講　座

コンピュータの設計自動化（4）*

倉　地　　正**

8. 故障検査・診断設計

8.1 検査データ作成プログラム

電子計算機における自動検査データ作成・診断の技術は機能ICレベル，ICカードレベル及び装置レベルの3段階に適用される．いずれの場合にも検査データ作成は**図-22**に示す2つの処理の繰り返しによって行われ必要な基準に到達するまで続けられる．それぞれに使われる手順は**表-3**に示すような種々のものがあり，用途により使い分けられている．

8.2 自動テストベクタ作成

テストベクタの自動作成の手順には特定の故障に対してシステマチックにその検出入力パタンを導き出すものや不特定多数を対象としたもの等各種のものがある．以下代表的なものを紹介する．

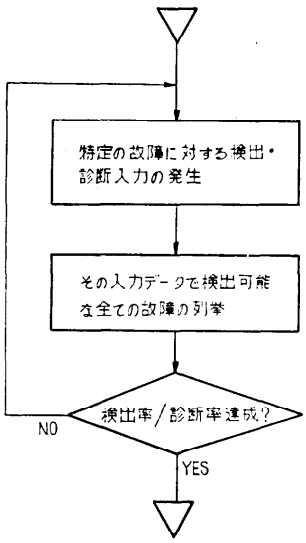


図-22　検査データ作成のフロー

8.2.1 乱数発生法

文字通り入力端子に対して乱数により値を与えるもので，どんな回路にも適用でき，時間も早いが論理回路の構造を考慮しないため検査が進むにつれ検出効率が悪くなってくる．実用システムでは他の方法と組み合わせて検出率の高い間使われることが多い．

8.2.2 パス活性化法[83]

この方法は論理回路の構造に着目してテストベクタを発生する最初に発表された手法であり，故障の影響を1本の径路（活性化径路）に沿って出力端子まで導く前方操作（forward trace）と，活性化径路の各信号を所定の状態にするのに必要な入力端子の値を決定する後方操作（backward trace）とで構成される．この方法はかなり有効でありIBMシステム/360のFLT作成に使用されたが，出力からの径路が途中で分岐し再収れんする場合（reconvergentfanout）には問題が

表-3　故障検査・診断用手法

(a)　テストベクタ発生法

手　順	速　度	到達可能検出率	多重故障処理	対　象
乱数発生法	早い	低	可	順序回路†
パス活性化法	かなり早い	中	不可	組合せ回路
Dアリゴリズム	遅い	高	不可	組合せ回路
ブール微分法	遅い	高	不可	組合せ回路
論理式シミュレーション法	中	高	可	非同期回路†
M0M1 アルゴリズム	中	中	不可	順序回路†
SPOOF	遅い	高	可	組合せ回路
9値モデル法	遅い	高	不可	順序回路†

† 組合せ回路も含む

(b)　検出可能故障導出法

手　順	速　度	必要メモリ容量	多重故障処理	対　象
並列故障シミュレーション	遅い	少	可	順序回路
ディダクティブシミュレーション	中	多	不可	順序回路
コンカレントシミュレーション	遅い	多	可	順序回路
TEST-DETECT	遅い	中	不可	組合せ回路
ハードウェアシミュレーション	早い	一	可	順序回路

* Design Automation of Computers by Tadashi KURACHI (Computer Hardware Development Dept., Ome works, Tokyo Shibaura Electric Co., Ltd.)
** 東京芝浦電気(株)青梅工場ハードウェア開発部

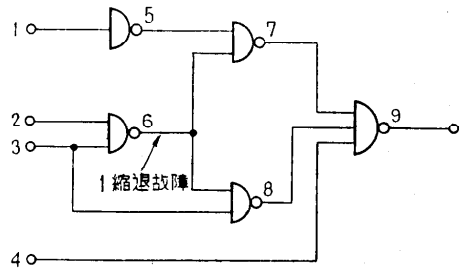
ある事が指摘されている．

8.2.3 Dアルゴリズム[84]

この方法は故障が検出可能である事を表わすのに **D*** 及び $\bar{\mathbf{D}}$ という状態を導入し，これを出力端子に伝搬させるような回路状態を求めるアルゴリズムである．

