解　説

機能メモリのアーキテクチャとその並列計算への応用

6. LSI CAD 分野への応用†

佐藤政生††　久保田和人††

1. LSI 設計の流れ

大規模集積回路 (LSI: large scale integrated circuits) は，半導体デバイスの開発および集積化技術の発展により，現在では，1チップに数百万個以上のトランジスタを搭載することが可能となってきた．その使用範囲は計算機の CPU やメモリだけでなく，飛行機，自動車，など多くの分野に広がり，多種多様な LSI の開発が必要となってきた．めまぐるしく変わる社会のニーズの中で LSI を多品種少量生産するためには，LSI を短期間で設計する大規模集積化技術のますますの開発・発展が望まれる．ところが，高集積化に代表されるデバイス（ハードウェア）技術の進歩はめざましいが，与えられた仕様から LSI のマスクパターンまでの設計を支援する計算機ソフトウェア (CAD: computer-aided design) 技術の発展速度は遅いのが現状である．それは，LSI 設計が非常に多くの回路データや制約条件を処理しなければならなく，理論的には，設計の最適化が非常に困難な問題のクラス (NP 困難) に属する部分問題から成り立っているからであろう．そこで最近では，ソフトウェアだけに頼るのではなく，ハードウェアアクセラレータ (CAD エンジン) によって処理時間を短縮しようという試みがなされるようになってきた．

LSI は通常，図-1 のように，実現すべきシステムの仕様書が与えられると機能設計，論理設計，回路設計，レイアウト（配置・配線）設計，テスト設計などの各工程を経て設計される．機能設計はシステムをいくつかの機能ブロックに分割し，各機能ブロックごとに ALU，メモリ，レジ

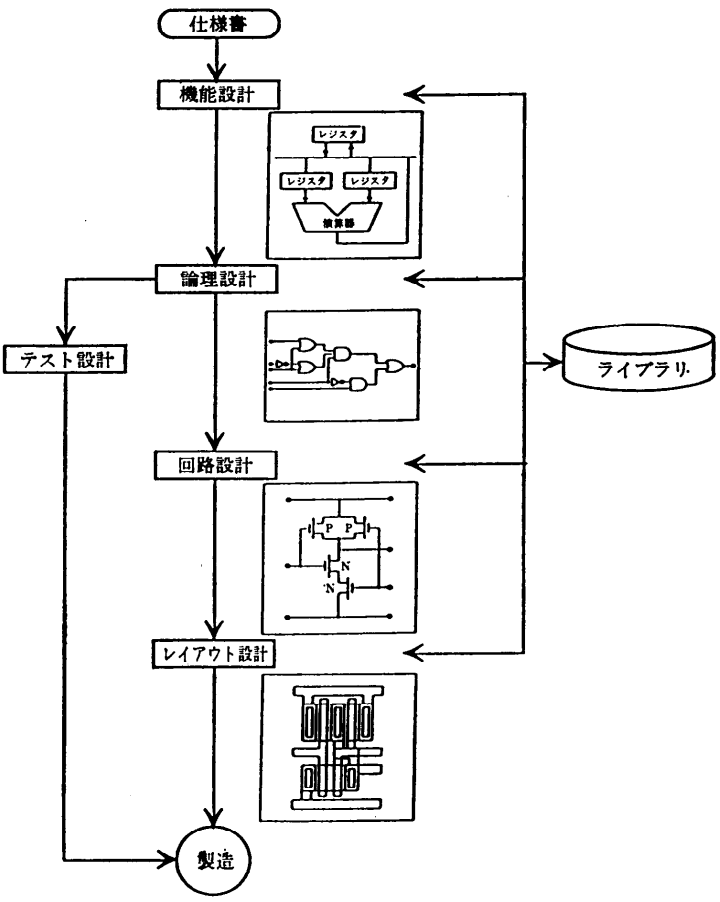


図-1　LSI 設計の流れ

† Applications of CAM to LSI CAD by Masao SATO and Kazuto KUBOTA (Department of Electronics and Communication Engineering, School of Science and Engineering, Waseda University).

†† 早稲田大学理工学部電子通信学科

スタなどの機能素子によって仕様を満足する回路を構成する工程である．論理設計は，機能素子をNAND や NOR, F/F (flip-flop) などの論理素子で表現する工程である．作成された論理構造に対しては論理シミュレーションが行われる．またこれと同時に，テスト設計である故障シミュレーション，テストパターン生成などが行われる．回路設計は，論理素子をトランジスタ素子によって表現する工程である．このとき，トランジスタ素子で構成された電子回路の特性を求めるために回路シミュレーションが行われる．レイアウト設計は，上流の工程で設計された回路素子を置く位置を定め，素子間を結ぶ配線の径路を決める工程である．この工程の最終段では，配置，配線設計によって作成されたマスクパターンがライブラリで定められた設計規則を満足するか否かの検証が行われる．テスト設計は，上述の故障シミュレーションやテストパターン生成に加えて，LSI が製造された際にそれが所望の機能を備えているか否かの検証を行う工程である．

現在の計算機は1次元の情報は非常に高速に処理することが可能であるが，図形のような2次元以上の情報処理を得意としていない．一方，機能メモリ (CAM) に基づくハードウェアエンジンは扱う情報が多次元であっても，メモリ内をいくつかのフィールドに分割して扱うことが可能なので処理時間にほとんど影響がない．このフィールドをあるときは図形の x 座標値，あるときは信号線の論理値といったようにみなすことができるので，CAM は非常に融通性が高く，LSI 設計への適用手法が多く提案されている．

