検査容易化構造をもつ順序回路のテスト自動生成†

林 照峯†† 畠山一実†† 森脇 郁†††
鈴木 茂†††† 高倉正博†††††

スキャン設計方式に基づいて論理回路を設計するとき，スキャンの規則性を崩してもスキャン回路の一部を通常回路と兼用したいという場合がしばしば現れる．本論文ではこれらのスキャン・イン／アウト手順が規則的に表現できないような論理回路に対し，スキャン・パターンを含むテスト系列を自動的に生成できるテスト生成手法を提案する．本手法はスキャン手順に対応する4段階のテスト生成手続きを構成している点に特徴がある．また，テスタビリティ・メジャーの拡張により通常回路の動作とスキャン回路の動作の切り分けを意識しながらテストを生成できるようにしたこと，パルス信号線の事前認識によりパルス値に伴う矛盾発生を低減させたことなどによって，テスト生成能力の向上を図っている．2～7Kゲート規模の論理回路6ケースに対する性能評価実験の結果，平均で本手法は従来の9値拡張Dアルゴリズムの検出率87.8%を約10%上回る検出率98.3%のテスト系列を生成できた．

1. まえがき

論理 LSI の大規模化と品種数の増大に伴って，そのテストをいかに効率よく作成するかが重要となってきた．このため，論理設計の段階から検査の容易化を考慮するいわゆる “Design for Testability” の必要性が認識されるようになってきた．なかでも，スキャン設計方式[1)~3)]は順序回路に対するテスト生成の問題を組合せ回路に対するそれに置き換えて解くことを可能にするため，検査の容易化に有効である．また，組合せ回路に対する高速なテスト生成アルゴリズム[4),5)]も開発されており，これらはスキャン設計方式をさらに強力なものにしている．

しかし，スキャン設計方式はこのような長所をもつにもかかわらず，必ずしも広く論理設計者に受け入れられているとは言えない．これにはいくつかの理由があるが，その一つは厳格な論理設計規則が論理設計の自由度を少なくしていることであろう．これに対して，スキャン設計方式をもう少し自由度を増して用いようとする動きがある．たとえば，部分的にスキャン・イン/アウトが可能でないフリップ・フロップ（FF と略す）の存在を許しながらも，故障検出率の高いテストを生成できる不完全スキャン・パス[6)]と呼ばれる方式が考えられている．

また，論理回路のオーバ・ヘッドを減らす等の目的で，通常動作のためのゲートとスキャン動作のためのゲートとを共用する方法がしばしば用いられる．この場合には，FF のスキャン・イン/アウトが規則的でなくなることがあり，FF または外部ピンではさまれた組合せ回路のテスト生成は自動的に行えても，これを外部ピン・イメージのパターンに変換するのが容易でなくなることがある．さらに，スキャン用回路部分はテスト生成回路モデルのなかに含まれないため，この部分の故障に対する故障辞書が作成できないという問題もある．したがって，このような回路に対しては順序回路全体をテスト生成回路モデルとして扱う必要がある．

一方，順序回路に対しては拡張Dアルゴリズムというテスト生成手法がよく用いられている．しかし，この手法は順序回路に検査容易化のための回路を付加した場合でも，期待するほど故障検出率の向上を図ることができないことが多い．というのは，この手法ではもとの回路と付加回路を区別しないで処理するため，付加回路の動作を必ずしもうまく利用しきれないことが多いからである．逆に言えば，このような順序回路に対して故障検出率を向上させるためには，その付加回路をいかに有効に活用できるような手法とするかがポイントとなる．

本論文では，スキャン・イン/アウトの手順が規則的に表現できないようなスキャン構造をもつ順序回路を対象に，スキャン回路を有効に利用しながらテストを自動生成する一つの手法を提案する．

---

† Automatic Test Generation for Sequential Circuits with a Structure for Testability by TERUMINE HAYASHI, KAZUMI HATAYAMA (Hitachi Research Laboratory, Hitachi, Ltd.), KAORU MORIWAKI (Kanagawa Works, Hitachi, Ltd.), SHIGERU SUZUKI (Odawara Works, Hitachi, Ltd.) and MASAHIRO TAKAKURA (Hitachi Engineering, Co., Ltd.).

†† (株)日立製作所日立研究所
††† (株)日立製作所神奈川工場
†††† (株)日立製作所小田原工場
††††† 日立エンジニアリング(株)

2. 対象とする論理構造

本章では対象とする論理回路の構造を明らかにするため，スキャン構造をいくつかのタイプに分類しておく．

Level Sensitive Scan Design (LSSD) 方式[1]，Scan Path 方式[2]，Random Access Scan 方式[3]，などのスキャン設計方式で設計された論理回路においては，FF のスキャン位置とその状態からスキャン・イン/アウトのためのパターン列を規則的に表現することができる．ここでは，このようなスキャン構造を定形スキャン構造と呼ぶ．一方，これらを規則的に表現できないようなスキャン構造を非定形スキャン構造と呼ぶ．

また，ある FF に対するスキャン・イン/アウトが他の FF を動作させずに行えるスキャン構造を組合せ型スキャン構造と呼び，そうでないものを順序型スキャン構造と呼ぶ．たとえば，Random Access Scan は組合せ型であり，LSSD や Scan Path は順序型である．

