$$\left.\begin{aligned}\varepsilon_y{}^{i+1}&=-\frac{h^5}{5!}z_i+\frac{h^6}{6!}y_i+\frac{h^7}{7!}z_i-\cdots\\ \varepsilon_z{}^{i+1}&=+\frac{h^5}{5!}y_i+\frac{h^6}{6!}z_i-\frac{h^7}{7!}y_i-\cdots\end{aligned}\right\}\quad(A\cdot1)$$

(A·1) を (2·3) に代入し，次の関係

$x_{i+1}=x_i+h$, $y_i\simeq r\sin x_i$, $z_i\simeq r\cos x_i$

を用いて整理すると，次式が得られる．

$$\varepsilon_r\simeq-\frac{h^6}{120}\left[\left(\frac{\sin h}{h}-\frac{\cos h}{6}\right)-\frac{h^2}{42}\left(\frac{\sin h}{h}\right)-\frac{\cos h}{8}+\cdots\right]\quad(A\cdot2)$$

$$r\varepsilon_\theta\simeq-\frac{h^5}{120}\left(\cos h+\frac{h\sin h}{6}-\frac{h^2}{42}\cos h-\cdots\right)$$

附　録 2.　丸めの誤差の規則的集積

打切り誤差が無視できるときには，次の階差方程式が得られる．

$$\Delta y=z\Delta x+\varepsilon_y'$$
$$-\Delta z=y\Delta x+\varepsilon_z'$$

ここで ε_y' および ε_z' は丸めの誤差である ((2·1) 参照)．

次の関係

$$\varepsilon_y'=\delta_1 z,\quad \varepsilon_z'=\delta_2 y$$

が成立するとすれば，次の関係式が得られる．

$$\frac{\Delta y}{\Delta x}=z\left(1+\frac{\delta_1}{\Delta x}\right),\quad -\frac{\Delta z}{\Delta x}=y\left(1+\frac{\delta_2}{\Delta x}\right)$$

したがって次の解が得られる．

$$y\simeq A\sin\left(1+\frac{\delta_1+\delta_2}{2\Delta x}\right)x$$

$$z\simeq A\cos\left(1+\frac{\delta_1+\delta_2}{2\Delta x}\right)x,$$

すなわち，$\Delta x=h$ とおいて

$$\left.\begin{aligned}r&\simeq A,\\ \theta&\simeq\left(1+\frac{\delta_1+\delta_2}{2h}\right)x_0\end{aligned}\right\}\quad(A\cdot3)$$

集積誤差は

$$\left.\begin{aligned}\varepsilon_r&\simeq0,\\ \varepsilon_\theta&\simeq\frac{\delta_1+\delta_2}{2h}x_0\end{aligned}\right\}\quad(A\cdot4)$$

電子計算機の配線に関する自動データ処理*

淵　一博**　西野博二**

1. はじめに

1台の計算機を作り上げる過程に尨大な量のデータ処理が伴うことは，計算機製作を実地に経験した人すべてが痛感するところであろう．一方計算機自体に，そのようなデータ処理をもとり込む能力があるのは論をまたない．既にある計算機を利用できるとき，新しい計算機の製作に関するデータ処理をそれにまかせようとするのは極めて当然のなりゆきである．

電気試験所で現在調整中の ETL Mark 4 B[1] の製作に当って，そのデータ処理の大きな部分を占める“配線”指令書の作製を ETL Mark 4 A に行わせる試みをしたので，その経験を報告する．それがますます広範囲な自動データ処理の発展にいささかなりとも役立てば幸いである．

2. 計算機製作におけるデータ処理

一つの製品を完成する過程には，研究，計画，設計に始まって，製作，検査に至るまで，極めて多種多様の作業が含まれている．実験的な小規模の製品の場合は，これらの作業を意識的に分析しなくても済むことも多い．しかし，電子計算機では試作の場合でも，作業を意識的に分析し，その上に立って作業手順を再構成しなければならないくらい大量な作業量があり，この中には幾多の情報処理過程が存在する．

