プリント配線板自動設計支援システム

吉田 武史*, 久津輪 敏郎**,
小堀 研一**, 江端 克彦***

An Automatic Supporting System for PWB Design

Takeshi YOSHIDA*, Toshiro KUTSUWA**,
Ken-ichi KOBORI** and Katsuhiko EBATA***

*大阪工業大学大学院電気工学専攻 (〒535 大阪府大阪市旭区大宮5-16-1)

**大阪工業大学電子工学科 (〒535 大阪府大阪市旭区大宮5-16-1)

***広島電機大学電子工学科 (〒739-03 広島県広島市安芸区中野6-20-1)

*Master Course Electrical Engineering, Osaka Institute of Technology (5-16-1 Omiya, Asahi-ku, Osaka-shi, Osaka 535)

**Department of Electronic Engineering, Osaka Institute of Technology (5-16-1 Omiya, Asahi-ku, Osaka-shi, Osaka 535)

***Department of Electronic Engineering, Hiroshima-Denki Institute of Technology (6-20-1 Nakano, Aki-ku, Hiroshima-shi, Hiroshima 739-03)

概要 ICやLSIなどの論理素子は近年急速に発展し、その用途は広範囲にわたるものとなった。それに伴い、デジタルシステムの設計も重要視されてきた。その設計手法としては、コンピュータを利用した支援システムを用いるものが主流となっている。その中でも最近ではハードウェア記述言語を用いた手法が普及しつつある。しかしその反面、これらのシステムは大規模で、非常に高価格な上、言語や論理回路等についてかなり高度な知識を必要とし、ユーザが容易に設計できるものではない。ユーザに非常に負担をかけているのが現状であり、設計の容易化、自動化の目的にはほど遠いものである。筆者らはこの目的を達成するために、手書き状態線図を入力とする小規模ながら容易に設計できるシステムの開発を行ってきた。手書き状態線図を入力とすることは種々の困難があるが、将来設計自動化を進めるにはこれの可否が重要と考えられるので、その研究を行い成果を上げることができた。また、既存のプリント配線板CADシステムとリンクすることによって論理設計から配線板設計まで一貫して行えるようになった。

Abstract

Recently, digital systems have been applied to various industrial fields with advance of IC and LSI. The design systems using hardware description languages have been also developed for design automation of LSI. But these systems are very expensive and difficult to use. Therefore, we have developed an automatic design system that is easily suitable for designing small digital systems. This system can be used without special knowledge and technique about hardware design. This system provides designers total design environment from logical specification to PWB layout.

1. まえがき

ICやLSIなどの論理素子はさまざまな産業分野で利用され、急速に発展し続けてきた。それに伴いデジタルシステムの設計も重要視され、その設計方法、技術もさまざまなものが提案され、設計環境は急速に整備されつつある。中でもコンピュータを用いた設計支援システムは、デジタルシステムを

設計する上で重要な設計手段の1つになっており、ハードウェア記述言語 (Hardware Description Language)¹⁾であるVHDLやVerilog-HDLなどは、積極的に導入されつつある。ハードウェア記述言語を用いることにより、ソフトウェアにおける高級記述言語によるプログラミングと同等な程度にまで設計の自動化が図れるようになった。

一方でこうしたシステムは非常に高価格であり、

また言語に対する相当高度な知識を必要とする。すなわち、ハードウェア技術者が記述言語を利用して設計するためには、言語の記述（プログラミング）について相当に習熟していなければならず、細かなサブモジュールを多数用意する必要がある、非常に面倒である。

一方、従来のソフトウェア技術者がこれを利用しようとするれば、要素定義や動作定義などのために論理回路やハードウェアについてかなりの知識を必要とし、容易に設計できるものではない。ユーザに非常に負担をかけているのが現状である。

