設計自動化 53－6
（1990．7．10）

連想メモリを用いた線分展開法の一実装手法

石川拓也* 滝澤哲郎* 久保田和人* 佐藤政生** 大附辰夫*

* 早稲田大学理工学部
** 拓殖大学工学部

VLSIの設計期間は年々短くなってきており、その規模の増大と共に扱われるデータ量も増大の一途をたどっている。そのためCADでは処理速度の向上と共にメモリ効率の向上が望まれている。格子構造を用いない配線手法「グリッドレスルータ」はデータをベクトル表記で扱うためメモリ効率がよく、様々な手法が提案されてきている。処理速度向上を目的として、我々は連想メモリを用いた図形処理用プロセッサCHARGEを試作し、実験を行ってきた。本稿では、グリッドレスルータの一つである「改良線分展開法」を連想メモリプロセッサ上に実装し、その処理の詳細及び実験結果を示す。

A hardware implementation of line expansion algorithm based on Content Addressable Memory.

Takuya Ishikawa* Tetsuro Takizawa* Kazuto Kubota* Masao Sato** Tatsuo Ohtsuki*

* School of Science and Engineering, Waseda University
3-4-1 Ohkubo, Shinjuku-ku, Tokyo 169, Japan
** Faculty of Engineering, Takushoku University
815-1 Tatemachi, Hachioji-shi, Tokyo 193, Japan

Various gridress algorithms have been presented. These algorithms need complicated data structures to reduce computional complexity. We have proposed a CAM based hardware engine which can solve some basic geometrical search problems in constant time. In this paper, we present a hardware gridless routing algorithm.If a path exists, this algorithm finds it in O(k) time, where k is the number of searched vertices(k is less than the number of vertices of routing region). Its performance data on a prototype machine is also shown.

1．はじめに

VLSIの配線処理時に扱うデータは増加の一途をたどっている。また、VLSIの設計期間は短くなってきており、そのために設計処理の速度向上が求められている。迷路法に代表される「格子構造を用いる配線手法」では、処理が単純である反面、配線領域面積に比例したメモリを必要とするため、使用メモリの面では不利になりつつある。また複雑な設計規則の変化に対応しにくいという欠点も持っている。このような欠点を解消するために、格子構造に依存しない配線手法「グリッドレスルータ」の研究が進んでいる。グリッドレスルータは図形処理によって配線径路を求めるものであり、図形データを直接扱うので複雑な設計規則に対応しやすく、格子を用いないので使用メモリが少ない、などの特徴を持っている。また今回用いた配線手法である改良線分展開法は、配線領域を逐次探索することにより径路を求める方法のひとつで、配線領域に全体に及ぶ前処理がなく、メモリの使用効率がよい、また、個々の配線に対して各々独立した設計規則を用いることが容易である、という特徴がある。

処理速度向上を実現する手段の一つとして、専用ハードウェアを用いることが挙げられるが、我々はこの専用ハードウェアについて研究を行っている。これは、図形処理を目的としたもので連想メモリ(Content Addressable Memory,以下CAMと略す）を用いている。我々は、このハードウェアをCHARGE(CAM based hardware engine for geometrical problems)と名付け、実際に試作を行い、各種図形処理問題やグリッドレスルータの一つである改良線分探索法をCHARGEに実装し、その評価を行ってきた。CAMを用いることにより、ソフトウェアでは複雑なデータ構造を必要とする問題に対しても簡単なデータ構造にでき、また各種部分問題に対してそれぞれ別のデータ構造を用意する必要もなくなる。さらに、処理の時間複雑度も、ソフトウェアによるものに比べて低く抑えることができる。

本稿では、まず連想メモリとその動作について説明し、これを用いて図形処理問題を効率良く解く方法について述べる。つぎに我々の試作した「連想メモリを用いたハードウェアエンジンCHARGE」について説明し、最後に改良線分展開法を実装した際のアルゴリズムと、その実験結果について述べる。

2．図形処理の連想メモリによる効率化

2．1　連想メモリの構成と動作

連想メモリは機能メモリの一つであり、RAMがアドレスによりデータをアクセスするのに対し、連想メモリでは参照したいデータを示し、それが一致するデータを直接アクセスすることができる。この動作を一致検索と呼ぶ。一回の一致検索に要する時間は、記憶アレイ中のデータ数に関わらず一定である。また、RAMと同様にアドレスによってアクセスすることもできる。図1に連想メモリの構成図を示す。