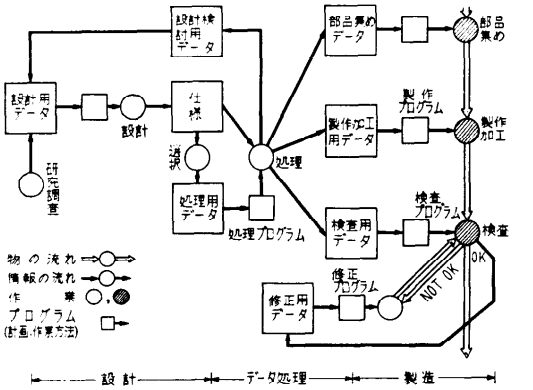
---

* Automatic Data Processing in the Wiring of Digital Computers, by Kazuhiro Fuchi and Hiroji Nishino (Electrotechnical Laboratory, Tokyo)

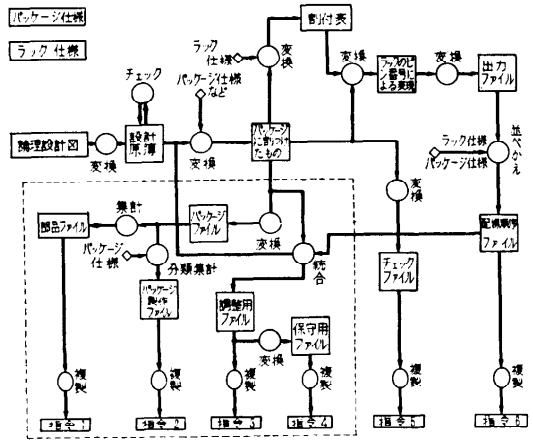
** 電気試験所電子部

ここで“製作におけるデータ処理”として取上げようとするのは，設計（図）に表現された情報を，実際の製造作業への指令という形の情報に変換する過程である．

全体の過程を一般的に流れ図にまとめれば**第1図**のようになる．円で表わした各作業過程には，作業指令としてのデータが与えられる．その主な指令データ作製の過程は**第2図**のように構成できる．この作業のど



第 1 図　製作過程の流れ図



第 2 図　データ処理の流れ図

の部分まで自動化するかは，利用できるデータ処理装置を考慮したデータ処理計画によってきまるわけである．

われわれが実際に計算機で行ったのは，そのデータ処理の中で最も大きな部分を占める配線指令の作製の自動化であり，第2図で点線で囲んだ以外の部分である．次節で配線指令の作製に関する理論的な考察をし，4節で実際の作業について述べる．

3. 配線問題

3・1 条件のない最短配線の問題[2]

最短配線の問題は，n 個の端子が与えられたとき，それをお互に電気的に接続し，しかも全体の配線の長さを最小にすることである．無意味な組合せを除けば組合せの数は有限であるから時間を厭わなければ，一つ一つ調べることによって最短になる配線法は確かに見つかる．しかし，これは実際には実行不可能である（例えば，$10!=3.6\times16^6$，$20!=2.4\times10^{18}$ であるから）．そこで，**確かに**最短な配線が求まる**有効な**手続きを得るということが問題になる．

端子からの線の分岐数に制限のない場合には，一つの解がある．これは NBS（アメリカ）で実際に用いられた[2),3)]．これは筆者等の方法と関連があるので，その方法の基礎となる基本定理だけを簡単に紹介しておく．

まず言葉の定義を述べる．端子を node，任意の二つの node の間の接続を branch と呼ぶ．一つの branch にはその「長さ」が与えられている．

二つの node の間に唯一つの path しかないとき，この node と branch の集りを tree と呼ぶ．node の部分集合がそれだけで tree をなしているとき，subtree と呼ぶ．tree の長さは branch の長さの合計であるが，最小の長さを持つ tree を minimum tree と呼ぶ．minimum tree は一つとは限らない．

