A LANGUAGE FOR BOOLEAN FUNCTION REPRESENTATION AND MANIPULATION

Iong Chen and B. D. Carroll
Electrical Engineering Department
Auburn University
Auburn, Alabama 36830

ABSTRACT

Boolean algebra is used extensively in the analysis and design of digital logic circuits, in the generation of test patterns for logic circuits, and in numerous other practical applications. Hand calculations involving Boolean equations become impractical when the equations involve a large number of variables or when the number of equations is large. Computerized processing of Boolean equations can significantly extend the range of problems that can be solved using Boolean algebra.

A language ABAL (Auburn Boolean Algebra Language) is described in this paper that permits machine representation and manipulation of Boolean functions. Functions may be specified in algebraic form or as lists of minterms or maxterms. Types of operations available in the language include functional form changes, simplification rules, prime implicant or prime implicate generation, functional minimization, functional combinations using Boolean operators, and truth table generation. ABAL is written in BASIC-PLUS for execution on a DEC PDP 11/40 RSTS/E System.

INTRODUCTION

Boolean algebra is used extensively in the analysis and design of digital logic circuits, in the generation of test patterns for logic circuits, and in numerous other practical applications. Fundamental principles of Boolean algebra and logic design are routinely taught to undergraduates in most modern electrical engineering curricula. However, the range of problems to which Boolean algebra can be applied by engineering students and engineering practitioners is severely limited when traditional manipulation and evaluation techniques are used. That is to say, hand calculations using Boolean algebra become impractical when a large number of variables or a large number of equations are involved. The advent of large scale integration (LSI) has made the problem more acute since logic circuits equivalent to several thousand gates can now be manufactured on a single chip. Such logic circuits are beyond the scope of Boolean algebra representation and analysis when only traditional methods are employed.

Computerized processing of Boolean equations can significantly extend the range of problems that can be solved using Boolean algebra. Many programs have been written for minimizing Boolean functions, for ANDing or ORing functions, and for simulating logic circuits.[1,2,3] These programs are most frequently written in some high-level language such as FORTRAN, APL, or BASIC. Assembly language subroutines are sometimes used to perform the basic Boolean operations.

High-level languages such as FORTRAN, APL, or BASIC are not, however, well suited for processing Boolean equations since Boolean operations are not available in the languages. Also missing are the means for representing Boolean functions. A language specifically designed to handle Boolean algebra and Boolean equations is needed. Auburn Boolean Algebra Language (ABAL) is such a language and will be described in this paper.

APPLICATIONS OF ABAL

ABAL was originally envisioned as a means for providing machine manipulation of Boolean equations as required by the multiple-fault test pattern generation procedure of Bossen and Hong.[4] However, numerous other uses of the language have since been identified. For example, ABAL can be used to develop drills in the fundamentals of Boolean algebra for the beginning logic design course. Other applications of ABAL include logic design problems such as the simplification of logic functions to minimize the gate count and the verification of the function realized by a logic circuit design. Finally, ABAL can be utilized in any problem where Boolean equations are used to represent a physical or an abstract structure.

ILLUSTRATION OF ABAL

The listing of a typical ABAL program is given in Figure 1. Output produced by the program is shown in Figure 2.

DESCRIPTION OF ABAL COMPILER

Figure 3 shows the subprogram structure of the ABAL compiler. The function of each subprogram is given below.

BOOLE - Creates a file for ABAL source program
TXEDIT - Orders the statements of the source program
SCANER - Analyzes the syntax and builds the symbol table
SMATIC - Checks the semantic correctness and generates object code
COPILE - Interprets the object code
MINMAX - Generates minterm list, maxterm list, don't care list, or truth table
PI - Generates prime implicants, prime implicates, or minimal SOP or POS

SOPPOS - Generates SOP or POS expansion
COMP - Applies complement rule
UNIQUE - Applies uniqueness theorem
ABSORB - Applies absorption theorem
CONSEN - Applies concensus theorem

Testing of the ABAL compiler has been accomplished using the strategies listed below. While this approach is an effective way of identifying errors, it does not provide for a proof of correctness of the compiler.

1. Exercise every type of legal statement in the language
2. Exercise every subprogram in the compiler
3. Exercise every subroutine in each subprogram
4. Exercise every keyword in the language

CONCLUSION

ABAL provides an effective tool for performing calculations involving Boolean algebra. However, several desirable features are missing from the current version and are planned for future versions. These features include conditional and unconditional branching, subroutines, and diagnostic messages. It is also desirable to increase the limits on the number of variables, the length of expressions, and the

Features of ABAL are described in the following section. Later sections contain an example ABAL program and output and a discussion of ABAL applications. The ABAL compiler is described in another section. Conclusions based on experience with ABAL are presented in the final section.

DESCRIPTION OF ABAL

ABAL is an interpretive language that permits machine representation and manipulation of Boolean functions. Functions may be specified in the language in either algebraic form or as lists of minterms or maxterms. Don't care conditions may be combined using Boolean operators, simplification rules can be applied to functions in a selective manner, functional form changes may be specified, functions may be minimized, prime implicants or prime implicates may be generated, and truth tables may be produced by employing features of the language. The current version of ABAL is written in BASIC-PLUS for running on a PDP 11/40 RSTS/E time sharing system.

