解　説

2. 方式・機能・論理設計における CAD

2.6 論　理　合　成†

南　谷　　崇††

1. まえがき

論理合成はスイッチング回路理論の創生期以来変わらぬ課題であり，その自動化は常に論理設計者の夢であった．初期の頃は，論理式簡約化，最小コスト多段回路網合成等，主として最適化アルゴリズムの研究に力点が置かれた．しかし最近では，半導体技術の驚異的進歩により VLSI チップ上に実現し得るシステムの規模が人手設計の可能な範囲を越えつつあることを背景として，最適設計でないことによる多少のコスト増を覚悟してでも「正しい」論理設計を「短時間に」完成させる必要のあることが，自動論理合成システム実現の主要な動機となっている[1)~3)]．

本稿では，ディジタルシステムの論理設計工程における論理合成問題とそこで用いられる基本的技法を整理し，最近の動向を概観する．個々のアルゴリズムに関してはそれぞれの文献に譲る．

2. 論理設計工程

ディジタルシステムの論理設計とは，所望の論理的な動作仕様を，一定の設計制約の下で，所与の基本素子で実現される物理的構造へ変換する営みである．動作仕様の表現としては，アルゴリズム，アーキテクチャ，命令セット，有限状態機械，論理関数，入出力時系列等，種々の形式が考えられ，それらの抽象度レベルも異なる．一方，与えられる基本素子には，電子回路素子（トランジスタ，ダイオード等），論理素子（AND, NAND, フリップフロップ等），機能素子（マルチプレクサ，デコーダ，加算器等），LSI 素子（PLA, ROM 等）など，規模及び形式における多様性があると同時に，それを実現する素子技術（バイポーラ，MOS 等）も多様である．また，課される設計制約としては，例えば，チップ面積，素子数，速度，電力，タイミング条件，クロック分配，ファンイン／ファンアウト制限，入出力ピン制限，テスト容易化設計規則，レイアウト容易化設計規則，等がある．

動作仕様の表現，使用できる基本素子，課される設計制約に応じて，例えば，

- データパスは機能集中型か，機能分散型か？
- その場合の処理は逐次型か，並列型か，あるいはパイプライン型か？
- 制御部はマイクロプログラム形式か配線論理形式か？ その場合の論理回路形式は PLA か，ROM か，個別の論理ゲートか？
- タイミング方式は同期式か，非同期式か？

等，種々の設計様式の選択が論理設計に先立つ段階でなされるのが普通である．

論理設計の自動化もしくは CAD を考える場合，変換の出発点である動作仕様の記述と到達点である基本素子の接続構造の記述とは，その抽象度（詳細化の程度）において大きな隔たりがあるので，通常，中間にいくつかの記述レベルが設定され，それぞれのレベルに適したハードウェア記述言語（本特集「ハードウェア記述言語とその応用」参照）を用いて各レベルの設計仕様が表現される．実際には設計対象によって中間レベルの記述形式はさまざまであり，各レベル間の境界も必ずしも明確ではないが，おおよそ**図-1** に示すような三つの階層にまとめることができる[1)]．

1) 動作記述レベル：実現すべきシステムの振舞いもしくはアルゴリズムを具体的実現構造と無関係に記

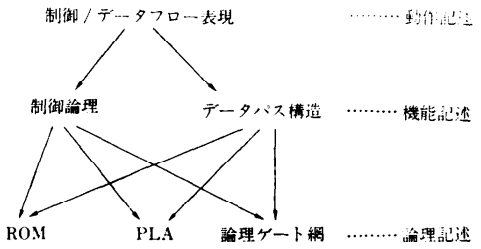


図-1　論理設計の階層

† Logic Synthesis by Takashi NANYA (Department of Computer Science, Tokyo Institute of Technology).

