INFORMATICA, 2004, Vol. 15, No. 3, 329–336
© 2004 *Institute of Mathematics and Informatics, Vilnius*

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

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

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*

Received: October 2003

**Abstract.** In this paper a useful educational tool is presented for minimizing low order Boolean expressions. The algorithm follows the Karnaugh map looping approach and provides optimal results. For the implementation, C++ was used on the CodeWarrior for Palm Operating System environment. In order to make the overall implementation efficient, the object oriented approach was used. Two step-by-step examples are presented to illustrate the efficiency of the proposed algorithm. The proposed application can be used by students and professors in the fields of electrical and computer engineering and computer science.

**Key words:** Boolean simplification, digital logic tool, Karnaugh maps, Palm and PDA application.

## 1. Introduction

Today there exist many techniques for minimizing Boolean logic expressions. A simple but very interesting 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. 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.

K-map is a graphical representation of a truth table using Gray code order. It is suitable for elimination by grouping (looping) redundant terms in a Boolean expression. By defining an algorithm for optimizing the number of loops, it is possible to entirely simplify a given Boolean expression. Quine (1952) and McCluskey (1956) reported a tabular algorithmic technique for the optimal Boolean function minimization. Almost all available techniques have been embedded into many computer aided design packages and in all the logic design university textbooks (Wakerly, 2000; Katz, 1994; Mano and Kime, 2004; Hayes 1993; Givone, 2003; Chirlian, 1982). Unfortunately, almost all the techniques along with the Espresso technique (Brown and Vranesic, 2002) do not always guaranty optimal solutions.

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 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), 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** | **CD** | **Cd** |
|---|---|---|---|---|
| **ab** | 0 | 0 | 0 | 0 |
| **aB** | $1^1$ | 0 | 0 | 0 |
| **AB** | $1^2$ | $1^2$ | 0 | 0 |
| **Ab** | 0 | $1^2$ | $1^2$ | $1^1$ |

Analyzing the above K-map table, the following looping observation can be made for each 1 present in the Table. (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, intuitively the smaller the looping possibility the higher the looping 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, 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, 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.
*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 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.

## 2.1. *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. The Kmap object controls everything related to the K-map table as a whole, such as initializing and simplifying. In order to apply these functions, the Kmap object creates 16 "smaller" objects representing each box. Each "smaller" object (KmapElement) learns its own properties and its methods never change any other object's properties, but they can trigger an event, such as when a looping occurs. Breaking the program down this way is advantageous because 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. 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.

### 2.2. *Program Flow*

As soon as an input is presented a Kmap object is created. As a result 16 children of class KmapElement are created and initialized. During the initialization phase each KmapElement gathers data about itself such as its possibilities to be paired, quaded or octeted. The instance variables of this object are updated as soon as a pair, quad or octet is formed.

After initialization, the Kmap object defines the way that the simplification is going to take place according to the presented algorithm. The actual pairing, quading, octeting and updating of the instance variables are done by the functions of the KmapElement class. The order in which these functions are executed is determined by the Kmap class, because it is where the algorithm is defined. The simplification happens only once, which means that there are no trials or secondary simplifications. All the functions in the KmapElement have options of choosing sharing and priority in any combination of the two. Each of the functions is executed only on objects that meet the requirements. In the case where the function completes its main task, it updates the instance variables of the neighboring cells that need to know the occurred looping.

## 3. Examples

Two salient examples, simple yet illustrative of the theoretical concepts presented in this work, follow below.

### 3.1. *Example* 1

Consider the following Boolean Expression:

$$F = abCD + abCd + aBcD + aBCD + ABcD + ABCD + Abcd + AbcD. \quad (1)$$

The following K-map table is generated. (Superscript/subscript show pairing/quading possibilities.)

| | **cd** | **cD** | **CD** | **Cd** |
|---|---|---|---|---|
| **ab** | 0 | 0 | $1_0^2$ | $1_0^1$ |
| **aB** | 0 | $1_1^2$ | $1_1^3$ | 0 |
| **AB** | 0 | $1_1^3$ | $1_1^2$ | 0 |
| **Ab** | $1_0^1$ | $1_0^2$ | 0 | 0 |

For each one on the table the following data can be collected:

- $abCD$ has two possibilities to be paired,
- $abCd$ has one possibility to be paired,
- $aBcD$ has two possibilities to be paired, and one to quad,
- $aBCD$ has three possibilities to be paired and one to quad,
- $ABcD$ has three possibilities to be paired and one to quad,
- $ABCD$ has two possibilities to be paired and one to quad,
- $Abcd$ has one possibility to be paired,
- $AbcD$ has two possibilities to be paired.

