RBCQ同期機構およびその同期方式の提案と性能評価

早川 潔[†] 本多弘樹[††]

近細粒度並列処理で，先行制約を保証する同期を，高速にかつ不必要な待ち時間なく行うための同期機構・方式の1つとして，本論文では，バリアキューによるブロッキングを利用してタスク間の先行制約を保証するRBCQ同期機構とその同期方式を提案する．RBCQ同期機構は，簡潔な回路で構成されているので，高速な同期動作が可能である．また，RBCQ同期方式は，ブロッキングを利用する他の同期方式（One-PE同期方式）に比べ，不必要なブロッキングが少なく，バリア参加情報の数も少なくできるという特徴を持つ．RBCQ同期機構を近細粒度並列処理のテストベッドOPASにインプリメントし，評価プログラムを用いて性能評価を行った．その結果，RBCQ同期方式を用いたプログラムの処理速度が，共有メモリのフラグ変数をチェックする同期方式を用いたプログラムの処理速度より42%向上した．

A Proposal and Evaluation of The RBCQ Synchronization Mechanism and Compile Method

KIYOSHI HAYAKAWA[†] and HIROKI HONDA[††]

Several hardware synchronization mechanisms have been proposed to reduce synchronization overhead on fine-grain parallel processing. In this paper, we propse the RBCQ synchronization mechanism and method. The mechanism and method allow to ensure the order constraint between tasks by using barrier blocking. The RBCQ synchrnization method is less sychronizer-dependent precedences than the other method using barrier blocking (One-PE synchronization method). And the RBCQ synchronization mechanism is less circuit delay time, because of simpler circuits. We imprement the RBCQ synchronization mechanism on OPAS. And we also evaluate the RBCQ synchronization method. As the results, the RBCQ synchronization method is 42% faster than shared memory base (flag check) synchronization method.

1. はじめに

密に結合されたマルチプロセッサのプロセッサ間同期オーバヘッドを少なくするために，種々のハードウェア同期機構・方式☆が提案されている[1),2),6),7),9),11)]．これらの同期機構の多くは，バリア同期を一部拡張した同期動作を行う同期機構（バリア型同期機構）である．本論文では，近細粒度並列処理で，タスク間の先行制約を保証するためのバリア型同期機構について議論する．

バリア型同期機構の同期ポイントの指定方法として，命令フィールド中に同期ポイントを示すタグを設ける方法と同期ポイントを示す専用命令（同期コード）を並列オブジェクトコード内に配置する方法が考えられている[6)]．

マルチプロセッサを汎用プロセッサで構成する場合，同期コードによる方法のほうがタグによる方法より簡単に構成できる．同期コードによる方法の場合，同期機構依存先行制約[8)]による不必要な待ち時間の削減とともに同期コード数の削減も重要となる．

同期コード数が少ない同期機構として，SBM同期機構[7)]があげられる．SBM同期機構の線状&任意参加バリア[12)]による実行制御を用いて先行制約を保証する場合，バリアを通過しようとするプロセッサに不必要な待ち時間が生じる場合がある．不必要な待ち時間は，線バリアによるものとブロッキング[7)]によるものに分けられる．

SBM同期機構上で，線バリアによる不必要な待ち

† 電子技術総合研究所
Electrotechnical Laboratory
†† 電気通信大学大学院情報システム学研究科
Graduate School of Information Systems, University of Electro-Communications

☆ ここでいう同期機構とは，同期処理回路そのものを意味し，同期方式とは，同期機構を使用して先行制約を保証するための方法を意味する．

時間がない同期方式として，One-PE 同期方式[2]が提案されている．しかし，One-PE 同期方式でもブロッキングによる不必要な待ち時間が生じてしまうという問題がある．

そこで，本論文では，不必要なブロッキングが少ない同期機構として，RBCQ☆同期機構とその同期方式を提案する．RBCQ 同期機構は，簡潔な回路で構成されているので，高速な同期動作が可能である．また，One-PE 同期方式に比べ，バリア参加情報（後述）の数を少なくできるという特徴を持つ．

RBCQ 同期機構・方式を性能評価するために，RBCQ 同期機構を近細粒度並列処理のテストベッド OPAS に実装し，また RBCQ 同期方式を並列化 C コンパイラに組み込んだ．この並列処理システムで，評価プログラムを実行し，他の同期方式と RBCQ 同期方式とを比較することにより，RBCQ 同期方式の有効性を検証する．

