ハイレベル・データフロー形並行処理計算機の検討

A CONSIDERATION OF HIGH-LEVEL DATA FLOW MULTIPROCESSORS

阿江 忠　高橋 浩一　松本 健治　相原 玲二　天満 尚二
Tadashi Ae　Koichi Takahashi　Kenji Matsumoto　Reiji Aibara　Shoji Tenma
広島大学工学部
Faculty of Engneering, Hiroshima University

1. まえがき

マルチプロセッサシステムの歴史は古く，ILLIAC Ⅳ，PEPE，Cmmp，Cm*，QA-1など数多く列挙できる。これらのマルチプロセッサシステムの目的は並列処理による高速化にあるが、処理速度はプロセッサ台数の意外に小さいところで飽和してしまう傾向にある。汎用性を目的としたシステムではなく、専用機として使用するコンピュータならば、マルチプロセッサシステムも有望であり、分野によっては大形コンピュータなみの速度をもつ低価格のマルチプロセッサシステムが実現可能なことが実証されつつある。さらに、計算機構からただちにアーキテクチャが導かれるような分野ではこのようなシステムの試作がすでに行なわれている[1][2]。しかし、他方いくつかの応用分野を対象とするときに、アーキテクチャが必ずしも明確に定まるとは限らない。単純なアーキテクチャを考えると、いわゆる"フォン・ノイマン・ボトルネック"のためにオーバーヘッドは減少しない。ところで、非ノイマン形計算機としてデータフローコンピュータが提案されている[3]。しかしながら、一連のデータフローコンピュータは実現の段階では種々の問題点をかかえている。データフロー言語[4]の提案からもわかるように、データフローコンピュータの発想はトップダウン的なものであり、現在の技術でデータフローコンピュータのコストパフォーマンスを考えるのは早計であろう。一方、先に述べたマルチプロセッサの発達を考えたときデータフロー的に働くマルチプロセッサ，いいかえれば並列処理計算機への進展は自然なものといえよう。

本稿は、このようなボトムアップ的に発達してきたマルチプロセッサシステムのデータフロー化の可能性について検討したものである。使用目的は画像表示システムを想定しているが、同様の動作を行なうシステムでは本稿と同様に考えることができよう。

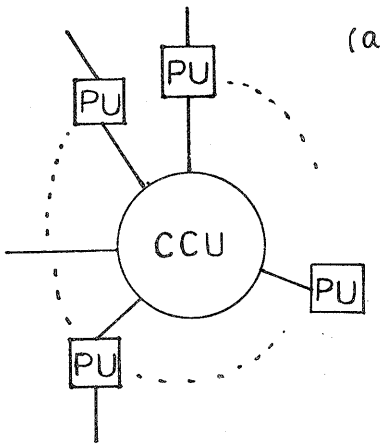
2. 並列処理マルチプロセッサシステムの分類

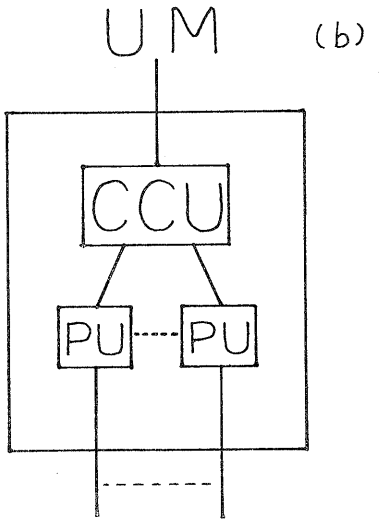
従来から、Flynnの分類すなわち Single / Multiple Instruction Single / Multiple Data Stream を用いて、コンピュータの動作を表現することが多い。しかしながら、マルチプロセッサシステムでは、SIMDでも細かくみるとMIMD的な動作ともみなせるような場合がある。ことに、Instructionがsingleなのかmultipleなのか不明確なことも多い。

一応、Flynnの分類にしたがうとすれば、本稿で議論するのは、MIMD形のシステムであり、並列処理（parallel processing）というより並行処理（concurrent processing）という呼び方がなされているシステムを対象とする。

我々はMIMD形システムの種々の構成法を議論するため、一つのモデルを導入する。

一般にマルチプロセッサシステムは図1.(a)





UNIT OF MULTIPROCESSORS

CCU : COMMUNICATION AND CONTROL UNIT
PU : PROCESSOR UNIT

図1 マルチプロセッサ
(プロセッサ集合体の1つの単位)

- のようにPU (Processor Unit) の集合をCCU (Communication and Control Unit)[†]で結んだものとして表せる。一般に、このシステムの規模は任意であるが、ある適切な一つの規模を UM (Unit of Multiprocessors) として以降扱うことにする。UMは、複数個存在して相互に結合され拡張しうることも想定しておりCCU, PU ともに外に向った結合も可能とする。

これらの準備のもとに、MIMD形マルチプロセッサを細分しよう。我々は、表1に示すようにCCUとPUのそれぞれの機能を高、低にわけることで、4つのタイプに細分する。

CCU \ PU	低	高
低	I	II
高	III	IV

表1. MIMD形マルチプロセッサシステムの細分

タイプ I

CCU, PUとも低機能の場合であり、CCUの機能は低いから特殊な場合はともかく、専用機であっても目的ごとにシステムを設計するのは現状ではかなりむずかしい。低機能の単位を多数個あつめ、かつ、機能の低いCCUにより結合して高機能なシステムをつくる(あるいはつくれるか否かを判定する)問題については、今後、さらに検討が必要である[††]。

タイプ II

CCUは低機能、PUは高機能の場合で、いわゆる単一目的マルチプロセッサに相当する。むろん、このようなタイプも有用であるが、データフローコンピュータを考察する上では、一応除外して考える。

