主記憶共有型多重プロセッサシステム

Multi-Processor System with shared Main Memory

渡辺 定久　金田 悠紀夫　大岸 洋　内堀 義信　山崎 晴久
SadahisaWATANABE　YukioKANEDA　HiroshiŌGISHI　YoshinobuUCHIBORI　HaruhisaYAMAZAKI
電子技術総合研究所　村田電機製作所
Electrotechnical Laboratory　MurataElectricWorkLtd.

1．はじめに　小形で高速の演算エレメント（ＰＥ）多数を結合したアレイプロセッサの研究は、多量の計算時間を要する大形の連立一次方程式の解法計算や偏微分方程式の高速計算を目指して、１９６０年代前半からＩＬＬＩＡＣⅣシステムを始めとしていくつか精力的に進められてきた。

一方、最近のミニコンピュータやマイクロコンピュータの高性能化と低廉化に着目してこれら小形コンピュータ多数を組合せた多重プロセッサシステムが将来の有力な計算機システムの一形態となると考え、汎用性のある多重プロセッサシステムの開発研究も進められており、C.mmP.，ＰＲＩＭＥ等のシステムがその代表例といえる。しかし現在までのところ、これらのシステムに対する評価は不確定の点が多く、確立された高い評価を受けているとは必ずしもいえない。これは単に「並列」といっただけでは一般的、抽象的すぎ具体的な意味付け一定量的な並列性の把握一が困難で具体性を持った研究の展開や評価が困難となるためと考えられる。そこでわれわれは大規模システムの解析に焦点をしぼり、現在までに開発された代表的で、豊富な蓄積のある計算アルゴリズムを取り上げ、この方面の応用に対し極めて有効に働く一アルゴリズムの持つ並列性を十分生かし、効率よくしかも柔軟に対応できる一並列処理システムを小形プロセッサを組合せて実現し、これを基本としてより一般的で、より大規模な多重プロセッサシステムの研究をハードウェア、ソフトウェア、応用の三面から進めていくことを計画している。大規模システム解析に用いられる手法としては、建造物、船などの構造解析に出てくる有限要素法における連立一次方程式の解法計算、固有値や固有ベクトルの計算、大形プラントのスケジューリングなどに出てくる大規模線形計画計算、流体力学計算などに出てくる偏微分方程式の数値解析計算などを上げることができる。これらはいずれもスパース行列一非零要素数は通常全要素の１％以下で問題が大形化するにつれてスパース性が増す一に対する計算が主で、スパース性を生かした計算アルゴリズムがそれぞれ開発されている。並列計算という立場でこれらの計算アルゴリズムをながめてみると、多レベルの並列性が中に含まれていることが判る。オ１レベルとしては行列中の要素のアドレス計算と2要素間の演算の並行計算がある。オ2のレベルとしては、行ベクトルまたは列ベクトルに対する同一タイプの演算で結果が同一サイズのベクトルとなるものの並行計算で各要素を並行して求める。ベクトル同志の加減算、ベクトルに対する定数の乗除算などがこれに当る。オ3のレベルは、複数の列ベクトルまたは行ベクトルに対して同時にレベル2の計算を行う場合で、代表的な例としては、行列または部分行列の全体要素を並行して計算する場合がこれに当る。オ4のレベルはオ3レベルの計算を複数個並行して行う場合で線形計画計算においてマルチプライシングを行ったときのＦＴＲＡＮ計算とか実対称帯行列の固有値計算における三重対角化の過程にあらわれる特殊な鏡像変換計算などはそのアルゴリズムを分析するとレベル4と見なすことができる。これら計算の流れ図は図１のようになる。図

1から明らかなことは、最も内側の並列計算パスは二重であり、最も外側の並列計算パスは数～数十重となることが多いが、レベル2、レベル3の並列計算パスは行列の行数または列数のオーダとなることが多く極めて多重度の高い並列計算パスとなる。また最も内側にあらわれる逐次処理を行う計算パスは極めて短い（例えば浮動小数点の乗除算1個程度）のが普通である。全並列計算パスを同時に並行計算するためには、1．極めて多数のプロセッサが必要。2、1で述べた多数のプロセッサにデータと命令を円滑に高速に分配しなければならない。3、極めて多くの fork，join 処理をオーバヘッド少く行わねばならない。等が必要となる。1、2、3、の要請を100%満足させるシステムは実現不可能であるが、これらの点をできるかぎり満足したシステムを想定してみると、プロセッサ数～十数台からなり、主記憶を共有したシステムが考えられる。特にプロセッサユニットは高速で、並行動作が可能な高速浮動小数点演算ユニットを持ちバーチャルアドレスを実現するマッピング機構を付加したものが望まれる。主記憶は大容量高速で、主記憶部とプロセッサ部との結合はマトリクス・スイッチが考えられる（図2）。3で述べた点を考慮すると、各プロセッサユニットが空き状態となることなく均等動作し、かつ fork、join の処理回数を極力少くすることがシステム全体の効率向上につながると考えられる。そのためには、全並列計算パスを生かすというアルゴリズムではなく、全体の計算の流れからプロセッサユニットの個数だけの並列計算パス群を抽出し、各群に含まれる並列パスを整理統合し、不要となった fork，join 部の処理を取り除き、処理時間の長い単一計算パスに変換する。各計算パスをプロセッサに分配して並行計算を行うという方法が最も効率的と考えられる。われわれは以上に述べた事柄を実システムで確認するため、NOVA02システムを中核とした多重プロセッサシステムのパイロットモデルを開発している。以下パイロットモデルを中心にハードウェア構成ならびにサポートソフトウェアについて概説する。

