ADAPTIVE MAP FOR SIMPLIFYING BOOLEAN EXPRESSIONS

Dr. Mohammed H. AL-Jammas

Department of Computer and Information Engineering,
College of Electronics Engineering, University of Mosul, Mosul - Iraq

ABSTRACT

The complexity of implementing the Boolean functions by digital logic gates is directly related to the complexity of the Boolean algebraic expression. Although the truth table is used to represent a function, when it is expressed algebraically it appeared in many different, but equivalent, forms. Boolean expressions may be simplified by Boolean algebra. However, this procedure of minimization is awkward because it lacks specific rules to predict each succeeding step in the manipulative process. Other methods like Map methods (Karnaugh map (K-map), and map Entered Variables) are useful to implement the Boolean expression with minimal prime implicants. Or the Boolean function can be represents and design by used type N's Multiplexers by partitioned variable(s) from the function. An adaptive map is a combined method of Boolean algebra and K-map to reduce and minimize Boolean functions involving more than three Boolean variables.

KEYWORDS

Adaptive map, Boolean function, Entered variable, K-map, Partitioned variable

1. INTRODUCTION

Because binary logic is used in all digital computers and digital devices, the cost of the circuits is an important factor addressed by designers [1]. Finding simpler and cheaper, but equivalent, realizations of a circuit can reap huge payoffs in reducing the overall cost of the design. Mathematical method, map methods, tabular methods, and, other methods are used to simplify and implements Boolean functions.

2. MATHEMATICAL METHOD

In mathematical method, the Boolean algebra, like any other mathematical system, is defined with set of elements, set of operators, and number of unproved axioms or postulates. All Boolean expressions, regardless of their form, can be writes in two standard forms: the (**SOP**) **S**um-**O**f-**P**roducts form or the (**POS**) **P**roduct-**O**f-**S**ums form. Standardization makes evaluates, simplified, and implements the Boolean expressions more systematic and easier [2].

Boolean algebra is formal to express a digital logic equations, and represents a logical design in an alpha-numeric way. The Boolean algebra format, and many of logic manipulates rules and techniques were formalized around 1850 by George Boole, an Irish mathematician. It was used a systematic approach to solving problems in logic and reasoning. With the advent of modern electronics, and digital systems in particular, Boolean algebra found a natural home. In addition to being used as a tool for deductive reasoning, it is now an almost indispensable tool to design the digital logic circuits and machines.

DOI : 10.5121/ijcses.2013.4601

3. MAP METHOD

However, the map method presented here provides a simple, straightforward procedure for minimize Boolean functions. This method regarded as a pictorial form of a truth table. The map method is also known as *Karnaugh map* or *K-map*.

The *Karnaugh map*, like Boolean algebra, is a simplification tool applicable to digital logic. The *Karnaugh map* is simplified logic expression faster and more easily in most cases. Boolean simplification is faster than the *Karnaugh map* for a task involving two or fewer Boolean variables. It is still quite usable at three variables, but a bit slower. At four input variables, Boolean algebra becomes tedious. *Karnaugh maps* are both, faster and easier, work well for up to six input variables. For more than six to eight variables, simplification should be by **CAD** (**C**omputer **A**utomated **D**esign) [3], or use other methods.

4. TABULAR METHOD

In tabular methods for function minimization have been devised that can be implemented by a computer and can therefore be used to minimize functions having a large number of input variables. One such method has become known as the **Q**uine-**M**cClusky (**Q-M**) algorithm. Typical of these methods, the **Q-M** algorithms first finds the prime implicants and then generates the minimum cover [4].

5. ENTERED VARIABLE K-MAP METHOD

The K-maps are only useful up to and including functions with six variables, but in the case of a function having a larger number of variables and providing the function does not contain too many terms it can be useful to plot it on a reduced dimension map. Such a map is one in which the individual cells can now contain variables, so that a map for m variables can be used to represent functions having (m+1) or even (m+2) variables [5]. The entered variable mapping, which is a logical and very useful extension of conventional (1's and 0's) mapping methods developed previously. Entered variable K-maps are the most common form of graphical representation, two types of variables shown in K-map, the map variable which represent the index of K-map, and the entered variables (1's, 0's, don't care, and variable).

6. SHANNON'S EXPANSION THEOREM

Designing with Multiplexers resolves around applying a theorem called Shannon's Expansion Theorem. The theorem can be stated as follows:

$$\mathbf{F} = \sum_{i=0}^{2^n-1} \mathbf{I}_i . \boldsymbol{m_i} \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \qquad (\mathbf{1})$$

Where m_i is a minterm consisting of n select signals that are applied to the select inputs and I_i is the input data (used for applying 2^n input data signals) [6], the input signals (1's, 0's, don't care, and variable).

