設計自動化28−1
(1985・9・17)

論理合成環境

Logic Synthesis Environment

高木茂
Shigeru Takagi
日本電信電話株式会社 武蔵野電気通信研究所
Musashino Electrical Communication Laboratories, N.T.T.

1．まえがき

論理回路自動合成は長年の研究課題であり，論理関数の最小化[1][2][3]，回路変換[4][5]，動作仕様記述に基づく装置全体の合成[6]〜[9]，等様々なレベルで検討されてきたが，研究の歴史の長さ，報告書の数からして，中心テーマは，論理関数の最小化であった

しかし，論理設計技術書[10]〜[13]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．残りの大部分には，演算器の構成法，装置全体の構成法等，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，装置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，加算器の構成法の種類，各構成法のゲート量，速度等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，これら加算器の構成法を記憶しておき，ビット幅，性能等，具体的な加算器の仕様が与えられた時，適切な構成法を選択し，トップダウンに詳細な回路を生成している．

また，近年のデバイス技術の進歩によりゲートコストが低下する一方，VLSIの修正困難さのため設計の無謬性が重要となってきている．このため，論理合成の研究においても，適用領域を広くすることに重点がおかれるべきと思われる．

本論文は，『既存の論理自動合成アルゴリズムの基盤のうえに，設計者にゆだねられていたマクロな設計技術を逐一アルゴリズム化・蓄積すれば，設計者はこれら豊富な設計ツールの中から自分の目的に沿った手法を選択し，問題に適用することにより，多くの設計作業を計算機にまかせられる』という発想に基づき作成している総合的論理合成支援の実験システム（論理合成環境）の概要，技法について報告する．

マクロな設計技法をアルゴリズム化するには以下の技術が重要となってくる．

①仕様記述の抽象度の高度化

従来，機能仕様は真理値表，論理式等で与えていた．これらは，論理的に厳密ではあるが，機能のマクロな意味をとらえるには向いていない．例えば，加算機能を真理値表で表すことはできるが，真理値表より，それのマクロな機能（加算機能）を推定することは困難である．　回路のマクロな機能を陽に記述できるようにする必要がある．本論文では，マクロな機能の種類とそれを実現する回路のバリエーションの範囲を分類・整理し，機能種別毎に回路構成の特徴を表すパラメータとその値の組で仕様を記述する方法を採る．

②マクロな機能記述に基づく詳細論理の生成

従来は，与えられたものを簡約化するということが中心課題であったのに対し，抽象度の高い機能記述より，適切な論理式を生成することが課題になる．本論文では，機能種別毎に多段の論理式を組み立てる記号処理プログラムをつくる方法を採る．

③多数の技法の管理，利用法

マクロな設計技法は機能種別毎に異なり，また同一機能種別といえども複数存在する．これら多数の設計技法を管理し，設計者にとって全体が把握しやすく，かつ，使い易いインタフェースを提供することが重要となる．本論文では，オブジェクト指向プログラミングの手法に沿って，各設計技法を整理する．また，仕様はインタラクティブに入力できるよう考慮する．

以下，第2章ではシステムの概要と多数の技法の管理法，ユーザインタフェースについて述べ，第3章では仕様記述の種類について述べ，第4章では主要な設計技法について述べ，第5章では，マクロな機能仕様から詳細な論理を生成する技法の例について述べる．

なお現在は，マクロな設計技法のうち，主として組み合わせ論理回路に関する技法を蓄積している段階である．

2．システムの概要

本システムはエキスパートシステム構築ツールである汎用知識表現推論環境KRINE [14] (Knowledge Representation and inference Environment) (図1)のうえに作られている．

KRINE はフレーム(オブジェクト) を基盤とし，LISP ,PROLOG,RULE,DAEMON,等の手続き記述方式およびグラフィックパッケージを統合したシステムである．