以下に一致検索を行うための具体的な操作を示す。まずインデックスレジスタに検索データ、マスクレジスタにマスクデータをセットする。マスクデータは検索データの特定のビットのみを有効にするためのものである。次に一致検索を指示すると、記憶データの中に一致するものがあれば、各ワードに付随する応答レジスタにそれが記録され、さらに一致するデータの有無が一致検索線に現われる（図2）。また、一致したデータは、複数分離回路(priority encoder)により1ワードずつ順に読み出すことができる。

一致検索機能と検索結果に対する論理演算機能をもとに記憶アレイ上のデータの中から最大値、最小値、あるいは外部から示した値より大きいデータ(greater than)、小さいデータ(less than)等を検索することができる。ここで最大値・最小値検索をまとめて極値検索、greater than,less thanをまとめてしきい値検索(threshold search)という。また、これらの検索を総称して関係検索という。この関係検索もデータ数によらず一定時間で行える。

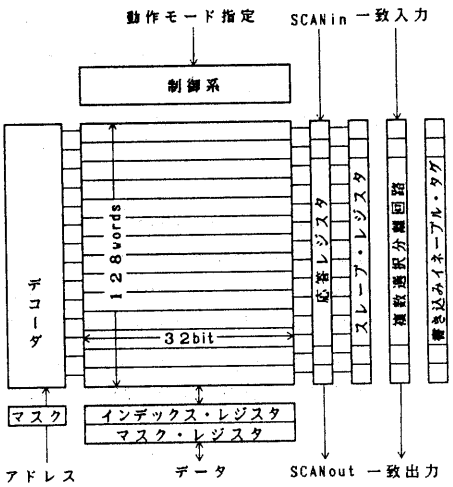


図1．　連想メモリの構成

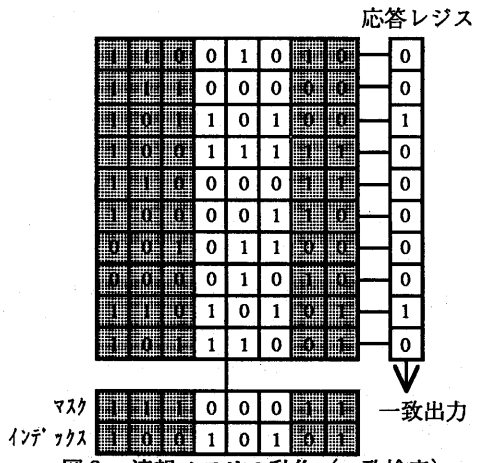


図２．連想メモリの動作（一致検索）

２．２　連想メモリの図形処理への適用

LSI設計における図形処理の特徴として、扱うデータ数が非常に膨大であり、データの挿入・検索・削除などの動的なデータ処理を必要とする操作がランダムに生じる、また図形処理において複数種類の基本的問題を解く必要があることなどが挙げられる。連想メモリにおいては、全てのデータをメモリ効率の良いベクトル表記で扱え、また動的な問題に対しても処理の手間が悪くならない。この連想メモリを用いることによってLSI設計における複雑な図形処理の効率化を図ることができる。以下に、図形処理の具体例として、線分に隣接する線分の探索問題（図3）の解法を述べる。データ構造としては図4のようなものを用いる。

X軸と平行な横線分の集合Sと1線分$H(X_{LH},X_{RH},Y_H)$が与えられたとき、Hをy軸方向に平行移動させ、Hが最初に出会う横線分を求める操作には、

Ⅰ）$X_R \geqq X_{LH}$を満たすデータを選ぶ。

Ⅱ）Ⅰ）を満たし、$X_L \leqq X_{RH}$を満たすデータを選ぶ。

Ⅲ）Ⅱ）を満たし$Y \geqq Y_H$を満たすデータを選ぶ。

Ⅳ）Ⅲ）を満たすデータの中でYが最小のものを選ぶ。

という操作を行えばよい。この操作は、しきい値検索３回と極値検索１回で行え、CAMではデータ数をｎとした場合その計算複雑度はＯ（１）、空間複雑度はＯ（ｎ）となる。この線分隣接問題は、今回CAM上に実装した改良線分展開法の基本をなす探索操作である。

