ISSN 2456-8066
International Journal of Advanced Engineering and Management
vol. 2, no. 6. 2, pp. 141-145, 2017
DOI: https://doi.org/10.24999/IJOAEM/02060034

Software Implementation of Break-Up Algorithm for Logic Minimization

Koustuvmoni Bharadwaj and Sahadev Roy

Abstract

In this paper we propose a robust technique to minimize multiple input digital circuits. This technique is simple and efficient to determine the minimum number of gates required to realize a multiple input digital circuit. Rather than using sizeable truth table for multiple input combinational circuits having at least one operation per minterms, proposed technique produced a minimal solution by breaking the minterms and arranging them in adjacent groups. This technique also overcomes the problems related to simplification using Karnaugh Map for more than four inputs system. Also we have proposed an algorithm to implement this technique in Java programming language and out provide results.

Keywords

Breakup Algorithm;
Java;
Logic Minimization;
Minterms;
Prime Implicants;
Prime Implicant Table.

I. INTRODUCTION

Digital function minimization is a classical problem, for which various techniques have been proposed by researchers over the time based on the work by Shannon in 1938 [1]. Minimization using Chart Method was proposed by Veitch in 1952 [2] which was further carried on by Karnaugh in 1953 [3], popularly known as Karnaugh Map or K-map which can be used to minimize upto 6 variables efficiently. If the number of variables increases complexicity increases [4]. Quine-McCluskey Method is also an important method used for minimization in software implementation, proposed by Quinne in 1952 [5] and improved by McClauskey [6-7] by using don't care [8] prime implicants [9].

Implementation of Logic Switching circuit has two types: Two-level synthesis and multilevel synthesis [10]. Implementation of multilevel synthesis faces problems like propagation delay, errors and complexity in designing although it requires fewer gates and fewer connections compared to two level synthesis [11-16]. Two level synthesis is further classified into Exact techniques (eg: K-Map method, Quinne Mcclauskey method) and Heuristic techniques (Variable Entered Map [13], ESPRESSO-II [14], ESPRESSO-MV [15], etc.). Exact techniques give minimum representation but computational complexity is more compared to Heuristic approaches. Heuristic approaches are usually preferred for software implementation. T.S. Rathore proposed an effective method in the year 2014, which reduced the truth table size by half by reducing the truth table from N variables to N-1 to N-2 variables and so on, but the number of steps involved increases along with the number of effective truth tables.

Therefore we propose a concept based to the "Breakup Technique" to reduce the truth table by breaking it down into two and implementing this technique in software, thus providing an optimal solution with minimum gate counts.

II. INTRODUCTION TO THE ALGORITHM

In the proposed algorithm, the inputs are given in the form of a .txt file. These inputs are stored in the system which is defined location saved by the user. These are store in a two dimensional array and arrange the in an ascending order. The inputs can have only three characters '0','1'and '-'. After arranging we check for difference in 1 bits among two input minterms, and if there exist such pairs we replace them with '-' character in both the minterms and we store them in another array and repeat the same three times to get the minimized terms. In the end we plot the prime implicant table using the input minterms and the minimized prime implicants.

The algorithm has been described below elaborately.

1. Declare three Two Dimensional Character Array, 'charmap[][]', charmap_copy[][] and charmap1[][] to store the input binary numbers.
2. Define the path of the input file.
3. Using the 'buffer reader' function read each element line by line and store them in the Two Dimensional Character Array, charmap[][].

4. Check if the number of columns is same for each row, if not break there; else continue till the last term.
5. Calculate the number of Input Product Terms, i.e. the number of rows and the number of Input Variable, i.e. the number of Columns.
6. Copy the values of charmap[][] into charmap_copy[][].
7. Call the minimize(charmap[][]) function.

I. First the elements of the charmap[][] array are sorted in ascending order and the the duplicate elements are removed using the sort(charmap) function where each row of the array are converted into a string and then compared and then again stored in the charmap array as char.

II. Three variables r1,r2 and c are initialised which depict row1, row2 and column of the charmap array.

III. Using for loop, value of a particular column of both the rows are compared to find a difference in 1 bit.

IV. While comparing the elements of the two rows if there is a difference in bits, a flag, 'flag0' is incremented.

V. After the elements of the whole row are checked, we check the value of 'flag0'. If the value of 'flag0' is 1 we replace that particular bit by '-' and store the whole row in charmap1[][]. Else we store the value of the row in charmap1[][]. For Eg: if minterms '0010' and '0011' are compared, we see both the values differ by a bit '1' hence we store the value '001-' in charmap1[][].