The following statement types are available in ABAL.

1. DEF - Used for function definitions
2. WRITE - Used for output specification
3. Assignment - Used for specifying functional combinations or for specifying constants
4. END - Used for terminating a program

Operators provided in ABAL are given below.

1. AND(*)
2. OR(+)
3. EXCLUSIVE-OR(@)
4. NOT(')

In addition, the following generators are provided in ABAL for applying frequently needed Boolean algebra theorems and procedures to previously specified functions.

1. SOP - Generates the sum-of-products expansion of a function.
2. POS - Generates the product-of-sums expansion of a function.
3. MSP - Generates a minimal sum-of-products form of a function.
4. MPS - Generates a minimal product-of-sums form of a function.
5. PICAN - Generates the prime implicants of a function.
6. PICAT - Generates the prime implicates of a function.
7. MIN - Generates the minterms of a function.
8. MAX - Generates the maxterms of a function.
9. TRUTH - Generates the truth table of a function.
10. COM - Removes the product terms of the form $_{xx}$' from a function.
11. UNI - Removes duplicate terms from a function.
12. ABS - Applies the absorption theorem to a function.
13. CON - Applies the consensus theorem to a function.

ABAL source program may contain as many as 300 statements. However, the sum of the number of DEF statements and the number of assignment statements cannot exceed 50. Also, the number of variables in a source program cannot exceed 24. Boolean expressions cannot exceed 256 characters in length. These limitations are imposed by the current implementation of ABAL and not by the language itself.

number of statements. Future work on ABAL will also be concerned with the implementation language and with the feasibility of direct execution in hardware of ABAL.

REFERENCES

1. S. G. Shiva and H. T. Nagle, Jr., "Computer Aided Design of Digital Networks," Electronic Design, Vol. 22, 1974.

2. S. A. Szygenda and E. W. Thompson, "Digital Logic Simualtion in a Time-Based, Table-Driven Environment - Parts 1 and 2," Computer, Vol. 8, No. 3, March 1975, pp. 24-36, 38-49.

3. B. D. Carroll, "A Simulator for Undergraduate Logic Courses," Computers in Education Division of ASEE Transactions, Vol. VII, No. 5, May 1975, pp. 57-72.

4. D. C. Bossen and S. J. Hong, "Cause-Effect Analysis for Multiple Fault Detection in Combinational Networks," IEEE-TC, Vol. C-20, No. 11, Nov. 1971, pp. 1252-1257.

```
10      DEF BFA(X,Y,Z)=((X@Y)'*Y)+(Y'@Z)
20      WRITE SOP BFA(X,Y,Z)
30      WRITE POS BFA
40      WRITE COMPLEMENTSP BFA
50      WRITE ABSORPTIONSP BFA
60      WRITE CONSENSUSSP BFA
70      WRITE MINTG BFA
80      WRITE MAXTG BFA
90      DEF BFB(X,Y,Z)=MIN(0,1,3,4,7)+DONT(2,6)
100     BFC(X,Y,Z)=BFA(X,Y,Z)+BFB(X,Y,Z)
110     WRITE TRUTH BFC(X,Y,Z)
120     WRITE MINTG BFC
130     WRITE MAXTG BFC
140     WRITE PICAN BFC
150     WRITE PICAT BFC
160     WRITE MSP BFC
170     WRITE MPS BFC
180     END
```

Figure 1

```
SUM OF PRODUCT FORM OF  BFA  IS  :
Y'*Z' + Y*Z + X'*X*Y + Y*X + X'*Y'*Y + Y*Y'

PRODUCT OF SUM FORM OF BFA  IS  :
Y+Y'+Z * X'+X+Y'+Z * Y'+X+Z * X'+Y+Y'+Z * Y+Z' * X'+X+Y+Z' * Y'+X+Y+Z' *
 X'+Y+Z' * Y'+Y+Z'

BOOLEAN FUNCTION BFA AFTER APPLYING
COMPLEMENT THEOREM IS
Y'*Z' + Y*Z + Y*X

BOOLEAN FUNCTION BFA AFTER APPLYING
ABSORPTION THEOREM IS
Y'*Z' + Y*Z + Y*X + Y*Y'

BOOLEAN FUNCTION BFA AFTER APPLYING
CONSENSUS  THEOREM IS
Y'*Z' + Y*Z + Y*X + Y'*Y*X'

THE MINTERMS OF THE BOOLEAN FUNCTION  BFA  ARE:
 0
 3
 4
 6
 7

THE MAXTERMS OF THE BOOLEAN FUNCTION  BFA  ARE :
 1
 2
 5
```

