計算機アーキテクチャ 101−14
（1993. 8. 20）

メモリ結合型マルチプロセッサMC1について

酒居 敬一[†] 藤田 聡[†] 相原 玲二[††] 阿江 忠[†]

[†]広島大学 工学部 [††]広島大学 集積化システム研究センター

本稿では、メモリ結合型マルチプロセッサMC1について述べる。従来のメモリ結合型マルチプロセッサは共有メモリバスによって結合されていて、それぞれのプロセッサはキャッシュを持っていた。その結果として、プロセッサ数が増加すると、内容の同一性を保つのが困難になってくる。そこで、マルチポート共有メモリをプロセッサ同士の相互結合に使うことにして、プロセッサにはメモリへ接続するためのポートを複数用意する。そうすると、データの受渡しがメモリへのread / writeで実現でき、より高速で簡単なデータ転送が可能となる。現実には、同時にread / writeできるマルチポート共有メモリはないので、エミュレータボードを製作する。

MEMORY CONNECTED MULTIPROCESSOR MC1

Keiichi Sakai [†] Satoshi Fujita [†] Reiji Aibara [††] Tadasi Ae [†]

[†] Faculty of Engineering, Hiroshima University
[††] Research Center for Integrated Systems, Hiroshima University

Kagamiyama 1-4-1 Higashihiroshima City Hiroshima Prefecture 724 JAPAN

In this paper, we describe Memory Connected multiprocessor MC1. Usually the memory connected multiprocessor is connected by the shared memory bus, and then the processor has the cache between processor and shared memory. Consequentry, the more number of processor connected to the shared memory bus increases, the more it is difficult to keep coherency. This is a serious demerit toword a massively paralell processor. We desided to use the multiport shared common memory with the multi read/multi write capability among processors. Data send or receive is realized by memory write or read, respectively data can be transfered faster and easily. Actually, such a multiport shared common memory does not exist yet, and therefore, we realize an emulater board that is faster than the single bus connected shared memory.

1 はじめに

メモリ結合型のプロセッサ間結合方式には、図1のようにバスを用いる共有メモリのように普通のメモリを仮想的に multi read/multi write 化した型と、図2のようにマルチポート共有メモリの各ポートにプロセッサが接続される型の2つに大別できる。前者ではふつう図1のようにプロセッサと共有されるメモリの間にキャシュメモリを持ち、キャシュにヒットする限りでは、共有されるメモリをプロセッサが持っているローカルなメモリと同じ速度でアクセスできるようになっている。しかし、キャシュメモリと共有されるメモリの内容の同一性を保つ方法が、プロセッサ数の増加とともに困難になってくる。このような仮想共有メモリに対し、後者は実共有メモリと呼ぶべきものであるが、2以上のポートを持つ共有メモリは提案あるいは試作にとどまっている[3,8-10](光結合3次元集積回路回路技術によるマルチポート共有メモリの実現も期待されている[2-7])。

よって本稿では、後者のマルチポート共有メモリが将来実現するまでの代替手段として、ディスクリート素子によるマルチポート共有メモリのエミュレータボードを試作し、プロセッサどうしの結合にマルチポート共有メモリを用いる。また、この結合形式をマルチポート共有メモリ結合と呼ぶことにし、プロセッサどうしの結合にマルチポート共有メモリを用いたものをマルチポート共有メモリ結合型マルチプロセッサと呼ぶ[1]。このマルチポート共有メモリ結合型マルチプロセッサは、トランスピュータなどと同様、結合アーキテクチャは自由に選べるという特徴を持っている。

2 マルチポート共有メモリ

マルチポート共有メモリは、同時アクセスが可能で、どのポートから見える内容も全く同じである[3]。これをマルチプロセッサにおけるプロセッサ間のメッセージ転送に使えば、非常に高速な、つまりバンド幅の広いメッセージ転送が無手順で実現できるはずである。

そこで有望なのは、光結合3次元集積回路技術によるマルチポート共有メモリである。図3に示したのが、光結合3次元集積回路技術によるマルチポート共有メモリのブロック図である[2-6]。ひとつのポートに1層が対応して配され、各層は図の上下方向に光配線によって結合されている。ポートからあるアドレスに書き込まれると各層の対応するアドレスにブロードキャストされる。ポートから読み出される時は、それぞれの層から読み出されるので、全く問題なく同時読み出しが実現できる。