本論文では，組合せ型のスキャン構造をもつ順序論理回路を対象とし，かつ非定形スキャン構造をもつ場合に特に有効なテスト自動生成手法について述べる．

3. テスト生成の基本方式

スキャン構造をもつ論理回路に対して，ある単一縮退故障を検出するためのテストは以下の四つのパターンの系列からなる．

（1）故障検出に必要な FF の状態をイニシャライズするためのパターン列（スキャン・イン・パターン列）．

（2）故障を外部出力ピンまたは FF の入力ピンに伝播するためのパターン．

（3）FF の入力ピンに伝播された故障を FF の出力ピンに伝播させるパターン．

（4）FF 出力ピンに伝播された故障を外部出力ピンに伝播するパターン列（スキャン・アウト・パターン列）．

このうち，(1)と(4)のパターン列は FF のスキャン位置と状態から機械的に求められるようにしておくことにより，テスト自動生成で扱わなくてもすむようにするのが普通である．一方，(2)のパターンは FF または外部ピンではさまれた部分組合せ回路ごとに自動生成される．また，(3)は FF のテスト生成用素子モデルの工夫により(2)のパターン生成で同時に求められるようにしておく方法がよく用いられる．

しかし，非定形スキャン構造をもつ回路においては必ずしもこの方法が効率的とは言えないことがある．たとえば，**図1**に示す回路は**表1**に示すようにスキャン・パターンが規則的でないため，このパターンを誤りなく求めること自体が(2)のパターンを求めることと同程度に複雑である．したがって，(1)～(4)のパターンをすべて求めることができるテスト生成方式が望ましい．

本論文で述べる方式は組合せ型スキャン構造をもつ回路に対してこれを実現するために，以下の4段階でテスト生成を行うようにしている．

(A1) CCTG (Combinational Circuit Test Generation Stage):

FF または外部ピンではさまれた組合せ回路のテストパターンを生成する(上記(2)に対応)．この部分には組合せ回路用のテスト生成アルゴリズムを用いる．

(A2) FFTG (Flip-Flop Test Generation Stage):

FF の出力に故障を伝播させるパターンを求める(上記(3)に対応)．

表 1 スキャン・パターンの例
Table 1 An example of scan patterns.

機能＼外部ピン		入力ピン					出力ピン
		PI_1	PI_2	PI_3	PI_4	PI_5	PO_1
Scan In 0	FF_1	0	0	−	−	⎍	−
	FF_2	0	0	−	−	⎍	−
Scan In 1	FF_1	0	⎍	0	−	0	−
	FF_2	⎍	0	−	1	0	−
Scan Out	FF_1	0	0	−	0	0	(FF_1-Q)
	FF_2	0	0	−	1	0	(FF_2-$\bar{Q}$)

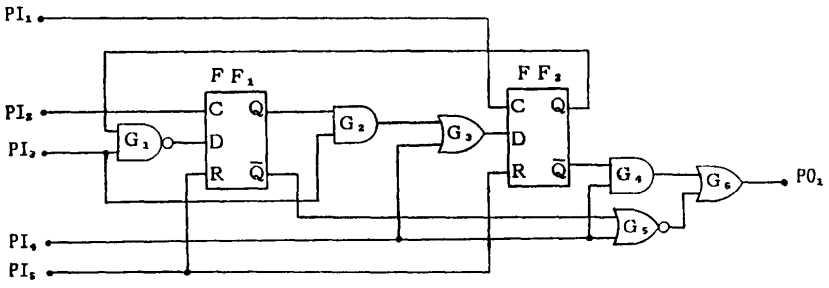

図 1 組合せ型非定形スキャン構造をもつ回路の例
Fig. 1 An example of circuit with irregular and combinational scan structure.

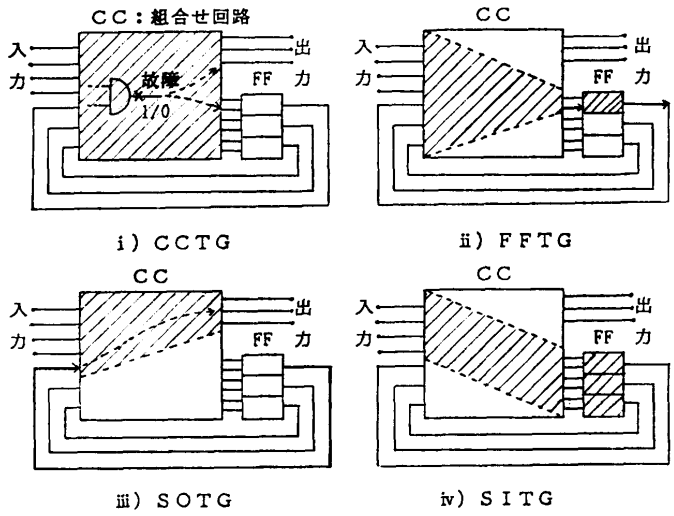

図 2　4段階テスト生成方式
Fig. 2　Four stage test generation method.

(A3) SOTG (Scan Out Test Generation Stage):
FF 出力まで伝播された故障を外部出力ピンに伝播するパターンを求める（上記(4)に対応).

(A4) SITG (Scan In Test Generation Stage):
必要な FF の状態をイニシャライズするパターン列を求める（上記(1)に対応).

ここで，(A1)において故障が外部出力ピンに伝播されるときは(A2)および(A3)を行う必要はない.

図 2 は組合せ回路と FF のループからなる一般的な順序回路モデルにおいて，(A1)～(A4)の各段階における処理で必要となる回路部分を示している．すなわち，CCTG においては組合せ回路部分を，FFTG においては故障を伝播させる FF とその FF に信号を供給する組合せ回路部分を，SOTG においては故障伝播された FF の出力ピンおよび外部入力ピンから外部出力ピンに至るパスを構成する回路部分を，SITG においては状態イニシャライズが必要な FF と外部入力ピンからこれらの FF に至るパスを構成する回路部分のみを処理の対象とする.