図-23 を用いて**D**アルゴリズムを簡単に紹介する．図において信号線6の1縮退故障を検出する場合を考える．そのためNAND 6に出力が $\bar{\mathbf{D}}$ となるような状態 (Primitive **D**-cube of failure と称する) を設定する．次に信号線6を入力としている素子7に対し $\bar{\mathbf{D}}$ を出力へ伝搬するような状態 (Propagation **D**-cube) を設定する．この操作はパス活性化法の前方操作と類似であり **D**-drive と呼ばれる．この操作を出力端子に**D**または $\bar{\mathbf{D}}$ が伝搬するまで繰り返す．出力に**D**または $\bar{\mathbf{D}}$ が現われたら今度は素子の出力の状態が決っており入力が未定のものについて入力状態を決める Consistency 操作を行う．このようにして求められた入力端子の状態が求めるテストベクタである．**D**-drive または Consistency 操作の途中で矛盾が生じて先へ進めなくなることもあるが，この場合には Propagation **D**-cube または入出力関係の別の組み合わせを採用して続行する．

Dアルゴリズムは解が存在すれば必ず見つけ出すという利点があるため多くのテストベクタ作成システム



(a) 回路例

ステップ ＼ 信号線	1	2	3	4	5	6	7	8	9
1. 故障注入		1	1			$\bar{\mathbf{D}}$			
2. NAND (7) 伝搬					1		D		
3. NAND (8) 伝搬								D	
4. 出力へ伝搬				1					$\bar{\mathbf{D}}$
5. Consistency 操作	0								

(b) ステップごとの状態

図-23 Dアゴリズムの例

* **D**は正常回路で1，故障が存在するとき0であるような故障，すなわち0縮退故障が存在する状態を表わす．

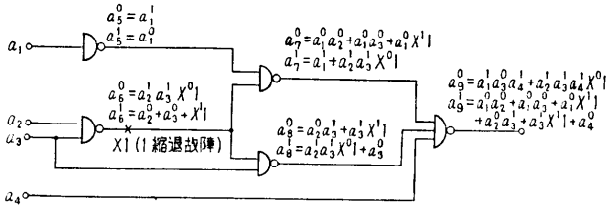


図-24 論理式シミュレーション法の例

においてそのままあるいは一部修正した形で広く採用されている．

8.2.4 ブール微分

故障検出問題を代数的に扱う方法にブール微分がある．図-23 において各信号線の状態を x_i で表わした時出力 x_9 は入力信号と x_6 で表現すると

$$x_9=\bar{x}_4+\bar{x}_1x_6+x_3x_6$$

となる．x_6 の故障が x_9 に影響を与えるためには

$$\frac{dx_9}{dx_6}=x_9(x_6=1)\oplus x_9(x_6=0)$$
$$=(\bar{x}_4+\bar{x}_1+x_3)\oplus\bar{x}_4$$
$$=\bar{x}_1x_4+x_3x_4=1$$

となればよい．この dx_9/dx_6 は x_9 の x_6 に関するブール微分と呼ばれる．

x_6 の1縮退故障を検出するためには

$$\bar{x}_6\frac{dx_9}{dx_6}=x_2x_3\frac{dx_9}{dx_6}$$
$$=x_2x_3x_4=1$$

となればよい．すなわち (x_1, x_2, x_3, x_4) が $(0,1,1,1)$ または $(1,1,1,1)$ の時 x_6 の1縮退が検出可能である事が分かる．

このブール微分による方法は条件を満たす全てのテストベクタを同時に求めうる特徴があるが，式の操作が計算機で扱うのは必ずしも効率的でないのと，テストベクタとしては任意の1パタン求まれば良いのでこの方法は余り採用されていない．

