μ－パケットネットワークによるメモリ共有型マルチCPUシステム　ASTRAL－I

MEMORY SHARING MULTI CPU SYSTEM BY μ-PACKET NETWORK　ASTRAL-I

松原康夫　唐沢博　米井章憲
Yasuo MATSUBARA　Hiroshi KARASAWA　Akinori YONEI
樋川哲也　三好光照　高澤嘉光
Tetsuya HIKAWA　Mitsuteru MIYOSHI　Yoshimitsu TAKASAWA
山梨大学工学部計算機科学科
Computer Science Department Yamanashi University

1、はじめに

ASTRALは、ASYNCHRONOUS TRANSFER ARRAY LOGICの略であり、情報を非同期に伝播させて処理することを意味する。本システムは、ミクロなレベルでパケットの蓄積交換を行うものであり、その制御のために、ハードウェアペトリネットを使用する。

マイクロコンピュータが普及して、安価に手に入れることができるようになったため、マイクロコンピュータを多数接続して高性能のシステムを構成することが現実的となってきた。すでに100台以上のマイクロコンピュータを接続して動いているシステムが存在し、その有効性が示されている。しかし、特殊な問題に対する性能はともかく、汎用性に関しては従来の大型コンピュータシステムに立ちおちできない。特に、速度が問題で処理そのものは同じことの繰り返しであるような場合だけでなく、より高度で多様な処理内容への発展が望まれる。

本論文で述べる研究は、最初から完結したシステムを作ることよりは、マルチCPUシステムや並列処理に関する、種々の問題点を検討することに主眼を置いた。今回は、5台のCPU(Z-80)と5つのメモリモジュール、全体をコントロールする1－ボードコンピュータから成る小規模なシステムを作成するが、最終的な目標としては、100台以上の大規模なシステムを念頭に置く。又、今回はハードウェアを作成することに重点を置くので、ソフトウェアについては後日発表する。

マルチCPUシステムには、CPU間の結合度合が、疎なものから密なものまでいろいろに分類することができる。本システムは、最も密な場合である、主記憶共有型である。この形態を選んだのは、2つの理由からである。1つは、今後ソフトウェアを作成する上で、もっとも柔軟性に富み、多様な形態の処理を実験をでき、もっとも並列処理らしいことができることである。もう1つはこの形態が望まれているにもかかわらず、ソフトウェア及びハードウェア上、特に後者において、困難な点が多く、性能とコストがつり合わない。そこで、これらの問題点を克服することを、研究目標とするためである。

特に今回目標としたのは、メモリインタフェースの問題である。メモリインターフェースは、コストが、CPUの数が増えても、急激にコストが高むことがないこと、それ自身競合を起さないこと、システムの拡張や変更に耐える柔軟性を持つこと等のいくつかの条件を充たす必要がある。これらの点を考慮して、メモリインタフェースとして、μ－パケットネットワークを採用することにした。

並列処理一般から言えば、SIMD方式は、ある範囲の応用に対して、重要である。しかし、本研究では、より広範囲の問題に対して適要可能で、多様な処理のできる、MIMD方式を採用する。実際、データベースマシンには、MIMD方式が望しいことが言われている。

パケットネットワークは、情報を非同期に伝播させるが、これを制御するために、ハードウェアペトリネットを用いている。MIMD方式では、各CPUが同時に、全く異る処理を行っており、そのタイミングはまちまちである。そのため、全体を1つのクロックで動作させることは、最も遅い部分にシステム全体を合わせることになり、能率の低化の可能性がある。現在、一応動くことを目的としているので伝播速度は低い値に抑えてある。この方法の可否の判断は今後に待たれる。

2. メモリインタフェース

主記憶共有型のマルチCPUシステムでは、N個のCPUとM個のメモリモジュールを接続するハードウェアが必要となる。(一般にMはNと同程度) これをメモリインタフェースという。

マルチポート方式、共有バス方式、プロセッサ専用バス又はスイッチマトリクス方式が良く使われている。この他に、リング上のデータが同期して移動するベルトコンベア方式も考えられている。マルチポート方式は、予じめメモリにポートを用意しておくので拡張性がなく、大規模なシステムには向かない。共有バス方式はコストが小さいことから良く使われるが、バスの競合が問題となるので、小規模のシステムに限られる。競合を緩和するために複数のバスや階層構成を採ることもある。いずれにしても制御は複雑になる。プロセッサ専用バス又はスイッチマトリクス方式では、CPUとメモリモジュールの各々にバスラインが用意してあるため、バスラインに関する競合は起らない。しかし相互接続がN×M個必要であり、この部分のコストはNの自乗にほぼ比例するため、余り大規模なシステムではコストパフォーマンスの問題がある。これらについては、1つのアクセス信号が、1つのバスラインを占有しているのに対して、ベルトコンベア方式では、アクセス信号はパケットの形にまとまっている。これには接続数が少いという利点はあるが、シフトから次のシフトへのインタバルを、一番遅いメモリのアクセスタイムに合わせなければならず、CPUからメモリへのアクセスは、N+M倍の時間になってしまう。特に、応答時間が極端に異るようなデバイスを接続することはできない。

SIMD型システムのプロセッサ間のデータ転送に関してもいろいろの方法が存在する。対数構造転送、シャッフル交換網、ILLIAC-IVのように、特定の位置関係にあるもの同志でやりとりして伝播するもの等がある。SIMD型では、全体が1つのプログラムで制御されるので、前もって、データの伝達経路を計画することができる。これに対してMIMD型においては、各CPUが同時に異ることを行っているので、その場にならないと、データがどのように転送されるかわからない。

