論理回路図の自動作成*

北村拓郎** 杉本隆夫**

Abstract

It is an important factor in the logical design of a large computer system that a large volume of logic diagrams is automatically prepared with high accuracy and in short turn-around time.

In this paper, a system of automatic preparation of logic diagram is presented which uses the Digityzer and a computer.

The Digityzer generates input data for the computer from the logic diagram drafts which logical designers make.

The computer records logic diagram data on the file in the shape of the image of the logic diagram and maintains the file up-to-dately. The computer can generate the logic connection file for the implementation design phases and supply paper tape data for a plotting machine which draws the logic diagrams for photo prints.

The system can present an efficient tool for change control that has a great significance in the development phase of a computer system. The system has been applied to the design of the NEAC-2200 model 700 computer.

The method that is adopted in the above system can be probably applied with slight modification to the other field of design and management, for examples, the automatic preparation of flow chart of computer programming and PERT chart for job control.

1. はじめに

論理回路図（以後論理図と呼ぶ）は，少数の論理記号，記号間を接続する線分，アルファベットの簡単な基本要素から構成されるものである．しかし，論理図の作成は，計算機などの設計において，次のような理由から，多くの設計工数を要する困難な設計作業となっていた．

（i） 一般に記号間を接続する線分が不規則的に複雑に接続されること．

（ii） 計算機などの装置全体を表わすためには，大量の論理図を必要とすること．

（iii） 論理図の記述に冗長度が少なく，一点の書き誤りといえども致命的な欠点となり，したがって，正確性が極度に要求されること．

（iv） 特に，装置の開発時においては，多くの論理図変更がすみやかに正確に行なわれる必要があること．

したがって，論理図の自動作成の必要性が早くから認識され，すでに 1958 年には実施例の文献発表をみることができ，その後改良された形での自動作成について，いくつかの文献で発表されている[2),3)]．

しかし，従来のこれらの方法では，次のような点で，改良を要すると思われる．

（i） 論理設計者が書いた論理図の原稿から，計算機に入力するためのパンチ・カード・データを作ること（この過程で誤りの混入する可能性が多い）．

（ii） 論理図上の論理記号の配置に制約の多いこと（設計者が論理図の原稿を書くのに不便を感ずる）．

（iii） 論理記号間の接続線のルートを直接的に設計者が制御できないこと（見にくい論理図ができるおそれがある）．

（iv） 接続線のルートを定めるための計算機の処理が複雑になる．

（v） ライン・プリンタによって論理図が出力される（論理記号の表現などが見やすさにかけ，また，写真法による後処理の手数がかかる）．

このたび，われわれは上記のような点に改良を加え

---

* The Automatic Preparation of Logic Diagram, by Takuo Kitamura, Takao Sugimoto (Nippon Electric Co., Ltd., Tokyo)

** 日本電気株式会社・コンピューター開発本部・装置部

ることを考え，入力情報の作成および作図を行なうディジタイザと，論理図データを処理し，ファイル・メンテナンスを行なう計算機システムにより，論理図の自動作成を行なう一方法を開発した．この方法の概要については，すでに報告したが[4]，ここでは計算機による論理図データの処理に重きをおいて，さらに詳しく報告する．

2. 自動作成システムの概要

本稿で述べる論理図の自動作成システムの流れが，**Fig. 1** に示されている．

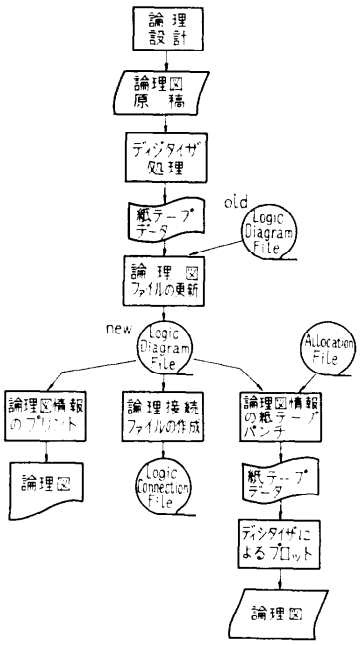

Fig. 1 Flow of Automatic Preparation of Logic Diagram

論理設計の結果として，論理回路が論理図原稿で表現される．論理図原稿は自動作成のための若干の制約条件のもとに，論理設計者によって，論理記号を用いて描かれる．論理図原稿はディジタイザと呼ばれる機械を用いて，操作者によって処理される．

この処理の結果として，論理図原稿に描かれた図面形式の論理図情報が，平面座標を表わす数値と，論理記号を表わす若干の文字に変換されて，計算機へ入力できる形式，すなわち，紙テープ・データとして出力される．この紙テープ・データは，論理図の一葉ごとにまとめられて，計算機システムの論理図ファイルに変換されて格納される．論理図ファイルは，上述の紙テープ・データを計算機の記憶装置に読み込んで，記憶装置の上で論理図の像を作り，この記憶像を磁気テープに移したものである．また，この論理図ファイルからライン・プリンタに論理図の出力が行なわれ，論理図プリントが論理設計者にもどされる．

