文　献　紹　介

A: 数　値　解　析　　B: プログラミング　　C: 計　算　機　方　式
D: 回路および機器　　E: オートマトン　　F: 応　用　そ　の　他

A-1. 最適化のシーケンシャルアルゴリズムおよびその応用

V.S. Mikhalevich: Sequential Algorithms of Optimization and Their Applications [Proc. of IFIP Congress 65, pp. 165～172].

種々の分野に現われる最適化問題を繰り返えし計算によって近似的に解く算法が数種紹介されている．まず離散的な最適化問題は，ベルマンの最適性の原理を適用して，次のような手順で解を求めることができる．離散的な時間を t_k ($k=0, 1, \cdots\cdots, N$)，状態変数，制御変数を x_k, u_k とし，x_k, u_k の間には

$$x_k = h_k[x_{k-1}, u_{k-1}]$$

なる関係があるとする．問題は制御クライテリオン

$$g_0(x_0, u_0) + \cdots\cdots + g_{N-1}(x_{N-1}, u_{N-1})$$

を最小にすることである．x_k の定義されている空間に有限個数の点 $x_{k_i}, \cdots\cdots, x_{k_{i+n}}$ からなる集合 A_k を考える．A_k に属する任意の点に，u_{k-1} に課せられた条件 $\tilde{U}_{k-1}(\ni u_{k-1})$ に属するような u_{k-1} で到達できる点 x_{k-1} のうち A_{k-1} に属する集合を $A^u_{k-1}(x_{k_i})$ とする，そこでまず $x_{1_i} \in A_1$ に対して $A^u_0(x_{1_i})$ を見つける．$A^u_0(x_{1_i}) \neq \phi$ であれば

$$\zeta_1(x_{1_i}) = \min g_0(x_{0j}, u_0)$$

なる $\tilde{u}_0(x_{1_i})$ がある．$x_{0j} \in A^u_0(x_1) \neq \phi$ にするような x_1 の集合 ($\subset A_1$) を B^u_1 とする．次に $x_{2_i} \in A_2$ に対して $B^u_1(x_{2_i}) = B^u_1 \cap A^u_1(x_{2_i})$ を見つける．$B^u_1(x_{2_i}) \neq \phi$ ならば

$$\zeta_2(x_{2_i}) = \min [\zeta_1(x_{1_j}) + g_1(x_{1_i}, u_1)]$$

なる $\tilde{u}_1(x_{2_i})$ がある．$B^u_1 \neq \phi$ にするような x_2 の集合 ($\subset A_2$) を B^u_2 とする．以下この手順を $k=N$ まで繰り返せばこのとき得られる列 $\tilde{u}_0, \tilde{u}_1, \cdots\cdots, \tilde{u}_{N-1}$ は最初に選んだ $A_0, A_1, \cdots\cdots, A_N$ の範囲内で最適である．A_k の中の点数を増して行けば，上記の列 $\tilde{u}_k$ は最適制御に近づいて行く．上記の手法は過程が確率的な場合にもそのまま適用できる．たとえばマルコフ過程の場合，関係 h_k はマルコフの遷移関数で置き換えられる．離散的な場合について，著者の開発した方法をも含めてさらに二つの方法が紹介されている．これらの方法は，たとえば道路の最適計画，ガスパイプ系の最適計画，生産ラインの最適配分等に応用すると有効である．

連続系の場合にはポントリヤーギンの最大原理とベルマンの最適性の原理とを併用すると効果が上る．

総体的にこの繰り返えし法をさらに発展させるには，統計的決定理論の成果を導入する必要がある．

（三上　徹）

B-2. 動的な FORMAT 指定

J. E. Ranelletti: Dynamic Format Specifications [Comm. ACM, Vol. 8 August 1965, pp. 508～510]

従来の FORTRAN の FORMAT ステートメントは，コンパイル時にほとんど全部定義され固定されてしまい，実行時に自由に変更できない．この論文では，FORMAT ステートメントを実行時に自由に変更できるようにする方法を述べている．

数値表現形式の変換は FORMAT の中では，E, A, F などで表わされるが，そのくり返えし回数，データフィールドの幅などを表わす数値部に数値の代わりに＊を置く．＊は実行時に，入出力ステートメントのリストの中の最初の変数の値に置き換えられ，それに従って数値表現形式の変換が行なわれる．たとえば

```
      READ INPUT TAPE 2, 4, N, M
    4 FORMAT (2I3)
      READ INPUT TAPE 2, 5, N, (M, (A
      (I, J), J=1, M), I=1, N)
    5 FORMAT (*(*F 15･5/))
```

によって，大きさの異なる入力データのアレイを自由に読み込むことができる．もちろん，DIMENSION で A の大きさは十分大きくとっておく．

また，数値表現形式の変換の仕方そのものを，実行時に変えることもできる．すなわち FORMAT の中の A, E, F の部分に T を置いておき，入出力ステートメントのリストの最初の変数の値でこの T を置

きかえる．たとえば

```
      READ INPUT TAPE 2, 5, N, M
5     FORMAT (A 1, I 3)
      READ INPUT TAPE 2, 10, M, N, (A(I)
      I=1, M)
10    FORMAT (*T 11,3)
```

この例では，最初のカードイメージの値に従がって，以後の変換が行なわれる．

この＊と T の指定を解読するようにコンパイラを作り変えることはあまり難かしくない．著者はこれを Control Data 3600 の IOH パッケージに組込んで成功をおさめた． （佐藤 文孝）

B-3. 機械の論理と制御をシミュレートする言語 LOCS

M. S. Zucker: LOCS: An EDP Machine Logic and Control Simulator 〔IEEE. Trans. EC-14 No. 3. June 1965, pp. 403～418〕

EDP の機械の論理と制御をシミュレートする新らしい言語 LOCS が述べてある．LOCS への入力は LOCS 言語で書かれた機械の記述とテストプログラムである．LOCS の出力は，性能を示す統計，計算結果，テストプログラムおよびシミュレートされた機械に関する診断データである．この論文では LOCS システムの入力，出力操作のあらましを述べ，次に簡単な例での動作をのべている．

