計算機アーキテクチャ 110－3
設 計 自 動 化 73－3
（1995. 1. 19）

CAM搭載ハードウェアエンジンとアプリケーション実装環境

前田志門　松下章　大野慎介　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)を試作してきた．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故障シミュレーションを取り上げ，CHARGE IIに実装した結果を報告する．

A CAM-Based Hardware Engine and its Environment for Application Implementation

Shimon MAEDA, Akira MATSUSHITA, Shinsuke OHNO and Tatsuo OHTSUKI

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

Abstract

CAM(Content Addressable Memory) receives an index data and a mask data as the input, and 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 fault simulation is implemented on it and compared with software implementation.

1 はじめに

近年のLSI回路の大規模化・複雑化にともないLSI設計におけるCADアプリケーションをより効率良く高速に、さらに実用的に処理するための研究が盛んになっている．その一つの方向性として，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というアプローチがある．

連想メモリ(CAM:Content Addressable Memory)[1]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，一致検索があり，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る．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，しきい値検索や，極値検索を備えている．

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，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・システムを構築してきた．加えて多層配線プログラムをアプリケーションとして実装してきた[7]．

本稿では，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．さらにプログラム開発用ソフトウェア・ツールとして用意したマイクロコード・コンパクション等の実装環境について報告をする．また，アプリケーションとして，故障シミュレーションをCHARGE IIに実装し，その実験結果の考察を行う．

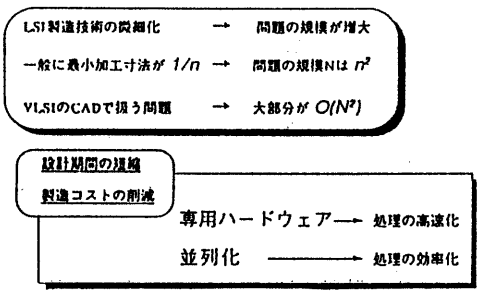

図 1: ハードウェア・エンジンの有効性

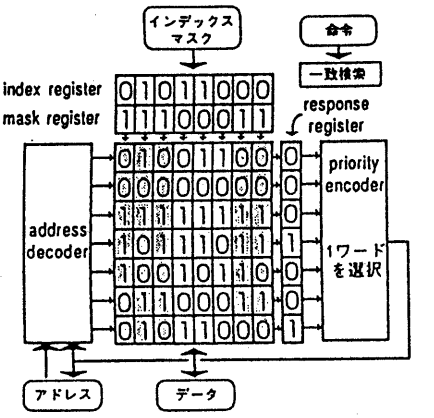

図 2: CAMの構造と一致検索の様子

2 連想メモリの機能

2.1 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ることが出来る．

2.2 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．

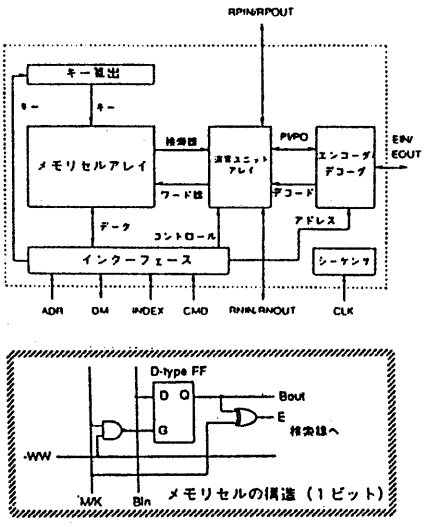

図3: CAM LSIのブロック図とメモリセル

2.3 CAM LSIの持つ諸機能

(1) 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．

(2) 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．

(3) 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.