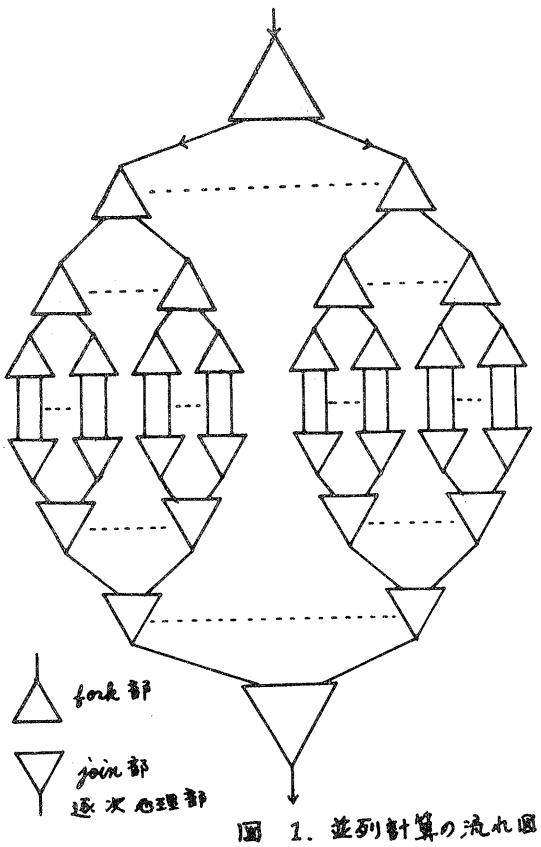

図 1. 並列計算の流れ図

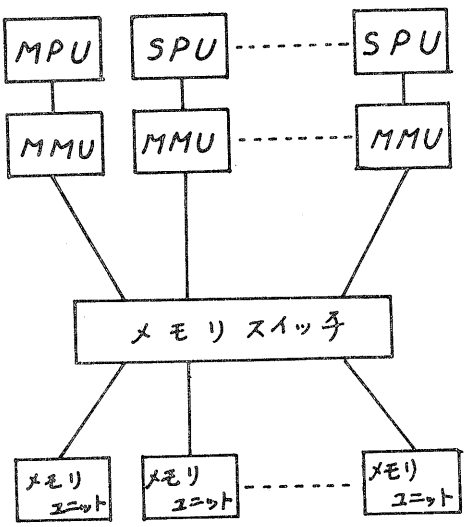

図 2. システム構成

2. パイロット システムの構造

2.1 主記憶共有システム

図3は 我々のシステムの構成の大要を示すブロック図である。図に見るように、我々のシステムは、複数個のプロセッサ（MPU，およびSPU）が、ハード的には、同じレベルで共有メモリにアクセスできると同時に、プロセッサ間通信機構（ICD）を介して，相互に通信できるようになっている。したがって、システム構成上の、中心課題は、共有メモリ管理機構と、ICDを、どのように具体化するかという点にしぼられる。

共有メモリ管理方式としては、まず、MPUが、各SPUが使用するメモリエリヤ すなわち、論理アドレスのページと共有エリアのフィジカルアドレスのページとの対応表を作成し、各SPUはこの表にしたがって、アドレス変換を行う方式を採用する。この場合各プロセッサからのメモリリクエストが同じアドレスに対して同時に行なわれるのを防止するためのチェック機構（CC、CHECK）が必要となる。また、アドレス変換を必要としない場合には、共通エリアが各プロセッサにわりつけられ、各プロセッサごとに管理されることになる。

プロセッサ間通信は、通信機能を、複数のプロセッサに分散するのと、中央のマスタープロセッサに集中するのと、大きく2つに分れる。本システムの場合は、前者の方法とした。これは、システムの汎用性を考慮し、プロセッサの動作レベルの重みを、同じレベルにするためである。但し、PU1は、外部装置接続と、主記憶管理機構のため、動作レベルは、他のプロセッサより、やや重くなる。これにより、各プロセッサ間の相互通信が、直接行なえる。

次に、通信方法であるが、大別して、通信要求の割り込みを行なう方法と、割り込みとデータの転送を行なう方法がある。これは、前者の方法をとった。この方法は、通信装置が、簡単であること、主記憶が共有であるため特別なデータ通信媒体がなくてもよいこと、メーカーから提供されるOSを中心として使用するので、割り込み処理に制限を受けることなどの理由があるためである。

