設 計 自 動 化 35－1
（1986. 12. 16）

制御論理回路自動合成方式の検討

戸次 圭介　横田 孝義　浜田 亘曼

（株） 日立製作所 日立研究所

本論文では、マイクロプログラムのアドレス順序情報と分岐機能を定義する情報から、マイクロプログラムのアドレス制御を行う論理回路を自動合成するアルゴリズムについて検討する。本方式は宣言的データとして表現された制御構造モデルを用いて論理回路の自動合成を行うものであり、モデルの蓄積により容易に設計空間の拡張が可能となる。また論理型言語prologを用いて本アルゴリズムの実装を行ない、実際に回路の合成を試みた結果、本アルゴリズムが実用可能となることが確認できた。

" A METHOD FOR CONTROL LOGIC SYNTHESIS" (in Japnese)

by Keisuke BEKKI, Takayoshi YOKOTA, Nobuhiro HAMADA

(Hitachi Research Laboratoy, Hitachi Ltd., 4026 Kuji-cho,
Hitachi-shi, Ibaraki-ken, 319-12 Japan)

In this paper, we propose a method of synthesizing the control logic for sequencing microprogram from the specification of the microprogram addressing and branching. This method is based upon modifying declaratively described control structure models. So, the system can be easily extended by user defining other control structure models. We implemented this system using logic programming language Prolog and have shown its effectiveness.

制御論理回路自動合成方式の検討

戸次　圭介　横田　孝義　浜田　亘曼

（株）　日立製作所　日立研究所

１．はじめに

近年のＶＬＳＩの高集積化に伴い、それを活用して高機能のＬＳＩを出来るだけ短期間に設計したいという要求が高まってきている。実装、診断等の技術は古くから研究がなされ実用化されているものが多いが、ＶＬＳＩの機能設計や論理設計は、その作業のほとんどが人手で行なわれているのが現状である。

そこで、これらの機能設計や論理設計の負担を軽減するため論理設計支援システムについて研究を進めてきた[1)~5)]。本論文では、論理設計支援システムの一部を構成するマイクロプログラム方式による制御系回路の中でマイクロプログラムのアドレスを制御する部分の自動合成方式について述べる。

２．制御系回路自動合成

一般にマイクロプログラム方式による制御論理回路は、図１に示す回路ブロック構成により実現することができる。制御記憶部には、データパス上にレジスタ転送を実行するためのプログラムが格納されている。各部は、マイクロプログラムのパイプラインの深さにより若干異なるが、普通以下の２サイクルで動作する。

１）制御記憶から、次アドレス計算部により計算された番地に格納されているマイクロ命令を読出す。
２）１）で読出されたマイクロ命令の情報の一部をデコードし、データパス上のレジスタ転送やフラグセット操作を実行する。また同時にマイクロ命令の一部の情報と論理回路の状態（フラグレジスタの値）から、次に読出すべきマイクロ命令が格納されている制御記憶のアドレスを計算する。

上で述べたようなサイクルは、マイクロプログラム

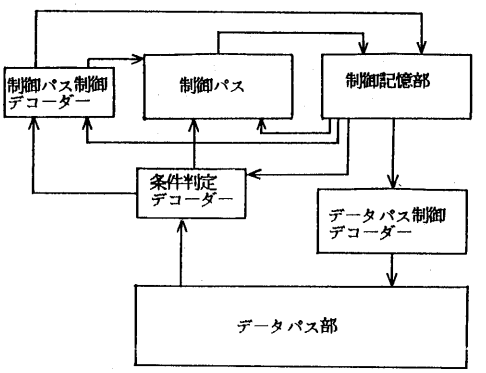


図１　制御回路のブロック構成

が有する分岐機能によって制御される。このマイクロプログラムの分岐の方式は次の５つのタイプに分類できる。[6),7)]

