Formula Manipulation System for Boolean Functions (BALOC-2)

YUUJI YOSHIDA* AND TERUO FUKUMURA*

1. *Introduction*

For the aim of automated formula manipulation of Boolean functions, we developed the Formula Manipulation System for Boolean Functions (BALOC-2; Boolean Algorithm Oriented Compiler-2). Boolean Functions cannot be manipulated by the existing formula manipulation systems for polynomials, rational functions or analytical functions. This system utilizes many features of computers in binary operations and so Boolean functions are processed with high speed and efficiency. The Boolean functions are expressed in the canonical form of ring sum, which are uniquely defined for the functions, and stored in memory partitioned into pages. The system consists of BALOC-2 language, compiler and utility routines for execution. All of these are based upon FORTRAN and a few bitwise operations are implemented by assembler language.

2. *Specification of BALOC-2 Language*

BALOC-2 language is embedded in FORTRAN. This method for designing new languages is the most familiar one and the resulted language is very powerful in both formula manipulations and numerical calculations. Some new statements are added to FORTRAN for formula manipulations and relational operations. In the following sections, only these statements and the related matters are described.

2.1 *Two New Types of Identifiers*

Two types of identifiers, BASIC and FORMAL, are newly introduced, corresponding to Boolean variables and Boolean functions respectively. To declare these identifiers, two statements are provided;

(1) BASIC list,

(2) FORMAL list.

Arrays of these types up to one dimension can be also declared by these statements.

2.2 *Descriptions of Boolean Functions*

Boolean functions in BALOC-2 are described by using constants (0 or 1),

This paper first appeared in Japanese in Joho-Shori (Journal of the Information Processing Society of Japan), Vol. 12, No. 7 (1971), pp. 406~413.

* Faculty of Engineering, Nagoya University

BASIC variables, FORMAL variables and several Boolean operations. Boolean operations are .EQV. (equivalence),+(ring sum), .IMP. (implication), /(inclusive or), *(logical product) and' (negation) The precedence relation among them is in this order. For example, the expression

$$h(x, y, z)=f(x, y, z)\vee\overline{g(x, y, z)}\vee x\cdot\bar{y}\cdot\bar{z}$$

is described in BALOC-2 as Fig. 1, where the third statement is assignment statement.

```
FORMAL  F, G, H
BASIC   X, Y, Z
H=F/'G/X*'Y*'Z
```

Fig. 1. Boolean functions in BALOC-2

2.3 *Additional Statements*

(1) FSUBST $f_1, (b_1, g_1), \cdots, (b_n, g_n), f_2$

Boolean function g_1 is substituted into Boolean variable b_1 in Boolean function f_1, and g_2 is substituted into b_2, and so on. The result of the successive substitutions is substituted into Boolean function f_2.

(2) VSUBST f_1, V, n, f_2

V is one dimensional integer array with length n, each element of which takes one of 0, 1, or 2. $V(1), \cdots, V(n)$ are substituted into n Boolean variables of f_1 in the order specified by the variable list of BASIC statement and the result is substituted into f_2. If $V(i)$ equals to 2, then the substitution is omitted.

(3) NEGATE $f_1, (b_1, b_2, \cdots, b_n), f_2$

The Boolean variables $b_1, b_2, \cdots, b_n$ in f_1 are replaced by those negated and the result is substituted into f_2.

(4) PERMUTE f_1, b_1, b_2, f_2

The Boolean variables b_1 and b_2 in f_1 are mutually permuted and the result is substituted into f_2.

(5) TEST f_1, v_1

If f_1 is equivalent to 0 (false), then v_1 is set to 0. If f_1 is equivalent to 1 (true), then v_1 is set to 1. Otherwise, v_1 is set to 2.

(6) EQUAL f_1, f_2, v_1

If f_1 is equivalent to f_2, then $v_1=1$, otherwise, $v_1=0$.

