計算機アーキテクチャ　107－1
（１９９４．　７．　２１）

細粒度マルチマイクロプロセッサMSBMの構成と性能評価

本石　彰†，濱田智雄†，岩根雅彦†，米沢敏夫††

†：九州工業大学工学部　　††：㈱東芝北九州工場

ハードウェアバリア同期，グループ共有メモリを細粒度並列処理で評価するテストベッドとして細粒度マルチマイクロプロセッサ MSBM を開発した．MSBM は静的スケジューリングされたジョブを多重実行する．16PU(Processing Unit), IIU(Integrated Interface Unit) および HC(Host Computer)から構成され，IIU は拡張CAM によるハードウェアバリア同期機構を，各PU は DPM によるグループ共有メモリ構成を持つ．Whetstone ベンチマークプログラムを用いた性能評価の結果，ハードウェアバリア同期による並列処理は最高 2.02 倍の速度向上がみられその有効性が明らかになった．ジョブの多重実行では最悪 1.086 倍の応答時間増加がバス競合によって起きた．

Architecture and Performance Evaluation of MSBM : A Fine-Grained Multi-Microprocessors

Akira MOTOISHI†, Tomoo HAMADA†, Masahiko IWANE†, Toshio YONEZAWA††

† : Faculty of Engineering, Kyushu Institute of Technology
†† : Kitakyushu Works Toshiba Corporation

The fine-grained multi-microprocessors: MSBM (Multiple Static Barrier MIMD) were developed as the test bed evaluating the hardware barrier synchronization (HBS) and the group shared memory (GSM). It multiply executes the jobs scheduled statically. It consists of 16PU, Integrated Interface Unit (IIU) and Host Computer. IIU has the HBS-mechanism using a modified CAM, and PU has the GSM-configuration using a DPM. The results of the performance evaluations using the Whetstone benchmark programs show that a paralell processing by the HBS improves the processing speed (2.02 times), and the multiple execution of the jobs increases these response times (1.086 times).

1．はじめに

並列化コンパイラによって，プログラム全域からの潜在的並列性抽出およびスケジューリングをスタティックに行い，複数プロセッサで並列処理させることが注目されている[1][2]．この場合その並列性は数命令程度の細粒度となることが多い．しかし細粒度並列処理を効率よく行えるマルチプロセッサは少ない．

細粒度並列実行ではプロセッサ間のデータ通信と同期が頻繁に行われ，これが並列処理のボトルネックとなる．OSCAR[3]は命令実行時間の推定が容易なプロセッサを統一クロックで動作させることで処理時間の正確な推定を可能とし，バスアクセスのタイミング制御をコンパイラで行い同期動作を排除している．しかし，これには精密なスケジューリングが求められるため，何らかの同期動作を行う方がスケジューリングの負担軽減になる．そのためには，効率を低下させないように同期機構を高速なハードウェアによって実現することが不可欠となる．エラスティックバリア[4]はマスクと複数カウンタで任意のプロセッサ間の同期を可能としているが，配線数，回路量が増大する．SBM[5]はバリア同期間に完全順序関係を導入しバリアキューによって単一のバリア同期列として任意のプロセッサ間の同期を可能としており，DBM[6]はバリア同期間に半順序関係を導入しバリアキューとCAMによって任意のプロセッサ間の複数のバリア同期を同時に可能としているが，これらはループの繰り返し回数が不定なときなどでは正しく同期がとれないことがある．

プロセッサ間通信は高速な結合網としてクロスバスイッチなどが最良であるがハードウェアコストが大きい．共有メモリの場合，プロセッサとの結合方式は単一バスがハードウェアコストを抑えられるが，バスやメモリポートでの競合が頻発するため，キャッシュメモリおよびバスの多重化などによりそれを緩和することが多い．しかしこの場合ハードウェア量の増大およびキャッシュコヒーレンスの問題が発生する．OSCARは3本のバスによるプロセッサエレメント間直接通信などで転送能力を高めている．分散型共有メモリは共有メモリのアドレスを分割して各プロセッサに分散するため，効率の良い共有データの配置を考慮しなければならない．

以上からハードウェアによるバリア同期機構とグループ共有メモリ機構を提案し，それらの細粒度並列処理における評価，および並列化コンパイラ開発のためのテストベッドとして細粒度並列計算機ＭＳＢＭ(Multiple Static Barrier MIMD)を開発した．

