論　文

順序回路の検査パターン作成の発見的一手法*

壺屋光邦** 天宮伍郎**
有馬俊弘** 奥田二郎**

Abstract

With the advent of LSI, the problem of test for LSI and IC cards increasingly becomes difficult. To cope with the difficulty, there are two approaches: one is to design easy-testable circuits, the other is to develop powerful algorithms for test data generation of logical circuits.

As to the test data generation algorithm, it seems through our study that the conventional algorithms require more complicated program structure and more computer time to process relatively large circuits with more than 500 logic blocks.

This papers concerned with the new heuristic test data generation algorithm, called M0M1 algorithm. This algorithm is theoretically based on special type of values expressed in Boolean vectors, their logical operations, and several basic theorems. In implementation, the algorithm Enjoys the advantage that the main process is simple logical operations of Boolean vectors and backward search on the state table without tedious table operations.

1. まえがき

近年コンピュータなどのデジタル回路は複雑化し，その故障診断に関する技術の向上が期待されている．本論文は，パッケージ部品の故障診断に関するもので，その検査データを自動作成する新しい手法を紹介する．

従来のこの種の手法には，Dアルゴリズム[1),4)] やブール階差法などがある．前者は発見的手法***であり，後者は解析的手法であるが，数百ゲート以上の理論回路の検査データの作成には，解析的手法はほとんど役に立たない．現在Dアルゴリズムは広く一般に使われている手法であり，その改良も多く発表されている．

本論文で紹介する手法は，*M0M1* アルゴリズムと名付けており，カット手法とブールベクトル演算の2つの概念から成立っている．以下この2つの概念の説明をし，その後で本アルゴリズムの手順の概略を示す．最後にこれを実際に使用している検査システムでの運用結果を示す．

2. 故障診断とは

理論回路が故障している場合，ある理論素子の出力信号線または入力信号線が固定的に理論値0または1になっていることが多い．このような故障を**0縮退障害** (s-a-0)，**1縮退障害** (s-a-1) という．この他にも間欠的な故障を起こすこともあるが，一般には，検査データの作成で対象とする故障は縮退障害だけである．

故障のない回路を M_0 とし，その回路中に縮退障害をもった回路を $M_1, M_2, \cdots\cdots M_i, \cdots$ とする．回路 M_i の入力パターンを T としたときの M_i の出力パターンを $M_i(T)$ とする．このとき $M_0(T) \neq M_i(T)$ であれば M_0 と M_i を識別できる．このような T を，M_i の**検査パターン**という．T を入力変数で表わし，$M_i(T)$ を理論式で表現したとしよう．このとき

$$M_0(T) \oplus M_i(T) = 1 \quad (\oplus \text{は排他的論理和})$$

---

* A New Heuristic Test Sequence Generation Algorithm For Sequential Circuits, by Mitsukuni TSUBOYA Goro AMAMIYA Toshihiro ARIMA Jiro OKUDA (Engineering Department, Electronic Switching Div., Nippon Electric Co., Ltd.)

** 日本電気(株)電子交換事業部設計技術部

***ここでいう発見的手法とは，試行錯誤的という意味ではなく，手順によって解が異なるという意味である．

なる論理方程式の解は，M_i の検査パターンである．つまり多変数からなる論理方程式を，短い処理時間で一般的に解く方法があれば，検査パターンの作成は簡単であるが，そのような方法は見つかっていない．

3. 論理式のグラフ表現

論理式のグラフ表現について説明する．

論理式 $y=\overline{(\bar{x}_1+x_2)\cdot(x_1+\bar{x}_2)+x_1\cdot x_2}$ を回路表現したものを **Fig. 1** に示す．これを **Fig. 2** のように，論理積(・)を直列な枝で，論理和(+)を並列な枝で示し，グラフ表現することができる[3]．

このグラフ上の各枝 $e_1, e_2, \cdots e_6$ は回路図で，出力 y からバックトレースして求めることができる．たとえば Fig. 1 で，$y \to g_5 \to \bar{g}_4 \to \bar{x}_2$ というようにバックトレースすることにより，枝 e_5 に対応する $\bar{x}_2$ が求まる．