8.2.5 論理式シミュレーション法

Bell 研究所の LAMP システム中のテストベクタ発生法[85]として採用された方法は論理回路素子の出力を1(0)にセットする条件を論理式の形で求め，その結果が次々と出力側の素子へ伝搬されてゆき，外部端子まで伝えるものである．**図-24** はこの方法による例を示す．$a_i^0(a_i^1)$ は信号 i が値0(1)を取るとき真になる論理変数を表わし，$X^01(X^11)$ は故障1(信号線6の1縮退故障) が存在しない (する) 場合真になることを示す．外部出力端子での式を X^01 及び X^11 でまとめて

$$a_9{}^0=A+BX^11+CX^01$$
$$a_9{}^1=D+EX^11+FX^01$$

と表現したとき，故障1を検出するテストは

$$T=B\cdot F+C\cdot E=a_2{}^1a_3{}^1a_4{}^1$$

で求められ $(a_1, a_2, a_3, a_4)=(x, 1, 1, 1)$ となる．

8.2.6 その他の方法

簡単な代数演算を繰り返し適用してテストベクタを求める M0M1 アルゴリズム[86),87)]とよぶ発見的手法がある．この方法では 0,1 の論理値の他に $X, U, 0*$ $1*$ を導入し，外部入力端子の値を $X\rightarrow 1*\rightarrow 0$ または $X\rightarrow 0*\rightarrow 1$ と変化させることにより回路中のゲートを所定の値にセットすることを利用している．この方法は順序回路に対しても適用でき余り複雑でない論理回路に対して能率が良いことが報告されている．

また論理回路を各入力論理変数に出力端子に至る径路に相当する番号をつけた SPOOF と称する論理式で表現し，テストベクタを求める方法[88)]も発表されているが計算機プログラムとして実現するのは必ずしも容易ではないようである．

8.2.7 順序回路への適用

前述の各種の方法の多くは組み合わせ回路を対象にしたものである．これを順序回路に適用するためには回路を記憶素子の所で擬似入出力に分割し，組み合わせ回路のカスケード接続として表現する場合が多い．しかしこれは回路規模が大きくなったのと等価であり，高い検出率が得にくくなることの他に大部分のアルゴリズムに使われている単一故障仮定が成立しなくなるという問題がある．

これに対しては D アルゴリズムに故障のない回路または故障のある回路の一方のみの指定を許す状態値を導入した9値モデルによる方法[89)]が有効であると思われる．また順序回路を適当な大きさの部分回路に分割し，その全ての内部状態を遷移させる入力ベクタの合成でテストベクタを作る方法[90)]も研究されている．

実用システムにおいてはゲートを外部端子を基準にあらかじめレベルづけしておき外部端子に近いものから優先的に値を割付けてテストベクタ発生の時間の短縮をはかるようにした例[91)]もある．また故障分解能を高めるための手順[92)]に関する発表もある．

8.3 検出可能故障の導出

前述のテストベクタ作成法は特定の1個の故障を検出する入力パタンを作り出すが，それは同時に他の故障をも検出可能である．与えられた入力パタンにより検出可能な全ての故障を求める手順がこの節の対象である．これは故障診断辞書を作成するためにも重要な役割を果す．

8.3.1 故障シミュレーション

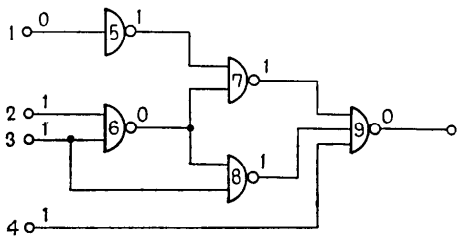
故障を含む回路モデルの動作をプログラムによるシミュレーションにより求め特定の入力パタンを与えたとき正常回路の出力と異なるかどうかを判定する故障シミュレーションは回路形式に関係なく適用でき，多くのテストベクタ作成システムに採用されている．故障シミュレーションの手法については **5.3.5** で示した．

8.3.2 TEST-DETECT

TEST-DETECT は J. P. Roth らにより D アルゴリズムと同時に発表された方法[84)]であり，出力側に近い素子から順番にその D（または $\bar{D}$）状態が外部出力端子へ伝搬するかまたは既に検出可能であることが判明している素子へ単独に伝搬する場合，検出可能であると判断する．**図-25** に TEST-DETECT 法の例を示す．

8.3.3 ハードウェアシミュレーション

ソフトウェアによる故障シミュレーションは実際のハードウェアを必要とせず，設計変更等にも容易に対処できるが回路の規模が大きくなると必要な計算機時間も飛躍的に増大するので装置レベルの回路に適用するのは難しい．そのため実際の装置に擬似故障を導入