Figure 2

```
THE TRUTH TABLE OF BOOLEAN FUNCTION  BFC  IS  :
 0  0  0      1
 0  0  1      1
 0  1  0      D
 0  1  1      1
 1  0  0      1
 1  0  1      0
 1  1  0      1
 1  1  1      1

THE MINTERMS OF THE BOOLEAN FUNCTION  BFC  ARE:
 0
 1
 3
 4
 6
 7
THE DONT CARES OF BOOLEAN FUNCTION BFC  ARE  :
 2

THE MAXTERMS OF THE BOOLEAN FUNCTION  BFC  ARE :
 5
THE DONT CARES OF BOOLEAN FUNCTION BFC  ARE  :
 2



PRIME IMPLICANTS OF    BFC    ARE
( 0 , 6 )     Z'
( 0 , 3 )     X'
( 2 , 7 )     Y


PRIME IMPLICATES OF    BFC    ARE
( 2 , 2 )     X+Y'+Z
( 5 , 5 )     X'+Y+Z'


MINIMAL SUM OF PRODUCT OF    BFC    ARE

X' + Z' + Y


MINIMAL PRODUCT OF SUM OF     BFC    ARE

X'+Y+Z'
```

Figure 2 (Con't)

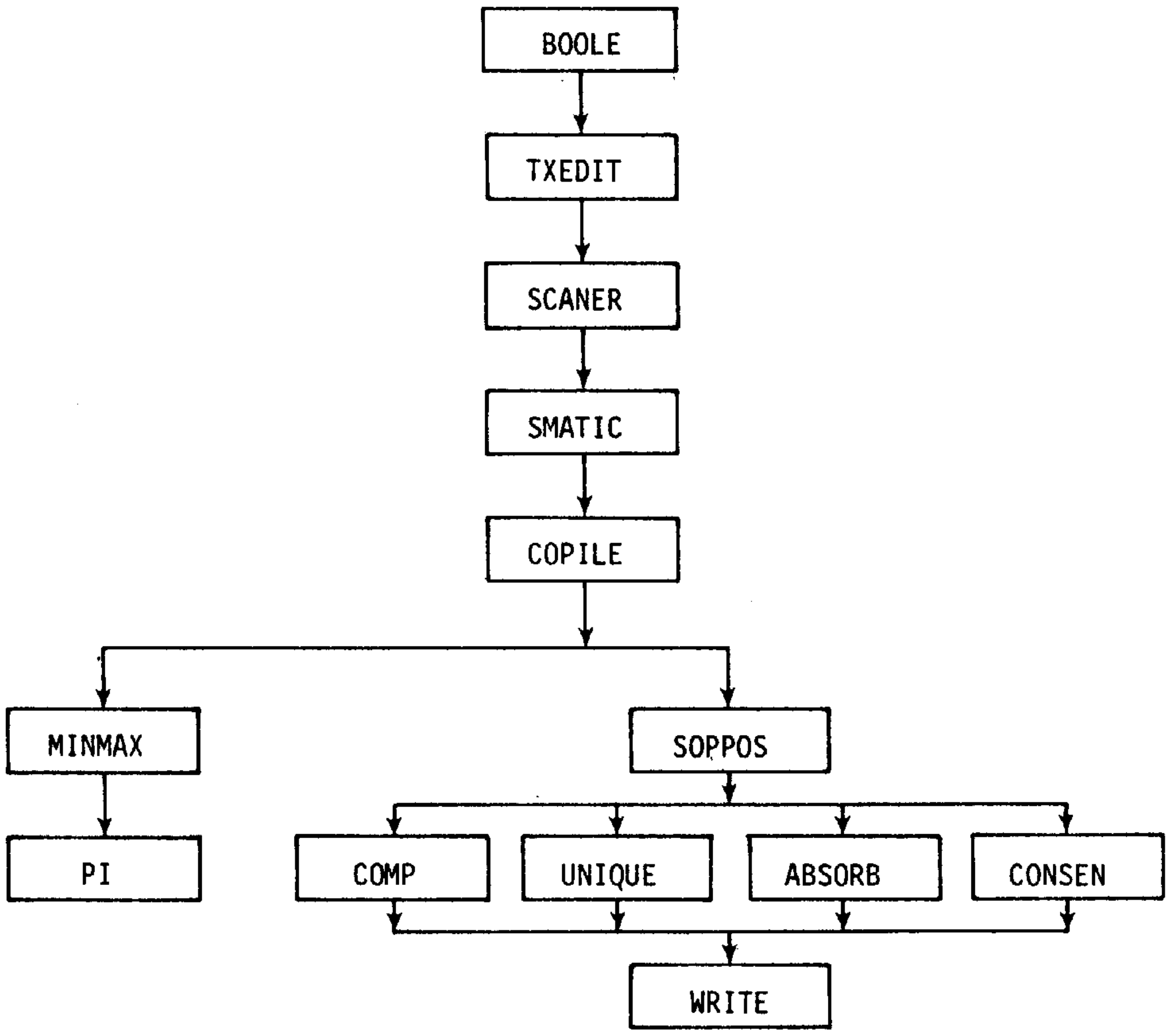
BOOLE
TXEDIT
SCANER
SMATIC
COPILE
MINMAX
SOPPOS
PI
COMP
UNIQUE
ABSORB
CONSEN
WRITE

Figure 3