4. テスト生成手法

4.1 テスト生成用論理値

本手法においては**表 2** に示す 15 個の論理値を用いる．このうち，X, U, 0, 1, D, $\bar{D}$ はDアルゴリズム[7]など一般のテスト生成手法で用いられる論理値と同じ意味をもつ．P, N はそれぞれ正極性または負極性のパルスを，S_0, S_1 はそれぞれ FF 出力信号線が0または1の状態であることを表しており，これらはスキャン・イン・パターンを求めるときのみに用いられる．また，Z, G_0, G_1, F_0, F_1 はトライステート・ゲートを扱うための論理値である.

4.2 時系列管理

順序回路において，ある故障を検出するテストは一般に複数時刻からなる系列で構成される．以下では本手法における時刻の扱い方を述べる.

いま，CCTG が処理する時刻を t_0 (t_0 は基準時刻) とする．FFTG においてはクロック信号線に正極性のとき 0100，負極性のとき 1011 の時系列を与えて故障値を FF に伝播させる．このうち，はじめの3時刻は正パルスを 010 に負パルスを 101 に展開して扱うことを表している．このように展開したのはクロック信号線上の故障を扱いやすくするためである．また，最後の1時刻はスキャン・アウトのために用意した時刻である．いま，上記4時刻のうちクロックが "on" になりうる2番目の時刻を t_0 に対応させる．そうすると，FFTG においては (t_0-1)～(t_0+2) を，SOTG では (t_0+2) の時刻を扱うことになる．また，SITG では FF の状態イニシャライズのために $t \leqq (t_0-1)$ の時刻 t を扱う.

表 2　テスト生成用論理値とその意味
Table 2　Meaning of logic values for test generation.

論理値	意味	使用条件 CCTG	FFTG	SOTG	SITG	備考
X	don't care	○	○	○	○	通常のテスト生成手法で用いる論理値
U	unknown	○	○	○	○	
0	0 level	○	○	○	○	
1	1 level	○	○	○	○	
D	good 1, faulty 0	○	○	○	*	
$\bar{D}$	good 0, faulty 1	○	○	○	*	
P	positive pulse (⎍)	×	×	×	○	クロックパス上の信号線のみ
N	negative pulse (⎍)	×	×	×	○	
S_0	0 state	×	×	×	○	FFの出力信号線のみ
S_1	1 state	×	×	×	○	
Z	high impedance	○	○	○	○	トライステート出力信号線のみ
G_0	good 0, faulty Z	○	○	○	*	
G_1	good 1, faulty Z	○	○	○	*	
F_0	good Z, faulty 0	○	○	○	*	
F_1	good Z, faulty 1	○	○	○	*	

○：使用可　×：使用不可　*：$t \leqq (t_0-2)$では使用不可

表 3 Dキューブと C キューブの例
Table 3 Examples of D-cubes and C-cubes.

i) 2入力ANDのCキューブ

No	入力 I_1	入力 I_2	出力 O_1	時刻	適用条件
1	0	X	0	t	$t \leqq (t_0-2)$ or
2	X	0	0	t-1	$O_1(t-1) \neq 0$
3	0	X	0	t-1	$t \geqq (t_0-1)$
	0	X	0	t	and
4	X	0	0	t-1	$O_1(t-1) = 0$
	X	0	0	t	
5	1	1	1	t	
6	P	1	P	t	$t \leqq (t_0-2)$
7	1	P	P	t	
8	N	1	N	t	
9	1	N	N	t	

ii) 2入力ANDのDキューブ

No	入力 I_1	入力 I_2	出力 O_1	時刻	適用条件
1	d	1	d	t	1次元活性化
2	1	d	d	t	
3	d	d	d	t	2次元活性化

iii) Dラッチの C キューブ

No	C	D	S	R	出力 Q	出力 $\bar{Q}$	時刻	適用条件
1	0	X	0	0	0	1	t-1	$t \geqq t_0$
	0	X	0	0	0	1	t	
2	0	X	0	0	1	0	t-1	
	0	X	0	0	1	0	t	
3	0	X	0	0	S_0	S_1	t-1	$t = (t_0-1)$
	0	X	0	0	0	1	t	
4	0	X	0	0	S_1	S_0	t-1	
	0	X	0	0	1	0		
5	0	X	0	P	S_0	S_1	t	$t \leqq (t_0-2)$
6	P	0	0	0	S_0	S_1	t	
7	0	X	0	0	S_0	S_1	t-1	
	0	X	0	0	S_0	S_1	t	
8	0	X	P	0	S_1	S_0	t	
9	P	1	0	0	S_1	S_0	t	
10	0	X	0	0	S_1	S_0	t-1	
	0	X	0	0	S_1	S_0	t	

iv) Dラッチの D キューブ

No	C	D	S	R	出力 Q	出力 $\bar{Q}$	時刻	適用条件
1	0	d	0	0	U	U	t-1	D入力からの故障伝播
	1	d	0	0	d	$\bar{d}$	t	
	0	d	0	0	d	$\bar{d}$	t+1	
	0	d	0	0	d	$\bar{d}$	t+2	
2	0	1	0	0	0	1	t-1	C入力からの故障伝播
	d	1	0	0	d	$\bar{d}$	t	
	0	1	0	0	d	$\bar{d}$	t+1	
	0	X	0	0	d	$\bar{d}$	t+2	
3	0	0	0	0	1	0	t-1	
	d	0	0	0	$\bar{d}$	d	t	
	0	0	0	0	$\bar{d}$	d	t+1	
	0	X	0	0	$\bar{d}$	d	t+2	
4	0	X	0	0	0	1	t-1	S入力からの故障伝播
	0	X	d	0	d	$\bar{d}$	t	
	0	X	0	0	d	$\bar{d}$	t+1	
	0	X	0	0	d	$\bar{d}$	t+2	
5	0	X	0	0	1	0	t-1	R入力からの故障伝播
	0	X	0	d	$\bar{d}$	d	t	
	0	X	0	0	$\bar{d}$	d	t+1	
	0	X	0	0	$\bar{d}$	d	t+2	

注) d は D または $\bar{D}$ をあらわす.

