計算機アーキテクチャ 107－3
（１９９４．７．２１）

汎用マルチマイクロプロセッサFMMの構成と性能評価

野口善昭, 本石彰, 茶屋道宏貴, 岩根雅彦
九州工業大学工学部

少量のハードウェアで同期と通信のオーバヘッドを削減するために，包含検索機能をもった拡張CAMによる半動的ハードウェアバリア同期機構と書き込み選択放送機能をもった拡張グループ共有メモリ構成を提案し，これらの機構の評価のための実験機としてFMMを設計した．バリア同期機構からバリア識別情報を削除することによって生じるバリアの不一致を，拡張CAMへのバリアパターンの半動的な登録削除方法を確立することによって解決した．並列化されたWhetstoneベンチマークを使用してMSBM計算機でシミュレーションによる評価をしたところ，6PU以上で並列実行したときは逐次実行と比べて計算時間が0.46倍となった．

Architecture and Performance Evaluation of FMM: A General Purpose Multi-Microprocessors

Yoshiaki Noguchi, Akira Motoishi, Hirotaka Chayamichi, Masahiko Iwane
Faculty of Engineering, Kyushu Institute of Technology

In general purpose multi-microprocessor systems, decreasing the overhead of interprocessor communication and synchronization for running fine to coarse grain size parallel programs. FMM with the semi-dynamic barrier mechanism(SDB) and the extended group shared memory system(EGSM) to reduce the overhead is shown. The barrier pattern in the modified CAM can be updated dynamically at the beginning of running the procedure using the SDB. The EGSM consisting of the system shared memory, the group shared memory and the local memory has the ability of multicasting. The performance of FMM is evaluated using the paralleled Whetstone benchmark on MSBM.

1 はじめに

汎用マルチプロセッサでは細粒度から粗粒度までの並列性をもった多種多様な複数のプログラムが効率的に並列処理される必要がある．並列処理の効率を妨げる大きな要因はプロセッサ間の同期と通信によるオーバヘッド[1]である．プロセッサ間同期の方法の一つにバリア同期があり，その機構はソフトウェアまたはハードウェアで実現されている．細粒度の並列性をもったプログラムではソフトウェアのバリア同期はバリア共有変数の操作による遅れが大きいのでハードウェアのバリア同期機構[2, 3]が提案されている．ファジーバリア[4]はバリア同期識別情報とマスクによって，エラスティックバリア[5]は同期カウンタとマスクによって任意のプロセッサ間のバリア同期を可能にしているが，配線量，回路量ともに増大する．SBM[6]はバリア同期情報の代わりにバリア同期間で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の任意のプロセッサ間の同期をとる．DBM[7]はバリア同期間に半順序関係を導入してバリアキューとCAM (Content Addressable Memory)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．SBMおよびDBMでは解の収束の程度によってループの繰り返し回数が異なるときには同期の管理が困難である．

密結合マルチプロセッサにおけるプロセッサ間通信は共有メモリによって行われるのでプロセッサと共有メモリを接続する相互接続網や共有メモリモジュールにおいて競合がおこる[8, 9]集中型共有メモリにおける相互接続網としてハードウェア量の少ないバス結合が多く使用されている．プロセッサが十数台以上になるとキャッシュメモリによりバス競合を削減させているが，ハードウェア量が増加するだけでなくキャッシュコヒーレンス問題が生じる[10]．各プロセッサが局所メモリと共有メモリをもつ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時の競合を減少できるが共有メモリをプロセッサに分散させるためにアドレス変換機構が必要である[11, 12]．疎結合マルチプロセッサにおける通信は入出力操作と同様に行われるので直接通信路のないプロセッサ間ではオーバヘッドが生じるが，隣接したプロセッサへ一度に多量のデータ通信を行い，粗粒度の並列性をもつある種のプログラムに対しては格子結合網が有効である[13]．MSBM[14]がハードウェアバリア同期機構とグループ共有メモリ構成をもった細粒度並列計算機として開発されているが，効率よく静的コードスケジューリングを行うには制約が大きい[15]．

細粒度の並列性をもった多種多様な複数のプログラムを効率よく処理するために，拡張CAMを用いたハードウェアによる半動的バリア同期機構，プログラム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，格子結合網による隣接プロセッサ間通信機能をもったマルチマイクロプロセッサを設計した．

