マイクロプログラム方式の制御論理回路合成方式の検討†

戸次圭介†† 横田孝義†† 浜田亘曼††

高機能 LSI の論理設計期間を短縮するためには，優れた論理 DA システムが必要となる．論理設計はデータパス，制御回路，およびこれらの構成要素となる機能モジュールのそれぞれの設計に分類することができる．制御回路には結線論理方式とマイクロプログラム方式があるが，大規模で複雑な論理回路はほとんど後者の方式が採用されている．そのためマイクロプログラム方式の制御回路の設計を自動化することが重要となる．そこで本論文では，マイクロプログラム方式の制御回路の構成方式をモデル化し，この構成モデルに基づいた論理合成アルゴリズムを検討した．これにより，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部分の論理回路を自動合成することが可能となった．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．

1. まえがき

近年の VLSI の高集積化に伴い，それを活用して高機能の LSI をでき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 DDL や ISPS 等の仕様記述言語による抽象度の高い仕様記述[21)-23)]から詳細論理回路やマスクパターンを自動生成するシリコンコンパイラの研究が盛んに行われるようになってきた[13)-19)]．現在，詳細論理回路までの設計を自動化するシステムは，およそ以下3つの論理合成アルゴリズムにより構成されると考えられている[20)]．

（1） データパス合成アルゴリズム
（2） 機能モジュール合成アルゴリズム
（3） 制御回路合成アルゴリズム

（1），（2）のアルゴリズムについては，比較的早期より着手されてきた部分でありその成果についての報告も多い．また，制御回路の構成方式には，結線論理方式とマイクロプログラム方式があり，比較的小規模な論理回路の制御回路には前者の方式が用いられ，大規模な論理回路の制御回路には後者の方式が用いられている．結線論理方式による制御回路の自動合成アルゴリズムについては高木による報告がある．この方式を用いると，マイクロプログラムとして実現する論理とマイクロプログラムの読み出しを制御する論理が，1つの大きな組合せ論理として実現される．これはマイクロプログラム方式と全く異なる点である．現在のところマイクロプログラム方式による制御論理を自動合成するアルゴリズムについての報告は見当たらない．そのため本論文では，マイクロプログラム制御方式の制御回路の回路モデルを記述し，記述されたモデルに従って詳細論理回路を自動合成する方式について検討する．

以下，2章ではマイクロプログラム制御方式の回路構成モデルについて考察する．3章では構成モデルに基づいた論理合成アルゴリズムについて検討する．4章では本アルゴリズムを論理型言語 Prolog を用いてプログラム化し，実際に回路の合成を試みた結果について報告する．

2. 制御回路モデル

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，データパス部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み出す．
2) 1)で読み出されたマイクロ命令の情報の一部をデ

† A Method for Microprogramming Control Logic Synthesis by KEISUKE BEKKI, TAKAYOSHI YOKOTA and NOBUHIRO HAMADA (The 3rd Department, Hitachi Research Laboratory, Hitachi Ltd.).

†† (株)日立製作所日立研究所第3部

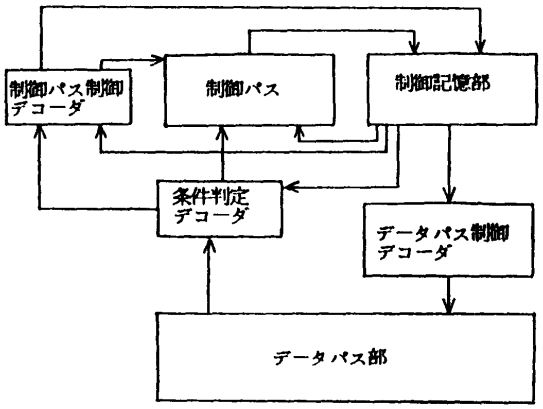


図 1 制御回路のブロック構成
Fig. 1 Schematic block diagram of control logic circuit.