なお，本論文では以下の事項を前提とする．

- ターゲットマシンはプロセッサ台数が十数台の共有メモリ型マルチプロセッサとする．
- 汎用プロセッサでのインプリメントの容易さを考え，同期コードによる同期ポイント指定方式を採用する．
- 並列処理方式は，逐次プログラムの基本ブロックを近細粒度タスクに分割し，そのタスク間の並列性を利用して並列処理[4]するものとする．
- 並列化コンパイラが，コンパイル時に，各プロセッサ用のオブジェクトコードを生成する．
- 各プロセッサはそれぞれ独自のクロックで動作しているため，タスクの実行時間を正確に見積もることができない．ただし，見積り実行時間と実際の実行時間のずれは比較的短い時間（1〜2 タスクの実行時間）とする．

2. キュー型同期機構と不必要なブロッキング

2.1 キュー型同期機構の基本構成

キューを用いてバリア参加情報間の全順序関係を保証する同期機構をキュー型同期機構と呼ぶことにする．ここでいうバリア参加情報とは，同期に参加するプロセッサをビットベクトルで表した情報である．

キュー型同期機構のハードウェア構成を図 1 に示す．

図 1 のバリアキューでは，各縦の列が 1 回分の同期に対応するバリア参加情報を示す．バリア参加情報中の各ビットは各プロセッサ（PE）に対応し，ビットを

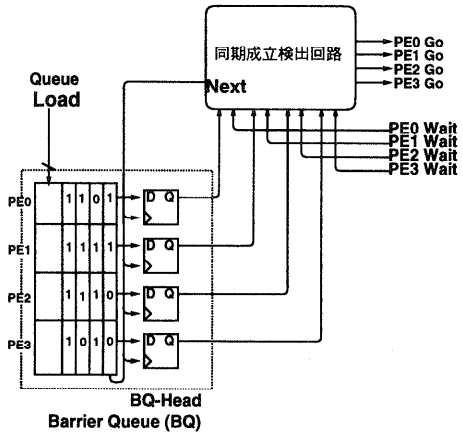


図 1 キュー型同期機構のブロック図
Fig. 1 The block diagram of the que type synchronization mechanism.

1 とすることで対応するプロセッサが同期に参加することを指定する．

PE Wait 信号は，その信号に対応したプロセッサが同期コードに到達し，待ちの状態であることを示す信号であり，PE Go 信号は，待ち状態になっているプロセッサに，同期成立の検出を通知する信号である．プロセッサは，同期コードを実行する際，PE Wait 信号を送出し，PE Go 信号が返されるまで，待ち状態となる．

同期成立検出回路は，PE Wait 信号とキューの先頭に格納されているバリア参加情報を基に，同期検出を行う．その同期検出状況を基に，PE Go 信号および Next 信号の送出を行う．

キュー型同期機構の代表的なものとして，SBM 同期機構[7]が提案されている．SBM 同期機構でのプロセッサの待ち状態は，線バリアによるものとブロッキングによるものとに分かれる．

線バリアによる待ち状態はバリアに参加する全プロセッサのいっせいの待ち状態なので，先行–後続タイプの待ち状態（後続タスクは先行タスクを待つが，先行タスクは後続タスクに同期ポイントの通過を知らせるだけで待たない）として適用すると，不必要な待ち状態が生じてしまう可能性が高い．

そこで，ブロッキングによる待ち状態を先行–後続タイプの待ち状態に適用する同期方式として，One-PE 同期方式が提案されている．

2.2 ブロッキングによる実行順序制御

ブロッキングとは，プログラム実行中，バリアキュー内の 2 番目以降の同期に参加しているプロセッサが，先頭のバリアに参加しているプロセッサの同期コードの

☆ Reduced Blocking and Compressed barrier Queue の略

実行終了を待つ現象をいう．ブロッキングは，キュー型同期機構で，バリア参加情報に対応している同期コード間に，順序付けをしているために生じる現象である．

このブロッキングの順序付けを利用して，タスク間の先行制約を保証する同期方式が，One-PE 同期方式である．One-PE 同期方式では，先行タスクに対応するバリア参加情報を先，後続タスクに対応するバリア参加情報を後の順で，バリアキューに投入することにより，タスク間の先行制約を保証する．また，プロセッサ単独で同期動作を行うことにより，線バリアによる待ち状態をなくしている．