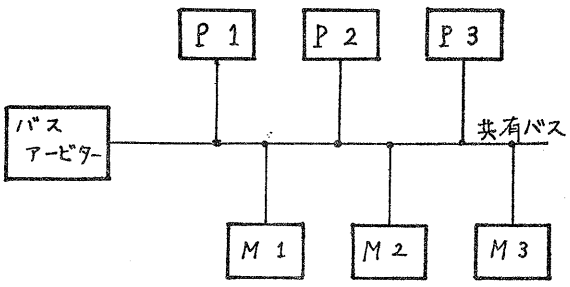


図 1. 共有バス方式

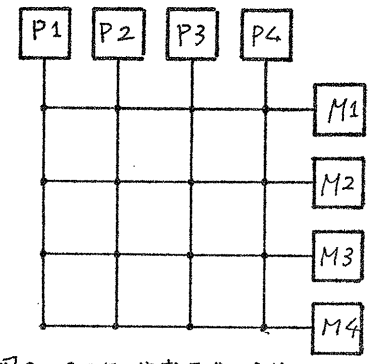
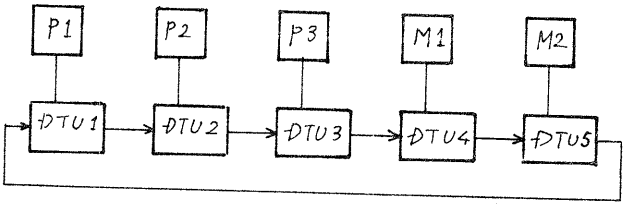


図2. プロセッサ専用バス方式



DTU : Data Transfer Unit

図3. ベルトコンベア方式

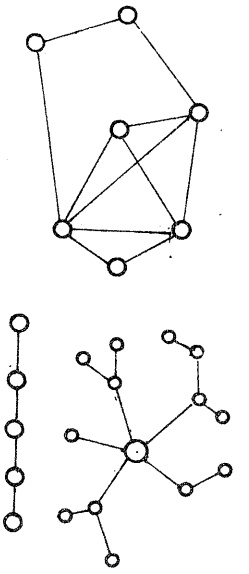


図4. コンピュータネットワークの各種の形態

計算機複合体の中でも最も疎な結合である。コンピュータネットワークにおいては、パケット交換方式が用いられる。パケットには、データの他に行先や発信者等の制御情報を含む。ノードに相当するコンピュータは、パケット単位で蓄積して交換を行う。このため非常に多くのルーチングが可能である。実際のネットワークは、多様な形態があるが、この方式の柔軟性からどのような形態にも適要可能である。先に述べたベルトコンベア方式に較べても、一ケ所の遅いアクセス時間に全体を合わせる必要はない。

密結合のマルチCPUシステムにおいて、パケット交換方式を、メモリインタフェースに採用することが考えられる。このことにより、アクセスの情報が、空間的にまとまっていること、全体が非同期で動作するので異る応答時間に耐える、柔軟なルーチングが可能である、等の利点が考えられる。特に、アクセスの競合に関しては、1ケ所で集中して仲裁するのではなく、ネットワークのノード毎に仲裁することになる。コンピュータネットワークにおいてはビットシリアルが多いが、μ-パケットネットワークでは伝達時間を短くするために、32ビットからなるパケットを、パラレルに転送する。

ハードウェアのコストを節約するために、ネットワークを構成する各ノードには、高々1コのパケットだけを蓄積できるようにする。パケットが次のノードに移ろうとするとき、次のノードに1コ、パケットが存在すれば、空くまで待つことになる。

CPUとメモリモジュールの間の連絡は、共有バス方式や、プロセッサ専用バス方式に較べると間接的である。CPUがメモリに対するアクセス要求を出すと、マスタ・アダプタによって32ビットのパケットに変換されてネットワークを伝播し、目的のスレーブアダプタに到達する。スレーブアダプタは、パケットの内容によって適当なアドレスに読み書きを行い、その結果を元のCPUに返すパケットとして、ネットワークに送出する。これを受け取ったマスタアダプタは、CPUの、WAITを解除して、必要ならデータを返し、メモリアクセスを終了する。

このような間接的な方法をとっているため、伝播の途中で何らかの変換を行うことは容易である。又、1つのアクセスがパケットの形にまとまっているため、ノード間でのパケットの伝播が充分速ければメモリインタフェースとしての伝達速度を満足し、そしてノードが充分たくさんあれば、メモリインタフェースそのものに関する競合は起らない。

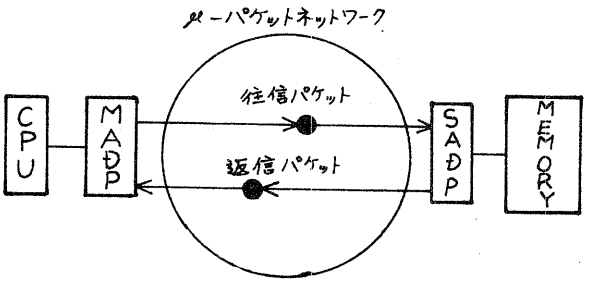


図5 CPUからメモリへのアクセス

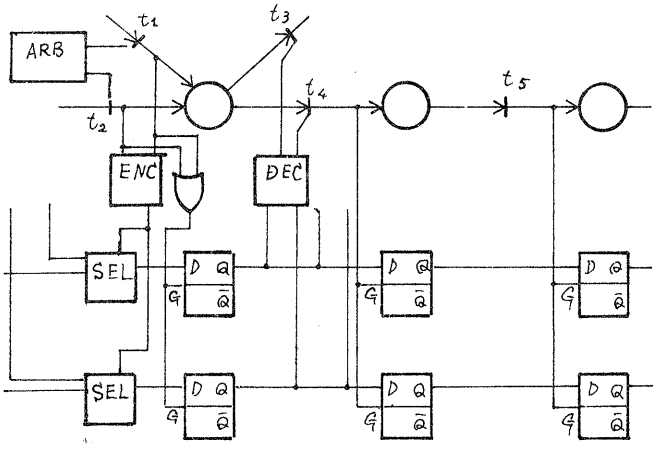
3 μ-パケットネットワークの構成

パケットの伝播を制御するために、ハードウェアペトリネットを使用する。各ノードは、パケットを貯える32ビット分のラッチと1つのプレース、行先を判定するためのデコーダからなる。ノードとノードの間には、パケットの移動を制御するためにトランジッションが設けられ、ラッチのデータ入力にはデータセレクタが入る。

