論　文

シフトレジスタ形しきい値素子回路網の計算機設計*

阿　江　　忠**

Abstract

With respect to the design of an autonomous sequential circuit, a unified theory using linear algebra on Galois field is well-known. In this paper we discuss the design of a shift-register-type sequential circuit using a threshold element. In general the behavior of a sequential circuit using threshold elements has been rather investigated. On the other hand, the synthesis of such circuits seems to be difficult, since the synthesis of a threshold function itself requires a number of procedures.

The refore we introduce a method using a computer and make a program for the purpose of hardware realization. One problem is how to treat "don't cares" since a required output sequence has not usually the length equal to 2^m, where m is the number of shift registers. In order to explain this treatment we use the concept of problem solving method. The method to decide whethre a function f is a threshold one or not depends on 2-asummability. As a result a program for design is obtained within $m=7$ and it is shown to be profitable.

1. まえがき

しきい値素子回路網の順序機械としての動作については，自己組織系の動作モデル[1]として，あるいは，パターン認識における学習モデルとしての位置づけがもっとも適していると考えられるため，オートノマス動作に関して主に研究がなされている[2)~4)]．

一方，出力周期系列を与えたとき，そのような出力を与えるオートノマス動作をするしきい値素子回路網を設計する問題を"構成問題"と呼ぶとき，構成問題の一般的解法はしきい値素子自体のもつ性質の難しさのため必ずしも容易ではない[5)]．メモリ素子をn個用いる回路網は2^n個の状態数をもつが，一般には周期系列の長さは2のべきであるとは限らない．したがって構成問題においてしきい値関数を決定しようとする場合，与えられた条件には関数値が0, 1のどちらでもよい定義域が生じる．すなわち，Don't Care（以下D. C と略す）が存在するから，計算機設計のプログラム作成にあたっては，このことを考慮に入れなければならない．

ところで，所望の出力周期系列を発生するオートノマス順序回路の設計は，シフトレジスタ形モデルに関してすでに統一的方法が確立されている[6)]．ただし，ガロア体上の議論であるから，ゲート回路はring sum演算となり，これよりしきい値ゲートへの変換法は一般にはできない．

本稿においては，構成問題の複雑さを軽減するため，ひとまず，シフトレジスタ形構成のしきい値素子回路網を対象に構成問題を論じた．上述のガロア体上の議論のような統一的表現はできないから，アルゴリズム的方法を用いなければならない．したがって，いかに能率のよい計算機設計をするかが問題となる．

本稿では Problem Solving 的な見地から問題をとりあげ，所望の周期系列に対してしきい値素子を真理値表で与える計算機設計法を以下で述べる．

2. シフトレジスタ形しきい値素子回路網

本稿の設計問題はつぎのように要約される．所望の周期系列を $a_1, a_2, \cdots, a_n$ ($a_i \in \{0, 1\}$ とし，あとは同じ系列を繰返す）と与えたとき **Fig. 1** に示すシフト

* Computer Design of a Threshold Logic Network whose memories are Shift Resistors, by Tadashi AE (Dept. of Electronics, Fac. of Eng., Hiroshima University)

** 広島大学工学部電子工学科

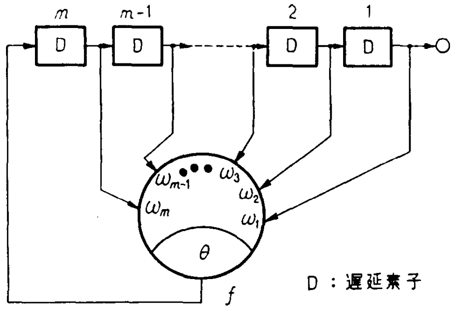

Fig. 1 Shift-type Sequential circuit with a threshold

Table 1 Relation between a output sequence and a memory one.

