超並列計算機のための同期処理機構とその評価

岡本一晃[†1] 松岡浩司[†2] 廣野英雄[†1]
横田隆史[†3] 佐藤三久[†4] 坂井修一[†5]

超並列計算機において，同期処理の高速化は重要な問題であり，ハードウェア支援による処理の効率化が効果的であることはよく知られている．しかし一方で，超並列計算機では実装上の制約が大きく，すべての同期処理をハードウェア化することは現実的でない．本論文では，超並列計算機のための同期処理機構として，メッセージ処理を基礎とした単純なマイクロ同期機構を提案し，これをソフトウェアで操作し組み合わせることで，さまざまな同期処理を効率良く実現できることを示す．さらにこのマイクロ同期機構を超並列計算機 RWC-1 に実装し，その処理性能を評価する．その結果，マイクロ同期機構が特に細粒度並列処理において大きな性能向上をもたらすことが確認された．同時に，バリアなどのマクロな同期においても，マイクロ同期機構をソフトウェアで組み合わせることで，専用のハードウェアを持つ他の並列計算機にひけをとらない性能を示すことが確認された．

Micro Synchronization Mechanisms on a Massively Parallel Computer

KAZUAKI OKAMOTO,[†1] HIROSHI MATSUOKA,[†2] HIDEO HIRONO,[†1]
TAKASHI YOKOTA,[†3] MITSUHISA SATO[†4] and SHUICHI SAKAI[†5]

This paper presents a micro synchronization mechanism for massively parallel computers. The mechanism is based on a message processing scheme, and realizes both low implementation cost and high performance of micro synchronization. Macro synchronization, as a barrier, is also performed efficiently using combination of that mechanism with software control. In this paper, we shaw the micro synchronization mechanism on RWC-1. Evaluations of its hardware cost and basic performance are also described.

1. はじめに

並列計算機においては，並列に動作するプロセッサ間の同期をいかにしてとるかということが重要な問題である．中でも汎用を指向する超並列計算機ではさまざまな粒度の並列問題を対象としており，特に細粒度の並列処理を扱う場合において同期にかかるコストが全体の性能に大きく影響するため，同期処理のオーバヘッドをいかに低くおさえるかが重要なポイントになる．しかも，処理の並列度が大きくなるほど同期をとる組合せの数が爆発的に増大し，さらに同期の形態も多様となるが，こうしたさまざまな同期についてそれぞれの処理にともなうオーバヘッドを軽減し，高速な同期処理を実現しなければならない．

同期処理のオーバヘッドをおさえて高速な同期を実現するには，専用の同期機構を装備し，ハードウェアの支援を受けることが有効であると考えられる．実際，現存する並列計算機の多くには，同期のための何らかのハードウェアが装備されており，同期処理の高速化を図っている[1)~12)]．こうした専用の同期機構が，特に細粒度の並列処理を扱うような超並列計算機において，大きな効果を発揮することは容易に推測できる．

しかし，その一方で実装面を考えた場合，超並列化のための要素プロセッサ（PE）はできるだけ小規模であることが望ましく，またその開発費をおさえることからも余分なハードウェア投資はできるだけ避けたいという要望がある．特に細粒度並列処理においては，多様な形態の同期のすべてをハードウェアが支援することは現実的でなく，効果の薄いハードウェア投資は

†1 三洋電機株式会社東京情報通信研究所
SANYO Electric Co., Ltd.
†2 日本電気株式会社 C&C メディア研究所
NEC Corporation
†3 三菱電機株式会社先端技術総合研究所
Mitsubishi Electric Corporation
†4 新情報処理開発機構つくば研究センタ
Real World Computing Partnership
†5 東京大学工学系研究科電気工学専攻
The University of Tokyo

避けなくてはならない．したがって，超並列計算機の同期機構は，ハードウェア支援とソフトウェア処理との組合せで実現し，両者のバランスは処理効率と実装コストとのトレードオフで決定されるべきであると考えられる．言い換えれば，超並列計算機の同期機構においては，どこまでの機能をハードウェア化して支援するかの選択が重要であるといえる．

このような背景をふまえ，本論文では超並列計算機のための単純なマイクロ同期機構を提案し，これを組み合わせて使用することで細粒度並列処理にともなうさまざまな同期処理を効率良く実現できることを示す．そしてこれを超並列計算機に実装し，その有効性を検証する．

本論文では最初に超並列システムにおける同期機構の要件を示し（2章），次にこれを満足する超並列計算機向け同期機構について述べる（3章）．さらにその実現例として，超並列計算機 RWC-1 の同期機構について述べ（4章），そのハードウェアコストと基本性能について評価を行う（5章）．最後に，現状と今後の課題について述べる（6章）．

2. 超並列システムにおける同期処理機構

並列計算機における同期処理を大別すると，(1) 命令や小規模の命令ブロックを単位とするミクロな同期と，(2) バリアのようなマクロな同期との2種類に分けられる．前者については，命令間のデータ依存関係を保持するための同期処理がその代表例としてあげられる他，新たに fork したスレッドからの戻り値を待つ場合，ベクトル演算のような定型的な繰返し演算におけるベクタの要素どうしの対応付けなどもこれに含まれる．一方後者は，たとえば，並列に処理されるようスケジューリングされた複数のタスクにおいて，全体で処理の順序関係を保証するために歩調を合わせる場合がこれに該当する．前者のミクロな同期処理は，特に細粒度並列処理の性能に支配的であり，ハードウェア支援による処理の効率化が効果的であると考えられる．これまでにも，命令レベル並列処理の代表とされるデータ駆動型計算機において，さまざまな同期機構が提案され，その有効性が示されている[13]．これに対し後者のマクロな同期処理は，データ並列処理のような大規模数値演算において必要とされ，多くの並列計算機でバリア同期を高速に行うためのハードウェア機構を実装している[10)~12)]．またバリア同期のバリエーションとしては，これまでに Fuzzy Barrier[14]，Elastic Barrier[15]，重複可バリア[16] など多数が提案されており，そのハードウェア支援も検討されている．

このように効率面から考えると，同期処理にはハードウェア支援による高速化が重要であり，特に汎用を指向する超並列計算機においてはミクロな同期とマクロな同期との両方を高速に処理する必要がある．

一方，ハードウェアの実装面から考えると，特に超並列計算機において過剰なハードウェア投資は望ましくない．ここで実装面から見た超並列計算機の同期処理機構の要件を考えると，

(1) 付加回路のハードウェア量が小さく，かつ複雑でないこと
(2) 入出力線数を増やさないこと

の2点があげられる．このうち1点目においては，近年の半導体集積技術の向上により量的な問題は解決されつつあり，むしろ複雑な付加回路によって生じるクロック長の増大や，並列動作時の検証・調整作業の混迷化などの問題が大きい．また2点目の線数の問題では，プロセッサやスイッチなどの LSI のピン数や，基板上および基板間の配線などに制約がある．たとえば，バリア同期専用の信号線の導入は，全体の実装規模が大きくなった場合に線のコストが無視できなくなる可能性がある．

