

論 理 設 計

自 動 化 シ ス テ ム

伊 藤 誠 和 田 義 弥 植 松 誠 也 河 野 豪 一

(山 梨 大 学 工 学 部)

はじめに 論理設計のCADを目的とした設計自動化システムの第一版が8割程度完成したので報告する。本システムはミニコン(64KB, フロッピーディスク)を用い、システムは①構造展開システム、②マクロ展開システム、③シミュレータ④トランスレータからなる(図1)。構造展開は階層的に記述されたシステムを、目的とするレベルまで、Top down方式で展開する。マクロ展開は任意入力数のゲートやレジスタ等を展開ライブラリに依り物理素子レベルに展開する。シミュレータは、構造記述とそれに対応する機能記述を得てシミュレーションする。機能記述はRTLレベルの言語である。構造記述展開レベルまでの遅れを考慮したシミュレーションが可能である。

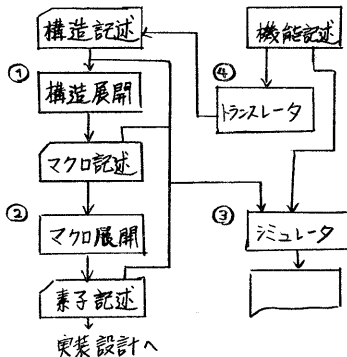


図1 論理設計自動化システム

トランスレータは制限された機能記述を構造記述に変換する。トランスレータのサブシステムとして、論理式処理があり、AND, OR, NOT多段論理の簡単化処理をする。

2. 構造展開システム

2.1 束信号 束信号は複数の論理信号を $A[2:0]$ のような配列型、 (A, B, C) のようなベクタ型で表現したものであり、 $A[2:0] = (A, B, C)$ は、 $A[2] = A$, $A[1] = B$, $A[0] = C$ と同等である。本システムではこの束信号を利用できる。

2.2 構造記述 本システムでは、記述対象素子を〈名前〉(〈束信号〉, …, 〈束信号〉) のように、名前と入出力信号名で規定する。素子としては構造展開される素子、マクロ展開される素子、基本素子があり前者には〈名前〉の前に“*”と“.”をつけて識別する。図2は構造記述の一般形式である。同じ信号を持つ素子の端子は結線されるものとする。構造記述の中の*〈構造素子名〉は別のところで構造記述される必要がある。

```

str <名前>(<信号リスト>)
net <内部信号リスト>
* <構造素子名>(<信号リスト>)
. <マクロ素子名>(<信号リスト>)
-----
eos
  
```

図2 構造記述

2.3 構造展開システム 構造展開システムは構造記述にしたがい構造を下位構造に展開する。

展開システムの入力には複数の構造記述が登録されたファイル、出力はマクロ素子ファイルである。展開は会話型とし、任意のレベルで展開を停止できるように、新しい構造名が出たとき、その展開の有無を問合せる。

図3に展開例を示すが、展開前の構造をコメントとして出力し、展開のレベルを段付して出力する。展開システムはPASCALで約1000行である。

```

STR HA(INPA, INPB, OUT, CARRY)
NET X;
AND2(INPA, INPB, CARRY)
OR2(INPA, INPB, X)
INV(CARRY, *CARRY)
AND2(X, *CARRY, OUT)
EOS

```

```

STR FA(INPA, INPB, CI, OUT, CO)
NET S, CY[1:2];
*HA(INPA, INPB, S, CY[1])
*HA(CI, S, OUT, CY[2])
OR2(CY[1], CY[2], CO)
EOS

```

```

STR ADD(A[0:3], B[0:3], CI, SUM[0:4])
NET CY[0:2];
*FA(A[0], B[0], CI, SUM[0], CY[0])
*FA(A[1], B[1], CY[0], SUM[1], CY[1])
*FA(A[2], B[2], CY[1], SUM[2], CY[2])
*FA(A[3], B[3], CY[2], SUM[3], SUM[4])
EOS

