Chapter 3.12
LOGIC-Minimiser:
A Software Tool to Enhance Teaching and Learning Minimization of Boolean Expressions

Nurul I. Sarkar
Auckland University of Technology, New Zealand

Khaleel I. Petrus
University of Southern Queensland, Australia

ABSTRACT

Boolean algebra, minimization of Boolean expressions, and logic gates are often included as subjects in electronics, computer science, information technology, and engineering courses as computer hardware and digital systems are a fundamental component of IT systems today. We believe that students learn minimization of Boolean expressions better if they are given interactive practical learning activities that illustrate theoretical concepts. This chapter describes the development and use of a software tool (named LOGIC-Minimiser) as an aid to enhance teaching and learning minimization of Boolean expressions.

LEARNING OBJECTIVES

After completing this chapter, you will be able to:

- List and describe three main features of LOGIC-Minimiser.
- Explain how LOGIC-Minimiser can be used in the classroom to enhance teaching and learning Boolean expression minimization.
- Describe the Q-M algorithm for the minimization of Boolean expressions.
- Define the following key terms: Boolean expression, SOP, logic gate, logic minimization, and K-maps.

INTRODUCTION

It is often difficult to motivate students to learn minimization of Boolean expressions because students find the subject rather abstract and technical. A software tool (named LOGIC-Minimiser) has been developed that gives students a hands-on learning experience in minimizing Boolean expressions. LOGIC-Minimiser was developed in C language under MS Windows and is suitable for classroom use in introductory Boolean algebra courses. Based on user input (i.e., logic expression), the system displays the sum of product (SOP) functions as well as minimized logic gate diagrams. Test results demonstrate the successful implementation of LOGIC-Minimiser, and the simplicity of the user interface makes it a useful teaching and learning tool for both students and instructors.

This chapter describes the development of LOGIC-Minimiser and its usefulness as an aid to teaching and learning minimization of Boolean expressions. The chapter concludes with a discussion of the strengths and weaknesses of LOGIC-Minimiser and its future development.

BACKGROUND AND MOTIVATION

Boolean algebra, minimization of Boolean expressions, and logic gates are essential concepts included in electronics, computer science, information technology, and engineering. These concepts play a fundamental role in computer hardware and digital systems design. We believe that it is extremely important to incorporate practical demonstrations into these courses to illustrate theoretical concepts and therefore provide an opportunity for hands-on experience. These demonstrations will significantly enhance student learning about Boolean expression minimization.

In fact, very little material has been designed and made available for public access to supplement the teaching of Boolean expression minimization. This is revealed by searches of the Computer Science Teaching Center Web site (http://www.cstc.org/) and the SIGCSE Education Links page (http://sigcse.org/topics/) on the Special Interest Group on Computer Science Education Web site. We strongly believe, as do many others (Bem & Petelczyc, 2003; Hacker & Sitte, 2004; Ibbett, 2002; Leva, 2003; Shelburne, 2003; Williams, Klenke, & Aylor, 2003), that students learn more effectively from courses that provide for active involvement in hands-on learning activities.

Boolean expression minimization is one of the most challenging subjects to teach and learn in a meaningful way because students find the topic full of technical jargon, dry in delivery, and quite boring. Sarkar, Petrus, and Hossain (2001) have developed LOGIC-Minimiser in C under MS Windows to give students an interactive, hands-on learning experience in minimization of Boolean expressions. LOGIC-Minimiser can be used by a teacher in the classroom as a demonstration to enhance the traditional lecture environment at an introductory level. Also, students can use the system in completing tutorials on Boolean expression minimization and to verify (interactively and visually) the results of in-class tasks and exercises on Boolean expression minimization. LOGIC-Minimiser can be used either in the classroom or at home as an aid to enhance teaching and learning Boolean expression minimization.

Minimization of Boolean expressions using traditional methods such as truth tables, Boolean algebra, and K-maps can be very tedious and is not well-suited for expressions involving more than six variables. A more useful approach, the Quine-McCluskey (Q-M) algorithm, also called tabular method, is an attractive solution for minimizing complex Boolean expressions involving variables of any length. Moreover, the algorithm lends itself to a fast and easy machine implementation.

