Design and development of Karnaugh Map simulator to simplify Boolean Expression

Mundhe Anil Krushna
Department of Electrical Engineering
JSPM'S Rajarshi Shahu College of Engineering, Pune, India
email id:mundheanil84@gmail.com

Banale Shashank Santosh
Department of Electrical Engineering
JSPM'S Rajarshi Shahu College of Engineering, Pune, India
email id: banaleshashank0804@gmail.com

Manjusha A. Kanawade
Department of Electrical Engineering
JSPM'S Rajarshi Shahu College of Engineering, Pune, India
email id: makanawade_elec@jspmrscoe.edu.in

***Abstract*— This research paper proposes the design and development of a simulator to simplify Boolean expressions using Karnaugh Map (K-map). The simulator is developed using Html, CSS and JavaScript allows users to input Boolean expressions, display Karnaugh Map and output simplified Boolean expressions. The simulator is tested thoroughly and found to be accurate and efficient in simplifying Boolean expressions. The proposed simulator provides an effective tool for engineers and designers to simplify Boolean expressions, ultimately reducing the complexity of digital logic circuits.**

Keyword – Boolean Expression, Karnaugh Map, Digital Logic Circuits,Virtual Simulator.

I. Introduction

Karnaugh map (K-map) is a graphical tool used in digital circuit design and analysis to simplify Boolean expressions and minimize the number of gates required to implement the circuit. K-map provide a systematic method for minimizing Boolean functions, reducing the complexity of digital circuits, and improving their performance. They are widely used in the design of digital circuits, including microprocessors, memory devices, and communication systems.

Karnaugh map (K-map) is a graphical tools used in digital circuit design and analysis to simplify Boolean expressions and minimize the number of gates required to implement the circuit. K-map provide a systematic method for minimizing Boolean functions, reducing the complexity of digital circuits and improving their performance. However, simplifying complex Boolean expressions using K-map can be challenging, especially when the number of variables is large. Karnaugh map simulations are a powerful tool for simplifying Boolean expression using K-map, providing designers with a quick and efficient way to optimize digital circuits.

Karnaugh map simulations use software tools to automate the process of simplifying Boolean expressions using K-map. The simulations can be based on heuristic algorithms that find a good solution quickly, or exact algorithms that find the optimal solution to the simplification problem. K-map simulations are particularly useful for designers who need to optimize their circuits quickly or who do not have the expertise to perform manual K-map simplification.

Karnaugh map (K-map) is a powerful tool for simplifying Boolean expressions in digital circuit design. K-map provide a visual representation of a Boolean function, where each cell in the map represents a possible combination of input variables. By grouping adjacent cells with the same output value, K-map can be used to simplify Boolean expressions and reduce the number of gates required to implement a circuit.

To use K-map for simplification, we need to follow a set of steps to create the map and group adjacent cells. These steps are:

1. Write the truth table for the Boolean function.
2. Identify the number of variables in the Boolean function and create a K-map with the appropriate number of rows and columns.
3. Label the rows and columns of the K-map with the input variable values.
4. Fill in the K-map with the output values from the truth table.
5. Group adjacent cells with the same output value in the K-map to form implicants.
6. Combine the implicants to form a simplified Boolean expression.

Let's consider an example to illustrate these steps. Suppose we have the following Boolean function:

$$F(A,B,C) = \Sigma(0,2,4,5,7)$$

Prepare the truth table as shown in Table 1 for this function:

Table 1: Truth Table

A	B	C	F
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

From the truth table fill the cell values in K-Map structure as shown in Fig. (1).