通信の管理機能も、中央のマスタプロセッサに集中する方法と、各プロセッサに分解する方法とに、大きく分けられる。これは前者の方法に、よって行うこととした。これは、通信内容の転送媒体に主記憶を使用するため、主記憶の管理にも影響を 受けるためである。

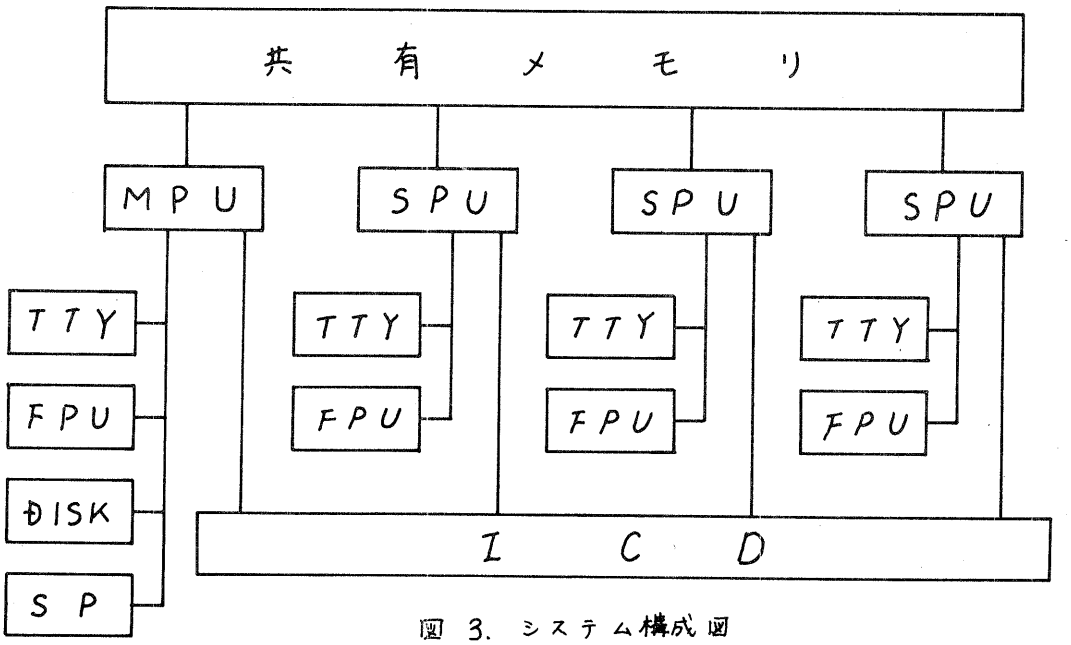

図 3. システム構成図

プロセッサとしては、市販のミニコンを使用できれば都合がよい。我々は、ミニコンとして、日本ミニコン社製ＮＯＶＡ－０２を使用することを予定している。図４はＮＯＶＡ－０２のメモリと各バスライン、ＣＰＵとの関係を示すブロック図である。

これらデータの流れを いくつかの例で見ると、メモリの読み出しでは、ＭＢＯ　ＭＰＸ（メモリ　バッファ　アウトプット　マルチプレクサ）より、出力されたアドレスが、Ｄ－ＭＰＸ（データ　マルチプレクサ）を通り、Ｍ（アドレスレジスター）に、格納される。ＭＡのアドレスを用いて、読み出されたデータは、ＭＢ（メモリボード内メモリバッファ）に格納され、ＭＥＭ　ＢＵＳ（メモリ→ＣＰＵバス）を通り、ＣＰＵのＭＢに転送される。ＲＥＡＤ　Ｉ／Ｏ命令の場合、Ｉ／Ｏ　ＢＵＳ（インプット、アウトプット　バス）のデータは、Ｄ－ＭＰＸを通ってメモリボード内のＭＢを中継レジスターに、一度格納される。さらに、ＭＥＭ　ＢＵＳを通って、ＣＰＵに転送される。

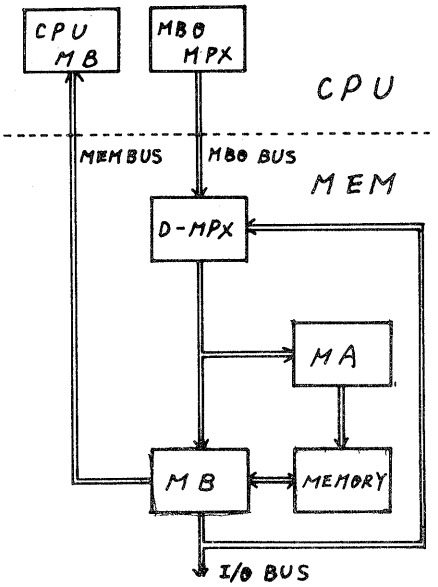

図４．NOVA-02 ブロック図

図５に、ＭＭＵ（メモリ管理ユニット）の構成と、ＩＣＤ、主記憶装置、Ｉ／Ｏバスの、関係ブロック図を示す。４台のプロセッサは、バスライン、外部装置制御ライン、メモリ制御ライン、プロセッサタイミングクロックラインが、１台のＭＭＵに、接続されている。ＭＭＵ内の４台のＭＵ（ＭＡＰユニット）は、それぞれのプロセッサを受け持っている。