ペトリネットは各種の問題をモデル化してその性質を調べるのに使われている。あるいは、ペトリネットを使って非同期回路の設計をしたり、直接それをハードウェアペトリネットでインプリメントすることもある。本システムで使用するハードウェアペトリネットは、厳密にはペトリネットではないが、ペトリネットで表現し得るものである。ここでは、便宜上ペトリネットと呼ぶ。詳しくは別の機会に発表するが、図8に示すように、アークを往復の2本の信号線でスピードインデペンデントに構成するものである。

μ-パケットネットワークを構成する上で必要なトランジションは、1入力1出力のものである。これが発火可能となるためには、入力プレースにおけるラッチの内容がある条件を満足しなければならない。これを発火条件と呼び、デコーダで判定している。又、プレースがデータラッチを伴うと考えるかわりに、プレース内のトークンが、各種の情報を持ったパケットであると考えてもよい。つまりトークンが属性を持つわけである。

厳密な意味のペトリネットと異る点の1つは、1つのプレースには高々1個のトークンしか入れないことである。もう1つの異る点は、トークンが属性を持ち、このトークンの入ったプレースを入力プレースとするトランジションの発火可能性がコントロールされることである。特に2つのトランジションの共有出力プレースにおいては、2つが同時に発火可能となったときにはどちらか一方だけを発火させることが必要となる。そうしないと、1つのプレースに2つのトークンが同時に入ろうとするからである。この制御のためにアービターが必要となる。アービターには、Req-Ack 方式と Req-Grant 方式があるが、ここでは Req-Grant 方式を使用する。本システムでは3つのトランジションの共有出力プレースが存在するので3-アービターが必要となる。トランジションからはアービターにReqを出し、この中1つのトランジションに対してだけGrantを返す。するとGrantをもらったトランジションだけが発火してトークンが移動する。



ARB：アービター　，ENC：t_1とt_2のどちらが発火するかを示すエンコーダー

SEL：データセレクタ．エンコーダの出力で、データを選ぶ

DEC：データラッチの内容によって、t_3又はt_4のどちらかを発火させるデコーダー

図6 μ-パケットネットワークの構成

このように、1つのプレースに同時に2つ以上のトークンが入ろうとすると、1つだけが入って他を待たせるため、ネットワークの形によっては、デッドロックを起す可能性がある。その例を図9に示す。

メモリインタフェースにおいては2つ以上のCPUからのアクセスが衝突を起すことが

ある。共有バス方式ではバスラインに接続されたCPU間で、バスラインに対するアクセス競合を解決するためのバスアービターを必要とする。このアービターはN-アービターであり、各CPUとの間にReqとGrantの2つの線をそれぞれ配線する必要がある。又、プロセッサ専用バス方式では、各メモリモジュール毎に共有バスを持つのと同じことなので、N-アービターをM個必要とし、Req線とGrant線の対は、N×M本必要である。どちらにしても集中して、競合の仲裁を行う必要がある。スイッチマトリクスについても、どこかで競合の仲裁を行う必要があるので、同様のことが言える。

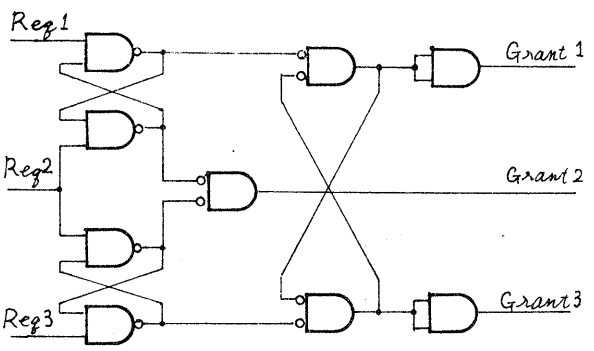


図7 3-アービター

μ-パケットネットワークにおいては、どこか1ケ所で仲裁するのではなく、ハードウェアペトリネットの構成上必要なアービターで仲裁していることになる。つまり、いたるところで衝突が起り、これをその場で仲裁している。

メモリインタフェースにおいては全モジュールに、全く同一の情報を同時に送り出したいことがあり、これをブロードキャストという。今回はこの機能をインプリメントしなかったが、μ-パケットネットワークにおいてこれを行うためには、1つのトークンを必要に応じて2つ以上に増殖させればよい。このためのハードウェアとしては、1入力多出力のトランジッションを使用すればよい。又、ブロードキャストとは逆に、すべてのCPUからの応答がそろったところで、始めて元のCPUに応答を返すことも考えられる。このためには、多入力1出力のトランジッションを使用すればよい。

本システムでは、制御の簡単化のため、リードでもライトでも、CPUからのアクセスには必ずその度毎に応答信号を返しているが、データ転送を高速化するためには、応答信号を省くか、又は一定数の往信パケットに対して1つの返信パケットを返す等の方法が考えられる。チャネル装置のDMA転送に相当することが、CPUをホールド状態にしたり、サイクルスチールを行ったりしなくても実現可能である。

構成上、ハードウェアペトリネットを用いることによって、全体のシステムを非同期又は速度独立に構成することが可能となった。このことは各CPUが独立に動作するMIMD型のシステムにおいては重要なことである。このことによって、システムの最とも遅い部分に全体を合わせる必要がなくなった。どのよう

