設 計 自 動 化 35－3
（1986. 12. 16）

ALTES/RA: レジスタファイルを含む論理回路のテストパターン自動生成プログラム

猿山 秀一　荻原 拓治　村井 真一

三菱電機（株） 情報電子研究所

レジスタファイルを含む論理回路のテストパターン自動生成プログラムＡＬＴＥＳ／ＲＡを開発した。テストデータは、テストパターン・ジェネレータと故障シミュレータを繰り返し適用することにより、効率よく生成される。テストパターン・ジェネレータは、組合せ回路用のアルゴリズムを時間展開して、一段レジスタを含む順序回路に対して時系列テストパターンが生成できるように拡張した。また、故障によりアクセスできなくなるレジスタには固有初期値を割当て、正常回路で０と１になり故障回路で固有初期値となる複数のテストパターンを生成するため、従来の方法では生成できなかった故障に対してもテストパターン生成が可能となった。

ＡＬＴＥＳ／ＲＡをレジスタファイルを含む実際のＬＳＩ２品種に適用したところ、故障検出率１００％のテストパターンが自動生成できた。

ALTES/RA : Test Generaor for Logic Circuits with Register Files

Shuichi Saruyama , Takuji Ogihara and Shin-ichi Murai

Information Systems and Electronics Development Lab.
Mitsubishi Electric Corpration , 5-1-1 Ofuna Kamakura 247 , Japan

This paper describes an automatic test pattern generation program ALTES/RA for logic circuits with register files. It consists of fault generation phase, testability measure calculation phase, and test data generation phase. Test data are generated by a test pattern generator with fault simulator in test data generation phase. Test patterns are effectively generated by new algorithm. That is, conventional test pattern generator could not generate test patterns for some faults which cause registers unaccessable, however, this generator can generate test patterns for such faults by the algorithm assigning the unique initial values to such registers and treating their values as a kind of fault signal.

ALTES/RA has been successfully applied to two LSI's including register files, and test patterns with 100% fault coverage has automatically been generated.

1．はじめに

論理回路の大規模化に伴い、そのテストはますます困難になりつつある。この問題を解決する手法の一つとして、スキャン設計のようなテスト容易化設計［1］が行われている。しかし、スキャン設計は回路内の記憶素子をすべてシフトレジスタにしてスキャンパスを構成するため、ゲート数が増加する。従って、レジスタファイルのような記憶素子を多く含む論理回路に適用することはチップ面積の点から非現実的であった。従来、このような回路に対しては人手でテストパターンを作成し、故障シミュレーションを行っていたが、故障検出率の高いテストパターンを作成するには多くの時間を要していた。

このような問題点を解決するため、スキャン設計を適用していないレジスタファイルを含む論理回路のテストパターン自動生成プログラムＡＬＴＥＳ／ＲＡを開発した。本報告では，ＡＬＴＥＳ／ＲＡの構成と、テストパターン生成アルゴリズム及びその評価結果について報告する。

2．レジスタファイル

レジスタファイルを含む論理回路の一般的構成を図１に示す。レジスタファイルは、デコーダ、セレクタ及びレジスタアレイから構成されており、その他の部分は組合せ回路である。

レジスタファイルの特徴としては、順序回路であっても、レジスターレジスタ転送のない一段レジスタ回路であり、一般順序回路（例えばカウンタ）よりも比較的テスト生成容易な構造である。ＡＬＴＥＳ／ＲＡは、このような一段レジスタの順序回路に対してテストパターン自動生成が可能である。