LOCS は五つの部分からなっている．

(1) 言語
(2) モニタ
(3) アセンブラ
(4) 翻訳機 (translator)
(5) 実行機 (executor)

実際の機械の機能及び特別な出力は LOCS 言語の operational spec. で書かれる．operational spec. は，シミュレーションされる目的によって，どのような詳細さでも書くことができる．単に機械の命令の集合をシミュレーションしたいならば，必要な記憶装置と命令の制御の順序が知れていればよいので，簡単な LOCS ステートメントで書ける．性能の評価あるいは誤りの診断のためには，細かい論理および制御の記述を行ない，個々のクロックにおいてどのように動作するかを決定することが大切である．

special output は，出力用のテープに書く命令であるが，これは実際の機械のシミュレーションの際に，シミュレータのクロック時間，特定の記憶内容などを特定の時間にテープに出すものである．機械の記憶装置のシミュレーションは実際のコアのビットとシミュレートされる機械のビットを対応させることで行なわれる．

次にこれらの機械の記述は，シミュレーションする IBM 7090 のコードに翻訳される．テストプログラムはシミュレートされる機械の命令語で書かれる．出力は，テストプログラム実行の最初と最後のシミュレータクロック，制御シーケンスのある点でのシミュレータクロック各々の記憶要素への読み書きの回数，名前のつけられた制御シーケンスの行なわれた回数，組合わせ回路の使用回数である．その他デバッグのための出力を出す．本論文ではさらに詳しく．実際の小計算機の例についてのべている．

このようなプログラムは，自動設計プログラムと同時に使用することにより，非常に強力な計算機設計上の道具となると考えられ，なお一層強力なシミュレータの出現がまたれる． （岡田 康行）

B-4. プログラムの自動簡単化について

J. Nievergelt: On the Automatic Simplification of Computer Programs 〔Comm, ACM. Vol. 8, No. 6, June 1965 pp. 366～370〕

ここで考える自動簡単化とは，プログラムの実行内容については何らの予備知識が与えられていないとして，プログラムの命令群の配列だけに注目し，実行内容の同値性を保存した変換を施しながら，より短い，速い，storage の少ないプログラムに改善することである．

著者は，次のような簡単化を上げて，これらの簡単化は，compile 中よりも，compile 完了直後に実行すれば有効であるとしている．

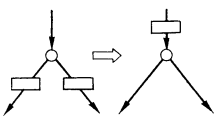
(1) Jump Simplification
  a. 無条件 jump の jump 先に再び無条件 jump 命令がありA番地へ飛ぶとき，最初の jump 先をAに書き換えられる．
  b. 条件つきで jump した jump 先でそれと反対条件つきの jump があれば，前の jump 命令の番地は，あとの jump の次の番地に換えられる．
  c. 無条件 jump を最少にするようにプログラムを配列しなおす．
(2) Chain Simplification

a. 連続した無関係な命令の順序は互いに可換.

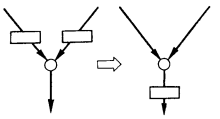
b. 一つの命令で得た情報ないし結果を，それを利用する以前に他の命令群で書き換えてしまうとき，最初の命令は取り除ける.

(3) Node Simplification

同一の命令群は，ある条件で図の様に一つにすることができる.



Fing 3.



Fing 4.

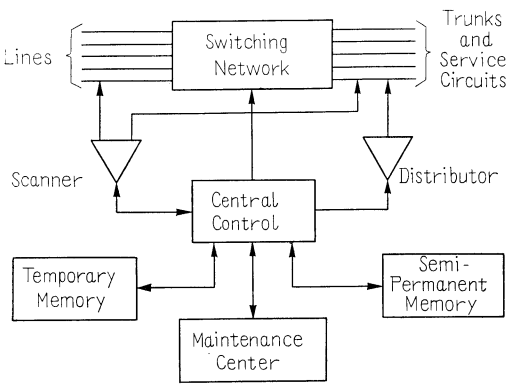
さらに著者は，フローチャートで書かれたプログラムは一種の state diagram であると考えられ，incomplete machine の状態簡約化の手法が応用できることを示唆している.　(村上　国男)

C-5. プログラム記憶方式による電話交換

F.S. Viglinate: Fundamentals of Stored Program Control of Telephone Switching System. [ACM Proc. 19 th National Conference. N 2.2-1～N 2. 2-6]

ベルシステムでは，蓄積プログラム方式による電話交換システムを商用化した．この新方式は，従来の交換方式が，金物に依存した交換システムであったのに比べ，ソフトウェアで交換回路を制御するから，加入者状況の変化に順応しやすく，新サービスの追加なども容易となる点が長所である．下図のブロック図は，そのシステム構成を示す.

走査機は，加入者線からの入力情報およびトランクの空き状態の情報を，高速に捕捉して，それを一時記憶 (Call Store) に転送する．一時記憶で呼がまとめられると，CC (Central Control) は，その情報を使って通話路網を制御する．通話路網は，加入者線とトランクの接続を可能にする.



半固定記憶は，蓄積プログラムと翻訳情報を twistor card に蓄える．twistor card を用いる理由は，その内容が半永久的な性質のものであることと，高速で経済的なメモリであるということである.

モニタプログラムは，システムプログラムがきめられた手順で呼を処理するように監視するほかに，保守プログラムの run も制御する.

CC の命令には，高速を計るための wired macro，誤り発見と診断向きの命令，一般的な命令，の三つの型のものがあり，算術演算よりも論理決定向きの命令が多い.