論理設計者は論理図プリントを調べることによって，論理図が自動作成システムに所定どおり登録されたかどうかをたしかめることができる．設計変更，またはその他の理由によって，論理図に変更が必要とされるときには，論理設計者が修正を要する論理図に関する部分的な論理図原稿を作成する．次いで，ディジタイザの操作によって，修正部分に関する論理図の修正紙テープ・データを作成する．旧論理図ファイルは，修正紙テープ・データによって更新されて，新論理図ファイルが作成される．一般に，論理設計の全期間を通じて，論理図ファイルのデータは，上述のような過程によって，何回も更新登録されて，徐々に累積されていく．

論理図ファイルは，論理図データの更新，蓄積，あるいは図面の出力に適した形であって，論理回路の製造用データ（たとえば，布線，配線データ）を作成するためには，必ずしも適した形式でないので，これを論理接続ファイルに変換する．論理接続ファイルには，論理回路の電気的な接続関係を表わす情報だけが格納される．

論理設計が一応の段落をみせ，論理図データが累積された時点で，論理図ファイルから，論理図をプロッタによって，プロットするための紙テープデータが作成される．紙テープ・データは，プロッタにかけられて，論理図が描かれる．この論理図は，所定のA2版の用紙に描かれ，そのまま青写真用の原紙として用いられる．なお，プロッタはディジタイザの一部に使用されているものを共用することもできる．

論理設計に引き続いて，実装設計が終了した時点では，論理接続ファイルから，この論理図の自動作成システムとは，別のプロセスによって，実装ファイルが作られる．実装ファイルの実装に関する情報（たとえば，論理素子の実装位置情報など）を引き出して，プロッタのための紙テープ・データに追加することによって，論理図の中に，実装情報を含んだ論理図を作成することもできる．

以上で，論理図の自動作成システムの概要を述べたが，本システムのおもな特徴と眼目は，次のようであ

る．

（i） 論理図原稿から直接的にディジタイザによって，計算機の入力用紙テープ・データを作成することができ，従来の論理図原稿からカード・パンチ用原稿を作り，計算機入力用カード・パンチを行なう方法に比べて，一工程を減らすことができる．

（ii） 論理図ファイルが論理図の記憶像をそのまま格納している形式であって，頻繁な論理図データの更新が容易に能率よく行なわれる．

（iii） 論理設計者の確認用として，論理図プリントが，論理図ファイルの更新とともに直ちに得られる．

（iv） 論理図として見やすく取り扱いやすい論理図が，原紙の形でプロッタによって描かれる．

（v） 製造用の各種データを作成するために，論理接続ファイルが論理図ファイルから自動的に作成される．

（vi） 使用する機械としては，4 K語程度の記憶容量を有する小型計算機と，計算機とはオフラインで使用されるディジタイザで十分であって，比較的安価である．

3. 計算機入力データの作成

3.1 論理図原稿の書き方

論理設計者は，3mm ピッチの格子状の罫が引かれているＡ２用紙（使用範囲は 126×156 ピッチ）へ **Fig. 2** に示す機械書きに適した論理記号を使って論理図原稿を作る．

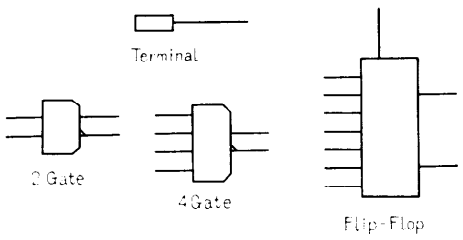

Fig. 2 Some Examples of Logic Symbol

3.2 ディジタイザ

ディジタイザは論理図作成の機械化を目的として製作された装置で，以下の機器で構成されている．

（i） ディジタル X-Y プロッタ　計算機入力用紙テープ・データ作成時には，そのベッド上にセットされた論理図原稿から，論理記号・結線を拾い出すのに使い，作図時にはベッド上の所定用紙に論理図を描くのに使う．

（ii） 穿孔タイプライタ　論理図原稿に書かれている文字情報を紙テープへ出力するのに使う．

（iii） 紙テープ読取機　計算機から出力された紙テープを読み取って，X-Y プロッタに論理図を描かせるのに使う．

（iv） 制御部　上記機器間の接続を制御する部分と，X-Y プロッタのペンを操作する操作ボタン部より成る．

Fig. 3 はディジタイザの全景であり，**Table 1** はディジタイザのおもな仕様である．

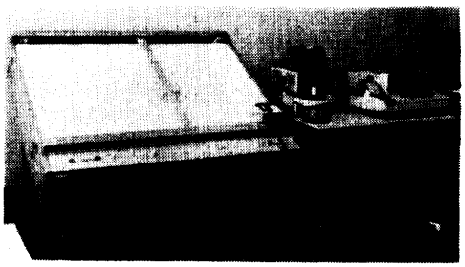

Fig. 3 A View of Digityzer

Table 1 Specification of Digityzer

使用範囲 (mm)	700×650
移動速度 (mm/s)	30
移動方向 (度)	縦・横　±45
精度 (mm)	0.1
読み取りピッチ (mm)	0.1, 1.0, 2.5, 3.0, 5.0