†† 東京工業大学工学部情報工学科

述するレベルであり，制御の記述とデータフローの記述を含む．アルゴリズムレベルあるいは命令セットレベルと呼ばれるレベルに相当する．

2) 機能記述レベル：演算器，レジスタ，マルチプレクサ等を基本要素としたデータパスの構造，及び1)で記述された動作に必要な演算，データ転送操作等の制御系列を記述する．レジスタ転送レベルと呼ばれるレベルに相当する．

3) 論理記述レベル：使用可能な論理素子あるいは論理モジュールを基本要素としたデータパス及び制御部の論理回路を表現する．

図-2 に論理合成の概念を示す．何らかの機能を果たすブロックAの動作（入力Xと出力Yの関係）の記述と，Aより単純な機能を持つ基本要素 a, b, c, d, e が与えられたとする．これらの基本要素の適当な結合で実現された構造で定められる入力xと出力yの関係がブロックAの動作と（定義されている範囲で）等価であるならば，このブロックAの抽象的な動作記述（上位記述）から具体的な構造記述（下位記述）への変換を論理合成という．構造記述における各基本要素は，それぞれの機能が定義されているので，必要ならばそれを動作記述としてさらに下位の構造記述へ変換される．また逆に，ブロックAを基本要素とした構造記述によって，さらに上位のブロックの動作記述を実現することもできる．

一般に，論理合成は一意的ではない．すなわち，ある（上位の）動作記述を実現する（下位の）構造記述はいくつもあり得る．同じ記述レベル内での異なる構造記述の間の変換，例えば AND, OR, NOT を基本要素とする回路から NANDを基本要素とする回路への変換を論理再変換という．特に，何らかの尺度に関して最適な構造記述への論理再変換は最適化という．**図-3** にこれらの関係を示す．

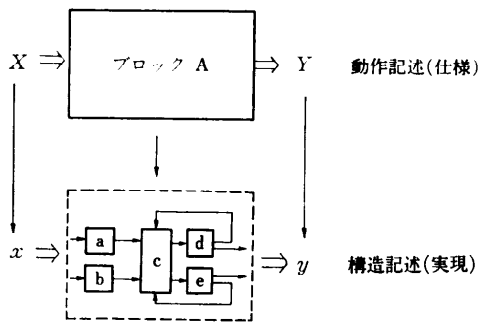


図-2　論理合成の概念

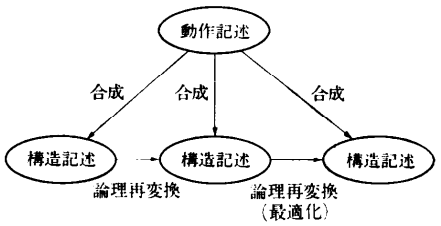


図-3　論理合成と最適化

結局，隣接する記述レベル間の論理合成で得られるいくつかの下位記述の中から設計制約あるいは何らかの最適化尺度に基づいて一つを選択することを，階層的に繰り返す工程が論理設計であると考えることができる．

3. 制御論理の合成

図-1 の動作記述レベルにおける制御／データフローの表現から，順序機械の状態遷移関数／出力関数，マイクロコード系列等，機能記述レベルの表現形式への変換である．

3.1 合成過程

図-4 は，従来別々に論じられている順序機械[4]形式（右列）と（水平型）マイクロプログラム[5]形式（左列）の簡約化過程を，互いに等価と考えられる二つの表現形式を横並びの同一レベルに配置して，示したものである．一般的な制御論理の合成過程は，以下に述べるように，これら二つの形式を併合したものと考えることができる．

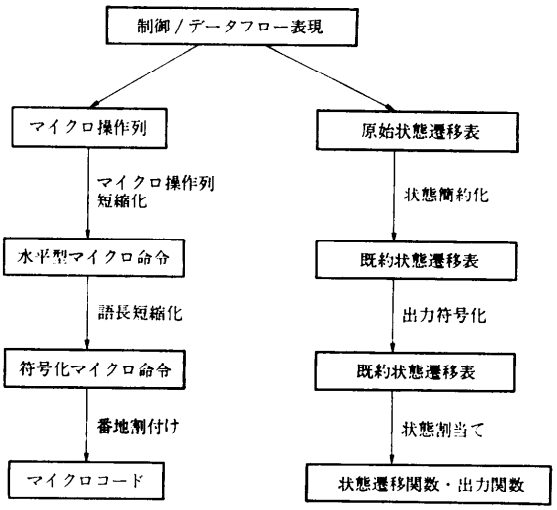


図-4　制御論理の合成

1) 記述言語に依存した前処理によって，動作仕様から，データの転送，演算等の基本動作の系列と，制御フローの分岐条件を求める．（マイクロ操作系列の記述または原始状態遷移表の作成に相当する．）

