

## 二進木結合並列計算機 Coral 68 K の開発とその評価†

高橋 義造<sup>††</sup>      遠藤 俊雄<sup>†††</sup>  
 松尾 賢二<sup>††††</sup>      槌谷 一<sup>††</sup>

大規模計算を高速に行うために二進木結合並列計算機 Coral 68 K を開発し、基本的なプログラミング開発環境を整備した。Coral 68 K は 63 台のプロセッサ要素を DMA 回路によって二進木状に結合したものをホスト計算機となるワークステーションに接続したものである。各プロセッサ要素はマイクロプロセッサ MC 68000, DMA コントローラ MC 68450, 16 KB ROM, 512 KB RAM より構成されている。Coral 68 K のプログラミング環境としてコンパイラ、メッセージプロセッサ、デバッガ、ランタイムサブルーチンがある。メッセージプロセッサはホスト計算機から放送されるメッセージを受け取り、それに書かれているコマンドにしたがって処理を行うようになっている。この計算機を用いて各種の定型的な並列処理問題を計算し、速度向上率やプロセッサ利用率などの並列処理効率を実測し、二進木結合並列計算機の有効性を評価した。

### 1. はじめに

大規模な計算を比較的小型のプロセッサ要素を多数使用することによって高速に処理することを目的とする並列計算機の研究が各方面で行われており、各種の方式が提案されると共に、実際にプロトタイプが試作され、並列アルゴリズムの開発に利用されたり、実規模の並列計算機を開発するための設計資料を得る目的で評価に使用されている。我々は高並列度システムに適応する疎結合並列計算機について検討した結果、超 LSI 化に適した二進木結合方式に着目し、先に小規模の実験機 Coral '83 を開発し並列アルゴリズムの研究に使用してきた<sup>1)</sup>。その経験からこの方式の並列計算機が実際の問題の処理に有効であるとの確信を持つに至ったので、実地的な問題に使用できる実規模の二進木結合並列計算機 Coral 68 K の開発に取り掛かり、このほど完成した。この論文は Coral 68 K の構成と、そのプログラミング環境、および各種の定型的な問題の並列処理を行って実測した性能にもとづいて二進木計算機の評価結果について述べるものである。その前にまず二進木計算機の特長とこれまでに開発された二進木計算機と Coral 68 K の比較を第 2 章で行い、第 3 章でハードウェアの構成、第 4 章でプログラ

ミング環境、第 5 章で性能評価の結果について述べ、第 6 章の結論ではこれらの開発経験と評価に基づく二進木計算機の方式の評価を行う。

### 2. 二進木計算機の特長

疎結合並列計算機の代表的な結合方式にはトーラス結合、ハイパーキューブ結合、二進木結合がある。この 3 種類の結合方式の特性を平均接続数、平均通信距離、平均放送距離で示すと表 1 のようになる。ここに平均接続数とは 1 台のプロセッサ要素を他のプロセッサ要素と接続するための接続回路の数の平均値であり、平均通信距離は任意の 2 台のプロセッサ要素間に存在する接続回路数（つまりこの 2 台が通信するのに必要な中継回数に 1 を加えた値）の平均値である。平均放送距離は任意の 1 台のプロセッサ要素から放送した情報がすべてのプロセッサ要素に到着するまでの中継回数に 1 を加えた値の平均値である<sup>2)</sup>。（なお参考文献<sup>2)</sup>には特定のプロセッサ要素からの放送距離が示されている。）

二進木結合の特長として次の事項があげられる。

- (1) 再帰的構造であり、構成が簡単である。
- (2) 階層構造であるので探索問題、分割統治法による並列処理に適している。
- (3) 平均および最大接続数が小さいのでプロセッサ要素の構造が簡単である。
- (4) 平均通信距離および平均放送距離が小さい。
- (5) 接続が平面的であり、WSI（ウェハー・スケール・インテグレーション）向きである。

これらの中で(3)(4)の性能はハイパーキューブに次ぐものであるが、(5)の特長はそれにはない特長で

† Development and Performance Evaluation of the Coral 68 K Binary-Tree Parallel Computer by YOSHIZO TAKAHASHI (Department of Information Science and Intelligent Systems, Tokushima University), TOSHIO ENDO (Toshiba Inc.), KENJI MATSUO (IBM Japan, Ltd.) and HAJIME TSUCHITANI (Department of Information Science and Intelligent Systems, Tokushima University).

†† 徳島大学工学部知能情報工学科

††† (株)東芝

†††† 日本 IBM(株)

表 1 各種疎結合並列計算機の結合方式の性能比較  
Table 1 Performances of some typical loosely coupled multiprocessors.

| 結 合 方 式  | 平均接続数      | 平均通信距離       | 平均放送距離       |
|----------|------------|--------------|--------------|
| トーラス     | 4          | $\sqrt{N}/2$ | $\sqrt{N}$   |
| 二 進 木    | 2          | $2 \log_2 N$ | $3 \log_2 N$ |
| ハイパーキューブ | $\log_2 N$ | $\log_2 N/2$ | $\log_2 N$   |

ある。

また一方では二進木結合の欠点として次の事項が指摘されている。

- (1) プロセッサ要素によって通信負荷にアンバランスがある。
- (2) 任意のプロセッサ要素間に複数の通信路がないので信頼性が劣る。

通信負荷のアンバランスは問題の分割法と割当法を選ぶことによって回避できる。また信頼性の問題は次のように考えると重要な欠点とは考えなくてもよいことがわかる。すなわち根の部分に故障するとシステム全体が使用できなくなるが、葉の部分だとそのプロセッサ要素が使用できないだけである。 $N$  台のプロセッサ要素のうちの 1 台が故障したとして、それが葉である確率は  $1/2$  であるが、根である確率は  $1/N$  にすぎない。したがって 1 台のプロセッサ要素が確率  $f$  で故障したときに使用できなくなるプロセッサ要素の数の期待値  $u$  は次のように計算され、さほど大きいものではないことがわかる<sup>3)</sup>。

$$u = f \cdot \log_2 N \quad (1)$$

我々が 1983 年に開発した小規模の二進木結合計算機 Coral '83 の使用経験から実用規模の問題を処理するのに必要な計算処理能力、メモリ容量、プログラミング環境の開発のめどを得たので、今回 16 ビットマイクロプロセッサを 63 台結合した実規模の二進木結合並列計算機 Coral 68 K を設計し開発したものである。なおこれまでに実際に製作された二進木構造の並列計算機としては上記の Coral '83 のほか、九州大学の Hyphen-C 16<sup>4)</sup>、豊橋技術科学大学の DON<sup>5)</sup>、コロンビア大学の Non-Von<sup>6)</sup> などがあるが、いずれも Coral 68 K とは構成や処理の方式が異なっている。すなわち Hyphen-C 16 は二進木の葉に

プロセッサ要素がありノードはスイッチになっており、DON は二つの木を葉で接続したダブルトリー構造で、Non-Von は葉とノードのプロセッサを別々の結合網で接続している。この点では Coral 68 K は二進木の葉もノードもプロセッサになっており、いわば純二進木構造の結合方式である。コロンビア大学の Dado 2<sup>7)</sup> は同じ構造で、プロセッサ要素の数も 1,023 台と多いがこれには処理能力の小さい 8 ビットのマイクロコントローラが使用されており広い範囲の実用規模の計算を行うのは無理であろう。