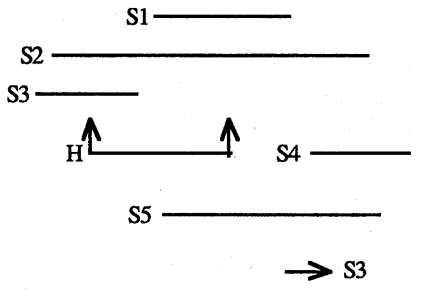


図3　線分隣接問題

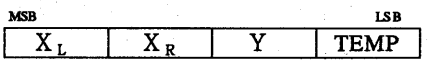


図4　データ構造

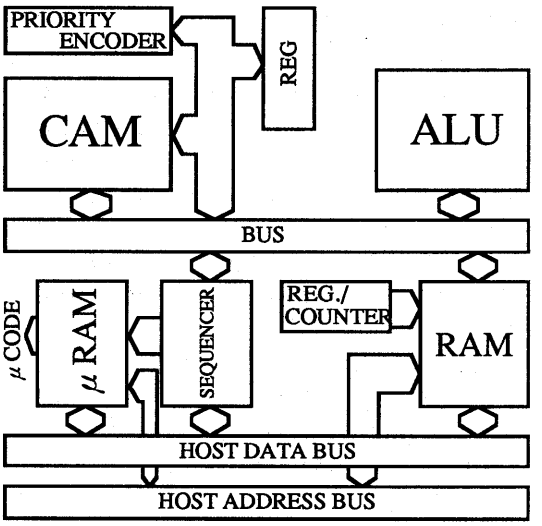


図5．CHARGEの構成図

３．図形処理プロセッサCHARGE

図形処理プロセッサCHARGE[3][4]はNTTによるCAMチップ[5]（4K bit/chip:32bit/word×128word）を用いている。このチップはクロックパルスに同期して与えられるコマンドにより動作し、30種のコマンドを持つ。CHARGEではクロックを200nsとしている。

CHARGEはホストコンピュータのバスに接続しておりスレーブプロセッサとして用いられる。CHARGEの構成を図5に示す。内部は大きく分けてシーケンサ部・RAM部・ALU部・CAM部の４つに分けられ、各モジュールはシーケンサによるマイ

クロコードによって並列に動作する。CHARGEのプログラム開発環境を整えるため、専用アセンブラとハードウェアエミュレータも用意した。

このCHARGE上にグリッドレスルータのひとつである改良線分探索法を実装し、ソフトウエアによるものと比較した結果、数倍の処理速度向上が確認できた。

4．改良線分展開法

4．1　CAM上での改良線分展開法

今回実装した改良線分展開法は、線分展開法を改良したもので基本的にはダイクストラ法に基づいたものである。まず、図6(b)に示すように始点Sを通る線分を発生させ、この線分を垂直方向に障害物にあたるまで掃引（展開）することによって配線領域を探索する。探索した長方形領域に終点Tがなければ、この長方形の外周上に必要に応じて線分を発生させ同様の処理を行う。これをTにたどり着くまで繰り返し、その後バックトレースを行い径路を得る(図6(h))。本稿では、発生させた線分をactive line（以下ＡＬとする）、ＡＬをコストに基づいて保持して次に展開させるＡＬを求めるためのデータ構造をpriority queue（以下ＰＱと表す）とする。CAMを用いた場合、ＰＱ用の特別なデータ構造を用いる必要はなく、CAMに格納した各線分にＰＱの内か外かを示すフラグをつけるだけでよい。

なお、ここで取り扱う配線モデルは、一層・垂直及び水平な線分で囲まれた領域（複合長方形領域）で斜め配線は扱わないこととし、２点S(ソース)・T(ターゲット)間の配線をおこなうものとする。また、既配線が存在していてもよいこととする。各線分はその中心線で表すが、設計規則に違反しないよう配線幅や配線間隔を考慮している。

以下、改良線分展開法のアルゴリズムを示す。

（0）データ初期化

図形探索用データ構造ＰＱを初期化し、空にする。

（1）ソースＳからのＡＬの発生

Ｓから配線領域内に最初のＡＬを発生させる。このＡＬ上にターゲットＴがあれば配線をして終了。なければこのＡＬをCAMに格納し、ＰＱ内に入れる。このＡＬのコストは０とする。

