Optimisation of Mixed Polarity Reed-Muller Functions

Meng Yang
State Key Lab of ASIC and System, Fudan University, Shanghai, China
Email: mengyang@fudan.edu.cn

Jinmei Lai
State Key Lab of ASIC and System, Fudan University, Shanghai, China
Email: jmlai@fudan.edu.cn

***Abstract*—This paper presents a genetic algorithm (GA) search method in order to obtain better circuit implementation of the mixed polarity Reed-Muller functions. By combining global searching ability of genetic algorithm and local searching ability of simulated annealing, the proposed GA method could achieve fast convergence. It differs to traditional genetic algorithm, in which the proposed GA forms an intermediate population by using 2/3 population from previous generation and 2/3 population from current generation at the annealing stage. Annealing is then applied to the intermediate population to generate a new population. In the next generation selection, crossover and mutation operations are used for the newly generated population. The calculation of cost function of proposed algorithm is based on parallel tabular technique to overcome the disadvantage of the traditional tabular technique. The results of the tested benchmark indicated that this algorithm is highly effective for searching the best polarity and it could achieve 29% area reduction and 3.68X speedup.**

***Index Terms*—logic synthesis, Reed-Muller, mixed polarity, genetic algorithm, computer aided design**

I. Introduction

Mixed polarity Reed Muller (MPRM) [1] is one of the canonical AND/XOR forms, which have at least 3^n different numbers of expansions. XOR gates have large area and low speed attributes compared to AND/OR gates. It is widely known that FPGAs have made the delay and area of all types of gates equal. For instance, in Xilinx look-up table (LUT) type FPGA, the LUT can realise any function of up to six variables with the same area and delay. Hence, as field programmable gate arrays (FPGA) becomes available; circuits implemented in MPRM form can be more practical.

To widen the search space and achieve better synthesis results, Reed Muller (RM) expansions and sum-of product (SOP) expansions have been investigated. In some cases, the circuit can be better simplified in RM expansion using AND/XOR forms, whereas for some other circuits using AND/OR form will be the case. Extensive research has been carried out to find the optimal representation solutions. Numerous methods have been proposed for fixed polarity Reed Muller (FPRM) [2-4] and MPRM [5-10] in terms of area minimisation and/or power minimisation. In addition, methods proposed for minimisation of Dual RM were investigated [3, 11, 12, 13]. In [9] the authors presented a class of two-level RM expressions called reduced Kronecker expressions (RKROs) developed a method for an exact minimisation and applied genetic algorithm (GA) to minimise RKROs. Recently genetic algorithm methods [14, 15] were presented to obtain the optimal MPRM solution. Genetic algorithms have been found that they could produce good results within acceptable computation time. However, to find the optimal or even the best solution among a large number of polarities in an efficient way is challenging.

The main aim of the paper is to focus on the CPU time reduction without degradation of quality of solution. A GA method using parallel tabular technique in cooperation with multi-thread technique is proposed to finding efficient solutions among the large number of polarities in MPRM domains in a parallel manner. The remainder of the paper is organised as follows. Section II gives some definitions. MPRM conversion using multi-thread technique is given in Section III. The optimal Polarity via GA using multi-thread technique is given in Section IV. Section V discusses the comparison results in details with respect to other approaches. Conclusions are then given in Section VI.

II. Preliminaries

A. Theory

Given a truth table for a Boolean function, standard algebraic form of the function can be derived. The canonical SOP expansions are based on AND/OR, where m_i are the minterms; a_i = 0 or 1 and it indicates the absence or presence of minterms, respectively.

$$f = \sum_{i=0}^{2^n-1} a_i m_i \tag{1}$$

If all the variables are present in every term of (1), then the OR can be replaced by XOR giving XOR SOP, where $\oplus$ is the XOR operator.

doi:10.4304/jsw.8.11.2770-2774

$$f = \oplus \sum_{i=0}^{2^n-1} a_i m_i \quad (2)$$

B. Definitions

In the RM representations of the Boolean functions, each variable can be employed in one of three modes, as either true, complement or mixed mode.

Definition 1: If each variable in (2) appears in its true or complemented form but not both, the expressions are known as fixed polarity RM (FPRM) expressions, which have 2^n different polarities or expansions.