（ｉ）無条件分岐
（ii）条件テストの結果に応じて制御パス上のレジスタ転送を変え、条件分岐を実現する方式。
（iii）制御記憶のアドレスを指定するアドレスレジスタの一部に条件テストの結果を挿入することにより、条件分岐を実現する方式。
（iv）制御記憶のアドレスを指定するアドレスレジスタの一部に論理回路中の特定のレジスタの値をそのまま挿入することにより、多重分岐を実現する方式。（機能分岐）
（ｖ）マイクロサブルーチンの呼び出しと復帰

このような分岐操作は、図１に示す条件判定デコーダ

ーと制御パス制御デコーダーの２つの組合せ論理を用いて制御パスを制御することにより実現できる。これら２つの組合せ論理は、具体的にそれぞれ以下のような役割をもつ。

（ａ）条件判定デコーダー

(ii)及び(iii)の分岐機能は、次に実行すべきマイクロ命令が格納されている制御記憶のアドレスを、条件テストの結果に依存し決定する。具体的には、マイクロ命令によって指定されるテスト条件（回路の状態、主としてフラグレジスタの値）が成立しているかどうかを判定し、その判定結果に応じてアドレスを決定するというものである。

条件判定デコーダーは、(ii), (iii)の条件分岐を実現するために、マイクロ命令よって指定されるテスト条件が成立しているかどうかを比較判定し、その判定結果を制御記憶のアドレスを指定するアドレスレジスタや制御パス制御デコーダーに出力する組合せ論理である。

（ｂ）制御パス制御デコーダー

普通、制御パスは一般の論理回路のデータパス同様レジスタ、マルチプレクサ、スタック、カウンタ等、機能回路モジュールのネットワークとして構成される。マイクロプログラムの分岐機能は、制御パス上のレジスタ転送を制御することによって実現される。制御パス制御デコーダーは、マイクロ命令に格納されている分岐方式を指定する情報と条件判定デコーダーにおける条件判定の結果から、制御パス上のレジスタ転送を実行するのに必要な制御信号を発生する組合せ論理である。

以下では、マイクロプログラムのアドレス順序情報から制御パス、条件判定デコーダー、制御パス制御デコーダーを自動合成する方式について述べる。

２．１制御パスの合成

（１）制御回路モデルと分岐機能

制御パスの詳細な構成方式は、その用途、目的によって大きく異なるため、種々の制御回路モデルの情報を蓄積しておかなければ、広く設計空間を張ることができない。また制御回路モデルの情報を簡単に入力するには、制御回路モデルの情報を宣言的データとして記述できなければならない。さらに、１つの制御回路モデルによって張ることができる設計空間が広ければ広いほど、用意すべき制御回路モデルの量は少なくなるため、制御回路モデルを高機能で汎用性の高いものにする必要がある。

このため論理設計者によって入力された動作仕様に対し、システムが有する制御回路モデルは、不必要な回路を含んだ冗長なものになる場合がある。制御回路モデルの縮約アルゴリズムは、選択した制御モデルに含まれるこのような冗長回路を除去するアルゴリズムである。

