SOFTWARE FOR THE MINIMIZATION OF THE LOGIC FUNCTIONS

Mohammad Osiur Rahman[1]
&
Dr. Md. Nurul Islam[2]

Abstract:
The design and operation of software based on Karnaugh mapping method was undertaken to simplifying the logic function and to draw the relevant circuits. In the present software, the user selects the input variable and gives the decimal value of the corresponding minterms. The software constructs the Karnaugh map (K_map) corresponding to the input variable. Entries of 1 are made in the cells corresponding to the minterms that make the function a 1 in Sum-of-Product (SOP) form and 0's are entered in the remaining cells. The software simplifies the Boolean function by correctly grouping the adjacent 1's.On simplification the software implement logic circuit automatically. The software convert numbers from one number system to another, perform basic binary arithmetic, evaluate and simplify a Boolean and logic functions. It also analyzes a combinational logic circuit to obtain its switching functions, design a combinational logic circuit, simplify the design, implement the design using SSI, MSI, and programmable logic devices. It further analyzes synchronous sequential logic circuits, design synchronous logic circuits using state diagram and ASM chart, simplify the design circuits, and implement the design circuits using SSI, MSI, and programmable logic devices. Simulate the design circuits using all software. It is also applicable for Neural Networks software for the minimization of the logical functions and may follow Canonical Binary Functions.

Keyword: **K_map, Sum-of-Product (SOP), minterms**

Introduction

Once the expression for a logic circuit has been obtained, we may be able to reduce it to a simpler form containing fewer terms or fewer variables in one or more terms. The new expression can then be used to implement a circuit that is equivalent to the original circuit but that contains fewer gates and connections.

It is wondered how a computer can do something like software minimization! These are things that, just a few decades ago, only humans could do. Now computers do them with apparent ease. A "chip" made up of silicon and wires do something that seems like it requires human thought by logic function.

If you want to understand the answer to this question down at the very core, the first thing you need to understand is something called **Boolean logic.** The great thing about Boolean logic is that, once you get the hang of things, Boolean logic (or at least the parts you need in order to understand the operations of computers) is outrageously simple.

---

[1] Lecturer, Department of Computer Science & Engineering, IIUC
[2] Professor, Department of Computer Science & Engineering, IIUC

There are three, five or seven simple gates that you need to learn about, depending on how you want to count them (you will see why in a moment). With these simple gates you can build combinations that will implement any digital component you can imagine. These gates are going to seem a little dry here, and incredibly simple, but we will see some interesting combinations in the following sections that will make them at a lot more inspiring.

The simplest possible gate is called an "inverter", or a **NOT gate**. It takes one bit as input and produces as output its opposite. The table below shows a logic table for theNOT gate and the normal symbol for it in circuit diagram: You can see in this figure that the NOT gate has one input called **A** and one output called **Q** ("Q" is used for the output because if you used "O", you would easily confused it with zero). The table shows how the gate behaves. When you apply a 0 to A, Q produces a 1. When you apply a 1 to A, Q produces a 0. Simple.

Some time cannot occur at the input. These times can produce a 0 or a 1 at the output without effecting system operation. They are called **don't-time** states and can be assigned a 0 or 1 in to minimize the function.

The Karnaugh Map *[1]* (K_map) is a graphical device used to simply a logic equation or to convert a truth table to its corresponding logic circuit in a simple, orderly process. Although a Karnaugh Map can be used for problems involving any number of input variables, its practical usefulness is limited to six variables. The following discussion will be limited to problems with up to four inputs.

Simplification of a Boolean function plotted on a K_map is a process of correctly grouping *[2]* the adjacent 1s. Entries of 1 are made in the cells corresponding to the minterms that make the function a 1 in Sum-of-Product (SOP)*[3]* form. 0's are entered in of the remaining cells.

The over all system block diagram chart is shown blow.

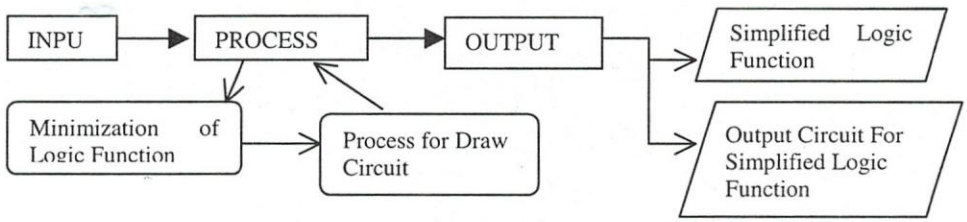


Fig: 1: Overall System Flow Chart

Design Procedure of the Software

Number of input variable: In this system there are four variables used. They are: (a) "1" for a single variable A (say), (b) "2" for two variables AB, (c) "3"for three variables ABC and (d) "4" for four variables ABCD. After selecting the input variables, possible product of variables are selected. These

are used as decimal values of the corresponding minterms. (Flow chart shown in fig 2)