(7) DEPEND f_1, b_1, v_1

If v_1 is not redundant in f_1, then $v_1=1$, otherwise $v_1=0$.

(8) ERASE $f_1, f_2, \cdots, f_n$

The values of $f_1, f_2, \cdots, f_n$ are erased and the storages for their representation are freed.

3. *BALOC-2 Compiler*

BALOC-2 compiler is the FORTRAN program (about 1300 statements) and

translates the BALOC-2 source program to FORTRAN program. In addition to the translation of statements described above, the expression number and bit position in word are assigned to FORMAL and BASIC variables, respectively. The numbering of variables is subject to the variable list in the corresponding declaration statements. An example of translation is shown in Fig. 4.

4. *Internal Representation of Boolean Functions and Storage Management*

There are two essential problems in the implementation of formula manipulation system. One is how to represent formula in storage and the other is how to allocate storage to formula dynamically. The solutions of these problems have significant influences on the performance of the system. Our solution is as follows.

4.1 *Representation of Boolean Functions in Storage*

In BALOC-2 system Boolean function is represented in the canonical form of ring sum. By this form a Boolean function of n variables $f(x_1, x_2, \cdots, x_n)$ is described as below;

$$f(x_1, x_2, \cdots, x_n)=\sum_{i=1}^{N} x_1^{v_{i1}} \cdot x_2^{v_{i2}} \cdot \cdots \cdot x_n^{v_{in}} \tag{1}$$

where $\sum$: summation by ring sum,

N: number of term,

v_{ij}: if i-th term contains j-th variable, then $v_{ij}=1$, otherwise $v_{ij}=0$.

$$x_j^{v_{ij}}=\begin{cases}1 & (v_{ij}=0)\\ x_j & (v_{ij}=1)\end{cases} \tag{2}$$

It can be seen that the function is determined uniquely by the list of index $\{v_{ij}\}$ in (1). If vector $(v_{i1}, v_{i2}, \cdots, v_{in})$ is stored as n bits string in one word of memory, a Boolean function in (1) can be stored in N words. This representation makes it possible to internally process various operations on Boolean functions at high speed by the effective use of machine instructions for bitwise logical operations. For example, logical product of two Boolean functions, $f^{(1)}(x_1, x_2, \cdots, x_n)$ and $f^{(2)}(x_1, x_2, \cdots, x_n)$, whose internal form are $\{v_{ij}^{(1)}\}$ and $\{v_{ij}^{(2)}\}$ respectively, is given as follows;

$$\begin{aligned} & f^{(1)}(x_1, \cdots, x_n) \cdot f^{(2)}(x_1, \cdots, x_n) \\ &= \left(\sum_{i=1}^{N_1} x_1^{v_{i1}^{(1)}} \cdot \cdots \cdot x_n^{v_{in}^{(1)}}\right) \cdot \left(\sum_{j=1}^{N_2} x_1^{v_{j1}^{(2)}} \cdot \cdots \cdot x_n^{v_{jn}^{(2)}}\right) \\ &= \sum_i \sum_j x_1^{(v_{i1}^{(1)} \vee v_{j1}^{(2)})} \cdot \cdots \cdot x_n^{(v_{in}^{(1)} \vee v_{jn}^{(2)})} \end{aligned} \tag{3}$$

The list of indices $((v_{i1}^{(1)} \vee v_{j1}^{(2)}), \cdots, (v_{in}^{(1)} \vee v_{jn}^{(2)}))$ in (3) can be computed easily by logical or instruction. Furthermore, owing to the uniqueness of this form some statements described in 2.3 can be executed correctly.

Although this form has many advantages mentioned above, it has some disadvantage at the same time. That is, it is not familiar with us, and it cannot

be converted so easily to other canonical form. The latter is essential to read-in/print-out of Boolean functions.