論理設計技法は多数ある．これら技法の管理，追加の容易さ，システムの使い易さを考慮し，本システムを構築する指針を以下のように定めた．

①設計技法の記述

設計技法を目的（実現すべき機能種類）に沿って階層的に分類，整理する(図2)．機能仕様の記述形式は各目的(機能) に最も適切な形式を設定する．即ち，論理設計技術書の各機能ブロックの記述を参考にし，充分高い抽象度を保ちつつ，機能の意味をあいまい性なく記述できるよう設定する．各設計技法は機能仕様記述を入力とし，多段のブール式を出力する．設計技法はLISP，あるいはPROLOGのプログラムとする．

②ユーザーインタフェース

設計技法は多数ある．また，機能仕様の記述法は，設計技法の種類毎に異なる．これらを記憶することは負担となる．現在蓄積されている設計技法をCRT 上に表示し，設計者に使いたい技法をポインタ・デバイス(マウス等) で指示させる．システムは，その設計技法が必要とするデータをメニュー等のインタラクティブな方法で問い合わせ，それに対する設計者の応答を得て，合成プログラムを起動する．

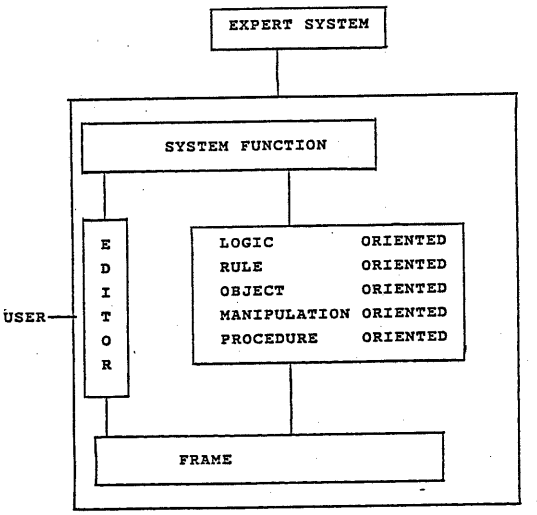


図１．知識表現推論環境KRINE の概要

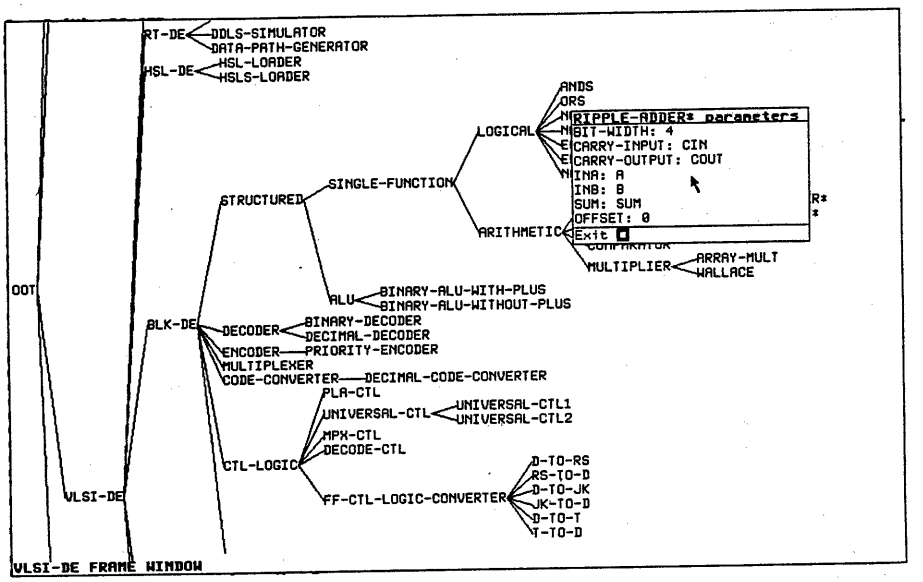


図２．設計技法の階層を表す画面と機能仕様の入力例

③設計技法の管理

①及び②の項目を実現する枠組みとしてオブジェクト指向のプログラミング手法を採用する(図3). すなわち，各設計技法を各々オブジェクトとする．各オブジェクトには，設計技法を実現する合成プログラムと共に，ユーザーに対する問い合わせ処理，デフォルト値等，その設計技法に特有な処理を行うプログラム，データを持たせる．設計技法間の階層関係は，オブジェクト間の階層関係に対応付ける．下位オブジェクトで共通する処理があれば，それら共通する処理を実行するプログラムは上位オブジェクトのみに持たせ，オブジェクトのメソッド遺伝機能により，下位オブジェクトでもその処理が実行できるようにする．設計技法の表示は，KRINE のオブジェクトの階層関係をCRT 上でツリー状に表示する機能を利用して実現する．

