International Journal of Research (IJR) Vol-1, Issue-9, October 2014 **ISSN 2348-6848**

Karnaugh Map

Jaiveer Singh & Raju Singh
Department of Information and Technology
Dronacharya College of Engineering, Gurgaon, India
Jaiveer.16915@ggnindia.dronacharya.info
Raju.16930@ggnindia.dronacharya.info

Abstract–
In this paper, we outline the use of karnaugh map in order to simplify logical expressions. The karnaugh (pronounced as "car-no") map was propose my Maurice Karnaugh, a tele-communication engineer, at Bell Laboratories in 1953. Realization of expressions having two or three variables is what this paper aims at. Systematic way of representing and labelling a k-map is also discussed in the paper. Further it mention the properties of k-map which should be used while carrying out sum of minterms i.e., sum of products (SOP) and product of maxterms i.e., product of sum (POS) of a logic expression.

1. Introduction

At present there exist many techniques for minimizing Boolean logic expressions. A easy but very effective method was proposed by Karnaugh (1953). It is well known that the Karnaugh-map (K-map) technique is an elegant teaching resource for academics and a systematic and powerful learning tool for a digital designer in minimizing low order Boolean functions [14]. The reasons for simplifying logic expressions are obvious. By simplifying the logic function we can reduce the original number of digital components (gates) required to implement digital circuits. Therefore, by reducing the number of gates, the chip size and the cost will be reduced and the computing speed will be increased. However, simpler handling of the Karnaugh map made us choose this technique that by visualizing the error positions in the map, offers remarkable ease in code generation and also understanding.

In this paper a K-map based logic minimization algorithm and its personal digital assistant (PDA) implementation is proposed for simplifying up to four-variable Boolean functions. The algorithm and implementation is found to have excellent results. The proposed PDA application is a useful tool for students and professors [13]in the fields of electrical and computer engineering and computer science. It provides a fast and portable way to check and solve problems in digital logic, discrete mathematics and computer architecture courses.

2. Algorithm

The proposed algorithm is based on the looping of redundant terms. In order to take a closer look on how to loop two (create a pair), four (create a quad), or eight (create an octet), [12]1's in order to get the smallest possible number of groups in a K-map table setting, consider the following simple example. Note that the lower case letter represents the complement value, for example "a" means complement of "A":

F = aBcd +ABcd +ABcD +AbcD +AbCD +AbCd.

	cd	cD
ab	0	0
aB	1^1	0
AB	1^2	1^2
Ab	0	1^2

Analyzing the above K-map table, the following looping observation can be made for each 1 present in the Table [1]. (Superscript numbers show the pairing possibility of a cell.)

Cell

- aBcd has one possibility to be paired, with ABcd,
- ABcd has two possibilities to be paired, with aBcd and ABcD,
- ABcD has two possibilities to be paired, with ABcd and AbcD,
- AbcD has two possibilities to be paired, with ABcD and AbCD,
- AbCD has two possibilities to be paired, with AbcD and AbCd,
- AbCd has one possibility to be paired, with AbCD.

The basic principle behind the proposed algorithm is the method for choosing the next candidate cell to loop. A classical approach to this problem is to assign different priorities to cells. In the proposed algorithm, cells are assigned priorities based on their looping possibilities [10], intuitively the smaller the looping possibility the higher the looping

PDA-based Boolean Function Simplification: a Useful Educational Tool 331

priority. Back to the previous example, it is obvious that there are two cells that have one possibility to be paired, namely aBcd and AbCd, so we assign them the highest priority. These two cells get paired the first, aBcd gets paired with ABcd and AbCd gets paired with AbCD. After these pairings, the pairing possibilities of ABcD and AbcD are decremented by one [2], leaving both with one possibility to be paired. Finally they get paired together having an optimal result with three pairs.

F = Bcd+AbC +AcD

The observation reveals the presence of a consistent rule that lies beneath this logic. Extending the priority from possibility idea [9], the following algorithm is derived for the optimal looping of cells (1's) in a four variable K-map table.

Step 1: Find and loop a possible octet.

Step 2: Find and loop cells that have one possibility to pair.

Step 3: Find and loop cells that have one possibility to quad.

Step 3a: Repeat Step 2 for new cells with one possibility to get paired.

Step 3b: Repeat Step 3 for new cells with one possibility to get quaded.

Step 4: Find and loop cells that have two possibilities to get quaded without sharing.

Step 4b: Repeat Step 4 until no quads found.

Step 5: Find and loop cells that have two possibilities to get quaded with sharing, choose one quad out of two, with the least sharing [8].

Step 6: Find and loop cell that have two possibilities to be paired without sharing.

Step 7: Find and loop cell that have two possibilities to be paired with sharing.

Step 7a: For each success repeat Step 2.

Step 8: If there are cells that have a value of one and are not quaded or paired, then these cells either

a) Can not be looped or,

b) Have more than 2 possibilities to get paired or quaded.

Step 8a: If a) is valid then, no further action is needed.

Step 8b: If b) is valid then change the possibilities of one of these cells to 2 and go to Step 3 to repeat the procedure.