4.3 テスト生成手順

(1) CCTG: 組合せ回路用のテスト生成アルゴリズムである D アルゴリズムに一意活性化操作[5]を取り入れた手法を用いてテストを求める. ここで, D アルゴリズムを用いたのはスキャン・インが必要な FF 数をなるべく少なくするためである.

(2) FFTG: 故障値が FF のデータ入力ピンに伝播された場合には, 各素子の時刻 (t_0-1) と時刻 (t_0+1) における論理値を時刻 t_0 における論理値と同一の値にする. (故障値が FF のクロック入力ピンに伝播された場合には, そのままとする.) つぎに, FF の D キューブを用いて故障値をその FF の出力ピンに伝播させる. さらに, そのとき決まった素子の論理値を拡張 D アルゴリズムの後方操作によって求める.

(3) SOTG: この部分も CCTG と同様に D アルゴリズムに一意活性化操作を加えた手法を用いる. ただし, 故障伝播された FF を除くすべての FF に対して, その状態を利用しないパターンを求める.

(4) SITG: この部分は拡張 D アルゴリズムの後方操作を用いる. 時刻 $t \leqq (t_0-2)$ は真にスキャン・インを行うための時刻であり, パルス論理値 P, N や FF の状態論理値 S_0, S_1 を使用することができる.

表 3 に 2 入力 AND ゲートとセット・リセット付 D ラッチの D キューブと C キューブの例を示す.

また, 表 4 は図 1 の回路のゲート G_2 の 0 縮退故障 (s. a. 0 と書く) に対するテスト生成過程を各段階ごとに示したものである. 最初に, CCTG では G_2 の故障を FF_2 の入力ピンにまで伝播するパターンを求める. このとき, FF のピンも外部ピンと同等に扱う. 次の FFTG では, まず FF のデータ入力ピンに故障伝播されていることから, 各素子の時刻 (t_0-1) と時刻 (t_0+1) における値を時刻 t_0 における値と同一値にセットする. その後で, FF_2 に対し表 3 の 1 番目の D キューブを適用し, 故障を FF_2 の出力側に伝

表 4 テスト生成例
Table 4 An example of test generation.

テスト生成段階	時刻	外部入力ピン PI_1	PI_2	PI_3	PI_4	PI_5	FF FF_1	FF_2	ゲート G_1	G_2	G_3	G_4	G_5	G_6	出力 PO_1
CCTG	t_0	X	X	1	0	X	1	X	X	D	D	X	X	X	X
FFTG	t_0-1	0	0	1	0	0	1	U	X	D	D	X	X	X	X
	t_0	1	0	1	0	0	1	D	X	D	D	X	X	X	X
	t_0+1	0	0	1	0	0	1	D	X	D	D	X	X	X	X
	t_0+2	0	X	X	X	0	X	D	X	X	X	X	X	X	X
SOTG	t_0+2	0	X	X	1	0	X	D	X	X	X	$\bar{D}$	0	$\bar{D}$	$\bar{D}$
SITG	t_0-2	X	P	0	X	0	S_1	U	1	X	X	X	X	X	X
	t_0-1	0	0	1	0	0	1	U	X	D	D	X	X	X	X
	t_0	1	0	1	0	0	1	D	X	D	D	X	X	X	X
	t_0+1	0	0	1	0	0	1	D	X	D	D	X	X	X	X
	t_0+2	0	X	X	1	0	X	D	X	X	X	$\bar{D}$	0	$\bar{D}$	$\bar{D}$

播させるパターンを求める．SOTG では時刻 (t_0+2) における FF_2 出力ピンの故障値を外部出力ピン PO_1 に伝播させるパターンを求める．最後の SITG では，FF_1 を1状態にするパターンを求める．この場合，時刻 (t_0-2) において FF_1 の状態イニシャライズを行うことができる．

その結果，外部入力ピン $(PI_1, PI_2, PI_3, PI_4, PI_5)$ に対する $(0, P, 0, 0, 0)\rightarrow(0, 0, 1, 0, 0)\rightarrow(1, 0, 1, 0, 0)\rightarrow(0, 0, 1, 0, 0)\rightarrow(0, 0, 1, 1, 0)$ の5時刻からなるテスト系列が得られる．

5. 性能向上のための技法

前章でテスト生成手法を述べたが，このなかにはいくつかのヒューリスティックな処理を必要とする部分が存在する．以下では主に検出率の面で性能向上を図るため，これらの処理に施した工夫点について述べる．

5.1 コスト関数

後方操作におけるCキューブの選択，前方操作における活性化パスとDキューブの選択には，各素子に対して定義されるコスト関数（テスタビリティ・メジャー）がよく用いられている．本手法においても基本的にはこの方法を用いることができる．本手法において特に注意すべき点は，時刻 $t\leqq(t_0-2)$ の SITG において FF の状態を利用せずに外部入力ピンのパターンを求めなければならないこと，SOTG において FF にしか到達しないパスに故障伝播しないようにすることである．このため，コスト関数として以下のものを用いることにした．

