論理チェッカーのための論理回路図面自動生成アルゴリズム

若林, 哲 / Wakabayashi, Satoshi / 三宅, 祥二 / 檀, 良 / Dang, Ryo / Miyake, Shoji

(出版者 / Publisher)
法政大学工学部
(雑誌名 / Journal or Publication Title)
法政大学工学部研究集報 / 法政大学工学部研究集報
(巻 / Volume)
26
(開始ページ / Start Page)
43
(終了ページ / End Page)
52
(発行年 / Year)
1990-02
(URL)
https://doi.org/10.15002/00003927

論理チェッカーのための論理回路図面自動生成アルゴリズム

若林　哲*・三宅祥二**・檀　良**

Automatic Routing Algorithm for Logic Diagrams Used in Logic-checker

Satoshi WAKABAYASHI*, Shoji MIYAKE** and Ryo DANG**

Abstract

A simulation program for automatic placing and routing is described for constructing logic diagrams from text format outputs of the Logic-Checker[1-3]. The automatic placing and routing algorithms are based on the levelized-method and the maze-solving method, respectively.

§1. はじめに

我々は，論理回路設計における回路自動修正機能〔論理チェッカー〕[1-3]の研究を行っている。これは，論理回路設計者が設計した論理回路を論理シミュレータにかけるときに，現在のシミュレータは設計者の入力した期待値とシミュレーション結果が一致しているかどうかの判断だけを行っている。その点を改良して，その期待値を満たす論理回路にゲートを置き換えを行い，出力するものである。この機能は，近年のLSIの需要の急増に伴い，LSIの設計者の需要も急増しており，その結果，エキスパートと呼ばれるような設計者は非常に不足している。そこで我々は，このような機能を考案し，多少論理回路の設計が可能な設計者が設計を行う場合に，適切な指示が行えるこのような，いわゆるエキスパート・システムに近い機能を持った論理シミュレーション・システムの開発を行っている。

本研究は，さきに述べた論理チェッカーの結果出力がテキスト形式で出力されたものからスケマティックな図面を自動的に生成し，画面等に出力するためのアルゴリズムについての研究である。これはゲート位置を自動的に配置し，端子間を自動的に配線するものである。配置はあらかじめ準備された仮想図面上の配置位置に論理ゲートを自動配置してゆく。配線は，端子間の配線領域に自動配線してゆくものである。

* 大学院工学研究科電気工学専攻（博士後期課程）
** 工学部電気工学科電気電子専攻

§2. 論理チェッカーの概要

本章では，論理チェッカーが論理ゲートの修正を施すためのアルゴリズムの概要について述べる。論理チェッカーとは，論理回路に対する出力の期待値を入力することにより，その期待値と論理シミュレーションの出力結果との一致，不一致にかかわらず，期待値信号にかかわるゲートをゲート単位で置き換え，シミュレーションを行い，その出力結果が期待値と一致した場合，そのゲートを置き換え可能ゲートとして出力するというものである。

この機能の全体の流れはFig. 2.1にフローチャートで示す。論理シミュレーションの結果と期待値とが異なった場合に，これを正しい結果に導くには，どのゲートを修正または変更すれば良いかという判断を，自動的にコンピュータを用いて行う[1]。まず，結果と期待値の異なった出力段に最も近いゲートを，入出力ピンの合致するゲートに置き換えてシミュレーションを行う。期待値を満たすゲートが存在した場合には，そのゲートを置き換え可能ゲートテーブルに記録する。存在しなかった場合には記録しない。これを入出力ピンの合致するゲート全てに対して行う。次に，そのゲートの入力を出力とするゲートを同様に調べて行く。そして，期待値を与えられている出力ピンに影響を与えるゲート全てを調べ終わると，結果として置き換え可能ゲートを，テキスト形式で表示するものである。

この論理チェッカーは，期待値を入力することにより回路の修正箇所を示すものなのだが，まず置き換えるゲートを回路内で一つだけに限定し，そのゲートを入力ピンの数が同数であるという置き換え条件を満たすゲートに置き換え，シミュレーションを行う。その結果を期待値と比較し，一致していれば置き換えたゲートを置き換え可能ゲートテーブルに保存する。このとき置き