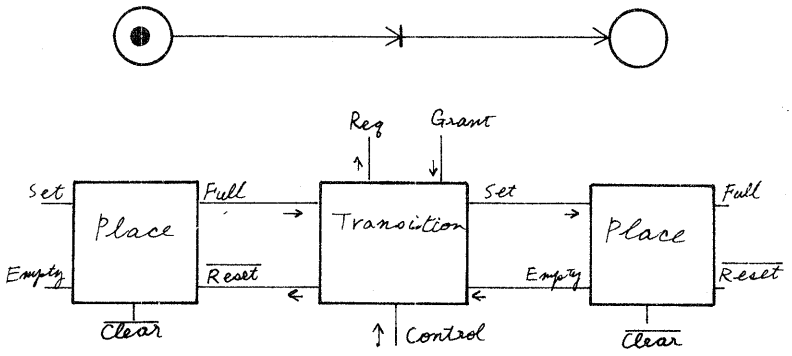


図8 ハードウェアペトリネットの構成

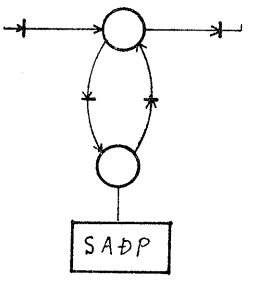


図9 デッドロックを起す形態

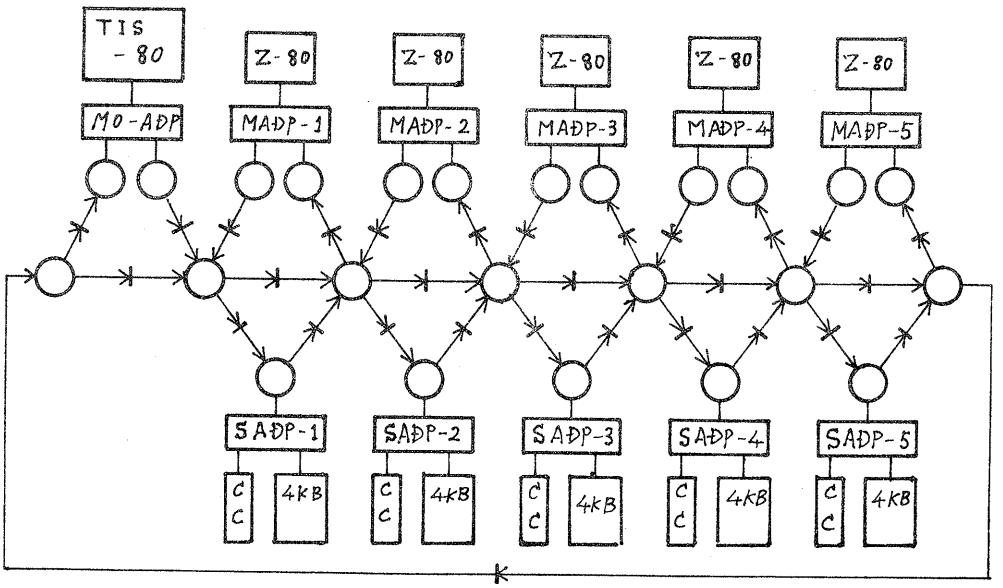
な速度を持つデバイススはCPUでも同様の方法で、ネットワークに接続することが可能である。このことは、ネットワークの持つルーチングの柔軟性と共に、システムの柔軟性、拡張性を与えている。

4、システムの構成

ASTRAL-Iの本体は、5台のZ-80 CPUとその制御回路、5つのメモリモジュール及びこれらを相互に接続する、4ーパケットネットワークから成る。1台のCPUと1つのメモリモジュール、その制御回路を合わせたものを、1つのモジュールとする。これらはモジュール1からモジュール5までである。

又、ASTRAL-Iを総合的に管理する、モジュール0をこれに接続する。ASTRAL本体がデバッグ機能を持たないため、モジュール0を使って、外部から、プログラムの配置やその実行等の管理を行う。モジュール0は、信頼性と可用性、つまりいつでもすぐ使えることが必要なので、市販のワンボードコンピュータ(TIS-80 サンエイ株式会社)を中心に、少し拡張したものを使用する。

一般に、コンピュータにおいては、CPUの初期設定や、状態の読み取り、プログラムのイニシャルロードや実行開始等のために、コンソールパネルが必要である。マルチCPUシステムの場合、各CPU毎に設ける場合と、全体で1つだけ設ける場合が考えられる。コンソールパネルを作成するためには、各種のスイッチや、メモリへ書き込みや読み取りの回路が必要であり、コストが高むこと、又マイクロコンピュータでは、CPUの状態の初期設定や読み取りはハードだけではできないことと、さらに、5台のCPUをマニュアルで操作するのが困難なことから、別々に設けるのは止めた。現在、市販されているマイコンキットでは、CPU内部のレジスタやつうがを、ソフトウェアで、表示、設定を行っている。ASTRALでは、モジュール0から、モジュール1～5のCPUをリセット状態にしたり、割込みをかけることができる。それだけでなく、モジュール0



MADP: Master Adapter, SADP: Slave Adapter.
CC : CPU Controle, MO-ADP : Module 0 Adapter.

