Computerized Automatic Logic Analysis System:CALAS

Tohru MOTO-OKA*, Nobunori SUGIURA**, Tsutomu SHIINO**
and Atsushi TAKEUCHI**

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 expression 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 a relatively small capacity of storage and executing in a relatively short time, (2) It can treat 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 common system.

1. Introduction

In an automatic logic analysis system, a logical system written in a Boolean expression can be converted to a register transfer representation which is equivalent to sequential control flows, so utilization of this automatic logic analysis system enables to compare a specified logic with the designed logic circuit operation and carry out checking and correction of design errors. By combining an automatic logic synthesis system or manual logic synthesis system and forming a return loop in the design in order to make the design more speedy and correct, the design of a large-scale logic circuit is also expected to become feasible. From this point of view, a basic algorithm for automatic logic analysis and an experimental program RLC 23 prepared chiefly for checking of the principle have been described in the literature 3) on the basis of the logic design language LDS offered previously in the literature 1), 2).

In this paper, the authors will describe the Computerized Automatic Logic Analysis System (CALAS) which has been developed as a practical system applicable to actual logic circuit design. In CALAS, curtailment of the required capacity of storage and shortening of the analysis time have been realized because the effective

This paper first appeared in Japanese in Joho－Shori (Journal of the Information Processing Society of Japan), Vol. 16, No. 7 (1975), pp. 568～575.

* Faculty of Engineering, University of Tokyo
** Software Systems Division, OKI Electric Industry Co., Ltd.

table driven method was used as the internal data representation and also because the sparse matrix method was used as the expression of various matrices. Besides, a variety of processing functions useful for a practical system have been incorporated; including the detection of asynchronous loops, simplification of output logical expressions, expansion of modulized logic circuits etc. In addition, as a means of performing efficiently, the design involving repetition of analysis and revision, and control of analysis by various analysis control parameters have been enabled.

2. Outline of System

This system analyzes input logic circuit models described in the level 2 language (Boolean expression) of LDS language[1] and converts to the representation by the level 3 language (register transfer type format). Fig. 1 shows the outline of the system performance of CALAS.

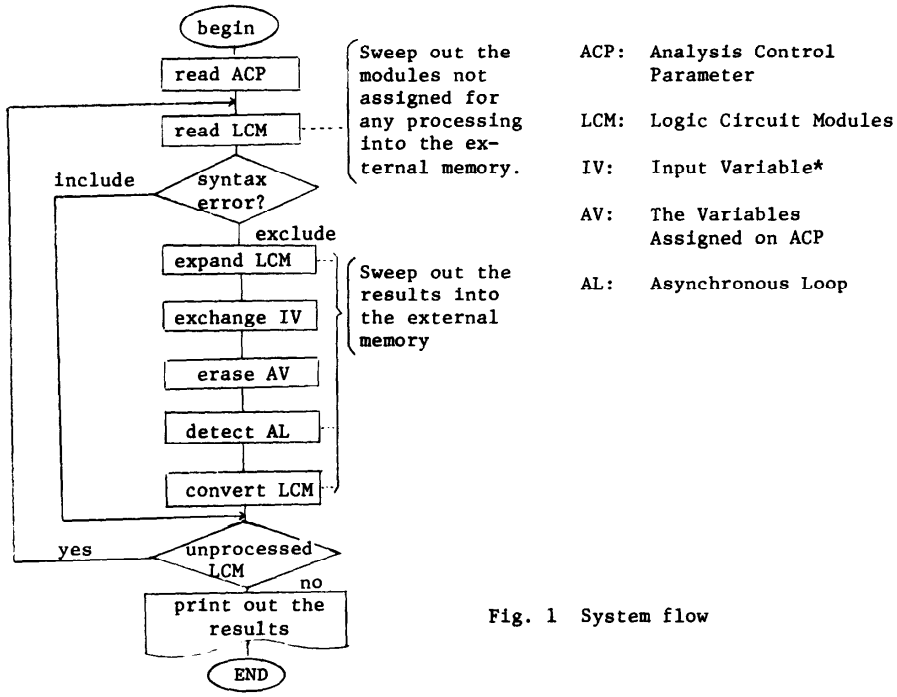

Fig. 1 System flow

3. Expansion Process of LCM

Expansion processing, in which the LCM is demodulized and substituted into the external LCM, is carried out by incorporating the contents of the auxiliary logic equation table into the logic equation table which stores the contents of the module

* The variables which indicate the input terminals of a modulized logic circuit are defined as "input variables" and those which indicate the output terminals as "output variables".

outside the module assigned for expansion.

