



自動論理解析システム：CALAS*

元 岡 達** 杉 浦 宣 紀***
 椎 野 努*** 武 内 惇***

Abstract

CALAS is the automatic logic analyzer on digital systems which is implemented in FORTRAN. The input language in which given digital systems are described is a Boolean equation type language. The output is described in the language which is equivalent to sequential control flows. Therefore, CALAS is the converter between 2 types of system description languages.

The characteristics of CALAS are as follows; (1) A large scale system is analyzed efficiently because of requiring relatively small capacity of storage and executing in relatively short time, (2) It can deal a system with asynchronous loop circuits, (3) The system behavior is understood easily, because of simplification of logic equations and (4) Various analysis modes can be specified by the command system.

1. ま え が き

論理設計の自動化に関しては、これまで多くの記述言語や、実験システムの報告がなされている^{1)~5)}。

しかし、その中で実用的なシステムとして現実の設計に広く用いられているのは、主にゲートレベルの論理シミュレータに限られており、論理回路の合成を含む完全自動化システムの実現には、まだ、多くの解決すべき問題が残されている。

自動論理解析システムでは、ブール式で表現された論理システムを、状態遷移形表現に自動的に変換するので、これを用いて仕様論理と設計論理回路動作とを照合し、設計誤りのチェック、修正を行なうことができる。自動論理合成システム、あるいは、人手による論理合成と組合せて、設計の帰還ループを構成することにより、設計の迅速化、正確化を図り、これによっ

て大規模論理回路の設計も可能になることが期待される。このような観点から、文献 5)では、先に文献 3)で提案された論理設計言語 LDS をもとにして、自動論理解析を行なう場合の、基本アルゴリズム、および、主として原理的確認のために作成された実験プログラム RLC 23 について述べられている。

本稿では、この論理解析アルゴリズムを用いて、現実の論理回路設計に使用しうる実用システムとして開発した自動論理解析システム CALAS (Computerized Automatic Logic Analysis System)について述べる。この種のシステムが実用に供しうるかどうかは、取り扱っている論理回路の規模（計算機内のメモリ占有量）、および、計算に要する時間の長さに依存する。CALASでは、言語に FORTRAN を使用し、内部データ表現にはテーブル形式をとったこと、および、各種マトリクスの表現にスパースマトリクス法を採用したことなどにより、計算機内メモリ占有量の縮小化、および、解析時間の短縮化が可能になった。また、非同期ループ検出機能、出力論理式の簡単化機能、モジュール化された論理回路の外側論理回路への展開処理など、実用システムとして有効な各種の処理機能を持たせてある。さらに、解析、修正のくり返しによる設計を効果的に行なう手段として、入力変数****間の関係の

* Computerized Automatic Logic Analysis System: CALAS by Tohru MOTO-OKA (Faculty of Engineering, University of Tokyo) Nobunori SUGIURA Tsutomu SHIINO and Atsushi TAKEUCHI (Software Systems Division, OKI Electric Industry Co., Ltd.).

** 東京大学工学部電気工学科

*** 沖電気工業（株）ソフトウェア事業部

**** 一つのモジュール化された論理回路の入力端子を示す変数を入力変数、出力端子を示す変数を出力変数と定義する。

指定, 解析する必要のない状態(遷移禁止状態)の指定, 値の決まらない状態を定義する制御変数の数が一定値を越えると解析を一時停止させる指定(計算時間が膨大になるため解析を続けるかどうか使用者に判断を求める)など, 種々の解析制御パラメータによる解析の制御を可能にした。また, RLC 23 において設定されていた入力論理回路表現上の種々の制限を大部分解除し, 自由度を大幅に高めている。

本稿では主として, CALAS の処理方式, 内部データ構造, テーブル構成による解析手法, 非同期ループの検出方式, および, 出力表現を簡潔にするための論理式の簡単化手法などについて述べる。

2. システムの概要

本システムは, LDS 言語³⁾のレベル2言語(ブール式表現)で記述された入力論理回路モデルを解析し, レベル3言語(状態遷移形式)による表現に変換するものである。Fig. 1 に CALAS のシステム動作概要を示す。まず, 解析制御パラメータ指定カード(7章参照)を解釈し, 解析制御パラメータ値をセットする。次に, 入力論理記述を最も内側のモジュールから, 構文の誤りを調べながら解釈し, システムの内部表現に

変換する。このとき, 処理指定のないモジュールは外部記憶にそのままの形で掃き出す。構文に誤りのなかったモジュールは, 展開処理(解析制御パラメータの指定による), 入力変数間の関係式の処理と変数の消去処理, 非同期ループの検出処理を行ない, 結果をレベル2記述に変換し, 外部記憶に掃き出す。さらに, 代入計算の順序付けなど変換処理の前に必要な種々の前処理を行なう。