このような困難な点を改善するために状態線図（手書きでない）や流れ図をCRT上で対話的に処理して設計するものも開発されている²⁾。

しかし筆者らは、将来さらに設計の容易化、自動化を進めていくためには、手書きの状態線図の入力の可否が重要であると考えて、あえてこれについて研究を進めてきた。

手書き状態線図を利用する場合は、文字の認識など困難な問題が多いが、これについて成果を得ることができた。また既存のプリント配線板CADシステムとリンクすることによって、論理設計から配線板設計まで一貫して行えるようになった。

なお本システムはパソコンをベースとした小規模なシステムであり、設計できる配線板も比較的小規模のものになっている。しかし、今後はさらに記述言語等とのデータの互換性をはかり、機能回路やサブモジュールの作成など手間のかかる部分に本システムを利用するなど、互いに相補う形で利用すれば設計の容易化、自動化が一層図れるものとする。

2. システムの概要

本システムは、図1に示すように大きく3つの処

図1. 本システムの構成

理部に分けられる。設計順にまず手書き状態線図認識部、次に論理合成部、最後にプリント配線板ネットリスト作成部である。

手書き状態線図の認識部では、まず基本設計仕様を基に、回路動作と各状態および入出力の関係を状態線図で表す。この状態線図をイメージスキャナによって画像データとしてコンピュータ上に取り込む。そして、図形要素の認識を行い、遷移表データとして出力させ、次の論理合成部の入力データとする。

論理合成部では組合せ回路、順序回路の設計を対話形式で行っていくことができる。ここでの順序回路の設計データには、前述のように手書き状態線図認識部で得られた遷移表データを利用する。また組合せ回路の設計では、入力データとして論理式を与えることにより設計を行うことができる。設計方法としては、各入力データや共通変数の設定、フリップフロップの選択などがほぼ自動的に行え、かつ対話形式で容易に修正変更できるため、設計者に負担をかけることはない。こうして最終結果としては合成、簡略化された論理式が得られる。

最後のプリント配線板ネットリスト作成部では、市販されているプリント配線板作成用CADであるOrCAD-PCB IIにデータを受け渡して設計できるようにするために、合成後得られた論理式を展開し、ネットリストを作成する。

3. 各処理部の処理方法

3.1 状態線図認識部^{3),4)}

状態線図認識部における認識手順を図2に示す。

図2. 手書き状態線図の認識処理の流れ

まず手書きの状態線図を、イメージスキャナにより読み込ませる。このときの入力用紙サイズは、A 4 までの用紙での読み込みが可能である。また複雑な状態線図に対しては、状態線図を分割し、複数の用紙にわたって記入し入力を行い、認識後、自動的に統合を行う方法によって対応している。

次に、取り込まれた画像を基に、円、矢印、文字の順に認識を行う。状態線図では円は状態を表し、矢印は状態の移行、つまり遷移関係を表す。また文字は状態名および入出力関係を表す。したがって円から順に矢印、文字と認識することによって合理的に認識できると考えられる。円、矢印の認識はそれぞれに対して条件を設定し、図形の輪郭を追跡する輪郭線追跡法⁹⁾(Border following)により、それぞれの図形の条件を満たしているかを判別することによって認識を行う。

文字位置は状態を表す円内部、矢印の近辺を探索することにより抽出することができる。こうして抽出された文字位置より文字の切り出しを行い、認識を行う。文字認識は本システムでは、階層型のニューラルネットワーク⁹⁾を用いている。

ニューラルネットワークは、認識パターンに対して雑音などが含まれた場合でも高い汎化能力を示す反面、ずれに対しては弱い特徴がある。そこで、認識率を向上させるために、文字認識を行う前に認識対象になる文字に対して、大ききの正規化、位置の正規化⁹⁾などの文字の加工処理を施している。

本システムで用いているニューラルネットワークの構造は入力層、中間層、出力層の3層構造であり、各層のユニット数は入力層256個、中間層60個、出力層6個となっている⁷⁾。状態線図中で利用できる文字は0, 1, 2, 3, S, /の6文字である。これら6文字をあらかじめ用意したパターンを学習することによって、認識させることができる。文字は入出力関係を表す文字は5文字まで、状態名を表す文字は3文字まで使用できる。

