解　説

VLSI設計 CAD の最近の動向†

高　木　　茂††

1. まえがき

VLSI の規模が増大し，その応用分野も急激に広がりつつある．これにともない以前にも増して必要な機能を早く正しく実現できる設計支援システムが重要となってきている．このため機能・論理回路・タイミングの検証，設計ワークステーション，シリコンコンパイレーション，自動設計（論理合成）などのさまざまなCAD技術が研究されている．本稿はこれらの技術のなかで設計の無謬性，設計時間の短縮に最も貢献すると予想される自動設計を中心に，1984年の本誌大特集：『論理装置 CAD の最近の動向』[1]以後の動向を概観する．なお個々のアルゴリズムの詳細については原論文を参照されたい．

2. VLSI 設計の流れ

VLSI 設計では複数の設計段階を設定し，抽象的な仕様から出発して段階的に詳細化を行うのが通例である．設計段階の設定法はVLSIの規模，設計者，類似設計をしたことの有無により異なる．また境界も必ずしも明確ではないが，おおよそ次の3段階にまとめられる．

1) 方式設計

VLSI で処理すべき問題を想定し，その問題を所定のコスト，性能で実現するために命令セット（プログラム蓄積方式の場合），大まかなアーキテクチャ（パイプライン方式，並列計算方式など）およびアーキテクチャ上でのその命令セットの実行アルゴリズムを決定する．方式設計の結果VLSIの論理はいくつかのサブシステムに分割され，各サブシステムは有限状態マシンとしてその動作仕様が規定されることが多い．この段階は命令セットレベル設計あるいはシステムレベル設計と呼ばれることもある．

2) 機能設計

方式設計結果をもとに演算器，レジスタ，バス，マルチプレクサなどの機能ブロックを構成要素としたデータパスの構造およびデータパスを制御する制御論理を設計する．機能ブロックの詳細な回路構成はまだ未定である．またマイクロプログラムの設計もこのレベルである．この段階はレジスタ・トランスファ設計と呼ばれることもある．

3) 詳細論理設計

データパス系，制御系の各機能ブロックの詳細論理を定め基本素子の接続構造に変換する．この段階は単に論理設計あるいは回路変換と呼ばれることもある．

一般に設計の抽象度が高いほど自由度が大きく，また性能，コストなどの正確な評価もむずかしいため自動化アルゴリズムの作成が困難とされている．しかし設計工数を削減する必要性に迫られ，また最近の知識処理技術の応用がこれら難問の解決に役立つかもしれないとの期待より，研究領域は下位の設計レベルからしだいに上位の設計レベルに拡がりつつある．以下，各レベルにおける自動設計手法を紹介する．

3. 方式・機能設計と CAD

本章では方式設計および機能設計レベルにおける自動化アルゴリズムを紹介する．

3.1 仕様記述

自動設計すべき対象の仕様はなんらかの形式的な仕様記述言語で与えなくてはならない．仕様記述は設計者が行うので，設計者にとってその枠組みで考えやすく，記述しやすく，かつ曖昧性なく表現できる言語でなくてはならず，その構文，意味の付与のしかたは重要である．このためさまざまな仕様記述言語が提案されている[2),3)]．しかし自動設計システムでは言語間のシンタックス，意味の多少の差異をとりはらった本質的な情報のみからなる中間表現に基づいて合成を行うのが通例である．したがって自動設計の難易度は仕様記述のなかで本質的な情報として何が規定されてお

† Recent Studies on VLSI System, Functional and Logic CAD by Shigeru TAKAGI (NTT Electrical Communication Laboratories).

†† NTT 電気通信研究所

り，何が自由度として残されているかに依存する．

たとえば方式設計（命令セット）レベルと機能設計レベルの最も大きな相違は，後者はクロックを単位として処理アルゴリズムが規定されているのに対し，前者はクロックの概念が存在しないことである．クロックの指定のないことが設計の自由度を1次元増やし，方式設計アルゴリズムの定式化をより難しくしている．またたとえば，機能設計レベルにおいても，①純粋な動作記述から演算器，バスなどを自動設計しなくてはならない場合と，②演算器，バスなどの主要な機能ブロックは宣言されており，これらをどのように使って動作を実行するかを記述した仕様に基づき細部を自動設計する場合とでは自由度は異なる．