さて Fig. 2 のグラフ上では，1つの連鎖は論理式 y を加法展開したときの1つの項を表現している．たとえば枝 $e_1e_4e_5$ からなる連鎖は，$\bar{x}_1x_3\bar{x}_2$ なる項を表現している．したがって $\bar{x}_1x_3\bar{x}_2=1$ は $y=1$ なる方程式の1つの解である．しかしグラフ上の任意の連鎖が全て $y=1$ の解であるわけではない．たとえば連鎖 $e_1e_3e_5$ からは，$\bar{x}_1x_1\bar{x}_2$ なる項が得られるが，これは論理的に0であり $y=1$ の解ではない．このような連鎖は**矛盾をもつ**という．

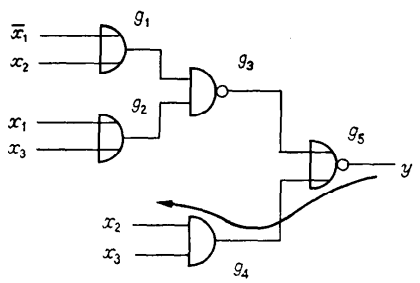

Fig. 1 The circuit model of logic

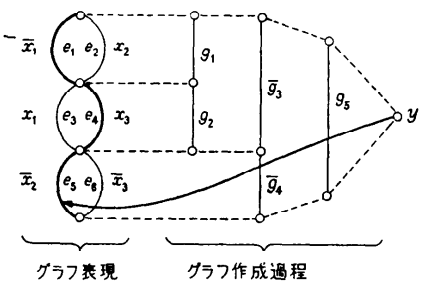

Fig. 2 The graph model of logic

4. カット手法

論理方程式の解を求めるには，論理式をグラフ表現し，そこから矛盾のない連鎖を見つければよい．これは**カット手法**という方法で行うことができる．

カット手法を説明するために，**Fig. 3** のグラフ(a)から $y=1$ なる方程式の解を求めてみよう．

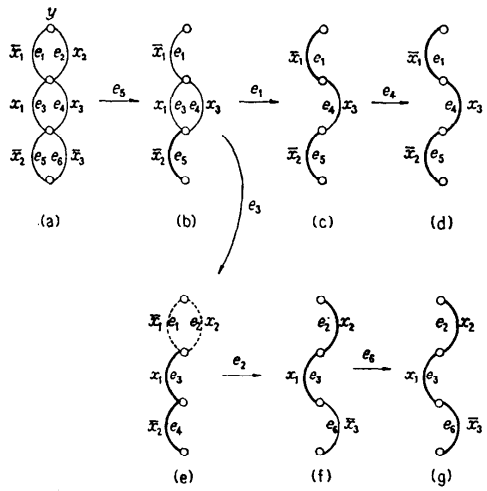

Fig. 3 The cutting method

(a)で枝 e_5 を選び，$\bar{x}_2=1$ として(b)を得る．

(b)で枝 e_1 を選び，$\bar{x}_1=1$ として(c)を得る．

(c)で枝 e_4 を選び，$x_3=1$ として(d)を得る．

このようにして矛盾のない連鎖 $e_1e_4e_5$ が得られる．このときの解は，$x_1=0$, $x_2=0$, $x_3=1$ である．

上の手順において，(b)で枝 e_3 を選び，$x_1=1$ としたとすると，(e)のように連鎖が途切れてしまう結果となる．この場合は途切れた枝の1つを復元してやる．たとえば(e)で枝 e_2 を復元すると，$x_2=1$ から(f)が得られる．さらに(f)から(g)を作れば矛盾のない連鎖 $e_2e_3e_6$ が得られる．このときの解は $x_1=1$, $x_2=1$, $x_3=0$ である．

以上がカット手法の概略である．

5. 論理空間の写像

われわれは，グラフ表現せずに論理回路表現を使って，論理方程式を解く方法を考えた．**Fig. 4** (次頁参照) のように，3値論理空間 L を 13 値論理空間 S で表現し，これに対して G **変換**，B **変換**という操作を行う．この操作はグラフのカット手法と手順が同じで