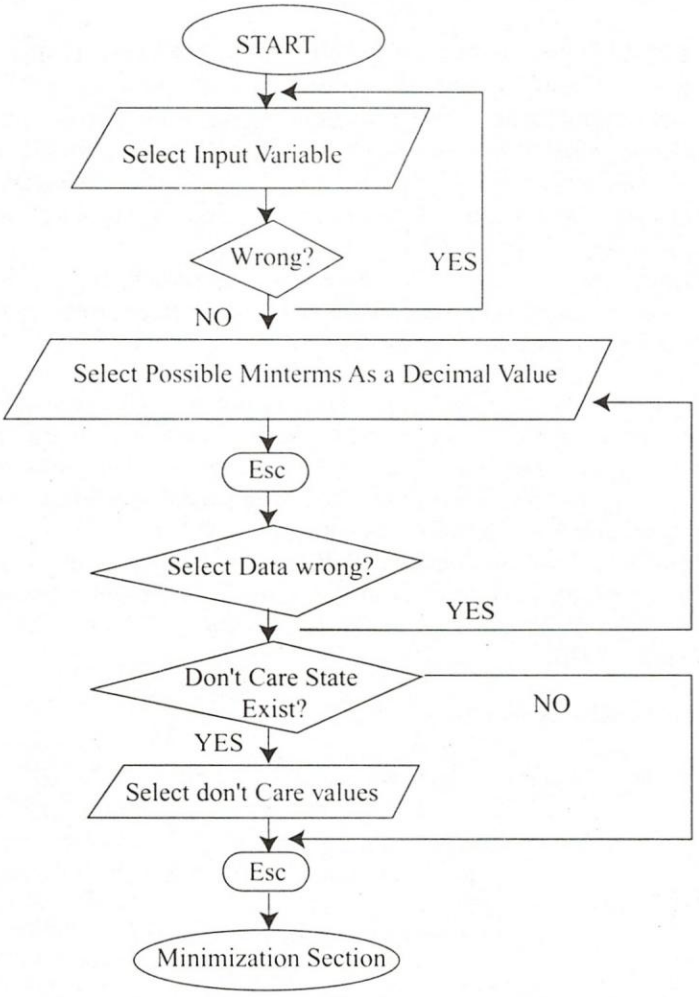


Fig: 2: Input Portion Flow Chart.

For Minimization of Logic Functions Grouping Rule [6] are used as follows

Rule 1: If all entries in an N-variable K_map are 1s, the group size is 2^N and the corresponding minimized function is given by f(A,B,.....) = 1 .
Rule 2: For N variables, the largest nontrivial group size is 2^(N-1). All group sizes are powers of 2. For N variables, group sizes of 2^N, 2^(N-1), 2^(N-2), 2^(N-3),.....2^0 are allowed.
Rule 3: In any group it must be possible to start at a cell and travel from cell to cell, with moves of one unit length, through each cell in the group without passing through any cell twice and return to the starting cell.

Simplifying Algorithm

Using the rules for forming groups, we now apply the rules to arrive at a minimum SOP function as follows (Flow chart shown in fig-3):

Step_1: Construct the K_map.
Step_2: Enter a 1 on the Karnaugh map for each fundamental product that corresponds to 1 output in the truth table. Enter 0s elsewhere.
Step_3: If all entries in an N-variable K_map are 1s, the group size is 2^N and the simplified function is given by f(A,B,.....) = 1 . Else go to Step_4.
Step_4: Encircle the octets, quads, and pairs *[4]*. Remember to roll and overlap to get the largest groups possible.
Step_5: If any isolated 1s remain, encircle them.
Step_6: Eliminate redundant groups if they exit.
Step_7: For each group, determine which literals remain constant in each cell in the group. These literals when ANDed together form the product terms of the SOP function. For x groups, there will be x product terms.
Step_8: Write the Boolean equation by ORing the products corresponding to the encircle groups.

Procedure For Don't Care *[5]*

Step_1: First, all don't care consider as normal input value, and then compute the simplified logic function
Step_2: Only for don't care input, compute the simplified logic function.
Step_3: Then two simplified function is compared.
Step_4: Common Product terms are removing.