本解説では，これらの適用例を紹介していくわけであるが，紙面の都合により，視角的に最も理解しやすいと思われるレイアウト設計への応用を中心に行ってゆくことにする．

2. レイアウト設計への応用

2.1 機能メモリによる基本的図形探索手法

LSI のレイアウト設計工程は素子間の配線，素子と配線が設計規則を満足するか否かを調べる幾何学的設計規則検証などの処理から成り立っている．これらの処理は幾何的な図形データであるマスクパターンに対する処理となるため，2次元平面上の図形処理問題と考えることができる．ここでは，これらの図形処理問題を構成する基本的図形探索問題を，機能メモリ (CAM) を用いて処理する手法について述べる[1]．

本解説では，CAM は以下に示す機能を保持するものとする．

（1） 一致検索機能

CAM は記憶アレイ，インデックス・レジスタ，マスク・レジスタ，レスポンス・レジスタの4つの部分から成り立っているとする．一致検索は記憶アレイに格納されているデータのうち，インデックス・レジスタの内容（マスク・レジスタによって部分的にマスクされる）と一致するものについて，そのデータが格納されているワードに対応するレスポンス・レジスタを1にセットする機能である（図-2）．

（2） 検索結果に対して演算を行う機能

レスポンス・レジスタの内容と現在の検索結果について AND や OR などの演算を行い，結果をレスポンス・レジスタに格納する機能のことである．この機能を用いれば，CAM 内のデータについて，ある値より大きいワードを求めたり（しきい値検索），一番大きいワードを求める（極値検索）ことができるようになる．

（3） 並列書込み機能

選択された複数のワードについて並列に外部から与えた値を書き込む機能のことである．図形処理を行う場合には，ある検索キーで検索された図形データの集合に対して対応するワードの特定のビットに1を書き込む(フラグをたてる)必要があるが，この機能を用いることにより，検索された

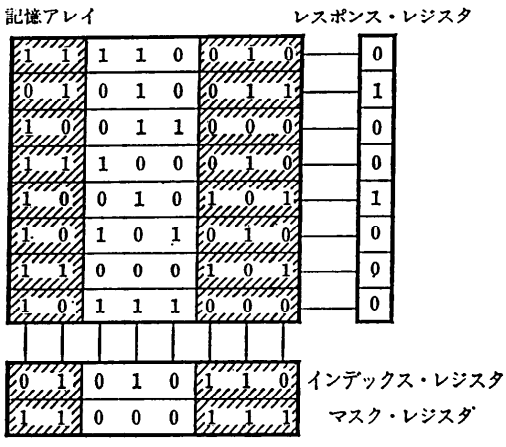


図-2 CAM の構成

データの数によらず一定時間でフラグをたてることが可能となる.

つぎに, LSI のレイアウト設計で用いられる基本的図形探索問題の例を示す.

(1) 線分交差問題

水平線分の集合 H に対し, 質問垂直線分 v が与えられたとき v と交差する H の要素を列挙する問題 (図-3).

(2) 領域探索問題

水平線分の集合 H に対し, 質問領域 R が与えられたときに, R と交差する (または, 含まれる) 線分を列挙する問題 (図-4).

(3) 線分隣接問題

水平線分の集合 H に対し, 質問水平線分 h が与えられたとき, h から上 (下) 方向の長方形領域内にある H の要素のうち最も h に近い要素を求める問題 (図-5).

これらの探索問題は, データの挿入, 削除がランダムに発生する動的な (Dynamic) 問題である. この種の問題は計算幾何学の分野で研究されており, それぞれの問題についてソフトウェアで効率よく解くためのデータ構造とアルゴリズムが提案されている. しかし, ソフトウェアの場合, 現在までに提案されているいかなる手法を用いても, 最悪の場合には処理時間の手間が検索の対象となる図形数に比例するものしか知られていない. さらに, 問題ごとに異なるデータ構造を必要とし, データの挿入, 削除, 検索が繰り返し行われる動的な問題では効率が下がるという欠点をもつ. これに対して CAM を用いた解法では, 上記の問題を図形のデータ数に依存しない一定時間で解くことができる. さらに, 動的な問題に対しても効率は落ちない. 以下では, (3)の線分隣接問題が CAM を用いることにより一定時間で解くことができることを示す. なお, (1), (2)の問題も同様の手順で解くことが可能であることは明らかであろう.

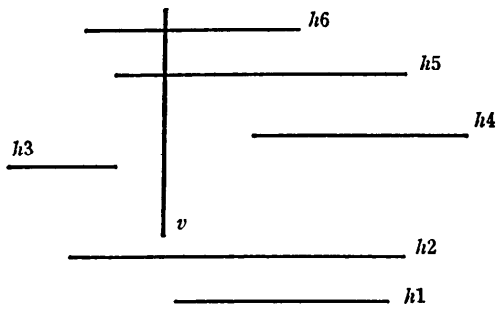


図-3 線分交差問題

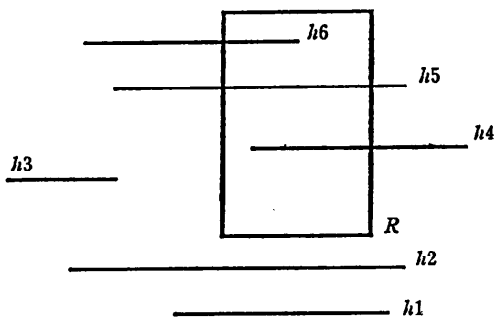


図-4 領域探索問題

線分隣接問題において質問線分は座標 QXl, QXr, QY, 被質問線分は Xli, Xri, Yi で表されているものとする. 被質問線分を図-6(a)に示したフォーマットで記憶アレイに格納し, 質問線分を(b)のフォーマットでインデックス・レジスタに格納する. 条件を満たす線分は次の手順で求められる.

Step 1. $QXr \geqq Xli$ を満たす線分を選ぶ.

Step 2. Step 1 を満たし, $QXl \leqq Xri$ を満たす線分を選ぶ.

Step 3. Step 2 を満たし, $QY \leqq Yi$ を満たす線分を選ぶ.

