論理装置設計自動化に関する研究

水原 登　伊藤 誠（山梨大学 工学部）

1. はじめに

近年，計算機が容易に利用でき，その汎用性は増大しているが，研究室等では特殊のインタ・フェース，あるいは簡単な処理装置の設計・製作が必要になることが少なくない．この論理設計・製作をすべて人手で行うと，単純なるミス（ICのピンの誤り等）や，設計変更した場合の書類管理が十分でなく，無用の時間を費すことが多い．

そこで，我々は実験規模（IC換算して50～100で複数の基盤にまたがらない）程度の設計の自動化システムを生成し，その基本部分が完成し，インタ・フェース製作等に実際に適用して効果をあげたので紹介する．本システムは，適当な言語で装置を記述すると，最終的には，使用するIC名と，ICのピン間の接続関係を出力するもので，出力結果に基き，ピン間の接続（電源ラインも含む）をすれば装置が製作できる．プログラムは主としてFORTRANで書かれ，ユーザ領域56KBのFACOM270／30（補助記憶装置としてドラム，マガジン（小型MT）を使用）で実行される．

現在までに，ALERT，LDT，CDLさらに，我国でも，LDS，TBM，ADL等の設計言語が提案され，システムが試作されているが，大型の処理装置が必要であったり，SSI規模のICのみを利用していたり，IC間のピン接続という実装情報の生成までの一貫した処理がなされていない，等のため，我々の目的には合致していない．本システムは，上記高級言語の部分的実現というより，むしろ，文献*の設計システムの部分的拡張とも考えられる．

我々のシステムは，言語の記述能力自体は，上述の設計言語に比すと貧弱な点はあるが，実験規模（パッケージ単位）の論理装置を設計言語で記述すると，製作に必要な結線情報が得られること，さらに，利用したいI.C.（集積回路）の実装情報をシステムに登録するのみで，そのICが利用可能になる等の特徴をもつ．

* F.G. Linnemann, C.E. Minich: "Logic Degin System", Share IEEE D.A. Workshop, pp.329~346, 1971.

2. システムの概要

システム全体は，図2－1に示すように，五つの処理部分からなっており，以下にその概要を述べる．

① コンパイラ

論理装置設計言語LORANで記述されたプログラムを解読・整理し，以下の②，③，④各部に必要なデータを生成・ファイルする．同時に文法エラーのチェックも行う．

② 制御論理生成部

コンパイラから与えられた，シーケンスに関する情報（制御表）より，順序回路に相当する論理式を生成する．

③ 論理式の処理部

①，②より得られた論理式を，それぞれ，NANDゲート，FF（フリップ・

フロップ）を用いたモジュール表現に変換する．

④モジュール実装情報作成部

ＮＡＮＤゲート，ＦＦ，その他のＩＣの名前，機能，ピンの接続状態，ピン数等をファイルに登録しておき，実装を行う際のデータとする．現在，約30種のＩＣを登録しているが，必要に応じて100種までの登録が可能である．また，登録内容の変更，削除も簡単に行なえる．

⑤実装部

本設計システムの最終段階で，①，③より得られたモジュール表現に，④で得られた実装情報を参照して特定のＩＣ，ピン番号を割付け，さらに入出力の負荷計算をした上で，回路図に相当する結線表を生成する．

３．記述言語

普通，設計者が論理装置を設計する際には，論理回路は論理式で，また順序回路はタイムチャート，ないしは，フローチャートで記述される場合が多い．一方最近の集積技術の進歩は著るしく，かなり複雑な論理機能を１チップ上に集積することが可能になっている．

これらの背景により，計算機システム全体を記述する設計言語の一歩として，任意の論理式を記述する論理処理文，フローを記述する制御文，ＳＳＩ，ＭＳＩ，ＬＳＩ等の特殊機能を記述するモジュール文から構成される言語ＬＯＲＡＮを紹介する．

