International Conference on Sustainable Technologies for
Industry 4.0 (STI), 24-25 December, Dhaka

An Extensive Karnaugh Mapping Tool for Boolean Expression Simplification

Md. Saidur Rahman*, Rifat Hasib†, Babe Sultana‡, Md Gulzar Hussain§, Mahmuda Rahman¶,
Muhammad Aminur Rahaman‖
Department of Computer Science and Engineering
Green University of Bangladesh, Dhaka-1207, Bangladesh
{md.saidur.rah5*, rifathasib†}@gmail.com, {babe‡, gulzar§, mahmuda¶,
aminur‖}@cse.green.edu.bd

***Abstract*—The fundamental concept in the design of digital circuits is to reduce the complexity of hardware, make a circuit as simple as possible, which also reduces costs. For obtaining this, we use boolean expression to achieve a lowest possible number of terms and do not encompass any obsolete couple. Karnaugh Map (K-map) and Quine-McCluskey (QM) approach are the most popular specific methods to simplify the Boolean expressions. In this research paper, we have developed an extensive Karnaugh mapping tool which overcomes the existing research's limitation in this related area. We have simultaneously considered to ensure that, our extensive K mapping tool is made easier for primary level digital logic design learners where they can operate with a big number of variables, and run time is optimal. We have implemented the algorithm and the performance evaluation results carried out by making a comparison with C-Minimizer Algorithm and shows that, our extensive K mapping tool outperforms existing approaches in terms of computation time and the percentage of relative error is much lower(2%) which is fulfilled the satisfactory level.**

Keywords: Karnaugh map, Quine McCluskey(QM) technique, Minterms, Boolean Expression.

I. Introduction

Simulation at the logic level is now one of the most widely used electronic circuit activities throughout both construction and test periods. As the quantity of logic to be incorporated into a single chip increases rapidly, higher attempts are needed to optimize the design process. The design of digital logic circuits including Programmable Logic Array (PLA) and Programmable Array Logic (PAL) is one of the main goals [1], where the amount of logic gates should be kept as low as possible, thus reducing the cost of manufacturing of these devices. Among them Karnaugh map is a graphical mechanism to simplify Boolean Expression in the form of a minimum sum of products. Circuit-wise, this results carried in a minimum application of two levels. Its mainly used for many tiny design issues. True indeed that many bigger designs are carried out using separate algorithm computer implementations, but we will gain a lot of insight into the digital logic circuit after studying Karnaugh maps.

In boolean logic to circuit design, we can see that the basic boolean operations are AND, OR and NOT. These operations can be come together to create complicated expressions that can also be translated straight into a physical circuit. For a particular circuit the building cost of a physical circuit is assumed by its boolean expression. Shorter can bring small cost than larger. If a large circuit functionality can be carried out with a small one and the output functionality is same as well as, we all take small one for minimum cost. Many works have been done for simplification of boolean function. First at all, postulates and theorems is used to simplify the boolean function where there are no specific rules [2]. The first and only way to simplify seems to apply postulates theorems and several other common techniques of manipulation.
The popular tabular method which is called as Quine McCluskey (QM) technique is suitable for large number of variables. This tabular method is a system-wise action that can be performed step by step which was first developed by Quine [3] and later raised by McCluskey [4]. As far as we know, it is working with large number of variables. So, it is suitable for computer manipulations. QM methods as works as follows two major activities: a) Determination of Prime Implicates b) Selection of Essential Prime Implicates [5]. The saddening part of this QM algorithm is that, this algorithm is a NP complete and the algorithm's run time increases exponentially with the length of the input [5]. So, for the primary level DLD (Digital Logic Design) students it is very tough to handle QM tabular method. The popular generalization technique which is called mapping method, first suggested by Veitch [6] and mildly altered by Karnaugh [7], offers a simple, straightforward operation to minimize boolean functions. Basically the K map is very fast for small number of variables at the educational level and best of our knowledge, when we get more than 5 variables, it's difficult to do by side. In this research paper our aim is to work for primary level students who are able to handle with large number of variables when they try to minimize boolean function using mapping method.