本稿では，2章で基本設計について述べ，次いで3章ではＭＳＢＭのハードウェア構成を述べる．そして4章ではWhetstoneによるバリア同期，GSM機構ハードウェアの性能評価について記し，最後に5章でむすびを述べる．

2．基本設計

2.1. スタティックバリア

並列処理される1つの応用プログラムは主プログラムと複数のサブプログラム(サブルーチンやループ)から構成され，これらをプロセデュアと呼ぶ．OSにより実行される1つの応用プログラムをジョブと呼び，1ジョブに割り当てられたプロセッサの集合をグループと呼ぶ．

全てのプロセッサ間同期は，コンパイラによるスタティックスケジューリングによって実行前にバリア同期命令を命令流に挿入しておくことで行う．そして，ジョブの開始前にバリア同期機構に対してジョブ内で行われるバリアパターン(バリア同期を行うプロセッサを示すパターン)を登録し，実行中に動的に変更したりせずに終了後に削除する．この方式をSBM(Static Barrier MIMD)方式と呼ぶことにする．ここでは，SBMのバリア同期形態の異なる2つの方式，sBM(single-Barrier MIMD)とmBM(multiple-Barrier MIMD)を考える．

sBMはバリア同期を行う場合常にジョブ内の全プロセッサで行わせるものである．つまり1ジョブ (1グループ)にただ1つのバリアパターンが対応する．mBMは複数個のバリアパターンを1ジョブ(1グループ)に対応させ，複数同期を並列独立に行うものである．1プロセデュア内ではsBM同様に全プロセッサでバリア同期をとらせ，並列化可能なプロセデュアにそれぞれ1バリアパターンを対応させ並列実行する．ただしプロセッサはジョブ内のどれか1つのバリアパターンにのみ属することになる．このプロセデュア単位の並列性もコンパイラで抽出し，バリアパターンもスタティックスケジューリングで実行前に決定する．

また，SBMのような並列化では並列度が低いことが多いため，無使用のプロセッサを別のジョブに割り当て，SBMジョブを多重実行し，MSBM(Multiple SBM) とすれば，システム全体のスループットが向上することが見込まれる．

2.2. バリア同期

2.2.1. バリア同期のハードウェア化

バリア同期は同期をとる全プロセッサの実行が同期位置に達するまで先に到達したプロセッサが待機することで，どのプロセッサの実行も同期位置を越えることができないようにするものである．細粒度並列処理に共有変数などを利用したバリア同期（ソフトウェアバリア同期）を用いるとオーバーヘッドが大きくなり並列処理によって逆に実行速度が低下するという事態が起こりうる．よってバリア同期機構を高速な専用ハードウェアによって実現しオーバーヘッドを縮小する．

また，本来は一部プロセッサ間の同期だけで良い場合でもsBM,mBMのように全プロセッサで同期をとるようにすること，つまり識別の必要がない単一のバリアで同期をとることは，同期識別情報を不必要としハードウェア処理をシンプルにすることができる．しかしこのために，本来は同期をとる必要がないプロセッサであっても，図1のように他のプロセッサが同期をとるならば同期を行わせる必要がある．

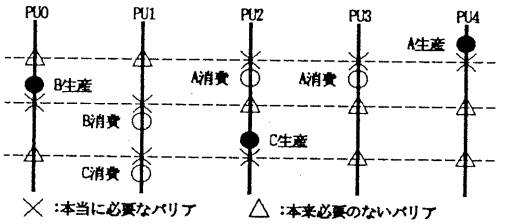

図1. 常時全プロセッサによるバリア同期

2.2.2. バリア同期の拡張

SBMのコンパイラは実行時間の推定を行いバリア命令を設定する．しかしこれを厳密に行うことは容易ではなく，行えても実行前に予測不可能な遅れが発生することも考えられる．そこで通常のバリア同期の他にファジーバリア(Fuzzy Barrier)同期[7]をサポートする．ファジーバリア同期は，通常のバリア同期のように一点で待ち合わせる同期ではなく，広がりのある領域内での同期を許すものである．つまり，同期をとる全てのプロセッサの実行が同期領域に突入するまで，先に同期領域の終点に実行が達したプロセッサが待機することで，同期を取るどのプロセッサも同期領域を脱することができないようにするものである．