表１に、ＡＬＴＥＳ／ＲＡの扱う素子タイプを示す。図１のような論理回路に対しては、レジスタアレイ部を

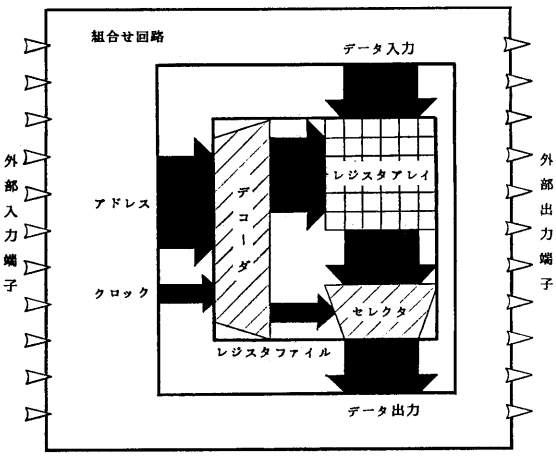

図１．レジスタファイルを含む論理回路の一般的構成

表１．ＡＬＴＥＳ／ＲＡの扱う素子タイプ

外部端子	外部入力（ＰＩ）　外部出力（ＰＯ） 双方向（ＢＩＤＲ）
基本ゲート	ＡＮＤ／ＮＡＮＤ　ＯＲ／ＮＯＲ ＮＯＴ　ＸＯＲ　ＴＲＩバッファ
記憶素子	Ｄラッチ

1bitＤラッチに展開し、デコーダ、セレクタ部等その他の部分を基本ゲート（ＡＮＤ、ＯＲ等）に展開してテストパターン生成を行う。

次に縮退故障（０縮退故障：ＳＡ０、１縮退故障：ＳＡ１）がレジスタファイルに及ぼす影響について述べる。

図２は、簡単な２×４bit のレジスタファイルであり、アドレス信号は入力デコーダ、出力セレクタに共通である。図２中に、代表的な３種類の故障を示す。

①Ｄラッチのデータ入力に影響を及ぼす故障

②Ｄラッチのクロック入力に、正常回路ではクロックが入らないが、故障回路ではクロックが入る（以下、正常回路で０、故障回路でクロックＣの信号値を０／Ｃのように表す）故障

③Ｄラッチのクロック入力に、正常回路ではクロックが入るが、故障回路ではクロックが入らない（Ｃ／０）故障

特に、上記③のような故障は、故障回路においてＤラッチの内部状態を初期化することができず、不定値Ｘのまま残ってしまう。従って、従来のアルゴリズムでは故障信号を作り出すことができず、このような故障を検出することはできなかった。

ＡＬＴＥＳ／ＲＡのテストパターン・ジェネレータは、故障回路においてクロックが入らなくなったＤラッチの初期値に固有初期値を導入することにより、このような故障に対してもテストパターン生成が可能である。

3．ＡＬＴＥＳ／ＲＡの構成

ＡＬＴＥＳ／ＲＡの構成を図３に示す。

●故障生成　等価故障を除去した代表故障のみ生成する。なお、扱う故障は単一縮退故障である。また、未使用論理の故障やＧＮＤ／ＶＣＣ固定信号線のＳＡ０／ＳＡ１のような検出不能故障もあらかじめ除去される。

●試験容易性尺度計算　４節で述べる可制御性・可観測性尺度からなる試験容易性尺度を計算する。この尺度は、テストパターン生成時に制御、観測容易な経路の選択基準として使われる。

