細粒度マルチプロセッサMSBM

岩根雅彦[†] 本石彰[†]
野口善昭[†] 米澤敏夫[††]

細粒度並列処理を効果的に行うためには高速なプロセッサ間の同期と通信が必要である．多重プログラミング環境に対して，少量のハードウェアを持った同期と通信機構を提案する．1つは包含検索機能付き機能メモリによるバリア同期機構であり，1つはマルチキャスト機能を持ったグループ共有メモリ構成である．これらを評価するためのテストベッドとして細粒度並列計算機MSBMを開発した．MSBMは1台のホストコンピュータ，グループ共有メモリを持つ16台のプロセッサ，機能メモリによるバリア同期機構を持つインタフェースから構成される．タスク内で同期をとるプロセッサ組(バリアグループ)をバリア同期機構にタスク実行開始前に登録し，実行中は動的に変更せずに終了後に削除する．これを静的バリア同期管理SBMとよび，ただ1つのバリアグループをタスクに設定するSsBMとタスク内で複数のバリアグループを設定するSmBMを考える．SsBMでのdoacrossループによる評価では16プロセッサで8.39倍の速度向上を得た．さらにWhetstoneベンチマークで性能評価を行った．SsBMでの速度向上比は個々のモジュールで1.03から2.29，Whetstone全体では1.32であった．SmBMでの速度向上比はWhetstone全体で2.35である．2つのモジュールを多重実行したときの実行時間の増加は0.00から0.09であった．MSBMにおける同期機構および通信機構の有効性が確かめられた．

Multiple Static Barrier Management MIMD (MSBM) Multi-processors for Fine Grain Tasks

MASAHIKO IWANE,[†] AKIRA MOTOISHI,[†] YOSHIAKI NOGUCHI[†]
and TOSHIO YONEZAWA[††]

The high-speed interprocessor communication and synchronization are required to achieve the efficient processing of fine grain parallelism. Two mechanisms with small amount of hardware to improve them are proposed under the multiprogramming environment. One is the hardware barrier synchronization (HBS) using the modified content addressable memory. The other is the group shared memory organization (GSM) with the ability of multicasting. The parallel computer for fine grain tasks, called MSBM, has been developed to evaluate these mechanisms as a test bed. MSBM consists of 1 hostcomputer, 16 processors with GSM and the interface unit with HBS. The processors synchronizing each other are registered into HBS before the execution of the program and are removed from HBS after the execution. This is called the static barrier management (SBM). The speedup ratio is 8.39 for doacross-loop program using 16 processors. The speedup ratio is 1.03 to 2.29 for each module of Whetstone program and 1.32 for whole Whetstone program under the single barrier registration per the program. The speedup ratio is 2.35 for whole Whetstone program under the multiple barrier registration. The increment of the execution time is 0.00 to 0.09 under the multiprogramming of 2 modules.

1. はじめに

コンパイラなどでプログラム全域からの潜在的な並列性の抽出およびスケジューリングを静的に行い，複数のプロセッサで細粒度並列処理させることが注目されているが，細粒度並列処理を行えるマルチプロセッサは少ない[1]．

細粒度並列処理ではプロセッサ間のデータ通信と同期が頻繁に行われ，これがオーバーヘッドとなる．このオーバーヘッドを削減するために，精密なスケジューリングによってプロセッサ間の同期を極力排除する方法[2]やハードウェアによるバリア同期機構[3]が提案されている．後者には，バリア同期概念の拡張と

† 九州工業大学工学部電気工学科
Department of Electrical Engineering, Faculty of Engineering, Kyushu Institute of Technology
†† (株)東芝 北九州工場
Kitakyushu Works, Toshiba Corporation

してのファジーバリア[4]，エラスティックバリア[5]，ウルティメイトバリア[6]がある．またバリア同期をとりあうプロセッサ組（バリアグループ）の指定方法として，同期識別タグによる指定[4]，マスクレジスタによる方法[4),5]，バリアキューを用いたSBM[7]およびCAM (Content Addressable Memory) を用いたDBM[8]がある．ファジーバリアおよびエラスティックバリアでは各プロセッサに分散された同期用ハードウェアで同期がとられる．一方SBMおよびDBMのバリア同期ではバリアプロセッサ[9]と呼ばれる1つの専用の同期用ハードウェアで同期処理が行われる．このようなバリア同期機構は単一タスクシステムを主な対象としており，タスク間の実行順序が不定である多重プログラミングシステムについて十分な考慮がされていない．

プロセッサ間通信として単一バスによる共有メモリ構成が多く使用されている．細粒度並列処理のようなプロセッサ間通信が頻繁に行われる環境では，バスでの競合が頻発するためにバスの多重化[1]やキャッシュメモリ[10]によりバスでの競合を軽減させているが，ハードウェア量の増大やキャッシュコヒーレンス問題が生じる．また各プロセッサが局所メモリと共有メモリを持つ部分分散型共有メモリのひとつである部分共有マルチリードメモリでは，共有メモリの読み出し競合を軽減できるが共有メモリをプロセッサ分散させるためになんらかのアドレス変換機構[10]が必要である．

そこで，細粒度の並列性を持った複数のタスクを効率よく処理するために，MCAM (Modified CAM)[11]によるバリア同期機構およびタスクごとに独立した共有メモリ空間とマルチキャスト機能による部分分散共有メモリ構成を有する細粒度並列計算機MSBM (Multiple Static Barrier Management MIMD) を機能の評価および並列アルゴリズムの開発のためのテストベッドとして開発した．