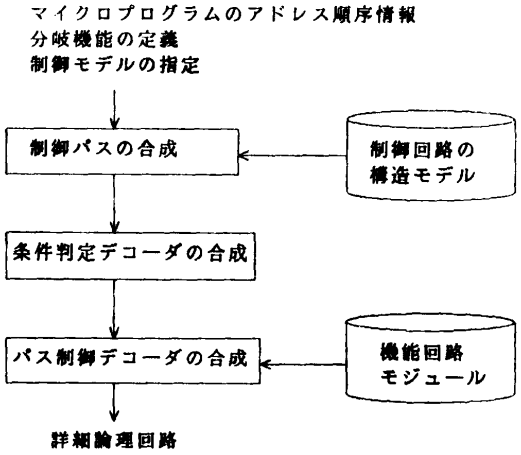


図 2 処理の流れ
Fig. 2 Flow of control logic synthesis.

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，次に読み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4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，次アドレスを現アドレス＋1にしたりマイクロ命令のリテラル値にしたりする操作のように，制御パス上のデータ転送経路を変え，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)

(iv) 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と制御パス制御デコーダの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

条件判定デコーダは，(ii)の条件分岐を実現するために，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に出力する組合せ論理である．

(b) 制御パス制御デコーダ

普通，制御パスは一般の論理回路のデータパスと同様にレジスタ，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，制御パス上のデータ転送経路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，制御パス制御デコーダを自動合成する方式について述べる．

3. 制御論理合成アルゴリズム

図2は，本論文で検討する制御論理合成アルゴリズムの処理手順を示したものである．以下図2に従ってアルゴリズムを検討する．

3.1 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，実現すべきマイクロプログラムの分岐方式によって大きく異なる．そのため，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数は少なくなるため，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．

しかしこの点を強調すると，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．これを避けるために考案した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．

表 1 アドレス分岐機能の記述に使用する記号
Table 1 Defined symbols to describe branch scheme.

モジュール	制御モデルの記述に使用されているモジュール mif : mir に割り当てられた次アドレスを指定するフィールド lcf : mir に割り当てられたループの回数を指定するフィールド external_register : 外部レジスタ
リテラル	数字 (10進数) で表現する．
分岐条件テスト	test, =, true : 条件テストの成功を表現する． test, =, false : 条件テストの失敗を表現する．
演算子	+, −, and, or, not, push, pop <− : データの転送を表現する．

図 3 に制御構造モデルの例を示す．なお図3においてパス統合部は，機能分岐や条件分岐を統合的に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．

図 4 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 (Register Transfer Expression for Control Circuit) 述語を用いて定義するものである．この図において，第4行めは，条件分岐 cjp (2, 1) を定義したもので，分岐条件テストが成立した時，rar (0)←1 を実行する．また**表 1** に，アドレス分岐機能を記述するために許される記述の一覧を示す．また**図 5** に示す micro_code 述語は，マイクロプログラムのアドレス順序情報を，制御パス上のレジスタ転送のニーモニックを用いて表現したものである．

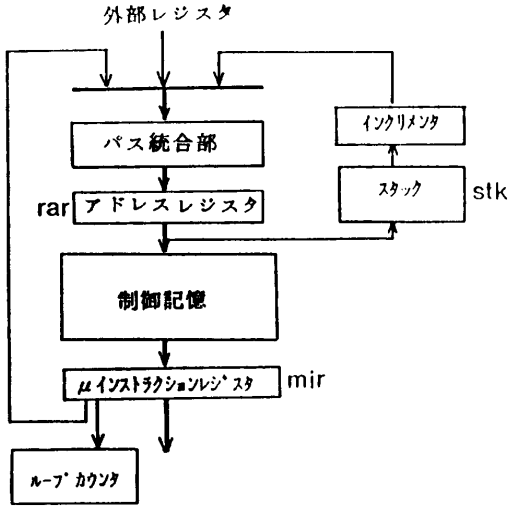


図 3 制御回路モデルの例
Fig. 3 Control logic circuit model for microprogramming.

