論理設計自動化システム

伊藤誠，河野豪之，牧田敏彦，藤沼良一
（山梨大学工学部）

はじめに

論理設計の自動化を目的とした記述言語とそのトランスレータがほぼ完成したので報告する．このシステムは機能レベルから素子レベルまでの記述を可能とし，トランスレータ出力も主要部分はマクロ素子レベルとなっている．このため，マクロ展開ライブラリを変更するのみで，トランスレータ出力を種々のICファミリを用いて実装できる．

1．設計自動化システムの構成

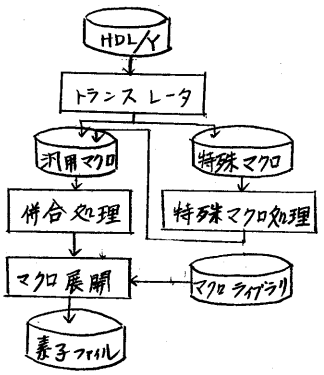


図1 設計自動化システム

本設計自動化システムの全体を図1[1]に示す．HDL/Y（Hard were Description Langage/Yamanashi）で記述された設計対象は，トランスレータにより，特殊および汎用マクロ記述に変換される．特殊マクロ処理は，論理式の簡単化，制御論理の合成を行い，汎用マクロを生成する．汎用マクロは併合処理をされた後，マクロライブラリの記述にしたがい素子ファイルに変換される．以後は実装設計の段階である．

2．マクロ素子

HDL/Yの説明の前に，トランスレータの出力となるマクロ素子を説明する．マクロ素子には，特殊マクロ，汎用マクロ，素子の3レベルの記述がある．一般形は，

<素子記号><素子名>（<入力信号列>/<出力信号列>）

である．汎用マクロは，SCALD[2]のマクロ素子と同等の概念である．たとえば，

・REG（D[15:0]，CLK，CLR/R[15:0]）

は16ビットのレジスタ（CLKでトリガ，CLRでクリア）を記述したものである．これは，たとえば，

R273（D007，----，D000，CLK，CLR/R007，-----，R000）
R273（D015，----，D008，CLK，CLR/R015，-----，R008）

の二つの素子記述にマクロ展開される．R273は8ビットレジスタMSI（SN*273）への実装を前提としている．マクロ素子記述は別に定義されているマクロ展開ライブラリを用いて展開される．

特殊マクロは，展開ライブラリでは記述できない特殊な機能の記述である．たとえば，

$ST（si，cf，sk），$OT（si，ck，oe）

は，状態siから条件cfで状態skに推移し（STマクロ），状態siで条件ckのとき，出力信号oeを生成する（OTマクロ）ことを示す．HDL/Yトランスレータは，HDL/Yの記述から直接制御回路を合成しないで，この$STと$OTマクロを発生する．このマクロから，結線論理を合成したり，PLAのROMパタンを合成

するのは，特殊マクロ処理の仕事となる．その他の特殊マクロとして，

$ EXP（<信号名>{:=}<論理式>,<実行条件>,<クロック名>）

がある．これは<論理式>の値を信号名として出力する組合せ回路を示すが，一般には，簡単化処理が必要なので，これをマクロ素子に変換するのも特殊マクロ処理とした．信号名の次が":="のときは，論理式の値を<クロック>に同期して記憶すべきことを示す．特殊マクロ処理は，HDL/Yトランスレータの第2パスと考えることもできよう．

3. HDL/Y記述の階層構造

HDL/Yは設計対象をunit単位に分割し記述する．unitは宣言部および，一つの主シーケンスと複数のサブシーケンスから構成される．シーケンスは，複数のブロックから記述される．ブロックにはレジスタ転送レベルの記述とブロック間の推移記述が含まれ，これらは同時に実行される．

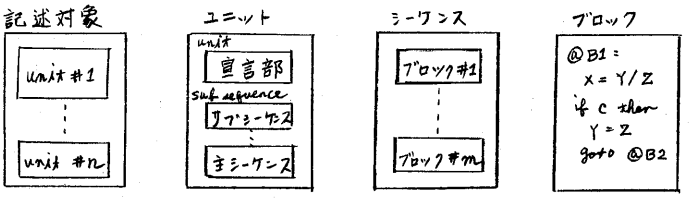


図2 HDL/Yの階層記述

この他にオペレーション，ファンクション，インラインマクロ記述機能が用意される．

```
operate XTOY
  BUS = X  ;
  Y := BUS   eoo
  ----
@B1: !XTOY
  ---      eob
```