4. 処理プログラム

本論理図自動作成システムの処理プログラムで，特に問題となる点は，次のようである．第１に，論理図のデータを論理図ファイルに登録していく過程において，使いやすく，かつ，処理能率のよいプログラム・システムを作成することである．使いやすいとは，入力データに対する本来は不必要な制限，たとえば，入力データの順序制限などをできるだけ排除し，およそ，人間が紙と鉛筆と消しゴムで，図面を作成し修正していく場合と同様に，入力データを表現しうること，および誤った疑いのある入力データに対しては，default メッセージを出力するとともに，その影響を最小限にとどめ，処理を続行することなどである．第２は，正しく表現された論理図データから，能率よく論理接続データを作成することである．これらの問題を解決するためには，ファイルの形式を適切に定め，処理プログラムのアルゴリズムに工夫が必要となる．

以下に，われわれが採用した各ファイルの形式と，処理プログラムのおもな方法について述べる．

4.1 論理図ファイルの更新

ディジタイザで作られた更新紙テープ・データを，更新に適した形で更新データ・ファイルへ格納した後，このデータによって，論理図ファイルの Comment, Map, Message の各アイテムを更新し，新しい論理図ファイルを作成する．**Fig. 4** は更新処理の流れ図である．

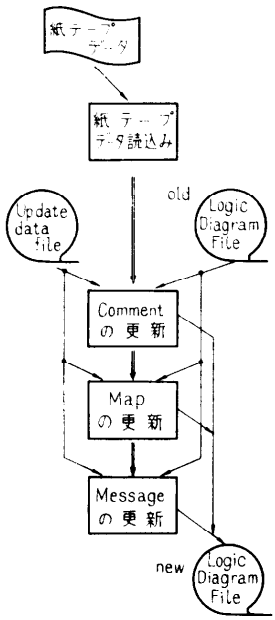

Fig. 4 Flow of Updating Logic Diagram File

4.1.1 ファイル構造

(1) 論理図ファイル

論理図ファイルの構造は，その用途，更新の容易さ，処理速度などを考慮して定められ，磁気テープ上でのデータの配列は，**Fig. 5** に示されるようなものである．

Comment Item 更新に伴う論理図の註釈事項を表わし，アイテム中にある Identifer に従って順序づけられている．

Map Item Map Item の Logic diagram name の箇所には，論理図の名称，版数が格納される．Map の箇所には，論理図の論理記号と結線を，計算機の内部記憶装置へ写像した像のある区画の部分像が格納される．個々のアイテムは，Identifier によって，どの

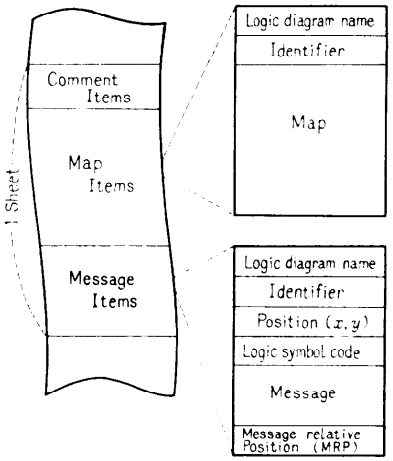

Fig. 5 Stracture of Logic Diagram File

部分像であるか区別される．論理図の全体の像は，いくつかの Map Item から構成されている．

Message Item Message Item の Message の箇所には，論理図上の文字情報が格納され，この文字情報は，必ず論理記号に付随して入力される．文字情報には，たとえば，論理図の入出力端子記号に付加される論理信号名称などがある．Logic symbol code の箇所には，文字情報が付加すべき論理記号コードが記入される．Position には Logic symbol code の図面上での位置座標 (x, y) が格納される．Message relative position には，Message 情報の (x, y) position からの相対位置 (MRP) が格納される．Message Item は (x, y) position によって論理図ファイルの中で順序づけられている．

(2) 更新データ・ファイル

更新データ・ファイルのデータは，論理図ファイルの Comment Item に対する更新データ，Map Item に対する更新データ，Message Item に対する更新データに大別される．これらの各データは，**Fig. 6** のように配列されている．

Map Item に対する更新データは，(i) 論理記号の除去に関するデータ，(ii) 線分の除去に関するデータ，(iii) 論理記号の追加に関するデータ，(iv) 線分の追加に関するデータの順に配列されている．

論理記号の更新データは，論理図名称，Identifier のほかに，図面上における論理記号コード，論理記号の記号原点 (**Fig. 7** における SGP) の位置座標から成っている．線分の更新データは，論理図名称，Identifier のほかに，線分の始点終点を示す 2 組の位置座