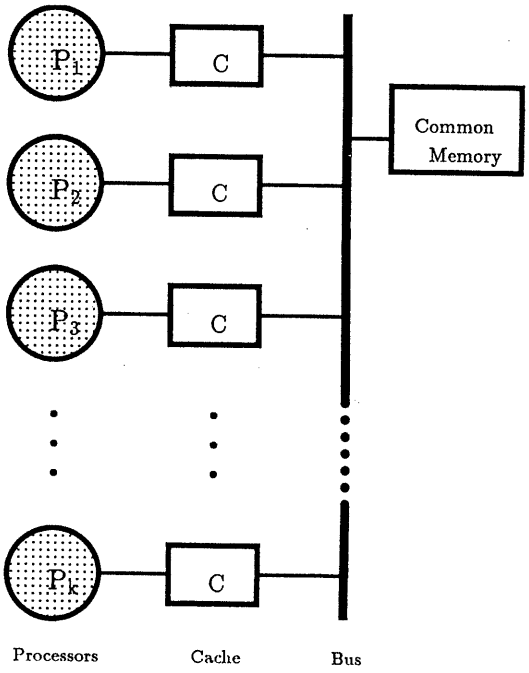

図1: バスを用いる共有メモリ

しかし、今のところ、光結合を用いたマルチポート共有メモリは完成されていない。従来は、図1のような共有されるメモリをバスにつなぎ多数のプロセッサがそれぞれ排他制御を行なって仮想的な共有メモリとして使用し、キャッシュを間に入れていた。そこで本稿では、光結合3次元集積回路技術によるマルチポート共有メモリが完成するまでの代替手段として、ディスクリート素子によりそのエミュレータボードを製作することにする。しかし、単にスタティック RAM をアービタの制御で仮想的にマルチポート化すると図1と本質的に変わらず、仮想的なマルチポート共有メモリと同様に sin-

