Chapter- 1

INTRODUCTION

1.1 Boolean Function

A Boolean function can be defined as follows.

(i) A set of Boolean variables is taken as independent variables.

(ii) Another Boolean variable is assigned to a dependent variable.

(iii) A rule is formulated which assigns a value to the dependent variable for each set of values of the independent variables. In mathematical notation it can be depicted as

\[ a = f(x, y, z) \]

where \(x, y, z\) are the independent variables and "\(a\)" is the dependent variable and the rule assigning values to \(a\) for each set of values of \(x, y, z\) is "\(f\)".

For example, if we consider a rule as "the output variable "\(a\)" is true for every value of "\(x\)" when "\(y\)" is true but "\(z\)" is false" then the above function can be written as follows.

\[ a = f(a, b, c) = \overline{x}y\overline{z} + xy\overline{z} \] (1.1)

The above eq. (1.1) is called a logic equation and is used to describe Boolean function similar to a "truth table". Each input variable as it is occurring (in uncomplemented complemented form) in the eq. (1.1) is known as a "literal" Boolean functions are generally represented in two canonical forms, namely, Sum of Product
(SOP) and Product of Sum (POS). Each term present in a SOP form is called a minterm which is obtained by ANDing all the independent variables (input variables) in a row of truth table having a value “true” for the corresponding dependent variable (output). The minterms are numbered in increasing decimal numbers (in 8-4-2-1 code) obtained by assigning a ‘1’ to each uncomplemented literal and a ‘0’ to a complemented one. The POS form can be obtained from SOP using the “duality” in Boolean algebra. The Boolean functions are generally specified in terms of SOP/POS for carrying out Boolean function minimization.

1.2 Necessity of Boolean Function Minimization

Boolean function minimization is

(i) reduction in the number of minterms forming a Boolean functions, and

(ii) reduction in number of literals forming the minterms

without violating in any way the rule “1” governing the given Boolean function.

Interpreting the above definition in practical terms, it can be readily realized that reduction of minterms results in reduction of gates while realizing the given Boolean function and the second condition results in reduction in number of inputs to a gate, i.e., size and fan-in (of the gates) of the circuit to be realized is reduced resulting in saving of cost. These two are the reasons necessary to go for the minimization of Boolean functions.

However, with the advent of new implementation technologies of digital circuits (VLSI circuit design) the actual gate count is no longer the most important criterion of design. Even then the technologies like multilevel custom design, FPGA, PLA, etc., require two-level minimization techniques at some phase during the design process. Development of two-level Boolean minimization methods is, therefore, one of the open
research areas. The present work is an attempt to further explore the possibility of
development of a Boolean function minimization method which is suitable both as a
paper-and-pencil method and one which can be readily computerized.

A few methods of Boolean function minimization are considered in the
following section.

1.3 Introduction to Methods of Boolean Function Minimization

A number of methods have been proposed by researchers from time to time and
the trend is still going on. However, only the most frequently used methods are
described in the present chapter.

1.3.1 Minimization using Theorems of Boolean Algebra

Boolean functions can always be minimized using the theorems and
postulates of Boolean algebra. The logic adjacency theorem (namely,
\( \overline{A}B + AB = A \) ) is used for elimination of a variable from two minterms
differing in one literal. However, this method is very tedious as it is very difficult
for one to remember all the theorems and postulates and moreover it does not
provide any indication that a function has reached its minimal state and cannot be
minimized further. As a result in some case the function is made more complex then
it was. Further, a simple graphical mapping method circumventing above
difficulties has been developed which is as under.