次に認識結果のCRT上での確認および修正は、本システムではエディタを用いて行う。エディタの操作はマウスのみで行うことができるため、簡単に修正を行うことができる。こうして認識された結果として正しければ遷移表形式で出力される。

図3. 論理合成処理の流れ

3.2 論理合成部^{8),9)}

論理合成部では、状態線図認識部で得られた遷移表データを順序回路設計の入力データとして、論理式に自動変換し合成を行う。また、同時に式入力によって組合せ回路の自動合成も行うことができる。合成部の特徴としては次のようなものがあげられる。

- 1) 組合せ回路の入力には通常使用している論理式を直接入力できる。
- 2) 論理式の簡略化は、木探索法のアルゴリズム^{10)~12)}を採用することにより、高速化を図っている。
- 3) マンマシンインタフェースを考慮し、対話形式での入力を意識した設計となっている。

以上のような特徴を持つが、実際に合成部では、図3に示すような手順に従って処理を進めていく。

3.2.1 組合せ回路の式入力と真理値の生成処理

組合せ回路の合成を行う場合は、組合せ回路の論理式の入力を行う（行わない場合は次の処理に進む）。式の入力の例は次式(1)の通りである。

$$Y_1 = A ! B + AB ! C ; A ! B ! C \quad (1)$$

ここで Y_1 は出力関数、 A, B, C は入力変数、 $!$ は前の入力変数の否定 (NOT) を表し、 $+$ は論理和 (OR)、 $*$ は論理積 (AND)、 $;$ 以降には、冗長項を積和形で表す。なお、 $*$ は省略可能である。論理式に使用できる記号を表1に示す。なお、多数の式の区切りは $/$ で示す。

表 1 . 論理式で利用できる記号

等号	=
否定 (NOT)	!
論理積 (AND)	*
論理和 (OR)	+
排他的論理和 (E×OR)	@
冗長項	;
カッコ	(,)

真理値の生成は入力式の値を計算し、その値が“1”となった真理値表上の番号^{*1}を式の表現に必要な真理値番号として採用する。

式(1)の場合、入力変数 C, B, A それぞれに各真理値 (0, 1) を代入して、 Y_1 の値を求めるとそれが“1”となる真理値番号は、{2, 5, 6, (4)} である。ただし、カッコ内は冗長な真理値番号である。この表現が Y_1 の論理式を表す。

3.2.2 順序回路の真理値生成処理

まず、順序回路の出力関数、状態関数、およびフリップフロップの入力関数を表現するために必要な真理値番号と状態線図や表に記述されていない冗長な真理値番号を10進符号化し、つまり遷移表を符号変換する。

筆者らの開発した論理合成プログラム（状態線図入力変換プログラム）では、入出力値は、

$$(X_2 X_1 : Y_2 Y_1)$$

の16進2桁で入力できるように、今回の手書き状態線図よりも一般性をもたせている。冗長値には16進のFFを用いる。したがって、10進で示せば255となる。これらを10進符号化して、遷移表に変換する⁸⁾。

1例として、図4の状態線図を基にまず遷移表を作成すると表2が得られる。Sは現在の状態、S*は次状態、Xは入力値、Yは出力値である。

次に10進符号化した遷移表は、表3となる。ここでtは真理値番号、Qは現在の状態の状態符号、Q*は次状態の状態符号、Xは入力符号、Yは出力符号である。

状態符号は状態名の数値部をそのまま10進符号として用いており、単純な状態割当に相当する。

図 4 . 状態線図

表 2 . 図 4 の遷移表

S	S*	X	Y
S0	S0	00	00
S0	S1	01	00
S0	S2	02	00
S1	S1	00	00
S1	S2	01	00
S1	S0	02	02
S2	S2	00	00
S2	S0	01	02
S2	S0	02	03

表 3 . 10進符号化状態遷移表

i	t	Q	Q*	X	Y
1	0	0	0	0	0
2	1	0	1	1	0
3	2	0	2	2	0
4	4	1	1	0	0
5	5	1	2	1	0
6	6	1	0	2	2
7	8	2	2	0	0
8	9	2	0	1	2
9	10	2	0	2	3

