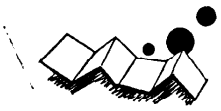
Vol. 22 No. 8　　情　報　処　理　　Aug. 1981

解　説



論理回路の試験，診断†

山田昭彦†† 舩津重宏†† 黒部恒夫†††

1. はじめに

論理回路の試験，診断が，CAD あるいは DA (Design Automation) の対象として取り上げられたのは，1960 年頃からと思われるが[1),2)] その後のデジタル・システムの急速な普及に伴い，信頼性の確保のためにますます重要な分野となってきている．特に VLSI のように，集積度（素子の数）が10万個以上の回路を対象とする場合には，設計の質および量の問題に対処することのほかに，製造された回路の検査が重要な問題となってくる．すなわち製造された回路が設計意図通りに電気的条件，論理的条件で動作するかどうか（試験），もし不良であるならば，どの箇所がどのように悪いか見付ける（診断）ことを効率良く行わなければならない．これら試験，診断の効率が，VLSI の実用上のリミットを抑えてしまうことも十分考えられるわけである[3)]．

一般的に論理回路の試験，診断として CAD の対象とされるのは，機能試験である．デバイスの検査としてはほかにも直流パラメトリック試験，交流動作特性試験等があるが，ここでは機能試験のみを議論の対象とする．回路の機能試験は，その入力端子（プライマリ入力）に一連の刺激パターンを印加し，その応答を出力端子（プライマリ出力）で観測することにより行われる．プライマリ入力に印加される一連の刺激パターンはテスト・パターンと呼ばれ，その応答はシンプトンと呼ばれる．VLSI の場合，このプライマリ入力の数が 100～200 位となることも，めずらしくないと思われる．これらのプライマリ入力に，すべての組み合わせのテスト・パターンを網羅的に印加することは現実的ではない．すなわち，できるだけ少ないテスト・パターン数で回路の論理機能を試験してやることが必要である．これがテスト・パターン発生技術である．また試験の結果得られるシンプトンから，回路内に故障が存在する場合，その故障箇所を指摘してやることも必要である．これは故障診断技術と呼ばれる．

VLSI の場合には，上述のような技術が適用しやすいように回路設計を行っておくこともまた重要な要素となってきている．特に最近のようにデジタル・システムが社会の中枢神経をはたすようになってくると，システムの中心となる VLSI の超高信頼化への要求もますます強くなるものと思われるが，そのためにもまず回路自身が試験容易なように設計されている必要がある．これを試験容易化技術と呼ぶ．

2. テスト・パターン発生技術

テスト・パターン発生技術を年代を追って眺めてみると（表-1 参照），1970 年代初めまでに，その基本と

表-1　テスト・パターン発生技術年表

1962	パラレル故障シミュレーション (2)
1966	パス活性化法 (4) D アルゴリズム (5)
1968	D アルゴリズムの順序回路への拡張 (6) ブール微分法 (7)
1970	スキャン・パステスト方式 (8), (23)
1972	ディダクティブ故障シミュレーション (9) TEGAS 2 システム (10)
1973	コンカレント故障シミュレーション (11) D-LASAR システム (12)
1974	LAMP システム (13)
1977	LSSD テスト方式 (14)
1979	ファンクショナルレベルテスト生成法 (15)
1980 年	VOTE システム (16)

† Test and Diagnosis of Logic Circuits by Akihiko YAMADA, Shigehiro FUNATSU (Computer Engineering Division, Nippon Electric Co., Ltd.) and Tsuneo KUROBE (IC Division, Nippon Electric Co., Ltd.).