●テストデータ生成　５節で述べるテストパターン・ジェネレータと故障シミュレータを繰り返し適用することによりテストデータを自動生成する。

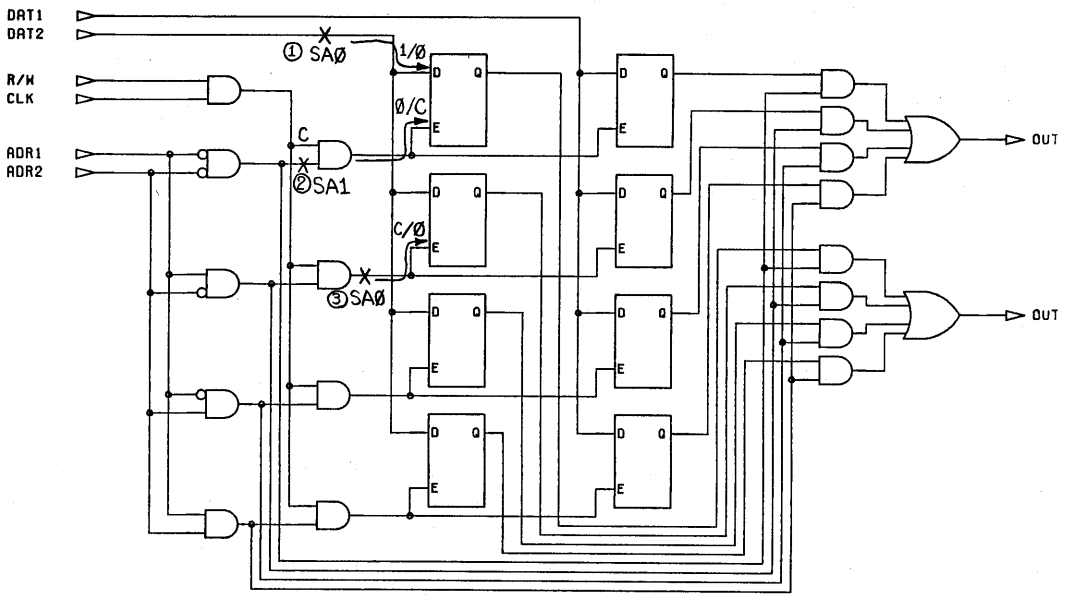

図２．レジスタファイルに対する故障の影響

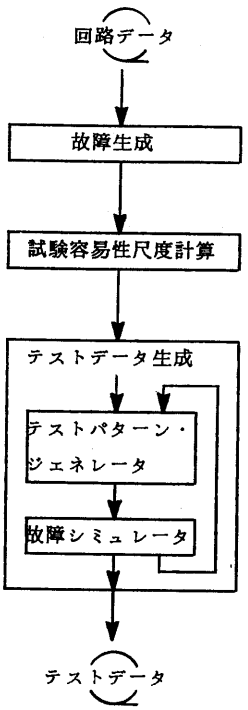

図３．ＡＬＴＥＳ／ＲＡの構成

４．試験容易性尺度計算

論理回路内の各ノードに対し、次に示す可制御性尺度・可観測性尺度からなる試験容易性尺度を計算する。

ＣＣ０（Ｎ）	：ノードＮの組合せ０可制御性尺度
ＣＣ１（Ｎ）	：ノードＮの組合せ１可制御性尺度
ＣＯ　（Ｎ）	：ノードＮの組合せ可観測性尺度
ＳＣ０（Ｎ）	：ノードＮの順序０可制御性尺度
ＳＣ１（Ｎ）	：ノードＮの順序１可制御性尺度
ＳＯ　（Ｎ）	：ノードＮの順序可観測性尺度

上記尺度は、各素子タイプに対し表２に示す式に従って、可制御性尺度は外部入力端子から、可観測性尺度は外部出力端子から順次計算される。

表２の式は、Goldstein 尺度［２］（以下Ｇ尺度と略す）とは若干異なる。Ｇ尺度では、組合せ素子に対するＣＣ０、ＣＣ１、ＣＯの各式には＋１の項があるが、本尺度では＋１の項がなく、尺度の意味が次に示すように異なる。

●Ｇ尺度

ノードＮをある信号値に制御、あるいはノードＮの信号値を観測するため、信号値を設定する必要のある最小組合せノード数

●本尺度

ノードＮをある信号値に制御、あるいはノードＮの信号値を観測するため、信号値を設定する必要のある見かけ上の最小外部入力端子数

すなわち、図４に示すような回路に対してＧ尺度では

ＣＣ０（Ｙ１）　≠　ＣＣ１（Ｙ２）

ＣＣ１（Ｙ１）　≠　ＣＣ０（Ｙ２）

であるが、本尺度では

ＣＣ０（Ｙ１）　＝　ＣＣ１（Ｙ２）

ＣＣ１（Ｙ１）　＝　ＣＣ０（Ｙ２）

となり、Ｇ尺度よりも実際の論理に近い。

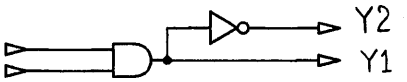

図４．尺度の相違

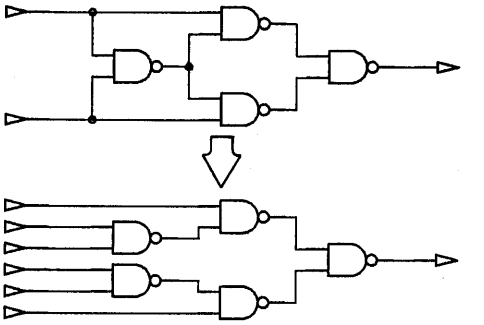

図５．再収斂回路と樹状回路

また、ＳＣ０、ＳＣ１、ＳＯに関しても次のように尺度の意味が異なる。

●Ｇ尺度

ノードＮをある信号値に制御、あるいはノードＮの信号値を観測するため、信号値を設定する必要のある最小順序ノード数

●本尺度

ノードＮをある信号値に制御、あるいはノードＮの信号値を観測するため、信号値を設定する必要のある見かけ上の最小クロック数

Ｇ尺度、本尺度とも再収斂分岐の影響を考慮していないため、再収斂分岐のある回路に対しては、図５に示すように見かけ上樹状回路に展開した場合の最小数であり、正確ではない。しかし、再収斂分岐に対しても正確な尺度を計算するのは、膨大な計算量になる。また、再収斂分岐を考慮した近似的な尺度も提案されている［３］が、テストパターン生成アルゴリズムにおいて自由度のある選択の基準として用いた場合、再収斂分岐を考慮しない尺度との差異はあまり見られないことから、計算の簡単な尺度を採用した。