表 2 は，制御モデルの構造を記述するための記号の一覧を示す．

図 6 は，これを用いて図3に示した制御回路モデルの構造を記述したものである．このように表2の記号を用いると制御モデルの構造を宣言的に記述することが可能となる．

```
分岐方式の          分岐のための
ニーモニック        動作定義
    ↓                  ↓
rt_con(jmp, [[rar, <−, mif]]).
rt_con(jps, [[stk, <−, rar], [rar, <−, mif]]).
rt_con(rtn, [[rar, <−, stk, +1], [pop, stk]]).        分岐の起動条件
rt_con(cjp(2, 1), [[rar(0), <−, 1],                          ↙
                   [rar(1, 3), <−, mif(1, 3)]], [test, =, 1]).
rt_con(cjp(2, 1), [[rar(0), <−, 0],
                   [rar(1, 3), <−, mif(1, 3)]], [test, =, 0]).
rt_con(cjp(2, 2), [[rar(1), <−, 1],
                   [rar(0), <−, mif(0)],
                   [rar(2, 3), <−, mif(2, 3)]], [test, =, 1]).
rt_con(cjp(2, 2), [[rar(1), <−, 0],
                   [rar(0), <−, mif(0)],
                   [rar(2, 3), <−, mif(2, 3)]], [test, =, 0]).
```

図 4 分岐機能の定義
Fig. 4 A definition of branch scheme for microprogram.

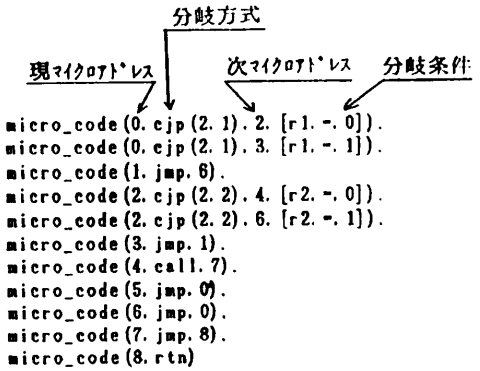


図 5 マイクロプログラムのアドレス順序情報
Fig. 5 A definition of microprogram sequence.

表 2 制御モデルの記述に使用する記号
Table 2 Defined symbols to describe control model structure.

モジュール	
レジスタ	r∗, mir (Micro Instruction Register), rar (ROM Address Register), external_register：外部レジスタ mif：mir に割り当てられた次アドレスを指定するフィールド lcf：mir に割り当てられたループの回数を指定するフィールド
スタック	stk∗
カウンタ	cntr∗ ∗cntr
インクリメンタ	inc∗
ALU	alu∗
マルチプレクサ	mux∗, mbr∗
パス統合部	path∗, ∗path
端子	
入力	in∗, ∗in
出力	out∗, ∗out
リテラル	10進数

∗ は任意の文字列

なお図2に示すように設計者が入力すべき情報は，マイクロプログラムのアドレス順序情報と分岐機能のニーモニック定義と制御モデルを指定するための情報である．また制御モデルと機能モジュールの情報は，ライブラリとしてシステム内に蓄積しておく情報である．

（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，高汎用性のため特定の用途に対しては，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） micro_code で引用される分岐機能を実行するために必要な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行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み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．

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7は，図3に示した制御回路モデルで図4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行った例を示す．この例では，冗長であったループカウンタの回路

```
link([mif,out],[mux,in1]).
link([external_register,out],[mux,in2]).
link([inc,out],[mux,in3]).
link([mux,out],[path,in1]).
link(0,[path,in2]).
link(1,[path,in3]).
link([path,out],[rar,in]).
link([rar,out],[stk,in]).
link([rar,out],[cm,address_in]),
link([cm,out],[mir,in]).
link([lcf,out],[lpcntr,in]).
```

図 6 制御モデルの構造記述例
Fig. 6 A definition of control model structure.

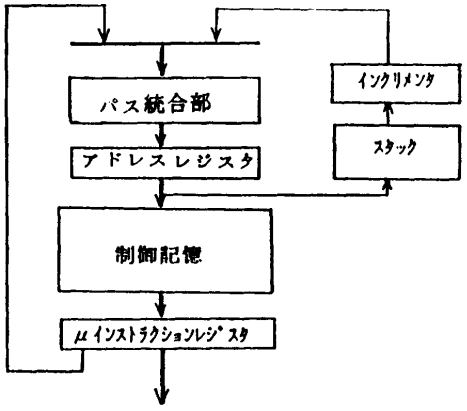


図 7 縮約された制御回路モデル
Fig. 7 Reduced and instanciated control circuit.