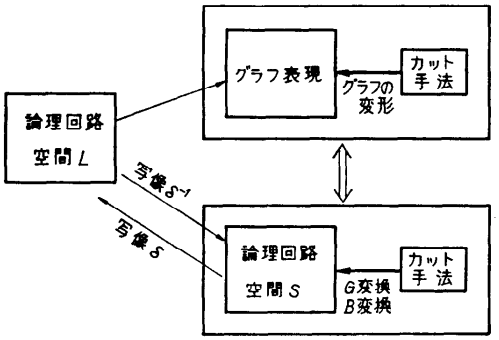

Fig. 4 Equivalent conversion of the cutting method

ある．最後に空間Sを空間Lに戻せば解が得られる．

以下，空間L，空間S，写像 δ，G変換，B変換について説明する．

6. 写像と変換の定義

通常の論理演算では2値{0, 1}だけであるが，これに don't care xを追加して $L=\{0, 1, x\}$とした**空間 L** は一般的であり，説明を要しないであろう．われわれは，空間Lに対して新しい**空間 V**を考え，**写像 δ** 及び**G変換**，**B変換**を **Fig. 5** のように定義した．ただし，$V=\{0, 1, 2, 3, 4, 5\}$．空間Lの変数 x_i と，空間Vの変数 $\dot{x}_i$ に対し，これらの関係を写像 δ を使って次のように表わすことにする．

$\delta\dot{x}_i=x_i$ または $\delta^{-1}x_i=\dot{x}_i$.

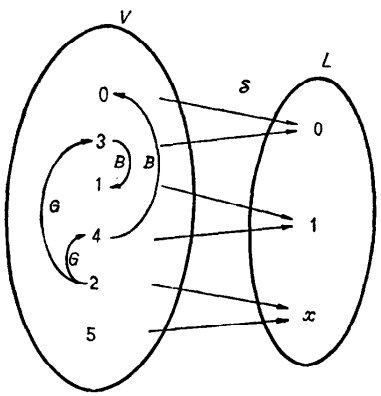

Fig. 5 Image of G-change, B-change and δ-mapping

たとえば，$\delta 0=0$，$\delta 3=0$，$\delta^{-1}0\in\{0, 3\}$ である．また，$\dot{x}_i$ のG変換を $G\dot{x}_i$，$\dot{x}_i$ のB変換を $B\dot{x}_i$ と書くことにすれば，$G2\in\{3, 4\}$，$B3=1$，$B4=0$ である．これ以外の変換は定義していないことに注意．以上の定義から空間Lに対する変換は，空間Vに対する変換と写像で表わせる．たとえば $x_i\in L$ で $x_i=0$ を $x_i=1$ に変換する場合は次のように表わせる．

$\delta(B(\delta^{-1}x_i))=\delta(B(\delta^{-1}0))=1$.

次に，空間 V に対しても空間 L と似た論理演算を決めると便利である．なぜならば，$y_1=x_1+x_2$ において $\dot{x}_1$ と $\dot{x}_2$ が既知であるなら，$\dot{y}_1=\delta^{-1}(\delta\dot{x}_1+\delta\dot{x}_2)$ から $\dot{y}_1$ が求まるが，$\dot{y}_1=\dot{x}_1+\dot{x}_2$ を定義しておけば，もっと簡単に $\dot{y}$ 知ることができるからである．

7. ブールベクトル演算

一般に 0, 1 を成分とするベクトル，たとえば

(011), (101), (0111)

などをブールベクトルという．その成分の個数をそのベクトルの次元という．われわれは6次元のブールベクトルを考え，次のような**ブールベクトル演算**を定義した

a_i, $b_i\in\{0, 1\}$,

$A=(a_1a_2a_3a_4a_5a_6)$, $B=(b_1b_2b_3b_4b_5b_6)$

としたとき，

$A+B=(a_1+b_1\cdots\cdots a_6+b_6)$

$A\cdot B=(a_1\cdot b_1\cdots\cdots a_6\cdot b_6)$

$\bar{A}=(\bar{a}_4\bar{a}_5\bar{a}_6\bar{a}_1\bar{a}_2\bar{a}_3)$*

とする．(この定義はドモルガンの定理 $\overline{A+B}=\bar{A}\cdot\bar{B}$ を満足している)．

前節で述べた集合Vを，次のような6次元ブールベクトルに対応させることにする．

0=(000000)

1=(111111)

2=(111000)

3=(100000)

4=(111011)

5=(010101)