```

```

(* FA(A[0], B[0], CI, SUM[0], CY.001[0]) *)
(* HA(A[0], B[0], S.002, CY.002[1]) *)
AND2(A[0], B[0], CY.002[1]);
OR2(A[0], B[0], X.003);
INV(CY.002[1], *CY.002[1]);
AND2(X.003, *CY.002[1], S.002);
(* HA(CI, S.002, SUM[0], CY.002[2]) *)
AND2(CI, S.002, CY.002[2]);
OR2(CI, S.002, X.004);
INV(CY.002[2], *CY.002[2]);
AND2(X.004, *CY.002[2], SUM[0]);
OR2(CY.002[1], CY.002[2], CY.001[0]);
(* FA(A[1], B[1], CY.001[0], SUM[1], CY.001[1]) *)
(* HA(A[1], B[1], S.005, CY.005[1]) *)
AND2(A[1], B[1], CY.005[1]);
OR2(A[1], B[1], X.006);
INV(CY.005[1], *CY.005[1]);
AND2(X.006, *CY.005[1], S.005);
(* HA(CY.001[0], S.005, SUM[1], CY.005[2]) *)
AND2(CY.001[0], S.005, CY.005[2]);
OR2(CY.001[0], S.005, X.007);
INV(CY.005[2], *CY.005[2]);
AND2(X.007, *CY.005[2], SUM[1]);
OR2(CY.005[1], CY.005[2], CY.001[1]);
(* FA(A[2], B[2], CY.001[1], SUM[2], CY.001[2]) *)
(* HA(A[2], B[2], S.008, CY.008[1]) *)
AND2(A[2], B[2], CY.008[1]);
OR2(A[2], B[2], X.009);
INV(CY.008[1], *CY.008[1]);
AND2(X.009, *CY.008[1], S.008);
(* HA(CY.001[1], S.008, SUM[2], CY.008[2]) *)
AND2(CY.001[1], S.008, CY.008[2]);
OR2(CY.001[1], S.008, X.010);
INV(CY.008[2], *CY.008[2]);
AND2(X.010, *CY.008[2], SUM[2]);
OR2(CY.008[1], CY.008[2], CY.001[2]);
(* FA(A[3], B[3], CY.001[2], SUM[3], SUM[4]) *)
(* HA(A[3], B[3], S.011, CY.011[1]) *)
AND2(A[3], B[3], CY.011[1]);
OR2(A[3], B[3], X.012);
INV(CY.011[1], *CY.011[1]);
AND2(X.012, *CY.011[1], S.011);
(* HA(CY.001[2], S.011, SUM[3], CY.011[2]) *)
AND2(CY.001[2], S.011, CY.011[2]);
OR2(CY.001[2], S.011, X.013);
INV(CY.011[2], *CY.011[2]);
AND2(X.013, *CY.011[2], SUM[3]);
OR2(CY.011[1], CY.011[2], SUM[4]);

```

図3 4ビット加算器のゲートレベル展開

3 マクロ展開

3.1 マクロ素子 マクロ素子は任意長の入力を持つゲートやレジスタ、整数値や実数値をとる抵抗や遅延素子であり、通常論理設計はこのマクロ素子のレベルで行われる。マクロ展開はこのマクロ素子を入出力線数に制限のある物理的な素子に展開する(図4)。

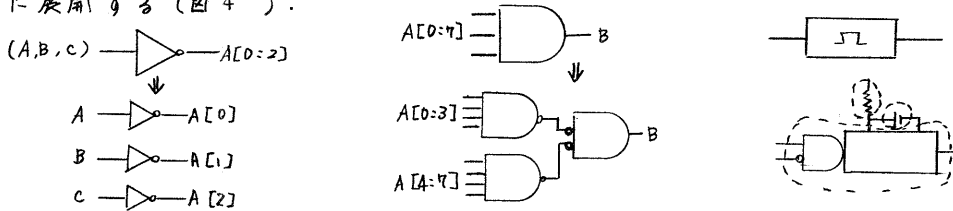


図4 マクロ展開例

マクロ展開の手法は、マクロ素子定義言語で記述されライブラリアンでライブラリとして保存される。ICファミリーや使用素子の变化はこのライブラリでかなり吸収できることが期待できる。