本論文では，バリアグループ管理の方法について述べ，つづいてMCAMによるバリア同期機構，部分分散共有メモリであるグループ共有メモリ構成，細粒度並列計算機MSBMのハードウェアシステム構成，さらにdoacrossプログラムによる基本性能の評価およびWhetstoneベンチマークプログラムを用いた単一タスク実行ならびに多重タスク実行での評価について述べる．

2. 基本設計

2.1 静的バリア同期管理

並列処理される1つの応用プログラムはメインプログラムやサブプログラムようないくつかのプログラムモジュールから構成されている．1つの応用プログラムの構成要素である個々のプログラムモジュールをプロセジャとよぶ．プロセジャごとに静的コードスケジューリング[12]を行ってバリア同期を挿入しておく．これらのプロセジャのリンクをとったのち，プロセッサを静的に割り当てて並列化されたタスクを形成する．すなわちオペレーティングシステムによりスケジューリングされて実行される1つの応用プログラムをタスクとよぶ．1つのタスクに静的に割り当てられたプロセッサの集合をプロセッサグループとよぶ．

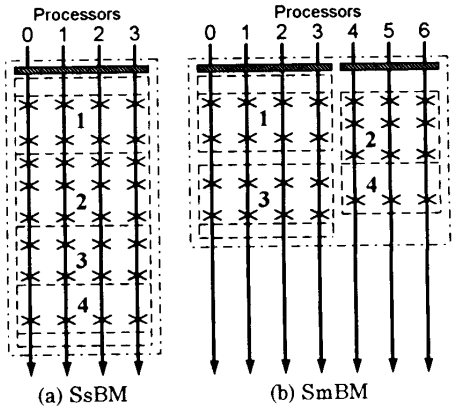

図1 SBM
Fig. 1 SBM.

タスクの実行開始前にバリア同期機構にこのタスクでバリア同期をとりあうプロセッサ組すなわちバリアグループを設定する．このタスクの実行中はバリアグループを動的に変更しないでタスク終了後にバリアグループを削除する．この方法を静的バリア同期管理SBM☆ (Static Barrier Management) とよぶ．

SBMにおいて2つの異なるバリア同期管理形態SsBM (Static single Barrier Management) とSmBM (Static multiple Barrier Management) を考える．SsBMでは1つのタスクにただ1つのバリアグループを設定して，タスクに割り当てられたすべてのプロセッサで同期をとる．SmBMでは1つのタスクに複数のバリアグループを設定し，独立かつ並列に複数のバリア同期をとる．1つのプロセジャ内ではSsBMと同様にプロセジャに割り当てられたすべてのプロセッサでバリア同期をとるが，並列実行可能なプロセジャにそれぞれ1つのバリアグループを設定して並列に実行する．**図1**の(a)にSsBM，(b)にSmBMを示す．図1の斜線での塗りつぶしはバリアグループのバリア同期機構への設定を，× はバリア同期を示

☆ 文献7)のSBMとは異なる．

す．また一点鎖線はタスクを，破線はプロセジャを示す．SsBM ではバリアグループは 1 つで，プロセジャは逐次実行される．SmBM では 2 つのバリアグループの設定のもとにプロセジャ1，3 とプロセジャ2，4 が並列に実行される．

SBM ではタスクまたはプロセジャごとに 1 つのバリアグループを設定するために本来必要としないバリア同期をダミーとして挿入しなければならないが，バリアグループの登録および削除がタスクごとに 1 回ですむ．また SBM ではタスクごとに並列度が異なるので，使用していないプロセッサに別のタスクを割り当てて多重実行する．システムを多重プログラミング環境，つまり多重静的バリア同期管理（Multiple SBM）とすることでシステム全体のスループットの向上をはかる．

2.2 バリア同期機構

2.2.1 単純なハードウェアバリア同期機構

ソフトウェアによるバリア同期ではバリア変数によって同期識別タグを持たせて任意のプロセッサ間のバリア同期をとる．これをそのままハードウェア化した場合にはハードウェア量が大きくなるので**図 2** に示す同期識別タグを削除したバリア同期機構が一般的である[3),6)]．単一タスクの実行では静的コードスケジューリングによってバリア同期の実行順序が一意に決定している．このようなバリア同期機構では，バリア同期をとらないプロセッサを同期をとる前にマスクすることによって任意のプロセッサ間でバリア同期をとることができる．これには各プロセッサ（PU）に 1 ビットのマスクが必要である．また n 本のバリア同期要求信号線とこれらの信号の論理積（Wired AND）をとって各プロセッサに通知する n 本のバリア同期成立信号線が必要である．すなわち n 個のフリップフロップと $2n$ 本の配線が必要である．

多重プログラミング環境においてタスク間の実行順序は静的コードスケジューリングでは決定できない．したがってバリア同期の実行順序は一意には決まらない．正しくバリア同期をとるためには n 台のプロセッサからのバリア同期要求をそれぞれ独立にほかのプロセッサに伝達するための $n \times (n-1)$ 本の信号線と，ほかのプロセッサからのバリア同期要求をマスクするための n ビットのマスクレジスタが各プロセッサに必要である．すなわち任意のプロセッサ間でバリア同期をとるために n^2 個のフリップフロップと $n \times (n-1)$ 本の配線が必要である．

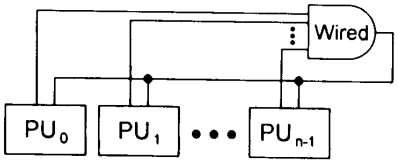

図 2 バリア同期検出機構
Fig. 2 Barrier synchronization detection mechanism.