（2）ＡＬの展開

待ち行列ＰＱ内のＡＬの内、コスト最小のものをみつける。ここでＰＱ内にＡＬがなければ径路は存在しない。見つけたＡＬをＰＱから取り出し、これを障害物に衝突するまで掃引し（以下ＡＬの展開とする）、衝突する障害物を求める（図7）。これは２．２で述べた線分の隣接探索を行う操作に相当し、CAMを用いることにより、データ数に関わらず一定時間で行える。ここで、展開した領域上にターゲットＴがあればバックトレースに移る。

展開した領域の外周上にＡＬを発生させＰＱ内に入れる。発生させたＡＬ（以下、子ＡＬとする）には、発生元（以下、親ＡＬとする）へのポインタを持たせておく。このポインタはバックトレース時に用いる。ここで、発生させた子ＡＬが図８の様に直線に重なる場合は２本のＡＬを併合する。

ＡＬを展開して子ＡＬを作る際にはその子ＡＬにコストを与えるが、今回は以下のようなコスト計算式を採用した。

$$D = k1 \times A + k2 \times B + k3 \times C$$

Ｄ：コストの増分

Ａ：ＡＬの曲がり

（ある時＝１、ないとき＝０）

Ｂ：ＡＬからＴまでのマンハッタン距離

Ｃ：親ＡＬと子ＡＬの距離

ここで、k1,k2,k3はネットごとに各々指定できる重みであり、この値を変えることによって径路を特徴づけることができる。ＡＬを展開して新たなＡＬを作る際はこの式によりコストの増分を計算し、もとの親ＡＬのコストにそれを加え、作った子ＡＬのコストとする。ここで、線分と線分の距離はその線分の中心と中心の距離とし、点と線分の距離は、点と線分の中心の間の距離とした。このコスト計算はALUで行う。

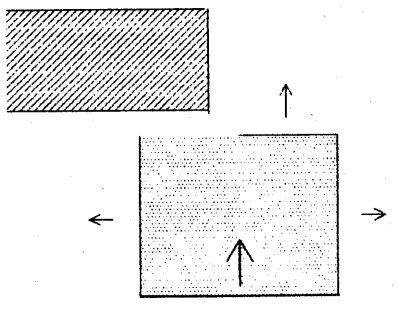
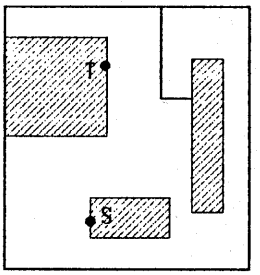
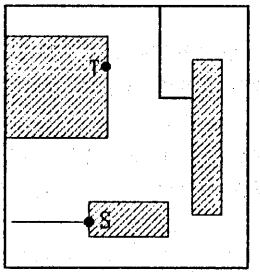


