情報処理学会第38回(昭和64年前期)全国大会

共有バス共有メモリ型マルチプロセッサ 支援機構の提案

7T-5

福田 宗弘　　松本 尚

日本アイ・ビー・エム株式会社　　東京基礎研究所

1. はじめに

スヌープ・キャッシュを装備した共有バス共有メモリ型マルチプロセッサ・システムでは，キャッシュ間の共有データの取扱い方法，およびプロセッサ間の同期の実現方法が共有バスの使用頻度を決定し，システムの性能に大きな影響を及ぼす。この共有データの取扱い方法をキャッシュ・プロトコルと呼ぶ。処理効率の向上を図るためには，キャッシュ・プロトコルをデータに応じて細かく制御すること，同期頻度の高い並列処理プログラムに対して高速な同期を実現すること，等が必要である。本稿では，共有データに応じたキャッシュ・プロトコルの変更を従来のシステムに搭載されているページ管理機構を利用して簡潔に行う機構を提案する。さらに，共有バスにプロセッサ台数分の同期信号線を付加し，各プロセッサに同期コントローラを装備することにより同期の高速化を図る方法について述べる。

2. キャッシュ・プトコル制御機構

2.1 ねらいと特徴

共有バスの使用頻度を軽減し，システム性能を改善することを目的として，システムの稼働中にキャッシュ・プロトコルを変更することが試みられている(1)。より良好な性能を得るために，種々の型の変数を割り付けたメモリ領域ごとにキャッシュ・プロトコルを変更する必要がある。以下に例を示す。

1. セマフォの鍵，プロセッサ間のデータの受渡しに変数を使用する場合: キャッシュへの書込み時に他のキャッシュの共有データを更新するupdate方式により，キャッシュのヒット率を高めることができる。

2. 局所変数をプロセッサが使用する場合: キャッシュへの書込み時に他のキャッシュの共有データを無効にする invalidate 方式により，バス・アクセスを軽減できる。

3. 全プロセッサが同一データを参照する必要がある場合: キャッシュ・ミスによる読込み時に，他のキャッシュがデータのリプレイスを引き起さずに，同一データを取り込むことができるとき，すべてのキャッシュが同一データを取り込むall read方式を採用する。データ数×プロセッサ数のバス・トラフィックをデータ数に抑えることができる。

上述の細かなキャッシュ・プロトコルの変更を実現するために，メモリ領域ごとにプロトコルのタイプを表す情報を付加し制御する必要がある。アドレスの各々にその情報を付加する方法では，その情報の記憶領域，管理に必要なハードウェア量は，増大し現実的ではない。そこで，従来のプロセッサ，または専用のメモリ管理ユニット (MMU) に装備されているページ管理機構を利用し，ページごとにキャッシュ・プロトコルの制御を行う機構を提案する。

提案方式では，ページごとにプロトコルのタイプを表す属性を付加する。変数や作業領域をそれに適したプロトコル・タイプを持つページ内に割り付ける。共有バスにプロトコル・タイプを示す信号線を付加し，各キャッシュ・コントローラがこの信号を監視することにより，ページ単位のプロトコル制御を行う。all read等の際にバス信号を使用してデータを受け取るキャッシュの指定を可能とする。

2.2実現方法

提案機構の基本構成を図1 に示す。各プロセッサにキャッシュに加えてグループIDコントローラを装備するとともに，キャッシュ・プロトコルのタイプを示す信号線を共有バスに付加する。ページ管理機構が使用するページ・エントリ内にプロトコルのタイプを示すフィールドを数ビット定義する。プロセッサは，メモリを参照する際にこれら数ビットを外部に出力することにより，このアクセスを処理するプロトコルの型を指定する。従来型のプロセッサ，またはMMUを用いて提案機構を実現する場合には，アドレスの上位2〜3ビットをプロトコル・タイプを示すフィールドとして使用する。32ビットの実アドレス線を持つプロセッサ，およびMMUでは，512Mbyteから1Gbyteの実メモリ空間が使用可能であり，実用上大きな影響を与えない。

キャッシュから共有バス上にアクセスが出るとき，このプロトコルのタイプを示す信号が送出される。他のキャッシュ・コントローラはこの信号によりプロトコルを決定し，共有データの処理を行う。また，グループIDコントローラは，all read時にキャッシュにデータを取り込むべきプロセッサ群を指定可能とする。これにより，OSの環境下で複数の無関係なタスクを同時に効率良く実行することができる。