---

†) PUが数多いときはCCUを Communication and Control Network と呼ぶ方が適切であるが、一つの単位としてはこのように呼んでおく。

††) 理論的には、セルラーオートマタ(PUは組合せ回路)→オートマトン複合体あるいはポリオートマタ(いずれもPUは順序回路)、として扱われているものに対応しよう。

タイプ Ⅲ

PU は低機能で、CCU は高機能の場合であり、インストラクションレベルでのデータフローコンピュータとしては、Rumbaugh の machine[3] がその代表的なものである。これに代表される MIT の machine はトップダウン的アプローチとしては興味深いものであり、Dennis の言うように VLSI 技術とともに、やがて開花するのかも知れない[5]。しかし、現状では次のタイプⅣよりもまだ多くの問題を有していると言わざるをえない。

タイプ Ⅳ

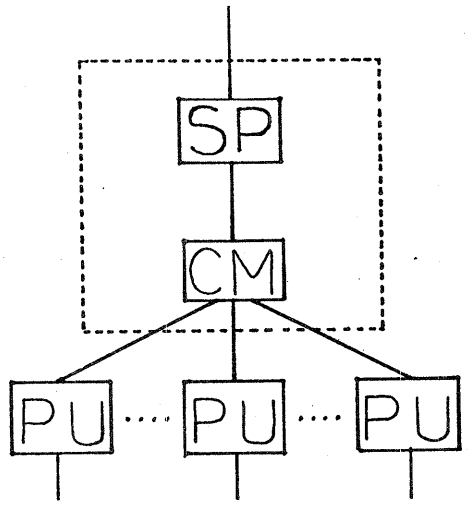
PU, CCU とも高機能の場合であり、タスクあるいはプロセジャーレベルでの並列性をもつデータフローコンピュータとなりうる。

ところで、マルチプロセッサシステムの歴史は古いけれども、PU を大量に使うシステムが実現できるようになったのは、マイクロプロセッサが出来てからである。PU すなわちマイクロプロセッサの数としては 100 以上の個数を用い、(MIPSは評価として必ずしも適当ではないが) 10 MIPS程度の速さを目的としたマルチプロセッサシステムの実現をここでは対象とする。このとき、タイプⅡと呼べるシステムには、すでに、全部または一部が完成しているものがある[1][2]。いずれも、偏微分方程式の計算を目的としているが、タイプⅡで専用化しうる対象は数多いといえようか？

むしろもう少し柔軟性を要求するシステムの方が多いのではないか…という立場で我々は研究を進めており、CCU にもある程度の機能をもたせたタイプⅣを考察の対象とする。(タイプⅣはタイプⅡを含むという見方も成立つ。)

3. マルチプロセッサにおけるボトルネックについて

前節で述べたタイプⅣをマルチプロセッサによりつくる典型的な方法は、図2のように、共有メモリ (CM と略す) を介してプロセッサ間相互に通信をさせる方法でありスーパバイザプロセッサ (SP と略す) が制御する。このような CM を介して PU を結合するＭＩＭＤ形アーキテクチャの問題点は、CM をアクセスするプロセッサ (PU 及び SP) 間の競合がボトルネックになることである。



SP : SUPERVISOR PROCESSOR
CM : COMMON MEMORY
PU : PROCESSOR UNIT

図2 タイプⅣを実現するマルチプロセッサシステム

3－1 静的な考察

たとえば、図3のように PU_i と PU_j がともに CM をアクセスしたいときでも、(a) のように PU_i にそのアクセス権を与える[+]。そして、PU_i に CM を使用させているときはもう一方の PU_j は wait 状態にあり、PU_i が CM の使用をやめて (release して) はじめて、(b) のように PU_j は CM を使用できる。もっと多くのプロセッサ

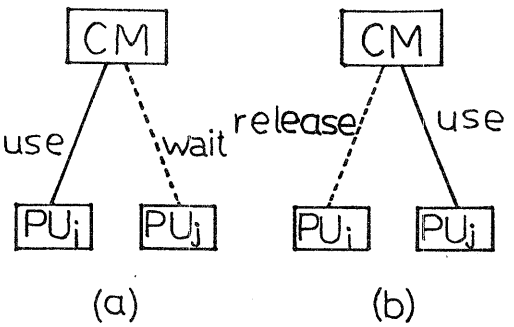


図3 プロセッサ競合の例

+) どのような policy によりアクセス権を与えるかはシステムによる。例えば、先着順などが多く用いられる。

が競合するときはこの拡張であり、アクセス権はどれか1つだけ有し、残りはすべてwait状態になる。つまり、MIMD形アーキテクチャでは、このようにCMを時間的にshareせざるをえない。このshareの方法としては、