i 行の真理値番号 t_i は10進符号化された X_i と Q_i を用いて生成される。

$$t_i = Q_i \times 2^k + X_i \quad (2)$$

k は入力値を2進化したときのビットの数である。例えば、 $i=9$ の時の真理値番号は、 $Q_i=2$ 、 $k=2$ 、 $X_i=2$ であるから、 $t_i=10$ となる。

これにより、生成される真理値の集合 T は、表3に示すように、

$$T = \{0, 1, 2, 4, 5, 6, 8, 9, 10\}$$

となる。未定義の真理値番号の集合 R は、

脚注 ※1：真理値表においてアルファベットの逆順に並べた入力の2進値を10進値化して表したものの。すなわち上から順に0, 1, 2, ……と10進番号化したもの。

$R = \{0, 1, 2, \dots, 2^{k+r}-1\} - T$ (3)
 で与えられ、 r は状態符号値を2進化したときのビットの数であるから、 $r = 2$ より、

$2^{k+r}-1 = 16-1 = 15$
 よって、
 $R = \{3, 7, 11, 12, 13, 14, 15\}$
 となる。

出力関数の真理値生成は次のように行う。出力符号値の個数(すなわち最大の出力符号の値)を y としたとき、 $m \geq \log_2 y$ を満たす最小の整数 m 個の出力変数を用意する。これらの変数名を順に $Y_1, Y_2, \dots, Y_m$ とし、各出力変数 Y_j に対する真理値番号の生成は、各真理値番号 t_i に対する出力符号値 Y_j に関して、次式を計算した(ビットごとの論理積をとった)とき、 P_j が真となる真理値番号を取ることで行う。

$$P_j = (Y_j \text{ AND } 2^{**} (j-1)) \tag{4}$$

ただし、 $j = 1, 2, \dots, m$ 。

言い換えれば、出力符号値を2進符号化して $Y_m, \dots, Y_2, Y_1$ で表し、各 Y_j について該当するビットが1である真理値番号 t_i を抽出することである。

状態関数の真理値生成も、同様に次のように行う。いま状態符号値のビット数を r とする。入力変数名は A からアルファベット順に $A, B, C, D, \dots, K$ (k 番目のアルファベット) と割り振るので、入力変数の個数が k ならば、状態符号の変数名は $(k+1)$ 番目のアルファベット (L) から順に $Q_{k+1}, Q_{k+2}, \dots, Q_{k+r}$ が割り振られる。

各状態変数 Q_{k+j} の真理値は、式(4)において Y_j の代わりに Q_{k+j}^* を、 m の代わりに r を代入して P_j が

表4．出力関数，状態関数の真理値生成
 (2進符号化状態遷移表)

i	t	現状態 Q _b Q _c	次状態 Q _b * Q _c *	入力 B A	出力 Y ₂ Y ₁
1	0	0 0	0 0	0 0	0 0
2	1	0 0	0 1	0 1	0 0
3	2	0 0	1 0	1 0	0 0
4	4	0 1	0 1	0 0	0 0
5	5	0 1	1 0	0 1	0 0
6	6	0 1	0 0	1 0	1 0
7	8	1 0	1 0	0 0	0 0
8	9	1 0	0 0	0 1	1 0
9	10	1 0	0 0	1 0	1 1