Following the presented algorithm:

1. There are no octets in the K-map table.
2. Terms $abCd$ and $Abcd$ are paired first, with $abCD$ and $AbcD$ respectively. ABcD and aBCD pair possibilities are decremented by one and $abCD$, $abCd$, $Abcd$ and $AbcD$ are marked as done.
3. The term $aBcD$ is looped in a quad. It is quaded with, $ABcD$, $aBCD$ and $ABCD$. All these cells are marked as done.
4. No other cells of value of one remain un-looped. All the cells have been included into pairs and quads. Therefore

$$F = abC + Abc + BD. \tag{2}$$

The above simplified Boolean expression, with three terms is the optimal solution for the given Boolean expression (1). The correctness of the result can also be verified using the software application.

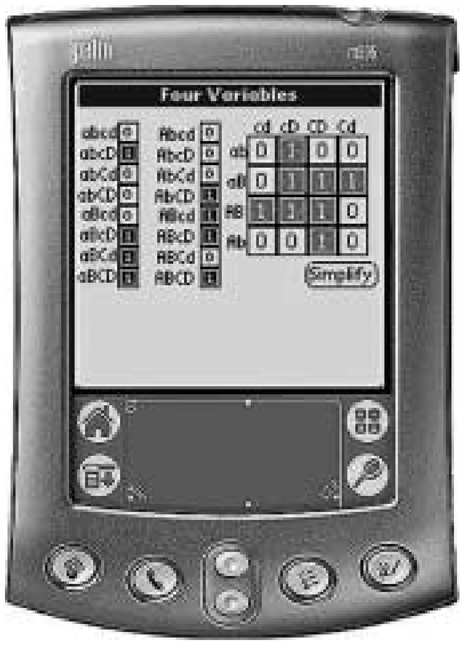
## 3.2. *Example* 2

Consider the following Boolean expression (Tomaszewski *et al.*, 2003):

$$F = abcD + aBcD + aBCD + aBCd + ABcd + ABcD + ABCD + AbCD. \tag{3}$$

Using the M505 Palm PDA the boxes, shown in Fig. 1, are selected according to each term of the given Boolean expression (3).

In this case a quad is possible. But according to the algorithm any quad of any type will not be looped before all the 1's that have one possibility to be paired are looped. Gathering all cells' looping possibilities we have,



| | | | |
|---|---|---|---|
| abcd | 0 | Abcd | 0 |
| abcD | 1 | AbcD | 0 |
| abCd | 0 | AbCd | 0 |
| abCD | 0 | AbCD | 1 |
| aBcd | 0 | ABcd | 1 |
| aBcD | 1 | ABcD | 1 |
| aBCd | 1 | ABCd | 0 |
| aBCD | 1 | ABCD | 1 |

| | cd | cD | CD | Cd |
|---|---|---|---|---|
| ab | 0 | 1 | 0 | 0 |
| aB | 0 | 1 | 1 | 1 |
| AB | 1 | 1 | 1 | 0 |
| Ab | 0 | 0 | 1 | 0 |

Fig. 1. K-map setting of expression 3.

- $abcD$ has one possibility to be paired,
- $aBcD$ has three possibilities to be paired & one possibility to be quaded,
- $aBCD$ has three possibilities to be paired & one possibility to be quaded,
- $aBCd$ has one possibility to be paired,
- $ABcd$ has one possibility to be paired,
- $ABcD$ has three possibilities to be paired & one possibility to be quaded,
- $ABCD$ has three possibilities to be paired & one possibility to be quaded,
- $AbCD$ has one possibility to be paired.

According to the algorithm the following looping combinations are obtained:

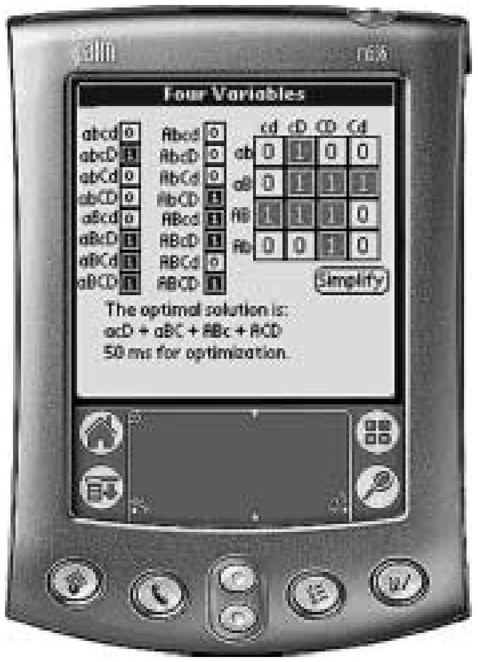
- $abcD$ is paired with $aBcD$ since $abcD$ has one possibility to be paired,
- $aBCd$ is paired with $aBCD$ since $aBCd$ has one possibility to be paired,
- $ABcd$ is paired with $ABcD$ since $ABcd$ has one possibility to be paired,
- $AbCD$ is paired with $ABCD$ since $AbCD$ has one possibility to be paired.