(1) R-A方式　PUがCMをアクセスするデータ長[+]ごとにshareする方法[++]。

(2) A-W方式　PUがCMをアクセスするデータ長とは無関係にマシンサイクルごとにshareする方法。

の2方法が考えられる。(個々の名称や詳しい説明については付録参照。)それぞれCMを専有している時間は図4，5のように示される。

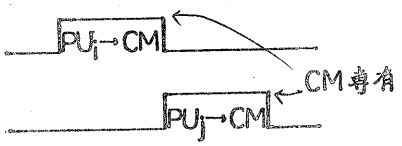


図4 R-A方式

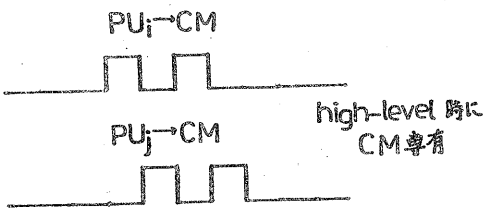


図5 A-W方式

直観的にわかるように、A-W方式は競合に関してR-A方式を改善した方式である。それがどの程度かを各PU上のメモリ(private memory)からCMへの平均転送時間について具体的に計算したものを付録に示す。いずれもプロセッサはIntel 8080Aであり、すべてのプロセッサが競合したワーストケースを扱っている。A-W方式の改善は,プロセッサ台数が10台以上では,著るしい。なお，A-W方式はDMA方式と比べても倍程度であり、ハードウェアコストは格段に小さい。さらに、DMA方式ではランダムアクセスができないという欠点はあるが、A-W方式では連続するデータでなくても、同様の転送速度でよい。さらにもう一つA-W方式の特長は無手順でCMをアクセスできるから、ソフトウェアの負担が非常に小さい。PUの処理時間の長い場合、A-W方式とR-A方式の違いは大差なくなる。一方、この両方式のハードウェアコストはほぼ同等とみなせるから、ソフトウェアのより簡単なA-W方式の方がコスト的にも優れている。

これらの理由からCMをshareする方式としてはA-W方式が適切な方式であると思われる。

以下では、A-W方式を採用するものとして話をすすめる。

3－2　動的な考察

以上は、最悪時の平均転送時間を評価対象としたもので、静的な状態での考察にすぎない。一方、システムを実際の動作における評価，いわば動的な評価を行なうには、通常、動的なパラメータ変化を統計的に把握し、マクロな考察を行なう方法がよく行なわれる[8]。

しかしながら、データフローコンピュータが汎用コンピュータとしてつくられるなら，ともかく、現状では何か専用機的な方向を模索するのが妥当であろう。そこで我々は、一応MIMD的な目的を規定し、具体的な手続きを走らせて、動的な評価を求める立場をとる。我々は画像表示という応用例をとりあげる。(静止画像のみならず動画をも含む)画像表示がMIMD形の動作が適していることを述べるために、図6の例をあげる。

図6では、画をセグメントにわけて、それぞれに手続きを適用して描いている。画を動かす場合には、さらに、いくつかの手続きを適用する。

+)パケット長と考えてよい。

++)ハンドシェイク方式の1つである。

+++)後述するように、SP 1台とPU 4台からなるpilotmachine AKOVSTではA-W方式が採用されている。A-W方式を実際に動かした報告としては2台のプロセッサの場合が文献(6)，(7)あたりにみられるが、我々のように5台の場合[13]は特に報告としては見当らないようである。

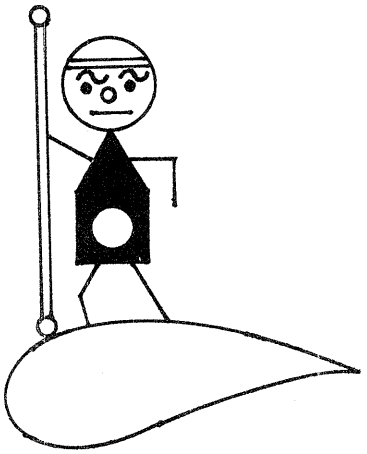


図6　Line, Circle, Paint, Smooth という Procedure で描かれる画の例

画をディスプレイ画面に描く場合、一般に非常に時間のかかるものであることは、よく知られている。

データ→画データ(picture data)→表示(display)の関係は、空間的，ならびに、時間的に図7のような関係をもつ。

並列性は時間軸に沿った処理と空間的な処理の両方に考えられる[9][10]。簡単のため、一枚の画を描く空間的な並列性を考えてもＭＩＭＤ的な記述が適当である。

データから画データへの変換は手続きによりなされるが、画データにも手続き、たとえば、Paint（多角形内の塗りつぶし）や、Smooth（スムージング）などは適用される。

