設 計 自 動 化 70－2
（1994．2．4）

連想プロセッサ・システムの構成とアプリケーションの実装

中村恵介　木村功　佐藤政生　大附辰夫

早稲田大学　理工学部

あらまし

機能メモリの一種である連想メモリ (CAM:Content Addressable Memory) は，外部から与えられたデータに対し，その一部分が一致するデータを検索することが可能で，しかもその処理時間は格納データ数によらず一定である．この連想メモリを搭載しEWS(Engineering Work Station) と接続することによりスレーブプロセッサとして動作するハードウェア・エンジン "CHARGE II" (CAM-based Hardware Engine for Geometrical Problems) を試作した．本稿ではCHARGE IIとその制御を行うハードウェア，及びソフトウェアツールから構成される連想プロセッサ・システムについて報告する．また，アプリケーションとして多層配線プログラムを取り上げ，CHARGE IIに実装することにより，汎用計算機上に実装したソフトウェアとの比較を行う．

A CAM Based Hardware System and its Application Implementation

Keisuke NAKAMURA, Isao KIMURA, Masao SATO and Tatsuo OHTSUKI

School of Science and Engineering, Waseda University
3-4-1 Ohkubo, Shinjuku-ku, Tokyo 169

Abstract

CAM(Content Addressable Memory), which is elsewhere called associative memory, receives an index data and a mask data as the input. CAM searches all the stored data simultaneously for matching between non-masked bits of the index data and the corresponding bits of the stored data. A hardware engine based on CAM is connected to an engineering work station as a slave processor. An integrated system embedding the CAM-based engine together with backup and application tools is proposed. This system consists of the hardware engine "CHARGE II," control hardware and several software tools. As an application, a wiring program is implemented on it and compared with software implementation.

1 はじめに

LSI 加工技術の微細化に伴い，レイアウト設計において扱う図形情報が大規模になると，ソフトウェアによる CAD では実用的な処理速度が得られなくなるなどの問題が生ずる．従って，ソフトウェアの持つ柔軟性を損なうことなく，ハードウェアを用いた並列処理を導入し，処理速度の改善を実現することが重要となっている．

半導体メモリの分野では，3 年で 4 倍という集積度の向上が達成されているが，この結果を直接並列度の向上に反映させることが可能な並列計算機構に連想メモリ (CAM:Content Addressable Memory) がある [1]．連想メモリは RAM の機能に加え，各ワードに演算機能を付加した機能メモリで，ワード数に相当する並列性を持っている．連想メモリの基本機能に，一致検索がある．これは，外部から与えられたデータに対し，その一部分が一致するデータを，格納データ中から高速に検索する機能であり，検索された結果は，各ワードに付加されたタグを設定することによって区別される．一致検索では，格納データ数によらない一定時間で検索が終了する．従って，レイアウト設計で多用される図形探索処理における座標検索を CAM で行えば，データ数 n に対し，ソフトウェアで $O(\log n)$ 程度の手間を必要とする探索問題を $O(1)$ の手間で解くことができる [2]．

我々は NTT で開発された 4kbit CAM LSI[3] を用いて，図形処理用ハードウェア・エンジン CHARGE (CAM-based Hardware Engine for Geometrical Problems) を試作し，その性能評価を行ってきた [2][4]．しかしこの CAM は Prolog マシンへの応用を前提として開発されたものであったため，我々はさらに図形処理に適した，高機能 CAM LSI を開発した [5]．この CAM は一致検索の他に，メモリ内の格納データと，外部から与えられたデータの一部分とを比較して，大きい／小さいワードを検索するしきい値検索や，メモリ内の全格納データの中で一部分が最大／最小のワードを検索する極値検索を備えている．これらの検索は一致検索を組み合わせれば実現できるが，この CAM では各検索の処理過程を LSI 内のハードウェアで実現しており，CAM を制御するソフトウェア側の負担が軽減されるだけでなく，検索に要するクロック数も削減されている．

この CAM LSI を用いて我々は新たに図形処理用ハードウェア・エンジン CHARGE II を製作し，実験を行ってきた [6]．このハードウェア・エンジンはシーケンサを持ち，RAM に書かれた 80 ビットのマイクロコード (マイクロプログラム) に従って処理を行う．ホスト計算機として CHARGE はパソコンに接続されていたが，CHARGE II は EWS (Sun4) に接続されており，データ処理やプログラム開発が容易になった．さらに我々は CHARGE II の制御，プログラムのデバッグ等を行うためのハードウェア，及びマイクロコード・アセンブラやシミュレータなどのソフトウェアを製作し，操作性を向上させるとともに，プログラム開発をより柔軟に行うことを目指した連想プロセッサ・システムを構築した．