2.2.2 MCAM によるバリア同期機構

多重プログラミング環境ではタスクごとのバリア同期が互いに干渉して性能低下を起こさないように複数のバリアグループを並列に検索する機構が必要である．これを実現するための MCAM によるバリア同期機構を**図 3** に示す．このバリア同期機構はシステム内プロセッサ（PU）台数に等しいビット長を持った BRR（Barrier Request Register），BGR（Barrier Grant Register），MCAM で構成される．BRR および BGR の各ビットはそれぞれ PU と Breq（Barrier request）信号および Bgrt（Barrier grant）信号で接続されている．また PU にはバリア同期用命令が用意されている．

多重静的バリア同期管理におけるこのバリア同期機構の基本動作はタスクの実行開始前に MCAM にバリアグループを書き込むことから始まる．次に PU が通常のバリア同期命令である Bar 命令を実行すると，Breq 信号をバリア同期機構に送ったのち PU は実行を中断する．バリアグループのすべての PU が Bar 命令を実行すると，バリア同期機構からこれらの PU に Bgrt 信号が送られて PU は実行を再開する．

バリア同期機構では送付された Breq 信号は BRR の当該ビットをセットする．BRR を MCAM の探索データとして MCAM のすべてのエントリを同時に探索する．包含関係が成立すれば当該バリアグループが MCAM から BGR に読み出される．すなわち，

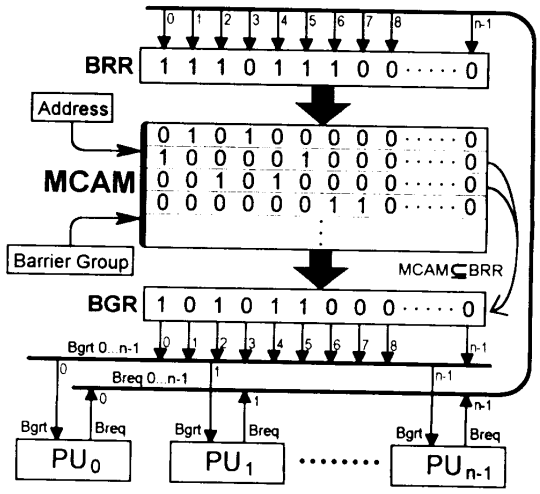

図 3 MCAM によるバリア同期機構
Fig. 3 Barrier synchronization mechanism using MCAM.

$$\text{if } MCAM_{ij} \subseteq BRR_j \quad \text{then } BGR_j := MCAM_{ij} \quad (j = 0, \cdots, n-1)$$

ただし，i はMCAM内 i 番目のエントリ，j はMCAM，BRR，BGRの j 番目のビット，n はシステムのPU総数を表す．また包含関係は $(MCAM_{ij} \subseteq BRR_j) \equiv (\neg MCAM_{ij} \vee BRR_j)$ である．なお，包含されるバリアグループがMCAMに複数存在する場合はそれらのバリアグループの論理和がMCAMからBGRに出力される．BGRに読み出されたバリアグループはBgrt信号としてPUに同期の成立を伝える．同時にBGRによって同期が成立したPUに対応するBRRのビットをリセットする．

マスクレジスタを持った単純なハードウェアバリア同期機構では同一PUを同時には異なるバリアグループの構成要素にできない．同様の制約のもとでMCAMによるバリア同期機構で任意のバリアグループを設定するためには，基本的に $\lfloor n/2 \rfloor \times n$ ビットのMCAM容量と $2n$ 本のバリア同期用信号が必要である．しかし包含論理は一致論理 $(MCAM_{ij} \wedge BRR_j) \vee (\neg MCAM_{ij} \wedge \neg BRR_j)$ よりも簡単であるのでMCAMは通常のCAMに比べて少ないハードウェアで構成できる．

2.3 グループ共有メモリ

プロセッサ間通信は共有メモリによる方法が簡単である．しかし頻繁にプロセッサ間通信が発生する細粒度並列処理，さらにその多重プログラミング環境で，共有メモリのアクセス競合を低減させるために分散型共有メモリを考える．

共有メモリを個々のプロセッサに分散させて，プロセッサグループ（単にグループとよぶ）ごとに任意の大きさの共有メモリ空間をとることができるグループ共有メモリ（GSM）構成をとる．GSMは**図4**に示すような部分分散共有メモリである．各プロセッサのメモリ空間は境界レジスタ（SMBASE）により局所メモリ（LM）とGSMに分けられる．同一グループのプロセッサのSMBASEは同一の値を設定する．MPUが局所バスを介してプロセッサのメモリの全アドレス空間にアクセスできるように，局所バスには2ポートメモリ（DPM）の1つのポート（portA）が接続されている．DPMのもう1つのポート（portB）はGSMバスインタフェースに接続されている．MPUからメモリの全アドレス空間に読み出しおよび書き込みが可能であるが，書き込み時にはGSMバスインタフェースのSMBASEとアドレスを比較し，GSMへの書き込みならばGSMバスにグループ番号，アドレス，データがマルチキャストされる．受け取り側のプロセッサではグループ番号を比較し，一致すればGSMバス上のデータをPortBからDPMに書き込む．タスクによってはGSM空間を多く必要とすることがあるため，DPMの任意の容量をGSMとして使用できるようにSMBASEを任意に設定可能とし，不必要なGSMバスアクセスを排除してバストラフィックの削減をはかる．

