Circuits, Systems, and Signal Processing (2022) 41:5895–5902
https://doi.org/10.1007/s00034-022-02040-4

SHORT PAPER

Simplifying Karnaugh Maps by Making Groups of Non-power-of-two Elements

Mario Garrido[1]

Received: 15 January 2021 / Revised: 20 April 2022 / Accepted: 20 April 2022 /
Published online: 23 May 2022

Abstract

When we study the Karnaugh map in the switching theory course, we learn that the ones in the map are combined in rectangles whose length and width must be a power of two. The result is the logic function described as a sum of products. This paper shows that we can also make groups where the length and width of the rectangles are equal to three. This results in a logic function that is simpler than the sum of products in terms of logic gates, leading to more hardware-efficient circuits. This idea is extended later in the paper to other groups of elements. Finally, a new perspective on the Karnaugh map that integrates the proposed approach with the conventional one is provided. This can be used in switching theory courses to improve the explanation of the Karnaugh map.

Keywords Boolean algebra · Digital circuits · Groups of non-power-of-two elements · Karnaugh map · Logic function · Simplification

1 Introduction

In Boolean algebra, there exist different algorithms to simplify logical functions. The most popular ones are the Karnaugh map [5, 6, 11] and the Quine–McCluskey method [7, 8]. Through the years, many new ideas to improve these algorithms have been presented. This includes new maps to handle larger number of variables [1], models to explain how the number of variables in the Karnaugh map is reduced [9], computer tools to apply the Karnaugh map efficiently [10], recursive algorithms to solve Boolean relations [2], and upgrades of the Karnaugh map [5, 11] such as using of

This work was supported by the Ramón y Cajal Grant RYC2018-025384-I of the Spanish Ministry of Science, Innovation and Universities.

✉ Mario Garrido
mario.garrido@upm.es

[1] Department of Electronic Engineering, Universidad Politécnica de Madrid, Madrid, Spain

XOR patterns [11] or projected sum of products [3, 4]. In this paper, a new upgrade of the Karnaugh map is presented, which is based on making groups of non-power-of-two elements.

When we study the Karnaugh map, we learn that the ones in the map must be combined in groups of $a \times b$ elements, where a and b are powers of two [5]. Other shapes and rectangles of other sizes are not allowed. A simple example is when there are three ones in a row of the Karnaugh map, as shown in Fig. 1a. In this case, two groups of two elements are created. One with the first element and the middle one, and the other group with the middle element and the last one. The alternative shown in Fig. 1b, where three elements are grouped together, is not allowed.

When the rule for grouping elements is followed, the result is a logic function represented as a sum of products (SOP). This representation has the advantage that it leads to a circuit with low delay [4]. By contrast, the resulting circuit is generally not efficient in terms of the number of logic gates. This paper presents a new perspective on the Karnaugh map that makes it suitable for the design of hardware-efficient circuits, not only for low-delay ones. This is achieved by making groups of non-power-of two elements in the map, which results in logic functions with less number of gates. Additionally, the proposed approach is easy to apply. In fact, an upgraded explanation of the Karnaugh map is provided in this paper, which can be used in switching theory courses.

The proposed approach is developed in the paper as follows: first, making groups of three, six and nine elements is studied in Sect. 2. Then, the ideas are generalized to groups of $2^n - 1$ elements in Sect. 3, which completes the approach. Later, in Sect. 4 it is explained how to use the proposed approach to improve the explanation of the Karnaugh map. In Sect. 5, an example of application of the proposed approach is presented. Finally, the main conclusions of the paper are summarized in Sect. 6.

2 Making Groups of Three, Six and Nine Elements

Figure 1a shows how to group three ones in a Karnaugh map according to the conventional approach, which consist of making two groups of two elements. The logic function for this case is:

$$f = a\bar{c}d + b\bar{c}d, \tag{1}$$

which can be implemented with five two-input logic gates. As a general criterion throughout the paper, the number of logic gates is counted as the number of two-input logic gates.

The alternative presented in Fig. 1b groups all the three elements together. In this case, the second row corresponds to $\bar{c}d$, whereas the three last columns correspond to the function $a + b$. This leads to

$$f = (a + b)\bar{c}d, \tag{2}$$

which can be implemented with three logic gates. Therefore, grouping three elements together is more hardware-efficient than making two groups of two elements.