4. Detection and Modulization of Asynchronous Loops

In actual logic circuits, asynchronous loops are often used as circuit elements having a sort of registering function. In the case of CALAS, asynchronous loops are detected as follows. First, the connection table among the variables is prepared from the definition expressions of the intermediate variables*, and the end variables** are consecutively detected and erased from outside the logic circuit. With respect to the variables remaining unerased, the connection relationship among the variables are traced on the connection table starting from the variable having the most intermediate variables in the right hand definition expression to detect a connection path of variables returning to the original variable in N Steps. It is an asynchronous loop (AL) consisting of N variables. The AL is driven out of the circuit as one module.

5. Level Conversion Processing (Conversion of LCM)

Let $|C_1^j|$ be the value of the control variable*** C_i at jth period. The state $\mathbf{S}^{j-1}(|C_1^{j-1}|,\ldots,|C_p^{j-1}|)$ is substituted in the logic equations. Then, the results of the logical operation performed in the operational part under the state $\mathbf{S}^{j-1}$ (assuming that the logic circuit is divided into the control part determining the values of the control variables, and the rest is operational part) are defined as the basic statement list****. And the results in the control part, when all the values $|C_1^j|, \ldots|C_p^j|$ of the control variable C^j after a unit time have been established, the state $\mathbf{S}^{j-1}$ transfers to the state $\mathbf{S}^j(|C_1^j|, \ldots,|C_p^j|)$; on the other hand, when $\mathbf{C}_s^j(C_{s_1}^j, \ldots, C_{s_t}^j)$ remains undecided in value among the control variables, the variables $\mathbf{A}(a_1, \ldots, a_v)$ necessary for determining the values of $\mathbf{C}_s^j$ are detected

---

* Variables indicating the terminals of a logic circuit other than the input terminals, the output terminals and the terminals of the storage elements.

** End variables are the variables of the output and input terminals and include the variables indicating new input and output terminals appearing when excluding the input and output terminals and the terminals of the storage elements from the original circuit.

*** Control variable indicates the storage elements of this control part of a circuit when dividing it into the control part and operational part, and data variable indicates that of operational part.

**** Basic Statement List is formed of the expressions defining the data variables, output variables, intermediate variables and observation variables which indicate the storage elements of the operational part. However, for simplicity of the output result, the expressions defining those data variables which do not change on state transition and those output variables whose values are "0" after state transition, are excluded. Also, the expressions defining those intermediate variables which do not appear in the definition expressions making up the Basic State List are excluded.

from the variables **X** (the input variables plus data variables***), and the values of C_s^j are computed for all the possible combinations of the values of **A** to obtain r transition states S^{jn} (n = 1, ..., r). In this case, the transitional condition expression showing the relationship between the state transition of the logic circuit and the variable A determining the transition is formed as follows: Namely, when the control values C_s^j take the same value $|C_s^{jn}|$ and transition to the same state S^{jn} takes place for u kinds of combinations of the A value A $(a_{\omega_1}, \ldots, a_{\omega\upsilon})(\omega = m_1, \ldots m_u)$, the transitional condition expression $T_{j-1,\,jn}$ is defined as follows:

$$T_{j-1,jn} = \sum_{\omega=m_1}^{m_u} \prod_{i=1}^{\upsilon} |a_{\omega_1}| \odot a_{\omega i} \qquad (1)$$

, where $|a| \odot a = |a| a \rfloor \, | \neg a | \, \neg a$,

Σ: logical sum, Π: logical product, $\rfloor$: Logical sum, $\neg$: negation

The expression (1) becomes

$$T_{j-1,jn} = \begin{cases} 1 & (A = A_\omega) \\ 0 & (A \neq A_\omega) \end{cases} \qquad (2)$$

, showing that transition to the state S^{jn} takes place only when the variables A take the combination of the values of $|A_\omega|$.