このファジーバリア同期を実現するために，通常のバリア同期命令を拡張したBar（Barrier）命令および，AdBar（Advanced Barrier）命令を全プロセッサに用意し，Bar命令単独で通常のバリア同期を，AdBar命令とBar命令でファジーバリア同期を実現する．先だってAdBar命令を実行せずにBar命令を実行したとき，同期成立検出機構に同期要求を行いプロセッサの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．プロセッサがAdBar命令を実行した場合，同期成立検出機構に同期要求を行うがプロセッサの実行は中断しない．プロセッサがAdBar命令を実行後にBar命令を実行したとき，既に同期成立検出機構から同期成立が伝えられていればプロセッサはその実行を中断することはない．同期成立が伝えられていなければプロセッサの実行を通常のバリア同期同様に中断し同期成立が伝えられるまで待つ．このようにAdBar命令とBar命令の間をバリア領域とするファジーバリア同期命令を実現する．図2はファジーバリア同期を実現する様子をそれぞれの示している．

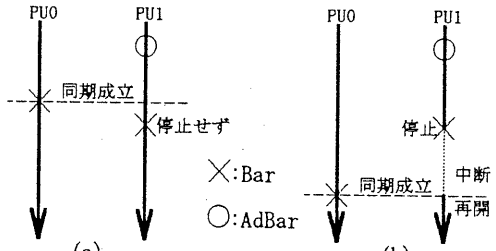

図2. Bar,AdBarによるファジーバリア同期

2.2.3. MCAMによるバリア同期機構

性能低下を起こさないためにはお互いの同期動作に干渉しないように複数のバリアパターンの同期成立を並列，高速に検査する機構が必要である．

拡張CAM（MCAM：Modefied Content Addressable Memory)[8]は検索データと包含関係のある記憶データを読み出すメモリである．MCAM内のi番目の記憶データMiと検索データSのj番目のビットをそれぞれMij，Sjとすると包含関係は次式のように表される．

$$Mij \subseteq Sj = \neg Mij \vee Sj$$

全てのビットjでこの関係が成り立つときMiはSに包含されることになり，MCAMの出力QはSに包含される全てのMiの論理和となる．

集中処理型のハードウェアバリア同期機構をこのMCAMを用いて構成する．

MCAMには図3のように，データビット長がプロセッサ数と一致するものを使用し，データMiの各ビットを個々のプロセッサに対応させる．このMCAMにバリア同期をとるプロセッサのパターン(バリアパターン)を登録しておく．また，Miと同様

なプロセッサとの関係を持つ２つのレジスタBRR(Barrier Request Register),BGR(Barrier Grant Register)を用意する．BRRは各ビットが対応するプロセッサのバリア同期要求を受けセットされ，現在バリア同期要求を行っているプロセッサのパターンを表す．このBRRをMCAMの検索データとして包含検索を行い成功した時，包含されたバリアパターンに対応する同期が成立したことなる．そして，包含される全バリアパターンの論理和がMCAMからBGRに出力され，BGRが同期が成立したプロセッサを表すことになる．このBGRのビットパターンを各プロセッサに送ることで同期成立を伝える．

このMCAMによるバリア同期機構では任意のプロセッサで同期をとれ，さらに数パターンの同期検出が並列かつ独立して行われる．また，包含される全てのバリアパターンの論理和が出力されるため，一度の検索処理で複数同期の成立を伝えることができる．

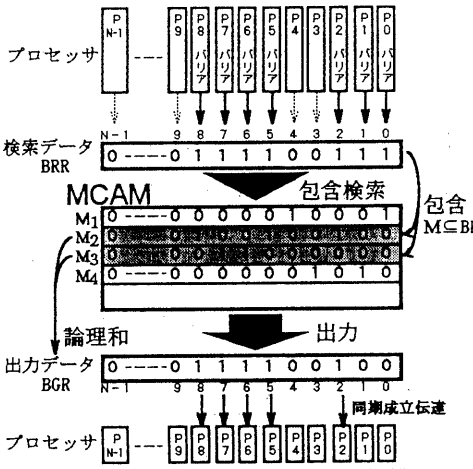

図3.　MCAMによるバリア同期機構

2.3. グループ共有メモリ

プロセッサ間データ通信は共有メモリを介す方法がシンプルであるが，頻繁にデータ通信が発生する細粒度並列処理行，さらにはその多重実行を効率よく行うには，共有メモリモジュールやプロセッサとの相互接続網で競合をできる限り抑える機構が求められる．