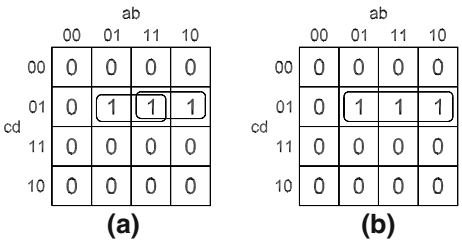

Fig. 1 Grouping three elements. **a** Conventional approach. **b** Proposed approach

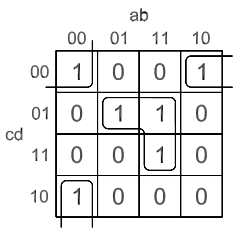

Fig. 2 Example where groups of three elements are made and these groups have an L shape

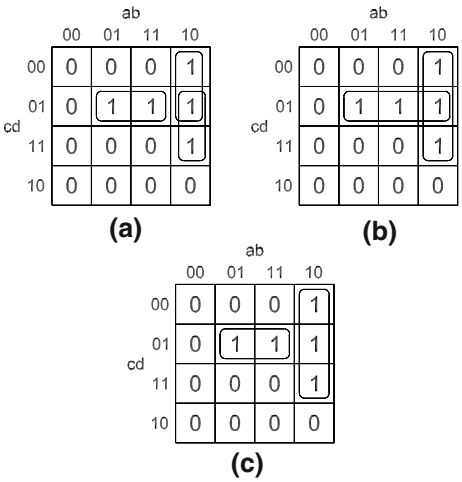

Fig. 3 Two groups of three elements. **a** Conventional approach. **b** Making groups of three elements. **c** Proposed approach

This idea can be extended to groups of three elements that form an L shape, as shown in Fig. 2. The three ones in the centre correspond to $bd(\overline{c}+a)$, and those in the corners correspond to $\overline{b}\,\overline{d}(\overline{c}+\overline{a})$.

When there are two groups of three elements that intersect as in Fig. 3, the conventional approach in Fig. 3a obtains the logic function by making three groups of two elements, which results in

$$f = b\overline{c}d + a\overline{b}\overline{c} + a\overline{b}d. \tag{3}$$

The circuit used to calculate this equation requires 8 logic gates. By contrast, making groups of three elements as in Fig. 3b results in

$$f = (a+b)\overline{c}d + (\overline{c}+d)a\overline{b}, \tag{4}$$

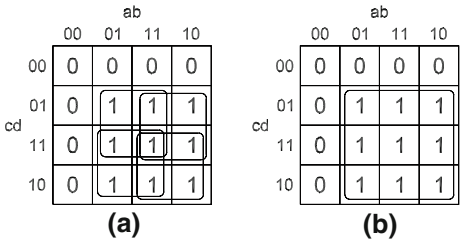

Fig. 4 Grouping a square of three-by-three elements. **a** Conventional approach. **b** Proposed approach

which requires seven logic gates. However, there is an even better alternative shown in Fig. 3c, which consists of a group of two elements and a group of three elements and leads to

$$f = b\overline{c}d + (\overline{c} + d)a\overline{b}, \tag{5}$$

and requires six logic gates.

The examples in Figs. 1 and 3 lead to two interesting conclusions. First, a group of three elements is more hardware-efficient than two groups of two elements. Second, a group of two elements is more hardware-efficient than a group of three elements. These conclusions serve as decision rules when making the groups.

Another interesting case is when there is a square of 3×3 ones, as shown in Fig. 4. The use of the conventional approach requires to make four squares of 2×2 and leads to

$$f = bd + ad + bc + ac, \tag{6}$$

which requires seven logic gates.

The alternative for this case is to group all the nine elements together. The three last columns correspond to the function $a + b$ and the three last rows to $c + d$, which results in

$$f = (a + b)(c + d). \tag{7}$$

In this case, the calculation only requires three logic gates, which is a significant reduction of the hardware cost with respect of grouping the elements in squares of 2×2.

As a final example of making groups with a number of elements that is multiple of three, Fig. 5 highlights the case of grouping six elements together. The conventional approach solves the Karnaugh map in Fig. 5a by making three groups of 2×2 and obtains

$$f = a\overline{c} + bd + ad, \tag{8}$$

which requires five logic gates.

The alternative of using two groups of six elements in Fig. 5b results in:

$$f = a(\overline{c} + d) + (a + b)d, \tag{9}$$

which also requires five logic gates.