ここで計算された尺度は、次節で述べるテストパターン・ジェネレータにおいて、自由度のある選択が生じた場合、制御容易な経路、または観測容易な経路を選択するための基準として使われる。

表２．可制御性尺度・可観測性尺度

素子タイプ	シンボル	CC0(Y)	CC1(Y)	CO(Xi)	SC0(Y)	SC1(Y)	SO(Xi)
ＰＩ	X ▷— Y	1	1	CO(Y)	0	0	SO(Y)
ＰＯ	X —▷ Y	CC0(X)	CC1(X)	0	SC0(X)	SC1(X)	0
ＡＮＤ	X_1 … X_N — Y	$\underset{i=1}{\overset{N}{MIN}}(CC0(Xi))$	$\sum_{i=1}^{N}(CC1(Xi))$	$CO(Y)+\sum_{j\neq i}(CC1(Xj))$	$\underset{i=1}{\overset{N}{MIN}}(SC0(Xi))$	$\sum_{i=1}^{N}(SC1(Xi))$	$SO(Y)+\sum_{j\neq i}(SC1(Xj))$
ＯＲ	X_1 … X_N — Y	$\sum_{i=1}^{N}(CC0(Xi))$	$\underset{i=1}{\overset{N}{MIN}}(CC1(Xi))$	$CO(Y)+\sum_{j\neq i}(CC0(Xj))$	$\sum_{i=1}^{N}(SC0(Xi))$	$\underset{i=1}{\overset{N}{MIN}}(SC1(Xi))$	$SO(Y)+\sum_{j\neq i}(SC0(Xj))$
ＮＯＴ	X —▷○— Y	CC1(X)	CC0(X)	CO(Y)	SC1(X)'	SC0(X)	SO(Y)
Ｄラッチ	D Q / E	CC0(D)+CC1(E)	CC1(D)+CC1(E)	CO(D) − CO(Q)+CC1(E) CO(E) − CO(Q)+CC0(D)+CC1(D)	SC0(D)+SC1(E)+1	SC1(D)+SC1(E)+1	SO(D) − SO(Q)+SC1(E)+1 SO(E) − SO(Q)+SC0(D)+SC1(D)+1
分岐	X — Y_1 … Y_N	CC0(X)	CC1(X)	MIN(CO(Yi))	SC0(X)	SC1(X)	MIN(SO(Yi))

5．テストデータ生成

テストデータは、テストパターン・ジェネレータと故障シミュレータを繰り返し適用することによって生成される。すなわち、まず未検出の故障を１つ選び、テストパターン・ジェネレータによってその故障を検出するテストパターンを生成する。次に、生成されたテストパターンに対し、不定値Ｘとして残っている部分に０１の乱数を割り当て、故障シミュレータによって同時に検出できる故障を除去する。

以上の繰り返しにより、テストデータを生成する。

5．1　テストパターン・ジェネレータ

従来、組合せ回路に対しては、Ｄアルゴリズム［４］、ＰＯＤＥＭ［５］、ＦＡＮ［６］等のテストパターン生成アルゴリズムが発表されており、またこれらのアルゴリズムをベースにＣＭＯＳ回路のトライステート系素子も扱えるように拡張したアルゴリズムも発表されている［７、８］。

テスト容易化設計の一つであるスキャン設計［１］を適用した論理回路に対しては、スキャンパスで囲まれた部分は組合せ回路と見なすことができるため、上に挙げたアルゴリズムにより、テストパターンの自動生成が可能であった。

これに対し、レジスタファイルを含むような論理回路は、ゲート数（チップ面積）の点から、スキャン設計を適用することは非現実的である。このような回路に対して、従来は人手でテストパターンを作成し、故障シミュレーションを行っていたが、故障検出率の高いテストパターンを作成するために多くの時間を要していた。

そこで、このような回路に対するテストデータ生成を自動化するため、クロック系の故障も考慮した新たなテストパターン生成アルゴリズムを開発した。

ここで述べるアルゴリズムは、基本的には組合せ回路用のテストパターン生成アルゴリズムであるＰＯＤＥＭを時間軸に対して展開したものである。従って、組合せ回路においては１つの故障を検出するテストパターンは１パターンであったのに対し、本アルゴリズムでは数パターンから成るテストパターン系列を自動生成する。さらに、故障によって記憶素子にクロックが入らなくなるような故障も検出できるテストパターンも生成可能である。

