マイクロプログラム ジェネレータ

馬場 敬信　萩原 宏
（京都大学 工学部）

1. はじめに

近年、高速で比較的大容量のＩＣメモリが、制御記憶として用いられるようになって、ダイナミック・マイクロプログラミングが実用の計算機で採用されるようになり、マイクロ・ダイアグノスティクス，エミュレーション，高級言語処理マシーンなどを中心に使われている。

これに伴って、マイクロプログラムの記述言語とその処理システムもいくつか考えられている。vertical type のマイクロ命令を持つマシーンに対して考えられたＭＰＬ[2]，シミュレーションと評価を主目的とするＭＰＧＳ[3]，ＩＢＭ３６０／４０以後のマシーンに対して使われたＣＡＳ[1]と、現在ＩＢＭで開発中のＭＤＳ[4]などがその主なものである。

これらのシステムに対し本論文のマイクロプログラム・ジェネレータの特徴はまず第１にhorizontal type マイクロ命令の生成を目標としていることである。現在、中・大型機の多くは、並列性を最大限に利用できるhorizontal typeマイクロ命令を採用しているが、プログラムの書きにくさがその最大の欠点であり、これの解消を目標としている。第２に、従来の記述システムで問題となっていた生成されるマイクロプログラムの効率を重視して、マイクロプログラミングを行う際に、プログラマが最適化のために用いる手法を、最適化法の中に取り入れている。第３に、マシーンの記述は、全体を表形式とし、高位の、機能的な面の記述を行うことにより、特定のハードウェア、テクノロジィに依存しない記述ができる。第４に、アルゴリズムの記述は、記述されたマシーンに対して行われ、式を用いた記述，ラベルによるシーケンスの記述ができる事が主な特徴である。

Fig.1に示すようにマイクロプログラム記述言語(μPL)は、システム記述部(ＭＤＳ)とアルゴリズム記述部(ＡＤＳ)とから成り、マイクロプログラムジェネレータはＡＤＳで記述されたアルゴリズムを、ＭＤＳで記述されたマシーンで実行可能なマイクロ命令列に変換する。一方、これはシミュレータに対する入力となる。

我々は、本論文に於て、マシーン記述の概要についてハードウェアとの対応を中心に述べると共に、ＭＤＳを変換してマシーン記述テーブル(ＭＤＴ)をどのように構成しているかについて述べる。

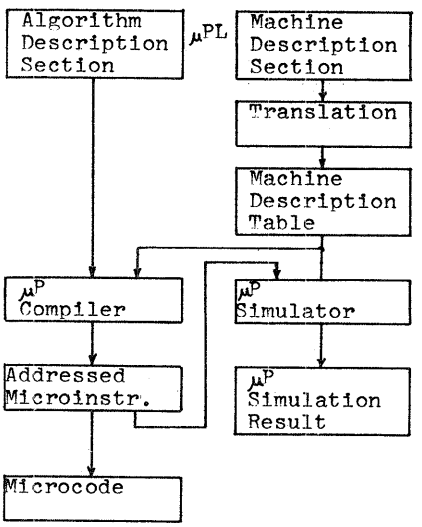

Fig.1 μP Generator

2. マシーン記述部（ＭＤＳ）

2－1. マイクロプログラム制御の方式について

マイクロプログラム制御を実際のマシーンで実現するために一般的に使われている手法と、以下の項で使用する言葉の定義を行う。

計算機の動作は、register to register 転送、flip-flop の状態のテスト、1ビットのシフト等の最も基本的な動作から成っている。これを micro operation(μOP)と呼びμOPに対する命令を micro order(μO)、その2進コードを micro operation code(μOC)と定義する。

ある一定の時間を区切ったとき(普通は machine cycle に一致)、その時間内に行ないうるμOPを制御する命令語が考えられるが、これを micro instruction(μI)と定義し、その2進コードを micro code(μC)と定義する。