Step 4. Step 3 を満たす線分の中で最小の Yi をもつ線分を選ぶ.

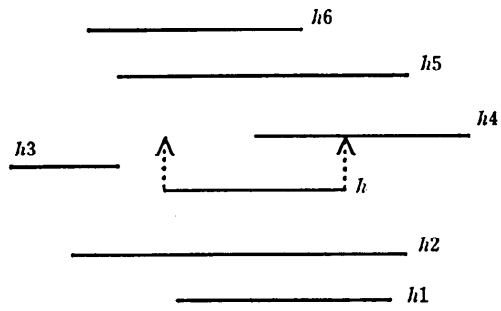
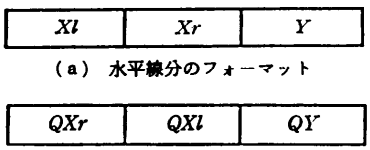


図-5 線分隣接問題



(a) 水平線分のフォーマット

(b) 質問水平線分のフォーマット

図-6 CAM に格納される線分のフォーマット

以上の処理は3回のしきい値検索と1回の極値検索を行うことによって解を求めることができる．これらの処理はすべてデータ数に依存しない一定時間で解くことができるため，線分隣接問題も一定時間で解ける．ソフトウェアでこの問題を解いた場合，最悪の場合処理の手間は図形データ数に比例することになる．

2.2 配線設計

与えられた回路接続情報をもとに，回路素子間に配線を行う処理が配線設計処理である．配線問題を図形処理問題として解く手法を一般にグリッドレス・ルータと呼んでいる．LSI の配線手法として有名なものに，配線領域を格子状に区切り，その格子を利用して配線径路を発見する迷路法[2]があるが，グリッドレス・ルータは，迷路法と比べて以下のような特徴をもっている．

（1） 図形データを直接扱うので，複雑な設計規則への対応が可能である．

（2） 迷路法のように格子を用いないので使用メモリが少ない．

（3） 配線結果のマスクパターンへの変換が容易である．

ここでは，グリッドレス・ルータの一つである「改良線分展開法」のアルゴリズム[3]について説明し，これを2.1で示したCAMを用いた図形処理手法を使って解いた場合にどれくらいの性能向上が図れたかを示す．アルゴリズムの記述にあたって言葉の定義をしておく．

Active line：径路探索の際に作成する補助的な線分．以下 AL と呼ぶことにする．

Heap：AL の管理を行う priority queue．新たに作成された AL は Heap に挿入される．AL は，後述するようにコストをもっているが，最小のコストをもつ AL が priority queue から取り出される．Heap の初期状態は空である．

LMS (line segment management system)：AL のほか，配線禁止領域の境界線分など幾何的なレイアウト情報をすべて保持しているデータ構造である．

二つの端子 S と T を結ぶ径路は以下のようにして求めることができる（図-7 参照）．

Step 1. S を通る AL を作成する（図-7 (a)）．もし，AL 上に T がある場合は，S と T 間に配線径路を作り Step 5 へ．そうでなければ，次のステップで上方向に配線領域を探索する AL と下方向に探索する AL の2本の AL を作成し Heap ならびに LMS に挿入する．このとき両者ともコスト0を与えておく．

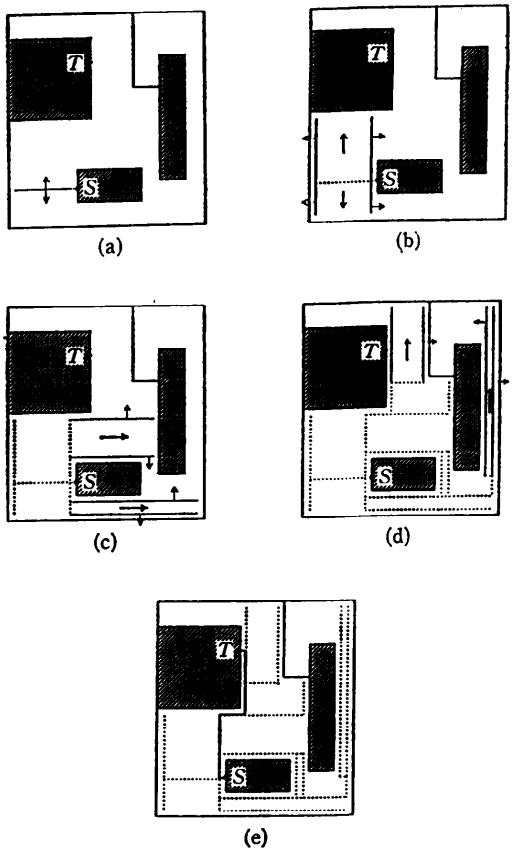


図-7 改良線分展開法

Step 2. Heap から AL を削除する．Heap が空なら Step 5 へ．このとき径路は存在しない．

Step 3. AL から（L とする），L とは垂直な方向に障害物にあたるまで掃引し，配線領域を求める（図-7 (b)-(d)）．これは，線分隣接問題を解くことと等価である．T が掃引した領域中にあれば Step 4 へ（図-7 (d)）．掃引した領域の境界に新しい AL を作成し，Heap ならびに LMS に挿入する．これらの AL には L へのポインタおよびコストをもたせる．ポインタは，逆追跡のときに使用する．コストは以下の式で与える．

$$G = aB + bD + cZ$$

G：コストの増分

B：曲がりの有無（曲がりあり：1，曲がりなし：0）

D: L からの距離

Z: T までのマンハッタン距離

ここで，a を大きくすると曲がりの少ない径路が求まる傾向が強くなり，b の値を大きくすると短い径路が求まる傾向が強くなる．c は，T に近づく方向の探索を優先させるものである．