以上のことから効率面と実装面の両面を考え合わせると，超並列計算機においては，同期機構として規模が小さく単純で高速なハードウェアを実装し，これを効率良く組み合わせて使用することでさまざまな同期に対応していく方法が，望ましいと考えられる．

3. マイクロ同期機構

上述の超並列システムにおける同期機構の要件をふまえ，本章ではメッセージによって通信を行う細粒度並列計算機のためのマイクロ同期機構について述べる．

前述のとおり，同期処理はミクロな同期とマクロな同期に大別できるが，特に細粒度並列処理において支配的な前者は，ハードウェア化による処理の効率化が全体性能の向上に効果的である．そこで，ミクロな同期の中でも最も単純なものとして，2つのメッセージの同期をとるだけのマイクロ同期に注目し，これを高速に処理する同期機構を考える．そして，すべての形態の同期処理は，このマイクロ同期をソフトウェアで組み合わせることで実現することとする．

ここで述べるマイクロ同期機構は，同期情報を含むメッセージとそれを受け取るメッセージ処理機構，および同期処理を起動する同期命令とで実現される．

3.1 同期メッセージ

マイクロ同期は2つの同期メッセージをメモリ上で待ち合わせて実現する．各々の同期メッセージは，同

期情報として待ち合わせを行うメモリアドレスと，同期が成立したあとの処理を指し示す命令アドレスを運ぶ．また，必要に応じて2つのメッセージの識別情報（左 or 右），およびメッセージ長などが同期情報に含まれる．ここで同期メッセージは，通常のメッセージとは独立した特殊メッセージであってもよいが，通常のメッセージと同じ枠組みで考えることも可能であって，その方が汎用性に富む．この場合，同期情報はメッセージのヘッダ部に組み込むのが効率的である．

3.2 メッセージ処理機構

メッセージにより伝達される同期情報の獲得は，受信処理の一環としてメッセージ処理機構で行われる．メッセージ上の同期情報は，ハードウェアにより自動的にプロセッサの特殊レジスタに格納される．ここで獲得された同期情報は，同期処理時に参照される．

3.3 同期命令

マイクロ同期機構は，上述の同期情報に基づき対となる2つのメッセージを検出し，同期が成立したかどうかを判断する．同期が成立すれば同期情報が指し示す命令を実行し，不成立のときは当該メッセージをメモリ上に退避して後続メッセージの到着を待つ．このマイクロ同期機構の起動を，同期命令により行う．

処理の効率面から考えると，同期処理もメッセージ処理などと同様にすべてをハードウェアで自動的に処理されるのが望ましい．しかし，メッセージ処理とは違って同期処理にはメモリアクセスがともなうため，ページフォルトなどの例外が発生する場合が考えられ，すべてをハードウェアで対応するのは実装上困難である．また，さまざまな同期形態を実現するために，マイクロ同期をソフトウェアで組み合わせていく際，その起動をソフトウェアで制御できる方が柔軟性に富む[17]．

同期命令は，パイプライン化されたマイクロ同期機構の起動のみを1クロックで実行する命令であり，RISC 形式の実行形態をも乱すことなく実装される．

4. 超並列計算機 RWC-1 の同期機構

前章で述べた細粒度並列計算機のためのマイクロ同期機構の実現例として，本章では超並列計算機 RWC-1[18] に実装したマイクロ同期機構について述べ，次章でこれを評価する．

4.1 超並列計算機 RWC-1

RWC-1 は図1のとおり分散メモリ構成の超並列マルチスレッド計算機であり，メッセージの到着によってスレッドの起動や切替えが行われる[19]．特に細粒度の並列処理機能の強化のため，メッセージは 64 Byte 以下の小さなデータサイズで授受され，かつプロセッサの汎用レジスタ間でデータを直接転送することで，通信のオーバヘッドを低減している．

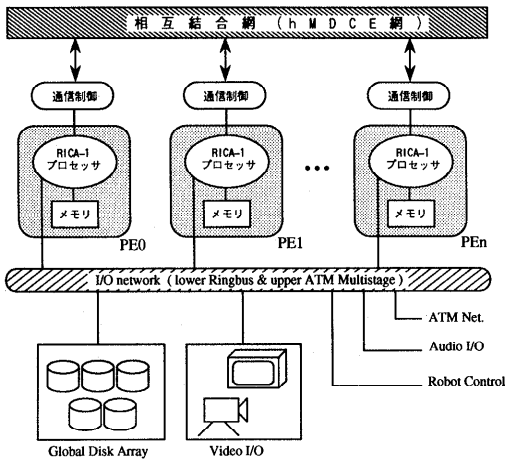


図1 RWC-1 の構成
Fig. 1 Organization of RWC-1.

個々の PE は1つのプロセッサ LSI（RICA-1）[20] と2つのルータ LSI[21] とから構成されており，独自の超並列向け相互結合網[22] によって結合されている．RICA-1 プロセッサ（図2）は，RISC アーキテクチャに基づく演算処理パイプラインと，マルチスレッド処理のための通信パイプラインとが高度に融合された，マルチスレッド型プロセッサである．RICA-1 の演算処理パイプラインは4ステージからなり，通常のスーパスカラ型 RISC プロセッサとして動作する．内部に 64bit32 ワードの汎用レジスタと，スレッドの環境を保持する制御レジスタとからなるレジスタセットを持ち，それを実行優先順位に対応して複数セット用意することで，高速なスレッド切替えを実現している．一方，通信パイプラインは超並列向けプロセッサアーキテクチャ RICA[23] に基づき，1）メッセージ処理（スレッド起動），2）同期処理，3）スレッド実行，4）メッセージ生成および送出，からなる循環パイプラインを基本としている．

RWC-1 では，受信されたメッセージがハードウェアにより直接汎用レジスタ上にロードされ，同時にスレッドが起動される．これらの処理は完全にパイプライン化されており，高速なコンテクストスイッチを実現している．また RWC-1 の同期処理は，データ駆動計算機のようにハードウェアが直列にパイプライン化されているわけではなく，必要に応じてスレッドの中からソフトウェアにより起動する．これを実現するためのハードウェアとして，メッセージを利用してスレッ