3.2 マクロ素子定義言語 マクロ素子定義言語はマクロ素子を展開する手法を示す。図5の例をとって、この言語を説明する。macro文はマクロ名の宣言と引数の型の宣言である。JからNは整数、OからRは実数、他は束信号である。束信号のサイズはこれが呼び出されたときに定まる。#Aは束信号のサイズを示す。最初のif文は、AとBのサイズが一致しないときエラーメッセージを出すことを示す。I=0は整数変数への代入文である。while文は次の論理式が満たされている間、eowまで実行を続けることを示す。"-----"でく

```
macro INV (A, B)
if #A < > #B then
error ("INV MISMATCH", #A, #B)
I = 0,
while I < #A do
I = I + 1
"INV(&A[I], &B[I])"
eow eom
```

図5 マクロ定義例

くられ下文が、マクロ展開の出力を指定する。"&"は仮引数A[I]に対応する束引数(束信号名)を示す。&A[I]は束信号AのI番目の信号となる。eomはマクロ定義の終了を示す。この例では用いられていないが&*はマクロ展開の通し番号の指示で、マクロ素子内の局所信号名として用いられる。

定義言語の中の式は四則演算以外に、2の対数、2の中乗が利用できる。

3.3 マクロ展開システム マクロ展開システムのサブシステムとして、ライブラリ保存システムがあり、これはマクロ定義をページに分割してディスクファイルに保存したり、消去、変更、追加を行う。マクロ展開システムはこのライブラリとマクロ素子ファイルを入力とし、ライブラリを解釈しながら各マクロ素子を展開する。主記憶はバッファ扱いとし、ライブラリは仮想記憶方式で参照するため、大きなライブラリも利用可能である。図6にラインプリンタインタフェス回路の展開例を示す。展開レベルはTTLバックでIC数換算で23個であった。これは、人手の場合と比べて一個少ない。使用言語はPASCALで、ライブラリ保守と展開システムの合計で2500行である。

4 シミュレータ

4.1 シミュレータの構造 シミュレータは部分的に展開された構造記述をシミュレーションする。展開された最下層の構造に対し、機能記述言語によるその

MACRO PULSE(A, O, B)

P=1.0E-10

R=0*0.14E-8/P

WHILE (R<1.4E3)+(R>4.0E4) DO

IF R<1.4E3 THEN P=P/10 EOI,

IF R>4.0E4 THEN P=P*10 EOI,

R=0*0.14E-8/P EOW

"SHOT(&A, GND, VCC, CXRX&*, CX&*, &B, ^&B);"

"RG(VCC, CXRX&*, &R);"

"CAP(CXRX&*, CX&*, &P);"

EOM

MACRO ANDS(A, B)