（１）固有初期値

ＡＬＴＥＳ／ＲＡのテストパターン・ジェネレータでは、表３に示す信号値を扱う。Ｘｎ及び$\overline{Xn}$は、ここで導入した固有初期値とその補数を表す信号値であり、故障によって初期化できなくなったＤラッチの固有識別番号ｎとともに表される。このＸｎ、$\overline{Xn}$は故障回路にしか現れない信号値で、不定値の一種ではあるが、Ｄラッチｎに固有の信号値であり、各ゲートの真理値表上では、０、１に準じた扱いをする。表４にＡＮＤ、ＯＲ、ＮＯＴ及びＤラッチの真理値表を示す。

表３．信号値

シンボル	意味
0	信号値０（Ｌｏｗ電圧）
1	信号値１（Ｈｉｇｈ電圧）
Z	ハイ・インピーダンス
X	不定値
C	クロック（ポジティブ・パルス）
$\overline{C}$	クロック（ネガティブ・パルス）
Ｘｎ	識別番号ｎの記憶素子の固有初期値
$\overline{Xn}$	Ｘｎの補数

（２）テストパターン生成アルゴリズム

テストパターン生成アルゴリズムのフローを図６に示す。基本的にはＰＯＤＥＭ法を時間展開したアルゴリズムとなっている。すなわち、回路内部にＤラッチを含むため、初期目的値からの外部入力端子と信号値の選択（図６ｂ、ｄ）及び故障信号を伝搬させるための初期目的値の選択（図６ｃ）において時間軸方向の操作が拡張されている。

●外部入力端子と信号値の選択（図６ｂ、ｄ）

故障の存在する信号線の正常信号値がまだ不定値Ｘの場合は故障の縮退値と逆の値が、または故障信号を伝搬させるような素子の入力値が、それぞれ初期目的値として選ばれる（図６ａ、ｃ）。

ＰＯＤＥＭ法では、通常、この初期目的値から外部入力端子に向けて不定値Ｘの経路をバックトレースし、到達した外部入力端子とその時の目的値が設定値として得られる。この際、各素子タイプに対して以下に示すように目的値を変化させながらバックトレースする。

ＡＮＤゲート

ｉ）出力信号線の目的値が０の場合、不定値である入力信号線のうち最も０に制御容易な信号線に対して目的値０でバックトレースする。

ii）出力信号線の目的値が１の場合、不定値である入力信号線のいずれか１つに対して目的値１でバックトレースする。

ＯＲゲート

ｉ）出力信号線の目的値が０の場合、不定値である入力信号線のいずれか１つに対して目的値０でバックトレースする。

ii）出力信号線の目的値が１の場合、不定値である入力信号線のうち最も１に制御容易な信号線に対して目的値１でバックトレースする。

表４．真理値表

ＡＮＤゲート

I_1 \ I_2	0	1	Z	X	C	$\overline{C}$	X_n	$\overline{X_n}$	X_m	$\overline{X_m}$
0	0	0	0	0	0	0	0	0	——	
1	0	1	E	X	C	$\overline{C}$	X_n	$\overline{X_n}$	——	
Z	0	E	E	E	E	E	E	E	——	
X	0	X	E	X	X	X	X	X	——	
C	0	C	E	X	C	E	X	X	——	
$\overline{C}$	0	$\overline{C}$	E	X	E	$\overline{C}$	X	X	——	
X_n	0	X_n	E	X	X	X	X_n	0	X	X
$\overline{X_n}$	0	$\overline{X_n}$	E	X	X	X	0	$\overline{X_n}$	X	X

但し $n \neq m$

ＯＲゲート

I_1 \ I_2	0	1	Z	X	C	$\overline{C}$	X_n	$\overline{X_n}$	X_m	$\overline{X_m}$
0	0	1	E	X	C	$\overline{C}$	X_n	$\overline{X_n}$	——	
1	1	1	1	1	1	1	1	1	——	
Z	E	1	E	E	E	E	E	E	——	
X	X	1	E	X	X	X	X	X	——	
C	C	1	E	X	C	E	X	X	——	
$\overline{C}$	$\overline{C}$	1	E	X	E	$\overline{C}$	X	X	——	
X_n	X_n	1	E	X	X	X	X_n	1	X	X
$\overline{X_n}$	$\overline{X_n}$	1	E	X	X	X	1	$\overline{X_n}$	X	X

但し $n \neq m$

ＮＯＴゲート

I	O
0	1
1	0
Z	E
X	X
C	$\overline{C}$
$\overline{C}$	C
X_n	$\overline{X_n}$
$\overline{X_n}$	X_n