図3 オペレーション

```
reg c[0:5]
  ----
@: .COUNT[CNT,CLR/C]
  ----
@B2 : CNT = 1
```

図4 インラインマクロ

```
function COUNT(A)
  count(x , / $A)  eof
  ----
@B3 : COUNT(C)
  -----
@B5 : COUNT(C)
```

図5 ファンクション

図3はオペレーションの使用例で，XをBUS経由でYレジに記憶するオペレーションをXTOYとして定義している．ブロックB1の!XTOYはオペレーションの実行要求で，オペレーションの内容をブロックの中にそのまま記述した場合と同等の効果を持つ．同一のレジスタ転送が複数のブロックで記述される場合は，それをオペレーション定義しておく方が記述も簡明であるし，後述するマクロ素子の併合処理も簡単となる．

図4は，マクロ素子をHDL/Yの中に記述した例である．これは名前なしブロックにのみ記述可能である．（名前なしブロック内の記述はその実行に条件がつけられない）．トランスレータはこのマクロ素子には何も処理を加えないでそのまま，6ビットのカウンタマクロとして，出力マクロファイルに追加するだけである．しかし，ブロックB2に制御が移るとCNTにパルスが生成されるため，COUNTマクロのレジスタCの値が一つ増加する（.COUNTマクロは，カウンタMSIに対応する素子に展開されるものとする）．インラインマクロを強化してパラメータ付のマクロ素子定義を可能にしたのがファンクションである（図5）．functionの定義そのものは，マクロ素子の生成形式を定義するだけである．ブロックB3とB4

で，function の定義にしたがい

.COUNT（B3，/C） と .COUNT（B4，/C）

の二つのマクロ素子が生成され，これらは併合処理により，

.OR((B3, B4),@G00), .COUNT(@G00, /C)

に変換される．ORマクロは B3 と B4 の論理ORを @G00 とするゲートに，後者は@G00により，計数を行うカウンタに対応する．function 機能はまだトランスレータの中には組込れていない．

4. HDL/Y の記述例とその特徴

HDL/Y の文法はすべて図6のような構文図で定義されているが，簡単のため，ここでは主として例を用いながら紹介する．

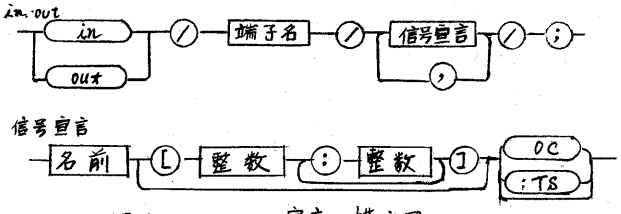


図6 In out 宣言の構文図

(a) 宣言部

```
unit EX (CLK)
in / BUS / D[15:0].RD.WAT ;
out / IO /BFO[7:0]:OC, STRB:TS;
reg BUF[7:0], ST ;
net DT[7:0] ;
equ UD=D[15 8], LD=D[7:0] ;
```

図7 宣言部の例

図7は，ユニット EX（同期クロック CLK）が利用する信号の定義例である．in out は外部との入出力信号名とサイズ（信号線の数）の定義である．信号名は，シーケンス部では端子名をつけて，DBUS.RDのように参照するから，端子名が一意なら同じ信号名があっても識別できる．信号名の次の〝:OC〟は open collector 出力，〝:TS〟は tri state 出力を示す．equ は名前の置換とサブレンジの定義である．複数の信号から構成される信号をここでは束信号と呼ぶが，束信号の一部を参照するときはあらかじめこの宣言をしておく必要がある．reg は記憶型の net はゲート出力型の内部信号名の定義である．内部信号名は，マクロ素子記述内の信号名としてトランスレータから出力されるときは頭にユニット名がついて，他のユニットの信号名と区別をつけられる．ユニット名，端子名は3文字，信号名は6文字以内とする．宣言部には他に定数を名前で参照するための const 文がある．

(b) ブロックとシーケンス

```
reg A[8:0], B[8:0]
net X[7:0], Y[7:0]
sequence
@L1: A:=X ; B:= X/Y ; goto @L2 eob
@L2: A:=X & Y ; goto @L3 eob
@L3: --- [ ] eob eos
```

図8 シーケンス例

図8は L1, L2, L3 の三つのブロックを持つシーケンスの例である．L1, L2, L3 をブロック名と呼ぶが，これは順序機械の状態に相当する．時刻 t で X の値が A に，X と Y のビット毎の OR が B に記憶されると，次の時刻 t+1 ではブロック L2 がアクティブとなり，X と Y の AND が A に記憶される．ユニットの中で，1ブロックのみがアクティブとなることができる．ただし，名前なしブロック（〝@:〟で始まるブロック）は常にアクティブであるが，この中ではレジスタ代入処理（〝:=〟を持つ代入文）は記述できない．