$w_0(G)$：FF の状態利用可能としたとき，素子Gの出力値を0にすることの難しさ．

$w_1(G)$：FF の状態利用可能としたとき，素子Gの出力値を1にすることの難しさ．

$m(p)$：外部出力ピンと FF を出力端点とみなしたとき，素子の入力ピン p への故障伝播によるテスト生成の難しさ．

$w_0'(G)$：FF の状態利用不可としたとき，素子Gの出力値を0にすることの難しさ．

$w_1'(G)$：FF の状態利用不可としたとき，素子Gの出力値を1にすることの難しさ．

$m'(p)$：外部出力ピンのみを出力端点とみなしたとき，素子の入力ピン p への故障伝播によるテスト生成の難しさ．

これらの関数の定義から，SOTG と時刻 $t\leqq(t_0-2)$ の SITG では w_0', w_1', m' を，FFTG と時刻 $t=(t_0-1)$ の SITG では w_0, w_1, m を用いる．ここで，w_0 と w_0' は Goldstein[9] の可制御性メジャー CC^0 に対応しており，同様に w_1 と w_1' は CC^1 に，m と m' は可観測性メジャー CO に対応している．また，その計算方法は文献10) の方法を基本にしている．

例として $I_1, I_2, \cdots, I_n$ の n 入力をもつ AND ゲートとリセット付Dラッチに対するコスト計算式を示す．

i) n 入力 AND ゲート

$$\begin{cases} w_0(G)=\min\limits_{1\leq i\leq n} w_0(I_i)+\Delta(G) \\ w_1(G)=\sum\limits_{i=1}^{n} w_1(I_i)+\Delta(G) \\ m(I_i)=\sum\limits_{j\neq i} w_1(I_j)+\min\limits_{k\in K} m(k) \end{cases} \quad (1)$$

$$\begin{cases} w_0'(G)=\min\limits_{1\leq i\leq n} w_0'(I_i)+\Delta(G) \\ w_1'(G)=\sum\limits_{i=1}^{n} w_1'(I_i)+\Delta(G) \\ m'(I_i)=\sum\limits_{j\neq i} w_1'(I_j)+\min\limits_{k\in K} m'(k) \end{cases} \quad (2)$$

ii) リセット付Dラッチ

$$\begin{cases} w_0(Q)=\min\{w_1'(R), w_1'(C)+w_0'(D)\}+w_0'(C)+w_0'(R)+\Delta(Q) \\ w_1(Q)=w_0'(C)+w_1'(C)+w_1'(D)+w_0'(R)+\Delta(Q) \\ m(C)=\min\{w_1'(R), w_1'(C)+w_0'(D)\}+w_0'(C)+w_0'(R)+w_1'(D)+\min\limits_{k\in K} m'(k) \\ m(D)=w_0'(C)+w_1'(C)+w_0'(R)+\min\limits_{k\in K} m'(k) \\ m(R)=w_0'(C)+w_1'(C)+w_1'(D)+w_0'(R)+\min\limits_{k\in K} m'(k) \end{cases} \quad (3)$$

$$\begin{cases} w_0'(Q)=w_1'(Q)=\infty \\ m'(C)=m'(D)=m'(R)=\infty \end{cases} \quad (4)$$

ここで，$\Delta(G)$ はGに出力分岐があるとき1，ないとき0である．また，K は出力分岐先の入力ピンの集合を表す．

(1)～(4)式の例からもわかるが，ここでのコスト関数の主な特徴として以下があげられる．

1) 素子の出力分岐をコスト関数に反映している．これはテスト生成手順の中で生ずる矛盾が出力分岐点で起こることに着目したものである．

2) FF 出力ピンに対して常に $w_0'=w_1'=\infty$ であり，FF 入力ピンに対して常に $m'=\infty$ である．これにより，FF の状態を利用しない場合のコスト関数を実現している．

3) m と m' を素子に対して定義するのでなく素子の入力ピンに対して定義している．このようにしたの

表 5 コスト関数の計算例
Table 5 An example of cost functions for sequential circuit.

i）可制御性コスト

コスト＼素子	PI_1	PI_2	PI_3	PI_4	PI_5	FF_1	FF_2	G_1	G_2	G_3	G_4	G_5	G_6
w_0	0	0	1	1	1	3	3	4	1	2	1	1	2
w_1	0	0	1	1	1	3	3	1	4	1	4	4	4
w_0'	0	0	1	1	1	∞	∞	∞	1	2	1	1	2
w_1'	0	0	1	1	1	∞	∞	1	∞	1	∞	∞	∞

ii）可観測性コスト

コスト＼素子ピン	FF_1			FF_2			G_1		G_2		G_3		G_4		G_5		G_6		PO_1
	C	D	R	C	D	R	I_1	I_2	I_1	I_2	I_1	I_2	I_1	I_2	I_1	I_2	I_1	I_2	
m	5	3	4	5	3	4	4	6	5	8	4	4	2	4	2	4	1	1	0
m′	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	2	∞	2	∞	1	1	0

は，FF やトライステート・ゲートなど入力ごとに意味の異なるような素子に対して，コスト関数のより厳密な扱いを可能にするためである．

図1の回路に対するコスト関数の計算例を**表5**に示す．

5.2 パルス信号線の認識と活用

本手法ではパルス論理値P，Nを扱う．この場合に考慮すべきことは，パルス値を与えてもテスト生成できないような信号線上にパルス値を与えないようにすることである．**図3**にパルス値の与え方が適切でないために矛盾が生ずる例を示す．この例ではクロック外部入力ピンにパルス値を与えずに，クロック制御のための外部入力ピンにパルス値を与えたため，FF_2 のクロック入力ピン（正極性）に負パルス値Nが現れ矛盾が発生している．このような不適切な決定を避けるためにはあらかじめパルス値P，Nが現れる可能性のある信号線を求めておく必要がある．

