IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 5, 2019, accepted February 2, 2019, date of publication February 12, 2019, date of current version March 7, 2019.

*Digital Object Identifier 10.1109/ACCESS.2019.2899035*

# Fast Minimization of Fixed Polarity Reed-Muller Expressions

**ZHENXUE HE[1], LIMIN XIAO[2], ZHISHENG HUO[2], TAO WANG[3], AND XIANG WANG[3]**

[1]School of Information Science and Technology, Hebei Agricultural University, Baoding 071001, China
[2]School of Computer Science and Engineering, Beihang University, Beijing 100191, China
[3]School of Electronic and Information Engineering, Beihang University, Beijing 100191, China

Corresponding author: Zhenxue He (hezhenxue@buaa.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61772053, Grant 60973106, Grant 61232009, and Grant 81571142, in part by the China Postdoctoral Science Foundation under Grant 2018M641154, in part by the Scientific Science and Technology Research Projects of Universities in Hebei under Grant BJ2018012, in part by the Project of Hebei Natural Science Foundation under Grant G2018204093, in part by the Baoding Science and Technology Support Plan under Grant 15ZG050, and in part by the Hebei Science and Technology Project under Grant 15227535 and Grant 15227414.

**ABSTRACT** Logic minimization has recently attracted significant attention because in many applications it is important to have a compact representation as possible. In this paper, we propose a fast minimization algorithm (FMA) of fixed polarity Reed–Muller expressions (FPRMs). The main idea behind the FMA is to search the minimum FPRM with the fewest products by using the proposed binary differential evolution algorithm (BDE). The BDE can efficiently maintain population diversity and achieve a better tradeoff between the exploration and exploitation capabilities by use of proposed binary random mutation operator and improved selection operator. The experimental results on 24 MCNC benchmark circuits demonstrate that the FMA outperforms the genetic algorithm-based and simulated annealing genetic algorithm-based FPRMs minimization algorithms in terms of accuracy of solutions and solving efficiency. To the best of our knowledge, we are the first to use differential evolution algorithm to minimize FPRMs. The FMA can be extended to derive a minimum mixed polarity Reed–Muller expression.

**INDEX TERMS** Logic minimization, fixed polarity Reed-Muller expressions, differential evolution algorithm, mutation operator, selection operator.

## I. INTRODUCTION

Logic functions can be expressed either in AND/OR/NOT based Boolean logic or in AND/XOR based Reed-Muller (RM) logic. For some circuits, such as the arithmetic circuits, parity check circuits and communication circuits, RM representation are more efficient than their Boolean representation in power, area, speed, and testability [1]–[3]. Notably, functions that do not produce efficient solutions in the Boolean representation can often be realized efficiently in the RM representation [4]–[6]. Therefore, it is necessary to establish synthesis schemes for RM logic circuits, particularly as look-up-table based Field Programmable Gate Array (FPGA) technology has become increasingly available and the relative cost of EXOR gates has become a non-critical factor restricting the use of RM design approaches [7], [8]. Fixed Polarity RM Expressions (FPRMs) are one of the canonical RM expressions. FPRMs have attracted wide attention due to their remarkable superiority in designing easily testable circuits, detecting symmetric variable of switching functions, designing multi-level circuit and Boolean matching [9], [10]. Furthermore, for many practical functions, FPRMs require fewer products than sum-of-products expressions.

Logic minimization has become a fundamental research topic. For an $n$-variable logic functions, it has $2^n$ distinct FPRMs corresponding to $2^n$ different polarities. A minimum FPRM is one with the fewest products. Therefore, to search the best polarity corresponding to minimum FPRM from polarity optimization space is a typical NP-hard problem. Genetic Algorithm (GA) and its variants have been widely used in the polarity optimization of FPRMs due to the simplicity, robustness and inherent parallelism of GA. (e.g., [11]–[15]).

However, the existing polarity optimization approaches, which are mainly based on GA or its variants, have a low convergence speed or are easily trapped into the local optimal

The associate editor coordinating the review of this manuscript and approving it for publication was Walter Didimo.

solution due to the inherent defects of GA. Firstly, the selection operation makes the variance and entropy of population evolve toward the reduction, which would result in the decrease of population diversity. Secondly, there exists a contradiction between the dispersion of population and convergence, which leads to the slow convergence speed. Lastly, the selection mechanism based on fitness tends to the pure random selection when there is not much difference among the fitness of individuals, which would make the GA fall into the local optimal solution.

In this paper, we propose a Fast Minimization Algorithm (FMA) of FPRMs. Compared to existing minimization algorithms of FPRMs, our main contributions are as follows.

1) We propose a Binary Differential Evolution (BDE) algorithm to solve the discrete binary-encoded combination optimization problem, which can efficiently maintain population diversity and achieve a better tradeoff between the exploration and exploitation capabilities by use of proposed binary random mutation operator and improved selection operator.

2) We propose a FPRMs minimization algorithm, called FMA, which uses the BDE to search the minimum FPRM with the fewest products. To the best of our knowledge, we are the first to use differential evolution algorithm to minimize FPRMs. It can be extended to derive a minimum mixed polarity Reed-Muller expression.

3) We compare FMA with the GA based and Simulated Annealing GA (SAGA) based FPRMs minimization algorithms on MCNC benchmark circuits. Experimental results demonstrate the effectiveness and superiority of FMA.

The remainder of this paper is structured as follows. In Section II, we introduce a few preliminaries that are relevant to our study. The fast minimization algorithm of FPRMs is described in detail in Section III. Section IV presents the experimental results. Our conclusions are presented in Section V.