Step 4. ポインタを用いて逆追跡を行い配線径路を得る（図-7（e））．

Step 5. LMS 内の AL を削除し，配線径路を構成する線分を LMS に挿入する．

次に，ソフトウェアを用いて上記のアルゴリズムを解いた場合と CAM を用いた場合の配線処理時間の違いを図-8 に示す．CAM は LMS を実現するために用いられている．配線領域を表す線分数が数百という小さな例題においても，基本的な図形探索処理は CAM のほうが 15 倍高速であり，配線処理は 3 倍高速である．この倍率は線分数の増加にしたがい大きくなる傾向にある[4]．この結果から CAM を用いた場合の処理の高速性が分かる．

2.3 幾何学的設計規則検証

LSI のマスクパターンは，最小幅，最小間隔などの製造プロセスなどから決まる制約を満たさなければならない．マスクパターンの制約違反の箇所は，マスクパターンの幾何学的な性質のみに着目すれば発見できるため，図形処理問題へと帰着することができる．ここでは，幾何学的設計検証問題のうち，最小間隔検証問題をとりあげ，CAM を利用して解く手法[5]，ならびにその性能評価について述べる．

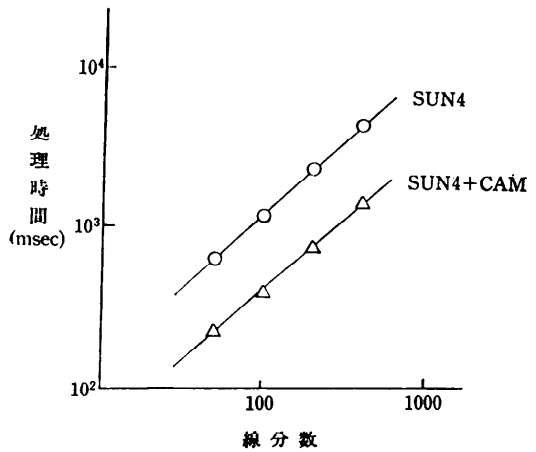


図-8　線分数と処理時間の関係

一般に，最小間隔を構成するのは図-9 に示すように凸頂点対もしくは凸頂点と線分のいずれかである．したがって，すべての凸頂点から一定距離以下の領域にほかの図形が存在するかどうかを調べることで，すべての設計規則違反を検出することができる．ここで，点から一定の距離以下の領域というと円になるが，円の領域を探索するのは CAM を用いた手法では困難であるため，ここでは，この円を包含する最小の正方形領域を探索する（図-10）．これにより検出された線分は必ずしも設計規則に違反するとは限らないため，これらの線分は後処理の厳密距離計算によって再び判定される．

最小間隔検証手法はワークリスト法に基づき，幅 $2D$（D は設計規則の最小間隔）の縦方向スリットと交差する線分をワークリストに保持する（図-11）．ここで，幅 $2D$ の分だけ線分を保持しているのは，すべてのデータを CAM に格納するのは容量的に困難であるためで，探索漏れの起こらない必要最小限のデータを CAM に保持させるためである．アルゴリズムを以下に示す．なお，

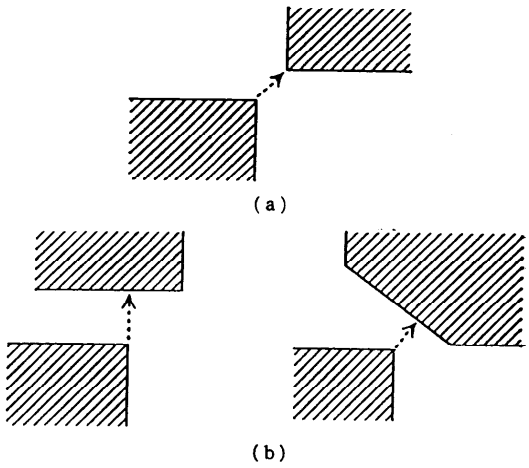


図-9　最小間隔の構成

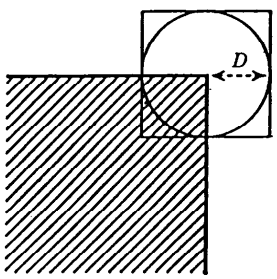


図-10　凸頂点からの領域探索

入力される図形データは輪郭取りされており，図形の境界線分データがその左端点のX座標でソートされているものとする．

Step 1（初期化）．スリットの右端を入力データの先頭データの左端点に重なるようにし，このスリットと交差する線分をワークリストに入力する．

Step 2（スリット移動先の探索）．現在のスリットの右端より右側で最も近い頂点を求める．この頂点は，現在のスリットの右端より大きいx座標をもつワークリスト内線分の右端点の中で最も小さいものと，入力データの先頭データの左端点とを比較し，その小さいほうとなる．Step 4 でスリットを移動するときには，スリットの右端がこの頂点と重なるようになる（**図-12**）．

Step 3（最小間隔検証）．現在のスリットの中心線と移動したと仮定したときのスリットの中心線との間（図-12 の斜線の領域）にある凸頂点から一辺の大きさが$2D$の正方形領域探索を行う．この領域探索により検索された線分は，前述のように必ずしも設計規則に違反しているとは限らないため，ホストコンピュータにデータを送り，頂点と検出された線分間の距離を厳密計算して設計規則に違反しているかを判定する．