次にモジュールの変換処理を行ない, 結果を内部表現からレベル3記述に変換し, 外部記憶に掃き出す。解析を終了したモジュールを外部記憶に掃き出すことにより, 大規模論理回路の解析を可能にしている。このようにして, 内側のモジュールから順次解析を行ない全モジュールの解析終了後, 出力処理プログラムにより外部記憶に記憶されている内容を整理して出力する。

3. 内部データ構造

内部データは, メモリの節約, 解析の迅速化に主眼をおき, 以下のようなテーブル構造で記憶している。

各モジュール, 変数に関する情報は, それぞれ Fig. 2 (a), (b) に示す構造のモジュール表, 変数表に記憶される。モジュール名, 変数名は通常2ワード(8文字以内)で記憶されるが, 8文字以上31文字以下のものは Fig. 2 (c) のように, 次の行を用いて記憶される。入力論理式は, Fig. 3(次頁参照)に示すような論理式表に変数表の番地と数字符号を用いて記憶さ

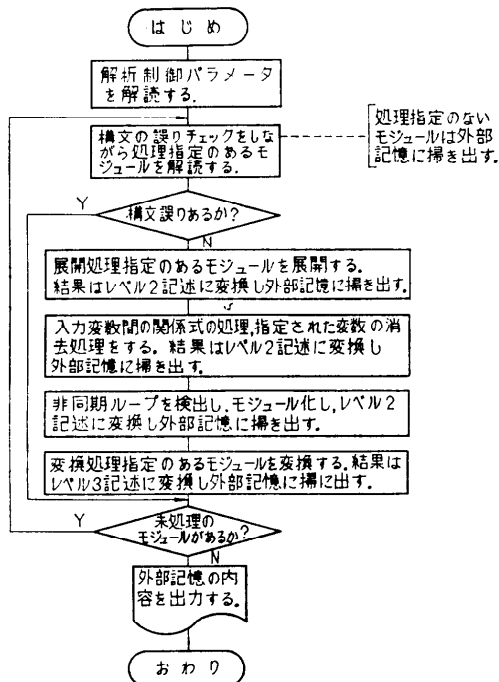


Fig. 1 System flow

モジュール名 (2ワード)	モジュール 種別 (1ワード)	変数表への ポインタ (1ワード)	論理式 表へのポインタ (1ワード)	モジュールの 階層レベル (1ワード)
------------------	-----------------------	-------------------------	--------------------------	---------------------------

(a) モジュール表
MODULE table

変数名 (2ワード)	変数種別 (1ワード)	変数の値 (1ワード)	論理式表 へのポインタ (1ワード)	補助論理式 表へのポインタ (1ワード)
---------------	----------------	----------------	--------------------------	----------------------------

(b) 変数表
VARIABLE table

A B C D E F G H I *	
J K L M N O P Q R	-----

(c) 8文字以上のモジュール名, 変数名の記憶法
(*は継続マーク)
module name, variable name storing
format (* continuation mark)

Fig. 2 MODULE table, VARIABLE table

内 容	意 味
Yの変数表における番地 =を表す数字記号	Y =
Aの変数表における番地	A &
Bの変数表における番地に負号を付けたもの	B
┐を表す数字記号	┐
Cの変数表における番地	C
式の終り記号(¥)を表す数字記号	¥

Fig. 3 EQUATION table

れる。たとえば、論理式* $Y=A \& B | \neg C \text{ ¥}$ は変数 A, B, C, Y がそれぞれ変数表における番地で表わされ、演算子 $\&, |$ はそれらの左側にある変数の変数表の番地の符号によって、Fig. 3 のように記憶される。すなわち、 $A \&$ は A の変数表の番地、 $B |$ は B の変数表の番地に負号をつけたものとなる。このような記憶法により、論理式記憶領域は大幅に縮小され、また、後述するパタン照合による代入計算をきわめて効率よく行なうことができる。また、制御変数**値の列($|C_1|, \dots, |C_r|$)で定義される状態 S は、できるだけビット操作をさけるため次式のようにコード化され、Fig. 4 に示す構造の状態表に記憶される。

$$|S| = \sum_{i=1}^P |C_i| \times 2^{i-1} \quad (1)$$

ただし、 $|C_i|$ は制御変数 C_i の値である。

4. 解析処理手法

主な解析処理は次の三つである。

- (1) モジュールの展開処理
- (2) 非同期ループの検出とそのモジュール化
- (3) レベル変換処理

以下、各々の処理について述べる。

4.1 モジュールの展開処理

展開処理指定されたモジュールでは、入力処理の段階で、各変数名はそのモジュール名で修飾された形で変数表に記憶される。論理式は論理式表と同じ構成の補助論理式表にいったん記憶される。展開処理は、展開処理指定されたモジュールの外側のモジュールの内容が記憶されている論理式表内に、その補助論理式表