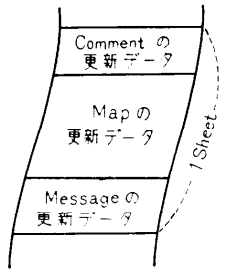

Fig. 6 Date Aligment of Update Data File

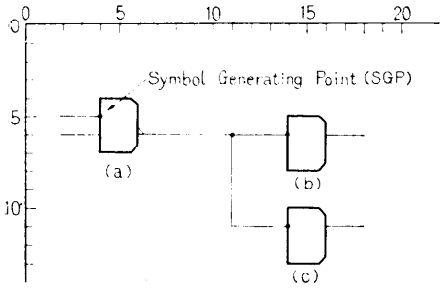

Fig. 7 An Example of Portion of Logic Diagram

標から成っている．

Message Item, Comment Item に対する更新データは，論理図ファイルの Message Item, Comment Item と同様の形式で，更新データ・ファイルに格納されいてる．Comment Item に対する更新データは，Identifier を第1鍵語，制御情報（論理図ファイルへ追加すべきデータか，除去すべきデータかを示す情報）を第2鍵語として分類され配列されている．Message Item に対する更新データは，(x, y) position の位置情報を第1鍵語，制御情報を第2鍵語として分類され配列されている．

4.1.2 Map 構造

論理図原稿の格子上に描かれている論理記号および結線が，計算機の内部記憶装置へ写像された像を Map と呼ぶ．

論理図の一格子の状態を表わすために6ビットを用いる．この6ビットを Mesh code と呼び，Map における Mesh code の格納される場所を Mesh と呼ぶ．A 2 版の論理図原稿は，126×156 格子点を有しているから，これを写像して Map を作るためには，19,656 Mesh を必要とする．

Mesh code の種類は $2^6=64$ であるが，このうちおもなものの意味は，**Table 2** に示されている．Mesh code によって論理図の各格子点の状態が一義的に定められるので．Map から論理図の文字情報を除く図面を構成することができる．**Fig. 8** は Fig. 7 の論理図の一部から，Map の一部を作成した例である．Map の各 Mesh は内部記憶装置の各番地に規則的に対応している．

Table 2 Some Examples of Mesh Code

コード（8進数）	機能	コード（8進数）	機能
00	ブランク	30	否定
01	横の線素	31	論理記号の入力点
02	縦の線素	〃	省略
03	マップの境界を示す	40	記号の下側 *1
04	2 Gate	41	記号の左右側 *2
05	4 Gate	42	記号の上下側 *3
06	6 Gate	43	記号の左右側
07	8 Gate	44	記号の左上角
10	論理記号の入力点	45	記号の上下側
11	⊣ 分岐点	46	記号の内部
12	⊣ 分岐点	47	記号の右側 *4
13	⊣⊢ 分岐点		
〃	省略		省略

＊1：入力線接続可能点　＊3：出力線接続可能点
＊2：入出力線接続可能点　＊4：入力線接続可能点

4.1.3 論理図ファイルの更新

論理図ファイルの更新の大きな流れは Fig. 4 に示されている．紙テープの更新データを更新データ・ファイルに変換した後に，Comment の更新，Map の更新，Message の更新の順序で，論理図ファイルを更新データ・ファイルによって更新し，新しい論理図ファイルを作成する．

（1） **Comment Item の更新**

論理図ファイルの Comment Item と，更新データ

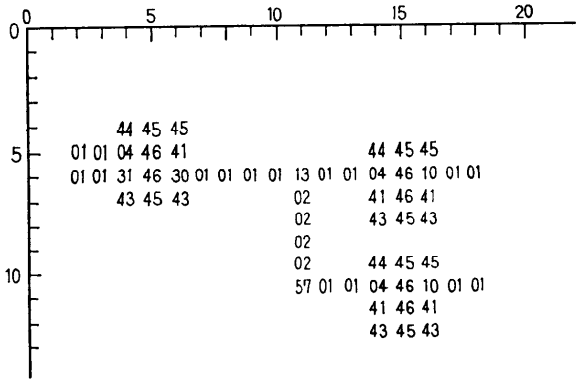

Fig. 8 An Example of Portion of Map on Memory

ファイルの Comment Item の Identifier を比較して，更新データの制御情報に従って論理図ファイルの対応アイテムを更新する．

（2） Map Item の更新

論理図ファイルから，一葉の論理図のすべての Map Item の Map 情報を，内部記憶装置に移して Map を作成し，更新データ・ファイルの論理記号と線分に関する更新データで，Map の対応 Mesh を変更する．引き続いて，Map を論理図ファイルのすべての Map Item に変換して，Map Item の更新を行なう．ここで注意すべきことは，ごく少量の更新データに対しても，すべての Map Item が Map に変換され，再び Map からすべての Map Item が作られることである．