2.3 不必要なブロッキング

One-PE 同期方式で生じるブロッキングには，タスク間の先行制約を保証するために必要なブロッキングだけではなく，不必要なブロッキングもある．不必要なブロッキングとは，ブロッキングによる同期コード間の順序付けが原因で，あるタスクを実行しようとするプロセッサが，そのタスクとの間に先行制約のないタスクの実行終了を待つ現象をいう．不必要なブロッキングは，元来，先行制約が必要ないタスク間に，同期機構依存先行制約がついてしまったために生じる．

たとえば，**図 2** の実線円間の実線矢印の先行制約を One-PE 同期方式で保証することを考える．ここで図 2 の円はタスクを表す．

タスク間の先行制約を保証するために，実線円のタスクを実行するプロセッサのみが参加することを示すバリア参加情報を生成し，タスクの見積り実行時間が早い順にそのタスクに対応したバリア参加情報をバリアキューに投入する．その場合，図 2 の網掛け範囲がバリア参加情報に対応し，網掛けの間に全順序関係が生じる．この全順序関係から実線円間に点線矢印の同期機構依存先行制約が生じてしまっている．

3. RBCQ 同期機構

不必要なブロッキングをなくすために，HBM 同期機構[7]を使用して One-PE 同期方式を行うことが考えられる．HBM 同期機構は，連想メモリを同期機構内に追加し，不必要なブロッキングを起こす原因となる同期コード間の順序付けをなくすことを可能にした同期機構である．

しかし，連想メモリとそれを制御する回路を組み込まなければならないので，ハードウェア量が増加する．連想メモリのレイテンシや，不必要なブロッキングを起こさないようバリア参加情報を連想メモリに投入する制御のオーバヘッドから高速な同期動作は見込めない．

そこで，不必要なブロッキングを少なくすることが可能な同期機構として RBCQ 同期機構を提案する．

RBCQ 同期機構は，複数プロセッサのグループ間でブロッキングし，グループ内ではブロッキングしない動作を可能にする．つまり，RBCQ 同期機構は「ある同期に参加しているプロセッサは，直前の同期に参加しているすべてのプロセッサが同期ポイントを通過しなければ，同期ポイントを通過することはできない」という動作を行う．

RBCQ 同期機構を利用することにより，不必要なブロッキングを削減できる．たとえば，図 2 の先行制約を RBCQ 同期機構を使用して保証する場合，**図 3** のように，One-PE 同期方式とは異なり，複数のプロセッサが同期に参加でき，さらにそれらのプロセッサ間には順序制御なくタスクを実行できる．よって，図 3 の同期機構依存先行制約は図 2 のそれより少ない．

3.1 RBCQ 同期機構での同期検出回路

RBCQ 同期機構の同期検出回路を**図 4** に示す．RBCQ 同期機構は，バリアキューおよび同期成立検

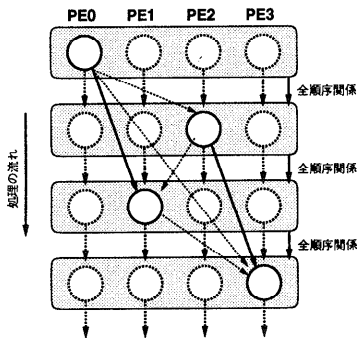


図 2 One-PE 同期方式の同期機構依存先行制約
Fig. 2 The synchronizer-dependent precedences of the One-PE synchronization method.

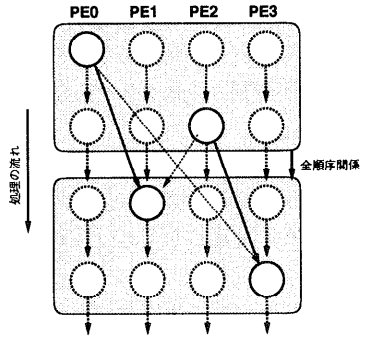


図 3 RBCQ 同期方式の同期機構依存先行制約
Fig. 3 The synchronizer-dependent precedences of the RBCQ synchronization method.

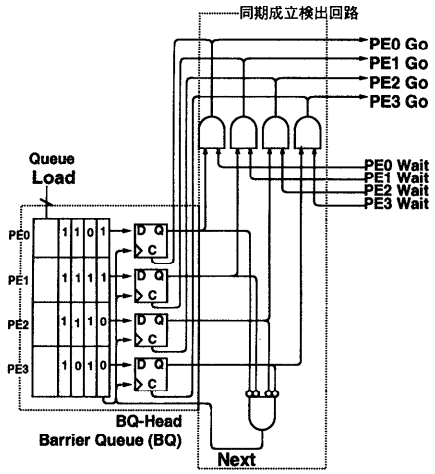


図4 RBCQ同期機構の回路（PE 4台）
Fig. 4 The circuit of the RBCQ synchronization mechanism (4 PEs).