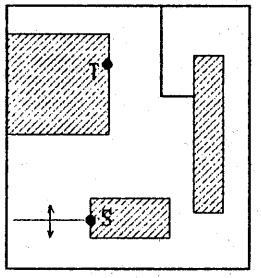
図7．ＡＬの展開



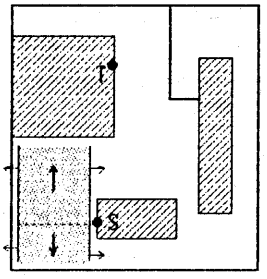
（ａ）配線領域



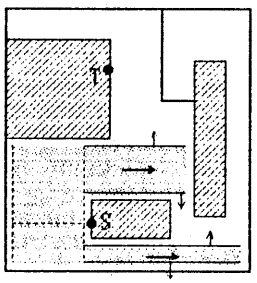
（ｂ）



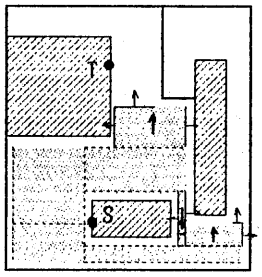
（ｃ）



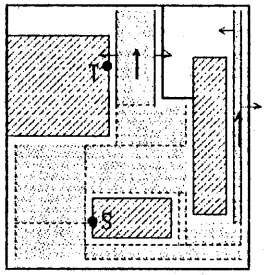
（ｄ）



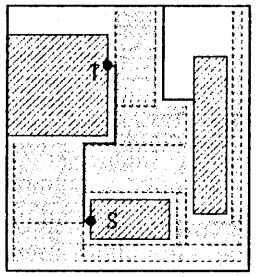
（ｅ）



（ｆ）

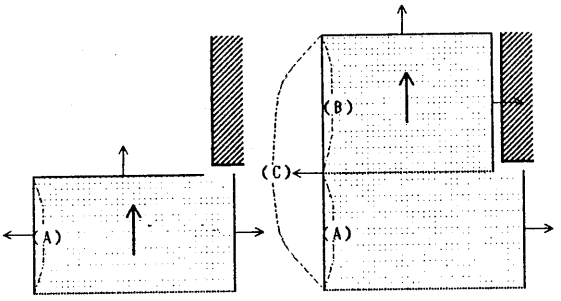


（ｇ）



（ｈ）バックトレース

図６　改良線分展開法



2本の active line(A),(B) は併合されて (C) になる

図8. ＡＬの併合

（3）バックトレース

ターゲットＴからソースＳまでバックトレースを行う。各ＡＬは親ＡＬへのポインタを持っているので、それを調べて行くことによりＳまでの配線径路を求めることができる。

（4）後処理

求めた配線径路をCAMに格納し、不要になったＡＬのデータをCAM上から削除する。

このアルゴリズムをCAMに実装する際のデータ構造を図9(a)に示す。CAMは１ワード３２ビットなので、それを８ビットずつＡ～Ｄの４ブロックに分け、Ｄブロックにはそのワードに格納されているデータの種類を示すＩＤ、およびフラグを格納した。各ＡＬはコストを持つが、そのコストはＡＬと同ワードには格納できないため、CAM内にコストのみを格納するワードを設け、それをコストワードとした。このコストワードには、そのコストを所有するＡＬのアドレスを持たせてある。各ＡＬの親ＡＬを示すポインタはRAMに格納したが、このポインタをCAM内に格納することも可能である。

全体的なデータ構造を図9(b)に示す。

ﾌﾞﾛｯｸ	A	B	C	D
横線分	Ｘｌ	Ｘｒ	Ｙ	ﾌﾗｸﾞ
縦線分	Ｘ	Ｙｕ	Ｙｄ	ﾌﾗｸﾞ
端子	Ｘ	－－	Ｙ	ﾌﾗｸﾞ
コスト	ALのｱﾄﾞﾚｽ		コスト	ﾌﾗｸﾞ

ＣＡＭ(32bit:A-D 8bit x 4)

ﾎﾟｲﾝﾀ	（未使用）	親へのpt.

ＲＡＭ(32bit:pt.10bit)

図9(a) データ構造

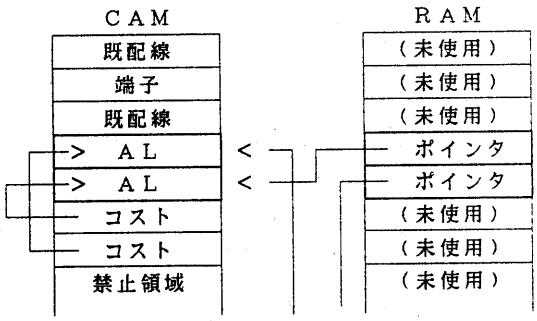


図9(b) 全体的なデータ構造

４．２ CAM実装における特徴

改良線分展開法をCAM上に実装する場合の特徴をまとめると以下のようになる。

（1）ＰＱ(priority queue)をCAMで実現している。

ＡＬ展開の際は、コスト最小のＡＬを選ぶ必要があるが、CAMではコスト最小のものを極値検索を行なうことでデータ数に関わらず一定時間で求めることができるので、ＰＱ用の特別なデータ構造を用いる必要がない。

（2）各種データがCAM内で混在している。

CAM内データのアクセスはCAM内のどこにデータがあっても一定時間で行え、データの場所を意識する必要がない。混在したデータの区別は各ワードの下位につけたＩＤ・フラグを用いて一定時間（実際には数クロック）で行え、データの混在による処理効率の低下は全くない。

（3）処理の時間複雑度がソフトウェアの場合に比べて向上している。

CAMを使用することにより、各種検索を一定時間で行える。そのため、配線処理全体の時間複雑度は、配線領域内の図形頂点数をｎとした場合O(n)で抑えられる。

（1）及び（2）に述べた通り、CAMを用いることにより、データ構造を簡単にすることができる。

４．３ 時間複雑度・空間複雑度

改良線分展開法をCAMに実装した場合の時間複雑度と空間複雑度について考察する。ｎは配線禁止領域の頂点数、ｋは探索された頂点数のことである。

（1）時間複雑度

一回のＡＬの展開に要する検索、即ち線分探索

はCAMを用いていることから、データ数に関わらず一定時間で行える。検索したデータに対し新たなＡＬを作成するわけであるが、これも一定時間で行える。探索された頂点数はｋであるから、ＡＬをソースからターゲットまで見つけるまで展開する操作においては、一定時間の操作をO(k)回行うことになり、その時間複雑度はO(k)となる(ただしｎ≧ｋ)。