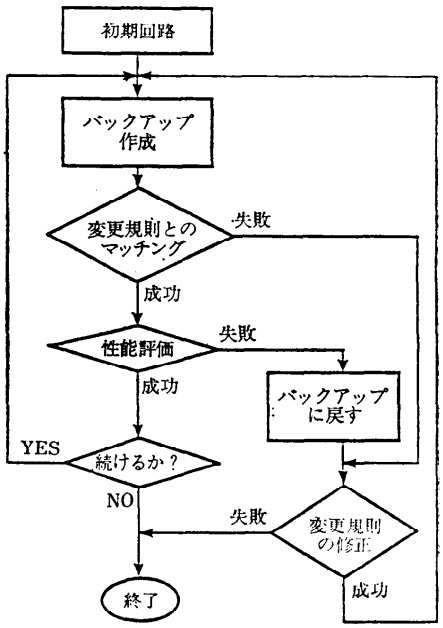
合成品質を妥当な線に保つ，設計者の意図を設計結果に反映させやすくする，あるいは自動設計アルゴリズムの作成を容易にするためには自由度をあまり広げ

```
(PROGRAM PARCOR-FILTER
   (DCL-IO (S   (INPUT   (STREAM   (SERIAL)))
                (WIDTH   (8))
                (TYPE    (ARRAY    (N))
                         (FIXED    (7)))
                (ROLE    (PORT     (INPUT))))
           (E   (OUTPUT  (STREAM   (SERIAL)))
                (WIDTH   (8))
                (TYPE    (ARRAY    (N))
                         (FIXED    (7))
                         (PRECISION (7)))
                (ROLE    (PORT     (OUTPUT))))
           (RK  (INPUT   (STREAM   (SERIAL)))
                (WIDTH   (8))
                (TYPE    (ARRAY    (M + 1))
                         (FIXED    (7)))
                (ROLE    (PORT     (INPUT))))
           (M   (STEP    (12)))
           (N   (POINT   (128)))
           ((FT GT GTO)
            (TYPE (ARRAY (M + 1)))))
  (DCL-RESTRICTION
    (SAMPLING-FREQUENCY ((12.5 ^ 3))))
  (PROCESS
    (DO ((I 1 (I + 1)) (I =16))
        (GTO{I} = 0.)
        (FT{I} = 0.))
    (E{1} = S{1})
    (M1 = M + 1)
    (DO ((J 2 (J + 1)) (J = N))
        (FT{1} = S{J})
        (GTO{1} = S{J - 1})
        (DO ((I 2 (I + 1)) (I = M1))
            (FT{I} = FT{I - 1}
                   - RK{I - 1} * GTO{I - 1})
            (GT{I} = GTO{I - 1}
                   - RK{I - 1} * FT{I - 1})
            (GTO{I} = GT{I}))
        (E{J} = FT{M1}))))
```

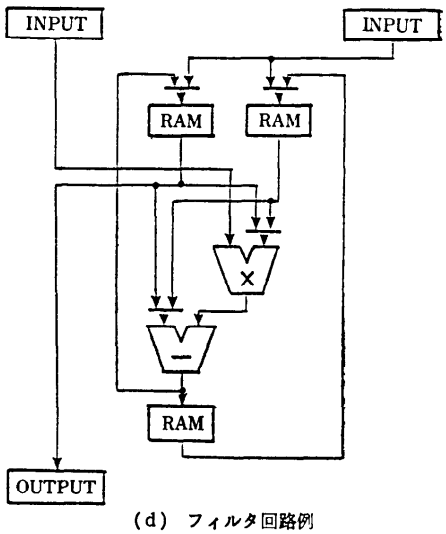
（a） フィルタの仕様記述例

```
(RULE4 (AND (#OPERATOR1 ^FUNCTION ADD)
            (#OPERATOR2 ^FUNCTION SUB))
       =>
       (OPMGD OPERATOR1 OPERATOR2))
```

（b） 回路の変更ルール例



（c） 回路の変更手順



（d） フィルタ回路例

図-1 信号処理アルゴリズムに基づくデータパス系自動設計[9]

ないアプローチが現実的という考えもある．本稿では設計の本質を明らかにすること，および波及効果を狙いとして自由度の大きい仕様記述を出発点として自動設計するアルゴリズムを主体に紹介する．

3.2 データパス系設計

VLSI 全体の処理方式，サブシステムへの分割などの最も高度な方式設計を自動化するシステムの報告はまだないようである．クロックを陽に指定しない処理アルゴリズムに基づきデータパスを自動設計する研究は CMU-DA システム[4]や竹沢らのシステム[5]がある．CMU-DA システムでは ISPS (Instruction Set Processor Specification) 言語で記述された仕様から VT (Value Trace) と呼ばれる有向グラフを生成し，これをもとに演算ノードの併合，変数のレジスタへの割り当てなどを行って ALU，レジスタ，マルチプレクサからなるデータパスを得ている．

竹沢らは信号処理の分野を想定し特殊な仕様記述法に基づいて合成している．図-1(a) は PARCOR-FILTER の仕様記述例であり，DCL-IO に続くリストが変数の属性の宣言部，DCL-RESTRICTION に続くリストが制約条件に関する宣言部，PROCESS に続くリストが処理アルゴリズムの記述部である．DCL-IO 部ではたとえば，変数 S はストリーム形式シリアル入力で，8 bit 幅，1 から n までの一次元配列で，入力ポートから入力されるということを宣言している．DCL-RESTRICTION 部ではフィルタのサンプリング周波数が 12.5 KHz であることを宣言している．PROCESS 部は COND 節，WHILE-DO 文，あるいは決まった回数だけ繰り返す DO 文など構造化プログラムに似た形式でアルゴリズムを記述している．データパスの合成は以下の手順で行っている．まずアルゴリズムにあらわれる演算動作ごとに演算器を設け，並列度最大（性能最大）のデータパス構成を求める．次に図-1(b)に示すような演算器の併合など，主として低コスト化をめざす回路変更ルールを上記データパスに適用して改善する．図-1(b)は加算器と減算器を一つの演算器に併合するルール例である．回路変更ルールを適用するごとにデータパスの性能評価を行い，目標性能を満たさない場合には回路をもとに戻すなどの制御を行っている (図-1(c))．図-1(a)の目標性能を満たすデータパスとしては 4 種類得られ，そのなかの一つは図-1(d)のようになるとのことである．