†† 日本電気(株)コンピュータ技術本部
††† 日本電気(株)集積回路事業部

なる手法のほとんどが確立されていることがわかる．そして以来現在に至るまで，それら手法を改良しあるいはそのまま用いて，多くのシステムが作られ続けている．いいかえると，この十年ほどの間はテスト・パターン発生技術に大きな革新はなかったことになる．しかし，VLSI の出現によって，現在の CAD システムの限界が浮きぼりにされ，単に従来技術を改良するだけでは，この限界を突き破るのが非常に困難であることが明らかとなり，テスト・パターン発生技術の新たな飛躍が望まれている．

2.1 故障モデル

テスト・パターン自動発生および故障シミュレーションは，たいていの場合あらかじめ設定された故障を対象として行われる．現実に発生する故障は様々であるが，計算機処理をするためには何らかのモデルが必要であり，単一スタック故障がよく用いられる．

単一スタック故障モデルでは，故障はゲートまたは機能ブロックの入力あるいは出力が，0または1に縮退した状態として定義され，同時に2個以上の故障が回路に存在することはない．

このモデルがよく用いられるのは，モデル自身が単純でプログラミング上扱いやすいことに加え，ゲートの入出力のスタック故障のほとんどを検出するテスト・パターンは，実際に回路内で発生する故障もほとんど検出できることが経験上知られているためである．ほかに，オープン，ショート，多重スタック故障が主に単一スタック故障を補う目的で用いられる．

2.2 テスト・パターン自動発生

テスト・パターン自動発生手法の代表的なものには，パス活性化法，Dアルゴリズム，ブール微分法などがある．これらの手法は，仮定された故障を出発点として，正常回路と故障回路が異なる動作をするようにプライマリ入力を定め，同時にその相違がプライマリ出力で観測できるようにすることからなっている．これら手法のうち，最もよく使われるDアルゴリズムを簡単な例（図-1）で説明する．

Dアルゴリズムでは，D値という特殊な値を用いる．正常回路の値／故障回路の値で示すと，1／0は D，0／1は $\bar{D}$ である．

ゲートBの出力にスタックアット0故障が仮定されている（図-1(a)）．正常回路と故障回路とが異なる動作をするためには，ゲートBの出力値はDとなる必要があり，ゲートBの入力は全て1とならねばならない（図-1(b)）．また，この D 値をプライマリ出力へ伝搬させるには，ゲートEの第2入力が0，ゲートFの第2入力が1でなければならない（図-1(c)）．これらゲートB，E，Fの各入力値を確定できるようにプライマリ入力の値を決める．この時，ゲートEの第2入力を0とするためにI4を0とすると，ゲートFの第2入力を1とすることに失敗するので，I3を0とする．この例ではほかに二者択一的要素はなく，プライマリ入力の値が定まって，テスト・パターン発生に成功する（図-1(d)）．

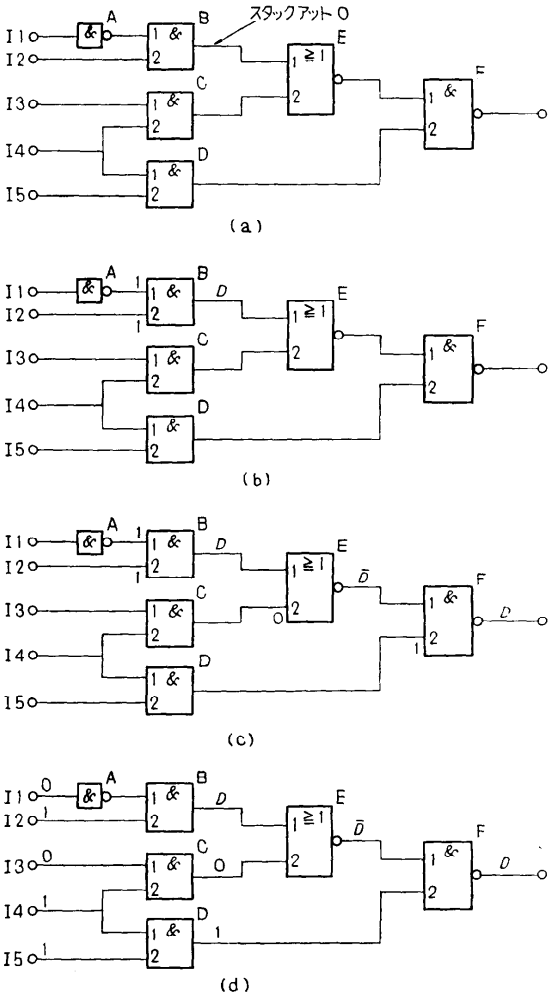


図-1　テスト・パターン発生手法（D アルゴリズム）

2.3 故障シミュレーション

故障シミュレーションは，与えられたテスト・パターンに対して，回路内に仮定した全ての故障の振舞いをシミュレートし，プライマリ出力上での正常回路との相違を明らかにし，テスト・パターンの有効性の検証および故障診断用の辞書を作成するために使われる．