と外部レジスタからの入力が除去されている．

（3） パス統合部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，柔軟な分岐操作を統合して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 micro_code 述語で引用している分岐機能に必要な制御パス上のレジスタ転送をすべて実行可能とする制御パスの自動合成を行う．

（i） パス統合部の出力先を CMAR とし，そのビット長を N ビット，CMAR の i ビットめを CMAR (i) $(0 \leqq i \leqq N-1)$ と記述する．

任意の i $(0 \leqq i \leqq N-1)$ に対し，CMAR (i) を終点とするすべてのレジスタ転送から，パス統合部の入力点を抽出する．この入力点の集合を L_i と定義する．

(ii) 以下の手続きに従い，L_i $(0 \leqq i \leqq N-1)$ 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) $L_i \in F_e$ が成立し，$L_i = L_{i+1}$ が成立するならば，$L_{i+1} \in F_e$ として 4)へゆく．

3) $L_i \in F_e$ が成立し，$L_i \neq L_{i+1}$ が成立するならば，$L_{i+1} \in F_{e+1}$ 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 に対してマルチプレクサ mbr (i) を定義する．$F_i = \{L_{i1}, L_{i2}, \cdots L_{im}\}$ 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 $R = L_{i1} \vee L_{i2} \vee \cdots \vee L_{im}$ の各要素である．

図 8 は本アルゴリズムを用いて，図3に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．

3.2 条件判定デコーダの合成

条件判定デコーダは，LSI ごとに機能の異なる部分である．これは，論理回路の状態（フラグレジスタの値等）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のマッチングをとることにより条件判定を行い，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を与えなければならない．本論文では，このフィールド構成方式については**図 9** に示すデフォルトを設定する．

以下では，条件判定デコーダの合成アルゴリズムを説明する．

(1) 入力パターンの抽出

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

i) 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．

ii) Bの要素の数を n，要素を a_i $(1 \leqq i \leqq n)$ 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．

iii) $R = A_1 \vee A_2 \vee \cdots \vee A_i \vee \cdots \vee A_n$ 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

$F: r_i \rightarrow r_i$

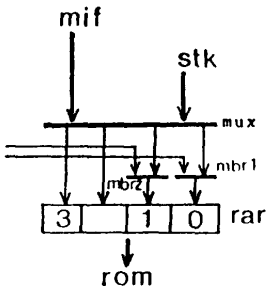
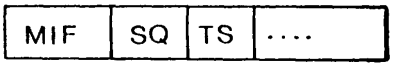


図 8 合成された多重分岐用マルチプレクサ
Fig. 8 Synthesized logic circuit for address control.



SQフィールド：分岐機能を指定するフィールド
TSフィールド：分岐テストを指定するフィールド
MIFフィールド：次アドレスを指定するフィールド

図 9 マイクロ命令のフォーマットアドレス順序制御部
Fig. 9 Instruction fields structure for address control.

フィールドに割り付けられた符号 → テストする条件

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

図 10 TS フィールドの符号化情報
Fig. 10 Encoded instructions defined in TS field.

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)

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

次に $F: R \rightarrow L$ で定義される集合 L を求める．

iv) 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．したがって，$a_i \in B$ 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

図9の TS フィールドでは，条件分岐で使用するすべての判定すべき条件（例えば，$r1=1$ 等）が指定できなければならない．そのため，条件分岐で使用される分岐条件をすべて抽出し，その符号化を行う．**図 10**

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．これは，TS フィールドの値が0のとき $r1=1$ が成立するかどうか判定し，1のとき $r2=1$ が成立するかどうか判定するように符号化している．(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条件判定デコーダへの入力ビットパターンを抽出することができる．

(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が満足すべき真理値情報を得ることができる．**図 11** は，図6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の真理値情報を示す．

3.3 パス制御デコーダの合成

制御パス制御デコーダは，制御パスを構成しているマルチプレクサ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出しを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．

```
                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 11 条件判定デコーダの真理値情報
Fig. 11 Truth table for condition decoder.

(1) パス上の機能モジュールの動作への展開

制御パス制御デコーダ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 12 は図4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 in 1, add 1, 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，制御パスおよ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 13** 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 cin(0) をもち，cin (0)=0 のとき in 2 に入力された信号を

```
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 12 分岐操作と機能モジュールの動作
Fig. 12 Behaviors of functional logic circuits to perform branch control operation.