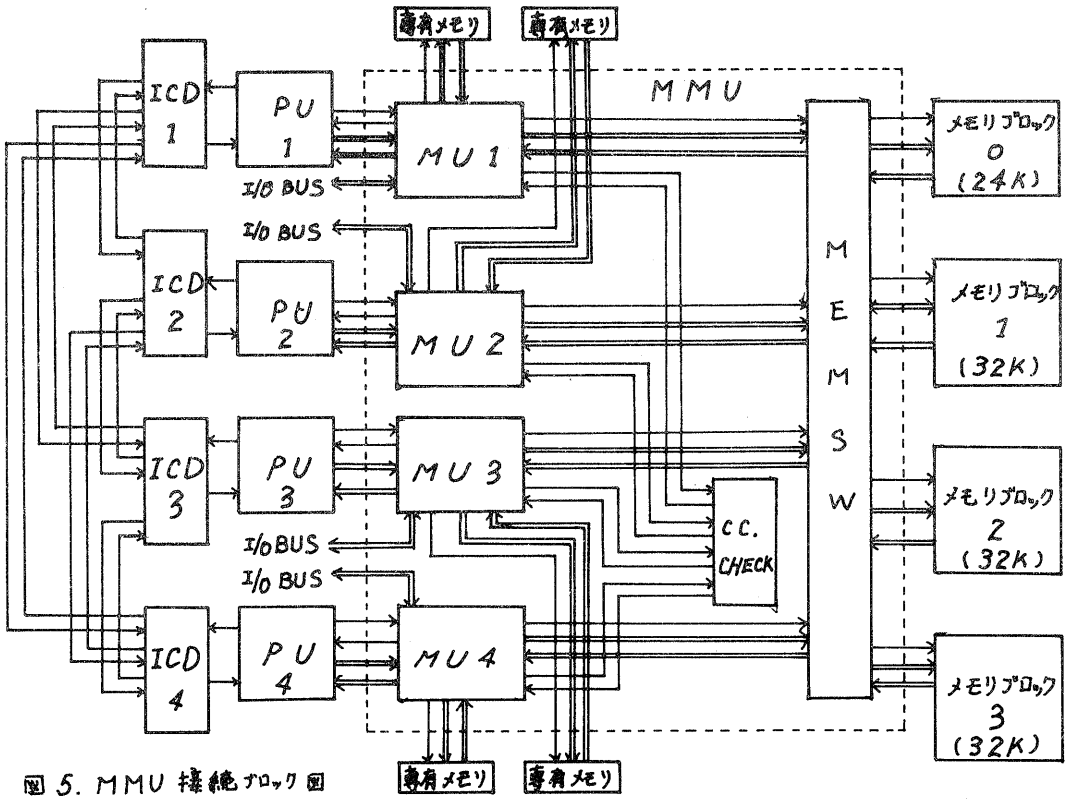

図５．ＭＭＵ接続ブロック図

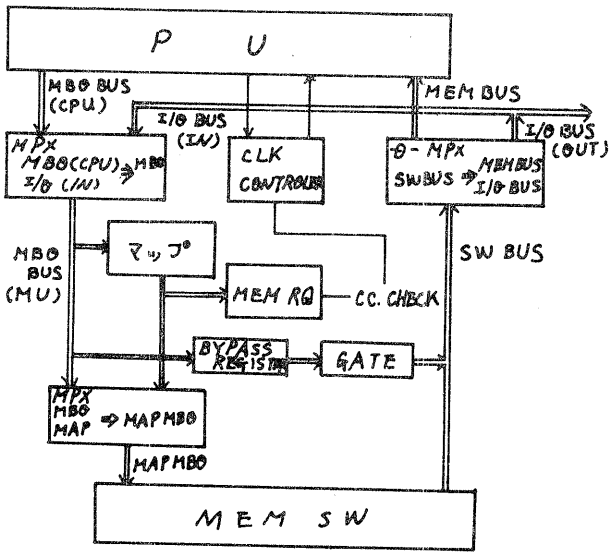

図 6. MU ブロック図

MUの機能としては、アドレス変換、アドレスチェックを行い、MEM RQ（メモリ リクエスト）を発生し、RP（CPU SELECT）を、受け取るまで、待機する。RPを受けとると、MEM SW（メモリ スイッチ）を、制御し、アドレスをメモリに転送する。つづいて、プロセッサに、メモリサイクルを実行させる。図6に、MUの機能ブロック図を示す。