* $\&$: 論理和、 $|$: 論理積、 $\neg$: 否定、 ¥ : 論理式の終り記号である。

** 回路を制御部分と演算部分に分けたとき制御部分の記憶要素を表わし、回路の状態を定義するものを制御変数、演算部分の記憶要素を表わすものをデータ変数と定義する。

*** 論理回路の端子のうち入力端子、出力端子、記憶素子の端子以外の端子を示す変数。

**** 端変数とは入出力端子の変数のことで、もとの回路から、入出力端子、記憶要素の端子を除いたとき生じる新しい入出力端子を示す変数も含む。

状態のラベル名 (1ワード)	状態コード (2ワード)	補助論理式 表へのポインタ (1ワード)
(a) 状態表 STATE table	(b) 遷移表 TRANSFER table	

Fig. 4 STATE table, TRANSFER table

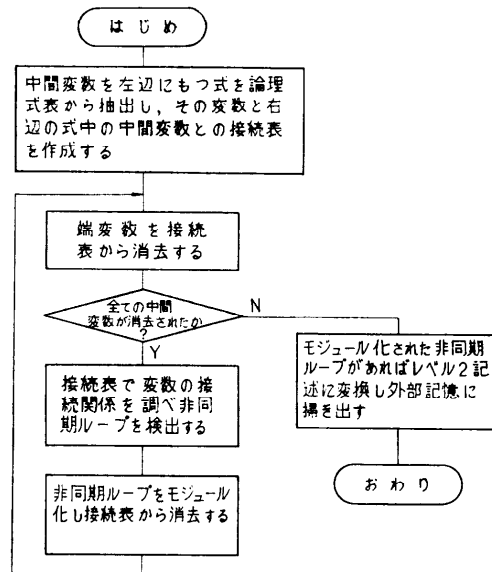


Fig. 5 Asynchronous loop detection and modulization

の内容を組み込むことにより行なわれる。

4.2 非同期ループの検出とモジュール化

現実の論理回路では、しばしば非同期ループが一種のレジスタ機能を持った回路素子として用いられる。このような非同期ループは、その構成変数を制御変数と定義しないかぎり、回路の状態を構成する要素にはならず、代入計算をループさせ、解析を不可能にする。また、大規模な論理回路の設計に当たって、その煩雑さから設計を誤り、非同期ループを構成してしまう場合もある。したがって、論理回路の解析に当たっては、非同期ループの検出が不可欠となり、CALASではFig. 5に示すようにして非同期ループの検出を行なっている。まず、中間変数***の定義式から変数間の接続表を作成し、論理回路の外側から端変数****を順次検出消去する。その結果、消去されずに残った変数について、右辺の定義式中に中間変数を最も多く含む変数を起点に、変数間の接続関係を接続表上で追跡し、 N ステップでもとの変数にもどる変数の接続パスを検出する。これが N 変数からなる非同期ループである。こ

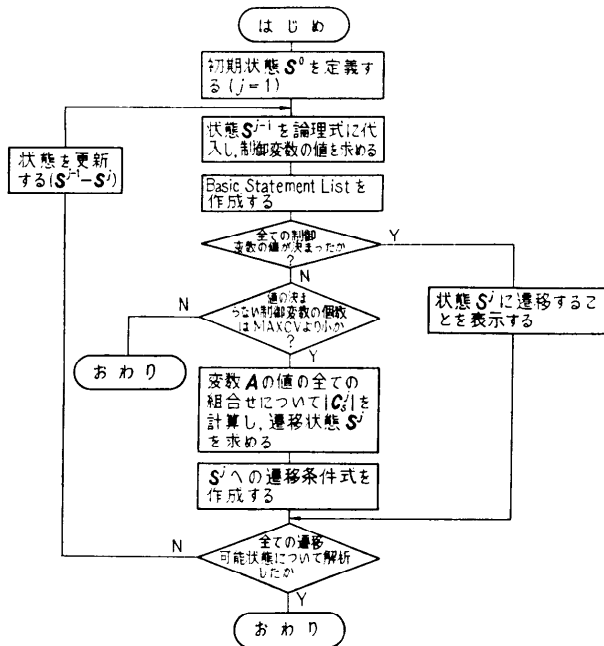


Fig. 6 Level transfer flow

の非同期ループは一つのモジュールとして回路からくり出す。このとき、非同期ループへの入力端子と出力端子は、それぞれ出力変数、入力変数と再定義する。次に、端変数の検出消去処理からくり返す。このようにして、非同期ループを検出し、展開されないモジュールとしてレベル 2 記述に変換し、外部記憶に書き出す。4 変数以上からなる非同期ループについては構造解析はせずに、まとめて出力する。