バックトレースは、ターゲットからソースまでＡＬをたどって行く操作である。ＡＬからその親の（発生元の）ＡＬを見つける操作は、ポインタを参照するだけで済むので、一定時間で行える。このポインタをたどる操作を、ソースにたどり着くまで行うわけであるが、そのときたどるＡＬ本数は、ＡＬ展開時に作成したＡＬ総本数よりも少ないので、バックトレース全体について考えるとその時間複雑度はO(k)となる。

結局、ソースからターゲットを見つけるまでのＡＬを展開する操作の時間複雑度と、バックトレースの時間複雑度を考え合わせると、この改良線分展開法アルゴリズム全体としての時間複雑度は、O(k)となる(n≧k)。

（２）空間複雑度

CAM内に格納される配線領域の初期データは、１つのデータにつき１ワード必要なので、その空間複雑度はO(n)となる。ＡＬ展開時にCAM内に格納されるＡＬ及びコストのデータは、ＡＬの展開回数の時間複雑度がO(n)となることから、その空間複雑度もそれぞれO(n)となる。よって、全体としての空間複雑度はO(n)となる。

以上のCAMを用いた場合の計算複雑度と、今回比較実験に用いたヒープ探索木データ構造のソフトウェアによる手法の計算複雑度を表１に示す。

表１　改良線分展開法の計算複雑度

	CAM	ソフトウエア
時間複雑度	O(k)	$O(k \log^2 n)$
空間複雑度	O(n)	O(n)

ｎ：図形の頂点数
ｋ：ＡＬ展開時に探索された頂点数
（ただし、ｎ≧ｋ）

５．実験結果と評価

４で述べたアルゴリズムで、改良線分展開法を連想メモリプロセッサ上に実装し、実験を行った。実験は、座標値が０～２５５の正方平面内に乱数で禁止領域と２つの端子（ソースＳとターゲットＴ）を発生させ、その２つの端子を結ばせるようにした。配線領域内の頂点数は32,104,200,392の４種類にし、それぞれ10種づつのデータを用いて実験を行い平均値をとった。CAMのサイクルタイムは200nsである。また、今回はソフトウェアとの比較実験も行った。ソフトウェアはＣ言語で実装され、CAMと同じデータを汎用コンピュータ(SUN4/110)上で処理させた。それぞれの処理時間を表２、及び図10に示す。

表２　改良線分展開法の処理時間

	処理時間(ms)			
	n=32	n=104	n=200	n=392
CAM	6.0	14	28	35
ｿﾌﾄｳｴｱ	50	129	288	394

n:配線領域内図形頂点数

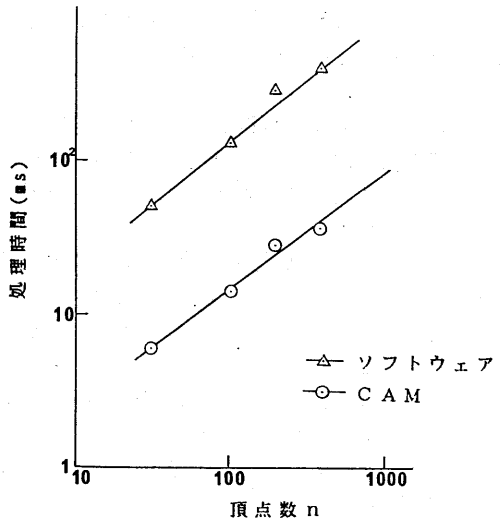


図10　改良線分展開法の処理時間

５．２　評価

CAMのグラフを見ると、最大の傾きの理論値は１であるが実際の傾きは約0.8であり、ＡＬの展開が領域全体に行きわたる前にターゲット端子を見つけて配線を行ったことが確認できる。この傾きの点に関してはソフトウェアによるものについても同様のことがいえる。実際の実行時間を比較すると、CAMはソフトウェアよりも約10倍程度速いことが分かる。また、今回の実験では、CAMの容量の制約から400頂点までの実験しか行えなかったが、データ数を多くした場合、処理速度の面でのCAMの効果が更に大きくなると期待できる。