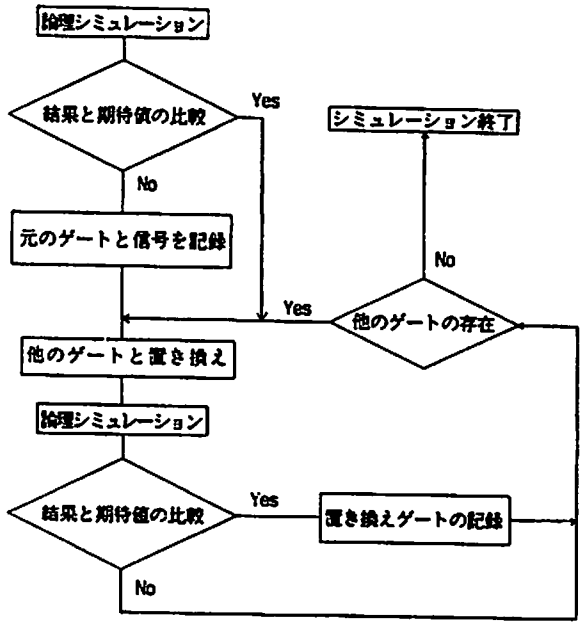
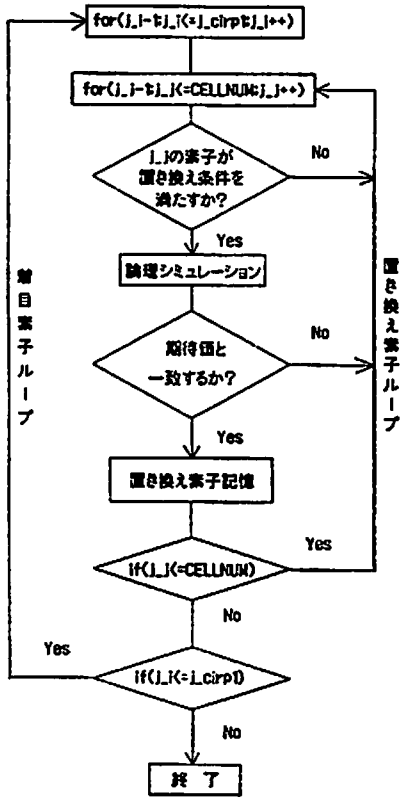


Fig. 2.1　論理チェッカーのフローチャート



J_cirp1: 置き換える素子の数
CELLNUM: 登録されている素子数

Fig. 2.2　1ゲート置き換えの
フローチャート

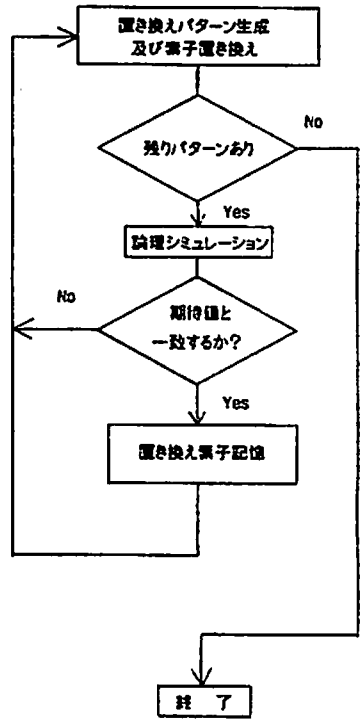


Fig. 2.3　複数ゲート置き換えの
フローチャート

換えるゲート以外は，初期入力ゲート状態のままとする。そして，回路内の期待値置き換えが終了したら，保存しているゲートを表示する。これらの流れはFig. 2.2のフローチャートに示す。次に回路内のゲート全部を対象に，複数個の置き換えを考慮し，置き換えるべきすべてのゲートについて着目した。そして，考えうるすべてのゲート置き換えのパターンについてシミュレーションを行い，結果が一致した場合は，そのときの置き換えたゲートを，置き換え可能ゲートテーブルに記録するというものである。そして，パターンの生成方法は，1ゲート置き換え時に，置き換え可能なゲートをメモリ上にデータベース化し，そのデータベースを基にして置き換えを繰り返し，パターンを生成する方法である。これらの流れはFig. 2.3のフローチャートに示す。

以上が，論理チェッカーの機能の細かな説明である。最初の置き換えはゲートを1対1対応で1ゲートだけ置き換えるため，置き換えは複数ゲートをまとめて調べていないことから処理時間は短い。後の置き換えは同様1対1対応での置き換えであるが複数ゲートを調べているため，非常に時間がかかる。しかし，将来的に複数ゲートをあるブロックと考え，同様の動作をするゲート数の異なるブロックに置き換える処理を行うにはこの機能が，大いに威力を発揮してくる。内部動作の基本的な流れはFig. 2.1のフローチャートに基づいている。