本稿では，連想プロセッサ・システムの概要と，アプリケーションの実装を報告する．まず，我々の開発した CAM の機能，及びこの CAM LSI を搭載したハードウェア・エンジンについて述べ，連想プロセッサ・システムの構成を示す．また，アプリケーションとして，配線手法の一つである多層化線分展開法 [7] を CHARGE II に実装し，汎用計算機上に実装したソフトウェアとの比較実験を行う．

2 レイアウト設計における図形処理

VLSI のレイアウト設計における素子間配線や設計規則検証においては，素子を図形と見なして図形間の幾何学的関係を扱う処理が多用される．以下に代表的な 2 次元の図形処理を挙げる．

(A) 線分探索処理

平面上に与えられた x 軸に平行な横線分集合 $S = \{s_1, s_2, \ldots s_n\}$ と質問点 $p(x_0, y_0)$ に対し，質問点 p から直線を y軸の正方向に延ばしたとき，交差する線分の中で質問点 p に最も近い線分を報告する (図 1(a))．この処理は全ての横線分 s_i $(i = 1, 2, \ldots n)$ に対して，両端点の座標を $(x_i^{(\mathrm{left})}, y_i)$, $(x_i^{(\mathrm{right})}, y_i)$ (ただし $x_i^{(\mathrm{left})} < x_i^{(\mathrm{right})}$) としたとき，

$$\begin{array}{ccccc} x_i^{(\mathrm{left})} & < & x_0 & < & x_i^{(\mathrm{right})} \\ & & y_0 & < & y_i \end{array}$$

を満たす線分 s_i の中で y_i が最小であるものを報告する．

(B) 領域探索処理

平面上に与えられた長方形集合 $R = \{r_1, r_2, \ldots r_n\}$ と質問長方形 $q(x_0^{(\mathrm{left})}, x_0^{(\mathrm{right})}, y_0^{(\mathrm{bottom})}, y_0^{(\mathrm{top})})$ (ただし $x_0^{(\mathrm{left})} < x_0^{(\mathrm{right})}$, $y_0^{(\mathrm{bottom})} < y_0^{(\mathrm{top})}$) に対し，質問長方形 q と交差する長方形 (q に含まれる長方形も含む) をすべて報告する (図 1(b))．全ての長方形 r_i $(i = 1, 2, \ldots n)$ に対して，それぞれの座標を $(x_i^{(\mathrm{left})}, x_i^{(\mathrm{right})}, y_i^{(\mathrm{bottom})}, y_i^{(\mathrm{top})})$ (ただし $x_i^{(\mathrm{left})} < x_i^{(\mathrm{right})}$, $y_i^{(\mathrm{bottom})} < y_i^{(\mathrm{top})}$) としたとき，

$$\begin{array}{ccccc} x_i^{(\mathrm{left})} & < & x_0^{(\mathrm{right})}, & x_0^{(\mathrm{left})} & < & x_i^{(\mathrm{right})} \\ y_i^{(\mathrm{bottom})} & < & y_0^{(\mathrm{top})}, & y_0^{(\mathrm{bottom})} & < & y_i^{(\mathrm{top})} \end{array}$$

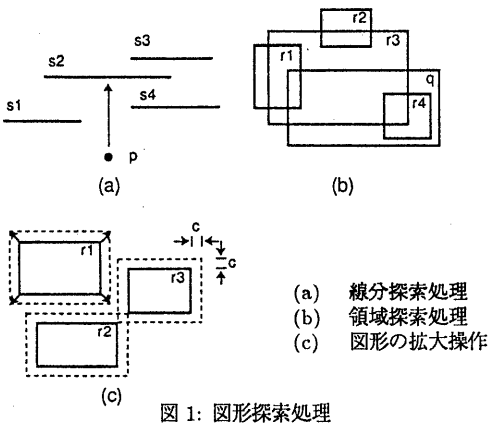

図 1: 図形探索処理

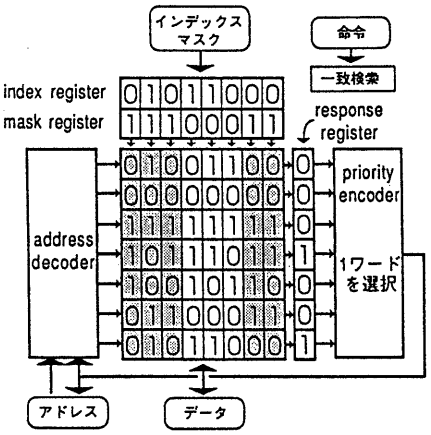

図 2: CAM の構造と一致検索

を満たす長方形 r_i を報告すればよい.