３．１　記述言語の構成

ここでは，ＬＯＲＡＮの構成をまず超言語で記述し，以後，各文の機能を順を追って示す．

Ａ）超言語による記述

ＬＯＲＡＮ言語をバッカス記法を用いて示す．

(1)　＜ＬＯＲＡＮ記述＞ ::= ＜制御／論理処理部＞＜構造部＞

(2)　＜制御／論理処理部＞ ::= ＢＧＮ␣ＣＮＴ＜装置名＞
{＜制御文＞ | ＜論理処理文＞}...
ＥＮＤ␣ＣＮＴ

ＢＧＮ␣ＣＮＴ，ＥＮＤ␣ＣＮＴは，それぞれ，begin control，end control を表わす．

(3)　＜構造部＞ ::= ＢＧＮ␣ＭＤ
＜モジュール文＞...
ＥＮＤ␣ＭＤ

ＢＧＮ␣ＭＤ，ＥＮＤ␣ＭＤは，それぞれ，begin module，end module を示す．

(4)　＜制御文＞ ::= ＜ＧＯ　ＴＯ文＞｜＜ＪＯＩＮ文＞｜＜ＷＡＩＴ文＞｜＜ＩＦ節＞＜ＧＯ　ＴＯ文＞

(5)　＜論理処理文＞ ::= ＜ＳＥＴ文＞｜＜ＬＥＴ文＞｜＜ＣＮＴ文＞｜＜ＩＦ節＞＜ＳＥＴ文＞｜＜ＩＦ節＞＜ＣＮＴ文＞

(6)　＜ＩＦ節＞ ::= ＩＦ（＜論理式＞）

(7)　＜ＧＯ　ＴＯ文＞ ::= ＧＯ　ＴＯ［＜ラベル名＞，］...

(8) ＜ＪＯＩＮ文＞::＝ＪＯＩＮ＜ラベル名＞
(9) ＜ＷＡＩＴ文＞::＝ＷＡＩＴ（＜論理式＞）
(10) ＜ＳＥＴ文＞::＝ＳＥＴ＜論理変数名＞＝＜論理式＞
(11) ＜ＬＥＴ文＞::＝ＬＥＴ＜論理変数名＞＝＜論理式＞
(12) ＜ＣＮＴ文＞::＝ＣＮＴ［＜制御変数名＞，］...
(13) ＜モジュール文＞::＝＜機能名＞（［＜論理変数名＞,］...／［＜論理変数名＞］...）
(14) ＜ラベル名＞::＝＜英字＞｜＜ラベル名＞＜英字＞｜＜ラベル名＞＜数字＞
(15) ＜論理式＞::＝＜項＞｜＜論理式＞＋＜項＞｜－＜項＞
(16) ＜項＞::＝＜１次子＞｜＜項＞＊＜１次子＞
(17) ＜１次子＞::＝＜論理変数名＞｜（＜論理式＞）
(18) ＜名前＞::＝＜英字＞｜＜名前＞＜英字＞｜＜名前＞＜数字＞
(19) ＜正論理変数名＞::＝＜名前＞
(20) ＜負論理変数名＞::＝－＜正論理変数名＞
(21) ＜論理変数名＞::＝＜正論理変数名＞｜＜負論理変数名＞
(22) ＜制御変数名＞::＝＜名前＞
＜機能名＞::＝＜名前＞
＜装置名＞::＝＜名前＞