N=1 M=0

IF #A<=2 THEN "NAND2(&A[1:2], ^&B);", N=#A+1 EOI

IF #A=3 THEN "NAND3(&A[1:3], ^&B);", N=#A+1 EOI

IF #A=4 THEN "NAND4(&A[1:4], ^&B);", N=#A+1 EOI

WHILE A>N+3 DO

"NAND4(&A[N:N+3], W&MA&*);",

N=N+4, M=M+1 EOW

IF #A=N THEN "INV(&A[N], W&MA&*);", M=M+1 EOI

IF #A=N+1 THEN "NAND2(&A[N:N+1], W&MA&*);", M=M+1 EOI

IF #A=N+2 THEN "NAND3(&A[N:N+2], W&MA&*);", M=M+1 EOI

IF M=2 THEN "NOR2(WOA&*, W1A&*, &B);" EOI

IF M=3 THEN "NOR3(WOA&*, W1A&*, W2A&*, &B);" EOI

IF M=4 THEN "NOR4(WOA&*, W1A&*, W2A&*, W3A&*, &B);" EOI

IF M>4 THEN ERROR("ANDS TOO MANY INPUT", A) EOI

EOM

(4) 定 義 例

```
1: .BREC V('BDALC0:12], DA[C0:12]);
2: .BREC V(('BBS7, 'BSYNC, 'BDOUT, 'BDIN, 'BIAKI, 'BINIT), (IO, SYNC, DOUT, DIN, IAKI, INIT));
3: .TST(DA[C1:12], 2003, CALL);
4: DFF(CALL, SYNC, , , CALED, 'CALED);
5: DFF(DA1, SYNC, , , BUFFER, 'BUFFER);
6: AND2(DOUT, SYNC, WT);
7: AND2(DIN, SYNC, RD);
8: .DECOD2(BUFFER, CALED), WT, ( , , 'STWT, 'BFWT);
9: .DECOD2(BUFFER, CALED), RD, ( , , 'STRD, 'BFRD);
10: OR2(ERR, DONE, ROST);
11: .INTM(DA6, STWT, ROST, IAKI, DIN, INIT, IRQ, VEC, IAKO, INTENB);
12: .ORS((STWT, BFWT, STRD, BFRD, VEC), RPLY);
13: .DELAY((LPIMG, LPPALL, LPBUSY, LPOFL, 'LPTIME), 1000, LPS[1:5]);
14: .RCV(LPS[1:5], ('ERR, 'FALL, 'BUSY, 'OFL, TIME));
15: AND2('OFL, 'TIME, WAIT);
16: AND2('BUSY, WAIT, DONE);
```

(b) L P イ ン タ フ ェ ス 定 義 例 (部 分)

```
1: BREC V('BDALC0], DA[C0])
2: BREC V('BDALC1], DA[C1])
3: BREC V('BDALC2], DA[C2])
4: BREC V('BDALC3], DA[C3])
5: BREC V('BDALC4], DA[C4])
6: BREC V('BDALC5], DA[C5])
7: BREC V('BDALC6], DA[C6])
8: BREC V('BDALC7], DA[C7])
9: BREC V('BDALC8], DA[C8])
10: BREC V('BDALC9], DA[C9])
11: BREC V('BDALC10], DA[C10])
12: BREC V('BDALC11], DA[C11])
13: BREC V('BDALC12], DA[C12])
14: BREC V('BBS7, IO)
15: BREC V('BSYNC, SYNC)
16: BREC V('BDOUT, DOUT)
17: BREC V('BDIN, DIN)
18: BREC V('BIAKI, IAKI)
19: BREC V('BINIT, INIT)
20: NAND4('DA[C1], 'DA[C2], 'DA[C3], 'DA[C4], WOA001)
21: NAND4('DA[C5], 'DA[C6], 'DA[C7], 'DA[C8], W1A001)
22: NOR2(WOA001, W1A001, CALL)
23: DFF(CALL, SYNC, , , CALED, 'CALED)
24: DFF(DA1, SYNC, , , BUFFER, 'BUFFER)
25: AND2(DOUT, SYNC, WT)
26: AND2(DIN, SYNC, RD)
27: DECO2(BUFFER, CALED, 'WT, , , 'STWT, 'BFWT)
28: DECO2(BUFFER, CALED, 'RD, , , 'STRD, 'BFRD)
29: OR2(ERR, DONE, ROST)
30: DEF(DA6, STWT, , 'INIT, INTENB, 'INTENB)
31: AND2(INTENB, ROST, I1M002)
32: DFF(VCC, I1M002, 'INIT, VEC, I2M002, I2M002)
33: AND2(I3M002, IAKI, VEC)
34: DFF(IRQ, DIN, , 'INIT, I3M002, I3M002)
35: AND2(I2M002, I1M002, IRQ)
36: AND2(I3M002, IAKI, INTENB)
37: NOR4(STWT, BFWT, STRD, BFRD, WOA003)
38: INV(VEC, W1A003)
39: NAND2(WOA003, W1A003, RPLY)
40: RG(LPIMG, LPS[1], 100)
41: CAP(LPS[1], GND, 1.42E-8)
42: RG(LPPALL, LPS[2], 100)
43: CAP(LPS[2], GND, 1.42E-8)
44: RG(LPBUSY, LPS[3], 100)
45: CAP(LPS[3], GND, 1.42E-8)
46: RG(LPOFL, LPS[4], 100)
47: CAP(LPS[4], GND, 1.42E-8)
48: RG('LPTIME, LPS[5], 100)
49: CAP(LPS[5], GND, 1.42E-8)
50: RCV(LPS[1], 'ERR)
51: RCV(LPS[2], 'FALL)
52: RCV(LPS[3], 'BUSY)
53: RCV(LPS[4], 'OFL)
54: RCV(LPS[5], TIME)
55: AND2('OFL, 'TIME, WAIT)
56: AND2('BUSY, WAIT, DONE)
```

(c) マクロ展開結果

図6 マクロ定義と展開例

構造の動作記述が必要である。展開が標準的なゲートレベルになったとき機能記述は不要で、この場合ゲートレベルシミュレータとなる。シミュレーションは固定遅延時間とハイインピーダンス状態を考慮したイベント・ドリフ方式とした。

