Software Implementation of the Quine-McCluskey Algorithm for Logic Gate Minimisation

Nurul Sarkar, Khaleel Petrus
School of Information Technology
Auckland University of Technology
Auckland, New Zealand
Nurul.Sarkar@aut.ac.nz
Khaleel.Petrus@aut.ac.nz

Hosneara Hossain
Department of Computer Science
North South University
Dhaka, Bangladesh

ABSTRACT

The minimisation of complex logic gates is important to simplify the hardware design of programmable logic arrays (PLAs) and programmable array logic (PALs). The result of minimisation is a considerable reduction of production cost of these digital systems. Quine-McCluskey (Q-M) is an attractive algorithm for simplifying Boolean expressions because it can handle any number of variables. A menu-driven keyboard-based software package has been developed, in C language under MS Windows, to implement the Q-M algorithm for logic gate minimisation. Based on user input (i.e. logic expression), the system displays the sum of product (SOP) functions, as well as minimised logic gates diagram. Q-M algorithm and its software implementation are described. The experimental results demonstrate successful implementation, and the simplicity of the user interface, makes the package a useful teaching and learning tool for both students and tutors.

KEYWORDS

Logic gate, Boolean expression, Quine-McCluskey algorithm, Sum of product (SOP)

1. INTRODUCTION

One of the prime objectives of designing digital logic circuits including the PLA and PAL, is to keep the number of logic gates as minimum as possible, therefore reduce the production cost of these systems. To simplify the complexity of a circuit, the designer must find another circuit that computes the same function as the original but does so with fewer gates (or perhaps with simpler gates, for example, two-input gates instead of four-input gates).

Boolean algebra, reduction of Boolean expressions, and logic gates are often included in computer science and/or engineering courses as fundamental concepts involved in computer hardware and digital systems design. Furthermore, it is essential to explain and demonstrate how complex logic expressions are minimised (simplified) to produce the final simplified diagram.

Minimising Boolean expressions using the traditional methods such as truth tables, Boolean algebra, and K-maps can be very tedious and not well suited for expressions involving more than six variables. Fortunately, the Quine-McCluskey (Q-M) algorithm, also called tabular method, is an attractive solution for minimising complex Boolean expressions involving variables of any length. Moreover, the algorithm can easily be machine implemented. Both students and lecturers can use the software tool to verify (interactively and visually) the results of Boolean expression minimisation. Pursuing this goal, a software package has been developed (written in C language under MS Windows) that not only automate the minimisation (reduction) of Boolean expressions but also display minimised expression and its logic circuit diagram.

Digital Systems design has been addressed in many references eg. Mano (1984), Tanenbaum (1999), Green (1985), and Greenfield (1977). Quine McCluskey algorithm is described extensively in literature eg. Costa (2001), Hideout (2001), Carothers (2001), and Hintz (2001). Grimsey (2000) examined the strengths and weaknesses of various methods of minimising Boolean expressions, including truth tables, Boolean algebra, and Karnaugh maps (K-maps). Lockwood (2001) presented simple text based program for the implementation of the algorithm. However, it is of limited use as a teaching and learning tool. Leathrum (1997) described and presented a text based menu driven program for Q-M algorithm, but the user interface is rather difficult to use.

The paper is organised as follows. In section 2, the Q-M algorithm for logic gate minimisation is described . The software development process is described in section 3. Test result, which verifies the successful implementation of the Q-M algorithm is presented in section 4. Section 5 presents conclusions and future work.

2. QUINE-MCCLUSKEY (Q-M) ALGORITHM

The Q-M algorithm is a way to reduce Boolean expression to its simplest form. It is designed particularly for use with problems containing six variables or more, but can be used for smaller problems as well. The algorithm is based on repeated applications of the distributed law and the fact that X OR (NOT X) is always true. The Q-M method is a systematic way of selecting the pairs to be used for simplification. The main steps in the Q-M algorithm are summarised below:

- Representing all addends as sums of minterms
- Grouping the minterms that has the same number of ones
- Merging the terms that differ in only one bit (this is done in several steps)
- In order to find the irredundant cover we use the Min-Cover Algorithm:
  1. Find all distinct minterms.
  2. Find all essential prime implicants.
  3. Find all the minterms that are covered by the essential prime implicants.
  4. Remove all minterms and prime implicants found in 1-3.
  5. Choose that prime implicant that covers most of the remaining minterms.
  6. Repeat 5 until all minterms has been covered.