図10 ASTRAL-Iシステム構成

は、メモリの役割りを演じることができ、そのアドレスは X'0000' 番地から X'0FFF' 番地までであるので、リセットを解除すると、モジュール0をアクセスする。これによって、モジュール0は、初期設定その他の操作を他のCPUに対して行うことができる。すなわち、モジュール0は、コンソールパネルの役割を持つことになる。

各モジュールには、マスタアダプタとスレーブアダプタが存在する。マスタアダプタは、CPUと、ネットワークを接続し、スレーブアダプタは、ネットワークと、メモリモジュールと制御回路を接続する。現在1つのメモリモジュールは4KBであるが、これは将来、拡張される可能性がある。制御回路には、いくつかのレジスタが存在し、このビットを操作することによって同じモジュールのCPUをリセットしたり、割込みをかけることができる。又、アドレス変換レジスタも、ここに含まれる。他のCPUからも、これらのレジスタに書き込むことができる。

モジュール0は、マスターとしての役割と、スレーブとしての役割の両方を果さなければならないので、これとネットワークを接続するためには特殊なアダプターが必要である。実際には、エミュー80のバスラインに、I/Oポートとして接続し、データラッチに書き込んだり、トークンを送り出す機能と、到着したトークンの、データラッチの内容を読み取り、トークンが存在することを確めたり、トークンを消すような操作を、プログラムで行うことができる。

マスターとしての機能は、CPUの初期設定の他に、各モジュールのメモリにプログラムをイニシャルロードすること、適当なタイミングで各CPUに割込みをかけることである。スレーブとしての機能は X'0000'〜X'0FFF' 番地のメモリアクセスに対して、適当な応答を返してメモリをシミュレートすることである。この2つの機能は、単独で使用されるのではなく、組み合わせて使用される。これによって、多種多様なサービスをモジュール0が行うことができる。

モジュール0は、常にパケットの到着をセンスしていて、到着したパケットの内容に対応して、各種のルーチンが実行され、適当な応答パケットが送り返される。

各種のI/Oがモジュール0に接続される。現時点ではASTRAL本体のデバッグが、モジュール0とこれらのI/Oを使用して行われるが、デバッグが終れば、各CPUは、ルーパケットネットワーク本体に、その他のI/Oを接続することが考えられる。補助記憶として、フロッピーディスクはカセットテープを接続する予定であるが、現在は紙テープ読取り装置が接続されている。会話型の端末としては、キャラクターディスプレイとキーボードが接続されている。

当初は、FACOM U-200 をホストとした、クロスソフトウェアにより、ハードウェアのデバッグと、基本的なソフトウェアの実験を行っていく。

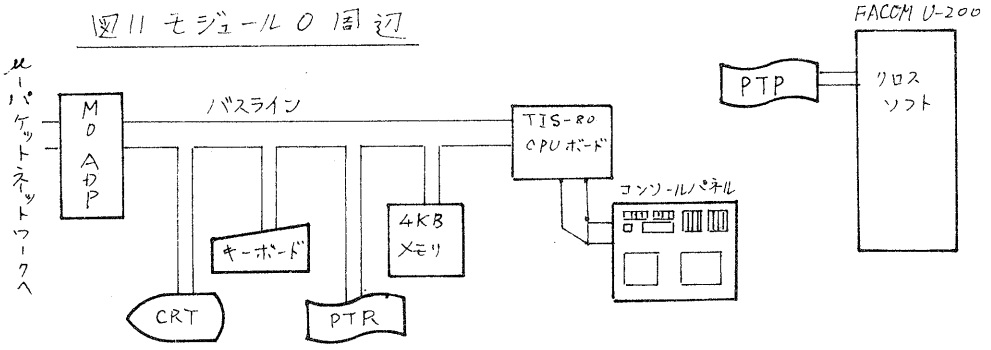
5. ソフトウェアから見た構造

5つのメモリモジュールには、固定した物理アドレスが与えられている。Z-80のアドレスバスは16ビットからなるが、このうち上位4ビットがモジュール識別として使用され、下位の、12ビットは、モジュール内のアドレスとして使用される。Z-80のI/O命令は、アドレスとして8ビットだけを使用する。このうち上位4ビットは、やはり行先モジュール識別として使用される。I/O命令によって作られたパケットは、行先モジュールにおいて、CPUの制御回路で使用される。

メモリアクセス、I/Oアクセスいずれの場合でも、行先モジュール識別の4ビットには、CPUからの直接のアドレスに、CPU制御回路のアドレス変換レジスタの4ビットの内容が加算されたものが、実際のパケットに付加されて送り出される。加算するかしないかは、CPU制御回路のビットで、I/Oやメモリアクセスのそれぞれについて指定できる。アドレス変換機能は、プログラムが、5つのメモリモジュールにどう配置されるかに対処するために設けられた。処理形態にはいろいろな場合が考えられるが、例えば、次のような場合が考えられるだろう。

(1) 5つのメモリモジュールが、1つのプログラムに占められ、5つのCPUは、同一のプログラムを、リエントラントに実行する。

図11 モジュール0周辺



(2) メモリモジュール毎に異るプログラムが置かれ、各CPUは、モジュール内のプログラムを実行する。

(3) その他、2つのプログラムを、モジュール1～2と、3～5に置き、それぞれのCPUで並列に実行する場合等。

マイクロコンピュータの機械語では、相対的なアドレッシングが使えないため、1つのプログラム(実行形式)は必ず決った論理アドレスに置いて実行しなければならない。このため、以上のような種々の処理形態に対処するためには、何らかの手段で、論理アドレスを物理アドレスに変換することが望ましい。本システムの方式は、このための最低限の機能を与えている。

Z-80には3つの割込みモードがある。モード0では、8080Aと同様に、データバス上に任意の命令を送ることによって、適当な割込処理ルーチンに分岐させることができる。モード1では、無条件に、X'0038'番地へのリスタート命令を実行する。モード2では、Iレジスタの内容がアドレスの上位8ビットとして使われ、下位7ビットは外部から与えられ、最下位の1ビットは0としたアドレスに分岐する。本システムでは、種々の形態の実験を行うために、モード2を採用する。しかしハードウェアを節約するために、外部からの7ビットは常に一定とする。