### 3. Coral 68 K の構成

#### 3.1 システム構成

以前に開発した二進木計算機のプロトタイプ Coral '83 は 15 台の 8 ビットマイクロプロセッサより構成されており、プロセッサ要素の接続方式は今回の Coral 68 K と同じ二進木であるが、プロセッサ間結合回路がプログラム入出力方式であるために 9 KB/秒と遅く、またメモリ容量もきわめて小さいので十分なプログラム開発環境を持つことができなかった。また並列度の大きいところで果たして二進木構造が並列計算機に適しているのかを評価するにはプロセッサ台数が不足であった。これに対して今回開発した Coral 68 K は 16 ビットマイクロプロセッサを使ったプロセッサ要素 63 台より構成したので処理能力が大幅に向上し、大型計算機と匹敵する性能を持つことができた。またプロセッサ台数が増えた分だけ増えるプロセッサ間通信のオーバーヘッドを抑えるために DMA 方

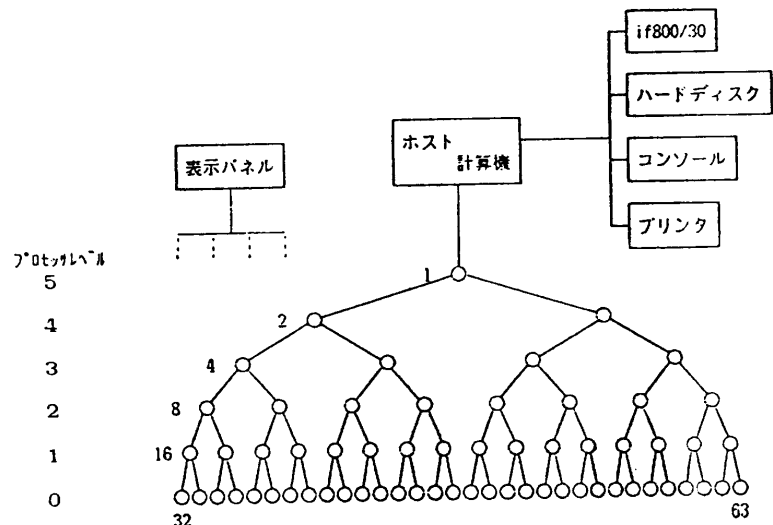


図 1 Coral 68 K の構成図

Fig. 1 Organization of the Coral 68 K.

式による結合回路を採用した。また各プロセッサ要素のメモリ容量を大幅に増やして二進木計算機特有のプログラム開発環境を持たせた。この結果 Coral 68K は実用問題の並列処理に使用できるようになった。

Coral 68K は図1に示すように二進木状に結合された63台のプロセッサ要素と、その根に接続されたホスト計算機より構成されている。ホスト計算機は16ビットのワークステーションで、プロセッサ要素で実行するプログラムの作成と管理、コンパイル、ローディングを行うほか、プロセッサ要素で実行中のプログラムの入出力サービス、ファイル管理、デバッグなどを行う。ホスト計算機と木の根に当たるプロセッサ要素（以後ルートプロセッサと呼ぶ）の間は16ビットの並列インタフェースによって接続した<sup>8)~10)</sup>。

プロセッサ要素は図2に示すように独立した1本の筐体に納めた。63台のプロセッサ要素を相互にDMA方式による16ビットのチャンネルで二進木状に結合し、各プロセッサ要素からは筐体表面の表示パネルと各シャーシに付属したテストパネルの制御信号を引き出している。表示パネルは実行中の各プロセッサ要素の動作状態をプログラマが直接監視して並列プログラムのデバッグや、動的な負荷のバランスの模様などを評価できるようにするために設けたものである。これは図2の筐体の表面に二進木状に配置された合計126個のLED（発光ダイオード）から構成し、各プロセッサ要素の動作状態およびプロセッサ要素相互間で行われている通信とその方向通信が一目で見えるようにした。LEDには図3に示すように各プロセッサ要素に

対応して1個ずつの赤色のものと、各接続回路ごとに1個ずつの2色（緑・橙）のものがある。赤のLEDはプロセッサ要素からの命令によって駆動され、プログラムの実行状態を表示するのに使う。また2色LEDはプロセッサ要素間通信回路によって直接駆動され、TOP方向から受信が行われる時には緑が、TOP方向に送信が行われる時には橙色が一定時間点灯するようにした。したがって小さなデータの転送が一瞬に行われても、そのことを知ることができる。

63台のプロセッサ要素を同時に動作させる並列プログラムのバグをソフトウェアだけで行うのは容易ではない。そこで細かな実行中のプログラムの動作状態をプログラマに知らせることができるようテストパネルを筐体の裏側に設けることにした。これはプロセッサ要素ごとに3個のLEDをおいたものでプログラムによって3ビットの情報を表示することができる。これはまたテストプログラムによるハードウェアの診断結果の表示にも使用している。

Coral 68Kの構成要素と仕様を表2にまとめて示す。

### 3.2 プロセッサ要素の構成

各プロセッサ要素の構成はすべて同一で図4に示すとおりである。演算部として16ビットMPUのMC-

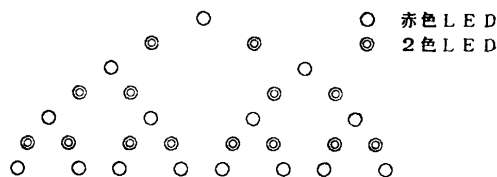


図3 表示パネルのLED  
Fig. 3 LED array on the display panel.

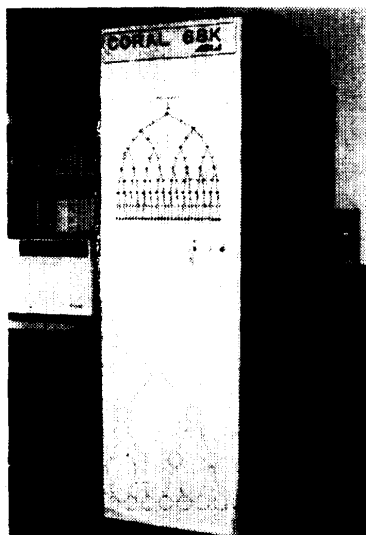


図2 Coral 68Kの外観  
Fig. 2 The Coral 68K

表2 Coral 68Kの仕様  
Table 2 Specifications of the Coral 68K.

| 要素      | 項目      | 仕様                |
|---------|---------|-------------------|
| ホスト計算機  | モデル     | シャープ IX-5         |
|         | CPU     | MC 68000 (10 MHz) |
|         | メモリ     | 4 MB              |
|         | ハードディスク | 67 MB             |
| プロセッサ要素 | OS      | UNIX SYSTEM V     |
|         | 台数      | 63                |
|         | CPU     | MC 68000 (10 MHz) |
|         | ROM     | 16 KB             |
|         | RAM     | 512 KB            |
| データ転送速度 | ホスト-PE間 | 3.7 KB/秒          |
|         | PE-PE間  | 2.0 MB/秒          |