3．機能仕様の記述法

設計技法の種類毎にそれに適した機能仕様の記述法を設定することにより，少ない記述量で回路の意図を明確に表現できるようにする． 本論文では次の4種類を採用している．このうち最初の3 種類は従来より，広く使われている方法である．．

①ブール式

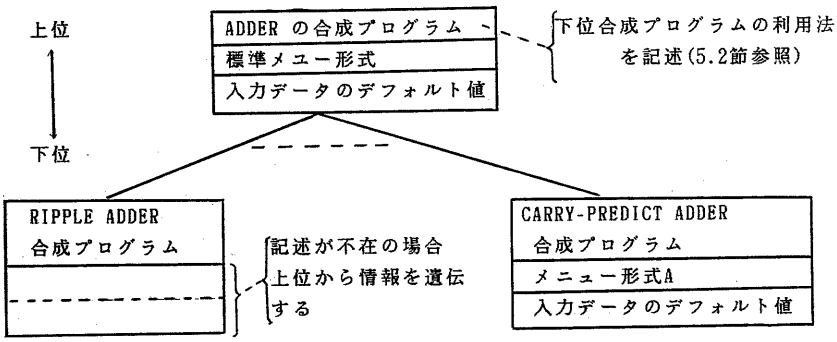
ブール式は，表1 に示す論理演算子を使った，任意の深さのネスティングを許す式である．なお，実際の処理システムでは，LISPのS 式のシンタックスを使い，プレフィックス記法で記述している．

例) X=A&(B @ C) or ^(D or ^(B or C))

表1 論理演算子の種類

記号	意味
OR	OR
&	AND
^	NOT
^&	NAND
^OR	NOR
@	EXCLUSIVE OR
^@	EXCLUSIVE NOR
=	EQUAL
^=	NOT EQUAL

設計技法とオブジェクトの対応付け



設計技法の起動法

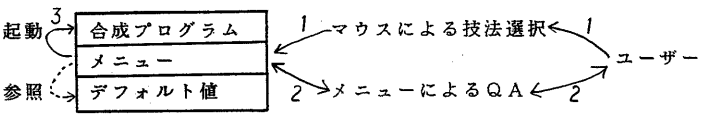


図3．設計技法をオブジェクトとして表現，管理する手法の概要

②真理値表
③キューブ
④特殊形式

これは，ALU ようにブール式で機能を表現するとその量が爆発的に増大し，記述が困難，あるいは，加算器，乗算器，のように概念が確立されており，その機能名称と若干の補足的キーワードを与えれば，ブール式でなくとも論理的にあいまい性なく言い尽くせる機能ブロックを記述するための形式である．機能ブロック毎に，設計者にとって意味がわかり易く，かつ回路設計をするうえでポイントとなる個所をおさえられるよう形式を設定する必要がある．

いくつかの例(ALU，加算器，マルチプレクサ）を図4 に示す(意味を日本語で記述し直してある）．ALU の様な多機能演算器の仕様は，データ幅，機能種別とそれらの選択信号の表であたえる．また，加算器のような単機能演算器の仕様は，データ幅の指定，CARRY 入出力の有無等の若干のパラメータで与える．