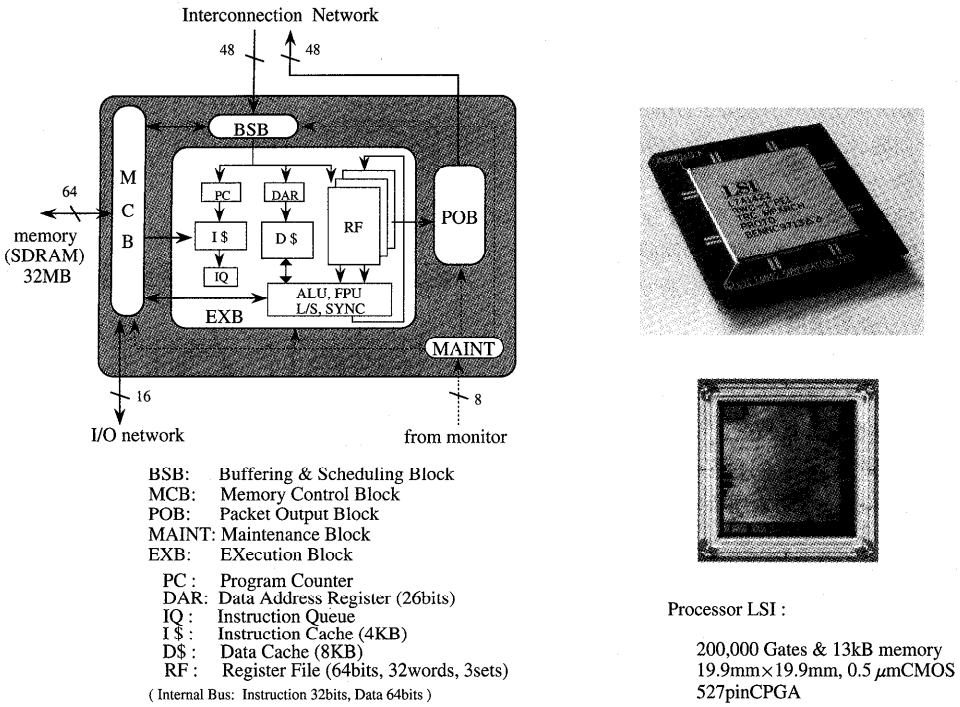


図 2 RICA-1 プロセッサ
Fig. 2 RICA-1 processor.

ド間の同期をとるマイクロ同期機構を実装している．さらにメッセージ生成および送出についても専用のパイプラインを実装しており，起動のみを命令（mkpkt）により 1 クロックで行うことで他の命令実行との重畳化を実現している．

メッセージ転送はルータ LSI を介し，独自の相互結合網を通して行われる．ここでは超並列向けの結合トポロジ hMDCE の採用により，遠隔プロセッサへの通信遅延をおさえると同時に，virtual cut through 方式によるパイプライン化で，高速なメッセージ通信を実現している[24]．

4.2 RWC-1 のマイクロ同期機構

RWC-1 のマイクロ同期機構は，前節で述べた細粒度並列計算機のためのマイクロ同期機構に基づいて実装されている．すなわち，2 つのメッセージをメモリ上で待ち合わせるデータ駆動型同期を，スレッド上の命令実行により陽に起動するものである．マイクロ同期処理を専用の命令で起動することにより，例外発生時の対応をソフトウェアでとれるようにすると同時に，さまざまな形態の同期に対応することを容易化している．さらにこの同期命令をスレッド中の任意の位置で実行可能にすることで，柔軟なプログラミングを可能にしている．

図 3 に RWC-1 のマイクロ同期機構を示す．図において RWC-1 のメッセージは，行き先を指示するためのヘッダ部と，引数を運ぶためのデータ部から成っている．ヘッダ部には同期情報が埋め込まれており，内訳として待ち合わせを行うメモリアドレス（LVOA），同期成立時に実行する命令アドレス（LVIA），メッセージ長（SIZE），および 2 入力オペランドの左右を識別するための L/R フィールドが含まれている．これらの同期情報は，メッセージ受信時にハードウェアにより自動的に，それぞれが特殊レジスタに格納される．ただし待ち合わせアドレスが格納されるレジスタは，汎用レジスタの 1 つとしてプログラムから参照することができる．

4.2.1 待ち合わせアドレス

2 つのメッセージを待ち合わせるメモリ上のアドレスは，同期命令が指定する汎用レジスタの値によって指し示される．通常のデータ駆動型同期の場合は，メッセージに搭載されたメモリアドレスを利用して処理を行うが，RWC-1 では他のメモリアドレスで待ち合わせることもできる．したがって，データ駆動計算機に見られるフレームの概念に縛られることなく自由にプログラミングができ，また動的にアドレスを決定することも可能であって，柔軟性に富む．

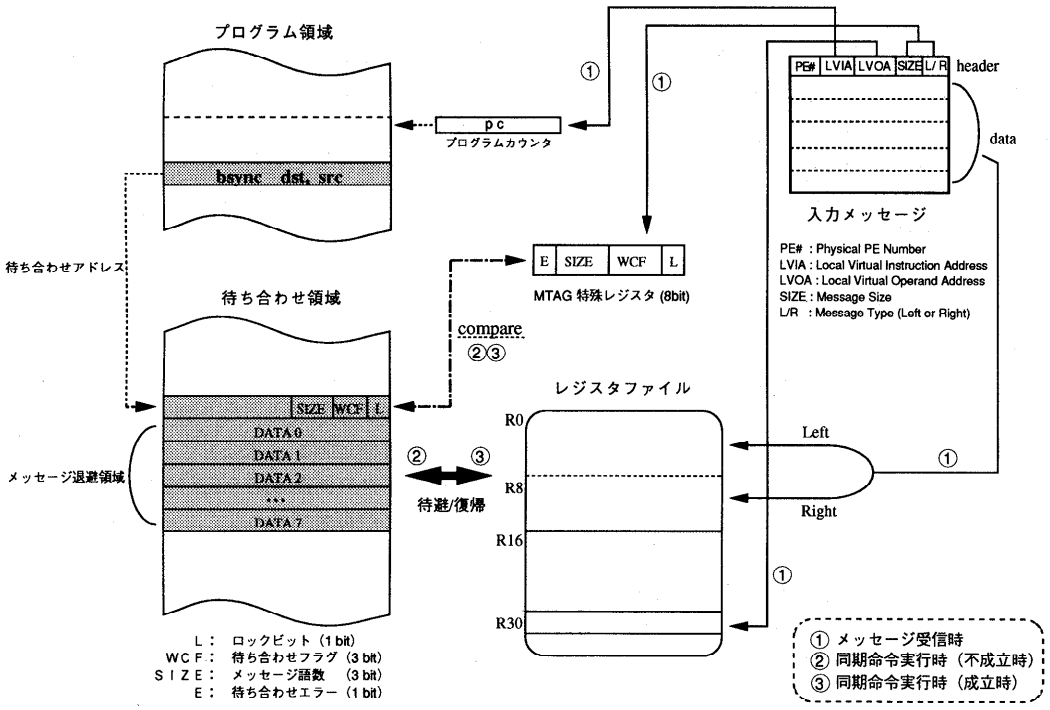


図 3 RWC-1 のマイクロ同期機構
Fig. 3 Micro synchronization mechanism on RWC-1.