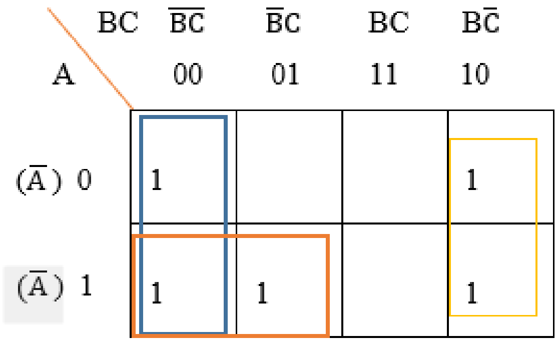


Fig 1. K-Map with filled cells

Make a group of adjacent cells having cell value 1 by following rules of k-Map. For example, we can group the four cells at (0,0), (0,1), (1,0), and (1,1) because they all have an output value of 1. This gives us the implicant AB. We can also group the two cells at (0,3) and (1,3) because they both have an output value of 1 form truth table. This gives us the implicant AC'. Finally, we can group the two cells at (1,0) and (1,3) because they both have an output value of 1. This gives us the implicant BC.

Now combine these implicants to form a simplified Boolean expression:

$$F(A,B,C) = AB + A\bar{C} + BC$$

This expression is equivalent to the original Boolean function, but it is much simpler and requires fewer gates to implement in a digital circuit.

II. Problem Statement

To simplify the complex Boolean expression by using K-Map simulator.

III. Innovative Content

K-map, have been widely used in digital circuit design for decades. In this paper, Karnaugh map introduced a method for simplifying Boolean expressions using a two-dimensional map. The innovative content of Karnaugh's paper lies in the simplicity and effectiveness of the K-map technique, which allowed engineers to simplify complex digital circuits more efficiently than previous methods. Karnaugh's paper remains a cornerstone of digital circuit design, and K-map continue to be used today.

IV. Design Methodology

The problem formulation for K-map in digital circuit design can be approached using programming simulation techniques. One such approach is to use a computer program to simulate the mapping and grouping process in K-map. The program can take as input a Boolean expression and generate a K-map grid, where each cell represents a unique combination of input variables. The program can then identify adjacent cells with the same logical value and group them into prime implicants. The resulting expression can be optimized further by identifying redundant terms using techniques such as Petrick's method or the Quine-McCluskey algorithm [1]. The innovative content in this approach lies in the use of programming simulation to automate the K-map simplification process, which can save time and reduce the chance of human error. Further research in this area aims to develop new algorithms and techniques for optimizing the K-map simplification process using programming simulation, and to explore the use of K-map in more advanced applications such as machine learning and artificial intelligence [1].

A mathematical model for K-map can be implemented using HTML, CSS, and JavaScript. The model can be built by creating an HTML page that displays the K-map grid and the Boolean expression to be simplified [1]. CSS can be used to style the K-map and create the visual representation of the Boolean expression. JavaScript can be used to implement the mapping and grouping process, identify the prime implicants, and optimize the resulting expression. The innovative content in this approach lies in the use of web technologies to create an interactive and dynamic K-map simplification tool that can be accessed from any device with a web browser [2]. The model can also be easily modified and updated as new algorithms and techniques are developed. Further research in this area aims to explore the use of HTML, CSS and JAVASCRIPT Programming to optimize the K-map simplification process and to create more advanced K-map tools using web technologies [2].

V. Development of Simulater

In this Experiment, simulator is developed by using HTML and JavaScript for K-map simplification tool. To develop this experiment, the first step would be to create an HTML file and add the necessary HTML elements such as buttons, input fields, and display areas as shown in Fig 2. CSS can be used to style the elements and create a visually appealing interface. The JavaScript code would then be added to implement the K-map simplification algorithm. The code uses event listeners to detect user input and update the K-map and expression display accordingly [3]. The solveKMap() function is called when the user clicks the "Solve K-Map" button and it applies the Quine-McCluskey algorithm to the minterms and don't cares entered by the user [3] as shown in Fig 2 . The resulting prime implicants are displayed in the "notebook" textarea and the optimized expression is displayed in the "booleanfunction" div.