6．おわりに

本稿では、ベクトル型データを用いたグリッドレスルータの一つである改良線分展開法を連想メモリプロセッサ（CAM）上に実装し、その特徴、アルゴリズム、および時間複雑度、空間複雑度について述べ、実際の処理時間をソフトウェアによる手法のものと比較した。

我々は、既にCAM上にグリッドレスルータの一つである改良線分探索法を実装し、その効果を確認しているが、今回実装した改良線分展開法は探索法に比べ、

- 配線ごとに設計規則を設定でき、またコストでその配線を特徴付けできるなど、柔軟性が高い。
- 径路探索に先立って行う処理（前処理）がいらない。
- 径路探索をする際、ターゲットを見つけた時点でその処理を打ち切るので、あらかじめ径路全体にエスケープラインを生成する改良線分探索法よりもメモリ効率がよい。

等の利点があり、径路探索に要する時間も改良線分探索法と改良線分展開法でほとんど差が無いことから、改良線分展開法は非常に有効な配線手法であると言える。

また、CAM上に改良線分展開法を実装する際の利点としては、以下のようなものがあげられる。

- 各種の検索が、データ数に関わらず一定時間で行える。
- データ構造が簡単であり、データがCAM上で混在していても構わない。
- priority queue（ＰＱ）用のデータ構造を作る必要がない。
- 時間複雑度、実際の処理時間が、汎用計算機によるものよりも向上し、被探索データ数に比例した手間で処理が抑えられる。

現在はCAMの容量が小さいため大規模な実験を行うことはできない。今回は１ワードで一つの線分を示すようにしたため座標値に８ビットしか用いることができなかった。実用的な問題を扱うためには座標値を24～32ビットにする必要があり、そのためには、１ワードのビット数を増やす、または、１線分を表すのに複数のワードを使用する、というようなことが考えられる。現実的には後者の方が有効と考えられるが、現在用いているCAMチップでは容量不足である。しかし、通常の１Ｍビットの SRAMが１ビット当り６トランジスタで構成されていることを考えると、CAMでは１ビット当り１１トランジスタが必要であるので、１ＭのSRAMと同等のプロセスでCAMを製作すれば500Kビット程度のCAMが実現できると期待でき、このような大容量のCAMを用いればより実用的な問題に対処できるようになると思われる。また、今回用いたCAMは本来、図形処理用に開発されたものでないこともあり、図形処理をする上で、機能的に不十分な面もある。今後の課題としては、既に製作したシミュレータ等を用いて、このような容量的・機能的な問題点を解決し図形処理をさらに効率化するCAMチップ、及びそれを用いた図形処理用ハードウェアエンジンのアーキテクチャの検証を行い、それらを実現することが挙げられる。

＜参考文献＞

[1] C.Y.Lee:"An Algorithm for Path Connection and its Applications",IRE.Trans.EC-10,pp.346-365(1961).

[2] 奥川：”連想メモリとその応用”、bit,vol.15,No.4,pp.318-329(1983).

[3] 井出、石和、小島　他：”連想メモリを用いた図形処理装置の試作”、信学技法、VLD87-106,pp.71-78(1987).

[4] 鈴木、大附：”連想メモリを用いたVLSI設計用図形処理ハードウェア”、電子情報通信学会論文誌、Vol.J72-A,No.3,pp.550-560(1989).

[5] 小倉、山田、丹野　他：”4Kb CMOS連想メモリLSI”、信学技法、SSD83-78,pp.45-53(1983).

[6] 小倉、山田：”20Kb CMOS連想メモリLSI”、昭和６１年度信学総全大447,pp.2-235(1986).

[7] 久保田、石和、鈴木、大附：”グリッドレスルータの連想メモリプロセッサへの実装”、設計自動化、Vol.44,No.9,pp.67-74　(1988).

[8] W.Heyns,W.Sansen and H.beke:"A Line-Expansion Algorithm for the General Routing Problem with a Guaranted Solution",Proc.17th DA Conf., pp.243-249 (1980).

[9] 小島、佐藤、大附：”線分展開法の改良とその評価”、設計自動化、Vol.48,No.6,pp.1-8(1989).