(a) 入力(0,1,1,1)を与えた時の状態

素子番号	検出可能故障	理　由
9	s-a-1*	外部出力端子
8	s-a-0*	9へ単独に伝搬
7	s-a-0	〃
6	s-a-1	9で $\bar{D}$ が収束し単独になる
5	—	7で消滅
4	s-a-0	9へ単独に伝搬
3	s-a-0	7へ単独に伝搬 (8では D, D 入力で消滅)
2	s-a-0	6へ単独に伝搬
1	—	5で消滅

* s-a-1(0): 1(0) 縮退故障

(b) (a)の状態での検出可能故障

図-25 TEST-DETECT 法の例

しテストデータを与えて故障に対する応答を収集するハードウェアシミュレーション[93]が採用されることもある．通常 IC とプリント基板の間にリレーを組み込んだ故障挿入装置をはさみ制御用コンピュータによって各種の擬似故障を作り出す．ソフトウェアシミュレーションに比べ論理的な故障以外の IC の電源オープン，ピン間ショート等も作り出せるという特徴もある．

8.4 検査容易回路

論理回路の検査・診断性の向上は近年増々重要になっており，またテストデータ作成のための設計工数も無視できなくなっている．検査容易回路に関する理論的研究も数多く行われているが[94]，実用的には順序回路の検査性向上が重要な課題である．

短い入力シーケンスで記憶素子を初期状態に設定可能にしたり，論理深度を浅くするための回路変更は大部分の設計において実施されているが，順序回路中に含まれる記憶素子を検査モードのときシフトレジスタとして接続し，任意の状態の設定及び読み出しを可能にした構成は検査データ作成問題を組み合わせ回路として扱えるため非常に効果がある．

9. LSI のインパクト

最近の半導体の集積度向上の度合は著しく，電子計算機の設計へも大きな影響を与えている．その一つの形は市販の IC の MSI 化，LSI 化であり，これは従来の実装法のものにもどんどん採用され装置の小型化，低価格化に貢献している．他の一つは LSI 実装方式を採用した最新の大型電子計算機の方向である．これらが計算機 DA システムに与える影響を考察してみる．

9.1 現行の実装法

市販の IC の LSI 化は機能演算ブロック，レジスタアレイ，ビットスライスマイクロプロセッサ等の形で現われてきている．機能が高級になるに従って IC のピン数も増加しており各種のサイズの IC パッケージがプリント板の上に混載されるため，実装 DA システムは同一サイズの部品の代りにサイズの異なる部品の配置問題を扱う必要が生じてきた．しかしこれは標準化された部品配置位置を設けて処理すれば割合容易に処理する事ができ，従来システムの小変更で対処できる．

LSI 化の影響は検査データ作成面に大きく，自動検査データ作成システムでは通常回路をゲートレベルに分解して扱うため，回路規模の増大と順序回路の論理深度の増大が問題になっている．IC ベンダから供給される仕様書にも内部構造は記してないものが増えており，回路構造をもとにして検査データを作る方式に対しては回路モデル作りが結構面倒な仕事になり始めている．このため，機能レベルの回路記述の混在をも認めるようにしたり[47]，特定の LSI に対してはマクロな回路として扱うような試み[43]も行われているが，必ずしも十分汎用性があるとは言い難く，今後の手法の開発が重要である．

9.2 LSI 実装法

LSI の特徴を生かした高密度実装の LSI 計算機に関する DA システムについてはまだ余り発表されていないが，設計法や DA システムの使われ方も従来の実装法とは異なった配慮が必要であると思われる[95]．

9.2.1 設 計 法

LSI 化が進むに従ってチップの品種が増加する傾向がある．しかしこれは同一品種当たりの生産量の低下となり，コストアップにつながるので，できる限りリピータビリティの高くなるような設計法が要求される．LSI 計算機に関する設計法としては**図-26** に示すような二つのアプローチが考えられる．

第一は Top Down 方式であり，LSI の論理は意識しないで論理設計を行い，その後 LSI チップの制約条件すなわちピン数やゲート数の制限を満たすよう分割を行い，それに従って LSI チップのレイアウト設計を行う方式である．この方法は設計の自由度は大きいが必然的に LSI の品種が増えてしまうので LSI チップの構成法に工夫が必要である．その解決法として**マスタスライス方式，ビルディングブロック方式**または PLA (Programmable Logic Array) 等が提案され，実用化されている．