4.2 機能記述言語 この言語は、従来のレジスタ転送レベルの言語に属するが、構造記述への変換とシミュレーションの効率を考慮して、若干の修正をしている。

```
func <機能名> (<束信号リスト>)
  <宣言部>
  {
    / 条件 / { <実行文1>,
    ----- <実行文n> }
  }
eof
```

```
net I0 [7:0]: R, MSKF: R ;
    DA [7:0], M [9:0, 7:0]
state s1, s2, s3
clock clk 10, 30, 20
```

図7 機能記述の構造

図8 宣言部例

図7に機能記述の型を示す。fun文で機能名と外部信号を定義する。これは、対応する構造と同じ名前と信号を持つ必要がある。記述は<宣言部>と複数の<ブロック>から構成される。<宣言部>の例を図8に示す。束信号の場合サイズを指定する。メモリは2次元配列で表現する。各信号は記憶(レジスタ)型のため、":R"宣言をする。その他、同期回路の状態名をstate、クロックをclock文で記述する。

<ブロック>は必ず"/"で囲って実行条件を指定する。シミュレータは、各時刻で、条件を満たすブロックのみ実行する。条件は単信号、その立上り、下り、状態名等が指定できる。

実行文としては、レジスタ代入文で処理を記述するが、

```
(A [3:0], B) = (C, A [3:0])
```

のように、ベクタ型の中に配列型束信号を書くことができる。また、配列型信号のサイズが宣言した型と同じときは、A [] = B []のように省略記述できる。また、マルチプレクサの使用を考慮して、mpx代入文があり、

```
A [ ] = mpx C of / 0 / D [ ] & E [ ];
                / 1 / D [ ] / E [ ] eom
```

のような記述が可能である。

また、同期型制御を書くためにgoto文、同期または非同期の実行のためのif(式) xlen --- else --- 文、デコーダを意識した多分岐のためのcase<信号> of /値/ ---- ; /値/ ---- eoc 文等も用意されている。

演算としては、算術四則演算の他、束信号間のbit wise AND, OR, NOTと、束信号内のAND, OR, NOTが利用できる。

4.3 シミュレーションの手法 シミュレータは、展開された構造記述と、対応する機能記述を入力とする。機能記述はまず、内部形式に変換してライブラリ形式に保存する。内部形式は、信号名を番号に変換しグローバル信号はコモン領域、ローカル領域はスタック領域を静的に割当て使用する。束信号は4バイト(32ビット)の領域を割当る。式は逆ポーランドに変換し、その他のkey wordも数字に変換しておく。

シミュレーション手法はイベント・ドリブン型で、構造間の信号（グローバル信号）については、その信号を用いるすべての構造を結合しておき、その信号のイベント生起時に、接続する全構造の機能記述をシミュレーションする。ローカル信号のイベントは、その構造のみをシミュレーションする。特定の構造のシミュレーションは、各ブロックの条件を接続しておき、条件を満たさないブロックはシミュレーションの対象としない。

4.4 会話型シミュレーション・コマンド シミュレーションを会話型にするため、条件付のコマンドを指定できる。たとえば

/* A / ? B []

は A の変化時に乗信号 B の値を出力するコマンドである。このコマンドに対し、シミュレータはプリントモジュールをあたかも構造（素子）のように信号に結合する。その他

/ SYS / ! STOP

は SYS 信号の変化でシミュレーションの停止をさせたり、

/* RQ / ACK = 0

で RQ の立上りで ACK を 0 にするようなコマンドも指定できる。動作時間の遅延は、

A = B (* 30 *)

のように指定する。（* --- *）はシミュレータ以外ではコメント扱いである。

本シミュレータは現在作成中であり、まだ完成していないが、解釈ルーチン組込型の素子記述レベルのシミュレータを試作したので、その例を図 9 に示す。このシミュレータは PASCAL で 800 行である。

5. トランスレータ

トランスレータは機能記述を構造記述に変換する。いわゆる、論理合成処理を行う。ただし、機能記述全体の処理は無理があるので、

- 1) 2次元配列処理
- 2) 一致以外の関係演算
- 3) 算術演算

は除外する。上記三種は、論理設計者がその構造をもっと詳細に記述することが必要である。トランスレータは、例えばレジスタ代入文を、論理演算マクロとレジスタマクロに展開する（図 10）