時刻 ＼ レジスタの番号	m	$m-1$ ………	3	2	1
0	a_m	a_{m-1} ………	a_3	a_2	a_1
1	a_{m+1}	a_m ………	a_4	a_3	a_2
2	a_{m+2}	a_{m+2} ………	a_5	a_4	a_3
⋮	⋮	⋮	⋮	⋮	⋮
m	a_{2m}	a_{2m-1} ………	a_{m+4}	a_{m+2}	a_{m+1}
⋮	⋮	⋮	⋮	⋮	⋮
$n-1$	a_{m-1}	a_{m-2} ………	a_2	a_1	a_n
n	a_m	a_{m-1} ………	a_3	a_2	a_1

レジスタ形しきい値素子回路網において（初期条件を適当に与えることにより）出力端子から順次この系列を発生させるような Fig. 1 のしきい値素子を求めることである．このとき出力周期系列の周期長 n に対し必要なシフトレジスタの数 m の下限は $\log_2 n$ であるから，m はこの値にできるだけ近いことが望ましい．

このような問題の設定において，しきい値関数 f が満たさなければならない関係式は，時刻とともに **Table 1** のようにシフトレジスタのメモリは遷移するから，

$$\left.\begin{array}{l} a_{m+1}=f(a_1, a_2, \cdots, a_m) \\ a_{m+2}=f(a_2, a_3, \cdots, a_{m+1}) \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ a_{m+i}=f(a_i, a_{i+1}, \cdots, a_{i+m-1}) \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ a_n=f(a_{n-m}, a_{n-m+1}, \cdots, a_{n-1}) \\ a_1=f(a_{n-m+1}, a_{n-m+2}, \cdots, a_n) \end{array}\right\} \quad (1)$$

となる．出力周期系列の長さ n が $n=2^m$ のときはよいが，一般には $n<2^m$ となるからしきい値関数は D.C を生じる．

なお，$m=n$ とすれば trivial に f が決められることからわかるように m を最大 n まで許すなら，しきい値関数 f の存在はつねに保証される．

所望の出力周期系列は，過渡状態の系列をもたないものとしたが，過渡状態の系列をもつ場合もほとんど同じ方法でとり扱える．

3. しきい値関数と Don't Care

一般にブール関数がしきい値関数であるための必要十分条件としてつぎの定理が知られている．

〔定理 1〕[7]

ブール関数が一個のしきい値関数となるための必要十分条件は，オンセットとオフセットを互いに異なる半空間に分離する線形な超平面が存在することである．

なお，オンセットとは n 変数ブール関数を 2^n 次元超立方体で表わしたとき，関数値が 1 となる頂点の集合であり，オフセットは関数値が 0 となる頂点の集合である．

与えられたブール関数が，しきい値関数であるか否かを実際に判定する方法に関しては，いろいろな方法が提案されている[7][8]．しかしながら，実際に計算機プログラム化するという観点からはやはりつぎの asummability が適当であると思われる．

〔定義〕[7]

ブール関数 F においてオンセットの頂点のうちから重複を許してとり出したものを $x_1, \cdots, x_k$，オフセットの頂点から同様にしてとり出したものを $y_1, \cdots, y_k$ とする．このとき，$\sum_{i=1}^{k} x_i=\sum_{i=1}^{k} y_i$ となる頂点の選び方があれば F は k-summable であるという．また，頂点をどのように選んでも上式の等号が成立たないとき F は k-asummable であるという．もし，任意の k に対して k-asummable であれば単に asummable という．

〔定理 2〕[7]

ブール関数 F がしきい値関数であるための必要十分条件は F が asummable であることである．

なお，変数の数 n が 7 以下ならば k-asummable の k は 2 で十分なことが知られている[7]．（その他詳細は文献 9）に詳しい．）

関数がしきい値関数であると判定できたら，D.C をもつ関数の場合これを完全に埋めなければしきい値関数は完全には決定しないので具体的な回路設計ができない．

ところで，しきい値関数の判定と D.C がある場合の関数構造の決定を別々に行なうことは（heuristic な方法のときもそうであるが）計算機アルゴリズムをつくる場合には得策ではない．D.C を割当てるアルゴ