割込モードの指定、Iレジスタの内容、割込みマスク等は、予じめ、ソフトウェアでセッティングしておく必要がある。ところが、これらのプログラムを、モジュール毎のメモリに置くことは、他のプログラムやデータのための領域を圧迫してしまうので望しくない。そこでこれらのセッティングは、モジュール0の責任となる。

モジュール0に何らかのサービスを依頼するときは、X'0××××'番地へのメモリアクセスか、X'0×'番地へのI/Oアクセスを行えばよい。モジュール0では、対応する処理ルーチンを実行し、応答パケットを返す。この場合、モジュール1～5から見たモジュール0は、単に、機能的なメモリ、又はI/Oに過ぎない。

モジュール0の働きを示すために、最初に、プログラムをメモリにロードし、各CPUを初期化して指定アドレスから実行させる場合を考えてみる。

1) システムリセットボタンが押され、各CPUはリセット状態に入り、μ-パケットネットワークもクリアされ、どんなトークンも存在しない状態になる。

2) モジュール0が、補助記憶又は入力装置からプログラムを読み込み、各メモリモジュール毎にプログラムを格納する。

3) 例えば、モジュール1のCPUのリセットを解除する。すると、モジュール1のCPUは、X'0000'番地の命令をFetchするパケットを送出する。

4) モジュール0は、初期化のために必要な命令を、いくつかのパケットに分解し、命令の、Fetchパケットが来るごとに、返信パケットとして返してやる。

5) 最後に、実行開始番地への分岐命令を、命令Fetchパケットに返信パケットとして返してやる。モジュール1のCPUは、指定番地から実行を開始する。

このようにして、モジュール0は、各CPUを制御することができ、任意の番地からプログラムを実行したり、止めたりできる。現時点では、モジュール0は、行先モジュール識別が0であるようなパケットに対してのみ干渉できるだけであるが、さらに権限を拡大して、一定の条件を満たすパケットには何らかの処理を行うようにすることも可能である。

又、モジュール0を介して、他のCPUへ働きかけることもでき、他のCPUを、メモリアクセスの中で呼び出すことや、返信パケットを返すのを遅らせることによってCPU間の同期を取ることなど、興味深い実験が可能である。

1つのCPUには原則として1つのプロセスを対応させるが、CPU間の同期問題は、単にソフトウェアだけでなく、ハードウェアのデバイス等をも含めて解決すべき問題である。

6、μ-パケット

μ-パケットネットワークの中を伝播するμ-パケットについて述べる。これは見方を変えれば、ペトリネットのトークンが属性を持ったものと見ることもできる。通常のコンピュータネットワークにおけるパケットと同様に、行先識別と発信者識別、又行先で伝えるべき情報等を含んでいる。μ-パケットは32ビットから成り、ビットパラレルに伝達される。

ビット0は常に0とする。これは将来、機能や規模を拡大するときのための予備である。

ビット1は、0のときマスターからスレーブへの往信パケットであり、1のときスレーブからマスターへの返信パケットであることを示す。

ビット2～5は、行先のモジュール番号を示す。これはアドレスラインの上位4ビットに、アドレス変換レジスタの内容を加えて物理アドレスに変換したものが現われる。

ビット6は、0のときメモリアクセス、1のときI/Oアクセスであることを示す。

ビット7は、0のときリード、1のときライトであることを示す。

ビット8～19の、12ビットはモジュール内でのアドレスを示す。

ビット20～23の4ビットは、発信者のモジュール番号を入れる。

ビット24～31の8ビットは、データである。

パラレルに伝送し、各ノードも単純なものにしたいので、必要最低限の情報だけを含む。シーケンス制御やバイトカウント等、順序入れかえやパケットの紛失等に対処するための冗長な情報は含まない。このため、μ-パケットネットワークの構成は、パケットの順序が入れかわったり、パケットが失われたりしないようになっている。特にバッファが溢れたときには、パケットを捨てるのではなく、溢れないように、パケットの伝播を待たせている。

上に述べた32ビットの情報は、どの時点でも全てが必要なものではない。CPUが、あるメモリにアクセスしたときに、パケットの情報が、どの時点でどのように、不要となるかを追ってみる。

1) CPUから、メモリアクセスのパケットを送出する。ビット24～31のデータ部分は、書き込みのときには必要であるが、リードのときには不要である。その他のビットについては、全て必要である。

2) パケットが、ネットワークの幹線ノードから、行先のスレーブアダプタに取り入れられる。スレーブアダプタにおいては、ビット2～5の行先モジュール識別、ビット1のマスターとスレーブの区別、ビット0は、全く不要となる。

3) アダプタはメモリーアクセスであることを確認して、モジュール内アドレスにアクセスし、その結果を返信パケットとしてネットワークに送り出す。このときビット0は常に0、ビット1はスレーブからマスタへを意味する1とする。行先モジュール識別には、往信パケットの発信者識別の内容を入れてやる。ビット24～31のデータ部分は、アクセスがライトのときは不要、リードのときは必要である。ライトのとき、CPUに返すべき情報はないが、CPUが次の動作に移るタイミングを作るために、必ずパケットを返信する。又、これ以

外のビット6～23は全て不要である。

4）ネットワークから返信パケットがマスターアダプタに取り込まれる。アクセスがリードのときデータビットが必要。それ以外は全て不要。

以上のように、パケットにおいて必要なビットは、先に行くに従って減少していく。これらの各段階に対応して、ネットワークのノード間における情報線の接続は、不要なビットの配線を省略することができる。本システムでは工数の減少のため必要最小限の配線だけを行っている。その様子を表1にまとめる。しかしモジュール0のアダプタとネットワークの間は、モジュール0が各種のサービスを行えるように、32ビットを全て配線している。