Boolean functions with a large number of inputs can be accommodate by constructing Multiplexer trees (cascade Multiplexers) to implement functions, but it is not easily reduced or

implement functions with a larger than normal number of inputs. It is not necessary to write every binary entry for every signal in the truth table, since we can reduce the output column by simply listing the binary values of the partitioned-off signal (or signals). The type of Multiplexers design simply agrees with the number of input signals partitioned off in the truth table of the function. For example, we called type 2 Multiplexer when implement the function when partitioned off two signals.

7. ADAPTIVE MAP

To implement the Boolean function with large number of input signal, it is suitable to use entered variable method, or use type *n* Multiplexer. The complexity of these methods is how to find the input data of Multiplexer, or find the entered variables in K-map.

In this paper, make a combination between Boolean algebra, K-map, Entered variables, and Shanno's theorem methods that suggest how to find the variables to implements function by different types of multiplexers, or by Map Entered variable method.

An adaptive map is build with a variable location of sub-cubes in K-map. The sub-cubes locations are adaptive with the partitioned off variables or with entered variables.

7.1. The Partitioning off One Variable

Assume a function with four variables,

$$\boldsymbol{f(A,B,C,D) = \sum_{i=0}^{2^n-1} I_i \,.\, m_i} \qquad \ldots\ldots\ldots\ldots (2)$$

n=4 number of variables.

The variable ***D*** is a less significant bit ($2^0 = 1$), while variable ***C*** is ($2^1 = 2$), and variable ***A*** is ($2^3 = 8$). These values used to implements function with new location of sub-cubes when any of these variables is partitioned off.

The flowing Adaptive map represents the case when partitioned off one variable.

Partitioned off variable ***D*** (2^0=1)
The difference of location between each two
sub-cubes in the same column = $2^0 = 1$

	ABC							
D	000	001	010	011	100	101	110	111
0	I_0 m_0	I_2 m_2	I_4 m_4	I_6 m_6	I_8 m_8	I_{10} m_{10}	I_{12} m_{12}	I_{14} m_{14}
1	I_1 m_1	I_3 m_3	I_5 m_5	I_7 m_7	I_9 m_9	I_{11} m_{11}	I_{13} m_{13}	I_{15} m_{15}

Partitioned off variable ***B*** (2^2=4)
The difference of location between each two
sub-cubes in the same column = $2^2 = 4$

	ACD							
B	000	001	010	011	100	101	110	111
0	I_0 m_0	I_1 m_2	I_2 m_4	I_3 m_6	I_8 m_8	I_9 m_9	I_{10} m_{10}	I_{11} m_{11}
1	I_4 m_4	I_5 m_5	I_6 m_6	I_7 m_7	I_{12} m_{12}	I_{13} m_{13}	I_{14} m_{14}	I_{15} m_{15}

In the adaptive map shown in (a), the equivalent prime implicants of each column (from column 0 to column 7) can be found using direct Boolean algebra.

$P_0 = I_0.m_0 + I_1.m_1$, $P_1 = I_2.m_2 + I_3.m_3$, , , $P_7 = I_{14}.m_{14} + I_{15}.m_{15}$.

The P_o, P_1, , , P_7 are inputs data to the (8X1) Multiplexer with (***A,B,C***) are the select signals, or an entered variable in map entered variable method that shown below:

The same way in map show in (b), the equivalent prime implicants can find by:

$P_0 = I_0.m_0 + I_4.m_4$, $P_1 = I_1.m_1 + I_5.m_5$, , , $P_7 = I_{11}.m_{11} + I_{15}.m_{15}$

The P_0 , P_1 , , P_7 are the inputs data to the (8X1) Multiplexer with (***A,C,D***) are select Signals

	BC			
A	00	01	11	10
0	P_0	P_1	P_3	P_2
1	P_4	P_5	P_7	P_6

	CD			
A	00	01	11	10
0	P_0	P_1	P_3	P_2
1	P_4	P_5	P_7	P_6

7.2. Partitioned off Two Variables

Here we assume a function with five variables,

$$f(A,B,C,D,E) = \sum_{i=0}^{2^5-1} I_i \,.\, m_i \qquad \ldots\ldots\ldots\ldots(3)$$

The variable ***E*** is less significant bit (2^0 = 1), while variable ***B*** is (2^3 = 8), and variable ***A*** is (2^4 = 16). These values used to implement the function with new location of sub-cubes when two of these variables are partitioned off.