μI列が microprogram(μP)である。

定義から明らかなように、μIはそのマシーンのμOを要素として持つ集合であり、従ってμCはμOCの集合を何らかの形でコード化したものである。コード化法によって、従来、Fig.2のような vertical type(machine code type)のものと、horizontal type(function field type)のものが考えられている。

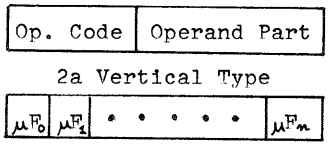

Fig.2 μI Format

vertical type μIは、いくつかのμOPのシーケンスをコード化したものであり、数種類のμIが含むμOPの全体がそのマシーンのすべてのμOPを含む。具体例としては、Interdata 3, 4 がある。

horizontal type μIは、もともとμIの各ビットが1つのμOPに対応するものとして考えられているが、[5] 実際には、マシーンの構造上、同時に指定できないものがあるので、これらに対するμOCがまとめてコード化されているのが普通でありfield(μF)と呼ばれている。

前者の利点は、μPの書き易さであり、後者の利点はマシーンの並列に行える動作を最大限に利用できるということである。現在、後者の利点に重きを置く中・大型機では、horizontal type を採用している。

ここでは、horizontal type を対象としているが、vertical type への適用も容易である。

2-2 制御記述

マシーン記述部では、動作の記述をμOP単位とし各μOPにはそれを制御するμOが制御記述として付けられる。この際、μOの任意の組合せが意味のある動作を制御できるわけでなく、その組合せには一定の制限条件がある。

(1) 同一μIで指定の必要な場合

(i) terminal を介してのregister間のデータの転送

例. terminal LT → terminal BUSB → …… → counterR

(ii) ALUの動作を指定する複数個のμO

例. μO_1: shift left　μO_2: 桁数指定

(iii) 演算を行うμOと演算結果に対するテストを指定するμO

例. ALUA+ALUB → BUSA　BUSA=0?

(iv) テストμOとこれに対応するブランチ・アドレスを指定するμO

例. BUSA=0 なら $(0B00)_{16}$へ

BUSA≠0 なら アドレスレジスタのインクリメント

(v)定数のビットパターンと定数を指定するμO

(i),(iii)は、terminalに記憶の機能がないことによる。他は、いくつかのμOが組になって一つの動作を指定する場合である。

(2)同一μIで指定できない場合

(i)同一fieldに属するμOは原則として同時に指定できない。

(ii)ある変数をdestinationとする転送μOまたはセットμO(μO_1)に対し、同一変数をsourceとする転送μOまたはこの変数に対するテストμO(μO_2)があって記述順がμO_i, μO_jで

$$T(\mu O_i) < T(\mu O_j) \quad i, j = 1, 2$$

のとき。ただしTはμOの実行されるタイミングを、<は実行順序を表し右辺の方が早く実行されることを表す。

例. μO_1:counterR←registerU, μO_2:decrementRで、$T(\mu O_1) < T(\mu O_2)$なら、μO_2は次のμIに入れなければならない。

(iii)アルゴリズム記述部でプログラムの流れが変るとき、その前後のμO

(3)異なるμI中のμO間に一定の関係がある場合

(i)メモリに対して読出し、もしくは書込みを指定するμOのあるμIと、データが実際に読出されて使われるμI、或いは書込みデータの転送を行うμIとが異なる場合。

(ii)instructionの終了をその前のμIで指定する必要のある場合。

2-3 制御記述の仕様

2-2のようなμO間の関係を記述するために、μOPを制御するμOの組を次のような仕様で表現する。

(1)μOの論理式表現

μOの指定を禁止すること、いくつかのμOを同時に指定すべきこと、及び、いくつかのμOのどれか1つを指定すべきことをそれぞれ¬, ・, +の論理記号を用いて表す。

(2)条件指定

1つのμOが2つ以上のμOPを制御する場合に必要となり

<μOの論理式>

で、μPコンパイラは、論理式で表現されたμOが指定されることを確かめる。

(3)位置指定

実際に動作の行われるμIの位置を基準として、μOを指定すべきμIの位置を次のように表す。

[μOの論理式](n) <μOの論理式>(n)