The remainder of the chapter is organized as follows. First we examine various open source software tools suitable for logic-gate design and minimization. We then describe LOGIC-Mini-

miser in teaching and learning contexts. Then, software implementation of LOGIC-Minimiser is discussed, and the educational benefits of the software are highlighted. An example of a classroom plan and LOGIC-Minimiser in practice is discussed. Test results which verify the successful implementation of LOGIC-Minimiser are presented, followed by a conclusion and future research directions.

RELATED WORK

A detailed discussion of digital systems design and minimization of Boolean expressions in general can be found in Green (1985), Greenfield (1977), Mano (1984), and Tanenbaum (1999). The Quine-McCluskey algorithm is described extensively in the computer hardware and digital logic design literature (Carothers, 2003; Costa, 2004; Hideout, 2003; Hintz, 2003). Grimsey (2000) examined the strengths and weaknesses of various methods of minimizing Boolean expressions, including truth tables, Boolean algebra, and Karnaugh maps (K-maps).

A variety of open source and commercial software tools exist for modelling and simulation of logic circuit design and Boolean expression minimization. These powerful tools can have steep learning curves; while they may be good for doing in-depth performance modelling of computer hardware and logic design, they often simulate a hardware environment in far more detail than is necessary for a simple introduction to the subject.

Lockwood (2003) presented a program for the implementation of the Q-M algorithm. However, it is of limited use as a teaching and learning tool because of its text-based interface that is not user-friendly. Leathrum (2003) described another text-based menu-driven program for the Q-M algorithm, but the user interface is rather difficult to use. Costa (2004) developed a package called "bfunc" for Boolean functions minimization. It is an MS-DOS-based program and is considered an alternative to the K-map method of simplifying Boolean functions. Burch (2002) proposed a tool named Logisim, a graphical system for logic circuit design and simulation which is suitable for classroom use. Logisim is a Java application and can be run on both Windows and Unix workstations. While Logisim is an excellent tool for building a variety of complex combinational circuits, but it is not suitable for logic gate minimization. Other tools such as Digital Works 3.0 (2001) and LogicWorks (1999) are similar to Logisim in that they provide a graphical toolbox interface for composing and simulating logic circuits. LOGIC-Minimiser, which we describe in the next section, has its own unique features, including simplicity and ease of use either in the classroom or at home, to enhance teaching and learning Boolean expression minimization.

ARCHITECTURE OF LOGIC-MINIMISER

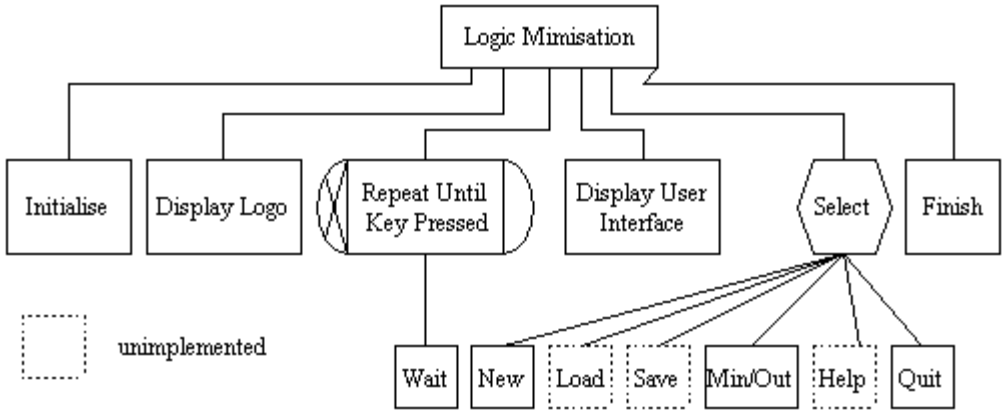
Figure 1 shows the structured diagram of LOGIC-Minimiser. The main features of LOGIC-Minimiser are briefly described.

- **New**: This feature allows users to enter a new set of variables for minimization.
- **Min/Out**: This feature allows users to view a minimized sum of product (SOP) expression and logic circuit diagram.
- **Quit**: This feature allows users to exit from the program at any time.

The following three features have not been implemented yet and are considered as future work.

- **Load**: This feature will allow users to view existing data (i.e., minimized minterms) for further analysis and modifications.