真になる真理値番号 t_i を取ることによって行う。すなわち、各状態変数について状態符号値が1であるところの真理値番号を抽出する。

図4の例では、出力符号変数名は、 $m = 2$ より Y_1, Y_2 である。 $j = 1, Y_1$ のとき、 $(Y_1 \text{ AND } 2^0)$ であり、表3から表4が得られ、

$$Y_1 = \{10, (R)\},$$

$$Y_2 = \{6, 9, 10, (R)\}$$

となる。また状態変数は、 $r = 2$ であり、

$$Q_c = \{1, 4, (R)\},$$

$$Q_b = \{2, 5, 8, (R)\}$$

となる。

3.2.3 フリップフロップの入力設定
 フリップフロップの入力設定は、JK型、D型の各々の入力動作に従って、10進符号化した、または2進符号化した遷移表と照合しながら行われる。例として、JK-FFの場合について述べる。

JK-FFの入力動作表を表5に示す。状態遷移表における現在の状態符号値 Q と次の状態 Q^* の符号値の変化をJK-FFの状態変化に対応させて、その状態変化を引き起こすためのJK-FFの入力を決定する。

図4の例で言えば、表4における Q_c と Q_c^* および Q_b と Q_b^* の関係を表5の q_u と q_u^* との関係に対応させて、フリップフロップ u (図4の場合、 $u = C, D$) の入力 J_u, K_u を決定する。そのための入力の真理値番号は、

$$J_c = \{1, (4, 5, 6, R)\}$$

$$K_c = \{5, 6, (0, 1, 2, 8, 9, 10, R)\}$$

$$J_b = \{2, 5, (8, 9, 10, R)\}$$

$$K_b = \{9, 10, (0, 1, 2, 4, 5, 6, R)\}$$

となる。

以上のことから、図4の状態線図の場合には、回路の入力変数名に A, B 、出力変数名に Y_1, Y_2 、フリッ

表5．JKフリップフロップuの動作表

現状態 q _u	次状態 q _u *	入力	
		J _u	K _u
0	0	0	255
0	1	1	255
1	0	255	1
1	1	255	0

255 : redundant

プフロップC,Dの状態変数名には Q_c, Q_d ,それへの入力変数名には J_c, K_c, J_d, K_d が用いられる。そして,上で得られたこれらの真理値番号表現(論理式表現に対応)によって,図4の動作を表すことができる。

3.2.4 簡単化と論理式の表現処理

本システムでは,前節で生成された各々の関数の真理値を用いて,木探索法のアルゴリズム^{10)~12)}による簡単化を行っている。

完全系の選択でNAND形を選択すると,積項形の主項を生成して最小被覆より最簡な積和形の論理関数が求まる。一方,NOR形の選択を行うと,和項の素項を生成して,最簡な和積形の論理関数を求める。つぎに,ド・モルガンの公理をアルゴリズム化して,最小化された積和形からはNAND形に,和積形からはNOR形に変換される。

出力結果となる論理式に変換されるとき,順序回路の場合は入力変数名として,組合せ回路に用いられる変数名以外の変数(一文字)がアルファベット順にとられる。フリップフロップの状態変数名には前述の Q_{k+j} が用いられる。また,組合せ回路の場合は入力した式から簡単化された表現が得られる。

3.3 プリント配線板ネットリスト作成部¹³⁾

本システムでは,論理合成部で得られた論理式を展開することにより,素子の接続テーブルを作成し,OrCAD-PCB IIで利用できるデータ形式のネットリストを生成する。処理の流れを図5に示す。

3.3.1 接続情報テーブルの作成

論理合成部では,論理式形式の出力データが得られた。そこでプリント配線板や回路図の作成を行うためには,式を展開することによって,ゲートレベルでの接続関係の抽出や使用する素子の選択を行わなくてはならない。したがって本システムでは,論

理式の出力関数とフリップフロップの入力関数より,接続情報テーブルの作成を行った。このとき最初に素子選択としてはあらかじめ登録されている素子が選ばれ,表6に示すような接続情報テーブルが作成される(表6は式(1)の例である)。

