EXTENDED K-MAP FOR MINIMIZING MULTIPLE OUTPUT LOGIC CIRCUITS

Palash Das[1], Bikromadittya Mondal[2]

[1]Department of Computer Science and Technology, Bengal Engineering and Science University, Shibpur, Howrah, India.
[2]Department of Computer Science and Engineering, B P Poddar Institute of Management and Technology, Kolkata, India.

ABSTRACT

Minimization of multiple output functions of a digital logic circuit is a classic research problem. Minimal circuit is obtained by using multiple Karnaugh Maps (K-map), one for each function. In this paper we propose a novel technique that uses a single Karnaugh Map for minimizing multiple outputs of a single circuit. The algorithm basically accumulates multiple K-Maps into a single K-Map. Finding minimal numbers of minterms are easier using our proposed clustering technique. Experimental results show that minimization of digital circuits where more than one output functions are involved, our extended K-Map approach is more efficient as compare to multiple K-Map approach.

KEYWORDS

Boolean Algebra, Karnaugh Map, Digital Logic Circuit, Clustering.

1. INTRODUCTION

Simplification of logic function actually reduces the number of digital logic gates required to implement digital circuits. This results the reduction of the size of the circuit. The cost of the circuit will also be reduced. There are a number of techniques proposed to minimize logic functions. The Boolean algebra was proposed by Boole [1]. Then C.E. Shannon [2] showed the design of digital circuits using Boolean algebra. Karnaugh [3] proposed a new technique for simplifying Boolean expressions using a map. Quine and McCluskey [4][5] proposed an algorithmic based technique for simplifying Boolean logic functions. S. K. Petrick [6] also did significant work on Boolean function minimization. Heuristic based techniques [9][10] were proposed for fast minimization of Boolean functions.

Generally Boolean functions are expressed in terms of two standard forms: the sum–of–products and the product–of–sums. Each combination of variables in a sum-of-products function is called a *minterm*; in the product-of-sums form, they are called *maxterms*. This paper presents the use of minterms to create Extended Karnaugh Map (K-map), although the same technique can also be used for creating Extended K-map for Boolean functions by maxterm expressions. This Extended K-map can accommodate more than one output functions at a time and helps to design the entire minimized circuit at a time. It is obvious that the number of variables of all the functions will be same as this new technique of Extended K-map has been developed to design specific circuits of a particular instance of time. If there are m number of n variable output functions then this Extended K-map will have 2^n cells which will accommodate all m functions and will produce minimized Boolean expressions for designing the entire circuit. Figure 1 shows the block diagram of the system. It is efficient as it reduces the design complexity significantly in case of multi-output circuits.

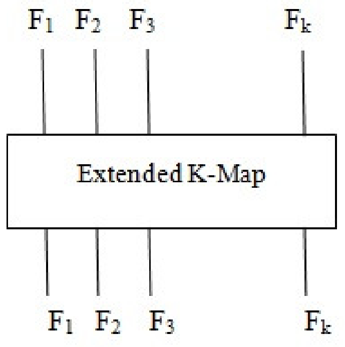

Figure 1: Block diagram of an Extended K-map

2. PROPOSED WORK

This work is actually the extended version of K-map which is based on basic K-map principle but with some additional features. The Extended K-map algorithm is presented in section 2.1. The cluster generation and selection algorithm is given in section 2.2. The illustration of our proposed algorithm is given in section 2.3.

2.1 The Extended K-map Algorithm

Input: k number of n variable Boolean functions.
Output: k number of minimized Boolean expressions for designing the circuit.

```
Step 1: Draw a map (matrix) of $2^n$ cells; where n is the number of variables of all functions
Step 2:
        Step 2.1: Initialize all variables
        Step 2.2: Set array_functions = all output functions to be minimized,
                  NO_OF_FUNCTIONS = length of the array_functions
Step 3: REPEAT the following steps till NO_OF_FUNCTIONS! = 0
        Step 3.1: generate_cluster will return the clusters one by one of the function passed as
                  parameter.
        Step 3.2: All the returned clusters will in taken into a stack. And later they will be taken
                  out in last in first out (LIFO) basis for getting the minimized outputs
        Step 3.3: NO_OF_FUNCTIONS; (decrementing the number of remaining functions to be
                  minimized)
Step 4: END OF LOOP
Step 5:
        Step 5.1: REPEAT until the stack is empty
        Step 5.2: Now pop the first cluster
        Step 5.3: Find the elements (groups) of the cluster and form the minimize Boolean
                  expressions(in SOP or POS form) using basic K-Map principle to design the
                  minimized circuit
```