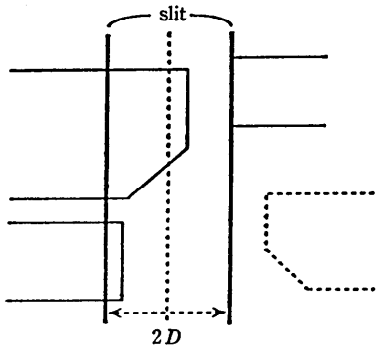


図-11 スリットと交差する線分

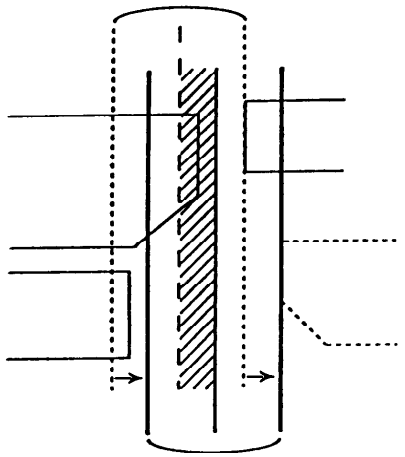


図-12 スリットの移動

Step 4（スリットの移動：ワークリストの更新）．移動後のスリットと交差しない不要な線分（**図-13** の太点線で表されている線分）をワークリストから削除する．そして，移動後のスリットと交差する線分（図-13 の太実線で表されている線分）をワークリストに挿入する．

Step 5. Step 2 に戻る．

以上の操作をスリットが領域全体を操作するまで繰り返す．

この手法の時間複雑度は，入力された図形の線分数をnとすると$O(n)$となる．一方，同様の処理をソフトウェアのみで行うと$O(n^{1.5})$となる．図形数と処理時間について実験を行った結果が**図-14** である．CAM を用いることにより，線分数が1千本程度のときに5倍，1万本程度のときに12倍高速に処理を行っている．

3. シミュレーションとテスト生成への応用

3.1 論理シミュレーション

論理シミュレーションは，論理回路情報と入力信号系列を入力とし，各時刻における回路の状態を計算によって求め出力するものである．論理シミュレーション手法としては，S—アルゴリズムとT—アルゴリズムが知られている．以下では，CAM に基づくそれぞれの手法[6]について述べる．

S—アルゴリズム（空間優先評価アルゴリズム）は，時刻を逐次進めながら各時刻における回路の

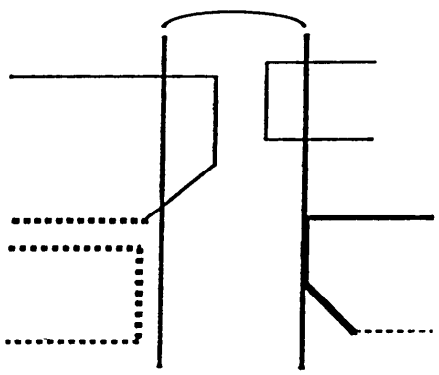


図-13 ワークリストの更新

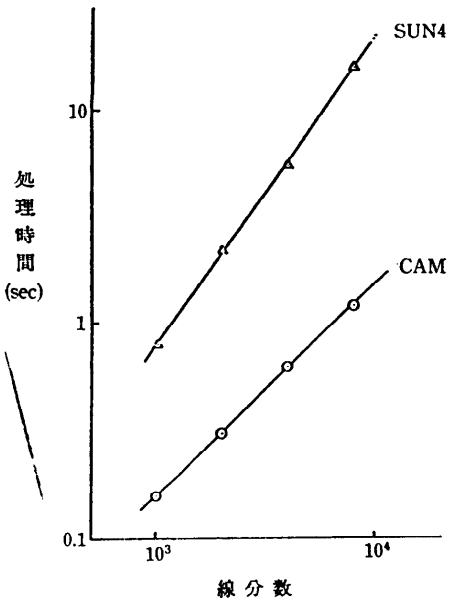
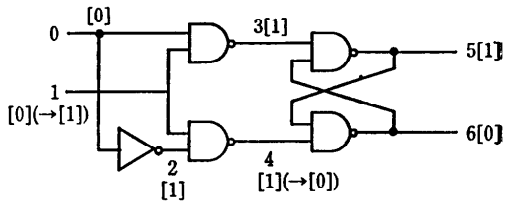


図-14 線分数と処理時間の関係

状態を計算してゆくものである．通常，回路中の全ての論理素子についてその出力を求める必要はなく，入力信号値が変化したものだけを処理すればよい．この信号値の変化をイベントといい，各時刻のイベントを処理してゆく手法がイベント法（選択的追跡法）である[7]．ここでは，回路は2入力 NAND ゲートからなる単位遅延モデルとし，入力として論理回路の信号線リスト（ネットリスト：各ゲートの入出力端子につながる信号線名の一覧）および外部入力におけるイベントの系列が与えられるとする．

Step 1. CAM のビットを**図-15**（b）のようなフィールドに分割する．ここで，各ワードは，一つの NAND ゲートに対応しており，Net I1, Net I2, Net O には入出力のネット番号が，I1, I2, O, New O には各ネットの信号値が，E には信号値の変化の情報が格納されることになる．ネットリストに基づき各ワードの Net I1, Net I2, Net O の部分に対応するゲートの入出力端子が属するネット番号を書き込む．図-15 では，インバータは2入力の NAND ゲートとして扱われており，Net I1, Net I2 には入力ネット番号の0が，Net O には出力ネット番号である2が書かれている．

Step 2. 入力および内部にイベントが生じていれば，そのイベントを一つ取り出す．なければ，



（a） ネット番号と信号値

NetI1	NetI2	Net$O$	I1	I2	O	New$O$	E
0	0	2	0	0	1	1	0
0	1	3	0	0→1	1	1	0
1	2	4	0→1	1	1	1→0	0→1
3	6	5	1	0	1	1	0
4	5	6	1	1	0	0	0

（b） CAM 上でのデータ

図-15 D フリップフロップに対する S—アルゴリズム

Step 5 へ．たとえば，ある時刻において，図-15（b）のようにデータが CAM に書かれていたとき，**図-15**（a）のようにネット1の値が0から1に変化したとすると，これがイベントとなる．

Step 3. イベントが生じたネットにつながっているゲートを探索し，その入力値をセットする．図-15 の例では，Net I1 (Net I2) が1であるワードの I1 (I2) のフィールドに1が書き込まれる．