Karnaugh map is really an effective tool with few variables in algebra logic processing. This tool is used for various purpose in engineering subject. Author [8] motivated by the Karnaugh map, they introduce a Karnaugh-map-like online cable virtualization embedding algorithm that involves: online scheduling technique and Karnaugh-map-like embedding algorithm. The identification of all candidate keys is an

978-1-7281-6099-3/19/$31.00

significant step in the design of relational databases. Author [9] give a simpler way for using the concept of an algorithm for relational database and the Karnaugh map technique to handle functional dependency sets to get all candidate keys. Above all discussions, working with large number of variable for K- mapping technique will be a great contribution in this field. Author [10] works with a prominent fundamental problem of digital circuit design which is generalized by Type 2 problem. This article is part of their ongoing attempt to modernize methods through ' large ' boolean algebraic methods to handle basic digital circuit design issues. A altered method of the Karnaugh map is proposed by author [11] which provide the students with the capacity to see precisely how a K-map was made and enable them to ask any questions more obviously during the digital logic design lesson. But there is no concept for working with large number of variable. Author [12] proposed an EXTENDED K-MAP which is using a single Karnaugh map to minimize various circuit outputs. And, in their algorithm basically shows that, the multiple K-maps accumulated in a single K-map. Their experimental result and analysis gives us an idea that, they only work with (n<= 5) variables not more than. So, obviously our contribution will be a great one in this area.

The rest of this paper is oriented as follows. An overview of the kaurnaugh map tool is described in section II. The details of our implementation are described in section III. In section IV, we have described our performance evaluation with impact of result analysis. And finally, we have turned out our conclusion with the direction of future works in section VI.

II. Kaurnaugh Map Tool Overview

Kaurnagh maps are used to simplify boolean function by exploiting the pattern-recognition capacity of humans. In this paper, a framework has been designed to implement kaurnagh map. The system takes a boolean expression as input. Then depending on n variables it forms 2^n position in the K-map where each position containing a binary digit that represents the output function for that input combination. Then the system creates minterm groups with adjacent 1s having an area of power of 2 (i.e. 1, 2, 4, 8,...). Finally the algebraic minterms are derived by examining which variables stay the same within each box.

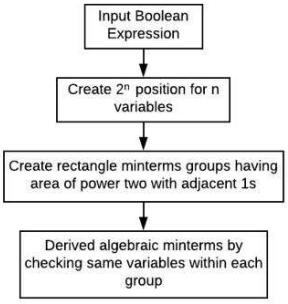

Fig. 1: Block diagram of the proposed kaurnaugh map implementation system

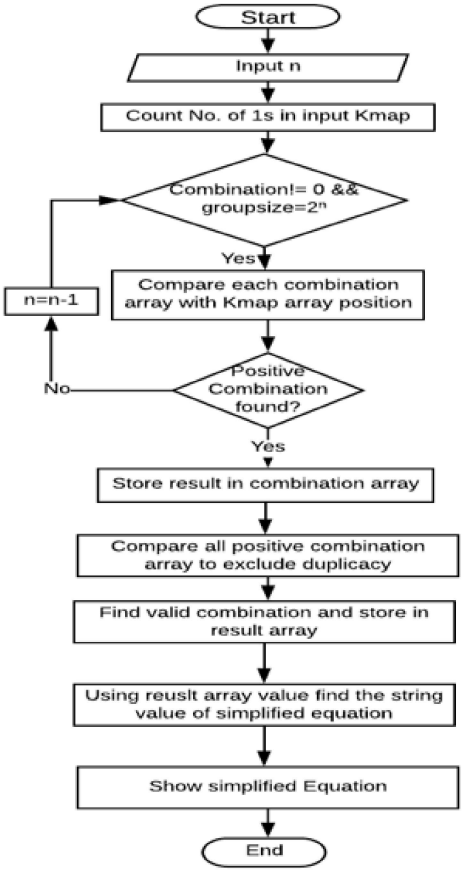

Fig. 2: Work flow diagram of the kaurnagh mapping tool.