図9はサブシーケンスとその呼び出し例を示す．サブシーケンスは一般に，リ

```
@B1 : call SUB(0)
@B5 : call SUB(1)
@B7 :
```

```
subsequence SUB(FLAG);
@S1 :
@S5 : return(0)@B7 ;
      return(1)@B8 eob
```

図9　サブシーケンス

ンクレジスタ（図9の場合FLAG）を持つ．サブシーケンスを呼び出すときは，リンクレジスタに値をセットしておく．サブルーチンからの戻りは，リンクレジスタの値により定まる．図9の場合，B1からFLAG＝0とした後サブシーケンスSUBのS1がアクティブとなり，S5に達するとブロックB7に戻る．

(c) 論理式と代入文

HDL／Yでは，束信号間の／(OR)，%(EXOR)，&(AND)および束信号の^(NOT)演算が可能である．図10の(1)式はXとAをビット毎にANDし，さらにBとのORをとった後，Aに記憶する（レヂスタ記憶）することを示す．(2)式は，Aのすべてのビットを0Rして ZF(Zero Flag) に記憶することを示す．ZFは1ビットの信号となる．(3)式は一致演算の例で，Aの値が3(ADRS)かつZF=1のとき CALは1となる．CALの値は記憶されないから，AおよびZFの変化と同時に変化する．

```
reg A[7:0], B[7:0],ZF
net X[7:0], CAL
const ADRS = 3
  A := B/ X&A ---------- (1)
  ZF := /A    ---------- (2)
  CAL = <A=ADRS>&ZF ---(3)
```

図10　代入文

代入文の左辺と右辺のビット数は同じでなければならない．また，式中の束信号線に範囲をつけることはできないから，この場合equ宣言をしておく必要がある．例を図12に示す．

```
equ LA=A[3:0],
    UA=A[7:4]
@B: UA:=LA
    LA:=UA
```

図12 部分代入文

```
reg C[2:1]
A:=mpx C of
/1/ A & X ;
/2/ A / X ;
/3/ A% X ;
/4/ ^A eom
```

図13 mpx文

図13はCの値にしたがい，Aに代入される論理式を定める文で，トランスレータはMPX（マルチプレクサ）マクロを生成する．HDL／Yではカウンタ，アダ等の組込演算は許していない．これは，カウンタにも，同期／非同期，2進／10進，ロード・クリア機能等多種のMSI ICが存在する．これらを言語レベルで指定するのは，言語が物理的条件に左右されることになり，独立性が悪い．HDL／Yでは，これらはすべて，インラインマクロ，または，ファンクションとして記述することになる．

(d) 制御文と条件式実行

HDL／Yではブロック間の推移をgoto @<ブロック>名で示す．'@'はブロック名を他の名前と区別するために用いている．('@'はこの他にも，トランスレータが自動生成する名前にも付加される)．サブシーケンスの呼び出し戻りには，call，return を用いる（図9参照）この他に，HDL／Yでは条件付実行の記述のために，if，after（指定されたクロック後に実行することを指示する)文がある．各文はeoi (end of if)，eoa (end of after)等で区切られるから，

```
if CAL then A := X
   else B := X  eoi
```

(a) if文

```
after 3 goto @B4 eoa
```

(b) after

```
case C of
/0/ goto @B1
/1/ goto @B2
/2/ goto @B3
/3/ goto @B(4) eoc
```

(c) case文

図14　if，after，case文

複数の処理を記述できるが，同一の文の中に複数個の無条件 goto や call を記述することは，文法的に禁止している．

5. トランスレータ

ここでは，HDL/Y 記述とそれから生成されるマクロ素子について説明をし，あわせてトランスレータの機能を説明する．

HDL/Y	@B2 : if X&Y goto @B3	@B1 : A := A/X	@B4 : case C of /0/ goto @B5 /1/ goto @B6 eoc
マクロ	$EXP (@C00 = X&Y); $ST (B2, @C00, B3);	.$OT(B1, , @E00) . OR((A[7:0], X[7:0])/@G00[7:0]); . REG(@G00[7:0], @E00, CLK/A[7:0]);	$ST(B4, @C01, B5); $ST(B4, @C02, B6); $CASE(C/(@C01, @C02));
	(a) goto 文	(b) 代入文	(c) case 文