Figure 1. Structured diagram of LOGIC-Minimiser



- **Save**: This feature will allow users to store outputs on disk for later use and further modifications.
- **Help**: This feature will provide help on various topic related to minimization of Boolean expressions.

SOFTWARE IMPLEMENTATION

The Q-M algorithm is used to reduce a Boolean expression to its simplest form. It is designed particularly for use with problems containing six variables or more but can be used equally well for a smaller number of variables. The algorithm is based on repeated applications of the distributed law and the fact that XOR (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 summarized below:

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

A structured analysis and design has been employed to design the package. C programming language under MS Windows has been used in the implementation.

USEFULNESS AND BENEFITS OF LOGIC-MINIMISER

For simplicity and ease of use, it has been decided to implement LOGIC-Minimiser with a menu-driven, keyboard-based interface with a few menu options. The interface is self-explanatory, which makes the package well-suited for both students and teachers for classroom use. Therefore, the package can be an integral part of a 2-hour session for teaching and learning the Q-M method for logic gate minimization. An in-class task will be given to the students to produce a minimized logic diagram on paper. After a prescribed period of time (for example, 20 minutes), LOGIC-Minimiser will be introduced to the students on a step-by-step basis to verify their solution and learn more about minimization of Boolean expressions.

LOGIC-Minimiser provides the following main benefits:

- **Hands-on**: It facilitates an interactive, hands-on introduction to minimization of Boolean expressions.
- **Modelling**: It provides a simple and easy way to develop a variety of SOP functions and models. Students can experiment with minterms of various sizes and develop a sound knowledge and understanding of Boolean expression minimization.
- **Ease of use**: The use of a menu-driven interface makes LOGIC-Minimiser easy to use and a user-friendly tool. The software can be easily installed and run on any PC operating under MS Windows.
- **Economical/usefulness**: It enhances face-to-face teaching with online learning and can be used either in the classroom or at home to provide hands-on experience.
- **Robustness**: It was tested on various PCs across campuses and was found to be robust.
- **Challenging**: It provides an environment for students to test their knowledge on Boolean expression minimization.

EXAMPLE OF CLASSROOM PLAN

In this section we present a detailed lesson plan (2-hour session) which can be used in teaching and learning minimization of Boolean expressions using LOGIC-Minimiser. The learning outcomes focus on learning the Q-M algorithm as well as use of the software tool for verifying results of Boolean expression minimization. The lesson plan incorporates a number of resources and classroom activities, including revision of Boolean expressions, brainstorming, teaching, example, worksheet, demonstration of software package, and use of the package to verify the worksheet exercises.

Lesson Plan

Table 1 lists the learning outcomes, resources, and various activities that can be conducted in the classroom in teaching minimization of Boolean expressions effectively. It can be used for a 2-hour lecture session on the minimization of Boolean expressions. The lesson plan includes a guided worksheet (see Table 2) suitable for classroom use.

HOW TO USE THE SYSTEM

LOGIC-Minimiser is easy to use and can be run from any PC operating under MS-DOS/Windows. To run the package, the user can either double-click on “newqm.exe” or type “newqm” at the DOS prompt. The main steps of using this package (from Windows) are summarized below:

Table 1. Lesson plan (2-hour session with 10-minute break)

By the end of this session students will be able to: • Outline steps in minimization of Boolean expressions using Q-M algorithm. • Use LOGIC-Minimiser to verify the minimization of Boolean expressions.	
Resources required	• LOGIC-Minimiser • Data Show • Computer Laboratory • Whiteboard • Worksheets
Time (minutes)	**Activity**
10	Quickly review of Boolean expressions
5	Brain storming (ask the class what they know about Boolean expression minimization)
15	Explain Q-M algorithm
5	LOGIC-Minimiser demonstration
15	Solve workout/example problems
10	Break
20	Worksheet Exercises (ask the class to work in pairs and solve worksheet exercises)
20	Use LOGIC-Minimiser and verify results of minimization (worksheet exercises)
10	Conclusion and checking learning outcomes
Review: What went well:	What could be improved:

- **Run**: Double-click on "newqm.exe."
- **Entering minterms**: Select the New option to enter a new set of minterms. The user will be prompted for the number of variables to be used. After entering the appropriate number of variables, a matrix of cells with index numbers will appear on the screen. At this point the user can enter each minterm by selecting a cell by pressing the Enter key on the keyboard.
- **Accepting data**: When a set of minterms has been entered, press the F8 key to accept.
- **Display diagram**: The minimized logic-gate diagram can be seen on the screen in graphics mode. The user can zoom the diagram using the F1, F2, and F3 keys for 100%, 50%, and 20% scaling, respectively. To go back to main menu, press the F10 key.
- **Display minterms and output**: Select the Min/Out option from the main menu to see

Table 2. Boolean expression minimization worksheet

Consider the minimization of following logical function:

$F = A.B.C.D + A.B.C.\overline{D} + A.B.\overline{C}.D + A.\overline{B}.C.D + \overline{A}.B.C.D + \overline{A}.B.C.\overline{D}$

The above function can be written as:

$F(A,B,C,D) = \sum(0,1,2,4,8,9)$

Where 0, 1, 2, 4, 8, 9 are the decimal values of the minterms.
It is required to apply the minimization on this Boolean function with the use of Quine-McCluckey's algorithm, firstly by hand and then verify the solution using LOGIC-Minimiser.

Solution:

1. Write the first list as:

	A	B	C	D
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
4	0	1	0	0
8	1	0	0	0
9	1	0	0	1

2. Deduce the second list (You have to complete this list)

	A	B	C	D
0,1	0	0	0	-
0,2				
0,4				
0,8				
1,9				
8,9	1	0	0	-

3. Third list (You have to do it all yourself)

	A	B	C	D
0,8,1,9				

4. Now build the chart which relates minterms with the prime implicates as shown:

	0	2	4	8	9
$\overline{A}.\overline{B}.\overline{C}$	X	X			

5. Now use the LOGIC-Minimiser to cross check your solution.

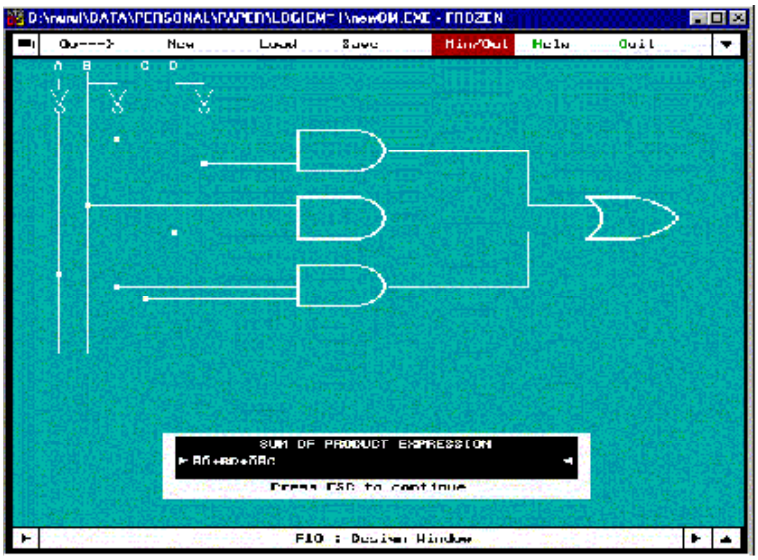
the list of minterms (that have been entered) and the minimized output expression.

- **Exit from the program**: Select the Quit option from the main menu to exit from the program at any time.

TEST RESULTS

To evaluate the performance of LOGIC-Minimiser, the software has been installed on various PCs and tested with various Boolean expres-

Figure 2. Example of four-variable minimization with minimized output expression and logic gate diagram



sions, each involving a different number of input variables. Then the test results were validated manually. Figure 2 shows a sample test result for four-variable (A, B, C, and D) Boolean expression minimization. The following minterms were entered from the keyboard: [0,2,3, 5,7,8,10,13,15], and the software produced the simplified logic-gate diagram as well as the output expression, as shown in Figure 2.

EVALUATION

An earlier version of LOGIC-Minimiser had been presented at the National Advisory Committee on Computing Qualifications conference in Napier, New Zealand (Sarkar & Petrus, 2001). The discussion during the conference presentation was quite encouraging, and many staff members from various polytechnic institutions expressed their interest in using the package in their classes.