III. Methodology

A. Creating 2^n Positions & Minterm Groups

To construct K-map the system create 2^n positions for n variables where each positions are ordered in gray code to ensure that only one variable changes between adjacent cells in each pair. A binary digit is stored in each position of K-map to represent the functions output for that inputs combination. Our system proposed an algorithm which consists of two one dimensional array where one is used to hold the position of kaurnaugh map and the other is used to hold the function output for that particular position. As for example, Fig.3 shows the index mapping between the position array and function output array of a 3 variables boolean function. Based on the number of variables the system generates all the possible groups of minterms for the given function and the combinations are represented by an two dimensional Array List. Then the Array List is passed to the **Group_checker** function is to check the possible group of minterms with adjacent 1s for that particular boolean expression.

B. Checking Group

In kaurnaugh map group of minterms are formed having an area of power of 2 i.e. 1, 2, 4, 8, 16, ... The **Group_checker**

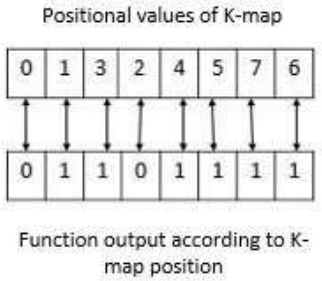

Fig. 3: An example of mapping between position and function output of a boolean function consists of 3 variable.

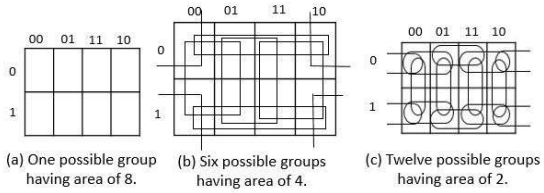

Fig. 4: An example of all the possible combinations of minterm groups of 3 variables boolean function.

Algorithm 1: K_map_Algorithm

Input: kMap_positionArray[], function_output[], variableSize
Output: validCombinationArray[],resultGroup[]
Initial variableSize ← values from users;;
Initial kMap_positionArray ← values from users;
Initial function_output ← values of function input according to position;
Initial combination[][] ← possible combination value as array elements;
Initial match ← 0;
Initial start ← 0;
Initial ending ← combination[][].length;
for *i= 0 to ending* **do**
  initial validCombination ← call_groupChecker(kMap[], combination[i], i);
  if *validCombination ≠ 0* **then**
    increase match by 1;
  end
end
Initial resultGroup[]← call positiveCombination[groupSize];

function checked every possible combination that can be formed by adjacent 1s. It also considered the overlapping groups in order to make larger groups without any 0s. Fig. 4 shows that, there are 2 possible minterms groups of having area 4 for the example shown in the Fig. 3. Then the **Group_checker** function return the number of groups to the main function .

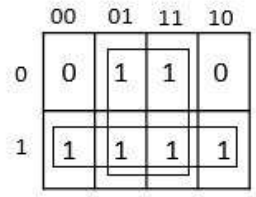

Fig. 5: Groups of minterms for the boolean function given in the Fig. 2

Algorithm 2: Group_Checker Algorithm

Input: kMap[], combinationArray[], combinationNumber, maxCombinationCount, groupMax1
Output: validGroupCombination
Initial positiveCombination[] ← maxCombinationCount;
Initial start ← 0;
Initial ending ← combinationArray[].length;
Initial flag ← 0;
for *i= 0 to ending* **do**
  if *kMap[combinationArray[i]] = 1* **then**
    increase flag by 1;
end
if *flag = groupMax1* **then**
  positiveCombination[combinationNumber]← 1;
  increase start by 1;
  return 1;
else
  return 0;
end

C. Derivation of Algebraic Minterms

when groups are formed the following algorithm 3 counts the number of 1s in each group. According to number of 1s it defined how many variables are required to express this minterm. When number of variables are defined, it considered the common bit among the positions and derived the variable for this position.