2.2 The Clustering Algorithm

This algorithm actually describes the process of making cluster of the corresponding output functions that has been passed as parameter. It returns all the clusters one by one and stores in to the stack. The *array_functions* is basically a pointer to an array which holds all the *minterm* or *maxterm* values of the corresponding functions passed as argument to the process. We are considering SOP forms for this process assuming the POS form of the algorithm can be easily changed as like our basic K-map principle.

Step 1: In the first iteration initialize the empty cell by putting 1's on top in the cell for the corresponding minterms of the first function.
 Step 1.1: Make initial groups of those 1's using basic K-map principle
 Step 1.2: Assign a name to each group (e.g. X1, X2 etc.)
 Step 1.3: Take all those groups of the first function into a cluster (e.g. C1={X1, X2})
Step 2: In the other iteration put 1's directly on the cells if the cells are still empty after putting the value of previous functions
Step 3: If the cell is not empty; check the previous symbol
 Step 3.1: If the previous symbol of the cell is '1', make it don't care by putting 'X' just below the '1'
 Step 3.2: If the previous symbol of the cell is 'X', make it don't care by putting '1' just below the '1'
 Step 3.3: Make groups using basic K-map principle of the uncovered 1's and the value of the cell just changed from '1' to 'X' or 'X' to '1'
 Step 3.4: Assign a name to each of those groups (e.g. X3, X4 etc.)
 Step 3.5: Take all those groups of each function into a cluster (e.g. C2={X1, X3, X4})
Step 4: Return the clusters one by one to the top of the stack

2.3 Example

Let us consider the following four 4-variable functions for minimization.

F1 = A'B'CD + A'BCD + ABCD + AB'CD + ABC'D' + ABC'D + ABCD'
F2 = A'B'C'D' + A'B'CD' + AB'C'D' + ABCD
F3 = A'B'C'D' + A'B'CD' + AB'C'D' + AB'CD'
F4 = A'B'C'D' + A'B'C'D + A'B'CD + A'B'CD' + AB'C'D' + AB'C'D + AB'CD + AB'CD'

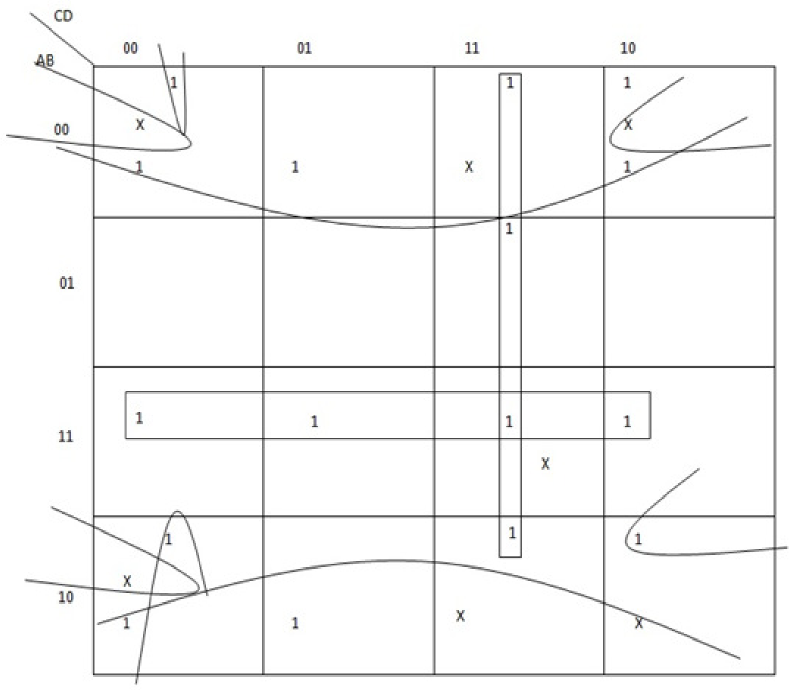

Figure 2: Extended K-map accommodating all four functions

Figure 2 shows the Extended K-map that holds all the four functions. The grouping is done based on our Extended k-map algorithm.