CC. CHECK（CONCURRENT CHECK）は、各MUが、発生したメモリリクエストを、認識すると、優先順位の高いCPUを、選択して動作させる。

CC. CHECKは、メモリブロック毎に行い、前回メモリサイクルを、行ったプロセッサを、記憶しておき、その情報によって、優先順位が変化する。

ICD（プロセッサ間通信装置）は、通信を行なおうとしているプロセッサが、相手のプロセッサのICDのフラグをセットすると、セットされたICDは、プロセッサに、プログラムインターラプトを、要求する。受信側プロセッサが、割り込みを受け付けると、プロセッサ間通信が、達成される。通信できる内容は、割り込み要求だけで、その他の内容は転送できないので、共有メモリ内に、メッセージを、セットしておく必要がある。

2.2 MMUによるメモリ管理機構

2.2.1 アドレス変換方式

プロセッサが持っている絶対アドレス指定範囲を越えて、アドレス指定しようとする場合、当然アドレス変換を、行なう必要がある。本システムでは、1kWを、1ページとして、ページアドレスについて、マップレジスターによるアドレス変換を、行なっている。

図7において、論理アドレスは、32kWであるのに対して、実行アドレスは1.8kW指定できる。論理アドレスの、ページ部（5ビット）で、マップレジスターの内容（7ビット）を、読み出し、ページアドレス部を、5ビットから7ビットに、変換することによって、1.8kW（8kWの専有メモリを含む）の、任意の32kWを、アドレス指定できるようになっている。

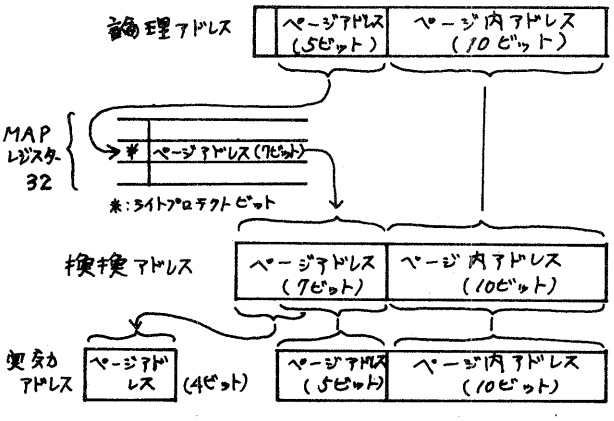

図 7. アドレス変換

また、プロセッサが、持っているアドレス指定方式は、全て実行できる。

アドレス変換に関しては、変換を行なわない、スーパーバイザモードと、ユーザーマップを用いて変換する ユーザーモードがあり、データーチャネルは、別に変換用マップを、持っている。

2.2.2 メモリSW（MEM SW）

主記憶（コアメモリ）は、通常、4K、8K等の単位で、実装されていることが多い。図7の、変換アドレスと、実行アドレスで示したように、メモリボードに、一転送するデータ（15ビット）が、実装ボード内アドレスとなる。このアドレスは、ページ内アドレスと、ページアドレスの一部によって構成される。しかし、この15ビットのアドレス全てが、有効なのではなく、実装されているメモリの容量によって、有効語長は、変化する。図では、NOVA0Sで、8KW単位のメモリボードを、用いる条件で、記入されている。したがって15ビット中、2ビットは、無効であるが、アドレス語を構成するために、挿入されている。図中、4ビットで、構成されているページアドレスを示したが、このアドレスは、実装単位（ボード単位）毎の、アドレス（BS：BOARD SELECT）として、用いられ、MEM SW（メモリ スイッチ）の、制御信号にも、用いられる。

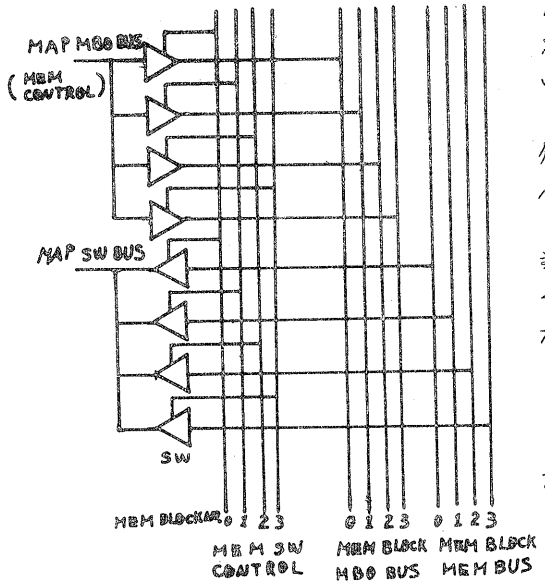

図8. MEM SW

MEMSWによって、切り換えを、必要とする信号線は、メモリの入出力データバス、制御信号合せて、40～50信号線である。

ボード毎に、切り換えたのでは、MEMSWの数と配線量は、膨大となる。また、CPUが、4台であるなら、プログラムを、メモリ効率よく作成すれば、使用するメモリは、4ブロックで充分である。