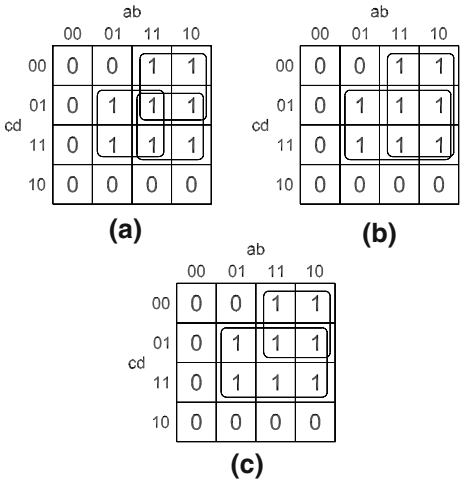

Fig. 5 Making groups of six elements. **a** Conventional approach. **b** Using only groups of six elements. **c** Proposed approach

Finally, by using a group of six elements and a group of four elements as in Fig. 5c, the resulting logic function is:

$$f = a\bar{c} + (a + b)d. \tag{10}$$

In this case, the number of logic gates is reduced to 4.

This example illustrates the facts that a group of four elements is more hardware-efficient than a group of six elements, whereas a group of six elements is more hardware-efficient than two groups of four elements.

3 Making Groups of $2^n - 1$ Elements

The ideas for groups with a number of elements that is a multiple of three can be generalized to groups of $2^n - 1$ elements where $n \in \mathbb{N}$. These elements must be embedded in a rectangle of size $2^i \times 2^j$ where both $i, j \in \mathbb{N}$ and $i + j = n$. According to this, there will be a single element in the $2^i \times 2^j$ rectangle that is not a one. This element will be excluded from the group by using an OR function.

As an example, Fig. 6 shows a group of seven elements. In this case, $i = 1$, $j = 2$, $n = 3$, $2^i \times 2^j = 2^n = 8$ and $2^n - 1 = 7$. According to the conventional approach in Fig. 6a, the ones are grouped in three groups of four elements, leading to

$$f = cd + bd + ad. \tag{11}$$

This logic function requires five logic gates.

According to the proposed approach, the ones are embedded in a rectangle of 2×4 whose logic function is d. Inside the rectangle, the function is $(a + b + c)$. This leads to

$$f = d(a + b + c), \tag{12}$$

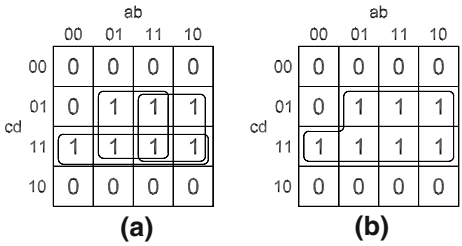

Fig. 6 Grouping seven elements. **a** Conventional approach. **b** Proposed approach

which results in three logic gates. Again, this strategy reduces the number of logic gates.

4 Upgrading the Explanation of the Karnaugh Map

The proposed approach allows for presenting the Karnaugh map as an optimization tool with two possible goals: to reduce the delay of the circuit or to obtain a hardware-efficient digital circuit.

In order to reduce the delay of the circuit, we derive the SOP expression with the conventional approach by using the following rules:

- Group the ones in the Karnaugh map in squares or rectangles of $2^i \times 2^j$ elements.
- Borders of the Karnaugh map are connected to the opposite borders, which allows for connecting elements from opposite borders.
- A one in the map may be included in one or several groups.
- Each group must include at least a one that is not included in any other group. Otherwise, the group is redundant.
- Groups must be made with the aim of making the smallest number of groups and include the largest number of ones in these groups.

In order to obtain a hardware-efficient circuit, we incorporate the ideas presented in this paper and consider making groups of a number of elements that is not a power of two. This transforms the design rules into:

- Group the ones in the Karnaugh map in squares or rectangles of $a \times b$ elements where $a, b \in \{1, \ldots, 4\}$, or groups of $2^n - 1$ elements embedded in a square or rectangle of size $2^i \times 2^j$, being $i + j = n$.
- Borders of the Karnaugh map are connected to the opposite borders, which allows for connecting elements from opposite borders.
- A one in the map may be included in one or several groups.
- Each group must include at least a one that is not included in any other group. Otherwise, the group is redundant.
- Groups must be made with the aim of making the smallest number of groups and include the largest number of ones in these groups. However, a group of two ones is preferable to a group of 3 ones where the first or last one is already included in another group. Likewise, a group of 4 ones is preferable to a group of 6 if the 2 ones of difference are already included in another group.

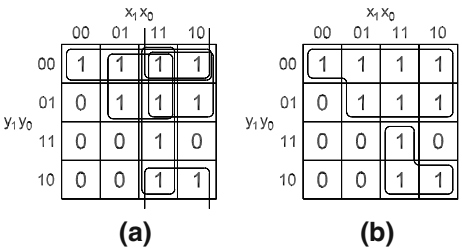

Fig. 7 Calculation of $x_1x_0 \geq y_1y_0$. **a** Using the conventional approach. **b** Using the proposed approach

5 Example of Application

In this section, an example of application of the proposed approach is presented. Let us consider two 2-bit unsigned binary numbers x and y, where $x = x_1x_0$ and $y = y_1y_0$. In this context, we want to design a circuit that determines whether $x \geq y$.

