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

B-22. 組織設計のためのドキュメンテーション

P.H. Vince: Documentation for System Design 〔Computer Journal Vol. 7, No. 1, April, 1964 pp. 8～14〕

これは筆者の英国 IBM での経験をもとにして，組織設計をする際のドキュメンテーションの一つの方式を示したものである．組織設計には，組織の分析，組織要請の決定，および設計，評価の各相がある．仕事はまず，機械の先入観なしに客観的に始められなければならない．

アナリストの仕事には，組織モデルの設定と，その事業の目標の精確な定義（これには経営者との一致が必要）がある．モデルは各対象について異なり得るが，その一般的なスキームとして，入力，出力，操作，手持能力という概念分けによるものを示す．目標については，単にもうけるというような抽象的なものでなく，具体的なものでなければならない．精密工業を例にとれば，標準品の生産，注文生産，役務およびコンサルテーションの提供など8項目があげられる（計算機業そのものが，各種の要求を含んだよいモデルである，という指摘は面白い）．

分析において必要な（有用な）いくつかの書式を，アナリストの仕事と関連させて提示すること，がこの論文の主眼である．組織の活動の構造を示すものとして，組織図，活動図がある．これらと諸原資の利用をマトリックス状に表わした，原資利用表（Resource usage sheet）が作られる．

また，活動図の各ステップを詳しくしたものとして，作業情報行列（Functional information matrix）がある．これは，各作業について，入力と出力との相互作用をグラフで表わしたものである．その各要素は，たとえば，C（計算），I（指示），T（転写），V（更新）のような記号で表わされる．さらに，ファイル一覧表（File sheet），メッセージ一覧表（Message sheet）も作られる．以上のような書式を実例を挙げて，分析の過程とともに説明する．

ここに述べた，一般的な形から特殊に進む行き方のほかに，現在の出力のすべてを調べ上げて，そのどれが目標達成に必要かを決めていく行き方もある．いずれにしても，諸活動を評価するために，ドキュメンテーションの総括が必要である．このために階段状のマトリックスを持った活動図（Activity chart）がつくられる．

以上のような書類（Document）を作製し，その中で原価などが計算され，組織設計の検討がなされる．その結果によっては，再設計ということで，上のような手順が繰り返えされることによって，組織設計が進んでいくのである．　（淵　一博）

B-23. FORTRAN から ALGOL へのほんやくルーチン

D. Pullin: A FORTRAN to ALGOL Translator 〔Computer Journal Vol. 7, No. 1, April 1964 pp. 24～27〕

この論文は ALGOL と FORTRAN の長所に関する論争を解決する一方法，すなわち FORTRAN source language（FORTRAN II または IV）から ALGOL source language を得るほんやくルーチンについて述べたものである．

基本的な考え方は FORTRAN プログラムは大部分 function または subroutine として指定されるサブプログラムの集まりとして記述できるから，これらを一つの単位と考え，そのおのおのを読み込み，それらと equivalent な ALGOL statement に変換することである．これを具体的に FORTRAN の個々の statement がどのように ALGOL の statement にほんやくされるかということが例示されている．

たとえば FORTRAN IF についていえば，

```
IF (A(I)−B) 10, 20, 30
```

は

```
if A[I]−B<0 then go to l10;
if A[I]−B=0 then go to l20 else go to
   l30;
```

と与えられる．

その他 DIMENSION, COMMON および EQUIVALENCE の各 statement の処理についても説明されている．特に FORTRAN の input/output state-

ment については Elliott ALGOL に認容できる型に変換する新しい手順 setformat (a, b) と forout (y, z) が与えられる．これらは input/output の装置の撰択，データの伝送，および input, output の数値の順序と個数を決めるリストの作成などに使われる．またリストの中に DO ループがあれば，それは for statement に置換される．このシステムを異なった ALGOL compiler に働かせる場合は，上に述べた setformat と forout の procedure を書き直すだけでよい．

このほんやくルーチンは ALGOL で書かれており将来 FORTRAN が修正された場合容易に組み込むことができる上に，すでに ALGOL compiler があれば数多くの FORTRAN ライブラリーを利用することができる．

最後に FORTRAN subroutine のプログラムとこのほんやくルーチンによって得られた ALGOL procedure のプログラムが与えられている．（長谷文子）

B-24．句構造文法の決定問題

Peter S. Landweber: Decision Problem of Phrase-Structure Grammars 〔IEEE Trans. on Electronic Computers EC-13, Aug., 1964, pp. 354～362〕

Chomsky によって初めて句構造文法が言語の文を生成する装置であることが示された．文法はその規則に制限を加えることによって，0, 1, 2, 3 型に分類される．この論文では，各型の文法を対象とした決定問題が扱われている．決定問題の証明については，この論文以前に，すでに3型文法の場合は Rabin and Scott らによって，2型文法の場合は Bar-Hillel らによって証明されている．ここではそれらの証明を独自の方法（実は審査員によって証明の一部が誤りであることが指摘されている）によって行ない，さらに1型についてもその方法を拡張して行なう．それらの結果をまとめた一覧表が与えられている．

各型文法に対する決定問題

問　　題	文法の型			
	3型	2型	1型	0型
任意の文法で生成された言語は空であるか？ ($L_G=0$)	肯	肯	否	否
任意の文法で生成された言語は無限か？ (L_G=無限)	肯	肯	否	否
任意の文法で生成された言語は最終の文字のつながりに等しいか？ ($L_G=V_T^*$)	肯	否	否	否
二つの文法は同じ言語を生成するか？ ($L_{G_1}=L_{G_2}$)	肯	否	否	否
二つの文法の言語の一つが他に含まれるか？ ($L_{G_1}\leq L_{G_2}$)	肯	否	否	否
二つの文法に共通な言語は空であるか？ ($L_{G_1}\cap L_{G_2}=0$)	肯	否	否	否
二つの文法に共通なる言語は同じ型か？	真	否	真	真
ある文法により生成された言語の補集合は同じ型の言語か？	真	否	？	否
任意のつながり ϕ, ψ に対して，ある文法で ϕ が ψ から導かれるか？ ($\phi\Rightarrow\psi$)	肯	肯	肯	否
位意のつながり ϕ, ψ に対して，ある文法で ϕ を含むあるつながりが ψ から導かれるか？ ($\phi^*\Rightarrow\psi$)	肯	肯	否	否
ある文法の言語の文が二とおりの方法で導かれるか？（G はあいまいである）	肯	否	否	否
ある文法に対して，同じ型の文法で，同じ言語を生成するあいまいでない文法があるか？	真	？	？	真

肯：この問題を決定する算法が存在する
否：この問題を決定する算法が存在しない
真：この問題はいつでも成り立つ
？：未解決

（五十嵐実子）

B-25．新しいプログラミング言語

Daniel D. McCracken: The New Programming Language 〔Datamation, Vol. 10, No. 7, July, 1964, pp. 31～36〕.