リズム，すなわち関数構造決定のアルゴリズムが完全であれば構造決定のできない関数はしきい値関数でないと判定できる．

このような観点に基づく計算機用の関数構造決定アルゴリズムを次に述べる．

D.C の処理方法は，人間が行なうのと機械が行なうのとでは大きな違いがある．人間は全体を把握したのち細かい操作を行なえるから，変数の数が少ないときや極端にオンセットやオフセットが少なく直観的思考の働らくときはさほど D.C の処理を考えなくてもよい．しかしながら，計算機処理においては，たとえ変数が少なくてもきっちりアルゴリズム化しておく必要が生じる．D.C に能率よく関数値 0 または 1 を割当てようとするとき次のような方法が考えられる．(まず，$n \leq 2^m$ となる最小の m が与えられたとする．)

(1) 瞬時割当て法

D.C すべてに対し関数値 0 あるいは 1 をそれぞれ同時に割当てる．詳しくは次のような区別ができよう．

(1-1) D.C すべてに対し関数値割当てを適当に(例えば，ランダム)に行ない，関数の判定を行なう．しきい値関数にならないときは割当てをやり直す．

(1-2) D.C を含む関数，すなわち不完全定義関数に対する判定法[9]でしきい値関数か否かを判定し，しかるのち正確な D.C 割当てを行なう．

(1-1) の方法は D.C のない完全定義関数に対する判定法ですむという利点はあるが，最大手数は D.C の数を $p(=2^m-n)$ とするとき 2^p で与えられる．もちろん関数がしきい値に“近い”とき，すなわち，どのような D.C 割当てでもしきい値関数となるときは実際の手数はずっと少なくてすむ．しかし，最悪状態ではよい方法とはいえない．また，(1-2) の方法は不完全定義関数の asum. 判定は手数が増えること[9]および，結局，正確な D.C 割当てアルゴリズムが必要になるという点で実用的でない．

そこで本稿では次の(2)の方法を用いた．

(2) 逐次割当て法

D.C を 1 つずつ割当て，その都度一応の判定をしておく．

(2)の方法は D.C があっても完全定義関数の判定法を適用していき，もし，不都合が生じたらもとに戻ってやり直す方法で関数決定と判定が混在している．

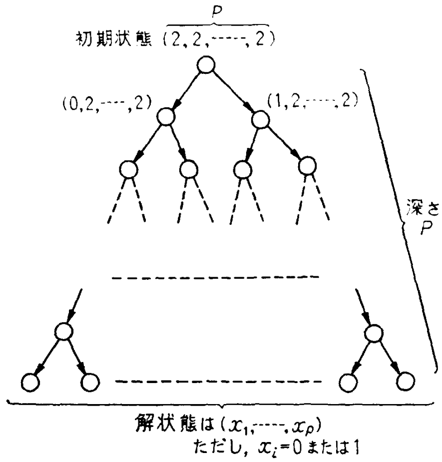

Fig. 2 Birnary tree representation of don't care assignment.

すなわち，0 (または 1) を一つの D.C に割当て完全定義関数に対する k-asum. 判定をする．しきい値ならば先へ進むし，非しきい値ならば逆に 1 (または 0) とする．どちらも割当て不能ならばもとに一ステップ戻ってやり直す．(どのような割当ても不可能なら，はじめの m を $m+1$ にして繰返す．)

この方法を Problem Solving 的見地から考察するために **Fig. 2** ように 2 進木 (binary tree) 表示する．このような状態空間 (state-space) 法[10]の木表現で解を求める代表的方法には Depth-first 法と Breadth-first 法とがあるが，ともかく，しきい値関数を得るような割当て可能解を“一つ”求めることを目的とするならば Depth-first 法が望ましい．(もちろん，すべての解を求めるとするなら Breadth-first 法も考えられる．) この場合，解状態 (goal state) に対応する頂点の深さ (木の根からの距離) はつねに P である．2 進木の各頂点は P 次元ベクトルのラベルをもち，ベクトルの要素は未定 (D.C に相当) のときは 2 としておき，(2-2) の操作の一ステップごとに 2 を 0 (オフセット) または 1 (オンセット) に変える．操作のステップはベクトルの要素を下位から順に行なうとし，2 進木において同じ深さの頂点どおしは辞書式順序で左からならべる．