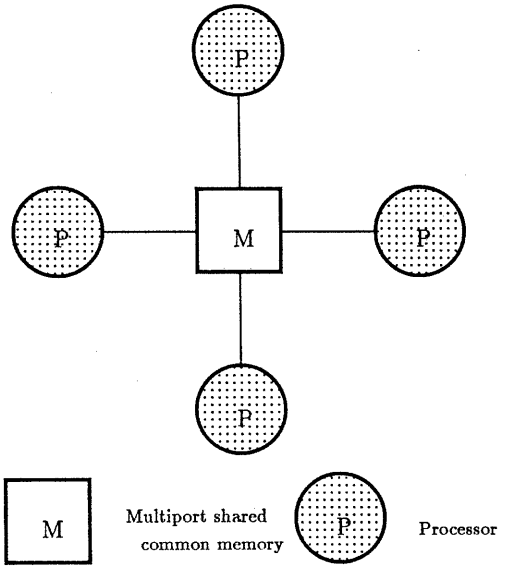

図 2: マルチポート共有メモリ結合

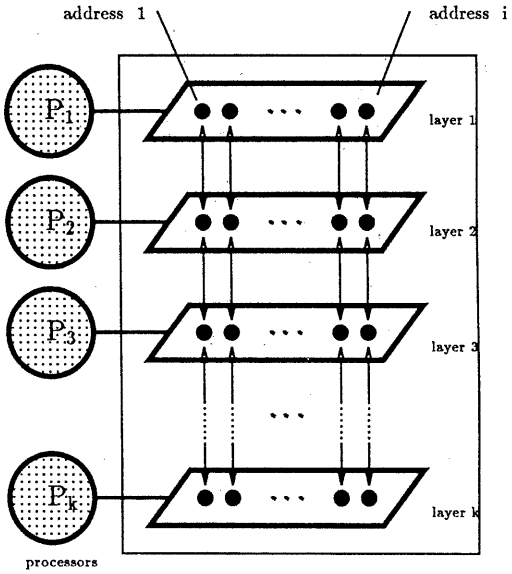

図 3: 光結合 3 次元 IC マルチポート共有メモリ

gle read/single write のままである。

そこで、ひとつのマルチポート共有メモリボードの内部構成をいくつかのメモリブロックに分割する。各メモリブロックは、セレクタにより切替えられる。そして、アービタが各メモリブロックに対してメモリアクセスの衝突を避けるように制御する。この方式では、read/write ともに平等の扱いをするので、read/write ともに平均的なアクセスタイムは同じである。またアービタやメモリの制御は高速で製作の容易な GAL(Generic Array Logic) を使用する。このブロック図を図 4 に示す。マルチポート共有メモリに接続されるおのおののプロセッサから見たメモリイメージは全く同等であり、各プロセッサからは 1 ワードおきに 2 組の S-RAM が交互にアクセスされる構成になっている。これは、ひとつのマルチポート共有メモリをアクセスしようとする時にそれぞれのプロセッサがアクセスするメモリブロックが異なっている限りでは実際に同時アクセスが可能である。実際の使用におけるプロセッサからマルチポート共有メモリへのデータ転送は、単なる memory read/write によって行なうので、無手順で高速であるという特徴を持たせることができる。

以上述べたことは、マルチポート共有メモリ結合型マルチプロセッサを構成する上で最も重要な事項なので、まずはマルチポート共有メモリのエミュレータボードを試作してその性能を評価することにした。

3 ディスクリート素子による設計

マルチポート共有メモリをディスクリート素子で試作する場合、内部で市販の S-RAM を共有されるメモリとして使うため、同時に 2 つ以上のプロセッサとアクセスできない。

そこで、メモリブロック一つあたり各プロセッサへ 1 組の制御信号線を用意する。ひとつは、メモリリクエスト信号でプロセッサのバスサイクルがマルチポート共有メモリに対するものであるときプロセッサ側から出力される。もうひとつは、プロセッサ側からのメモリリクエストに対してマルチポート共有メモリ側から返されるメモリアクノリッジである。それらの信号線の役割は、同じメモリブロックに対して 2 つ以上のプロセッサから同時にメモリリクエストがあったとき、プロセッサからは見えない

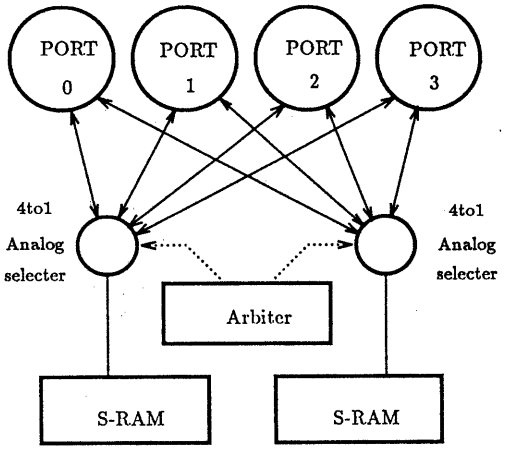

図 4 試作したマルチポート共有メモリ

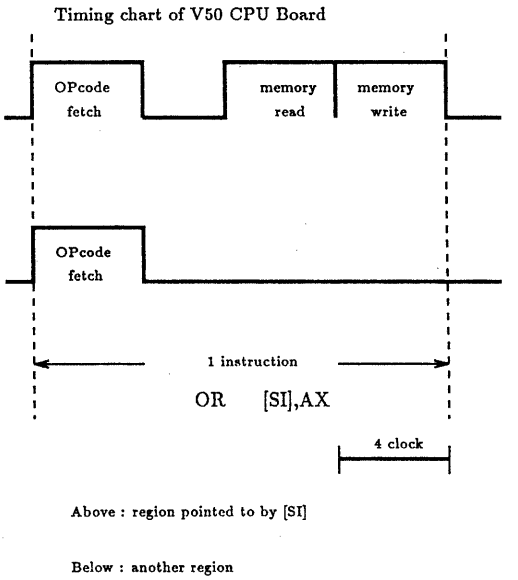

図 5: 評価に用いたプロセッサのタイミング

存在であるアービタが先着順にメモリアクノリッジをプロセッサに返すというものである。この機構により、プログラムを書く側からはローカルメモリに対するアクセスと同様の感覚で同じマルチポート共有メモリに接続された他のプロセッサへのデータ転送を可能にする。