マスタスライス方式は 1 つのチップの中に入れるゲ

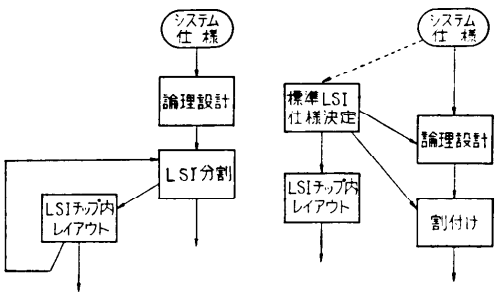


図-26 LSI 計算機の設計法

ートの種類及び数は固定しておき，品種ごとにそれらの接続を変更することにより論理を構成する方式である．この場合には LSI の拡散パタンは全品種共通にしておき，配線パタンのみを可変にすれば良いので，品種ごとに全パタンを設計するのに比べ，はるかに少ない工数で設計を行う事ができる．このための DA プログラムはプリント板と類似の径路結線アルゴリズムを用いる事ができるが配線エリアが限定されているため自動的にパタン化できない線に対し人手修正が必要となる．

ビルディングブロック方式は MOS LSI に比較的多く採用される方式で基本ゲート群をファンクショナルブロックとして用意しておき，このブロック列を列間の配線が可能な最短間隔に配置してゆく方式である．自動設計が可能であり，100% 結線が保障される．

PLA は論理を積和標準形式に表現し多入力 AND ゲート及び OR ゲートの入力を選択的に接続し論理を構成する方法である．マスタスライス方式に比べ，はるかに簡単なプログラムで自動設計できるが，一般に有効論理密度が低くまた算術演算等論理段数を多く必要とする論理は実現しにくい欠点がある．しかしながら拡散パタンのみならず配線パタンまでも大部分を共通化できるため品種が増えたことによるコストアップの割合は少なく LSI 化の有力な方式の一つである．

Top Down 方式に対抗する設計法は Bottom Up 方式である．これは方式設計が終るとまず LSI の品種を決め，論理設計はこれらの品種組み合わせで行う方式である．これは従来のベンダーから供給される IC のファミリーを使って論理設計を行う方式に似ており品種の数はおさえることができるが，かなり集積度の低い単位ゲートに近い品種も用意しておかないと全体のロジックを吸収できない欠点がある．Bottom Up 方式による LSI の品種の決定に有力な方法として Bit Slice 法がある．これは計算機のデータパスの部分を数ビット以下の信号伝搬方向の論理に切り取る方式で，ピンーゲート比を高める事ができ，リピータビリティも向上させる事ができる．Bottom Up 方式においては計算機用 DA と LSI 用 DA の部分を比較的独立に扱うことができ，従来の DA を割合生かす事ができる．

9.2.2 従来の DA との違い

LSI 計算機に対する DA システムは従来の IC 計算機用 DA と比較していくつかの特徴がある．

まず第一に論理シミュレーションの重要さをあげることができる．これは LSI においては設計変更による論理の修正が全く不可能なため，設計の段階で論理を完全にデバッグしておく必要があるためである．

次に検査データ作成が難しくなり高度のアルゴリズムが要求されるようになる．LSI においては一般にゲートの数を増やすことは容易であるが入出力ピンの数はかなり少なくおさえるため論理設計の段階から検査を容易にするための付加回路を考慮しておく必要があり，またゲート価格が相対的に安くなっているためこのアプローチが正当化される．

また一般 LSI 用の CAD システムと比べてみると Top Down 方式においてはかなり多品種の LSI を短時間に設計する必要があるためインタラクティブ・デザインによりチップ面積を減少させることよりも自動設計の割合を強めた設計工数の縮減の方がより重視される．