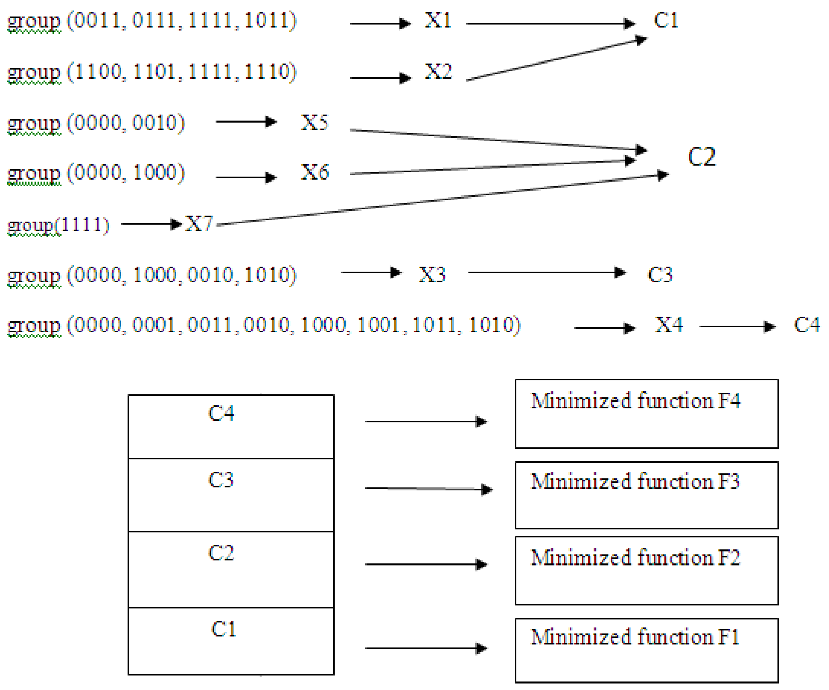

Figure 3: Stack of clusters

From the stack we will get clusters one by one which are nothing but the minimized forms of the corresponding functions.

C1= {X1, X2} $\longrightarrow$ NO_OF_FUNCTIONS -- OUT1 = AB + CD
C2= {X5, X6, X7} $\longrightarrow$ NO_OF_FUNCTIONS --OUT2 = B'
C3= {X3} $\longrightarrow$ NO_OF_FUNCTIONS --OUT3 = B'C'D' + A'B'D' +ABCD
C4= {X4} $\longrightarrow$ NO_OF_FUNCTIONS --OUT4 = B'D'

3. Experimental Results And Simulations

Table 1 shows the experimental results of our proposed technique for different circuits.

Table 1: Experimental results

Number of variables	**Functions under experiment**	**Minimized output using Extended K-Map**	**Reference Extended K-Map**
2	1. F1(A,B) = E (1, 2) 2. F2(A,B) = E (1, 3)	1. K1 = B' 2. K2 = A'	Figure 4
3	1. F1 = E1 (0, 1, 2, 3, 4, 6) 2. F2 = E2 (3, 5, 6, 7) 3. F3 = E3 (0, 2, 4, 6)	1. K1 = A' + C' 2. K2 = AB+BC+CA 3. K3 = C'	Figure 5
4	1. F1 = E1 (3, 7, 11, 12, 13, 14, 15) 2. F2 = E2 (0, 2, 8, 15) 3. F3 = E3 (0, 2, 8, 10) 4. F4 = E4 (0, 1, 2, 3, 8, 9, 10, 11)	**1.** K1 = AB + CD **2.** K2 = B'C'D'+ A'B'D' +ABCD **3.** K3 = B'D' **4.** K4 = B'	Figure 6
5	1. F1 = E1 (1, 2, 3, 5, 7,11, 13, 17, 19, 23, 29, 31) 2. F2 = E2 (8,9,13,12) 3. F3 = E3 (0, 1, 5, 4) 4. F4 = E4 (8,12) 5. F5 = E5 (24, 25, 27, 26, 30, 31, 29, 28)	1. K1 = A'B'E + B'C'E + A'B'C' D +AB'DE +A'CD'E+ ABCE+A'C'DE 2. K2 = A'B'D' 3. K3 = A'B'D' 4. K4 = A'BD'E 5. K5 = AB	Figure 7