IBM の SHARE から NPL (New Programming Language) と称するコムパイラ言語が発表されたが，これは SHARE で6名からなる委員会で作成されたもので，1963年10月から1964年5月の間に会合をかさね，最終的なレポートを提出するに至ったものである．

NPL は，ALGOL の方向にそうて FORTRAN を展開したような外観を有し，ユーザの見地からみると，FORTRAN より簡単でより有力であるとして，NPL の特長を要約して 22 カ条で示した説明書をあげている．この説明は FORTRAN を知っていることを前提としている．その主な事項をあげる．

（1） ステートメントはセミコロンで区切られ，continuation カードの考えはない．プログラムは入力媒体に関係なく字の直線的なつながりから成るものとされている．遠隔伝送を可能にしている．

（2） 適当な Declaration を用いると，変数は complex あるいは logical な字またはビットとして定義され，字とビット単位の操作が可能になっている．

（3） 複合ステートメントがあり，DO と END でかこまれたステートメントのグループは一つのステー

トメントとして制御される．

（4） IF ステートメントに ELSE path が含まれ，論理演算を便にしている．

（5） DO ステートメントは発展して，インデックス変数の制限がなくなり，論理演算のため WHILE が付加されている．

（6） 入出力に標準的なフオーマットを使用すれば，FORMAT の情報は省略できる．

（7） Array については，procedure で実際のサイズをきめられ，ダイナミックなメモリー割付が可能である．

（8） Procedure は他の procedure を，いかなるレベルまでも含め得る．procedure は多数の entry と return の点を持てる．

（9） COBOL と同様なデータの取扱いが可能であり，procedure の編集によって完全に制御できる．

（10） Multi-processing と Concurrent operation に対するプログラムが書ける．プログラムは計算の一部が完了するまで suspend でき，大規模なリアル・タイム・システムにおける利用が考えられている．

現在では，NPL は IBM 360 に用意することになっているが，それ以上の発展は，産業界の NPL 使用の効果の判断，ユーザの熱意などに左右される．著者の意見では，NPL は，まず IBM のユーザの科学計算に採用され，ついで事務計算と IBM 以外のユーザに徐々に普及していくだろうとしている．

なお，NPL で書いたプログラムが 4 例示されている．（河辺教雄）

B-26. 機械の構造・動作を記述する形式言語

H.P. Schlaeppi: A Formal Language for Describing Machine Logic, Timing and Sequencing (LOTIS) 〔IEEE Trans. on Electronic Computers, EC-13, No. 4, Aug. 1964, pp. 439～448〕

ディジタル計算機の構造や動作を記述する言葉を，人間の見やすい形に作っておけば，設計者にとっても設計の助けに計算機を使う場合にも非常に便利である．このような考えから hardware を記述する言葉を作ったのであるが，完全な文法は後日発表されるものとして，でき上ったものを見ると ALGOL 60 によく似ている．

まず，物と名前を一対一に対応させる宣言部と，動作の手順を表わす過程部に分かれている．後者の本質的な側面として data flow と control をあげ，これらの記述を明確，簡素に工夫して，機械の hierarchy や concurrency をこなしているのがこの論文の核心である．

data flow を素過程 step に分解してみると，ある path を働かせることにほかならないから，一つの assignment statement に対応される．この“働かせる”ということをこまかに考察して，タイミングを数えられる場合（同期式）と，step の終りと次の step の始まりが同一な場合（非同期式）などを区別している．

回路のスイッチに operator を対応させ，回路網に expression を対応させる点は，大よそ ALGOL 60 の真似をしている．途中に待ち合わせを含まない step の一連を sequence と呼び，独立に実行されないものをひとまとめにして group と呼んでいる．すると，control とは沢山の sequences を順序づけるものになる．branch と call の機能をもたせ，あらわに宣言されていない間接的なシーケンスをも呼び出せるよう工夫している．

ひとつの group にひとつの表示灯を対応させ，実行中かどうかを条件文で調べることができる．このことと，ある group を black box と見たてて function と呼ぶことから concurrency や hierarchy の記述が簡便になっている．最後に非同期式パラレル加算器の命令による実行を記述した例が示されていることを付記しておく．（川合英俊）

B-27. FORMAC 入門

J.E. Sammet and E.R. Bond: Introduction to FORMAC 〔IEEE Trans. on Electronic Computers EC-13, No. 4, Aug. 1964, pp. 386～394〕

FORMAC とは FORmula MAnipulation Compiler によるもので，非数値計算にも利用できるように開発された実験的なプログラミングシステムである．

方程式 $Y=X^2+2.5\frac{Z}{X}$ があるとき，FORTRAN では変数には数値のみを指定できるから X に 5，Z に 4 を指定すると結果は 27 となる．これを行なうプログラムは

$X=5$,　　$Z=4$,　　$Y=X**2+Z*2.5/X$

であって，実行段階では 27 が Y で表わされる単一の記憶部分に格納される．一方 FORMAC においては変数には数値または記号を指定できる．たとえば X に

表現（$A+B$）を，Zに数値4を指定すれば結果は

$$(A+B)^2+\frac{10}{A+B}$$

である．これを FORMAC プログラムで表わせば

```
LET X=A+B
    Z=4
LET Y=X**2+Z*2.5/X
```

となり実行の結果は（$A+B$)**2+10/($A+B$）という表現がYという記憶部分に格納される．

FORTRAN における三角関数，対数に対応する operators はもちろん，微分，組合せ，階乗，一つおきの階乗の四つの operators が FORMAC には追加されている．

FORMAC の命令と宣言を次に示す．

1) FORMAC 変数を求める命令

LET	指定された表現を構成せよ
SUBST	表現をともなった変数を代入せよ
EXPAND	かっこをはずせ
COEFF	変数またはその“べき”の係数を求めよ
PART	表現を項，因子，指数に分離せよ

2) FORTRAN 変数を得る命令

EVAL	数値化せよ
MATCH	二つの表現が同値か恒等か比較せよ
FIND	従属関係を決定せよ
CENSUS	語，項，因子を数えよ

3) 雑命令

BCDCON	内部形式から BCD 形式に変えよ
ALGCON	BCD 形式から内部形式に変えよ
AUTSIM	自動簡単化中に成される計算を制御
ERASE	不必要な式を消去せよ

4) FORMAC DECLARATIONS

ATOMIC	basic な変数の宣言
DEPEND	明白な従属関係の宣言
PARAM	SUBST と EVAL のためのパラメータ対の宣言
SYMARG	FORMAC の変数としてサブルーチンの引数を宣言し，プログラムの始めに flag を立てる．