Definition 2: If each variable in (2) appears as true or complemented at the same time, the expressions are known as MPRM expressions, which have 3^n different polarities.

Example 1: $f(x_1, x_2, x_3) = x_1 \oplus x_1\bar{x}_2 \oplus \bar{x}_2 x_3$ for fixed polarity in which variable x_1 and x_3 in true forms and variable x_2 in complement form.

Example 2: $f(x_1, x_2, x_3) = x_1 \oplus x_1\bar{x}_3 \oplus \bar{x}_2 x_3$ for mixed polarity in which variable x_1 in true form, variable x_2 in complement form and variable x_3 in mixed form.

Definition 3: The different MPRM expansions are identified by a polarity number. To calculate the polarity of any function, each variable is replaced by 0, 1 or 2 depending on whether the variable is used in true, complemented or mixed form, respectively, as follows.

$$p_j = \begin{cases} 0 & \text{if } x_j \text{ appears in true form} \\ 1 & \text{if } x_j \text{ appears in complement form} \\ 2 & \text{if } x_j \text{ appears in mixed form} \end{cases} \quad (3)$$

The polarity is the decimal equivalent of the resulting ternary number.

Example 3: Given a three-variable function $f(x_1, x_2, x_3) = \bar{x}_2\bar{x}_3 \oplus x_2\bar{x}_3 \oplus x_1\bar{x}_2\bar{x}_3 \oplus x_1x_2 \oplus x_1x_2\bar{x}_3$, the polarity is 7 since variable x_1 appears in true form, variable x_2 appears in mixed form and variable x_3 appears in complement form.

III. Polarity Conversion via Parallel Tabular Technique

It is convenient to convert one of MPRM expansions to another by using tabular technique [3, 6]. However, the disadvantage of traditional tabular technique is that the generation of the on-set terms is in sequence. To overcome this problem, a new method is proposed to generate all the new terms of according to a required polarity. The generation of new terms can be implemented to each term independently without result conflict. The updating process is equivalent to the sequentially canceling pairs in the traditional tabular technique. The procedure is as follows:

Step 1: List all on-set terms in the index table and set the number of counts to 1.

Step 2: Select one of on-set terms in the index table.

Step 3: Generate all possible 2^j-1 new terms for the selected on-set term listed in the index table, considering 1 as don't care condition terms but leave the jth unchanged when $r_j = 2$ and $m_j = 2$

Step 4: If the newly generated term is already in the index table, increment the count by 1. Otherwise, add it to the index table and set the number of counts to 1.

Step 5: Repeat Step 2 –4 for all on-set terms.

Step 6: Alter the column heading to indicate the polarity of the variable and change the bias of the variable.

a) If $r_j = 0$ and $m_j = 1$ or 2, alter column heading according to the polarity of the variable x_j. The bias of the variable is unchanged.

b) If $r_j = 1$ or 2, and $m_j = 0$, alter column heading according to the polarity of the variable x_j. The bias of the variable is unchanged.

c) If $r_j = 1(2)$ and $m_j = 2(1)$, alter column heading according to the polarity of the variable x_j. The bias of the variable is reversed.

Step 7: Output the on-set terms of the required expansion, in which the number of count in the index table is odd number.

Since all the on-set terms are generated in parallel manner, different thread of Multi-core CPU can be used to each on-set term to enhance the speed without any conflicts. The pseudo code is shown in Algorithm 1, where THREADS is the number of threads of Multi-core CPU and join_all() function is used to wait each threads to complete its independent process.

```
Algorithm 1: MPRM conversion using multi-thread technique
Inputs: a given MPRM expansion u
Output: another MPRM expansion v
    Initialise the on-set table T according to expansion u
    for all THREADS t do {
        for all on-set coefficients/THREADS c in T do {
            Generate on-set coefficients N
            for all n in N do {
                if (n NOT exists T) {
                    add n to the on-set table T
                    Initialise the number count to 1
                }
                else
                    Increment the number count
            }
        }
    }
    join_all(); //wait
    for all on-set coefficients c in T do {
        if (the number count of c is odd number)
            c is the on-set coefficients of expansion v
    }
    Output result according to on-set coefficients of v
}
```