オペレータは k-asum. 判定である．不完全定義関数の判定法 (k^*-asum. 判定，一般には $k^*>k$) を用いればこれはつねに下向きであるが，この場合は前述のように上向きになることもある．初期状態 (startstate) の頂点のラベルは $(\underbrace{2,2,\cdots,2}_{P})$ であらわし，D.C がすべて割当てられた状態，つまり，解状態の頂点のラベルは $(x_1,\cdots,x_P)$ (ただし，$x_i=0$ または 1) で与えられる．

4. 設計アルゴリズムの大要

設計アルゴリズムの要約は **Fig. 3** のとおりである．これは主プログラムのフローチャートをあらわしているが，k-asum. 判定の k は2としている．ここで手数のかかるのは D.C を割当てるサブルーチンの作成である．実際に作成したサブルーチンの理論的根拠はつぎの命題による．

〔命題 3〕

D.C のあるブール関数 f が D.C 以外の vertex の集合については asum. であるとしたとき，D.C を適当にうめることにより D.C のないしきい値関数 f' を得ることができる．(証明) 文献5) 参照．

D.C の割当ては，オンセット (関数値が1) とオフセット (関数値が 0) の2種が存在するが，どちらを先に試みればより能率的かを知ることはむつかしい．そこで本文のプログラムでは，試行錯誤の順序として

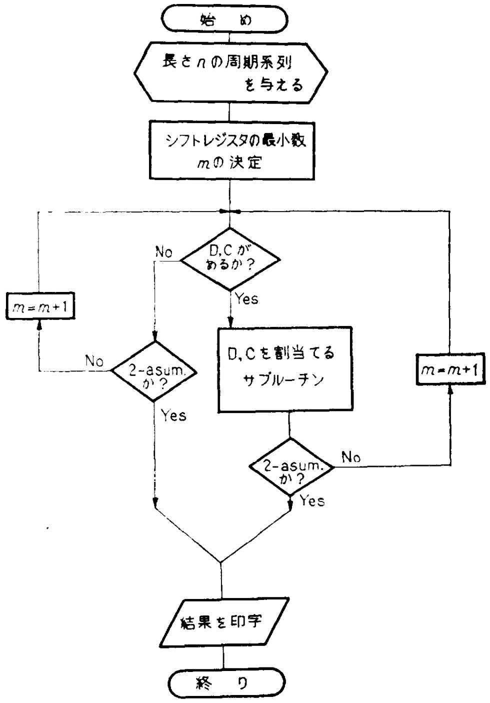

Fig. 3 Flowchart of main program.

オンセット優先，すなわち，まず関数値1を割当てしきい値か否かを判定し，“否”のときに関数値0を割当てる方法をとった*．

Fig. 2 において i 番目 $(i=1,\cdots,P)$ の D.C がすでにオンセットに割当てられているとする．このとき，$i+1$ 番目の D.C がオンセット，オフセットともに割当て不可能であれば i 番目に戻り，i 番目の D.C をオフセットに割当て可能か否かをみる．可能ならばオフセットに割当てて $i+1$ 番目に進む．もし，不可能なら $i-1$ 番目まで戻り同様のことを繰返す．この方法は Fig. 2 の2進木において最右端から解状態を探索することを意味する．本文の設計問題では，周期系列以外の系列には何ら条件をつけていないから，2進木の初期状態から解状態へ至るパスが一つ求まれば設計は完了する．しかし，そのようなパスが一つも存在しないときは，はじめに設定したシフトレジスタの数 m では実現できないから，m を $m+1$ として同じことを繰返す．前述のように周期系列の長さ n 以下の値で必ず設計は完了する．

実際のプログラムを **Fig. 4** (次頁参照) に，結果の例を **Fig. 5** (p. 28 参照) に示す，なお，Fig. 3 のフローチャートでは途中の印字は省略しているが，実際は参考のため印字している．しきい値関数か否かの判定は 2-asum. で行なっているためシフトレジスタの数 m の上限は7である．実際，このプログラムで周期系列の長さ n が 14 までの設計を行なわせたところ (すべてを試みたわけではないが) この上限内のシフトレジスタ数ですべて構成可能となった．なお，一つの周期系列に対する解を求める計算時間は長くとも数分 (TOSBAC 3400) である．