動的な評価を行なうために実験システムとして AKOVST というマルチプロセッサを用いた。

AKOVST は図8に示されるようなマルチプロ

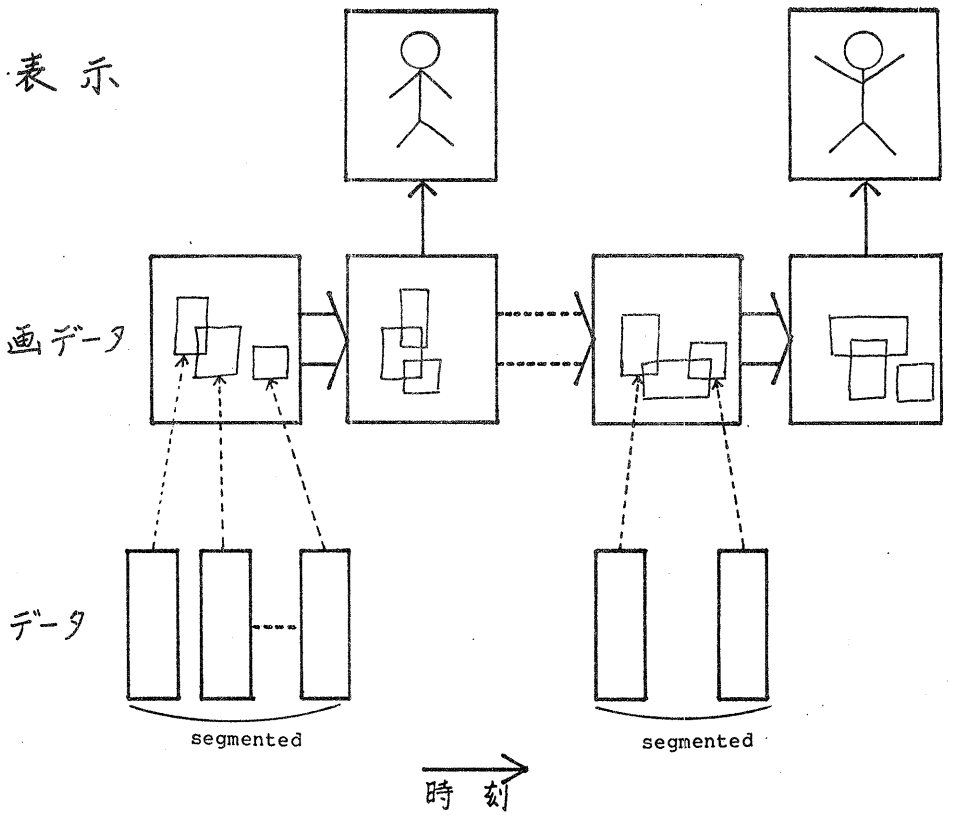


図7. 画像表示システムにおけるデータ、画データの空間的および時間的構造

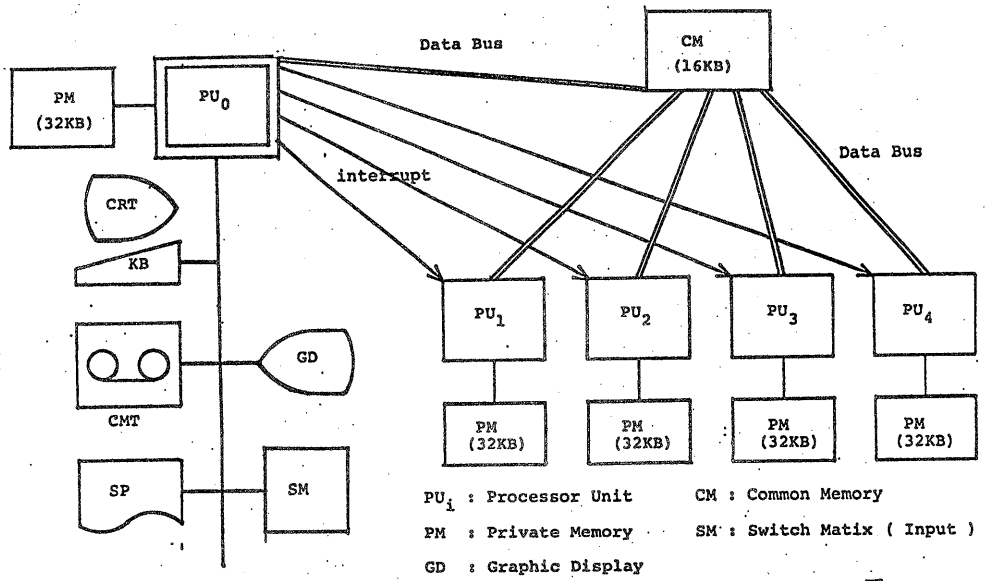


図8。マルチマイクロプロセッサ

セッサで多目的使用を目的に作られている[11]†。

図1に示したUMとしては多少PUの数が少ないが、PU数1，2，3，4から得られるデータをもとに、プロセッサ数を増加させたときに、処理速度の向上割合はどうなるかを試算したのが図9である。ただし、プログラムは単一の仕事をするものと仮定している††。なおrは、

$$r = \frac{\text{PU への転送時間}}{\text{PU での処理時間}}$$

で定義されるパラメータであり、オーバヘッド係数と呼ぶ。rが小さいと、オーバヘッドが小さいので、プロセッサ台数nに対し処理速度は1/nに近づく。rが大きいとこの逆となる。

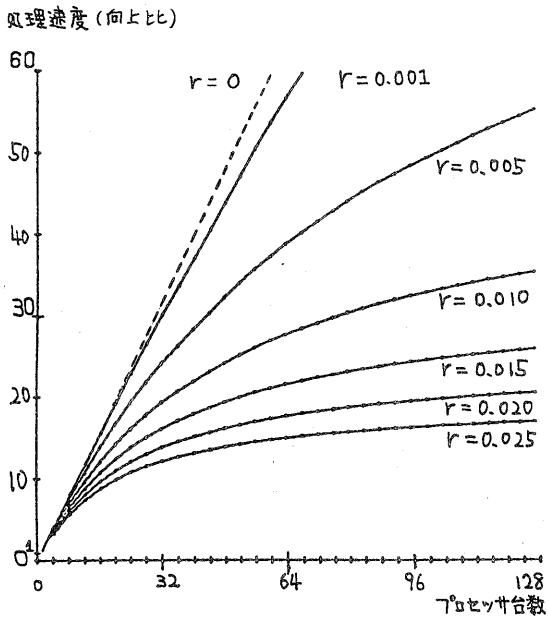


図9 並列処理の効果(A-W方式)

---

†) 1978年に製作開始され
第1年度(1978) ハードウェア
第2年度(1979) 基本ソフトウェア
第3年度(1980) アプリケーション ソフトウェア
の順にそれぞれ開発され、当初の目的[11]は一応達成されている。なお、CMをマルチポート化するためのshare方式は、むろん、A-W方式が使われている。

††) MIMD形のマルチプロセッサで単一の仕事をさせるとSIMD的な動作をするが、ミクロにはプロセッサ間には実行時間のばらつきがあるから、文献(12)でいうSAMDに近い動作をする。

画像表示のためにひんぱんに使われる手続き表2が AKOVST にはインプリメントされているが、Lineを除く手続きはいずれも図9の曲線の r=0.001～0.025 の範囲内にある†。

手続き名	機　能
Line	直線を引く
Circle	円を描く
Paint	塗りつぶす
Zoom	ズーミングを行う
Rotate	回転させる
Move	移動させる
Smooth	Spline または Bezier によりスムージングを行う