以下＜英字＞，＜数字＞はＦＯＲＴＲＡＮに準ずる．ただし，＜モジュール文＞の＜機能名＞は，３節ｂ)で述べる．

Ｂ）制御文

制御文は，処理の流れを記述するもので，並列処理や，待ち合わせ処理の記述を考慮した文が含まれている．ここでは，各制御文を例をあげながら説明する．

(1) ＧＯ　ＴＯ文

例３－１）Ｓ１：ＧＯ　ＴＯ　Ｓ２，Ｓ３，Ｓ４

ＦＯＲＴＲＡＮと違って，ラベル名を複数個並べることができ，制御パスを一ヶ所から複数に分岐できる．

(2) ＪＯＩＮ文

例３－２）Ｓ７　ＪＯＩＮ　Ｓ１０

ＪＯＩＮ文は，必ずＧＯ　ＴＯ文と対になって使われ，この対により，ＦＯＲＫ－ＪＯＩＮ型の並列処理を行なう．

(3) ＷＡＩＴ文

例３－３）『ＷＡＩＴ（Ａ＊Ｂ）

この文は，待ち合わせ処理用に導入された文で，Ａ＊Ｂの値が１になると，次の文に制御を移すことを表わす．

(4) ＩＦ節

例３－４）『ＩＦ（Ａ＋Ｂ）　ＧＯ　ＴＯ　Ｓ１

括弧内が論理式であることを除けば，ＦＯＲＴＲＡＮと同様の機能である．

Ｃ）論理処理文

論理処理文には，まず，任意の論理式を記述するために用意された文として，式の内容が特定のタイミングで実行されるＳＥＴ文と，各タイミング毎に実行されるＬＥＴ文の２種類がある．

例３－５）　Ｓ２　『ＳＥＴ　Ａ＝Ｂ＋Ｃ（Ｄ＋Ｅ）
『ＬＥＴ　Ａ＝Ｂ＋Ｃ＊Ｄ

ＳＥＴ文では，右辺の論理式がＳ２のタイミングで実行され，かつ，記憶される．これに対し，ＬＥＴ文では，単に右辺の論理式の値を左辺の論理変数に置き換えることを意味し，ハード的には，単なる組み合せ論理回路となる．

論理処理文には以上の他に，制御変数の概念がある．これは，次項に述べるモジュール文と組み合わさって一つの意味をもつ．

例３－６）　Ｓ３　'ＣＮＴ　ＣＯＵＮＴ，ＳＨＩＦＴ

この例では，ラベルＳ３に制御が移ると，制御変数ＣＯＵＮＴ，ＳＨＩＦＴを論理値１にし，つぎのタイミングには（つぎの文に制御が移るから），ＣＯＵＮＴ，ＳＨＩＦＴの値を０にすることを示す．

Ｄ）モジュール文

＜モジュール文＞は，ＬＯＲＡＮ言語の特徴の一つで，ソフトウェアの面からみると，ＦＯＲＴＲＡＮの組み込み関数，あるいは，システムに登録されたサブルーチンと同様の機能をもっており，システムにそのＩＣに関する実装情報の登録をすれば（図２．１の④参照），ＬＯＲＡＮで再び定義することなしに利用できるようにしたものである。設計者はこれにより，ＩＣの物理的な内部構造を知る必要もなく，提供された登録済みのＩＣのマニュアルを手引きに，容易にＩＣが組み込まれるのが大きなメリットである．

モジュール文は，機能素子単位に書かれた表現形式をとっており，各機能素子を暗箱と考え，各暗箱を

＜機能名＞（＜入力１＞，・・・・，＜入力ｎ＞／＜出力１＞，・・・・，＜出力ｎ＞）

の形で記述したものである．

例３－７）　'ＭＤ　ＳＲ（Ｄ４，Ｄ３，Ｄ２，Ｄ１，ＳＩ，ＭＣ，ＳＨＩＦＴ，ＬＯＡＤ／Ｍ４，Ｍ３，Ｍ２，Ｍ１）

これは、並列ロード機能をもった右シフトレジスタのモジュール文である。並列ロードするときは、ＭＣを論理値１にし、ＬＯＡＤを１→０にすると，（Ｄ４，Ｄ３，Ｄ２，Ｄ１）→（Ｍ４，Ｍ３，Ｍ２，Ｍ１）なるロードを行なう。シフトを行なうには、ＭＣを０にし、ＳＨＩＦＴを１→０にすると，（Ｍ４，Ｍ３，Ｍ２，Ｍ１）→（ＳＩ，Ｍ４，Ｍ３，Ｍ２）なるシフトが行なわれる。

さて、制御変数文の例３－６）のＳＨＩＦＴは、上のモジュール文中のＳＨＩＦＴと同じ名前であるので、同一信号として扱かわれる。従って、ハードウェアの面からながめれば、制御変数ＳＨＩＦＴが１→０に変化する瞬間に、このシフトレジスタはシフトを行うことになる。この例に限らず、＜制御／論理処理部＞と＜構造部＞との間の、共通の変数名によって、情報のやりとりが行なわれることになる。