この接続情報テーブルにおいて,接続番号“0”は出力端子,もしくはフリップフロップの入力端子を意味した特別な番号である。したがって,接続端子から次の“0”の手前までの情報が1つの関数を表している。

このテーブルは配列に格納されており,接続番号は素子番号で示される素子が接続される先の,素子の接続番号が格納されている配列の添字を表している。

出力関数 Y_1 の場合を例としてあげると表6に表されるように,6番目の配列に入っている素子番号200の素子(2入力NAND素子)は4番目の配列の接続番号が1であることから,1番目の配列に入っている接続番号の素子,すなわち,3番目の配列の素子番号70の素子(出力端子 Y_1)に接続されることを意味している。

以下同様にして,接続番号,入力数,素子番号,およびそれらが格納されている配列番号によって,使用する素子や素子間の接続関係が明らかとなる。図6は表6の例を基に回路の合成を行ったものである。

3.3.2 回路構造に対する処理

作成された接続情報テーブルには,そのままでは,

表6 接続情報テーブル

接続番号	入力数	素子番号
0	1	70
1	2	200
4	2	200
4	3	201
7	1	12
7	1	14
10	1	10
10	1	13
10	1	15

配列の番号

1	2	3	4	5	6	7	
0	1	70	1	2	200	4

図5 ネットリスト作成処理の流れ

図6. 表6を基に合成した回路

重複して使用できる共通ゲートが存在するので、回路を構成する上で共通ゲートは削除してやる必要がある。そこで本システムでは、共通ゲートを削除する2, 3の処理を行って対処している。

3.3.3 ネットリスト変換処理

接続情報テーブルの作成により、接続関係および使用する素子の情報が明確になった。そこで次に、OrCAD-PCB II上で利用するためには、接続情報テーブルをOrCAD-PCB IIで読みこめるネットリスト形式のデータに変換を行う必要がある。OrCAD-SDT (回路図作成ツール) では、主なプリント配線板自動作成のネットリスト形式に対応することができる。その中で本システムではOrCAD-PCB IIのサポートする専用のネットリスト形式に変換する。

このネットリストに必要な内容は、基本的に次の2点である。

- 1) 各部品の外形モジュール (部品) データ
- 2) ピン間の接続データ

外形モジュールデータは、プリント配線板上で素子を配置していくために必要なデータである。通常外形モジュールの選択はOrCADがあらかじめ用意しているスタッフファイルより、自動的にそれぞれの素子にあった外形モジュールを選択し、ネットリストを作成する。本システムでも同様に、あらかじめ登録している標準素子に対して、それぞれ外形モジュールを登録しておき選択する。ピン間の接続データは、それぞれの素子の接続関係を表すものである。したがって、接続情報テーブルより接続関係を抽出し、変換することにより作成することができる。

以上の処理によって、図4に示した手書き状態線図を入力することによって作成されるのが図7に示すネットリストである。その後OrCAD-PCB IIで

プリント配線板の設計を行うわけであるが、受け渡すネットリストによって図8に示すようなモジュールが表示されるので、これを最適配置、配線を行うことにより、図9に示すようなプリントパターンを得ることができる。

4. まとめ

本論文は、デジタルシステムの設計の1手法として提案するものである。まえがきでも述べたように、現在はハードウェア記述言語による設計が主流

```

({OrCAD PCB NetList Format
NET_LIST
|
T.Yoshida
O.I.T

Revised: December 13, 1993
Revision:1

Time Stamp-}
(DC 5 E641 14DIP300 U1 7400
(1 a!)
(2 b)
(3 300)
(4 b!)
(5 c)
(6 301)
(8 302)
(9 d)
(10 e!)
(11 303)
(12 d)
(13 f)
)
(DC 5 E642 14DIP300 U2 7410
(1 d!)
(2 e)
(12 304)
(13 f!)
)
.
.
.