機能設計レベルからの自動設計手法についてもいくつかの報告がなされている[6]~[8]．筆者らは機能設計記述言語 DDL[9] にもとづく VLSI 自動設計システム (図-2) を研究している[10]．本システムは DDL 仕様記述をトランスレートし演算代入を指示する演算動作とその実行条件の表にまとめる．この表をもとにデータパス系の合成，制御系の合成を行う．次に各機能ブロックを論理式に展開後，回路に変換する．以下データパス系の合成アルゴリズムを示す[11]．機能設計レベルでは各クロックサイクルごとの処理内容は仕様として与えられており，設計課題の一つは仕様を満たしつつコストのできるだけ低いハードウェア構成を得ることにある．本システムではハードウェアの並列動作性に着目し，並列に動作しないハードウェア資源を併合する手法をとっている．またデータパス系を演算系/データ転送路系と二つのサブシステムに分けて設計し，重要なものから段階的に構成することによりコスト準最小の構造を得ている．すなわち，

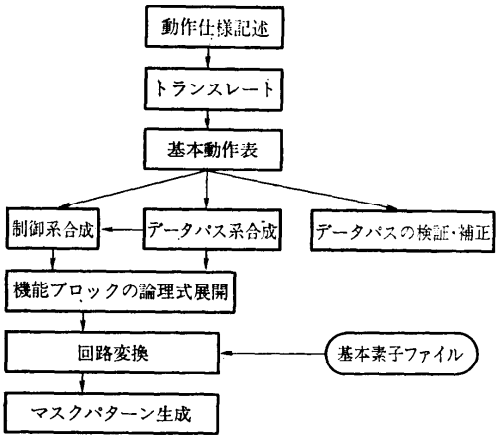


図-2 DDL 機能仕様記述に基づく VLSI の自動生成手順[10]

（1） 演算動作の実行条件を調べ演算動作相互間の並列動作/非並列動作を判定する．

（2） 並列に動作しない演算動作グループに対し，1 演算器を生成する．

具体的には図-3(a)に示すように，まず任意に一つ演算動作を選びそれを核とした演算器を生成する．次にこの演算器と並列動作せず，かつ併合したときにハードウェア構成的に妥当となる演算動作を順次探して併合していく．併合できる演算動作がなくなれば新たに核演算器を設け繰り返し処理を行う．演算動作を核演算器に併合する概念を図-3(b)に示す．併合処理により演算器の機能，各機能の選択条件，演算器とレジス

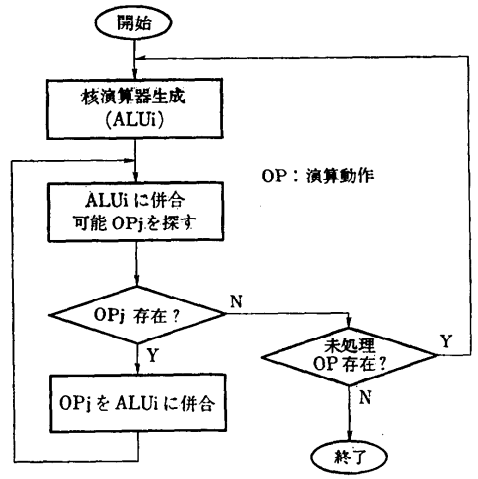
タなど資源間の転送路についての論理的な情報（転送元，転送先，転送条件）などが合成されていく．併合処理はルールベースプログラムとなっている，図-3(c)に併合ルール（意味を英訳してある）の例を示す．

（3） 演算器合成後転送路の合成を行う．転送路の並列動作性を解析し，互いに並列に動作しない転送路のグループをバスに併合する．その後，必要な箇所にマルチプレクサ，トライステートゲートなどを付加する．

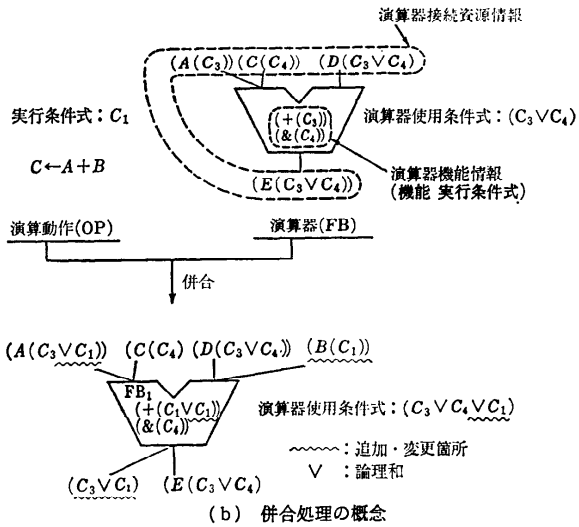
これらの手法により，従来の合成アルゴリズムに比較しハードウェアコストを低下させる見通しが得られた．