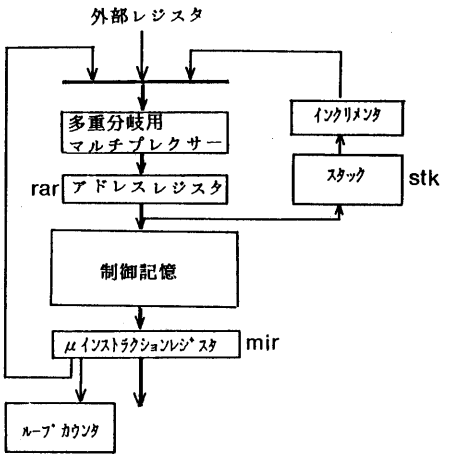


図２　制御回路モデルの例

図２に制御構造モデルの例を示す。なお図２において多重分岐用マルチプレクサは、機能分岐や条件分岐を実現する一連のマルチプレクサ群であり、この合成アルゴリズムについては、本節（３）で述べる。

```
rt_con(jmp, [[rar, <-, mif]]).
rt_con(jps, [[stk, <-, rar], [rar, <-, mif]]).
rt_con(rtn, [[rar, <-, stk, +, 1], [pop, stk]]).
rt_con(cjp(2, 1), [[rar(0), <-, 1]], [test, =, true]).
rt_con(cjp(2, 1), [[rar(0), <-, 0]], [test, =, false]).
rt_con(cjp(2, 2), [[rar(1), <-, 1]], [test, =, true])
rt_con(cjp(2, 2), [[rar(1), <-, 0]], [test, =, true]).
```

図３　分岐機能の定義

図３に、制御回路で実現する分岐機能のニーモニック定義を示す。分岐機能を実現するために必要な制御パス上のレジスタ転送とそのニーモニックを rt_con 述語を用いて定義するものである。また図４は、マイクロプログラムのアドレス順序情報とこのアドレス順序を実現するために用いる制御パス上のレジスタ転送

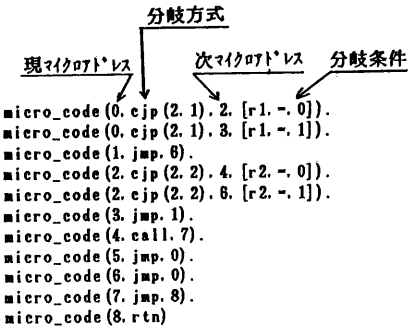


図4　マイクロプログラムのアドレス順序情報

との対応をmicro_code 述語を用いて示したものである。

（2）制御モデルの縮約アルゴリズム

前述したように制御構造モデルは、不必要な回路を含んだ冗長な回路になっている場合がある。本設計支援システムでは、以下のアルゴリズムを用いて冗長な部分の除去を行う。

（i）rt_con 述語で定義されるすべての制御パス上のレジスタ転送を抽出する。

(ii)（i）のレジスタ転送を実現するのに必要なすべての経路（パス）を抽出する。

(iii)制御記憶からのプログラム読出しを実現する制御パス上のレジスタ転送を抽出し、これを実行するのに必要なすべての経路を抽出する。

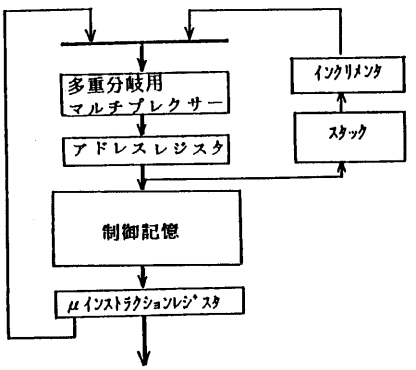


図5　縮約された制御回路モデル

以上（i）～(iii)の手続きにより抽出された経路の構成が、冗長性の少ない制御論理構成となる。

図5は、図2に示した制御回路モデルで図3に示した制御パス上のレジスタ転送を実現するとき、本アルゴリズムを用いて制御モデルの縮約をおこなった例を示す。この例では、冗長であったループカウンタの回路が除去されている。

（3）多重分岐用マルチプレクサーの合成アルゴリズム

マイクロプログラムを用いる制御回路は、機能分岐、条件分岐、無条件分岐、サブルーチンの呼び出し、復帰等、柔軟な分岐操作を実行することができる。この分岐操作は、制御記憶のアドレスを指定するアドレスレジスタへのレジスタ転送を制御することにより、実現する。そのため、アドレスレジスタを終点とするレジスタ転送動作は非常に複雑なものとなるため、マルチプレクサを組み合わせてレジスタ転送路の選択回路を構築することが必要となる。本システムでは以下のアルゴリズムに従い rt_con 述語により表現される制御パス上のレジスタ転送がすべて実行可能となるパスの自動合成を行なう。