出回路で構成される．同期成立検出回路内では，以下の処理が行われる．

- BQ-Headのバリア参加情報とプロセッサが送出するPE Wait信号とのANDをPE Go信号とする．
- これと同時に，そのPE Go信号に対するバリア参加情報ビットが格納されているBQ-Headのフリップフロップをクリアする．
- BQ-Headの内容がすべて0になった時点でNext信号を出力し，バリアキューを右にシフトさせ，次のバリア参加情報をキューの先頭に設定する．

3.2 RBCQ同期方式

RBCQ同期機構を使用してタスク間の先行制約を保証するための同期コード配置決定方法およびバリア参加情報列生成方法として，RBCQ同期方式を提案する．

RBCQ同期方式での同期コード配置は，先行タスクの直後および後続タスクの直前とする．また，各同期コードの見積り実行開始時刻をそれぞれ先行タスクの見積り実行終了時刻と後続タスクの見積り実行開始時刻とする．

RBCQ同期方式でのバリア参加情報列の生成手法では，同期コードをグループ化し，各グループ間の順序付けを行う．それを基にして，バリア参加情報列の生成を行う．

同期コードのグループ $S_i(i = 1,\ldots,n)$（n はグループ数）は，以下の条件の下で，各グループ内の同期コード数が最大になるように構成する．

- 同一グループ内には，同じプロセッサで実行される同期コードは存在しない．
- 同一グループ内の各同期コードに対応するタスク間に，先行制約がない．
- グループ S_i 内の同期コードの見積り実行開始時刻の集合を T_i とすると，T_i と T_{i+1} の間には，$\max T_i \leq \min T_{i+1}$ の関係が成り立つ．ただし，$\max T_i = \min T_{i+1}$ で，$\max T_i$ の同期コードに対応したタスクと，$\min T_{i+1}$ の同期コードに対応したタスクとの間に先行制約がある場合，先行タスクに対応した同期コードは S_i のグループに属し，後続タスクに対応した同期コードは S_{i+1} のグループに属している．

S_i 内の同期コードを実行するプロセッサのグループを P_i とし，P_i 内のプロセッサが参加することを示すバリア参加情報を生成し，それらを i の値が小さい順に並べ，バリア参加情報列とする．

先行タスクに対応する同期コードが属しているグループを S_i とすると，上記の条件により，その先行タスクと先行制約の関係にある後続タスクに対応する同期コードは，必ず S_{i+1} 以降のグループに属する．なぜなら，各タスクはリストスケジューリング[3]を用いてスケジューリングされているので，後続タスクの見積り実行開始時刻は先行タスクの見積り実行終了時刻以降にスケジュールされているからである．

RBCQ同期方式の同期コード配置の決定およびバリア参加情報列の生成手順を以下に述べる．

(1) スタティックスケジューリングにより決定されたプロセッサへのタスク割当て結果と先行制約から，文献5)の手法を使用し，同期が必要な先行制約を求める．

(2) (1)で求めた先行制約の先行タスクの直後と，後続タスクの直前に，同期コードを配置する．

(3) どの同期コードも登録されていないグループを作成し，そのグループを $S_j(j = 1)$ とする．

(4) グループに登録されていない同期コードの見積り実行開始時刻が最も早い同期コードを1つ選択する．ただし，見積り実行開始時刻の同じ同期コードが複数存在し，かつそれらの同期コードに対応したタスク間に先行制約がある場合，先行タスクに対応した同期コードを選択する．他の場合，任意の同期コードを選択する．選択された同期コードを C とする．選択する同期コードがない場合は(9)へ，

(5) グループ内の同期コードを実行するプロセッサのうちで，同期コード C を実行するプロセッサと同じものがあれば(8)へ，なければ次へ，

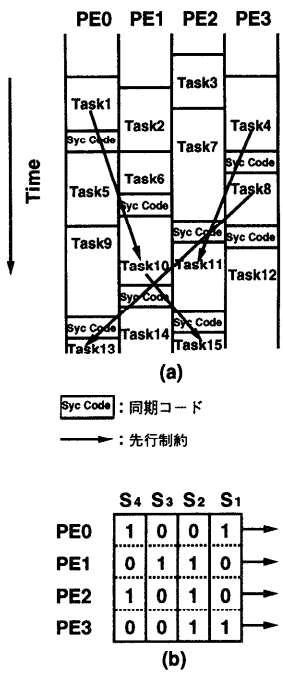


図 5 同期コード配置例とその例に対するバリア参加情報．(a) 同期コード配置，(b) バリア参加情報

Fig. 5 An example of arrangement of the synchronization code and barrier masks. (a) Arrangement of the synchronization code, (b) Barrier masks.