ついで、データ転送の高速さをどの程度エミュレートできるかという問題がある。前述の機構により、プログラムを書く立場から意識するのは不可能であるが、実際にアクセスしようとするマルチポート共有メモリからのメモリアクノリッジがすぐに返されないときのことを考えてみる必要はある。実際に、そういう事態が起こればデータ転送の効率低下をひき起こすので、これまではキャッシュを設けていたのである。そこで本稿では、共有メモリへつながるバスの使用頻度はマルチポート共有メモリをデータ転送にのみ用いるとするとせいぜい 50%程度と考え、ポート数 4 に対して 2 メモリブロックを内部に持つ構成としてメモリアクセスを分散させることにした。設計の際に、マルチポート共有メモリに対するプロセッサのバスサイクルを短縮することと、アービタを高速にすることにより、マルチポート共有メモリによるデータ転送の高速性も可能な限りエミュレートできると考えたわけである。

4 試作したボードの評価

4.1 実験方法

実験では、マルチポート共有メモリの同一領域へのアクセスが集中した場合を想定して、試作したマルチポート共有メモリでどのくらい効率の低下が起こるかを測定した。

まず、2 つのメモリブロックは 1 ワードおきにメモリ空間に現れるので先に述べたように連続アクセスでは効率が落ちない。そこで、実際の使用とは異なるが、マルチポート共有メモリにインストラクションを書き込みそこに制御を移すという方法をとる。そうすることによってインストラクションフェッチによるメモリのリードも加わり同一領域へのアクセスがブロック転送などによるデータアクセスより集中した状態が実現できる。ここで使用した命令は、命令実行時間の割にマルチポート共有メモリをアクセスする時間の長い命令を使用する。例えば、下のようなものでタイミングチャートを図 5 に示す。

$$OR \qquad [SI], AX$$

この命令はソースインデックスレジスタ SI で指し示されるところのデータを一度テンポラリレジスタに読み出してアキュムレータ AX と OR をとった結果をまた同じところに書き戻す。つまり、read と write が 1 命令の中にある。また先ほど述べた、インストラクションフェッチによるメモリ読み出しが、ソースインデックスレジスタ SI で指される領域とそうでない領域交互に起こるので、read:4 write:4 insuruction fetch:4/2 の合計 10 クロックのメモリアクセスを行なえる。この命令の実行時間は 15 クロックなのでかなりの比率である。つまり、命令実行時間 15 クロックのうち 10 クロックはメモリをアクセスしているわけで、2 つのプロセッサが同時にこの命令を実行すると、ほぼ交互にメモリをアクセスすると考えると 1 命令の実行に 20 クロックほど必要となるはずである。次に測定方法だが、ジャンプ命令を埋め込んでループさせるとその分の計算が必要なために面倒である。そこで、評価に用いたプロセッサは、64*Kbyte* のセグメントの壁に当たるとセグメント内の先頭へ折り返すことを利用して、1 セグメント全部をこの命令で埋めつくした。そうすると、CPU のアドレスを監視することで信号変化の周期から 1 命令平均の命令実行時間を知ることができる。

4.2 実験結果および考察

実測値では、1 命令あたりの平均で 17.5 クロックとなった。1 命令おきに 2 つの CPU がマルチポート共有メモリをアクセスすると、1 命令の実行時間は 20 クロックとなるはずであったものが、それよりも短くそれほど効率が低下しなかった。これは、CPU のメモリサイクルを可能な限り短縮する設計の効果であると思われる。

つぎに、実際の使用ではインストラクションフェッチはプロセッサボードに搭載してあるローカルメモリより行なわれることと、ブロックワード転送のような連続的にメモリをアクセスする時は連続したメモリブロックが続いて選択されることがないという理由により 4 ポート同時アクセスが起こってもプログラム全体の実行時間程度の尺度で見た場合そう大きな効率の低下は起こらないと考えられる。試作が終ってデバッグのときに問題があった点は、