（i）制御記憶のアドレスレジスタ CMAR とし、そのビット長を N ビット、CMAR の i ビットめを CMAR(i) ($0 \leq i \leq N-1$) と記述する。
任意の i ($0 \leq i \leq N-1$) に対し、CMAR(i) を終点とするすべてのレジスタ転送から、多重分岐用マルチプレクサーの入力点を抽出する。この多重分岐用マルチプレクサーの入力点の集合を L_i と定義する。

(ii) 以下の手続きに従い、L_i ($0 \leq i \leq N-1$) を要素とする集合族 F_k を定義する。

1) $L_0 \in F_0$ 、i＝0 とし、2)へゆく。

2) $L_i \in F_*$ が成立し、$L_i = L_{i+1}$が成立するならば、$L_{i+1} \in F_*$として 4)へゆく。

3) $L_i \in F_*$ が成立し、$L_i \neq L_{i+1}$が成立するならば、$L_{i+1} \in F_{*+1}$として 4)へゆく。

4) i=N-1 ならば終了、それ以外は i=i+1 として 2)の操作に移る。

(iii)任意の集合族 F_i に対しマルチプレクサーmbr(i)を定義する。$F_i = \{L_{i1}, L_{i2}, \ldots, L_{im}\}$ とするとき mbr(i)の出力点は

{CMAR(i1), CMAR(i2), ... CMAR(im)} であり、入力点は集合 $R\text{-}L_{i1} \vee L_{i2} \vee \cdots \vee L_{im}$ の各要素である。

図6は本アルゴリズムを用いて、図3示したレジスタ転送が実行できる制御パスを合成した例を示す。

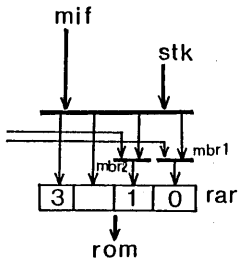
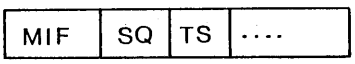


図6　合成された多重分岐用マルチプレクサ

2．2 条件判定デコーダーの合成

条件判定デコーダーは、論理回路の状態とマイクロ命令の一部のフィールドによって定義されるテスト条件と論理回路の状態とを比較することによって条件判定を行ない、その判定結果を出力する組合せ論理である。この条件判定デコーダーを合成するには、マイクロ命令においてテスト条件を定義する部分のフィールド構成方式ついては与えられていなければならない。本論文では、このフィールド構成方式については図7に示すデフォルトを設定する。

以下では、条件判定デコーダーの合成アルゴリズムを説明する。



SQﾌｨｰﾙﾄﾞ　:分岐機能を指定するﾌｨｰﾙﾄﾞ
TSﾌｨｰﾙﾄﾞ　:分岐ﾃｽﾄを指定するﾌｨｰﾙﾄﾞ
MIFﾌｨｰﾙﾄﾞ:次ｱﾄﾞﾚｽを指定するﾌｨｰﾙﾄﾞ

図7　マイクロ命令のフォーマット
アドレス順序制御部

（1）条件パターンの抽出

分岐条件が成立するときの条件判定デコーダーの入力ビットパターンは、以下の操作により求める。

1）アドレス順序情報で使用されているすべての条件分岐を抽出する。この条件分岐の集合をBとする。

2）Bの要素の数をn、要素を a_i $(1 \leq i \leq n)$ とするとき、$^{\forall}a_i \in B$，に対し条件 a_i で使用されているレジスタをすべて抽出する。このとき、a_i に対し抽出したすべてのレジスタの集合を A_i とする。

3）$R\text{-}A_1 \vee A_2 \vee \cdots \vee A_i \vee \cdots \vee A_n$　で定義される集合Rを求める。Rはレジスタの集合であるため、任意の要素 r_i に対して写像Fを以下のように定義することができる。

r_i が1ビットのレジスタのとき

F： $r_i \rightarrow r_i$

r_i がmビットのレジスタのとき

(r_i-r_i (0, m-1) と表現できるとき)