4.3 レベル変換処理

レベル 2 記述からレベル 3 記述への変換処理法を Fig. 6 にしたがって述べる。

まず、状態 S^{j-1} ($|C_1^{j-1}|, \dots, |C_r^{j-1}|$) を論理式表に記憶されている論理式に代入する。論理式表の構造から、論理演算を Fig. 7 に示す 10 種類の演算パターンに分類し、これらと論理式表の内容をパターン照合して代入計算を行なう。次に、論理回路を制御変数の値を決定する制御部分と、その他の演算部分とに分けたとき、状態 S^{j-1} において演算部分で実行される論理演算の結果を、Basic Statement List*として補助論理

* 演算部分の記憶素子を示すデータ変数、出力変数、中間変数、観測変数（使用者がその値を見るために指定した変数）の定義式から構成される。ただし、出力結果の簡潔化のため、状態遷移により値を変えないデータ変数、値が "0" となる出力変数の定義式は削除する。また、Basic Statement List を構成する変数の定義式中に現れない中間変数の定義式もここから削除する。

演算パターン	処 理
$0 \& f$	f 中に 1, または同一レベルの), または $\neq$ が現われるまで式を消去する。
$0 f$	0 を消去する。
$1 \& f$	1 を消去する。
$1 f$	f 中に同一レベルの), または $\neq$ が現われるまで式を消去する。
$f \& 1$	1 を消去する。
$f \& 0$	f 中に 1 または同一レベルの (, または $=$, または $\neq$ が現われるまで式を消去する。
$f 1$	f 中に同一レベルの (, または $=$, または $\neq$ が現われるまで式を消去する。
$f 0$	0 を消去する。
$f X$	(処理しない)
$f \& X$	(処理しない)

f : 論理式

同一レベルのカッコ: その演算が含まれるカッコ

X : 値の決まらない変数

Fig. 7 Logic operation pattern

式表に移す。一単位時後の制御変数 C^j の総ての値 $|C_1^j|, \dots, |C_r^j|$ が決まったとき、状態 S^{j-1} は状態 $S^j(|C_1^j|, \dots, |C_r^j|)$ に遷移する。

一方、制御変数のうち値の決まらないもの C_i^j ($C_{n1}^j, \dots, C_{nr}^j$) が残ったとき、入力変数とデータ変数を合せて変数 X とすれば、変数 X のうち C_i^j の値を決定するのに必要な変数 $A(a_1, \dots, a_v)$ を検出し、変数 A の取りうる全ての値の組合せについて、 C_i^j の値を計算し、 r 個の遷移状態 S^{jn} ($n=1, \dots, r$) を求める。このとき、論理回路の状態遷移と、それを決める変数 A との関係を示す遷移条件式を次のようにして構成する。すなわち、 u 種類の変数 A の値の組合せ $A_\omega(a_{\omega 1}, \dots, a_{\omega v})$ ($\omega=m_1, \dots, m_u$) に対して制御変数 C_i^j が同一の値 $|C_i^{jn}|$ をとり、同一の状態 S^{jn} に状態遷移が起るとき、遷移条件式 $T_{j-1, jn}$ を次式で定義する。

$$T_{j-1, jn} = \sum_{\omega=m_1}^{m_u} \prod_{i=1}^v |a_{\omega i}| \odot a_{\omega i} \quad (2)$$

ただし、 $|a| \odot a = |a| |a|$, $|a| |a|$: 論理積, Σ : 論理和, Π : 論理積を示す。

(2)式は

$$T_{j-1, jn} = \begin{cases} 1 & (A = A_\omega) \\ 0 & (A \neq A_\omega) \end{cases} \quad (3)$$

となり、変数 A が $|A_\omega|$ の値の組合せをとるときだけ状態 S^{jn} に状態遷移が起ることを示す。(2)式は $a_1, \dots, a_v$ の論理積の論理和として表わされ、変数の個数 v が多いとき膨大な論理式となることがあ

る。したがって、(2)式を簡単化(5章参照)し、解析結果の検討を容易にした。このようにして得られた遷移条件式は、入力変数とデータ変数のみで構成される。

また、単位時間前の状態の代入計算により値の決まらない制御変数が残ったとき、その制御変数の定義式中に含まれる中間変数を遷移条件式の構成に用いると、遷移条件式が簡単になり、構成の迅速化を図ることができる場合もある。すなわち、値の決まらない制御変数 C_i^j の定義式を $f_i(f_1, \dots, f_i)$ とし、変数 A の値の代入計算の結果得られた C_i^j の値を $|C_{i,j}^{j*}|, \dots, |C_{i,j}^{j*}|$ とすると、遷移条件式 $T_{j-1,jn}$ は次式で定義される。

$$T_{j-1,jn} = \prod_{i=1}^i |f_i| \odot f_i = \prod_{i=1}^i |C_{i,j}^{j*}| \odot f_i \quad (4)$$