したがって、メモリを、4群に分け、8バスライン（4入出力バス）と、4制御ライン群の切り換えを行なう。

図8は、MEM SWの、1データラインを示した。制御線用SWは、図の上半分だけでよい。

2.2.3 同時チェック（CC CHECK）

同一のメモリブロックを、2台以上のCPUが、アクセスした場合、メモリサイクルが、重なり合うと、読み出されたデータ等の、内容の信頼性は、ほとんどなくなってしまう。このような状態に、ならないように、制御するのが、CC、CHECK（CONCURRENT CHECK）の、機能である。ブロック図を、図9に、示す。

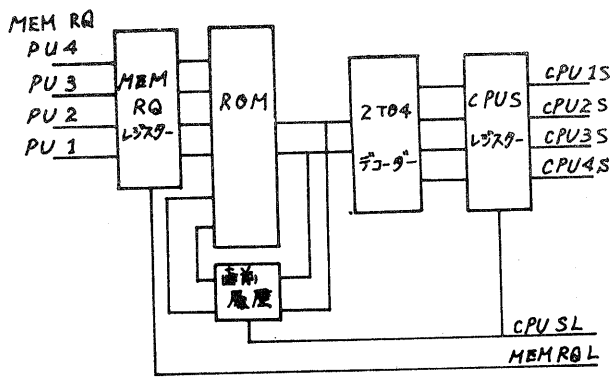

図9. CC. CHECK

各MUより、出されたMEM RQ（メモリアクセスリクエスト）を、レジスターに、一旦受け取り、リクエストの状態と、直前履歴レジスターの、内容により、選択されるプロセッサが、決まる。

メモリサイクルの間、CPU S（プロセッサ）を選択する信号）は、メモリサイクルの間、保持されている。

直前履歴レジスターは、CC CHECKが、管理しているメモリブロックの各々に、最後にアクセスしたCPUが、2ビット（4台）で、記憶されていて、そのプロセッサの、優先順位が、最低になっている

ことを示す。優先順位のオスの条件は、プロセッサのコード（番号順）である。
選択されなかったＣＰＵは、選択されるまで、クロックが停止し、いかなる動作も行なえなくなるが、これは高々（プロセッサ台数×メモリサイクル時間）ですむので、問題にならない。

２．２．４　ＭＭＵ　制御命令

（１）　ＬＯＡＤ　ＭＡＰ……アドレス変換に、使用するＭＡＰの書込み命令
（２）　ＬＯＡＤ　ＰＲＯＴＥＣＴＩＯＮ　ＣＯＮＴＲＯＬ----各種保護機能の、動作開始命令
（３）　ＥＮＡＢＬＥ　ＵＳＥＲ　ＭＡＰ----プログラムの、アドレス変換を、開始する命令（ユーザーマップ）
（４）　ＩＮＩＴＩＡＴＥ　ＰＡＧＥ　ＣＨＥＣＫ----ユーザマップの内容を、ＳＴＡＴＵＳレジスターに格納する。
（５）　ＲＥＡＤ　ＳＴＡＴＵＳ……ＳＴＡＴＵＳレジスターの内容を、ＣＰＵに転送する。
（６）　ＲＥＡＤ　ＩＮＳＴＲＵＣＴＩＯＮ　ＡＤＤＲＥＳＳ……エラーを生じた命令の論理番地をＣＰＵに送る。
（７）　ＲＥＡＤ　ＩＮＶＡＬＩＤ　ＡＤＤＲＥＳＳ……エラーを生じた論理番地を　ＣＰＵに送る。
（８）　ＳＵＰＥＲＶＩＳＯＲ　ＣＡＬＬ……スーパーバイザーモードにして、指定番地へＪＵＭＰする。
（９）　ＥＮＡＢＬＥ　ＳＩＮＧＬＥ　ＣＹＣＬＥ……一時的に、ユーザーモードにする命令。
（１０）ＩＮＴＥＲＶＡＬ　ＳＴＥＰ……メモリサイクル毎に、指定された時間ＣＰＵが停止する。
（１１）ＡＤＤＲＥＳＳ　ＳＴＯＰ……指定された論理番地のアクセス直前で、ＣＰＵが停止する。

２．３　プロセッサ間通信装置

プロセッサ間通信装置（ＩＣＤ）は、各プロセッサの外部デバイスとして、ＭＭＵを経由しないで、接続されており、通信を扱かうデバイスは、それぞれ固有の、かつ、各ＣＰＵ内で、同一のデバイスコードを持っている。通信方法は、図１０に、示すように、送信側プロセッサのデバイスで、通信相手のプロセッサを示すコードを持ったデバイスに対して、フラグセット命令で、送信側プロセッサを示すコードを持ったデバイスのＢＵＳＹフラグを、セットする。これにより、受信側プロセッサでは　送信側プロセッサに割り当てられたデバイスが、インターラプト要求を発生する。受信側プロセッサに、受け付けられて通信要求転送は終了する。ＢＵＳＹフラグのクリヤはインターラプト処理中に受信側で行なわれる。フラグセットの完了、通信要求転送の終了は、送信側プロセッサのフラグセンス命令で、確認される。受信側プロセッサはどのプロセッサからの通信要求であるかをデバイスコードから認識できる。このＩＣＤで通信できる内容は、割り込み要求だけであるので通信メッセージは、共有メモリ内にあらかじめ書き込んでおく必要がある。