最も単純な故障シミュレーションの方法は，正常回路と各故障回路ごとの論理シミュレーションを行うことであるが，経済的ではないのでパラレル，ディダクティブ，コンカレントの3手法がよく用いられる．そのうち最もよく用いられているのはパラレル法で，これは計算機のワードを利用してブール演算を行うことによってスピードをあげている．たとえば1ワード32ビットの計算機で1度に10ワードのブール演算を行えば，319個の故障回路と1個の正常回路を1度にシミュレートできる．

回路規模が増大すると，コンカレント法がCPU時間で優位にたつため，VLSIにはコンカレント法を適用するところが増えると思われる．この方法では，故障の影響で正常回路と異なる状態が生じない限り，故障に関する演算を行わないように工夫している．

2.4 テスト・パターン発生技術に与えるVLSIのインパクト

VLSI化，すなわち回路の複雑化と回路規模の増大に伴い，テスト・パターン発生技術は重大な影響をうける．現存するテスト・パターン自動発生技術は，もともと組み合わせ回路用であり，順序回路は時間軸をもとに展開して，擬似的に組み合わせ回路として扱えるようにしており，テスト・パターン自動発生中の展開後のゲート数は回路の複雑さに比例して増加する．

回路規模の増大はテスト・パターン自動発生，故障シミュレーションの両方にCPU時間に関して影響を与える．たとえばゲート数がn倍になると，扱うスタック故障数もほぼn倍となり，テスト・パターン自動発生の対象規模はn^2倍ということになる．さらにテスト・パターン数もn倍になるとすれば，故障シミュレーションの対象規模はn^3倍になる．そこで，CPU時間をゲート数Gの関数として見積もってみると，パラレル故障シミュレーションは単純に対象規模に比例すると考えられるので，そのCPU時間はほぼG^3の関数となる．ただし，コンカレント法では故障回路が演算されるのは，正常回路と異なる場合のみであるので，ほぼG^2の関数であると推定される．テスト・パターンの自動発生は複雑な組み合わせ問題と考えられており，そのCPU時間は最悪ケースでGの指数関数，回路が組み合わせ回路で最もうまくいったケースでもG^2の関数であると思われる．これは千ゲート回路のテスト発生にCPUを1時間使ったとすると，10万ゲート回路では1万時間のCPUの使用が必要となることを示している．

上記のことよりわかるように，現存のテスト・パターン発生技術は大きなカベに突き当たっている．ファンクショナルレベルでテストを生成する試み等が行われているが，残念なことにこれを打ち破るような画期的な手法が現われる徴候は，今のところ見あたらない．代わりに必要とされるのは，VLSIの設計がテスト・パターン発生技術の限界を考慮してなされることである．これは試験容易化技術に関する問題であり詳細は後述するが，そのめざすところは，テスト・パターン自動発生と故障シミュレーションに要するCPU時間をゲート数に比例するようにすることである．

3. 故障診断技術

試験の結果得られるシンプトンより，回路内に故障が存在する場合その故障個所を探し出さなくてはならない．VLSIのように回路内に多数(10万素子以上)のトランジスタを実装し，しかもその内部状態をプロービング等の手段によりアクセスできない場合には，シンプトンのみが故障診断の唯一の手掛かりとなる．

3.1 故障辞書方式[17]

すでに述べたように，故障シミュレーションにより回路内のあらかじめ想定した故障についてテスト・パターンに対応するシンプトンが得られている．印加されたテスト・パターンに対する，各想定故障に対応したシンプトンを見出しとして編集したテーブルを故障辞書と呼び，このテーブルを参照した診断方式を，故障辞書方式と呼ぶ．