Map の変更は更新データの配列順序と同様に，次のような順序で行なわれる．

（i） 論理記号に関する除去
（ii） 線分に関する除去
（iii） 論理記号に関する追加
（iv） 線分に関する追加

論理記号の更新 論理記号の更新は，次のような方法によって行なわれ，**Fig. 9** に図示されている．

（i） 更新しようとする論理記号の記号コードで，論理記号を表わす Mesh code を論理記号テーブルから求める．論理記号テーブルは，あらかじめ用意されたテーブルで，各論理記号に対し，各論理記号を表わす Mesh code を，記号の形状に並べて用意しておく．すなわち，論理記号テーブルには，論理記号の Map が用意されている [Fig. 9 の（a），（b）]．

（ii） ふるい Map の更新しようとする Mesh にある Mesh code を取り出す [Fig. 9 の（c），（d）]．

（iii） （i）で求められた論理記号テーブルの Mesh code とふるい Map の Mesh code から，新しく Map に挿入すべき Mesh code を，論理記号更新テーブルを利用して定め，いま問題となっている Map の Mesh を，この Mesh code に置き換える．論理記号更新テーブルには，除去テーブルと追加テーブルがあり，それぞれ2次元のテーブルで，論理記号テーブルの Mesh code とふるい Map の Mesh code とを，テーブルの2つの相対番地の形で与えれば，求める Mesh code が得られるものである．このテーブルによって，たとえば，すでに引かれている線分の上に論理記号が置かれるようなときは，Erorr code に置き換えるなどの Error 検出も行なわれる [Fig. 9 の（b），（d），（e）]．

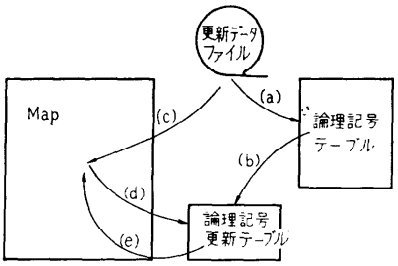

Fig. 9 Updating Diagram of Logic Symbol

（iv） （ii），（iii）の各過程を（i）で求めた論理記号の部分 Map に従って行なう．

線分の更新 線分の更新データに含まれている線分の始点座標から終点座標までの Map 上の線分の Mesh に対して更新を行なう．

この方法は，論理記号の更新の場合と，ほぼ同様である．ただし，論理記号テーブルのようなものは必要とせず，更新データから直接的に，更新データの Mesh と Mesh code を計算する．論理記号更新テーブルに対応するものは，線分更新テーブルであるが，どちらも，ほぼ同様な機能を持っている．

（3） Message Item の更新

Message Item の更新は，Comment Item の更新とほぼ同様である．ただし，論理記号に付加する Message の相対位置（MRP）に関する特殊な処理が伴うけれども，詳細については省略する．

4.2 論理接続ファイルの作成

論理接続ファイルの接続情報に関して，Fig. 7 の論理回路を例にとって説明する．

Fig. 7 の論理回路の（a）の論理記号から（b），（c）の論理記号への接続に関しては，論理接続ファイルでは，次のような接続情報で表わされる．

$$\underset{(1)}{BB}\ \underset{(2)}{B}\ \underset{(3)}{2\,NAND} \rightarrow \underset{(4)}{CE}\ \underset{(5)}{1}\ \underset{(6)}{1\,AND},\quad \underset{(7)}{DE}\ \underset{(8)}{1}\ \underset{(9)}{1\,AND}$$

（1），（4），（7）の BB, CE, DE はブロック位置と呼び，論理図を3ピッチ単位で区切った領域を示す．（1）BB は論理記号（a）の記号原点（SGP）の存在する領域である．同じく（4）CE,（7）DE は論理記号（b），（c）の記号原点（SGP）の存在する領域である．（5）の1は論理記号（b）の第一入力であることを示す．（6）の1 AND,（9）の 1 AND は論理記号（b），（c）

が1入力を有するAND回路であることを示す．(8)の1は論理記号(c)の第一入力であることを示す．

以上で，論理記号(a)の否定出力が論理記号(b)，(c)の第一入力に接続されていることが表わされ，論理図上の領域で論理記号(a)，(b)，(c)が一義的に区別され，各論理記号が2NAND, 1ANDなどの文字情報で表わされている（この2NAND, 1ANDなどの文字情報を論理タイプと呼ぶ）．