このシステムでは，特に信頼性と保守には注意が払われていて，40年間に合計数時間のダウンしか許さないように，また処理を誤る確率も 10,000 分の 1 の程度におさえるように設計されている．そのために，メモリ，通話路網，分配装置，走査機は，全て2重化されている.　(石野　福弥)

C-6. 浮動小数点加算方式の解析

D.W. Sweeney: An Analysis of Floating-point Addition. [IBM Sys. J., 1965, pp. 31～42]

この論文は浮動小数点演算方式設計の資料を得るために，いろいろな基数で表現された数値について，浮動小数点加減算のときの，桁合わせのシフト数と，加算後の正規化のシフト数をいろいろなデータについて調べている．データ収集はある計算センタで IBM 704 で計算された実際の問題のうち代表的なもの六つを選んで行なわれた．一つの問題で終りまで行かないうちに 250,000 回の浮動小数点加減算が現われた場合にはそれで trace を中止した．まず第1表に trace された全命令のうち，浮動小数点加減算命令の割合が示されている (平均 11.0 %)．第2表には基数が 2 の場合について，桁合わせシフトと正規化シフト数が 0 から 255 についてその頻度が示されている．さらに演算の結果が 0 になる場合と，結果がオーバーフローする場合も示されている．この結果 27 桁以上シフトをする場合は非常に少なく，ほとんどが 110 から 140 の間にあるが，これは一方の operand が 0 (0×2^{-128}) の場合である．第3表では桁合わせシフトの結果実際に加

算が行なわれる場合と，減算が行なわれる場合について分類してある（これは符号と絶対値表示の場合に必要）．その結果桁合わせ 3 桁以内のものが 55％，正規化シフト 1 桁以内のものが 86％であった．また，この表から再補数化が必要な場合は平均 6.8％しかないことがわかるので，結果の符号はいつも指数の大きい方の operand に一致するようにすれば，ほとんどの場合正しい結果を与え，結果が負になった時だけ再補数化を行なえばよいことがわかる．また第 3 表を使って浮動小数点加減算の平均時間を求める方法が示されている．

次に第 8 表および第 9 表には基数が 2，4，8，10，16，32，64 の場合について，桁合わせシフトと正規化シフトの割合が示されている．これから基数が大きくなるとシフト数は少なくなることがわかる．しかし基数が大きくなると正規化された数値に対して，最初に 0 がつくことがあるので，同じ素子の数に対して有効桁数は少なくなる．シフトの方法にも基数を小さくして，1 桁，2 桁，3 桁……とシフトのパスをいくつもつける方法もあるが，素子の数がふえることになる．これらのことを考慮して IBM System/360 では基数 16 が選ばれた．

以上の資料は倍長演算回路の設計にも役立つと思われる．また倍長演算をプログラムで行なう場合，アキュムレータだけが倍長であれば，非常に能率よくできることが示されている．また桁合わせではみ出した桁のうち 1 桁だけがまた有効桁に戻ることがあり得るので，1 桁だけ保持すればよいので，直列回路の場合，倍長のシフトの必要はない．また並列回路でアキュムレータの下半分について別々に符号をもつようにすれば補数化のための時間と回路が省略できることが示されている．（相馬 嵩）

C-7．大規模記憶装置利用に関する進んだ考え方

C.B. Poland: Advanced Concepts of Utilization of Mass Storage 〔Proc. IFIP Congress 1965, pp. 249～254〕.

コアメモリに納まらないような大量のデータの処理の問題点の通論であり，テープのような純 sequential な装置，ディスクのような direct access の装置および磁気カードのような装置の比較を行ない，これらの装置で構成し得るシステムについて論じている．

テープではなんらかの key を用いて sequential に data を並べるのが唯一の方法であるが，direct access の装置では key を address に用いる方法と index sequential な方法とがあり，また data を任意場所に格納し index をそれに合わせて訂正することも可能である．data 処理では普通 random access-random process が最良と考えられているが，index sequential な方法がより実用的な場合も多く，費用も memory capacity に比例する場合と batch length に比例する場合とある．data 処理に要する時間は速いことが是非必要な場合もあるが，一般的には大部分のデータを long batch で経済的に処理し，かつ，たとえば毎朝指定される特定のデータに関する処理を短時間に行なえる方式が便利なことが多い．

前述の Index Sequential な方式とは，data の key に従って辞書で単語を引く要領で所要の被処理データを見出す方式であり，撰択の最初のレベルで数トラック分のヘッドがまとめて撰択され，次のレベルで特定のトラックが撰択されるようなシステムを考えている．データの長さが不定の場合にはテープが最も取扱いが簡単であるが，direct access の装置では簡単でなく一つの方法として各データを基本長＋α の形とし，α を他のファイルに入れて置く方法がある．この場合，実際上の便宜のためにこの α の分は Index Sequential のところで述べた最初のレベルで撰択される数トラックに入れる方法がとられる．場合によっては各データのうち使用頻度の高い部分と低い部分を別のファイルに格納することもあり，また問合わせを扱う場合には各データに関する処理が起きる度にそれを別のファイルに記録し，個々のデータに関する問合わせに際しては，主ファイルと上記別ファイルの両者を参照して現状を report する方式が経済的である．磁気カードなどでは on-line，off-line 動作を適宜混用して特色ある便宜が得られる．

最後に大切なこととして周辺機器は進歩が早いので，特定の機器の特別の性質を利用した凝ったシステムプログラムを用いないようにとの注意が具体例をあげて述べてある．（佐々木 彬夫）

C-8．多数のデータを順次配列する場合および無作為配列する場合の諸問題

A. I. Dumey: Considerations on Random and Sequential Arrangement of Large Numbers of Records. 〔Proc. IFIP Congress 1965, pp. 255～260〕

データを記憶装置に格納して置く場合に，大きく分

けると読み取りヘッドが固定していて情報がその前を通り過ぎて読み取りを行なうものと，ヘッド，情報の両者が動くものとすることをまず述べ，それぞれの具体例を豊富にあげてある．また交換可能のマガジンによるメモリも重要であるとして例をあげ，テープと容量を比較している．

実際に読み書きに要する時間はデータの位置決定，読み取り，release などより成っており，その制御は全面的に CPU の負担にならないようにメモリの方にも制御がつくが，実際にどのくらい CPU の負担になるかは方式の詳細によるとして，種々の場合をあげている．