ビット	内容	MADP→幹線	幹線→SADP	SADP→幹線	幹線→MADP
0	通常モード／特殊モード	○	×	○	×
1	マスター→スレーブ／スレーブ→マスター	○	×	○	×
2～5	行先モジュール識別	○	×	○	×
6	メモリアクセス／I/Oアクセス	○	○	×	×
7	リード／ライト	○	○	×	×
8～19	モジュール内アドレス	○	○	×	×
20～23	発信モジュール識別	○	○	×	×
24～31	データ	○	○	○	○

○配線　×配線省略

表1 ll-パケットのビットマップと配線の省略

ビット0は通常は0であるが、1のときには、他のビットと全く異る意味を持たせることができる。これを使う拡張としては、ブロードキャストや、DMA転送、あるいは、存在しないモジュールにアクセスしたときの処理等が考えられる。

7．アダプタ

マスターアダプタは、CPUがメモリまたはI/Oへのアクセス信号を出したとき、ll-パケットを送り出すとともにCPUをWAIT状態にし、返信パケットが返ってきたら、CPUのWAIT状態を解除し、必要な情報を与える機能を持つ。CPUからはアクセス信号が出されて、WAIT状態が解除されるまで続くが、アクセス信号の立ち上りで1つのトークンを作り出すため、特殊なプレース回路が必要となる。又、返信パケットに対しても1つのプレースを用意し、トークンが返ると、CPUのWAIT状態を解除する。CPUのアクセス信号が消えることによって、トークンがクリアされる。

ビット0と1は、0に固定される。

行先モジュール識別に関しては、Z-80のI/O命令では下位8ビットにしかアドレス信号が出ないので、2-wayのデータセレクタを用いる。

ビット6については、Refreshのときにも Memory Requestがでるので、$\overline{\text{Request}}$ とアンドを採って接続する。

ビット7について、RDを接続する。

ビット8～19にはアドレスラインの11～0を接続する。

ビット20～23は、モジュールの識別番号を入れる。デバッグの便宜のためDIPスイッチで、指定している。

ビット24～31はZ-80のデータバスの7～0を接続する。

スレーブアダプタは、到着したパケットによってメモリスロCPU制御レジスタにアクセスする。いずれの場合でも、到着したパケットの発信局識別を、返信パケットの行先モジュール識別に接続する。メモリのアクセスには時間がかかるので、26ビット分のラッチが必要である。I/O指定のときは、直接、目的のラッチに書き込めばよい。

CPU制御レジスタは、X'F'番地とX'E'番地のモジュール内アドレスに存在する。X'F'番地のレジスタは、上位のビット7,6,5が、それぞれリセット、メモリアドレス変換、I/Oアドレス変換を指定する。下位のビット3,2,1,0はアドレス変換レジスタであり、同じモジュールのCPUからパケットが出されるとき、アドレス変換を行うならばアドレスバスの上位4ビットにこれが加えられて行先モジュール識別となる。X'E'番地は割込みレジスタである。これに8ビットのデータを書き込むと、このモジュールのCPUに割込みがかけられる。書き込むとき、この書き込みパケットの発信者識別が、別の4ビットのレジスタに貯えられる。X'E'番地のレジスタは、X'E'番地から読み取ることができる。X'F'番地を読み取ると、上位4ビットは、X'F'番地に書き込んだ内容が、下位4ビットは、割込みレジスタの発信者識別が読み取られる。アドレス変換レジスタは書き込みはできるが、読み取りはできない。このように、8ビットのデータを書き込むことによって、割込みがかけられ、そのデータと発信者を知ることができるので、一種のメッセージバッファとして使うことができる。

以上に述べた制御は1つのプレースとスレーブアダプタによって行われる。I/O指定のときは、すぐに返信パケットが返され、メモリ指定のときにはアクセス時間がかかるので、このプレース内にトークンの滞在する時間が、場合によって異る。

モジュール0については、他のモジュールと全く異るアダプターが置かれる。これは、ワンボードコンピュータから見て、1つのI/Oポートとして見える。これにも、送信プレースと受信プレースがある。両方とも32ビット分のデータラッチを持つ。ワンボードコンピュータから、送信プレースのラッチに書き込みができ、受信プレースのラッチは読み取ることができる。両方のプレースともトークンの有無を調べることができ、送信プレースにはトークンを書き込むこと、受信プレースはトークンをクリアすることが可能である。このため、発信者識別についても自由に設定でき、各種のデバッグを行うことができる。

8. 問題点

本システムの作成において、後に述べる、いくつかの要求を満足することが必要であった。

(1) ハードウェアの構成法、アーキテクチャの研究
(2) システムが、ソフトウェアの各種の実験に使用可能
(3) コストを小さくする
(4) 工数を少くする。

CPUの実行速度を上げるためには、モジュール内のCPUとメモリは直接結合することが考えられる。そうすると工数が増え、インターフェースの統一性を欠くことになるので、ネットワークを介してのみ接続した。又、将来の拡張性のためには、不必要な情報線も配線することが望ましいが、工数を少くするため省略した。

最初の構想では、ネットワークはリング構造でなく、双方向のネットを設ける予定であったが、具体的な設計から、マイクロCPUと4KBのメモリに対して、ネットワークのハードウェアが重み、バランスのとれないことがわかった。ネットワークの構成は、システムの目的に応じた形態を採るべきである。

手順としては、確実な部分からインプリメントし、しだいに拡張するというステップをふむべきであったが、期間が限られていること、ワンボードコンピュータの入手が遅れたことから必ずしもそのステップをふむことができなかった。現在、一応のインプリメントを終え、ワンボードコンピュータ周辺のデバッグを行っている。

μ-パケットネットワークの、パケット伝播速度は、1つのノードから次のノードに移るのに約200nsかかる。これは、回路を、スタンダードのTTLで組んでいるためである。これを、HCMOSシリーズに置きかえることにより、1/5程度の時間で済むものと思われる。しかし、今回は、確実に動作させることを目標とし、又他の回路とのインタフェースの便宜からも、スタンダードのTTLを採用した。