定理　n 個の node があり，そのうち k 個が subtree をなしていて，しかも minimum tree の一部であるとする．この k 個の node のうち任意の1個と，残りの $n-k$ 個の node のうち任意の1個の間の branch のうちで最短のものは，minimum tree に含まれなければならない．

上の条件を満す最短の branch で等しい長さのものがいくつかある場合は，それぞれ，k-node の subtree を含むいずれかの minimum tree（いくつかある）の branch をなす．

この定理によって，最短配線，前記の言葉でいえば minimum tree を求める手続きが構成でき，それに従って計算機のプログラムを作ることができる．この方法で注目すべきことは，branch の長さ自体は問題にならず，その長さの順序だけで minimum tree が定まるという点である．NBS で上記の方法を用いて，計算機 Pilot の配線を求めるのに，IBM 704 を使って全体として約 200 時間要した由である[3)]．

3・2 条件付最短配線の問題

node に接続される branch の数が制限される時は前節の方法は使えない．ひとつの node に接続できる branch の最大数を C とすれば，前節の問題は $C=n-1$ であった．一般に $2 \leqq C \leqq n-1$ の任意の C の値で，問題が解ければそれに越したことはないが，それが難しければせめて $C=2$ の場合で，問題が解けないであろうか．この場合の最短配線は，前節で求める minimum tree より長くなることは明らかであるが，別の理由から $C=2$ の方が都合がよいことがある．

$C=2$ の場合は簡単にいえば，n 個の node を1本の線で結ぶことである*．これは実際の配線上の利益から，われわれの欲するものである．1本の線で結ぶことは，配線，ハンダ付の手間からみて一番よい（誤りを冒す可能性が一番少ない）．また，C は2を越えないのだから，配線ないしは配線指令書の誤りのチェックになる．これは ETL Mk 4A および 4B の経験では有効であった．

われわれの研究室では，配線指令書は各出力端子ごとに上の条件で作り，それに従って配線するから $C=2$ のチェックは配線作業中にできる．それと，別に用意した各入力端子ごとの，それに対する出力端子の表によるチェック（配線順序に無関係なチェック）を併用して（もちろんその際発見した誤りを直した結果）Mk 4A および Mk 4B の配線は誤りが皆無に近かった．これは計算機の製作の点では馬鹿にならない利点である．この配線法の欠点は，配線が幾分長くなるために配線のインダクタンスや漂遊容量が増加することであるが，クロック周波数 200 kc 程度の中速度の計算機では実際上まだ問題にならない．

しかし，$C=2$ でも**確かに最短の得られる**（数学的に厳密な）有効な手続きが発見できなかった．しかし配線を自動的に求めたいという気持は強かったので，近似的な解を求めた．

（a） 筆者等の方法（1）

第1近似として次のものを採用する．n 個の node のうちから適当に出発点を選び，それに最も近い node を探す．次に残りの $(n-2)$ 個のうち最も近いものをさがす．これを繰返し，node の一系列を得る．これを基本解と呼ぶ．

次にこれを改良することを考える．可能な配線は，1列に並べるのだから $n!/2$ 組ある．任意の二つの順列は，順列内の任意の二つの要素の交換を繰返すことによって相互に転換できるから，この交換に注目する．すなわち任意の交換によって配線の長さが短くなれば，そのように改良する．どのような交換によっても短かくならなければ，これを極小解として残す．交換の種類は $n(n-1)/2$ 個あるから，これを全部試みる手順を決めておく（プログラミングの節参照）．すると一つの基本解から一つの極小解が得られる．実験の結果，この改良法は思った程には有効でなかった．そこで，出発点として n 個の node 全部を試み，n 個の極小解のうち最短のものを現実解として採用することにした．