(C) 拡大・縮小操作

平面上に与えられた長方形集合 $R = \{ r_1, r_2, \ldots r_n \}$ をすべて c だけ拡大する (図 1(c)). 領域探索と同様に全ての長方形 r_i $(i = 1, 2, \ldots n)$ に対して, $x_i^{(\mathrm{left})}$, $y_i^{(\mathrm{bottom})}$ からは c を引き, $x_i^{(\mathrm{right})}$, $y_i^{(\mathrm{top})}$ には c を加えればよい.

3 連想メモリの機能

我々の開発した CAM LSI は 2 節で示したような図形処理における, 座標間の比較演算, 算術演算等を全格納データに関して並列に行うことができる [5].

3.1 連想メモリの基本機能

図 2に連想メモリの基本構造を示す. メモリアレイの各ワードには RAM と同様, 固有のアドレスが与えられており, 外部からアクセスすることが可能である. これとは別に, 外部から与えたデータの一部分が一致するワードを検索する一致検索機能を持っている. 検索結果は, 各ワードに附属する 1 ビットのレスポンスレジスタ (RR) に格納され, この RR 値を用いて, アクセスするワードを選択することもできる. 一致検索においては, インデクスレジスタに検索データ, マスクレジスタに検索対象となるビットを設定する. 検索命令を実行すると, マスクされない部分が検索データと一致する全てのワードの RR に 1 が格納される. この検索に要する時間はメモリアレイのワード数によらず一定である.

また, 一致検索を繰り返すことにより, メモリ内の格納データと, 外部から与えられたデータの一部分とを比較して, 大きい／小さいワードを検索するしきい値検索や, メモリ内の全格納データの中で一部分が最大／最小のワードを検索する極値検索が実現できる [5][6]. これらの検索に要する時間はメモリアレイのワード数によらず, しきい値検索では検索データの 0 または 1 のビット数, 極値検索では検索データのビット数に比例する.

検索結果が複数存在する場合, プライオリティ・エンコーダ (PE) を用いて 1 ワードを選択することが可能である. PE の出力するアドレスを用いて, 検索されたワードにアクセスすればよい.

3.2 CAM LSI の構成

図 3に CAM LSI のブロック図を示す. 本 LSI は 36 ビット×128 ワードの連想メモリセルを持ち, 36 ビットのデータバス (DM), 及び RAM のアドレスバスに相当する 7 ビットのアドレスバス (ADR) を備えている. また検索データや各種マスクを与えるための 36 ビットのインデクスバス (INDEX), 及び CAM の動作命令を与えるための 20 ビットの命令入力 (CMD) を持つ. さらにクロック入力 (CLK) を持ち, 外部クロックに同期して動作する.

メモリセルアレイ メモリセルはインデクスレジスタとの一致判定を行う EXOR ゲートを持つ (図 3). この出力から一致検索結果が生成され, ワードごとに検索線として演算ユニットに送られる.

演算ユニットアレイ メモリセルアレイの各ワードはそれぞれ 1 個の演算ユニットを持つ. 各演算ユニット内にはレスポンスレジスタ (RR) が 2 個あり, 独立に機能する. また, 1 ビット ALU を持ち, 検索結果に対し, RR 値との論理演算または算術演算を行ってから RR に格納することが可能である.

プライオリティ・エンコーダ RR 値が 1 となるワードが複数存在する場合, ワードのアドレスによって決まる

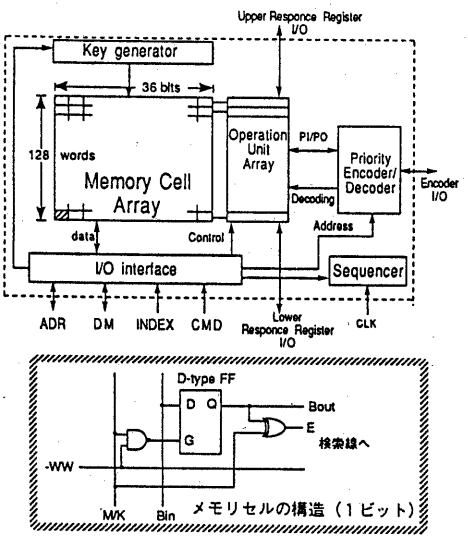

図 3: CAM LSI のブロック図
とメモリセル

優先度に従い，その中から 1 ワードを選択する．

キー算出回路　しきい値検索，極値検索をオンチップで実現するため，与えられた検索キーデータとマスクデータからインデクスレジスタ値とマスクレジスタ値を算出する回路である．キー算出回路の出力を用いて，LSI 内のシーケンサが一連の一致検索を実行し，これらの検索が自動的に行われる．