論理接続ファイルでは，論理図上のすべての論理接続が，上述のようにして表わされている．

論理図ファイルから論理接続ファイルを作成する大要は次のようであり，**Fig. 10** に図示されている．

(i) 論理図ファイルから，計算機の内部記憶装置に Map を作る．

(ii) Map を走査して，論理記号に対応する論理タイプと，論理記号の記号原点 (SGP) を算出する．

(iii) さらにもう一度 Map を走査し，論理記号間の接続関係をトレースして接続情報を作る．

(iv) 論理タイプと接続情報を編集して，論理接続ファイルとする．

4.2.1 論理タイプの算出

Map を走査して，論理記号の原点 (SGP) を示す Mesh を捜し，記号原点から論理記号の形状を Map の上でトレースして，論理記号の入力線数と出力の接続状態（否定，肯定などの区別など）を求める．論理記号の論理タイプは，論理記号の入出力の状態と，記号原点(SGP)の Mesh code によって定められ，記号原点の座標と合わせて中間データとして Work File へ出力する．たとえば，Fig 7 の回路の場合には，Fig. 8 の Map から，座標 (4, 5) の Mesh code から，論理タイプ 2NAND が求められ，同じく座標 (14, 6)，(14, 11) の Mesh code 04 から，論理タイプ 1AND が求められる．

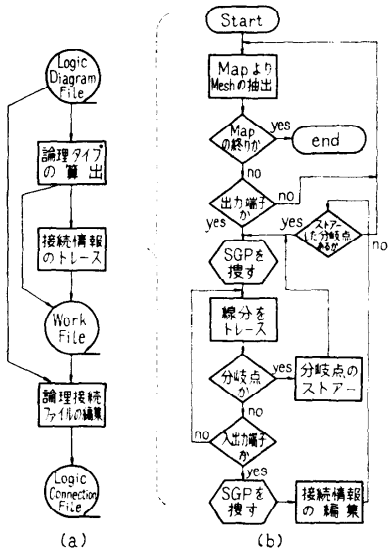

Fig. 10 Flow of Generating Logie Connection File

4.2.2 接続情報の作成

接続情報の作成は Fig. 10 の(b)のような流れに従って，次のように行なわれる．

(i) Map を走査して，論理記号の出力端子を示す Mesh を捜す．

(ii) (i)で求められた出力端子が属する論理記号の記号原点 (SGP) を記号の縁をたどって捜し，記号原点 (SGP) の座標と，出力端子の記号原点 (SGP) に対する相対座標を求める．

(iii) 出力端子に接続している線分を示す Mesh を，他の接続している論理記号の入力または出力端子までトレースする．このとき，接続線が分岐を示すような Mesh [たとえば，Fig. 8 の座標 (11, 6)] では，分岐点に関する処理が必要になる．

(iv) 線分に対するトレースが行なわれて，論理記号の入力点または出力点に至ると，(ii)と同じようにして記号原点 (SGP) を求めて，その座標と入出力端子の記号原点からの相対座標を算出し，接続情報として Work File に出力する．

次に，線分のトレースにおける分岐点に対する処理方法について，さらに詳しく説明する．

分岐点に対する処理は，DIR レジスタ，Direction テーブル，Branch Point Stack を使用して行なわれ **Fig. 11** にこの様子が図示されている．

論理記号の出力端子から線分をトレースする前に，トレースすべき方向を DIR に記憶しておく．DIR の

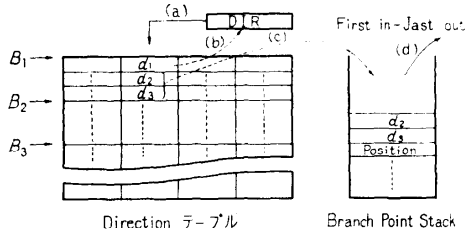

Fig. 11 Direction Table and Branch Point Stack (d_n: direction)

示す方向にトレースを行ない分岐点に達すると，分岐点の Mesh code (B_i とする) と DIR の内容とから，Direction テーブルを引いて，次にトレースすべき方向を求め DIR に格納し，さらに，分岐点からのびている他の方向を，分岐点の座標とともに Branch Point Stack に格納する．Direction テーブルでは，分岐点の Mesh code と，現在トレースしてきた方向である DIR の内容が与えられれば，分岐点からトレースすべき方向のいくつかの d_j が求まるようにテーブルの構造が定まっている．このうちの一つの d_j を DIR に格納し，他の d_j は Branch Point Stack に格納すればよい．

Branch Point Stack は First in-last out の stack memory で，分岐点が多重に存在した場合においても，簡単に処理できることを目的としている．

分岐点から一つの方向にトレースが行なわれ，論理記号の入出端子に至った場合には，求められた接続情報の出力処理を行なった後，Branch Point Stack を調べ，空でなければ stack の中にある d_j を一つ読み出して DIR に格納し，stack に貯えられている位置座標からトレースを続ける．この stack から読み出された d_j は stack から消去される．トレースの終了は Branch Point Stack の内容が空になったことを検出したときである．

