計算機アーキテクチャ　63－16
(1986・11・21)

OS機能をPEに分散したマルチプロセッサについて

A multi processor with distributed OS functions on each Processor Element.

島田明宏、舟渡信彦、坂上誠司、千葉徹、河田亨

Akihiro Shimada、Nobuhiko Funato、Seiji Sakaue、Toru Chiba、Toru Kawata

シャープ株式会社、技術本部、コンピュータシステム研究所

Computer Systems Laboratories, SHARP Corp.

1. まえがき

EDPやCAD/CAE等の幅広い情報処理の分野ではコンピュータ技術やネットワーク技術への様々の要求がある。OAを背景とした高度のユーザーインターフェースを備えた機器やDAを背景とした設計･解析技術には、多量の処理や演算を短時間で実行することに対して強い要求がある。これに対応して処理時間を短縮するために、マルチコンピュータや専用ハードウェアによる多数のアプローチの報告がある。本研究は、汎用のハードウェアをベースにして高い処理能力を持ったマルチプロセッサを安価に実現することを目標に開発を行ったものである。本システムは、汎用コンピュータとしての機能を備えたものであると同時に、容易にマルチCPU構成による専用エンジン的な付加装置にもなるものである。本システムでは複数のCPUをバスで結合しているが、ネットワークでプロセッサエ･エメント(PE)を接続したシステム構成も実現できることを目標としている。

2. ハードウェア構成

2.1 システムアーキテクチャ

本マルチプロセッサシステムのハードウェア概念アーキテクチャを図2-1に示す。各プロセッサエレメント(PE)はその内部にローカルバスを持ち、そのバスにはローカルI/O、ローカルメモリが結合されている。即ち、通常のCPUオペレーションは、PEの内部アクセスにおいて完結する構造となっている。

システムバスには、グローバルメモリ、グローバルI/O等のデバイスが結合されており、PEは必要に応じ自己のバスインタフェースを呼ぶことにより、システムバスに結合するデバイスをアクセスすることができる。また各PEのローカルメモリもバスインターフェース(BUS I/F)を通して結合されているため、各PEはグローバルメモリ、グローバルI/O等の共有資源はもとより、他PEのローカルメモリを直接アクセスすることができ、PE間の通信がバスの速度で行うことができる。本マルチプロセッサシステムでは、このシステムバスをCPU間同期のための制御情報の転送、PEとグローバルI/O間のデータ転送および共有メモリへのアクセスに用いる。

CPUとしては、MC68020(16MHz)を用いており、このCPUを搭載したPEを、システムバスに最大14個まで接続できる。従って、このCPU能力を最大限利用できたとすると30MIPSを超える処理能力を得ることができる。またシステムバスとしては、非同期プロトコルの32bitバスであり、MC68020と適合性がよいVME規格のバスを採用している。これに加えて、ボード間のデータ転送ルートとしてIPTバス(Inter Processor Transfer Bus)を持っており、あるPEのローカルメモリから別のPEのローカルメモリへの高速なDMA転送を行うことができる。

本マルチプロセッサシステムのメモリマップは図2-2に示す通りである。すべてのCPUのアドレス空間は同一空間上に配置し、各PE上のCPUは他PEのローカルメモリにもアクセスのできる構造になっている。即ち、MC68020のアドレス空間4Gbyteを16分割し256Mbyteのブロックを16個作って、最下位のブロック(ブロック#0)を自己のワークエリアとしてローカルバス経由でアクセスし、最下位以外のブロックはシステムバス経由でアクセスする。その際、ブロック#1からブロック#14にはCPU#1からCPU#14のワークエリアを割当て、最上位のブロック#15にはグローバルメモリとグローバルI/Oを割当てる。このようなメモリマップの構造により、物理的にはメモリをローカルメモリとして各PEに分散させることによってアクセス競合を抑えながら、論理的にはメモリを共有することによって柔軟性のあるPE間通信を実現している。

2.2 二重バス構造

本マルチプロセッサシステムは概念的にはネットワーク型のシステムであり、前節で述べたようにメモリをローカルメモリとして各PEに物理的に分散させ、かつOSも次章で述べるように各PEに搭載しており、各PEがすべて対等なマルチマスタ方式のシステムとなっている。このようなマルチプロセッサシス