10. インタラクティブ・デザイン

コンピュータの設計自動化は従来バッチによる自動処理が中心であり，途中で設計者が介入することは比較的少なかった．これは半導体，特に LSI の設計が CAD と呼ばれ，設計者と計算機の対話で進められるのと対称的である．その理由としては計算機の場合の方が一度に扱う論理の量が多く，設計の省力化に対する要求がより強いことと全ての処理に対して CAD 的な扱いをするには計算機の処理能力が不足気味であった事が考えられる．しかし設計の段階の中にはバッチ処理では十分な効果が得にくい分野もあり次第に対話的に設計を進めるインタラクティブ・デザインの手法が採用されはじめて来た．現在インタラクティブ・デザインの対象としては設計情報の入力や修正時のエラーチェック，ゲート割付や配置等の試行錯誤を行いたい分野及びプリント板等のパタン設計のように図形処理を必要とする分野が中心である．

10.1 インタラクティブ・デザインの形態

対話形式で設計を進めるために必要なシステムとしては通常大型計算機によるタイムシェアリング機能を使ったものと，ミニコンによる図形処理専用システムを用いたものとがある．大型計算機を用いたものは処理の内容により TTY やキャラクタディスプレーを端末としたもので文字情報の扱いを中心にしたものとグラフィックディスプレー端末を用いた図形処理用のものに分ける事ができる．

タイムシェアリング機能を用いて端末から設計情報

を入力する場合のメリットは，通常入力をするとすぐエラーチェックを行えるため入力・修正のサイクルを短縮できることや，設計者の作業に計算機処理待の時間がなくなり思考の連続性が保てることである．論理シミュレーションや人手でゲートを IC に割りつけたり，IC をプリント板上で配置したりする作業は通常計算機に特定の指示に対する処理を行わせ，その結果により次の指示を判断する作業の繰り返しであるのでそのターンアラウンド時間の短縮が設計期間の短縮に効果的であり，インタラクティブ・デザインの効果が著しい分野である．

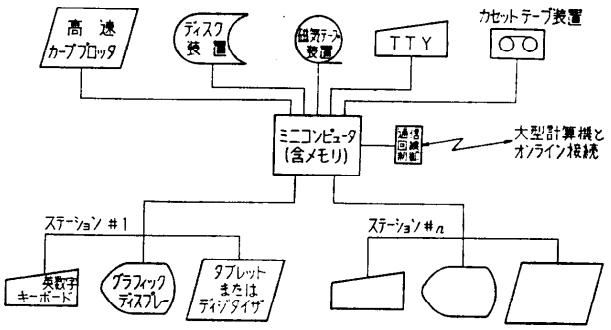


図-27 ミニコンによるグラフィックシステムの構成例

プリント基板のパタン設計や LSI のレイアウト設計にはグラフィック端末を使用したインタラクティブ・デザインが有効である．大型計算機を用いたオンライン端末の場合には他の設計自動化処理と同じデータベースをアクセスすることができ，また処理能力も高いので複雑な操作を短時間で行うことができるが，図形処理のための高度システムを開発する必要がある．一方ミニコンを中心とする図形処理専用システムが数社から市販されており，これは基本的な汎用図形処理用ソフトウェアがベンダーから提供されるので容易に設計自動化システムの中に組み込むことが可能である．大型計算機に比べ端末の数も余り多くないのでレスポンスはかなり早くなるが，システムの使用効率はどうしても低くなり勝ちである．ミニコンシステムの場合にはディジタイザ，カーブプロッタ等の図形入力機器を直接制御可能なので複雑なパタンを処理する場合には有利である．しかし処理能力の点からミニコンシステムのみで全ての処理を行うことは難しいので通常磁気テープまたは通信回線を介して大型機側の設計自動化システムと接続され，ミニコン側では自動パタン発生後の人手修正やパタン出力等の処理を分担しているシステムが多いが[96),97)]，パタン入力の手段として利用しているシステムもある[98)]．

ミニコンシステムを大型計算機の端末として使用して入力，配置，パターン発生と修正等プリント板設計における全ての処理をグラフィック端末から行うようにしたシステムも報告されている[99),100)]．

図-27 にミニコンによるグラフィックシステムの構成の例を示す．

10.2 バッチシステムとの比較

インタラクティブ・デザインとバッチシステムの特徴の比較を**表-4** に示す．それぞれ長所と短所を有するのでどちらか一方のみで処理するのではなく互いの長所を生かした複合システムが今後の方向と思われるが，ハードウェア及びソフトウェアの発達と共にインタラクティデブ・デザインの比率が高まってゆくものと思われる，