To assess the educational value of LOGIC-Minimiser, we administered a survey to the students of the introductory digital logic subject. The survey was repeated for two consecutive years. Overall results revealed that the majority of students found the package useful and user-friendly, with an overall rating of 4 out of 5.

The questionnaire also posed five open-ended questions: (1) How well did you understand minimization of Boolean expressions before entering this course? (2) How easy did you find the software package to use? (3) How well did the package help in understanding the minimization of Boolean expressions? (4) Would you like to have more software tools of this kind as part of your course? (5) Would you prefer to learn minimization of Boolean expressions in a hybrid mode (i.e.,

minimization by hand and verification of results by software tool)?

CONCLUDING REMARKS

A software tool (LOGIC-Minimiser) has been developed that can be used in the classroom to enhance the teaching and learning of various aspects of Boolean expression minimization. LOGIC-Minimiser is easy to use and can be run from any computer operating under MS-DOS and MS Windows. It was tested on various PCs and was found to be robust. Many staff members from various polytechnic institutions expressed their interest in using the software in the classroom.

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

LOGIC-Minimiser is available free of cost to faculty interested in using it to supplement their teaching. More information about LOGIC-Minimiser can be obtained by contacting the first author.

SUMMARY

Boolean algebra, minimization of Boolean expressions, and logic gates are essential concepts included in electronics, computer science, information technology, and engineering. These concepts play a fundamental role in computer hardware and digital systems design. We believe that it is extremely important to incorporate practical demonstrations into these courses to illustrate theoretical concepts and therefore provide an opportunity for hands-on experience. These demonstrations will significantly enhance student learning about Boolean expression minimization. This chapter described the development and use of LOGIC-Minimiser as an aid to enhance teaching and learning Boolean expression minimization. It was tested on various PCs and was found to be robust.

REVIEW QUESTIONS

1. List and describe three main features of LOGIC-Minimiser.
2. Discuss the usefulness of LOGIC-Minimiser in teaching and learning contexts.
3. Describe the main steps in the Q-M algorithm for the minimization of Boolean expressions.
4. Define the following key terms: Boolean expression, logic gate, logic, minterms, and K-maps.
5. Explain how LOGIC-Minimiser can be used in the classroom for demonstration.
6. List and describe further enhancements to LOGIC-Minimiser.

REFERENCES

Anonymous. (2006). Digital Works. Retrieved January 5, 2006, from http://www.spsu.edu/cs/faculty/bbrown/circuits/howto.html

Bem, E. Z., & Petelczyc, L. (2003, February 19-23). *MiniMIPS: A simulation project for the computer architecture laboratory.* Paper presented at the Proceedings of the 34th Technical Symposium on Computer Science Education (SIGCSE'03), Reno, NV (pp. 64-68).

Burch, C. (2002). Logisim: A graphical system for logic circuit design and simulation. *Journal*

of Educational and Resources in Computing, 2(1), 5-16.

Carothers, J. D. (2003). *Quine-McCluskey algorithm*. Retrieved September 20, 2004, from http://www.ece.arizona.edu/~csdl/474aslide4

Costa, A. (2004). *Boolean functions simplification (logic minimization)*. Retrieved December 27, 2004, from http://www.dei.isep.ipp.pt/~acc/bfunc

Green, D. C. (1985). *Digital techniques and systems* (2nd ed.). Longman.

Greenfield, J. D. (1977). *Practical digital design using ICs*. Wiley.

Grimsey, G. (2000). The truth, the whole truth, and/or nothing but the truth. *Journal of Applied Computing & Information Technology, 4*(1), 42-52.

Hacker, C., & Sitte, R. (2004). Interactive teaching of elementary digital logic design with WinLogiLab. *IEEE Transactions on Education, 47*(2), 196-203.

Hideout, G. (2003). *The Quine-McCluskey method of logic reduction*. Retrieved September 20, 2004, from http://www.geekhideout.com/qmm.shtml

Hintz, K. (2003). *Quine-McCluskey method*. Retrieved September 20, from *http://www.cpe.gmu.edu/courses/ece331/lectures/331_8/sld001.htm*

Ibbett, R. N. (2002, June 24-26). *WWW visualization of computer architecture simulations*. Paper presented at the 7th annual SIGCSE conference on Innovation and Technology in Computer Science Education (ITiCSE), Aarhus, Denmark (pp. 247).