いま診断対象回路Mが3ビットのプライマリ出力を持つものとして，Mに4つのテスト・パターンT_1〜T_4を印加した結果，得られたシンプトンがそれぞれ

$(010)_{T_1}$
$(101)_{T_2}$
$(111)_{T_3}$
$(000)_{T_4}$

であったとする．この場合，故障シミュレーションにより表-2に示すような故障辞書が得られていたとすれば，シンプトンどうしの比較により故障M_{F_2}と診

表-2 故障辞書

シンプトン				故障
T_1	T_2	T_3	T_4	
(010)	(001)	(110)	(000)	Mφ (故障なし)
(110)	(001)	(110)	(000)	M_{F_1}
(010)	(101)	(111)	(000)	M_{F_2}
(110)	(101)	(110)	(000)	M_{F_3}
(010)	(011)	(010)	(000)	M_{F_4}
(010)	(001)	(010)	(001)	M_{F_5}

断される．

故障辞書方式の場合には，シンプトンのデータ量が問題となる．特にVLSIのようにシンプトンのビット数とテスト・パターン数が増大する場合には，シンプトンの圧縮をはかることが有効である[18]．圧縮は一般的にマッピングを用いて行われ，マッピングによるシンプトン圧縮の程度は，同類項（シノニム）の発生する確率との兼ね合いにより決定される．なおシンプトンの圧縮を行った場合には，試験を実行した結果得られるシンプトンに対しても同様の圧縮操作を行ってやる必要がある．

このほかにも故障辞書方式として，シンプトンの代わりに，各テスト・パターンのパス／フェイルを見出しとして用いる手法もあるが，シンプトンを用いる手法に比べデータ量の点では有利であるが，診断分解能が悪化する欠点がある．（たとえば表-2の例では，診断結果として M_{F_2} のほかに M_{F_4} が指摘されてくる．）

故障辞書方式をVLSIに適用する場合の問題点として，以下の4点が挙げられる．

1) 一般に故障辞書作成に時間がかかり過ぎる．

2) 単一スタック故障に対する故障辞書作成が実用上の限界である．

3) 診断データ量が多大となる．

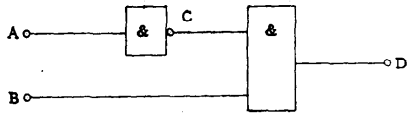
4) デバイスの製造プロセスが安定していないと，誤診断を行う危険がある．

これらの問題を解決するためには，後述のパーティショニング技術，演繹方式を併用することが考えられる．

3.2 演繹方式[19]

この方式は故障辞書方式とは異なり，試験結果より回路内部の信号線が正常に動作しているか否かを演繹してゆく．図-2を用いてそのアルゴリズムの概要を示す．

（a） 診断対象回路を示す．



(a)

	A	B	C	D
T_1	0	1		1
T_2	1	1		0
T_3	0	0		1

(b)

	A	B	C	D
T_1		1	1	1
T_2				0
T_3				1

(c)

	A	B	C	D
T_1		1	1	1
T_2		1		0
T_3				1

(d)

	A	B	C	D
T_1		1	1	1
T_2		1	0	0
T_3				1

(e)

	A	B	C	D
T_1	0	1	1	1
T_2	1	1	0	0
T_3				1

(f)

	A	B	C	D
T_1	0	1	1	1
T_2	1	1	0	0
T_3	0			1

(g)

	A	B	C	D
T_1	0	1	1	1
T_2	1	1	0	0
T_3	0		1	1

(h)

	A	B	C	D
T_1	0	1	1	1
T_2	1	1	0	0
T_3	0	1	1	1

(i)