Algorithm 3: Number of 1s Counter

Input: kMap[]
Output: Numbers of 1's
Initial counter ← 0;
Initial lentth ← kMap[].length;
for *i=0 to length* **do**
  if *kMap[combinationArray[i]] is 1* **then**
    increase counter by 1;
  end
end
return counter;

IV. Performance Evaluation

A. Environmental Setup

We evaluate the developed kaurnaugh mapping desktop tool under the following environment:

- **Operating System:** Windows 10 64bit
- **Processor:** Intel Core i7-2.00 GHz
- **RAM:** 16 GBytes
- **IDE:** Microsoft Visual Studio Express 2019[13]
- **Programming Language:** C#

B. Results and Discussion

1) Impacts of increasing number of Variables with relative error: After completion of the tool, a group of 100 students who are studying Digital Logic Design (DLD) and pursuing a Bachelor of Science in Computer Science and Engineering tested it. Three sets of Boolean algebra expressions with variable numbers 3, 4, 5, 6, and 7 are provided to the students to test the tool where every set contain different numbers of expressions. It is done because expression with higher number of variables is harder to simplify than the lower number of variables. Table I shows absolute and relative error for boolean algebra expressions with variable numbers 3, 4, 5, 6, and 7.

TABLE I: Table of relative error

Number of Variables	Number of Expressions taken for testing	Number of Correctly Simplified Expressions	Absolute Error	Relative Error
3	100	98	2	0.020
	200	195	5	0.025
	300	292	8	0.027
4	100	96	4	0.041
	200	190	10	0.053
	300	282	18	0.064
5	100	93	7	0.075
	200	187	13	0.070
	300	279	21	0.075
6	50	44	6	0.136
	100	83	17	0.205
	150	126	24	0.190
7	25	22	3	0.136
	50	42	6	0.14
	75	63	12	0.190

From Table I we can see that relative error is increasing with the increase in the number of expressions and number of variables. We found minimum relative error 0.02 for 3 variables and maximum for 6 variables expressions. Fig 6 shows the relation between the value of the number of variables, number of expressions and relative error which contain in Table I.

2) Computation Complexity: For testing the efficiency we have done analysis of computation time analysis which is found from the developed application. We have tested time needed for each number of variables which is given in the Table II. Using these data graph 7 can be drawn where we can see computation time is increasing with increase of the number of variables. From the graph we can see, for 7 variable we need only 312.43 milliseconds which is much efficient.

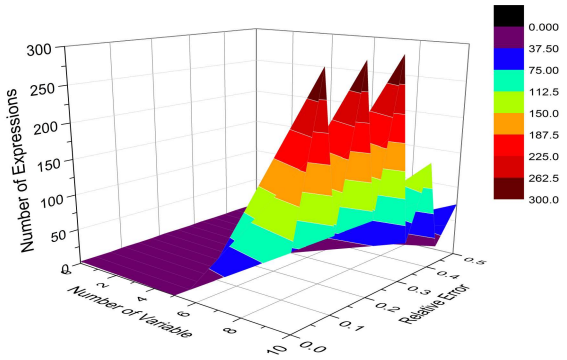

Fig. 6: Comparison of number of expressions Vs number of variables Vs relative error

Comparison is done with the result found on paper [6] to compare the efficiency of our system as no recent work on tool development to minimize the boolean expression. Table II and 7 shows the comparison and we can see that our Karnaugh mapping tool need less time than C-Minimizer Algorithm for 3 to 7 variables. The reason for this less time is, our Karnaugh mapping tool removes the repeated indices of minterms with use of gray code sequence.