図 15 HDL/Y と生成マクロ

図 15 (a) は条件付 goto 文をトランスレートした例で，条件論理はそのまま $EXP マクロとなり，その出力が @ C00 (`C'は条件論理を示し，`00'はシステムが生成する番号である) である．goto 文に対し，$ST (State transition) マクロを生成する．(b) は代入文の処理例で，まず，$OT マクロで実行条件 @E00 を生成する．この場合，ブロック B1 で無条件に E00 が 1 となる．束信号間の式処理は簡単化をしないので，直ちに OR マクロを生成し，この出力を REG マクロ (CLK は同期クロック) を生成する．(c) は case 文の処理例で最後に $CASE マクロを生成している．この $CASE マクロは，MSI IC レベルではデコーダに変換される．.OR, $CASE マクロは入出力信号の数が変化するため，それらを `( )' で囲んで一つのパラメータ扱いとしている．マクロ展開時には 2 入力 OR，3-8 デコーダ等固定長素子になる．オペレーションの変換例を図 16 に示す．オペレート定義中の代入文は XTOY を実行条件としたマクロに変換される．一方オペレーションの実行要求は，XTOY を出力とする $OT マクロを生成する．なお，.GATE マクロは，XTOY が 1 のとき X を Y に出力することを示す．BUS が TS (Tri State) や OC (Open Collecter) 指定をもつときは，.GATE の代りに .GTTS, .GTOC マクロが生成される．

```
operate  XTOY
 BUS=X ;          .GATE(X[7:0],XTOY/BUS[7:0]);
 Y:=BUS  e00      .REG(BUS[7:0],XTOY,CLK/Y[7:0]);
 ------           ---------
@B1: !XTOY ;      .OT(B1,  ,XTOY);
 -------
```

図 16 オペレーションの変換

egu 宣言を持つ信号名は，マクロ生成時には egu 宣言時の右辺の名前に置換される．図 12 の場合は図 17 のマクロが生成される．

```
$OT(B,  ,$E01)
.REG(A[3:0], $E01, CLK/A[7:4])
.REG(A[7:4], $E01, CLK/A[3:0])
```

図 17 egu 宣言を持つ信号の変換

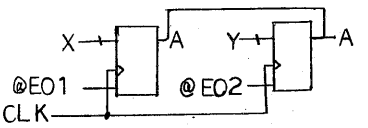
最後にトランスレータの検査能力について述べる．HDL/Y の文法的な検査を行っているが，意味的な検査は全く行っていない．たとえば，推移するブロックの有無，同時に複数のブロックへの推移等も検査していない．ただし，代入文の両辺の束信号のサイズ，$EXP マクロ中の論理式の構文検査等は行っている．

6. マクロ素子の併合

トランスレータからの出力マクロをそのまま実装設計入力とすることはできない．図18の場合，このままではレジスタCが二つ生成され，しかも，その出力間が結合されてしまう．この場合，二つのマクロを図19のように併合する必要がある．

```
@L1 ; A := X ;      .REG(X[7:0],@E01,CLK/A[7:0])
- - - - -
@L2 ; A := Y ;      .REG(Y[7:0],@E02,CLK/A[7:0])
```

(a) HDL/Y記述　　(b) 生成マクロ

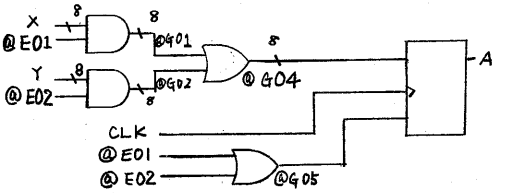


(c) 対応回路

図18 マクロ生成例

```
.GATE(X[7:0],@E01/@G01[7:0])
.GATE(Y[7:0],@E02/@G03[7:0])
.OR((@G02[7:0],@G03[7:0]/@G04[7:0])
.OR((@E01,@E02)/@G05)
.REG(@G04[7:0],@G05,CLK/A[7:0])
```

(a) 併合結果



(b) 対応回路

図19 併合例

一般に，同じ信号を出力に持つマクロ素子は併合の対象となる．併合処理は，一般的には各入力信号をORすれば良いが，すべてのマクロ素子に同じ処理を適用することはできない．この併合処理は次の課題である．

7. 特殊マクロ処理

特殊マクロ処理の中には，$EXPマクロで記述される論理式の簡単化処理が含まれるが，この処理手法について述べる．簡単化の方針は，①原則として式の変形はしない．ただし，指定があればAND-OR-NOTの標準形（最速論理となる）に展開する．②冗長な変数や，項は消去する．ただし，複数の項の組合せの変更による簡単化はしない．③共通な項があれば，それを利用する．である．ここで，項とは，$x1 \cdot x2 \cdot \overline{x3}$ のような変数のAND演算を言う．この方針による簡単化例を図20に示す．