上の改良法の実際をみると，これは n が少ないとき，あるいは局所的にしか有効でない．大きく離れた点の交換はほとんど意味のないことが多い．node の分布はいくつかのブロックを形成していることが多い．このブロックの並ぶ順序は基本解のままで，上の改良法ではほとんど変更されない．そこでブロックを見分けて，その中に代表点をとり，まず代表点だけについて最短になるように並べ，その後残りの node をつけ加えるという案が考えられる．しかし，手続きがあまり複雑になるので使わなかった．

（b） 筆者等の方法（2）*

$C=n-1$ の場合に用いられた方法を流用することが考えられる．これも便法であって確かに最短に到達する保証はないし，反例が作れる．

これは，まず $n(n-1)/2$ 個のすべての branch の長さを計算し，それを大きさの順に並べておく．方針として，最短の branch から採用していく．$C=2$ になった node については，その node に関係する残りの branch を全部消す．この手続きを続けてすべての node をつなぐ．ただし単純に上の手続きを適用すると，ループを作ることがあるから，branch を採用する前にループの検定を行う必要がある．

（c） 別の方法

逆に進むことも考えられる．最初 $n(n-2)/2$ 個の branch を全部採用しておく．この時すべての node につき $C=n-1$ である．次に長い順から branch を取除いていく．最初に $C=1$ となった node 2個を両端とする．あとは $C=2$ になった node について残った二つの branch を解の branch として保存する．もちろんこれでも反例は作れる．これ以外に，

* このこと自体は NEAC 2201 の製作から始まった．

* これと同じ方法は筆者らと独立に，矢島・高田両氏が KDC-1 によって実験が行われている[4]．

Monte Carlo 法の考え方を取り入れるという着想もある．杉浦氏（電試）によって検討されたが実用的という判断は下せなかった．

4．作　　業

4・1　作業手順

第2図の作業の流れの中で，配線表の自動作製に関連して，実際に行った作業を列挙する．

1．論理設計図から原簿となるカードを作る．このカードはパッケージ1枚あたり1枚ずつのホール・ソート・カードを用いた．記載事項はパッケージの名前（ラックに割つけたときの），番地，入力端子に接続する増幅器の名前，その他である．

2．このカードの記載事項をフレキソライタを用いて紙テープに移す．このテープをDテープと呼ぶ．これ以後の段階では計算機が処理を行う．

3．Dテープから，パッケージの名前と番地だけを抜き出したSテープを作る．

4．Dテープの内容は入力端子についての情報である．次にSテープとDテープから出力端子についての情報を作る．すなわち一つの増幅器に注目して，その名前のある入力端子の番地を探す．実際には10個の増幅器（20個の出力端子）ごとに一度に走査した．この結果得たテープをTテープと呼ぶ．

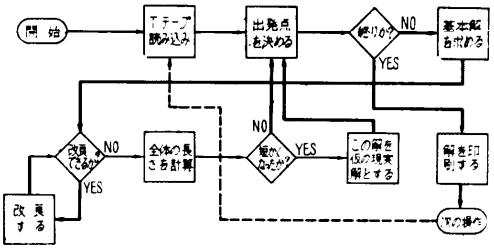
5．Tテープの情報は結線すべき端子の番地であるが，配列は走査によって得られた順序のままなので，ただちに配線指令にはできない．これを配置換えして最短配線（に近いもの）にして，結果を配線指令書として形にととのえて印刷し，同時に紙テープに複製をとり保存しておく．これをRテープと呼ぶ．