本論文では拡張CAMを用いたハードウェアによる半動的バリア同期機構，FMMのハードウェア構成および並列化 Whetstone ベンチマークによる性能予測について述べる．

2 半動的バリア同期機構

2.1 前提

共有データのための部分共有メモリとバリア同期機構をもった多数のマイクロプロセッサPUとホストコンピュータHCで構成されたマルチマイクロプロセッサシステムにおいて，マスタスレーブ型オペレーティングシステムのもとで複数の応用プログラムが多重実行される環境で次の前提を考える．

並列処理される一つの応用プログラムはその各々がプロセデュアと呼ばれる一つの主プログラムと幾つかのサブプログラムにより構成される．さらにプロセデュアはマイクロプロセデュアと呼

ばれる逐次あるいは並列に実行される命令列で構成される．ユーザがオペレーティングシステムのコマンドにより実行することができる一つの応用プログラムをジョブと呼び，プロセデュア，マイクロプロセデュアに対するそれぞれの処理の実体をプロセス，マイクロプロセスと呼ぶ．ジョブを構成するプロセデュアは個別にコンパイラなどによって静的スケジューリングがなされて，バリア同期が設定され，並列化されている．リンカーおよびローダによりこれらのプロセデュアはジョブとして結合されジョブ単位でスケジューリングを行い，ジョブを固定のマイクロプロセデュアの集合と考えてジョブ実行前にマイクロプロセデュアを要素プロセッサへ割り当てる静的割り当て[16]を行う．このとき並列に動作可能なマイクロプロセデュアは異なった要素プロセッサに割り当てる．図1において丸印はマイクロプロセデュア，点線はプロセデュア，一点破線はジョブである．そして実行中のジョブが終了したときPUを他のジョブに解放するものとする．なお一つのジョブに割り当てられた要素プロセッサの集合をグループと呼ぶ．

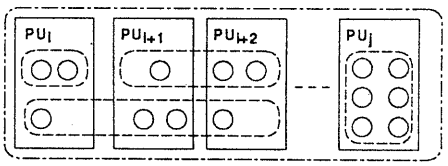

図1: プロセデュアのPU割り当て

2.2 拡張CAM[17]による半動的同期機構

通常のバリア同期のほかに，バリア同期識別情報の削除によって生じる不都合を補うダミーバリア (Dummy Barrier) 同期，および広がりのある領域内での同期を許すファジーバリア (Fuzzy Barrier) 同期を導入する．これらの同期を実現するためにDmBar (Dummy Barrier) 命令，AdBar (Advanced Barrier) 命令，Bar (Barrier) 命令をPU出力命令として用意する．PUはDmBar命令を実行するとバリア同期ユニットMCAMUにバリア同期要求を送り処理を続行する．PUはAdBar命令を実行するとMCAMUにバリア同期要求を送り処理を続行するが，このとき同期が成立しておればMCAMUはPUに同期成立を知らせる．PUがBar命令を実行したとき，すでにAdBar命令が実行されて同期成立が知らされていればPUは処理を続行する．まだ同期が成立していなければ同期成立まで処理を中断する．PUがBar命令に先だってAdBar命令を実行していなければバリア同期要求をMCAMUに送り処理を中断する．そして同期が成立すればMCAMUはPUに同期成立を知らせてPUの処理を再開させる．すなわちBar命令単独で通常のバリア同期，DmBar命令でダミーバリア同期，AdBar命令とBar命令のペアでファジーバリア同期を実現する．

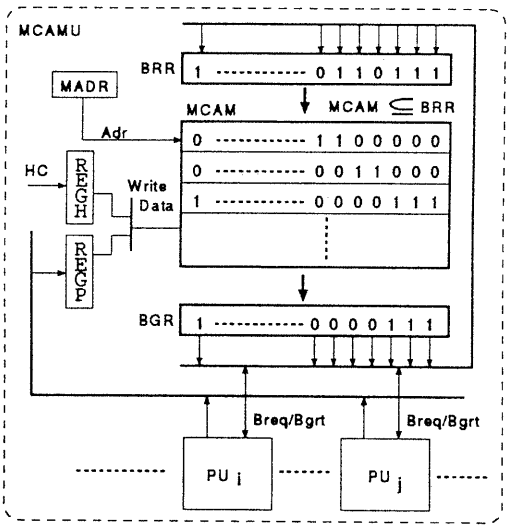

図2: MCAMUの構造

図2に示すようにバリア同期機構はMCAMUと個々のPU内のBarif (Barrier interface) で構成される．またMCAMUはシステム内PU台数に等しいビット長をもったBRR (Barrier Re-