表-4 インタラクティブ・デザインとバッチ処理の比較

	インタラクティブ・デザイン	バッチ処理
最適設計	試行錯誤による最適化が可能. 経験の付加により効率化.	原則としては1パス方式. 組込アルゴリズムにより固定.
使用しやすさ	入力データのエラーは対話的に修正可能. 使用しながら学習可能.	エラーがあるとプログラムはストップ．全体のやり直しが必要.
汎用性	広範囲の対象が処理可能.	対象は一般に固定.
例外処理	設計者の洞察力により判断基準可変. 結果を見て短時間にフィードバック可能.	例外処理がしにくい. ターンアラウンド時間が長い.
設計の質	設計者の能力・経験により異なる.	常に一定の水準が得られる.
設計効率	計算機との対話において連続的に設計が進められるので効率がよい. 設計期間が短い.	ターンアラウンド待ちの間思考が分断されるので効率がよくない. 設計期間が長い.
プログラム構造	基本操作がコマンドの形で独立に用意されており，使用者が順序，パラメータを制御する.	基本操作を上位プログラムが組込アルゴリズムにより制御する.
システム開発	人間工学，効率的なリソース管理，エラー回復機能等考慮する必要あり.	比較的容易.
経済性	端末，通信回線等を長時間専有するので一般に高価	インタラクティブ・デザインに比べ経済的

11. 将来の課題

コンピュータの分野における設計自動化の現状について説明を行ってきたが，最後に今後更に研究・開発が必要であると考えられる分野について触れてみる．

まずデータベースについては設計自動化に適した効率の良い処理方式を確立する必要がある．取り扱う設計情報の種類及び量は今後増々増加する方向にあり，

また処理形態もインタラクティブ・デザイン等多様化する傾向にあるので，それらに対処できるデータ構造及び処理方式が望まれる．

ファームウェア設計の分野においては，オペレーティングシステム等のファームウェア化の影響でマイクロプログラムの量が増えるのに対処するため実用的な自動生成や最適化の手法の開発が重要である．

論理設計の分野においては能率の良い設計情報の入力方法が必要である．従来の1レベルの単純な記述の代りに設計の初期の段階から詳細設計の段階まで使用できるように構造化された記述を許すような階層化構造の言語とその処理方法及び多レベルの機能のシミュレーション方法を確立することが望まれる[101]．また自動設計検証の技術の確立が望まれるが，ソフトウェアの分野においても研究中であり近い将来類似の方法によるハードウェアの論理の検証が行われるようになるであろう．

論理設計の分野においても自動生成技術の実用化が重要である．第一段階としては機能レベルの構造記述を個々の論理レベルの素子に展開する効率のよい手法が望まれるが，次のステップとしては動作記述を構造記述に変換する技術の実用化が必要である．論理設計の場合には性能を最適化するか，ハードウェア費用を最適化するかにより各種の選択があり，また使用テクノロジーの違いにより制約条件も異なるので，それらをパラメータで与えられることが必要である．

プリント板，LSI 等の自動結線処理は設計自動化の中でも歴史の古い技術であるが，大部分の方法は一度パタンを引くとその位置を固定してしまうので後で引くパタンの妨害になってしまい結線率を高くしにくい欠点がある．後で引くパタンの進路に存在するパタンを移動させる能率のよいアルゴリズムが切望されている．

論理機能の検査の分野では LSI の進歩によりゲート数の増加が著しく，大規模論理回路の検査データ作成が次第に困難になって来ている．特に論理深度の深い順序回路については無効索探が多くなり，検査データ発生に要する計算機時間が禁止的に増加するので，性能のよいアルゴリズムの開発が待望される．しかし検査データの発生の容易さは回路構造にも大いに依存するので検査容易回路の研究と，与えられた回路を検査容易な等価回路に変換する技術の開発も重要である．

12. おわりに

コンピュータの設計自動化について解説を試みた．紙面の都合もあり重要な技術にしか触れる事ができなかったが更に広範な内容については各種のサーベイ[102)~105)]，入門書[106)]及び設計自動化に関する研究会[107)]や会議[108)]が行われているので参照されたい．本文の内容の一部は本学会設計自動化研究連絡会における討議を参考にした事を付記したい．本文が設計自動化の分野への理解に役立てば幸いである．

参考文献

83) D. B. Armstrong: On Finding a Nearly Minimal Set of Fault Detection Test for Combinational Logic Nets, IEEE Trans. on EC, Vol. EC-15, pp. 66～73 (1966).