3.3 CAM LSI の持つ諸機能

(A) 一致検索　INDEX バスに検索データ，DM バスにマスクデータを与え，CMD バスに命令を与える．CMD には，命令コード，ALU 関数，検索結果を格納する RR 等を指定する．これらのデータを取り込んだ後，1 クロックで検索結果が目的の RR に格納される．

(B) しきい値検索　一致検索と同様に INDEX バスに検索キーデータ，DM バスにマスクデータを与え，CMD バスに命令を与える．これらのデータを取り込んだ後，CAM は自走して一致検索を繰り返す．1 + 0.5w クロック (w は検索キーデータの有効ビット数) 以下で，マスクされない部分が検索キーデータよりも大きい／小さいワードが全て選択され，RR に 1 が格納される．

(C) 極値検索　DM バスにマスクデータを与え，CMD バスに命令を与える．これらのデータを取り込んだ後，しきい値検索と同様に CAM は自走して一致検索を繰り返す．3 + 2w クロック (w は極値検索を行う有効ビット数) 以下で，全ワードにおけるマスクされない部分の最大値／最小値が内部のインデクスレジスタに求められた後，自動的に一致検索が 1 回行われて，解となるワードの RR に 1 が格納される．しきい値検索，極値検索においては，検索中，Busy 端子が 1 となり，この間外部からのアクセスはできない．

4 連想プロセッサ・システムの構成

本システムのハードウェアは，上述の CAM を搭載したハードウェア・エンジン CHARGE II と制御ボード，及びそれぞれが接続されている EWS (Sun4) から構成される．また，ソフトウェア・ツールとして，デバッガ，シミュレータ，マイクロコード・アセンブラなどがある (図 4)．

4.1 ハードウェア・エンジン CHARGE II の構成

CAM LSI は連続して命令を与えることによりパイプライン式に高速実行が可能であるため，CHARGE II はプロセッサとしてシーケンサ LSI を持ち，RAM(M) に書かれた 80 ビットのマイクロコードに従って自走する VLIW(very long instruction word) 型アーキテクチャを用いている．CAM にはインデクスとマスクを同時に与えるため，36 ビット×2 バス構成とし，これらのバスには，64 個のレジスタを持つレジスタファイル LSI と 64k ワード RAM(D) から任意にデータを出力することが可能である．以下では，シーケンサ，CAM, RAM などをモジュールと呼ぶ．

CAM　本ハードウェア・エンジンには現在，我々が開発した CAM LSI が 16 個実装されており，36 ビット×2,048 ワードが利用可能である．

マイクロコード RAM(M)　80 ビット×32k ワードの RAM(M) に書かれたマイクロコードは，表 1のように

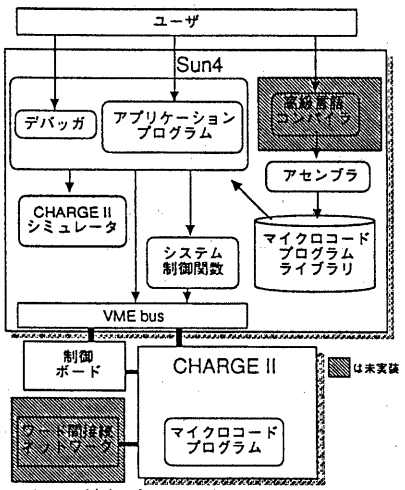

図 4: 連想プロセッサ・システム

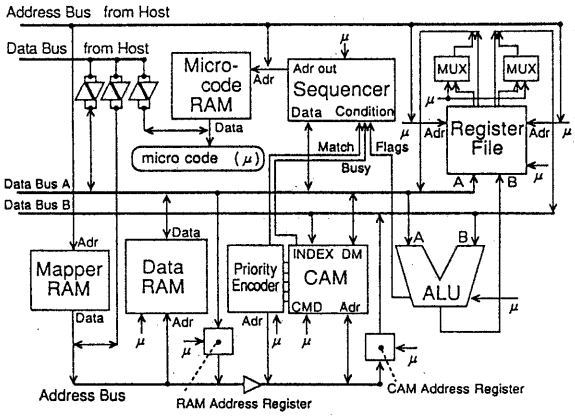

図 5: CHARGE II ブロック図

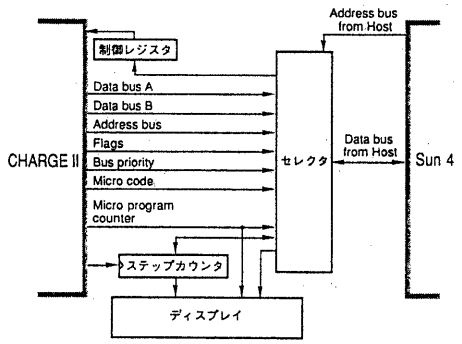

図 6: 制御ボードのブロック図

全モジュールを制御する．RAM(M) のアドレスはシーケンサ LSI によって制御される．

表 1: マイクロコード 80 ビットの制御先モジュール

20-bit ... CAM	20-bit ... Register File
26-bit ... シーケンサ	1-bit ... RAM(D)
9-bit ... ALU	4-bit ... その他