1.3.2 Karnaugh Map Method (K-map)

The Karnaugh map method is an excellent and the most widely used
paper-and-pencil method for readily minimizing Boolean functions having small
number of variables. A K-map for a Boolean function specifies values of the function for every combination of function's variables like a truth table. Thus a "n" variable Karnaugh map has $2^n$ entries (cells). The entries (cells) are labeled with decimal equivalent of binary numbers (in 8-4-2-1 code) formed by joining the column with the row indices. The cells in the map are so placed that the adjoining cells differ in only one bit. The map entry for a cell is made 1 if the corresponding minterm is present in the given Boolean function else it is zero. The K-map up to four variables has cyclic adjacency (i.e., difference of one bit) among its rows and columns. The adjacent cells having 1 entered in them can be grouped to form a pair, quad, octet, etc., and the variables which do not change their values within a group are then extracted as prime implicants of the given function. K-map for more than four variables becomes three dimensional in order to retain cyclic adjacency among its rows and columns. The cyclic adjacency feature is, however, absent in planer (2-D) version of K-map as given in text books. Use of K-map becomes very difficult beyond six variables and as such K-map method is not used for Minimization of Boolean functions having more than six variables.

To circumvent above difficulties of K-map a tabular method capable of handling more than six number of variable was developed which is described in the next subsection.

1.3.3 Quine-McCluskey’s Tabular Method (QM Method)

Quine-McCluskey’s (QM) tabular method is an algorithmic one and is readily suitable for development of computer program for minimization of Boolean functions having large number of Boolean variables. Many of the CAD tools for two-level and multilevel logic syntheses are based on this method.
In this method all the given minterms are written along a column arranging them in the increasing order of number of 1’s which are appearing for a uncomplemented literal in a minterm. Sets of minterm differing by one in number of 1’s are partitioned drawing a line among the sets. Each minterm of a set is compared with minterms of next set having just one higher number of 1’s to obtain a minterm differing in only one bit. If such another minterm is found it is grouped with the minterm at hand to form a pair and the entry of the pair is made in the next column marking the change of bit as ‘-’ and the two minterms involved are tick marked. In this way all the partitions are covered and possible pairs are obtained. The next column, i.e., the one for pairs, is partitioned and one bit differing pairs are grouped to form the next larger groups, i.e., quads in a similar manner. Search for one bit differing groups along a column continues till no further one bit differing groups are left. The pairs, quads and other larger groups thus obtained are called the implicants.

A minterm left unmarked in the first column must be essentially included in the minimized expression without any reduction and is, therefore, called the “essential prime implicant” of the function. All the unmarked groups called “prime implicants” are further optimized by making a grid chart to select set of prime implicants which at least once cover all the minterms present in the given Boolean function. The set of prime implicants which are to appear in the minimized expression is called the minimum cover of the given Boolean function.

From the above description it can be readily seen that the minimization process in QM method comprises of the following three steps.

(i) implicant generation

(ii) prime implicant selection

(iii) covering problem solution

1 - 5
All the CAD tools developed till date use these steps in a serial manner to produce minimum expression for a given Boolean function.

A commonly used CAD tool for Boolean function minimization is described in the following subsection.

1.3.4 ESPRESSO

Espresso is a program for two-level Boolean function minimization, readily available from the University of California, Berkeley. It combines many of the best heuristic techniques developed in earlier programs, such as mini and presto. The basic ideas employed by the program are as follows:

i. Rather than start by generating all implicants and then finding those that are prime, espresso expands implicants into their maximum size. Implicants that are covered by an expanded implicant are removed from further consideration. This process is called EXPAND. How well this works depends critically on the order and direction in which implicants are expanded. Much of the power of espresso lies in its methods for directing and ordering the expansion.

ii. An irredundant cover (that is, one for which no proper subset is also a cover) is extracted from the expanded implicants. The method is essentially the same as the Quine-McCluskey's prime implicant chart method. This step is called IRREDUNDANT COVER.

iii. At this point, the solution is usually pretty good, but under certain conditions it can still be improved. There might be another cover with fewer terms or the same number of terms but fewer literals. To try to find a better cover.
espresso first shrinks the prime implicants to the smallest size that still covers the logic function. This process is called REDUCE.