4.2.2 待ち合わせフラグ領域

データ駆動型同期においてメモリ上で待ち合わせを行う場合，メモリ上には先着メッセージを退避する領域とともに，待ち合わせの状態を示すためのフラグ領域が必要である．データ駆動型同期の待ち合わせフラグは通常，メモリ上に相手がいるかどうかを示す presence bit，相手が右オペランドか左オペランドかを示す L/R bit，の 2 ビットから成る．RWC-1 の場合はこれに加え，後述する同期付構造体をサポートするために，Waiting Queue を作っているかどうかを示す Q bit を持っており，合計 3 ビットにより構成されている．また RWC-1 のメッセージは 8 ワードまでの可変長であるため，メモリ上に退避されたメッセージのワード数を格納するフィールドが必要である．さらに同期処理で必要な不可分ブロックを保証するため，Lock Bit を用意しており，これらがメモリ上に書き込まれる．

4.2.3 同 期 命 令

RWC-1 のマイクロ同期は，以下に示す同期命令 (bsync) により実現される．bsync 命令の書式は，

・ `bsync dst, src`

である．その動作はデータ駆動型同期に基づき，src が示すレジスタの内容を待ち合わせアドレスとして以下に示すように待ち合わせを行う．

(1) 上述の待ち合わせフラグをチェックし，その判別結果を dst レジスタに書き込む．
(2) 同期が成立したら，退避されている先着メッセージを汎用レジスタ上に復帰し，待ち合わせフラグをクリアして次命令に進む．
(3) 同期が不成立の場合は，汎用レジスタ上のメッセージをメモリに退避し，待ち合わせフラグを更新してスレッドを中断する．

また，同期付構造体に基づくキュー操作が必要な場合は，フラグを更新すると同時にロックをかけ，キュー操作の不可分性を保証する．このときは，メッセージの退避および復帰は行わないで，そのまま次命令に進む．

RWC-1 の同期命令は，スレッド中の任意の位置から実行が可能であり，柔軟なプログラミングが可能である．

4.2.4 マイクロ同期処理

前述の同期機構および同期命令を利用して，RWC-1 のマイクロ同期処理は，以下のように実現される．

先着したメッセージは，ハードウェア処理によりレジスタファイル上にロードされ，スレッドが起動される．このとき，メッセージのタイプ (左 or 右) によって，メッセージがロードされるレジスタファイルの位置が変わる．また，メッセージが持っている命令アド

レスはプログラムカウンタへ，待ち合わせアドレスは汎用レジスタへ，パケットサイズ，待ち合わせフラグはそれぞれ同期処理用の特殊レジスタに保持される．スレッドの中で bsync 命令が実行されると，上述のとおりレジスタ上のメッセージを退避し，フラグを更新して当該スレッドを終了する．対となるメッセージが到着すると，同様にスレッドが起動される．スレッドの中で bsync 命令が実行されると，退避されている先着メッセージをレジスタ上に復帰し，待ち合わせフラグをクリアして後続の命令に進む．こうしてメッセージによるマイクロ同期が成立する．

4.2.5 バリア同期

バリア同期は，図 4 のようにマイクロ同期をカスケード状に組み合わせることによって実現する．これにより任意の組合せの部分バリアや，複雑な形態のバリアなどにも柔軟に対応できる．

さらにメッセージの優先度処理と組み合わせることにより，Fuzzy Barrier などの面状バリア同期にも対応することが可能である．RWC-1 では優先度処理により，スレッドの切替えをオーバヘッドなく行えるため，これを利用して面状バリアが実現できる．すなわちカスケード状に組み合わせた同期処理部分を高い優先度で処理することにより，スレッド本体の実行と同期処理とを並行して処理することができる．

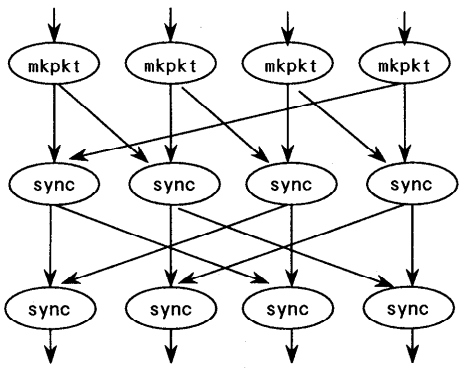


図 4 RWC-1 のバリア同期
Fig. 4 Barrier synchronization in RWC-1.

4.2.6 同期付構造体の実現

細粒度並列演算においては，I-structure などの同期付構造体が有効であることが知られている[25]．特にオブジェクト指向の分野でよく見られるように，生産者-消費者の関係が 1 対 1 で対応するような場合，I-structure を拡張した Q-structure と呼ばれる同期付き構造体が有効であるとされる[26),27]．ほかにも多くの同期付構造体が提案されているが[28),29]，これらはみなマイクロ同期機構を応用してソフトウェアで実現することが可能である．

図 5 に Q-structure の実現例を示す．Q-structure は生産者–消費者間のデータ依存関係を対応づける場合に用いられる構造体で，書き込み要求と読み出し要求との間で 1 対 1 の同期をとる．書き込みと読み出しとの双方とも，対応する相手が未着の場合にはキューを作成して待ち合わせを行う．ただし待ち合わせの性質上，双方が同時にキューを作成することはない．図 5 では，bsync 命令を用いて Q-structure を実現しており，呼び出し側は書き込みデータをメッセージに載せて送出する（q_write）．また読み出すときには戻りアドレスをメッセージに載せて送り出し，戻りメッセージを受けるために一度スレッドを中断する（q_read）．

5. 評　　価

本章では，RWC-1 のマイクロ同期機構について評価する．RWC-1 は現在 128 PE 構成のプロトタイプシステムを調整中であり，これを用いて実機上で評価

```
/* Qstructure handler :                */
/*   global address is automatically  */
/*     loaded to $r30                 */

q_struct:
        bsync  WORK, ($r30)
        bbs    7, WORK, lock_fail
        bbs    3, WORK, mng_queue
        mkpkt  $r8,$r0,0,0,0,0,1   /* retuen */
mng_queue:
        キュー操作
lock_fail:
        サスペンド処理
```

callee 側

```
/* Qstructure access routine:    */
/*     $r1 is global address      */

q_write:    /* $r2 is write data */
        mkpkti  $r2,$r1,q_struct,0,7,0,0,1,1  /* right packet */
        jmpr    $r29      /* return */

q_read:     /* $r30 is frame top address  */
        st      ($r30), $r29   /* save return address */
        mkcnt   $r0,$r30,rtn,0,7,0,0,1,0  /*return continuation */
        mkpkti  $r0,$r1,q_struct,0,7,0,0,0,1   /* left packet */
        break   0       /* suspend */

rtn:
        ld      $r29, ($r30)   /* restore retunn address */
        jmpr    $r29       /* return */
```

caller 側

図 5 Q-structure の実現例
Fig. 5 Programming sample of Q-structure.