３．３　記述例

ここでは、乗算器を例にあげて、ＬＯＲＡＮの記述例を示す。乗算器は８ビットの乗算器で、乗算のフローを図３．１に示す。このフローに従って乗算器は図３．２のように記述できる。

４．制御論理の合成

４．１　状態推移表の作成

図４．１は乗算器を処理した結果の状態推移表である。状態推移表は、ＬＯＲＡＮの記述から，制御の流れの情報のみを抽出し、表にまとめたものであり，制御回路生成の際に使用される。

4.2 状態割当

現在、状態割当プログラムとしては、1)正方形法による状態割当，2)カウンタ・デコーダ法，3)1状態1FF割当法 以上3種開発しているが，本システムでは、指定の無い限り3)の方法を用いており、1)，2)はオプションとなっている。

図4.1に、3)による状態割当を施した結果，生成される論理式は、

S1;MULTI , S2;S1+S2*-CE , S3;S2 , S4;S3*CE

となる。ここで，代入演算子";"は、SET文と同様右辺の値を下FFに記憶することを示す。

5. 論理式からモジュール文への変換

ここでは、以上の過程で得られた論理式を論理回路として実現するために、NANDゲートや、下FFを用いたモジュール表現へ変換することを目的とする。

例5.1) A=B+C(D+E)を分解すると次のようになる。

A=B+N1 , N1=C*N2 , N2=D+E

モジュール表現への変換は、これらにNANDゲートを割り付ける。

NAND2(-B,-N1/A), NAND2(C,N2/-N1), NAND2(-D,-E/N2)

この際，-B，-N1，-Dに必要なインバータは、この段階では生成されず，実装段階で最終的に必要なものだけ生成することにより，冗長なインバータの生成を防いでいる。論理段数の最小化をはかることにより、時間的遅れを少くすると共に、パラメータによる指定入力以上の入力数をもつ項は、自動的に分解するように考慮している。以上の手続きにより、論理式はモジュール表現に変換され，モジュール表現ファイルに貯えられ、これでLORANで記述されたすべての指令は、モジュール表現に変換されたことになる。

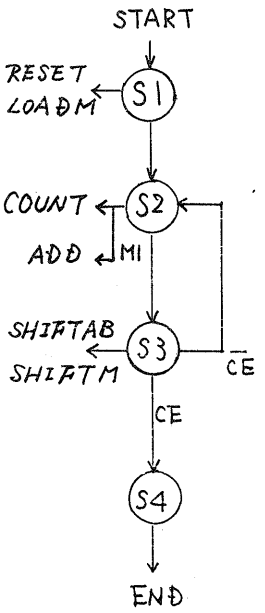

図3.1
乗算器のフロー

```
CONI INI
 :BGN CNT (MULTI)
S1 :CNT RESET,LOADM,MCM
S2 :CNT COUNT
 / :IF(M1) CNT ADD,MCAB
S3 :CNT SHIFTA,SHIFTM
 / :IF(-CE) GO TO S2
S4 :END CNT
 :BGN MD
 :MD COUNT16(COUNT,C1,RESET,T/CE,*,*,C1)
 :MD AD(A4,A3,A2,A1,R4,R3,R2,R1,GND/AD4,AD3,AD2,AD1,CO4)
 :MD AD(A8,A7,A6,A5,R8,R7,R6,R5,CO4/AD8,AD7,AD6,AD5,*)
 :MD SR(D8,D7,D6,D5,GND,MCM,SHIFTM,LOADM/*,*,*,M5)
 :MD SR(D4,D3,D2,D1,M5,MCM,SHIFTM,LOADM/*,*,*,M1)
 :MD SR(AD4,AD3,AD2,AD1,A5,MCAB,SHIFTA,ADD/A4,A3,A2,A1)
 :MD SR(AD8,AD7,AD6,AD5,GND,MCAB,SHIFTA,ADD/A8,A7,A6,A5)
 :MD SR(T,T,T,T,A1,GND,SHIFTA,T/B8,B7,B6,B5)
 :MD SR(T,T,T,T,B5,GND,SHIFTA,T/B4,B3,B2,B1)
 :MD EXT(T.0,R8.1,R7.2,R6.3,R5.4,R4.5,R3.6,R2.7,R1.8,
1D8.9,D7.10,D6.11,D5.12,D4.13,D2.14,D2.15,D1.16,MULTI.17/
1A8.18,A7.19,A6.20,A5.21,A4.22,A3.23,A2.24,A1.25,B8.26,
1B7.27,B6.28,B5.29,B4.30,B3.31,B2.32,B1.33,S4.34)
 :END MD
 :LORAN END
```