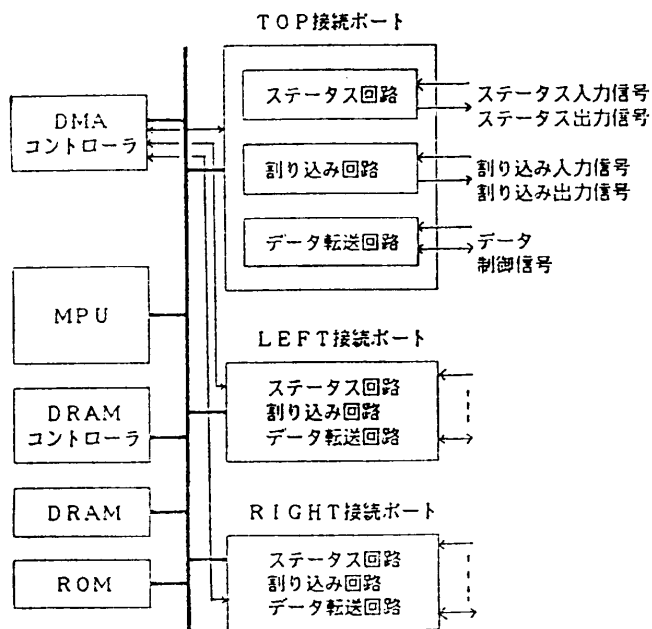


図 4 プロセッサ要素の構成図

Fig. 4 Block diagram of the processing element.

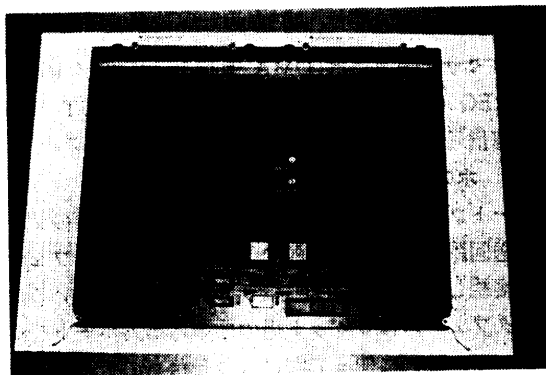


図 5 プロセッサ要素の外観

Fig. 5 A processing element.

68000 (10 MHz), 16 KB の ROM, 512 KB の RAM, および他のプロセッサ要素と接続するための 3 方向の接続ポートを 1 枚の 4 層プリント配線基板上に搭載した。プリント基板の寸法は 32 cm × 26 cm で、96 ピンの DIN コネクタが 2 個と、電源供給用のピンブロックを 4 個付けた。図 5 にこのプロセッサ要素の外観図を示す。

経済的な理由だけから演算部には算術プロセッサを付加しなかったが、このため Coral 68K は浮動小数点計算の性能が劣ることになり応用分野が主として整数演算型の問題に限られるという制約を受けることになってしまった。

ROM 部には 64K PROM を 2 個使用し、アドレス

空間の先頭の 16 KB を割り当てた。ここには割り込みベクトル、IPL プログラム、メッセージ処理プログラムを書き込んでいる。RAM 部には 16 個の 256 K DRAM と DRAM コントローラ (MB 1422) を使用し、アドレス空間の 16 KB からの 512 KB を割り当てた。

### 3.3 接続ポート

各プロセッサ要素には TOP, LEFT, RIGHT の 3 方向に対してそれぞれ独立した接続ポートを設けた。各接続ポートには図 4 に示すようなステータス回路、割り込み回路、およびデータ転送回路がある。ステータス回路には 3 ビットのステータス信号のための出力および入力信号線があり、それぞれ相手のプロセッサ要素のステータス回路の入力信号線と出力信号線に交差して接続される。

ステータス信号はオペレータに各プロセッサ要素の状態を知らせるためと、隣どうしのプロセッサ要素間で短い情報を直接伝達するための二つの目的に使用する。前者の目的のためにステータス出力信号がテストパネルの 3 個の LED を駆動し、また最上位の 1 ビットが筐体表面の表示パネルの赤色 LED を駆動するようにしている。後者の目的のためには通信の場合のハンドシェイク信号、割り込み原因の通知、データの要求信号などに使用する。

データ転送回路には DMA 方式を採用した。2 台のプロセッサ要素間で高速データ転送を行う方法として通信メモリや FIFO メモリなどの共有メモリを使用する方法と、共有メモリをおかず DMA によってそれぞれのローカルメモリ間でデータを移動する方法が考えられる。通信メモリでは競合問題を解決する必要がある上に一度に転送するデータ量に制限がある。また FIFO メモリでは競合問題はないが現在入手できる FIFO メモリの容量が小さいのでデータ量が著しく制限される。共有メモリなしの DMA 方式ではこのような問題は起こらない。特に DMA コントローラ MC 68450 には 4 チャンネルが内蔵されているので非常に都合がよい。ただし 5.2 節に述べるように DMA コントローラに設定時間が大きいという問題点もある。今回開発したデータ転送回路は 16 本の双方向データ線と 5 本の制御線で構成する。3 方向の接続ポートのデータ転送回路は 1 個の DMA コントローラ MC 68450 によって制御し、バーストモードで 16 ビットデータの転送を行う。このモードのデー

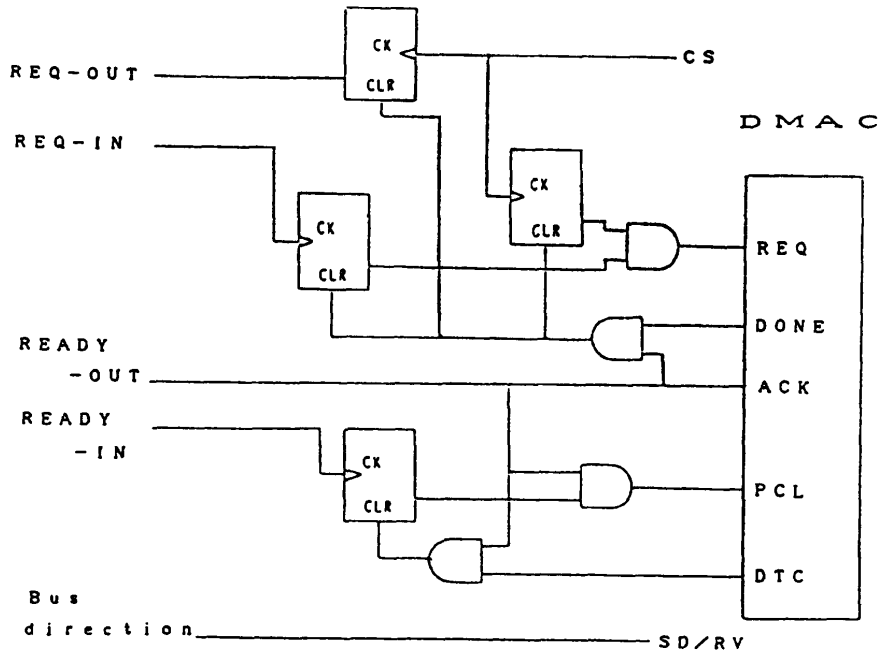


図 6 制御線の接続  
Fig. 6 Control lines interconnected.

