並列推論マシンPIM/pの要素プロセッサのアーキテクチャ

篠木 剛*, 松本 明**, 近山 隆**, 後藤 厚宏**, 服部 彰*
(*富士通株式会社, **新世代コンピュータ技術開発機構)

1. はじめに

我々は, 第5世代コンピュータプロジェクトの一環として並列推論マシンPIM/pの開発を行っている. 本論文では, PIM/pの要素プロセッサ(CPU)のアーキテクチャの設計過程での検討内容の一部とアーキテクチャの概要について報告する.

PIM/pは図1のように, 100台規模のCPUを, 共有バス結合とクラスタ間ネットワーク結合の二階層の構造でつなぐ.

2. 要素プロセッサの設計のポイント

要素プロセッサの設計では, 特に次の点に重点をおいた.

(1)記号処理用アーキテクチャ

-データタイプ処理とガーベジコレクションの高速処理

特に, 実時間ガーベジコレクタ-MRB方式[2]-の高速化のために, MRB処理のオーバヘッドを小さくすること

(2)マルチプロセッサ用のアーキテクチャ

-共有バスの通信量の低減

クラスタ内は8台のCPUと1つの共有メモリが共有バスで密結合されており, このバス負荷を低減すること.

本論文ではこれを中心として報告する.

-プロセッサ間の通信自体を高速化すること

クラスタ内のCPU間通信及びクラスタ間の通信のオーバヘッドを小さくすること

3. 共有バス負荷の低減

多数のCPUが共有バスをとおして共有メモリをアクセスする並列計算機では, CPUの処理が高速になるにつれ, 共有バス上を流れる情報のトラフィックによるバスネックが大きな問題となる. そこで, 共有バス上を流れる情報量を小さく抑えることにCPUの設計の重点を置いた.

(1)マクロ命令

マシン命令を設計するにあたってまず問題となったのが, マシン命令のレベルを高く設定し, KL1-B言語(KL1コンパイラがだすWAMレベルの命令列)をマイクロエミュレートする方式にするか, RISC的に低く設定し, KL1-B言語をさらに展開して, 直接ハードワイヤードに実行する方式にするかということであった. 我々は, 以下の理由で, 後者を選んだ.

-マシン命令レベルを高く設定するとコードのoptimizationの余地が狭められてしまうこと

-マイクロエミュレート方式の場合, VLSI化1チッププロセッサでは, マイクロプログラム用メモリをチップ外のメモリにたよらざるおえないため, そのチップクロスが原因でマイクロへのdispatch時間が相対的に大きくなること

しかし, 命令レベルを低く設定するとコンパイルドコードサイズが大きくなるため, 命令フェッチによる共有バス負荷が問題になる. そこで, 高機能な命令を定義するための機能(マクロ命令)を導入することにした. これは, CPU側にサブルーチン群を格納する局所メモリをもち, これらをコンパイルドコードから高速に呼び出す機構である. 局所メモリに格納される命令は, コンパイルドコード用命令とほぼ同じである. そのため, 複雑なマイクロ命令とちがって, システムプログラマによって簡単に作成することができる.

(2)ライトバックキャッシュとキャッシュ制御命令

KL1言語の単一代入的性格から, 従来言語に比べてメモリライトが多いという性質があるため, メモリライト命令毎に共有バスアクセスが起こるstore through方式は適さない. そのため, write back方式のキャッシュ(Illinois方式)を採用することにした.

従来の計算機のメモリアクセス命令では, read命令とwrite命令しかもっていない. これらの命令はある意味で汎用すぎるため, 無駄なデータが共有バスを流れることがある. そこで無駄なデータが流れることをプログラムから抑制できるように, キャッシュの動作を制御しつつメモリアクセスする命令を追加した.

(i) Direct write命令

通常のメモリライト命令では, キャッシュミスヒット時, データの書込に先立ち, キャッシュブロックを共有バスをとおしてフェッチするが, このフェッチが不要であることがプログラムではわかる場合がある. 初めて使うメモリ領域への書込がこれにあたる. 一般に, KL1言語を含む記号処理言語では, Consのような, ヒープの未使用領域の先頭にデータを書き込むという処理は多い. Consの他に, KL1言語では, ゴール(軽いプロセスのようなもの)の発生がこれと同様で, 発生したゴールの情報を新しい領域に書き込むという処理になる. ゴール単位で並列処理を行なうKL1ではこれが頻繁におこる. そこで, キャッシュミスヒットしても, 旧データをフェッ

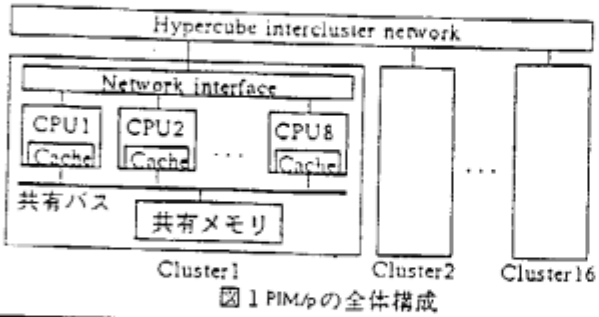

図1 PIM/pの全体構成

The Architecture of a Processor Element of PIM/p
Tsuyoshi Shinogi*, Akira Matsumoto**,
Takashi Chikayama**, Atsuhiro Goto**, Akira Hattori*
*Fujitsu Limited, **ICOT

チすることなく，キャッシュエントリを確保するだけで，キャッシュにデータを書き込むメモリライト命令(direct write命令)を用意することにした．

direct-write <reg>, <offset>(<addressreg>)