今日では複数個の入出力を扱うことが常識となっているが，問題の一つは，データ抜きのプログラムのみで working memory からハミ出すことがしばしばあることである．その場合情報を周辺機器から得ることとなるが，それは electronic speed から mechanical speed へ移ることを意味するので，できる限り速く情報にアクセスすることが大切となる．この問題はデータを sequential に並べることや，また処理し易い形にすることによって解決する場合が多いが，緊急処理を要するデータに問題が残る．これは単なる例外処理または tub file を用いることで解決されるけれども，tub file の場合にはアドレスの割り振りに決め手がない．一般に用いられているのはメモリの一部に backet を設け，その over flow を監視して行く方法である．

デイスクなど多くの大容量記憶装置では，アクセスのための動作が1回行なわれると，その後は読み取りまたは書き込みに要する正味の時間で大量のデータを呼ぶことができるので，データを sequential に並べて置けば実効的に特定データ即時呼び出し機能を有するテープと同じこととなる．最初のアクセスには二十の扉のような方法で index を調べるわけである．Sequential System を用いる場合には．sequence の外へ出ることが少ないとその長所が発揮できない．ramdom system と sequential system とどちらが良いともいえない一つの例は，大規模なメモリを sorting に用いる場合である．一般に両者競合になる場合，そのおもな理由は sequential system の方が data の access に機械的動作が少なくてすむからであろう．しかし特殊な場合で sequeutial system の方が速く出力が得られる場合も存在する．

C-9. FORTRAN ステートメントを直接に処理する小形計算機

A. J. Mebourne and J. M. Pugmire: A Small Computer for the Direct Processing of FORTRAN Statements〔Computer J. Vol. 8, No. 1, April, 1965, pp. 24～27〕

マイクロプログラムされた FORTRAN コンパイラを持った小形計算機について述べてあり，従来の，コンパイラを持った同程度の計算機との比較が行なわれている．

記憶装置は1バイト，16 ビットで構成されていて，語長がオペレータのコンソールから，$2 \leq m \leq 9$ で，m バイトに固定される．

コントロールは固定記憶装置（fixed-store）の中にあるマイクロプログラムによって行なわれる．そのおもな構成は，従来の計算機（X）と共通な基本操作を行なう部分（P_1）と，コンパイラ array の取り扱い，関数の実行，スタック機能などを含む部分（P_2）より成っている．

入力はタイプライタによって行なわれ，FORTRAN ステートメントは変換マイクロプログラムにより二つの段階を経て，命令の記憶場所に最終の形で格納される．Arithmetic statement の場合は独立したサブステートメントとして扱われ，逆ポリッシュ記号に変換される．またデータの記憶場所には array や identifier 等のための場所が確保される．

プログラムの実行は，各ステートメント（たとえば Do ステートメント）の処理を行なう，ステートメント・マイクロプログラムがあり，さらにこれをコントロールするマスタ・マイクロプログラムが二つのスタックを使用することによって行なう．

現在入力中のステートメント，入ってしまったステートメント，実行中のプログラムに対し，容易に修正が行なえるようになっている．

従来の計算機 X と，P_1 と P_2 より成る計算機 P とを比較すると，コストの点では，ほぼ P_2 の分だけ高くなる．実行時間の点では，X のためにハンドコードしたもの，FORTRAN で書いたもの，X でコンパイルされたものでは，P での実行時間がハンドコードしたものより長く，コンパイラを経たものより短くなった．また P によって要求される命令の記憶場所は X で要求されるより少ない．

P_2 のマイクロプログラムの語数を最小にするには，

ソース・プログラムと目的プログラム間の構造が非常に似かよっていることと，ソース・言語と目的言語に独立な（機械設計者によって選ばれた）言語で P_2 自体が記述されることである．

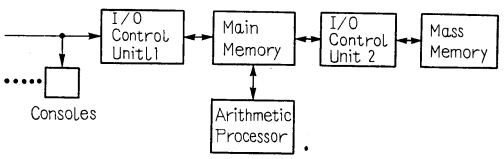
（藤井 狷介）

C-10. Time Sharing System の待合わせ

D. Fife and R. Rosenberg: Queuing in Memory Shared Computer 〔ACM. Proc. of 19 th Conference 1964, pp. H 1-1～H 1-13〕

最近多くの Console から一つの計算機の memory を time sharing で使って計算を行なう，いわゆる time sharing system が脚光をあびている．この論文は time sharing system における待合わせの確率モデルによって，コンソールからの計算命令に対する応答の統計的性質を解析的に求めた結果について述べている．

計算を簡単にするために，次のようなモデルを作成した（第1図参照）．



第1図 典型的な Shared Computer System

(1) メインメモリは五つのブロックに分れ，コンソールからのインプットされた仕事のためのプログラムは，ドラムまたはディスクから読み出されて，必ず一つのブロック内に入る．したがって同時に五つの仕事が実行できる．

(2) インプットのバッファは15とする．すなわち実行中の五つの仕事のほかに15の仕事が計算機に入って待合わせをすることができ，それ以上のインプットに対しては busy となる．

(3) 仕事は到着順に実行し，重要度，処理時間などによる処理の優先順位をつけない．また処理時間に制限を設けない．

(4) 計算機からのアウトプットは，同時にいくつのコンソールへでもできる．

(5) 仕事の到着はポアソン分布に従う．

(6) ドラムまたはディスクからプログラムを読み出す時間およびプログラムの実行時間は指数分布に従う．このようなモデルが処理している仕事の数によってモデルの“状態”を定義し，状態遷移方程式を解くことによって，種々の統計的性質を算出している．

結果は，平均処理時間で正規化した仕事の平均到着間隔，プログラムの平均読出し時間をパラメータとして，平均応答時間，busy 率，user の平均待合わせ時間，メモリの平均使用率などが与えられている．