グループ共有メモリ(GSM：Group Shared Memory)構成の概略を図4に示す．GSM構成は各プロセッサのメモリの内容をグループごとに完全一致させることでグループごとの共有メモリを実現する．共有データの参照時は常に自身内のメモリから読み出せばよいため，不要なバスアクセスを排除できバス競合による性能低下を軽減できる．また，各プロセッサでは共有メモリ領域を局所メモリ(LM：Local Memory)と同様に直接メモリ空間にマッピングするため，通常のメモリ命令でアクセスできる．GSM内容のグループ内一致は，あるプロセッサがGSMに書き込みを行ったとき共有バス(GSMバス)に所属グループの番号を出力させ，これを識別情報とするマルチキャストを同一アドレスへ行うことで実現される．GSM領域への書き込みかLM領域への書き込みかの判断は各プロセッサ内のレジスタ(SMBASE)とアドレスを比較することで行う．メモリデバイスとして2ポートメモリ(DPM:Dual Port Memory)を用い，SMBASEレジスタによりLMとGSMとに分ける．DPMのportAをプロセッサ内部に，portBをGSMバスに接続することで，GSM内容の一致のためのGSMバスを介したメモリ書き込が，プロセッサ内部かのメモリアクセスを阻害することがなくなり性能低下を軽減できる．

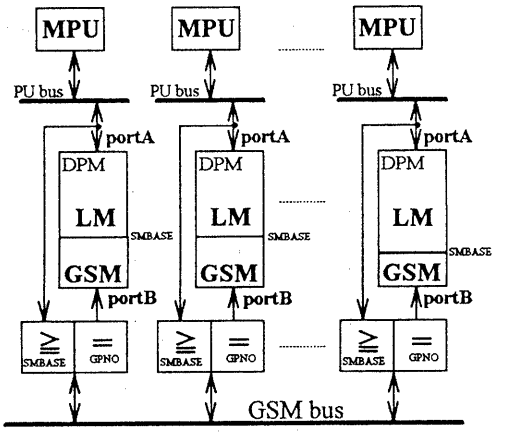

図4.　GSM構成

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は再開させるものである．あるプロセッサがlock要求またはunlock要求を発行すると，その要求を属すグループの番号と共に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，グループ(ジョブ)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. システム概要

ＭＳＢＭは図5に示すようにホストコンピュータ(HC:Host Computer)，統合インターフェイスユ

ニット(IIU:Integrated Interface Unit), および16台のプロセッシングユニット(PU:Processing Unit)から構成されている.

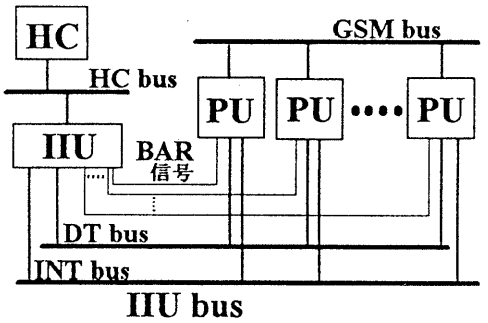

図5. ＭＳＢＭのハードウェア構成

HCにはi386MPUで構成されたパーソナルコンピュータを用いている. IIUとの間はHCバスで接続され, ユーザーインターフェイス, PUの統括管理などを行う. またバリア同期機構の一部としてバリアパターンの管理も行う.

IIUはHC-PU間のインターフェイスおよび集中処理型バリア同期機構の役割を担っている.

PUは割り当てられた逐次命令流を各々独立したクロックで実行する既製のMPUを中心とするユニットである. 個々にメモリを所有しプログラムコードやデータを格納する. バリア同期機構のPU側インターフェイス, GSM構成とlock/unlock機構のGSM機構, HCからPUにデータをマルチキャストさせる機構, およびPUからIIUを介してHCに割り込み処理を要求する機構などが備えられている.