3.高速同期機構

3.1ねらいと特徴

並列処理の1 つのアプローチとして，順次プログラムの並列化がある。たとえば，do文等のループを同時に複数のプロセッサを用いて並列に処理するループ展開(2)(3)，プログラムを1 命令程度に細分し，同時に複数のプロセッサにより実行するVLIW型コンパイラ(4)等がある。上述の応用例では，データの依存関係および実行順序を維持するために多数の同期を必要とする。多数の同期をメモリ参照によって実現した場合，同期に伴う処理は順次化され並列処理の効果を充分に発揮することができない。

そこで，同期信号のバスの付加と各プロセッサへの同期コントローラの装備によって同期の高速化を試みる。プロセッサは，同期コントローラに対して同期要求の発行，および返答の受信を行う。同期コントローラは，同期信号バスを用いて他のコントローラと通信し合い，同期の制御を行う。

提案方式では，2 種類の同期を提供する。1 つは，すべてのプロセッサが同期要求を発行するごとに，一斉に待ち合わせを行うものである。この同期をbarrier同期と呼ぶことにする。もう1 つは，あるプロセッサから論理的に隣接するプロセッサへの待ち合わせを次々に実現させるものである。この同期をpipeline同期と呼ぶことにする。この2 種類の同期を並列に処理するプログラムに応じて適用する。たとえば，VLIWの実現では，barrier同期を用いて並列処理可能な命令の実行を一斉に進める。また，データの依存関係を持つループ展開 (do across) では，繰り返しの各々が，依存箇所の入口点で毎回 barrier同期を行うことにより依存箇所を順に実行できる。複雑な依存関係を持つ do across等のpipeline処理では，コンパイル時に各繰り返しのステップ数等を詳細に計算する必要がある。barrier同期を利用したコンパイルが不可能な場合，実行時にpipeline処理の並列度を変化させる場合には，pipeline同期が有効である。

pipeline同期を用いたdo acrossの実行例を図2.1に示す。特に，あるプロセッサでの同期受付け順序を維持する必要がある。たとえば，

A Proposal of Support Mechanizm for Shared Bus Shared Memory Multiprocessors
Munehiro FUKUDA, Takashi MATSUMOTO
IBM Research, Tokyo Research Laboratory,

プロセッサiからプロセッサi+2への同期sync3は，プロセッサiにおける3回目の同期要求であるとともに，プロセッサi+2における3回目の同期受付けであることがわかる。そこで，同期要求を行うプロセッサは，同期先プロセッサでの同期受付け回数を数えることにより，適切に同期要求を発行することができる。以上の機構をハードウェアで実現することにより，同期のオーバヘッドを削減する。

また，プロセッサ資源の有効利用を目的として，プロセッサのグループ分けを行ない，同一グループ内でのプロセッサの同期を実現する。このとき，barrier同期において，一部のプロセッサは同期をとる必要のない場合が発生する。頻繁に同期をとるプログラムでは，同期の必要のないプロセッサにダミーの同期要求命令を付加する。長期間にわたって他のプロセッサと同期をとる必要がない場合には，プロセッサのグループ分けを動的に変更する。

3.2実現方法

同期コントローラは，図2.2に示すようにbarrier同期ユニットとpipeline同期ユニットからなる。プロセッサは，アプリケーションに応じて2つのユニットの一方を選択する。システム上のすべての同期コントローラは，同一のクロックにより駆動される。同期信号バスは，プロセッサ台数分の信号線を持ち，各線はプロセッサの各々に割り当てられる。

barrier同期ユニットは，syncレジスタ，コンパレータ，タイミング制御回路からなる。プロセッサは，同期をとるプロセッサに対応するsyncレジスタのビットをオンにする。タイミング制御回路は，プロセッサからの同期要求を受け付けて，同期信号バスの自分に割り当てられた信号線をオンにする。コンパレータは，同期信号バスをサンプリングし，syncレジスタの値と比較し，同期の成立を検出する。

pipelin同期ユニットは，syncレジスタ，syncカウンタ，コンパレータ，カウンタ・アレイ，およびsync受付け回路からなる。プロセッサは，syncレジスタに論理的に隣接するプロセッサIDをセットし，同期要求を発行する。syncカウンタは，その同期要求の回数を保持する。カウンタ・アレイを構成する各カウンタは，プロセッサの各々に割り当てられ，同期信号バス上に送出された信号を検出し，同期受付け回数を保持する。コンパレータは，syncカウンタと同期先プロセッサの同期受付け回数を比較し，値が一致したときに，同期信号を送出する。sync受付け回路は他のプロセッサからの同期要求を受け付ける。