Step 8c: Repeat Step 7 until no cells that qualify for this step are found.

It is noted that the looping possibilities of a cell are reduced by one when a looping takes place and this cell can be looped with any of the cells [7] that just formed a pair, quad or octet. It is noted that sharing, in the above algorithm, means that a cell is included in more than one looping.

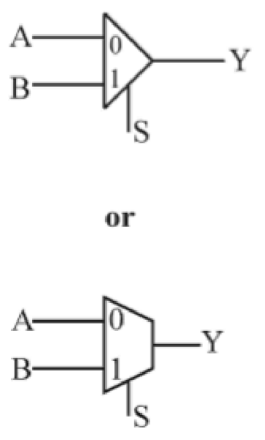
3. Design

The program was developed using the CodeWarrior For Palm OS 7.0, which supports C, C++ and Java. The implementation of the program was done in C++ using the Object Oriented paradigm. The program is divided into two major classes: a parent class(Kmap), and a child class (KmapElement). Each of these classes represents a logical division of the K-map table [6]. The Kmap object controls everything related to the K-map table as a whole, such as initializing and simplifying.In orderto apply these functions,the Kmap object creates 16 "smaller" objects representing each box. Each "smaller" object (KmapElement) [3] learns its own properties and its methods never change any other object's properties [11], but they can trigger an event, such as when a looping occurs. Breaking the program down this way is advantageousbecause the amount of complete, detailed, organized and correct information about the K-map is maximized. The parent object holds 16 children of the class KmapElement and administers the way the simplification methods are called. The child class KmapElement represents one element of the Kmap [5]. This object has all the properties, such as: pairing and quading possibilities, the value of the cell and the looping status of this cell. These objects can learn about other objects of this class through their parent since they have a reference of the parent.

4. Example

Consider the truth table for a basic 2-input multiplexer. We can view the truth table as a sort of specification which says what a circuit should do.

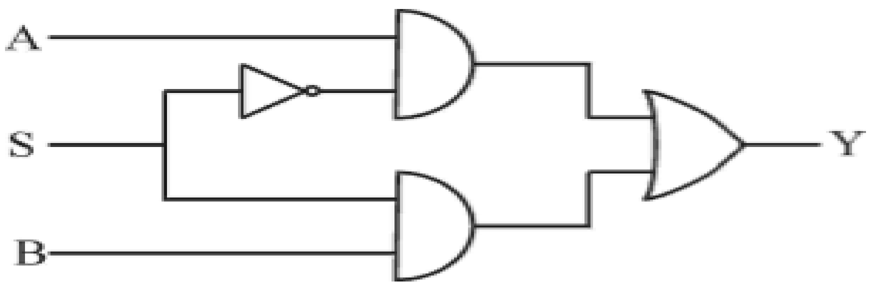
2-input multiplexer



S	A	B	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

We can write this as a standard sum of products (SSoP):
Y = S’. A .B’ + S’. A . B + S . A’. B + S . A . B = m2 + m3 + m5 + m7 = E(2, 3, 5, 7)

We can reduce this circuit using Boolean algebra (specifically, the
distributive axiom and the identity axiom):
Y = S’. A . (B’ + B) + S . B . (A’ + A) Y = S’. A . (1) + S . B . (1) Y = S’. A + S . B



Consider the following arrangements of cells:

2-input

a’· b’ 00	a’· b 01
a · b’ 10	a · b 11

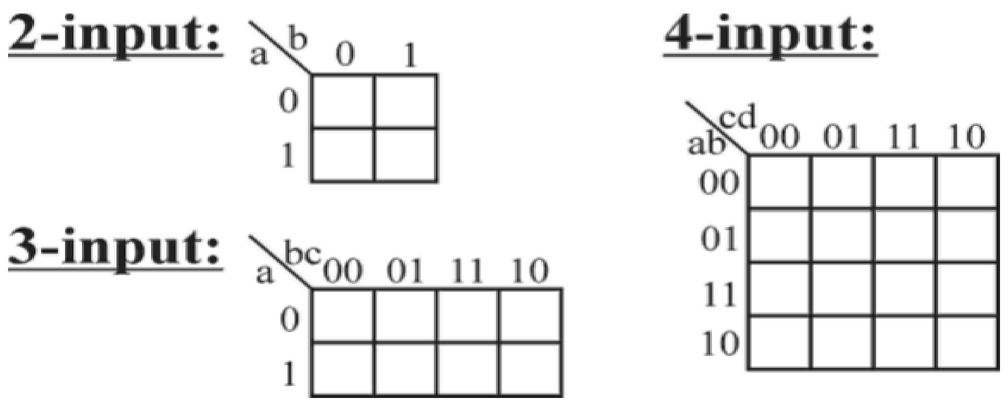
3-input

a’· b’· c’ 000	a’· b’· c 001	a’· b · c 011	a’· b · c’ 010
a · b’· c’ 100	a · b’· c 101	a · b · c 111	a · b · c’ 110

