構造記述関数による組合せ回路の検査系列の生成について

樹下　行三　　高松　雄三，柴田　正治
（広島大・総合科学）　（佐賀大・理工）

1. まえがき

論理回路の故障診断の問題は，回路のLSI化に伴って増々重要になりつつある。この問題は，かなり古くから取扱われており多くの手法†が発表されている。その中で入出力端子間の経路に着目し一つの経路だけで信号の変化を伝搬させる一次元経路活性化法，および多重経路活性化に拡張したDアルゴリズムなどがよく知られている。これらの手法は，本質的には組合せ回路に対するものであるが，現実には少し規模が大きくなると純粋に組合せ回路であるものは少く，順序回路である場合が多い。しかしながら，順序回路の検査系列生成は，組合せ回路に比べて更に複雑であり，これまで実用化に至っている手法はないようである。

論理回路において，一つの故障を仮定してそれを検査する入力を求めるという問題は，NP－Completeであるということが知られており[2]，すべての仮定故障に対して検査入力を求めることはその手数上容易ではない。そこで，大規模回路に対しては検出率を犠牲にして，適当な入力を与えてその入力によって検出可能な故障を出来るだけ多く求める手法がある。これは論理シュミレーションを並用して実行する故障シュミレーションであり，これらの手法では，適当な計算時間で実用上有効な検出率を有する検査系列が生成出来るとされている。

先にも述べたように被検査対象となる論理回路は，一般に順序回路であることや，また，LSI，VLSIとして回路の集積度が高くなることから，検出率を上げることが困難になり新しい観点からの取扱いが要求されている。この方向として，検査容易設計が脚光を浴び，その一つとして，スキャンパスを含む回路設計が行われている[3],[4]。この考え方は，順序回路の故障検査を組合せ回路の検査に帰着させ，論理回路の検査系列生成を組合せ回路の上で取扱おうとするものである。このような観点から検査の問題を考えると，再び組合せ回路の故障検査系列の生成が重要な問題となる。これが本論文で組合せ回路を対象としている動機である。

本論文では，論理回路のゲート接続情報を含んだ論理関数記述（以下，構造記述関数という）を用いて回路表現をし，論理関数の計算処理という形で検査系列を生成する方法を提案する。このような構造記述関数は，Armstrong[5]，Clegg[6]などにより提案されており，検査系列生成に関する議論が行われているが，記憶容量の点（これらの構造記述関数では回路が大きくなるとその表現式が急激に大きくなる）などから実用化はなされていないようである。ここでは，圧縮した形の構造記述関数を導入しこの問題を解決している。

このような構造記述関数を用いても，生成アルゴリズムがNP－Completeであるということには変りがないが，以下述べる構造記述関数を用いる方法では，アルゴリズム自身の記述（プログラムの大きさ）は簡単であり，完全な検査入力の生成とか近似的検査入力の生成が比較的容易にできる。この方法でも，作業用の記憶容量が回路の規模

† 歴史的な文献は多くあるが，直接関係するもの以外は省略する。なお，これらの文献に関するまとめに文献(1)がある。

と伴に増加するが，最近では，計算機の使用可能な記憶容量が増大しているので，これが計算時間の短縮につながるならば，この方法は一つの有効な手段を与えるものであろうと考えている。

2. 構造記述関数

ここでは，本論文で基本となる構造記述関数（Structure Description Function，以下，SD関と略す）を定義し，その算出アルゴリズムを示す。また，以下の議論で必要となる諸定義について述べる。

本稿で対象とするゲートは，AND，OR，NOT，NANDおよびNORの5種とする。

次の条件を満足するような元の回路と等価な樹枝状回路で実現される関数をSD関と定義する。従って，以下この式を<u>等価樹枝状回路表現</u>（Equivalent Fanout-free Form，EF関と略記する）と呼ぶ。

「入力端子から出力端子へ至るすべての経路を，入力変数が同一であっても別個の入力とする，入力に補元の形を許したANDおよびORのみからなる元の回路と等価な樹枝状回路。」

このようなEF関は次の手順Aを用いて得られる。

<u>[手順A] EF関を求める手順</u>

以下，i個の$L(i)$, $L(i-1)$, …, $L(1)$の系列を$\tilde{L}(i)$で表す。また，回路の信号線にはラベルA，B，……が付されているものとし，ラベル$\bar{A}$, $\bar{B}$, ……は正負の符号を有するものとする。

[A·1] 初期設定

(1-1) $i \leftarrow 1$, $j \leftarrow 1$.