出力し，また cin (0)＝1 のとき in 1 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．

```
<<< Frame Name >>> : mux
--- Slot ------ Facet ---- Value ----------------------------
akomethod        value    multiplexer    機能回路モジュールの種類
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]      モジュールの関係
```

図 13 フレームで記述された機能回路モジュールの例
Fig. 13 Functional logic circuit specification described using frame.

（3） パス制御デコーダ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および動作情報から，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の入力は，図9のマイクロ命令における分岐機能を指定する SQ フィールドの値と条件判定の結果で構成される．したがって各 SQ フィールドの符号化情報および条件判定結果の情報から，制御パス制御デコーダの真理値情報を合成することができる．**図 14** はこのアルゴリズムで合成した制御パス制御デコーダの真理値情報の例を示す．

```
                SQ    mux  mbr1  mbr2  rar  sik  ink
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 14 制御パス制御デコーダの真理値情報
Fig. 14 Truth table for control decoder.

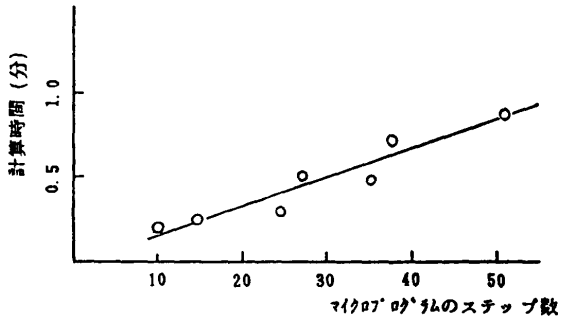


図 15 マイクロプログラムのステップ数と計算時間
Fig. 15 Synthesis time vs total steps of microprogram.

4. 評　価

以上検討してきたアルゴリズムを7 KLIPS の Prolog 処理系上でプログラム化し，その評価を行った．本アルゴリズムは，そのほとんどが制御回路の構造記述における経路探索や回路内で成立する論理関係の真理値情報への展開処理等，すべての条件を満足する解の探索処理からなっている．これらの処理は Prolog がもつバックトラックおよびユニフィケーション機構を用いることにより容易に記述することができた．

次に7種類のアドレス順序情報に対し本アルゴリズムを適用し詳細論理回路を合成した．ここで得られたパス統合部の詳細なマルチプレクサ構成は，いずれも人手による設計結果に近いものであった．したがって，本方式を真理値情報から高品質な詳細論理回路を合成するアルゴリズムと組み合わせて用いると，実用可能な詳細論理回路を自動合成することができる．

図 15 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．本方式においてマイクロプログラムのステップ数が処理に影響するのは，条件判定デコーダの合成部である．したがって図 15 は，条件判定デコーダ合成部の性能評価と考えることができるであろう．この処理は本文で述べたように Prolog の機能を利用した探索処理であるため，マイクロプログラムのステップ数に対し指数関数的に計算時間が増大すると予測される．一般のマイク

ロプロセッサは数kステップのマイクロプログラムによって構成される．このような大規模な回路に条件判定デコーダ合成処理を適用するときの性能評価については今後の課題である．

5. むすび

本論文ではマイクロプログラム制御方式の制御論理回路を自動設計するアルゴリズムについて検討した．得られた結論は以下のとおりである．

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再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の構成方式を提案した．また，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．

謝辞 本論文作成に当たり，本研究の機会を与えてくださった当所所長川本幸雄博士，谷中雅雄第3部部長，VLSI の設計自動化手法について御指導御討論いただいた前島英雄博士，桂晃洋氏に感謝いたします．

参考文献

1) 横田，戸次，浜田：フレーム表現による CMOS 論理機能ブロックの対話設計支援，信学，情報システム全国大会論文集，Ⅲ，pp. 197-198 (1985. 11).
2) 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 VLSI CAD への知識工学の応用シンポジウム論文集，pp. 11-15 (1986. 1).