F： $r_i(0, m-1) \rightarrow r_i(0), r_i(1), \ldots, r_i(m-1)$

次に　F： R→L　で定義される集合Lを求める。

4）Lは1ビット単位のレジスタを要素とする集合であり、ブール代数の変数とみなすことができる。従って、$^{\forall}a_i \in B$ に対し条件 a_i が成立するようにLの各要素（ブール変数）に値を与えることが可能である。ここで、条件 a_i を満足するようにLの要素に値（0あるいは1）を与える写像をSとし、

S： $a_i \rightarrow X_i$

と記述する。Lのすべての要素にSを適用することにより分岐条件が成立するときの入力ビットパターンを求めることができる。

（2）テスト条件の符号化

本論文で採用したアドレス制御を定義する部分のフィールド構成では、条件分岐で使用されるすべての分岐条件がTSフィールドで定義できなければならない。そのため、条件分岐で使用するすべての分岐条件を抽出し、その符号化を行う。図8はすべての分岐条件を符号情報としてTSフィールドに割り付けを行った例を示したものである。（1）の結果とこのTSフィールドの符号化情報から条件テスト成功時の入力ビットパターンを抽出することができる。

```
test_code([0], [r1, -, 1]).
test_code([1], [r2, -, 1]).
```

図8　TSフィールドの符号化情報

（３）出力パターンの抽出

前述したように条件分岐には、制御パス上のレジスタ転送を条件判定結果に応じて切り変えることにより実現する方式と、アドレスレジスタの一部に条件テストの結果を直接挿入することにより実現する方式がある。制御パス上のレジスタ転送を切り変える方式が使用されている場合、制御パス制御デコーダーが制御パス上のレジスタ転送を実行するための制御信号を出すため、条件テストの結果は制御パス制御デコーダーに出力される。また条件テストの結果をアドレスレジスタに挿入する方式が使用されている場合、各テスト条件に対しその条件判定結果はアドレスレジスタの所定のビットに出力される。条件判定デコーダーの出力ビットパターンは、テスト条件と判定結果の出力先との対応関係を解析することにより抽出される。

以上（１）～（３）の処理を実行することにより条件判定デコーダーが満足すべき真理値情報を得ることができる。図９は、図４に示したアドレス順序情報に対し本アルゴリズムを適用し得られた条件判定デコーダーの真理値情報を示す。

```
                TS r1 r2    t1 t2
dec2_io([0, x, 0], [0, 0]).
dec2_io([0, x, 1], [0, 1]).
dec2_io([1, 0, x], [0, 0]).
dec2_io([1, 1, x], [1, 0]).
```

図９　条件判定デコーダーの
真理値情報

２．３制御パス制御デコーダ合成アルゴリズム

制御パス制御デコーダーは、制御パスを構成しているマルチプレクサーやスタック等の機能回路モジュールの制御信号を発生するための組合せ論理である。そのためには、アドレス順序制御のためのレジスタ転送と制御記憶からマイクロプログラムの読みだし実行するためのレジスタ転送を解析して、制御パスの制御点に必要な制御信号を抽出しなければならない。制御パス制御デコーダー合成アルゴリズムは、以下の（１）～（３）の手続によって構成される。

（１）制御パス上の機能モジュール回路動作への展開

制御パス制御デコーダーは制御パス上の各レジスタ転送を正確に実行するための制御信号を発生させる組合せ論理であるため、まず必要な制御信号の抽出を行わなければならない。本アルゴリズムでは以下の手続きに従って必要な制御信号の抽出を行う。

（ｉ）制御パス上で実行すべきレジスタ転送を抽出し、ソースレジスタからディスティネーションレジスタへの経路探索を行ない、経路上に存在する機能モジュールの抽出を行う。

(ii) レジスタ転送で実行する操作を実現するためには、（１）で抽出した各機能モジュールにどのような動作をさせればよいか解析する。

(iii) (i), (ii)の操作をすべての制御パス上のレジスタ転送に適用する。

図１０は図３に示した制御パス上のレジスタ転送にアルゴリズムを適用し、制御パス上の各機能モジュールの動作に展開した例を示すものである。各機能回路モジュールの動作は、入出力端子の宣言と in1, add1, rt, pop といったキーワードによって表現される。