テムは共有メモリ型のマルチプロセッサシステムに比べ、

(1) CPUアクセスをできるだけPE内部で完結させることによりアクセス競合を抑えることができる

(2) 各PEが機能的に独立しており、一個のPEの故障が全体に波及することがないため、システム全体としては高稼動率を達成することができる

(3) 処理内容に応じて、いくつかのPEを専用サブシステムとして設計することにより、性能/価格比の向上を図ることができる

といった特長を有している。しかし、1)複数の協調型プロセスからなる一つのジョブの処理を高速化するため各PEに負荷を分散させる場合や、2)プロセスのPEへの割当てを動的に行い各PEの負荷を均等化することによりシステムのスループットを向上させたい場合、には不利な方式である。いずれの場合にも、PE間の通信のためにシステムバスへのアクセスが生じ、通信量によってはアクセス競合のためシステムの性能が低下する。前者の場合には、PE間通信量はジョブの性質によって異なるが、後者の場合にはプロセスの移動が必要となるため、通信量は通常かなり大きなものとなる。そこで、本マルチプロセッサシステムではこれらの場合に対応するためのハードウェア基盤として、システムバス以外に、PE間で大量データを高速に転送するための専用バスを備えた二重バス構造のシステムとなっている。本システムではこの専用バスをIPTバス(Inter Processor Transfer Bus)と呼ぶ。

IPTバスの仕様は表2-1に示す通りであり、データ転送手順は以下の通りである。

(1) システムバスを介して、転送先PEのDMAレジスタにベースアドレス、転送語数および制御情報を書き込む。ただし、DMAレジスタとは図2-3に示されたBase Address Register、Word Counter RegisterおよびControl Registerの3つのレジスタのことである。
(2) 転送元PEのCPUは自分のDMAレジスタにも同様の情報を書き込む。
(3) IPTバスを獲得する。
(4) 転送先アドレスをIPTバスに出力する。
(5) 転送先PEのCPUをホールドする。
(6) 転送先PEが転送元PEにアクノリッジを返す。
(7) 転送元PEが転送先PEにアクノリッジを返す。
(8) データ転送を開始する。
(9) 転送元ローカルメモリと転送先ローカルメモリのいずれかにダイナミックRAMのページ境界がくると、一時的に転送を終了する。
(10) アービトレーションの結果、再びIPTバスを獲得すると、データ転送を再開する。
(11) 要求されたデータ転送が完全に終了するまで、(9)と(10)の動作を繰り返し実行する。

IPTバスは同期型のバスであり、あるPEのローカルメモリから別のPEのローカルメモリへDMA転送を行う。この転送はDMAコントローラのレジスタを介することなく転送先PEのローカルメモリへ直接に行っており、またダイナミックRAMのページ・モード・オペレーションを利用しているため、32Mbyte/secの高速転送が可能となっている。

なお、このIPTバスはPE間のデータ転送だけでなく、PEとマスタディスクコントローラとの間のデータ転送にも用いられており、ディスクアクセスの高速化に寄与している。

2.3 CPUボード

CPUボードの内部構成を図2-4に示す。本ボードは、基本的には、次の4個のブロックからなる。

(1)CPUブロック
(2)ローカルメモリブロック
(3)ローカルI/Oブロック
(4)バスインタフェースブロック

CPUブロックは、32bitマイクロプロセッサMC68020(16MHz)、演算プロセッサMC68881(12.5MHz)、ページ方式のメモリ管理ユニットMC68851およびキャッシュメモリより構成され、高い数値演算能力を持った仮想記憶システムを構築するのに必要なハードウェア機構を備えている。ただし、キャッシュは現状のシステムでは実現されていない。キャッシュとしては、ライトスルー方式の論理キャッシュを考えている。また、今回の試作システムに搭載しているOSはOS-9をマルチプロセッサ用に拡張した実記憶のOSであり、セグメント方式のメモリ管理を行っているため、メモリ管理ユニットMC68851は使用していない。

ローカルメモリはダイナミックRAMにより構成され、最大16Mbyteまで実装可能であり、CPUブロックからボード内のローカルバス経由で高速にアクセスができる。

ローカルI/Oブロックは、マイクロプロセッサMC68000を内蔵したインテリジェント・コミュニケーション・コントローラであり、シリアルチャネルを4チャネル(このうち2チャネルはDMAコントローラ

MC63450により高速のシリアル転送が可能)の他、汎用パラレルチャネルおよびSCSIインタフェースを持っており、外部とのコミュニケーションに十分な能力を備えている。また、このローカルI/Oブロックと CPUブロックとが通信を行うために、スタティック RAMにより構成される128Kbyteのデュアルポートメモリを持っている。

バスインタフェースブロックは、本ボードと VMEバスおよびIPTバスとのインタフェース回路 (BUS I/F) と、IPC コントローラ(Inter Processor Communication Controller)から構成されている。このうちBUS I/Fには、前節で述べたIPTバスによるDMA転送のためのレジスタや制御回路が含まれる。IPCコントローラはPE間通信を行うためのハードウェアであり、IPCレジスタと呼ばれる16bitのレジスタを内蔵している。このレジスタは他のPEから自由にアクセスができ、リクエストまたはレスポンスを出力するPEが、相手PEのIPCレジスタに自分のPE番号およびコミュニケーション・バッファへのインデックスとともにコマンドを書き込むことにより相手PEのCPUに割り込みをかけることができる。割り込みがかかると、相手PEにおいて通信用の割り込み処理ルーチンが起動され、PE間の通信が行われる。