仕様記述に基づきトップダウンに自動設計を行うアプローチのほかに，設計者と対話的に設計を進めようという試みもある．横田らは設計者が入力した概略データパスの補完，詳細化，矛盾，冗長の削除などを行う知識を知識ベース化して貯え，これに基づきデータパスの適切な改良の提案，あるいは自動対処を行う論理設計の知的支援系[12]を研究している（図-4）．そこでは，たとえば内部バスとトライステートゲートでない資源の出力が接続されていた場合に，それらの間にトライステートゲートを挿入するルールをLONLI (Prolog 系言語) を用い図-4(b)に示すように記述している．

また筆者らは，①自動設計後にもとの仕様が変更された場合に設計済みの回路の変更必要箇所を認識しデータパスを補正する，②装置の概略構造になんらかの制約条件が存在し，その概略構造に沿って設計を進めなくてはならない，あるいは③設計者の設計した概略データパスを動作仕様を満たすように補正することを狙いとしたデータパス補正プログラムを作成した[13]．そこでは図-5(a)に示すように，演算動作のデータパス上での実行可能性を，①演算動作に必要な機能（演算子）を実



（a） 演算器系合成手順



（b） 併合処理の概念

IF
1) there is an ALU,
2) there is an operation,
3) the ALU and the operation never work simultaneously,
4) the ALU does't include function of the operation,
5) the function of the operation is mergeable to the ALU,
6) the left input port of the ALU is already connected to the first operand of the operation,
7) the right input port of the ALU is already connected to the second operand of the operation,
8) the output port of the ALU is not connected to the sink of the operation,

THEN
1) add the required function with its condition to the ALU,
2) update the transfer condition from the first operand,
3) update the transfer condition from the second operand,
4) add a new path from the output port of the ALU to the sink with transfer condition to the ALU

（c） 演算器合成ルール例

図-3 演算器系合成手法の概要[11]

行可能なサブデータパスが存在し，②サブデータパスと演算動作で参照あるいは代入される資源との間にデータ転送路が存在することととらえている．そしてデータパスの不備の原因を次のように分類している．

① ハードウェア制約条件の違反

出力端子同士の結線，ファンアウト負荷制限オーバなどの論理仕様とは無関係に検出できる不備

② 機能の不足

③ データ転送路の不足

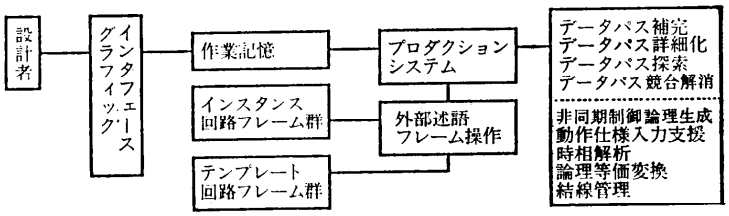
④ 資源競合

一般に不備は複合して起こり得，また不備の原因を唯一に特定することは困難である．とくに相違の程度が大きい場合には，その原因はさまざまに解釈できる．そこで筆者らは，図-5(b)に示すような失敗原因を複数組み合わせた失敗パターンに分類・整理し，各失敗パターンごとにそれを認識するルールを設けることとした．失敗パターンにより不備の原因と箇所を同定後，その原因ごとに図-5(c)に示すような補正ルールを適用している．約100個のルールを簡単なマイクロプロセッサのデータパスの補正に適用した結果，データパスと動作仕様の整合性が高い場合はこれらの手法が有効であるが，整合性が低い場合は複数の演算動作を同時に考慮した補正ルール，データパス全体を見渡した補正ルールを考案するなどの高度化が必要であることがわかった．

3.3 制御系設計

制御系設計の基本技術として論理関数の最小化，状態割り当て，オートマトン記述に基づく制御回路の自動設計など，さまざまな研究がなされている[14)-23)]．筆者らはこれら基本技術を統合し，かつさまざまな制御回路を生成しうる自動設計システムが重要であろうとの観点より，図-6 に示す制御回路自動設計システムを開発している[24),25)]．そこでは動作仕様記述とデータパス構造を入力とし，以下の手順で制御回路を合成している．

(1) 動作仕様に記述されている演算，データ転送などの動作をデータパス上でどのように実行するかを解析し，制御すべき構成要素とそれへの制御信号をもとめるのが制御点解析である．制御点解析の原理は，演算動作を実行するサブデータパスを探し，みつかったサブデータパスと演算動作で参照あるいは代入される資源との間のデータ転送路を探索することである．制御点解析を行うことにより，制御回路を設計するのに必要な論理的情報を得ることができる．



(a) 処理系の概略構成

```
rule (8,if,[[T,output_terminal],[connect,T,output(X)],
          [not,inserted,input(T)],
          [not,inserted,output(X)],
          [not,kind(X,A),tri_state]],
      then,[insert(output(X),input(T),[tri_state,N])]).
```

(b) トライステートゲート挿入ルール例

図-4 論理設計知的支援系[13)]