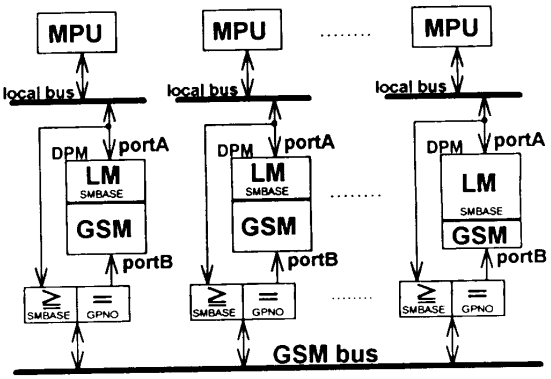

図4 GSM構成
Fig. 4 GSM organization.

また，GSMの相互排除処理のためにlock/unlock機構を用意する．lockはグループ内の他のPUの動作を停止させ，unlockは再開させる．あるプロセッサがlock要求またはunlock要求を発行すると，その要求をプロセッサが属するグループの番号とともにGSMバスに乗せる．それ以外のプロセッサでGSMバスのグループ番号が自分のグループ番号と一致するものは，lockならば処理を中断し，unlockならば中断していた処理を再開させる．このように，タスクごとに独立して相互排除処理を行える．

3. ハードウェア構成

3.1 システム構成

並列計算機MSBMは**図5**のようにホストコンピュータ（HC），統合インタフェースユニット（IIU），および16台のプロセッサ（PU）から構成されている．HC–IIU間はHCバスで，IIU–PU間はDTバス，INTバスおよびBAR信号線で，PU相互はGSMバスでそれぞれ結合されている．

HCは80386MPU，8MBメモリ，各種入出力装置およびIIUインタフェースから構成されたパーソナルコンピュータである．MS-DOSが稼動しておりユーザインタフェース，入出力管理などを行う．さらにIIUのためのBIOSレベルの基本ソフトウェアが稼動しており，PU管理やバリアグループ管理などを行うタスク管理ソフトの基礎となっている．タスク実行に際し

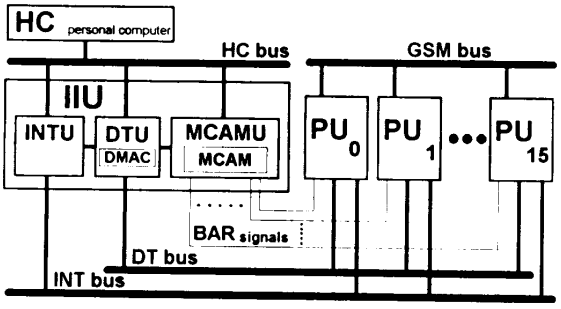

図5 MSBMのハードウェア構成
Fig. 5 Hardware organization of MSBM.

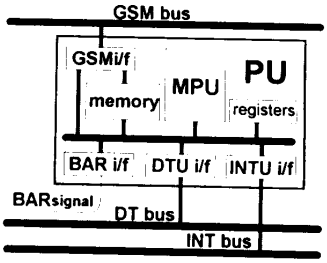

図6 PUの構成
Fig. 6 Organization of PU.

ては必要数のPUでグループを形成し，コード転送するとともに同期機構にタスクで使用する全バリアグループを登録する．そして実行終了後にPU解放とともにバリアグループの同期機構からの登録解除を行う．

IIUはデータ転送ユニット（DTU），MCAMユニット（MCAMU）および割り込みユニット（INTU）で構成される．DTUはDMAを用いてPU，グループまたは全PU単位でHCメモリとPUメモリ間のデータ転送を行う．MCAMUは一種のバリアプロセッサであり，16ビット×8のMCAMを持ちMCAMによるバリア同期機構を実現している．各PUとの接続は基本設計ではBreq信号とBgrt信号の2本であったが，BAR信号1本で実現し配線数を削減している．またバリアグループのMCAMへの登録をHCからのみとすることでハードウェアを簡素化している．INTUはPUからHCへの割り込み処理要求であるサービスコール（SVCC）の中継処理を行う．PUからSVCC要求を受け取るとPUからパラメータを受け取り，HCへ割り込みをかけパラメータを渡す．

3.2 プロセッサ

PUは**図6**に示すようにMPU，メモリ，各種内部レジスタおよび各種インタフェース（i/f）から構成され，割り当てられた逐次コードを各々独立したクロックで実行する．

MPUは8088MPU+8087NDPである．メモリは基本設計ではすべてDPMとしたがROM:32KB，RAM:208KBおよびDPM:16KBで構成し費用削減を行っている．ROMには初期化ルーチンやHCとの基本的なインタフェースとなる最低限のソフトウェアが格納されている．RAMおよびDPMの一部でLMを構成しコードや局所データを格納する．DPMのその他の空間はグループ内の共有データを格納するGSMを構成している．

内部レジスタとして，PU選択レジスタ，GSM境界レジスタ，PU番号レジスタ，グループ番号レジスタ，SVCCのパラメータ用レジスタおよび実行開始指示などのHCとの小規模通信用のレジスタなどが用意されている．DTUi/fはHC-PU間データ転送時のPUバスの制御を行う．PU，グループおよび全PUの転送単位の選択は，PU選択レジスタへのHCからの書き込みによって行われる．