3. ソフトウェア構成

本システムでは、各PE上のOSが個々のプロセスを独立して管理しており、そのOSの基本的な機能を次に上げる。

①メモリ管理
②プロセス管理
③割り込み管理
④システムコールのサービス
⑤システムの初期化

マルチプロセス機能を有する既存OSであるOS-9をベースとして、その上にマルチプロセッサに対応したPE間の通信や同期機能を効率よく動かすための拡張を行った。システムが全体として複数PEを並列に動作させるためには、OS間で協調してカーネルの処理を進めることが必要である。本システムでは、OS間の協調を実現するための手段として、OSのサービス単位であるシステムコールを使用する。すなわち、各OS間でのサービスの依頼や実行はシステムコールを単位として行う。本システムのソフトウェアの概略としては、図3-1に示す構成をとっており、既存OSにおけるアプリケーションとシステムとのインターフェース部をローカル処理とバスを経由した他PEへのアクセスに分割している。

3.1システムソフトウェアの構成

本システムでは、複数PE上で多数のプロセスを並列に実行させるために、OS機能の拡張を行っている。マルチプロセッサに対応して拡張した主な機能を以下に述べる。

(1)プロセッサ名によるカーネル管理機能の拡張
(2)グローバル・パス名によるI/O機能の拡張
(3)セマフォによる同期制御機能
(4)共有変数機能の付加

本システムでは、バス(ネットワーク)で結合したすべてのPEに独立したOSが存在し、それぞれが独立に各PE上のユーザプロセスの管理を行っているが、OSのシステムコールにPEの識別子(PEID)が扱える機能を加えることにより、他PE上へのプロセスの起動や停止などの制御機能、及びグローバルなパス名により他PE上の入出力デバイスにアクセスする機能、を実現している。プロセス間の同期は、IPCコントローラからの割り込みによりOSのIPCコントローラの制御部を経由して目的プロセスに通知される。(図3-2)

これらのパスとプロセスの管理をシステム全体にわたって一貫して行うために、パス及びプロセス管理用のテーブルを設けている。パス/プロセス管理テーブルにはシステム全体でユニークなパス/プロセス番号が登録されており、各PE上のOSがこのテーブルを参照することによりパス及びプロセスをシステム内で特定することができる。リクエストを出したPE上のOSは、システムコールのパラメータからプロセスを特定し、対応するPEに割り込みをかける。割り込みをかけられたPEでは、パラメータを受け取り、要求されたサービスを自PE上で提供する。パスについても同様に割り込み要求の授受を行い、目的PE上にパスについてのシステムコールを発行することで異なるPE上のプロセス間通信を実現している。(図3-3)

3.2プロセスの制御

プロセスの起動、停止、PEへの割当てを行うために、以下のシステムコールを提供している。これらの関数では、従来のプロセス名にPEの指定を付加することにより、他PE上でのプロセスの起動や停止の制御を行うことができる。

(1)プロセスの生成

目的PE上へのプロセスの生成は、プロセス名にPE名を付加して、以下のようにシステムコール

p_fork(PE名!プロセス名)

をコールすることにより行う。

(2)子プロセスの終了との同期

親プロセスよりforkされた子プロセスの終了を確認するために、次のシステムコールを用いる。

p__wait( )

(3)PEへのプロセス割当て

プロセスをPEへ割当てる際に、システム全体の負荷を均等に分散させるためには、

①静的割当て法
②動的割当て法

に代表される各種の手法が考えられる。複数CPUと共有メモリ(マルチキャッシュ)で構成されるような実用的なマルチプロセッサでは、メモリ空間内に存在するOSがユニークであり、メモリ空間内でのプロセスの移動が不要であるため、CPUへのプロセス割当ては動的に実現することが容易であるが、PEの数だけOSが存在するシステムでは、

①動的割当ては共有メモリ上のプロセスに限定
②PE上のプロセスの管理を移譲
③PE間でのプロセスの移動

などの手法を導入することが必要である。③ではプロセスの移動にともなうオーバヘッドが非常に大きい。ただし、プロセスはポジションインディペンデント(PIC)なコードで構成しているため、他メモリ空間への移動は可能である。②では、プロセスの管理情報を他PEに引き渡すが、プロセス自体の移動は行わないためオーバヘッドは少ないが、システムの構造上他PEのローカルバス上でアクセス競合を引き起こすことになり効率的でない。上記の理由から動的な割当てを行うプロセスについては、①の手法を使用することとする。本システムの現状としては、明示的にPEを指定する静的割当て法を実現している。