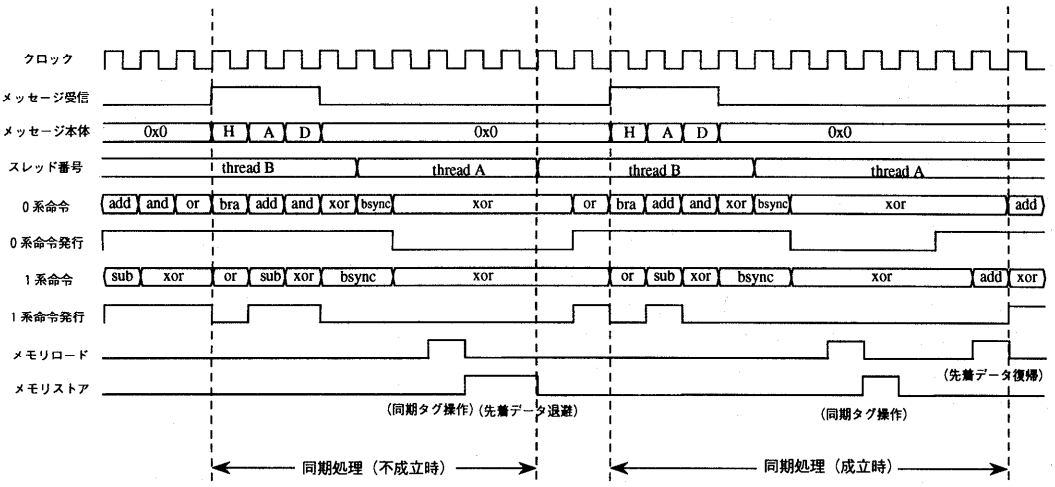


図 6 マイクロ同期処理の動作タイミング
Fig. 6 Timing diagram of micro-synchronization.

表 1 RWC-1 の同期機構のハードウェア量
Table 1 Hardware cost of synchronization on RWC-1.

プロセッサ	
付加回路量	1605 gates
付加入出力ピン	なし
基板	
外部付加回路	なし
付加信号線	なし

を行った☆.

5.1 ハードウェア量

RWC-1 の同期機構にかけるハードウェア規模を，**表 1** に示す．メッセージ処理機能を利用して実装しているため同期のための付加回路はわずかで，プロセッサの論理回路において全ゲート数に占める割合は，1 パーセントにも満たない．また同期命令は load/store 命令の拡張形で，メモリアクセスには load/store pipeline を利用しているため，その起動は 1 クロックで行え，かつ付加回路によるクロック長増大も避けられた．さらにメッセージ転送をベースに同期をとるので，付加的なピン数および信号線数はゼロであり，特に線数の制限が厳しい超並列計算機に適合しているといえる．

5.2 マイクロ同期性能

bsync 命令により起動されるマイクロ同期の，静的な処理性能について評価する．前述のとおり bsync 命令は，先に受信したメッセージをメモリ上に退避し，同期の対になるメッセージを受信したときに，退避した先着メッセージを汎用レジスタ上に復帰して，後続の命令実行を続行する．

表 2 マイクロ同期機構導入の効果
Table 2 Cycles of micro-synchronization mechanism.

メッセージ語数	1 word	8 word
同期機構を用いた場合	20 clocks	34 clocks
同期機構を用いない場合	61 clocks	95 clocks

ここでマイクロ同期の処理時間は**図 6** のとおりである．図に示すとおり，同期が不成立のときの処理が 9 clock（先着メッセージを受信してから bsync 命令が起動されるまでに 4 clock，bsync 命令が実行され先着メッセージがキャッシュ上に退避される時間が 5 clock），同期が成立したときの処理は 11 clock（同じく後続メッセージの受信処理が 4 clock，bsync 命令が実行され先着メッセージが復帰されるまでの時間が 7 clock）である．したがって，このマイクロ同期機構により 1 回の同期に要する処理時間は 20 clock になる．すなわちプロセッサが 50 MHz で動作する場合，1 回の同期が 400 nsec で行えることになる．また，メッセージ受信処理と先着メッセージの復帰処理の一部について他スレッドの実行と重畳化でき，これにより実行効率をさらに向上させている．図 6 から同期処理のうち 10 clock が他のスレッドの命令実行とオーバラップしており，マイクロ同期のパイプライン実行時間が 10 clock であることが読みとれる．

同じマイクロ同期を本同期機構を用いずにソフトウェアで処理した場合との比較を，**表 2** に示す．ここではメッセージ処理機構とロック命令を利用してマイクロ同期を実現した場合を想定している．表から同期機構を利用した場合に比べ，3 倍以上の処理時間が必要となることが分かる．しかも同期機構を用いない場

☆ 現在 RWC-1 は調整中のため動作周波数 33 MHz で評価を行ったが，本論文では最終動作予定の 50 MHz に換算して記述する．

合，ロックに失敗したときにキュー操作などが必要になり，その処理量はさらに増大する．

5.3 さまざまな同期性能

RWC-1 のマイクロ同期機構を利用したいくつかの同期性能について，128 PE 構成の RWC-1 プロトタイプを用い評価を行った☆．

5.3.1 細粒度 fork-join における同期の評価

RWC-1 にマイクロ同期機構を導入した効果を調べるため，細粒度並列プログラムによる評価を実機を用いて行う．最初に，フィボナッチ関数を用いて，本同期機構の有効性を検証した．ここでフィボナッチ関数は，探索問題用のトイベンチマークとしてではなく，典型的な fork-join 構造を持つプログラムとして採用している．そのためフィボナッチ関数を再帰的に処理し，関数ごとにスレッドを分割していて，スレッド間の同期にマイクロ同期機構を利用している．

図 7 から，Fib（10）の処理において，マイクロ同期機構の導入が 4〜5 割程度の性能向上を実現していることが分かる．一方，Fib（15）の処理においては，PE 台数が少ないときにマイクロ同期機構の効果が薄い．これは，関数の引数が大きくなると，生成されるスレッド数が飛躍的に増大し，これを起動するメッセージの数が増えるため，システムが持つメッセージキューが溢れてしまうことが原因と考えられる．この場合溢れたメッセージキューは，PE のローカルメモリ上に退避されるが，この退避処理が占める時間がスレッド実行時間に比べて小さくないため，相対的にマイクロ同期の導入効果が薄れている．しかし，この場合でもなお 2〜3 割程度の効率化が実現されており，マイクロ同期機構が細粒度並列処理に有効であることを示している．

さらに同じシステムを用い，fork された関数内部での処理に重みづけをして同様の評価を行った．これは，より現実的な問題への指標として，マイクロ同期機構の性能への影響が，スレッドの粒度に対しどのように変化するかを見るものである．**図 8** において，横軸は関数内部に挿入したダミー処理のステップ数である．関数内のダミー処理が大きくなるほど，すなわち並列処理の粒度が大きくなるほどマイクロ同期機構の導入効果が薄れていくが，グラフより関数内のダミー処理が 100 ステップのとき，約 14%の性能向上がマイクロ同期機構の導入により得られていることが分かる．さらにダミー処理が 150 ステップのときでも，なお 5%程度の効率化が実現されている．このことから，スレッド内の処理が 100 ステップ程度の細粒度並列処理には，マイクロ同期機構の導入が有効であることが実証された．