3) 横田，戸次，浜田：論理設計知的支援用対話型仕様入力方式，第 32 回情報処理学会全国大会論文集，4L-1 (1986. 3).
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，第 32 回情報処理学会全国大会論文集，4L-2 (1986. 3).
5) 横田，戸次，浜田：演算回路自動合成方式の検討，情報処理学会，設計自動化研究会資料，34-3 (1986. 10).
6) 萩原：マイクロプログラミング，産業図書，東京 (1977).
7) 馬場：マイクロプログラミング，昭晃堂，東京 (1985).
8) Mano, M. M.: *Computer System Architecture*, Prentice-Hall Inc. (1976).
9) Mead, C. and Conway, L.: *Introduction to VLSI Systems*, Addison-Wesley (1980).
10) Agerwala, T.: Microprogram Optimization; A Survey, *IEEE Trans. Comput.*, Vol. C-25, No. 10, pp. 962-973 (1976).
11) 高木：制御回路自動合成の一手法，情報処理学会，設計自動化研究会資料，27-1 (1985. 7).
12) Brown, D. W.: A State Machine Synthesizer, *Proc. 18th DA Conference*, pp. 301-305 (1981).
13) Thomas, D. E. et al.: Automatic Data Path Synthesis, *IEEE Comput.*, Vol. 16, pp. 59-70 (1983).
14) Kowalsky, T. J. et al.: The VLSI Design Automation Assistant to Silicon, *IEEE, Design & Test*, Vol. 2, No. 4, pp. 33-43 (1985).
15) 上原貴夫：機能，論理設計アルゴリズム，電子通信学会誌，Vol. 69, No. 4, pp. 370-376 (1986).
16) 浜田亘曼ほか：LSI/CAD 用エキスパートシステム，日立評論，Vol. 68, No. 3, pp. 69-76 (1987).
17) Yokota, T. et al.: A VLSI Design Automation System Using Frames and Logic Programing, *Proc. 3rd Conference on Artificial Intelligence Applications*, pp. 296-301 (Feb. 1987).
18) Southard, MacPitts, M. J.: An Approach to Silicon Compilation, *IEEE Comput.*, Vol. 16, pp. 74-82 (1983).
19) Johannsen, D. and Blocks, B.: A Silicon Compiler, *Proc. 16th Design Automation Conference*, pp. 310-313 (1979).
20) 高木 茂：VLSI 設計 CAD の最近の動向，情報処理，Vol. 28, No. 5, pp. 581-589 (1987).
21) 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).
22) 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).
23) German, S. E. et al.: Zeus: A Language for Expressing Algorithms in Hardware, *IEEE Comput.*, Vol. 18, No. 2, pp. 55-65 (1985).

(昭和62年9月28日受付)
(昭和63年4月14日採録)

戸次 圭介（正会員）

昭和33年生．昭和57年大阪大学原子力工学科卒業．昭和59年同大学院修士課程修了．同年(株)日立製作所日立研究所に入社．知識工学を応用した VLSI 設計支援システムの研究に従事．電子情報通信学会会員．

横田 孝義（正会員）

昭和31年生．昭和54年東京工業大学電子物理工学科卒業．昭和59年同大学院博士課程修了．工学博士．同年(株)日立製作所日立研究所に入社．知識工学を応用したVLSI設計支援システムの研究に従事．人工知能学会，IEEE各会員．

浜田 亘曼（正会員）

昭和41年，大阪大学工学部電気工学科卒業．同43年，同学科修士修了．同年(株)日立製作所日立研究所入社．以来アナログ制御系，計算制御システム，高水準言語，分散型計装システム等の開発に従事．その間，昭和52年には米国カーネギーメロン大学，計算科学部にて客員研究生として分散型OSの研究に従事．現在，知識処理，EWSを応用したVLSI設計支援システムの研究開発に従事．工学博士．