図3.2
乗算器の記述

図4.1
乗算器の状態推移表

CONTI

ASTML	TCOND	PSTML
S 1	1	S 2
S 2	1	S 3
S 3	- C E	S 2
S 3	C E	S 4

6. 実装自動化システムのあらまし

本システムの主フローを図2.1に示す。実装設計を行う前に、各論理機能素子（ゲート、FF、カウンタ、メモリ等）のモジュール表現を定め、その実装情報を同図①の登録編集システムを用いてファイルしておくことが必要である。逆に言えば、登録さえしておけば、実装システムの変更なしに、任意のICの論理機能を利用できる。

設計者は、登録されたモジュール表現にしたがって、任意の論理システムを記述し、カードにパンチする。その後の処理は、実装設計プログラムが自動的に行なう。システムは、必要ならばインバータ（NOT素子）を追加し、ICおよびピンを割り付け、負荷解析の結果と共に、ICのピン間の結線表を出力する。

7. モジュール表現

モジュール表現には、論理機能を記述する論理モジュール表現と、外部端子を示す特殊モジュール表現がある。今後、特にことわらない限りモジュール表現といえば、論理モジュール表現を示すものとする。

7.1 論理モジュール表現

モジュール表現は一般に

〈機能名〉(〈入力変数リスト〉/〈出力変数リスト〉)

の形で記述される。簡単のため例をあげて説明する。

例として非同期入力の同期化回路をあげる。ASYINが非同期入力，SYOTが同期化出力，CLは同期クロックである。この回路を、モジュール表現すると次のようになる。

FF(ASYIN,-ASYIN,CL,T,T/*,-FFA)
NAND2(ASYIN,-FFA/KB)
FF(-KB,KB,CL,T,T/SYOT,-SYOT)

FFは、プリセット，クリア入力付きのJ-Kマスタ・スレーブ型FFであり，入力変数リストは左からJ、K，クロック，プリセット，クリアであり，出力変数リストはQ，−Qを示す。ここで変数の前の"−"は，論理否定を示す。NAND2は2入力NANDを示す。また、Tは論理値"1"を，＊は必要としない出力変数を示す。

7.2 擬似モジュール表現

擬似モジュール表現は、実装に必要な情報を、モジュール表現に類似した形で記述するもので、現在、〈外部端子記述〉と、〈テスト端子記述〉の2つが用意されている。記述形式を次に示す。

EXT(〈入力端子リスト〉/〈出力端子リスト〉)
〈入力端子リスト〉::={〈入力変数〉.〈端子番号〉,}...
〈出力端子リスト〉::={〈出力変数〉.〈端子番号〉,}...

〈EXT文〉は、特定の変数が外部より（に）接続されることを示すと共に、基板上のピン位置を設計者が指定するのに用いる。〈端子番号〉は整数を用いて指定する。これは、複数の基板にまたがるシステムを設作するとき、外部端子の位置をそろえるために用いる。〈テスト端子記述も同様であり、

TEST(〈入力端子リスト〉/〈出力端子リスト〉)　と記述する。

8. モジュール表現の実装情報の登録

8.1 実装情報の記述形式

モジュール表現で記述されたシステムを物理的に実現するためには，モジュール表現の論理機能を実現するデバイス（IC）を選択し，そのICの仕様にしたがって，必要な結線を行なうことが必要である。そのためには，①．ICの選択，②．ICの物理的な形，③．電源　④．モジュール文の各変数に対するピン番号と，FAN-IN，FAN-OUTの大きさ等の情報が必要である．以下，例をあげて記述形式の説明を行なう。