シーケンサ　　マイクロコードから与えられる命令及び ALU や CAM からの条件フラグを読み込み，次に実行すべきマイクロコードが格納されている，RAM(M) のアドレスを決定する．

レジスタファイル　　2 ポートの入出力を持つ 64 個のレジスタを備えた汎用 LSI である．両ポートには異なるデータの出力が可能で，CAM の入力データを生成する．

データ RAM(D)　　CAM と共通のアドレスバスを持ち，CAM の補助記憶として用いられる．

マッパ　　ホストに CAM, RAM(D) の内容をマッピングする際に，アドレステーブルとして用いられる RAM で，予めホストから CAM, RAM(D) のアドレスを書き込んでおく．マッパは CHARGE II が自走している間は用いられない．

4.2 制御ボードの構成

CHARGE II 単体の実行を外部から観測する働きをし，全バスの値，条件フラグ，各メモリ値の取得が可能である．また，CHARGE II の動作を制御することも可能で，マイクロコードのデバッグに用いられる (図 6)．本ボードの制御はホスト (Sun4) が直接行う．

セレクタ　　CHARGE II の各所からバス値，条件フラグ，各種信号線の値を取り出し，ホスト (Sun4) に送るためのセレクタ (バッファ) である．制御レジスタ，ステップカウンタ，ディスプレイに対しては双方向のバッファとしてはたらく．

制御レジスタ　　ホストから状態を書き込むことにより，システムの動作を制御する．

ステップカウンタ　　CHARGE II のクロック数をカウントする．カウンタを制御することにより，マイクロプログラムの特定ルーチンが要したクロック数を計測することが可能である．

ディスプレイ　　システムの状態をリアルタイムで表示する LCD や LED を持ち，ホストによって制御される．

4.3 ホスト計算機 (Sun4) とのインターフェース

CHARGE II 及び制御ボードはインターフェースボードを通して，Sun4 の VME バスに接続されており，EWS による直接アクセスが可能である．各モジュールは VME アドレス空間上へ図 7のように接続されている．このアドレス空間を Sun4 の仮想記憶へマッピングすることにより，Sun4 上のソフトウェアから自由にアクセスが可能となる．また，データ RAM(D), CAM のマッピング状態はマッパにより自由に設定可能である．

マッパは 64k ワードからなる RAM で，ホストから予め RAM(D) や CAM のアドレスを書き込んでおく．VME ア

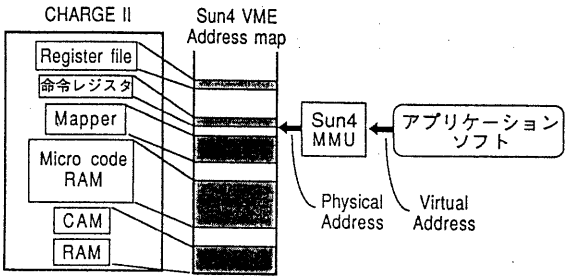

図 7: VME アドレス空間へのマッピング状態

ドレス空間のRAM(D)やCAMの部分がアクセスされたときに，マッパはRAM(D),CAMへアドレスを出力し，アドレス変換を行う．

4.4 ハードウェア・エンジンの実行方法

CHARGE IIの動作は，ホストが，図7の制御ワードに適当な値を書き込むことにより制御される．CHARGE IIは待機(Host mode)と実行(Local mode)の両状態をスイッチし，実行状態においては，制御レジスタ値により高速実行またはステップ実行を行う(図8)．ホストがCHARGE IIの各モジュールにアクセスできるのは待機状態のみで，この間にマイクロプログラムの転送，マッパ設定や，レジスタ，RAM(M),CAMへのデータ転送等を行う．実行状態では，制御ボードを通して実行状況，信号値の観測が可能である．ホストが起動命令を与えると，CHARGE IIのシーケンサはマイクロプログラムに従って処理を制御する．実行中はホストへBusyフラグが送られる．マイクロプログラムが終了番地に達すると，待機状態に戻る．ステップ実行ではホストがクロックを制御し，1ステップずつ処理を行う．ステップ実行を利用することにより，マイクロコードのデバッグ等が可能である．