ALU の機能記述例
データのビット幅:32
関数機能:

制御コード	関数
S2&S1&S0	F=A
S2&S1&S0	F=A or B
S2&S1&S0	F=1
S2&S1&S0	F=A@B
S2&S1&S0	F=A+B+Cin

加算器の仕様記述例
データのビット幅:16
CARRY 入力　　: 有り
CARRY 出力　　: 不要
論理段数:
ゲート数:

マルチプレクサの仕様記述例
データのビット幅:8
マルチプレクス数:4
セレクト信号　　: 有り
ストローブ信号　: 不要

図4．抽象度の高い機能仕様の記述例

4．主要な設計技法

現在，蓄積している主な設計技法を以下に示す．

4.1　基本的技法

(1) 機能仕様記述形式間の変換
①任意のブール式の積和標準形への変換

任意のネスティングしたブール式を積和標準形に直す．展開は基本的な手法を用いている．即ち，
(i) 論理和，論理積以外の演算子はその定義に従って論理和，論理積，否定の組合せに書き替える，
(ii)否定はド・モルガンの法則を用いて展開する，
(iii) ネスティングした式は分配律
(A&(B or C)= A&B or A&C) を用いて展開する，
(iv) 0になる項 (P&＾P=0)や重複したリテラル
(P&P=P) の除去．吸収律(P or P&Q=P)を用いた簡約化，を行う．

この手法で得られる展開結果は，最簡なブール式ではない．
②積和標準形と真理値表間の相互変換
③積和標準形とキューブ間の相互変換

(2) 論理式の簡約化
2 レベルの積和標準形の論理式を簡約化する技法である．2 種類の簡約化技法MINI[1] ，とPRESTO[2] を持っている．

例) A&B&C&＾D or　A&＾B&C&＾D or
A&B&＾C&＾D　or A&＾B&＾C&＾D
= A&＾D

(3) 論理式の因子化
2 レベルの積和標準形のブール式を共通因子をくくりだすことにより多段化する技法である[15]．ゲート数を減らす技法の一種である．

例) A&B&C&D or A&B&C&E or A&B&F or A&B&G or H
= A&B&(G or F or C&(D or E)) or H

(4) 回路変換操作
多段の論理式を基本素子の接続関係に変換する技法である．即ち，
①多段の論理式を出力側からレベルソートする，
②1つの論理式を基本素子網に変換するのは極性伝播法[5] による．極性伝播法を適用する際には，論理式の出力負荷を駆動できる基本素子の中から選ぶ．
③論理式の出力負荷は，その論理式の一段前のレベルの全論理式を基本素子網に変換後，この論理式出力の接続される基本素子の入力負荷を積算して求める．

得られた基本素子網は基本素子の品種，ファンイン数，ファンアウト数制限をみたしている．1 万ゲート規模の回路の変換時間は1MIPS のパーソナルマシンで1000秒程度である．

4.2 高位機能ブロックの設計技法

高位機能ブロックをその高い抽象度の仕様から多段の論理式に変換する技法群である．主な機能ブロックを以下に上げる．これらを生成する基本的手法は第5章を参照されたい．

- ・論理演算器
- ・加算器
- ・ALU
- ・乗算器
- ・比較器
- ・マルチプレクサ
- ・デコーダ
- ・エンコーダ類(バイナリイ・エンコーダ，プライオリティ・エンコーダ，コード・コンバータ)

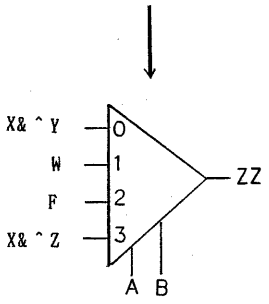
4.3 制御回路合成用の設計技法

①規則的制御回路の設計技法

規則的な回路部品を使って制御回路を構成する方法が各種考案されている．この設計技法は，与えられた論理式をそれら規則的制御回路構成に変換する手続きである．規則的制御回路構成法としては，PLA,万能論理生成回路，マルチプレクサ等を使った方法がある．