3 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)．

3.1 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，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．

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．

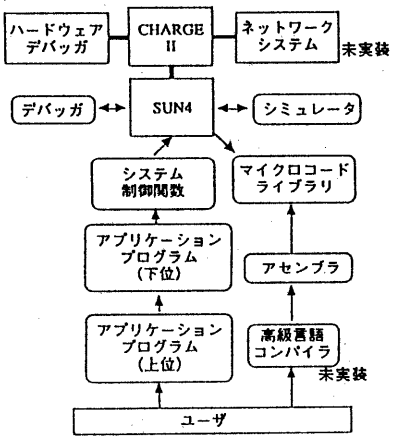

図 4: 連想プロセッサ・システム

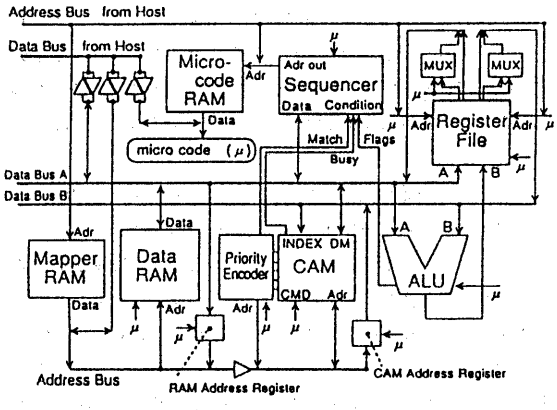

図 5: CHARGE II ブロック図

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．

3.2 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．本ボードの制御はホスト (Sun4) が直接行う．

3.3 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 6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．

3.4 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6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．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すことによって行われる．

3.5 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．

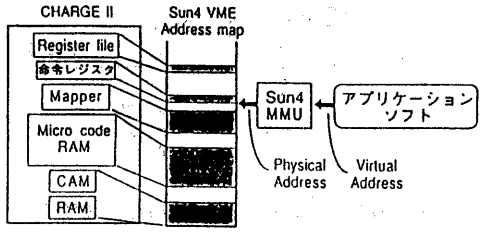

図 6: VME アドレス空間へのマッピング状態

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．

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．

3.6 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．

3.7 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 (図??)．

3.8 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．

3.9 マイクロコード・コンパクション

CHARGEIIの採用しているVLIW型アーキテクチャは並列処理システムの一つであり，マイクロプログラムにより，シーケンサ,ALU,CAM,RAMの各部分は並列に動作する．そこでプログラム開発にあたり，その並列性を加味した開発が効率性の向上に必要である．マイクロコード・コンパクションは開発環境整備の一環として，コード中の並列性を抽出して再配置するものである．これにより，連想プロセッサ・システムの並列性を意識することなくプログラムを開発することが出来る．

4 並列故障シミュレーションの実装

故障シミュレーションは，ある入力テストパタンによってどのような故障を検査することが出来るかを評価するためのシミュレーションで，主として，故障辞書作成，テストパタンの故障検出率の計算，またはテストパタン生成などのために使用される．一般に，逐次計算機上の故障シミュレーションは回路規模をL（ゲート数）として，計算複雑度が1テストパタンあたり$O(L^2)$といわれ[9]，ますます集積化するLSIに対して処理が困難になっている．本稿ではCAMの並列性に着目し,CHARGEIIに故障シミュレーションを実装することによって処理速度の向上をはかった．

4.1 特徴

故障シミュレーションの高速化をはかるためには，シミュレーションの演算スピードを上げること以上に，膨大な数の故障（回路）の処理を効率良く行なうことが重要である．我々はこの点に着目し,CAMを用いて大量の故障回路を並列に（同時に）処理している．ただし，ゲート評価（演算）は逐次計算である．また,1ワード32ビットと限定されている連想メモリに対し，1故障回路に複数ワードを割り当て，さらに工夫をこらすことで効率良くネットリストをビットに格納している（ビットの再利用）[8]．

この並列故障シミュレーションはシミュレーション・スピードが速いだけでなく，大量の故障を同時（並列）に扱うことができる．多数のテストパタンを流す中，初期段階では効率が悪いが，多くの故障を扱っているうちに初期のネックをゲインするだけの効率を上げる．