Step 4. Step 2 へ．

Step 5. ゲートの出力値を計算する．例題では，回路は NAND ゲートから構成されているので，11, 12 がともに 1 である場合には 0，それ以外の場合には1が New O のフィールドに書き込まれる．

Step 6. ゲートの出力値が変化していれば，フラグをたてる．これは，O と New O が異なっているか否かを調べることで分かる．違いが生じている場合には，Eフィールドに1を書き込むと同時に，New O の値を O に書き込む．このEフィールドは次時刻のイベント処理のときに利用される．図-15 の例では，ネット4の信号値に変化が生ずるようになる．

Step 7. 1単位時間分のシミュレーション終了．Step 2 へ．

上述の入力値の書き込み，出力値の計算，イベントの検出はマスクレジスタを用いた一致検索および並列書き込みを用いることで全ワードで並列に行うことができる．京都大学で作成された FMPP (Function Memory type Parallel Processors) のエミュレータでは，357 K イベント/秒のシミュ

レーション速度があることが報告されている．

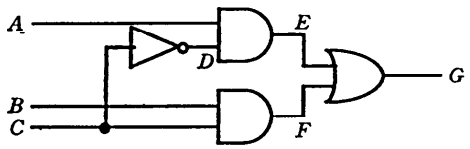
T—アルゴリズム（時間優先評価アルゴリズム）はS—アルゴリズムとは逆に，あるゲートに対してすでに決定している入力信号系列をまとめて評価（ゲートの入力値から出力値を計算し，出力信号線の値を更新すること）するものである．CAM に基づいた T—アルゴリズムでは，**図-16**（b）のように，CAM の各ワードを一つのシミュレーション結果に対応させ，各ビットをネットに対応させる．図-16（b）には，同図（a）の回路に対して長さ4の入力信号系列を与えた場合のシミュレーション結果が示されている．この処理は，まず入力信号系列を A, B, C のネットに対応するビットに書き込み，次に入力段から順次（レベル・ソート順に），ネットの信号値を計算してゆくことによってなされる．CAM の並列動作により，この処理は入力信号系列の書き込み後は，ゲート数に比例する時間で入力信号系列全てに対するシミュレーションを実行できる．FMPP エミュレータによる実験では，651 K ゲート/秒/ワード数の速度があることが報告されている．

T—アルゴリズムに基づくシミュレーションを組合せ回路の全入力パターンに対して行うことにより，回路が計算する論理関数の真理値表を作成することができる．T—アルゴリズムは，回路だけでなく論理式の評価も行うことが可能なので，仕様として与えられた論理式と設計された回路を真理値表に展開して比較することができる．CAM の1ワードにこの二つの真理値表を作成すれば，それらの比較を並列に行え，出力が一致しない入力条件をゲート数と式の長さに比例した時間で検出することができる．したがって，このアルゴリズムは論理検証にも応用することが可能である．

3.2 故障シミュレーション

故障シミュレーションは，与えられた回路に故障があることを前提として，入力信号系列に対して論理シミュレーションを行うことである．ここでいう故障とは，ある信号線の論理値が0または1に固定されている縮退故障（0縮退故障または1縮退故障という）のことである．故障シミュレーションは，入力信号系列で検出できる，もしくは，できない故障を求めたり，故障検出確率を求めたり，次節にあるようにテストパターンを生成するのに用いられる．

以下では，文献 8）で提案されている手法について述べる．CAM 内部では**図-17**（b）のように回路データが格納される．ここでは，各ワード Wi は i 番目の故障 fi が起こった場合のシミュレーションのデータを記憶するのに用いられている．ただし，$W0$ は故障のない正常回路に対するデータである．各ワードの故障番号フィールドには故障番号が2進法で記録されている．ほかの各ビットは一つのネットに対応し，その信号値が記録される．ここでいうネットとは，分岐がない結線要求を意味し，**図-17**（a）のネット2のように途中で分岐する場合には，ネット 5, 6 と新たに番号を付けることにする．入力信号系列が CAM に書き込まれたならば，後は前節で述べた T—アルゴリズムと同様の方法によって各信号値を計算してゆく．T—アルゴリズムと異なる点は，ゲートの出力値を計算するときに，故障ネットに出会

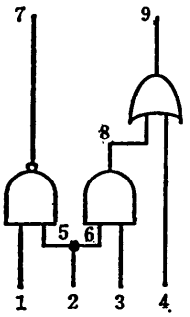


（a） ネット番号

A	B	C	D	E	F	G
0	1	1	0	0	1	1
1	0	0	1	1	0	1
1	0	1	0	0	0	0
0	0	1	0	0	0	0

（b） CAM 上でのデータ

図-16 データセレクタに対する T—アルゴリズム



（a） ネット番号

	故障番号	ネット番号
W0（正常回路）	00000	101100101
W1(f1)	00001	101000100
W2(f2)	00010	101100111
W3(f3)	00011	111011011
⋮	⋮	⋮
WF(fF)	11101	101000000

（b） CAM 上でのデータ

図-17 故障シミュレーション

ったらその信号値を故障信号値に書き換える（故障の挿入という）ことである．故障 fi の挿入はワード Wi に対してのみ行えばよいので，通常のメモリと同じ read/write 命令を用いて行うことができる．シミュレーション終了後の出力信号の比較は，各外部出力について，まずその正常回路（ワード $W0$）の信号値を読み出し，全ワードについてこれと一致検索を行うことにより実現される．また，検出できた故障，できなかった故障の集合も取り出すことができる．ソフトウェアではゲート数の2乗に比例する時間がかかるのに対し，CAM を用いると線形時間で実行可能となる．