Finally the following result is obtained,

$$F = acD + aBC + ABc + ACD. \tag{4}$$

Using a Palm PDA and pressing the “simplify” button the above derived result (4) is displayed in the Palm screen, depicted in Fig. 2.

The simplified Boolean expression (4) is the optimal solution for the given Boolean expression (3).



**Four Variables**

| | | | |
|---|---|---|---|
| abcd | 0 | Abcd | 0 |
| abcD | 1 | AbcD | 0 |
| abCd | 0 | AbCd | 0 |
| abCD | 0 | AbCD | 1 |
| aBcd | 0 | ABcd | 1 |
| aBcD | 1 | ABcD | 1 |
| aBCd | 1 | ABCd | 0 |
| aBCD | 1 | ABCD | 1 |

| | cd | cD | CD | Cd |
|---|---|---|---|---|
| ab | 0 | 1 | 0 | 0 |
| aB | 0 | 1 | 1 | 1 |
| AB | 1 | 1 | 1 | 0 |
| Ab | 0 | 0 | 1 | 0 |

```
The optimal solution is:
acD + aBC + ABc + ACD
50 ms for optimization.
```

Fig. 2. The result as in Eq. 4.

## 4. Note

It is noted that the proposed simplification algorithm was tested correctly for all the 65535 combinations, using four variables, in conjunction with the internet Quine-McCluskey-based Boolean minimization algorithm (Tomaszewski *et al.*, 2003).

## 5. Conclusions

In this paper a useful educational tool was presented. The application can minimize a Boolean expression using a Palm PDA. Today PDA's are very popular among students and 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

Brown, S., and Z. Vranesic (2002). *Fundamentals of Digital Logic with VHDL Design*. McGraw-Hill, New York.
Chirlian, P.M. (1982). *Digital Circuits with Microprocessor Applications*. Matrix Publishers, Oregon.
Givone, D.D. (2003). *Digital Principles and Design*. McGraw-Hill, New York.
Hayes, J.P. (1993). *Digital Logic Design*. Addison Wesley Publ., New York.
Karnaugh, M. (1953). The map method for synthesis of combinatorial logic circuits. *Trans. AIEE, Communications and Electronics*, **72**, 593–598.
Katz, R.H. (1994). *Contemporary Logic Design*. Benjamin/Cummings Publ, Redwood City, CA.
Mano, M., and C.R. Kime (2004). *Logic Computer Design Fundamentals*. Prentice Hall, New York.
McCluskey, E.J. (1956). Minimization of Boolean functions. *Bell System Technical Journal*, **35**(5), 1417–1444.
Quine, W.V. (1952). The problem of simplifying truth tables. *American. Math. Monthly*, **59**(8), 521–531.
Tomaszewski, S.P, I.U. Ilgaz and G.E. Antoniou (2003). WWW-based Boolean function simplification. *International Journal of Applied Mathematics and Computer Science*, **13**(4), 577–583.
Wakerly, J.F. (2000). *Digital Design*. Prentice-Hall, New York.

**L. Bitincka** was awarded the bachelor of science in computer science (Magna Cum Laude) from the Montclair State University in 2003. He is currently with Department of Biochemistry and Biophysics of the University of California San Francisco, San Francisco, CA, USA.

**G. Antoniou** was awarded the doctor of electrical engineering (informatics) degree from the National Technical University of Athens in 1988. He is currently a professor in the Department of Computer Science at Montclair State University, NJ, USA. His present research interests include system theory, multidimensional digital signal processing and FPGA/VHDL applications.

## Bulio funkcijos supaprastinimo PDA realizacija: naudingas mokymo įrankis

Ledion BITINCKA, George E. ANTONIOU

Šiame straipsnyje aprašomas naudingas mokymo įrankis žemos eilės Bulio reiškinių minimizavimui. Algoritmas sudarytas pagal Karnaugh atvaizdavimo metodą. Jį naudojant, gaunami optimalūs rezultatai. Realizavimas atliktas C++ kalba su CodeWarrior Palm operacinės sistemos aplinkoje, naudojant objektiškai orientuotą metodą. Pasiūlyto algoritmo efektyvumą demonstruoja pateikti du smulkiai aprašyti pavyzdžiai. Sukurtą mokymo įrankį gali naudoti dėstytojai ir studentai elektros ir kompiuterių inžinerijos bei informatikos srityse.