3.3プロセス間コミュニケーション

本システムには、異なるPEにまたがるプロセス間のデータ交換手段として、名前付きパイプ(named pipe)によるメッセージ伝達機能が有る。名前付きパイプは、プロセス間のデータ交換手段として次の機能を実現している。

(1)通信チャネル

通信チャネルの獲得は、共有メモリ内に設けた通信チャネルを管理するテーブルにチャネル名称を登録することにより行う。チャネル名にはシステム全体にグローバルな名称を使用しており、この名前により任意のプロセスが関数pp__link( )をコールすることにより登録済みチャネルとのリンクを確立することができる。通信チャネルの作成･削除などの処理は、テーブルへの登録を動的に管理することにより実現しており、そのテーブル構造を図3-4に示す。

pp__create( )
pp__delete( )　　delete前のlink-countを返す
link-countは登録チャネルに連結されているプロセスの数である。

(2)非同期式メッセージ伝達

本システムでは、メモリ内にFIFOバッファを設けることにより非同期式メッセージ通信機能を実現している。基本的な通信機能は、封鎖型のメッセージ通信であるが、チャネル状態を調べる関数pp__check( )により非封鎖型メッセージ伝達も行うことができる。

pp__write( )
pp__read( )
pp__check( )　　チャネル状態(empty、full、ready)を得る

(3)通信チャネルの構成

通信チャネルは、基本的に単方向のメッセージを伝達するものであるが、システム･コール単位でのメッセージ通信が排他的であることを利用して1プロセスが多数のプロセスとの間でメッセージを交換することができる。これに加えて、通信チャネルの指定がグローバル名であるため、親子関係にないプロセス間のメッセージ通信が可能となる。これらの複雑な通信網を構成するために次の関数を使用する。

pp__link( )

後の章で述べる連立方程式の解を求める実験では、pp__link( )を用いて1個の管理プロセスと複数の作業プロセスという関係を構成してガウスの消去法を並列に実行している。

3.4共有変数

本システムでは、複数プロセスに分割されたアプリケーションを、複数PE上で並列に動作させるために以下の拡張機能を新たに加えている。

3.4.1共有変数のアクセス

共有変数は、OSの機能として提供されるデータモジュールとロック機構を用いて実現した。データモジュールは、共有メモリ上に割当てることを原則としており、すべてのPEからアクセスできる様に構成している。(図3-5)

(1)共有変数用メモリの動的割り当て

ユーザプロセスは、共有変数を共有変数テーブルに登録(変数定義)し、これに必要なメモリ領域を確保する。プログラムモジュールとのリンクは、共有変数テーブルに格納されている変数名の参照により行う。共有変数の登録や削除は、共有変数テーブルを動的に管理することにより行う。

ps__create( )
ps__link( )
ps__delete( )

ここで、他プロセスからリンクされている共有変数をps_delete( )で消去することは禁止している。これは、link countを用いて実現できる。

(2)共有変数アクセスの相互排除

プロセス間での共有変数アクセスの相互排除を実現するために、ロック機構を設けた。本ロック機構では、きわどい領域を不可分に実行する機能の他に、共有変数の定義とメモリ領域の獲得機能を備えている。(図3.6)

pl_create( )
pl_link( )
pl_lock( )
pl_unlock( )
pl_delete( )

pl_create( )、pl_link( )、pl_delete( )については共有変数と同様であるが、pl_lock( )、pl_unlock( )はイベント管理機能を用いて共有資源の排他的アクセスを保証する関数である。

4. 基礎的実験

本マルチプロセッサシステムの性能を評価するために、PE間の通信速度の測定と、協調型プロセスからなるジョブの処理時間の測定を行った。PE間の通信速度の測定には、異なるPE上に存在する二つのプロセス間のパイプによる通信を取り上げ、協調型プロセスからなるジョブの処理時間の測定には、科学技術計算分野における代表的な問題である、連立一次方程式の求解を取り上げた。

ただし、現段階では、ハードウェア的にもソフトウェア的にも未完成の部分があるため、これらの実験におけるPE間の通信には、システムバス経由でパイプによるプロセス間通信を用いているのみであり、基礎的な実験にとどまっている。