IV. Genetic Algorithm to Find the Optimal Polarity

Genetic algorithm encodes potential solutions to the problem as chromosomes and applies recombination and mutation operators to generate further solutions. After evaluation, the best solutions are used to replace the weaker ones. Each individual solution within the

population is randomly initialised and represented in mixed polarity by a fixed number of bits equal to the number of variables for the given function using ternary code as given in definition 3. The GA uses a tournament selection method where the main parameter of selection is the tournament size which can be changed by the operator. The single-point recombination operator is used. Thus, recombination is an action of choosing randomly a crossover point and combining two different parts from the two parents to form a new offspring. The uniform mutation alters a single gene of the individual randomly. Using mutation can encourage diversity within the population and minimise the chances of population stagnating at any local optima.

The fitness function computes the quality of new individuals. Initially, it computes the fitness of all the chromosomes. Then it computes the fitness for the new offspring of each evaluation. The fitness function is implemented by using parallel tabular technique explained in Section III to calculate the number of terms for the polarity determined in the selected individual. Tournament replacement controls the composition of the new generation for each evolution loop. Replacement is carried out if the new offsprings are not found in the population. After replacement it forms an intermediate population by selecting two thirds of previous population and two thirds of current population. Simulated annealing is applied to intermediate population according to $P(w_i)$. $P(w_i)$ is probability of individual w_i selection for annealing process. T_0 is initial temperature and set to 100, $j = 1,2,\cdots,n$ and $k = 1,2,\cdots,n$. n is intermediate population size.

$$P(w_i) = \frac{e^{f(i)/T_k}}{\sum_j e^{f(j)/T_k}} \tag{4}$$

$$T_k = \frac{1}{\ln\left(\frac{k}{T_0} + 1\right)} \tag{5}$$

A usual strategy is to stop evolution after a fixed number of evaluations. Pseudo code of the proposed GA is given in Algorithm 2, where THREADS is the number of threads of Multi-core CPU. join_all() is used to wait each threads to complete its independent process. thread_adapter() creates thread groups for create_thread() for parallelization. create_thread() creates different threads of multi-core CPU. evpop and annealing are evaluation for whole population and annealing processes. popcurrent and popinter are current population and intermediate population respectively.

```
Algorithm 2: Find optimal Polarity via GA using multi-thread technique
Inputs: Various parameters setting
Output: The best polarity expansion
    Read benchmark
    Initialise population
    for (int i = 0; i < THREADS; ++i) {
        ta = thread_adapter (evpop, popcurrent, i)
        create_thread(ta)
    }
    join_all() //wait
    while (number of generation--)
    {
        Select two parents randomly
        Single point crossover
        Mutation
        for (int i = 0; i < THREADS; ++i) {
            ta = thread_adapter (evpop, popcurrent, i)
            create_thread(ta)
        }
        join_all() //wait
        if (new offspring not exist)
            Replacement
        Form a new intermediate population with 4N/3 by
        selecting 2/3 from parents and 2/3 from offsprings
        for (int i = 0; i < THREADS; ++i) {
            ta = thread_adapter(annealing, popinter, i)
            create_thread(ta)
        }
        join_all() //wait
        Select N individuals to form a new generation
    }
    Output the best result
```

V. Results

The results are obtained on an Intel dual-core CPU of 2.4 GHz and 4 GB RAM under Windows 7. The difference is that GA methods used in [14, 15] are based on minterm conversion method and traditional tabular technique respectively whereas the GA in this paper utilises parallel tabular and multi-thread techniques. Each result is taken after running the GA algorithm ten times in order to reflect the quality of these results. The GA parameters are population size, tournament size and number of generations. From the test, it was found that a good population size was between 20 and 30. Tournament size parameter is used to control the selection pressure. From the test, it was found that the best result was produced when the tournament size was between 3 and 6.

Therefore the tournament size was chosen in this range for all benchmark examples. The number of generations indicates how many times the evaluation loop of the GA will be running. It is different for each example depending on the size and complexity of the given function.