Table1. Comparison between the two minterms row wise and their stored value

	C1	C2	C3	C4
Row1	0	0	1	0
Row2	0	0	1	1
Stored Value	0	0	1	-

VI. The charmap1[][] is sorted and duplicate elements are removed and the values are copied into charmap[][].

8. Display the charmap[][] array after minimising for the first time.
9. Repeat steps 7 and 8 for two more times. We have seen with trial and error method that repeating these steps more than 3 times does give the same result, so we repeat only 3 times.
10. Finally the minimised terms are displayed.
11. We also plot the Prime Implicant Table by declaring an array Table[][], where rows are the input minterms stored in array charmap_copy stored in ascending order and the columns are the minimised terms stored in ascending order.
12. For a particular element in the table array, table[i][j], we compare the $(i-1)^{th}$ element of charmap_copy[][] and $(j-1)^{th}$ element of charmap[][]. If both the values differ only by the element '-' then it is a prime implicant and we marked it by 'X' else we left it empty.

III. Result Analysis and Verification

Example 1:

F (A, B, C, D, E, F, G, H)
= ∑ m(24,25,28,29,105,152,233).

Feed in the input through a text file where the minterms are stored in their respective binary form.

Number of Input Product Terms: 7
Number of Input Variable: 8

The given input is:
00011000
00011001
00011100
00011101
01101001
10011000
11101001

Minterms after minimizing 1st time:
-0011000
-1101001
00011-00
00011-01
0001100-
0001110-

Minterms after minimizing 2nd time:
-0011000
-1101001
00011-0-
Minimized Product Terms:
-0011000
-1101001
00011-0-

Now on finding the expression from the minimized product terms, we get,

F (A, B, C, D, E, F, G, H) = $\overline{A}\,\overline{B}\,\overline{C}\,D\,E\,\overline{G}$ + $\overline{B}\,\overline{C}\,D\,E\,F\,\overline{G}\,\overline{H}$ + $B\,C\,\overline{D}\,E\,\overline{F}\,\overline{G}\,H$

Example 2:

F (A, B, C, D, E) = ∑ m(2, 3, 6, 7, 11, 12, 13,18,19, 22, 23, 24, 28, 29).

Number of Input Product Terms: 14

Number of Input Variable: 5

The given input is:

00010
00011
00110
00111
01011
01100
01101
10010
10011
10110
10111
11000
11100
11101

Minterms after minimizing 1st time:

-0010
-0011
-0110
-0111
-1100
-1101
0-011
00-10
00-11
0001-
0011-
0110-
10-10
10-11
1001-
1011-
11-00
1110-

Minterms after minimizing 2nd time:

-0-10
-0-11
-001-
-011-
-110-
0-011
00-1-
10-1-
11-00

Minimized Product Terms:

-0-1-
-110-
0-011
11-00

No. of Minimized Product Terms: 4

Now on finding the expression from the minimized product terms, we get,

F (A, B, C, D, E,) = $\overline{B}D$ + $AB\overline{D}\,\overline{E}$ + $\overline{A}\,\overline{C}DE$ + + $BC\overline{D}$

IV. CONCLUSION

This is an organized method used in the simplification process of logic functions which requires very less time and avoids cumbersome manipulations and calculations compared to Karnaugh map method. From this paper we have seen that minimized functions can be obtained by combining minterms by sorting and forming sets. We have seen that this algorithm is very much effective in reducing complexity of multiple input/output digital circuits which is beyond the capacity of a Karnaugh map. This algorithm is very much helpful in designing the combinational circuits, which are the building blocks of sequential circuits; also it is used in designing PLA.

REFERENCES

[1] Roy, S., & Bhunia, C. T. (2015). On synthesis of combinational logic circuits. *International J of Computer Applications*, *127*(1), 21-6. DOI: 10.5120/ijca2015906311.

[2] Veitch, E. W. (1952, May). A chart method for simplifying truth functions. In *Proceedings of the 1952 ACM national meeting (Pittsburgh)* (pp. 127-133). ACM..

[3] Karnaugh, M. (1953). The map method for synthesis of combinational logic circuits. *Transactions of the American Institute of Electrical Engineers, Part I: Communication and Electronics*, *72*(5), 593-599.

[4] Roy, S., & Bhunia, C. T. (2014, August). Constraints analysis for minimization of multiple inputs logic programming. In *Proceedings of International Conference on Signal and Speech Processing ICSSP-14, India* (pp. 61-64).

[5] Quine, W. V. (1952). The problem of simplifying truth functions. *The American Mathematical Monthly*, *59*(8), 521-531.

[6] McCluskey, E. J. (1956). Minimization of Boolean functions. *Bell Labs Technical Journal*, *35*(6), 1417-1444.

[7] McCluskey, E. J. (1956). Detection of group invariance or total symmetry of a Boolean function. *Bell Labs Technical Journal*, *35*(6), 1445-1453.