INTUi/fはMPUがSVCC要求発行の出力命令を実行したときINTUへSVCCの中継処理要求を行い，承諾されるとINTUとパラメータ（16ビット×2）の転送を行う．PU間の要求調停はデイジーチェーン接続で行われる．

各PUのGSMi/fはGSMバスで接続されGSM構成とlock/unlock機構を実現する．GSM書き込み要求とlock/unlock要求はPU番号順に優先順位が設定された同じ調停機構で同等に調停される．

BARi/fではMPUの出力命令による通常のバリア同期Bar（Barrier）命令のほかにAdBar（Advanced Barrier）命令を加えた．そしてAdBar-Bar間をバリア領域とし，領域での同期を許すファジーバリア同期[4]を実現している．先だつAdBar命令なしのBar命令実行では同期要求を行い実行を中断し同期成立まで待つ通常のバリア同期命令となる．AdBar命令を実行した場合，同期要求するが実行を続ける．AdBar命令実行済みでBar命令を実行したとき，同期が成立していれば実行を中断せず，成立していなければ実行を中断し同期が成立するまで待ち，ファジーバリア同期となる．

PUの各i/f，レジスタおよびIIUの各ユニットは，FPGA（Field Programmable Gate Array）を用いて製作されている．BARi/fおよびMCAMUなどのバリア同期関連のハードウェアの1PUあたりの割合は，8088MPU，8087NDPおよびメモリを除いたハードウェアに対し5%程度であり，8088MPU，8087NDPを含めたハードウェア全体に対しては無視できる規模である．**図7**にMSBMの全景を示す．

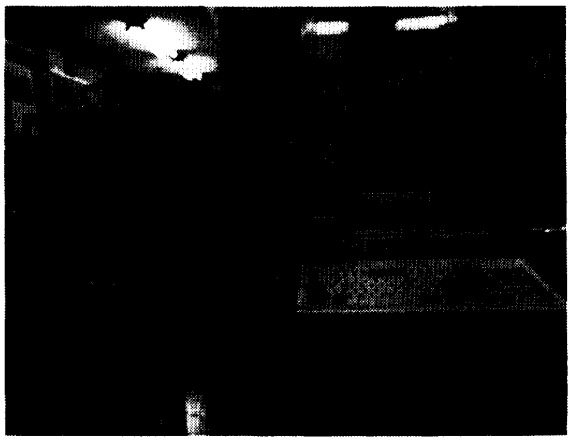

図7 MSBM の全景
Fig. 7 Picture of MSBM.

```
for i:=0 to n-1 do begin
   a[i] := b[i];
   c[i] := a[i+1];
end
```

(a) serial

```
 p = number of PU  (n >> p)
 k = 0(PU0),1(PU1),2(PU2)...
for i:=k to n-1 step p do begin
   t := a[i+1];
   Barrier;
   a[i] := b[i];
   c[i] := t;
end
```

(b) parallel (doacross)

図8 doacross プログラム
Fig. 8 doacross programs.

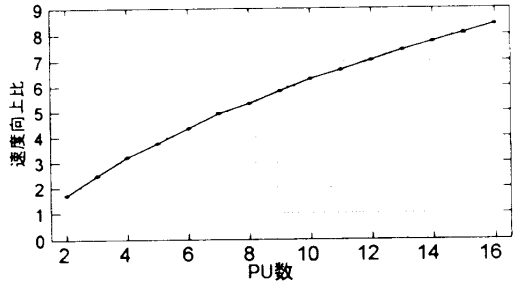

図9 doacross プログラムの速度向上比
Fig. 9 Speedup ratios of doacross programs.

4. 性能評価

4.1 doacross ループによる評価

do ループの並列処理によってバリア同期の基本性能の評価を行う．図 8(a) に示す簡単なプログラムをバリア同期を用いて並列化した細粒度 doacross ループプログラム (b) を実行する．逐次版の実行時間 Ts および各 PU 数による実行時間 Tp を測定し，逐次版に対する各 PU 数での速度向上比 Ts/Tp を算出した．その結果を図 9 に示す．

並列処理される部分の割合が大きいため，大きな速度向上が認められ，16PU で 8.39 倍の速度向上を得ている．ただし PU 数が増加するほど速度向上の伸びがなくなっている．これはバリア同期によって毎回すべての PU がほぼ同時に共有メモリに書き込みを行うため，競合による遅れが PU 数に比例して増加してしまうためである．なおバリア同期を使用しないループディストリビューション[1]による並列処理では，16PU での速度向上比は 7.72 であった．