Circuits Implementation
The simplified logic function, which we get from K_map, is Sum-of-Product (SOP) form. Each of the SOP expressions consists of two or more AND terms (Products) that are ORed together. Each AND term consists of one or more variables appearing in either complemented or un-complemented form. For example, in the SOPs expression –

$ABC + \overline{A}B\overline{C}$. The first AND product contains the variables A, B and C in their un-complemented (not inverted) form. The second AND term contains A and C in the complemented (inverted) form.

For four input variable, in the K_map there are maximum eight-isolated cell they are (0,3,5,6,9,10,12,15) and (1,2,4,7,8,11,13,14) group only. So that the maximum number of inputs of OR gate is eight. Thus we use two or more but less than or equal to eight output of the AND gates are the input of the OR gate.

Since each AND term consists of Two or more input variable and in the present system maximum four input variables are used so that the number of input of AND gates is four.

In this system, in order to implement simplified logic circuit, there are only three kinds of logic gates are necessary. They are: NOT gate (inverter), AND gate and OR gate.

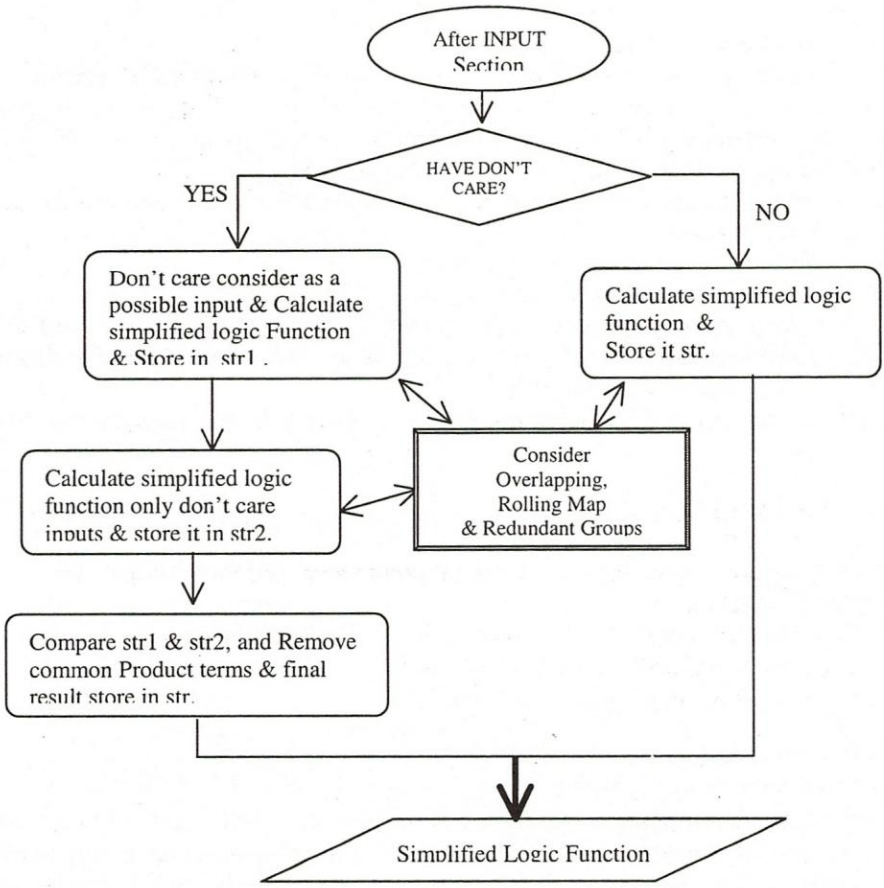


Fig 3: Karnaugh Map Minimization Portion

Algorithm to Draw Circuit

First of all, we get a Sum-of-product form. Now apply the following steps:

Step_1: Calculate how many AND gates are used and Calculate the input variables either complemented or un-complemented which is used and paste the input variable in appropriate place and store X_coordinate.

Step_2: Detect the appropriate place of the Logic gates to overcome the overlapping of the logic gates.

Step_3: Paste AND gates in appropriate place and Save the output coordinate of the AND gates.

Step_4: Appropriate OR gate is selected on the base of number of AND gates and direct inputs.

Step_5: Link between output of AND gates or direct input and input of the OR gate.

Protection from overlapping of the logic circuits

In this system, I get from K_map analysis and circuit implementation analysis, maximum eight AND gates is used at one time and if two or more AND gate is used then one OR gate is used. Since maximum four input variables is used, maximum four input line are used as input line of AND gate.

The result which we get from K_map is Sum-of-Product form, and 4_input AND come first, next 3_input AND, then 2_input AND and at last direct input come.