例 8.1）2入力NAND素子の実装情報

```
'BEGIN MODULE                                    (1)
'LEVEL 0                                         (2)
'MODULE NAND2(A,B/C)                             (3)
'NAME  SN7400                                    (4)
'SIZE  14DIP                                     (5)
'POWER (7,GND,12/14,+5V,12)                      (6)
'PINCN (1.1,2.1/3.10)(4.1,5.1/6.10)              (7)
1       (10.1,9.1/8.10)(13.1,12.1/11.10)         (8)
C  QUAD 2 INPUT TTL NAND GATE                    (9)
C  COST 100 YEN                                  (10)
'END  MODULE                                     (11)
```

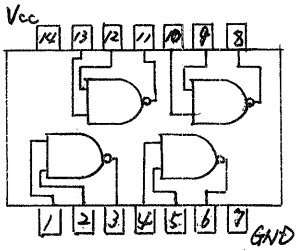

図 8.1 SN7400 ピン接続図

この例は，2入力NANDを4個実装したT.I.社のIC，SN7400を用いて，モジュールNAND2の実装情報を記述した例である。記述は，BEGIN MODULEで始まり，END MODULEで終了する。第2行目のLEVEL 0は，このモジュールが基本素子であることを示す。

第3行目はMODULE文である。NAND2が機能名，A，Bが入力，Cは出力である。

第4行目はNAND2の論理機能を実現するICの型番であり，製作，または，発注するときの資料となる。

第5行目はICのパッケージ形式を示し，例の場合は，14ピンのDual line In Packageであることを示す。

第6行目はICの電源と，必要な容量を示す。例の場合は，7ピンを接地（GND）し，14ピンに+5Vの電圧をかけ，それぞれの使用電流は12mAであることを示す。

第7，8行は，MODULE文の変数名と，ICのピン接続との対応関係を示す。第7行の（1.1，2.1／3.10）は，1ピン，2ピンがそれぞれ入力で，入力電流は1単位電流であり，また，3ピンは出力で，許容電流は10単位電流であることを示している。各ピンの記述順序ならびに個数は，第3行目のMODULE文と一致させる必要がある。SN7400の場合，2入力NAND素子4個が1つのパッケージにまとめられており，それらはPINCN文として1つの文に記述される。参考のため，SN7400のピン接続図を図8.1に示す。

8.2 実装情報の登録システム

実装情報登録プログラムは，カードから実装情報記述を読み取り，それを内部表現に変換して，マガジンに貯えるのが主機能であるが（図2.1），その他の機能

として次のものがある。

1) 文法誤りのチェック --- 各文の順序・形式、また、モジュール文とピン指統文との対応等についてチェックを行なう。
2) 重複登録のチェック
3) 編集 --- 登録内容の変更、削除、修正を行なう。
4) 一覧表の作成 --- 図8.2にモジュール登録表の一部を示す。同図において、NOはモジュール番号、RNは、実装設計時に、マガジンからドラムに転送される際の転送先アドレスを示す。

9. 実装設計の概略

9.1 内部形式への変換

モジュール表現は、VNTと呼ぶ変数名表と、CNTと呼ぶ指統表に分解して主記憶に記憶される。VNTは、変数名とCNTへのポインタから構成される。CNTは、モジュール表現を記憶するもので、モジュール名は番号(図8.2)で、変数名はCNTのポインタで記憶する。1つのモジュール表現は、シーケンシャルに記憶されるが、モジュール間、および、異なるモジュールに現われる同一変数は、CNT上でリング構造で結びつける。また、CNTにはIOおよびピンの割り付け結果、さらに、各ピンのロード・ソース電流値等も記憶される。これらの値の記憶には、1語(2バイト)を用いる必要がないのでパックして記憶するため、1変数当りには3語のメモリを必要とするのみである。

9.2 インバータの追加

モジュールの入出力変数には、－<変数名>という形が許されており、これは<変数名>の論理否定を示す。この場合、たとえば入力変数に－Aがあり、出力にはAのみが存在する(あるいは、この逆)ときは、NOT素子(インバータ)INV(A/－A)というモジュール表現を追加する必要がある。このステップでは、変数名表と指統表を調べ、必要があればインバータを指統表に追加する。インバータの決定を論理設計終了後に自動的に行なうことにより、冗長なインバータの使用を避けることができる。