(1-2) 出力端子に接続されている信号線（Aとしよう）を取り出し，$L(i) \leftarrow A$とし[A·2]を行う。

[A·2] 以下の手順を行え。

(2-1) $i \leftarrow i+1$とし，ラベル$L(i-1)$から入力端子へ向い，ラベル$L(i-1)$がゲートGの出力線であればGの入力線のうちマークされていない入力線（Bとしよう）を取り出しBをマークする。(2-3)へ。そうでなければ，ラベル$L(i-1)$が接続されている入力側の信号線（Cとしよう）を取り出し(2-2)へ行く。

(2-2) $L(i) \leftarrow C$，Cが入力端子に接続されていれば(2-5)へ。そうでなければ(2-1)へ行く。

(2-3) Gが否定素子（NOR，NANDおよびNOT）ならば，ラベル$L(i-1)$の符号の反転したものをBに付し（B^*と書く），$L(i) \leftarrow B^*$とする。そうでなければ，$L(i) \leftarrow B$とする。

(2-4) Bが入力端子に接続されていれば(2-5)へ。そうでなければ，(2-1)へ行く。

(2-5) 経路P_jとして$\tilde{L}(i)$を登録する。[A·3]を行う。

[A·3] 以下の手順を行え。

(3-1) ラベル$L(i)$が出力端子に接続されていれば，[A·4]を行う。ラベル$L(i)$がゲートGの入力線であれば(3-3)へ。そうでなければ，(3-2)へ行く。

(3-2) $i \leftarrow i-1$として(3-1)へ。

(3-3) Gにマークされていない入力線（Dとしよう）が存在すれば(3-4)へ。そうでなければGの入力線すべてのマークを消し，$i \leftarrow i-1$として(3-1)へ行く。

(3-4) $L(i) \leftarrow D$として，P_jとP_{j+1}の関係$R(P_j, P_{j+1})$を次のように決定する。

$$R(P_j, P_{j+1}) \leftarrow G_N \cdot G_P \cdot Sgn(L(i-1))$$

ここで，G_Nは$\tilde{L}(i)$上のゲートの数であり，また，

$$Sgn(L(i-1)) = \begin{cases} 1 & (\text{ラベル}L(i-1)\text{が正}) \\ -1 & (\text{ラベル}L(i-1)\text{が負}) \end{cases}$$

$G_P=\begin{cases}1 & (G\text{がOR又はNANDのとき})\\ -1 & (G\text{がAND又はNORのとき})\end{cases}$

$j \leftarrow j+1$ として，〔A・2〕へ行く。

〔A・4〕 $R(P_j, P_{j+1})$ の絶対値の大きい順に，正のときは P_j と P_{j+1} との演算がOR，負のときはANDとなるように括弧付の式を構成する。

（手順A終り）

〔例〕

上記の手順Aを用いて図1の回路のEFF（E_f と書く）を求める。

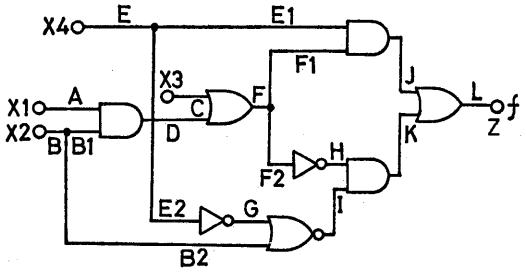


図1．論理回路の例

〔A・1〕 (1-1) $i \leftarrow 1$, $j \leftarrow 1$. (1-2) $L(1) \leftarrow L$.

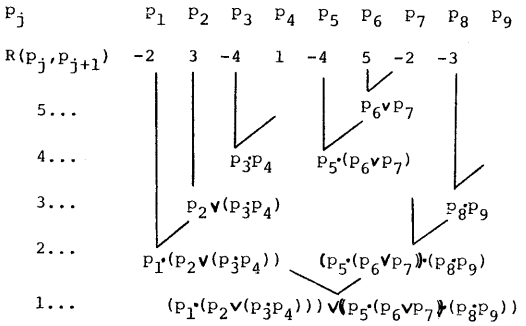
〔A・2〕 (2-1) $i \leftarrow 2$, Jを取り出しマーク. (2-3) $L(2) \leftarrow J$. (2-4), (2-1) $i \leftarrow 3$, E_1 を取り出しマーク. (2-3) $L(3) \leftarrow E_1$. (2-4), (2-1) $i \leftarrow 4$, Eを取り出す. (2-2) $L(4) \leftarrow E$. (2-5) P_1 として EE_1JL を登録。

〔A・3〕 (3-1), (3-2) $i \leftarrow 3$. (3-1), (3-3) マークされていない F_1 が存在. (3-4) $L(3) \leftarrow F_1$, $R(P_1, P_2) \leftarrow 2 \times (-1) \times 1$, $j \leftarrow 2$.