HC-IIU間はHCバスで, IIU-PU間はIIUバスで, PU相互はGSMバスでそれぞれ結合されている. HCバスは, HC-IIU間のデータやコマンドなどの転送に使用される. IIUバスは, DTバス, INTバスおよび, BAR信号線から構成されている. DTバスは, IIU-PU間のデータおよびコマンドの転送に使用される. INTバスは, PUからIIUを介してのHCへの割り込み要求およびそれに伴うパラメータデータの転送に使用される. BAR信号線は各PU個別に16本存在し, 対応するPUからはバリア要求を, IIUからは対応するPUにバリア同期成立を伝える役目をする. GSMバスはGSM機構のための共有バスであり, GSMの内容更新およびlock/unlock要求に使用される.

3.2. 統合インターフェイスユニット(IIU)

IIUの構成を図6に示す. IIUは, HC-PU間の通信を行うデータ転送ユニット(DTU:Data Transfer Unit), PUからHCへの割り込み処理の中継役を行う割り込み処理ユニット(INTU:Interrupt Unit), およびMCAMを用いたPU間バリア同期成立検出行うMCAMユニット(MCAMU:MCAM Unit)から構成されている. DTUはDTバスと, INTUはINTバスと接続されている. また, 各PU個々からのＢＡＲ信号線はMCAMUに接続されている. またMCAMなどの各ユニットは, FPGA(Field Programmable Gate Array)を用いて製作されている.

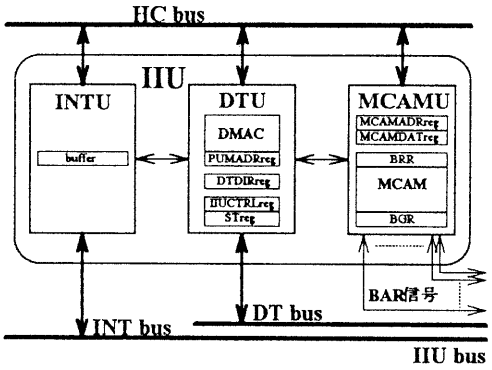

図6. IIUの構成

DTUはHCバスおよびDTバスを使用して行われるHCメモリとPUメモリの間のDMA(Direct Memory Access)転送を行うユニットである. そのためDTUにはDMAC, PUメモリアドレスレジスタ(PUMADRreg), 転送方向レジスタ(DTDIRreg)などが存在しHCから制御される. HCメモリに対するアドレスはDMACが出力し, PUメモリに対するアドレスはDMACと同調してインクリメントするカウンタであるPUMADRregが出力しDMA転送時を行う. また, HCからIIU, PUへのコマンドに対する処理も行い, IIUやPUを制御するためのレジスタ(IIUCTRLreg)や状態を示すレジスタ(STreg)が存在する.

INTUはPUからHCに対する割り込み要求処理の仲介を行う. PUからの要求を受けるとINTバスを介して2byteのパラメータデータを受け取りバッファに格納した後HCに割り込みをかける. バッファはHCバスに接続されており, HCは読み込むことが可能である.

MCAMUは各PUとバリア同期要求/成立を示す1本のBAR信号線で個々に接続されている. MCAMUはMCAMによるバリア同期成立の検出シーケンスを独自のクロックによって繰り返し行っている. MCAMへバリアパターンの登録は８エントリまで可能であり, MCAMU内のレジスタ(PUADRreg, PUDATreg)を介しアドレスを指定してHCからのみ行うことができる.

3.3. プロセッシングユニット(PU)

PUの構成を図7に示す. PUはMPU, メモリ, PU内部レジスタ, DTUi/f(i/f:interface), INTUi/f, GSMi/f,

およびBARi/fから構成されPUバスで接続されている．IIUと各PUは，DTバス，INTバスおよびBAR信号線で接続されており，それぞれDTUi/f，INTUi/fおよびBARi/fがインターフェイスとなっている．PU同士はGSMバスで接続されており，GSMi/fがそのインターフェイスとなっている．またこれらインターフェイスおよびレジスタは，FPGAを用いて製作されており，4PUで1枚のPUボードを形成している．