The flowing Adaptive map is represent the case of partitioning two variables

Partitioned off variables ***E*** (2^0=1), and ***D*** (2^1=2)
The difference between sub-cubes in same column is 2^0 and 2^1

ABC

DE	000	001	010	011	100	101	110	111
00	I_0 m_0	I_4 m_4	I_8 m_8	I_{12} m_{12}	I_{16} m_{16}	I_{20} m_{20}	I_{24} m_{24}	I_{28} m_{28}
01	I_1 m_1	I_5 m_5	I_9 m_9	I_{13} m_{13}	I_{17} m_{17}	I_{21} m_{21}	I_{25} m_{25}	I_{29} m_{29}
10	I_2 m_2	I_6 m_6	I_{10} m_{10}	I_{14} m_{14}	I_{18} m_{18}	I_{22} m_{22}	I_{26} m_{26}	I_{30} m_{30}
11	I_3 m_3	I_7 m_7	I_{11} m_{11}	I_{15} m_{15}	I_{19} m_{19}	I_{23} m_{23}	I_{27} m_{27}	I_{31} m_{31}

Partitioned off variables ***B*** (2^3=8), and ***C*** (2^2=4)
The difference between sub-cubes in same column is 2^2 and 2^3

ADE

BC	000	001	010	011	100	101	110	111
00	I_0 m_0	I_1 m_2	I_2 m_4	I_3 m_6	I_{16} m_{16}	I_{17} m_{17}	I_{18} m_{18}	I_{19} m_{19}
01	I_4 m_4	I_5 m_5	I_6 m_6	I_7 m_7	I_{20} m_{20}	I_{21} m_{21}	I_{22} m_{22}	I_{23} m_{23}
10	I_8 m_8	I_9 m_9	I_{10} m_{10}	I_{11} m_{11}	I_{24} m_{24}	I_{25} m_{25}	I_{26} m_{26}	I_{27} m_{27}
11	I_{12} m_{12}	I_{13} m_{13}	I_{14} m_{14}	I_{15} m_{15}	I_{28} m_{28}	I_{29} m_{29}	I_{30} m_{30}	I_{31} m_{31}

In the adaptive map shown in (a), the equivalent prime implicants of each column (from column 0 to column 7) can be found using direct Boolean algebra.

$P_0 = I_0.m_0 + I_1.m_1 + I_2.m_2 + I_3.m_3, , , P_7 = I_{28}.m_{28} + I_{29}.m_{29} + I_{30}.m_{30} + I_{31}.m_{31}$

The P_0 , P_1 , , P_7 are the inputs data to the (8X1) Multiplexer with (***A,B,C***) are select signals, or use an entered variable in the map shown below

The same way in map (b), the equivalent prime implicants can find by:

$P_0 = I_0.m_0 + I_4.m_4 + I_8.m_8 + I_{12}.m_{12}, , , P_7 = I_{19}.m_{19} + I_{23}.m_{23} + I_{27}.m_{27} + I_{31}.m_{31}$

The P_0 , P_1 , , P_7 are the inputs data to the (8X1) Multiplexer with (***A,D,E***) are select signals

BC

A	00	01	11	10
0	P_0	P_1	P_3	P_2
1	P_4	P_5	P_7	P_6

DE

A	00	01	11	10
0	P_0	P_1	P_3	P_2
1	P_4	P_5	P_7	P_6

7.3. Partitioned off Three Variables

A function with five variables can be represent as,

$$f(A,B,C,D,E) = \sum_{i=0}^{2^5-1} I_i \,.\, m_i \qquad \ldots\ldots\ldots\ldots(4)$$

The flowing Adaptive map represents the case of partitioning off three variables.