表2　試作システムに備えられている手続き

一般に、図形を描くプログラムはいろいろな手続きを用いる。描画の際、最終的に用いられる手続き（図6の例参照）のほかに、種々の計算のための手続きが必要であり、これらの手続きには時間のかかるものも多い。その意味でオーバヘッドへの影響が小さい場合もあるが、手続きをつくるためのアルゴリズム次第で実行速度の異なることも多く、種々のアルゴリズムが適用可能なMIMD形コンピュータが望ましい。

4. データフロー制御機構

手続きレベルでの並列処理を行うデータフローコンピュータでは、各 PU への入出力は図10のようなパケットであらわされる。

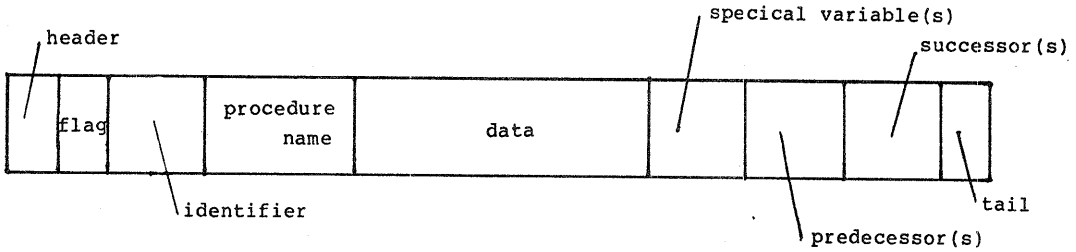


図10　パケットの形式

基本的な考え方は Dennis 流のものと変らないが、手続きレベルであるため具体的には異なってくる。図11が本稿の基本メカニズムであるが、activity store 内のactivity template にしたがい、フラグ(flag)が決まる。フラグがたてば、パケットは operation packet として PU に送られ実行される。procedure name は画像処理に用意される基本手続き (Line, Circle,...etc.) と、ユーザが定義するいくつかの手続き（数値演算などを含む）の1つを指定する††。data は画像データのみならず、通常の数値及び記号データを含む。special variable(s) は、制御用の変数であり、token(s)の働きをする。result packet として戻ってきた

†)たとえば、r=0.022 (Circle), r=0.005 (Paint), r=0.001 (Smooth)が実験結果より求められる。

††)パケット長を短かくするため、個数には制限がある。

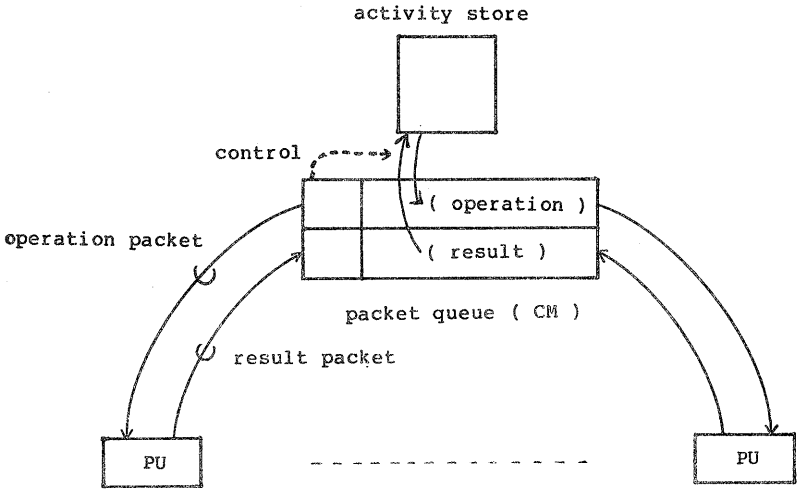


図11. 手続きレベル・データフローの基本機構

とき、token(s)にしたがい、activity store を更新(update)する。predecessor(s)は、指定があるときに書かれる†。successor(s)も同様である。predecessor(s)は、token(s)と同様、activity template (図12) に係わることがある。

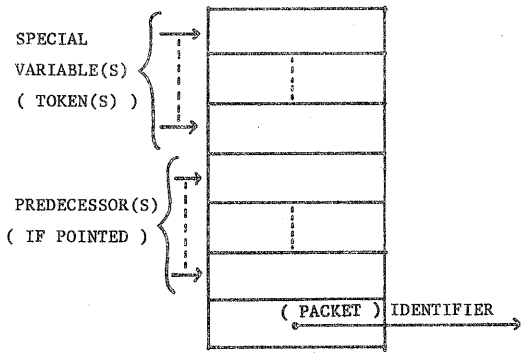


図12 ACTIVITY TEMPLATE の例

図11のパケット処理機構を現状で完全にハードウェア化することはむずかしい。前節までの検討の結果、対象を手続きレベルの並行処理に限ることにより、パケット処理機構におけるオーバヘッドに対する条件は、インストラクションレベルのそれに比べて緩やかなものと考えられる。

したがって

(i)プロセッサによる機構化††

(ii)プロセッサとハードウェアモジュールとの組合せによる機構化

(iii)ハードウェアのみによる機構化

を考えたとき、(ii)の方式が有効であろう。このとき、プロセッサとしてはバイポーラのビットスライス型を用い、activity storeのハードウェア化を検討中である。

PU はハイレベルゆえ、通常の浮動小数点演算機構など、†††画像計算に必要なものはむろん備えるものとするが、それ以外に画像データ演算機構も有すものとする。一般に画像データ演算機構は複雑なものとなるが、データ構造を制限すれば、†††† 十分実現可能である[10]。つまり、2つの画の交わりの検出などの手続きも、ほどほどの時間で行えるようにも配慮している。