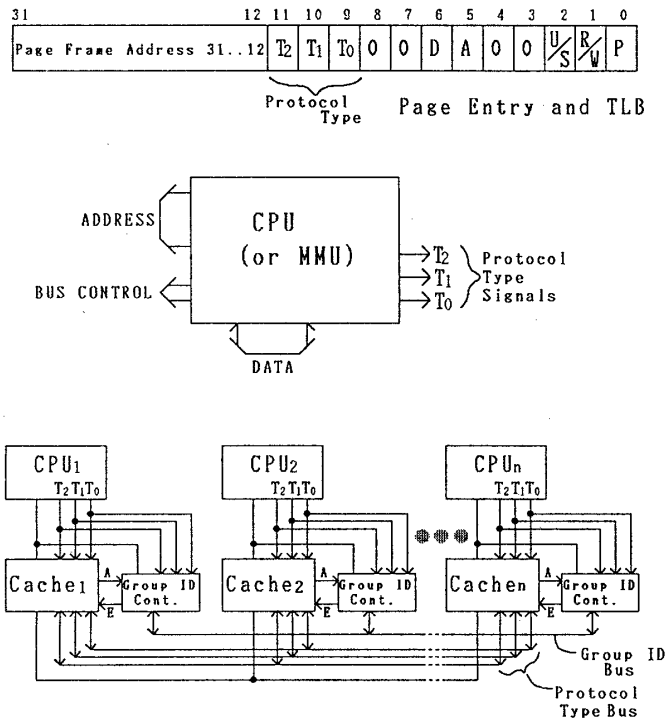

図1　キャッシュ・プロトコル制御機構の構成

4.むすび

スヌープ・キャッシュを備えた共有バス共有メモリ型マルチプロセッサのシステム性能を向上することを目的として，キャッシュ・プロトコル制御機構，および高速同期機構を提案した。今後の課題は，提案機構の実現，およびこの機構を利用したループ展開等のプログラムの実行により，その有効性を確認することである。

尚，貴重な助言を頂いた本研究所中田武男氏，鈴木所長に感謝します。

文献

1. 大庭,清水: 高速並列処理ワークステーション(TOP-1)スヌープ・キャッシュ. 情処第37回全国大会 (1988) pp. 176-177.
2. M.Wolfe: Multiprocessor Synchronization for Concurrent Loops. *IEEE Software* (Jan.1988) pp. 34-42.
3. S.P.Midkiff, D.A.Padua: Compiler Generated Synchronization for Do Loops. *Proc.Int'l Conf.Parallel Processing, CS Press* (1986) pp. 544-551.
4. J.R.Ellis: Bulldog:A Compiler for VLIW Architectures. *The MIT Press* (1985).

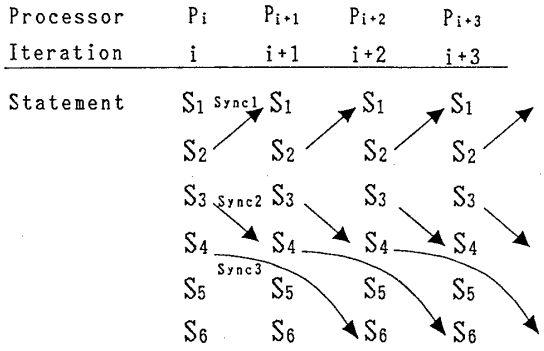

図2・1　Do Acrossの実行例

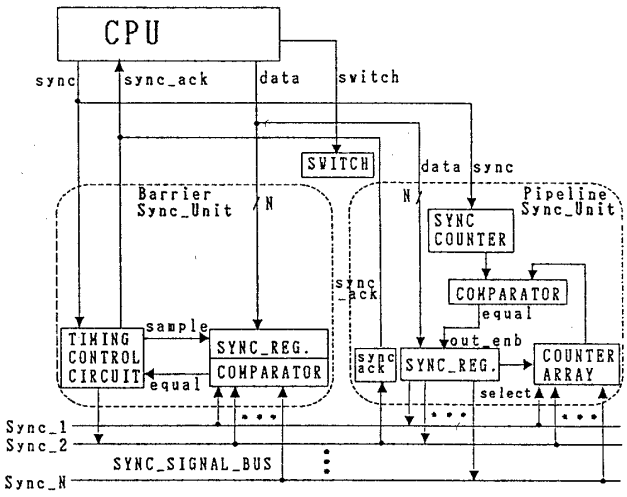

図2・2　同期コントローラの構成