図-2 演繹方式のアルゴリズム

（b） 試験結果を示す．プライマリ出力Dは論理値0，1を示しているので故障ではない．

（c） Dは正常である．したがってテスト・パターン T_1 での入力信号 B=C=1．（水平形インプリケーション）

（d） B はプライマリ入力で，T_1 で B=1 の論理値をとっている．したがって T_2 でも B=1 となり得る．（垂直形インプリケーション）

（e） D が正常であり，T_2 で D=0，B=1．したがって T_2 では C=0 である．

（f） C は論理値0，1を示しているので正常である．これより A の値が定まる．

（g） A はプライマリ入力であり，T_1 で A=0．したがって T_3 でも A=0 となり得る．

（h） C は正常であるから，T_3 で C=1．

（i） T_3 で D=1 となっているので，$B \neg C=1$．C=1 は(8)で求められている．したがって B の故障と診断．

演繹方式の利点として，以下の2点が挙げられる．

1） 故障辞書作成が不要である．

2） 単一スタック故障以外の故障も診断可能である．（回路の初期化が不能となるような故障でも，診断可能である．）

本方式をそのまま VLSI に適用するには，手順の複雑さ（おそらくテスト・パターン発生技術が直面している問題の複雑さと同程度の複雑さを持つものと思われる．），診断分解能等の点で問題があると思われるが，すでに述べた方式および後述の試験容易化技術との併用により，回路規模の増大に比例した手順の増加で済むような診断技術の展開が期待される．

4. 試験容易化技術

すでに述べたように，VLSI に対する試験を効率よく行うためには，現在提案されているテスト・パターン発生プログラムや故障シミュレーション・プログラム等の能力向上には限界があり，試験対象回路が上述の CAD ツールを用いて試験容易なように設計されていることが必須の条件となってきている．これら試験容易化設計技術は今後ますます発展することが期待されているが，ここでは現在実用化されている技術を紹介することにする．

試験容易化設計を有効に実施するためには，設計された回路が試験しやすいか否かを簡単に判定し，もし試験し難いと判定されたならば，どの部分をどのように変更すれば試験容易な回路となるのかという情報が，回路設計段階で設計者にフィード・バックされることが必要である．一般に試験容易性（テスタビリティ）は，可制御性（コントローラビリティ）と可観測性（オブザーバビリティ）の関数と考えることができ，それぞれ以下のように定義することができる[20]．

- 可制御性

  回路Sのプライマリ入力を励起することで，S内のサブ回路 S(i) の入力にテスト・パターンを印加できる容易さ．

- 可観測性

  回路Sのプライマリ出力の観測により，S内のサブ回路 S(i) の応答が決定される容易さ．

これら可制御性，可観測性が定量的に測定できれば，特定な回路に関する試験容易度の判定が行えることになる．

4.1 試験容易度の判定

現在までに実用化されている試験容易度判定プログラムで，よく知られているものとして TMEAS[21]，SCOAP[22] の2つが挙げられるが，ここでは SCOAP の手法について紹介する．

対象回路は標準的なセル（たとえば AND，OR 等）で構成されるものとし，回路内の各ノードに対して，回路トポロジーからのみ決定される以下の6つの試験容易度の尺度を設定する．

CC^0(N)：ノード N の組み合わせ的0可制御性
CC^1(N)：〃 1可制御性
CO(N)：〃 可観測性
SC^0(N)：ノードNの順序的0可制御性
SC^1(N)：〃 1可制御性
SO(N)：〃 可観測性

更に各標準セルにはセルの深度というものを定義しておく．

組み合わせ的セル深度：組み合わせセル（AND，OR 等）は1．順序セル（フリップ・フロップ等）は0．

順序的セル深度：組み合わせセルは0．順序セルは1．

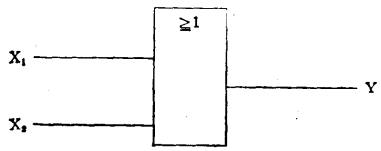


図-3 2入力 OR ゲート

以上のように定義したセル深度を用いて，各種標準セルの入出力ノード間に関する試験容易度の尺度の関係を求めておく．たとえば 図-3 に示すような 2 入力 OR ゲートに関する入出力ノード間の尺度関係は以下のように求められる．

$$CC^0(Y)=CC^0(X_1)+CC^0(X_2)+1$$
$$CC^1(Y)=\mathrm{Min}\,[CC^1(X_1), CC^1(X_2)]+1$$
$$SC^0(Y)=SC^0(X_1)+SC^0(X_2)$$
$$SC^1(Y)=\mathrm{Min}[SC^1(X_1), SC^1(X_2)]$$
$$CO(X_i)=CO(Y)+CC^0(X_{3-i})+1 \quad i\in\{1,2\}$$
$$SO(X_i)=SO(Y)+SC^0(X_{3-i}) \quad i\in\{1,2\}$$