この他に FORTRAN の statements, declaration の全部を用いることができる．　（星野幸夫）

B-28. List 処理および Embedding による言語の使いやすさの拡大

D.G. Bobrow and J. Weizenbaum: List Processing and Extension of Language Facility by Embedding 〔IEEE Trans. on Electronic Computers, EC-13, No. 4, Aug., 1964, pp. 395～400〕

プログラム言語を拡大するのに二つの方法が考えられる．第1の方法は言語の syntax を拡大するものであり，この方法によればその言語の中で解釈される表現のクラスが拡大される（eloquence の拡大）．第2の方法は言語の semantics を拡大するものであり，この方法によれば同一の表現に対し，拡大する以前にはなされなかった operation が拡大した後には実行されることになる（functional domain の拡大）．

二つの言語 L, L′ に対し，L が L′ の proper subset であり，L で書かれた program が L′ においても同じ解釈を持つとき，L は L′ に embed されるという．すでに存在するある言語 L の能力より少し大きな能力を持つ言語 L′ を作ろうとするとき，L と独立に全く新しい言語 L′ を作るより，L が L′ に embed されるように L′ を選ぶほうが，新しい言語 L′ の習得および L の compiler の利用という点から見て有利である．この embedding を行なう方法として三つの方法が考えられる．第1の方法は L′ で書かれた statement を L の statement に変換する preprocessor を作る方法である．第2の方法は L に対し新しい basic subroutine をつけ加える方法である．この方法によれば，L がすでに十分な文法構造を有するときには，guest language も host language L の文法構造を利用することができる．第3の方法は，guest language の interpreter を L で書く方法である．この方法では guest language の syntax は L の syntax と根本的に異なっていてもさしつかえない．

本論文では実例として METEOR, SLIP という二つの言語をあげている．METEOR は string の変換に便利な prototype notation を LISP につけ加えたものであり，guest language として COMIT とほとんど同じものを採用している．言語の拡大の方法としては第1の syntax そのものを拡大する方法を取り，embedding の方法としては第3の interpreter による方法を採用している．SLIP は list 処理および記号操作の能力を FORTRAN につけ加えたものである．言語の拡大の方法として第2の semantics を拡

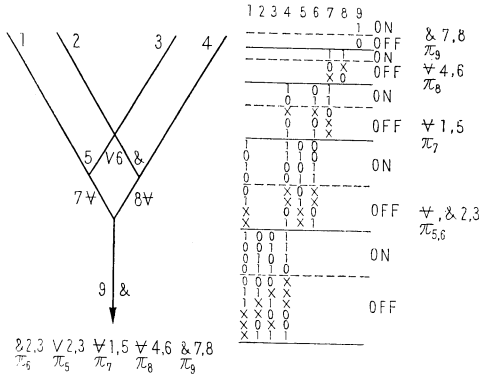
大する方法を取り，embedding の方法としては第2の basic subroutine をつけ加える方法を採用している．（小林孝次郎）

B-29. スイッチング回路の故障点発見法

J.M. Galey, R.E. Norby and J.R. Roth: Techniques for the Diagnosis of Switching Circuit Failures〔IEEE Trans. on Comm. and Electronics No. 74, Sept. 1964, pp. 509～514〕

一出力の組み合わせ回路において，故障を発見し，その場所を指適するに必要な入力信号群を求める組織的な方法について述べたものであって，このためのプログラムが IBM 7090 用に作られた．一例をあげると，8入力のパリティチェック回路において，102 の可能な故障のうち，どれか一つが起ったか否かを発見するためには，4組の試験入力を加えてみればよく，この組み合わせを，IBM 7090 では 2.12 分で求めることができた．この方法は帰還のある回路にも適用できる．

まず組み合わせ回路において，“1”出力（on）を与える入力信号の組（On-Array）と，“0”出力（off）を与える入力信号の組（Off-Array）とを求める組織的な方法としては第1図に示するようなブーリアングラフを元にして，出力側から順次入力枝路の信号の組を求めて行く方法をとる．×印は1でも0でも任意にとり得ることを示す．



第1図

次に故障回路に関しては，どれか一つの枝路のみが1または0に固定されるような種類の故障のみを考える．いま i 番目の枝路が1に固定されたとしたとして，このときの On-Array と Off-Array を前述の方法で求める．この On-Array 中 i 番目の枝路が1のものを0とした入力の組と，正しい回路の Off-Array とを比較し，同じものがあれば，これは正しい回路では0出力を，故障回路では1出力を出す入力信号組となる．これを $GB_i^1(0, 1)$ と表わす．Gは正しい回路，B_i^1 は i 番目の枝路が1に固定された故障回路を意味し，$(0, 1)$ はGでは 0，B_i^1 では1出力を出すことを意味する．同様な方法で，$GB_i^1(1, 0)$，$GB_i^U(0, 1)$，$GB_i^U(1, 0)$ 等が求まる．

これらの入力信号組を，各枝路が故障の場合について求め，その中から，どこかに故障があるとき，少なくともどれか一つの入力信号に対して正しい回路と異なった出力が出るような入力信号の最少組をみつけ出すと，それが故障発見用の信号群となる．

最初に述べたパリティチェック回路の例では，この信号群が4信号からなり，故障点の検出には，8信号が使われる．これによって 51 個の枝路が0あるいは1に固定される 102 種の故障が検出できる．

なお本文中 Fig. 2 の STEP 4 のところで $GB_8^1(1, 0)$ とあるのは $GB_8^1(0, 1)$ のミスプリントである．（元岡 達）

C-30. 高速乗算回路の提案

C.S. Wallace: A Suggestion for a Fast Multiplier〔IEEE, Trans. on Electronic Computers, EC-13, No. 1, Feb. 1964, pp. 14～17〕

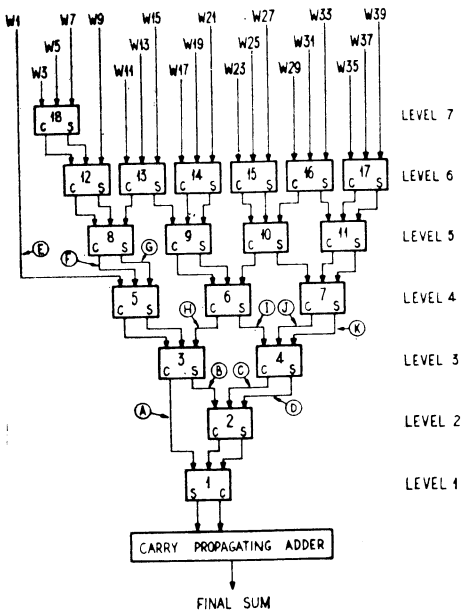
最近の大形科学用計算機においては，乗除算の時間が arithmetic unit 動作時間の半分近くをしめるから，乗除算の金物に投資するのが最も効果的であるという立場に立ち，40 ビットの乗算ならびに除算が各1 μs, 3 μs 以下という方法を提案している．