We consider, 8_input OR gate as follows:

(1) First input: If 1st input is come from 4_input or 3_input AND gate then connected the last (8th) input line of the OR gate. On the other hand, if 2_input AND gate then connected the first (1st) input line of the OR gate.

(2) Second input: If 1st input is 4_input AND gate and 2nd input is 4_input AND gate then connect 7th input line of the OR gate, Else if 2nd input is 3_input AND gate then shift the Y_coordinate of output of 3_input AND gate value and connect the 7th input line of the OR gate. Else if 2_input AND gate then connect the 1st input line of the OR gate .On the other hand, if 1st & 2nd input is 2_input AND gate then 2nd input AND gate connects to the 2nd input line of the OR gate.

(3) Third input: Check Four /Three input AND gate then connects bottom to top orderly input line of the OR gate, else connects top to bottom orderly input line of the OR gate.

(4) 4th, 5th, 6th, 7th, 8th input: Repeat the step of third input.

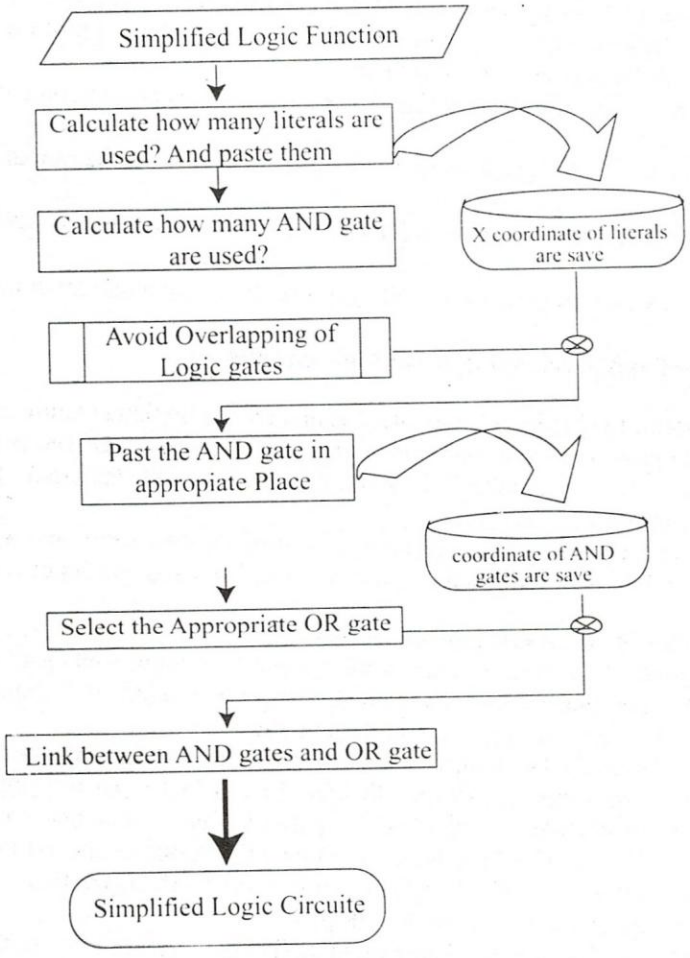


Fig 4: Simplified Circuit Drawing Portion

Operation of this Software

When system is run the variable selection window is displayed on the screen. Then the input variable number is selected by entering the appropriate decimal number. Then the input value is taken from the table by entering input candidate minterms values. After

completing the data entry, then Esc key is pressed. Shows the corresponding K_map. Press Enter then shows the result. Follows menu to show the simplified circuit.

Conclusion

The system is tested thoroughly for any possible error by using actual data. Error checking is done as tight as possible. As the software is written in C's large memory model, memory management cannot be done efficiently. The software works only when the number of input variables is less than or equal to four. The "Circuit drawing part" of this system will be used to draw any circuit for given logic function. The system is developed in such a way that further enhancement like – to extent the number of input variables, to draw circuits by using NAND, XOR, NOR etc gates, can is possible to implement.

REFERENCES:

[1] Ronald J. Tocci. "Karnaugh Map Method", Digital Systems (July, 1996), pp 114 - 12.

[2] Ronald J. Tocci. "Simplifying Logic Circuits", Digital Systems (July, 1996), pp 102.

[3] Ronald J. Tocci. "Sum-of-Products Form", Digital Systems (July, 1996), pp 100.

[4] Albert Paul Malvino. "Karnaugh Simplifications ", Digital Computer Electronics (1991), pp 73 - 75.

[5] Dr. V. K. Jain. " Don't Care States ", Switching Theory And Digital Electronics (1992), pp 88.