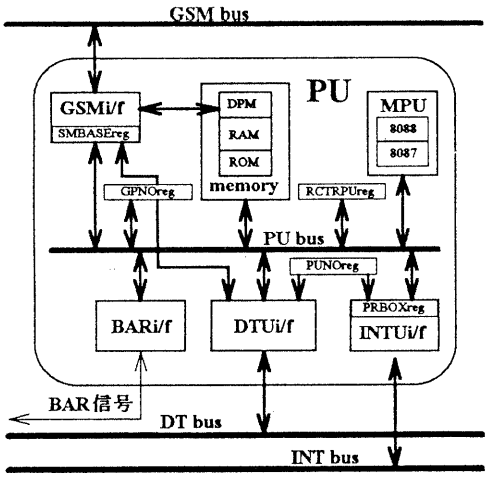

図7. PUの構成

MPUには8088MPU+8087FPUを使用している．メモリはコスト低減のためにROM:32KB，RAM:208KBおよびDPM:16KBで構成されている．ROMには初期化およびHCとの基本的なインターフェイスなどを行うPUモニタが格納されている．RAMおよびDPMの一部で局所メモリ(LM：Local Memory)を構成しプログラムコードやローカルデータを格納する．DPMの残りでグループ内の共有データを格納するGSMを構成している．PUには各種目的用に内部レジスタが用意されいる．PUNOregはPU固有の値でありDIPスイッチで物理的に設定される． GPNOregは属すグループの番号を格納する．RCTRPUregはHC，PUから設定，参照が可能であり，HCからのPUへの簡単な通信に利用可能である．SMBASEregはGSMi/fで使用するGSMの開始番地を格納する．PRBOXregはINTUi/fで使用されHCへの割り込み(SVCC)時にHCに提示するパラメータを格納する．

DTUi/fはHC-PU間データ転送時のPUバス解放，ゲート開閉などの処理を行い選択したPUのメモリ，内部レジスタに対するデータのマルチキャストを実現している．HCでデータ転送対象PUを指定するOUT命令(SELPU)が実行されると，DTUは全PUのDTUi/fに出力データを転送する．指定はPU番号(PUNO)，グループ番号(GPNO)，および全PUであり，SELPUのデータに識別ビット，PUNOまたはGPNOを含ませる．各PUのDTUi/fはそれをデコードしPUNOreg，GPNOregと比較し選択されているかを判断する．選択されたPUのみがPUバス解放，DTバスとPUバスを接続し，DTUもしくはHCにメモリまたは内部レジスタのアクセスを許す．この接続の際，DTUの転送方向レジスタから出力される信号(HC→PU，PU→HC)を用いてゲートのデータ方向が設定される．

INTUi/fはINTUとINTバスで接続され，PUでHCへの割り込み処理要求命令(SVCC:SerViCe Call)が実行されたときINTUに割り込み要求を行う．その後INTUから応答があるとPRBOXレジスタの内容を1INTバスを介してINTUに送る．このPRBOXreg2byteの内4bitはHCがSVCCを要求したPUを認識出来るようにPUregの内容が自動的に格納される．FUからINTUへの処理要求の調停はPUをディジーチェーン接続することで行われる．

GSMi/fはGSM構成とlock/unlock機構を実現する部分である．この2種類の処理はともにGSMバスのグループ番号バスを使用するため同一の使用権調停が行われる．この調停はPUNOregの値を使用したFuterbus方式[9]で行われる．lock命令，unlock命令はMPUのOUT命令で実現され別々のI/Oポートに割り当て区別する． lock/unlockによるMPUの実行中断，再開はMPUへのWAIT，READY要求で行う．

BARi/fはIIUのMCAMUと1本のBAR信号線で接続されている．MCAMUへのバリア同期要求，成立信号監視およびWAIT要求によるMPUの実行中断，再開などの処理を行っている．ここで通常のバリア同期とファジーバリア同期を実現している．Bar命令，AdBar命令はMPUのOUT命令で実現され別々のI/Oポートに割り当て区別する．BAR命令の動作は各種フラグの状態で決定される．

4．性能評価

4.1. Whetstone[10]の並列化

Whetstoneプログラムは異なる特徴を持つ複数のモジュールで構成されている．そこで性能評価は各モジュールを1のジョブとみなして行う，また，ベクトル化やステートメントレベルの並列化が不可能なため，並列処理コードの生成を行うためにはコンパイル手法に高度なものが要求される．今回は細粒度スケジューリング手法[11]に従い人の手により並列版のコードを生成した．以下に今回の評価に使用した各モジュールの並列版コードの特徴を記す．ただしGSMWRとはジョブ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．