加算については，三つ以上の数を一度に加えるのは論理が複雑になり得策でない．

ここでは三つの入力から sum と carry の二つの出力を作り出す pseudo adder を考え，この pseudo adder を1段通るごとに summand を一つずつ減らす方式をとっている．この pseudo adder の利点は carry propagation を行なわないため論理が簡単になり，高速並列動作が可能なことである．

第1図に 20 summands の和を作るための 18 pseudo adder の接続を示す．

乗算速度を上げるには，summand の数を減らすのも一方法である．ここでは2ビットずつをひとまとめにして summand を半分に減らす方法が述べられている．



第1図　The adder tree

また除算は逆数の計算により求める方式が提案されている．

この 40 ビット乗算回路の速度を推定するために

1) 1 pseudo adder 当りの delay 60 ns
2) fan out のための high current driver の delay 100 ns
3) carry propagating adder の settling time 100 ns
4) 結果をレジスタに入れる時間　100 ns

の如き仮定を行なうと，乗算時間は 750 ns となる．

必要な pseudo adder 数は 750 個，除算も行なうようにした場合，carry propagating adder を除く全半導体素子数は，トランジスタ 4,591，ダイオード 33,083 である．　（幸野真士）

C-31. DONUT: Threshold gate 計算機

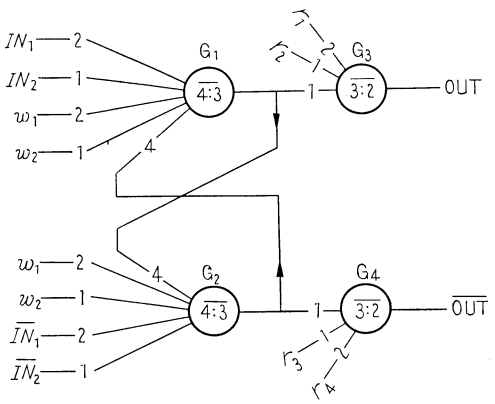
C.L. Coates, P.M. Lewis, II : A Threshold Gate Computer〔IEEE, Trans. on Electronic Computers EC-13, No. 3, June, 1964, pp. 240～247〕

DONUT (Digitally Operated Network Using Thresholds) とは基本論理素子としての "threshold gate" を評価するために製作された小形の汎用計算機で，実際の threshold gate について，部品や信号のバラツキを調べたり，部品の節約，システムの信頼性を検討した．メモリ，レジスタ，アキュムレータなどを含む DONUT の全システムは約 3,500 の threshold gate から成っており，その仕様を第1表に示す．

第1表　DONUT Specifications

Mode of Operation	Parallel
Internal Number System	Binary
Word Length	14 bits
Number of Instructions	15
Memory Size	64 words—Threshold Flip-Flop
Arithmetic Speed (including memory access)	
Add	7.1 μs
Subtract	10.2 μs
Multiply	48.5-107.8 μs

またシステム設計，論理設計について述べており，一例として典型的なレジスタを第3図に示す．各円は threshold gate を，円内はその gap を示している (4 : 3 は gate が "1" の出力を出す最小入力和が 4 であること，"0" の出力を出す最大入力和が 3 であることを示し，その上の棒は出力の否定を示している)．



第3図　A typical register

IN_1, IN_2 などはデータ，w_1, w_2, r_1, r_2 などは制御信号である．記憶は G_1, G_2 の重み4の入力により行なわれる ($w_1=w_2=0$ の場合)．IN_2 と $\overline{IN_2}$ をレジスタにセットする場合には $w_1=w_2=1$, $IN_1=\overline{IN_1}=0$ とし，IN_1 と $\overline{IN_1}$ をセットしたい場合には $w_1=1$, $w_2=0$ にすればよい．gap が 1 : 0 から 5 : 4 の範囲の回路設計は比較的易しいが，アキュムレータのように gap が 9 : 8, fan-in 32 を要求するものについては信号レベルなどに対する規定が厳しくなり，この公差

が実現可能かどうかを評価するのも，このプロジェクトの目的である．

部品の節約の問題に関しては NOR 回路との比較という形で検討したが，その結果を第6表に示す．

第6表 Comparison of Threshold and NOR Gate Realizations of DONUT

Gap	9:8	5:4	3:2		1:0—NOR Gate	
Fan-In	32	15	15	5	15	4
Whole Computer		614	882	1052	1702	2127
Accumulator Only	56	112	168	224	420	580

これらから threshold gate によれば，約1/4に部品を節約することが可能であるという結論が得られた． （幸野真士）

C-32. IBM システム 360 の構造

G.M. Amdahl, G.A. Blaauw and F.P. Brooks, Jr.: Architecture of the IBM System/360. 〔IBM J. Res. and Dev., Vol. 8, No. 2, April 1964, pp. 87～101〕

IBM システム 360 の設計思想を述べている．

I. 設計目標

前提：この 10 数年来の主要な発展傾向を考慮すること（広範な事務分野への応用，トータルシステムとして始めて発揮できる値打の増大，コンパイラの普及，ファイルメモリの発達，記憶容量の拡張可能性，実時間処理への応用）．

目標：最新の設計概念を取り入れること．
開放型（オープンエンデッド）設計．
一般性．
高能率．
プログラム コンパチビリティ．

最新の設計概念としては，

1．ファミリの形成 2．インタフェースの統一 3．(問題の解決量)/月 の意味での性能向上 4．プロセッサ，I/O，ファイルメモリの全体を最有効に働かせる．5．I/O チャネルと IOCS 6．完全なモニタシステム 7．I/O，メモリ，プロセッサをリダンダントに構成し得ること 8．32 kW は不十分である．
9．浮動小数点で 36 ビットは不十分な場合がある．
10．ハードウェアのチェック並びに誤りの診断
11．プログラムの誤りの診断
などをあげている．

II． 問題点

以下の如き諸点につき，種々検討を加え，取捨選択を行なった．

1． データの構成．
語は 2^n とするか，または 3×2^n とするか？
6 ビットコードか，または 4/8 ビットコードか？
浮動小数点数を 48 ビットにするか，64 ビットにするか？
浮動小数点数の 16 進表示の可否．
シグニフイカント アリスメチックの可否．
2． 2の補数表示か 符号＋絶対値表示か？
3． 可変長桁か，または固定長桁か？
4． ワードマーク方式かまたはカウント方式か？
5． ASC II 対 BCD コード
6． 命令方式
スタックの可否．
アドレスのきりつめの可否．
10 進アドレスの可否．
複数アキュミュレータの可否．
メモリの多重構造の可否． （村田賢一）

C-33. IBM システム/360 におけるサービス面を考慮した設計について

W.C. Carter, H.C. Montgomery, R.J. Preiss and H.J. Reinheimer: Design of Serviceability Features for the IBM System/360 〔IBM J. Res. and Dev. Vol. 8, No. 2, April 1964, pp. 115～126〕