(ii) Read invalidate命令

あるCPU(A)が，readしようとしてキャッシュミスヒットのため他のCPU(B)がもっているデータをフェッチし，引き続いてこのデータ値を書き換える(write)ような場合で，このCPU(A)以外がreadとwrite間にそのデータをアクセスする確率が低い場合がある．例えば，CPU(A)が他のCPU(B)から新しいゴールをもらう場合やCPU(A)が未束縛変数に値をbindする場合などがそれに当たり，これらはKL1では頻繁におこる．このような場合にread命令とwrite命令しかない計算機では，CPU(A)のread時にfetchコマンドがバス上を流れ，write時にそのデータをくれたCPU(B)がそのデータをもっているためinvalidationがバス上を流れる．このinvalidationコマンドが共有バスに大きな負荷をかけていることがわかった．このCPU(A)以外がこのreadとwrite間にそのデータをアクセスする確率が低い場合には，read時にそのデータをもっているCPUのキャッシュブロックをついでにinvalidateするようなバスコマンド(read&invalidate)を出すようにすることによって，write時のinvalidationコマンドを出さなくすることができる．このようなread&invalidateコマンドをだすマシン命令がread invalidate命令である．

read-invalidate <reg>, <offset>(<addressreg>)

4. シミュレーションによる評価

表1は，3節で述べた各項目の，2つのプログラムのシミュレーションによる共有バスサイクル数に関する評価データである．

- 標準　8台CPU，DirectWrite有，ReadInvalidate無，KL1-Bコードを実行
- コードサイズ2倍　KL1-Bコードを2倍にし，他は標準と同じ
- コードサイズ4倍　KL1-Bコードを4倍にし，他は標準と同じ
- DirectWriteなし　DirectWriteを普通のWriteにし，他は標準と同じ
- RI　heap領域へのReadをすべてReadInvalidateにかえて実行，他は標準と同じ

コードサイズが2倍，4倍になると各々平均で10%,70%以上，また，DirectWriteがないと平均で30%以上，バスサイクル数が増えている．これらは，マクロ命令の効果，DirectWrite命令の効果が大きいことを示している．RIのデータでは，ReadInvalidateの使用により5%程度しかバスサイクル数が低下してないが，これはheap領域へのすべてのReadをReadInvalidateにかえたためであり，ReadInvalidateの選択的な使用によりさらにバスサイクル数が低下することが期待される．

5. 要素プロセッサの概要

要素プロセッサ1台の構成と仕様の概略を，図2と表2に示す．図2において，局所メモリにマクロの本体が格納される．KL1データ処理のためCPU内部のデータ系は，データタイプ用タグ8bitとデータ用32bitの計40bitである．タグの1ビットはアーキテクチャ的にMRB用bitに専用化されている．命令は，特にパイプラインブレークが発生しない限り，表3にあるような4段のパイプラインに，毎サイクル投入され，実行される．パイプライン段数をあまり長くしてプロセッサを複雑にしないために，メモリアクセス命令ではレジスタとの転送に限った．メモリアドレシングは，バイトアドレシングとした．マシンサイクルは50ns程度の予定である．

6. おわりに

1台の要素プロセッサの性能は，GCのオーバヘッド込みで，append RPSで600KRPS程度の予定である．現在，アーキテクチャ設計がおわり，VLSI設計，ボード設計等が進行中である．

参考文献

[1] P.Bitar 他：Multiprocessor cache synchronization. Proc. of the 13th Annual Int. Symp. on Computer Architecture. 1986.
[2]T.Chikayama 他：Multiple Reference Management in Flat GHC. Proc. of the 4th Int. Conf. on Logic Programming. 1987.
[3]後藤他：並列推論マシンPIM/pの概要．本大会予稿集
[4]久門他：並列推論マシンPIM/pのネットワーク．本大会予稿集

表1 シミュレーションによる共有バスサイクル数

	標準	code size 2倍	code size 4倍	Direct write無し	RI
BUP	512,955	643,462	1,212,070	654,279	507,318
・/標準	100%	126%	238%	128%	99%
NQueen	337,195	343,962	354,285	461,140	305,726
・/標準	100%	102%	106%	137%	93%
平均	100%	114%	172%	132%	95%

表2 要素プロセッサの概略仕様

命令実行方式	命令実行段数4段の 1サイクルパイプライン方式
レジスタ	40bit＊32W
局所メモリ	50bit＊8KW
キャッシュサイズ	命令用，データ用各64KB
キャッシュ方式	256column 4set 32B/block 2sector方式のライトバック方式
共有メモリサイズ	256MB

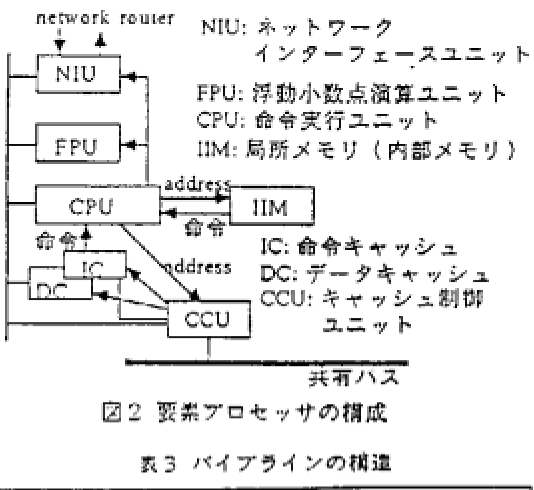

図2 要素プロセッサの構成

表3 パイプラインの構造

	ALU命令	メモリアクセス命令
D	Decode	Decode/Register read
A	・	Address 計算
T	Register read	Cache access
B	ALU/Register write	Cache access/Register access