(2) 制御回路にもきわめてさまざまな実現スタイルが存在する．どの実現スタイルが良いかは設計対象に依存する．自動設計システムとして重要なことは，必要に応じさまざまな回路を合成でき，設計者に選択の余地を増やすことにあると思われる．図-7 に制御系の実現スタイルの例を示す．これは MEALY タイプに属する二つの例である．図-7(a)は次状態遷移と制御出力論理を一つの PLA で実現するスタイルであり，図-7(b)は次状態遷移と制御出力論理を別々の PLA で実現するスタイルである．筆者らは，これら実現スタイルをテンプレートと呼びテンプレートごとに制御点解析情報よりテンプレート各部の論理を導く規則を設ける手法をとっている．

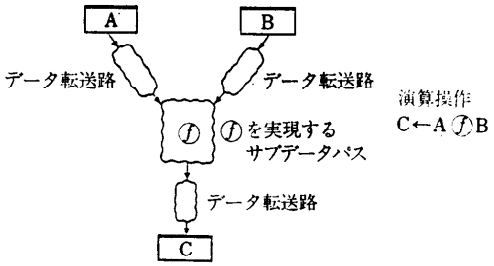
(3) 制御信号の多重化・符号化．同時には制御する必要のない機能ブロックへの制御信号を多重化，あるいは符号化することにより制御信号線本数，論理回路規模を削減することができる．これは水平型マイクロプログラムにおけるフィールド分割の問題と同様 NP 問題であり，制御対象の数が増加すると計算時間が急激に増加すること，および制御対象の種類によりフィールドの使い方が異なることより，発見的手法を使い，かつ制御対象の種類ごとに図-8 に示すようなコード化を行っている．

(4) 状態割り当ても NP 問題であるため，発見的手法を用いている．

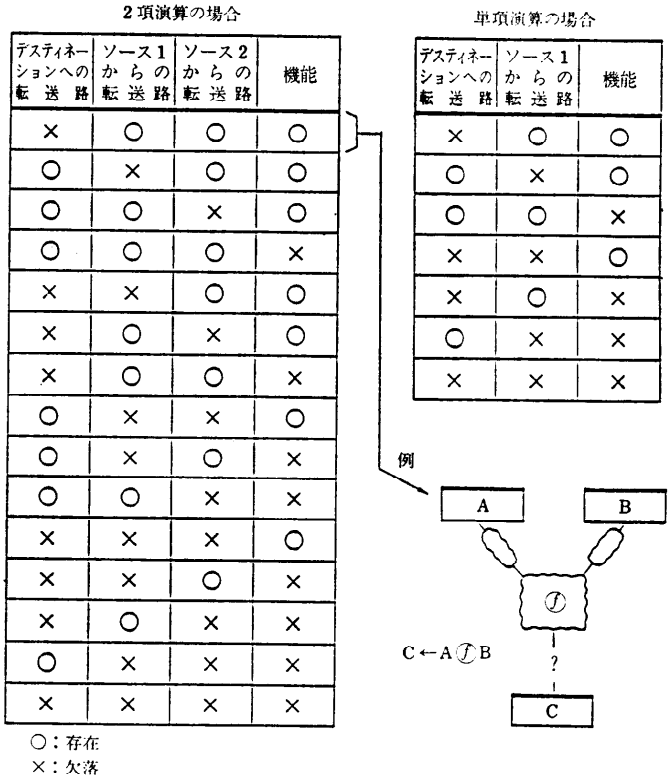
4. 論理設計と CAD

従来，論理設計レベルでは，①論理関数の最小化，因子化，PLA の面積最小化，②論理関数，記憶要素などをテクノロジイに依存した回路へ変換する手法，などの研究がなされてきた．回路変換についてはマクロ展開と冗長削除を組み合わせた方法あるいは極性伝播法などの研究が進み，与えられた論理関数が適切な形をしていれば人手設計に匹敵する変換結果が得られるレベルに近づいてきた[25)-27)]．

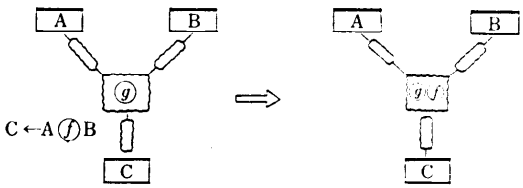
論理関数の最小化・因子化についても各種発見的手法により計算時間，合成品質が改善されつつある．これら手法では合成すべき回路の仕様を真理値表，ブール式などで与えキューブなどの内部表現に変換し処理するのが主流である．しかしこの手法で扱えるのは，最終解の積項数が高々数百程度におさまる，論理深度の浅い関数である．制御回路は論理深度の浅い関数となることが多いのに対し乗算器，ALU などデータパス系の機能ブロックは論理深度の深い関数となることが多い．したがって，これら論理深度の深い機能ブロックの合成には従来とは異なったアプローチが必要と思われる．論理回路のゲート量の多くは，これら論理深度の深い機能ブロックが占めており，従来の論理合成技術の適用範囲は限定されていた．一方論理設計技術書を紐解いてみると論理関数の最小化に割かれている頁数の比率は小さい．残りの大部分は演算器の構成法，装置全体の構成法など，のマクロなレベルの（抽象度の高い）設計技法が記述されている．実際の設計においても論理関数の最小化作業よりマクロなレベルの設計技術を具体的問題に適用し，論理回路を段階的に構成していく作業が多い．たとえば加算器設計を例にとると，技術書にはリップル加算方式，キャリ予測方式などさまざまな加算器構成法，各構成法のゲート量，速度に関する特徴が図あるいは文章で記述されている．設計者はこれら構成法を記憶しておきビット幅，性能など，具体的な加算器の仕様が与えられたときに適切な構成法を選択し，トップダウンに詳細な回