6. Simplification of Logical Expressions

The logical expression defined by the expression (1) takes a form of the logical sum of the logical product of the input variables and data variables. Its simplification has been effected by transforming it to the logical sum of their prime implicant terms.

7. Analysis Control Parameter

Analysis control parameters assign the contents of analysis processing to let the system exhibit its full capability as a man-machine system. They enable partial analysis of a logic circuit(for example, each element module)and such analysis as to consider limitations in the use of a circuit (for example, when the input signal patterns are limited, when the states to which transition does not take place are actually known, etc.).

For example: "CONVERSION" assigns the modules to be converted from the level 2 description to the level 3 description and, when the modules assigned for conversion are of multi-layer structure, processes consecutively from the inner modules.

"EXPANSION" assigns the modules to be expanded. Built-in functions ((various flip-flops (standard module), various gates (standard function) and delay element)) can all be expanded.

"INITIAL-STATE" assigns the initial state of a logical circuit. The control variables and their values are defined by this card.

"STOP-STATE" assigns the transition inhibited states (names of the states, names of the control variables) which do not require analysis of future transition. The transition states from one of the STOP STATES and the transitional condition expressions for them are computed, but the transitions of the subsequent states are not analyzed. etc.

8. Example of Analysis

An example of analysis of a simple computer circuit with 10 commands is shown. Fig. 2 is input data described in the level 2, where COMPUTE is the name of a module assigned for CONVERSION; and ADDER and COUNT are respectively the names of the modules not assigned for any processing. Fig. 3 shows the result of analysis. This analysis required an execution time of 56.27 sec (UNIVAC-1106).

9. Conclusion

In this paper, we have described the system outline of CALAS. By using an internal representation of the table structure where the circuit model descriptions are classified and given according to the content, this system has succeeded in efficient utilization of memory ((for example, models of 200 variables (64 control variables) and 1500 gates can be analyzed with 70 kW)) and speeding up to substitutional computation and composition of the transitional condition expressions. Besides, it enables detection of asynchronous loops, brief output of the analytical results by simplification of the transitional condition expressions and control of analysis by various analysis control parameters. The system, therefore, is useful for logic design of computers and LSI in our opinion.

10. Reference

1) Yasuyuki Okada and Tohru Motooka: Logic Design Language, Shingakukai-shi Vol. 50, No. 12, pp. 2353 - 2360 (1967)

2) Tohru Motooka: Trial of Automatic Logic Design of Digital Computers - Logic Design System (LDS), Shingakukai, EC-68-7 (May 1968)