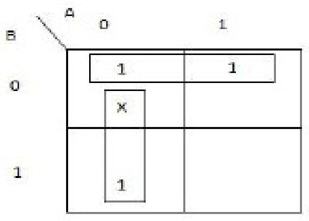
Figure 4

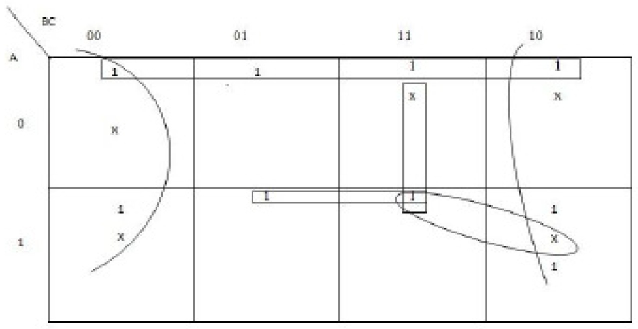
Figure 5

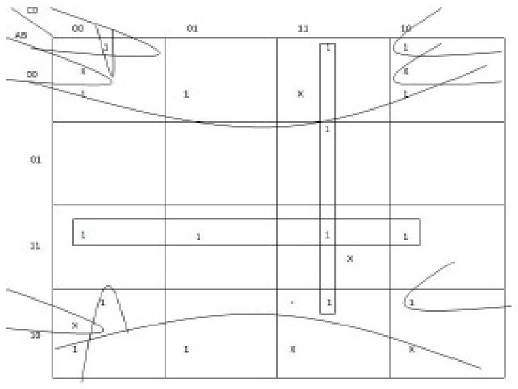
Figure 6

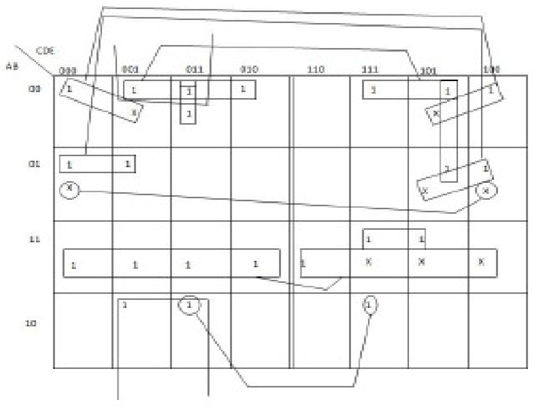
Figure 7

Figure 4, Figure 5, Figure 6 and Figure 7 show the minimization process of different multiple output circuits using the Extended K-map.

4. Complexity Analysis

Generally in case of basic K-Map [3] for n-variable function it needs 2^n spaces for a single function to be solved. So for k numbers of functions total number of spaces are $k*2^n$. So space complexity function will be

$$F_k = k * 2^n \text{ which is } O(2^n).$$

But in case of our Extended K-Map algorithm for n-variable function it needs 2^n spaces for all k function with an additional stack which has again k spaces for k number of functions. So space complexity function will be

$$F_k = k + 2^n \text{ which is } O(2^n).$$

So our algorithm is more space efficient as compared with previous one. Table 2 and Figure 8 show the complexity of our Extended K-map method compared with the complexity of basic K-map method.