タ転送中はバスが占有されるので3方向同時にデータ転送を行うことはできないが 2 MB/秒の高速転送が行われる。制御線は相手側のプロセッサ要素の制御線と図 6 のように接続し、それぞれ次のような信号をのせる。

1. 受信要求 (REQ-IN)
2. 送信要求 (REQ-OUT)
3. 受信可能 (READY-IN)
4. 送信可能 (READY-OUT)
5. バス方向 (Bus-direction)

送受信の方向はバス方向信号によって決定されるが、この信号は LEFT, RIGHT の方向には出力信号、TOP 方向には入力信号となっている。したがって TOP 方向のプロセッサ要素が常にバス方向の決定の主導権をとるのでバス調停は不要になる。一方が受信のとき相手側が送信するというようにプロセッサ間通信を協調的に行うような並列処理プログラムを書けば通信方向がぶつかる恐れはない。そうでない場合、例えば相手側が受信していない時に送信したいような時には、割り込み回路を使って相手側に割り込みをかけステータス信号で受信要求を知らせるようにする。なお割り込み回路の優先順位は TOP, LEFT, RIGHT の順になっており、デバッグに使用する割り込みスイッチからの割り込みの順位はこれよりも高い。

プロセッサ間通信は DMA コントローラが REQ-IN, REQ-OUT, READY-IN, READY-OUT 信号を確認信号に使うて MPU とは独立に行われる。

### 3.4 ホスト・プロセッサ要素間通信

ルートプロセッサの TOP 方向の接続ポートのデータ転送回路はホスト計算機の並列入出力インタフェースに接続し、16 ビット並列でデータ転送を行う。このインタフェースは 8 ビット用の標準入出力ポートを 2 チャンネル並列に使うて間に合わせたもので、DMA ではなくプログラム入出力方式になっている。そのためプロセッサ要素間に比べて転送速度が 2 桁ほど遅いが Coral 68K による計算の大部分はプロセッサ要素側で行うので全体の性能には大きな影響はない。

### 3.5 構造

Coral 68K の筐体には 4 台のシャーシを収納し、これに 63 枚のプロセッサ要素を分けて納めている。1 台のシャーシには 16 スロットがあり、これに図 7 (a) のようにプロセッサ要素を配置した。シャーシ 4 の余ったスロットにはホスト計算機とのインタフェースを納めた。バックパネルの配線の重なりをできるだけ避けるために 1 台のシャーシでのプロセッサ要素の配置は図 7 (b) のように中間順にした。このように配線が簡単になるのは二進木結合方式の特長の一つである。

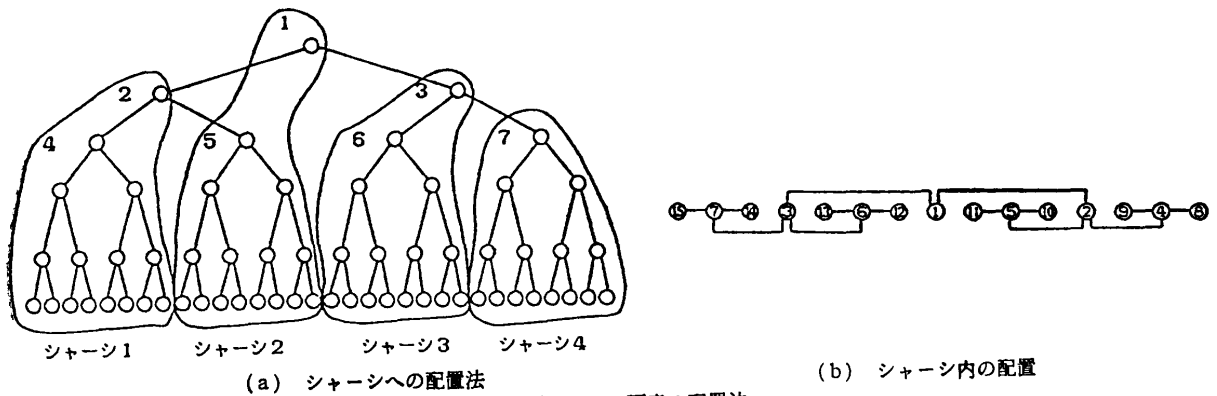


図 7 プロセッサ要素の配置法  
Fig. 7 Placement of the processing elements.

1台のプロセッサ要素あたり 15 W 程度の電力を消費するので冷却のために各シャーシ間と筐体の天井の 4 箇所にファンユニットを取り付けた。

テストパネルと表示パネルのための LED 駆動回路は外付けにし各パネルの裏に納めた。

#### 4. ソフトウェア開発環境

##### 4.1 プログラミング

並列処理プログラムの開発を行うには通常の計算機とは違った機能をもつソフトウェア開発環境が必要である。Coral 68K ではホスト計算機のソフトウェア開発環境をそのまま利用するほか、二進木計算機の機能を活用するためのソフトウェアを開発し、その一部は ROM に書き込んで使用できるようにした。その結果ホスト計算機の UNIX-System V、ホスト計算機用とプロセッサ要素用にそれぞれ用意したランタイムサブルーチンライブラリ、並列デバグ、ROM に書かれたメッセージプロセッサが Coral 68K のソフトウェア開発環境になる。

Coral 68K を使用するにはホスト計算機用と、プロセッサ要素用の 2 種類のプログラムを作る必要がある。プロセッサ要素ごとに違うプログラムを作ることできるが、通常は同じプログラムのコピーを全プロセッサ要素にロードして、プログラムの中でプロセッサ番号などを調べて各要素で違う処理をさせるようにしている。この方が多少メモリは無駄になるがプログラムのデバグが容易になる利点がある。どちらのプログラムも C 言語で書き、ホスト計算機でコンパイルする。ホスト計算機用のプログラムには表 3 に示すようなランタイムサブルーチンが使用できるが、この中で load ルーチンは全プロセッサ要素に同じプログラ

表 3 ホスト計算機用ランタイムサブルーチン  
Table 3 Run-time subroutines for the host computer program.

| ルーチン名                 | 機能           |
|-----------------------|--------------|
| crlopen ( )           | プロセッサ要素のオープン |
| crlclose ( )          | " クローズ       |
| crlputh (var)         | 2 バイトデータの送信  |
| crlgeth (&var)        | " 受信         |
| crlputi (var)         | 4 バイトデータの送信  |
| crlgeti (&var)        | " 受信         |
| crlputbk (&var, size) | ブロックデータの送信   |
| crlgetbk (&var, size) | " 受信         |
| load (filename, adr)  | プログラムのロード    |
| go (adr)              | プログラムの実行     |

ムを指定した番地からロードするためのものであり、go ルーチンは全プロセッサ要素に指定した番地から実行することを指令するためのものである。特定のプロセッサ要素にだけプログラムをロードしたり、実行させるには 4.2 節に述べるコマンドメッセージを使用する。

プロセッサ要素のメモリのアドレス空間は図 8 のように割り当てた。先頭の 16K バイトは ROM エリアで、0x10000 番地からの 512K バイトが RAM エリアになる。ROM には割り込み処理プログラムやメッセージプロセッサが書き込まれており、これが RAM エリアの先頭の 8K バイトにあるシステムエリアを管理する。ホスト計算機からロードするプログラムは 0x12000 番地以上のユーザエリアに格納する。したがってプログラムをコンパイルしてリンクする時にロードアドレスを指定する必要がある。0x100000 以降の番地は DMA コントローラやその他のデバイス