§3. アルゴリズムの概要

ここで報告するアルゴリズムは，全体図面をメッシュに切った升目[5~6]を，ゲート配置領域と配線領域の2種類の領域に分けて考えてゆく。メッシュに切った升目をFig. 3.1に示す。実際には，論理チェッカーはFig. 3.2に示すようなテキスト形式でファイルに格納されている。本論文で提案したアルゴリズムはFig. 3.2のようなテキストファイルを読み込み，自動的に論理ゲートを配置してゆき，端子間を配線してゆき，スケマティックな論理回路図面にして，ディスプレイ，またはプロッタ等に出力するためのものである。

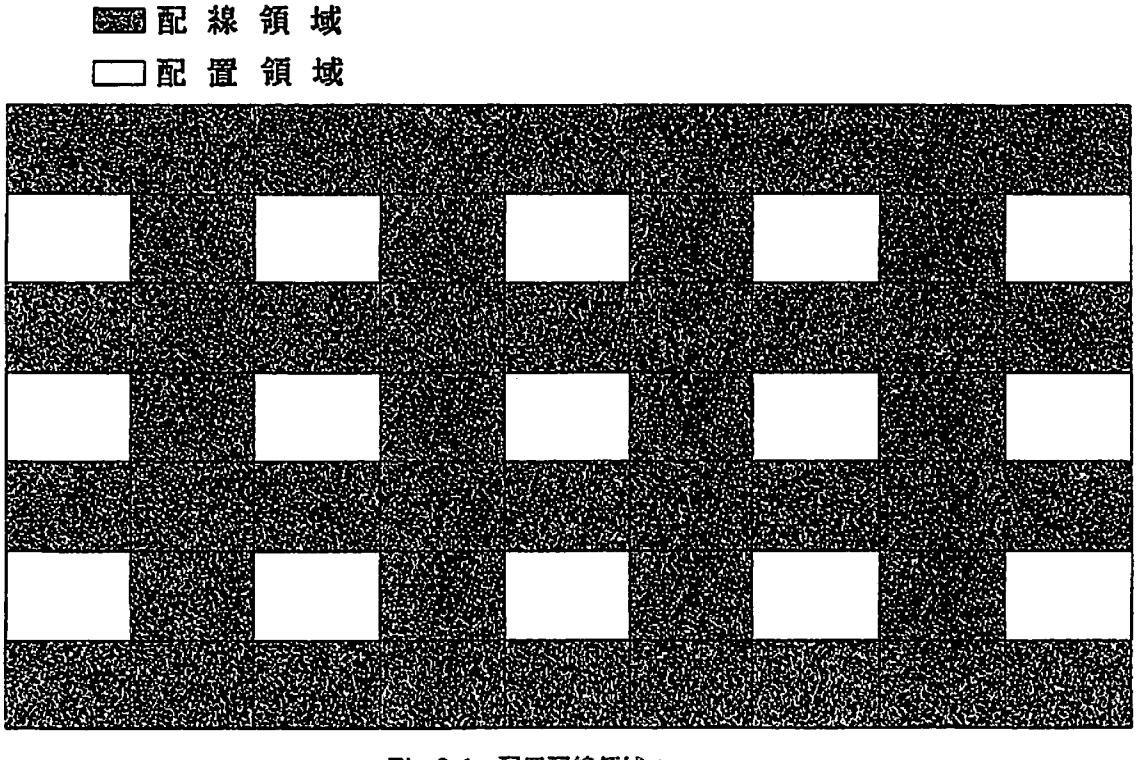


Fig. 3.1　配置配線領域メッシュ

```
4
INV(A:D)
NOR2(B:C:E)
BUFF(E:F)
NAND2(D:F:G)
```

Fig. 3.2　論理チェッカーの出力結果

3.1　配置アルゴリズム

まず最初に，論理ゲートをゲート配置領域に配置するため，論理ゲートにレベルを付けることから始まる。これは，入力端子から見てそのゲートが何段目になるかをレベルで表す。Fig. 3.3で示すように，入力端子Aから見てゆくと，入力端子Aのレベルを1とし，Aに接続されている論理ゲートINVのレベルを2とする。その出力Dが接続しているAND2のレベルを3とする。