計算にはこの種の計算を能率的に実行するように作成された RQA (Recursive Queue Analyzer) と称するプログラムを使用した．上に述べたモデルの条件をもっと複雑にして計算することもできる．

（関 栄四郎）

D-11. 計算機設計の半導体に対する影響

S.L.H. Clarke: The Influence of Computer Design on Semiconductors 〔Proc. IFIP Congress 1965, May. pp. 99～103〕

数年前からシリコン集積回路の急速な開発と計算機設計への導入が予言されていたが，現在までのところその進歩の速度は予想されたほどではない．たとえばシリコンの価格がトランジスタよりも下らないこと，信頼度推定が確かでないこと，速度の向上があまり著しくはないこと，容量 limitation の問題などがあり，典型的な例は IBM 360 が完全な集積回路でなく混合システムになったことである．ここではこの「進歩のおそさ」の原因として2点をあげている．

第1には実際のシステムで集積回路を使用することの困難性であり，アセンブリの方法，熱放散などが問題となるが，現実には，従来のパッケージなどを使用しても何とか解決できる．

第2に半導体メーカに flexibility がないことが問題であり，これは見かけよりも重要である．従来は装置メーカの要求に従って部品メーカが製品を作っていたが，集積回路では事情が異なってくる．1枚に載せられるゲートの数が増大するに従って，一般に使用できるブロックを決定することが非常に困難となり，半導体メーカが従来のように装置メーカの要求によって製作を始めていたのでは経済的に合わなくなる．

ここではこの第2の問題について根本的な解答を示してはいないが，一つの方法として集積回路を搭載した submodule の使用について述べている．さらに具体例としてトンネルダイオード Goto pair を用いた 50 Mc クロックの実験用計算機の試作について記している．現在はこれを 250 Mc にまで高速化する研

究が進行中で，相互接続の方法として，4相系で半波長 delay cable を使用することなどが検討されている．（苗村　憲司）

D-12．集積磁気記憶装置と集積超伝導記憶装置，その技術と成果並びに見通しの展望

J.A. Rajchman : Integrated Magnetic and Superconductive Memories A Survey of Techniques, Results and Prospects. 〔Proc. IFIP. Congress 1965, pp. 123～130〕.

2×10^8 ビット 8 μs および数百万ビット 1 μs 以下のコアメモリが今年または来年に市場に出ると思われるが，core stack そのものの値段と周辺の回路の値段は100万ビット 1 μs のメモリでほぼ同じであり，大容量になるにつれて core stack そのものの価格の比率は大きくなる．製造の便宜から考えて集積化の方向へ進むのが当然であるが，現在のところ集積化してないメモリを圧倒するようなものはまだ現われていない．集積化により10倍程度の性能の向上が可能であるが，その線に沿う筆者の研究二つを述べる．

集積化メモリでは素子の特性を揃えるのがむずかしく，電流一致方式は困難であり，語配列を用いるのであるが，選択ダイオードと磁気素子両者ともに集積化して初めて本当に経済的なメモリが得られる．筆者の研究の一つは laminated ferrite であり，各ビットは実効3 mil の磁心に相当し，ストリップ状の“巻線”をつけた2枚のフェライトシートの中間層を介して“巻線”が直交するように一体としたもので，64×64 ビットのものが上首尾に得られており，数百万ビッド 1 μs 以下のメモリの登場も間近かである．

筆者の研究の他の一つは超伝導薄膜メモリである．これは1平方インチ1万ビットの実装密度で，16,384 ビットのものが試作済みであり，動作原理の全てが実証されており，原理的にも実験的にも大容量の記憶装置を作る見込みに反する事実はどこからも出ていない．試作したものの構造は，錫の薄膜に碁盤の目に直交する巻線を付し，さらに裏側にジグザグのセンス巻線を付したもので直交する巻線の交叉点が各ビットを構成するものであるが，むずかしい点は各ビットの特性を所要の範囲に揃えることである．アドレスの選択はクライオトロンの tree で行なっている．

集積回路の発達により磁性体が完全に駆逐されるかどうかは興味あるところで，最近 MOS のみによる集積化メモリも報告されており complementary symmetry flip flop を使う方法が最も興味がある．この方法は小容量の高速メモリに向くものであり，大容量のものには薄膜トランジスタ TFT が適当で，TFT はまず content addressible メモリに使用されるであろう．集積化回路は今後も非常な勢いで発展し，メモリにおける非集積回路の優勢はいつかは破られるか，少なくも大きな影響を受けるであろう．

（佐々木　彬夫）

D-13．磁性体記憶装置の将来

G. Kohn : Future of Magnetic Memories 〔Proc. IFIP Congress 1965, pp. 131～136〕.

記算機の主記憶装置は現在磁性体を記憶素子とするものが最も多く用いられているが，この傾向は近い将来も続くであろうとして，その問題点および有望と考えられる形式をハードウェアの立場から概説したものである．磁性体を使用する記憶装置でも，磁気ドラム，磁気ディクスなどの補助記憶装置には言及されていない．取り上げられているのは主としてフェライト磁心と磁性薄膜であって，これを記憶容量 10^7 ビット以上の大容量記憶装置と，10^5 ビット程度以下の高速記憶装置の二つに分けて記述している．

大容量記憶装置については現在標準的な技術というものはないが，個々の記憶素子を試験し，構成する方法が限界に近づいていることは明らかで，何らかの形でバッチ構成をする必要があろう．フェライトを用いるものでは多孔板および棒状構造の二つがあげられる．磁性薄膜はもともとバッチ構成的のもので長所をもっているが，フェライトと薄膜のどちらが最後に勝つかは予測を許さない．大容量の素子をバッチ構成する時の問題は素子の欠陥の処置である．これにはどのように冗長度を導入するのかが重要である．つぎに性質の全く異るものとして光学的に選択される磁性体素子は記憶密度の大きい点で有望であるが，まだ材料の問題が多く残されている．