ここでnは整数で、nが負の場合は実際の動作の行われるときのμIより前に指定が必要なことを表す。[ ]は実際に指定の必要なことを、< >は条件指定を表す。

2-4 マシーン記述部の仕様

マシーン記述部は、8つの部分より成る。次に、各テーブルの仕様とその意味について述べる。

2-4-1 Field Definition Table(FDT)

horizontal type μIを分割するfield(μF)を記述する。各field間には、一定の実行順序があり、これは次に述べるTiming Tableによって記述される。例ではBBという6ビット長のfieldとP1という1ビット長のfieldがあってBBはT1というタイミングで実行されP1はparity bitであることを記述している。

```
(FDT)
  BB 6 T1
  P1 1 P

(MT)
  U 001000

(PST)
  P1 O BB,AB,AA,BA,SP,OP,EX,CL

(TIM)
  T1 1

(AGS)
  U [0:11] F TS[1:6],JA[0:5]/*UNC*/
```

Fig.3 テーブルの記述例

2-4-2 Timing Table(TIM)

1つのμI中でのμOの実行順序を記述する。タイミング名とその実行順序を表す値とから成る。

2-4-3 Mnemonic Table(MT)

μOの属するμFと対応するμOCとから成る。これとField Definition Tableによってμ Iの各fieldとμOCの関係が記述される。

2-4-4 Parity Specification Table(PST)

parityを取るμF名と、evenparityがoddparityかを指定する。例では、P1というodd parityをBB～CLまで取ることを表す。

2-4-5 Address Generation Schema(AGS)

アルゴリズム記述に対して制御記憶の絶対番地の割当てと、分岐のためのμOとビットパターンの生成を行うために、制御記憶のアドレスレジスタに対するアドレスの生成法を記述する。

アドレスの生成法としては

(1) increment(I) (2)μF(F) (3)address stack(V) (4)wired logic(H)

の4種類によるものを考えている。例は、Unconditional branchの場合で、アドレスレジスタの[0:11]に、μFのTSとJAからのビットパターンがセットできることを表す。

2-4-6 Variable Table(VT)

アルゴリズム記述部で変数として使用されるregister, counter, flip-flop, scratch pad memory及びmemoryについて記述する。

```
U     REG 32 OD2 |U[8:11]=0|/*ULU*/ [0:31]→BUSB[0:31]/*U*/
BUSB  TER 32 OD2 [0:31]→ALUB[0:31]/*NB*/
R     COU 8 OD2 |R[0;7]≠0|/*RNZ*/ 2∇[0:3]/*DECR*/ [1:7]→SPAR[0:6]/*G*/
NZ    FF  1 |NZ=0|/*NZO*/ BUSA≠0/*FIXCHK*/
MM    MEM 8 131 MAR MDR -1 T2 T4 /*[RW,I](-1)*/ /*[CW,I](-1)*/
SPM   SPM 32 128 SPAR SPMR 0 T1 T4 /*<SP>*/ /*<SP>*/ 10X/*U0*/,11X/*U1*/,18X/*U2*/
```

例を中心に説明する。registerUは、中が32ビットでULUというμOによって8から11ビットまでが0かどうかをテストできる。(以下/*, */にはさまれた部