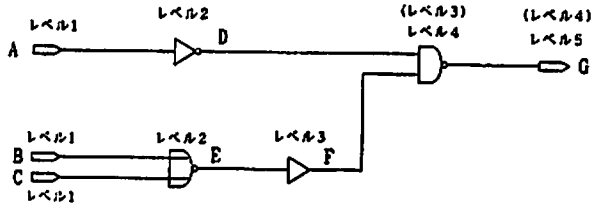


Fig. 3.3　ゲートのレベル付け

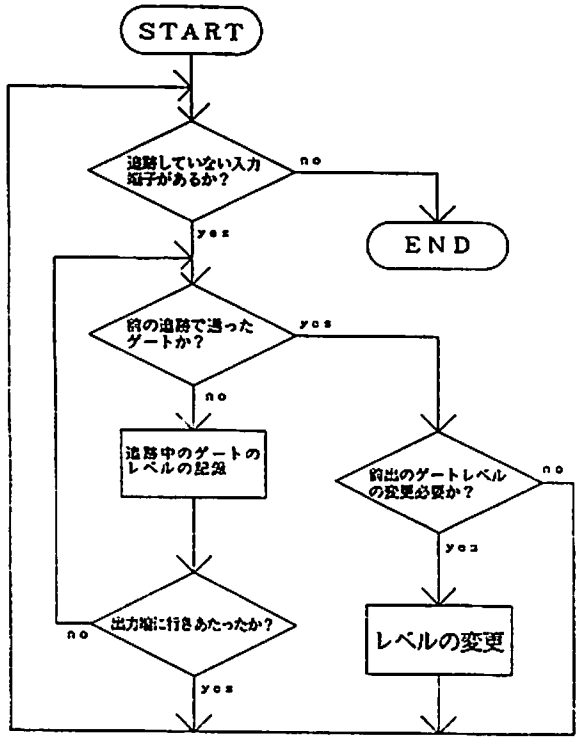


Fig. 3.4　レベル付けのフローチャート

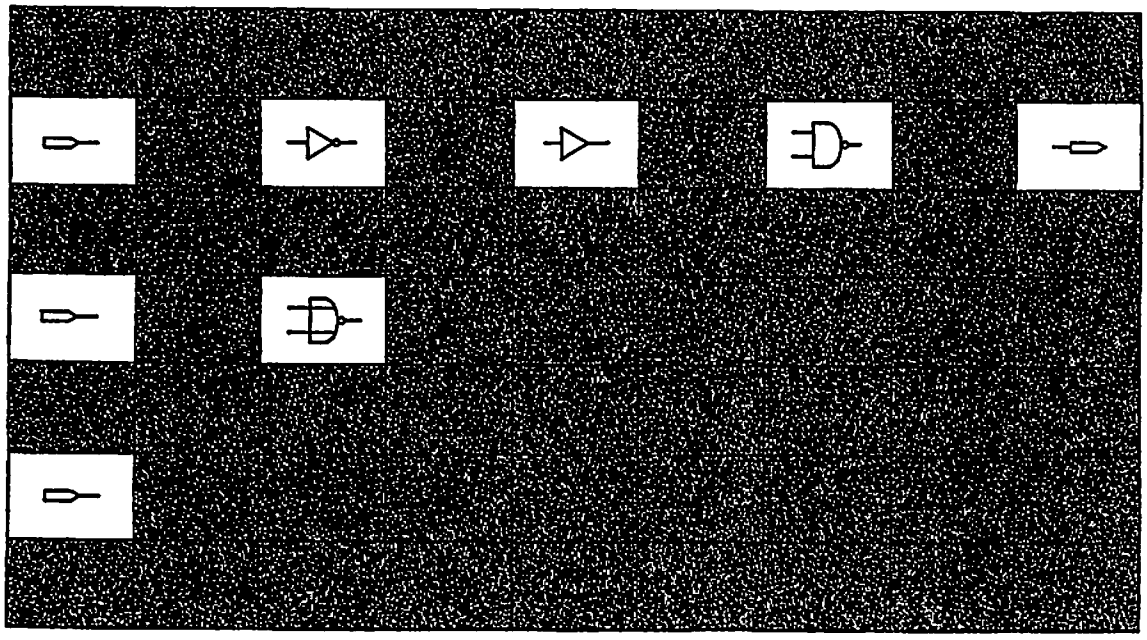


Fig. 3.5　配置領域へのゲートの配置

次に，入力端子Bから見てゆくと，Bに接続する NOR2 のレベルを2として，その出力Eが接続しているBUFFのレベルを3とする。さらに，その出力Fが接続しているAND2は，すでにレベルが3と付いているが，こちら側からみると4段目なので，新たにレベルを4に置き換える。同様に出力端子Gはレベルが4となっていたものも5となる。さらに，入力端子Cから調べると接続している NOR2 のレベルはすでに2と付いていてCから見ても同じなので変更せずにレベル付けを終る。レベル付けのフローチャートをFig. 3.4に示す。その結果，配置はFig. 3.5に示すような配置となる。この方法は，入力端子から信号が伝達する方向へ調べてゆきレベル付けを行い，ゲートを配置してゆくため短時間で，少ないメモリで最適に近いゲート配置が実現できるという特徴を持っている。