（２）制御パス上の制御点抽出

本論理設計支援システムにおいて、制御パス及びデータパスを構成する機能回路モジュールに関する情報は、フレームを用いて記述している。図１１は、条件分岐を制御するためのマルチプレクサ　mux に関する情報をフレームを用いて表現した例である。control_point スロットには、機能回路モジュールの制御点（制御入力端子）が格納されており、　また control スロットには制御信号と機能回路モジュールの動作との対応関係が格納されている。例えば図１１に示した機能回路モジュール mux は、制御点cin1をもち、 cin1=0 のときin2 に入力された信号を出力し、

```
branch(jmp, [[mux, in1], [mbr1, in1], [mbr2, in1], [rar, load]]).
branch(call, [[stk, push], [mux, in1], [mbr1, in1], [mbr2, in1], [rar, load]]).
branch(rtn, [[stk, pop], [inc, add1], [mux, in2], [mbr1, in1], [mbr2, in1], [rar, load]]).
branch(cjp(2, 1), [[mux, in1], [mbr1, in2], [mbr2, in1], [rar, load]]).
branch(cjp(2, 2), [[mux, in1], [mbr1, in1], [mbr2, in2], [rar, load]]).
```

図１０：分岐操作と機能モジュールの動作

```
<<< Frame Name >>> : mux
--- Slot ------ Facet ---- Value ----------------------------
  akomethod          value      multiplexer   機能回路ﾓｼﾞｭｰﾙの種類
  bit_length         value      3             ---- 入力数
  input              value      [in2,in1]
  control_point      value      [cin(0)]      ---制御点情報
  control            value      [[0],in2]  ┐
                                             ├--制御点と機能回路
  control            value      [[1],in1]  ┘  ﾓｼﾞｭｰﾙの関係
```

図１１　フレームで記述された機能回路モジュールの例

またcin1=1 のとき in1 に入力された信号が出力する。

本アルゴリズムは、制御パス上のレジスタ転送を実行するために必要なすべての機能回路モジュールの制御点を各フレームから抽出する。ここで抽出された制御点の集合は、制御パス制御デコーダーの各出力端子の信号出力先であり、１対１に出力端子と対応する。

（３）制御パス制御デコーダーの真理値情報の合成

（１）の手続きで抽出した機能回路モジュールの動作情報と、各機能回路モジュールのフレームに格納されている制御点及び動作情報から、制御パス上のレジスタ転送に対してどの機能回路モジュールにどのような制御信号ビットパターンを送ればよいか対応づける。

次にこの対応関係を満足するように、（２）の手続きで求めた制御点の各集合要素（ブール変数）に値を与え制御デコーダーの出力ビットパターンを求める。

また制御パス制御デコーダーの入力は、マイクロ命令のＳＱフィールドの値と条件判定の結果で構成される。従って各ＳＱフィールドの符号化情報及び条件判定結果の情報から、制御パス制御デコーダーの真理値情報ををを合成することができる。

図１２は本アルゴリズムで合成した制御パス制御デコーダーの真理値情報の例を示す。

```
                    mux  mbr1 mbr2 rar  sik  ink
        SQ
dec1_io([0,0,0],[0,0,0,1,0,0,0]).   -----jmp
dec1_io([0,0,1],[0,0,0,1,0,1,0]).   -----call
dec1_io([0,1,0],[1,0,0,1,1,0,1]).   ----rtn
dec1_io([0,1,1],[0,1,0,1,0,0,0]).   ----cjp(2,1)
dec1_io([1,0,0],[0,0,1,1,0,0,0]).   ----cjp(2,2)
```