```

図7. ネットリストファイル

図 8. モジュール (外形形状)

(a) レイヤー 1

(b) レイヤー 2

図 9. プリントパターン

になりつつあるが、これはユーザ（ハードウェアまたはシステム設計技術者）に記述言語や論理回路等についてかなり高度な知識を要求し、決して使いやすいシステムといえないのが現状である。このため、マンマシンインタフェースを考慮した、より使いやすいシステムの開発が望まれている。そこで筆者らは入力部において手書きの状態線図を認識させ、入力データとして利用することにより、より使いやすいシステムの開発を行った。

本論文では、そのシステムの概要、各処理部について説明した。本システムを利用することにより、おおよそ数百ゲート程度の回路規模のプリント配線板を実現できる。現状ではこのように比較的小規模の回路しか実現できないが、これをさらに改善し、記述言語など他のツールと互いに相補う形で利用すれば、設計の容易化、自動化に役立つものといえる。なお、その際に使用する変数名を自由に選べるようにするには、自動割り振りのものと希望するものとの入出力変数名対応テーブルをストアするか、または対話的に入力して置き換え処理を行えばよい。

今後の課題としては、他のツールとのデータの互換性を持たせることと、状態線図上で利用できる文字種を増すように文字認識部を改善することである。

謝 辞

適切なご助言をいただいた査読委員各位に謝意を表します。

(1994.5.13-受理 1994.9.29-再受理)

文 献

- 1) R. リブセット他著 杉山尚志監訳：“VHDL言語記述によるハードウェア設計へのアプローチ”，マグロウヒル，(1990.6)
- 2) 小島郁太郎，中山俊一：“ハードウェア設計者に馴染みのある図形表現で論理合成を活用”，日経エレクトロニクス，No614, pp.66-78, (1994.8)
- 3) 吉田武史，加藤貴之，長谷川勝士，山田豊士，久津輪敏郎：“プリント基板作成のための手書き状態線図の認識”，プリント回路学会第7回学術講演大会論文集，pp.77-78, (1993.3)
- 4) 久津輪敏郎，加藤貴之，他：“プリント配線板設計

- 自動化のための手書き状態遷移図の認識”, サーキットテクノロジー, Vol.9, No.1, pp.23-30, プリント回路学会, (1994)
- 5) 長谷川純一, 興水大和, 中山晶, 横井茂樹: “画像処理の基本技法〈技法入門編〉”, 技術評論社, (1986.8)
 - 6) ニューロンネットグループ, 桐谷滋: “入門と実習 ニューロコンピュータ”, 技術評論社, (1989.10)
 - 7) 長谷川勝士, 久津輪敏郎: “過学習度の判定式と過学習防止を考慮したバックプロパゲーション法”, 平成4年電気関係学会関西支部連合大会G303, (1992.11)
 - 8) K. Ebata, T. Kutsuwa and N. Kadomaru: “Automatic Synthesis System of Sequential Circuits”, IEEE, ISCAS Proceedings Vol.1, June, (1988)
 - 9) 角丸典子, 江端克彦, 久津輪敏郎: “CADを応用した論理回路の自動合成”, 電子情報通信学会春季全国大会論文集, pp.1-239, (1989)
 - 10) J. R. Slagle, C. L. Chang and R. C. T. Lee: “A New Algorithm for Generating Prime Implicants”, IEEE, Trans. on Computers, Vol.C-19, No.4, pp.304-310, April (1970)
 - 11) Douglas C. Schmidt and Larry E. Druffel: “An Extention of the Clause Table Approach to Multi-Output Combinational Switching Networks”, IEEE, Trans. on Computers, Vol.C-23, No.4 April (1974)
 - 12) 上林弥彦, 岡田康治, 矢島脩三: “節展開法を用いた論理関数の主項の生成”, 電子情報通信学会論文誌, Vol.J62-D, No.2, pp.89-96, (1979)
 - 13) 清水隆雄 著: “続 OrCAD入門(応用編)”, 電波実験社, (1991.7)