このように各種標準セルに関する入出力ノード間の関係式が得られたならば，対象回路内のすべてのノードに対する各種尺度は以下の手順により求められる．

● 可制御性尺度の計算

1° 個々のプライマリ入力ノード I について

$$CC^0(I)=CC^1(I)=1$$
$$SC^0(I)=SC^1(I)=0$$

そのほかのすべてのノード N について

$$CC^0(N)=CC^1(N)=SC^0(N)=SC^1(N)=\infty$$

2° 回路内のすべてのセルについて，入出力ノード間の尺度関係式を用いて可制御性尺度が定常値となるまで計算．

● 可観測性尺度の計算

1° 個々のプライマリ出力ノード U について

$$CO(U)=SO(U)=0$$

そのほかのすべてのノード N について

$$CO(N)=SO(N)=\infty$$

2° 回路内のすべてのセルについて，入出力ノード間の尺度関係式を用いて可観測性尺度が定常値となるまで計算．

以上のようにして求められた尺度により，回路内のどの部分(どの標準セル)が試験困難であるのかを知ることができる．

以上 SCOAP の手法についてその概要を紹介したが，一般に現在実用化されている試験容易度判定プログラムは，以下のような欠点を含んでいる．

1) 尺度が相対的なものであり，絶対的な尺度としては使用できない．すなわち設計へのフィード・バックに限界がある．

2) 試験を困難としている真の原因を指摘することがむつかしい．

しかしながら，これらの欠点も実用化をはかってゆくなかで，経験的に解消されることも可能と考えられるので，今後の展開が期待される．

4.2 試験容易化設計

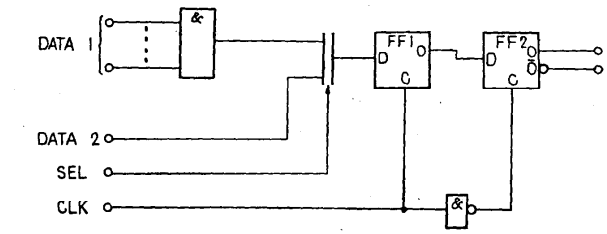
現在実用化されている試験容易化設計技術のうち，代表的なものとしてスキャン・パス方式[8),14),23)]を挙げることができる．回路の可制御性，可観測性の向上が試験容易化設計の要点となっていることはすでに述べた．VLSI 化回路に適した可制御性，可観測性の向上手段として，一般に以下の制約を考慮しなければならない．

1) 可制御性，可観測性の向上のための付加回路量はできるだけ少量に押える．

2) プライマリ入出力の増加はできるだけ押える．

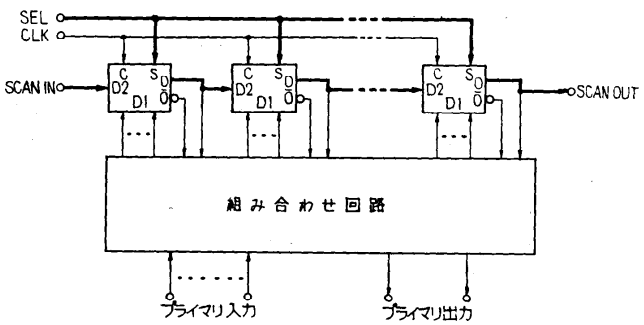
VLSI 化回路に対しては，特に 2) 項の制約が重要となる．これらの制約を満足する試験容易化設計技術として，スキャン・パス方式が注目されている．

図-4 にスキャン・パス方式を採用した回路の構成



FF 1: マスター・フリップ・フロップ
FF 2: スレーブ・フリップ・フロップ

(a)



(b)