以上のようにして求められた接続情報へ，前節で求めた論理タイプを編集して論理接続ファイルとする．この処理に関しては，その処理内容の説明は省略する．

5. 適 用 例

以上で述べた論理図自動作成システムを，実際に細部にわたる検討を行なって，論理図の描き方，ディジ

Table 3 Program Steps and the Time Required for a Sheet

プログラム名	論理図ファイルの更新	論理接続ファイルの作成	論理図情報のプリント	論理図情報の紙テープ・パンチ
ステップ	3,900 cards (assembler)	1,000 cards (assembler)	1,500 cards (assembler)	1,600 cards (assembler)
時 間 (min)	3～13	7	6	9
ディジタイザ	ディジタイザによる論理図の処理時間		ディジタイザによる論理図作図時間	
時 間 (hr)	約 2		約 1	

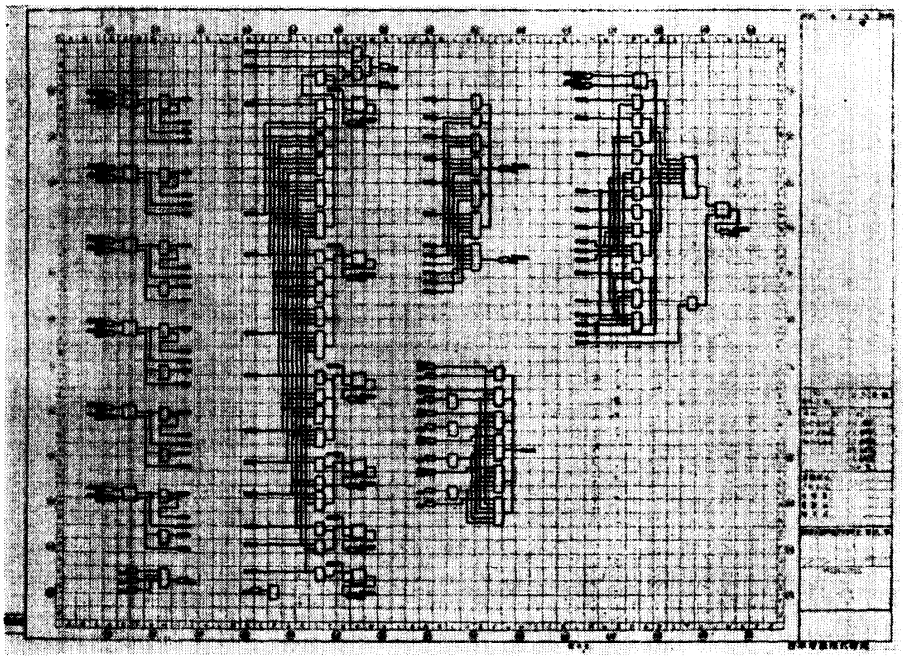

Fig. 12 An Example of Plotted Logic Diagram

タイザの使用法を定め，処理プログラムの作成を行なって具体化し，NEAC-2200 モデル 700 の論理設計に使用した．

使用した計算機は，4K語（1語 48 ビット）の記憶容量と，5台の磁気テープ・ユニットを有する NEAC-2400 である．

この場合の適用例における処理プログラムの規模，論理図1葉当りの処理時間，ディジタイザの操作時間は，**Table 3** のようである．論理図のプロッタとプリンタに出力された一例が **Fig. 12** と **Fig. 13** に示されている．

本システムの適用において，特に留意または苦心した点は，次のようなことである．第1に，論理図の描き方は従来の慣行を尊重し，かつ，機械書きに適した表現とするための工夫をすること．第2に，使用する計算機としては，小型計算機を専有化して使用することとしたが，このために，小さな記憶装置で処理能率のよいプログラムを作成すること．このために，FORTRAN などの高レベル言語は使用せず Assembler による巧妙なプログラム手法を駆使することに努力した．

次に，処理能力の点について考える．初版の論理図データはチェックも含めて，ディジタイザによって約2時間／葉を要して作成される．これを従来のカード・パンチ原稿，カード・パンチ方式のデータに換算すれば，カード約 500～700 枚程度のものとなりカード・パンチ原稿の書き誤りの発生などを除外したとしても，ディジタイザ方式が十分に経済性のあることがわかる．現在，ディジタイザを3台設置し，NEAC-2400 計算機を使用して，初版論理図で，1日当り 10 葉程度処理している．