Figure 7a shows the solutions to this problem using the conventional approach. The map is created by writing ones in the cells for which $x \geq y$. For instance, if $x = x_1x_0 = 10$ and $y = y_1y_0 = 01$, then $x = 2$ and $y = 1$ in decimal. Thus, it is fulfilled that $x \geq y$ and a one is written in the cell for which $x_1x_0 = 10$ and $y_1y_0 = 01$.

Once the map is created, the ones in the map are grouped by following the conventional approach. This leads to five groups of ones that result in the logic function

$$f = \overline{y_1}\,\overline{y_0} + \overline{y_1}x_0 + \overline{y_1}x_1 + x_1x_0 + \overline{y_0}x_1. \tag{13}$$

This logic function requires nine logic gates.

Figure 7b shows the solution of the same problem using the proposed approach. In this case, a group of seven elements and a group of three elements are formed, leading to the logic function

$$f = \overline{y_1}(\overline{y_0} + x_1 + x_0) + y_1x_1(\overline{y_0} + x_0), \tag{14}$$

which requires seven logic gates.

As a result, the proposed approach leads to a solution with less logic gates. Furthermore, this example shows another advantage of the proposed approach, which is the reduction of the number of groups with respect to the conventional approach. In the example in Fig. 7, the proposed approach only has to create two groups compared to the five groups of the conventional approach. This reduction in the number of groups makes it faster and more straightforward to obtain the logic function.

6 Conclusion

In this paper, a new way to understand the Karnaugh map has been presented. Contrary to previous approaches, the proposed approach enables groups of ones whose size is not a power of two. This results in a further simplification of the logic functions that result from the map, leading to digital circuits that are more hardware-efficient compared to the conventional approach, as they require a smaller number of logic

gates. By integrating the proposed approach with previous knowledge, the Karnaugh map has been presented as a tool to either minimize the delay of the circuit or reduce the number of logic gates. This enriches the explanation of the Karnaugh map and is a step forward toward the final research goal of designing simple and intuitive methods for deriving hardware-efficient digital circuits.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

Data Availability Data sharing was not applicable to this article as no datasets were generated or analysed during the current study.

References

1. Y.S. Abdalla, Introducing the Yasser-map as an improvement of the Karnaugh-map for solving logical problems, in *Proc. IEEE Int. Conf. Electr. Comput. Comm. Tech.* (2015), pp. 1–5
2. D. Baneres, J. Cortadella, M. Kishinevsky, A recursive paradigm to solve Boolean relations. IEEE Trans. Comput. **58**(4), 512–527 (2009)
3. A. Bernasconi, V. Ciriani, G. Trucco, T. Villa, Minimization of EP-SOPs via Boolean relations, in *Proc. IEEE Int. Conf. VLSI-SoC* (2013), pp. 112–117
4. A. Bernasconi, V. Ciriani, G. Trucco, T. Villa, Boolean minimization of projected sums of products via Boolean relations. IEEE Trans. Comput. **68**(9), 1269–1282 (2019)
5. M.E. Holder, A modified Karnaugh map technique. IEEE Trans. Educ. **48**(1), 206–207 (2005)
6. M. Karnaugh, The map method for synthesis of combinational logic circuits. Trans. Am. Inst. Electr. Eng. Part I Commun. Electron. **72**(5), 593–599 (1953)
7. A. Majumder, B. Chowdhury, A.J. Mondai, K. Jain, Investigation on Quine McCluskey method: a decimal manipulation based novel approach for the minimization of Boolean function, in *Proc. Int. Conf. Electron. Design Comput. Netw. Autom. Verif.* (2015), pp. 18–22
8. W.V. Quine, The problem of simplifying truth functions. Am. Math. Mon. **59**(8), 521–531 (1952)
9. M.R. Rahman, Mathematical study for reduction of variables in Karnaugh map, in *Smart Computing and Informatics*. ed. by S. Satapathy, V. Bhateja, S. Das (Springer, Berlin, 2018)
10. M.S. Rahman, R. Hasib, B. Sultana, M.G. Hussain, M. Rahman, M.A. Rahaman, An extensive Karnaugh mapping tool for Boolean expression simplification, in *Proc. Int. Conf. Sustain. Tech. Ind. 4.0* (2019), pp. 1–5
11. R.F. Tinder, Multilevel logic minimization using K-map XOR patterns. IEEE Trans. Educ. **38**(4), 370–375 (1995)