〔A・2〕 (2-1) $i \leftarrow 4$, Fを取り出す. (2-2) $L(4) \leftarrow F$. (2-1) $i \leftarrow 5$, Cを取り出しマーク. (2-3) $L(5) \leftarrow C$. (2-4), (2-5) P_2 として CFF_1JL を登録.

以下，〔A・3〕，〔A・2〕を繰り返し行うことにより，P_3, P_4, …, P_9 として，それぞれ，$ADFF_1JL$, BB_1DFF_1JL, $C\bar{F}F_2HKL$, $A\bar{D}F\bar{F}_2HKL$, $B\bar{B}_1D\bar{F}F_2HKL$, $EE_2\bar{G}IKL$, $B\bar{B}_2IKL$, （－記号は負のラベルを表す），が得られる。また，$R(P_j, P_{j+1})$, $j=2, 3, \cdots, 8$ は，3, −4, 1, −4, 5, −2, −3 となる。

〔A・4〕 $R(P_j, P_{j+1})$ から E_f を構成すると次のようになる。



従って，

$E_f = P_1 \cdot (P_2 \vee P_3 \cdot P_4) \vee P_5 \cdot (P_6 \vee P_7) \cdot P_8 \cdot P_9$

が得られる。

（例終り）

上記の手順Aで得られるEFFを展開し，経路 P_i の積和形で表した式を求めるとCleggの定義したSDFF（SP00Fと呼ばれている）になり，また，回路の信号線のラベルのかわりにゲート名で経路 P_i を表せばArmstrongの定義したSDF（enfと呼ばれている）となる。

SP00F または enf を用いると，それらは積和形で表されているので経路の個数の増大と伴に必要とする記憶容量は急激に増加し実用的でない。しかしながら，本稿で導入したEFFは経路の個数で表され，また，経路を表す情報はそのEFFを求める手法から容易に木構造で表すことが出来，効率良く記憶できるという利点を有している。

以下，論理関数fを実現する回路のEFFを E_f で表す。また，E_f を展開して積和形で表した式（SP00F）を S_f で表し，その積項を T_j で，T_j の部分積を t_j のように表すことにす

る。そうすると，S_f は，一般に，

$$S_f = T_1 \vee T_2 \vee \cdots \vee T_k \qquad (1)$$

と表される。

また，これらの式に現われる入力から出力へ至る経路を P_j で表す。一般に経路 P_j は $P_j = z_i^* \alpha$，ここで，z_i^* は z_i または $\bar{z}_i$ であり，α は入力 z_i^* から出力へ至る経路 P_j を表すラベルの系列，で表されるから，z_i^* を P_j の入力リテラルといい，入力リテラルが z_i^* である経路 P_j を $P_j\langle z_i^*\rangle$ と書く。

3. 経路活性化条件と経路微分

ここでは，2. で述べたSDFに経路に対するブール微分（以下，経路微分という）という概念を導入し，経路の活性化条件を考察する。

n 個の入力変数 $x_1, x_2, \cdots, x_n$ を有する論理回路において，n 個の入力変数に定数 $a_i \in \{0, 1\}$ または，ドントケア d を適当に割当てた n 次元ベクトルを入力といい I で表す。

〔定義1〕 与えられた回路に対して入力 $I_1 = (b_1, \cdots, b_{i-1}, a_i, b_{i+1}, \cdots, b_n)$ を加える場合を考える。ここで，b_j は a_j または d である。このとき，経路 $P_j\langle z_i^*\rangle$ 上の任意の信号線の信号値を1個所変化させたとき，その信号線を入力とするゲートの出力値が変化すれば，経路 $P_j\langle z_i^*\rangle$ は I_1 で a_i ― クリティカル経路であるという。