3. SOFTWARE IMPLEMENTATION

Structured analysis and design (Pryor, 2000) has been employed to design the package. C programming language under MS Windows environment has been used in the implementation. Figure 1 illustrates the structured diagram of the logic gate minimisation package.

3.1 User Interface

For simplicity and ease of use, it has been decided to implement menu-driven keyboard-based interface with few menu options. The interface is easy to use and self-explanatory which makes the package well suited for students and tutors for classroom use. Therefore, the package can be an integral part of a two-hour session for teaching and learning Q-M method for logic gate minimisation. An in-class task will be given to the students to produce minimised logic diagram on paper. After prescribed period of time (say 20 minutes), the package will be introduced to the students in a step by step basis to verify their solution.

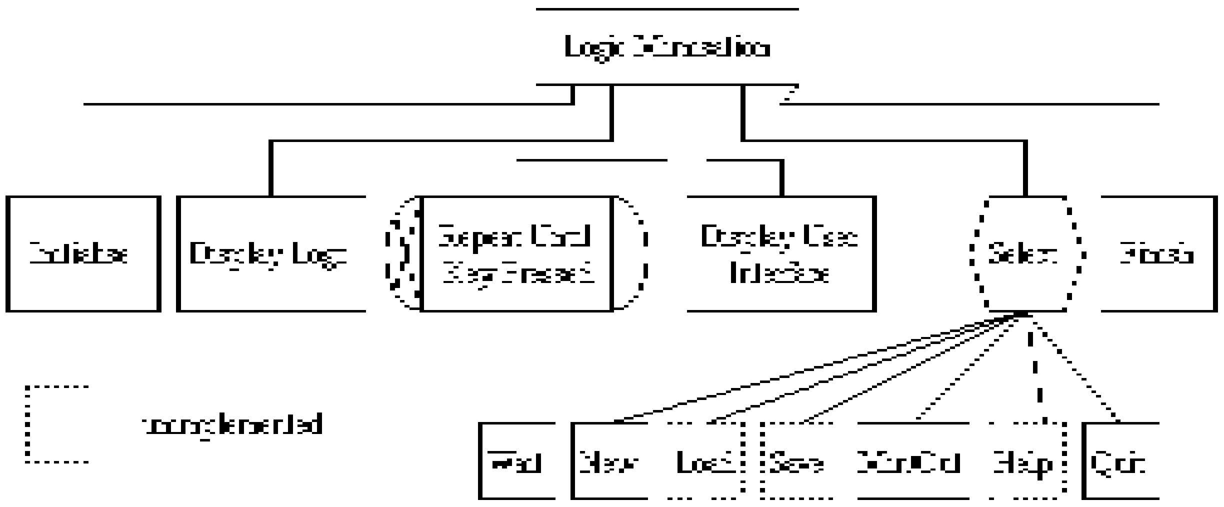

Figure 1.
Structured diagram of the logic gate minimisation package

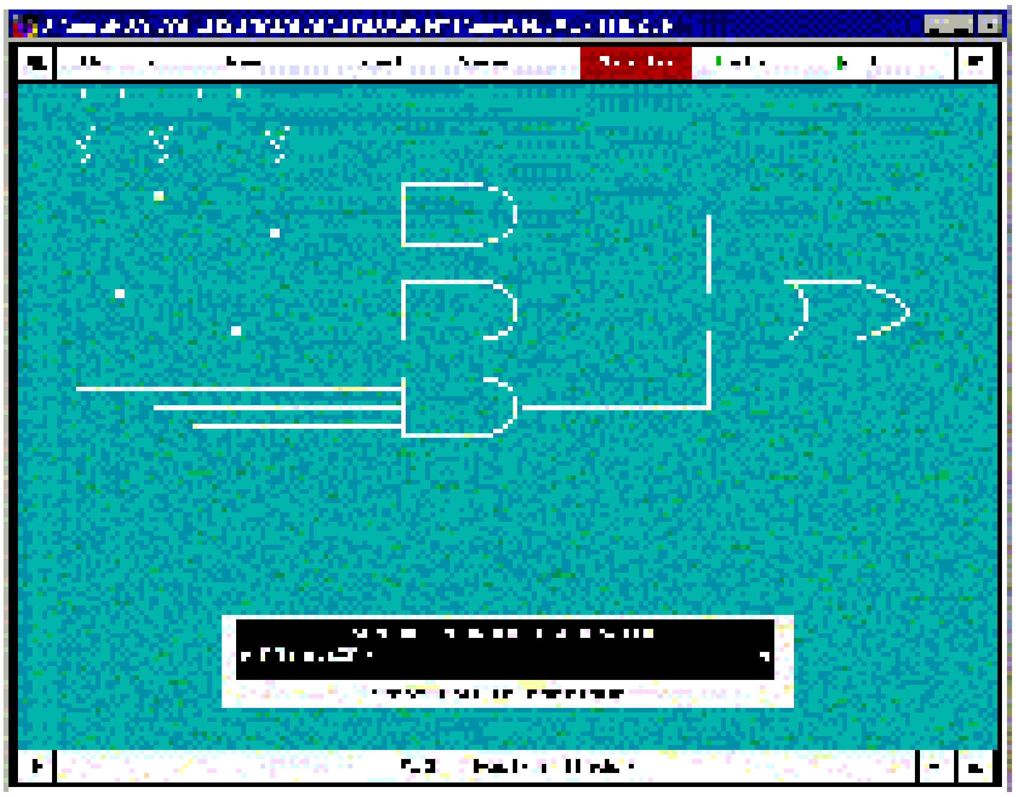

Figure 2.
Minimised output expression and logic gate diagram

4. RESULTS

To evaluate its proper operation, the package has been tested to minimise a number of Boolean expressions, each involving different number of input variables. Then the test results were validated manually. Figure 2 shows a sample test result for four variables (A, B, C, and D) Boolean expression minimisation. The following minterms have been entered from the keyboard: [0,2,3, 5,7,8,10,13,15], and the package had produced the simplified logic gate diagram as well as output expression (see Figure 2).

5. CONCLUSIONS AND FUTURE WORK

A software package has been developed for logic gate minimisation, which can be used as a teaching and learning tool for verifying results of Boolean expression minimisation. The package is easy to use and can be run from any machine operating under MS DOS/Windows. It was also tested on various PCs and found to be robust.