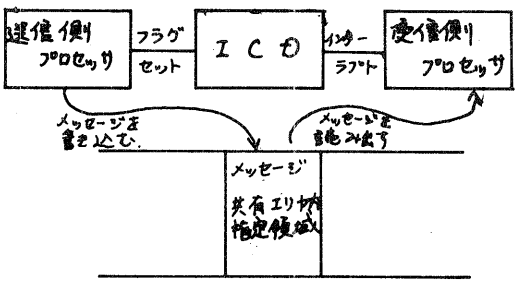

図 10. プロセッサ間通信方法

2.4 市販ミニコンによる実現

パイロットシステムの構成要素として、日本ミニコン社製NOVA0Sを想定し、これに本システムを実現するうえで最少限必要な改造乃至機能の追加を行うことにした。その大要はつぎの通りである。

2.4.1 CPU CLK AUTO STOP 機能

CPUを停止させる機能は、CC、CHECK等のMMUが動作するために必要な時間を得ること。また、メモリアクセス要求が常に、遅延なく受け付けられるとは限らないので、持ち時間を得るために付加した。

アドレスの転送を認識すると、フラグ（後述のメモリサイクル制御でも使用される）を、セットする。このフラグを受けて、アドレスが確定するとCPUCLK停止要求を行い、実行中サイクルの終了と同時にクロックを停止する。この制御端子はプロセッサユニットに 入力ピンが設けられているので、信号を与えるだけでよい。

2.4.2 メモリサイクル制御

2.4.1で、述べたCPUCLK停止機能では、NOVA0Sの場合メモリサイクルの直前では、CPU動作の停止が出来ない。これはメモリサイクルの直前のサイクル中に、アドレスの転送と、実行指令が行なわれているためで、メモリサイクル直前でCPUを停止させるためには、さらに、メモリサイクル制御機能が必要となる。

この機能は、メモリサイクル直前でCPU動作を停止させるために、2.4.1で述べたフラグを用いて、メモリサイクル実行を禁止して、プロセッサを停止させる。また再スタート時には、メモリサイクルの実行を指示する。

この機能を、付加するためには、プロセッサユニットの改造を必要とする。

2.4.3 バイパスレジスター

I/O命令において、ダイレクトチャンネルのデータ転送を、行なう動作では、メモリボード内にあるメモリバッファを中継レジスターとして、CPU－I/Oバス間のデータ転送を行なうが、メモリボードの拘束時間が長くなり、メモリ利用効率が低下する。またMMUはメモリのアクセスが終了すると、ただちに、メモリブロックの切り放しを行うため、転送用バッファを指定できなくなる。よって、MU内で中継を行うバッファレジスターを、付加した。

2.4.4 その他の機能

2.4章で、述べてきた機能は、NOVA0Sを用いるため、特に必要となった機能である。

これらの他に、MMUには付加された機能としては、保護機能やシステムの信頼性を増し、効率よく、使い易くする機能等がある。それらの機能とは

1. Runaway Defer Cycle Protection
2. Write Protection
3. Read Write Protection
4. Adress Invalid check
5. Interval Step Operation
6. Address Step
7. その他

などである。

3. 多重プロセッサの制御と計算の割付け

3.1 多重プロセッサ制御用OSの構造

本システムはC. mmp のように多ユーザに対するTSSサービスを行うというような汎用性を指向しているのではなく、a single program with parallel computable path の並行計算を主な目的としているので、OSとしては 1. 並行計算の開始、終了等の制御を行うためのプロセッサ間通信をサポートする。2. プロセッサの持つアドレス空間のバーチャル化をサポートする。等の機能を持つものを考えている。本システムは一台のマスタプロセッサの制御のもとに3台のスレーブプロセッサがあり、並列計算パス実行以外の様々の処理はすべてマスタプロセッサが行う方式をとっている。マスタプロセッサのOSとしてはNOVAが持っているRDOS (Real time Disc Operating System) のサービス機能を可能なかぎり利用し、プロセッサ間通信を行うのに必要なプロセッサ間通信サポート部とバーチャルアドレスを提供するバーチャルアドレス管理部をRDOSのユーザインタラプトサービス機能を利用して作成しRDOSに付加している。スレーブプロセッサはマスタプロセッサのコントロール下におかれるので、スレーブプロセッサのOSとしては付加されたI/O機器の最小限の制御、マスタプロセッサとの信号の授受、バーチャルアドレスの制御を行う部分から成り立っている。