4.1パイプによるプロセス間通信

前章で述べた名前付きパイプを用いて、異なるPE上に存在する二つのプロセス間で通信を行わせ、データ転送速度を測定した。この結果を表4-1に示す。なお、比較のため、同一PE内のプロセス間通信の場合のデータ転送速度も測定した。この表より、転送データ量によって若干の違いはあるが、いずれの場合も80kbyte/sec前後の転送速度を示しており、同一PE内のプロセス間通信の速度は、PEが異なる場合に比べ高々10%程度しか向上していない。システムバスのハードウェア的な転送速度は5~10Mbyte以上は得られているので、パイプを使用した場合にはOSによるオーバヘッドが非常に大きいことがわかる。今後、パイプのバッファサイズの調整(測定時のバッファサイズは90byte)やパイプ関係のOSルーチンの修正が必要である。

4.2連立一次方程式の求解

連立一次方程式を解くというジョブを1個の管理プロセスと複数個の作業用プロセスに分割し、管理プロセスと作業用プロセスがパイプによりプロセス間通信を行いながら実行した場合の処理時間の測定を行った。作業用プロセスが2個の場合の、このジョブ実行の様子を図4-1に示す。なお、対象とする連立一次方程式の係数行列はバンド構造とし、解法としてはガウスの消去法を採用した。

処理時間の測定結果を表4-2に示す。係数行列のバンド幅が小さい場合には作業用プロセス内での処理時間に比べ通信時間の割合が大きいため、複数の作業用プロセスを別のPEに割当てても、あまり大きな処理速度の向上は見られない。

今回の実験では、ハードウエア的にもソフトウェア的にも未完成の部分があるため、十分な測定データが得られていないが、現状のパイプの通信速度は、本手法に基づく連立一次方程式の求解には不十分である。従って、前節で述べた方法でパイプの通信速度を向上させるかまたは、共有変数を用いた手法に変更する等の対策が必要である。

5.あとがき

本稿では、バス結合のマルチプロセッサシステムについて述べ、その試作機を評価するための実験の結果について報告した。本マルチプロセッサシステムはメモリおよびOS機能を各PE(プロセッサ・エレメント)に分散させたネットワーク型のシステムであるが、システムバスの他にPE間で大量データを高速に転送する専用バス(IPTバス)を有する二重バス構造のシステムである点に特長がある。このため、プロセス間通信量の多いアプリケーションや動的なプロセス割当てにも対応できるハードウェア構成となっている。

しかし、現段階ではIPTバス関係のハードウェアは未完成であり、ソフトウェア的にもPE間の通信としては、パイプによる通信のみが実現できている状態であるので、今回の実験は基礎的なものにとどまっている。今後、IPTバス関係のハードウェアを完成させ、ソフトウェア的にも共有変数用メモリの割当てや共有変数アクセスの相互排除などの機能を実現し、より詳細な性能評価・検討を行う予定である。

謝辞

日頃御指導を頂いている西岡所長に深謝致します。また、貴重な御助言および活発な御討論を頂いた大阪大学の緒先生方に感謝致します。

参考文献

[1]高橋:「並列処理のためのプロセッサ結合方式」、情報処理Vol.23 No.3、1982年3月
[2]池原:「マルチプロセッサ方式における共用メモリアクセス競合の解析」、信学会論文誌 Vol. J63-D No.4、1980年4月
[3]布谷、住田、橋田:「マルチプロセッサシステムの性能評価」、電子通信学会誌Vol.66 No.12、1983年12月
[4]下条、宮原、高島:「通信競合を含めたマルチプロセッサにおけるプロセス割り当て問題」、信学会論文誌(D) Vol. J68-D No.5、1985年5月
[5]下条、宮原、高島:「分散システムにおけるプロセスの割り当て法の比較・検討」、信学会EC85-16、1985年8月
[6]日本モトローラ:「VMEbus アーキテクチャ・マニュアル」、CQ出版、1984
[7]Hwang,K., Briggs,F.A., :"COMPUTER ARCHITECTURE AND PARARELL PROCESSING," McGraw-Hill, 1984
[8]Andrew,G.,R. and Schneider,F.,B., :"Concepts and Notaions for Concurrent Programming," ACM Computing Surveys, Vol. 15, No.1 pp3-43
[9]Bitar,P., Despain,A.M., :"Multiprocessor Cache Synchronization Issues, Innovations, Evolution," 13th ISCA, 1986
[10]Cheriton,D.R., Slavenburg,G.A., Boyle,P.D.,:" Software-Controlled Caches in the VMP Multiprocessor," 13th ISCA, 1986
[11]Sweazey,P., Smith,A.J., :"A Class of Compatible Cache Consistency Protocols and their Support by the IEEE Futurebus," 13th ISCA, 1986