〔定義2〕 経路 $P_j\langle z_i^*\rangle$ が $I_1 = (b_1, \cdots, b_{i-1}, a_i, b_{i+1}, \cdots, b_n)$ で a_i ― クリティカル経路であり，かつ，$I_2 = (c_1, \cdots, c_{i-1}, \bar{a}_i, c_{i+1}, \cdots, c_n)$ で $\bar{a}_i$ ― クリティカル経路であるとき，$P_j\langle z_i^*\rangle$ は $I = (e_1, \cdots, e_{i-1}, d, e_{i+1}, \cdots, e_n)$ で活性化経路であるという。ただし，$c_j = b_j$ または $c_j = d$，$e_j = b_j \cap c_j$ $(j \neq i)$ であり，演算 $\cap$ は，$a \cap a = a$，$a \cap d = d \cap a = a$，$a \cap \bar{a} = \phi$（ϕ は定義されないことを意味する）である。

活性化経路においては，その定義から入力端子の変化は出力端子まで伝搬する。ところが，a_i ― クリティカル経路においては，必ずしも入力端子の変化が出力端子まで伝搬するとは限らない。

〔定義3〕 経路 $P_j\langle z_i^*\rangle$ を a_i ― クリティカル経路とする入力 I が存在しないとき，$P_j\langle z_i^*\rangle$ は非クリティカル経路であるという。

次に，回路の経路 P_j が与えられたとき，P_j が活性化経路かクリティカル経路かまたは非クリティカル経路であるかを判定する方法を述べよう。

〔定義4〕 次式で定まる dS_f/dP_j を S_f の経路微分という。

$$\frac{dS_f}{dP_j} = S_f(P_j = 0) \oplus S_f(P_j = 1) \qquad (2)$$

ここで，$S_f(P_j = a)$ は S_f に現われる P_j の値を a にした式を表す。

〔定義5〕 dS_f/dP_j のすべての P_l $(l \neq j)$ に対応する入力リテラル z_i^* を代入して得られる関数を経路活性化関数といい，$f_{P_j}(X)$ で表す。また，$f_{P_j}(X) = 1$ を満たす入力の集合を Π_j で表す。

〔性質1〕 経路 $P_j\langle z_i^*\rangle$ が活性化経路であるための必要十分条件は $I \in \Pi_j$ なる I が存在することである。ここで I は定義2である。また，$P_j\langle z_i^*\rangle$ がクリティカル経路であるための必要十分条件は $I_1 \in \Pi_j$ なる I_1 が存在することである。ここで I_1 は定義1のそれである。また，$\Pi_j = \phi$（ϕ は空集合を表す）のとき，かつそのときに限り $P_j\langle z_i^*\rangle$ は非クリティカル経路である。（証明略†）

†本稿では，性質の証明は紙面の都合で省略する[7]。

この性質から検査入力を生成することは経路微分を行い，その値を1とする入力集合を求めることに帰着される。従って，以下経路微分の算出法を考察する。

［性質2］ S_f の項 T_j において，経路 P_j が含まれている項を $T_{j_1}, \cdots, T_{j_m}$，そうでない項を $T_{i_1}, \cdots, T_{i_\ell}$ とするとき(2)は次の式で与えられる。

$$\frac{dS_f}{dP_j} = (t_{j_1} \vee \cdots \vee t_{j_m}) \cdot \overline{T}_{i_1} \cdots \overline{T}_{i_\ell} \quad (3)$$

ここで，$T_{j_k} = P_j \cdot t_{j_k}$ $(k=1,\cdots,m)$ である。

この性質を用いることにより経路微分が排他的論理和の演算を行うことなく得られる。ところが，式(3)を用いるためには E_f を展開した式 S_f を用いなければならない。このとき，回路の経路の数が増加すると項 T_j の数が急激に増加し，作業用の記憶容量の増大ばかりでなく計算時間の増加も免れ得ない。そこで，経路微分を E_f を用いて算出する方法を次に示す。

以下，E_f を否定したものを $\tilde{E}_f$ で表す。E_f ($\tilde{E}_f$) は P_j について正(負)関数であるから，

$$E_f = P_j \cdot E_{f_j} \vee E_f(P_j=0)$$

$$\tilde{E}_f = \overline{P}_j \cdot \tilde{E}_{f_j} \vee \tilde{E}_f(P_j=1)$$

と表すことができる。$E_f(P_j=a)$ は E_f の P_j の値を a とした式である。

［性質3］ S_f の経路微分は E_{f_j}, $\tilde{E}_{f_j}$ を用いて次式で与えられる。

$$\frac{dS_f}{dP_j} = E_{f_j} \cdot \tilde{E}_{f_j} \quad (4)$$

$\tilde{E}_f$ は E_f から容易に求めることができ，また，E_{f_j} ($\tilde{E}_{f_j}$) は E_f ($\tilde{E}_f$) から式の処理により切り出すことができる。式(4)を用いることにより ビット演算を導入することも可能となり，効率良く経路微分が得られる。