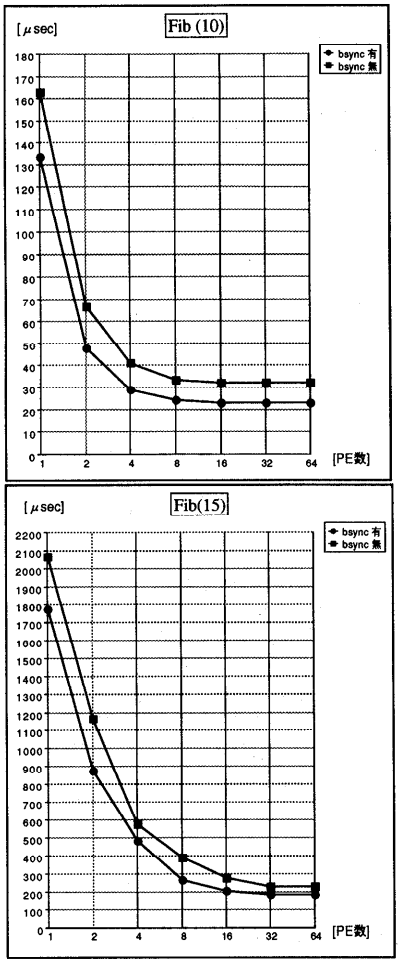


図 7 フィボナッチ関数の処理時間
Fig. 7 Performance of fibonacci function.

5.3.2 バリア同期の評価

マイクロ同期をソフトウェアで組み合わせることによりバリア同期を実現し，その性能を評価する．RWC-1 のバリア同期性能の測定結果を，**図 9** に示す．また，他の並列計算機のバリア同期性能との比較を**表 3** に示す．

表よりマイクロ同期を組み合わせて実現する RWC-1 のバリア同期が，専用のバリア同期機構をハードウェアで実装する他の並列計算機と比較しても，遜色のない性能を示していることが分かる．

☆ RWC-1 は 128 PE 構成のシステムが稼働を始めているが，今回は調整の都合上，実機での計測が 64 PE までしか行えなかった．そのため 128 PE の性能については，レジスタ転送レベルシミュレーションにより計測した．これは設計時の HDL 記述を用いたもので，クロックレベルで正確に動作を反映している．

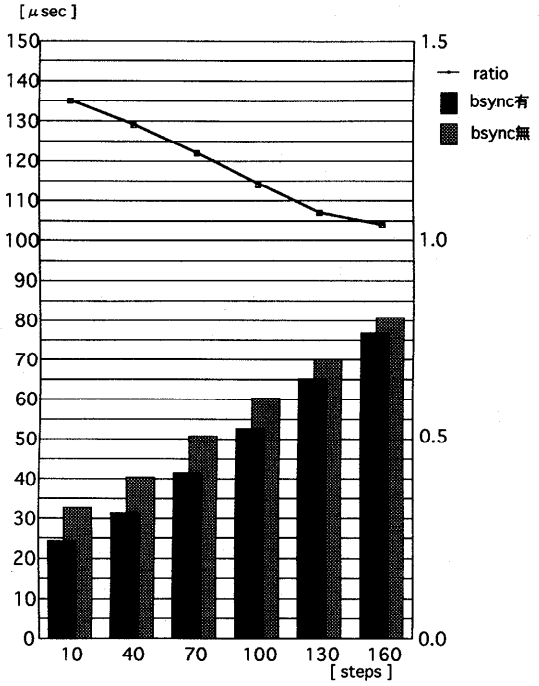


図 8 スレッドの粒度に対する性能の感度
Fig. 8 Performance sensitivity for thread granuality.

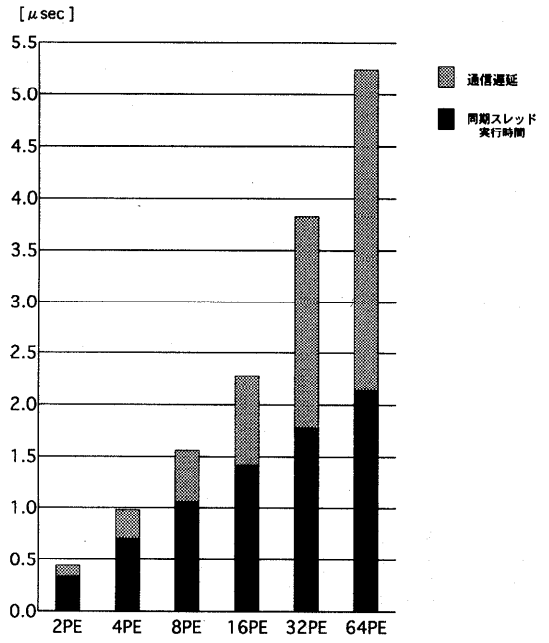


図 9 RWC-1 のバリア同期性能
Fig. 9 Evaluation of barrier synchronization on RWC-1.

一方，図 9 を見ると，バリア同期に要する時間のうち，実際にプロセッサが同期処理を行うスレッドを実行している時間に比べてメッセージが結合網上を転送される通信遅延が大きく，これがバリア性能を低下させていることが分かる．元来，並列計算機においてプロセッサ間の通信遅延は避けられない問題とされているが，特に RWC-1 においては千台規模の超並列システムの実装を想定しているため，1) 1000 PE 以上のシステムでランダム通信に強いトポロジの選択，2) リンクあたり 300 MB/sec の転送速度の確保，3) 10 m までの転送距離の保証，などの実装上の制約から，1 ホップあたりの通信遅延が小さくない．RWC-1 ではマルチスレッド処理による遅延隠蔽によってこの問題に対処しているが，上述のようなバリア同期においてはその性質上遅延を隠蔽することができず，これが性能に影響している．

そこで，面状バリアの導入により通信遅延の隠蔽を考える．RWC-1 上に面状バリアの一例である Fuzzy Barrier を実現したときのバリア性能を測定し，**図 10** に示す．ここでは，図 9 で示した同期スレッドの実行時間と，バリア同期および面状バリア同期の通信遅延を含んだ全体の処理時間との比較を示している．図より，バリア同期において大きなオーバヘッドとなっていた通信遅延が，面状バリアではほとんど隠蔽されているのが分かる．測定では 64 PE システムにおいて，2.2 μsec のオーバヘッドでバリアが実現できており，これは専用のバリア同期機構を持つ既存の計算機を凌ぐ値である．

表 3 バリア同期性能の比較
Table 3 Comparison of barrier performance.