4.2 Whetstone ベンチマークによる評価

4.2.1 Whetstone プログラムの並列化

Whetstone[13] プログラムは異なる特徴を持つ分離性の高い複数のモジュールで構成されている．そこでまず異なる性質のタスクに対する性能を求める意味で各モジュールを 1 つのタスク（モジュールタスクとよぶ）と見なして並列化した．すなわちモジュールごとに Duplication-Scheduling Heuristic[12] およびソフトウェアパイプライン手法[14] に従い，手作業でできる限り並列化を行って SsBM のバリア同期を挿入した．モジュール 8 以外は単項および二項演算を基本とした変数レベルでの並列化を行い，モジュール 8 は変数レベルでの並列化ができないので機械語レベルでの並列化を行った．表 1 に評価に使用した各モジュールの並列化プログラムの特徴を示す．バリア同期間隔は並列化コードから得られる推定値であり，複数で範囲表記（～）でないものは，同期がそれらの間隔でその順序で繰り返されることを意味する．また SMWR はタスク中の全メモリアクセスに対する GSM 書き込みアクセスの比率である．並列化率は並列化タスク中の並列化部分の比率を示し，限界値はアムダールの法則[15] によって得られた速度向上比の限界値を表す．なお，MCAM によるバリア同期は要求から成立まで待ち合わせがない場合 4 クロックで行える．

さらに Whetstone プログラム全体を SsBM および SmBM によって並列化した．SsBM では上述の並列化された各モジュールを逐次実行する形となる．最大の PU 数がモジュール 6 の 5 であるため，PU 数が 5 に満たないモジュールはそれに合わせてバリア同期のみのコードを余剰 PU で実行する．SmBM では各モジュールを並列実行する形となる．各モジュール間の依存関係を維持し，並列実行可能なモジュールを並列実行させるが，可能な限り PU 数は小さくする．図 10 にスケジューリング結果を示す．タスク全体の PU 数は 7 であり，2 つのバリアグループを MCAM に登録して実行する．並列実行するモジュールの呼び出しおよび終了待ち合わせは GSM 上の共有変数で行う．

4.2.2 モジュールタスクにおける評価

各 Whetstone モジュールにおいて，MCAM によるバリア同期機構を用いた SsBM での並列化タスク

表1 並列版モジュールの特徴
Table 1 Characteristics of parallel modules.

module	PU 数	バリア同期間隔 [マシンクロック]	SMWR[%]	並列化率 [%]	限界値
2	4	301	7.27	60.58	1.83
3	4	307	6.30	61.18	1.85
6	5	220～371	5.58	71.43	2.33
7	4	2478～2589	1.52	75.51	2.31
8	2	919	4.15	24.56	1.14
9	3	198, 32, 35	5.93	99.02	2.94
11	3	1237, 1748, 140	1.33	5.77	1.04

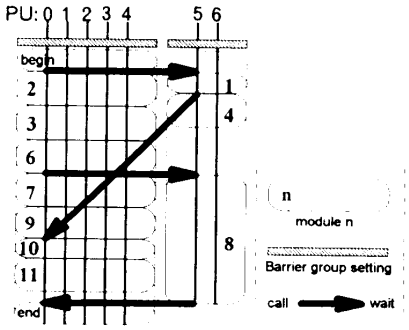

図10 SmBM でのモジュールのスケジューリング
Fig. 10 Modules scheduling under SmBM.

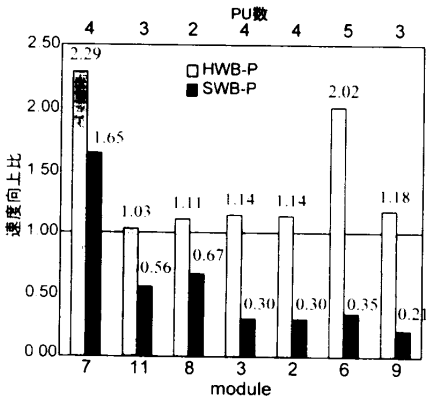

図11 逐次版 S に対する HWB-P, SWB-P の速度向上比
Fig. 11 Speedup ratios of HWB-P, SWB-P to serial modules.

(HWB-P) とソフトウェアによって必要なプロセッサ間でバリア同期をとる並列化タスク (SWB-P) を，逐次タスク (S) と比較して評価を行った．SWB-P ではバリア同期を共有メモリ上のカウンタとビジーウェイトによって実現し，カウンタ更新時には相互排除処理 (lock/unlock) を行っている．各モジュールタスクの S, HWB-P, SWB-P の実行時間 Ts, Thwbp, Tswbp を測定し，S に対する HWB-P, SWB-P の速度向上比 Ts/Thwbp, Ts/Tswbp を求めた．その結果を図 11 に示す．ただしバリア同期間隔を粒度の基準としモジュールタスクを左から粒度が大きい順に列挙している．

HWB-P ではモジュール 7, 6 の速度向上比は 2.29, 2.02 でその他のモジュールは逐次プログラムより少し良い程度である．一方，SWB-P ではモジュール 7 を除いて著しい性能低下が起きており，最悪の速度向上比は 0.21 である．モジュール 7 は粗粒度の並列性を持ち GSM アクセスが少なく，限界値も 2.30 である．HWB-P では限界値近くまで速度が向上しているが，SWB-P ではバリア同期によるオーバーヘッドのために速度向上比は 1.65 である．