quest Register),BGR (barrier Grant Register) およびMCAM(Modified Content Addressable Memory)で構成される．特にMCAMUには，エントリの有効，無効を示すvフラグが1ビット付加してある．PUがDmBar命令，AdBar命令，Bar命令を実行したときBarifは最初にDmフラグを検査して，セットしていればMCAMUからのBgrt (Barrier Grant)信号を受け取るまでWait信号によりPUの実行を中断させる．Dmフラグがセットしていないときまたは Bgrt信号の到着後，DmBar命令ではDmフラグをセットしてBreq (Barrier Request)信号をMCAMUに送りPUの処理を続行する．AdBar命令ではAdフラグをセットしてBreq信号をMCAMUに送りPUの処理を続行する．Bar命令ではAdフラグとGrフラグの両方がリセットしていればBreq信号をMCAMUに送ると共にWait信号によりBgrt信号が到着するまでPUの実行を中断する．単にGrフラグのみがリセットしていればBgrt信号が到着するまでPUの実行を中断する．GrフラグがセットしていればフラグをリセットしPUの処理を続行する．Barifから送付されたBreq信号はBRRの当該ビットをセットする．MCAMUはBRRをMCAMの探索データとして使用して包含関係が成立すればMCAMをBGRに読み出す．すなわち，

$$if\ MCAM_{ij} \subseteq BRR_j\ (j = 0,1,\cdots,n-1)$$

$$then\ BGR_j := MCAM_{ij}\ (j = 0,1,\cdots,n-1)$$

但し，iはMCAM内のi番目のエントリ，jはエントリi内のj番目のビット，nはシステムのPU総数を表す．また包含関係は次式で与えられる．

$$MCAM_{ij} \subseteq BRR_j = \neg MCAM_{ij} \vee BRR_j$$

BGRに読み出されたデータはBgrt信号としてBarifに同期の成立を伝える．同時にBGRとBRRの排他的論理和EXORをとってBGRに格納する．Bgrt信号を受け取ったBarifはDmフラグをリセットし，さらにAdフラグがセットしていればGrフラグのセットとAdフラグのリセットを行ったのちに，PUが処理を中断していれば再開する．なお，検索データに包含されるすべてのパターンの論理和が出力される．MCAMへの書き込みはHCからもPUからも可能である．

2.3 バリアパターンの管理

グループ内PU間でのバリア同期はBar命令，ダミーバリア同期はDmBar命令およびファジーバリア同期はAdBar命令とBar命令のペアで実現できるが，どのマイクロプロセスやサブプロセスと同期をとるかあらかじめMCAMに設定する必要がある．このようなバリア同期をとるPUの組を示すビット列をバリアパターンと呼ぶ．マイクロプロセス間またはジョブ間のバリアパターンのMCAMへの登録はジョブの実行開始時にHCが行うかあるいはPUが init_bar プリミティブを実行することによって行われる．MCAMからの削除はジョブの終了時にHCが行うかPUがreset_barプリミティブを実行することにより行われる．

init_bar(arg1,arg2,…,entry) : arg1,arg2,… で与えられたバリアパターンをMCAM上に登録し，そのときのエントリをentryに返す．そしてvフラグをセットする．

reset_bar(entry) : entryで示したMCAMのvフラグをリセットする．

図3においてノードはマイクロプロセス，矢印は実行順序，×はBar命令，△はDmBar命令を表す．マイクロプロセス7(3)で生産されたデータをマイクロプロセス11(12)で消費するときマイクロプロセス2(8)の直後のDmBar命令は本来不要であるがバリアパターンのMCAMへの登録削除回数を削減してオーバヘッドを少なくするために挿入した．マイクロプロセス9の直後のDmBar命令も本来不要である．PU0でinit_barが実行されるとバリアパターンがMCAMに登録される．バリア同期B1，B2，B3では