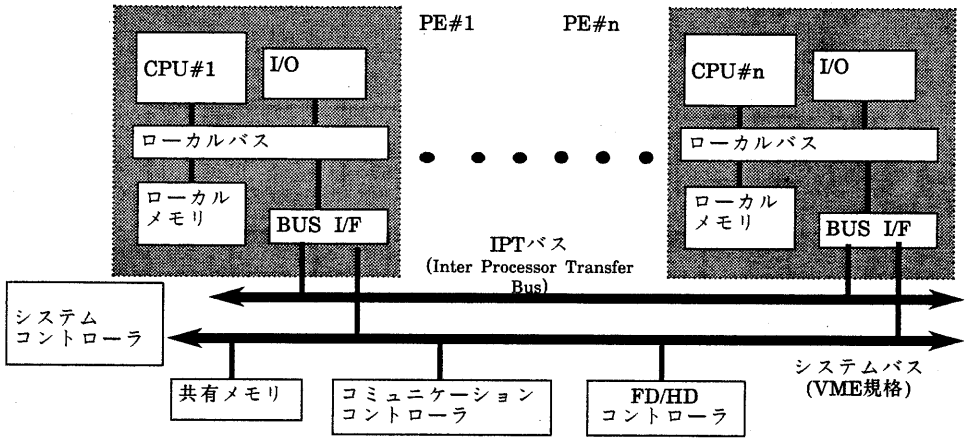

図2-1 ハードウェア概念アーキテクチャ

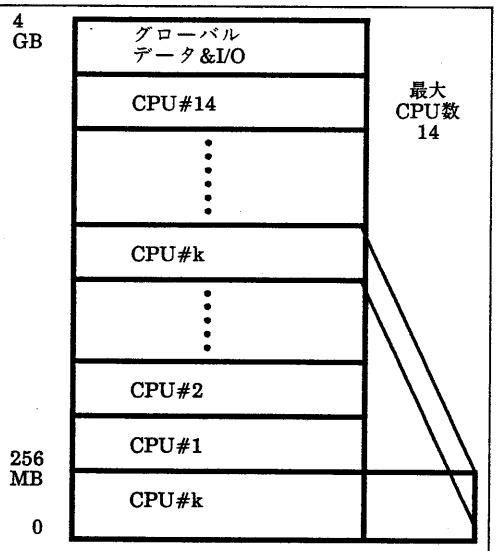

図2-2 メモリマップ

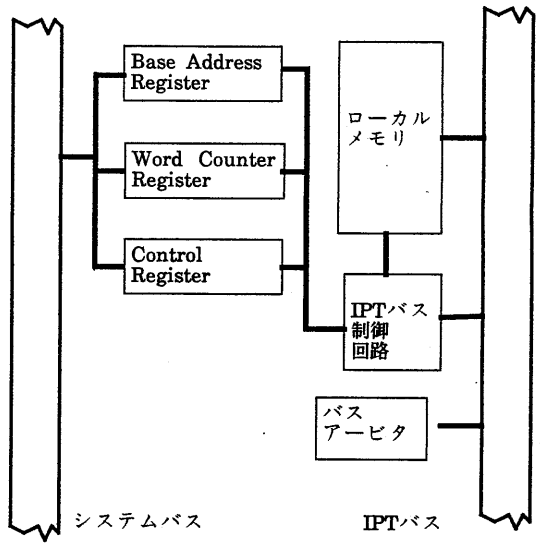

図2-3 CPUボード内のIPTバスインタフェース回路

表2-1 IPTバス(Inter Processor Transfer Bus)仕様

・転送速度	max. 32Mbyte/sec(クロック周波数8MHz)
・転送ビット幅	32bit
・転送方式	DMA転送(DMACのレジスタを介さないメモリ-メモリ間転送)
	DRAMのページモード・オペレーションを利用した高速転送
・アービトレーション	ラウンドロビン方式