2) 1)の基本動作系列において，データの相互従属性及び（もし与えられているならば）データパス資源の競合を考慮して，互いに並列動作可能な基本動作を一つの内部状態における動作としてまとめる．（マイクロ操作列短縮化[6]または内部状態の等価性あるいは両立性の概念による順序機械の状態簡約化[4]に相当する．）この結果，直接制御方式の水平型マイクロ命令系列または順序機械の既約状態遷移表に相当する制御論理の表現が得られる．

3) 互いに排他的な制御信号同士をグループにまとめて，各グループを符号化する．（水平型マイクロ命令の語長短縮化[7]または順序機械の出力符号化に相当する．）この結果，符号化制御方式の水平型マイクロ命令系列または出力の符号化された既約状態遷移表に相当する制御論理の表現が得られる．

4) 順序機械形式で実現する場合には，内部状態を符号化する（状態割り当て[4]）．その結果，内部状態遷移条件を表す論理関数（状態遷移関数[4]）とデータパス制御条件を表す論理関数（出力関数[4]）が得られる．一方，マイクロプログラム形式で実現する場合には，「次に実行すべきマイクロ命令の番地」を考慮して制御記憶への格納番地を割り付けることにより，マイクロコード系列が得られる．

3.2 問題の定式化と現状

マイクロ操作列短縮化は，系列の先頭以外に入口を持たず末尾以外に出口あるいは分岐点を持たないマイクロ操作列を対象とする局所的短縮化と，分岐点を含む制御フロー全体を対象とする大域的短縮化に分けられる．当然，後者の方が前者の積重ねより短縮化の効果は良いが，その計算の手間は非常に大きくなり，最適解を求めることは事実上困難である[8),9)]．

局所的短縮化問題も，最適解を保証するアルゴリズムは，少なくとも NP 完全問題と同程度の計算の手間を要することが示されているが[6]，最適解を必ずしも保証しないならば，多項式オーダのアルゴリズムがいくつか提案されている[10),11)]．

順序機械の状態数最小化は，極大両立集合を求める問題を含む[12]．これはグラフの極大クリークを求める問題と等価であり，クリーク問題は NP 完全であることが知られている[13]．

マイクロ命令系列が与えられ，各マイクロ命令が並列動作可能な複数個のマイクロ操作から成るとき，同一のマイクロ命令には決して現れないマイクロ操作同士，すなわち互いに排他的なマイクロ操作同士を一つのグループにまとめて符号化することによりマイクロ命令の語長を短くすることができる．これがマイクロ命令の語長短縮化である．二つのマイクロ操作が同じマイクロ命令に含まれることがないとき両者は両立的であると定義すると，語長短縮化は，マイクロ命令系列からすべての極大両立集合を求め，その中から解となり得る両立集合の組を選ぶ問題になる．したがって，語長最小化問題もクリーク問題に帰着するので，NP 完全である．

順序機械の状態割当ても古典的な問題であり，得られる論理関数が簡単な形式になるような状態割当て法の研究がなされているが[14]，最適化の尺度がはっきりしないため，一般的な最適解というものはまだ与えられていないといってよい．これと等価なマイクロプログラムの番地割付け最適化に関してもまだ十分な解は与えられていない[15]．

これまでに PLA を対象とした順序回路合成システムが多数報告されているが，いずれも 5. で述べる論理最小化と畳込みの自動化に主眼を置いており，順序機械の状態数縮小や状態割当て等，制御論理の合成を自動化したものは少ない[16)~23)]．

マイクロプログラム制御の合成に関しては，データパス構造と制御フローを入力として，ヒューリスティックなアプローチでマイクロコード圧縮をした結果，人手設計に比べて，並列性は同程度で 14% ほど大きな制御記憶を自動生成できたとの報告がある[24]．