4.2 並列故障シミュレーションの実装環境

3章で述べた連想プロセッサ・システムを用いて，以下のように並列故障シミュレーションを実装した．

まず，ベンチマーク論理回路から,CAMメモリに効率良く回路のネット番号を格納するためのファイル（ビット・アロケーションファイル）を作成する．なお，1（故障）回路あたりには複数ワードをあてがう．

つぎに，上記のビット・アロケーションファイルをもとにして故障シミュレーションで用いる入力マスクデータ，出力マスクデータ，ワード選択（タグ）データを作成する．これらの値は，RAM(D)に書き込まれるよう，設定される．また，同時にゲート呼び出し，故障挿入のアセンブリ言語命令（ルーチン）を自動的に書き出すようにする．これらのデータ，ルーチンとサブルーチンを合成することによりアセンブリ・プログラムを完成する．なお，故障シミュレーション用のサブルーチンはゲート評価，並列書き込み，故障挿入，故障検出によって構成される．アセンブラにより，マイクロコードが生成される．

ここで，デバッグツールを用いて開発したマイクロプログラムの動作確認を行ない，誤記述箇所を修正する．CHARGEII関数ライブラリの中からいくつかを組み合わせて制御プログラムを作成し，シミュレータを通して開発したプログラムのデバッグを行なう．以上の経緯を経て，並列故障シミュレーションの実装が行われた．

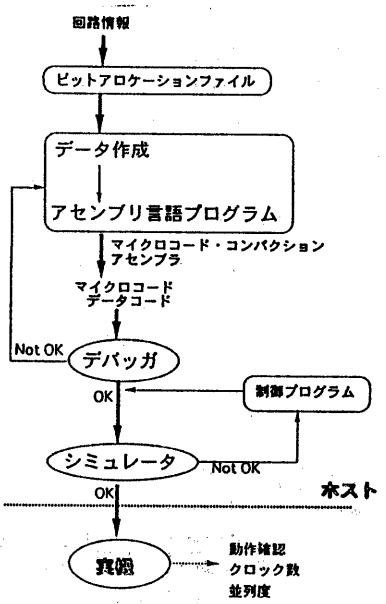

図 7: 故障シミュレーションのプログラム開発

4.3 ソフトウェアとの比較実験

本稿では,本故障シミュレーションと比較する（汎用計算機上で動く）ソフトウェア・アルゴリズムとして故障並列シミュレーションを選んだ.

選択した理由としては,まず,アルゴリズムが単純で高速性が望まれること.ビット演算を用いることで容易に並列性を実現できること.そして,同じ並列シミュレーションであることから,比較しやすいことなどが挙げられる.

両シミュレーションともベンチマーク論理回路を対象回路とし,ゼロ遅延単一縮退故障を扱うゲートレベル・シミュレーションである.使用した計算機は,本ハードウェア・シミュレーションがSun4/110(7Mips,14.28MHz)であるのに対し,ソフトウェアによるシミュレーションはSS2(Sparc-Station2)(28.5Mips,40MHz)である.

実験結果を表1,2に示す.どちらの手法とも,Fanout-Stemでの故障のみをその対象とした.実験結果によると,並列度（16個のCAMチップで同時に処理できる故障数）で1.59～64倍CAMを用いた方が上回り,処理時間では2.56～5.4倍高速となっている.図8にシミュレーション総合実行時間（1パスあたりの実行時間 × パス回数）の比較を示した.

なお,表2でもわかるとおり,シミュレーション時間の大部分がVMEアドレス空間へのマッピングで占められている.ホスト-ローカルのアクセス時間を短縮することによりシミュレーションの高速化が実現できる.