分はすべてその直前の動作を制御するμOの記述であり説明を省略する。) またその内容をBUSBに転送できる。BUSBについても同様であるがterminal変数はアルゴリズム記述部で使用できない点が異なる。

counter Rは、巾が8ビットでその値が0でないことをテストでき2ずつその内容をdecrementできる。またその内容をSPARに転送できる。

flip flop NZは、1ビットで、その値が0であることをテストできBUSA≠0なら1にセットされる。

memory MMは、1語8ビット、容量は131K語で、そのアドレスレジスタ名はMAR、データ・レジスタ名はMDRである。MARへのセットは、読み出すデータを使うμI或いは書込みデータをmemoryに送るμIの1つ前で指定する必要があり、読出し、書込みのタイミングとそれに必要なμOは、それぞれT2, T4及び[RW,I](-1),[CW,I](-1)である。

scratch pad memory とmemoryの違いは、前者の場合、データ・レジスタがterminalであってよいことと、アドレス・レジスタに定数をセットすること(例では$(10)_{16}$, ⋯ $(18)_{16}$がセットできる)が記述できることである。

2-4-7 Constant Table (CT)

μP制御方式計算機での定数の発生法には、wired logicによる方法と、μFのビット・パターンを用いる方法とがある。後者の場合μFへのビットパターンの与え方によって、発生させる定数を変えられる。例では、第1行が前者にあてはまり、ALUB上に32ビット巾の1が生成され、第2行は、BBとEXの2つのμFのビットパターンがBUSBに生成されることを記述している。

```
000000...000000001 32                    ALUB[0:31]/*C1*/
000000...0XXXXXXXX 32 BB[3:5],EX[0:4] BUSB[0:31]/*C*/
```

2-4-8 Operator Table (OPT)

ALUの機能の記述は、従来の記述言語では、adder, complementerなどの個々のユニットを記述することにより行われていたが、ここではマシーンに独立な記述をするために、ALUの機能的な面のみに注目し、入力端子とオペレーション及び、結果の出力端子によって記述する。例の第1行は、flip flop SCの内容を左端に置いて、ALUMの内容をlogical right shift L、その結果をBUSAに出力することを表す。この際、UNderflowがSCにセットされる。第2行はflip flop SCARの内容とALUA, ALUBの内容を加算して、その結果をBUSAに出力することを表す。この際、OVerflowがSCARにセットされる。

```
ρ+SC/*SRLS*/  ALUM[0:31]                BUSA[0:31] UN SC
+SCAR/*ADBC*/  ALUA[0:31] ALUB[0:31] BUSA[0:31] OV SCAR
```

3. マシーン記述テーブル(MDT)

我々は、HITAC 8350により、マイクロプログラム・ジェネレータをインプリメント中であるが、現在、MDSよりMDTへの変換プログラムの作成がほぼ終っている。そこで、MDTの仕様のあらましを次に述べる。

3-1. MDT全体の構造

MDSを処理した結果は、Fig.4のように、互いにポイントしあうテーブルの形となる。記述されたマシーンは、このように取扱い易い形でコンパイラとシミュレータに提供される。

これらのテーブルと実際のCPUとの対応は、各枠の右上に書いたように、TIM, PST, MT及びFDTが、制御部に対応し、MTよりのポインタ(破線)は、各μOPに対する制御信号に対応する。VTには、各変数のattributeに応じて、更に細かい記述をするためのTestable Condition Table(TCT), TRansfer Table(TRT), Counter Control Table(CCT), Flip Flop Set(FFS), Memory & Scratch pad memory Table(MST)が付属している。