<pre> /S1 / A [] = A [] / B [] ↓ . OR ((A [], B [], @G00 []) . REG (@G00 [], S1, CLK / A [] </pre>	<pre> /S2 / case c of /O / goto S3 ; /1 / goto S4 eoc ↓ \$ ST (S2, @C01, S3) \$ ST (S2, @C02, S4) \$ DECODE (C, (@C01, @C02)) </pre>
--	---

図 10 トランスレート例

\$ ST は状態推移マクロで、特殊マクロである。これは、式処理と共にサブシステムでマクロに展開する。PLA、各状態に FF を割当る等のサブシステムは作成済である。式処理サブシステムは、多段の AND, OR, NOT 論理式を展開し、共通項の抽出や、冗長項の削除を行う。このサブシステムも完成しているが、時間が

EXTERNAL EVENT

NO. 1	TIME= 1	SIGNAL= A[0]	VALUE= 1
NO. 2	TIME= 1	SIGNAL= A[1]	VALUE= 0
NO. 3	TIME= 1	SIGNAL= A[2]	VALUE= 0
NO. 4	TIME= 1	SIGNAL= A[3]	VALUE= 1
NO. 5	TIME= 1	SIGNAL= B[0]	VALUE= 1
NO. 6	TIME= 1	SIGNAL= B[1]	VALUE= 1
NO. 7	TIME= 1	SIGNAL= B[2]	VALUE= 1
NO. 8	TIME= 1	SIGNAL= B[3]	VALUE= 0
NO. 9	TIME= 1	SIGNAL= CI	VALUE= 1
NO.10	TIME= 9	SIGNAL= B[3]	VALUE= 1

```

A A A B B B B C S S S S S
[ [ [ [ [ [ [ [ I U U U U U
3 2 1 0 3 2 1 0 M M M M M
] ] ] ] ] ] ] ] [ [ [ [ [
                                4 3 2 1 0
                                ] ] ] ] ]

```

TIME

1	1 0 0 1 0 1 1 1 1 1 * * * * *
2	1 0 0 1 0 1 1 1 1 1 * * * * *
3	1 0 0 1 0 1 1 1 1 1 * * * * *
4	1 0 0 1 0 1 1 1 1 1 * * * * *
5	1 0 0 1 0 1 1 1 1 1 * * * * *
6	1 0 0 1 0 1 1 1 1 1 * * * * *
7	1 0 0 1 0 1 1 1 1 1 * * * 0 1
8	1 0 0 1 0 1 1 1 1 1 * * * 0 1
9	1 0 0 1 1 1 1 1 1 1 * * 0 0 1
10	1 0 0 1 1 1 1 1 1 1 1 * 0 0 1
11	1 0 0 1 1 1 1 1 1 1 1 0 0 0 1
12	1 0 0 1 1 1 1 1 1 1 1 0 0 0 1
13	1 0 0 1 1 1 1 1 1 1 1 0 0 0 1
14	1 0 0 1 1 1 1 1 1 1 1 0 0 0 1
15	1 0 0 1 1 1 1 1 1 1 1 1 0 0 1
16	1 0 0 1 1 1 1 1 1 1 1 1 0 0 1
17	1 0 0 1 1 1 1 1 1 1 1 1 0 0 1
18	1 0 0 1 1 1 1 1 1 1 1 1 0 0 1
19	1 0 0 1 1 1 1 1 1 1 1 1 0 0 1
20	1 0 0 1 1 1 1 1 1 1 1 1 0 0 1

図 9 図 3 (b) の素子 フ ァ イ の
シミュレーション例

がかる割に効果が少ない。これは、変数の数が多い割に共通部分が多いという、入力の性質に依る。図11にトランスレータの例を示す。

トランスレータは、現在改版中であるが、旧版のシステムのサイズはサブシステムを含めて5000行である。

6. おわりに

構造記述、マクロ記述の入力は図形入力とすることも考えている。また、展開結果の出力(回路図)は是非とも必要である。機能レベルのシミュレータは、現在開発中であるが、LSIを多用した基板のシミュレーションを可能にできるようにしたい。条件付コマンドは、会話型シミュレーションを効果的にするものと期待している。