3) T. Motooka, F. Nomizo: Automatic Logic Analysis System, First USA-JAPAN Computer Conference, pp. 397 - 404, (1972)

```
CONVERSION COMPUTE ;
INITIAL_STATE C1=0,C2=0,C3=0,C4=0,C5=0,C6=0;
MAXCYC=  ;
TYPE=0 ;
PARAM_END


LEVEL2
 ELEMENT COMPUTE(X(10)),RESET;RI,WI,R(6),Y(10))
 LEVEL2
 PASS ADDER(X(10),Y(10);Z(10),OF)
   LEVEL2
    TERMINAL C(10);
    Z=X&Y&C(X&_Y&_C(_X&_Y&C(_X&Y&_C  •
    C(0:8)=X(1:9)&Y(1:9)(X(1:9)&C(1:9)(Y(1:9)&C(1:9)  •
    C(9)=0   •
    OF=X(0)&Y(0)&_C(0)(_X(0)&_Y(0)&C(0))
   END;
  PASS  COUNT(X(10),PM; Y(10))
   LEVEL2
    TERMINAL T(10);
    Y=_X&T(X&_T   •
    T(9)=1   •
    T(0:8)=(X(1:9)&PM(_X(1:9)&_PM)&T(1:9))
  END;
  TERMINAL A(10),C(6),D(6),E(6),M(10),OP1,OP2,OP3,OP4 ,
  TERMINAL C1,C2,C3,C4,C5,C6,T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,T11,
  TERMINAL T12,T13,T14,T15,T16,T17,T18,T19,T20,T21,T22,T23,T24,
  TERMINAL T25,T26,T27,T28,T29,T30,T31,T32,T33,T34;
  T1=_C1&_C2&_C3&_C4&_C5&_C6   •
  T2=_C1&_C2&_C3&_C4&_C5& C6   •
  T3=_C1&_C2&_C3&_C4& C5&_C6   •
  T4=_C1&_C2&_C3&_C4& C5& C6   •
  T5=_C1&_C2&_C3& C4&_C5& C6   •
  T6=_C1&_C2&_C3 C4& C5&_C6   •
  T7=_C1&_C2&_C3 C4& C5& C6   •
  T8=_C1&_C2 C3&_C4&_C5&_C6   •
  T9=_C1&_C2 C3&_C4&_C5 C6   •
  T10=_C1&_C2 C3&_C4 C5&_C6   •
  T11=_C1&_C2 C3&_C4 C5 C6   •
  T12=_C1&_C2 C3 C4&_C5&_C6   •
  T13=_C1&_C2&_C3 C4&_C5&_C6   •
  T14=_C1&_C2 C3 C4&_C5 C6   •
  T15=_C1&_C2 C3 C4 C5&_C6   •
  T16=_C1&_C2 C3 C4 C5 C6   •
  T17=_C1 C2&_C3&_C4&_C5 C6   •
  T18=_C1 C2&_C3&_C4 C5&_C6   •
  T19=_C1 C2&_C3&_C4 C5 C6   •
  T20=_C1 C2&_C3 C4&_C5&_C6   •
  T21=_C1 C2&_C3 C4&_C5 C6   •
  T22=_C1 C2&_C3 C4 C5&_C6   •
  T23=_C1 C2&_C3 C4 C5 C6   •
  T24=_C1 C2 C3&_C4&_C5&_C6   •
  T25=_C1 C2 C3&_C4&_C5 C6   •
  T26=_C1 C2 C3&_C4 C5&_C6   •
  T27=_C1 C2 C3&_C4 C5 C6   •
  T28=_C1 C2 C3 C4&_C5&_C6   •
  T29=_C1 C2 C3 C4&_C5 C6   •
  T30=_C1 C2&_C3&_C4&_C5&_C6   •
  T31= C1 C2&_C3&_C4&_C5&_C6   •
  T32= C1&_C2&_C3&_C4&_C5&_C6   •
  T33= C1&_C2&_C3&_C4&_C5 C6   •
  T34= C1&_C2&_C3&_C4 C5&_C6   •
```

```
  C1:=T7&(_OP1&_OP2&OP3&_OP4(_OP1&OP2&_OP4 [illegible]
      [illegible] •
  C2:=T7&(OP1&_OP2&_OP3&_OP4(_OP1&OP2&C5 [illegible]
      [illegible]
      [illegible] •
  C3:=T7&(_OP1&OP4)_OP1&_OP2&_OP3))T4|T5|T6|T8|T14|T15|T25|
      T26|T27|T3)   •
  C4:=T4|T5|T6|T7&(_OP1&_OP3&OP4 [illegible]
      [illegible] ADD R [illegible] RESET   •
  C5:=T[illegible]
      OP2&_OP3&_OP4)|T21|T22|T25|T26|T33&_A(0)   •
  C6:=T[illegible]
      _OP1&OP4|OP1&_OP2&_OP3&_OP4)|T15|T16|T20|T22|T26|T30   •
  D:=C&T1|C&T2|D&_(T1|T7)   •
  RI=T2|T8|T16|T19   •
    R=D&(T2|T8|T16|T20|T19)   •
  M:=M&(T5|T3|T15|T20)|A&T25|M&_(T5|T8|T15|T23|T25)   •
  OP1:=M(0)&T6|OP1&_T6   •
  OP2:=M(1)&T6|OP2&_T6   •
  OP3:=M(2)&T6|OP3&_T6   •
  OP4:=M(3)&T6|OP4&_T6   •
  E:=M(4:9)&T6|COUNT_Y(4:9)&(T10|T12|[illegible])   •
  C:=COUNT_Y(4:9)&T6|C&_(T6|T16)   •
  ADDER(A&(T1|T19),A&T1|COUNT_Y&T14)   •
  COUNT(M[illegible]   •
  A:=ADDER_Z&(T12|T4)|[illegible]
      A&_(T12|T14|T25|T26|T3|T32)   •
  WI=T26   •
  Y=M&T25
 END;
END
```