次章でこの経路微分に基づいた検査入力生成アルゴリズムを述べよう。

4. 検査入力生成アルゴリズム

3. で述べたように検査入力を生成するには，経路微分を行い $f_{P_j}(X)=1$ を満たす活性化（クリティカル）入力を生成すればよい。このとき，被検査回路の規模，要求される検査入力により，次のような検査入力生成アルゴリズムが考えられる。

(1) 与えられた経路 P_j を活性化（クリティカル）する入力の生成

アルゴリズムPA：P_j について経路微分を行い，$f_{P_j}(X)=1$ とするすべての入力の集合 Π_j を求める。

アルゴリズムPB：P_j について経路微分を行い，$f_{P_j}(X)=1$ とする1つの入力を求める。

アルゴリズムPAを行うことにより，その経路が活性化，クリティカルまたは非クリティカル経路であるが決定できる。従って，回路の経路の遅延検査入力[8]を生成する場合有用であろう。アルゴリズムPBは経路微分に要する時間の短縮ができるから，大規模な回路に適用することができる。

(2) 被検査回路の検査入力集合の生成

アルゴリズムTA：すべての経路についてアルゴリズムPAを行う。

アルゴリズムTAを行うことにより，多重活性化，多重D伝搬に相当する検査入力を得ることができ，完全な検査入力集合を求めることができるが[7]，回路規模が大きくなるに伴い実用的な処理時間で得られなくなる。そこで，次のような近似的な手法を考える。

アルゴリズムTB：ある適当な経路の順序でアルゴリズムPBを実行し，すべての信号線が活性化（またはクリティカル）経路上にあれば終了する。

アルゴリズムTC：未検出信号線（活性化またはクリティカル経路上に未出現信号線）に対して，それを含む

経路を求めアルゴリズムPA（またはPB）でその経路の経路微分を行い検査入力を求める。この操作をすべての信号線が含まれるまで繰返し実行する。

アルゴリズムTB，TCは近似的検査入力集合を求めているが，少くともすべての信号線を含んでいるという意味で100%の検出率を得ようとするものである。

アルゴリズムTD：被検査回路のすべての入力信号線（および入力信号線のファンアウト線）をただ1つの活性化経路（もし存在すれば）で覆うように経路微分を行う。

このアルゴリズムを行うことにより，入力線のみの検査入力で全体の信号線の検査がどの程度可能かということを知ることができる（表1参照）。

5. プログラムの構成

ここでは，これまで述べたSDFを用いた検査入力生成法のプログラム化について述べる。

5.1 全体の構成および概要

作成したプログラムは，図2に示すように，(I) 回路記述から回路テーブルへの変換，(II) EFFの作成，(III) 経路微分，の3つの部分から構成されている。これらのルーチンはさらにいくつかのサブルーチンで構成されている。

5.2 サブルーチンの概要

ELMNTB：付録に示す回路記述フォーマットで書かれた被検査回路の回路記述データを読み込み，回路記述リストを印刷し，次に回路記述を2つの回路テーブル（内部表現）に変換する。このとき，回路記述データのエラー（未定義ラベル，多重定義ラベル，フォーマットエラーなど）をチェックする。