$x1 \cdot \overline{x2} + x2$ $\Rightarrow x1 + x2$ (a) 変数消去

$x1 \cdot \overline{x2} + \overline{x2} \cdot x3 + x1 \cdot \overline{x3}$ $\Rightarrow x1 \cdot \overline{x2} + x1 \cdot \overline{x3}$ (b) 冗長項の消去

$x1 \cdot x2 \cdot x3 + x4 \cdot x2 \cdot x3$
$\Rightarrow x2 \cdot x3 \cdot (x1 + x4)$ (c) 共通項の利用

図20 式の簡単化

図21に論理式処理の流れを示す．各項は，PASCALの集合（set）で表現し，式は集合変数の組として表現する．PASCALでは集合は，1語の中のビットの0,1で表現されるため，集合間のAND，OR，集合間の要素の有無等の処理は高速である．しかしながら，現在，冗長項・変数の消去は＃演

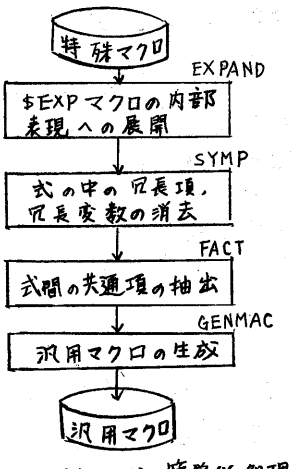


図21 論理式の簡単化処理

算（排除演算）手法[3]を用い，共通項の抽出には，集合演算[4]を用いているが，処理に要する時間は少なくはない．図22に簡単化の例と処理時間を示す．

CMD1 = A·(B+C·D+C̄+D̄) +E
CMD2 = A·(B·(Ā·(B̄·C̄)))
CMD3 = Ā·B̄·(Ā+B̄+C̄)·A·(B+C̄) + B̄
CMD4 = (Ā+B)·(B̄+C)+(B̄+C̄)
CMD5 = (A+B+C̄+A·D)·(A·C̄·(B̄+D̄))
CMD6 = (A·B̄+B·C)·(Ā·B̄+A·C+B·C̄)

(a)入力例（実際にはExpマクロ形式の入力）

CMD4 = Ā·B̄
CMD2 = A·B̄
CMD3 = B̄
POA1 = A·C̄·D̄
POA2 = C̄·CMD2
POA3 = C·D̄·CMD4
CMD1 = A+E
CMD5 = POA1+POA2+POA3
CMD6 = C̄+CMD4

(b)簡単化した結果
（実際には .AND, .OR マクロ出力）

展開時間	10秒
冗長の消去	5秒
共通項抽出	1秒
マクロ生成	1秒
総計	20秒

（ファイル入出力を含む）
(C)処理時間
（CMD1～CMD6）

図22 簡単化例

式展開ルーチンは，/A[0:3]（A00 + A01 + A02 + A03 に等しい）や，排他論理和，一致演算（<A[0:3]= 10 >=A00·Ā01·A02·Ā03）の展開，論理式の否定，論理式の括弧づけを尊重した展開等が可能である．

$STマクロと$OTマクロより制御回路を合成する問題は，順序回路の合成問題となり，状態に対する符号割当問題として種々の方面から研究されている．ここでは，① PLAを用いて合成するためPLAへの書込パターンを生成する．②結線論理を生成する．の二方面から開発を進めている．①のPLAパターンの生成システムはすでに試作が完了しているが[5]，field Programmable なSPLAの入手が困難なのでまだ実用化には達していない．②の方法は一状態一FF割当法を用いたマクロ素子生成プログラムを作成する予定である．

あとがき

使用計算機は，LSI-11（メモリ56kB）でフロッピーファイルを利用している．言語はOMSI PASCALコンパイラで，現在の総プログラムは5000行程度である．（LSI-11の命令実行速度は，転送約5μs，整数乗算約30μs）．モニタはRT-11を使用している．

今後，マクロの併合処理，HDL/Y記述から素子レベルまでをカバーする論理シミュレータ，自動生成回路の図面作成（ブロック図，展開図）等のシステムを開発する予定である．なお，トランスレータのコーディングは河野が，式処理部のコーディングは牧田が担当した．

参考文献

(1) 伊藤，河野：会話型設計援助システム
電子装置設計技術 2-3（79, 9, 18）

(2) McWilliams T.M：SCALD：Structures Computer Aided Logic Design
PP 271～277 15th DAC（1978）

(3) （例えば）アルーア：“ディジタル計算機の自動設計”
PP 47～53 産業図書

(4) 伊藤，稲垣，福村：“NAND論理を用いた論理回路網の合成”
電子通信学会 52-C（昭44年）