3.2 配線アルゴリズム

論理ゲートの配置が終わると，ゲート間の配線を始める。Fig. 3.6 に示すように，配線禁止領域（配置領域）を除いて，配線領域をさらに細かなメッシュに切る。実際には，このような全体図ではなく接続素子間の小領域だけを別なメッシュに複写して処理を行うのだが，全体図面の一部として述べてゆく。そして配線領域のメッシュに対して[7]，迷路法を用いて経路探索のための番号付けを行う。番号付けは，INV の出力端子DからNAND2の入力端子Dへ配線する場合には，NAND2側の端子を0と置き，垂直水平方向に接しているメッシュを1，その次の垂直水平方向に接しているメッシュを2と置いてゆく。そして端子Dにたどり着くまで番号付けを行う。それを表した図を，Fig. 3.7に示す。そして，INVの出力端子Dから逆に，数値の1小さい方向へ水平方向に進んでゆく。もし，1だけ小さい値が存在しない場合は，垂直方向に同様に進んでゆく。ただしこれ以外にもいくつかの条件が存在する。この例のように配置領域を飛び越すよう

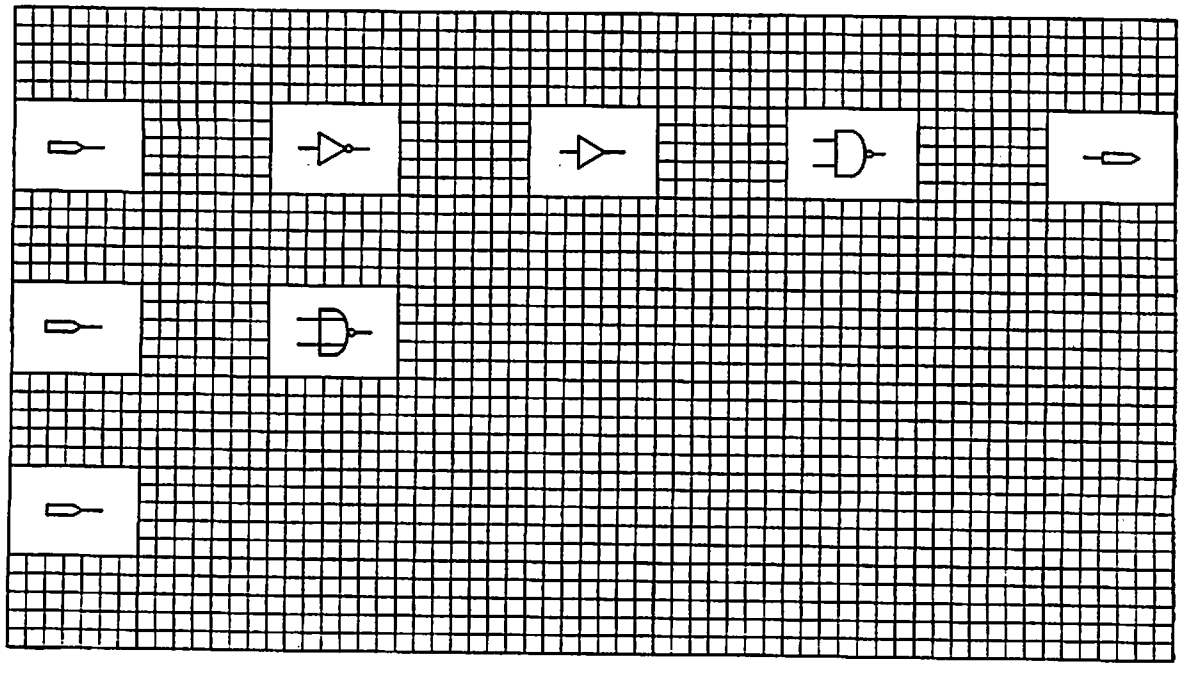


Fig. 3.6　配線領域メッシュ

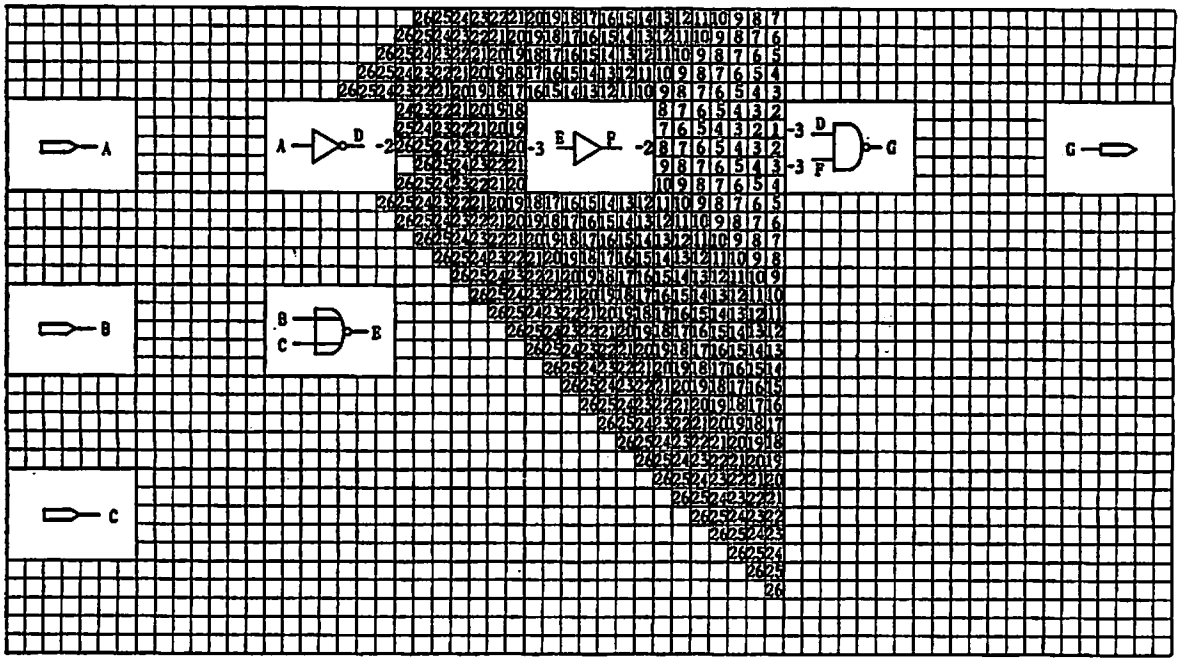


Fig. 3.7　経路探索番号付け

な配線の場合，間にあるゲートの入出力端子の接続点かどうかを判断し，その部分も配線禁止領域と同様に扱う。そして，また1だけ小さい値が存在しなくなった場合に垂直方向へ進む。この繰り返しによって，NAND2の入力端子Dまでを辿ってゆく。この配線アルゴリズムのフローチャートをFig. 3.8に，そして経路追跡結果をFig. 3.9に示す。

このとき，Table 3.1に示すように水平方向に進むメッシュを(−10)，垂直方向を(−11)，水平から垂直への回転を上方向(−13)，下方向(−14)，垂直から水平への回転を上から左(−15)，下から左(−16)のようにメッシュに記録してゆく。すでに配線がされている場合には水平(−10)，