又、ネットワークの構成が、図10に示すように、必ずしもモジュラリティが良くないのは、工数を減らすことと、伝播時間を小さくするためである。理想的には、モジュール間の接続は、1つのアークで済ませるべきである。

今回のインプリメントでは、メモリインタフェースに主眼を置いたが、CPU間の同期に関しても、ハードウェアによるデバイスを付加すべきである。ハードウェアベトリネットは、その目的の非常に有効な手段となる。これは今後の課題である。

実装については、デバッグの都合のため、全体を1つの平面上に構成した。デバッグの終った段階でコンパクトにする予定である。

9. おわりに

本システムは、5台のマイクロCPUを接続する小規模なシステムであるため、実用上はより簡単なメモリインタフェースで構成することができる。しかし、最終的には、大規模なシステムを目標とし、その問題点を研究し解決策を見つけることが、1つの目的である。

実際に100台以上のCPUを接続して実動しているシステムとして、SMS201が挙げられる。このように、単位となるCPUが、マイクロコンピュータのように低機能のものでも、問題によっては、汎用の大型コンピュータよりも高い性能を発揮することができる。しかし、本研究で目標とするのは、より汎用的で、多様な処理を目ざすものである。

明らかに、こういったシステムの性能を決定するのは、CPUやメモリ間を結合する、結合方式である。本システムではこの点に主眼を置いた。通常のコンピュータネットワークにおける、パケット交換方式の持つ、柔軟性その他の利点を、このような密結合のシステムに採用したのが、μ-パケットネットワークである。

本システムと同様の目標と、考え方を持つものとして、Cm*が挙げられる。Cm*においては、3段階の階層構造を持つ結合方式を採っており、ローカルには2段階のバスライン方式を使って、さらにその上では、パケット交換方式を使用する。これに対し、本システムでは、最もローカルなメモリアクセスからパケット交換方式を採用しており、これを可能とするため、独特の回路を用いている。

並列処理システムは、その適要する問題によって、全く異るシステムが望まれる。今後の方針としては、各種の問題に応じた、ソフトウェアをも含め、ハードウェアを実装していく。それとともに、大規模なシステムを構成する上での問題点を追求していく。

大規模なシステムを構成する上では、何らかの階層構造が必要になると思われる。しかし、ローカルに共有バスを用いると、グローバルなアクセスをしている間はローカルなバスを占有することになる。そこで、図12に示すように、ローカルな段階からパケット交換方式を採用することは、インタフェースの統一性からも利点があるものと思われる。又、大規模なシステムを構成する上で重要な点として、記憶空間の問題がある。本システムでは、全メモリを加えても20KBにしか達しないので、問題にならないが、最終的な目標に関しては、この問題を良く考慮する必要がある。

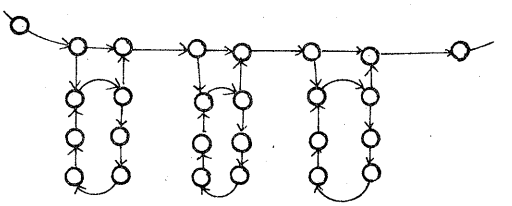


図12. 階層的なネットワークの構成

今回は、ハードウェアをインプリメントする

ことに終始したが、ハードウェアに関しても、問題点が明らかになるのはこれからである。ソフトウェアに関しては、基本的な制御プログラムから積み重ねていく必要がある。

10. 謝辞

有益な御助言をいただいた、慶應義塾大学の、相磯先生に感謝致します。又、常に励まして下さった田中先生に感謝致します。

文献

1) ハロルド・コーリン "ハードウェアとソフトウェアにおける並列処理" 産業図書

2) 加藤満左夫, 笛村憲司共著 "並列処理計算機" オーム社

3) "各種の形態をとるマルチマイクロプロセッサシステムが次々に登場"
NIKKEI ELECTRONICS (1977. 12. 26) pp. 53 - 75

4) 松原 "マルチマイクロCPUシステムの構想"
処理学会マイクロコンピュータとソフトウェアシンポジウム報告集 p.62
(1977. 7. 14 ~ 16)

5) 松原 "ペトリネットのハードウェアによる構成" 処理学会計算機アーキテクチャ研究会資料 27 - 1 (1977. 7. 12)

6) 有澤博・土肥康彦 "データベースマシン向きのデータ処理方式"
信学論 Vol J60 - D No11 (1977) p 921

7) J. L. Peterson " Petri Nets " Comp. Surv. Vol. 9 No 3 (1977)

8) 吉田裕, 西山禎彦 "コンピュータネットワーク用の通信網"
情報処理 Vol 16 No 7 (1975) p. 597

9) K. Spe. Højberg. " An Asynchronous Arbiter Resolves Resource Allocation Conflicts on a Random Priority Basis " Computer Design / August (1977)

10) 元岡達 "コンピュータコンプレックスの展望"
情報処理 Vol 15, No 7 (1974) p. 525.

11) 松原 "新しいメモリインタフェース AS Cone"
S52年度信学会情報部門全国大会講演論文集
S3 - 4 p 439

12) D. Misunas "Petri Nets and Speed Independent Design"
C.A.C.M. Vol 16 No 8 (1973) p. 474

13) R.E. Swartwout "One Method for Designing Speed Independent Logic for a Control".
Proc. of the 2nd Annual Symp. on Switching Circuit Theory and Logical Design. Oct. (1962) p. 94.

14) Tomas Lang "Interconnections Between Processors and Memory Modules Using the Shuffle-Exchange Network"
IEEE. Trans. Comp. Vol. C-25 No 5. (1976) P.496