(6) グループ内の同期コードに対応するタスクと，同期コード C に対応するタスクとの間に，先行制約があれば (8) へ，なければ次へ，

(7) 同期コード C をグループに登録し，(4) へ戻る．

(8) 新たにグループを作成し，そのグループを $S_j(j = j+1)$ とする．同期コード C を登録し，(4) へ戻る．

(9) バリア参加情報列を生成する．グループ内の同期コードを実行するプロセッサを求め，それらが参加することを示すバリア参加情報を生成し，同期コードのグループ番号が小さい順にバリア参加情報を並べる．

たとえば，図 5 (a) の矢印の先行制約を保証するために，RBCQ 同期方式では，図 5 (a) に示す位置に同期コードを配置し，図 5 (b) のバリア参加情報列を生成する．

4. 関連研究

4.1 RBCQ 同期機構の同期モデル

バリア型同期機構は，バリア同期モデルとして，4 つの項目別に 9 種類のモデルに分類される[12]．RBCQ 同期機構は，「面状&任意参加&全順序関係&オーバラップ不可能」なバリア同期モデルの入口同期ポイントが直前の出口同期ポイントに接している特殊な場合と考えることができる．

4.2 他のバリア型同期機構と RBCQ 同期機構との比較

バリア型同期機構として，SBM 同期機構[7]，Fuzzy Barrier[1]，Elastic Barrier[6]，重複可能なバリア型同期機構[10]が提案され，また，それぞれの同期機構を利用した同期方式も提案されている．これら従来のバリア型同期機構・方式と RBCQ 同期機構・方式には，以下に示す相違点がある．

タスク間先行制約の保証方法 従来のバリア型同期方式では，入口同期コードと出口同期コード間の 2 種類の同期コード間に生じる順序関係により，タスク間の先行制約を保証するのに対し，RBCQ 同期方式では，1 種類の同期コード間に生じる順序関係により，タスク間の先行制約を保証する．このことにより，1) 2 種類の同期コードを制御する必要がないので，RBCQ 同期機構の回路は簡潔で高速な動作が可能になり，2) 連続した同期コードを 1 つにまとめることができるので，同期コード数を少なくすることができる．

ダミーの同期コード数 同期コードによる同期ポイント指定方式の場合，タスクの粒度が細かいほど，タスク実行時間に対する同期コード実行時間のオーバヘッドが大きくなり，効率良い並列処理を妨げる一因となるので，ダミーの同期コードはできるだけ少ない方が望ましい．SBM 同期機構を除いた従来のバリア型同期機構では，先行制約に関係ないタスクの前後にも，ダミーの同期コードを配置する必要があるのに対して，RBCQ 同期機構では，その必要がない．よって，RBCQ 同期方式は，同期コード数が少なくて済み，また，ダミーコードを配置する手順が必要ないので，同期コード配置アルゴリズムも簡単になる．

同期処理回路の応答時間 SBM 同期機構の PE Go 信号出力回路は，OR 回路と AND 回路の Tree 構成であるのに対して，RBCQ 同期機構のそれは，AND 回路の 1 段のみで構成されている．このため，SBM 同期機構の PE Wait 信号が入力されてから PE Go 信号が出力されるまでの応答時間は，プロセッサ台数が増加するにつれて長くなる．一方，RBCQ 同期機構の応答時間は，プロセッサ台数が増加しても一定であり，しかも SBM 同期機構の応答時間より短い．

5. 性能評価

RBCQ同期方式を他の同期方式と比較することにより性能評価を行う．評価プログラムを，各同期方式を用いて，近細粒度並列処理のテストベッドOPAS[2]で実行する．実行された評価プログラムの処理速度およびバリア参加情報の数を比較する．

5.1 比較する同期方式

本論文で比較する同期方式は，

- RBCQ同期方式
- SBM同期機構を利用したOne-PE同期方式
- 共有メモリを利用したフラグチェック同期方式

である．

フラグチェック同期方式とは，特別な同期機構を用いず，共有メモリに確保したフラグ変数を利用したBusy Waitによる同期方式である．

5.2 評価プログラムの記述言語

評価プログラムは，C言語で記述した．プログラム中の各基本ブロックに対して，各代入分をタスクとして，近細粒度並列処理を行う．Cのソースプログラムを著者らが開発したC並列化コンパイラによりコンパイルする[2]．