Ｄラッチ

E \ D	0	1	Z	X	X_n	$\overline{X_n}$
0	Q	Q	Q	Q	Q	Q
1	0	1	E	X	X_n	$\overline{X_n}$
Z	E	E	E	E	E	E
X	X	X	E	X	X	X
C	0	1	E	X	X_n	$\overline{X_n}$
$\overline{C}$	E	E	E	E	E	E

但しＱは１時刻前の出力

Ｅはエラー（禁止）

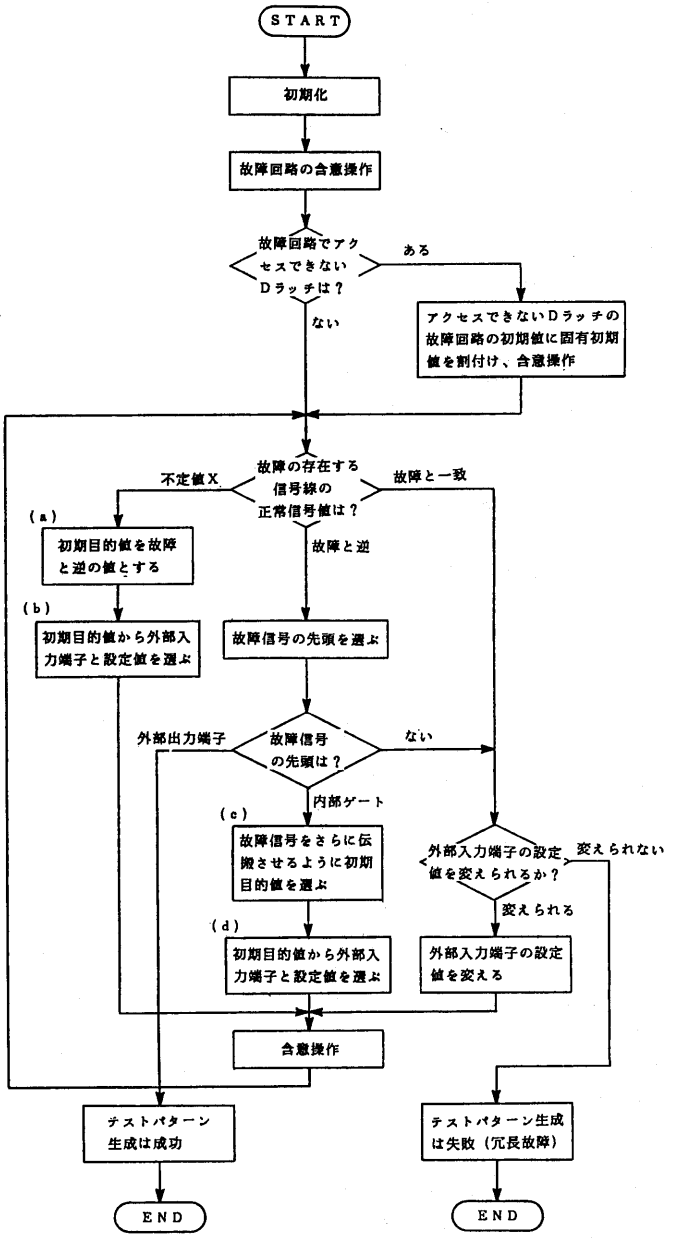

図６．フローチャート

ＮＯＴゲート

入力信号線に対して、出力信号線の目的値を反転させてバックトレースする。

Ｄラッチ

ｉ）クロック入力が不定値Ｘの場合、クロック入力信号線に対して次の目的値をＣとしてバックトレースする（図７ａ）。

ⅱ）クロック入力が信号値Ｃの場合、データ入力信号線に対して出力信号線の目的値をそのまま次の目的値としてバックトレースする（図７ｂ）。

ⅲ）クロック入力が信号値０の場合、１周期前の同一Ｄラッチの出力信号線に対し目的値をそのままとしてバックトレースする（図７ｃ）。

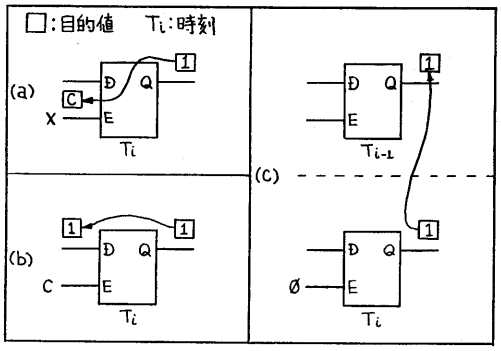

図７．バックトレース（Ｄラッチ）

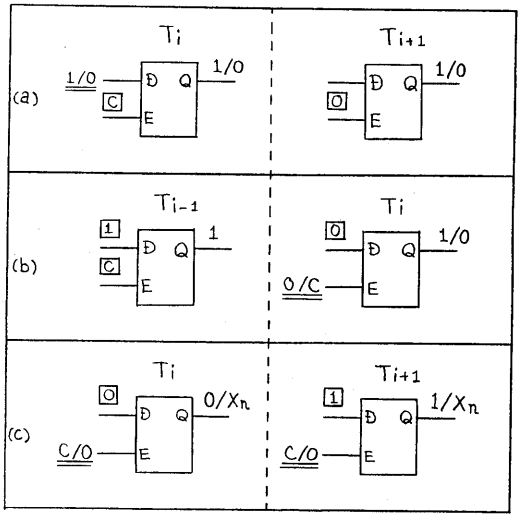

図８．故障信号伝搬のための初期目的値

●故障信号伝搬のための初期目的値の選択（図６ｃ）

故障信号がある素子の入力に伝搬した場合、さらにそれを出力に伝搬させるためには、その素子の他の入力に信号値を設定しなければならない。故障信号が素子の入力に伝搬した場合、各素子タイプに対して他の入力にどのような初期目的値を選択するかを次に示す。

ＡＮＤゲート

他のすべての入力信号線に対し、初期目的値１を選ぶ。

ＯＲゲート

他のすべての入力信号線に対し、初期目的値０を選ぶ。

Ｄラッチ

Ｄラッチに対しては次の３通りの故障信号があり、それぞれ図２の故障①、②、③に対応する。

（ｉ）故障信号がデータ入力に伝搬した場合、クロック入力信号線に初期目的値Ｃを選択する（図８ａ）。また、この時刻内で故障信号が外部出力端子まで伝搬できない場合は、次の時刻でそのＤラッチのクロック入力信号線に初期目的値０を選択する。この初期目的値が満たされた場合は、故障信号が次の時刻まで保持される。保持された故障信号を次の時刻以降で外部出力端子まで伝搬させるように試みる。