運用形態は，IC 化計算機の論理設計に適したものとし，最大 60 個，平均 40 個程度の IC を搭載する IC 化論理パッケージが設計されるに従って，1パッ

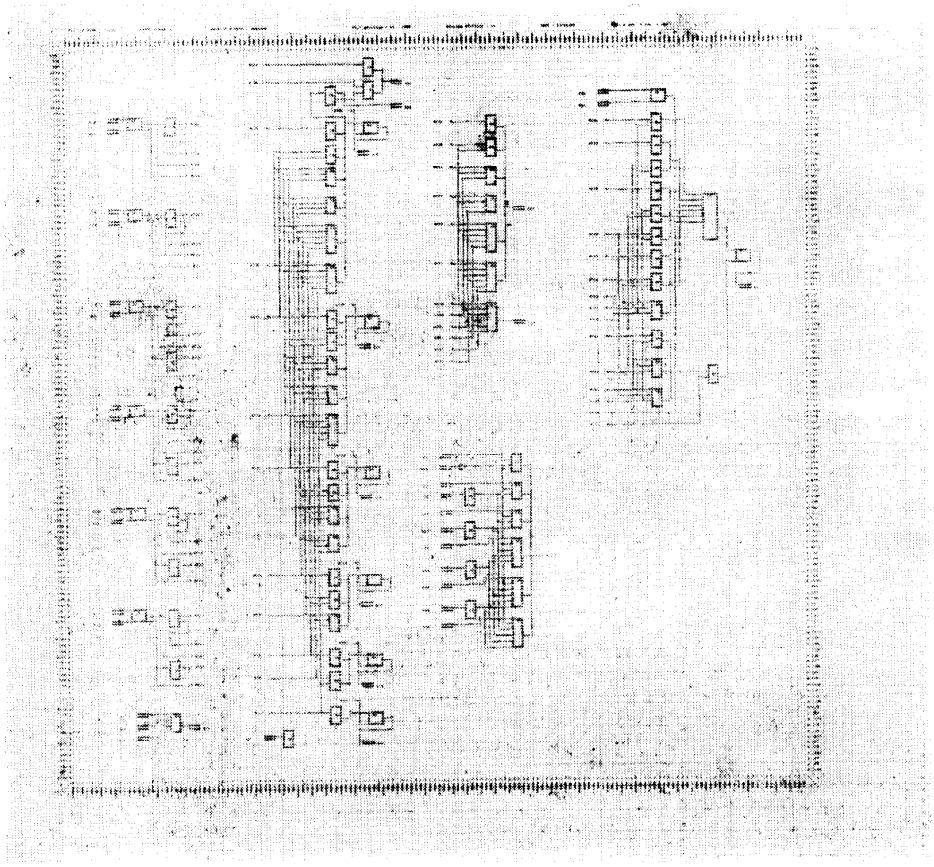

Fig. 13 An Example of Printe Logic Diagram

ケージの大部分は1葉の，残りは2葉の論理図で表現されるパッケージ論理図として登録される．パッケージ論理図は計算機システム全体では300種類に及ぶ．単位装置に関するパッケージ論理図のすべてが登録されると，単位装置の装置論理図が作成される．装置論理図の作成のために，特定のパッケージを装置論理図に定義するデータと，パッケージ間の論理接続関係を表わすデータは，カード形式で入力され，別途処理されて本稿で述べた自動作成システムに融合される．

6. むすび

論理回路図の自動作成の一方法について報告した．この方法は，計算機用入力データの作成機械と作図機械として専用のディジタイザを，回路図データのデータ処理機械として計算機を用い，大量の回路図データを正確に，迅速に処理して質のよい論理回路図を提供することを目的とするものである．

この方法は実際にNEAC-2200/700計算機の設計に適用されて，ほぼ所期の目的が達せられることがたしかめられた．

今後の改良点としては，現在，人間によって操作されて，計算機用の入力データを作成しているディジタイザをさらに自動化して，論理回路図原稿を自動的にトレースすることが考えられる．

本稿で述べたものとは別のアプローチの方法としては，経済性の問題などが解決すれば，オンライン化したグラフィック・ディスプレイを使用した論理回路図の自動作成も考えられよう．この場合のプログラム処理についても，入出力処理の部分を除けば，本稿の考え方は大部分適用できる．また，処理プログラムの方法として，本稿では，記憶装置のMapを中心とした処理が重要な役割をしているが，これに代わる方法としては，List形式による論理図データの表現が考えられる．大型計算機などを使用してList構造を記述するに適したプログラム言語が使用できるときには，有力な方法となろう．

論理回路図は，少数の記号と線分との簡単な基本要素から構成される図面であるが，これと同じような図面は他にもいくつか考えられる．たとえば，論理設計で用いられるブロック・チャート，フロー・チャート，計算機のプログラミングのために用いられるフロー・チャート，日程管理に用いられるパート線図など．これらの図面についても多少の修正でもって，本稿で述べた論理回路図の自動作成の方法と同じような方法が適用できるものと思われる．

終わりに，終始ご指導・ご鞭達をいただいたコンピューター開発本部装置部の宮城部長，小林課長および本自動化システムの開発と実用化に協力された関係者の方々に感謝の意を表する次第である．

参考文献

1) M. Kloomok, P. W. Case, and H. H. Graff: The Recording Checking, and Printing of Logic Diagram, Proceeding of the EJCC, p. 108 (December 1958).
2) P. W. Case, et al.: Solid Logic Design Automation, IBM Journal, p. 127 (April 1964).
3) 山田 博，他：論理回路の布線表，回路図の自動作成について，電子計算機研究会資料（1966年9月）．
4) 北村拓郎，他：論理回路図の自動作成，昭和43年度電子通信学会全国大会 S2—1.

（昭和44年5月19日受付）