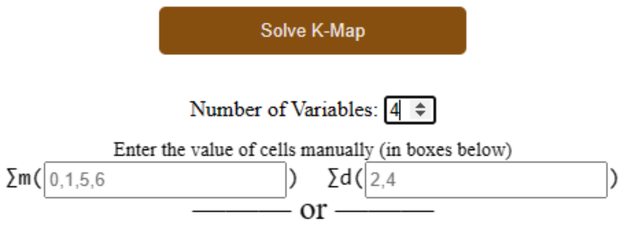
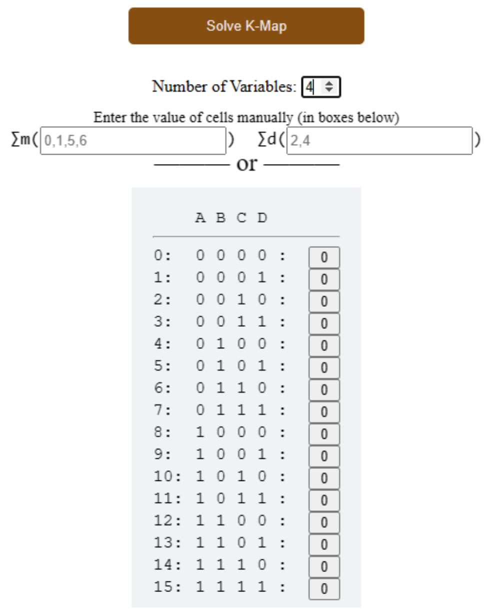


Fig 2. Selection of no. of variables

To develop this experiment simulation, the Second step would be to define variables to store the K-map data, such as the truth table and sequence of variables. The truth Table variable is an array that will hold all possible input combinations and their corresponding outputs. The sequence variable is an array that will hold the order of the variables in the K-map. The varCount variable is the number of variables in the K-map, which is set to 2 in this example [4] form Table 1 . The variables variable is a string that contains the letters A through H, which are used to label the variables. The k-Map variable is an array that will hold the visual representation of the K-map, while cell Size, row, col, offsetX, and offsetY are variables used to determine the size and position of each cell in the K-map [4].



Number of Variables: 4

Enter the value of cells manually (in boxes below)

Σm(0,1,5,6) Σd(2,4)

or

```
    A B C D

0:  0 0 0 0 :  0
1:  0 0 0 1 :  0
2:  0 0 1 0 :  0
3:  0 0 1 1 :  0
4:  0 1 0 0 :  0
5:  0 1 0 1 :  0
6:  0 1 1 0 :  0
7:  0 1 1 1 :  0
8:  1 0 0 0 :  0
9:  1 0 0 1 :  0
10: 1 0 1 0 :  0
11: 1 0 1 1 :  0
12: 1 1 0 0 :  0
13: 1 1 0 1 :  0
14: 1 1 1 0 :  0
15: 1 1 1 1 :  0
```

Fig 3**.** Creation of truth table.

The function first calculates the coordinates of the clicked cell based on the mouse position and the size of the cells. It then checks if the cell is a valid one based on the number of rows and columns in the K-map [5] as shown in Fig 3. If the clicked cell is valid, the function converts the binary coordinates of the cell to a decimal index, and finds the corresponding input element in the HTML page [5]. It then toggles the value of the input element between 0, 1, and 'x' (don't care). After updating the input element, the function calls two other functions: writeMinterms() and createKMap() [5]. WriteMinterms() updates the minterm and don't care lists based on the current input values, while createKMap() updates the visual representation of the K-map based on the input values as shown in Fig 3.

This function simplifyBooleanFunction() takes a Boolean function as input and returns the simplified version of the same Boolean function. It performs a series of simplification steps using Boolean algebraic laws and identities [5].

The input function is first checked if it is a constant (either 0 or 1), in which case it is returned as it is from Table 1. Otherwise, it is stripped of any white spaces and all variables are converted to lowercase letters as shown Fig 1.