モジュール 11, 8 は中粒度，モジュール 3, 2, 6, 9 は細粒度の並列性を持っている．これらのモジュールではソフトウェアによるバリア同期処理時間がバリア同期間の本来の処理時間と比較して大きいことが性能低下の原因になっている．すなわち SWB-P では粒度が小さいほどバリア同期によるオーバーヘッドが顕著に現れてくる．モジュール 11, 8 は並列化部分が少ないのでその限界値は 1.04, 1.14 と小さいが，HWB-P では限界値に近い速度向上比 1.03, 1.11 を得ている．モジュール 3, 2, 6 は並列化部分は 60～70%であるが，大きい SMWR のために通信によるオーバーヘッドが相対的に大きく，速度向上比は限界値 1.85, 1.83, 2.33 よりも 1.14, 1.14, 2.02 と低くなっている．モジュール 9 は並列化部分が 99%で限界値は 2.94 であるが，最も細粒度であるうえにデータ依存が逆依存関係にある．すなわち逆依存解消のための一時変数の導入によるオーバーヘッド，非常に短いバリア同期間隔のほかにすべての PU が同時に GSM に書き込むことによる競合の発生のために速度向上比は 1.18 と小さい．このように HWB-P における速度向上比の低下は共有メモリ構成が主な原因である．そして MCAM によるバリア同期機構を用いた SsBM は細粒度並列処理に有効であることが示された．なおモジュール 8 以外のモジュールもより細かい機械語の並列化を行えば，もう少し速度向上比の向上が見込まれる．

表2 WhetstoneのSsBM, SmBMによる速度向上比
Table 2 Speedup ratios of Whetstone under SsBM, SmBM.

	SsBM	SmBM
PU数	5	7
速度向上比	1.32	2.35

4.2.3 プログラム全体における評価

Whetstoneプログラムの各モジュールをプロセジャ(モジュールプロセジャとよぶ)とし，プログラム全体をタスクとする．SsBMおよびSmBMによる並列化タスクを逐次タスクSと比較し評価する．Sに対するSsBM, SmBMの速度向上比を求め，その結果を**表2**に示す．

SsBMでは並列化されたプロセジャを逐次実行する．すなわち，5台のプロセッサでプロセッサグループおよびバリアグループを構成して各モジュールプロセジャを逐次実行している．SsBMにおける同期通信のオーバーヘッドを除いたときの理想的な速度向上比は1.54である．これはプログラム全体の実行時間の40%近くを占めるモジュール8の速度向上比が低いことが主な原因である．速度向上比の実測値は1.32であり，理想値より少し低い程度であった．

一方SmBMでは7台のプロセッサでプロセッサグループを構成し，5台のプロセッサで1番目のバリアグループを，2台のプロセッサで2番目のバリアグループを構成する．モジュール2, 3, 6, 7, 9, 10, 11が1番目のバリアグループで，モジュール1, 4, 8が2番目のバリアグループで実行される．モジュール2, 3, 7, 9, 11は並列化されたプロセジャである．またモジュール1, 4, 10は逐次プロセジャであるのでバリア同期は存在しない．1番目のバリアグループで実行されるモジュールプロセジャと2番目のバリアグループで実行されるモジュールプロセジャが並列に実行される．各モジュールのThwbpから予測されるモジュール実行のクリティカルパスは2, 3, 6, 8であり，逐次タスクに対するSmBMの予測速度向上比は2.37である．モジュール自身の並列化に加えてモジュール間の並列化を行っていることにより，大きく性能が向上している．速度向上比の実測値は2.35であり予測値と大きく違わない．予測値がバリア同期機構でのバリアグループ間の干渉が無いときの値であることを考えると，そういった干渉による性能低下が起きていないことがわかる．

このようにSmBMはPU数が増大するが速度向上比に優れ，SsBMは速度向上比は劣るがPU数を抑えることができる．

表3 多重実行タスクの単独実行タスクに対する実行時間比
Table 3 Execution time ratio of multiple tasks to single task.

測定タスク(module)	同時実行タスク(module)						
	7	11	8	3	2	6	9
7	1.00	1.00	1.00	1.00	1.00	1.01	1.00
11	1.00	1.00	1.00	1.00	1.01	1.01	1.01
8	1.00	1.00	1.00	1.01	1.01	1.02	1.01
3	1.01	1.00	1.02	1.03	1.02	1.02	1.02
2	1.01	1.00	1.00	1.02	1.02	1.08	1.02
6	1.01	1.01	1.04	1.03	1.05	1.01	1.07
9	1.02	1.01	1.04	1.03	1.02	1.09	1.00

4.2.4 多重プログラミング環境での評価

多重プログラミング環境下でのタスクの多重実行時には，MCAMによるハードウェアバリア同期機構によりバリア同期の競合による性能低下は起こらないが，GSM書き込み時のバス競合による性能低下が予想される．2つのモジュールを多重実行した場合の実行時間を単独実行時の実行時間と比較し，多重プログラミング環境の性能評価を行う．1つのモジュールタスクがGSMバス優先権の高いPU群で繰り返し実行されている状態で各モジュールタスクを任意のタイミングで実行開始させたときの平均実行時間Tmを測定し，実行時間比Tm/Thwbpを算出した．その結果を**表3**に示す．

各タスクのGSMバスアクセス間隔のばらつきから，競合の発生にランダム性があり平均的な性能低下は小さくなっており，最大でも1.09倍である．SMWRが小さなタスクの性能低下はわずかである．SMWRが大きなタスクどうしでの性能低下は少し大きなものがあるが，その平均値は1.02であり多重実行ではほとんど影響はないと見なせる．またGSMバスアクセス間隔がほぼ等しい同一モジュールタスクどうしの多重実行では大きな性能低下が起きていない．これは一度競合が発生すればGSMバスアクセスにずれが生じ，以後，競合の発生が少なくなるためと考えられる．