Leathrum, J. F. (2003). *Quine McCluskey tabular minimization method*. Retrieved September 20, 2004, from http://www.ece.odu.edu/~leathrum/ECE241_284/support/quine.html

Leva, A. (2003). A hands-on experimental laboratory for undergraduate courses in automatic control. *IEEE Transactions on Education, 46*(2), 263-272.

Lockwood, J. W. (2003). *Quine-McClusky algorithm; computational techniques; cygwin freeware (GPL) tools*. Retrieved September 20, 2004, from http://www.arl.wustl.edu/~lockwood/class/coe460/

LogicWorks. (1999). *Capilano Computing Systems Ltd*. Retrieved January 10, 2006, from http://www.logicworks4.com

Mano, M. (1984). *Digital design*. Prentice Hall.

Sarkar, N., & Petrus, K. (2001, July 2-5). *Logic gate minimization demonstration*. Paper presented at the 14th annual conference of the National Advisory Committee on Computing Qualifications (NACCQ), Napier, New Zealand (p. 456).

Sarkar, N., Petrus, K., & Hossain, H. (2001, July 2-5). *Software implementation of the Quine-McCluskey algorithm for logic gate minimization*. Paper presented at the 14th annual conference of the National Advisory Committee on Computing Qualifications (NACCQ), Napier, New Zealand (pp. 375-378).

Shelburne, B. (2003). *Teaching computer organization using a PDP-8 simulator*. Paper presented at the SIGCSE'03 Technical Symposium on Computer Science Education (pp. 69-73).

Tanenbaum, A. S. (1999). *Structured computer organization* (4th ed.). Prentice Hall.

Williams, R. D., Klenke, R. H., & Aylor, J. H. (2003). Teaching computer design using virtual prototyping. *IEEE Transactions on Education, 46*(2), 296-301.

KEY TERMS AND DEFINITIONS

Boolean Expression: An expression which results in a Boolean (binary or TRUE/FALSE) value. For example 4 > 3 is a Boolean expression. All expressions that contain relational operators like >, <, and so forth are Boolean. Logical gates and their combinations are used to implement physical representations of Boolean expressions.

K-Maps: This term refers to Karnaugh maps, a logical minimization method based on graphical representation of Boolean functions in which each row in the truth table of the Boolean function is represented as a box. Unlike the truth table, K- map values of input must be ordered such that the values of adjacent columns vary by one single bit.

Logic Gate: An electronic device (based on transistors) used for implementing logical functions. The inputs and outputs of the gate are Boolean (i.e., binary) values. Gates can be used to implement various Boolean functions. NOT gates take one input and have one output. The AND, NAND, OR, and NOR gates may take two or more inputs and have one output. XOR gates take two inputs and have one output. Logical functions are all combinational functions, that is, their output depends on the input. Gates can also be used to implement latches and flip-flops which have an internal state and are used to implement sequential logical systems.

Logic Minimization: Simplification of Boolean expressions with the aim of reducing the number of logical gates. This is done by reducing the number of minterms into a number of prime implicants in which as many variables as possible are eliminated. The tabular method makes repeated use of the rule $\bar{A} + A = 1$.

LOGIC-Minimiser: Software package developed at the Auckland University of Technology to enhance teaching and learning minimization of Boolean expressions. The package was implemented in C programming language.

Minterms: This term refers to the product of Boolean variables. These variables can appear either as themselves or their inverses. A minterm corresponds to exactly one row in the truth table of the Boolean function. If we have four variables A, B, C, and D, then a minterm can be something like A.B.C.D or .B.C.D, and so forth.

Quine-McCluskey Algorithm: Table-based reduction method for simplification of Boolean expressions. This method is quite versatile as compared with other algorithms. It can handle any number of inputs and can easily be implemented on machines. The method starts from the truth table of the Boolean function.

Sum of Product (SOP): A two-level expression which represents a sum of minterms of a logical function. It is two-level because it is implemented by two layers of logic gates. The first level represents the product of Boolean variables of the logical function and the second level represents summing the products with OR operator.

This work was previously published in Tools for Teaching Computer Networking and Hardware Concepts, edited by N. Sarker, pp. 303-318, copyright 2006 by Information Science Publishing (an imprint of IGI Global).