5. 考　察

しきい値関数 f を本文では真理値表として求めた．このことは変数が増すと不都合ではあるが，実際 f をしきい値関数1個で実現しようとするときハードウェア的には，(i) 変数の個数には上限があること，(ii) 回路設計上真理値表が必要となることが多いこと，という理由からこれを採用した．他の表現法 (たとえば特徴パラメータなど[7]) は今のところハードウェア向きではないと思われる．(ii) の理由について補足すると，重み，しきい値で表現したしきい値関数の

* Problem Solving 的にいえば heuristic function が定まれば適当な選択法が示されることになるが今のところむつかしいと思われる．

```
      RUN=1000,LIN=10000
FTC
            COMMON  B(40)
            DIMENSION  A(20),IA(20)
          1 READ (5,100)  (A(I),I=1,20)
            DO 91  IL=1,40
            IF (A(IL).EQ.2.0)  GO  TO  300
         91 CONTINUE
        300 IL=IL-1
            DO  52  I=1,IL
         52 IA(I)=A(I)
            WRITE(6,101)  IL
            WRITE(6,102)  (I,I=1,IL)
            WRITE(6,102)  (IA(I),I=1,IL)
            IF (IL.EQ.1)  GO TO 299
            GO TO 298
        299 WRITE(6,103)
            GO TO 1
        298 DO 53  J=1,7
            JQ=2**J
            IF (IL.LE.JQ)  GO TO 301
         53 CONTINUE
        301 DO  54  I=1,IL
         54 B(I)=A(I)
            DO  55  I=1,J
            IO=IL+I
         55 B(IO)=A(I)
            CALL  DIT(J,IL,$1)
            GO TO 301
        100 FORMAT(20F4.0)
        101 FORMAT(1H1,'***  GIVEN PERIODIC SEQUENCE  ***',5X,'* THE LENGTH
           *OF THE GIVEN SEQUENCE *',I3)
        102 FORMAT(1H0,40I3)
        103 FORMAT(1H0,5X,'* POSSIBLE *',5X,'0-VARIABLE (WITHOUT DELAY-',
           *'ELEMENT)')
            END
FTC
            SUBROUTINE  DIT(J,IL,*)
            COMMON  B(40)
            DIMENSION IC(130,7),ID(130,7),ICQ(5000),IDQ(5000),NUM(50),ILI(20),
           *ICD(130,7),ICR(130),IDR(130),IAB(130),MM(130)
            JJ1=0
            DO 56 II=1,IL
            IF(B(II).EQ.1.0)  GO TO 303
            GO TO 56
        303 JJ1=JJ1+1
            DO 57 I=1,J
            KK=II+I
         57 IC(JJ1,I)=B(KK)
         56 CONTINUE
            JJ2=0
            DO 58 II=1,IL
            IF(B(II).EQ.0.0)  GO TO 304
            GO TO 58
        304 JJ2=JJ2+1
            DO 59 I=1,J
            KK=II+I
         59 ID(JJ2,I)=B(KK)
         58 CONTINUE
            KC=1
            KD=1
            DO 70 KK=1,JJ1
            ICR(KK)=0
            DO 70 K=1,J
         70 ICR(KK)=ICR(KK)*3+IC(KK,K)
            DO 71 KK=1,JJ2
            IDR(KK)=0
            DO 71 K=1,J
         71 IDR(KK)=IDR(KK)*3+ID(KK,K)
            DO 72 I=1,JJ1
            DO 72 KK=I,JJ1
            ICQ(KC)=ICR(I)+ICR(KK)
         72 KC=KC+1
            DO 73 I=1,JJ2
            DO 73 KK=I,JJ2
            IDQ(KD)=IDR(I)+IDR(KK)
         73 KD=KD+1
            JT1=KC-1
            JT2=KD-1
            DO 75 KK=1,JT1
            DO 75 K=1,JT2
            IF(ICQ(KK).EQ.IDQ(K))  GO TO  987
         75 CONTINUE
            I1=2
            WRITE(6,104)
            GO TO 320
        987 IF(J.LE.4) GO TO 306
            GO TO 307
        306 I1=1
            WRITE(6,200)
            GO TO 320
        307 I1=0
            WRITE(6,200)
            GO TO  320
        320 WRITE(6,105)
            KI=1
            IK=MINO(JJ1,JJ2)
         91 IF(KI.NE.1)  GO TO 333
            LO=1
            LH=IK
            GO TO 93
        333 IF(JJ1-JJ2)  310,95,60
         60 LO=JJ2+1
            LH=JJ1
            GO TO 92
        310 LO=JJ1+1
            LH=JJ2
            GO TO 94
         93 GO TO (31,32,33,34,35,36,37),J
         31 WRITE(6,106) (((IC(I,L),L=1,J),(ID(I,M),M=1,J)),I=LO,LH)
            GO TO 99
         32 WRITE(6,107) (((IC(I,L),L=1,J),(ID(I,M),M=1,J)),I=LO,LH)
            GO TO 99
         33 WRITE(6,108) (((IC(I,L),L=1,J),(ID(I,M),M=1,J)),I=LO,LH)
            GO TO 99
         34 WRITE(6,109) (((IC(I,L),L=1,J),(ID(I,M),M=1,J)),I=LO,LH)
            GO TO 99
         35 WRITE(6,110) (((IC(I,L),L=1,J),(ID(I,M),M=1,J)),I=LO,LH)
            GO TO 99
         36 WRITE(6,111) (((IC(I,L),L=1,J),(ID(I,M),M=1,J)),I=LO,LH)
            GO TO 99
         37 WRITE(6,112) (((IC(I,L),L=1,J),(ID(I,M),M=1,J)),I=LO,LH)