4. データパスの合成

図-1 の動作記述レベルにおける制御／データフロー表現から，機能記述レベルにおけるデータパス構造への変換である．この場合，データパスの構造記述で用いられる基本要素自身の具体的実現についてはブラックボックスとする．

4.1 合成過程

データパスの基本要素は，

記憶要素：汎用レジスタ，スタックポインタ，

演算要素：算術／論理演算器，シフタ等，

結合要素：バス，マルチプレクサ，双方向ゲート等，

の 3 要素に分けられる．

データパス合成の手順は次のようになる．

1) 「制御論理の合成」における手順 1)と同様にして基本動作系列（マイクロ操作列）を，また手順 2)と同様にして，並列動作可能な基本動作群（水平型マイクロ命令）の系列を求める．ただしここでは，データパス資源はまだ与えられていないので，並列動作可能性の判定は，データの相互従属性のみに基づく．

2) 1)の基本動作群系列に現れる変数及び演算をできるだけ少ない数のレジスタ及び演算器に対応させる．

3) もし可能ならば，いくつかのレジスタをまとめてスクラッチ・パッド・メモリを構成する．また，いくつかの異なる種類の演算器をまとめて ALU を構成する．

4) レジスタ，スクラッチ・パッド・メモリ等の記憶要素と単能演算器，ALU 等の演算要素をできるだけ少ない数の結合要素で接続する．

4.2 問題の定式化と現状

無向グラフの節点集合を，各ブロックがそれぞれ極大クリークを構成するように，できるだけ少ない数のブロックに分割する問題をクリーク分割問題という．上記の合成手順 2)～4)は以下に述べるように三種類の問題に分けられるがそれらはいずれもクリーク分割問題として定式化される[28]．

1) 記憶要素割付け問題：二段階に分かれる．

ⅰ）与えられた基本動作系列において，ある変数の値が定義 (WRITE) されてから，最後に参照 (READ) されるまでの期間を，その変数のライフタイムと呼ぶことにすると，ライフタイムが互いに重ならない二つの変数同士は，データパスにおいて物理的に同一のレジスタに割り付けることができる．したがって，変数をグラフの節点とし，ライフタイムの重ならない変数に対応する節点同士を枝で結んだグラフを G とすると，できるだけ少ないレジスタを割り付ける問題は G に対するクリーク分割問題になる．

ⅱ）与えられた動作系列において互いに同時にはアクセスされない複数のレジスタは，一つのグループにまとめてスクラッチ・パッド・メモリとすることができる．したがって，レジスタをできるだけ少ない数のメモリにまとめる問題はやはりクリーク分割問題に帰着する．これはまた **3.2** で述べた水平型マイクロ命令の語長短縮化問題と本質的に同じである．

2) 演算要素割付け問題：

与えられた動作系列において同時には実行されない同じ種類の演算は，データパスにおいては物理的に同一の演算器に割り付けることができる．また，同時には実行されない異なる種類の演算は，一つのグループにまとめて ALU を構成する．これも各演算を節点とするグラフにおいて同時には実行されない節点同士を枝で結べばクリーク分割問題として定式化される．

3) 結合要素割付け問題：

記憶要素間あるいは記憶要素と演算要素の間をデータ転送路で結ぶ場合，転送元あるいは転送先が同一である複数の転送路を一つにまとめて共通バスとすると，ハードウェア要素が簡素化され都合が良い．そこで，与えられた動作系列において交換律の成り立つ二項演算子に関するオペランドは適当に交換し，同時には使用されない結合（変数と演算子または変数と変数の順序対）はまとめてバス構造とする．この場合，各結合を節点とするグラフにおいて同時には使用されない節点間を枝で結ぶとやはりクリーク分割問題になるが，転送元あるいは転送先が同一である節点間の枝を優先的に処理してこの問題を解けばよい．その結果，一つの入力ポートに複数のバスが接続される場合にはマルチプレクサが挿入される．