[8] McCluskey, E. J. (1961, October). Minimal sums for Boolean functions having many unspecified fundamental products. In *Switching Circuit Theory and Logical Design, 1961. SWCT 1961. Proceedings of the Second Annual Symposium on* (pp. 10-17). IEEE.

[9] Green, D. H., & Khuwaja, G. A. (1993). Tabular simplification method for switching functions expressed in Reed-Muller algebraic form. *International Journal of Electronics Theoretical and Experimental*, *75*(2), 297-314.

[10] Roy, S. (2016). Breakup Algorithm for Switching Circuit Simplifications. *International Journal of Advanced Engineering and Management*, *1*(1), 1-11.

[11] Bose, S., & Saha, A. R. (1997). Some Algorithmic Improvements in Multi-Level Logic Minimisation. *IETE Journal of Research*, *43*(5), 371-381.

[12] Das, S. R., & Jone, W. B. (1992). On random testing for combinational circuits with a high measure of confidence. *IEEE transactions on systems, man, and cybernetics*, *22*(4), 748-754.

[13] Mohyuddin, N., Pakbaznia, E., & Pedram, M. (2011). Probabilistic error propagation in a logic circuit using the boolean difference calculus. In *Advanced Techniques in Logic Synthesis, Optimizations and Applications*(pp. 359-381). Springer New York.

[14] Dartu, F., Menezes, N., Qian, J., & Pillage, L. T. (1994, June). A gate-delay model for high-speed CMOS circuits. In *Proceedings of the 31st annual Design Automation Conference* (pp. 576-580). ACM.

[15] Roy, S. (2017). An Efficient Technique For Switching Functions Simplification. *International Journal of Advanced Engineering and Management*, *2*(1), 21-28.

[16] Roy, S., & Bhunia, C. T. (2013). Minterms generations algorithm using weighted sum method. *International Journal on Current Science & Technology*, *1*(2), 34-38.

[17] Roy, S., & Bhunia, C. T. (2014, January). Minimization algorithm for multiple input to two input variables. In *Control, Instrumentation, Energy and Communication (CIEC), 2014 International Conference on* (pp. 555-557). IEEE.

[18] Roy, S., & Bhunia, C. T. (2015). Simplification of Switching Functions Using Hex-Minterms. *International Journal of Applied Engineering Research*, *10*(24), 45619-45624.

[19] Roy, S., Saha, R., & Bhunia, C. T. (2016). Multiple Inputs Combinational Logic Minimization by Minterms Set. In *Proceedings of the International Conference on Recent Cognizance in Wireless Communication & Image Processing* (pp. 133-140). Springer, New Delhi.

[20] Rudell, R., & Sangiovanni-Vincentelli, A. (1985, May). ESPRESSO-MV: Algorithms for multiple-valued logic minimization. In *Proc. IEEE Custom Integrated Circuits Conf* (pp. 230-234).

[21] Wayne Current, K. (1979). Quaternary logic techniques for simplified integrated digital signal processing circuitry. *International Journal of Electronics Theoretical and Experimental*, *46*(6), 611-620.

[22] Rao, P. S., & Jacob, J. (1998, January). A fast two-level logic minimizer. In *VLSI Design, 1998. Proceedings., 1998 Eleventh International Conference on*(pp. 528-533). IEEE.

[23] Rao, P. S., & Jacob, J. (1998, January). A fast two-level logic minimizer. In *VLSI Design, 1998. Proceedings., 1998 Eleventh International Conference on*(pp. 528-533). IEEE.

[24] Poikonen, J. H., Lehtonen, E., & Laiho, M. (2012). On synthesis of Boolean expressions for memristive devices using sequential implication logic. *IEEE Transactions on computer-aided design of Integrated Circuits and Systems*, *31*(7), 1129-1134.

[25] Taherifard, M., & Fathy, M. (2015). Improving logic function synthesis, through wire crossing reduction in quantum-dot cellular automata layout. *IET Circuits, Devices & Systems*, *9*(4), 265-274.

[26] Rathore, T. S. (2014). Minimal Realizations of Logic Functions Using Truth Table Method with Distributed Simplification. *IETE Journal of Education*, *55*(1), 26-32.

[27] Choudhuky, A., Basu, M. S., & Das, S. R. (1964). On a Method of Simplification of Multiple-output Switching Functions. *International Journal of Electronics*, *16*(2), 223-237.

[28] Teodorovic, P., Dautovic, S., & Malbasa, V. (2013). Recursive Boolean formula minimization

algorithms for implication logic. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *32*(11), 1829-1833.

Author Details

Koustuvmoni Bharadwaj
Dept. of ECE, NIT Arunachachal Pradesh, Yupia, India
email: koustuv_red@yahoo.in
Sahadev Roy
Dept. of ECE, NIT Arunachachal Pradesh, Yupia, India
email:sahadevroy@gmail.com