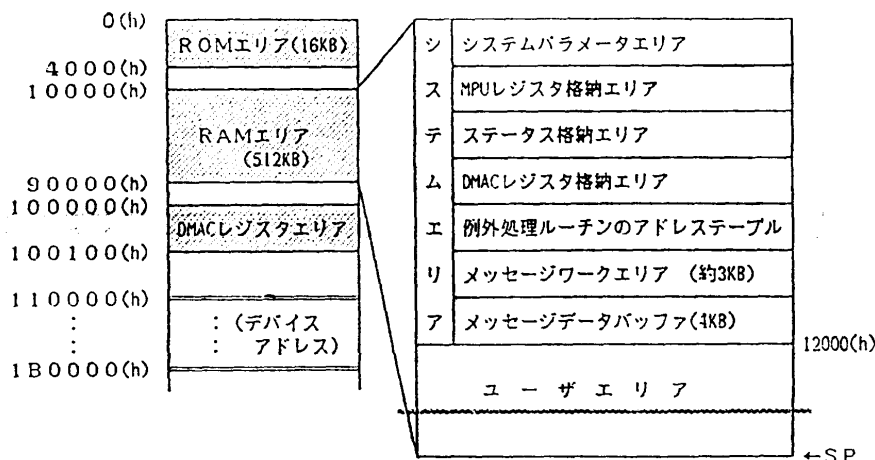


図 8 プロセッサ要素のアドレス空間  
Fig. 8 Memory space of a processing element.

表 4 プロセッサ要素用ランタイムサブルーチン  
Table 4 Run-time subroutines for the processing element program.

| ルーチン名                   | 機能                |
|-------------------------|-------------------|
| intrpt (dir)            | dir の方向に割り込み      |
| insts (dir)             | dir の方向のステータス入力   |
| outsts (dir, data, cnt) | " ステータス出力         |
| puth (var, dir)         | dir の方向の2バイト送信    |
| geth (&var, dir)        | " 受信              |
| puti (var, dir)         | dir の方向の4バイト送信    |
| geti (&var, dir)        | " 受信              |
| putbk (&var, size, dir) | dir の方向のブロックデータ送信 |
| getbk (&var, size, dir) | " 受信              |

のアドレスに使用している

プロセッサ要素用のプログラムのために、割り込み、ステータス入出力、データ転送のための表4に示すようなランタイムサブルーチンを用意した。ここで dir では TOP, LEFT, RIGHT の方向を指定する。離れたプロセッサ要素に転送する場合には途中のプロセッサ要素で中継する必要があるが、これはユーザプログラムで処理するか、あるいは別に開発された OS を使用しなければならない。

#### 4.2 メッセージプロセッサ

メッセージプロセッサはテストプログラム、初期化プログラム、割り込み処理プログラムなどと共に各プロセッサ要素の ROM に書き込まれている。各プロセッサ要素ではリセット割り込みでまずテストプログラムを実行し、メモリテストと接続ポートの診断を行

い、各プロセッサ要素は3方向の接続ポートに他のプロセッサ要素が接続されているかどうかを調べる。次に初期化プログラムでシステムパラメータを初期化し、それが終わるとメッセージプロセッサの実行に移る。メッセージプロセッサは TOP 方向よりシステムメッセージが送られてくるのを待ち、これを受け取るとその中に書いてあるコマンドに応じた処理を行い、処理が終了すると再びメッセージ待ちの状態になる。これを状態遷移図に表したのが図9である。またメッセージ処理中に割り込みスイッチが押されると処理を中断してメッセージ待ちになり、「継続」のシステムメッセージを受け取ると中断した所から処理を継続する。

初期化の時に設定されるシステムパラメータには次のようなものがある。

プロセッサ番号 (4バイト)

プロセッサレベル (4バイト)

LEFT 方向に連結されているプロセッサ要素数 (4バイト)

RIGHT 方向に連結されているプロセッサ要素数 (4バイト)

自分自身を含む部分木のプロセッサ要素数 (4バイト)

連結されている全プロセッサ要素のプロセッサリスト (8バイト)

初期化プログラムでは TOP 方向にプロセッサ要素が接続されていないプロセッサ要素はルートプロセッサであると判断し自分のプロセッサ番号を1に設定し、LEFT 方向には2を、RIGHT 方向には3を送る。この値を受け取ったプロセッサ要素はその値を自分の番

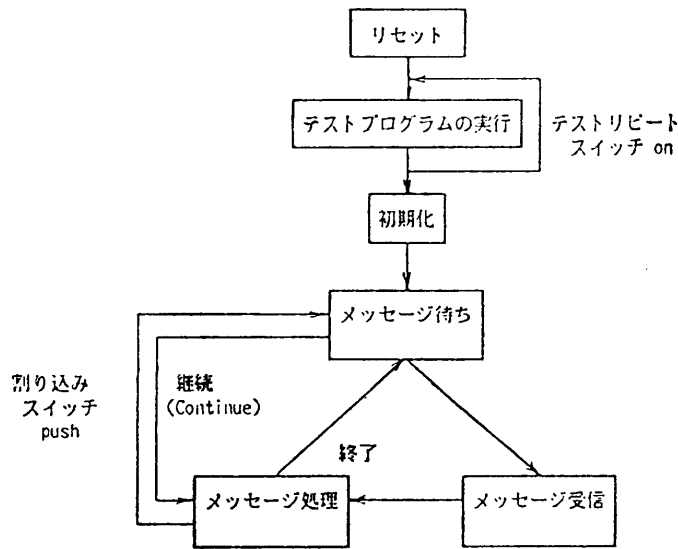


図 9 プロセッサ要素の状態遷移図

Fig. 9 State transition diagram in a processing element.

号とし、2倍したものを LEFT へ、2倍プラス1したものを RIGHT に送る。このようにしてルートプロセッサに連結されているすべてのプロセッサ要素の番号が幅順 (breadth-first order) に設定される。それらを 64 ビットデータのビット位置で表現したものがプロセッサリストである。これは葉のプロセッサ要素 (これをリーフプロセッサとよぶ) からの報告をもとにルートプロセッサで作って各プロセッサ要素に放送するようにしている。

メッセージ待ちの状態になるとプロセッサ要素は表示パネルの赤の LED をつけ、メッセージを受け取ると消す。システムメッセージは必ず TOP 方向から送られてくるので、これを受け取ったプロセッサ要素は LEFT, RIGHT にプロセッサが接続されていればこれに転送する。システムメッセージは 20 バイトの固定長のストリングで図 10 のようにコマンド部、宛先部、アドレス部、サイズ部の 4 つの部分よりなっている。

コマンド部にはコマンドのコードが書き込まれる。コマンドには次の種類があり、その宛先部のプロセッサリストで指定されたプロセッサ要素ではそれぞれ以下に述べる処理を行う。