上記のように，データパス合成問題の最適解を見つけることはいずれも極大クリークを求める問題もしくは等価的に極大両立集合を求める問題に帰着される．

データパス合成システムは，主として，カーネギ・メロン大学 CMU-DA システムの一環として精力的に研究と開発が行われている[25)-30)]．上記の定式化に従ってクリーク分割問題に対する近似的最適アルゴリズムの開発[28)]，ISP 記述[31)]からデータパスへの直接的な変換[29)]，データパスのコスト最小化[27)]，知識ベース・エキスパート・システムによるデータパス生成[26)]，演算順序とデータ記憶を時間の順序関係で記述したデータパス・モデルに対する整数計画法としての定式化[30)]，等が研究されている．

5. 論理回路の合成と変換

図-1 における機能記述レベルの基本要素の動作仕様表現もしくは論理関数から，論理記述レベルの構造表現である論理式，PLA/ROM パターン，論理接続図等への変換である．普通，狭義の論理合成とはこのレベルの変換を指す．

5.1 合 成 過 程

図-5 に，論理記述レベルでの三つの基本要素として ROM, PLA, 個別論理ゲートを用いた場合の論理回路の合成過程を示す．

1) 記述言語に依存する前処理によって，機能記述レベルの各基本要素（演算器，デコーダ，マルチプレクサ等）の動作を定義する論理関数，及び制御論理を定義する論理関数（状態遷移関数，制御出力関数，マイクロコード等）を求める．

2) 論理関数を，真理値表，キューブ表現，論理式等で表し，それぞれ ROM パターン，PLA パターン，論理ゲート接続表現とする．図-6 に同じ論理関数の三種類の記述形式と対応する実現形式を示す．

3) PLA パターンと論理ゲート接続表現について，もし可能ならば，与えられた設計制約及び素子技術制約のもとで，論理最適化あるいは論理再変換を行う．

4) PLA パターンについて，もし可能ならば，畳込みを行う．

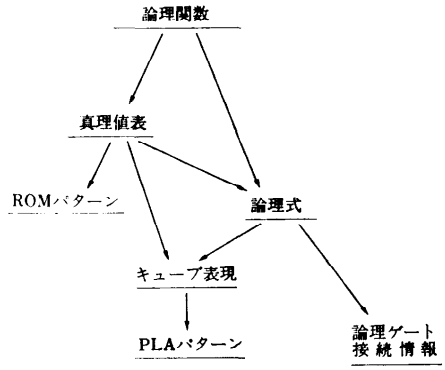
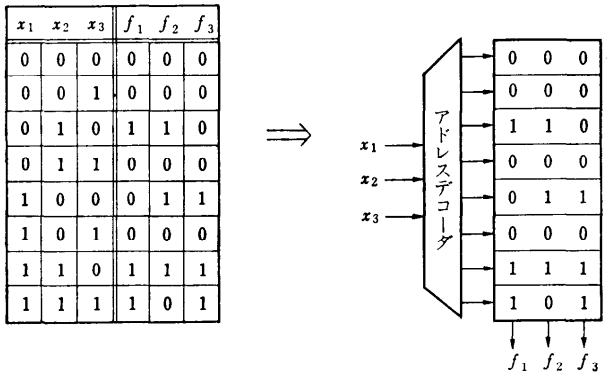
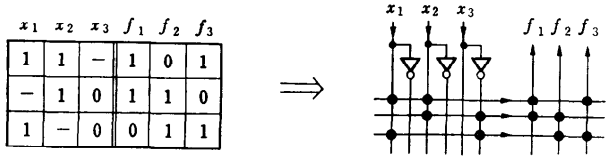