（ii）故障信号０／Ｃがクロック入力に伝搬した場合、通常１時刻前では正常回路、故障回路ともにクロック入力の信号線をＣにすることができる。従って、１時刻前で正常回路、故障回路ともにＤラッチを初期化し、故障信号の伝搬した時刻でＤラッチのデータ入力に前の時刻で初期化した値と異なるように初期目的値を選ぶ（図８ｂ）。この目的値が満たされれば、正常回路でのＤラッチの出力はクロックが入らないため１時刻前で初期化された値を保持し、故障回路ではクロックが入るため現時刻でのデータ入力の値を取り込んでしまう。従って、正常回路と故障回路で出力値が異なるため、新たな故障信号となって伝搬する。

（iii）故障信号Ｃ／０がクロック入力に伝搬した場合、故障回路においてはＤラッチにクロックを入れることができず、初期化ができなくなる。しかし、故障によって内部状態を初期化できないＤラッチに対しては、故障回路の初期値として固有初期値Ｘｎ（ｎはＤラッチの固有識別番号）が与えられている。そこで、正常回路での信号値０とＸｎ及び正常回路での信号値１とＸｎの２つの故障信号０／Ｘｎ、１／Ｘｎを新たな故障信号として、異なる時刻で外部出力端子まで伝搬させるようにする（図８ｃ）。実際の回路でこのような故障があった場合、Ｘｎは０または１のどちらであるかはわからないが、少なくともどちらかの値が変化せずに保持されていると考えられるので、外部出力端子まで伝搬した０／Ｘｎ、１／Ｘｎのどちらかが実際の故障の影響となるので、このような故障に対してもテストパターンが生成可能となる。

５．２　故障シミュレータ

生成したテストパターンで同時に検出される故障を除去するため、故障シミュレータを併用する。ＡＬＴＥＳ／ＲＡで用いる故障シミュレータは、コンカレント方式［９］に、先に述べた固有初期値Ｘｎを扱えるように拡張してある。すなわち、通常は外部出力端子に０／１や１／０のように正常回路と故障回路で異なる信号値で伝搬した故障は検出できたとして除去するが、それに加えて、０／Ｘｎ、１／Ｘｎの両方の信号値が外部出力端子まで伝搬した場合にも、その故障を検出できたとして除去するものである。