変数 A の値の組合せにより、すでに現われた状態に遷移が起るとき、遷移条件式は構成する必要はない。(4)式は明らかに(3)式を満たす。

遷移条件式を(2)式で構成(簡単化を行なう)するか(4)式で構成するかは、解析制御パラメータで指定できる。

上述のようにして得られた遷移条件式は、補助論理式表に記憶する。このとき、それぞれの遷移条件式の先頭番地をポインタとして、Fig. 4に示す遷移表に記憶する。ここで、遷移表の行番号は状態表の行番号と一致している。状態 S^{j-1} からの全ての遷移条件式を構成後、レベル3記述に変換し、外部記憶に掃き出す。全ての遷移可能状態が解析されるまで、同様の処理をくり返す。

5. 論理式の簡単化

この種のシステムでは、遷移条件式の構成に多くの時間を要し、その記憶に大きなメモリ領域を必要とするため、大規模回路を扱う上での障害になる可能性がある。

本システムでは、遷移条件式の構成過程において、論理式の簡単化を行なうことにより遷移条件式の記憶領域を縮小し、解析結果を簡潔化し見やすいものとしている。さらに、入力変数、データ変数の数が多いモデルに対しては、遷移条件式の構成時間も短縮される。(2)式で定義される論理式は、入力変数、データ変数の論理積の論理和形式をしており、これらの素項を求め、その素項の論理和形式に(2)式を変形することにより、論理式の簡単化を行なっている。

まず、遷移条件式の記憶エリアを縮小し、簡単化の処理時間を短縮するため(2)式の各論理積部分 $N_i^{(q)}$ を次式のようにコード化する。

$$N_i^{(q)} = \sum_{j=1}^v |a_{ij}| \times 10^{j-1} \quad (5)$$

($i=m_1, \dots, m_u$)

ただし

$$|a_{ij}| = \begin{cases} 0 & (i \text{ 番目の論理積に } a_j \text{ が現われたとき}) \\ 1 & (i \text{ 番目の論理積に } a_j \text{ が現われたとき}) \end{cases}$$

添字(0)は変数消去操作が(0)回なされたことを示す。このとき、

i) $|N_i^{(q)} - N_j^{(q)}| = 10^k$ のとき ($k \in \{0, 1, \dots, v-1\}$). $N_i^{(q)}$ と $N_j^{(q)}$ とは $k+1$ 番目の変数を消去して、一つの論理積 $N_i^{(q+1)}$ に縮約することができる。このとき、消去される変数 a_{ik+1} は $|a_{ik+1}| = 2$ とコード化し、論理積 $N_i^{(q+1)}$ を次式でコード化する。

$$N_{i1}^{(q+1)} = N_{(ij)}^{(q)} + 10^k \quad (6)$$

ただし $N_{(ij)}^{(q)} : N_i^{(q)}, N_j^{(q)}$ のうち大きなもの。

$N_i^{(q)}$ が $N_j^{(q)}$ 以外のものに対して、縮約することができるとき、同様にして $N_{is}^{(q+1)}$ を作成する ($s=1 \dots m$). m は $N_i^{(q)}$ が m 個の論理積と縮約されることを示す。

ii) $|N_i^{(q)} - N_j^{(q)}| \neq 10^k$ のとき ($k=0, \dots, v-1$) $N_i^{(q)}$ と $N_j^{(q)}$ とは変数一つを消去して縮約できない。このとき $N_{is}^{(q+1)} = N_i^{(q)}$ とする。

以上の操作を、消去される変数がなくなるまで行なう。この結果得られた論理積パターンを $\tilde{N}_i = (\tilde{a}_{i1}, \dots, \tilde{a}_{iv})$ ($i=1, \dots, \tilde{m}$) とすると、遷移条件式 $T_{j-1,jn}$ は次式のようになる。

$$T_{j-1,jn} = \sum_{i=1}^{\tilde{m}} \prod_{j=1}^v |\tilde{a}_{ij}| \odot \tilde{a}_{ij} \quad (7)$$

ただし、演算子 $\odot$ は $|\tilde{a}_{ij}| \neq 2$ のとき、演算子 $\odot$ を示し、 $|\tilde{a}_{ij}| = 2$ のとき $\tilde{a}_{ij}$ に関する演算は行わず、変数 $\tilde{a}_{ij}$ を論理積から消去する。

Fig. 8 (次頁参照) (a) には(2)式で構成した遷移条件式の例を示し、Fig. 8 (b) には簡単化された遷移条件式の例を示す。

6. 解析制御パラメータ

解析制御パラメータは、マン-マシンシステムとしての本システムの機能を十分に発揮させるため、解析処理内容を指定するものであり、論理回路の部分的解析を可能にし、回路の使用上からの制限(たとえば、入力信号パターンが限られている場合、実際には遷移の