OPTは、Arithmetic and Logical Unit(ALU)に対応し、各動作は、やはりMTにより制御される。

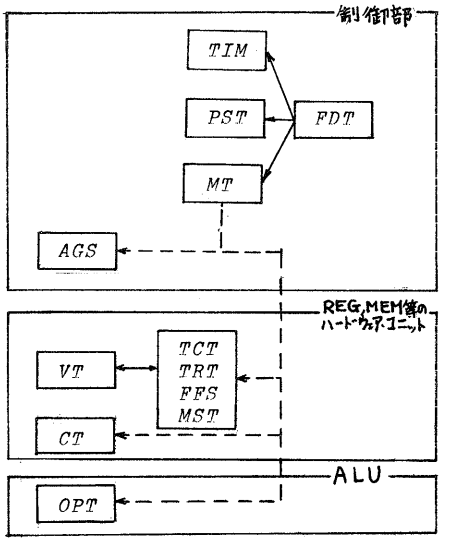

Fig.4 MDTの構造

3-2 制御部の構造

FDTが、制御部の中心で、TIM, PSTそしてMTへのポインタを持っている。各テーブルのエントリはMDSの記述に対応している。

AGSでは、各アドレスの生成法に応じてMDSの処理を行い、Fig.6のようなテーブルを構成する。

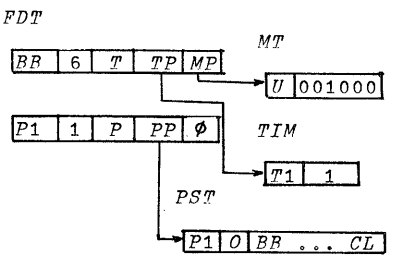

Fig.5 FDT, MT, TIM, PST

U	[0:11]	F	TS[1:6]	JA[0:5]	UNC

Fig.6 AGS

3-3 ハードウェア・ユニットの記述テーブル

register, terminal, counter, flip flop memoryそしてscratch pad memoryはアルゴリズム記述で最も頻繁に使用される部分であり、この記述テーブルは、一般のコンパイラで言えば変数テーブルに当るものである。

各変数に対し、identifier, attribute及びdimentionは共通にある。

U	R	32	OD2	→TCT	→TRT	ø	ø
BUSB	T	32	OD2	→TCT	→TRT	ø	ø
R	C	8	OD2	→TCT	→TRT	→CCT	ø
NZ	F	1	ø	→TCT	→FFS	ø	ø
MM	M	8	131	MAR	MDR	→MST	ø
SPM	S	32	128	SPAR	SPMR	→MST	→MST

Fig.7 VT

それ以外にregister, terminalについては. OperanD sideとTCTへのポインタ, TRTへのポインタがある。また. counterについてはcounter controlを記述するCCTへのポインタが必要である。flip flopについては. どのような値がセットできるかを記述するFFSへのポインタが, memory, scratch pad memoryについてはMAR, MDRと. それ以外のmemoryのread/writeのための記述テーブルであるMSTへのポインタがある。

CTには. 前述の如くHardware constantとFieldによるconstantがあるが. Fig.9に. 両方の例を挙げた。

(TCT)

U[8:11]	=0	ULU
R[0:7]	≠0	RNZ
NZ	=0	NZO

(TRT)

U [0:31]	BUSB[0;31]	U
BUSB[0:31]	ALUB[0:31]	NB
R [1:7]	SPAR[0:6]	R

(FFS)

BUSA[0:31]≠0	FIXCHK

(CCT)

2	V	R[0:3]	DECR

(MST)

-1	T2	T4	[RW,I](-1)	[CW,I](-1)
0	T1	T4	<SP>	<SP>
10X U0		11X U1	18X U2	ø

Fig.8 TCT, TRT, FFS, CCT, MST

(CT)

H	1	ø	32	ø	ALUB[0:31]	C1
F	0 FF	00...XXXXXXXX	32	BB[3:5]EX[0:4]	BUSB[0:31]	C

Fig.9. CT

3-4 論理演算装置(ALU)の記述テーブル

ALUを記述するOPTは. 殆どそのままの形でMDSよりMDTへ変換されるが. 予め決められたbinary, unaryの区別をテーブルに書込むことが必要である。

(OPT)

