の基本時間をすべて計測して, シミュレーションの評価を行う.

3.3 ロールバックの要因

ロールバックの発生の要因としては多くのものが考えられる. ここでは, ロールバック発生の主な要因として次の5つを指摘しておく.

(1) 与えられる回路の特性... 回路を構成するゲートの種類, ゲートの接続パターン.

(2) 与えられる入力信号の特性... 入力信号の値の変化パターン (変化の回数, 時間間隔). 複数種類の入力信号がある場合には, それらの関係.

(3) 回路の分割π... 与えられた回路を n(計算プロセッサの台数) 個のクラスタに分割する方法.

(4) 入力信号の回路への割当てσ... 与えられた回路の各入力端子に対する入力信号の割当.

(5) 処理速度τ_{eval}と通信速度τ_{com}... 計算プロセッサが1個のイベントを処理するのに要する時間τ_{eval}と計算プロセッサ間でイベントを転送するのに要する時間τ_{com}の値 (あるいは、その値の比).

実際には (1)-(5) の要因が (単独ではなく) 相互に関連し合ってロールバックを引き起こしていると予想される. そこで, 本論文では特に (3) に注目して並列論理シミュレーション実行時のロールバック発生について考察する.

表1: 実験結果

Circuit name	Partition 1		Partition 2	
	Number of rollbacks	Simulation time[τ]	Number of rollbacks	Simulation time[τ]
9symml	243	6261	113	6103
C432	568	10039	136	9509
C499	115	18631	241	18780
C880	486	15835	149	15497
C1355	1577	23458	1673	23360
b9	320	4402	125	4213
apex7	229	8747	341	9291
f51ml	171	5198	88	5113

4 適用実験

4.1 実験データ

提案する回路分割法の有効性を調べるために、シミュレーションモデルに基づく評価用のプログラムを作成して, ベンチマーク回路 [3] を用いてプロセッサ 3 台の場合について実験を行った. 外部入力端子には同一の規則的な信号を加えた. 実験結果を表 1 に示す. この表で分割法 1 は従来の回路分割法を、分割法 2 は提案する分割法を表す.

表 1 より実験に用いた 8 個のベンチマーク回路の中で 9symml,C432,C880,b9, f5lml の 4 つについては分割法 2 のロールバックの発生回数とシミュレーション時間がともに、分割法 1 と比べて値が小さいことが分かる. また, C1355 の場合, ロールバックの発生回数は分割法 1 の方が少ないが, シミュレーション終了時間は分割法 2 の方が少なかった. その他の 2 つの回路 (C499, apex) の場合は, 分割法 1 のロールバック発生回数とシミュレーション時間がともに分割法 2 に比べて少なかった.

4.2 考察

実験の結果、提案した回路分割法は, ゲート当たりの平均の入力線数及び出力線数が多い回路に対して、ロールバック発生回数の削減効果が見られた.

例えば, 提案した回路分割法 2 の方がロールバック発生数の少なかった回路 9symml の平均の入力線数及び出力線数の値は, それぞれ 2.39 及び 1.7 であった. また, 従来の回路分割法の方がその発生回数が少なかった回路 C499 の値はそれぞれ 1.72 及び 1.53 であった.

5 まとめ

本論文では, 楽観的方法を用いて並列論理シミュレーションを行う場合について, ロールバックの発生回数を削減するための回路分割法を新しく提案し, ベンチマーク回路を用いて評価実験を行った. その結果, 提案した回路分割法は, ゲート当たりの入力線及び出力線が多い回路に対して、ロールバックの削減効果が認められる方法であることが分かった.

今後は, 提案した回路分割法を利用して種々の特性を有するより多くの回路について適用実験を行なう計画である. それと同時に, 入力信号, 及び, 回路の特性を考慮した新しい回路分割法についても検討を進める予定である.

謝辞 本研究を進める上で貴重なご意見と温かいご激励をいただいた神戸大学 瀧 和男助教授, 三洋電機 松本幸則氏に深謝致します. また, 評価実験でご協力いただいた奈良高専専攻科西野 繁之氏に深謝致します.

参考文献

[1] P. Agrawal : "Concurrency and communication on hardware simulators", IEEE Trams. Computer-Aided Design, Vol. CAD-5, No.4, pp. 617-623(1986).

[2] D. R. Jefferson : "Virtual time", ACM Trans. on Programming Language and Systems, Vol.7, No.3, pp.404-425(1985).

[3] R. Lisanke:"Logic synthesis and optimization benchmarks user guide version 2.0", Technical report, Microelectronics Center of North Carolina(1988).

[4] 松本, 瀧 : "バーチャルタイムによる並列論理シミュレーション", 情処学論, Vol. 33, No.3, pp.387-395(1992).

[5] Y. Matsumoto and K. Taki : "Parallel logic simulation on a distributed memory machine", Proceedings of 1992 European Conference on Design Automation, pp.76-80(1992).

[6] T. Seko, S. Nishino,T. Kikuno: "Experimental Evaluation of Rollbacks in Parallel Logic Simulation for Tree-Connected Circuit" Proc. of JTC-CSCC'94,pp. 877-882(1994).