これでVに対する論理演算が定義できた．たとえば，

2+4=(111000)+(111011)

=(111011)=4

ところがこの演算の定義から，集合Vに属さない新しい要素が派生してくる．たとえば

$4\cdot 5=(010001)\notin V$

このような派生要素を全て包含し，ブールベクトル演算において閉じた**集合 S** を新しく定義する．

$S=\{0, 1, 2, \cdots\cdots, 12\}$

ただし，

* ベクトル成分の順序に注意．

6=(0 1 0 0 0 1)
7=(1 1 0 1 0 1)
8=(0 1 0 0 0 0)
9=(1 1 1 1 0 1)
10=(1 1 0 0 0 0)
11=(1 1 1 0 0 1)
12=(1 1 0 0 0 1)

集合 S の要素に対する論理演算を **Table 1** に示しておく．論理和演算は示していないが，ドモルガンの定理からすぐに求まる．たとえば，

$4+9=\overline{\bar{4}\cdot\bar{9}}=\overline{3\cdot 8}=\bar{0}=1.$

•	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	10	11	12
2	0	2	2	3	2	8	8	10	8	2	10	2	10
3	0	3	3	3	3	0	0	3	0	3	3	3	3
4	0	4	2	3	4	6	6	12	8	11	10	11	12
5	0	5	8	0	6	5	6	5	8	5	8	6	6
6	0	6	8	0	6	6	6	6	8	6	8	6	6
7	0	7	10	3	12	5	6	7	8	7	10	12	12
8	0	8	8	0	8	8	8	8	8	8	8	8	8
9	0	9	2	3	11	5	6	7	8	9	10	11	12
10	0	10	10	3	10	8	8	10	8	10	10	10	10
11	0	11	2	3	11	6	6	12	8	11	10	11	12
12	0	12	10	3	12	6	6	12	8	12	10	12	12

−	0	1	2	3	4	5	6	7	8	9	10	11	12
	1	0	2	4	3	5	7	6	9	8	11	10	12

Table 1 Logic operation: AND (•), NOT (−)

8. 写像と変換の定義拡張

先に空間 L と空間 V に関する写像 δ と G 変換，B 変換を定義したが，空間 V を**空間** S に拡張して新しく定義をしなおすことにする．

まず集合 S を次のように分割する．

$(S=S_1\cup S_2\cup S_3\cup S_4,\ S_1\cap S_2=\phi,\cdots\cdots)$

$S_1=\{1,\ 4\}$
$S_2=\{2,\ 9,\ 11\}$
$S_3=\{3,\ 7,\ 10,\ 12\}$
$S_4=\{0,\ 5,\ 6,\ 8\}$

これらの部分集合の論理演算 ($\bar{S}_1$, S_1+S_2 など) を，その部分集合の要素の論理演算として定義する．

* このように S_i は $\bar{S}_i$ の補集合ではないことに注意．

たとえば，$\bar{S}_1{}^*=\{\bar{1},\ \bar{4}\}=\{0,\ 3\}$.

このとき **Table 2** の論理演算が成立する．

•	S_1	S_2	S_3	S_4
S_1	S_1	S_2	S_3	S_4
S_2	S_2	S_2	S_3	S_4
S_3	S_3	S_3	S_3	S_4
S_4	S_4	S_4	S_4	S_4

+	S_1	S_2	S_3	S_4
S_1	S_1	S_1	S_1	S_1
S_2	S_1	S_2	S_2	S_2
S_3	S_1	S_2	S_3	S_3
S_4	S_1	S_2	S_3	S_4

Table 2 Logic operation: $S_i\cdot S_j$, S_i+S_j

われわれは，Table 2 の結果から次のような定義を行った．

写像 $\delta S=L$ の定義: $\delta S_1=1\in L$, $\delta\bar{S}_1=0\in L$, それ以外は $x\in L$ とする．

G 変換の定義: $GS_2=4\in S_1$, $G\bar{S}_2=3\in\bar{S}_1$.

B 変換の定義: $BS_3=1\in S_1$, $B\bar{S}_3=0\in\bar{S}_1$.