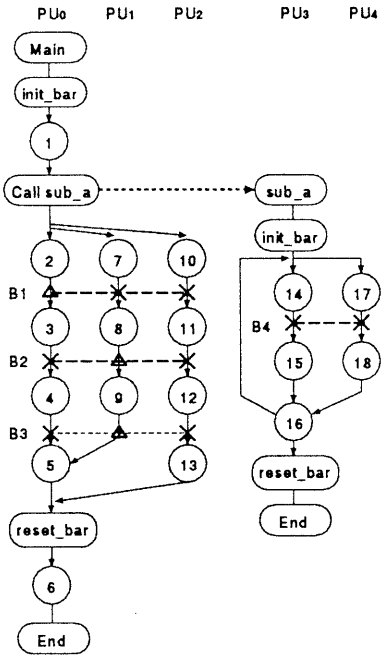

図 3: ジョブの実行順序

PU0, PU1, PU2 がバリア命令またはダミーバリア命令を実行するのでバリア同期が成立する．このバリアパターンは PU0 が reset_bar を実行したとき無効となる．sub_a はループ中にバリアを必要とするサブルーチンとする．PU3, PU4 に対するバリアパターンは PU3 の init_bar によって MCAM に登録され reset_bar によって抹消される．このように同じジョブでもバリア同期の設定は任意に独立に行うことができるので繰り返し回数が不定のループ中のバリア同期も問題なく処理できる．また，プロセデュア同志の静的スケジューリングはリンカーによって行われているので，通常メインルーチンはサブルーチンの終了を待たなくてよい．終了と待合わせる必要のあるときは粒度が大きいので共有変数による待合わせで十分である．

3 ハードウェア構成

3.1 システム構成

FMM は図 4で示すように HC (Host Computer), IIU (Integrated Interface Unit), および最大 64 台の PU (Processing Unit) から構成されている．

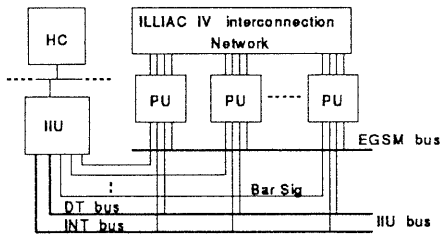

図 4: FMM ハードウェア構成

HC はパーソナルコンピュータ (MPU:i80386) を使用しユーザーインタフェースとプログラミング環境を提供するとともに，IIU，PU の制御を行う機能をもつ．

IIU は DTU (Data Transfer Unit), MCAMU (Modefied CAM Unit), INTU (Interrupt Unit) からなり，HC-PU 間のインタフェースを構成する．DTU は DMA による HC-PU 間及び隣接 PU 間のデータ転送機能を提供する．HC-PU 間の転送は HC バスと DT バスを使用し PU 単位，クループ単位またはすべての PU に転送が可能である．隣接 PU 間は ILLIAC 網を使用し各 PU は 4 方向の隣接 PU とのデータ転送が可能である．MCAMU はバリア同期機構を使用するためのバリアプロセッサであり，任意の PU の集合で同期を取ることができる．MCAMU へのバリアパターンの登録，削除は HC からは HC バスを，PU からは INT バスを介して行われる．INTU は INT バスを介した PU からのサービス要求をその種類に応じて処理する．図 5 にパラメータデータのフォーマットを示す．31 ビット目が 0 のときは割り込み要求データで 1 のときはバリア

パターン更新要求データである．Mod(Modifier) はIIU での処理種別を指定する．Mod がバリアパターン更新要求であれば，MCAM の Adr と Pos で指定した位置に Data のバリアパターンを書き込む．

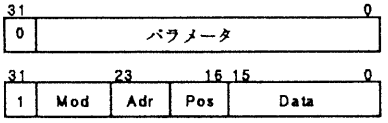

図 5: INT bus データ交換フォーマット

PU は 8088MPU+8087FPU，256KB メモリ，IIU インタフェース，EGSM インタフェース，NEWS インタフェースから構成される．そして，DMA 転送やデータのグループ共有のために PU 番号とグループ番号の識別機能を PU 内部にもつ．PU メモリは拡張グループ共有メモリ EGSM 構成をとり，LM (Local Memory)，GSM (Group Shared Memory)，SSM (System Shared Memory)，からなる．GSM にはグループで共有するデータを格納することができる．