CONCAT：回路テーブルの信号線名をパックする。

PATHTB：回路テーブルから，2.で述べた手順Aを用いてEFFを作成する。これを，回路の出力の数だけ繰返す。

PRNTEF：PATHTBで作られたEFFおよび経路のリストの印刷。

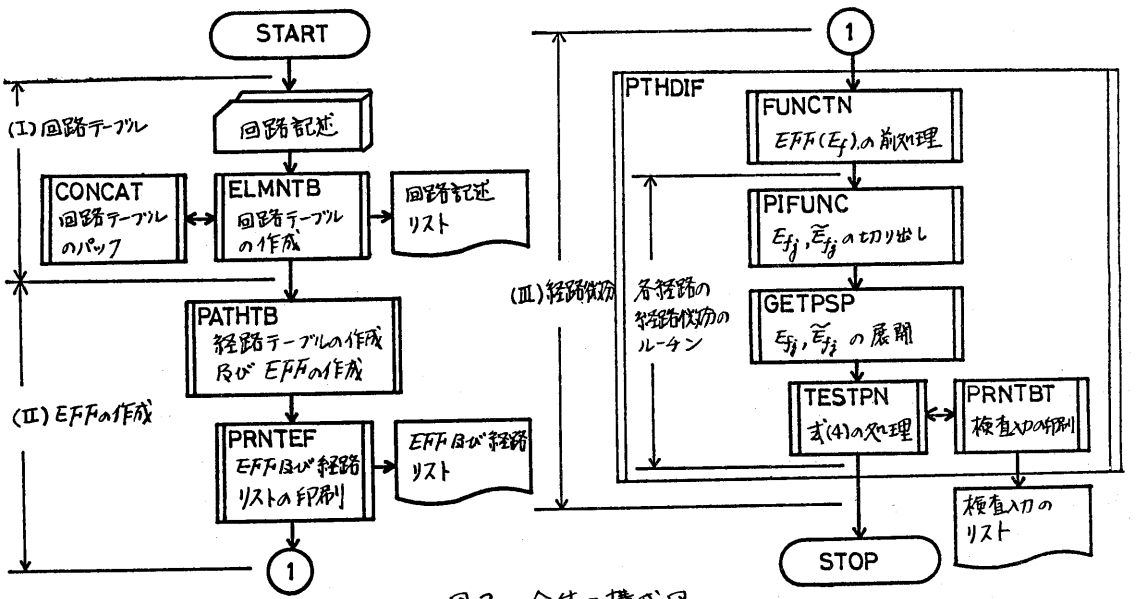


図2. 全体の構成図

PTHDIF：4.で述べた検査入力生成アルゴリズムを実行するルーチンであり，各アルゴリズムにより異なる構成になる。このルーチンには，基本的には式(4)を処理するために以下示す5つのサブルーチンを有している。

FUNCTN：EFF（E_f）から $\tilde{E}_f$ を作り，経路微分を行うための前処理を行う。

PIFUNC：E_f および $\tilde{E}_f$ から E_{fj}，$\tilde{E}_{fj}$ なる式の切り出しを行う。

GETPSP：E_{fj} および $\tilde{E}_{fj}$ に，それぞれ対応するハカリテラルを代入し，展開した式 $E_{fj}(X)$ および $\tilde{E}_{fj}(X)$ を求める。

TESTPN：$E_{fj}(X)$，$\tilde{E}_{fj}(X)$ から式(4)を用いて $fp_j(X)=1$ を満たす入力を生成する。

PRNTBT：TESTPNで生成された入力を印刷する。

5.3 アルゴリズムの選択法

4.で述べた検査入力生成アルゴリズムを実行するため，現在までに作成したプログラムは次のとおりである。

(1) アルゴリズムTA
(2) アルゴリズムTB
(3) アルゴリズムTD

これらの選択はサブルーチンPTHDIFを換えることにより行われる。

5.4 プログラム仕様

作成したプログラム仕様は次のとおりである。

(1) 使用言語：FORTRAN IV
(2) プログラムの規模
　(I) 回路テーブル作成：約210ステップ
　(II) EFFの作成：約270ステップ
　(III) 経路微分：約500ステップ

であり，メインプログラムを含めて合計約1Kステップのプログラムである。

(3) データ領域および作業領域

上記のプログラムを実行するには，被検査回路の信号線の総数 --- L
EFFの経路の数 --- P
ゲートの数 --- G

とすると，およそ次のデータ領域が必要である。

(I) 回路テーブル --- $5\times L+m\times G$
　m：ゲートの平均ファンイン数
(II) EFFの作成 --- $k\times P$
　k：定数（$5\leq k\leq 10$）
(III) 経路微分 --- $C\times P$
　C：定数（$\simeq 6$）
(IV) E_{fj}，$\tilde{E}_{fj}$ を展開するときの作業領域 --- W

作業領域Wは回路に依存するため評価は困難である。なお，表1の回路番号4以外のWはおよそ2.5KB程度であった。

6. 実行結果と評価

現在，作成したプログラムを数10～100ゲート程度からなる回路に適用し，EFFの作成，検査入力の生成を行っている。表1にその実行結果の一例を示す。使用計算機は九州大学大型計算機センターFACOM M-190およびM-200である。

アルゴリズムTAの実行結果から，EFFの作成については，(i)計算時間はほぼ回路の経路の数に比例する。(ii)経路情報の記憶容量は経路の総数の数倍（およそ3～5程度）である。(iii)経路の総数はゲート数の1～10倍程度である。などがわかる。これらのことから，EFFの処理に関しては，計算時間，記憶容量ともに実用的な結果であろうと考えている。また，経路微分に関しては，その処理時間が経路の数の増加とともに急激に大きくなる。この場合，近似的手法を用いる必要があろう（表1，アルゴリズムTD）。いずれにしても，回路の経路の数がその回路の複雑さを表す一つの有力な指標であることが確認される。