これを求める方法としては，人手でクロック外部入力ピンを指定した後で，クロック外部入力ピンからFF に至るパス上にあるゲートを求める方法が考えられる．しかし，非定形スキャン構造をもつ回路の場合，テストのためのクロック外部入力ピンと通常動作でのそれとが必ずしも同一ではないことがあるため，人手指定情報の中に誤りが混入しやすい．このことを考慮にいれ，ここでは，人手指定情報を必要としない方法とした．

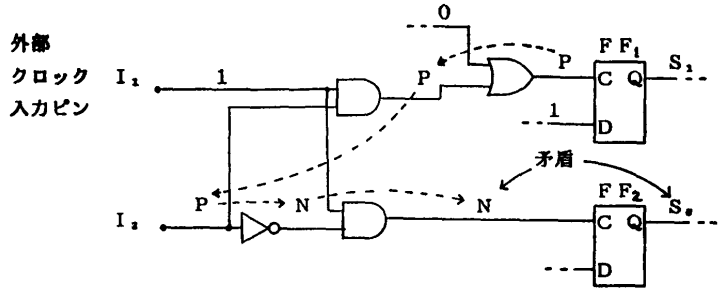

図 3 パルス値の与え方が不適切な例
Fig. 3 Unappropriate assignment of pulse value.

いま，すべての素子Gの出力信号線 l に対して p, n, p^*, n^*, pn^*, s, x のどれかの値をとるパルス信号線フラグ関数 $f(G)$ を導入する．ここで，これらの値は以下の意味をもつ．

$f(G)=p\cdots\cdots l$ は正パルス信号線である．
$f(G)=n\cdots\cdots l$ は負パルス信号線である．
$f(G)=p^*\cdots\cdots l$ は正パルス信号線の可能性あり．
$f(G)=n^*\cdots\cdots l$ は負パルス信号線の可能性あり．
$f(G)=pn^*\cdots\cdots l$ はパルス信号線の可能性がある（ただし，極性は不明）．
$f(G)=s\cdots\cdots l$ はレベル信号線である．
$f(G)=x\cdots\cdots l$ はパルス信号線かレベル信号線か不明．

ここで，pn^* と x は $f(G)$ の計算の途中でのみ現れる一時的な値である．二つのフラグ値間の強弱関係を記号＞，＝，＜で表現し，各フラグ値間の強さを，

$$s>p=n>pn^*>p^*=n^*>x$$

で与える．この強弱関係はある同一の信号線に相異なるフラグ値が要求されたとき，強い方のフラグ値に決めるために用いる．ただし，p と n あるいは p^* と n^* が同時に要求されたときは，例外としてそれぞれ s および pn^* に変えるものとする．

以上の準備をした上で，イベント追跡法に基づく $f(G)$ の計算手順を示す．

（1）FF のクロック入力ピンから外部入力ピンまたはFF に至るまでの後方追跡によって到達できるすべての素子を求める．これらの素子に対して入力側からソート番号をつける（外部入力ピンにもソート番号をつける）．

（2）まず，すべての素子 G に対して $f(G)=x$ とおき，そのあとで，つぎのどれかの条件を満たす素子Gに対して $f(G)=s$ とする．

i）素子Gにソート番号がついていないとき．

ii）素子 G がコスト関数 $w_0'(G)=\infty$ または $w_1'(G)=\infty$ のとき．

iii）素子Gが AND, OR, NAND,

NOR, NOT でなくかつ外部入力ピンでないとき（すなわち，トライステート・ゲートなどの特殊な素子や FF, EOR の出力信号線はパルス信号線とはならないものとみなす）．

iv) 素子Gの出力ピンから FF を通らずに外部出力ピンに至るパスが存在するとき．

（3）FF のクロック入力ピンのソース素子に対して，そのクロック極性に応じて $f(G)=p$ または $f(G)=n$ とおく．ここで，フラグ値を p または n に決めた素子を要素とする集合を α とする．

（4）集合 α が空であれば処理終了．そうでなければ集合 α の中からソート番号の最も大きい素子Gを求め，$\alpha=\alpha-\{G\}$ とする．素子Gの種類と $f(G)$ の値によって，Gのソース素子に対するフラグ値を決める．このとき表6のi）の規則（OR ゲートを例に示してあるが，AND, NAND, NOR, NOT ゲートについても同様）を用いる．これによって，フラグ値が変化した素子の集合を β とし，フラグ値が s 以外の値に変化した素子の集合を γ とする．$\alpha=\alpha\cup\gamma$ として（5）へ．

（5）集合 β が空なら（4）へ，そうでなければ（6）へ．

（6）集合 β の中からソート番号の最も小さい素子Gを選び $\beta=\beta-\{G\}$ とする．Gのシンク素子のうちソート番号がついているすべての素子について，表6のii）の規則（これも OR ゲートについて示してあるが他のゲートについても同様）を用いてフラグ値を更新する．このとき，フラグ値が s 以外の値に変化した素子を α の要素の中に，フラグ値が変化した素子を β の要素の中に含め（5）へ．

フラグ値の計算例を図4に示す．

以上述べた手順によって求めたフラグ値のうち，テスト生成処理で利用するのは p と n のみである．すなわち，時刻 $t\leqq(t_0-2)$ においては，$f(G)=p$ のとき，Gには論理値Nと1を与えないようにし，$f(G)=n$ のとき，Gには論理値Pと0を与えないようにする．これにより，図3の例で述べたようなパルス値の導入に伴うテスト生成能力の低下を防いでいる．