表 1: ソフトウェアによる実験結果

回路名	並列度†1	パス回数†2	実行時間／パス (s)
c17	32	1	0.04
c432	32	2	2.34
c499	32	1	4.21
c880	32	3	8.30
c1355	32	8	12.61
c1908	32	11	20.80
c2670	32	13	28.94
c3540	32	17	39.12
c5315	32	22	55.94
c6288	32	45	55.81
c7552	32	39	86.45

†1 1回のシミュレーションで同時に扱える故障数. †2 全故障を処理するために必要となるシミュレーション回数.

5 おわりに

本稿では我々の開発してきた連想プロセッサ・システムについて述べ,アプリケーションとして並列故障シミュレーションを実装し,その結果を示した.本システムのソフトウェア・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・プログラムの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アプリケーションの用途範囲やプログラム開発に柔軟性がある.我々はこれまでにCHARGEのアプリケーションとして線分探索法,線分展開法,スパース行列処理,設計規則検証を,そしてCHARGEIIのアプリケーションとして多層配線,論理シミュレーション,故障シミュレーション（本稿）を実装し,CAD問題におけるハードウェア・エンジンの有効性を実証してきた.

今後の課題として,ハードウェア・エンジンの操作性のさらなる向上を目指し,高級言語コンパイラの実装,また,ホスト-ローカル間アクセスの高速化が挙げられる.

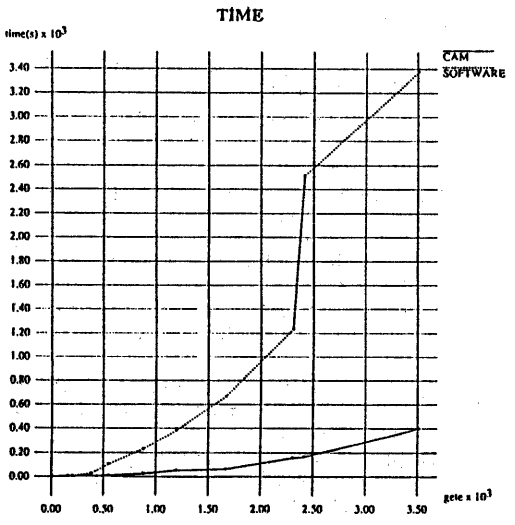

図 8: シミュレーション総合実行時間の比較

表 2: 連想プロセッサ・システムによる実験結果

回路名	ワード数†1	FS 数†2	並列度†3	パス回数	シミュレータ実行時間／パス	
					$SUN4(s)$†4	実機の実行時間 (s)
c17	2	2	1024	1	0.79	0.000162
c432	4	53	512	1	2.78	0.00387
c499	3	26	682	1	2.86	0.00470
c880	5	79	409	1	5.61	0.00866
c1355	5	226	409	1	9.23	0.0126
c1908	6	352	341	2	12.01	0.0194
c2670	13	399	157	3	16.77	0.0263
c3540	8	533	256	3	20.67	0.0368
c5315	12	683	170	5	30.88	0.0528
c6288	8	1424	256	6	27.23	0.0610
c7552	19	1223	107	12	32.96	0.0770

†1 1故障回路当たりに要する CAM のワード数. †2 Fanout-Stem 数. †3 16 個の CAM チップ（2048 ワード）で同時に処理できる故障数. †4 VME バスのマッピング所要時間および転送時間.

謝辞

本研究に関し, 貴重な御助言を賜わりました中村恵介氏（現 ソニー）に感謝致します.

参考文献

[1] 安浦寛人:"機能メモリによる超並列処理," 情報処理,Vol.32 No.12,(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] 中村恵介, 木村功, 佐藤政生, 大附辰夫:"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," 情処研報,DA70-2, (1994).

[8] Ohtsuki,T.,Kubota,K.,Kuwabara,Y.,Nakamura,K.,:"A New Content Addressable Memory Chip for VLSI Design,"Advanced Research Center For Science and Engineering,Technical Report No.93-16 (1993).

[9] 石浦菜岐佐, 矢島脩三:"連想記憶を用いた線形時間故障シミュレーション," 第4回軽井沢ワークショップ資料, pp.66-68, (1991)