9.3 IOおよびピン割付

この手続きが実装段階の主軸となる。ここで行なうことは、まず、§2でマガジンに記憶された実装情報をドラムに読み込み、CNTとVNTに展開されたモジュール文をつき合わせ、対応するIOと、ピン番号を割り付けることである。

本システムで採用した割り付け手順は、モジュール毎に実装情報を読み出し、対応するIOを決定し、そのIOをモジュール割り付け表(MASMT)に登録する。1つのIOが複数個のモジュールを実装している場合、まず、最初のモジュールのピン群を割り当てる。次のモジュール表現で、再び同じモジュールがでてきたら、2番目の実装モジュールのピン群を割り付ける。

モジュールの名前と、実装情報のドラム上の位置を記憶したモジュール表(MDムT)は主記憶上にソートして配置し、実装情報はドラム上に配置して、記憶の節約と、処理速度の向上を計っている。

この方法では、モジュール表現で記述された順にIOに実装していくため、場合によっては、モジュール表現の位置がずれると、実装位置が隣接しないことになる。しかしながら、実装位置が隣接すべきモジュールはモジュール表現上でも

隣接して記述されることが多いので、ICそのものの位置は、次節の手順で決定される。また、ピン決定と同時に負荷量をCNT上に記憶し、後述の負荷解析のデータとする。

また、新しいICを割り付ける毎に、実装情報の〈POWER文〉により、必要な電源をVNTとCNTに登録する。したがって、MOSIC等、2電源を要求するICの電源も、記述にしたがって自動的に処理できる。

9.4 ICの配置

このステップでは、与えられた基板の上のICの位置を決定する。入出力、および、テストピン(外部端子)の位置は、設計者により与えられる。配置アルゴリズムは、外部ピンと接続関係の深いものよりICを配置し、後に、配線長が短くなるように再配置をする、という2段構成をしている。また、ICのサイズも実装情報の中に含まれているから、MSI, LSI等も万能基板を用いて配置が可能である。さらに、基板のサイズ等の情報は外部から与えられるから、データさえあれば、任意の基板を使用できる。*

9.5 ICの割付表, および結線表の出力

すべてのモジュール表現に対し実装を行なった後、各変数ごとにリストをたどって、ロード・ソースの解析を行ないながら、実装結果を出力する。この際検出される事項は、1) NO LOAD, 2) NO SOURCE, 3) DOUBLE LOAD, 4) OVER LOAD の4項目である。出力結果は図10.1(c)に示すが、この表に従って、与えられたユニットのピンを結線することにより、必要な論理システムを製作することができる。

```
        ** MODULE TABLE **

NAME            NO          RN
AD              14          1995
COMPA4          35          1972
COUNT10         26          1983
COUNT16         13          1996
D2-4DECODE      29          1980
D4-1MULPX       24          1985
DFF             19          1990
EXOR2           25          1984
FF              18          1991
```

← 図8.2 モジュール登録表

```
:CARD
  1   EXT(ASYIN.1,CL.2,T.3/SYOT.4,-SYOT.5,*.6)
  2   FF(ASYIN,-ASYIN,CL,T,T/*,-FFA)
  3   NAND2(ASYIN,-FFA/KB)
  4   FF(-KB,KB,CL,T,T/SYOT,-SYOT)
:RUN
```

図10.1 (a) 同期化回路の実装出力

10. 実装システムの適用例

7.1節で記述した例を、本システムに適用した結果を示す。図10.1(a)は入力リスト, (b)がモジュール割付表(MASMT), (c)が結線表である。結線表において、-ASYINという変数は、3番IC(SN7404)の2ピンより出力されて、1番IC(SN7476)の16ピンに入力されることを示す。

図10.2(a)に7節の例で、インバータを2入力NANDで記述した例を示す。この場合、インバータの自動追加が不要になるので、同図(b)のように、必要なICの個数は2個となる。