図-4 スキャン・パス方式

を示す．図-4(a)は回路内に使用されるフリップ・フロップ回路を示し，図-4(b)は全体の回路構成を示している．図-4(a)よりわかるように，ここで使用されるフリップ・フロップは2系統のデータ入力(DATA 1, DATA 2) を備えており，選択信号 (SEL) により選択可能である．図-4(b) において，太線で示した部分が対象回路の可制御性，可観測性の向上のために付加された回路である．すなわち回路内のすべてのフリップ・フロップが，選択信号 (SEL) を制御することにより，スキャン・イン端子 (SCAN IN) を入力とし，スキャン・アウト端子 (SCAN OUT) を出力とするシリアル・シフト・レジスタとして動作するように構成されている．このように構成された回路は，以下の手順により，各フリップ・フロップの制御，観測がプライマリ入出力より直接実行できることになり，可制御性，可観測性が大幅に向上することになる．

1° 選択信号をシフト・レジスタ動作状態に設定．

2° スキャン・イン端子より各フリップ・フロップにテスト・パターンを印加．(スキャン・イン)

3° プライマリ入力にテスト・パターンを印加．

4° 選択信号を反転．

5° プライマリ出力での応答を観測．

6° クロック信号 (CLK) を1クロック印加．

7° 選択信号をシフト・レジスタ動作状態に設定．

8° スキャン・アウト端子より各フリップ・フロップの内容を順次読み出して観測．(スキャン・アウト)

スキャン・パス方式の採用による効果をまとめると，以下のようになる．

1) 可制御性，可観測性が大幅に向上する．

2) 付加回路量が少なくて済む．

3) 複雑な順序回路を簡単な組み合わせ回路として試験することができる．

4) 対象回路を適当な大きさのサブ回路に分割 (パーティショニング) して，各サブ回路ごとに試験することができる．

5) 設計者の経験に依存しない，均質の試験容易性を持つ回路を設計できる．

特に，3)，4)項はテスト・パターン発生プログラムの負荷を軽減するのに多大の貢献をしている．すなわちVLSIにより実現される複雑な順序回路のテスト・パターン発生を簡単な組み合わせ回路のテスト・パターン発生の問題に置き換えることが可能となり，更にスキャン・パスを構成するシリアル・シフト・レジスタを擬似的なプライマリ入力あるいはプライマリ出力として取り扱うことが可能となることにより，回路機能に関係なくこれらプライマリ入力／出力に囲まれた適当な大きさのサブ回路(パーティション)に分割してテスト・パターン発生等を行うことができるようになる．たとえばすでに述べたように千ゲート回路のテスト発生に CPU を1時間使っている場合，10万ゲート回路をスキャン・パス方式により100個の千ゲート・サブ回路に分割できるならば，CPU も100時間で済ませられることになり，テスト・パターン発生時間等をゲート数増加に比例させることができる．

このほか，最近になりスキャン・パス方式と符号解析法[24]を組み合わせて，回路内に試験機構を組み込んでしまう方式[25]が提案されている．符号解析法は，線形フィード・バック・シフト・レジスタを利用したシンプトン圧縮の一方法であるが，ここでは，スキャン・パス方式により入出力がレジスタで囲まれたサブ回路に対して，入力レジスタを擬似乱数発生器として用い，出力レジスタを符号解析器として用いることにより自己テストを行う方法が提案されている．この方式は故障診断には無力であるが，テスト・パターン発生をあらかじめ行っておく必要がなく，しかも実時間で試験が実行できるという点で今後の展開が注目されている．

5. おわりに

論理回路の試験，診断に関して，CAD 技術の現状および今後の方向について概観し，更に試験を容易にする回路設計技術について紹介した．これからのVLSI の設計，製造のためには，これらの技術がともに協力して効果的な CAD システムを構築してゆくことが必須と思われる．本稿が論理回路の試験，診断 CAD の理解の一助となれば幸いである．

参考文献

1) Eldred, R. D.: Test Routines Based on Symbolic Logic Statements, JACM, Vol. 1, No. 1, pp. 33-36 (1959).

2) Seshu, S. and Freeman, D. N.: The Diagnosis of Asynchronous Sequential Switching Systems, IRE Trans. on EC, EC-11, pp. 459-465 (1962).