以上の定義は，第7章の定義を全て包含している．

Fig. 6 に S の部分集合のイメージを表わしておく．

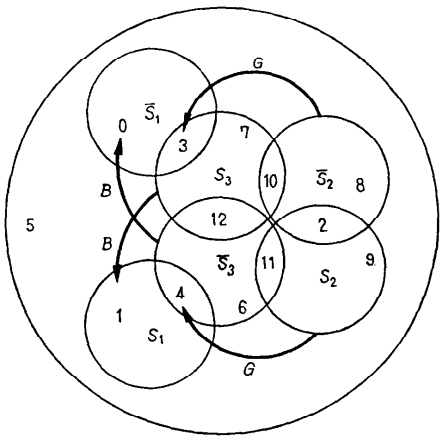

Fig. 6 Definition of G-Change, B-change

9. カット手法の変形

空間 S における G 変換，B 変換を行う手法を，簡単な例を使って説明する．これは先に述べたカット手法の変形である．

Fig. 7 (次頁参照) の順序回路で，$y=1$ なる方程式の解を求めてみよう．ただし次の条件があるものとする．

$x_1=1$, $x_2=x$, $x_3=x$ ただし，x_1 の値は強制指定であり，x_3 の初期値は不明であるとする．

ステップ 1: 与えられた条件から，L 空間を S 空間に

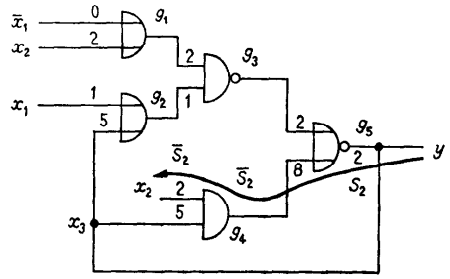

Fig. 7 The sequential circuit

写像すると，$\dot{x}_1=1$, $\dot{x}_2=2$, $\dot{x}_3=5$ となる．

次に S 空間の論理演算を行う．

$\dot{g}_1=2$, $\dot{g}_2=1$, $\dot{g}_3=2$, $\dot{g}_4=8$, $\dot{g}_5=\dot{y}=2$.

ステップ 2: $\dot{y}\in S_2$ だから y の G 変換を考える．$\dot{g}_5\in S_2$, $\bar{\dot{g}}_4\in S_2$, $\bar{\dot{x}}_2\in S_2$ であるので，$y\to g_5\to\bar{g}_4\to\bar{x}_2$ というようにバックトレースする．これは先に述べたように，Fig. 2 のグラフの枝 e_5 を見つけたことに相当する．$\dot{x}_2\in\bar{S}_2$ だから，$\dot{x}_2=G\,2=3$ という G 変換を行う．

$\dot{g}_1=3$, $\dot{g}_2=1$, $\dot{g}_3=4$, $\dot{g}_4=0$, $\dot{g}_5=\dot{y}=3$.

ステップ 3: $\dot{y}\in S_3$ だから y の B 変換を考える．$\dot{g}_5\in S_3$, $\bar{\dot{g}}_3\in S_3$, $\dot{g}_1\in S_3$, $\dot{x}_2\in S_3$ であるので，$y\to g_5\to g_3\to g_1\to x_2$ というようにバックトレースする．$\dot{x}_2\in S_3$ だから，$\dot{x}_2=B\,3=1$ という B 変換を行う．

$\dot{g}_1=1$, $\dot{g}_2=1$, $\dot{g}_3=0$, $\dot{g}_4=5$, $\dot{g}_5=\dot{y}=5$.

ステップ 4: $\dot{y}\in S_4$ だから，もはや $\dot{y}$ に対して G 変換も B 変換も不可能である．

以上はフィードバック信号線 $(y,\ x_3)$ を考慮しなかったが，次にこれを考えてみる．ステップ 2 の結果から $\dot{y}=3$ つまり $y=0$ を得ている．したがって x_3 の値はこれを引き継ぐことになる．そこで

$$x_1=1,\ x_2=x,\ x_3=0$$

という条件を設定しなおして手順を続行する．

ステップ 5: $\dot{x}_1=1$, $\dot{x}_2=2$, $\dot{x}_3=0$.

$\dot{g}_1=2$, $\dot{g}_2=1$, $\dot{g}_3=2$, $\dot{g}_4=0$, $\dot{g}_5=\dot{y}=2$.