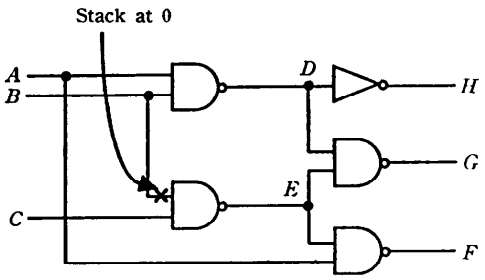
3.3 テストパターン生成

ある故障に対して，入力信号系列の中からその故障を検出可能な入力信号パターンの集合を求めることによって，その故障に対するテストパターンは作成される．これは次のようにして実行される[6]．組合せ回路とその中の一つの故障（複数の故障の組でもよい）が与えられたとき，すべての入力信号の組合せについて，正常回路に対しては論理シミュレーション，故障回路に対しては故障シミュレーションを行う．このとき，故障の影響を受けない部分のシミュレーションは共有できる．外部出力信号値を比較して，正常回路と故障回路が異なっている入力信号パターンがその故障に対するテストパターンとなる．テスト生成問題は，NP 困難に属する非常に難しい問題であるが，CAM に基づく T—アルゴリズムを用いると，回路の規模に比例した時間でテスト生成を行うことが可能となる．

例として**図-18**（a）の回路を考える．図の×の位置に0縮退故障があるとする．正常回路と故障回路を同図（b）のようにシミュレーションを行う．ここで，x' は故障回路に対する信号値である．この例では，故障の影響が出力 F, G に現れる．故障を検出する入力パターンは F と F', G と G' が異なっているものであるから，この場合には，F に対しては $(A, B, C)=(1, 1, 1)$, G に対しては，$(A, B, C)=(0, 1, 1)$ がテストパターンとなる．

3.4 回路シミュレーション

回路シミュレーションとは，設計者が回路構成と素子特性を入力として与え，計算機シミュレーション



（a） 故障のある回路

A	B	C	D	E	E'	H	G	G'	F	F'
0	0	0	1	1	1	0	0	0	1	1
0	0	1	1	1	1	0	0	0	1	1
0	1	0	1	1	1	0	0	0	1	1
0	1	1	1	0	1	0	1	0	1	1
1	0	0	1	1	1	0	0	0	0	0
1	0	1	1	1	1	0	0	0	0	0
1	1	0	0	1	1	1	1	1	0	0
1	1	1	0	0	1	1	1	1	1	0

（b） CAM 上でのデータ

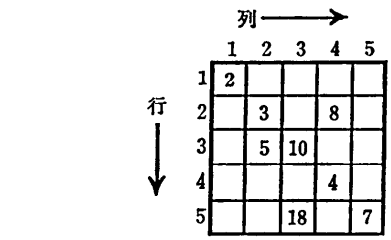
図-18 テスト生成

ションにより回路の特性を求める処理である．この処理は一般に，非線形連立方程式を解析せねばならず，ニュートン法の各ステップごとに繰り返し線形連立方程式を解くことに帰着される．連立方程式の解法としては回路シミュレーションの場合は集束性の問題から，反復法はあまり用いられずに，LU 分解などの直接法が用いられる．ここで，取り扱う行列は大規模なスパース行列となるため，このような行列を高速に処理する手法が必要となる．以下では，CAM を用いたスパース行列処理算法[9]について述べ，汎用計算機で行ったスパース行列処理との比較について述べる．

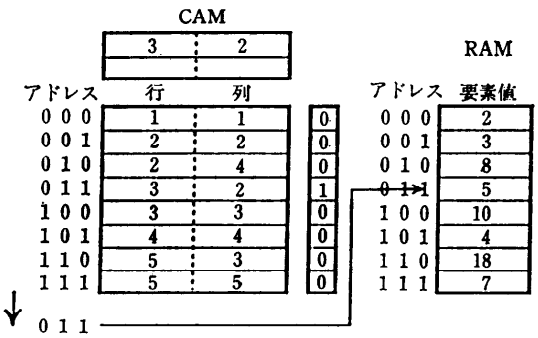
汎用計算機でスパース行列を格納する場合は，メモリを節約するため被零要素のみを格納することになる．このような状況下では，行列要素の抽出に手間がかかり，行列処理全体の効率を落としかねない．CAM を用いた場合の行列要素の保持は，**図-19** のように行う．まず，CAM と RAM を用意しそれぞれのワードが一対一に対応するようにアドレスを与える．ある行列の要素の格納は，CAM に行列要素の行番号および列番号を，RAM の同じアドレスには要素値を格納する．このとき選ぶアドレスは要素の行番号，列番号によらず任意でよい．ある i 行 j 列の要素が格納され

ている RAM のアドレスは，i, j という値が格納されている CAM のワードを検索することで求められる．同様に，特定の i 行の要素を検索する処理は，列番号の部分をマスクし行番号が i のワードを検索することで行われる．

この原理をもとに構成したハードウェア上で，LU 分解をクラウト法を用いて行った場合の処理時間を汎用機と比較した結果を図-20 に示す．CAM により行列の次元数が 200 のときには 66 倍，1,000 のときには 122 倍の高速化が得られている．



(a) 格納する行列



(b) CAM を用いた要素の格納および3行2列要素の抽出

図-19 CAM を用いた行列操作

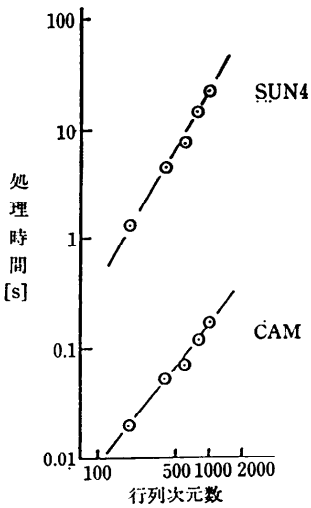


図-20 スパース行列処理時間

4. 今後の展望

機能メモリ (CAM) に基づいた CAD エンジンの LSI 設計への応用を紹介してきた．CAM はその融通性の高さから，多くの分野に適用可能であるため，LSI 設計専用ハードウェアではなく，ワークステーションなどの汎用ホスト計算機のバックエンドプロセッサとして，また，汎用計算機のメインメモリの一部として使用するのが将来のあり方であり，その期待は非常に大きい．今までは多次元情報処理を高速に行うために複雑なデータ構造を導入するなどソフトウェア的な改善努力を行ってきたが，今後は汎用 CAM アクセラレータの実現により，処理時間の短縮がはかれるだけでなく，ソフトウェアの開発にかかっていた時間を短縮することが可能となる．最後に，このような汎用 CAM アクセラレータを実現する上での課題を4つ以下に述べる．