サービスコールの累積百分率をコールの長さの函数として表わすと，初期故障期間を経たのちはその分布曲線の最大値，メジアン，平均値は一定の関係を保つことが経験的に知られており，これらの量を下げることを目的として，ここでは論じられている．その優先順位は

（1） 不定期のメインテナンスコールの最大値を大幅に下げる．

（2） 不定期のメインテナンスコールのメジアンおよび平均値を十分に下げる．

またプログラムコンパビリティと単位回路の標準化が，サービス面に潜在的な利益をもたらすことから，

（3） 計算機構成とモニタプログラムといった組み合わせに最も効果的な，統一された標準のサービス用プログラムおよび手順を開発すること．

である．

(1) については，障害が間歇的であるため，診断に

多くの時間を要することが原因であるから，そのデータの蒐集法および分析法を確立し，故障個所をよけてプロセスを続行する，あるいは命令再試行などの並用によって解決をはかり，(2) の解決法として，従来直接プログラムによってわずかな範囲のエレメントの不良しか探し得なかったことと，命令遂行サイクルの途中の状態を容易に知り得なかった，ということから Fault Locating Tests の開発についてくわしく述べられている．

そして次のような機能がハードウェアは追加されたことが最後に要約されている．

（1） 情報および制御用のエラー探索回路．

（2） エラー発見時に直に制御を止め，CPU の状態をそのままメインメモリに移す．

（3） メインメモリの情報のパターンをそのまま CPU の記憶素子に移す．

（4） 独立制御によって指定した回数だけマシンクロックを進める．

（5） 計算機のあるべき状態と実際の状態を比較し得る特別な比較回路．

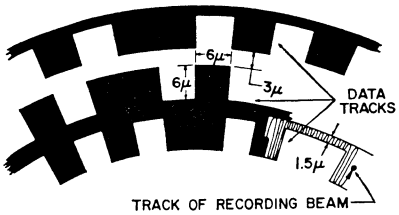
（6） 特別な命令あるいは全く独立した回路によるこのハードウェアの制御．　（岸　哲雄）

D-34. ディジタル情報の電子ビーム記録

R.F.M Thornley, A.V. Brown and A.J. Speth: Electron Beam Recording of Digital Information [IEEE Trans. Electronic Computers EC-13, No. 2, 1964, pp. 36～40]

読出専用の写真型記憶装置へのディジタル情報の記録に電子ビームを使用した方式が開発された．電子ビーム記録方式は従来の光学的記録方式と比較して，エネルギー密度が千倍も高く，また輝度および位置の制御の容易さなどの利点があり，記録密度 10^6 ビット/cm^2 で 10^6 ビット/秒の高速度記録を行なうことができる．これにより写真型記憶装置をたびたび内容変更を行なう 10^8 ビット以上の記憶装置として利用できるようになった．またこの読出しは光学的に 10^6 ビット/秒の速度で行なわれる．

本装置は IBM language translation machine に使用する円板記憶装置の記録用として設計されたもので，各円板は 5×10^7 ビットの記憶容量をもち，この記録に当たって従来の光学的方式では 24 時間かかったものが，電子ビーム方式では 1 分以内で記録できる．



第 1 図　Data format and recording techniques

情報は写真円板に同心円上に黒白のマークで第 1 図のような形状および寸法で記録される．この記録方法は回転円板上に直径 1 μ 以下の電子ビームを情報信号により変調した RF 信号で半径方向に偏光して行なうもので，この包絡線が情報である．情報の記録は磁気テープからトラック分の情報をバッファーメモリに一たん読み取り，円板回転に同期したタイミングで RF 信号を変調して行なう．またトラック密度を上げる目的で第 1 図に示すように変調器の出力信号の位相を反転し，基線の反対側に次のトラック情報を記録する．本装置は 100 トラックまで記録できるが，最終装置では 1,000 トラックの記録が必要となるので上記手段を 10 段階行なえるものを再設計中である．

記録材料としては必要な分解能を有し，市販中最高感度の Kodak 製の高分解能感光剤を使用した．この粒子は 0.5 μ 以下であるので解像力はむしろ電子ビームのエネルギー分布によって決定されている．

装置の記録機構は電子顕微鏡銃の陽極部を一部変更したもので，二つの独立した偏光方式すなわち変調 RF 信号用の静電偏光板および情報トラック撰択用の電磁コイルを備えている．写真円板の回転機構は回転中心の変動が ±1 μ 以内であるような精密ベアリングを使用し，ベアリングは真空装置中に入れてある．またこの駆動には速度可変のモータを磁気カップリングにより接続し，振動の伝達を少なくしてある．

（三田順業）

D-35. 円筒形磁性薄膜記憶素子の特性

C.J. Townsend and P.E. Fox: Cylindrical Film Memory Devices Characteristics [IEEE Trans. on Electronic Computers EC-13, No. 3, June, 1964, pp. 261～268.]

円筒形磁性薄膜記憶素子の諸特性を述べて，これから決定される試験条件をあげ，高速記憶素子として十分実用になるとしている．この素子は，外径 0.625

mm のガラス管にクロムと金を蒸着したのち，円周方向の磁化容易軸を有する厚さ 6,000 Å の Ni-Fe を電着し，長さ 2.5 mm に切断して製造する．切断したのはビット間のクリープを防止し，歩留りをあげるためである．

この素子に特有な端部効果，入力情報パターンの効果，妨害特性を中心にして特性を示す．素子の軸方向の長さに比して語線の幅（0.25 mm）は短いので，書き込み後の端部の磁化状態は書き込み以前の状態と同じである．しかるに，読取り出力は書込み部分と端部の磁化状態によるので，端部の磁化状態に左右されることになる（端部効果）．したがってパルス試験では書込み動作の前にリセットパルスを加えて端部の磁化状態を一定にし，最低読取り出力を測定している．

情報“1”を無限回読み書きしたのち，情報“0”を N回読み書きし，再び情報“1”の読み書きを行なった場合，あとの第1回目の情報“1”の読取り出力は Nの増加と共に減少し，リセットを導入したパルス試験で測定される読取り出力の値に漸近する（入力情報パターンの効果）．この現象は磁区の拡りによるものである．

語線には，0.8 A（幅 50 ns）の読取りパルスと 1.2 A（幅 70 ns）の書込みパルスを加える．読取り，書込み両パルスを分離すると，駆動回路の半導体の消費電力が減少し，端部効果を除去するのに有効とされる広く書き込んで，狭い部分を読み取ることが可能になる．なお，1.5 A の書込みパルスの与える漏洩磁界が及ぼす妨害は見出せなかった．