3.2 並列計算プログラムの構造

プログラムはソフトウェアページングの手法によって実現されているバーチャルメモリ上で記述されており、並列計算パスはDO FOR ALL文を導入することにより記述されている。DO FOR ALL文は DO を FOR ALL I=[m_1, m_2, m_3] の形式をとり、DO FOR ALL 文からあなるラベルで示される文までがDO FOR ALL block となり、並列計算の対象となる。Iは整数変数名でコントロールインデックスとなる。[m_1, m_2, m_3] はFORTRANのDO文に出てくる繰返し指定と同様でIがm_1, m_1+m_3, m_1+2m_3, … とm_2を越えない最大値までの値をとることを示している。DO FOR ALL block 内では別の DO FOR ALL 文が存在することを許していないし、DO FOR ALL block の途中に外からジャンプして入ってくることは許されない。DO FOR ALL block により生成された computation 群は順次プロセッサユニットによって並行計算が行われる。DO FOR ALL block 内で定義された変数は private 変数として取扱われ、private 変数は実行時に必要となるたびに生成され、パスの実行が終了すると delete される (図.11)。一例として、ガウス消去法の並列計算用プログラムを示す。アレイ, A(N, N+1) に方程式は読み込まれているとする (Nは方程式の元数)。プログラムの中核部のみを示すと

```
      PROGRAM GAUSS
      N1 = N-1
      DO  100  I = 1. N1
      DO  100  FOR ALL (J) / (I+1, N, 1)
      Q = A(-I, J) / A(I, I)
      I1 = I-1
      N2 = N+1
      DO  40  K = I1, N2
 40   A(J, K) = A(J, K) - Q*A(I, K)
100   CONTINUE
      A(N, N+1) = A(N, N+1) / A(N, N)
      DO  200  I = N1, 1, -1
      DO  150  FOR ALL (J) / (1, I, 1)
150   A(J, N+1) = A(J, N+1) - A(I+1, N+1)*A(J, I+1)
200   A(I, N+1) = A(I, N+1) / A(I, I)
      STOP
      END
```

となる．

3.3 プログラムの実行　OSはプロセスの概念を基本として設計されており、マスタプロセッサ上には1つのマスタプロセスが各スレーブプロセッサにはそれぞれ1つづつのスレーブプロセスが存在することになる。マスタプロセスはNOVA 0xの持つRDOSの機能にバーチャルアドレスのサポート、スレーブプロセスとの通信のサポート機能を加えたもので図11で示す計算パスの実行と他の並列計算パスのスレーブプロセスへの割付けを行い、fork部、join部における管理的処理を行っている。マスタプロセスを実現している制御プログラムはRDOSの諸プログラム、バーチャルアドレス空間をソフトウェアページングで実現するためのバーチャルアドレス空間サポートプログラム、スレーブプロセスへの並列計算パスの割付け等を行うプロセス間通信制御プログラムから構成されている。スレーブプロセスは、並列計算パスの実行とアドレス空間のバーチャル化のサポート、マスタプロセッサとの通信を担当している。スレーブプロセスはユーザプログラム中の並列計算パスを実行するだけのため、制御プログラムは簡単な割込処理プログラム、バーチャルアドレス空間を実現するためのサポートプログラム、マスタプロセスとの通信を行うプロセス間通信プログラムから構成されることになる。

4. 結言　システムの開発はハードウェアの設計製作をソフトウェア作製に先行して進めており、現在ハードウェアの実装設計とソフトウェアの概要設計を行っている状態で、その詳細については未定の点も残されている。今後は、ハードウェアシステム及びソフトウェアシステムの完成を目指すとともに、代表的なベンチマークプログラムを実行することにより、システムの効率についてのデータを収集しそれをもとにして、より本格的(大規模)な多重プロセッサシステムの可能性について検討を進めていくことを予定している。

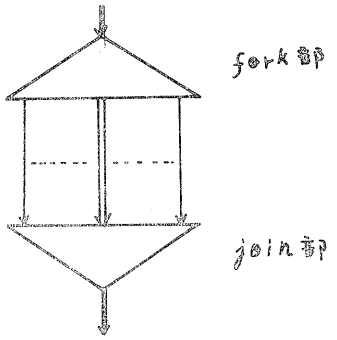

図 11. Do For All block の構造

参考文献

1 金田悠紀夫；"並列処理システムの動向" 電総研調査報告 第174号 昭和48年2月
2 C.G.Bell, A.Newell；" CMMP : The CMU Multiprocessor computer requirements and overview of the initial design", CMU-CS-72-112.
3 村田健郎，堀越清視；"対称帯行列を三重対角化するための新アルゴリズム" 情報処理 Vol.16, No.2, 昭和50年2月.
4 金田悠紀夫；"並列処理システムによる連立一次方程式と楕円形偏微分方程式の数値計算法" 情報処理 Vol.16, No.2, 昭和50年2月.
5 日本ミニコンピュータ(株) "NOVA 02" マニュアル類