---

* この部分のプログラムのみは、柔軟性を増すためプログラムを変更中であり、まだ完成していない。

11. 結言

本システムは、論理設計と実装設計を一貫して行う自動化システムであり、IC換算で最高100個程度のシステムを、記述言語より自動生成する。また、非同期システム等に対しては、モジュール表現から実装システムのみを利用して製造情報を得ることもできる。また、モジュール表現を入力としたシミュレーションシステムを現在作成中である。

本システムは、FORTRANで約5000ステップ、その他若干のアセンブラのTOOLルーチンから構成されており、処理時間は、例であげた乗算器で5分程度である。

このシステムを利用して最大の弱点と感ずるのは、部分的な設計変更ができない点である。一旦製作してしまうと、その後設計ミスが発見されたときの部分的変更は、すべて人手で行なわなければならない。

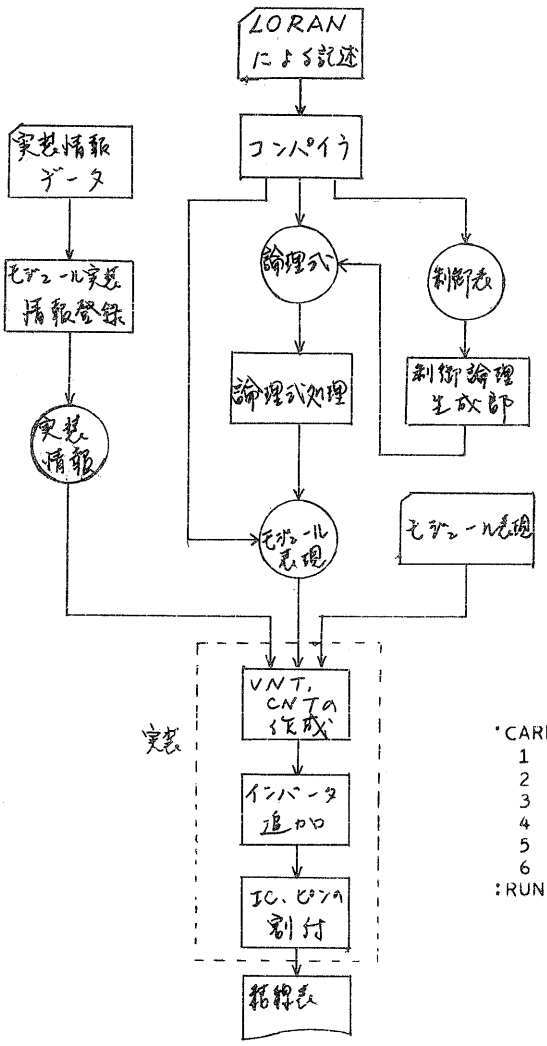

図2.1 全体のフロー

```
NO    *MODULE NAME*   *MODULE SIZE*
-------------------------------------
1         SN7476          16DIP
2         SN7400          14DIP
3         SN7404          14DIP
```

(b)

HENSUMEI	UNI	PIN	I/O	FAN
-ASYIN	1	16	1	1
	3	2	-1	10
-FFA	1	14	-1	10
	2	2	1	1
-KB	1	9	1	1
	3	4	-1	10
-SYOT	0	5	1	2
	1	10	-1	10
ASYIN	0	1	-1	8
	1	4	1	1
	2	1	1	1
	3	1	1	1
CL	0	2	-1	8
	1	1	1	2

(c)

図10.1 同期化回路の実装出力

```
*CARD
  1   EXT(ASYIN,1,CL,2,T,3/SYOT,4,-SYOT,5,*,6)
  2   FF(ASYIN,-ASYIN,CL,T,T/*,-FFA)
  3   NAND2(ASYIN,-FFA/KB)
  4   FF(-KB,KB,CL,T,T/SYOT,-SYOT)
  5   NAND2(ASYIN,T/-ASYIN)
  6   NAND2(KB,T/-KB)
:RUN

 IO    *MODULE NAME*   *MODULE SIZE*
-------------------------------------
 1         SN7476          16DIP
 2         SN7400          14DIP
```

図10.2 同期化回路の実装出力(インバータ記述)