(1) 初期化：先に述べた初期化と同じ処理、すな

わちシステムパラメタの設定を行う。

(2) ロード：このシステムメッセージに続いて TOP から送られてくる、サイズ部で指定されるサイズのプログラムをアドレス部で指定されるアドレスから始まるメモリ領域に書き込み、LEFT, RIGHT 方向にプロセッサ要素が接続されていればこれにプログラムを転送する。

(3) 実行：アドレス部で指定されたアドレスからプログラムを実行する。

(4) メモリダンプ：アドレス部とサイズ部で指定された範囲のメモリの内容を TOP に送る。このデータは図 8 中に示した 4 KB のメッセージデータバッファを使って中継されるので 4 KB より大きい場合には何度かに分けて転送する

(5) メモリライト：このメッセージに続いて送られてくるデータを指定された範囲のメモリ領域に書き込む。4 KB より大きい領域の場合は何回かに分けて転送する。

(6) レジスタダンプ：MPU のレジスタを TOP に送る。

(7) テスト：リセット時に行うのと同じプロセッサ要素のテストを行う。

(8) 継続：割り込みスイッチによって中断されたプログラムの MPU のレジスタ、出力ステータス、DMAC のレジスタをシステムエリアから回復し、プログラムを継続して実行する

#### 4.3 割り込み処理プログラム

プロセッサ要素への割り込みは 3 方向からのものと、プロセッサスイッチによるものである。プロセッサ要素の割り込みは送受信の要求や不規則に起きる計算の状態を隣接のプロセッサに知らせるのに使用するが、このときの割り込み原因を知らせるのにはステータス出力信号を使用する。割り込みスイッチをおすと割り込みが全プロセッサ要素にかかる。この場合にはプロセッサ要素は MPU のレジスタ、DMA コントローラのレジスタ、ステータス出力をシステムエリアのそれぞれの格納エリアに書き込み、メッセージ待ちの状態になる。

| コマンド(4バイト) | 宛先(8バイト) | アドレス(4バイト) | サイズ(4バイト) |
|------------|----------|------------|-----------|
|------------|----------|------------|-----------|

図 10 システムメッセージの形式

Fig. 10 System message format.

#### 4.4 デバugg

実行中のプログラムを任意の時点で止めてデバuggを行うことができるようにした。これは次のようにして行う。筐体正面の扉にある割り込みスイッチを押して全プロセッサ要素に割り込みをかけ、メッセージ待ちの状態にする。(このときは表示パネルの赤い LED がつく。) ルートプロセッサはホスト計算機に中断の信号を送る。ホスト計算機では通常ルートプロセッサから計算結果が届くのを待っているが、この信号を受け取ると処理を中断し、デバugg cdb を起動する。cdb は対話形式で動作し、プロセッサ要素のレジスタやメモリを読んだり書いたりするためのシステムメッセージを作ってプロセッサ要素に送る。最後に「継続」メッセージを送ればプロセッサ要素側では中断した所から処理が再開され、ホスト側でも元のプログラムに戻る。

### 5. 性能評価

#### 5.1 演算処理能力

Coral 68 K の 1 台のプロセッサ要素の演算処理能力はマイクロプロセッサ MC 68000 (10 MHz) のものであり、簡単なベンチマークを使って実測した結果は、整数演算で 0.8 MIPS, 実数演算で 0.006 MFLOPS である。5.3 節で述べるように並列処理効率は問題によって、また使用する並列処理アルゴリズムによってかわるからシステム全体の処理能力を一概に決めることはできないが、プロセッサ利用率を 80% と仮定すれば 63 台のプロセッサ要素では 40 MIPS, 0.3 MFLOPS ということになる。同じベンチマークを大型汎用計算機 FACOM M 360 で実行して測定した値と比較すると表 5 のようになる。

#### 5.2 プロセッサ間転送速度

プロセッサ要素間のデータ転送速度を実測した結果は図 11 のようになる。データブロックが 1 KB より大きいときには 2 M バイト/秒の転送速度が得られるが、これより小さいと転送速度は遅くなる。これは転送語数に無関係に DMA コントローラの設定にかかる時間がかなり大きいため、Coral 68 K のユーザプログラムはできるだ

表 5 演算処理能力  
Table 5 Processing powers.

| 機 種         | MIPS 値 | MFLOPS 値 |
|-------------|--------|----------|
| プロセッサ要素 1 台 | 0.8    | 0.006    |
| Coral 68 K  | 40     | 0.3      |
| FACOM M 360 | 2.67   | 0.64     |

け転送するデータのブロックが大きくなるように書く必要がある。ホスト-ルートプロセッサ間の転送速度も同じような傾向になるが、256 バイト以上のデータブロックでは 37 KB/秒の転送速度が得られる。

#### 5.3 並列処理効率

いくつかの並列処理プログラムについて使用するプロセッサ要素台数を変えて計算時間を実測し、Coral 68K の並列処理効率を測定した。また同時に一つの問題については大型計算機による計算時間と比較することによって計算能力の評価も行った。

並列処理効率は個々のプロセッサ要素の処理能力を複数台のプロセッサ要素を同時に働かせることによってどれだけ利用できるかを表したもので、並列計算機の評価に使用される。これは普通速度向上率とプロセッサ利用率で評価する。速度向上率は同じ問題を 1 台のプロセッサ要素で計算したときの計算時間を複数台のプロセッサ要素で並列に処理したときの計算時間で割ったものであり、プロセッサ利用率は速度向上率を使用したプロセッサ要素の台数 (すなわち並列度) で

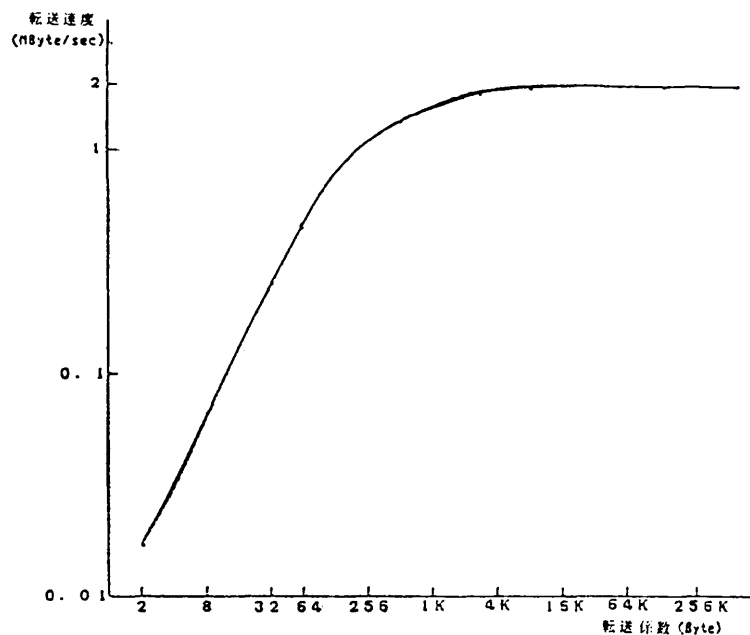


図 11 プロセッサ要素間転送速度  
Fig. 11 Interprocessor communication speed.