（a） 演算動作実行可能性検証の概念



（b） 失敗パターンの概念



（c） データパスの補正ルール例（機能追加）

図-5 データパスの補正の概念[13)]

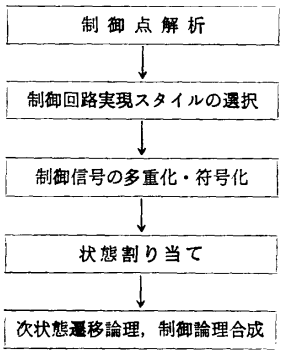
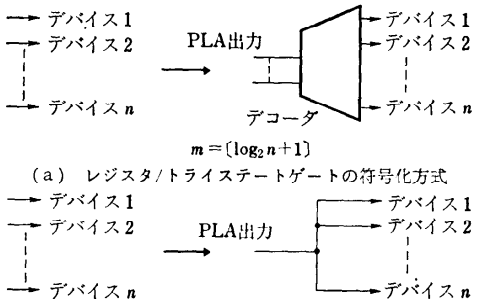


図-6 制御回路合成手順[24]



(a) レジスタ/トライステートゲートの符号化方式

(b) マルチプレクサ/ALU 制御信号の多重化方式

図-8 制御信号の符号化，多重化の概念[24]

路を生成している．

筆者らは回路の構成法ごとに多段の論理式を生成するプログラムを作成し，それを自動設計システムに登録し活用するアプローチをとっている[28),29)]．多段の論理式を生成する技法には，①ループを利用した式の生成，②再帰式を利用した式の生成，③定数の代入，④コードパターンの利用，⑤テンプレートと合成規則の利用，などさまざまな方法が考えられる．図-9 はループを利用して論理式を得る例を示す．この手続きは任意のビット幅 (Bit-width) のリップル加算器の論理式を生成することができる．図-9 において $T(i)$ は i ビット目の部分和を意味し，$G(i)$ は i ビット目のキャリ生成を意味し，$C(i)$ は i ビット目から $i+1$ ビット目へのキャリ伝播を意味し，$S(i)$ は i ビット目の和を意味する．

図-10 はテンプレートと合成規則を利用して多段の論理式を合成する例を示す．これは ALU を合成する手法の例である．このテンプレートの設定思想は，①ALU を出力段に排他的論理和を配し，キャリ予測論理，G信号生成論理，P信号生成論理，デコーダ論理から構成する．G_i は i ビット目からのキャリ生成を

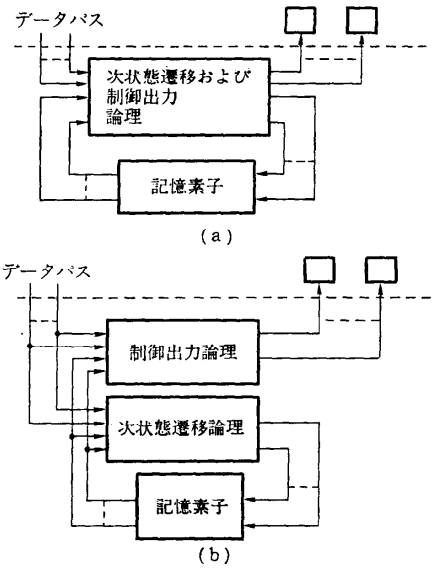
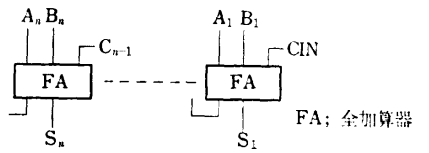


図-7 制御回路構成のモデル例[24]




```
DO i=1 to Bit−width
 COLLECT T (i)=A (i) @B (i); 部分和
 COLLECT G (i)=A (i) *B (i); キャリ生成
 COLLECT S (i)=T (i) @C (i); 和
 IF i=1
  THEN COLLECT C (i)=CIN    ; キャリ伝播
  ELSE  COLLECT C (i)=G (i − 1) orT (i − 1)
END−DO                          *C (i − 1)
```

図-9 多段の論理式の生成手続き例（リップル加算器）[28]

意図し，P_i は i ビット目の部分和を意図している．算術和実行時にはGおよびP信号よりキャリを予測し，出力段の排他的論理和に部分和Pとともに印可する．論理演算実行時には Carry-gate 信号を“0”とすることにより，キャリ予測論理の出力を“0”に抑止する．Pとして指定された論理演算実行結果が出てくるようにP生成論理を構成する．