例) マルチプレクサを使った構成法

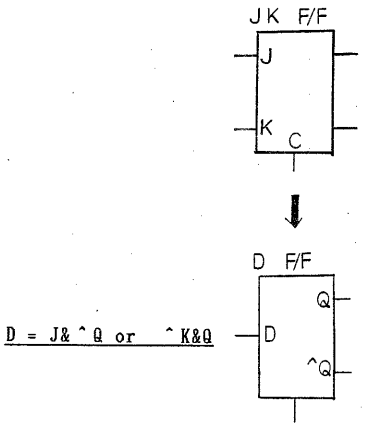
```
ZZ = X& ^ Y& ^ A& ^ B or  W&A& ^ B  or F& ^ A&B
     or X&  ^ Z&A&B
```



②フリップフロップの入力信号論理の相互変換

記憶素子として，D タイプ,RS タイプ等各種フリップフロップが存在する．これらフリップフロップの入力信号論理の相互変換をおこなう．

例) JKタイプF/F からD タイプF/F への入力論理の変換



4.4 状態遷移マシン設計技法

DDL 動作仕様記述に基づき，データパス機能ブロック，制御ブロックを合成する技法[9][17] である．

4.5 その他ユーティリティ

①定数削除

論理式に含まれる定数(0 1) を削除して，論理式を簡単にする．

②変数置換

変数を他の変数あるいは，定数に置換する．

③クリティカルパス解析

④ゲート数カウント

⑤レベルソート

5．論理式の生成手法

抽象度の高い機能仕様から詳細な論理式を生成する基本手法および，同一タイプの機能ブロックに複数の設計技法があった場合の制御方法を示す．

5.1 論理式生成の基本手法

抽象度の高い機能仕様から詳細な論理式を生成する手法は設計技法毎に異なる．それらのなかで使われているいくつかの基本的手法について示す．

(1) ビット方向への繰り返し

機能ブロックのデータ幅は，複数(n) ビットである場合が多い．論理演算のように，ビット幅方向で論理の形の変わらない機能ブロックでは，論理変数に添

字を付け，第ｉビット目の論理を構成する．次に添字ｉを１からｎまで変化させながら，式を繰り返しコピーすることにより全体の論理式が得られる．またアレイタイプの乗算器のように，２次元的に繰り返えされる回路構造の場合にも２個の添字を付加し，式の繰り返しコピー法により全論理式が得られる．

(2) リカージョンを利用した式の生成法

リカーシブに論理式を定義（生成）できる場合がある．

例えば，キャリー予測方式においては，第ｎビット目の部分和をＰ（ｎ），キャリー生成をＧ（ｎ）とすると，第ｎビット目のキャリー予測論理Ｃarry（ｎ）は次のリカージョンで定義できる．

Ｃarry(0)=Ｃin

Ｃarrry(n)= Ｇ(n-1) or Ｃarry(n-1)&Ｐ(n-1)

このリカージョンを用いるとネスティングしたブール式が得られる．それを積和標準形に変換することにより，よく知られたキャリー予測の論理式が得られる．

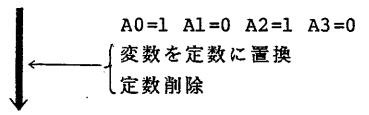
(3) 定数の代入

一般に機能ブロックは変数間に演算を施して結果を出力するように考案されている．しかし，一方が定数である機能仕様が与えられることがある． この場合は，入力が共に変数と想定した設計技法を流用して論理式を生成後，一方の変数を定数で置換し，定数削除の技法を適用することにより，低コストの論理式が得られる（図５）．