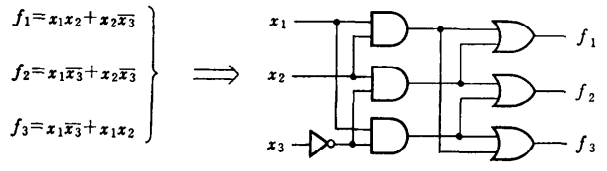
図-5　論理回路の合成



（a）　ROM による真理値表の実現



（b）　PLA によるキューブ表現の実現



（c）　論理ゲート網による論理式の実現

図-6　論理関数の実現

5.2　問題の定式化と現状

レジスタ転送レベルにおける論理機能もしくは論理関数の記述から論理式への変換[32]は，一般に記述言語に依存するが，最適化を考えなければこの変換は素直に行えるので，これまでにこのレベルの論理合成システムが多数作られている[1),2)]．しかし，種々の設計制約の下での最適化を考えると人手設計に劣るのでまだほとんど実用に供されていないのが現状のようである[3)]．論理最小化は古典的問題であり，まずすべての主項を求め，次にそれらの主項の最小被覆を求める，といういわゆる Quine-McCluskey の方法を基にして，主項の生成及び最小被覆に関する多くのアルゴリズムが知られている[33)]．しかし，これらの方法が実用に使用できる範囲は小さい[34)]．

PLA は積和2段構成の論理回路と等価な論理形式であることから，最近では PLA 論理最小化のための種々のヒューリスティック・アルゴリズムが提案され[35)~39)]，またそれらに基づいた PLA 自動合成システムが実現されている[17)~21)]．

AND アレイと OR アレイからなる PLA の基本構造を前提として PLA パターンの論理最小化（積項数最小化）を行っても，この基本構造自体は LSI チップ上の物理的配置として面積最小の形式とはいえない．PLA の畳込みとは，AND-OR 2段形式という PLA の基本的性質を失うことなく，一定の境界条件の下で PLA パターンが物理的に占める面積を最小にしようとする変換である[40)~42)]．PLA の畳込み問題もやはり NP 完全であるが[43)]，ヒューリスティック・アルゴリズムあるいはそれに基づいた PLA 自動合成システムが多数発表されている[44)~50)]．

ゲートアレイ，標準セル方式等で論理回路を実現する場合には，論理記述レベルでの基

本要素は NAND, AND-NOR, マルチプレクサ, フリップフロップ等, 多くの種類があり, 機能の大きさも多岐にわたる. また素子技術に依存した設計の制約も種々あり得る. このような場合, 大域的な最適化は非常に困難なので, 例えば,

$A \cdot /A = 0$

$A + /A = 1$

$A + /A \cdot B = A + B$

のようなブール代数の基本的な公式を利用した局所的な変換をヒューリスティックに繰り返し, 設計コストの許す範囲で所期の性能, 制約を満たす方法が用いられることが多い[3),51),52)].

上に述べたような種々の技法を用いて, これまでに多様な論理合成システムが発表されている. それらの内で代表的なもののいくつかは, 文献 1), 2)で紹介されている.

6. むすび

ディジタルシステムの論理合成問題を整理し, その最近の動向を概観した. 現在のところ, PLA 構造に向けての自動論理合成が, その論理設計, レイアウト, 検査の容易さゆえに, 実用レベルで成功する望みが最も高いように思われる. また, 最近では, 抽象度の非常に高いレベルからの論理合成の試みもあり, 注目に値する[53)].