高速記憶装置に関しては磁性薄膜が有望であろう．サイクル時間 200 ns～1 μs の範囲ではフェライト磁心もまた有力である．薄膜についてはそのサイクル時間を制限する要因を分析した結果より，非破壊読出し方式がすぐれていることを指摘している．また磁性薄膜で非破壊読出し方式を具体化する一つの構成方法を述べている．それについて実施した基礎的な実験結果によれば，読出しサイクル時間 15 ns，書込みサイクル時

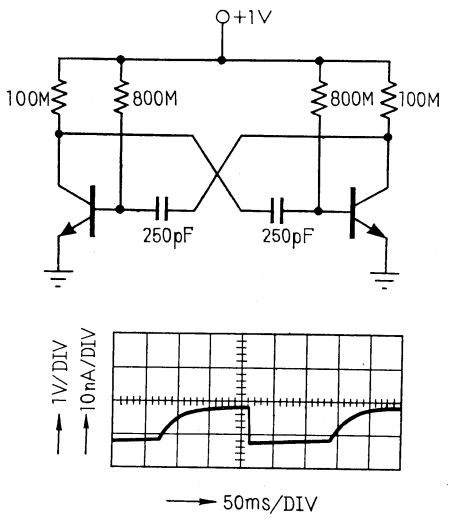
間 65 ns が得られており，将来この程度の速度で記憶容量 10^5 ビットくらいのものはできるだろう考えられる．　（石井　治）

D-14. ナノワット装置

W.W. Gaertner: Nanowatt Devices〔Proc. of IEEE. Vol. 53, No. 6, June, 1965, pp. 592～604〕.

本論文は豊富な引用文献（53 種）の調査から，これまでのミリワット回路についての検討を一歩進め，消費電力がナノワットの電子回路の可能性について考察し，実験例をあげつつ現在の技術レベルについて述べたものである．

トランジスタが 1 nW クラスで動作することを理論的に証明したのち，エミッタ電流（I_E）が 1 nA で h_{FE}=30，10 nA で h_{FE}=50 のトランジスタ実測例，19 nW で動作するマルチバイブレータ（第 1 図）の回路例を示している．



19ナノワットの電源で動作するアステーブル・マルチの回路とその波形

ナノアンペア回路で使用する抵抗は 100 MΩ 以上となるので，動作速度を制限するものは，各種のキャパシタンスである．ジャンクション容量の技術的限界が 10^{-15} ファラッドと思われるので，動作速度の上限は 1 Mc 程度であろう．容量を小さくするためには (1) すべての回路素子の断面積はそこを流れる電流値に比例して小さくすること，(2) 長さはできるだけ短くすること，が必要である．回路の浮遊容量を小さくするためには，基材としての Si は必ずしも適格ではなく，oxide isolation 等の新しい手段が必要であろう．

トランジスタはコレクタの大きさが 3×3 μ のもの，抵抗は幅 1 μ，長さ 100 μ，シート抵抗 100 μΩ/square のものが目標とされているが，相当困難であろう．ダイオートとしては動作レベルが低いことから，バックワードダイオード，トンネルダイオードが有望であろう．

なお，これらの超小形回路を製作する方法として，従来のマスクによるパターン作成方式に代わって，計算機で制御されたエレクトロンまたはイオンビームによる adaptive fabrication が注目されている．

抵抗体の雑音レベルは，動作に支障をもたらす程度ではない．　（鍵山　圭一郎）

E-15. 図表による多段論理合成法

S. B. Akers: A Diagramatic Approach to Multilevel Logic Synthesis〔IEEE. Trans. EC.-14 No. 2, April, 1965, pp. 174～181〕

Karnaugh 図表は論理関数の簡単化における視覚的補助手段として有用であるが，多段合成の場合にはあまり実用的でない．Akers がここで提案している図表はその点で Karnaugh 図表にまさるものといえる．

任意の論理関数 $f(x_1, ..., x_n)$ は，m 個（$m \leqq 2n$）のリテラル（$x_r, \bar{x}_s$ など）の単調関数として $F(X_1, ..., X_m)$ の形で表わされる（Akers は logically passive と呼んでいる）．F の値 1 を与える入力ベクトルの集合を $\alpha=\{\alpha_i\}$，値 0 を与える入力ベクトルの集合を $\beta=\{\beta_j\}$ とすれば（禁止ベクトルはいずれにも含まれない），任意の α_i と β_j は $\alpha_i \neq \beta_j$ を満たす．すなわち i, j の任意の値について，α_i において 1，β_j において 0 をとるリテラル X_{ij} が少なくとも一つは存在する．Akers の“α-β 図表”は，本質的にはこの X_{ij}（1 個または複数個）を i, j 要素とする行列である．

これに次の手順を適宜（任意の順序で）くり返えし適用して，図を簡単化しながら AND, OR ゲートの組合わせで合成していく．

(1) ある行(列)の要素がみな，他の行(列)にも含まれていれば，後者の行(列)は消去することができる．

(2) 各正方形から一つを残して他のリテラルをすべて消去することができる．

(3) 行と列は適当に順序を変えることができる．

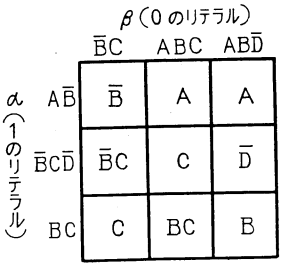
(4) 隣接した正(長)方形の内容が同じならばまとめて一つの正(長)方形とすることができる．

(5) 図をよこ方向に切って p 個の長方形に分割することができる．このとき fan-in 数 p の ORゲートを1個使用してその出力を関数出力とする．p 個の入力が，分割で得られた新しい図表に対応する．

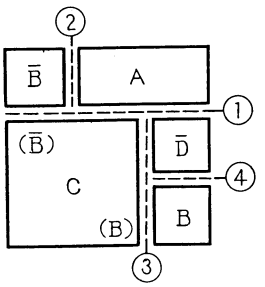
(6) 図をたてに切ることもできる．ANDゲートを使うこと以外は（5）に同じ．

以上の手順により，第1図の図表は第2図のように簡単化できて第3図のように合成される．厳密なアルゴリズムが示されてないので最小の合成法を得ることは保障されないが，fan-in，段数などを図表上で任意に選べる点で実用的な approach であるといえよう．