テンプレートの段階ではデコーダ，P，G 生成論理などの詳細は未定である．与えられた機能仕様に基づきこれらの論理を合成していく．まず各論理を“0”に初期設定する．次に機能仕様のおのおのについてその構文のタイプを調べ，タイプごとに図-10 に例を示

したような規則に従ってテンプレート各部の論理を合成していく．図-10 の規則は機能仕様が3項 (α, β, Cin) 間の算術和の構文をもっていたならば P, G, Carry-gate の論理を図に示してあるような論理式に従って変更することを意味している．ただし α, β は算術和を含まない論理式である．機能仕様例中の第3番目がこのタイプに該当する．最後に各部ごとに論理関数の最小化手法を適用する．

このように機能ブロックのサブ構造を制限すると少ない計算時間で，人手設計に匹敵する合成結果が得られることがわかった．

機能ブロックの構成法は多数存在する．これらを蓄積していくことが重要と思われる．

標準セル方式では NAND あるいは，NOR などの SSI レベルの基本素子 (標準セル) を定めておき，論理式をこれら基本素子の結合関係に変換し (回路変換)，次にこれら基本素子を配置・配線 (レイアウト設計) して VLSI を構成するのが通例である．VLSI の面積は配置・配線アルゴリズムの良否に依存することが知られており，面積を低減させるさまざまなアルゴリズムが研究されてきた．一方，最近高位の機能仕様から直接マスクパターンを生成するシリコンコンパイラが研究されている．シリコンコンパイラの技法は高速にパターンを生成できるものの，現状ではまだアーキテクチャが固定されており，柔軟性にかける，あるいは生成されたチップの面積が大きくなるなどの改善の余地が多くある．シリコンコンパイラと標準セルの配置・配線方式の中間方式として，論理回路レベルで VLSI 上の概略レイアウトを指定し，これに基づき自動設計システムが詳細な配置の決定，配線を行う方式も考えうる．筆者らはこの考えに基づき，機能ブロックごとにその論理式自動生成手続きを記述するとともに，生成された論理式を VLSI 上でどのように相対配置するかの情報を生成する手続きを記述する方法をとっている．この情報と各論理式を回路変換した結果を付き合わせ基本素子の配置を決定し，配線を行うことにより ALU，乗算器などの規模が大きく規則的な回路構造をもつ機能ブロックに対し，少ない計算時間で良好な配置・配線結果を得た．

論理回路の規則性，クリティカルパス，信号の流れなどの性質をレイアウトに反映させることにより，チップ面積の減少，伝播遅延の分散の低下などが期待できる[30]．今後これら論理回路の特徴をレイアウトに反映させる研究が重要と思われる[31]．

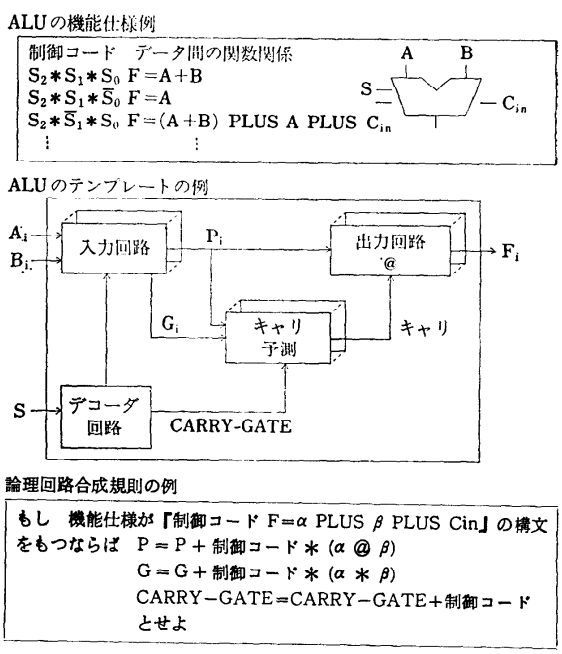


図-10 テンプレートを用いた論理回路合成の概念 (ALU の場合)[29]

5. むすび

論理回路の自動設計の手法に関して概観した．自動設計の研究は膨大で触れることができなかった研究成果も多いと思われる．自動設計は CAD 研究者の長年の夢であり一朝一夕には達成し得ない．特に性能，ゲート量などの制約条件が存在した場合，それに柔軟に対処できる自動設計アルゴリズムの研究は十分でなく今後の発展を期待したい．設計はオープンな問題であり一つの自動設計アルゴリズムですべての場合をカバーすることは困難と思われる．実用的なシステムを実現するにはいろいろな場面において，設計者とインタラクティブに共同作業をしやすくするインタフェース，さらには設計者個々の独創的設計手法をシステムに付加し他設計者がそれらを自由に使える，設計コミュニティを構成していけるような研究も重要と思われる．また論理回路の自動設計とシリコンコンパイラのアプローチのどちらが有望，あるいは融合して発達していくものなのか興味深いところである．