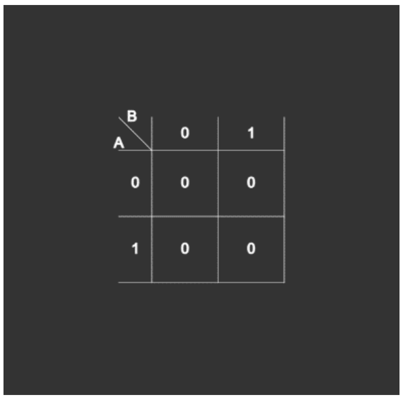


Fig 4. Simulator displays K-Map structure

The code assumes that the input values are already available in a truth table. It also assumes that the dimensions of the Karnaugh Map have been specified in row and col variables. The code uses the cellSize variable to specify the size of each cell in the HTML table that displays the Karnaugh Map [5] as shown in Fig 4. The variables and sequence variables are used to generate the Boolean expression. The output variable in create Boolean Function is used to store the final expression. The visited and K-map variables are used to keep track of which cells in the Karnaugh Map have been visited and their values.

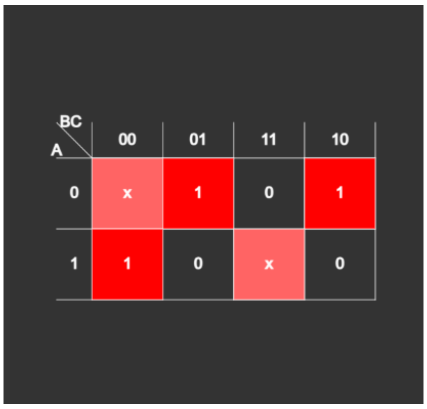
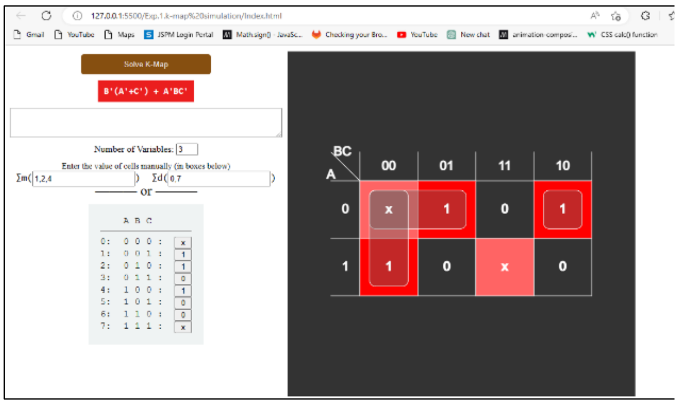


Fig 5. Simulator shows cells with values.

The function takes an optional parameter bg, which defaults to true. This parameter controls whether or not the background of the output values should be drawn. If bg is true, the function will draw a red square for each output value that is equal to 1, and a light red square for each output value that is a don't care ('x') [5]. If bg is false, the function will only draw the text of the output values. The function uses the push() and pop() functions to save and restore the current state of the canvas, so that any transformations applied to the canvas within the function do not affect other parts of the web page . The function translates the origin of the canvas to the center of the Karnaugh map. The function then iterates over each cell in the Karnaugh map. For each cell, the function checks if the k-map array (which likely holds the current state of the Karnaugh map has a value at that cell. If it does, the function checks the value of that cell. If the value is 1, the function fills a red square at the center of the cell. If the value is 'x', the function fills a light red square at the centre of the cell. If bg is false, the function does not draw any squares as shown in Fig 4 .