module2 ：実数配列変数の四則演算．並列度4．GSMWR:7.27%．細粒度．

module3 ：module2と同一演算をサブルーチンコール．並列度4．GSMWR:6.04%．細粒度．

module6 ：整数四則演算と変数型変換．並列度5．GSMWR:5.58%．細粒度．

module7 ：三角関数の四則演算．並列度4．GSMWR:1.52%．比較的粒度大．

module8 ：関数呼出しと実数四則演算．並列度2．GSMWR:4.15%．粒度中間的．

module9 ：関数呼出しと配列参照．並列度3．GSMWR:5.93%．細粒度．

module11：標準関数(sqrt, exp, ln)．exp関数のみ並列化．並列度3．GSMWR:6.33%．比較的粒度大．

4.2. ハードウェアバリア同期機構の評価

HWB-PではsBM方式で通常のバリア同期のみの設定を行った．SWB-Pでは共有メモリ上のカウンタによってバリア同期を実現した．カウンタの更新時には相互排除処理(lock/unlock)を行い，カウンタの参照時はDPMによるGSM構成を活かしバスアクセスなしに行っている．各モジュールのS, HWB-P, SWB-Pを単独で実行した時のそれぞれの平均応答時間:Ts, THswp, Tswbpを測定し，Sに対するHWB-P, SWB-Pの速度向上比：Ts/Thwbp, Ts/Tswbpを算出した．その結果を図8に示す．

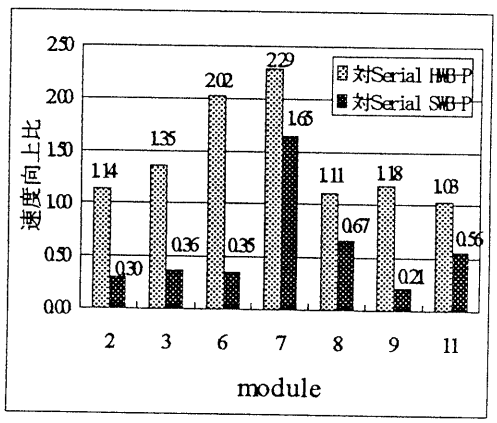

図8. 逐次版Sに対するHWB-P, SWB-Pの速度向上比

ハードウェアバリア同期による並列版(HWB-P:Hardware Barrier-Parallel)ジョブとソフトウェアバリア同期による並列版(SWB-P:Software Barrier-Parallel)ジョブを逐次版(S:Serial)ジョブと比較し，ハードウェアバリア同期機構の評価を行う．

ハードウェアバリアによる並列処理(HWB-P)によって，最高で2.29倍(モジュール7)の性能向上が認められた．比較的細粒度のモジュール6でも2.02倍の性能向上がみられた．モジュール11, 8は粒度の大きさに比べ速度向上比が1.03倍, 1.11倍と性能の向上が小さい．これはこれらのモジュールに並列化可能な部分が非常に少ないためである．

また，HWB-Pが全てモジュールにおいて逐次版より速度が向上しているのに対して，SWB-Pはほとんどのモジュールにおいて著しい性能低下が起きており，最悪で0.21倍(モジュール9)となっている．これは，これらのジョブとくに細粒度ジョブであるモジュール2, 3, 6, 9における同期と同期の間の処理時間がソフトウェアバリア同期の処理時間に比較して大きなことが主な原因であると考えられる．性能が向上しているモジュール7では同期と同期の間の処理時間が同期処理時間と比較して大きく，並列化が有効となっている．また，その直接的なオーバーヘッドに加え，各PUが同期のたびに相互排除を要求し他のPUの実行を阻害していることも一因と考えられる．HWB-Pでのモジュール6は並列度が大きいため，比較的細粒度であるにも関わらず大きな速度向上(2.02倍)を示している．しかしSWB-Pではこの高並列度の効果が現れていない．これは共有カウンタ更新時の相互排除による性能低下が並列度に比例するためと考えられる．しかし，共有メモリ参照時もバスアクセスするようなメモリ構成ならばさらに性能が悪化するものと思われる．

以上のように細粒度並列処理にはハードウェアによるバリア同期機構実現が不可決であることおよび，MCAMによるハードウェアバリア同期機構が有効であり，通常のバリア同期のみ設定でも効果があることが示された．

4.3. ジョブ多重実行

ジョブを多重実行する場合予測不可能なGSMバスの競合による性能低下が起こると予想される．2つのHWB-Pジョブを多重実行させた場合の応答時間を単独実行時の応答時間と比較し，性能評価を行う．