4.2 *Storage Management*

The size of list $\{v_{ij}\}$ changes dynamically during the processing, so available storages must be allocated to the lists dynamically. In BALOC-2 available storages are partitioned into groups of contiguous words (this group is called 'page') and storages are allocated in units of page. Fig. 2 shows the scheme concretely.

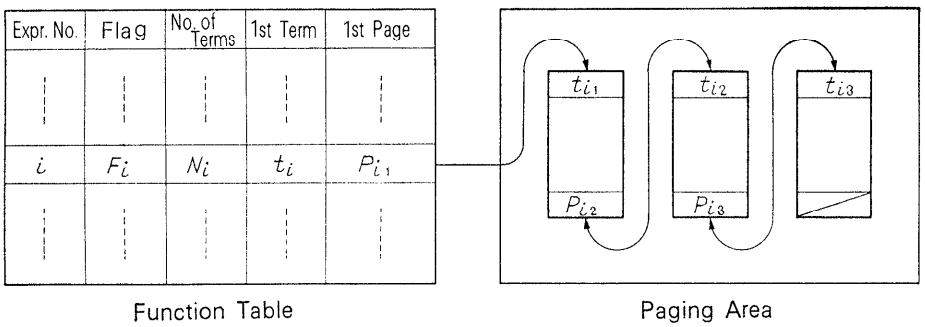

Fig. 2 Boolean function in storage

In the figure F_i denotes whether i-th expression has been defined. If so, $F_i=1$, otherwise $F_i=0$. N_i denotes the number of terms of i-th expression and $N_i=0$ means false of the expression. When $N_i=1$, no page is used and t_i has a meaning only in this case, denoting a word containing a single term. Unused pages are monitored by page controller and in the case of need they are allocated and freed pages are added to them.

Operations upon Boolean functions are performed through the process as shown in Fig. 3. 'ACC area' contains relatively much contiguous words and

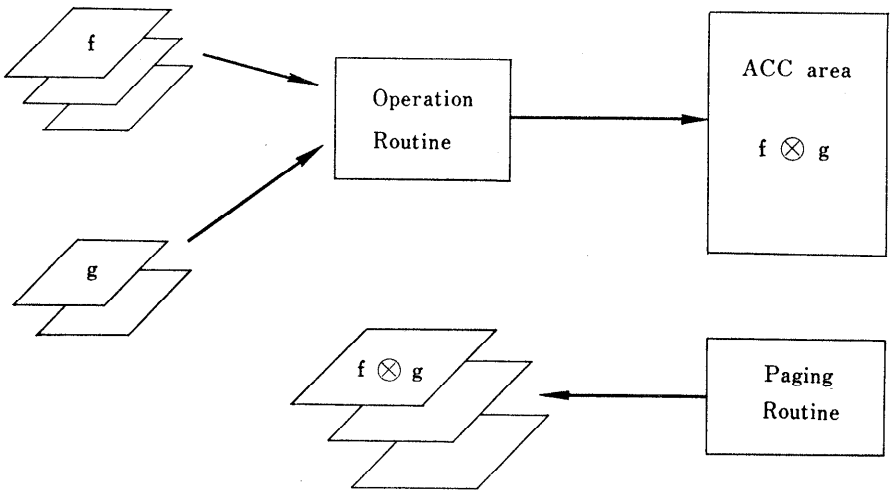

Fig. 3 Process of operations on Boolean functions

the result computed by the routine for the operation is once written into this area and then stored to paged memory by paging routine. In the case of our experimental system, a page consists of 100 words and there are 200 pages in all. ACC area contains 2000 words.