3.2 拡張グループ共有メモリ EGSM

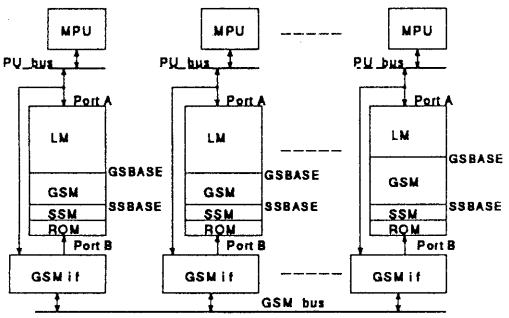

図 6: GSM 構成

拡張グループ共有メモリ EGSM (Extended Group Shared Memory) 構成を図6に示す．EGSM 構成は各 PU のメモリの内容をグループごとに完全一致させるグループ共有メモリ GSM および，システム内のすべての PU でメモリの内容を一致させるシステム共有メモリ SSM を実現する．特に SSM にはバリア管理のようなシステムに共通な情報を置く．共有データの参照時は常に自 PU 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 PU では共有メモリ領域を LM と同様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/SSM 内容のグループ内一致は，あるプロセッサが GSM/SSM に書き込みを行ったとき GSM バスに所属グループの番号と SSM アクセスを表す信号を出力させ，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/SSM 領域への書き込みか LM 領域の書き込みかの判断は各 PU 内のレジスタ GSBASE/SSBASE 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 DPM (Dual Port Memory) を用い，GSBASE レジスタにより LM と GSM，SSBASE レジスタにより GSM と SSM 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みが，プロセッサ内部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．

GSM/SSM の使用の際に必要な相互排除も GSM バスの機能として提供されており，GSM 転送用のグループロックと SSM 転送用のシステムロックの機能がある．

4 MSBM による性能予測

4.1 基本性能と並列化ベンチマーク

FMM の基本性能として，PU からのバリアパターンの更新，データ転送およびバリア同期の処理能力を設計データから算出した．バリアパターンの更新に 10μs を要した．HC と PU 間のデータ転送能力および ILLIAC 網による PU と PU 間のデータ転送能力は 500KB/s，EGSM バスによる PU と PU 間のデータ転送能力は 700KB/s

であった．バリア同期の処理時間として同期成立からPUの動作が再開するまでの時間とすると1.5μsであった．

細粒度静的スケジューリング法[15]によりWhetstoneベンチマークのモジュールを各々単独に並列化を行ってMSBMで実行したところ，各モジュールの並列度およびモジュール2を1とした相対実行時間は次のとおりとなった．

```
module2: 並列度3, 実行時間1,    逐次時間1.1
module3: 並列度4, 実行時間6.6,  逐次時間8.9
module4: 並列度1, 実行時間3.3,  逐次時間3.3
module6: 並列度5, 実行時間9.2,  逐次時間18.5
module7: 並列度4, 実行時間16.5, 逐次時間37.8
module8: 並列度2, 実行時間54.2, 逐次時間60.2
module9: 並列度3, 実行時間20.4, 逐次時間24.0
module11: 並列度3, 実行時間16.7, 逐次時間17.3
```

Whetstoneベンチマークの実行にあたり使用PU台数をあらかじめ与え，その台数以内で並列実行できるモジュールは並列実行するようにモジュールレベルのスケジューリングを行った．また並列化されたモジュールの実行開始直後にバリアパターンをMCAMに登録し，終了直前にそのバリアパターンを無効とした．

4.2 性能評価

図7に使用可能PU台数以内でモジュールの実行順序を入れ換えることなしにデータ依存性を考慮して，できる限りモジュールを並列実行させたときの相対実行時間の最良値を示す．縦軸はすべてのモジュールを逐次実行したときの実行時間を1としたときの相対実行時間，横軸は使用可能PU台数である．最大性能は6PUを使用して，次のスケジューリングを行ったときに得られた．このときの相対実行時間は0.46であった．

```
mod2-mod3-mod6-mod8
        mod4-mod7-mod9-mod11
```