iv. Since reduction yields a cover that is typically no longer prime, the sequence of steps REDUCE, EXPAND, and IRREDUNDANT COVER are repeated in such a fashion that alternative prime implicants are derived. Espresso will continue repeating these steps as long as it generates a cover that improves on the last solution found.

v. A number of other strategies are used to improve the result or to compute it more quickly. These include (a) early identification and extraction of essential prime implicants, so they need not be revisited during step 4; (b) using the function's complement to check efficiently whether an EXPAND step actually increases the coverage of the function (the min-terms covered by an expanded implicant may already be covered by another expanded implicant, so the newly expanded implicant should not be placed in the cover); and (c) a special "last gasp" step which guarantees that no single prime implicant can be added to the cover in such a way that two primes can then be eliminated.

Another recently developed CAD tool is described as under.

1.3.5 BOOM- A Heuristic Boolean Minimizer

BOOM an acronym for BOOlean Minimizer is another heuristic Boolean function minimizer which for most of the cases gives shorter time than ESPRESSO. It is briefly described as follows.

The function to be minimized is defined by its onset and offset, listed in a truth table. Thus the don't care set, often representing the dominant part of the truth table.
need not be specified explicitly. When minimizing a single-output function, the BOOM system uses three phases: coverage-directed (CD) search (generation of implicants), implicant expansion (generation of prime implicants), and solution of the covering problem.

1.4 The Proposed Method for Boolean Function Minimization

The proposed paper-and-pencil method derives its motivation from the facts that

(i) numbering of cells in K-map is carried out in 8-4-2-1 code (its equivalent decimal number are used) resulting in disorderly cell numbering which is difficult for a novice to remember,

(ii) the map becomes 3-dimensional for Boolean functions having more than four variables loosing its neatness of display which is a feature of a plane map; groupings of variables are not as immediately evident and alternate groupings are even less evident, and

(iii) it has been so far not possible to computerize K-map method as it is a graphical one.

The present work attempts to provide a method which has a regular (monotonically increasing) and much more flexible representation of minterms in a line or many number of lines, a planer and cyclic representation of adjacencies for even more than seven variables proposing a novel minterm numbering scheme based on Binary Reflected Gray Code (BRGC). The BRGC for any number of digits (bits) can be obtained recursively starting from the least significant bit (LSB). This fact has provided further motivation to explore algorithms to look for adjacency of minterms for each of its bit independently facilitating development of an algorithm which could be implemented on parallel basis on multiprocessor systems.
Most of the faster CAD tools developed are heuristic ones. These heuristic methods do not necessarily produce minimal solution but they can produce fairly good (near minimal) solution in a short time.

However, the proposed method is different from such methods in the respect that it tackles prime implicant generation step in a parallel manner on as many nodes as the number of the given Boolean function. Since all possible prime implicants are obtained in the parallel run an exact solution could be expected. The solution of covering problem is then achieved serially.

Having described the philosophy and some popular methods of Boolean function minimization a brief review of literature available in the area pertaining to advances made so far is presented in the chapter 2.

1.5 Organization of the Thesis

Chapter 2 contains a brief review of literature available on researches undertaken to develop various Boolean function minimization methods and algorithms for development of CAD tools.

Chapter 3 deals with the development of a new minterm numbering scheme based on BRGC and the proposed paper-and-pencil method of the Boolean function minimization.

In chapter 4 the proposed paper-and-pencil method is applied to simplification of Boolean functions in terms of Ex-OR/Ex-NOR gates and to Variable Entered Graphing. A real life application of the proposed method is also presented in chapter 4.
Chapter 5 deals with development and simulation of an algorithm for prime implicant formulation on parallel basis on the chosen topology of multiprocessor system.

Another parallel algorithm for Boolean function minimization is developed in the chapter 6 and the same is simulated for parallel implementation.

Lastly, in chapter 7 the work is concluded and the scope of further work in the areas related to the proposed method is explored followed by a list of references.

-oOo-