割ったものである。もちろん並列処理効率は並列プログラムのアルゴリズムによって左右されるが、ここでは Coral 68K で処理する場合に最も効果の上がると考えられるアルゴリズムを使用して測定を行った。

#### (1) ナップザック問題

これはそれぞれ異なる価値と重量を持つ  $m$  個の品物の中からナップザックの制限容量  $g$  以内で総価値が最大になるような組合せを選ぶ問題である<sup>11)</sup>。この問題の並列処理は最初の  $k$  個の品物のすべての組合せを  $n$  台のプロセッサ要素に等分割し、各プロセッサ要素が残りの  $(m-k)$  個の品物の組合せを探索する方式を取る。各プロセッサ要素では得られた解を LEFT, RIGHT 方向から送られてくる解と比較し、総価値の大きいものを TOP 方向に送る。 $m=22, g=700$  とし、品物の重量を 100 以下の乱数で与えたときに整数で計算した場合と実数で計算した場合の結果を表 6 に示した。この問題では負荷の分配が均等に行われ、プロセッサ要素間の通信も少ないので非常によい並列処理効率が得られる。同じ問題の整数計算を FACOM M360 で行った結果は 293 秒であった。この問題に関しては Coral 68 K は FACOM の 40 倍速いことになる。これが表 5 の性能比よりもかなりよくなっている理由は、このような探索問題では汎用計算機の高度の命令語を使用しないのでマイクロプロセッサの性能が大型機の性能に近づくことによるものである。

#### (2) ソーティング

表 7 は 156 K 個の整数データを  $n$  台のプロセッサ要素に分配し、個別にソートした後 LEFT, RIGHT 方向から送られるデータとマージし、TOP 方向に送

る方法でソートしたものである。二進木によるソーティングは  $O(N)$  の時間でできるとされているが<sup>12)</sup>、Coral 68K で行った結果はこの表に示すようになり、並列処理効率は決してよくない。これは個別ソートは完全に並列に行われるが、マージの並列度がレベルが高くなるにつれ低下するためである。

#### (3) ガウス・ジョルダン法

Coral 68 K で  $N$  元連立一次方程式  $A \cdot X=b$  をガウス・ジョルダン法で解くには行列  $A$  と列ベクトル  $b$  を行で等分割して各プロセッサ要素に分配し、1 番のプロセッサ要素から順番にピボット行を計算し、その結果を他の全プロセッサ要素に放送して各行の計算をする方法をとる。表 8 は  $N$  を変えて並列処理効率を測定した結果であるが、 $N$  の大きいところでは 50% の利用率が得られており、かなりよい結果と言える。なお  $N$  が 189 より大きいところでは 1 台のプロセッサ要素ではメモリ容量が不足して計算できないので  $N^3$  規則で見積った時間を元にして速度向上率を求めてある。

#### (4) 巡回セールスマン問題

この問題は最小距離のハミルトン閉路を求める問題である<sup>13)</sup>。この問題を並列処理するには最初の 3 都市の巡回順を全プロセッサ要素に分割して割り当て、各プロセッサ要素は残りの都市の巡回順を探索させるようにする。そして各プロセッサ要素は最小と思われる距離を得る度にそれを近傍のプロセッサ要素に知らせ、知らされたプロセッサ要素では探索中の経路の距

表 8 ガウス・ジョルダン法の実行結果

Table 8 Measured efficiency for Gauss-Jordan method.

| 次元数 $N$   | 63 | 126 | 252 | 378  | 504  | 630   |
|-----------|----|-----|-----|------|------|-------|
| 計算時間(秒)   | 3  | 16  | 95  | 292  | 655  | 1,327 |
| 速度向上率(倍)  | 14 | 20  | 27  | 21.6 | 31.3 | 32    |
| プロセッサ利用率% | 22 | 32  | 43  | 47   | 49   | 51    |

表 9 巡回セールスマン問題の実行結果

Table 9 Measured efficiency for traveling salesman problem.

| データ | プロセッサ要素数  | 1     | 3   | 7   | 15  | 31 | 63 |
|-----|-----------|-------|-----|-----|-----|----|----|
| 1   | 計算時間(秒)   | 2,255 | 758 | 356 | 163 | 89 | 47 |
|     | プロセッサ利用率% | 100   | 99  | 90  | 92  | 84 | 76 |
| 2   | 計算時間      | 1,149 | 394 | 196 | 101 | 58 | 36 |
|     | プロセッサ利用率% | 100   | 97  | 84  | 76  | 64 | 51 |
| 3   | 計算時間      | 1,474 | 496 | 257 | 108 | 56 | 36 |
|     | プロセッサ利用率% | 100   | 99  | 82  | 91  | 85 | 65 |

表 6 ナップザック問題の実行結果

Table 6 Measured efficiency for knapsack problem.

| プロセッサ要素数  | 1     | 3   | 7   | 15  | 31 | 63 |
|-----------|-------|-----|-----|-----|----|----|
| 整数計算時間(秒) | 400   | 166 | 68  | 32  | 15 | 7  |
| 速度向上率(倍)  | 1     | 2.4 | 6   | 13  | 27 | 57 |
| プロセッサ利用率% | 100   | 80  | 86  | 87  | 87 | 90 |
| 実数計算時間(秒) | 1,800 | 791 | 322 | 151 | 74 | 36 |

表 7 並列ソートの実行結果

Table 7 Measured efficiency for parallel sorting.

| プロセッサ要素数  | 1   | 3   | 7   | 15  | 31  | 63  |
|-----------|-----|-----|-----|-----|-----|-----|
| 実行時間(秒)   | 177 | 67  | 38  | 27  | 23  | 22  |
| 速度向上率(倍)  | 1   | 2.7 | 4.7 | 6.6 | 7.7 | 8.1 |
| プロセッサ利用率% | 100 | 88  | 67  | 44  | 25  | 13  |

表 10 10 クイーン問題の実行結果  
Table 10 Measured efficiency for ten queens problem.

| プロセッサ要素数  | 1   | 2   | 4   | 8   | 16 | 32 | 63 |
|-----------|-----|-----|-----|-----|----|----|----|
| 計算時間(秒)   | 115 | 57  | 29  | 14  | 8  | 5  | 2  |
| 速度向上率(倍)  | 1   | 2   | 4   | 8   | 14 | 23 | 58 |
| プロセッサ利用率% | 100 | 100 | 100 | 100 | 90 | 72 | 91 |

離がこの値を越えるとその経路の探索は打ち切るようにする。10都市の場合の実行結果を表9に示す。この問題は1台のプロセッサ要素で計算してもデータによって実行時間がかなり変わる。したがって実測した並列処理効率の値もデータによる変動が大きい。表9はいくつかの計算例についての結果を示したものである。

#### (5) $N$ クイーン問題

チェス盤の上に8個のクイーンをお互いに当たらないように並べる配置の数を求める問題はエイトクイーン問題とよばれ<sup>13)</sup>、この問題を  $N \times N$  目の盤に  $N$  個のクイーンを並べる問題にしたのが  $N$  クイーン問題である。これを並列処理するには最初の3列のクイーンの配置を分割して各プロセッサ要素に割り当て、残りの列について各プロセッサ要素が独立に探索し、求められた解の数をTOP方向に送って集計する方法をとる。 $N=10$ の場合の実行結果を表10に示す。この場合の並列処理効率は非常によい。