            GO TO 99
         92 DO 931 I=LO,LH
        931 WRITE(6,116)  (IC(I,L),L=1,J)
            GO TO 99
         94 DO 831 I=LO,LH
        831 WRITE(6,126)  (ID(I,L),L=1,J)
         99 IF(KI.EQ.2)  GO TO 95
            KI=KI+1
            GO TO 91
          2 DO 400 KK=1,JJ1
            NUM(KK)=0
            DO 400 K=1,J
        400 NUM(KK)=NUM(KK)*2+IC(KK,K)
            DO 401 II=1,JJ2
            IKK=JJ1+II
            NUM(IKK)=0
            DO 401 I=1,J
        401 NUM(IKK)=NUM(IKK)*2+ID(II,I)
            M=0
            IT=0
            IJJ=2**J
            JIT=JJ1+JJ2
            DO 402 KK=1,IJJ
            K=KK-1
            DO 403 I=1,JIT
            IF(K.EQ.NUM(I))  GO TO 402
        403 CONTINUE
            IT=IT+1
            M=M+1
            ILI(1)=K
            DO 404 NN=1,J
            MN=J-NN+1
            ILI(NN+1)=ILI(NN)/2
            ICD(M,MN)=ILI(NN)-2*ILI(NN+1)
        404 CONTINUE
        402 CONTINUE
            DO 25 KK=1,IT
            IAB(KK)=0
            DO 25 K=1,J
         25 IAB(KK)=IAB(KK)*3+ICD(KK,K)
            WRITE(6,201)  (NUM(I),I=1,JIT)
            II1=JJ1*(JJ1+1)/2
            II2=JJ2*(JJ2+1)/2
            IKC=II1
            IKD=II2
            M=0
            I=1
          4 DO 406 KK=1,JJ1
            IKC=IKC+1
        406 ICQ(IKC)=IAB(I)+ICR(KK)
            IKC=IKC+1
            ICQ(IKC)=IAB(I)*2
            M=M+1
            IP=IKC-JJ1
            DO 407 KK=IP,IKC
            DO 407 K=1,IKD
            IF(ICQ(KK).EQ.IDQ(K))  GO TO  3
        407 CONTINUE
            JJ1=JJ1+1
            ICR(JJ1)=IAB(I)
            DO 11 KK=1,J
         11 IC(JJ1,KK)=ICD(I,KK)
            MM(I)=1
            IF(I.EQ.IT) GO TO 21
            I=I+1
            GO TO 4
          3 IKC=IP-1
          8 DO 408 KK=1,JJ2
            IKD=IKD+1
        408 IDQ(IKD)=IAB(I)+IDR(KK)
            IKD=IKD+1
            IDQ(IKD)=IAB(I)*2
            IR=IKD-JJ2
            DO 409 KK=IR,IKD
            DO 409 K=1,IKC
            IF(IDQ(KK).EQ.ICQ(K))  GO TO 5
        409 CONTINUE
            JJ2=JJ2+1
            IDR(JJ2)=IAB(I)
            DO 12 KK=1,J
         12 ID(JJ2,KK)=ICD(I,KK)
            MM(I)=0
            IF(I.EQ.IT)  GO TO 21
            I=I+1
            GO TO 4
          5 IF(M.EQ.1)  GO TO 97777
            IF(I.LE.2) GO TO 9
          6 I=I-1
            IF(I.EQ.1)  GO TO 7
            IF(MM(I).EQ.1) GO TO 23
            IKD=IR-JJ2-1
            JJ2=JJ2-1
            GO TO 6
         23 IKC=IKC-JJ1
            JJ1=JJ1-1
            IKD=IR-1
            GO TO 8
          9 IKD=IR-1
          7 IF(MM(1).EQ.0)  GO TO 97777
            I=1
            MM(1)=0
            JJ1=JJ1-1
            GO TO 8
         21 I1=3
            GO TO 320
       2222 WRITE(6,203)  (MM(I),I=1,IT)
            GO TO 91111
         95 IF(I1.EQ.2)  GO TO 2
            IF(I1.EQ.1)  GO TO 97777
            IF(I1.EQ.0)  GO TO 91111
            IF(I1.EQ.3) GO TO 2222
      97777 J=J+1
            IF (J.GT.7)  GO TO 2111
            IF(J.GT.IL)  GO TO 2111
            GO TO 11111
       2111 WRITE(6,202)
            GO TO 91111
      11111 RETURN
      91111 RETURN1
        104 FORMAT(1H0,5X,'* POSSIBLE *',2X,'*  THE NUMBER OF THE DELAY-',
           *ELEMENTS *',I2)
        105 FORMAT(1H0,T23,'**  ON-SET  **',T53,'** OFF-SET  **')
        106 FORMAT(1H ,T26, I2,T55, I2)
        107 FORMAT(1H ,T26,2I2,T55,2I2)
        108 FORMAT(1H ,T26,3I2,T55,3I2)
        109 FORMAT(1H ,T26,4I2,T55,4I2)
        110 FORMAT(1H ,T26,5I2,T55,5I2)
        111 FORMAT(1H ,T26,6I2,T55,6I2)
        112 FORMAT(1H ,T26,7I2,T55,7I2)
        116 FORMAT(1H ,T26,7I2)
        126 FORMAT(1H ,T55,7I2)
        200 FORMAT(1H0,5X,'* IMPOSSIBLE *','*  THE NUMBER OF THE DELAY-','
           *ELEMENTS *',I2)
        201 FORMAT(1H0,5X,20I5)
        202 FORMAT(1H0,5X,'*  THE ASSIGNMENT OF DON''T CARE IS IMPOSSIBLE  *')
        203 FORMAT(1H0,2X,130I1)
            END
```