84) J. P. Roth et al.: Programmed Algorithm to Compute Tests to Detect and Distinguish between Failures in Logic Circuits, IEEE Trans. on EC, Vol. EC-16, pp. 567～580 (1967).

85) S. G. Chappell: "LAMP: Automatic Test Generation for Asynchronous Digital Circuits", B. S. T. J., (Oct., 1974), pp. 1477～1503.

86) T. Arima, M. Tsyboya et al.: A New Heuristic Test Generation Algorithm for Sequential Circuits, Proc. 10th DA Workshop, pp. 169～176 (1973).

87) 壺屋，天宮他：順序回路の検査パターン作成の発見的一手法，情報処理，Vol. 16, pp. 108～114 (1975).

88) M. J. Flomenhoft et al.: Algebraic Techniques for Finding Tests for Several Fault Types, Proc. 1973 Int. Symposium on Fault Tolerant Computing.

89) P. Muth: A Nine Valued Circuit Model for Test Generation, IEEE Trans. on Comp., Vol. C-25, pp. 630～636 (1976).

90) 山田，内藤：順序回路のテスト系列を求める一手法，昭和 51 年電子通信学会総合全国大会，p. 6-210.

91) A. Nakamura, J. Nose et al.: High Speed Generation of Fault Location Data for Logic Circuits, Information Processing 74, IFIP Congress Report, pp. 112～116 (1974).

92) C. H. Saltzmann, R. A. Cooper: A Technique for Automatic Digital Test Generation for Improved Fault Symptom Resolution, USA-Japan DA Symposium '75, pp. 30～32 (1975).

93) 北村，田代他：ハードウェア故障シミュレーションによる故障辞書作成方法，電子通信学会計算機研究会資料 EC 74-16 (1974).

94) R. G. Bennetts, R. V. Scott: Recent Developments in the Theory and Practice of Testable Logic Design, IEEE Computer, June 1976, pp. 47～63.
95) 上野，田渕：高密度実装に対する DA の問題点とその対応策に関する一考察，電子通信学会電子材料・部品研究会資料，CPM 75-98 (1975).
96) 平川，上田他：プリント板の配線設計と設計変更処理システム，情報処理学会設計自動化研究会資料，DA 28-2 (1976).
97) 池本，上川井他：高速論理 LSI 用 CAD (CAD 75)，情報処理学会設計自動化研究会資料 DA 28-3 (1976).
98) A. Inao, S. Ueda et al.: Design Automation System for Manually Routed Circuit Board, USA-Japan DA Symposium '75 Proc. pp. 114～118 (1975).
99) L. Marks: Use of On-line, Time-shared Graphic System to Design and Document Printed Circuit Boards, 13th Design Automation Conf. Proc., pp. 91～103 (1976).
100) H. N. Lerman: Computer Aided Design of Printed Circuit Boards Using Remote Graphics and TSO, 13th Design Automation Conf. Proc., pp. 104～108 (1976).
101) C. W. Rose: Design System——A Five Year View, COMPCON '76 Proc., pp. 190～193 (1976).
102) M. A. Breuer: General Survey of Design Automation, Proc. IEEE, Vol. 56-12, pp. 1708～1721 (1966).
103) M. A. Breuer: Recent Developments in Automated Design and Analysis of Digital Systems, IEEE Computer, May/June 1972, pp. 23～35.
104) 論理装置の設計・製造自動化，電気学会技術報告 (I部) 第 94 号 (1970).
105) 計算機設計自動化研究委員会報告 (昭和 46, 47 年度)，情報処理学会 (1973).
106) M. A. Breuer 編：Design Automation of Digital Systems, Prentice-Hall (1973).
林孝雄訳：ディジタル計算機の自動設計，産業図書 (1973).
107) 情報処理学会，設計自動化研究会.
108) ACM, IEEE 共催：Design Automation Conference (年1回開催).

(昭和 52 年 2 月 23 日受付)