ρ+SC	SRLS	32	U	ALUM[0:31]	ø	BUSA[0:31]	U	SC
+SCAR	ADBC	32	B	ALUA[0:31]	ALUB[0:31]	BUSA[0:31]	0	SCAR

Fig.10 OPT

4. MDTを用いたμOの生成法

4-1 quadrupleについて

ADS記述は. 構文解析の結果. Fig.11のようなquadrupleのリストに変換される。

FORMAT	SEMANTICS
(BP , , ,BA)	μPの実行番地を保存しておいてmicro subroutineへbranch
(ORP, , ,)	micro subroutineよりBP-quadrupleの次へ戻る
(GLA, , ,)	instruction fetch routineへbranch
(# , k , CV ,)	counter CVの内容を#が▽か△かに応じてkだけdecrement或いはincrementする
(:= , LE ,OPND2,LPC)	OPND2の内容をLE中のLPC個の変数すべてに転送する
(BOP,OPND1,OPND2,RES)	ALUによりOPND1,OPND2に2項演算BOPを施してRESを決定する
(UOP, ,OPND2,RES)	ALUによりOPND2に単項演算UOPを施してRESを決定する
(B , , ,BA)	BA番地への、無条件のbranch
(BC , , C ,BA)	BA番地への条件(C)付きのbranch

Fig. 11 quadrupleのリスト

4-2 各quadrupleに対するμOの生成

(1) (BP, , ,BA)と(ORP, , ,)

AGSよりaddress stackを用いてmicro subroutineへのbranch, returnをするに必要なμOを検索する。

(2) (GLA, , ,)

AGSよりinstruction fetch routineへ戻るに必要なμOを検索する。

(3) (#, k, CV,)

カウンタCVをVTより検索, 対応するcounter control欄でk#と同じものを探し対応するμOを検索する。

(4) (:=, LE, OPND2, LPC)

assignment及び次のunary, binary operationのquadrupleに対しては、ハードウェア・ユニット間のpathを見出すことが必要となる。このため例えば、

(:=, R[0:7], U[24:37], 1)

に対しては、TRTを用いてFig. 12のようなtreeを構成してpathとそのpathの転送に必要なμOを見

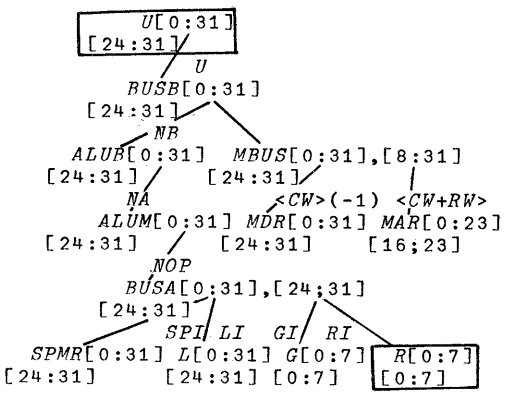

Fig. 12 treeの構成例

出す。

オペランドとして定数が使われた場合は、定数をrootとするtreeをCTを用いてFig. 13のように構成する。

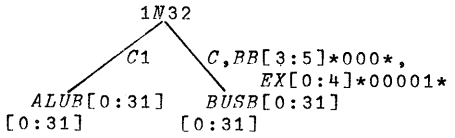

Fig 13 定数の生成例

(5) (UOP, ,OPND2, RES)

OPND2からのデータをALUの決められた入力端子(OPTにより記述)に送り、unary operation UOPを行うμOを生成する。

(6) (BOP,OPND1,OPND2, RES)

UOPと異なる点は、ALUへの2つの入力が同時に行われねばならないことと、各入力データが同一の入力端子を使用しないように、転送のための命令を補うなどの調整が必要なことである。

(7) (B, , , BA), (BC, , C, BA)

AGSより、それぞれ無条件、条件付き分岐に必要なμOを検索する。BC-quadrupleの場合はTCTよりテスト条件Cに対応するμOを検索して必要なμOを生成する。

5. おわりに

本システムの特徴は高級言語で記述されたμPのアルゴリズムよりhorizontal typeのμIを構成していく点にある。現在、HITAC 8350で、アセンブラによりインプリメント中であるが、今後、インプリメントによって、ここで述べた、μP記述言語(μPL)、特にマシーン記述部が現実のμP制御方式計算機を記述できるか、μPコンパイラはこれを用いてμPを生成できるか、またその効率はどうか、或いは、シミュレータは正しく動作するか、などの諸問題について、検討していく予定である。