表1. 検査入力生成プログラムの実行結果

**回路番号	ゲート数	信号線数	入力数	出力数	EFFの作成 時間(秒)	EFFの作成 経路テーブルの大きさ	EFFの作成 経路の数	検査入力生成 TA 時間(秒)	TA 活性*2	TB*3 時間*1(秒)	TB*3 %*4	TB 時間*1(秒)	TB 検出率*5
1	31	120	11	3	1.1	727	25	0.1	10	0.06	64		64
							98	2.7	24	2.0	75	3.5	63
							98	3.0	24	2.1	75		63
2	35	115	6	8	1.2	1005	39	0.1	7	0.08	64		49
							49	0.3	5	0.2	46		41
							24	0.06	5	0.04	60		59
							64	0.4	12	0.3	72	1.2	53
							44	0.2	7	0.1	68		65
							70	0.6	3	0.4	34		32
							52	0.3	12	0.2	73		64
							9	0.01	6	0.01	86		86
3	46	129	9	2	3.9	1330	144	23.3	144	4.8	100	2.2	42
							144	18.9	144	4.0	100		42
4	96	264	14	8	12.7	3190	384	335 *1	194	211	89	140	66
							72	0.6	50	0.4	95		86
							48	0.3	26	0.2	92		83
							56	1.0	56	0.7	100	11.3	94
							81	6.5	81	3.9	100		92
							24	0.2	24	0.2	100		100
							156	17.7	134	9.2	97		87
							108	3.0	86	1.8	96		87

7. むすび

ここでは，圧縮したSD下表現を導入し，そのSD下を用いた論理回路の検査入力生成法について考察した。

まず，SD下に経路微分を定義し，経路の活性化条件を明らかにした。また，これらの理論に基づいて作成した検査入力生成プログラムの概要およびその実行結果について述べた。EFFの処理に関しては実用的な結果が得られたと考えられるが，検査入力生成（経路微分）に関しては，経路の数の増加とそもそもその処理時間が大きくなる。したがって，4. で述べた近似的手法を用いる必要があろう。

しかしながら，アルゴリズムTAで得られる結果から，その回路の検証が行えると考えられるので，回路の規模，要求される検査入力の性質によりアルゴリズムを選べば，ここで述べた手法は回路の検証ということを含めて一つの有用な検査入力生成法であると思われる。

本研究は一部文部省科学研究費（課題番号 455146）による。

*1 計算時間は FACOM M-200（その他は，M-190）による。

*2 活性化経路の数。

*3 このTBでは，活性化経路のみに着目したアルゴリズムである。

*4 活性化経路で覆われる信号線の全体の信号線に対する比率(%)。

*5 この検出率では，等価故障を考慮していない。

** 回路番号 1,2,3,4 のカタログ番号は，それぞれ，SN5485, 9307, SN54LS280, SN54181 である。

参考文献

(1) Van Cleemput, W. M.: " Computer Aided Design of Digital Systems - A Bibliography, Vol. I,II,III", Pitman (1976,1977,1978).

(2) Ibarra, O. H. and Sahni, S. K. : " Polynomially complete fault detection problems ", IEEE Trans. Comput., Vol. C-24, 3, p.242, Mar. 1975.

(3) Yamada, A.,et al. :" Automatic test generation for large digital circuits ", Proc. 14th Design Automation Conf., June, p.78, 1977.

(4) Eichelberger, E. B. and Williams, T. W. : " A logic design structure for LSI testability ", ibid., p.462.

(5) Armstrong, D. B. : " On finding a nearly minimal set of fault detection tests for combinational logic nets ", IEEE Trans. Electron. Comput., Vol. EC-15, 1, p.66, Feb. 1966.

(6) Clegg, F. W. : " Use of SPOOF's in the analysis of faulty logic networks ", IEEE Trans. Comput., Vol. C-22, 3, p.229, Mar. 1973.

(7) 樹下, 高松, 柴田: "構造記述関数を用いた論理回路の故障検査法", 電子通信学会, 電子計算機研究会 (昭和55年2月発表予定).

(8) Shedletsky, J. J. : " Delay testing LSI logic ", Proc. 1978 Int. Symp. Fault-Tolerant Computing, June, p.159, 1978.