[6] Karim And John. " Binary Space and Karnaugh Maps ","Digital Electronics", pp 40 - 48.

Appendix

** Obtain a minimum sum-of–product expressions for the following switching function:

$$f(ABCD) = \sum m(0,2,5,7,8,10,15).$$

And draw the simplified logic circuit.

Figure App_1(a) shows the corresponding simplified logic function.

And Figure App_1(b) shows simplified logic circuit, which is the output of this system.

KARNAUGH MAP

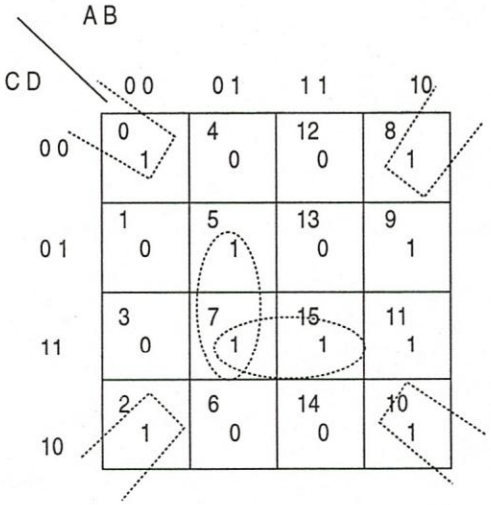
Select Input Variable: | 1 | 2 | 3 | 4 |

< N > EW

< C > IRCUIT

< E > XIT

Possible Input	Select Input	Don't Care
0	*	
1		
2	*	
3		
4		
5	*	
6		
7	*	
8	*	
9		
10	*	
11		
12		
13		
14		
15	*	



Reduce Result:

$$f(ABCD) = \overline{A}BD + BCD + \overline{B}\overline{D}$$

Fig : App_1(a)

Output Circuit For Simplified Logic Function

$$f(ABCD) = \overline{A}BD + BCD + B\overline{D}$$

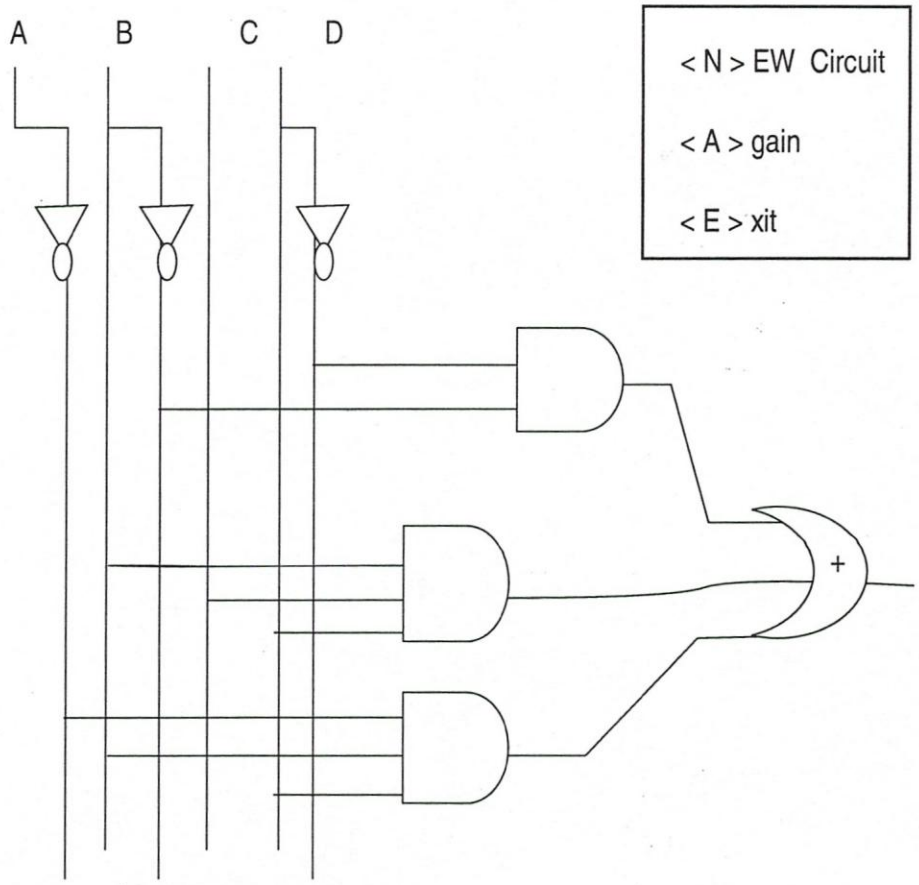


Fig : App_1(b)