ホスト上のアプリケーションはCHARGE IIに対し，必要に応じてデータの転送，及び起動を行う．CHARGE IIの処理が終了し，待機状態に戻った時点で，結果の格納されているモジュールから目的のデータを読み出せばよい．

4.5 マイクロコード・アセンブラ

マイクロプログラムの開発を容易にするためアセンブリ言語を作成した．この言語で記述されたファイルからアセンブラによって，80ビットのマイクロコードが生成される．

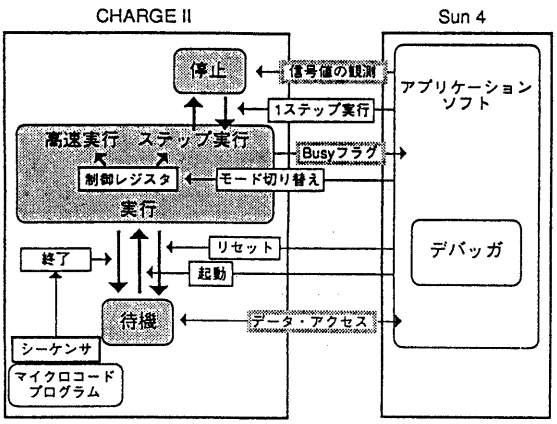

図 8: CHARGE II の状態遷移

以下にそれぞれの特徴をまとめる．

本アセンブリ言語の特徴

- 命令はマイクロコードの数ビットと1対1に対応．
- マイクロコードをモジュールごとの命令に分割．
- 複数のモジュールを制御する複合命令を導入．
- 命令 = 命令語 + 複数の引き数．
  (例) `SRCH_EQ rr=1 and_rr rr`
- 各モジュールごとに異なる命令語を使用．
- 命令語数を削減するとともに，
  引き数によりきめ細かい指定が可能．

各モジュールごとの命令語数

CAM.....19	Register File.....3
シーケンサ.....17	RAM(D).....3
ALU.....68	その他.....4

マイクロコード・アセンブラの特徴

- マイクロコード1ワードの命令は1行に並記．
  (例) `JMP Z LAB1; ADD r3 r2; CWRITE emp da`
- 処理に参加しないモジュールの命令は不要．
- プリプロセッサを用いたマクロが記述可能．
- 各モジュールに対する初期データを生成．
- バスの衝突，命令の重畳などの警告を出力．

4.6 CHARGE II シミュレータ

本ハードウェア・エンジンの機能を全く等しくシミュレートするソフトウェアである．実際のハードウェアに比べ，各所の信号状態を詳細に観測することができる．各メモリ内容やバス値のダンプ，クロック数のカウント等を行うことが可能である．ホスト上のアプリケーションプログラムを開発する場合のデバッグ等に用いられる．

4.7 マイクロコード デバッガ

マイクロコードのステップ実行や，各所の信号値の観測等をインタラクティブに行うソフトウェアで，マイクロプログラムのデバッグに用いられる．実際のハードウェアによる実行と，シミュレータを用いたソフトウェアによる実行が可能である．本デバッガはホスト上で動作し，CHARGE IIや制御ボード，またはシミュレータを呼び出し，処理を行う．以下にデバッガの持つ機能と働きを挙げる(図9)．

デバッガ制御　実行モード(ステップ実行，高速実行)の切り替えを行い，ブレーク・ポイントの設定，削除が可能である．また，ステップカウンタへのアクセス，カウントの制御，及びディスプレイ表示等を行う．

シーケンス制御　CHARGE IIの初期化，実行，停止を制御する．ステップ実行では，シーケンサを制御し任意ステップ数の実行，停止が可能である．

メモリアクセス　CAM, RAMなど全メモリへのアクセスが可能で，マイクロコードの書き込み，RAM(D)やレ

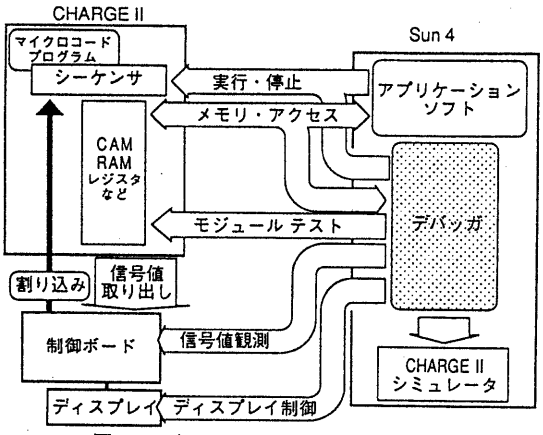

図 9: マイクロコード デバッガの機能

ジスタの初期設定を行う．またマッパに書き込む標準アドレス変換テーブルを持つ．

バス値，各種状態取得　制御レジスタをはじめとするシステム状態，各種信号値，および条件フラグ等を制御ボードから取得し，ディスプレイやホストのCRTに表示する．また，データバス，アドレスバス値，及び現在と次のマイクロプログラム実行番地等も取り込むことができる．