---

†)言語上では手続き名に対応する。successor(s)も同じ。ユーザが陽に書くのはsuccessor(s)であるから、コンパイル時に書かれるものとする。

††)現在稼動中の AKOVST は(i)のシミュレーションが可能である。

†††)マイクロプロセッサレベルではAMD社の、APU が有名

††††)たとえば、リスト構造を基本形とし、若干の拡張を備えれば、まずふつうの画には十分であろう。

5. 多数プロセッサへの拡張

図1で示したUMは、せいぜい8～16個程度のPUからなるものと考えている。したがって64～128個程度にPUが増加した場合は、まず、そのアーキテクチャが問題となる。

図2であらわされるUMをプログラム上のネスト構造に対応した木構造をそのまま対応させ拡張していくと図13のようになる。

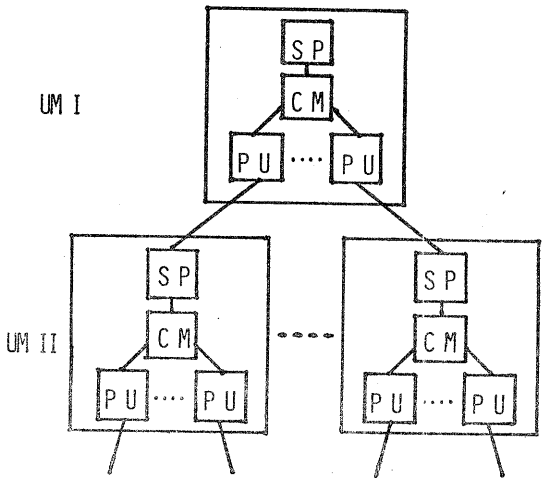


図13 木構造アーキテクチャ

最上位のUM Iを除いて通常UMはプロセッサSPがデータの転送にかかわることになり、3.で論じたオーバヘッドがまた問題になる。したがって、転送時間を大幅に増大させないための工夫がいる。

そこで、我々の考察した方式は共有メモリ(CM)を階層化するもので、ピラミッド共有メモリをなす。2レベルの場合を示したのが図14である†。

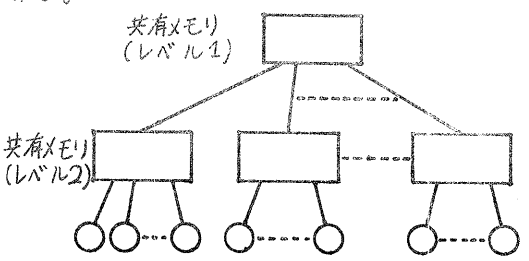


図14. ピラミッド共有メモリ

この方式の原理は、

レベルiで、CMを使用する権利を得たPUに限り、一つ上のレベル(レベルi-1)のCMを要求することができる。

というものである。また、CMを競合したPUのどれに使用権を与えるかは、各々のレベルにおいてハードウェア(3.で述べたA-W方式によるアービタ)により定める。(図15参照)

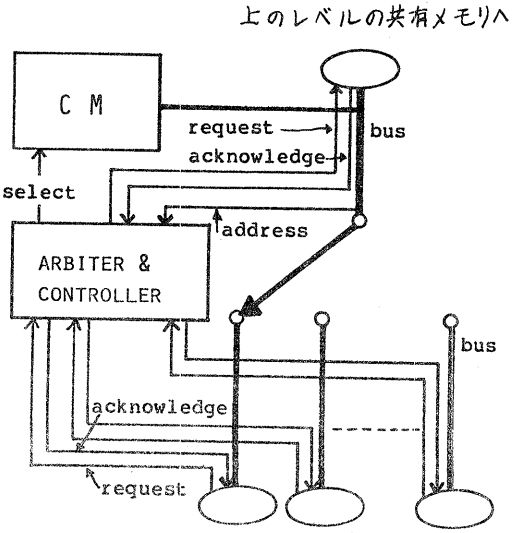


図15. ピラミッド共有メモリを実現する一方式

その結果、図13のようにプロセッサを介することなく、ハードウェアレベルで共有メモリとのやりとりが可能である。むろん、PU間の交信時間は1レベルのときと2レベルのときでは違いが生じるが、図13より、はるかに改善されよう。必要な木のレベル数は1つのUMがいくつのプロセッサを有するかに依存するが、たとえば、PU数が64のとき、UMに含まれるPU

---

†) レベルiの共有メモリはすべて同じ容量とし、マップも同じとする。異なるレベル間の共有メモリにはマップの重なりはない。

数を8として2レベルで構成する方法などがその一例である。拡張性はむろん考慮しておく。なお、共有メモリをレベル化する方法は、どんな結合構造でも適用できるから、アーキテクチャを木構造以外のものとする方法も考えられる。いずれにしても、多数プロセッサを用いるシステムには拡張、または変形に対する柔軟性をもつことが望まれるが、その要求も基本的には満足する方式である。

6. まとめ

手続きレベルの並行処理を行なわせるという制約のもとで、データフローコンピュータを実現する際に生じる問題点を検討してきた。

(i)データフロー機構については、補助的に実現可能なハードウェアモジュールを用いバイポーラ・プロセッサで制御する方式が考えられるが、処理におよぼすオーバヘッドについての評価が、今後、必要である。activity store など連想メモリの実現可能性についても同様である[†]。

(ii)プロセッサを多数個にする場合の問題はピラミッド共有メモリ方式が有望のようにみえる。スーパバイザを含む5台のプロセッサが共有メモリをshareするパイロットシステム(AKOVST)のハードウェアが完成して約2年間種々の実験を行なった。その結果、ピラミッド共有方式へ拡張することはアーキテクチャを固定するという欠点はあるが、成算のある方式といえる。