5.3 評価プログラム

評価プログラムは，1）差分法による流体シミュレーションのプログラムの一部，2）ディザ処理を行うプログラムの一部，である．評価プログラム1は「1対多」または「多対1」の先行制約が多いプログラム，評価プログラム2は「1対1」の先行制約が多いプログラムである．各プログラムのタスクスケジューリング結果と先行制約を図6と図7に示す．図6，図7において，円印がタスク，直線が先行制約を表し，左から右へタスクの実行が行われる．

5.4 評価システム：OPAS

OPASは近細粒度並列処理のテストベッドとして開発した共有メモリ型MIMDマルチプロセッサシステムである（図8参照）．4台のProcessing Element（PE）とShared Memory（S-RAM: 1 MByte）は，1本のVMEバスで結合される．

同期検出回路はSynchronization Controller（SC）内に実装される．SCは，PE Wait・Go信号線で各PEと結合され，同期成立の検出・PE Go信号の送出を行う．

5.5 同期コントローラ

SBM同期機構用同期コントローラおよびRBCQ同期機構用同期コントローラを製作した．

SBM同期機構用同期コントローラをTTL ICを使用して構成した[2]．この同期コントローラを用いて，One-PE同期方式の測定を行った☆．

RBCQ同期機構用同期コントローラをプログラマブルロジック（ALTERA社のEPF8282LC84）で構成した．プログラマブルロジックにダウンロードされる回路を，AHDLで記述した．

RBCQ同期機構用同期コントローラの回路は，同期

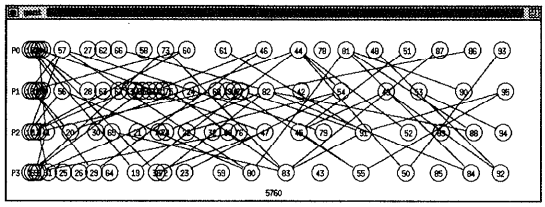


図6 評価プログラム1のタスクスケジューリング結果（PE 4台）
Fig. 6 The scheduling result of the evaluation program 1 (4 PEs).

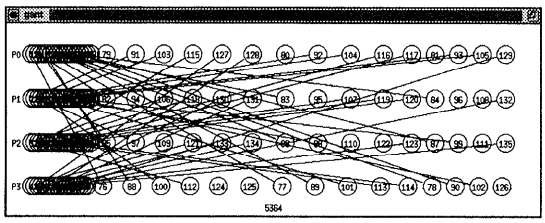


図7 評価プログラム2のタスクスケジューリング結果（PE 4台）
Fig. 7 The scheduling result of the evaluation program 2 (4 PEs).

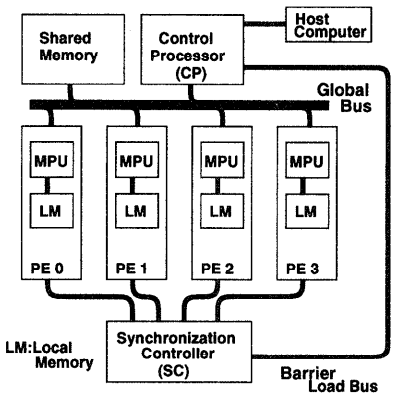


図8 OPASシステムのハードウェア構成
Fig. 8 The OPAS system.

☆ RBCQ同期機構用の同期コントローラでもOne-PE同期方式が動作可能である．しかし，OPASに実装したRBCQ同期機構用の同期コントローラでは，バリアキューの数が少なく，バリア参加情報の再投入が必要となり，その再投入のオーバーヘッドが計測されてしまうため，SBM同期機構用の同期コントローラを使用した．

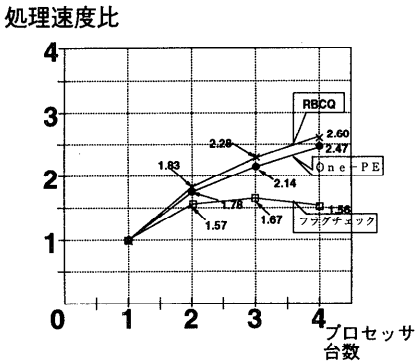


図9 評価プログラム1のプロセッサ台数と処理速度比の関係
Fig. 9 The Relation between the number of processors and speed up ratio in the evaluation program 1.