The results are compared to methods proposed in [14] and [15], which are shown in Table 1. It can be seen from the Table I that the other GA methods and the proposed GA actually found the optimum solutions for all the benchmarks attempted. Our method achieves the same performance in terms of on-set number but outperforms methods proposed in [14] and [15] on average with improvement of 16.14% and 23.41% respectively in terms of CPU time in microseconds.

Table II shows the improvement of on-set numbers of each benchmark in SOP and MPRM expansions. Averagely the on-set number is reduced by 29%. Table II also shows the improvement of CPU time before and after parallelisation. The CPU time is speedup on average 3.68 times.

TABLE I.

COMPARISON RESULTS IN TERMS OF ON-SET NUMBER AND CPU TIME.

Name	I/O	On-set number			CPU (ms)		
		[15]	[14]	Ours	[15]	[14]	ours
rd53	5/3	20	20	20	7.9	7.5	4.8
con1	7/2	14	14	14	9.4	8.5	7.5
rd73	7/3	63	63	63	9.9	9.0	8.1
rd84	8/4	107	107	107	9.4	8.8	7.6
clip	9/5	182	182	182	9.7	9.2	8.3
misex1	8/7	13	13	13	8.7	8.2	7.2
sao2	10/4	76	76	76	10.1	9.6	8.3
ex1010	10/10	810	810	810	9.8	9.8	8.5
inc	7/9	34	34	34	9.4	8.5	7.5
5xp1	7/10	61	61	61	9.0	8.7	7.8
total	-	1380	1380	1380	93.3	87.7	75.6
average	-	138	138	138	9.33	8.78	7.56
Imp %		0	0	0	23.41	16.14	0

TABLE II.

IMPROVEMENTS OF ON-SET NUMBER AND CPU TIME.

Name	I/O	On-set number			CPU (s)		
		SOP	MPRM	Imp (%)	Before parallel	After parallel	Speedup (times)
x4	17/15	520	374	28	43.3	12.7	3.4
pcler8	27/17	53	40	25	52.2	13.7	3.8
c8	28/18	79	52	34	49.7	13.4	3.7
count	35/16	169	64	62	56.8	14.6	3.9
unreg	36/16	48	48	0	55.4	14.2	3.9
b9	41/21	106	119	-12	65.6	17.3	3.8
cht	47/36	81	81	0	66.8	19.6	3.4
example2	85/66	329	234	29	69.5	18.8	3.7
apex6	135/99	656	491	25	83.4	23.2	3.6
x3	135/99	656	536	18	81.9	23.4	3.5
i6	138/67	202	239	-18	82.5	22.3	3.7
j7	199/67	264	268	-2	85.2	22.4	3.8
total	-	3163	2546	356	729	215	44
average	-	264	212	29	66	18	3.68

VI. CONCULSIONS

In this paper, new algorithm with cooperation of parallel tabular technique is proposed to optimise MPRM expansions. The proposed parallel tabular technique overcomes the disadvantage of traditional tabular technique, resulting all the new terms are generated at one time instead of generating in sequence. The experimental results show that the proposed GA can find the optimum solution in a reasonable time in all the examples attempted, achieving efficiency in terms of CPU time without quality of solution loss. The proposed GA outperforms other GA methods and achieves 29% improvement on average. The CPU time is speedup on average 3.68 times after parallelisation.

ACKNOWLEDGMENT

This work was supported by a grant (No. 11MS011) from State Key Lab of ASIC and System, China and the National High Technology Research and Development (863) Thematic Program of China (2012AA012001).

REFERENCES

[1] D.H. Green, “Dual forms of Reed Muller expansions,” IEE Proc. Comput. Digit. Tech., vol. 141, no. 3, 1994, pp. 184-192.

[2] M.K. Habib, “A new approach to generate fixed polarity Reed Muller expansions for completely and incompletely specified functions,” Int. J. Electron., vol. 89, no. 11, pp. 845 – 876, 2002.

[3] M. Yang, H. Xu, L. Wang, J.R. Tong and A.E.A. Almaini, “Exact minimization of large fixed polarity dual form of Reed Muller functions,” Proc. of Eighth IEEE Int. Conf. on Solid-State and Integrated Circuit Technology, October 2006, pp. 1931 – 1933.