3) たとえば大内淳義：VLSI 技術の情報処理へのインパクト，電子工業月報，Vol. 22, No. 12, pp. 2-18 (1980).

4) Armstrong, D. B. : On Finding a Nearly Minimal Set of Fault Detection Tests for Combinational Logic Nets, IEEE Trans. on

EC, Vol. EC-15, pp. 66-73 (Feb. 1966).
5) Roth, J. P.: Diagnosis of Automata Failures: A Calculus and a Method, IBM Journal of R & D, Vol. 10, pp. 278-291 (July 1966).
6) Kubo, H.: A Procedure for Generating Test Sequences to Detect Sequential Circuit Failures, NEC R & D, No. 12 (Oct. 1968).
7) Sellers, F. F., Hsiao, M. Y. and Bearnson, C. L.: Analyzing Errors with the Boolean Difference, IEEE Trans. on Computers, Vol. C-17, pp. 676-683 (July 1968).
8) Funatsu, S., Wakatsuki, N. and Arima, T.: Test Generation Systems in Japan, Proc. 12 th DA Conf., pp. 114-122 (June 1975).
9) Armstrong, D. B.: A Deductive Method for Simulating Fault in Logic Circuits, IEEE Trans. on Computers, Vol. C-21, pp. 464-471 (May 1972).
10) Szygenda, S. A.: TEGAS 2-Anatomy of a General Purpose Test Generation and Simulation System for Digital Logic, Proc. 9 th DA Workshop, pp. 116-127 (June 1972).
11) Ulrich, E. G. and Baker, T.: The Concurrent Simulation of Nearly Identical Digital Networks, Proc. DA Workshop, pp. 145-150 (June 1973).
12) D-LASAR User's Guide, University Computing Company (Nov. 1973).
13) Chang, H. Y.: LAMP: Logic Analyzer for Maintenance Planning, BSTJ, pp. 1431-1556 (Oct. 1974).
14) Eichelberger, E. B. and Williams, T. W.: A Logic Design Structure for LSI Testability, Proc. 14 th DA Conf., pp. 462-468 (June 1977).
15) Thatte, S. M. and Abraham, J. A.: User Testing of Microprocessors, Proc. COMPCON, pp. 108-114 (Spring 1979).
16) Ulrich, E., Lacy, D., Phillips, N., Tellier, J., Kearney, M., Elkind, T. and Beaven, R.: High-Speed Concurrent Fault Simulation witn Vectors and Scalars, Proc. 17 th DA Conf., pp. 374-380 (June 1980).
17) Chang, H. Y., Manning, E. G. and Metze, G.: Fault Diagnosis of Digital Systems, Wiley Interscience, N. Y. (1970).
18) Chang, H. Y. and Thomis, W. : Method of Interpreting Diagnostic Data for Locating Faults in Digital Machines, BSTJ, Vol. 46, No. 2, pp. 289-317 (1967).
19) Amramovici, M. and Breuer, M. A.: Multiple Fault Diagnosis in Combinational Circuits Based on an Effect-Cause Analysis, IEEE Trans. on C, Vol. C-79, pp. 451-460 (1980).
20) Hayes, J. P. and McCluskey, E. J.: Testability Considerations in Microprocessor-Based Design, Computer, pp. 17-26 (Mar. 1980).
21) Grason, J.: TMEAS: A Testability Measurement Program, Proc. of 16 th DA Conf., pp. 156-161 (June 1979).
22) Goldstein, L. H. : Controllability/Observability Analysis of Digital Circuits, IEEE Trans. CAS-26, pp. 685-693 (1979).
23) 小林，松江，柴：FLT に適したフリップフロップ回路，昭和43年電子通信学会全国大会，No. 892 (1968).
24) Nadig, H. J.: Signature Analysis-Concepts, Examples, and Guidelines, Hewlett Packard Journal, pp. 15-21 (May 1977).
25) Könemann, B., Mucha, J. and Zwiehoff, G.: Built-In Logic Block Observation Techniques, Test Conference, pp. 37-41 (Oct. 1979).

(昭和56年4月2日受付)