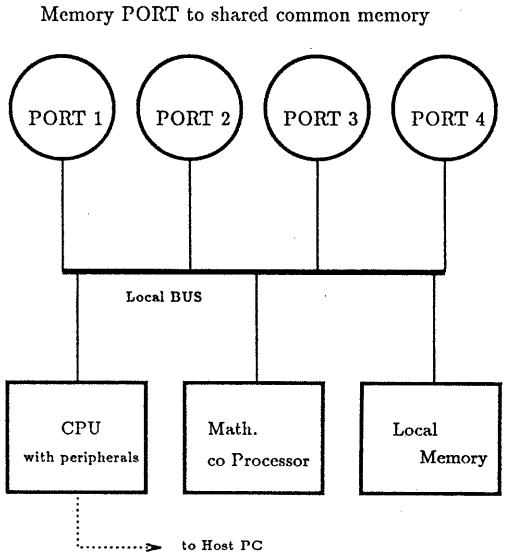

図 6: プロセッサのブロック図

切替えにアナログスイッチを使っていたためにプロセッサから見たメモリポートの入力容量がかなり大きくて、信号の電位が変化する度に流れるスパイク電流が隣接した信号線にクロストークを発生させていた。そのため、次節の試作では、74HC245 などのバスバッファを使用して、双方向性スイッチによるクロスポイントスイッチで、構成することにした。アナログスイッチと双方向性スイッチの違いは、ここでは、前者が信号の伝達方向に無関係に信号の伝達が可能であるのに対して、後者が信号の伝達方向を切替えれば信号が伝達できるという構造上の相違がある。試作では、信号の伝達方向を意識する必要がないだけ設計が容易であるので、前者を採用したわけである。

5 マルチプロセッサの設計

5.1 プロセッサの設計

今まで使用してきたプロセッサは、マルチポート共有メモリの評価用ということだった。そのため機能は貧弱であった。実際のマルチプロセッサとし

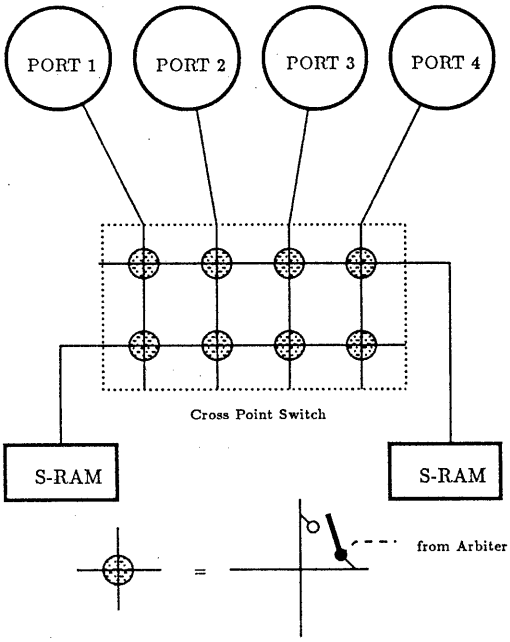

図 7: マルチポート共有メモリのブロック図

て機能させるためには、プロセッサの機能を強化しなければならない。

そこで、どこを強化するかという問題であるが、ソフトウェアを製作する立場からは CPU の性能が良いのが望ましいが、コンパイラの性能によっては、十分な性能が発揮できないことも良くある。8086 系列のプロセッサの場合 32*bit* コンパイラがあまり一般的でないので、特にその傾向が強い。またハードウェアを製作する立場からは 32*bit* 以上のデータバスまたはアドレスバスを持つプロセッサはバスの配線が多いため製作が困難になってくる。現実的にマルチプロセッサ全体に対する価格に制約があり、その範囲内においてプロセッサボードが数十枚からなるマルチプロセッサを製作せねばならない。そこで今回の設計でも、評価用のプロセッサと同じ V50 を使うことにした。このプロセッサは、周辺機能もまとめて一つのチップになっているのでメモリと若干の制御回路をつければ CPU ボードとしてスタンドアロンで動作させることが可能である。コンパイラは、Microsoft 社の C を採用することとした。これは、Microsoft 社の C に対応した ROM 化用ツールがあり、プロセッサに依存した制御が可能であることが主な理由である。またこのプロセッサは、ワープロなどに使われており入手性もよくて安いということもその理由にあげられる。

このブロック図を図 6 に示す。ローカルメモリには、ブート用 ROM (ホストパソコンを通じてプログラムのダウンロード・実行・デバッグを可能にするためのモニタを搭載) とデータ用 RAM それぞれ 256*Kbyte* 搭載する。また必要に応じて、80*x*87 数値演算プロセッサも搭載できる。図中点線で表したホストとの通信用の線は、シリアルバスにマルチドロップで接続され、シリアルバスを通じてホストパソコンからの指示によりプログラムのダウンロード・実行・デバッグが行なわれる。