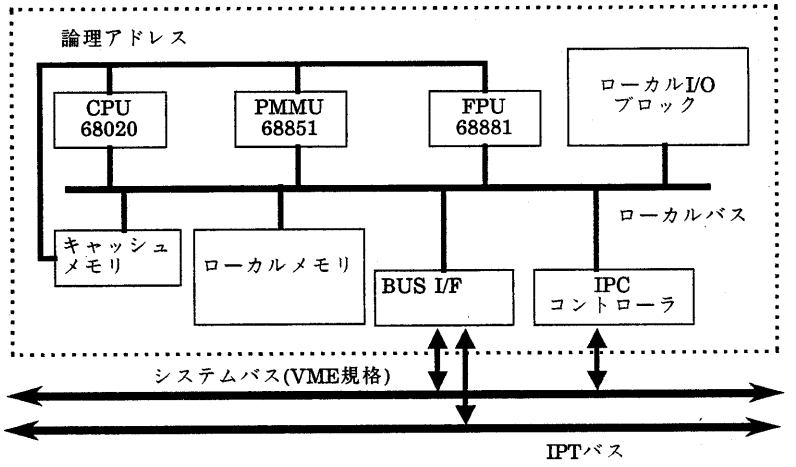

図2-4 CPUボード

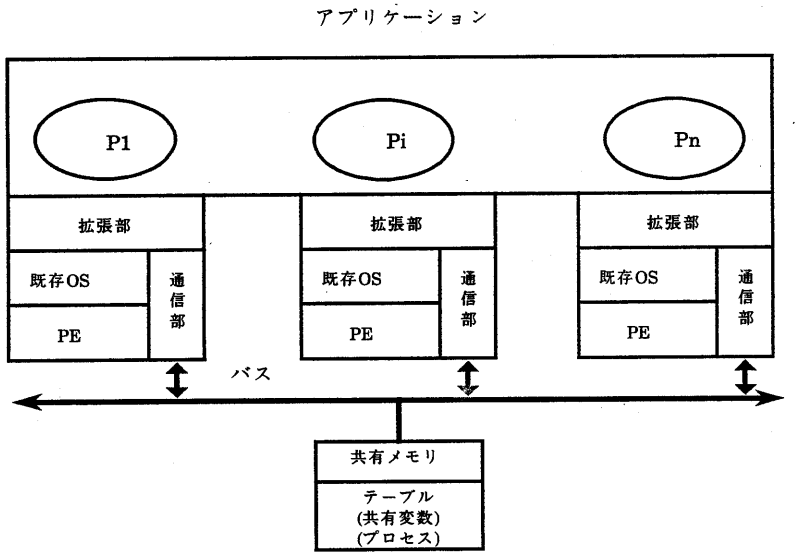

図3-1 ソフトウェア構成

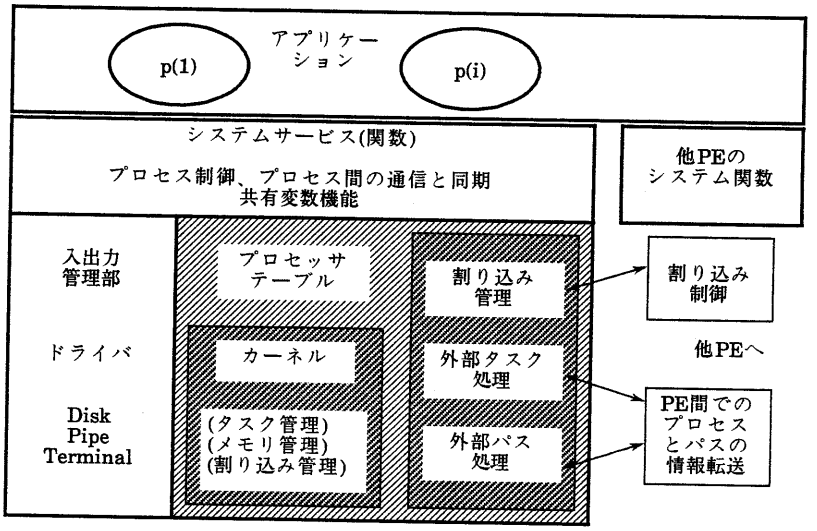

図3-2 システムソフトウェアの構成

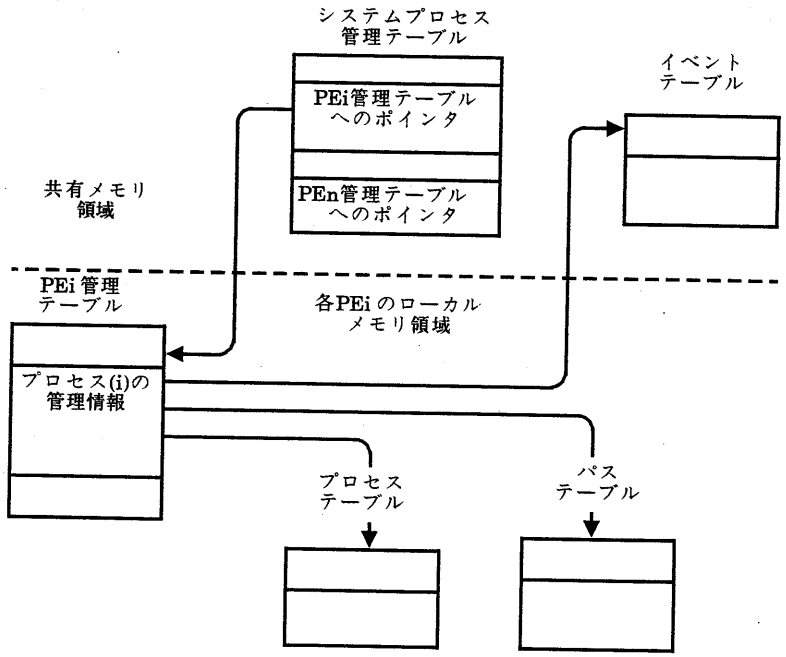

図3-3 パス、プロセス、イベントの管理情報の構造

チャネル名	パスID	リンク数	next-pointer
chan1	12	2	5
chan2	23	3	0(terminater)
chan5	3	1	3

図3-4 通信チャネルのテーブル構造

変数名	共有メモリ領域へのポインタ	変数サイズ	リンク数
var1	アドレス	20	2

図3-6 テーブル構造

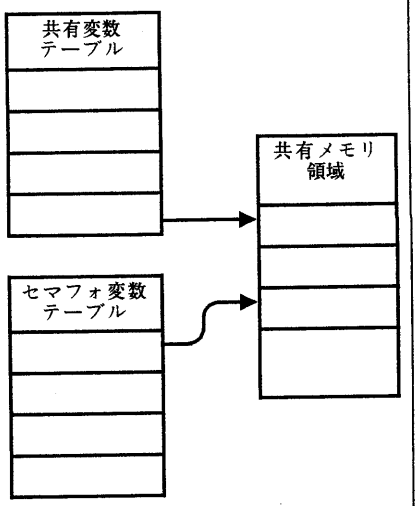

図3-5 共有変数とセマフォ

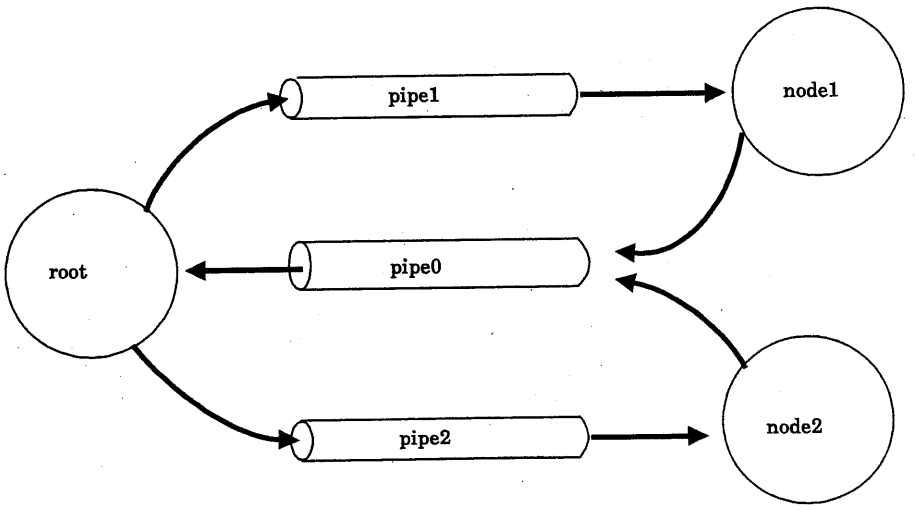

図4-1 連立一次方程式求解におけるプロセス間通信
(作業用プロセスが2個の場合)
root :管理用プロセス
node1、node2 :作業用プロセス

表4-1 パイプによるプロセス間通信

プロセスのPEへの割当て procA/procB	転送データサイズ [byte]	転送時間(往復) [m sec]	転送速度 [Kbyte/sec]