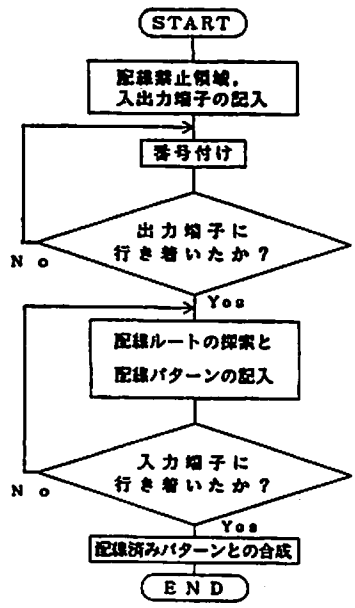


Fig. 3.8　自動配線フローチャート

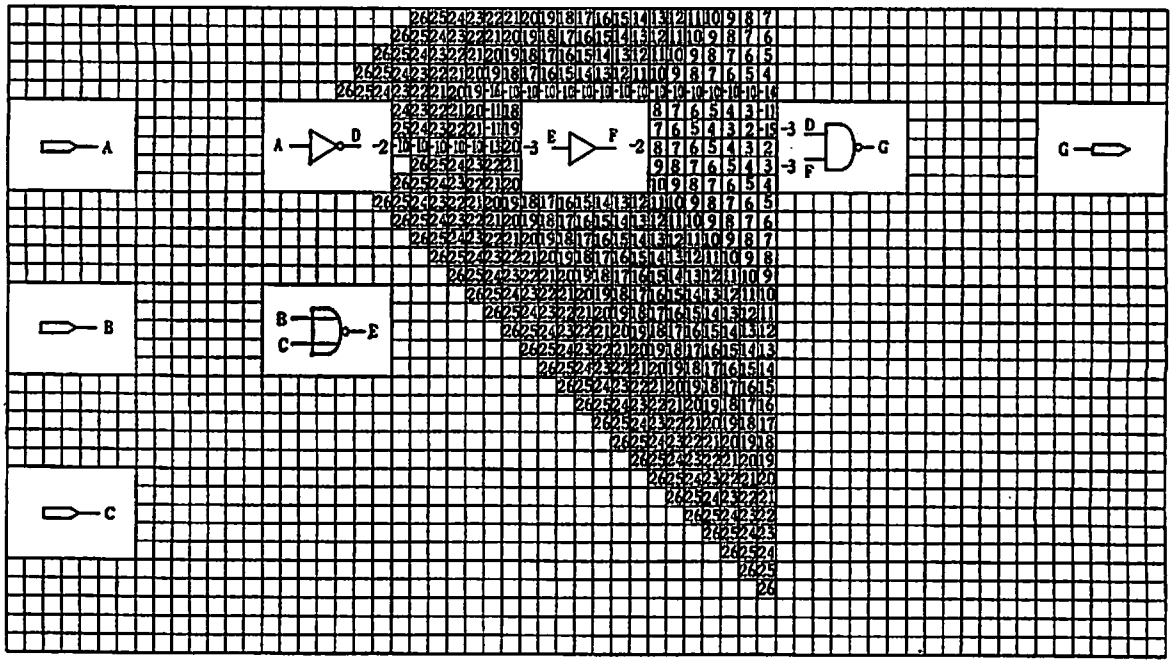


Fig. 3.9　経路追跡番号付け

Table 3.1　経路追跡用番号付け表

経路進行方向	形状	番号	経路進行方向	形状	番号
横方向直進	―	-10	横方向・上分岐	┴	-17
縦方向直進	|	-11	横方向・下分岐	┬	-18
交　差	┼	-12	縦方向・左分岐	┤	-19
左・上転回	┘	-13	縦方向・右分岐	├	-20
左・下転回	┐	-14	縦・横方向交差	┼	-21
上・右転回	└	-15			
下・右転回	┌	-16			

垂直(−11)以外は交差することができないため回転を余儀なくされる。水平(−10)，垂直(−11)の場合には，交差することが可能でメッシュは（−12）と交差を表す値に書き換えられる。また交点に対しては，水平に上からの交点（−17)，水平に下からの交点（−18)，垂直に左からの交点（−19)，水平に右からの交点（−20)，交差する交点（−21）のようにメッシュを書き換えてゆく，従来の手法では，回路図面全体を見て配線を行っていたが，このアルゴリズムは図面全体のうち配線する端子間の狭い部分で処理を行い，全体図面に書き移すという手法を用いているため，理論上は配線ゲート数に制限がなく，メモリの制約はあるが，全体図面をファイルとして扱うことにより相当大規模な図面生成を可能にしているといった特徴がある。

§4. 結　　果

以上のアルゴリズムを基にして，パソコン上でシミュレーションを行った。Fig. 3.3 の回路について配置及び配線シミュレーションを行った結果を Fig. 4 に示す。これは，パソコン上で BA-