5. *Illustrative Example*

In this section one simple example of application is given to show various processings performed by BALOC-2. In this case a Boolean function is read in and its dual function is computed. After both functions are printed out self-duality of the function is checked by EQUAL statement and the result is printed. The input function is

$$x_1x_2 \vee x_1\bar{x}_3\bar{x}_4 \vee x_2x_3$$
$$= x_1 \oplus x_1x_3 \oplus x_1x_4 \oplus x_2x_3 \oplus x_1x_2x_4 \oplus x_1x_3x_4 \oplus x_1x_2x_3x_4$$

Fig. 4 shows input data, source program, object program and printed results. From the printed result we see the dual function is

$$x_2 \oplus x_1x_3 \oplus x_2x_3x_4 \oplus x_1x_2x_3 \oplus x_1x_2x_3x_4$$
$$= x_1x_2 \vee x_1x_3 \vee x_2\bar{x}_3 \vee x_2\bar{x}_4,$$

indicating non-self-duality of the function as shown in the result. The printed format of Boolean function is almost as same as that in storage. This is slightly disadvantageous.

```
7        4    (No. of terms and var.)
1  0  0  0
1  0  1  0
1  0  0  1
0  1  1  0    (Product terms)
1  1  0  1
1  0  1  1
1  1  1  1
```

Fig. 4a Input format

```
BALOC-2 COMPILED LIST
SOURCE PROGRAM LIST
CARD NO.   LABEL      SOURCE STATEMENT
     1     B          FORMAL F, FDUAL
     2     B          BASIC X1, X2, X3, X4
     3     B          INPUT (F)
     4     B          NEGATE F, (X1, X2, X3, X4), FDUAL
     5     B          FDUAL='FDUAL
     6     B          EQUAL F, FDUAL, INDIC
     7     B          OUTPUT (F, FDUAL)
     8                IF (INDIC .EQ. 1) GO TO 1000
     9                WRITE (6,100)
    10           100  FORMAT (/1H, 19HF IS NOT SELF-DUAL./)
    11                STOP
    12          1000  WRITE (6,200)
    13           200  FORMAT (/1H, 15HF IS SELF-DUAL./)
    14                STOP
    15                END
```

Fig. 4b Source program

```
OBJECT PROGRAM LIST
    CALL EINP (1)
    CALL ENEG (1, 1, 100)
    CALL ENEG (100, 2, 100)
    CALL ENEG (100, 3, 100)
    CALL ENEG (100, 4, 2)
    CALL ENOT (2, 3, 100)
    CALL ETRF (100, 3, 2)
    CALL EEQU (1, 2, INDIC)
    CALL EOUT (1)
    CALL EOUT (2)
    IF (INDIC .EQ. 1) GO TO 1000
    WRITE (6, 100)
100 FORMAT (/1H, 19HF IS NOT SELF-DUAL./)
    STOP
1000 WRITE (6, 200)
200 FORMAT (/1H, 15HF IS SELF-DUAL./)
    STOP
    END
```

Fig. 4c Object program

VARIABLE NAME LIST

NAME	TYPE	EXP. NO. OR BIT POSITION
F	F	1
FDUAL	F	2
X1	B	1
X2	B	2
X3	B	3
X4	B	4

Fig. 4d Variable name list

```
EXPRESSION NO. 1       NUMBER OF TERMS=7
  1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
EXPRESSION NO. 2       NUMBER OF TERMS=5
  1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
F IS NOT SELF-DUAL.
```

Fig. 4e Computed results

6. *Concluding Remarks*

BALOC-2 system is similar to ALPAK system [1] in respect of internal representation of formula and similar to FORMAC system [2] in respect of language specification. There remain some important problems in BALOC-2 as partly mentioned in previous sections.

References

[1] W. G. Brown: The ALPAK system for nonnumerical algebra on a digital computer-I, BSTJ, 42, 5 (1963).

[2] J. E. Sammet: An overall view of FORMAC, IBM Systems Development Div., TR 00. 1367 (1965).

[3] Y. Yoshida and T. Fukumura: Formula Manipulation System for Boolean Functions based upon Symbol Manipulations (BALOC-3), J. IPSJ, 14, 1 (1973).

[4] Y. Yoshida and T. Fukumura: Symbol Manipulation System based upon FORTRAN (COSMOS-2), in this issue.