２変数加算器 (A+B)の論理式

（リップル方式，４ビット幅）

```
CA0  = A0&B0 or CIN&A0 or CIN&B0
SUM0 = CIN@(A0@B0)
CA1  = A1&B1 or CA0&A1 or CA0&B1
SUM1 = CA0@(A1@B1)
CA2  = A2&B2 or CA1&A2 or CA1&B2
SUM2 = CA1@(A2@B2)
COUT = A3&B3 or CA2&A3 or CA2&B3
SUM3 = CA2@(A3@B3)
```



変数と定数の加算器(5+B) の論理式

```
CA0  = CIN or B0
SUM0 = ^B0&^CIN or B0&CIN
CA1  = B1&CA0
SUM1 = B1&^CA0 or ^B1&CA0
CA2  = CA1 or B2
SUM2 = ^B2&^CA1 or B2&CA1
COUT = B3&CA2
SUM3 = B3&^CA2 or ^B3&CA2
```

図５．定数削除の技法の応用

(4) コードパターンの利用

コード変換などでは，コードパターンをそのまま利用できることが多い．すなわち，ある情報を表すコード体系が複数通りあったとする．それぞれのコードパターンの表を用意しておく．任意の2 つのコードA,B 間において，A のコード からB のコードに変換する回路は，A を真理値表の入力側パターン，B をそれに対する出力側パターンとみなし，この真理値表を簡約化し，ブール式に変換することにより得られる．図６は，BCD からEXCESS-3コードへの変換例をしめす．

真理値表

INPUTS	OUTPUTS
BCD	EXCESS-3
(0 0 0 0)	(0 0 1 1)
(0 0 0 1)	(0 1 0 0)
(0 0 1 0)	(0 1 0 1)
(0 0 1 1)	(0 1 1 0)
(0 1 0 0)	(0 1 1 1)
(0 1 0 1)	(1 0 0 0)
(0 1 1 0)	(1 0 0 1)
(0 1 1 1)	(1 0 1 0)
(1 0 0 0)	(1 0 1 1)
(1 0 0 1)	(1 1 0 0)
(1 0 1 0)	(1 1 0 1)
(1 0 1 1)	(1 1 1 0)
(1 1 0 0)	(1 1 1 1)
(1 1 0 1)	(0 0 0 0)
(1 1 1 0)	(0 0 0 1)
(1 1 1 1)	(0 0 1 0)

変換元コード ➡ 変換先コード

図６コードパターンの利用

(5) パターンマッチと合成規則

ALU のように，機能仕様がかなり複雑な場合には，機能仕様の構文を解析し，構文の型（機能の型）ごとに，論理の合成規則を設ける手法が有効である．詳細は文献16を参照されたい．

5.2 複数設計技法の選択法

同一種類の機能ブロックといえども，種々な回路構成法が存在する．たとえば，加算器には，リップル加算器，キャリー予測加算器，キャリーセーブ加算器等がある．

これら回路構成方式を陽に指定して設計したい場合もある一方，ゲート量あるいは速度で回路仕様を指定したい場合もある．後者のほうが，ユーザーインタフェースとしては抽象度が高い．このような機能を実現するため，以下の方法を採用している．

①設計技法の階層のなかで，複数の設計技法（リップル加算，キャリー加算，キャリーセーブ加算）を，それぞれ．機能ブロックの設計技法（加算器の構成法）の一種類と位置づける．
②機能ブロックの設計技法は，その下位の階層にある設計技法を順次起動し，合成結果が速度，ゲート量にかんする制約条件を満たしているかを調べる．条件の満足された時点で処理を終了する．
③どの設計技法を適用しても仕様を満たせない場合には，その旨のメセージを出力する．