ステップ 6: $\dot{y}\in S_2$ だから y の G 変換を考える．$\dot{g}_5\in S_2$, $\bar{\dot{g}}_3\in S_2$, $\dot{g}_1\in S_2$, $\dot{x}_2\in S_2$ であり，$y\to g_5\to\bar{g}_3\to g_1\to x_2$ とトレースする．$\dot{x}_2\in S_2$ だから，$\dot{x}_2$ を G 変換して $\dot{x}_2=G\,2=4$. $\dot{g}_1=4$, $\dot{g}_2=1$, $\dot{g}_3=3$, $\dot{g}_4=0$, $\dot{g}_5=\dot{y}=4$.

ステップ 7: $\dot{y}=4$ つまり $y=\delta\,4=1$ であり，解が求まった．

以上から，Fig. 7 の順序回路に次の 2 パターンを続けて入力することにより，出力を 1 にすることができる．

第 1 パターン：$x_1=1$, $x_2=0$ （ステップ 2 から）

第 2 パターン：$x_1=1$, $x_2=1$ （ステップ 6 から）

10. 故障診断への応用

前章では，ブールベクトル演算とカット手法を使って，順序回路の初期セットシーケンスを作り出せることを示した．次に故障診断への応用を考える．

故障のない回路を M_0 とし，故障のある回路を M_1 とする．回路中の素子 a の状態を，M_0 において a_0, M_1 において a_1 であるとし，a_0/a_1 と記述することにする．同様に $\dot{a}_0$, $\dot{a}_1\in S$ に対して，$\dot{a}_0/\dot{a}_1$ と記述し，

$$\delta(\dot{a}_0/\dot{a}_1)=\delta\dot{a}_0/\delta\dot{a}_1=a_0/a_1$$

とする．たとえば，$\delta\,4/3=\delta\,4/\delta\,3=1/0$. $\dot{a}_0/\dot{a}_1$ は集合 S の直積 S^2 の要素である．集合 S^2 の部分集合 $D,\bar{D}$ を次のように定義しておく．

$$D=S_1/\bar{S}_1,\quad \bar{D}=\bar{S}_1/S_1.$$

このとき，$\delta D=\delta S_1/\delta\bar{S}_1=1/0$.

$$\delta\bar{D}=\delta\bar{S}_1/\delta S_1=0/1.$$

つまり $\dot{a}_0/\dot{a}_1\in D$ または $\bar{D}$ ならば，$a_0/a_1=1/0$ または $0/1$ となる．これは素子 a において，正常回路 M_0 と故障回路 M_1 とが識別できることを意味する．

今 M_0 と M_1 が与えられたとき，G 変換及び B 変換を行うことによって，M_1 の検査パターンを求めることができる．以下その手法である $M0M1$ アルゴリズムを紹介する．M_0, M_1 は組合せ回路とし，M_1 では素子 f の出力が縮退 0 障害である，と仮定したときの手順を次に示す．ただし素子 f の状態を $\dot{f}_0/\dot{f}_1$ として表わす．

ステップ 1: M_1 に故障を注入する．つまり $\dot{f}_1=0$.

ステップ 2: $\dot{f}_0$ の値が $S=S_1\cup S_2\cup S_3\cup S_4$ のいずれかを判定する．$\dot{f}_0\in S_1$ のとき，$\dot{f}_0/\dot{f}_1\in D$ であるからステップ 3 へ行く．$\dot{f}_0\in S_2$ のとき，回路 M_0 において $G\dot{f}_0=4$ となるように G 変換を行う．$\dot{f}_0\in S_3$ のとき，回路 M_0 において $B\dot{f}_0=1$ となるように B 変換を行う．$\dot{f}_0\in S_4$ のとき，検査パターンは存在しない*.