自動プロッタによって情報パルス—出力電圧—特性を測定し，情報パルスを 110 mA（幅 100 ns）と決めた．書込みパルスの立下り時に情報パルスの波形が平坦部に達していれば，情報パルスの幅と立上り時間は書込み特性に無関係である．

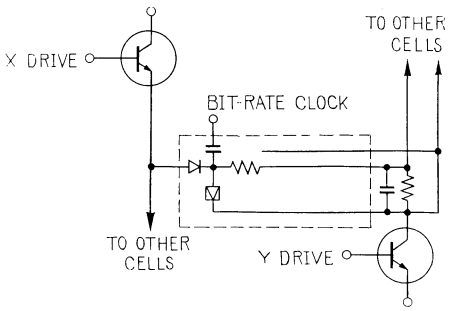
妨害情報パルスによる妨害特性を測定した結果，最悪の素子において，10^8 回の妨害パルスを加えた時の妨害閾値は，10^3 回の場合の 8% 減であった．したがって，妨害パルスの振幅を 10% 増しとして 10^3 回妨害することにより等価的に無限回の妨害パルスを加えた動作を行なわせると共に，その正常な動作を確認した．

数千個の素子が自動試験機によって試験され，定められた規格（I_W=1,080mA, I_R=720 mA, I_B=±100 mA, I_{BD}=∓151 mA, I_{WD}=20 mA, dV_1>10 mV, dV_2<−10 mV）を満足する素子がえられた．試験済みの 2,500 個の素子を 2,048 語の記憶装置のアレーに組み込んで動作させている．（村上 坦）

D-36. トンネル・ダイオード遅延線記憶装置

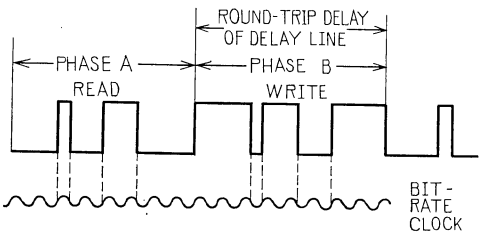
H.H. Harris, Jr., and W.D. Pricer: Tunnel Diode Delay-Line Memory. 〔IEEE Trans. on Electronic Computers, EC-13, No. 3, June 1964, pp. 269～272〕

この論文は，トンネル・ダイオードと遅延線を組み合わせた記憶方式および 256 語，8 ビット/語のモデル・セットについて述べている．第1図はそれぞれ1個のプリントされた遅延線，トンネル・ダイオード，



第 1 図

通常ダイオード，抵抗およびプリントされたコンデンサよりなる基本的な記憶素子を示している．情報は2レベルの電圧によって表わされ，電圧レベルの変化はクロックの位相に同期している．遅延線は，その特性インピーダンスより低い抵抗で終端してあるので，トンネル・ダイオードに戻ってくる反射パルスは逆位相となる．トンネル・ダイオードで反射パルスとビット・レイト・クロックが結合され，トンネル・ダイオードは逆の安定状態にトリガーされる．すなわち，一往復ごとにパルス列が逆転する．この特性を利用して，単一極性の制御で情報を消去したり，書き込みを



第 2 図

行なったりすることができる．情報とビット・レイト・クロックの関係は第2図の如くである．また読出期間中，同時にXおよびYを駆動すると，それに接続された通常ダイオードが導通し，トンネル・ダイオードが高電圧状態にスイッチする．したがって，蓄積された情報の再生を阻止できる．なお選択は記憶素子の通常ダイオードに接続されたXドライブ線の電圧を上昇させ，遅延線に接続されたYドライブ線の電位を低下させることにより行なわれる．また，X駆動回路は，読み出および書込信号によって制御され，読み出しを非破壊的に行ない，再生作用を続行させることも可能である．読出しの際，選択されたXドライバーのトランジスタは，電流検出型のベース接地増幅器として動作し，読出情報は，上記トランジスタのコレクタから取り出すようになっている．

情報は直列的に記憶されているので，必要に応じて直列—並列変換および並列—直列変換を行なわねばならない．8または16ビット/語程度の短かいビット構成では50 ns以下のアクセス・タイムを得ることが可能であろう．256語，8ビット/語のモデル・セットについて，記憶装置のブロック図，駆動波形などが示されている．　（五十嵐　良）

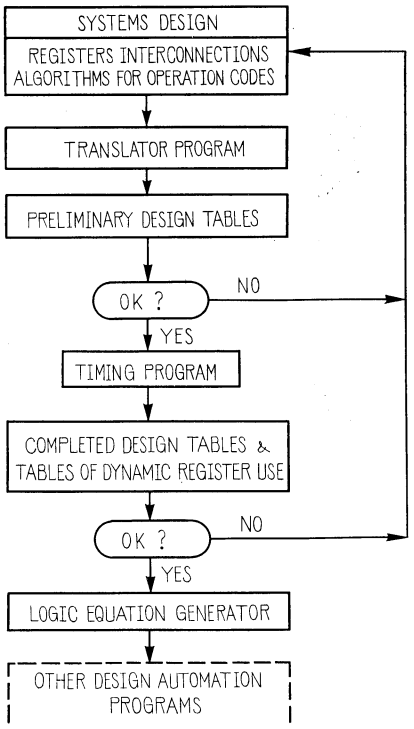
D-37. 論理設計トランスレータの実験

R.M. Proctor: A Logic Design Trnslator Experiment Demonstrating Relationships of Language to Systems and Logic Design. 〔IEEE Trans. on Electronic Computers, EC-13, No. 4, Aug. 1964, pp. 422～436〕

論理設計トランスレータシステムLDTは，トランスレータ，時間関係の解析，論理式作成の3部分から成る．これらを記述する言語は，レジスタの名前，長さ，を定義する宣言部と，演算，転送，条件，メモリとの入出力などの動力を記述する操作部がある．

レジスタの宣言においては，名前，長さのほか，それを幾つかの部分に分けて名前を付けることができる(ストラクチャーリスト)．特別なレジスタにおいては時間おくれが定義される．転送の宣言においては，各レジスタに対して，転送情報源となるレジスタが定義される．

トランスレータは各命令に対するシーケンスを情報径路に分解し，DESIGN TABLEに記入する．すべてのシーケンスを分解したあと時間解析を行なってタイミングを定める．これよりレジスタのダイナミックな使用状態に関する情報を表として得ることができる．