図中では示されていないが、プロセッサ間で同期をとる手段として使うための割り込み線が 6 本ラックには張られている。それぞれの割り込み線はプルアップされていて、オープンコレクタのインバータとプロセッサへの割り込みピンが接続される。各割り込み線は全プロセッサとワイアード OR 接続され、全プロセッサがインバータ出力を OFF した場合にすべてのプロセッサに対して、おのおのの割り込みを発生させることができる。例えば、各プロセッサのプロセスの実行区分ごとに同期をとる必要がある場合、プロセスの実行区分最初でインバータ出力を ON(=L level) にしておく。その後、プロセスの実行区分が終りに達したら、インバータの出力を OFF(Hi impedance) にするとともに、そのプロセッサは HALT(割り込み待ち状態) にする。もし、最も実行の遅いプロセッサがインバータの出力を OFF にしたのであれば、その時点で、割り込みがかかり、全プロセッサが同時に次のプロセスを実行できる。さらに、各プロセッサにおいて、バックグラウンドジョブを走らせることができるようにタイマ/カウンタが 1 本使用可能である。

5.2 マルチポート共有メモリの設計

先に述べた通り、メモリとプロセッサポートの間を接続する素子を変更した。プロセッサから見た場合のこのマルチポート共有メモリの動作は、全く変更なく入力容量が減少したことと、マルチポート

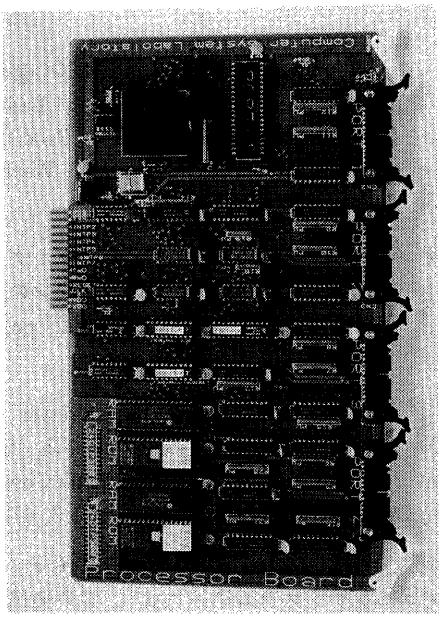

図 8: プロセッサボード

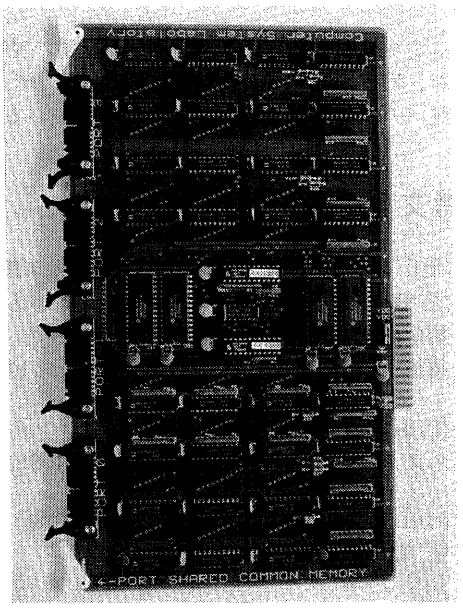

図 9: メモリボード

共有メモリの側にプロセッサボードから制御可能なアドレスラッチに変更されたくらいである。アドレスは、時分割でプロセッサから出されバスサイクルに入る直前確定するので、そのバスサイクルがマルチポート共有メモリに関するものであると確定した時点でアドレスをラッチする。そのためにアクセス時間を多くとることができ、メモリブロックの占有時間は試作版に比べてさらに減少させることが可能となった。

マルチポート共有メモリの構成を図 7 に示す。図中には表だってアービタの存在を書いていないが、クロスポイントスイッチを表す点それぞれがアービタの制御で動作する。このアービタと S-RAM の制御回路を含んだ GAL が S-RAM 1 ブロックごとに 1 個ある。それぞれのメモリブロックの大きさは、$64Kbyte$ でこのボード全体では $128Kbyte$ となる。このマルチポート共有メモリボードとプロセッサボードは 26 芯のフラットケーブルで接続することにより、自由な形態のマルチポート共有メモリ結合型マルチプロセッサの実験が可能になる。フラットケーブルは時分割多重されるデータ/アドレス線が 16 本、制御線が 7 本から構成されている。