B'(A'+C') + A'BC'

Number of Variables: 3

Enter the value of cells manually (in boxes below)

Σm(1,2,4) Σd(0,7)

or

```
      A B C
0:    0 0 0 :  x
1:    0 0 1 :  1
2:    0 1 0 :  1
3:    0 1 1 :  0
4:    1 0 0 :  1
5:    1 0 1 :  0
6:    1 1 0 :  0
7:    1 1 1 :  x
```

A \ BC	00	01	11	10
0	x	1	0	1
1	1	0	x	0

Fig 6. Overall window of Simulator

RESULTS

The Karnaugh map simulation would involve constructing a visual representation of the expression using a grid of cells, where each cell represents a unique combination of the input variables. The cells that correspond to logical 1's in the expression are filled in, and adjacent cells with the same value are grouped together to identify terms in the Boolean expression. Once the groups have been identified, the program would use Boolean algebra to simplify the expression by factoring out common terms and applying Boolean rules such as absorption, distribution, and complementation. The simplified expression would then be outputted by the program. The result of the Karnaugh map simulation can be used to analyze and optimize digital logic circuits, as well as to evaluate the complexity and efficiency of Boolean expression.

This Simulator is suitable for minimization of expressions. By applying the Karnaugh map simulation to different expressions, researchers can compare the accuracy, speed, and scalability of different minimization methods.

CONCLUSION

The designer system provides an interactive environment for students through this system make the student is able to perform the process of simplifying Boolean equations by using k-map. This happens by input equations and then applying certain steps of simplifying by fill them in the truth table and then fill k-map, grouping, removing according to rules and procedure until we obtain final expression which is simplest form of the equation. And this means reducing the cost and physical parts of the electronic circuits, also increases the efficiency of the computer. This research concluded that the simulation program designed helps the student to understand the concept of subject of simplification, which is one of the important topics in the study of computer architecture for students of computer technology as well as for students of electronic disciplines. Although, the simulation tools provide good solutions to develop the practical skills for the student in the higher education and recommend through this search The use of computers in training and learning through simulation programs helps in developing the skills of the student and cognition .Also promoting the use of eLearning as well as traditional education. This simulation provides friendly and interactive environment.

FUTURE WORK

Handling more variables: Improve the simulator to work with K-map that have more than five variables. This will involve developing techniques to represent and manipulate larger K-map efficiently.
User-friendly visualization: Create a more interactive and user-friendly interface for the simulator. This could include features like drag-and-drop functionality, real-time updates, and highlighting of related cells or groups.
Comparison with other techniques: Compare the performance of K-map simplification with other methods, such as the Quine-McCluskey method or binary decision diagrams. Determine the strengths and weaknesses of K-map in different scenarios.
Don't care conditions: Extend the simulator to handle "don't care" conditions in K-map. These conditions allow certain combinations of inputs to be ignored during simplification, leading to more optimized logic circuits.

ACKNOWLEDGEMENT

We would like to express my gratitude to Prof. Dr. S. L. Chavan, Head of Electrical Engineering Department for their guidance and support throughout this research paper. We would like to thank the developers and community for providing such a useful and powerful tool. Lastly, We would like to thank Electrical Engineering Department for their financial support, which made this research possible.

REFERENCES

[1] Huang, J.: Programming implementation of the Quine-McCluskey method for minimization of Boolean expression. Department of Biological Sciences, Faculty of Science, National University of Singapore. Retrieved 29/05/16 at https://arxiv. org/ftp/arxiv/papers/1410/1410.1059.pdf(2014

[2] Afaq Ahmad And Al, Development Of Verification Tool For Minimal Boolean Equation , IEEE Technology And Engineering Education (ITEE) ,Vol 8, No 3, https://www.researchgate .net

[3] Abdelrahman, E.: A C++ Karnaugh Map Minimizer-Infinite Variables. Code Project. Retrieved 10/08/2016 at http://www.codeproject.com/Articles/649849/A-CplusplusKarnaugh-Map-Minimizer-Infinite-Variab (2014).

[4] shrinidhi gindi and al,(2015) , Simplification And Simulation Of Digital Circuit In Object Oriented Programming ,IJSRD- International Journal For Scientific Research & Development , Vol.3, No.02 , ISSN(online)2321-0613,p209

[5] Frank M. B.: On the Suppression of Variables in Boolean equations. Discrete Applied Mathematics. Vol. 159 pp255-258. Elsevier (2011)