:L001F-CRES8-TRIG6-GOCIN6-GOCSNICRES8-TRIG6-GJCIN6-GOCSNICRES8TRIG6-GOCIN6-GOCSNI-CRES8-TRIG6;
GCTN6-GOCSNICRES8-TRIG6GOCIN6-GOCSNICRES8TRIG6GOCIN6-GOCSNI-CRES8-TRIG6-GOCIN6GOCSNICRES8-TRIG6-GOCIN6GOCSNI
CRES8TRIG6-GOCIN6GOCSNICRES8-TRIG6GOCIN6GOCSNICRES8-TRIG6GOCIN6GOCSNI-CRES8TRIG6GOCIN6GOCSNICRES8TRIG6GOCSNI
INAGOCSEN). L002(CRES8TRIG6-GOCIN6-GOCSN), L003(-CRES8TRIG6GOCIN6-GOCSN), L004(-CRES8TRIG6-GOCIN6GOCSN)'

(a) 簡單化前

```

      :L001X(,TRIG1CRESIGOCIN&GOCIN), L002(,CRES&TRIG&GOCIN&GOCIN), L003(,CRES&TRIG&GOCIN&GOCIN),
L004(,CRES&TRIG&GOCIN&GOCIN)*

```

(b) 簡單化後

Fig. 8 Example of simplification of transition condition equation

起こらない状態がわかっている場合など)を考慮した解析を可能にするものである。解析制御パラメータには、CONVERSION, EXPANSION, ELIMINATION, INITIAL-STATE, STOP-STATE, MAXCV, TYPE, TRACE, PARAM-END, がある。

“CONVERSION”は、レベル2記述からレベル3記述へ変換するモジュールを指定するものであり、こ

の指定のあるモジュールが多層構造をしているときは、内側の層のモジュールから順次解析処理を行なう。

“EXPANSION”は、展開処理を行なうモジュールを指定するものである。組み込み関数（各種フリップフロップ (Standard module), 各種ゲート (Standard function), 遅延素子）は、全て展開処理される。