## 6. 結 論

単純木構造の実規模の並列計算機 Coral 68 K を開発し、これを用いて各種の並列計算を行った結果から次のような結論が得られた。

(1) 二進木マシンの構造は結合方式が簡単であるために実装が非常に容易であり、数千台のプロセッサ要素でも製作することは可能であると考えられる。

(2) 階層構造であるためにナップザック問題、 $N$  クイーン問題などの探索問題ではよい並列処理効率を得られる。ただし巡回セールスマン問題のような探索の深さがデータによって大幅に変動するような問題では処理効率が落ちることがあるので注意しなければならない。

(3) 二進木構造に適しているとされた二進木ソートなどは予想に反してよい並列処理効率は得られないことが分かった。これはたとえ問題の処理過程を示すデータフローグラフが木構造に書かれていても、時間的に並列に実行できるようなデータが常に存在するのではなくては並列処理の効果が上らないことによる。

(4) 逆に二進木構造に向かないと一般に考えられている問題でもかなりよい結果が得られることがある。Coral '83 で行った偏微分方程式の Odd-Even SOR 法による並列処理では格子構造の問題を二進木にマッピングすることによってよい処理効率を得られている<sup>14)</sup>。これは平均プロセッサ間通信距離が短いという二進木結合の効果によるものである。

(5) 多数の問題の並列処理プログラムを開発した結果、ハイパーキューブや、トラスマシンと違ってデータの転送方向が3方向しかないために比較的プログラミングが容易であることが分かった。

(6) 二進木構造は任意のプロセッサ要素間に複数の通信路がないために信頼性が劣ることが指摘されているが、最近の電子部品の信頼性は高く、実際に Coral '83, Coral 68K では全く故障なく動作している。

(7) 二進木マシンではプロセッサ要素間の中継量にアンバランスが起こることが指摘されており、DON ではダブルトリー構造を取り、また NON-VON はリーフプロセッサを格子結合しているが、そのために構造が複雑になり、WSI 化に適しているという利点をかえって殺してしまっている。ナップザック問題のように中継量のアンバランスが問題にならない場合が多いのでこの点は余り心配する必要はないのではないかと考える。

(8) Coral 68 K ではプロセッサ間接続ポートに DMA コントローラを採用したために大量データの転送は非常に高速であるが、小さなデータの転送速度が遅い。したがって小さいデータを繰り返して転送するのは得策でなく、たくさんのデータをまとめて転送するようなプログラムを書くようにする必要がある。

以上のように二進木マシンの特長が実証された。この二進木マシンの特長を易化して数千数万の高性能のプロセッサ要素を単純木で接続すれば大規模な問題の処理に適用できるスーパーコンピュータが実現できよう。

謝辞 終わりに当たって Coral 68 K のプロセッサボードと筐体の設計製作に多大なご協力と助言を頂いたアジアエレクトロニクス株式会社に厚く感謝する。また、プロジェクトの初期にブレッドボードでの実験で苦勞された藤本和生、桑原昭、山本英彦の諸君、プロセッサボードのプリント基板化のための設計を担当した吉谷文徳君、筆者らとプロトタイプの実験を行った白方新洋君、今回の表示パネル回路の製作と評価プログラムを作成した須原康次、藤本英樹、光山正仁、

森和夫君らの卒業生諸君、および電源部分を担当された井上富夫技官に深い謝意を表する。

### 参 考 文 献

- 1) Takahashi, Y., Yamane, Y., Nishiyama, K., Yoshitani, F. and Inoue, K.: Efficiency of Parallel Computation on the Binary-Tree Machine CORAL '83, *J. Inf. Process.*, Vol. 8, No. 4, pp. 288-299 (1986).
- 2) 高橋義造: 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, pp. 201-209 (1982).
- 3) Takahashi, Y.: Fault Tolerance in Processor Networks, 第25回情報処理学会全国大会論文集, 1F-6 (1982).
- 4) 末吉俊則, 最所圭三, 有田五次郎: 階層構造高多重並列計算機実験システム HYPHEN C-16 について, 情報処理学会論文誌, Vol. 25, No. 5, pp. 813-822 (1984).
- 5) Imai, M.: A Double-Tree Structured Multi-computer System and Its Application to Combinatorial Problems, *Trans. IECE*, Vol. E 69, No. 9, pp. 1002-1010 (1986).
- 6) 小池誠彦: 超並列マシン, 情報処理, Vol. 28, No. 1, pp. 94-105 (1987).
- 7) Stolfo, S. J.: Initial Performance of the DADO 2 Prototype, *Computer*, Vol. 20, No. 1, pp. 75-87 (1987).
- 8) 藤本和生, 桑原 昭, 高橋義造: 68000 による 2 進木構造並列処理システム Coral プロトタイプ的设计, 第29回情報処理学会全国大会論文集, 5B-2 (1984).
- 9) 松尾賢二, 遠藤俊雄, 吉谷文徳, 高橋義造: 2 進木構造並列処理システム Coral 68K のプロセッサ要素の開発, 第33回情報処理学会全国大会論文集, 3C-3 (1986).
- 10) 遠藤俊雄, 松尾賢二, 白方新洋, 槌谷 一, 高橋義造: 2 進木マシン Coral 68K のシステム構成と性能評価, 情報研報, Vol. 87, No. 45, 87-CA-66-5 (1987).
- 11) 西川, 三宮, 茨木: 最適化 (岩波講座情報科学 19), 岩波書店, 東京 (1982).
- 12) Meads, C., Conway, L., 菅野, 櫛沢: 超 LSI システム入門, pp. 324-327, 培風館, 東京 (1981).
- 13) Wirth, N.: *Algorithms + Data = Programs*, pp. 143-147, Prentice-Hall, New Jersey (1976).
- 14) 吉谷文徳, 高橋義造: 2 進木マシン CORAL に

よる 偏微分方程式の並列処理, 第31回情報処理学会全国大会論文集, 3D-4 (1985).

(昭和63年4月8日受付)

(昭和63年11月14日採録)



高橋 義造 (正会員)

昭和7年生。昭和30年京都大学工学部電気工学科卒業。同年(株)東芝入社。自動制御, アナログ計算機, 計算機制御, 計算機の基本ソフトウェア, 各種オンラインシステム, 複合計算機等の研究開発に従事。昭和52年より徳島大学工学部情報工学科教授。並列計算機のアーキテクチャと計算アルゴリズムに興味をもって研究している。著書, 「制御工学」「電子計算機演習」(以上朝倉書店), 「計算機方式」(コロナ社)等。工学博士。電気学会, ACM 各会員。



遠藤 俊雄 (正会員)

昭和38年生。昭和61年徳島大学工学部情報工学科卒業。昭和63年同大学院修士課程修了。工学修士。同年(株)東芝入社。同社府中工場官公需制御システム部勤務。ビル管理等の計算機制御システムの設計に従事。電子情報通信学会会員。



松尾 賢二 (正会員)

昭和38年生。昭和61年徳島大学工学部情報工学科卒業。昭和63年同大学院修士課程修了。同年日本IBM(株)入社。情報機器開発統括本部勤務。金融システムのソフトウェア開発に従事。



槌谷 一 (学生会員)

昭和39年生。昭和62年徳島大学工学部情報工学科卒業。同年同大学院修士課程進学。関数型プログラムの並列処理方式の研究に従事。