ステップ 2 は，$\dot{f}_0\in S_1$ または S_4 になるまで繰返す．

ステップ 3: **Fig. 8** （次頁参照）のように，入力に D または $\bar{D}$ をもつゲートに着目する．Fig. 8 は OR ゲート，AND ゲートしか示していないが，他のゲートも同様である．Fig. 8 の a）の場合，$D+\dot{a}_0/\dot{a}_1=S_1/\bar{S}_1+\dot{a}_0/\dot{a}_1=(S_1+\dot{a}_0)/(\bar{S}_1+\dot{a}_1)=S_1/\dot{a}_1$ である．

* 実はこの手順では正しくない．これについては附録で述べる

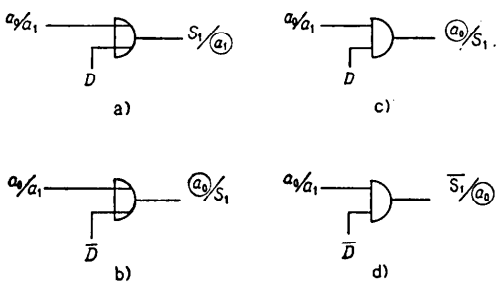

Fig. 8 D-drive operation

そこで $\dot{a}_1$ の値が $S=\bar{S}_1 \cup \bar{S}_2 \cup \bar{S}_3 \cup \bar{S}_4$ のいずれかを判定する．$\dot{a}_1 \in \bar{S}_1$ のとき，$\dot{a}_0/\dot{a}_1 \in D$ であるから他のゲートを調べる．$\dot{a}_1 \in \bar{S}_2$ のとき，回路 M_1 において $G\dot{a}_1=3$ となるように G 変換を行う．$\dot{a}_1 \in \bar{S}_3$ のとき，回路 M_1 において $B\dot{a}_1=0$ となるように B 変換を行う．$\dot{a}_1 \in \bar{S}_4$ のとき，他のゲートを調べる．同様にして Fig. 8 の b)，c)，d) の場合も判定できる．ステップ3は，回路の出力端子に D または $\bar{D}$ が出現するまで繰返す．もし全てのゲートについて調べても出力端子に D または $\bar{D}$ が出現しなければ，検査パターンは存在しない*．

以上が $M0M1$ アルゴリズムの概略手順である．

次に順序回路の処理方法について述べる．順序回路は **Fig. 9** の a) ように，組合せ回路とフィードバックというモデルに展開できる．そしてこれは b) のように展開して考えれば組合せ回路として扱うことが可能である[2),4)]．ただし次の点が普通の組合せ回路と異なる．フィードバックの初期値は不明であり，故障個所は複数個ある．$M0\,M1$ アルゴリズムでは，初期値が

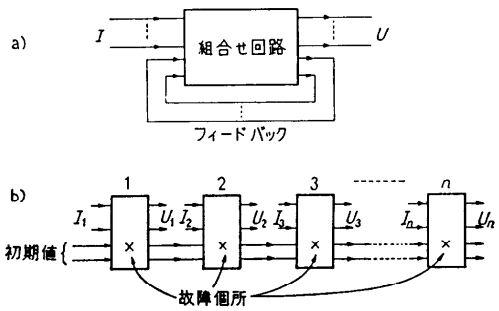

Fig. 9 The expanded model for a sequential circuit

*実はこの手順では正しくない．これについては附録で述べる．
**BPL は PL/1 のサブセットの言語機能を有する．
***故障シミュレータは，順序回路の非同期的な動作，ハザードなどを正しく解析するとともに，1つの検査パターンに対して全ての故障回路の動作を同時にシミュレーションする機能をもつ．

不明な素子または端子に対して $\dot{x}_i=5$ とすればよいが，複数個の故障を処理するために前に述べた手順で，ステップ2と3を区別せずに併合した処理を行うようにしている．

11. 実　例

$M0\,M1$ アルゴリズムを使って，実際に故障診断データの作成を行う検査システムでの結果を示す．このシステムは BPL 言語**とアセンブラ言語で作られていて，NEAC 2200/500 のコンピュータを使って運用される．**Table 3** に示したサンプルに対して，検査データの作成を行った結果を **Table 4** に示す．p_1，p_2 はプログラムパラメータであり，p_1 は1つの故障回路に対するコンピュータ時間の打ち切りを指定し，p_2 は順序回路に対するクロックパターンの制限数を指定する．$M0\,M1$ アルゴリズムの性能の評価は，

Table 3 Scales of sample circuits

	素子数	フリップフロップ数	全障害数	備考
Sample 1	66	0	194	組合せ回路
Sample 2	106	2	278	順序回路
Sample 3	273	32	892	順序回路
Sample 4	1464	228	5476	順序回路