Currently, the system minimises Boolean expressions involving variables of size 8, which is adequate for demonstration purposes. The package can be easily upgraded to accommodate variables of any length. The user options: "New", "Min/Out", and "Quit" have been implemented. More options, eg. "Save", "Load" and "Help" are still under development and incorporation of mouse-based user interface is also suggested for future work.

REFERENCES

Costa, A. (accessed April 20, 2001) "The Quine-McCluskey Method".
<http://www.dei.isep.ipp.pt/~acc/bfunc>

Carothers, J. D. (accessed April 20, 2001) "Quine-McCluskey Algorithm". <http://www.ece.arizona.edu/~csdl/474aslide4>

Grimsey, G. (2000) "The TRUTH, the Whole TRUTH, AND/OR NOThing but the TRUTH". New Zealand Journal of Applied Computing & Information Technology,4(1):42-52.

Green, D.C (1985) "Digital techniques and systems" 2-nd edition, Longman Scientific & Technical, England.

Greenfield, J.D., (1977) "Practical digital design using ICs" John Wiley & Sons, Inc., USA.

Hideout, G. (Accessed April 20, 2001) "The Quine-McCluskey Method of Logic Reduction",. <http://www.geekhideout.com/qmm.shtml>

Hintz. (Accessed May 13,, 2001) "Quine-McCluskey Method". <http://www.cpe.gmu.edu/courses/ece331/lectures/331-8/sld001.htm>

Lockwood, J.D. (2001) "Quine-McClusky algorithm; Computational techniques". Cygwin Freeware (GPL) Tools, Washington University, February 26, 2001 <http://www.arl.wustl.edu/~lockwood/class/coe460/>

Leathrum J.F. (1997) "Quine McCluskey Tabular Minimization Method". Old Dominion University, <http://www.ece.odu.edu/~leathrum/ECE241_284/support/quine.html>

Mano, M. (1984), "Digital design", Prentice Hall, USA.

Pryor, J. (2000) "PD_PP100 Course Notes". Computing Systems and Technology, Auckland University of Technology.

Tanenbaum, A. S. (1999) "Structured computer organization" 4th Edition, Prentice Hall, Inc.