計算機	性能	PE 数	専用機構
CM-5 (33 MHz)	4.8 μsec	32 PE	○
AP1000 (25 MHz)	10.6 μsec	(規模に依存せず)	○
RWC-1 (50 MHz) - 実機 -	5.24 μsec	64 PE	×
RWC-1 (50 MHz) - simulation -	6.98 μsec	128 PE	×

応用問題の質に依存するとはいえ，このように面状バリアが適用できる場合には，特に専用のハードウェアを設けることなくマイクロ同期機構の組合せだけで，より高速なバリアを実現できることが分かる．また通信遅延は PE 数の増加に従って大きくなるので，面状バリアは特に PE 数が多いときに効果的であるといえる．

5.3.3 同期付構造体の評価

同期付構造体の一例として，Q-structure の評価を行う．RWC-1 では，マイクロ同期機構を応用して Q-structure を実現する．RWC-1 の Q-strucuture の静的処理性能は，**表 4** に示すとおりである．Q-structure は，マイクロ同期機構を用いず lock 命令を利用しても構築

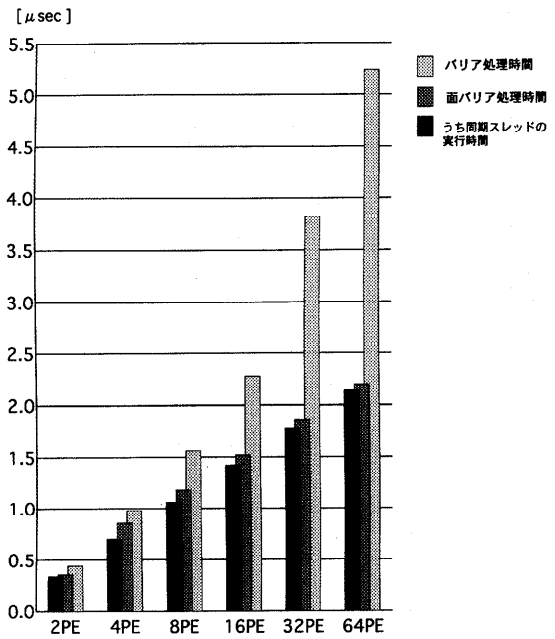
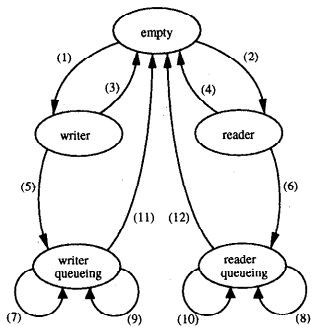


図 10 RWC-1 の面状バリア同期の処理性能
Fig. 10 Evaluation of fuzzy barrier on RWC-1.

表 4 Q-structure の処理性能
Table 4 Performance of Q-structure on RWC-1.

状態遷移	同期機構を用いた場合	同期機構を用いない場合
(1)(2)	0.16 μsec	0.60 μsec
(3)(4)	0.32 μsec	0.64 μsec
(5)(6)	1.36 μsec	1.40 μsec
(7)(8)	1.26 μsec	1.34 μsec
(9)(10)	1.30 μsec	1.38 μsec
(11)(12)	1.20 μsec	1.16 μsec



state transition of Q-structure processing

できるが，マイクロ同期機構を利用した Q-structure は，特にキュー操作が必要でない場合において高速な処理が可能である．表 4 から，キュー操作が必要でない場合，マイクロ同期機構を用いた方が 2〜4 倍の処理性能が得られることが読みとれる．また，キュー操作が必要な場合においては，マイクロ同期機構が lock 操作と同等の処理を行う．したがって，この場合処理に要する時間はマイクロ同期機構を用いない場合と同等である．

図 5 に示したシーケンスに基づき，実機上で Q-structure の読み出し時間を測定した．その結果，Q メモリが同一プロセッサ上にある場合，キュー操作がないときには 1.0 μsec，キュー操作が必要なときは 2.7 μsec で読み出しが完了した．一方，Q メモリが遠隔プロセッサ上にある場合は，読み出し時間がネットワーク構成や両プロセッサの位置関係に依存する．64 PE システムにおいての測定では，最短でキュー操作のない場合に 1.8 μsec，キュー操作が必要なときは 3.5 μsec で読み出せることを確認した．

6. おわりに

本論文では，超並列計算機 RWC-1 の同期機構について述べた．RWC-1 のような細粒度処理機構を持つ超並列計算機では，単純で高速なマイクロ同期機構が効率面とハードウェア規模の両面において優れ，かつ多様な同期形態に柔軟に対応できることを，実機による試作評価により検証した．今回の評価では，実機における評価が 64 PE までの中規模並列システムのものだけにとどまっているが，今後の課題としてより大規模な並列度のものの評価を，実機により行っていく．また，ページフォルト時の OS サポートの観点から，命令実行による同期処理の起動を採用しているが，これについての効果も実証していかなければならない．さらに，より実用的な応用プログラムについて，マイクロ同期機構の有効性を検証していく予定である．

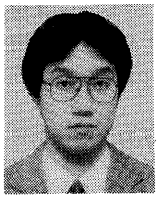
謝辞 本研究を遂行するにあたり，有益なご指導，ご討論をいただいた島田潤一 RWC 研究所長と並列分散システムパフォーマンスつくば研究室員の諸氏に感謝いたします．

参考文献

1) Shimada, T., Hiraki, K., et al.: Evaluation of a prototype data flow processor of the SIGMA-1 for scientific computation, *Proc. ISCA'86,* pp.226–234 (1986).
2) Papadopoulos, G.M. and Culler, D.E.: Monsoon: An Explicit Token-Store Architecture, *Proc. ISCA'90,* pp.82–91 (1990).
3) Kodama, Y., Sakane, H., Sato, M., Yamana, H., Sakai, S. and Yamaguchi, Y.: The EM-X Parallel Computer: Architecture and Basic Performance, *Proc. ISCA'95,* pp.14–23 (1995).
4) 富田：超並列処理計算機 JUMP-1 のアーキテクチャ，情報処理，Vol.36, No.6, pp.528–537