TABLE II: Comparison between our Karnaugh mapping tool vs C-Minimizer algorithm[6] based on computation time

Number of Variables	Our System (ms)	C-Minimizer Algorithm[6] (ms)
3	10.1	15
4	12.52	16
5	25.37	32
6	146.66	166.33
7	312.43	379.66

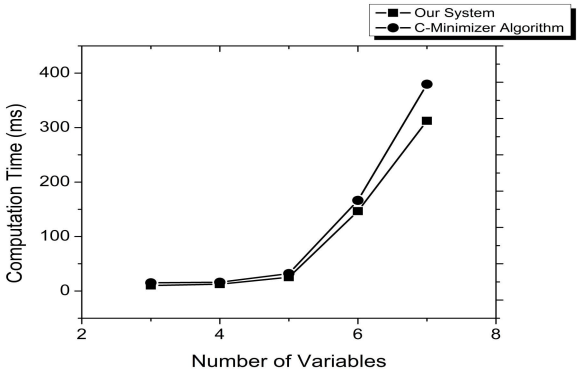

Fig. 7: Comparison between our Karnaugh mapping tool vs C-Minimizer algorithm[6] with number of variables Vs computation time(ms).

V. Conclusion & Future Work

In this research paper, we presented an attempt to represent an extensive Karnaugh mapping tool, which offers an easy

solution for simplifying boolean expressions with large number of variables. Here, we showed that, our findings demonstrate a positive correlation between the conceptual solution and the expected outcomes of the mathematical model. As we mentioned that, our aim was to be working for primary level digital logic design students, so, some situational relative error may cause anxiety for them. But, this percentage of anxiety level is much lower.

Working with more variables and decreasing the complexity of algorithm which causes huge computation time, that will be our future direction.

ACKNOWLEDGEMENT

This research was supported and funded by Green University of Bangladesh.

REFERENCES

[1] R. Czerwiński, D. Kania, and J. Kulisz, "Fsms state encoding targeting at logic level minimization," *Bulletin of the Polish Academy of Sciences: Technical Sciences*, 2006.

[2] M. H. Stone, "Postulates for boolean algebras and generalized boolean algebras," *American Journal of Mathematics*, vol. 57, no. 4, pp. 703–732, 1935.

[3] W. V. Quine, "The problem of simplifying truth functions," *The American Mathematical Monthly*, vol. 59, no. 8, pp. 521–531, 1952. [Online]. Available: http://www.jstor.org/stable/2308219

[4] E. J. McCluskey Jr, "Minimization of boolean functions," *Bell system technical Journal*, vol. 35, no. 6, pp. 1417–1444, 1956.

[5] T. K. Jain, D. S. Kushwaha, and A. K. Misra, "Optimization of the quine-mccluskey method for the minimization of the boolean expressions," in *Fourth International Conference on Autonomic and Autonomous Systems (ICAS'08)*. IEEE, 2008, pp. 165–168.

[6] D. Nasa and U. Ghose, "C–minimizer algorithm: A new technique for data minimization," *International Journal of Computer Applications*, vol. 44, no. 17, pp. 15–19, 2012.

[7] M. Karnaugh, "The map method for synthesis of combinational logic circuits," *Transactions of the American Institute of Electrical Engineers, Part I: Communication and Electronics*, vol. 72, no. 5, pp. 593–599, 1953.

[8] M. Yang, Y. Li, L. Zeng, D. Jin, and L. Su, "Karnaugh-map like online embedding algorithm of wireless virtualization," in *The 15th International Symposium on Wireless Personal Multimedia Communications*. IEEE, 2012, pp. 594–598.

[9] Y.-S. Zhang, "Determining all candidate keys based on karnaugh map," in *2009 International conference on information management, innovation management and industrial engineering*, vol. 4. IEEE, 2009, pp. 226–229.

[10] A. M. A. Rushdi, "Handling generalized type-2 problems of digital circuit design via the variable-entered karnaugh map," *International Journal of Mathematical, Engineering and Management Sciences (IJMEMS)*, vol. 3, no. 4, pp. 392–403, 2018.

[11] M. E. Holder, "A modified karnaugh map technique," *IEEE Transactions on Education*, vol. 48, no. 1, pp. 206–207, 2005.

[12] P. Das and B. Mondal, "Extended k-map for minimizing multiple output logic circuits," *International Journal of VLSI design & Communication Systems*, vol. 4, no. 4, p. 1, 2013.

[13] Microsoft. (2019) Visual studio express 2019. [Online]. Available: https://visualstudio.microsoft.com/vs/express/