表 6 OR ゲートに対するパルス信号線フラグ計算規則
Table 6 Computation rules of pulse line flag for OR gate.

i）後方追跡による計算の場合

条　件	フラグ計算規則
$f(G)=p$	すべての I_i に対して，$f(I_i)=p$ を要求する．
$f(G)=n$	1) $f(I_j)\neq s$ かつ $f(I_j)\neq p$ であるような j が一つのみ存在するとき，$f(I_j)=n$ を要求する．
	2) そうでないとき，すべての I_i に対して，$f(I_i)=n^*$ を要求する．
$f(G)=p^*$	すべての I_i に対して，$f(I_i)=p^*$ を要求する．
$f(G)=n^*$	すべての I_i に対して，$f(I_i)=n^*$ を要求する．
$f(G)=pn^*$	$f(G)=s$ とする．
$f(G)=s$	なにもしない．

ii) 前方追跡による計算の場合

条　件	フラグ計算規則
すべての I_i について $f(I_i)=s$	$f(G)=s$ とする．
すべての I_i について $f(I_i)=p$	$f(G)=p$ を要求する．
$f(I_j)=n$ である j が一つのみ存在	$f(G)=n$ を要求する．また，$f(I_i)=n^*$ であるすべての I_i について $f(I_i)=s$ とし，$f(I_i)=pn^*$ であるすべての I_i について $f(I_i)=p^*$ とする．

注）Gは計算対象 OR ゲート，I_1〜I_n はGのソース素子を表す．

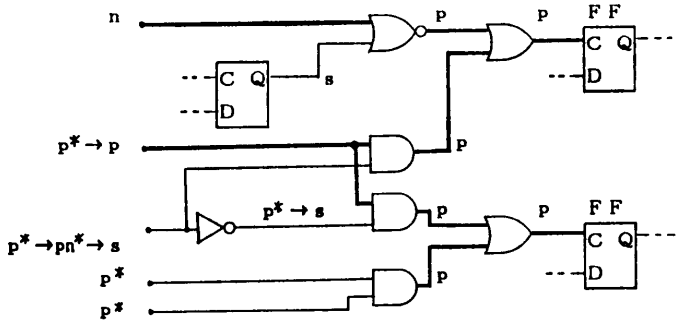

図 4 パルス信号線フラグの計算例
Fig. 4 Computation of pulse line flag.

5.3 含意操作の強化と処理範囲限定

アルゴリズム的にテスト生成を行う場合，テスト生成能力向上のためにはバックトラック回数を減少させることが必要である．このための一手段として含意操作（implication）を用いることが多い．含意操作には，AND ゲートの入力の一つが論理値0に決まったときその出力を0とする処理などの前方型，AND ゲートの出力が1に決まったときそのすべての入力を1とする処理などの後方型，AND ゲートの入力が一つを除きすべて1に決まったとき出力が0であれば残りの入力を0とする処理などの複合型がある．ここでは，検出率の向上を第一と考えてこれ

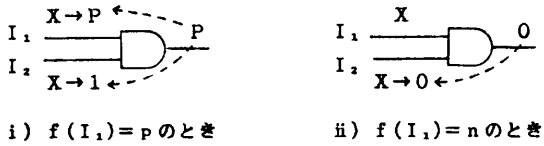

図 5 パルス信号線フラグを考慮した含意操作
Fig. 5 Consideration of pulse line flag in implication.

らをすべて行うことにした．また，パルス論理値に対しても含意操作を行うようにした．このとき，パルス信号線フラグの値も考慮した．たとえば，時刻 $t \leqq (t_0-2)$ において図5で示すように2入力 AND ゲートの出力が論理値Pに決まったとき，その一方の入力のフラグが p であれば，その入力を論理値 P とし他方の入力を論理値1とする．また，2入力 AND ゲートの出力が論理値0に決まったとき，その一方の入力のフラグが n であれば，他方の入力を論理値 0 とする．これらにより，パルス論理値の導入に起因するバックトラックを少なくした．

含意操作の強化は検出率の向上には有効であるが，処理時間の面では必ずしも有効ではない．ここでは無駄な含意操作をなるべく省くため，処理時間が最もかかる SITG において以下のことを考慮した．

時刻 $t \leqq (t_0-2)$ における SITG は FFの状態を設定するためのパターンを求めることを目的としているので，この目的に関係のない回路部分に対して処理を行う必要はない．そこで，状態設定の必要な FF からの後方追跡によって求められる部分組合せ回路部分に対してのみ含意操作を行うことにし，処理時間を短縮させた．

6. 性能評価結果

以上述べたテスト生成手法をプログラムとして実現し，数千ゲート規模の LSI 論理回路に対して性能評価実験を行った．表7は本手法（4SP 法と略す）の性能を従来の9値拡張Dアルゴリズム[8]（9 ED 法と略す）と対比して示したものである．この実験では，両手法のプログラムの実現方法や適用条件をできるだけ同一にし，その差がなるべく評価結果に反映されないように配慮した．たとえば，論理回路情報などのテーブル表現形式を両手法のプログラムとも同一にした．また，バックトラックの打ち切り制限回数を両者とも 25 回に設定して実験を行った．

実験の結果，4SP 法と 9ED 法の平均検出率はそれぞれ 98.3% と 87.8% であり，4SP 法の方が 9ED 法より検出率で約 10% 上回る性能を得ることができた．また，4SP 法は 9ED 法の約 1/2 強の時間で処理できた．両手法ともこれ以上処理時間を増やしたとしても，すなわちバックトラック打ち切り制限回数をもっと大きな値に設定しても，検出率の向上はわずかであることがわかっている．したがって，この結果は両手法の能力をよく表していると考えることができる．