Table 4 Computed results by NEAC 2200/500

	検出障害数 (SUCCESS)	解なし FAIL	未検出障害数	検査パターン数	処理時間 (分)	性能の評価 (%)	備考
Sample 1	194	0	0	194 (35)*	1	100	P_1= 2秒 P_2= 1パターン
Sample 2	269	5	4	285 (31)*	4	97	P_1=10秒 P_2= 2パターン
Sample 3	732	50	110	1311 (109)*	44	88	P_1=10秒 P_2= 3パターン
Sample 4	112	28	82	165	130	63	P_1=60秒 P_2= 2パターン

( )* は障害シミュレータと組合せて検査パターン数を減らした場合の数である．

SUCCESS/(SUCCESS+FAIL) としてある．**Fig. 10** (次頁参照) にパラメータ p_1 を変化させたときの性能を示す．なお Table 4 でカッコで示した値は，$M0\,M1$ アルゴリズムと故障シミュレータ*** を組合せて，検査パターンを減らした場合の数を示している．

12. むすび

$M0\,M1$ アルゴリズムは，簡潔な代数演算と発見的な手順からなる検査パターンの作成手法である．本アルゴリズムのプログラム化は簡単であり，繁雑な手続きを必要としない．J. P. Roth が提案した D アルゴリ

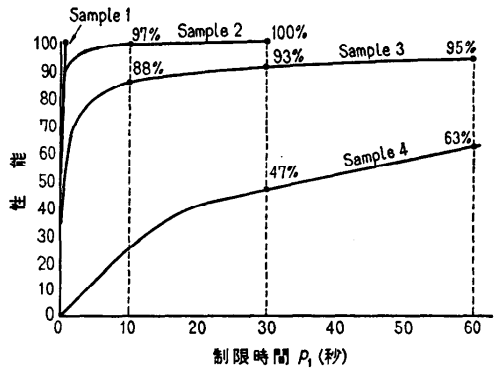

Fig. 10 Efficiency of $M0\,M1$ algorithm

ズム[1),4)]は組立式手法*であり，$M0\,M1$ アルゴリズムは繰返し式手法**であると言える．したがってどちらが優劣ということではなく，どちらも長短をもつ．

本アルゴリズムの性能は Fig. 10 からわかるように，あまり大きくない順序回路では，1故障につき数秒程度の制限時間で 80～100%の成功率をもつ．したがって実用上は，30～50 IC 程度のパッケージに対して，人手作成の 10～20 パターンを追加することにより，ほぼ全ての故障を検出することができる．

おわりに，本システムを共に開発した諸氏に深く感謝します．

参 考 文 献

1) J. P. Roth: Programmed Algorithms to Compute Tests to Detect and Distinguish between Failures in Logic Circuits, Computer (IEEE), vol. 16, No. 10, pp. 567～580 (1967).
2) 久保秀士：順序回路のテスト系列作成法について，NEC Res. Develop., No. 12, (1968).
3) Y. Koga & C. Chen & K. Naemura: A Method of Test Generation for Fault Location in Combinational Logic, Fall Joint Computer Conference, pp. 69～78 (1970).
4) J. P. Roth: A Heuristic Algorithm for the Testing of Asynchronous Circuits, Computer (IEEE), vol. 20, No. 5, pp. 639～647 (1971).

* constructive method
** iterative method

附 録

本論文中の第 10 章で，$M0\,M1$ アルゴリズムの手順の概略を述べたが，これは正確ではない．完全な手順にするには，B変換を次のように修正する．

入力端子 x_i をB変換する場合，$\hat{x}_i=3$ のとき新しく $\hat{x}_i=B3=1$ とするが，このとき $\hat{x}_i=0$ とおいた状態の入力パターンを記憶しておく．同様に，$\hat{x}_i=4$ のときは新しく $\hat{x}_i=B4=0$ とするが，このとき $\hat{x}_i=1$ とおいた状態の入力パターンを記憶しておく．そしてステップ2または3で行き詰った場合に，記憶してあるパターンの1つを取出してやり直す．もし記憶を全て取り出してしまったら，もはや検査パターンは存在しないと言える．

(昭和49年 5 月 1 日受付)
(昭和49年10月 1 日再受付)