付 録

A. 回路記述フォーマット

5. のSDFによる検査入力生成プログラムで用いた回路記述フォーマットについて述べる。ステートメントは大別すると, (1) 回路記述ステートメント, (2) コメントステートメント, の2つからなる。

A·1 ステートメントの形式

A·1·1 回路記述ステートメント

一連の回路記述ステートメントにより回路の接続情報を表すものであり, 次のような / (スラッシュ) で区切られた3つのフィールドからなる。

␣オペレーションフィールド/第1オペランド/第2オペランド/

(1) オペレーションフィールド: 回路の論理情報を表し, AND, OR, NAND, NOR, NOT, IN (入力線の定義), OUT (出力線の定義) および FOUT (ファンアウトの指定) から1つを第1カラムを空白にして第2カラムから左につめて記入する。

(2) 第1, 第2オペランド: オペレーションフィールドに記述されたオペレーションの対象となる信号線の組合せを記述する。このフィールドに記述される信号線はコンマ (,) 記号で区別される。

注1) このフィールドに空白記号があればすべて除いて処理される。

注2) 信号線は8文字以内の英数字で名前が付けられる。

注3) 2行以上にわたって記述する場合は, 2行目以降各行の第1カラムに等号 (=) 記号を記入する。

A·1·2 コメントステートメント

第1カラムにアスタリスク (*) 記号を記入することにより, その行すべてがコメントと見なされる。ただし, 回路記述リストの中ではプリントされる。

A·2 回路記述ステートメント

(1) AND, OR, NAND, NOR ステートメント

論理素子 (ゲート) の入出力信号線間の関係を記述するもので, 次の形式で表す。

␣OP./入力線1,...,入力線N/出力線/

ここで, OP. は AND, OR, NAND, NOR のいずれかである。

注1) 第1オペランドの入力線名は, で区切って記入する。

注2) 第2オペランドは1個の出力信号線名を記入する。

[例] ␣AND/A1, A2, A3/B1/

(2) NOT ステートメント

NOTゲートを記述するもので, 次の形式で表す。

␣ NOT / 信号線1 / 信号線2 /

注1) 第1,第2オペランドともに信号線名は各1個記入する。信号線1がNOTゲートの入力線,信号線2が出力線を表す。

[例] ␣ NOT / A5 / B5 /

(3) FOUT ステートメント

ファンアウト情報を記述するもので,第1オペランドに1個の信号線,第2オペランドに第1オペランドの信号線がファンアウトしている信号線を記述する。

␣ FOUT / 信号線1 / 信号線2, … , 信号線N /

[例] ␣ FOUT / A1 / B1, B2, B3, B4 /

(4) IN および OUT ステートメント

入力(出力)端子に接続されている信号線を記述するステートメントであり,次の形式で表す。

␣ IN / 入力端子名1, … , 入力端子名N / [入力信号線1, … , 入力信号線N] /
␣ OUT / 出力信号線1, … , 出力信号線M / [出力端子名1, … , 出力端子名M] /

[ ] は省略が許されることを示す。

注1) 端子名と信号線名は記入されている順に対応する。従って,記入される個数は同数でなければならない。

注2) 第2オペランドの [ ] の部分が省略されたとき,入力端子名と入力信号線名(出力信号線名と出力端子名)とは同一とされる。

[例] ␣ IN / PN1, PN2, PN3 / X1, X2, X3 /
␣ IN / INP1, INP2, INP3 //
␣ OUT / Y1, Y2, Y3, Y4 / P1, P2, P3, P4 /
␣ OUT / OPT1, OPT2, OPT3 //

(5) END ステートメント

回路記述の終りを示すのに用い,一連の回路記述ステートメントの最後のステートメントである。第1カラムを空白とする次の形式で表す。

␣ END

A·3 一般的注意事項

(1) 継続行に制限はない。

(2) 回路記述ステートメントは,ENDステートメントを除いて順序は任意でよい。

A·4 回路記述例

図1の回路の回路記述例を次に示す。

***** PROGRAM LIST *****

```
      ***** EX.1 *****
       IN/X1,X2,X3,X4/A,B,C,E/
       FOUT/B/B1,B2/
       FOUT/E/E1,E2/
       AND/A,B1/D/
       OR/C,D/F/
       NOT/E2/G/
       NOT/F2/H/
       NOR/G,B2/I/
       AND/E1,F1/J/
       AND/H,I/K/
       FOUT/F/F1,F2/
       OR/J,K/L/
       OUT/L/Z/
       END
```