図１２　制御パス制御デコーダーの真理値情報

３　評価

以上、検討してきたアルゴリズムを７klips の prolog を用いてプログラム化し、その評価を行なった。本アルゴリズムは、論理的記述、真理値情報を得るために網羅的探索処理から成っており、これらは prolog を用いて比較的容易に記述できた。

また図１３は、入力したマイクロプログラムのステップ数と制御回路を合成するのに要した計算時間との関係を示したものである。図13より本アルゴリズムの実行に要する計算時間は、マイクロプログラムのステップ数にほぼ比例して増加しているのがわかる。またステップ数が２４の場合とステップ数が２７の場合その計算時間は大きく変動している。これは、ガーベージコレクタの起動状況の差異によるものと考えられる。

一般のマイクロプロッセサーおよび専用プロセッサーは、数ｋステップのマイクロプログラムによって構成されるが、図１３の結果によると５０ステップ程度のマイクロプログラムを実行する制御回路の合成に要する計算時間は約数１０秒程度であり、また計算時間の増加傾向もほぼ線形的であることから、prolog の

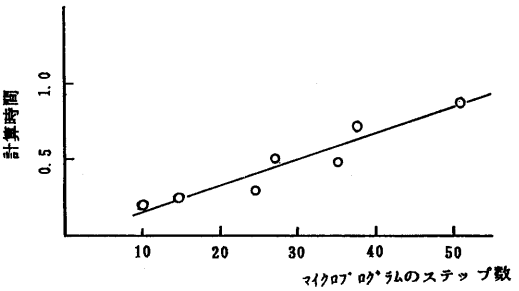


図１３　マイクロプログラムのステップ数と計算時間

処理系の高速化を勘案すると、本アルゴリズムは実用に耐えるものであると考えられる。

4　結言論

本研究で得られた結論は以下の通りである。

1）マイクロプログラム方式による制御論理回路の構成モデルを宣言的に記述することにより、制御回路構成に関する知識の蓄積が容易となり、種々の制御回路方式の構築が可能となった。

2）制御論理回路の合成アルゴリズムを検討し、論理型言語 prolog を用いて実現し、実際に合成を試みた結果、実用可能な回路が得られることを確認した。

3）論理合成手順のような論理的記述や真理値情報の網羅的探索等の処理はprologにより比較的容易に記述可能であることを確認した。

<<<参考文献>>>

1．横田、戸次、浜田: フレーム表現によるCMOS論理機能ﾌﾞﾛｯｸの対話設計支援:信学、情報システム全国大会、1985年11月

2．同上:論理設計知的支援用推論方式のﾌﾟﾛﾄﾀｲﾋﾟﾝｸﾞ:情処 VLSI CADへの知識工学の応用ｼﾝﾎﾟｼﾞｳﾑ論文集:pp. 11-15, 1986年1月

3．同上:論理設計知的支援用対話型仕様入力方式:情処、第32会(昭和61年前期)全国大会論文集、講演番号4L-1, 1986年3月

4．戸次、横田、浜田:論理設計知的支援用知識ベース構成方式:同上、講演番号4L-2, 1986年3月

5．横田、戸次、浜田:演算回路自動合成方式の検討:情処、設計自動化、34-3、1986年10月

6．萩原:ﾏｲｸﾛﾌﾟﾛｸﾞﾗﾐﾝｸﾞ:産業図書、1977年

7．馬場:ﾏｲｸﾛﾌﾟﾛｸﾞﾗﾐﾝｸﾞ:昭晃堂、1985年

8．M. Morris Mano:Computer Systerm Architecture:Prentice-hall Inc., 1976

9．C. Mead and L. Conway:Introduction to VLSI Systems:Addison-Wesley, 1980

10．T. Agerwala:Microprogram Optimization: A Survey: IEEE Trans. Computer, Vol. C-25, No. 10, Oct. 1976

11．高木:制御回路自動合成の一手法:情処、設計自動化 27-1、1985年7月

12．Brown. D. W.:A State Machine Synthesizer: Proc. 18th DA Conference, pp. 301-305 (1981)