“ELIMINASIION” は、入力変数間の関係式、およ

```

EVAL1
ELEMENT COMPUTE(X(10)),RESET(1:4),R(5),Y(10))
LEVEL1
REGISTER OP1,OP2,OP3,OP4;
PASS ADDC(X(10),Y(10));Z(10),CF)
LEVEL2
TERMINAL C(10);
Z=X8Y(C)Y-X8-C(1-X8-Y8C)-X8Y8-C
C(0:5)=X(1:5)Y(1:5)X(1:5)C(1:5)Y(1:5)C(1:5)
C(0)=Y
C=X(C(0))Y(C(0))-C(0)1-X(0)8-Y(0)6C(0)
END;
PASS COUNT(X(10),PM; Y(10))
LEVEL2
TERMINAL Y(10);
Y=X(1)X-T
T(0)=1
T(0:6)=Y(1:6)XPM1-X(1:6)8-YMT(1:6)
END;
L001: D:=C :L002*
L002: R:=1*
L003: R=D :L003*
L004: W:=X :L004*
L005: :L005*
L006: :L006*
L007: OP1:=W(1)
OP2:=W(1)
OP3:=W(2)
OP4:=W(3)
C:=COUNT,Y(4:9)
E:=W(4:9)
COUNT(1:1) :L017*
L007: D:=L008-OP18-OP28-OP38-OP4; L008-OP18-OP28-OP38-OP4; L018-OP18-OP28-OP38-OP4; L019-OP18-OP28-OP38-OP4; L020-OP18-OP28-OP38-OP4; L021-OP18-OP28-OP38-OP4; L022-OP18-OP28-OP38-OP4; L023-OP18-OP28-OP38-OP4; L024-OP18-OP28-OP38-OP4; L025-OP18-OP28-OP38-OP4; L026-OP18-OP28-OP38-OP4; L027-OP18-OP28-OP38-OP4; L028-OP18-OP28-OP38-OP4; L029-OP18-OP28-OP38-OP4; L030-OP18-OP28-OP38-OP4; L031-OP18-OP28-OP38-OP4; L032-OP18-OP28-OP38-OP4; L033-OP18-OP28-OP38-OP4; L034-OP18-OP28-OP38-OP4; L035-OP18-OP28-OP38-OP4; L036-OP18-OP28-OP38-OP4; L037-OP18-OP28-OP38-OP4; L038-OP18-OP28-OP38-OP4; L039-OP18-OP28-OP38-OP4; L040-OP18-OP28-OP38-OP4; L041-OP18-OP28-OP38-OP4; L042-OP18-OP28-OP38-OP4; L043-OP18-OP28-OP38-OP4; L044-OP18-OP28-OP38-OP4; L045-OP18-OP28-OP38-OP4; L046-OP18-OP28-OP38-OP4; L047-OP18-OP28-OP38-OP4; L048-OP18-OP28-OP38-OP4; L049-OP18-OP28-OP38-OP4; L050-OP18-OP28-OP38-OP4; L051-OP18-OP28-OP38-OP4; L052-OP18-OP28-OP38-OP4; L053-OP18-OP28-OP38-OP4; L054-OP18-OP28-OP38-OP4; L055-OP18-OP28-OP38-OP4; L056-OP18-OP28-OP38-OP4; L057-OP18-OP28-OP38-OP4; L058-OP18-OP28-OP38-OP4; L059-OP18-OP28-OP38-OP4; L060-OP18-OP28-OP38-OP4; L061-OP18-OP28-OP38-OP4; L062-OP18-OP28-OP38-OP4; L063-OP18-OP28-OP38-OP4; L064-OP18-OP28-OP38-OP4; L065-OP18-OP28-OP38-OP4; L066-OP18-OP28-OP38-OP4; L067-OP18-OP28-OP38-OP4; L068-OP18-OP28-OP38-OP4; L069-OP18-OP28-OP38-OP4; L070-OP18-OP28-OP38-OP4; L071-OP18-OP28-OP38-OP4; L072-OP18-OP28-OP38-OP4; L073-OP18-OP28-OP38-OP4; L074-OP18-OP28-OP38-OP4; L075-OP18-OP28-OP38-OP4; L076-OP18-OP28-OP38-OP4; L077-OP18-OP28-OP38-OP4; L078-OP18-OP28-OP38-OP4; L079-OP18-OP28-OP38-OP4; L080-OP18-OP28-OP38-OP4; L081-OP18-OP28-OP38-OP4; L082-OP18-OP28-OP38-OP4; L083-OP18-OP28-OP38-OP4; L084-OP18-OP28-OP38-OP4; L085-OP18-OP28-OP38-OP4; L086-OP18-OP28-OP38-OP4; L087-OP18-OP28-OP38-OP4; L088-OP18-OP28-OP38-OP4; L089-OP18-OP28-OP38-OP4; L090-OP18-OP28-OP38-OP4; L091-OP18-OP28-OP38-OP4; L092-OP18-OP28-OP38-OP4; L093-OP18-OP28-OP38-OP4; L094-OP18-OP28-OP38-OP4; L095-OP18-OP28-OP38-OP4; L096-OP18-OP28-OP38-OP4; L097-OP18-OP28-OP38-OP4; L098-OP18-OP28-OP38-OP4; L099-OP18-OP28-OP38-OP4; L100-OP18-OP28-OP38-OP4; L101-OP18-OP28-OP38-OP4; L102-OP18-OP28-OP38-OP4; L103-OP18-OP28-OP38-OP4; L104-OP18-OP28-OP38-OP4; L105-OP18-OP28-OP38-OP4; L106-OP18-OP28-OP38-OP4; L107-OP18-OP28-OP38-OP4; L108-OP18-OP28-OP38-OP4; L109-OP18-OP28-OP38-OP4; L110-OP18-OP28-OP38-OP4; L111-OP18-OP28-OP38-OP4; L112-OP18-OP28-OP38-OP4; L113-OP18-OP28-OP38-OP4; L114-OP18-OP28-OP38-OP4; L115-OP18-OP28-OP38-OP4; L116-OP18-OP28-OP38-OP4; L117-OP18-OP28-OP38-OP4; L118-OP18-OP28-OP38-OP4; L119-OP18-OP28-OP38-OP4; L120-OP18-OP28-OP38-OP4; L121-OP18-OP28-OP38-OP4; L122-OP18-OP28-OP38-OP4; L123-OP18-OP28-OP38-OP4; L124-OP18-OP28-OP38-OP4; L125-OP18-OP28-OP38-OP4; L126-OP18-OP28-OP38-OP4; L127-OP18-OP28-OP38-OP4; L128-OP18-OP28-OP38-OP4; L129-OP18-OP28-OP38-OP4; L130-OP18-OP28-OP38-OP4; L131-OP18-OP28-OP38-OP4; L132-OP18-OP28-OP38-OP4; L133-OP18-OP28-OP38-OP4; L134-OP18-OP28-OP38-OP4; L135-OP18-OP28-OP38-OP4; L136-OP18-OP28-OP38-OP4; L137-OP18-OP28-OP38-OP4; L138-OP18-OP28-OP38-OP4; L139-OP18-OP28-OP38-OP4; L140-OP18-OP28-OP38-OP4; L141-OP18-OP28-OP38-OP4; L142-OP18-OP28-OP38-OP4; L143-OP18-OP28-OP38-OP4; L144-OP18-OP28-OP38-OP4; L145-OP18-OP28-OP38-OP4; L146-OP18-OP28-OP38-OP4; L147-OP18-OP28-OP38-OP4; L148-OP18-OP28-OP38-OP4; L149-OP18-OP28-OP38-OP4; L150-OP18-OP28-OP38-OP4; L151-OP18-OP28-OP38-OP4; L152-OP18-OP28-OP38-OP4; L153-OP18-OP28-OP38-OP4; L154-OP18-OP28-OP38-OP4; L155-OP18-OP28-OP38-OP4; L156-OP18-OP28-OP38-OP4; L157-OP18-OP28-OP38-OP4; L158-OP18-OP28-OP38-OP4; L159-OP18-OP28-OP38-OP4; L160-OP18-OP28-OP38-OP4; L161-OP18-OP28-OP38-OP4; L162-OP18-OP28-OP38-OP4; L163-OP18-OP28-OP38-OP4; L164-OP18-OP28-OP38-OP4; L165-OP18-OP28-OP38-OP4; L166-OP18-OP28-OP38-OP4; L167-OP18-OP28-OP38-OP4; L168-OP18-OP28-OP38-OP4; L169-OP18-OP28-OP38-OP4; L170-OP18-OP28-OP38-OP4; L171-OP18-OP28-OP38-OP4; L172-OP18-OP28-OP38-OP4; L173-OP18-OP28-OP38-OP4; L174-OP18-OP28-OP38-OP4; L175-OP18-OP28-OP38-OP4; L176-OP18-OP28-OP38-OP4; L177-OP18-OP28-OP38-OP4; L178-OP18-OP28-OP38-OP4; L179-OP18-OP28-OP38-OP4; L180-OP18-OP28-OP38-OP4; L181-OP18-OP28-OP38-OP4; L182-OP18-OP28-OP38-OP4; L183-OP18-OP28-OP38-OP4; L184-OP18-OP28-OP38-OP4; L185-OP18-OP28-OP38-OP4; L186-OP18-OP28-OP38-OP4; L187-OP18-OP28-OP38-OP4; L188-OP18-OP28-OP38-OP4; L189-OP18-OP28-OP38-OP4; L190-OP18-OP28-OP38-OP4; L191-OP18-OP28-OP38-OP4; L192-OP18-OP28-OP38-OP4; L193-OP18-OP28-OP38-OP4; L194-OP18-OP28-OP38-OP4; L195-OP18-OP28-OP38-OP4; L196-OP18-OP28-OP38-OP4; L197-OP18-OP28-OP38-OP4; L198-OP18-OP28-OP38-OP4; L199-OP18-OP28-OP38-OP4;
```