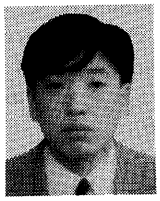
(1995).
5) 中澤，中村，朴：超並列計算機CP-PACSのアーキテクチャ，情報処理，Vol.37, No.1, pp.18–28 (1996).
6) Kuskin, J., Ofelt, D., Heinrich, M., Heinlein, J., Simoni, R., Gharachorloo, K., Chapin, J., Nakahira, D., Baxter, J., Horowitz, M., Gupta, A., Rosenblum, M. and Hennessy, J.: The Stanford FLASH Multiprocessor, *Proc. ISCA'94*, pp.302–313 (1994).
7) Felten, E.W., Alpert, R.D., Bilas, A., Blumrich, M.A., Clark, D.W., Damianakis, S.N., Dubnicki, C., Iftode, L. and Li, K.: Early Experience with Message-Passing on the SHRIMP Multicomputer, *Proc. ISCA'96*, pp.296–307 (1996).
8) Reinhardt, S.K., Pfile, R.W. and Wood, D.A.: Decoupled Hardware Support for Distributed Shared Memory, *Proc. ISCA'96*, pp.34–43 (1996).
9) Lovett, T. and Clapp, R.: STiNG: A CC-NUMA Computer System for Commercial Marketplace, *Proc. ISCA'96*, pp.308–317 (1996).
10) Thinking Machines Corporation: Connection Machine CM-5 Technical Summary (Nov. 1992).
11) 清水，堀江，石畑：AP1000の性能評価—メッセージハンドリング，放送，バリアの効果，情報処理学会計算機アーキテクチャ研究会 95-12 (1992).
12) Scott, S.L. and Thorson, G.M.: The Cray T3E Network: Adaptive Routing in a High Performance 3D Torus, Hot Interconnects IV Symposium Record, pp.147–156 (1996).
13) 坂井，岡本，松岡，廣野，横田：マルチスレッド計算機における同期機構とパイプライン構成，情報処理学会論文誌，Vol.38, No.8, pp.1613–1629 (1997).
14) Gupta, R.: The Fuzzy Barrier: A Mechanism for High Speed Synchronization of Processors, *Proc. 3rd Int. Conf. ASPLOS*, pp.54–63 (1989).
15) 松本：Elastic Barrier：一般化されたバリア型同期機構，情報処理学会論文誌，Vol.32, No.7, pp.886–896 (1991).
16) 高木，有田，曽和：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 論文集，pp.269–276 (1991).
17) 岡本，松岡，廣野，横田，堀，児玉，佐藤，坂井：超並列計算機RWC-1における同期機構，情報処理学会計算機アーキテクチャ研究会 101-2 (1993).
18) Sakai, S., Matsuoka, H., Okamoto, K., Yokota, T., Hirono, H., Kodama, Y. and Sato, M.: RWC-1 Massively Parallel Architecture, *Proc. HPCC'94*, pp.33–38 (1994).
19) 岡本，松岡，廣野，横田，坂井：超並列計算機におけるマルチスレッド処理機構と基本性能，情報処理学会論文誌，Vol.37, No.12, pp.2398–2407 (1996).
20) 松岡，岡本，廣野，横田，坂井，佐藤：超並列要素プロセッサRICA-1の高速メッセージハンドリング機構，並列処理シンポジウム JSPP'98 論文集，pp.79–86 (1998).
21) Yokota, T., Matsuoka, H., Okamoto, K., Hirono, H. and Sakai, S.: A High-Performance Router Design for the Massively Parallel Computer RWC-1, *Proc. Hot Interconnects IV Symposium*, pp.1–12 (1996).
22) Yokota, T., Matsuoka, H., Okamoto, K., Hirono, H. and Sakai, S.: hMDCE: The Hierarchical Multidimensional Directed Cycles Emsemble Network, *IEICE Trans. on Information and Systems*, pp.1099–1106 (1996).
23) Sakai, S., Kodama, Y., Sato, M., Shaw, A., Matsuoka, H., Hirono, H., Okamoto, K. and Yokota, T.: Reduced interprocessor-communication architecture and its implementation on EM-4, *Parallel Computing*, Vol.21, No.5, pp.753–769 (1995).
24) 横田，松岡，岡本，廣野，坂井，多昌，清水：超並列計算機RWC-1相互結合網ルータの実現，信学技報，CPSY97-48, pp.93–100 (1997).
25) Arvind, Nikhil, R.S. and Pingali, K.K.: I-structures: Data Structures for Parallel Computing, *ACM Trans. Programming Languages and Systems*, pp.598–632 (1989).
26) Sekiguchi, S., Shimada, T. and Hiraki, K.: Sequential Description and Parallel Execution Language DFC II for Dataflow Supercomputers, *Proc. ICS '91*, pp.57–66(1991).
27) 佐藤，児玉，坂井，山口：並列計算機EM-4における分散データ構造を用いたマルチスレッドプログラミング，情報処理学会計算機アーキテクチャ研究会 92-7 (1992).
28) Barth, P., Nikhil, R.S. and Arvind: M-structures: Extending a Parallel, Non-strict, Functional Language with State, *Proc. Functional Programming and Computer Architecture '91*, pp.538–568 (1991).
29) Jaroslav, M.: Functional Data Structures as Updatable Objects, *IEEE Trans. Softw Eng*, pp.1427–1432 (1990).

(平成10年 6 月 4 日受付)
(平成10年12月 7 日採録)



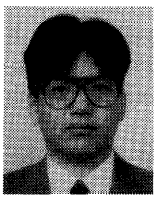
岡本　一晃（正会員）

1962年生．1986年慶應義塾大学理工学部電気工学科卒業．同年三洋電機（株）に入社．計算機システム一般，特にマルチスレッド計算機を中心とする並列処理アーキテクチャの研究に従事．1992年10月より1998年3月まで（技組）新情報処理開発機構に出向，超並列アーキテクチャの研究に従事．ICCD Outstanding Paper Award（1995年）受賞．



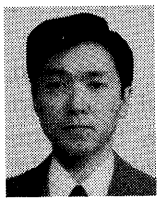
松岡　浩司（正会員）

1961年生．1984年東京工業大学工学部電気電子工学科卒業．1986年同大学院理工学研究科電子物理工学専攻課程修了．同年日本電気（株）に入社．1992年10月より1998年3月まで（技組）新情報処理開発機構に出向．現在，計算機システム全般，特にプロセッサアーキテクチャの研究に従事．ICCD Outstanding Paper Award（1995年）受賞．



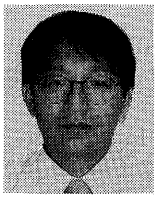
廣野　英雄

1968年生．1991年筑波大学第三学群基礎工学類卒業．同年三洋電機（株）に入社．1992年10月より1998年3月まで（技組）新情報処理開発機構に出向，超並列アーキテクチャの研究に従事．ICCD Outstanding Paper Award（1995年）受賞．電子情報通信学会会員．



横田　隆史（正会員）

1960年生．1983年慶應義塾大学工学部電気工学科卒業．1985年同大学院電気工学専攻修士課程修了．同年三菱電機（株）に入社，知識処理向けアーキテクチャおよび並列アーキテクチャの研究に従事．1993年12月より1997年3月まで（技組）新情報処理開発機構に出向．超並列アーキテクチャの研究に従事．工学博士．ICCD Outstanding Paper Award（1995年）受賞．電子情報通信学会会員．



佐藤　三久（正会員）

1959年生．1982年東京大学理学部情報科学科卒業．1986年同大学院理学系研究科博士課程中退．同年新技術事業団後藤磁束量子情報プロジェクトに参加．1991年，通産省電子技術総合研究所入所．1996年より，新情報処理開発機構つくば研究センタに出向．現在，同機構並列分散システムパフォーマンスつくば研究室長．理学博士．並列処理アーキテクチャ，言語およびコンパイラ，計算機性能評価技術等の研究に従事．日本応用数理学会会員．

坂井　修一（正会員）

1958年生．1981年東京大学理学部情報科学科卒業．1986年同大学院情報工学専門課程修了．工学博士．同年，電子技術総合研究所入所．1991年4月より1年間米国MIT招聘研究員．1993年3月より1996年2月までRWC超並列アーキテクチャ研究室室長．1996年10月より1998年3月まで筑波大学助教授（電子・情報工学系）．1998年4月より東京大学助教授（工学系研究科）．計算機システム一般，特にアーキテクチャ，並列処理，スケジューリング問題等の研究に従事．情報処理学会論文賞（1990年度），日本IBM科学賞（1991年），市村学術賞（1995年）ICCD Outstanding Paper Award（1995年）等受賞．IEEE，ACM，電子情報通信学会会員．