Partitioned off variables ***B*** (2^3=8), ***C*** (2^2=4), and ***D*** (2^1=2)
The difference of location between sub-cubes in the same column = 2^1 , 2^2 , and 2^3

BCD \ ***AE***	00	01	11	10
000	I_0 m_0	I_1 m_1	I_{16} m_{16}	I_{17} m_{17}
001	I_2 m_2	I_3 m_3	I_{18} m_{18}	I_{19} m_{19}
010	I_4 m_4	I_5 m_5	I_{20} m_{20}	I_{21} m_{21}
011	I_6 m_6	I_7 m_7	I_{22} m_{22}	I_{23} m_{23}
100	I_8 m_8	I_9 m_9	I_{24} m_{24}	I_{25} m_{25}
101	I_{10} m_{10}	I_{11} m_{11}	I_{26} m_{26}	I_{27} M_{27}
110	I_{12} m_{12}	I_{13} m_{13}	I_{28} m_{28}	I_{29} m_{29}
111	I_{14} m_{14}	I_{15} m_{15}	I_{30} m_{30}	I_{31} m_{31}

Partitioned off variables ***A*** (2^4=16), ***C*** (2^2=4), and ***E*** (2^0=1)
The difference of location between sub-cubes in the same column = 2^0 ,2^2 ,and 2^4

ACE \ ***BD***	00	01	11	10
000	I_0 m_0	I_2 m_2	I_8 m_8	I_{10} m_{10}
001	I_1 m_1	I_3 m_3	I_9 m_9	I_{11} m_{11}
010	I_4 m_4	I_6 m_6	I_{12} m_{12}	I_{14} m_{14}
011	I_5 m_5	I_7 m_7	I_{13} m_{13}	I_{15} m_{15}
100	I_{16} m_{16}	I_{18} m_{18}	I_{24} m_{24}	I_{26} m_{26}
101	I_{17} m_{17}	I_{19} m_{19}	I_{25} m_{25}	I_{27} m_{27}
110	I_{20} m_{20}	I_{22} m_{22}	I_{28} m_{28}	I_{30} m_{30}
111	I_{21} m_{21}	I_{23} m_{23}	I_{29} m_{29}	I_{31} m_{31}

In the adaptive map show in (a), the equivalent prime implicants of each column (from column 0 to column 4) can be fined use direct Boolean algebra.

$P_0 = I_0.m_0 + I_2.m_2 + I_4.m_4 + I_6.m_6 + I_8.m_8 + I_{10}.m_{10} + I_{12}.m_{12} + I_{14}.m_{14}$

The P_0 , P_1 , P_3 , P_4 are the inputs to the (4X1) Multiplexer with (***A,E***) are select signals, or use an entered variable in the map.

The same way in map (b), the equivalent prime implicants can find by:

$P_0 = I_0.m_0 + I_1.m_1 + I_4.m_4 + I_5.m_5 + I_{16}.m_{16} + I_{17}.m_{17} + I_{20}.m_{20} + I_{21}.m_{21}$

The P_0 , P_1 , P_3 , P_4 are the inputs data to the (4X1) Multiplexer with (***B,D***) are select signals

8. CONCLUSIONS

The most simplification methods are useful for less than 6 variables. A large number of input signals will be complex to simplify the function and implements with minimal logic gates, or use a suitable multiplexer. The adaptive map is a useful method used to reduce the K-map size and

simplify the function use entered variable method, or to use type *n* multiplexers with number of select signals less than the number of the input signals.

This method gives more imagination to designer to find different way to simplify any complex Boolean function with more than three input signals.

REFERENCES

[1] M. Morris Mano, and Michael D. Ciletti,(2006) *Digital Design*, Fourth Edition, Prentice Hall.
[2] Thomas L. Floyd, (2006) *Digital Fundamentals*, Ninth Edition, Prentice Hall.
[3] Tony R. Kuphaldt, (2007) *Lessons In Electric Circuits*, Volume IV – Digital.
[4] Richard F. Tinder, (2000) *Engineering Digital Design*, Second Edition, Academic Press (AP).
[5] Brain Holdsworth, and Clive Woods, (2002) *Digital Logic Design*, Fourth Edition, Copyright Material.
[6] Richard S. Sandige, (1990) *Modern Digital Design.* Mcgraw-Hill
[7] Richard S. Sandige, and Michael L. Sandige, (2012) *Fundamentals of Digital and Computer Design with VHDL*, McGraw Hill.

Author

Mohammed H. AL-Jammas (Jun'02) born in 1966 in Mosul-Iraq. He awarded BSc in Electronic and Communication Engineering from the University of Mosul, Mosul-Iraq in 1988. Next, he awarded the MSc in Communication from the University of Mosul, Mosul-Iraq in 1994, and PhD in Computer Engineering from the University of Technology, Baghdad-Iraq in 2007. From 2002-2006, Dr. Mohammed worked with the University of Technology in Baghdad. From 2007, he acts as an Assistance dean of the College of Electronics Engineering at the University of Mosul.

Through his academic life he published over 5 papers in field of computer engineering, and information security.