表５．実行結果（６．５ＭＩＰＳ計算機）

回路	ゲート数	Ｄラッチ数	故障数	故障検出率	パターン数	実行時間
＃１	９４１	６４	１５１３	１００％（９４％）	２４０	３３４sec
＃２	１３８７	１２８	２１９６	１００％（９０％）	２２４	６１７sec

（ ）内は固有初期値を用いない場合

６．実行結果

ＡＬＴＥＳ／ＲＡをレジスタファイルを含む２つの論理回路に適用した結果を表５に示す。２つの回路に対し、固有初期値を用いることにより故障検出率１００％のテストデータが自動生成された。固有初期値を用いない場合、故障検出率は９０～９４％までしか上がらず、固有初期値の有効性が確かめられた。

また、実行時間も５分から１０分（６．５ＭＩＰＳ計算機）程度で、人手でテストパターンを作成するのに比べ、はるかに高速にテストデータが自動生成されるようになった。

７．不完全スキャン設計への適用

ＡＬＴＥＳ／ＲＡのテストパターン・ジェネレータは、レジスタファイルのみをテストパターンの対象として限定されるものではなく、実際にはレジスタへのフィードバック・ループがなければ、多段であってよい。

現在、テスト容易化設計として実用化されているスキャン設計には、全記憶素子をスキャンパスに含める完全スキャン設計と、記憶素子の一部はスキャンパスに含めない不完全スキャン設計［１０］とがある。

ＡＬＴＥＳ／ＲＡの開発により、ＶＬＳＩのような大規模回路において、レジスタへのフィードバック・ループがなくなるようにスキャンパスを構成する不完全スキャン設計を適用することが可能となる。例えば、本報告で示したレジスタファイル部分はスキャンパスには含めず、周囲をスキャンパスで囲むだけでよい。これにより、チップ面積の増加がかなり抑えられ、またスキャンパスで囲まれた部分は、ＡＬＴＥＳ／ＲＡによってテストデータの自動生成が可能である。

現在、ＡＬＴＥＳ／ＲＡを不完全スキャン設計回路にも適用し、その評価を行っている。

８．まとめ

本報告では、レジスタファイルを含む論理回路のテストパターン自動生成プログラムＡＬＴＥＳ／ＲＡについて述べた。故障回路で初期化できなくなる記憶素子の初期値に固有初期値を導入することにより、クロック系も含め、高品質のテストデータが自動生成可能となった。

また、最後にＡＬＴＥＳ／ＲＡの不完全スキャン設計への応用も示した。

今後は、不完全スキャン設計回路も含め、より大規模な回路への適用評価を行うとともに、レジスタへのフィードバック・ループのある回路についても自動生成の検討を行う予定である。

参考文献

(1) T. W. Wiliams and K. P. Parker, "Design for Testability - A Servey", IEEE Trans. on Computer, Vol. C-31, pp. 2-15, Jan. 1982.

(2) L. H. Goldstein, "Controllability/Observability Analysis of Digital circuits", IEEE Trans. on Circuits and Systems, Vol. CAS-26, No. 9, pp. 685-693, Sptember 1979.

(3) 下野、藤原、尾崎、" 可検査性尺度に基づく検査入力生成効率の改善法"、情処学会　電子装置設計技術研究会１３－４、１９８２年２月.

(4) J. P. Roth, "Diagnosis of Automata Failures : A calculus and a method", IBM Res. and Develop., Vol. 10, pp. 278-291, July 1966.

(5) P. Goel, "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits", IEEE Trans. on Computer, Vol. C-30, No. 3, pp. 215-222, March 1981.

(6) H. Fujiwara and T. Shimono, "On the Acceleration of Test Generation Algorithms", IEEE Trans. on Computer, Vol. C-32, No. 12, pp. 1137-1144, Dec. 1983.

(7) T. Ogihara, et. al., "Test Generation for Scan Design Circuits with Tri-state Modules and Bidirectional Terminals", 20th DAC, pp. 71-78, June 1983.

(8) N. Itazaki and K. Kinoshita, "Test Pattern Generation for Circuits with Three-state Modules by Improved Z-algorithm", 1986 ITC, pp. 105-112, Sep. 1986.

(9) E. G. Ulrich and T. Baker, "The Concurrent Simulation of Nearly Identical Digital Networks", Proc. 10th Design Automation Workshop, pp. 145-150, June 1973.

(10) E. Trischler, "Incomplete Scan Path with an Automatic Test Generation Methodology", 1980 ITC, pp. 153-162, 1980.