1) 大容量 CAM チップの開発

CAM アクセラレータの有効性は 1 CAM チップに搭載可能なメモリ容量に大きく依存する．現在の技術をもってしても 500 K ビットから 1 M ビットの CAM チップの実現は可能である．

2) 高機能 CAM チップの開発

CAM アクセラレータで扱う個々のデータ（線分の端点の x, y 座標値など）のビット長が長くなった場合，各データは複数の隣接するワードに分けて格納することになる．このために，隣接するワードに対応するレスポンス・レジスタ間に通信機能を設ける必要が生ずる．逆に，この通信機能を用いることにより，隣接するワードに格納されているデータの組に対して並列に算術論理演算を行うことが可能となる．さらに，加算などを並列に行う場合には，各ワードがもつ検索結果とレスポンス・レジスタ間の演算機能に，AND, OR だけでなく EXOR などが追加されたほうが処理の効率が良くなる．上記のようなレスポンス・レジスタに関するワード間通信機能，多様な演算機能をもった CAM チップの開発が望まれる．

3) 多数 CAM チップ搭載ハードウェアの開発

CAM チップは単体で動作するわけではないので，CAM を高速に効果的に利用するためのアー

キテクチャの開発改善が必要である．また，1Mビット CAM チップが実現したとしても，1チップで扱える情報量は線分数にして約8,000本である．数百万トランジスタを搭載したLSIのマスクパターンの線分数が数千万から数億であることを考慮すると，多くの CAM チップを複数のプリント基板を用いて接続する必要がある．高速動作を要求されるためノイズの低減を考慮した接続方法の考案が求められる．

4) CAM とのインタフェースの整備

CAM を稼働させるソフトウェアの開発には二つの方法が考えられる．一つはソフトウェア製作者が CAM 用の命令を並べて CAM の機能を最大限に使用する方法であり，もう一つは CAM を稼働させる命令をライブラリ関数としてあらかじめ用意しておいて，ソフトウェア開発者はC言語のような一般言語によってこの関数を組み込みながらプログラムを作成する方法である．一般言語と CAM とのリンクはコンパイラによって行われる．汎用性を考えると後者の方法が好ましいのは明らかであるのでコンパイラの開発，ライブラリ関数の整備，などが望まれる．

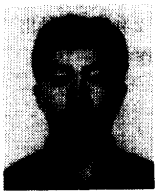
参 考 文 献

1) 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，電子情報通信学会論文誌A，Vol. J 72-A, No. 3, pp. 550-560 (1989).
2) Lee, C. Y.: An Algorithm for Path Connection and Its Applications, IRE. Trans, EC-10, pp. 346-365 (1989).
3) 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，情報処理学会設計自動化研究会資料，89-DA-48-6, pp. 1-8 (1989).
4) Sato, M., Kubota, K. and Ohtsuki, T.: A Hardware Implementation of Gridless Routing Based on Content Addressable Memory, 27th DA Conference, 38. 5, pp. 646-649 (1990).
5) 滝澤哲郎，石川拓也，久保田和人，佐藤政生，大附辰夫：連想メモリによる VLSI 設計規則検証，電子情報通信学会 VLSI 設計技術研究会資料，VLD 90-102, pp. 37-44 (1990).
6) 安浦寛人，渡辺章弘，左連隆吾，田丸啓吉：機能メモリ型並列プロセッサ FMPP 上での論理シミュレーション，電子情報通信学会コンピュータシステム研究会資料，CPSY 90-94, pp. 41-48 (1990).
7) 村井真一：ゲートレベル論理シミュレーション，情報処理，Vol. 25, No. 10, pp. 1119-1124 (1984).
8) 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回回路とシステム軽井沢ワークショップ資料，pp. 63-68 (1991).
9) 佐藤光一，久保田和人，佐藤政生，大附辰夫：連想メモリを用いたスパース行列処理プロセッサ，情報処理学会計算機アーキテクチャ研究会資料，Vol. 90, No. 60, 90-ARC-83-10, pp. 55-60 (1990).

(平成3年9月11日受付)

佐藤 政生（正会員）

1959年生．早稲田大学理工学部電子通信卒業．1986年同大学院博士課程修了．工学博士．1984年早稲田大学情報科学研究教育センター助手．1986年カリフォルニア大学バークレー校研究員．1987年拓殖大学工学部情報工学科助教授を経て，現在，早稲田大学理工学部電子通信学科助教授．電子回路の設計自動化，ノイズ解析，計算幾何学，グラフ理論等の研究に従事．1987年度丹羽記念賞，1990年安藤博学術奨励賞受賞．IEEE, ACM, 電子情報通信学会，プリント回路学会，日本 OR 学会各会員．



久保田和人（正会員）

1964年生．1988年早稲田大学理工学部電子通信卒業．1990年同大学院博士前期課程修了．現在，同大学院博士後期課程在学中．1991年早稲田大学情報科学研究教育センター助手．電子回路の設計自動化およびそれに関連した専用ハードウェア，計算幾何学等の研究に従事．IEEE, 電子情報通信学会各会員．