いずれにしても、理想的なデータフローコンピュータを何年がかりかで製作する方法とは別に、現在すでに存在するマルチプロセッサシステムを除々にデータフローコンピュータに近づけることも一方法と思われる[††]。無理なところは現状の技術でつくっておき、将来その部分を置きかえていく。手続きレベル並行処理計算機はこのようなアプローチに適している。応用として述べた画像表示は一例にすぎず、認識などを含む画像処理[16]への応用も将来考えられるし、手続きレベルであれば特に応用を限定するものではない。

本稿では手続きレベル並行処理計算機のハードウェア面しか述べなかったが、同時に、当面の目的である画像表示データフロー言語の設計にもとりかかっている[15][16]。これらソフトウェア面については別途報告したい。

最後に、日頃より種々の面でお世話になる広島大学工学部第二類(電気系)回路・システム大講座 吉田典可教授、市川忠男教授、前長建治教授をはじめとする教官諸氏、および当計算機工学研究室においてパイロットマシン(AKOVST)の設計、製作に従事された多くの諸氏に御礼申上げる。また、本研究の一部は科学研究費の補助によるものである。

文献

(1) R.Kober.et al : 'SMS 201-A powerful parallel processor with 128 micro-computer',Euro Micro Jour.5 PP.48-52(1979)

(2)星野 力ほか:'科学技術専用並列計算機PACSの開発(I)(II)(III)',電子通信学会総合全国大会 P1400,1401,1402 (昭55-03)

(3) J.E.Rumbaugh : 'A data flow multi-processor',IEEE Trans. Computer , c-26 , PP.138-146 (Feb. 1979)

(4) J.B.Dennis : 'First version of a data flow procedure language',Lecture Notes in Computer Science,19,PP.362-376,Springer-Verlag (1974)

(5) J.B.Dennis : 'The varieties of data flow computers',Proc.1st Intern.Conf on Distributed Computer Systems , PP.430-439 (1979)

(6) B.Loewer : 'The Z-80 in parallel ' , BYTE,3,7,PP.60-63,174 (July 1978)

(7)星野 力:'偏微分方程式解析のためのマイクロプロセッサ複合体',情報処理,20,11,PP.974-982(1979-11)

(8)池原 悟:'マルチプロセッサ方式における共有メモリアクセス競合の解析',信学論(D),J63-D,4,PP.334-341(昭55-04)

(9) T.Ae and K.Takahashi : 'Picture data processing in small parallel computer' , Proc.IEEE 2nd Workshop on Picture Data Description and Management , PP.266-271, Asilomar (AUG. 1980)

(10) T.Ae,K.Takahashi,H.Chiba and T.Ito : 'Computer graphics in multiple micro-processor system',EUROGRAPHICS 80 edit by C.E.Vandoni,PP.281-288 , North-Holland Pub.Co. (1980)

†)過去にアレイメモリを試作した経験から、効果とハードウェアコストを考えながら容量を定める必要があり、難しいところである。

††)したがって、本来のデータフローコンピュータとは大きな隔たりがあることは否めない。データフローコンピュータを広義に解釈しての話である。

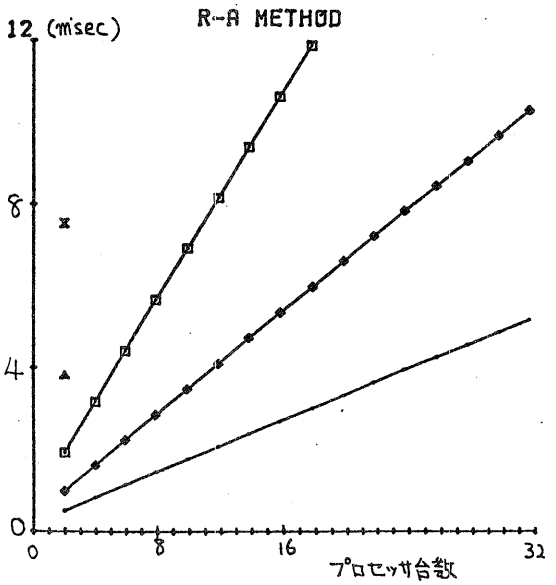
(11)阿江、大崎、Vuong：'小規模マルチプロセッサシステムの一方式'，信学技報 EC 78-35 (1978)
(12)元岡、錢木、喜連川、新岡：'SAMD計算機～A High Level Data Flow Machine～'，情報処理学会計算機アーキテクチャ研資 34-1 (1979-05)
(13)阿江、高橋：'インターリーブ形共有メモリをもつマルチマイクロプロセッサシステムAKOVST'，電子通信学会部門全国大会 381 (昭54-10)
(14)市川、坂村、諸限、相磯：'連想プロセッサ ARES'，信学論(D) J61-D，10，PP.743-750 (1978-10)
(15)高橋，阿江：'マルチプロセッサ向き画像処理言語の提案'，電気四学会中国支部連大 72312 (昭55-11)
(16) K.Takahashi : 'An experimental data flow multiprocessor with procedure-level concurrency',(prepared for Master Thesis, Hiroshima University) .
(17)阿江、高橋、千葉、松本：'マルチプロセッサによるグラフィックターミナルの高速化について'，信学技報 IE 80-39 (1980)
(18) T.Ae,K.Takahashi and T.Ito : 'A multiple microprocessor system with mutually diagnosing capability', Proc.International Computer Symposium (Vol I),PP.375-388,Taipei (Dec. 1980)