[4] S. Chaudhury and S. Chattopadhyay, “Fixed polarity Reed-Muller network synthesis and its application in AND-OR/XOR-based circuit realization with area-power trade-off,” IETE Journal of Research, vol. 54, no. 5, pp. 353-363, 2008.

[5] P.J. Wang, H. Li, “Low power mapping for AND/XOR circuits and its application in searching the best mixed-polarity,” Journal of Semiconductors, vol. 32, no. 2, pp. 025007, 2011.

[6] A.E.A Almaini and L. McKenzie, “Tabular techniques for generating Kronecker expansions”, IEE Proc. Comput. Digit. Tech., vol. 143, No. 4, pp. 205 – 212, 1996.

[7] B. Becker and R. Drechsler, “Exact minimisation of Kronecker expressions for symmetric function,” IEE Proc. Comput. Digit. Tech., vol. 143, no. 6, pp. 349 – 354, 1996.

[8] R. Drechsler, B. Becker and N. Gockel, “A genetic algorithm for RKRO minimisation”, Expert Syst. Appl., vol. 12, no. 1, pp. 127 – 139, 1997.

[9] M. Helliwell and M. Perkowski, “Fast algorithm to minimize multi output mixed-polarity generalized Reed – Muller forms,” Proc 25th IEEE/ACM Conf. on Design Automation, pp. 427 – 432, 1988.

[10] D. Debnath and T. Sasao, “GRMIN2 a heuristic simplification algorithm for generalised Reed Muller expressions,” IEE Proc. Part E, Comput. Digit. Tech., vol. 143, no. 6, pp. 376 – 384, 1996.

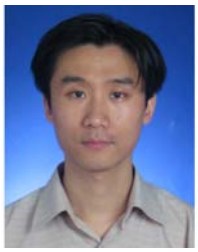
[11] L.Y. Wang, Y.S. Xia, X.X. Chen, “logic synthesis and optimization based on dual logic,” Journal of computer-aided design and computer graphics, vol. 24, no. 7, pp. 961-967, 2012.

[12] J. Cheng, X. Chen, K.M. Faraj, and AEA Almaini, “Expansion of logical function in the OR-coincidence system and the transform between it and maxterm expansion,” Proc. IET Computers and Digital Techniques, vol. 150, no. 6, pp. 397-402, 2003.

[13] M. Yang, J.R. Tong, J.M. Lai, “Optimisation of Fixed Polarity Canonical Or-Coincidence expansions”, Journal of Computers, vol. 8, no. 11, Nov. 2013.

[14] B.A. Al Jassani, N. Urquhart and A.E.A. Almaini, “Manipulation and optimisation techniques for Boolean logic”, IEE Proc. Comput. Digit. Tech., vol. 4, no. 3, 2010, pp. 184 – 192.

[15] B.A. Al Jassani, N. Urquhart and A.E.A. Almaini, "Optimization of MPRM functions using tabular techniques and genetic algorithms," The Mediterranean Journal of Electronics and Communications, vol. 4, no. 4, pp. 115-125, 2008.



Meng Yang received Bachelor of Engineering (Honor) degree in Electrical Engineering from Shanghai University, Shanghai, China, in 1999. He received Master of Science with distinction in Electronics and Communication Engineering and Ph.D. in Electronics from School of Engineering Edinburgh Napier University, Edinburgh, UK, in 2002 and 2006, respectively.

Currently he is a lecturer of Department of Microelectronics, School of Information Science and Technology, Fudan University, Shanghai, China. His research interests include algorithms in FPGA design automation, logic synthesis, and dynamic reconfigurable FPGA automation design. He has published more than 30 research papers.

Dr. Yang is a member of IET and Chairman of Young Member Section of IET Shanghai Branch.

Jinmei Lai received PhD degree in Shanghai Jiaotong University, Shanghai, China, in 1998. She was a Post-Doctor in Zhejiang University and Fudan University.

Currently she is a full professor of State Key Lab of ASIC and System, Fudan University, Shanghai, China. Her research interests include low power and reconfigurable architecture of FPGA and SOC, embedded IP core generation automation, logic synthesis and dynamic reconfigurable FPGA automation design, SOC testing automation. She has published more than 80 research papers and holds dozens of Chinese patents.