Fig. 4 Program for design

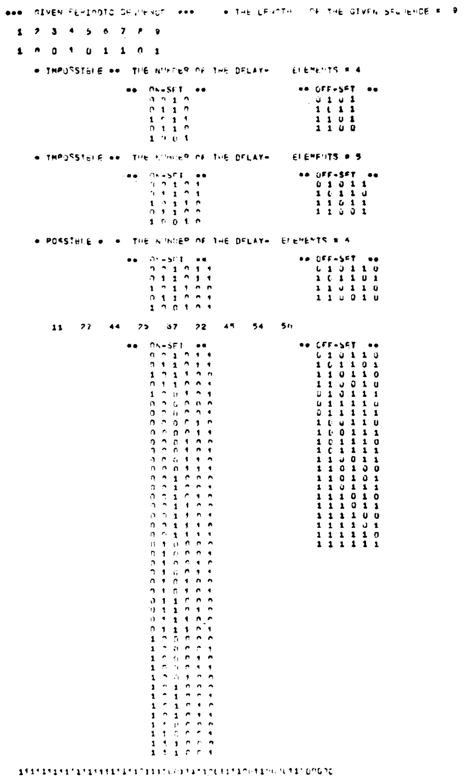

Fig. 5 An example of results

定義式からハードウェアにおける回路定数まで換算する一般的方法はまだ与えられていない．実際，RTL，DTL (TTL) などを用いしきい値素子をつくるとき，論理動作の調整は，抵抗変化や付加ダイオード数の変化で行なうことになり，結局真理値が必要となる．