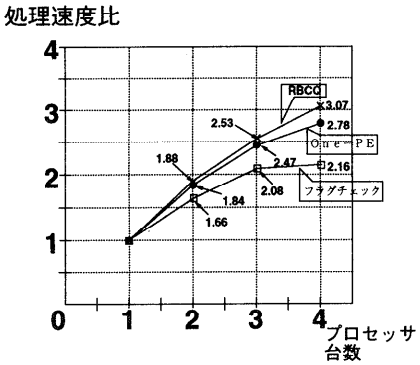


図10 評価プログラム2のプロセッサ台数と処理速度比の関係
Fig. 10 The Relation between the number of processors and speed up ratio in the evaluation program 2.

成立検出回路，バリアキュー（42個のバリア参加情報を格納可能）および信号制御回路で構成される．信号制御回路は，同期成立検出回路の出力を基に，Queue Load，Next およびPE Go信号を制御する．信号制御回路は，ジョンソンカウンタを用いたステートマシンで構成される．PE Wait信号が出力され同期が成立してから，PE Go信号が出力されるまでの時間は，最大でステートマシンのクロックサイクル時間62.5 ns+回路遅延16 ns，合計78.5 nsである（SBM同期機構では，最大68 ns）．

5.6 性能評価1

1つめの性能評価として，各評価プログラムの処理速度を比較した．各同期方式のプロセッサ台数と処理速度比（=各々のプロセッサ台数の各同期方式における処理速度/プロセッサ1台における処理速度）の関係を図9と図10に示す．RBCQ同期方式の同期機構依存先行制約が少ないことにより，各プログラムのすべてのプロセッサ台数で，RBCQ同期方式の処理速度はOne-PE同期方式のそれより速くなり，最大でOne-PE同期方式より10%，フラグチェック同期方式より42%の速度向上を得た．

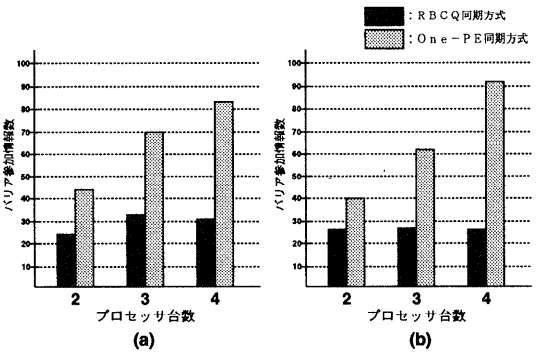


図11 評価プログラムのプロセッサ台数とバリア参加情報数の関係．(a) 評価プログラム1，(b) 評価プログラム2
Fig. 11 The number of barrier masks of the evaluation programs: (a) program 1, (b) program 2.

評価プログラム1におけるプロセッサ4台のタスクスケジューリングでは，「1つの後続タスクに対して複数の先行タスク」のタスク関係が多く存在している．フラグチェック同期方式では，このような場合，後続タスクの直前に先行タスクの個数分の同期コードが必要となる．そのため，プロセッサ4台の処理速度比が3台のそれより悪くなる一因になったと考えられる．評価プログラム2では，「1つの先行タスクに対して1つの後続タスク」のタスク関係が大部分なので，上記で示した性能低下はほとんどなく，プロセッサ4台の処理速度比は3台のそれより向上している．RBCQ，One-PE同期方式では，複数先行タスクが存在しても，後続タスク側の同期コードは1つだけで済むので，上記のようなフラグチェック同期方式で生じる性能低下はない．

5.7 性能評価2

2つめの評価として，バリア参加情報の数を比較した．評価プログラム1·2のバリア参加情報の数を図11に示す．プロセッサ4台で比較すると，評価プログラム1のRBCQ同期方式のバリア参加情報数はOne-PE同期方式のそれの約1/3，評価プログラム2では約1/4となった．

両評価プログラムにおいて，プロセッサ台数が多くなるにつれてOne-PE同期方式のバリア参加情報の数は増加するが，RBCQ同期方式のそれはほぼ一定になっている．この結果は，RBCQ同期方式では，同期コード数増加によるバリア参加情報ビットの増加分が，プロセッサ数の増加によって増えたバリア参加情報内のビットに割り当てられていることを示している．

6. おわりに

同期コード数が少なく，かつ不必要なブロッキングを削減できる同期機構・方式として，RBCQ同期機構・方式を提案した．近細粒度並列処理のテストベッドOPASにRBCQ同期機構，並列化Cコンパイラにその同期方式をそれぞれインプリメントし，評価プログラムを用いて性能評価を行った．評価プログラムの処理速度を計った結果，RBCQ同期方式の処理速度は最大でOne-PE同期方式より10%，フラグチェック同期方式より42%の速度向上を得た．また，評価プログラムで必要なバリア参加情報を数えた結果，RBCQ同期方式のバリア参加情報の数は，One-PE同期方式に比べて最大で約1/4にまで減少した．