7PU以上でこのプログラムを実行しても6PUでの実行以上の性能向上はできないのはモジュール8の並列実行時間がほかのモジュールの2倍以上であることに起因する．

MSBMのsingle-barrier MIMDモードで実行させたとき，5PUの使用で相対実行時間は0.75となった．一方，FMMでは5PUの使用で相対実行時間は0.6となった．これはモジュール8と9が並列実行でき，モジュール9の終了後に11を実行することによって短縮できた．

このプログラムではバリアパターンの登録，削除に要する実行時間に比べてモジュールの実行時間が非常に大きいので並列実行の効果が十分に得られた．これによりもう少し短い時間間隔でバリアパターンの登録，削除を行ってもかなりの性能向上が見込めると推測できる．

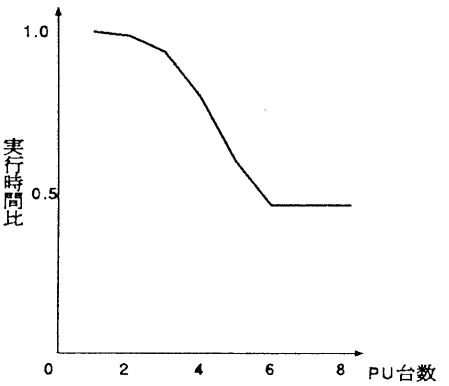

図7: Whetstoneの実行時間

5 むすび

細粒度の並列処理を効率よく行うための同期機構としてハードウェアによる半動的バリア同期機構とこの機構を効率よく使用するための拡張グループ共有メモリ構成を示した．そして，これらの機能を評価するためのテストベッドとして開発中の汎用並列計算機FMMのハードウェア構成と基本性能について述べた．Whetstoneベンチマークを用いてFMMの性能予測を行ったところ逐次実行に比べて相対実行時間は0.46，MSBMに比べ

て相対実行時間は0.61(=0.46/0.75)となり有効性が確認できた.

今後の課題として, FMMを製作すると共に半動的バリア同期機構に適した静的スケジューリングアルゴリズムを開発することである.

参考文献

1) 富田, 末吉: 並列処理マシン, オーム社(1989).

2) S.F.Lundstrom: Applications Considerations in the System Design of Highly Concurrent Multiprocessors, IEEE trans. on Computer, Vol.C-36, No.11, pp1292-1309(1987).

3) H.dietz et al: Static Synchronization Beyond VLIW, Supercomputing 89, pp416-425(1989).

4) R.Gupta: The Fuzzy Barrier: A Mechanism for High Speed Syncronization of Processors, 3rd INt. Conf. on ASPLOS, pp.54-63(1989).

5) 松本: Elastic Barrier: 一般化されたバリア型同期機構, 情処論, Vol.32, No.7.pp886-896(1991).

6) M.T.O'Keefe and H.G.Dietz: Hardware Barrier Synchronization: Static Barrier MIMD (SBM), 1990 Int. Conf. on Parallel Processing, pp.I5-I42(1990).

7) M.T.O'Keefe and H.G.Dietz: Hardware Barrier Synchronization: Dynamic Barrier MIMD (DBM), 1990 Int. Conf. on Parallel Processing, pp.I43-I46(1990).

8) D.J.Kuck, E.S.Davdson and D.H.Lawrie: Parallel Supercomputing Today and the Cedar Approach, 信学論, Vol.J71-D, No.8, pp.1361-1374(1988).

9) L.n.Bhuyan, Q.Yang and D.P.Agrawal: Performance of Multiprocessor Interconnection Networks, Computer, pp25-37, Feb.(1989).

10) M.Dubois and C.Scheurich: Synchronization, Coherence and Event Ordering in Multiprocessors, Computer, pp9-21, Feb.(1988).

11) 松田 他: マルチプロセッサシステムPARK上での並列Prolog処理系の実現, 情処論, Vol.29, No.1, pp38-46(1988).

12) 高橋義造編: 並列処理機構, 丸善(1989).

13) T.Hoshino: Ainvitation to the world of PAX, Computer, pp68-79, May(1986).

14) 本石彰, 濱田智雄, 米沢敏夫, 岩根雅彦: 細粒度マルチマイクロプロセッサMSBMの構成と性能評価, SWoPP琉球'94, (1994).

15) 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, (1994).

16) 渡辺: 並列処理概説, コロナ社(1991).

17) 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).