Fig. 9 Example of analysis

* $X(10)$ など数個の変数を一まとめにして表わす配列変数は、論理式中に現われる相異なる配列部分を、それぞれ異なる変数と見なして処理する。

り、メモリを有効に利用することができ(たとえばメモリ 70kW では、200 変数(制御変数 64 個)、1500 ゲートのモデルの解析が可能)、また、代入計算、および、遷移条件式の構成の迅速化を図ることができた。さらに、非同期ループの検出、遷移条件式の簡単化による解析結果の簡潔化、各種解析制御パラメータによる解析の制御などが可能であり、計算機やLSIなどの論理設計に役立つものと考ええる。

9. 謝 辞

本システムの開発は、日本電子工業振興協会論理設計自動化専門委員会の仕事の一部として行なった。同委員会言語仕様分科会で御討論いただいた委員諸氏、ならびに、日本電子工業振興協会各位に深甚なる謝意を表する。また、終始有益な御助言、御指導をいただいた沖電気工業(株)ソフトウェア事業部SE部次長山本正隆博士、開発に協力された細谷順一氏、東正明氏に深謝する。

参 考 文 献

- 1) M. A. Breuer: Recent Developments in the Automated Design and Analysis of Digital System, Proc. of IEEE, vol. 60, No. 1, pp. 12~27 (Jan. 1972)
- 2) H. Schorr: Computer-Aided Digital System Design and Analysis Using a Register Transfer Language, IEEE Trans. on EC, vol. EC-13, pp. 730~737 Dec. (1964)
- 3) 岡田康行, 元岡 達: 論理設計言語, 信学会誌, vol. 50, No. 12, pp. 2353~2360, (1967)
- 4) 元岡 達: デジタル計算機の自動論理設計の試み-Logic Design System (LDS), 信学会, EC-68-7, (May 1968)
- 5) T. Moto-Oka, F. Nomizo: Automatic Logic Analysis System, First USA-JAPAN Computer Conference, pp. 397~404, (1972)

(昭和49年11月26日受付)

(昭和50年1月23日再受付)