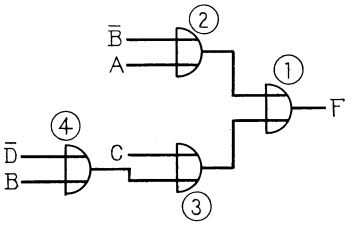
（苗村　憲司）



第1図　F の α-β 図表



第2図　簡単化と分割



第3図　合　　成

E-16. 線型順序回路に対する正規表現

J.A. Brozozowski: Regular Expressions for Linear Sequential Circuits〔IEEE. Trans. EC-14, No. 2, April, 1965, pp. 148～156〕.

1入力1出力の有限オートマトンの動作は正規表現によって完全に記述でき，状態図と正規表現は直接変換可能であって，対応する順序回路を直接構成することもできる．しかし，実現された回路は，外部入力とは別にスタートパルスをも必要とする．それ故，初期状態だけが指定されているような順序回路から，これを記述する正規表現を直接求めることはできない．

本文は，線型順序回路に限定し，回路構成や遅れ演算子による伝達関数と密接な関係を持つ，いくつかの“regular language”を導入して，この問題に考察を加えている．

いわゆる正規演算子のほかに AND (&)，ex. OR (+)，否定などを使用している．入力 R が一つの遅れ要素を通過した時の出力を Ri と表現できる．受理可能な入力系列，1で終る0，1の系列の全ての集合を $u=I_1$ とし，linor (linear integral operator) $A\equiv(a_0, ia_1, i^2a_2,\ldots, i^na_n)$，$a\epsilon_j\{\phi, \lambda\}$ を正規集合 R に対し，

$$RA\equiv Ra_1+Ria_1+Ri^2a_2+\cdots\cdots+Ri^na_n$$

なる性質を持つものと定義すると，feedback loop を持たない回路の正規表現は $R=uA$ と書ける．この場合は，回路から直ちに正規表現が得られる．

feedback loop のある時は，loop の所のみに着目して S 個の遅延の時には，$Ri^s+R=u$ なる方程式を得る．いま S 次の帰還演算子 i^s（ただし $i_0=\lambda$）を導入すると，この解は，$R=ui^s$ と書ける．

ここで，新しい関数を次のように導入する．

$$f_0(X)=ui=(0\vee 10^*1)^*10^*$$
$$f_0(iX)=(i0\vee i\ 1\ (10)^*i\ 1)^*i\ 1\ (i0)^*$$
$$\vdots$$
$$f_e=\overline{f_0(X)}$$
$$\vdots$$
$$J_k=\lambda+i+i^2+\cdots\cdots+i^k$$

上式を使用すると，次式が導ける．

$$ui^n=J_{n-2}\{0\,f_0(i^{n-1}X)+1\,f_e(i^{n-1}X)\}+f_0(i^{n-1}X)$$

遅れ演算子による伝達関数で回路を表現すれば，一般に

$$F(D)=\frac{D^{P-r}(1+a_1D+\cdots\cdots+a_mD^m)}{1+b_1D+b_2D^2+\cdots\cdots+b_nD^n}=\frac{D^{P-r}G(D)}{1+D^s}$$

と書け，分母は feedback，分子は前向き要素を表現しているから，本文の議論で，線型順序回路は直接，正規表現に変換できる．　（村上　国男）

E-17. ディジタル回路の冗長法

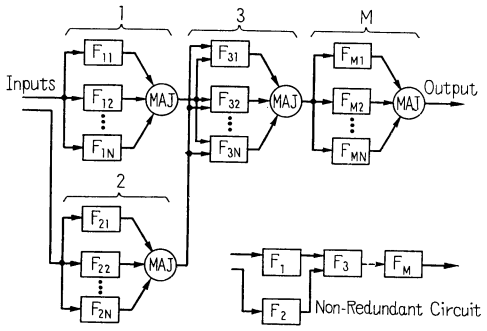
Rein Teoste: Digital Circuit Redundancy〔IEEE. Trans. on Reliability, Vol. 13, No. 2 June, 1964, pp. 42～60〕

いくつかの冗長法の例をあげ数式モデルを使って信

頼度の向上を推定し，各方法の比較をする．Majority Redundancy, Modified Majority Redundancy, Moore-Shannon Redundancy, Gate Connector Redundancy, Multiple-Line Circuit, Switching および Time Redundancy などをとりあげている．

冗長法の採用により，必要部品数・価格が増すだけでなく，それに伴って付加コスト（重量，大きさ，消費電力など）が増加するので，冗長法採用による信頼度の向上と付加コストの増加を考えて，全コスト最小を求めねばならない．

Majority Redundancy 装置を同じ信頼度 r を持



第1図 Majority redundancy applied on the Mth level.

った M 個の独立した部分に分け，各部分に冗長法を適用するもので（第1図），各部分は N 個のユニットとそのユニットの出力を入力とし，その入力の多数決論理により出力を生じる Majority organ からなる．その全体のシステム信頼度は

$$R_R=(1-\varepsilon)^M\Big[\sum_{n=\lceil N+1/2\rceil}^{N}\binom{N}{n}R_0^{n/M}(1-R_0^{1/M})^{N-n}\Big]^M$$