モジュールのテスト　各種メモリに対する読み書き，及びCAMに対する命令のテストを行う．

4.8 CHARGE II 関連関数ライブラリ

本ハードウェア・エンジンで多用される，初期設定，メモリアクセス，CHARGE IIの実行停止制御，クロック計測等をC言語の関数として準備した．ただし，高速性を必要とする部分ではこれらの関数を用いるよりも，直接ポインタを操作した方がよい．しかし，これらの関数を用いれば，ハードウェア・エンジンの細部にまで精通しなくとも，アプリケーションプログラムの開発が可能である．

5 多層配線プログラムの実装

本システムを用いたアプリケーションとして配線プログラムを実装し，ソフトウェアで実行した場合と比較する．

5.1 多層化改良線分展開法の概要

レイアウト設計において，迷路法のような格子構造を用いない配線手法(グリッドレスルータ)の一つに改良線分展開法 [8] がある．我々は，連想メモリを用いたハードウェア・エンジンで改良線分展開法を高速化する手法について報告を行ってきた [9]．今回実装した配線プログラムは，改良線分展開法を3層以上の多層配線ができるように拡張した手法 [7] に基づいている．この手法は多層の配線問題を扱うため単層に比べ，線分探索処理以外に領域探索処理が多用され，大きな処理時間を要している．

5.2 レイアウトデータ管理システム (DMS) の相違

本配線プログラムは，レイアウトデータを対象とした一種のデータベースシステムであるレイアウトデータ管理システム (DMS: Layout Data Management System)[10] を用いている．これにより，プログラム側からは図形処理の詳細は隠蔽されているため，ハードウェア・エンジンを用いたDMSを新たに作成し，ソフトウェアDMSと交換することにより，図形処理の高速化を試みた．

ハードウェア・エンジンを用いたDMS　CAMには，一つの図形データの座標を連続する複数ワードに格納し，各ワードの上位4ビット及びRAMにおける同ワードに図形の属性等を格納する．

ソフトウェアによるDMS　データ構造として高速とされるバケット [11][12] を用いて，EWS上に実装した．

5.3 ソフトウェアとの比較実験

実験は，3層からなる配線領域の1,3層に、一対の端子をランダムに発生させ，これを配線する処理を行った．配線領域には10000 × 10000 の座標を設定し，障害物として長方形をランダムに配置して配線を行う．バケットを用いたソフトウェアDMS(C言語)、及びCAMを用いたハードウェアDMSにより同じ処理を実行し、比較を行った．計算機はともにSun4/110(7 Mips, 14.28MHz) を用いた．実験結果を表2に示す．線分数は配線領域に含まれる障害物と端子の持つ線分数である．全処理時間は，図形データの読み込みから端子対を配線するまでの時間で，CAMの前処理時間は含まれていない．この前処理時間は，扱う線分数によらず一定で，今回の実験では約1秒である．

表2のCAMによるDMSにおける図形処理時間のうち，"Sun4" はホスト上のCHARGE II起動関数の実行時間である．ホストはCHARGE IIを起動後、この関数を出てしまうので，実際の図形探索処理時間は，CHARGE IIのクロック数から求められる値との和になると考えられる．

結果によれば，全処理時間で2.7 ～ 6.1倍，CAMを用いた場合が高速となっている．また、図形処理時間は、CAMの方が13倍 ～ 22倍高速となった．

また、図10にex.6.20.2 のデータに関する全処理時間の構成を示す．`dm_get_Objs` は図形探索関数、`malloc`, `free` はいずれも標準ライブラリのメモリ取得／解放関数である．

関数	CAM による DMS		ソフトウェアによる DMS	
	[sec]	(%)	[sec]	(%)
dm_get_Objs()	0.01	(0.8)	0.49	(10.7)
malloc()	0.04	(3.3)	0.77	(16.7)
free()	0.00	(0.0)	2.18	(47.4)
その他	1.18	(95.9)	1.16	(25.2)
全処理時間	1.23	(100.0)	4.60	(100.0)

ex.6.20.2 のデータより作成

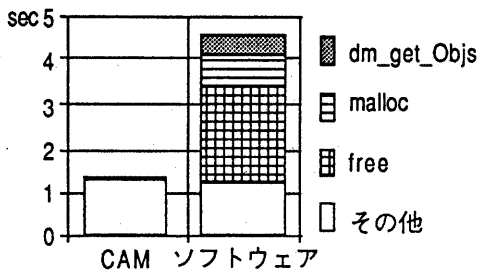

図 10: 全処理時間を構成する関数

CAM を用いることによって図形探索だけでなく、メモリ取得／解放も高速化されていることがわかる.