6 おわりに

現在のところ、32 プロセッサの 4 次元逆ハイパーキューブ[1,12]に必要な数の CPU ボードとメモリボードは完成したので、その製作に取り掛かっている。完成したボードの全体写真をそれぞれ図 8 と図 9 にしめす。それぞれのボードの黒い 4 個のヘッダコネクタによって相互結合できるようになっている。必要のないコネクタは解放しておけば動作に差し支えないようにしてある。また、コネクタにフラットケーブルの接続されていないポートにアクセスしても動作に差し支えないようになっている。はじめに述べたように、(接続できるプロセッサが 4 までであれば) 任意の形状 (メッシュ、de Bruijn、etc) のマルチポート共有メモリ結合型のマルチプロセッサが構成できる。トランスピュータと同様の特徴をメモリ結合にまで拡張したという表現も可能である。

我々の研究室では、1978 年以来、一貫してマルチプロセッサ製作あたっては、1 号機 AKOVST[8,9,13]、2 号機 UNIP[9,11] ともメモリ結合を用いてきている。今回の 3 号機 MC1 においてもそれを踏襲したわけである。メモリ結合に関していえば、デュアルポートでは古くからもちいられているが[14]、マルチポートと

してはごく最近である[15]。キャッシュ全盛の現在であるが、デバイスサイドの支援とともにマルチポート共有メモリの発展に努力したいと考えている。

参考文献

[1] 阿江 忠, 藤田 聡, 相原 玲二, 山中 太市郎, 酒居 敬一: "光インターコネクション向きメモリ結合型超並列プロセッサアーキテクチャ", 信学技報, CPSY92-25, pp.41-46, (August 1992).

[2] 林 巌雄, 阿江 忠, 小柳 光正: "光インターコネクション", 通信学会誌, 75, 9, (1992).

[3] 阿江 忠: 新しい計算機アーキテクチャ(飯塚 肇 編), 第4章, 丸善, (1990).

[4] M.Koyanagi, H.Tanaka, H.Mori and J.Iba: "Design of 4kbit × 4Layer Optically Coupled Three-Dimentional Common Memory for Parallel Prosessor System", IEEE J.Solid State Circuits, 25, 1, pp.109-116, (1990).

[5] 小柳光正, 広瀬全孝, 阿江 忠: "3 次元光結合共有メモリを用いた並列処理コンピュータシステム", オプトロニクス, No.126, pp.76-82, (June 1992).

[6] 阿江 忠, 相原 玲二: "マルチプロセッサシステムのための光結合共有メモリ", 第28回情報処理学会全国体会, 3C-7, (1984).

[7] 阿江 忠: VLSI コンピュータ, 第5章, 電子情報通信学会, (1988).

[8] 阿江 忠, 高橋 浩一, 松本 健治: "共有メモリ結合によるマルチマイクロプロセッサの並列動作について", 電子通信学会論文誌, Vol.J65-D, No.3, pp. , (1982).

[9] T.Ae and R.Aibara : "Experimentation and Analysis of Multiprocessor Systems", Proc.IEEE Real-Time Systems Symposium, L.A., pp.69-80, (1982).

[10] T.Ae, S.Tenma, H.Yamasaki and M.Kitagawa: "Distributed Real-Time Processing Lanuage on Multimicroprocessor System", Proc.IEEE Real-Time Systems Symposium, Washington D.C., pp.20-29, (1983).

[11] 相原 玲二, 阿江 忠: "マルチマイクロプロセッサによるソート/サーチエンジンの試作", 情報処理学会論文誌, Vol.26, No.2, pp.349-355, (March 1985).

[12] T.Ae et al. : "Hypercube is better than De Bruijn for Connectionist", 5th ISMM Int.Conf. on Parallel and DistributedComputing and Systems, Pittsburgh, Oct.1-3, (1992).

[13] 阿江, 大崎, VUong, : "小規模マルチマイクロプロセッサシステムの一方式", 電子通信学会電子計算機研究会資料 EC78-35, (1978).

[14] 星野 力: PAX コンピュータ—高並列処理と科学計算—, オーム社, (昭和 60).

[15] Daniel Litaize, Abdelaziz Mzoughi, Christine Rochange, Pascal Sainrat : "TOWARDS A SHARED-MEMORY MASSIVELY PARALLEL MULTIPROCESSOR", 19th. annual ISCA, pp.70-79, (1992).