次に，テスト系列数について見ると，4SP 法の方が 9ED 法よりも平均で約3倍多くなった．これはテスト生成を試行しかつ成功した故障の数が約 1.5 倍になったことと，一度の試行で生成されるテスト系列数が約2倍になったことによる．また，一度に生成されるテスト系列数が増えるのは，長い系列を必要とする検査しにくい故障についてもテスト生成していることと，FF の状態イニシャライズをほとんどスキャン回路のみで行っているためである．

評価実験により，本手法が組合せ型の非定形スキャン構造をもつ順序回路に対して有効であることを確認できた．

表 7 性能評価実験結果
Table 7 Experimental results of test generation.

項目 / 回路	ゲート数	FF数	検出率 (%)		処理時間 (相対値)		テスト系列数		テスト生成成功回数比	1回の生成系列長の比
			4SP	9ED	4SP	9ED	4SP	9ED	(4SP対9ED)	(4SP対9ED)
A	2258	96	99.8	97.1	0.4	1.0	1742	1625	1.4	0.8
B	3260	131	99.7	91.5	1.2	2.4	3628	1854	1.3	1.5
C	4972	193	96.1	77.9	5.5	15.3	7374	1014	1.5	4.8
D	4981	259	99.9	72.8	4.9	14.0	7573	1854	2.3	1.8
E	5708	311	98.6	91.7	8.3	7.3	8298	2642	1.7	1.8
F	7315	576	95.6	85.6	1.9	0.8	2514	1170	0.9	2.3
平均	4749	261	98.3	87.8	3.7	6.8	5188	1693	1.5	2.2

7. むすび

本論文ではスキャン・イン/アウトの手順が規則的に表現できないようなスキャン構造をもつ論理回路に対して，スキャン・パターンを含むテスト系列を自動的に生成できる4段階のテスト生成手法を提案した．とくに，スキャン動作と通常動作との切り分けを意識しながらテストを生成できるようにしたこと，パルス値の導入に伴う矛盾発生の可能性を

低減したことなどにより，テスト生成能力の向上を図った．さらに，数千ゲート規模の論理回路に対する性能評価実験により，本手法が実用的な性能をもつことを確認した．

参考文献

1) Eichelberger, E. B. and Williams, T. W.: A Logic Design Structure for LSI Testability, *Proc. 14th DA Conf.*, pp. 462-468 (1977).
2) Funatsu, S., Wakatsuki, N. and Arima, T.: Test Generation Systems in Japan, *Proc. 12th DA Conf.*, pp. 114-122 (1975).
3) Ando, H.: Testing VLSI with Random Access Scan, *Dig. COMPCON 1980*, pp. 50-52 (1980).
4) Goel, P.: An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits, *IEEE Trans. Comput.*, Vol. C-30, No. 3, pp. 215-222 (1981).
5) Fujiwara, H. and Shimono, T.: On the Acceleration of Test Generation Algorithms, *IEEE Trans. Comput.*, Vol. C-32, No. 12, pp. 1137-1144 (1983).
6) Trisher, E.: Incomplete Scan Path with an Automatic Test Generation Methodology, *Dig. 1980 IEEE Test Conf.*, pp. 153-162 (1980).
7) Roth, J. P.: Diagnosis of Automata Failures: A Calculus and a Method, *IBM J. Res. Dev.*, Vol. 10, pp. 278-291 (1966).
8) Goshima, S. et al.: Diagnostic System for Large Scale Logic Cards and LSI's, *Proc. 18th DA Conf.*, pp. 256-259 (1981).
9) Goldstein, L.H.: Controllability/Observability Analysis of Digital Circuits, *IEEE Trans. Circ. Syst.*, Vol. CAS-26, No. 9, pp. 685-693 (1979).
10) 林，五嶋，岡：順序回路に対するテスト発生の一手法，電子通信学会論文誌，Vol. J 64-D, No. 9, pp. 869-876 (1981).

(昭和61年7月14日受付)
(昭和62年1月14日採録)

林　照峯（正会員）

昭和22年生．昭和44年名古屋大学工学部電気学科卒業．昭和46年同大学院工学研究科修士課程修了．同年(株)日立製作所入社．以来，LSI，プリント板に対する設計自動化システムの研究開発に従事．現在，同社日立研究所第3部主任研究員．電子情報通信学会会員．

鳥山　一実（正会員）

昭和28年生．昭和51年京都大学工学部数理工学科卒業．昭和57年同大学院博士課程修了．工学博士．昭和57年4月(株)日立製作所に入社．以来，同社日立研究所にて，論理回路に対するテストデータの設計自動化の研究に従事．IEEE，日本オペレーションズリサーチ学会会員．

森脇　都（正会員）

昭和31年生．昭和51年松江高専電気工学科卒業．同年4月(株)日立製作所入社．現在，同社神奈川工場DA設計部にて，社内診断DAシステムの開発に従事．

鈴木　茂（正会員）

昭和27年生．昭和50年茨城大学工学部電子工学科卒業．同年(株)日立製作所入社．現在同社小田原工場に勤務．電子計算機の故障診断システム，設計自動化システムの開発に従事．

高倉　正博

昭和31年生．昭和54年電気通信大学電気通信学部物理工学科卒業．同年4月，日立エンジニアリング(株)入社．現在，同社ソフトウェアシステム部にて，LSIの故障診断DAシステムの開発に従事．