参考文献

1) Thomas, D. E.: The Automatic Synthesis of Digital Systems, Proc. IEEE, Vol. 69, No. 10, pp. 1200-1211 (1981).
2) Shiva, S. G.: Automatic Hardware Synthesis, Proc. IEEE, Vol. 71, No. 1, pp. 76-87 (1983).
3) Lipp, H. M.: Methodical Aspects of Logic Synthesis, Proc. IEEE, Vol. 71, No. 1, pp. 88-97 (1983).
4) 当麻, 内藤, 南谷: 順序機械, 岩波書店 (1983).
5) 萩原 宏: マイクロプログラミング, 産業図書 (1977).
6) Landskov, D. et al.: Local Microcode Compaction Techniques, Comput. Surv., Vol. 12, No. 3, pp. 261-294 (1980).
7) Dasgupta, S.: The Organization of Microprogram Stores, Comput. Surv., Vol. 11, No. 1, pp. 39-65 (1979).
8) Fisher, J. A.: Trace Scheduling : A Technique for Global Microcode Compaction, IEEE Trans. Comput., Vol. C-30, No. 7, pp. 478-490 (1981).
9) Tokoro, M. et al.: Optimization of Microprograms, ibid., pp. 491-504 (1981).
10) Dasguputa, S. and Tartar, J.: The Identification of Maximal Parallelism in Straight-line Microprograms, IEEE Trans. Comput., Vol. C-25, No. 10, pp. 986-992 (1976).
11) Ramamoorthy, C. V. and Tsuchiya, M.: A High-level Language for Horizontal Microprogramming, IEEE Trans. Comput., Vol. C-23, No. 8, pp. 792-801 (1974).
12) Unger, S. H.: Asynchronous Sequential Switching Circuits, John Wiley & Sons (1969).
13) Aho, A. V., Hopcroft, J. E. and Ullman, J. D.: The Design and Analysis of Computer Algorithms, Addison-Wesley (1974).
14) Miller, R. E.: Switching Theory, Vol. I & II, John Wiley & Sons (1965).
15) 馬場敬信: ファームウェア工学, 情報処理, Vol. 20, No. 7, pp. 622-646 (1979).
16) 南谷 崇: PLA の使い方, 産報出版 (1978).
17) 壺屋, 高木, 寺本, 南谷: PLA 設計サポートシステム, 情報処理学会第 19 回全国大会論文集, pp. 611-612 (1978).
18) Brown, D. W.: A State-Machine Synthesizer-SMS, 18th Design Automation Conf., pp. 301-305 (1981).
19) Kang, S. and vanCleemput, W. M.: Automatic PLA Synthesis from a DDL-P Description, ibid., pp. 391-397 (1981).
20) Teel, B. and Wilde, D.: A Logic Minimizer for VLSI PLA Design, 19 th Design Automation Conf., pp. 156-162 (1982).
21) 宮下, 竹田, 杉山: PLA 設計支援システム PLACAD, 研究実用化報告, Vol. 32, No. 6, pp. 1339-1352 (1983).
22) Edwards, M. D. and Aspinall, D.: The Synthesis of Digital Systems using ASM Design Techniques, Computer Hardware Description Language and their Applications, eds. Uehara, T. and Barbacci, M., North-Holland, pp. 55-64 (1983).
23) Forrest, J. and Edwards, M.D.: The Automatic Generation of Programmable Logic Arrays from Algorithmic State Machine Descriptions, VLSI 83, eds. Anceau, F. and Aas, E. J., North-Holland, pp. 183-193 (1983).
24) Nagle, A. W. and Parker, A. C.: Algorithms for Multiple-criterion Design of Microprogrammed Control Hardware, 18 th Design Automation Conf., pp. 486-493 (1981).
25) Tseng, C.-J. and Siewiorek, D. P.: The Modeling and Synthesis of Bus Systems, ibid., pp. 471-478 (1981).
26) Kowalski, T. J. and Thomas, D. E.: The VLSI Design Automation Assistant : Prototype System, 20 th Design Automation Conf., pp. 479-