レジスタの宣言

A(1, 48)(A 1(1, 12), A 2(13, 24), A 3(25, 26), A 4(37, 48))$

identifier　size　structure list

転送の宣言

A=(KE, A, LOGCU, ZERO, ARITH, C, LM)$

destination register　source register

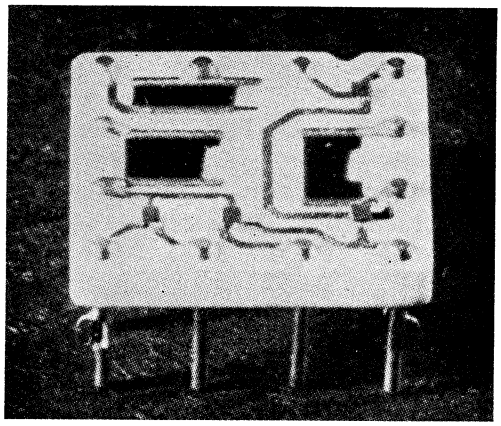
（加藤満左夫）

D-38. 固体論理技術：汎用高性能微小回路

E.M. Davis, W.E. Harding, R.S. Schwartz and J.J. Corning: Solid Logic Technology: Versatile High-Performance Microelectronics. 〔IBM J. Res. and Dev., Vol. 8, No. 2, April 1964, pp. 102～114〕

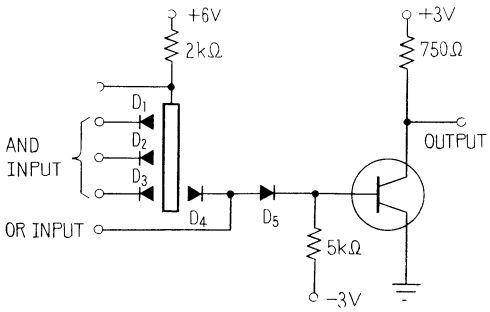
IBM 360に使われている基本モジュールについてその特徴，製法などが詳述されかつ本技術の応用面が述べられている．

本論文は，ソリッドロジック・テクノロジー(SLT



(a) Completed AOI module, without overcoating.

第1図 AND/OR INVERT logic module



(b) Logic circuit.

Power dissipation in mW

	On	off		On	off
Resistors (All R: 5%)	28	19	D_4	1	1
Transistor	7	0	D_5	1	1
D_1, D_2 D_3	0	2	Total	37	23

と呼ばれる新しい技術により，超小形電子回路を実装するに当たっての従来よりの問題点を解決し，高性能かつ汎用性にとむ回路を低価格で提供することを可能ならしめたものである．

基本回路は第1図（b）に示す如くアンド・オア・インバート（AOI）論理回路であり，約 12 ミリ平方のセラミックウェハ（95% アルミナ）の上に，トランジスタ，ダイオード対および抵抗が形成される（第1図（a））．

トランジスタおよびダイオード対はそれぞれシリコンプレーナ型で，従来の方法で1枚のウェハ上でエッチング，拡散などが施されるが端子処理以降の過程が異なる．個々に切られた半導体チップはガラスでおおわれ保護される．寿命試験の結果ハーメチックシールの必要がないことが判明した．これらのチップはセラミックのウェハにはんだボールを介してはんだづけされる．

配線は謄写印刷の技術を用いて印刷される．貴金属を含んだインキが用いられ 800°C で焼結する．抵抗も配線と同様に印刷技術により印刷される．抵抗体としてはパラジウム，銀，ガラスの混合物を有機溶媒に溶かしたものである．抵抗値のバラツキは ±15% であるが，最小低い所を目標としあとで機械的研摩法により希望値の ±1% の範囲に押えることができる．

モジュールの完成体は機械的な保護並びに高湿条件下におけるハンダの腐触をさけるためにプラスティックのケースに納められる．半導体および抵抗は 5,000 または 10,000 時間の寿命試験を行なった結果，十分満足すべき結果が得られた．回路はファンイン・ファンアウト共に5のとき 20 ns の遅延時間を示す．

本技術の応用としては，*LC* 共に可能であり，また一面上での線のブリッジすなわちクロス・オーバも可能である．またウェハの大きさについても制限なく，高速度の回路には小さなウェハを，複雑なシステムに対しては大きなウェハを用いることにより，広範囲にわたって能率の良い回路が構成され得る．

最後に本技術によればトランジスタ・ダイオード，抵抗などがそれぞれ独立に精度良く製作されるから，回路の消費電力が最小になるような設計が可能になり，また各素子が独立であるゆえ寄生容量が少なく回路の動作速度が理想的に高くし得ることが強調されている．（佐藤 繁）

E-39. Threshold Logic による文字の識別

S. B. Akers. and B. H. Rutteer : The Use of Threshold Logic in Character Recognition [Proc. IEEE. Vol. 52, No. 8, Aug. 1964, pp. 931～938]

Character Recognition は二つの部分，すなわち識別されるべき characters の特徴を求める receptor と，これらの特徴によって characters の名前を指定する部分 categorizer に分けられる．

Threshold gate は n 個の二進 inputs $x_1x_2\cdots\cdots x_n$ と正負の値をとる weights $w_1w_2\cdots\cdots w_n$ および数値をもつ threshold T とを有し，$\sum w_ix_i \geqq T$ なら 1，そうでないならば 0 を output するものである．

12 個の Letters が translation, stretching および

compressing されても識別されることが述べてある．

i 番目の Letter から receptor は 11 個の特徴 c_{ij}, $j=1,2\cdots\cdots11$ を threshold logic によって求め，特徴の存否に対応して 1, 0 を output する．たとえば，ある Letter A の特徴 c_{1j} の最適 weights w_{1j} を求めるには，まず全ての j に対して $c_{1j}=1$ となるようにするため，$c_{1j}=0$ となるような，$c_{ij}(i\neq1)$ をそれぞれの complement に置換えて，結果を matrix (c_{ij}) とする．このようにすると問題は

$$\sum_j w_{1j}c_{1j}-\max_{i=1}^{12}\left[\sum_j w_{1j}c_{ij}\right]$$

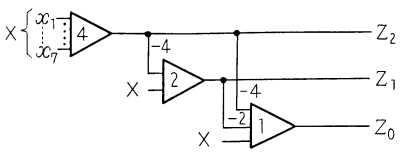
を最大にする正の weights w_{1j} を求めることになる．ただし $\sum w_{1j}=1$．これは，$c_{1j}=1$, $\sum_j w_{1j}c_{ij}=1$ であるから，第2項を最小にすればよいことになる．さらに (c_{ij}) の complement を (c_{ij}^*) すれば

$$v=\min_{i=2}^{12}\left[\sum_j w_{1j}c_{ij}^*\right]$$

をmaximum にする two-person zero-sum game の値 v を求めることに帰着する．この方法によって解かれた最適の weights w_{ij} が示されている．

このようにして求めた weights を categorizer において利用して識別している．

threshold gate を用いる方法はすでになされている方法たとえば curve following, waveform matching, positioning, moment 法などに代るべきものではなく，これらの方法に対して有効な手段を提供するものである．