付　録

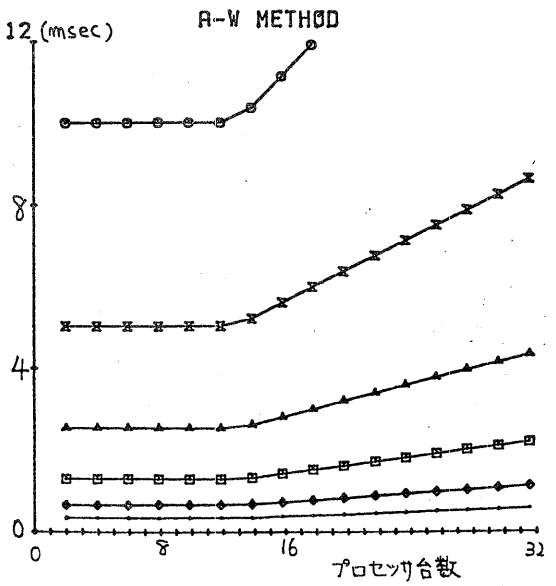
共有メモリ(CM)をshareする3つの方式(R-A, A-W, DMA)それぞれについて、平均転送時間をプロセッサ数に対して計算した図を、以下に掲げる。INTEL 8080A をプロセッサとし、各方式について CM をアクセスする標準的なプログラムを書き、クロック周波数を 2MHz として求めた(詳細は文献(17)参照)。

パラメータは転送バイト数であり、各図とも以下のとおりである。

パラメータ：転送バイト数
- 512 バイト
- 256 バイト
- 128 バイト
- 64 バイト
- 32 バイト
- 16 バイト



図A-1 競合時の平均転送時間(R-A方式)



図A-2 競合時の平均転送時間(A-W方式)

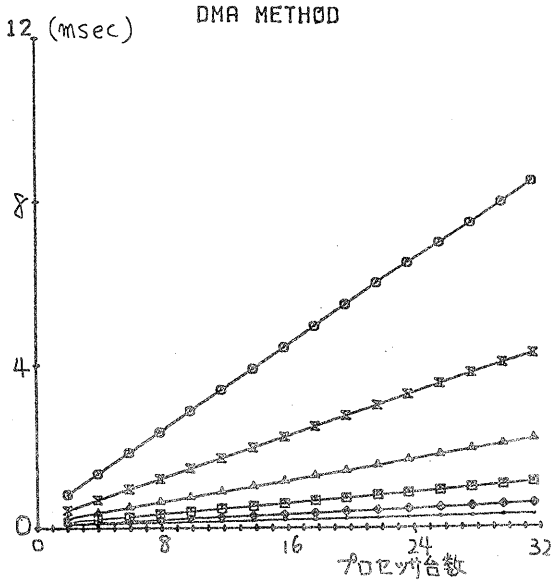


図 A-3 競合時の平均転送時間（DMA）

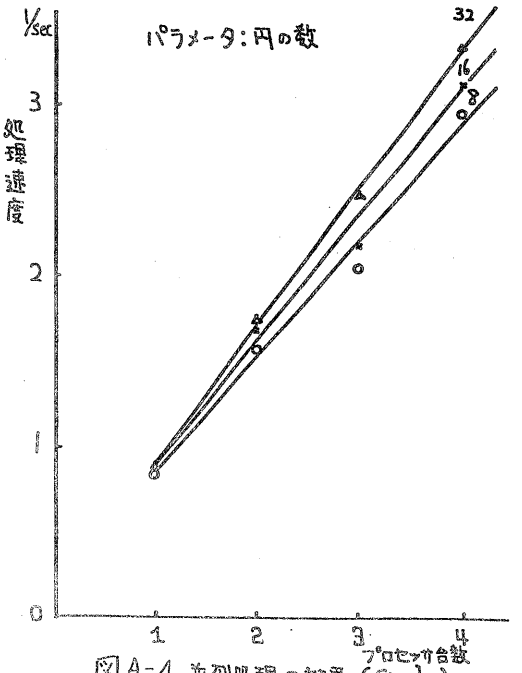


図 A-4 並列処理の効果（Circle）

画像表示に用いるいくつかの手続きの実行速度がプロセッサ数の増加（1，2，3，4）に対して、どのように速くなるかを述べる。直線（Line）は本数が64本程度になると、プロセッサ数に比例するようになる。円を描く（Circle，たとえば写真1）は図A-4のように、さらに比例度はよくなる。

塗りつぶし（Paint），スムージング（Smooth）の場合（たとえば写真2，3）になると、（紙数の都合上、図は省略するが）ほとんど比例した（r=0に近い）直線となる。

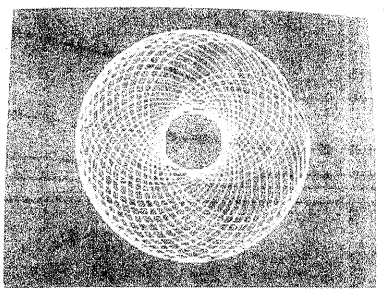


写真 1

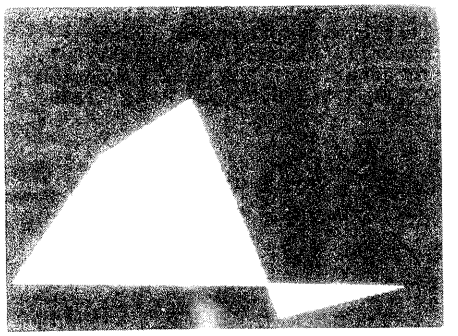


写真 2

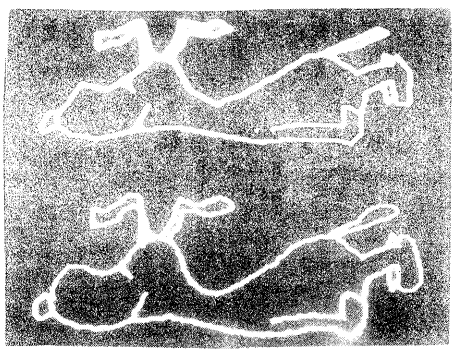


写真 3