4・2　プログラミング

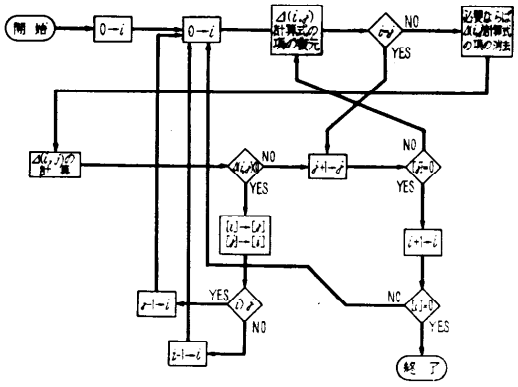
Sテープ，Tテープ作製に必要なプログラムは説明する程のこともない．Rテープの部分も，3・2節に述べた手順をそのまま具体化すればよい．

3・2節（a）の方法によるプログラムの一番粗いフロー・チャートは**第3図**のようになる．基本解の部分は単純である．改良の部分のフロー・チャートは**第4図**，ここで i, j は一列に並んだ node のうち i 番目と j 番目を交換することである．$\Delta(i,j)$ はこの交換による長さの増減である．

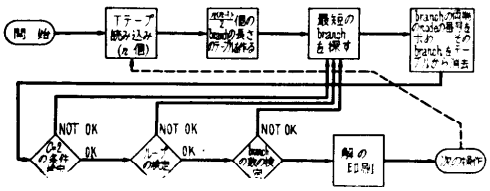
この方法によるルーチンの計算時間は4・3節に示すが，これでは，branch の長さを計算するのに，必要のつど，一々サブルーチンに飛んでいるので遅い．最初すべての branch の長さを計算してテーブルにしておき，あとはそれを利用することにすれば速くなる．3・2節（b）の方法によるルーチン（**第5図**にそのフローチャートを示す）ではテーブルを用いた．



第3図　最短配線のプログラム（1）



第4図　“改良”の部分



第5図　最短配線のプログラム（2）

4・3　作業の実際

4・1で述べた手順の各段階で要した時間，その作業の問題点を列挙する．

1．Dテープの作製

ひととおりのテープができるのに，延30人/時を要した．またテープの検査と修正に3日かかった．テープの長さは延約 200m．このテープはその後の総ての処理の基礎になるから，できる限り完全なことが必要であるが，これは極めて難しい．誤りの原因は作業者にあると同時に，フレキソライタなどが長時間使用すると不安定になったことにもあった．Dテープの誤りは後の処理段階で発見され，修正されたものもあるが，

最後までもち越されたものも少くない．このDテープ作製作業の手順にはあまり工夫をこらさなかったが，これは失敗であった．作業者およびフレキソライタの誤りの可能性を認めた上で，それをできるだけ完全に発見し，修正する手順を構成することは必要である．

2.　Sテープの作製

Sテープの作製には5時間の計算機時間を要した．この作業でDテープのフォーマットの誤りが発見された．その修正の時間は含んでいない．

3.　Tテープの作製

計算機で10時間を要した．2，3の段階での時間はほとんどテープを取扱う入出力機器で費やされた時間で，正味の計算時間はほとんどない．問題なのはテープの世話であって，適当なテープ巻取装置がなかったので上記時間のほとんど，計算機にかかりきりの人手がいった．

4.　Rテープの作製（指令書の印刷）

これが一番問題となる点である．一つの問題を解くのに要する正味の計算機時間とnodeの数の関係は実験的には大体，log－logのグラフで直線になる．実験式は初期のルーチンの場合

$$t \fallingdotseq \frac{1}{4}n^3 \quad [\text{sec}]$$

後に3・2節（b）の方法を用いコーディングしたものでは

$$t \fallingdotseq \frac{1}{150}n^3 \quad [\text{sec}]$$

である．前者では，$n=5$ で30秒，$n=10$ で4分，$n=20$ で35分，$n=30$ で2時間となる．後者ではそれぞれ，0.8秒，6秒，55秒，3分の程度の値となる．原簿カードの数は総計約700枚，そのうち増幅器パッケージ（入力端子数6，出力端子2）470枚，ゲート・パッケージ（入力端子16）230枚程度であり，解くべき問題の数は約900であった．