PE1/PE1	512	10	102
PE1/PE1	1024	30	68
PE1/PE1	4K	90	89
PE1/PE1	40K	860	93
PE1/PE1	200K	4270	94
PE1/PE2	512	20	51
PE1/PE2	1024	30	68
PE1/PE2	4K	100	80
PE1/PE2	40K	930	86
PE1/PE2	200K	4620	87

ただし、 procA,procB :プロセス名
PE1,PE2 :プロセッサ·エレメント名
データ転送はprocAとprocBの間での往復転送

表4-2 連立一次方程式の求解

(1)未知数100、バンド幅19

nodeプロセス数	nodeプロセスのPEへの割当て node1/node2	処理時間 [m sec]	相対速度
1	PE1	7210	1.00
1	PE2	7450	0.97
2	PE1/PE2	5490	1.31

(2)未知数100、バンド幅50

nodeプロセス数	nodeプロセスのPEへの割当て node1/node2	処理時間 [m sec]	相対速度
1	PE1	34260	1.00
1	PE2	33890	1.01
2	PE1/PE2	21420	1.60

ただし、 係数行列はバンド構造
解法はガウスの消去法を採用
root,node1,node2 :プロセス名
rootは管理プロセスであり、node1およびnode2は作業用プロセス
PE1,PE2 :プロセッサ·エレメント名
rootは常にPE1に割当てる