謝辞

日頃、御指導戴く渡辺勝正助教授に感謝する。

参考文献

1) S.S.Husson "Microprogramming:Principles and Practices" PRENTICE-HALL,Englewood cliffs,New Jersey, 1970

2) R.H.ECKHOUSE,JR "MPL:A high level microprogramming language"SJCC 1971 pp.169-177

3) M.Hattori,M.Yano & K.Fujino"MPGS:A High-Level Language for Microprogram Generating System" proc.ACM 25th,1972, pp.572-581

4) E.W.Dubbs,L.Parsons & J.E.Petersen"A Microprogram Design System Translator-An Introduction" 6th Annual IEEE Computer Society International Conference,pp.95-98

5) R.F.Rosin"Contemporary Concepts of Microprogramming and Emulation" Computing Surveys,Vol.1,No.4,Oct.1969 pp.197-212

6) C.V.Ramamoorthy & M.Tsuchiya "A Study of User-micro-programmable Computers" SJCC 1970.pp.165-181

7) M.J.Flynn & M.D.Maclaren "Microprogramming revisited" Proceedings ACM,National Meeting, 1967,pp.457-464

8) A.B.Tucher & M.J.Flynn "Dynamic Microprogramming Processor Organization and Programming" Com.ACM,Apr. 1971,Vol.14,No4,pp.240-250

9) R.W.Cook & M.J.Fiynn "System Design of a Dynamic Microprocessor" IEEE Trans.on Computers,VolC-19,No.3, Mar 1970,pp.240-250

10) K.E.Iverson "A Programming Language" New York,Wiley,1962

11) D.F.Gorman and J.P.Anderson"A Logic Design Translator" Proceedings FJCC,1962,pp.251-261

12) H.P.Schlaeppi "A Formal Language for Describing Machine Logic,Timing and Seqencing(LOTIS)" IEEE Trans.on Comp. Vol.C-13,1964,pp.439-448

13) H.Schorr "Computer Aided Digital System Design and Analysis Using a Resister Transfer Language"IEEE Trans on comp.Vol.C-13,1964,pp.730-737

14) Y.Chu "An ALGOL-like Couputer Design Language" Com.ACM,Vol.8,No.10,Oct.1965,pp.607-615

15) J.R.Durey & D.L.Dietmeyer "A Digital System Design Language(DDL)" IEEE Trans.C.Vol.C-17,No.9,Sep.1968 pp.850-861

16) G.B.Gerace "Digital System Design Automation-A Method for Designing a Digital System as a Sequential Network System" IEEE Trans.C.Vol.C-17,No.11,Nov.1968 pp.1044-1061

17) T.D.Friedman & S.C.Yang "Methods Used in an Automatic Logic Design Generator(ALERT)" IEEE.Trans.on.Computers Vol.C-18,No.7,Jul.1969,pp.593-614

18) E.P.Stabler "System Description Language" IEEE Trans. on Comp.Vol.C-19,No.12,Dec.1970,pp.1160-1173

19) 岡田,元岡"論理設計言語" 信学誌 Dec. 1967 pp.2353~2360

20) 萩原.黒住"計算機設計言語"情報処理 Vol.12,No.2,Feb.1971, pp 93-102

21) Hewlett-Packard Company "Microprogramming Guide for Hewlett-Packard Model 2100 Computer" California,U.S.A, Nov.1971

22) 日立製作所"HITAC 8350 RCM機構仕様書"1972

23) 馬場,萩原"マイクロプログラム制御方式計算機の記述について"信学会電子計算機研究会資料 1973年7月

24) 馬場"マイクロプログラム・ジェネレータ"情報処理学会ダイナミックマイクロプログラミングシンポジウム資料 1973年7月

25) 馬場,萩原"マイクロプログラムジェネレータ" 情報処理学会第14回大会予稿(予定)