本文の方法は，具体的にプログラムする段階では $m \leqq 7$ とした．もし，$m \geqq 8$ となると 2-asum. 判定が使えなくなり，サブルーチンを k-asum. 判定に作りかえる必要がある．このとき，設計基本方針には何ら変化はないけれども，実際の手数は極端に増し，この問題は k-asum. 判定におけるより正確な k を求めるというしきい値固有の問題[9] に還元される．

6. むすび

Fig. 1 の構成において関数 f が一般のブール関数とするならば，周期系列の長さ n に対してシフトレジスタの数 m はつねに $]\log_2 n[$ ($]x[$ は x に等しいか，x より大の最小の整数を表わす）とする構成がつねに可能である．しかしながら f の実際の実現はつねに簡単にできるとは言えない．

一方，ガロア体上の演算，すなわち ring sum 演算を用いる設計法は統一的になってきているが[6]，これまた実際の f の実現は必ずしも容易であるとはいえない．

これらのことを考慮すれば f をしきい値関数として実現する方法は一つの実用さを有しているといえよう．

なお，f を ring sum 演算とする方法と f をしきい値関数とする方法を比較したとき，必要とするシフトレジスタの数は両者どちらが有利であるとはいえない．すなわち，あるときは前者の方が少なく，あるときは後者の方が少ない．f をしきい値関数とする順序回路の動作は，オートノマス動作といえども特徴化はほとんどなされていないため[11]前者と長短を論じることはまだ早計といえよう．

最後に，日頃よりご鞭撻いただく大阪大学田中幸吉教授，北橋忠宏博士，広島大学吉田典可教授，多大な協力をいただいた大阪大学永見均氏，プログラム作成に当たってくれた極東マック吉村正明氏に深謝する．なお，使用した計算機は TOSBAC 3400（広島大学計算センター）である．

参 考 文 献

1) 甘利：自己組識しきい素子回路によるパターンの学習，信学論 (D), Vol. 55-D, No. 7, pp. 456～463 (1972)
2) S. Yajima, T. Ibaraki and I. Kawano: On Autonomous Logic Nets of Threshold Elements, IEEE Trans. Vol. EC-17, p. 385 (1968)
3) 石井，木村：与えられた状態列を実現するしきい素子回路網のテスト・シンセシス，信学論 (C), Vol. 51-C, No. 6, pp. 259～266 (1968)
4) 石井，宮崎：しきい素子回路網で実現できるサイクル集合の生成，信学論 (D), Vol. 55-D, No. 8, pp. 491～498 (1972)
5) 永見，阿江，吉田：任意周期系列のしきい値素子回路網による一実現，信学会オートマトンと言語研資 AL 72-71 (1972)
6) A. Gill: Linear Sequential Circuits-Analysis, Synthesis and Applications, McGraw-Hill Inc. (1966)
7) S. T. Hu: Threshold Logic, University of California Press (1965)
8) 福村：最近のスィッチング回路網理論（その 3）しきい値論理，信学誌 Vol. 53, No. 9, pp. 1276～1284 (1970)

9) 佐藤，北橋，田中：しきい値関数判定に関する一考察，信学論 (C). Vol. 54-C, No. 9, pp. 789～795 (1971)

10) N. J. Nilsson: Problem-Solving Methods in Artificial Intelligence, McGraw-Hill (1971)

11) 阿江，永見：しきい値素子を用いるシフトレジスタ形順序回路，信学会オートマトンと言語研資 AL 73-38 (1973)

(昭和49年1月8日受付)
(昭和49年5月31日再受付)