参考文献

1) 大特集：論理装置 CAD の最近の動向，情報処理，Vol. 25, No. 10 (1984).
2) 中村，小栗：ハードウェア記述言語とその応用，情報処理，Vol. 25, No. 10, pp. 1033-1040 (1984).
3) Hoshino, T., Karatsu, O. and Nakashima, T.: HSL-FX: A UNIFIED LANGUAGE FOR VLSI DESIGN, Proc. CHDL 85, pp. 321-336 (1985).
4) Thomas, D.E. et al.: Automatic Data Path Synthesis, IEEE Computer, Vol. 16, No. 59-70 (1983).
5) 竹沢，白井：アーキテクチャ設計支援エキスパート・システムの構成と設計環境，情報処理学会シンポジウム『VLSI CAD への知識工学の応用』，pp. 1-10 (1986).
6) Duley, J.R. and Dietmeyer, D.L.: Translation of a DDL Digital System Specification to Boolean Equation, IEEE Trans. Comput. Vol. C-13, No. 4. pp. 305-31 (1969).
7) Hoshino, T., Endo, N. and Karatsu, O.: An Automatic Logic Synthesizer for Integrated VLSI Design System, IEEE Custom Integrated Circuit Conference (1984).
8) 遠藤，星野，唐津：論理生成システム ANGEL の最適化手法，設計自動化研究会資料 23-5 (1984).
9) Duley, J.R. and Dietmeyer, D.L.: A Digital System Design Language (DDL), IEEE Trans. Comput. Vol. C-17, No. 9, pp. 850-861 (1968).
10) 高木：大局的設計知識を活用した LSI 合成エキスパートシステム：SAVE，情報処理学会シンポジウム『VLSI CAD への知識工学の応用』，pp. 27-35 (1986).
11) 高木：論理装置のデータパス構造自動合成の一手法，情報処理学会論文誌，Vol. 26, No. 4, (1985).
12) 横田，戸次，浜田：論理設計知的支援用推論方式のプロトタイピング，情報処理学会シンポジウム『VLSI CAD への知識工学の応用』，pp. 11-15 (1986).
13) 高木：データパスの検証・補正への1アプローチ，情報処理学会論文誌，Vol. 26, No. 1 (1985).
14) McClusky, E.J. Jr.: Minimization of Boolean Function, BSTJ, 35 (1956).
15) Hong, S.J., Cain, R.G. and Ostapko, D.L.: MINI: A Heuristic Approach for Logic Minimization, IBM J. Res. Develop., pp. 443-458 (1974).
16) Brown. D.W.: A STATE-MACHINE SYNTHESIZER, Proc. 18th DA Conference, pp. 301-305 (1981).
17) Brayton, R.K., Hachtel, G.D., Hemachandra, L.A., Newton, A.R. and Sangiovanni-Vincentelli, A.L.M.: A Comparison of Logic Minimization Strategies Using ESPRESSO: An APL Program for Partitioned Logic Minimization, Proc. ISCAS Rome, pp. 42-48 (1982).
18) 榎本，中島，村井，笹尾：組合せ回路合成システム—COMPO—，設計自動化研究会資料，20-1 (1984).
19) Brayton, R.K. and McMullen, C.T.: The Decomposition and Factorization of Boolean Expressions, ISCAS '82 pp. 49-54 (1982).
20) Brayton, R.K., Cohen, J.D., Hachtel, G.D., Trager, B.M. and Yun, D.Y.Y.: FAST RECURSIVE BOOLEAN FUNCTION MANIPULATION, Proc. ISCAS Rome, pp. 58-62 (1982).
21) 古関，山田：PLA 設計システム：PLAYER 回路とシステム研究会資料 CAS 84-132 (1984).
22) Mealy, G.H.: A Method for Synthesizing Sequential Circuits, Bell System Tech. J., 34: 5, 1045-1080 (1955).
23) Micheli, G.De.: OPTIMAL ENCODING OF CONTROL LOGIC, Proc. ICCD '84, pp. 16-22 (1984).
24) 高木：制御回路自動合成システム，電子通信学会論文誌，Vol. J69-D, No. 2 (1986).
25) Darringer, J.A.: A New Look at Logic Synthesis, Proc. 17th DA Conference, pp. 543-549 (1981).
26) Shinsha, T., Kubo, T., Hikosaka, M., Akiyama, K. and Ishihara, K.: POLARIS: PORARITY PROPAGATION ALGORITHM FOR COMBINATIONAL LOGIC SYNTHESIS, Proc. 21th DA Conference, pp. 322-328 (1984).
27) 斉藤，杉本：論理回路合成システム DDL/SX の開発，情報処理学会シンポジウム『VLSI CAD への知識工学の応用』，pp. 17-25 (1986).
28) Takagi, S.: DESIGN METHOD BASED LOGIC SYNTHESIS CHDL '85, pp. 49-63 (1985).
29) 高木：テンプレートを用いた ALU 回路の自動合成，電子通信学会論文誌，Vol. J68-D, No. 7 (1985).
30) 仲林，武藤，石川：論理構造を保存したレイアウト手法，電子通信学会回路とシステム研究会資料，CAS85-4 (1985).
31) 高木，仲林，石川：論理構造情報に基づくトップダウン配置手法，電子通信学会論文誌，Vol. J70-C, No. 1 (1987).

(昭和61年8月11日受付)