単純ではあるが，このような手法によれば，新しい設計技法を登録するだけで，システムが自動的にその技法を使って問題を解決することができる．

6．まとめ

論理設計者が使用している設計技法は極めて豊富である．従来の論理合成の研究で扱っている範囲は少ない．特に，機能ブロックの構成法，装置全体の構成法等のマクロな設計技法を扱った報告は少ない．　本論文では，これらマクロな設計技法をなるべく忠実にアルゴリズム化し，豊富な設計技法のアルゴリズムベースと使い易いインタフェースからなるシステム（論理合成環境）を構築することを提案し，実験システムにより基本的な手法を検討した．

研究は緒についたばかりであり，解決すべき課題は多い．以下に主な課題を記す．

①抽象度の高いレベルで機能仕様を記述しようとすると，機能ブロックの種類毎に記述形式を定めねばならない．　記述方式に対する，“単語の種類を少なくする”，“記述の抽象度を高く保つ”，“厳密さを保つ”，等の要求は互いに相反の関係にあり，適切な設定が課題である．
②１つの設計技法はあるゲート量，速度領域を想定して考案されている．　要求仕様の速度，ゲート量条件が単一技法のみでは満たされない場合に，複数技法を組み合わせる等の高度な問題解決方式が研究課題である．
③設計者と論理合成システム間の適切な役割分担，大量なデータの表示，管理法，ドキュメンテーション等，が実用システムを構築するうえで重要となる．

「参考文献」

(1) Hong,S.J.,Cain,R.G.,Ostapko,D.L. :"MINI:A Heuristic Approach for Logic Minimization", IBM J.RES.DEVELOP.,PP.443-458(1974)
(2) Brown,D.W. :"A STATE-MACHINE SYNTHESIZER", Proc.18thDA Conference,PP.301-305(1981)
(3) 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)
(4)Darringer,J.A. :"A New Look at Logic Synthesis",Proc.17th DA Conference,PP.543-549(1981)
(5)Shinsha,T.,Kubo,T.,Hikosaka,M.,Akiyama,K., Idhihara,k. :"POLARIS: PORARITY PROPAGATION ALGORITHM FOR COMBINATIONAL LOGIC SYNTHESIS", Proc.21th DA Conference,PP.322-328(1984)
(6)Hoshino,T.,Endo,M.,Karatsu,O. :"An Automatic Logic Synthesizer for Integrated VLSI Design System",IEEE Custom Integrated Circuit Conference(1984)
(7) 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)
(8)Duely,J.R.,Dietmeyer,D.L. :A DIGITAL SYSTEM LANGUAGE (DDL)", IEEE Trans. Comput. VOL C-17 NO.9 PP.850-861
(9)Takagi,S. :"RULE BASED SYNTHESIS, VERIFICATION AND COMPENSATION OF DATA PATHS",Proc. ICCD'84 PP.133-138(1984)
(10) GERRIT A. BLAAUW :"DIGITAL SYSTEM IMPLEMENTATION", PRENTICE-HALL,N.J.(1976)
(11) Frederic J.Mowle :"A Systematic Approach to Digital Logic Design", Addison-Wesley (1976)
(12)William I. Fletcher :"AN ENGINEERING APPROACH TO DIGITAL DESIGN", PRENTICE-HALL,N.J. (1980)
(13)Wienkel,D.,Prosser,F. :"The Art of Digital Design", PRENTICE-HALL,N.J. (1980)
(14)Ogawa,U.,Shima,K.,Sugawara,T.,Takagi,s. "Knowledge Representation and Inference Environment ;KRINE", Proc. FGCS'84 PP.643-651(1984)
(15)Brayton,R.K.,McMullen,C.T. :"The Decomposition and Factorization of Boolean Expressions", ISCAS'82 PP.49-54 (1982)
(16) Takagi,S. :"ALU LOGIC SYNTHESIS USING TEMPLATES", Proc. ISCAS'85 PP.443-446 (1985)
(17)高木茂　“制御回路自動合成の一手法”設計自動化研究会資料 27-1(1985)