トランスレータは、試作の域を出ないかも知れない。また、論理設計の未経験者が、機能言語で回路を合成するというより、類似回路の設計経験者が簡単に設計ができる点に意義を見つけるべきかも知れない。

参 考 文 献 (代表的な文献のみとする)

1. van Cleemput : SDL ; 14xR DAC (1977)
2. Mc Williams : SCALD ; 15xR DAC (1978)
3. 佐々木 他 : MEX ; 日経エレクトロ 1980. 12. 8
4. 上原 他 : DDL ; 日経エレクトロ 1979. 11. 12
5. 伊藤 他 : 会話型設計援助システム ; 電子装置設計技術 1979. 9
6. 伊藤 他 : マクロ転写システム ; 電子装置設計技術 1980. 6

```
FUNC IOMDL(D17:01,Z,CM,DT,'RD','M1','IORQ,CLK,RESET,IEI,IEO,INT,EX17:01,
'STRB,RDY)
```

```
NET
```

```
IOF:R,MSKF:R,IO17:01:R,MSK17:01:R,VCT17:01:R,XR17:01:R,
MD11:01:R,IE:R,AD:R,HL:R,RTIF:R;
```

```
(* RESET , COMMAND *)
① / RESET / { IOF=0; MSKF=0; IO17=0XFF; RTIF=0 }
② / CM / { IF (IOF) { IO17=D17; IOF=0 };
③ IF (MSKF) { MSK17=D17; MSKF=0 };
④ IF (^D101) { VCT17=D17 };
⑤ IF (D13:01==0XF) { MD17=D17:61;
⑥ IF (D17:61==3) IOF=1 };
⑦ IF (D13:01==0X7) { (IE,AD,HL,MSKF)=D17:41 }
}
(* DATA ACCESS FOR MODE 0,1 ONLY *)
⑧ / DT / { IF (^RD & (MD17==1)) { D17=XR17; RDY=0 };
⑨ IF ('RD & (MD17==0)) { XR17=D17; RDY=1 }
}
(* DATA HAND SHAKE FOR MODE 0,1 ONLY *)
⑩ / ^STRB / { IF (MD17==1) { XR17=EX17; RDY=1; INT=1 };
⑪ IF (MD17==0) RDY=0
}
EOF
```

```
STR IOMDL(D17:01,CM,DT,'RD','M1','IORQ,CLK,RESET,IEI,IEO,INT,EX17:01,
'STRB,RDY)
```

```
① $EXP(IOF:=0,RESET)
$EXP(MSKF:=0,RESET)
.REG(1,1,1,1,1,1,1,1),RESET,RESET/IO17:01)
$EXP(RTIF:=0,RESET)
② $EXP(@E00=IOF,CM)
.REG(D17:01,@E00,@E00/IO17:01)
$EXP(IOF:=0,@E00)
③ $EXP(@E01=MSKF,CM)
.REG(D17:01,@E01,@E01/MSK17:01)
$EXP(MSKF:=0,@E01)
④ $EXP(@E02=^D101,CM)
.REG(D17:01,@E02,@E02/VCT17:01)
⑤ $EXP(@E03=<D13:01=15>,CM)
.REG(D17:61,@E03,@E03/MD17:01)
⑥ $EXP(@E031=<D17:61=3>,@E03)
$EXP(IOF:=1,@E031)
⑦ $EXP(@E04=<D13:01=7>,@E03)
.REG(D17:41,@E04,@E04/(IE,AD,HL,MSKF))
⑧ $EXP(@E05=^RD & <MD11:01=1>,DT)
.GATTS(XR17:01,@E05/D17:01)
$EXP(RDY=0,@E05)
⑨ $EXP(@E06='RD & <MD11:01=0>,DT)
.REG(D17:01,@E06,@E06/XR17:01)
$EXP(RDY=1,@E06)
⑩ $EXP(@E07=<MD11:01=1>,'STRB)
.REG(EX17:01,@E07,@E07/XR17:01)
$EXP(RDY=1,@E07)
$EXP(INT=1,@E07)
⑪ $EXP(@E08=<MD11:01=0>,'STRB)
$EXP(RDY=0,@E08)
EOS
```