各ジョブと多重実行させた時の平均応答時間:Tmを測定し，応答時間増加率:Thwbp/Tmを算出した．その結果ををを表1に示す．

モジュール7, 11, 8の性能低下は極わずかであり，粒度が大きいとGSMバスの競合が少なく多重実行によって大きな性能低下が起きないことが確認できた．逆に，細粒度のジョブ同士を多重実行させた場合の性能低下が比較的大きく，最悪でモジュール6と同時実行したときのモジュール9の応答時間増加率が1.086倍であった．同じく細粒度のモ

表1. 多重実行ジョブの単独実行ジョブ(HWB-P)に対する応答時間比

測定ジョブ	同時実行ジョブ						
	mod2	mod3	mod6	mod7	mod8	mod9	mod11
mod2	1.01544	1.01509	1.07691	1.00513	1.00114	1.02439	1.00485
mod3	1.01646	1.03344	1.02805	1.00806	1.01581	1.02601	1.00332
mod6	1.04581	1.02302	1.00603	1.01170	1.03963	1.07391	1.00625
mod7	1.00194	1.00133	1.00679	1.00064	1.00164	1.00431	1.00027
mod8	1.00712	1.00385	1.01624	1.00234	1.00002	1.00580	1.00601
mod9	1.04488	1.03247	1.08588	1.01599	1.03925	1.00036	1.00578
mod11	1.00504	1.00349	1.01187	1.00136	1.00222	1.00568	1.00705

ジュール2,3の多重実行でのそれぞれの増加率は1.015倍，1.016倍と大きな低下なしに多重実行が行えている．

数ジョブ程度の多重実行ならば，粒度が大きいジョブはほぼ性能低下なしに多重実行が可能であるが，細粒が小さいジョブの多重実行では大きいと言えなくとも無視できない性能低下が起こりうることが判明した．

5．むすび

プログラムを並列処理する方式として，実行前に全ての並列性を抽出するSBM方式を示し，細粒度並列処理を効率よく行う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機構を示した．そして，それらの開発，評価のためのテストベッドとして開発され，稼働中の並列計算機，MSBMのハードウェア構成を述べた．

さらに，Whetstoneベンチマークプログラムを利用し，バリア同期およびジョブの多重実行時GSM機構の性能評価を行った．その結果，ソフトウェアによるバリア同期では逆に速度が低下してしまうところ，ハードウェアバリア同期では最高2.02倍の速度向上がみられ，細粒度並列処理におけるMCAMによるハードウェアバリア同期機構の有効性が明らかになった．またジョブ同士を多重実行させた場合，細粒度ジョブ同士で最悪1.086倍の応答時間増加がみられ，細粒度ジョブの多重実行ではGSM機構をしてもある程度の効率低下が起こることがわかった．

今後の研究としては，ファジーバリア同期の有効性の確認，mBMに対するコードスケジューリング手法の開発およびバリアパターンを半動的に処理する方法の開発などがあげられる．

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頂いた㈱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] 高橋義造編: “並列処理機構”，丸善株式会社(1989).

[3] 笠原博徳,成田誠之助,橋本親: “OSCAR(Optimally 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 (Aug.1988).

[4] 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 (Jul.1991).

[5] M.T.O'Keefe and H.G.Dietz : “ Hardware Barrier Synchronization : Static Barrier MIMD (SBM)” ,1990 Int'l Conf.on Parallel Processing, vol.I, pp.35-42(1990).

[6] M.T.O'Keefe and H.G.Dietz : “ Hardware Barrier Synchronization : Dynamic Barrier MIMD (DBM)” ,1990 Int'l Conf.on Parallel Processing, vol.I, pp.43-46(1990).

[7] R.Gupta: “ The Fazzy Barrier:A Mechanism for High Speed Synchronization of Processors ” ,Proc.Third Int.Conf.on ASPLOS, pp.54-63(Apr.1986).

[8] 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 pp45-52 (Mar.1994).

[9] Taub,D.M.: “ Arbitration and Control Acquisition in the Proposed IEEE P896 Futurebus” IEEE Micro., pp.28-41(Aug.1984).

[10] CURNOW,H.J. WITCHMANN,B.A.: “A Synthetic benchmark” ,The Comuter J.19:1.

[11] 濱田智雄,野口善昭,前川孝徳,岩根雅彦: “細粒度コードスケジューリング手法の提案とその評価”， SWoPP琉球'94.