なお用いられた threshold gate は，上図のようなものであって，Xによって inputs $x_1x_2\cdots\cdots x_7$ を示し，$z_2z_1z_0$ は inputs のうちの“1”の数を与える3ビットの2進数である．また gate 内に示される数は threshold を示し，inputs 上の数は weights を示す．ただし weight が与えられていないのは weight 1 とする．　　（星野幸夫）

E-40. 多レベルブール函数の簡単化

E.L. Lawler: An Approach to Multilevel Boolean Mimmization〔J. ACM, Vol. 11, No. 3, June 1964, pp. 283～295〕

単一リテラルの結合を零レベルと呼び，ϕ_1, $\phi_2\cdots\cdots$ が N レベルのとき ϕ_1, $\phi_2\cdots\cdots$ の結合で $N+1$ レベルを定義する．結合が和であるとき SN 形，積であるとき PN 形と呼ぶ．

N レベル形の簡単化ということは，与えられた函数をちょうど N レベル形で表わし，その中に含まれるリテラルの数を最小にすることである．また．すべての N の中で最小な形を絶対最小形という．

PN 形 ϕ が f を含むとき，ϕ は f の PN インプリカントと呼び，ある一つの PN インプリカントが他のそれを含むことが無いとき前者をプライム PN インプリカントという．

この論文における最小化の方法は，最小な SN 形はプライム $P(N-1)$ イリンプリカントの和で構成されるから，まず低いレベルから出発して多レベルのプライムンインプリカントを求めようというものである．

そのために，多レベルの最小形とプライムインプリカントの性質を論じ，多レベルのプライムインプリカントを求める例を挙げている．また求める手順を減らす方法を示している．

SN 形か PN 形のいずれかであるもの，すなわち同時に両者でないものを proper PN 形または SN 形という．ϕ が $N+2$ 以上のリテラルを含まない形であるとき，ϕ が最小 PN 形若しくは SN 形であれば絶対最小形であるとの定理から絶対最小形について論じている．　　（加藤満左夫）

F-41. 取引活動のゲーム・シミュレーション

R.J. Meeker, G.H. Shure and W.H. Moore, Jr.: Real-time Computer Studies of Bargaining Behavior: The Effects of Threrat upon Bargaining〔SJCC, AFIPS Conference Proceedings 1964, Vol, 25, 1964, pp. 115～123〕

計算機の能力が最高度に発揮するのは，シミュレーションとそれに伴なうデータの分析においてである．

この論文では，その一例として「売込活動におけるコミュニケーション・ゲームの戦術効果」を取り上げている．

ここで用いた実験方法は，PHICO 2000，テレビ，応答用コンソールを使い，二人一組として，18～24人を同時に実験した．各オペレータは誰と組になっているかがわからないように，個室に入れられ，テレビだけからチャネルの状態や，自分の行為の結果を知るだけで，お互が直接話し合う手段はなにもない．このチ

ャネルは，最高6単位からなっている．このチャネルに

（1） 6単位のメッセージを送ると 25 セントの儲け

（2） 4単位のメッセージを送ると 10 セントの儲け

（3） その他は無しである．

ただし，1試行で一人最高 25 セントまで稼ぐことができるが，一組の最高は 35 セントまでである．

ここで，1試行はコンソール上のスイッチを各組が交互にそれぞれ 15 回押す．

ただし，1スイッチでできる仕事は，以下の三とおりのうちの一つである．

1. 1単位のメッセージをチャネルに送る．
2. 1単位のメッセージをチャネルから引き出す．
3. なにもしない．

売込み戦術上の条件を表わすためにブロック行為というものを設けた．これは相手がチャネルにメッセージを送るのを邪魔するもので，この行為はしてもしなくても自由であるが，一人1試行に1回しか使うことができない．

このような条件のもとで，すべてのオペレータはできるだけ金を多く稼ぐように努力する．

この実験では以下の三とおりの場合について比較検討している．

1. 一組の2人ともブロックができる場合．
2. 片方だけブロックができる場合．
3. 両方ともブロックができない場合．

この結果，ブロックが使える場合は躊躇なく使い，ブロックされた方でもほとんど同時にブロックし返している．ブロックの手段を持たない人より持っている人の方が良く，持っていても使わない人の方がさらに良いことが解った．しかし，いったんブロックされたら，直ちに報復処置を取る方が，報復処置を取らないか，または取るにしても相当時間がたってから取る場合より商談がまとまりやすいことが解った．

（高橋生宗）

F-42. 大形非同期計算機のタイミングシミュレーション

R.L. Chew: Notes on Timing Simulation of a Large Asynchronous Computer〔Computer J. Vol. 7, No. 2, 1964, pp. 122〕

大形計算機アトラス1号機はケンブリッジ大学で作られたが，続いて2号機がケンブリッジ大学と連帯してフェランチで設計されている．この Atlas 2 で，いかに命令が重なり合って実行されるかという重要なやり方を，小形計算機シリウスでシミュレーションしたという報告である．

このようなシミュレーションを行なうと，プログラムにとって能率のよいループを開発したり，設計に役立てることができる．命令実行の重なり工合をはっきりさせるため，設計者は命令の各部を次のような節に分解した．すなわち，命令の要求，命令の待ち合わせ，命令の解読，被演算数の要求，被演算数の待ち合わせ，命令の完成，これらの6個の節である．待ち合わせの原因には，累算器，インデクスレジスタ，磁心記憶装置などが空くのを待つ場合があり，いずれも待ち合わせ時間はあらかじめ決められている．

注意しなければならないのは，磁心記憶装置が4箇の独立したスタックに分かれていること，トンネルダイオードでできた早いレジスタが 40 個あり，このうち32個にはプログラムの64命令以下のループなら入り得ること，さらに残りの8個は作業用に使用できることなどである．

したがって，調べようとするシーケンスについては，命令の数だけでなく，一つ一つの命令について次のようなこまかいことを指定しておく必要がある．すなわち，ジャンプ命令か否か，命令のあり場所，修飾に使用するインデクスレジスタの数，被演算数のあり場所，そして機能などである．

シーケンスに費やす時間は，ns 単位で各節についての時間を加えて求める．ただし，ある命令の解読は，その前に実行しつつある命令が被演算数を要求してから行なわれ，その結果がジャンプでなければ，その次に実行すべき命令の要求がなされる．このように三つの命令の実行が重なることがあるので，待ち合わせも，この命令の待ち合わせ，前の命令の被演算数の待ち合わせ，次の命令の待ち合わせ，この命令の被演算数の待ち合わせ，という順序で展開されるものとみなし，待ち合わせ以外の節の実行に要する時間は待ち合わせの時間に吸収されるものとしている．

こうして結果を μs 単位でプリントするプログラムをつくるのに約2カ月もかかったが，スーパーバイザーのループに有益な寄与をしている．さらに，サイクル時間の異なった磁心記憶装置や，累算器がどの程度有益なものかを評価するのに役立つと結んでいる．

（川合英俊）