4-input

a’·b’·c’·d’ 0000	a’·b’·c’·d 0001	a’·b’·c·d 0011	a’·b’·c·d’ 0010
a’·b·c’·d’ 0100	a’·b·c’·d 0101	a’·b·c·d 0111	a’·b·c·d’ 0110
a·b·c’·d’ 1100	a·b·c’·d 1101	a·b·c·d 1111	a·b·c·d’ 1110
a·b’·c’·d’ 1000	a·b’·c’·d 1001	a·b’·c·d 1011	a·b’·c·d’ 1010

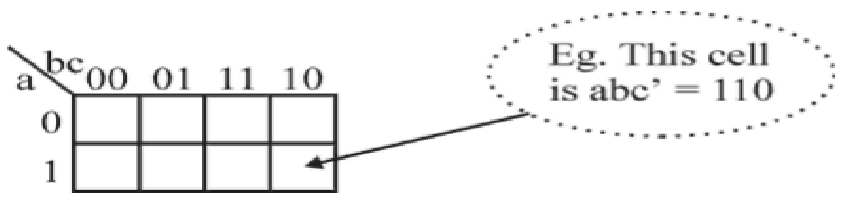
The cells are arranged as above, but we write them empty, like this:



Note that the numbers are not in binary order, but are arranged so that only a single bit changes between neighbours.

This one-bit change applies at the edges, too. So cells in the same row on the left and right edges of the array also only differ by one bit.

Note: The value of a particular cell is found by combining the numbers at the edges of the row and column.



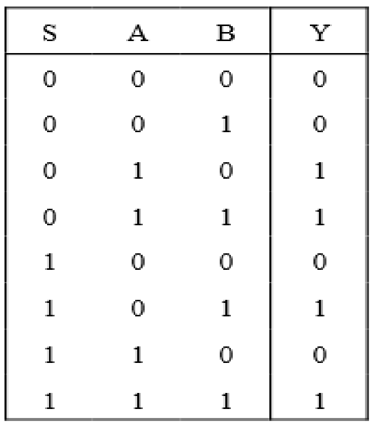
Also, in general, it is easier to order the inputs to a K-map so that they can be read like a binary number. (Show example.)

So, we have this grid. What do we do with it?

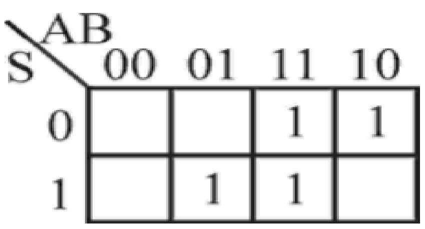
- We put 1's in all the cells that represent minterms in the SSoP [4]. (In other words, we find the 1's in the truth table output, and put 1's in the cells corresponding to the same inputs.)

Let's do this in relation to the 2-input multiplexer example:

If there are two neighbouring 1's in the grid, it means that the input bit change between the two cells has no effect on the output, and thus there is redundancy. This leads to a basic strategy.



S	A	B	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1



5. Conclusions

In this paper a useful educational tool was presented. The application can minimize a Boolean expressionusinga Palm PDA. Today PDA'sare verypopularamongstudentsand can be used easily in the classroom. For the implementation of the algorithm C++ coding was used on the CodeWarrior-Palm environment. The .prc file, which is executable on a Palm-based PDA, is 54K and is available from the authors for educational use.

Acknowledgement

The authors would like to thank the reviewers of the paper for their valuable comments.

References

[1] Brown, S., and Z. Vranesic (2002). Fundamentals of Digital Logic with VHDL Design. McGraw-Hill, New York.

[2] Chirlian, P.M. (1982). Digital Circuits with Microprocessor Applications. Matrix Publishers, Oregon.

[3] Givone, D.D. (2003). Digital Principles and Design. McGraw-Hill, New York.

[4] Hayes, J.P. (1993). Digital Logic Design. Addison Wesley Publ., New York.

[5] Karnaugh, M. (1953). The map method for synthesis of combinatorial logic circuits. Trans. AIEE, Communi- cations and Electronics, 72, 593–598.

[6] Katz, R.H. (1994). Contemporary Logic Design. Benjamin/Cummings Publ, Redwood City, CA.

[7] Mano, M., and C.R. Kime (2004). Logic Computer Design Fundamentals. Prentice Hall, New York.

[8] McCluskey, E.J. (1956). Minimization of Boolean functions. Bell System Technical Journal, 35(5), 1417–1444.

[9] Quine, W.V. (1952). The problem of simplifying truth tables. American. Math. Monthly, 59(8), 521–531.

[10] Tomaszewski, S.P, I.U. Ilgaz and G.E. Antoniou (2003). WWW-based Boolean function simplification. Inter- national Journal of Applied Mathematics and Computer Science, 13(4), 577–583.

[11] Wakerly, J.F. (2000). Digital Design. Prentice-Hall, New York.

[12] PDA-based Boolean Function Simplification: a Useful Educational Tool

[13] Ledion BITINCKA Department of Biochemistry and Biophysics, University of California San Francisco San Francisco, CA, 94143, USA e-mail: ledion@bitincka.com

[14] George E. ANTONIOU Image Processing and Systems Laboratory, Department of Computer Science Montclair State University Upper Montclair, New Jersey 07043, USA e-mail: george.antoniou@montclair.edu