ε: Majority Organ の故障率

Majority organ の出力と各ユニットの出力を比較してユニットの故障を検出し，それを取り除く手法として M.M.R. がある．その信頼度は

$$R_R(1-\varepsilon)^M[1-N(1-R_0^{1/M})^{N-1}+(N-1)(1-R_0^{1/M})^N]^M$$

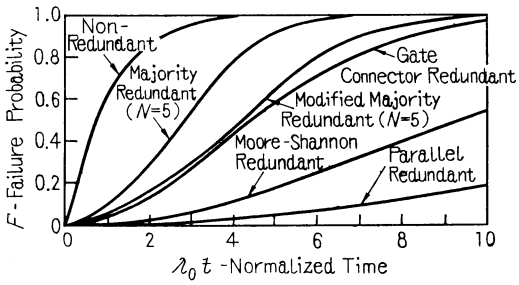
Moore-Shannon は1個のリレーのように働らく直並列に接続したリレー回路を提案した．リレーの故障として open type の故障と close type の故障とを想定して，その信頼度を求めた．

$R_R=\lceil 2R_0^{2/K}-R_0^{4/K}\rceil^K$ K: switch-like 回路数

冗長法は全ての故障に対して有効ではなく，ランダム故障に対してのみ有効である．装置の有用性を評価するに適した基準は MTTF である．故障確率が時間の経過と共に変化することを知ることは重要である．

第2図は Parallel Redundant が最も高信頼度を得られることを示しているが，これは open type の故障に対してのみ有効で一般的には Moore-Shannon Redundant が一番よいといえる．

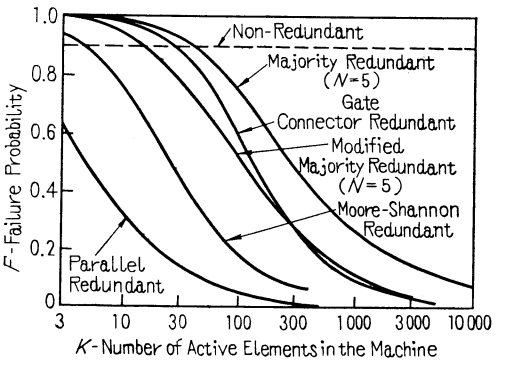
冗長法を選ぶ時には Reliability Gain と冗長部品の数の間の調整が必要であり，第1表は Parallel Redundant が一番よいこと，第3図は装置の使用部品数が多いほど冗長法による信頼度向上の効果が上ることを示している．（安部 匡晃）



第2図 Comparison of redundancy methods.

第1表

Type of Redundancy	Mean Time to Failure	Amount of Redundancy	Gain per Component
Majority	$3/\lambda_0$	approx. 5	0.6
Modified Majority	$4/\lambda_0$	approx. 5	0.8
Moore-Shannon	$10/\lambda_0$	4	2.5
Parallel	$18/\lambda_0$	2	9.0
Gate Connector	$4.5/\lambda_0$	approx. 6	0.75
Nonredundant	$1/\lambda_0$	1	1.0



第3図 Comparison of redundancy methods for $F_0=0.9$

F-18. 製鋼業に使われる制御用ディジタル計算機

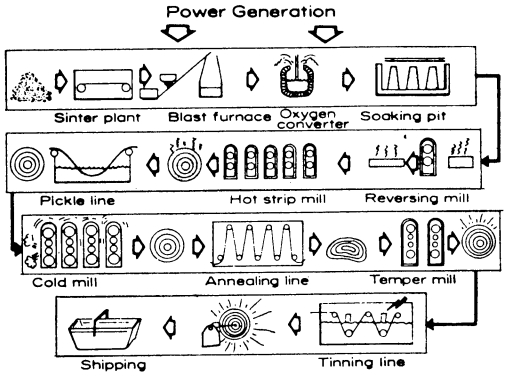
A.S. Brower : Digital Control Computers for the Metals Industry〔ISA J. Vol. 12, No. 2, February, 1965, pp. 51～63〕

一般に計算機で制御されるプロセスが備えていなければならない性質としては,

（1） 連続的で，製造能力が大きい
（2） 機械化されていて，人間の操作が不要
（3） 変動値に対して，速くて正確な調節ができる
（4） 生産能力に影響を与える変数が数多くある
（5） 科学的に根拠ある原理に基づいて動作している
（6） 改良が可能

などがあげられる.

最近の製鋼用プラントでは，焼結，高炉，酸素転炉，ホット・ミル，酸洗い，コールド・ミル，焼きなまし，鍛造ミル，錫メッキなど大体13の処理過程がある(第1図参照). このうち，洗じょう過程は比較的



第1図

単純なものであることから，また鍛造ミルは機械の大がかりな変更を必要とすることから，それぞれ計算機制御の対象から除かれる（本論文では全プラントにわたる制御ではなしに，各プロセスごとに個別の計算機制御を行なうことを考えている). 他方，ホット・ミルおよびコールド・ミルは経済的にも計算機制御を適応させやすいプロセスである.

なお，それぞれのプロセスのために開発される制御用計算機は，そのプラントのそのプロセスだけのためのものであるから，システムのプラニングに当たってはユーザ側において，そのプロセスに関する種々の角からの調査検討が前もって必要であり，またメーカが決定してからは仕様の詳細にわたって，両者の密接な協調がなされなければならない.

本論文では上記のほか，一般の制御用計算機の特長，信頼性，設置条件，プログラムの問題などについても触れている.　　（渡辺　豊丸）

F-19. 大学の計算機教育講座

Sam. D. Conte : The Computer Science Program at Purdue Unviersity〔ACM, Proc. 19 th National Conference, L 1. 2-1〕

Purdue 大学（米国インディアナ州立）では学内・学外における計算機に対する関心を考慮して 1962 年の秋学期から計算機科学に関する講座（The Computer Sciences Program）を開催することになった. この講座は数理科学科によって管理され，計算機に関する理論および応用の教授を目的とする. この講座に用いられる計算センターには10名の専任者および8名の数理科学科の教授が参加しており，他の分野の研究者も必要に応じて協同研究をすることになっている.

またこの講座は単にプログラミングの指導，数値計算を扱うのみならず，学部および大学院の学生に対して学士および修士の学位を授けている. 現在までこれらの学位をうけた学生の内訳は数学で39名一般工学の分野で4名，その他心理学，商学でそれぞれ1名となっている.　　（成田　誠之助）