作業の大部分は遅いルーチンで行なった．この場合 $n \geqq 20$ のものは時間がかかりすぎるので除外した．この除外した部分については，あとから速いルーチンで行った．延の計算機時間はおおよそ50時間であった．

作業手順の段階が進むにつれて，計算機自身の受持つ仕事が多くなり，人手がはぶける．原始的な段階の方に改良すべき点が多い．

5.　プログラミングに要した時間

上記の作業に必要なすべてのルーチンをプログラムするのに要した時間は，1日平均3，4時間の作業で，延2週間程度である．これは手直しも含む．

5.　あとがき—検討

われわれがMark 4Bの配線指令書を作るためにした仕事は大体以上のとおりである．最短配線の問題は数学的には未解決だし，テープの作製，チェック，計算機とのおつき合いなど単純労働にすぎない作業量も予想外にかかって，どの程度の節約ができたか疑問である．現在の段階で問題点を並べてみると，

1.　原理的問題　a.　作業手順
　　b.　最短配線の解法
2.　コストの問題　a.　プログラミング
　　b.　計算機時間
3.　実現性の問題　a.　データ処理装置利用上の便宜
　　b.　結果の活用度

このうち，最短配線の解法は，数学的に厳密な解は得られなかったけれども，筆者等の採用した方法でも大きな誤差はなく実用上さしつかえない．やはり一番大きな問題は，人手だけでやった場合と計算機を使用した場合を比較して，テープの作製時間，計算機時間等の点からみて，引き合うかということであろう．特に莫大なデータを取扱うので，それを計算機にかかる**完全な**形にする手間，主に入出力機器周辺で費やされる時間が大きい．Mk 4A製作のときも，Mk 4を使用して同じようなことを試みようとしたが，結局最初のテープ（Dテープに当る）を作り上げることができず放棄した経験がある．今度の場合，フレキソライタ，光電式リーダ，高速パンチなどによって辛うじてできたという程度である．磁気テープ装置があればかなり容易になると思われる．

また，このような処理は1度限りのものであるという議論がある．しかし，計算機製作に関するデータが完全な形の“ファイル”に構成できると，配線指令書だけでなく，たとえばチェック指令書や，部品手配，機械の改造の際の処理，保守のための作業指令など，1台だけしか計算機を作らない場合でも，種々の利用法が考えられる．またこのファイルの自動処理によって，論理設計検討用データも作製でき，設計作業に役立てる可能性もある*．単なる興味の問題というこ

---

* 配線表作製以外のデータ処理の自動化についてもいくつか試みた例が報告されている[5),6)]．しかもこの自動データ処理を，製作工程の自動化と結びつけることも考えられる（これもアメリカで試み始めたということを聞いている）．日本の計算機工業の水準が高まり，大量生産のそのような段階に達したとき，この種の自動データ処理が現実的な意味を持つことになろう．

にとどまらなければ，この活用の問題に，計算機を使うデータ処理の最大の意義があるだろう．

最後に当所，高橋回路課長の御指導に感謝する．

参考文献

1) 淵，西野：入出力計算機 ETL Mk 4B の方式，情報処理，Vol. 1，(1960) p. 16～22
2) H. Loberman & A. Weinberger: Formal Procedures for Connecting Terminals with a Minimum Total Wire Length, JACM Vol. 4 (1957) p. 428～437
3) 高橋：欧米の電子計算機技術の動向，通信学会電子計算機専門委員会資料 (1960)
4) 矢島，高田：計算機による論理回路布線設計，連大 (1960) No. 380
5) J. P. Malbrain & A. V. Banes: Automated Computer Design, Ramo-Wooldrige (1959)
6) H. L. Engel: Machine Language in Digital Computer Design, Proc. WJCC (1959) p. 182～186
7) K. Kloomok, P. W. Case & H. H. Graff: The Recording, Checking, and Printing of Logic Diagrams, Proc. EJCC (1958) p. 108～118