6 おわりに

本稿では我々の開発した連想プロセッサ・システムについて述べ, 配線プログラムを実装した結果を示した. 本システムの各ツールはハードウェア・エンジンを用いたアプリケーション開発を容易にすることを目指すとともに, ハードウェア・エンジンの操作性の改善を目的としている. また, 本システムは EWS をホストとしたことにより, 拡張性, 汎用性が向上しており, 図形処理以外の用途にも容易に応用が可能である.

現在 CAM 容量の面では実用性に劣るが, 汎用性を利用して, 図形処理以外のアプリケーションに対する実装を試みたい.

謝辞

本研究に関し、貴重な御助言を賜わりました久保田和人氏(現 東芝), 桑原泰雄氏(現 スクウェア) に感謝致します.

参考文献

[1] 安浦寛人, 渡邊章弘, 左達隆吾, 田丸啓吉:"CAM を用いた機能メモリ型並列プロセッサ上での並列アルゴリズム," 情処研報, AL21-2, (1991).

[2] 鈴木敬, 大附辰夫:"連想メモリを用いた VLSI 設計用図形処理ハードウェア," 信学論 (A), J72-A, NO.3, pp.550-560 (1989).

[3] Ogura,T., Yamada,S. and Nikaido,T.:"A 4kbit Associative Memory LSI," IEEE, J.Solid-State Circuits, Vol.SC-20, pp.1277-1282 (1985). Symp. on VLSI Circuits, Digest of Technical Papers, pp.109-110, (1990).

[4] 鈴木敬, 石和信政, 久保田和人, 大附辰夫:"図形処理プロセッサ CHARGE とその開発環境," 信学技報, CPSY88-2, pp.9-15 (1988).

[5] 桑原泰雄, 安部正秀, 久保田和人, 佐藤政生, 大附辰夫:"図形処理用連想メモリチップ," 信学技報, ICD92-54, pp.9-16 (1992).

[6] 桑原泰雄, 中村恵介, 久保田和人, 佐藤政生, 大附辰夫:"連想メモリを用いた図形処理用ハードウェアエンジン," 信学技報, CPSY92-17, pp.63-70 (1992).

[7] 石川拓也, 久保田和人, 佐藤政生, 大附辰夫:"改良線分展開法の多層化," 信学技報,VLD91-85, pp.41-48 (1991).

[8] 小島直仁, 佐藤政生, 大附辰夫:"線分展開法の改良とその評価," 情処研報,DA48-6, (1989).

[9] Sato,M., Kubota,K. and Ohtsuki,T.: "A Hardware Implementation of Glidless Routing Based on Content Addressable Memory," Proc.27th DA Conf.,pp.646-649 (1990).

[10] 小島直仁, 佐藤政生, 大附辰夫;"レイアウト・データ管理方式に関する一考察," 信学技報,VLD89-90, pp.51-58 (1989).

[11] Asano,T., Edahiro,M., Imai,H., Iri,M. and Murota,K.: "Practical Use of Bucketing Techniques in Computational Geometry," G.T.Toussaint,Ed., *Computational Geometry*, pp.153-195, North-Holland, (1985).

[12] 池田泰人, 池田栄一郎, 粟島亨, 久保田和人, 佐藤政生, 大附辰夫,"レイアウトデータ管理システム上の各種データ構造の評価," 情処研報,DA61-1 (1992).

表 2: 多層化線分展開法における処理時間の比較

データ名	線分数†1	展開回数	ソフトウェアによる DMS		CAM による DMS		
			全処理時間 [Sec]	図形探索処理時間 [Sec]	全処理時間†2 [Sec]	図形探索処理時間 (Sun4)†3 [Sec]	図形探索処理時間 (CH2)†4 [Sec]
ex.4.2.0	64	55	7.89	0.55	1.29	0.04	0.0092
ex.4.2.5	64	38	3.46	0.27	0.80	0.01	0.0062
ex.5.4.1	128	65	5.45	0.43	2.00	0.03	0.0112
ex.5.4.2	128	48	4.37	0.42	1.09	0.01	0.0084
ex.6.20.2	256	55	4.60	0.49	1.23	0.01	0.0109
ex.6.10.3	256	32	4.04	0.35	1.17	0.02	0.0069

†1 各層に含まれる図形の線分数. †2 初期設定に要する時間 (1.06 ～ 1.53 [Sec]) は含まない. †3 CHARGE II 起動関数の処理時間. †4 図形探索に要した CHARGE II のクロック数より計算 (クロック=4MHz).