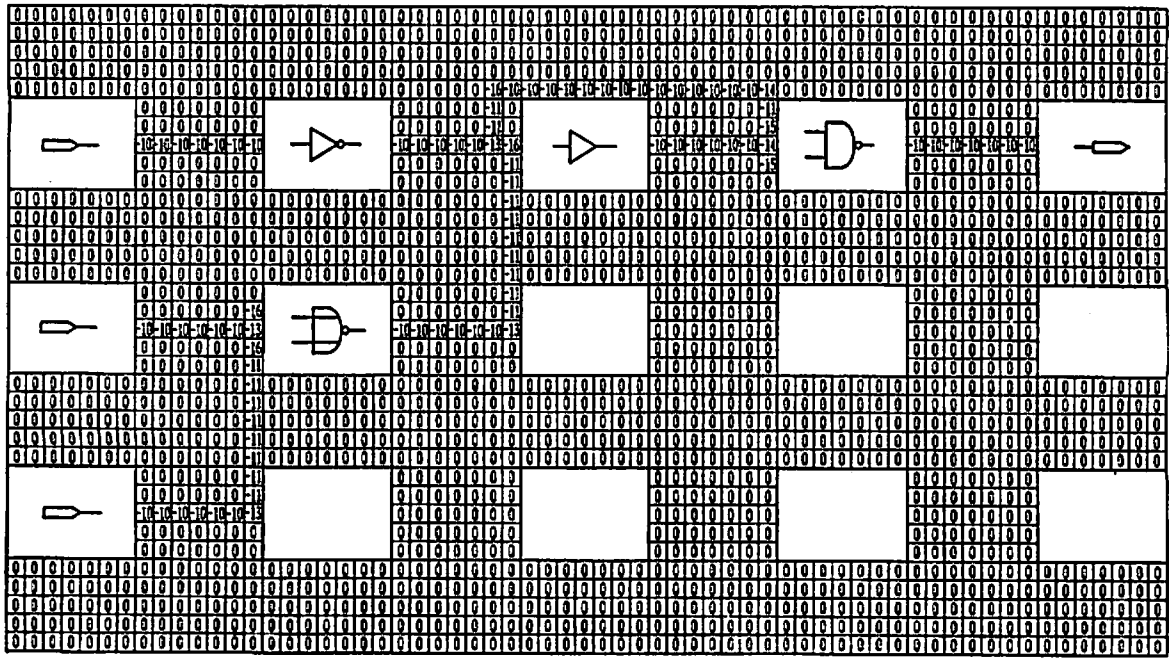


Fig. 4　配線番号付け結果

SIC 言語を用いて実際の配置メッシュを想定し自動的に配置を行ない，配線領域をさらに細分化し配線を行うメッシュの番号付けを行ったものである。ここに示すようにシミュレーションでは，端子間だけを小規模なメッシュで経路追跡を行い，不要な番号は0にしてしまっている。以上，結線，及び配置は間違いなく終了している。このほかにもいくつかのシミュレーションを行ったが，正しい結線が行われていることより，本手法に問題は無いであろうことがわかった。

§5. おわりに

テキスト形式出力から自動的に論理回路図をスケマティックな形式で生成するためのアルゴリズムを考案し，我々の開発している〔論理チェッカー〕の結果を用いて実験を行った結果，自動的に配置配線を行えることがわかった。その結果まだ多少の問題があることがわかった。以下にその点と対策について述べる。

(1) 配置領域を上から詰めてゆくため，配線の時に直進ではなく遠回りする場合がある。このため，再配置を検討している。再配置とは，結果のバッファのように上から詰めてくるため，一段下に配置されれば飛び越し配線も無くなり，すっきりとした図面になると考えられる。

(2) 配置領域でのゲート配置位置が固定のため，直線配線が行えない場合がある。この点に関しては，配置位置の中で多少上下に移動して，配線を直線にできるように微調整を行うことを考えている。それにより少なくとも多入力の場合には，配線の内1本は直線で結べるようになると考えられる。

以上の点を考慮し，今後のアルゴリズムの見直しを行ってゆきたい。

謝　辞

本研究を行うに当たり，ご協力いただきました当研究室・石川，平野両氏，ならびに院生，卒論生に深く感謝します。

参　考　文　献

1) 若林，佐藤，檀：論理チェッカー，論理回路設計への異常回路修正機能の付加，電子情報通信学会VLSI設計技術研究会技術報告，VLD 88-114，3月，1989.

2) S. Wakabayashi, J. Sato and R. Dang: A Cirrecting Eunction for Gate Errors in Logic Simulation. *Proceedings of the 17th IASTED International Symposium*, pp. 163-166, June, 1989.

3) S. Wakabayashi, H. Ishikawa and R. Dang, Logic-Checker: A Correcting function for Gate errors in Logic Simulation and its Implementation. *Proceedings JCS-CSCC'89*, pp. 453-458, June, 1989.

4) M.C. McFarland, A.C. Parker and R. Camposano: Tutorial on high-level synthesis. *Proceedings of the 25th Design Automation Conference*, ACM and IEEE, pp. 330-336, June, 1988.

5) 西尾，真鍋，藤田，宮田：自動論理合成システムLUNAの適用・評価—合成回路図面生成—．第34回情報処理全国大会 4F-8，3月，1989.

6) 真鍋，西尾，宮田：機能論理設計のための図面生成用自動配線アルゴリズム．第33回情報処理全国大会 2R-5，9月，1986.

7) C.Y. Lee: Algorithm for Path Connections and its Applications. *IRE on Trans. on EC*, vol. **EC-10**, no. 3, pp. 346-365, March, 1961.