Table 2: Complexity comparison

k	BK- Basic K-Map EK – Extended K-Map	n			
		2	3	4	5
1	BK	4	8	16	32
	EK	5	9	17	33
2	BK	8	16	32	64
	EK	6	10	18	34
3	BK	12	24	48	96
	EK	7	11	19	35
4	BK	16	32	64	128
	EK	8	12	20	36
5	BK	20	40	80	160
	EK	9	13	21	37
6	BK	24	48	96	192
	EK	10	14	22	38
7	BK	28	56	112	214
	EK	11	15	23	39
8	BK	32	64	128	256
	EK	12	16	24	40
9	BK	36	72	144	288
	EK	13	17	25	41
10	BK	40	80	160	320
	EK	14	18	26	42

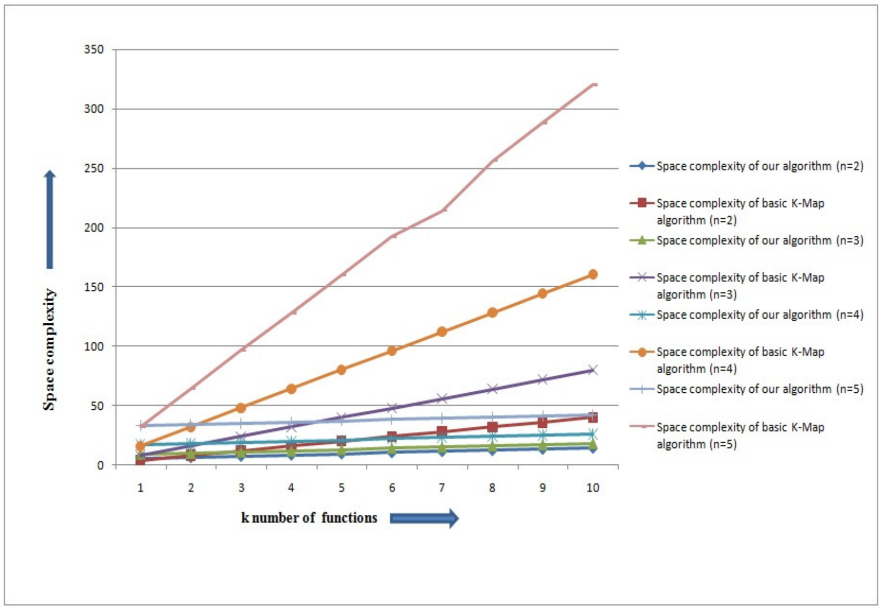

Figure 8: Graphical representation of complexity comparison

5. CONCLUSIONS

In this paper, we have proposed a new K-map which is more space efficient with increasing number of functions. Any number of functions can be minimized using this new technique. Future work may be done to extend this technique for minimizing large circuits.

REFERENCES

[1] Boole G. (1954): An Investigation of the Laws of Thought. — New York: Dover Publications.
[2] Shannon C.E. (1938): A symbolic analysis of relay and switching circuits. —Trans. AIEE, Vol. 57, No. 6, pp. 713–723.
[3] Karnaugh M. (1953): The map method for synthesis of combinatorial logic circuits. — Trans. AIEE Comm. Electron.,Vol. 72, No. 4, pp. 593–598.
[4] McCluskey E. J. (1956), "Minimization of Boolean functions", Bell System Tech. J., Vol. 35, No. 5, pp. 1417–1444.
[5] Quine W. V. (1952), "The problem of simplifying truth tables", Amer. Math. Month., Vol. 59, No. 8, pp. 521–531.
[6] Petrick S. K. (1959), "On the minimization of Boolean functions", Proc. Int. Conf. Information Processing, Paris: Unesco, pp. 422–423.
[7] McCluskey E. J. (1965), "Introduction to the Theory of Switching Circuits", New York, McGraw-Hill.
[8] Biswas N. N. (1971), "Minimization of Boolean Functions", IEEE Trans. on Computers, Vol. C-20, pp. 925-929.
[9] Hong S. J., Cain R. G., Ostapko D. L. (1974), "MINI: A Heuristic Approach for Logic Minimization", IBM Journal of Research and Development, Vol. 18, pp. 443-458.
[10] Rhyne V. T., Noe P. S., McKinney M. H., and Pooch U.W. (1977) "A New Technique for the Fast Minimization of Switching Functions", IEEE Trans. on Computers, Vol. C-26, pp. 757-764.