FIG 2 EXAMPLE OF ANALYSIS. INPUT DATA

```
  LEVEL3
   ELEMENT COMPUTE(X(10)),RESET;RI,WI,R(6),Y(10))
    LEVEL3
     REGISTER OP1,OP2,OP3,OP4 )
      PASS ADDER(X(10),Y(10);Z(10),OF)
       LEVEL2
        TERMINAL C(10);
        Z=X&Y&C(X&_Y&_C(_X&_Y&C(_X&Y&_C  •
        C(0:8)=X(1:9)&Y(1:9)(X(1:9)&C(1:9)(Y(1:9)&C(1:9))  •
        C(9)=0  •
        OF=X(0)&Y(0)&_C(0)(_X(0)&_Y(0)&C(0)
       END
      PASS_COUNT(X(10),PM;Y(10))
       LEVEL2
        TERMINAL T(10);
        Y=_X&T(X&_T   •
        T(9)=1        •
        T(0:8)=(X(1:9)&PM(_X(1:9)&_PM)&T(1:9))
       END;
     L001:   D:=C    :L002'
     L002:   RI=1•
             R=D    :L003'
     L003:   M:=X   :L004'
     L004:   :L005'
     L005:   :L006'
     L006:   OP1:=M(0)•
             OP2:=M(1)•
             OP3:=M(2)•
             OP4:=M(3)•
             C:=COUNT_Y(4:9)•
             E:=M(4:9)•
             COUNT(LIST2,1)    :L007'
     L007:   D:=E    :L008(_OP1&_OP2&_OP3&_OP4),L009
             (OP1&_OP2&_OP3&_OP4),L010(_OP1&OP2&_OP
             3&_OP4),L011(OP1&OP2|OP1&OP3|OP1&OP4),
             L012(_OP1&OP2&OP3&_OP4),L013(_OP1&OP2&
             OP3&_OP4),L014(_OP1&_OP2&_OP3&OP4),L01
             5(_OP1&OP2&_OP3&OP4),L016(_OP1&OP2&_OP
             3&OP4),L017(_OP1&OP2&OP3&OP4)'
     L008:   RI=1•
             R=D    :L018'
     L009:   WI=1•
             R=D    :L019'
     L010:   C:=F    :L002'
     L011:   :L001(OTHERWISE),L011(_RESET)'
     L012:   :L001(OTHERWISE),L010(_A(0))'
     L013:   :L001(_C(0)&_C(1)&E(2)&_E(3)&_E(4)&_E
             (5)),L020(OTHERWISE)'
     L014    RI=1•
             R=D    :L021'
     L015:   :L001(_E(0)&_E(1)&_E(2)&_E(3)&_E(4)&_E
             (5)),L022(OTHERWISE)'
     L016:   M:=A    :L023'
     L017:   A:=0    :L001'
     L018:   M:=X    :L026'
     L019:   M:=X    :L025'
     L020:   E:=COUNT_Y(4:9)•
             A:=LIST3•
             COUNT(LIST1,0)    :L013'
     L021:   M:=X    :L026'
     L022:   E:=COUNT_Y(4:9)•
             A:=LIST4•
             COUNT(LIST1,0)    :L015'
     L023:   WI=1•
             R=D•
             Y=M    :L027'
     L024:   :L028'
     L025:   :L029'
     L026:   :L030'
     L027:   :L031'
     L028:   :L032'
     L029:   :L033'
     L030:   :L036'
     L031:   :L001'
     L032:   A:=ADDER_Z•
             ADDER(A,M)    :L001(_ADDER.OF),L011(OTH
             ERWISE)'
     L033:   A:=M    :L001'
     L034:   A:=ADDER_Z•
             ADDER(A,COUNT.Y)•
             COUNT(_M,1)    :L001(_ADDER.OF),L011(OT
             HERWISE)
    END;
  END
```

FIG 3 RESULTS OF ANALYSIS