以上から多重プログラミング環境におけるGSM構成の有効性が確認できた．

5. むすび

バリア同期管理方式としてSsBM, SmBMの管理形態からなるSBMを示した．さらに静的コードスケジューリングされたタスクの多重プログラミング環境を実現する同期通信機構として，MCAMによるハードウェアバリア同期機構とGSM構成を提案した．そしてテストベッドとして開発した並列計算機MSBMを使用して，細粒度doacrossループプログラムによ

る評価を行った結果，16PUで8.39倍の速度向上を得た．さらに，Whetstoneベンチマークプログラムの，モジュールタスクにおける評価，SsBMとSmBMでのプログラム全体における評価，および多重プログラミング環境での評価を行った．その結果，モジュール単体ごとのSsBMでは1.03倍から2.29倍の速度向上がみられ，ソフトウェアバリア同期によって多くのモジュールで速度低下がみられた．また，Whetstone全体の並列処理では逐次処理の場合に対して，SsBMで1.32倍，SmBMでは2.35倍の速度向上がみられ，SmBMによる多重処理の有効性が示された．一方，2つの細粒度タスクの多重実行では，GSM構成によって最悪で単独実行時の1.09倍の実行時間に抑えられることが認められた．以上から，MCAMによるハードウェアバリア同期機構とGSM構成の多重プログラミング環境での有効性が確認できた．

今後の課題は，ファジーバリア同期の有効性の確認，タスク実行中にバリアグループを半動的にあるいは動的に処理することで並列処理効率を向上させることである．

謝辞 本研究を遂行するにあたり多大なご助力ならびにご支援をいただいた（株）東芝北九州工場の方々に深謝いたします．

参考文献

1) 笠原博徳：並列処理技術，コロナ社 (1991).
2) 笠原博徳，成田誠之助，橋本　親：OSCAR (Optimall Scheduled Advanced Multiprocessor) のアーキテクチャ，電子情報通信学会論文誌 D，Vol.J71-D, No.8, pp.1440–1445 (1988).
3) Lundstrom, S.F.: Applications Considerations in the System Design of Highly Concurrent Multiprocessors, *IEEE Trans. Comput.*, Vol.C-36, No.11, pp.1292–1309 (1987).
4) Gupta, R.: The Fazzy Barrier: A Mechanism for High Speed Synchronization of Processors, *Proc. Third Int. Conf. on ASPLOS*, pp.54–63 (Apr. 1986).
5) 松本　尚：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).
6) 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，電子情報通信学会研究会資料 CPSY91-15, pp.91–97 (1991).
7) O'Keefe, M.T. and Dietz, H.G.: Hardware Barrier Synchronization: Static Barrier MIMD (SBM), *1990 Int. Conf. on Parallel Processing*, Vol.1, pp.35–42 (1990).
8) O'Keefe, M.T. and Dietz, H.G.: Hardware Barrier Synchronization: Dynamic Barrier MIMD (DBM), *1990 Int. Conf. on Parallel Processing*, Vol.1, pp.43–46 (1990).
9) Dietz, H., Schwederski, T., O'Keefe, M. and Zaafrani, A.: Static Synchronization Beyond VLIW, *Supercomputing 89*, pp.416–425 (1989).
10) 高橋義造編：並列処理機構，丸善株式会社 (1989).
11) 岩根雅彦，重松保弘，定家健治，茶屋道宏貴，金沢高貴：FPGAによるバリア同期用機能メモリの開発，九州工業大学研究報告（工学），No.66, pp.45–52 (Mar. 1994).
12) El-Rewini, H., Lewis, T.G. and Ali, H.H.: *Task Scheduling in Parallel and Distributed Systems*, Prentice Hall (1994).
13) Curnow, H.J. and Witchmann, B.A.: A Synthetic Benchmark, *Comput. J.*, Vol.19, No.1, pp.43–49 (1976).
14) Monica, L.: Software Pipelining: An Effective Scheduling Technique for VLIW Machines, *SIGPLAN Conf. on Programming Language Design and Implementation*, ACM (June), Atlanta, GA, pp.318–328 (1988).
15) Hennessy, J.L. and Patterson, D.A.（富田眞治，村上和彰，新實治男訳）：コンピュータ・アーキテクチャ—設計・現実・評価の定量的アプローチ—，日経BP社 (1992).

(平成6年12月14日受付)
(平成8年3月12日採録)

岩根　雅彦（正会員）

1946年生．1968年京都大学工学部数理工学科卒業．同年（株）東芝に入社．1980年同社退職後トヨタ自動車（株），豊田工業大学を経て，1988年より九州工業大学工学部教授．工学博士．計算機アーキテクチャの研究に従事．電子情報通信学会会員．

本石　彰（学生会員）

1970年生．1994年九州工業大学工学部電気工学科卒業．1996年九州工業大学大学院工学研究科電気工学専攻博士前期課程修了．同年（株）ニコンに入社．並列計算機アーキテクチャの研究に従事．

野口 善昭（学生会員）

1971 年生．1995 年九州工業大学工学部電気工学科卒業．現在，同大学大学院工学研究科電気工学専攻博士前期課程在学中．並列計算機アーキテクチャの研究に従事．またオペレーティングシステムとプログラミング言語にも興味を持つ．

米澤 敏夫

1944 年生．1968 年金沢大学理学部物理学科卒業．同年（株）東芝に入社．1992 年同社北九州工場生産技術部長．1994 年より同社四日市工場工場長．1974 年大内記念賞，1979 年全国発明特別賞を受賞．半導体関連技術の開発，研究に従事．