483 (1983).
27) Hitchcock Ⅲ, C. V. and Thomas, D. E.: A Method of Automatic Data Path Synthesis, ibid., pp. 484-489 (1983).
28) Tseng, C.-J. and Siewiorek, D. P.: Facet: A Procedure for the Automated Synthesis of Digital Systems, ibid., pp. 490-496 (1983).
29) Hafer, L. J. and Parker, A. C.: Automated Synthesis of Digital Hardware, IEEE Trans. Comput., Vol. C-31, No. 2, pp. 93-109 (1982).
30) Hafer, L. J. and Parker, A. C.: A Formal Method for the Specification, Analysis and Design of Register-Transfer Level Digital Logic, IEEE Trans. CAD, Vol. CAD-2, No. 1, pp. 4-18 (1983).
31) Barbacci, M. R.: Instruction Set Processor Specification (ISPS): The Notation and Its Applications, IEEE Trans. Comput., Vol. C-30, No. 1, pp. 24-40 (1981).
32) Duley, J. R. and Dietmeyer, D. L.: Translation of a DDL Digital System Specification to Boolean Equations, IEEE Trans. Comput., Vol. C-18, No. 4, pp. 305-313 (1969).
33) Muroga, S.: Logic Design and Switching Theory, John Wiley & Sons (1979): (邦訳) 室賀, 笹尾: 論理設計とスイッチング理論, 共立出版 (1981).
34) 石川, 笹尾, 寺田: 論理式最小化の一手法とその適用限界について, 信学論 [D], Vol. J 65-D, No. 6, pp. 797-804 (1982).
35) Hong, S. J. et al.: MINI: A Heuristic Approach for Logic Minimization, IBM J. Res. Dev., Vol. 18, pp. 443-458 (1974).
36) Svoboda A. and White, D. E.: Advanced Logical Circuit Design Techniques, Garland Press (1979).
37) Arevalo, Z. and Bredeson, J. G.: A Method to Simplify a Boolean Function into a Near Minimal Sum-of-products for Programmable Logic Array, IEEE Trans. Comput., Vol. C-27, No. 11, pp. 1028-1039 (1978).
38) Brayton, R. K. et al.: A Comparison of Logic Minimization Strategies Using ESPRESSO: An APL Program Package for Partitioned Logic Minimization, Proc. of ISCAS 82, pp. 42-48 (1982).
39) Martinez-Carballido, J. F. and Powers, V. M.: PRONTO: Quick PLAP roduct Reduction, 20-th Design Automation Conf., pp. 545-552 (1983).
40) Wood, R. A.: A High Density Programmable Logic Arrays, IEEE Trans. Comput., Vol. C-28, No. 9, pp. 602-608 (1979).
41) 稲垣耕作: プログラマブルロジックアレーの高密度設計法, 信学論 [D], Vol. J 63-D, No. 9, pp. 739-746 (1980).
42) Hachtel, G. D. et al.: An Algorithm for Optimal PLA Folding, IEEE Trans. CAD, Vol. CAD-1, No. 2, pp. 63-77 (1982).
43) Luby, M. et al.: Some Theoretical Results on the Optimal PLA Folding Problem, Proc. ICCC 82, pp. 165-170 (1982).
44) Hachtel, G. D. et al.: Techniques for Programmable Logic Array Folding, 19th Design Automation Conf., pp. 147-155 (1982).
45) Grass, W.: A Depth-first Branch-and-bound Algorithm for Optimal PLA Folding, ibid., pp. 133-140 (1982).
46) DeMan, H. et al.: PLASCO: A Silicon Compiler for nMOS and CMOS PLA's, VLSI 83, ieds. Anceau, F. and Aas, E. J., North-Holland, pp. 171-181 (1983).
47) DeMicheli, G. and Sangiovanni-Vincentelli, A.: PLEASURE: A Computer Program for Simple/Multiple Constrained/Unconstrained Folding of Programmable Logic Arrays, 20th Design Automation Conf., pp. 530-537 (1983).
48) Liu, W. and Atkins, D. E.: Bounds on the Saved Area Ratio due to PLA Folding, ibid., pp. 538-544 (1983).
49) Hu, T. C. and Kuo, V. S.: Optimum Reduction of Programmable Logic Array, ibid., pp. 553-558 (1983).
50) Stebnisky, M. W. et al.: APSS: An Automatic PLA Synthesis System, ibid., pp. 430-435 (1983).
51) Darringer, J. A. and Joyner, W. H.: A New Look at Logic Synthesis, 17 th Design Automation Conf., pp. 543-549 (1980).
52) Vaidya, A. K., Dietmeyer, D. L. and Engh, M. K.: WISLAN—Technology Transformation and Optimization, in: Computer Hardware Description Language and their Applications, eds. Uehara, T. and Barbacci, M., North-Holland, pp. 43-54 (1983).
53) Subrahmanyam, P. A.: Synthesizing VLSI Circuits from Behavioral Specifications: A Very High Level Silicon Compiler and its Theoretical Basis, VLSI 83, eds. Anceau, F. and Aas, E. J., North-Holland, pp. 195-210 (1983).

(昭和59年7月25日受付)