今後の課題として，以下のことがあげられる．

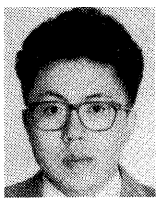
- 不必要なブロッキングが少ない同期コード配置・同期コードグループ生成アルゴリズムを検討する．
- RBCQ同期方式の冗長同期コードを削減するアルゴリズムを検討する．
- 他の同期機構をOPAS上にインプリメントしRBCQ同期機構との性能評価を行う．

なおこの研究の一部は，文部省科学研究費補助金（奨励研究(A) 09750405）による．

参考文献

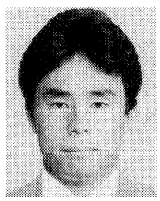
1) Gupta, R.: The Fuzzy Barrier: A Mechanism for High Speed Synchronization of Processors, *Proc. 3rd Int'l. Conf. Architectural Support for Programming Languages and Operating System*, pp.54–63 (1989).
2) 早川 潔，増村 均，本多弘樹：SBM同期機構を用いたOne-PE同期方式，電子情報通信学会論文誌，Vol.J78-D-I, No.2, pp.73–81 (1995).
3) Kasahara, H. and Narita, S.: Practical Multiprocesssor Scheduling Algorithms for Efficient Parallel Processing, *IEEE Trans. Comput.*, Vol.C33, No.11, pp.1023–1029 (1984).
4) 本多弘樹，水野 聡，笠原博徳，成田成之助：Fortranプログラム基本ブロックの並列処理手法，電子情報通信学会論文誌，Vol.J73-D-I, No.9, pp.756–766 (1990).
5) Phillip, L.S.: Minimaization of Interprocessor Synchronization in Multiprocessors with Shared and Privete Memory, *Proc. International Conference on Parallel Processing*, Vol.III, pp.138–142 (1990).
6) 松本 尚：細粒度並列処理実行支援マルチプロセッサの検討，情報処理学会論文誌，Vol.31, No.12, pp.1840–1851 (1990).
7) O'Keefe, M.T. and Dietz, H.G.: Hardware Barrier Synchonization: Static Barrier MIMD (SBM), *Proc. Int'l. Conf. Parallel Processing*, Vol.I, pp.35–42 (1990).
8) 高木浩光，有田隆也，曽和将容：問題が持つ先行関係のみを保証する高速な静的実行順序制御機構，情報処理学会論文誌，Vol.32, No.12, pp.1583–1592 (1991).
9) 早川 潔，本多弘樹：マルチプロセッサ上での細粒度並列処理のための1対1同期機構，情報処理学会論文誌，Vol.38, No.8, pp.1630–1637 (1997).
10) 高木浩光，有田隆也，曽和将容：細粒度並列実行を支援する種々の静的順序制御方式の定量的評価，並列処理シンポジウムJSPP'91，pp.269–276 (1991).
11) 高木浩光，有田隆也，曽和将容：重複可能なバリア型同期のためのスケジューリングアルゴリズムとその性能，電子情報通信学会技術研究報告，CPSY91-15, pp.91–98 (1991).
12) 山家 陽，村上和彰：バリア同期モデル—Taxonomyと新モデルの提案，および，モデル間性能比較，並列処理シンポジウムJSPP'93論文集，pp.119–126 (1993).

(平成9年10月1日受付)
(平成10年4月3日採録)



早川 潔（正会員）

昭和42年生．平成4年山梨大学工学部電子工学科卒業．平成9年同大学大学院工学研究科博士後期課程修了．同年電子技術総合研究所COE特別研究員．現在に至る．ハードウェア同期機構等の並列計算機のアーキテクチャの研究に従事．工学博士．電子情報通信学会，IEEE各会員．



本多 弘樹（正会員）

昭和36年生．昭和59年早稲田大学理工学部電気工学科卒業．平成3年同大学大学院博士課程修了．昭和62年同大学情報科学教育センター助手．平成3年山梨大学工学部電子情報工学科専任講師．平成4年同助教授．平成9年電気通信大学大学院情報システム学研究科助教授．現在に至る．並列処理方式，並列化コンパイラ，マルチプロセッサアーキテクチャなどの研究に従事．工学博士．電子情報通信学会，IEEE，ACM各会員．