## II. PRELIMINARIES

### A. FPRM

Any $n$-variable Boolean function may be represented canonically in a sum-of-products form as

$$f(x_{n-1}, x_{n-2}, \ldots, x_0) = \sum_{i=0}^{2^n-1} a_i m_i \tag{1}$$

where $\Sigma$ is an OR operator and $m_i$ are the minterms. $a_i$ are the coefficient of minterms, and $a_i = 1$ or 0 represents the presence or absence of minterms, respectively.

By applying shannon theorem, the Boolean function can be expressed as follows:

$$\begin{aligned} f &= \overline{x_{n-1}} f(0, x_{n-2}, \ldots, x_0) + x_{n-1} f(1, x_{n-2}, \ldots, x_0) \\ &= (1 \oplus x_{n-1}) f(0, x_{n-2}, \ldots, @_0) \oplus x_{n-1} f(1, x_{n-2}, \ldots, x_0) \\ &= f(0, x_{n-2}, \ldots, x_0) \oplus x_{n-1} [f(0, x_{n-2}, \ldots, x_0) \\ &\qquad \oplus f(1, x_{n-2}, \ldots, x_0)] \end{aligned} \tag{2}$$

Therefore, the XOR/AND expansion corresponding to variant $x_{n-1}$ of Boolean function is as follows:

$$f = f_0(x_{n-2}, \ldots, x_0) \oplus x_{n-1} f_1(x_{n-2}, \ldots, x_0) \tag{3}$$

where $f_1(x_{n-2}, \ldots, x_0) = f(0, x_{n-2}, \ldots, x_0) \oplus f(1, x_{n-2}, \ldots, x_0)$, and $f_0(x_{n-2}, \ldots, x_0) = f(0, x_{n-2}, \ldots, x_0)$.

By applying shannon theorem to each variant in turn, the Boolean function can be expressed by a FPRM as follows:

$$f^p(x_{n-1}, x_{n-2}, \ldots, x_0) = \oplus \sum_{i=0}^{2^n-1} b_i \pi_i \tag{4}$$

where $\oplus\Sigma$ denotes the modulo-2 addition, and $\pi_i = x_{n-1}x_{n-2}\ldots x_0$ represents the products of FPRM. $b_i \in \{0, 1\}$ represents whether or not$\pi_i$appears in the function, and $p = (p_{n-1}p_{n-2}\ldots p_0)$ is the polarity. In a FPRM, each variable appears either in true or in complement form, but not both. The polarity of FPRM can be represented by replacing each variable with 0 or 1 depending on whether the variable is used in true or complement, respectively. When a variable is used in true (complement), it can be replaced with 0 (1).

### B. DIFFERENTIAL EVOLUTION

Differential Evolution (DE) algorithm, which was proposed by Storn and Price [22], is a simple yet powerful global optimization algorithm to deal with continuous optimization problems. It has become a new research hotspot in evolutionary computation and has been successfully applied in scientific and engineering fields. Similar to GA, DE is also a population based algorithm, which is stochastic in nature to find global solution in feasible individual space. Moreover, only a few control parameters are required in comparison with other competing heuristic optimization methods [17]. The DE mainly includes population initialization and three evolutionary operators (i.e., mutation, crossover and selection) that are used to update the population.

#### 1) POPULATION INITIALIZATION

In DE, each individual is an $n$-dimensional vector that represents a candidate solution to the problem, which is randomly created in the search domain as follows:

$$x_{ij} = x_j^l + r \times (x_j^u - x_j^l) \tag{5}$$

where $i \in \{1, 2, \ldots, NP\}, j \in \{1, 2, \ldots, n\}$, $x_{ij}$ is the $j$-th component of the $i$-th individual $x_i$, $x_j^l$and $x_j^u$ are the lower and upper bounds of $x_j$, respectively. $NP$is the population size, and $r$ is a uniformly distributed random number in [0,1]. In initialization, all individuals are randomly generated with the uniform probability distribution.

#### 2) MUTATION OPERATOR

The basic idea of mutation is that the difference vector between two individuals is taken, and a scaled version of the difference vector is added to a third individual to create a new candidate solution. Various other mutation schemes are listed in Table 1 [18], where $x_{best}$ is the optimal individual in current

**TABLE 1. Mutation schemes of DE.**

| Mutation schemes | Mathematical equation |
|---|---|
| Best/1/exp | $v_i = x_{best} + F \times (x_{r1} - x_{r2})$ |
| Rand/1/exp | $v_i = x_{r1} + F \times (x_{r2} - x_{r3})$ |
| Rand-to-Best/1/exp | $v_i = x_i + F \times (x_{r1} - x_{r2})$ |
| Best/2/exp | $v_i = x_{best} + F \times (x_{r1} + x_{r2} - x_{r3} - x_{r4})$ |
| Rand/2/exp | $v_i = x_{r1} + F \times (x_{r2} + x_{r3} - x_{r4} - x_{r5})$ |

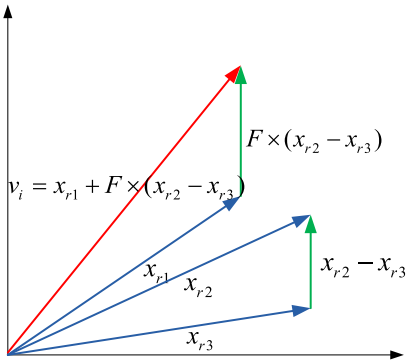


**FIGURE 1. "DE/rand/1" mutation operation.**

population. Equation (6) is one of most popular DE mutation schemes, which can be described by Fig.1.

$$v_i = x_{r1} + F \times (x_{r2} - x_{r3}) \tag{6}$$

where $v_i$ is a mutation vector, and $F$ is the scaling factor that controls the amplification of the differential vector. $x_{r1}$, $x_{r2}$ and $x_{r3}$ are candidate solutions, and $r1$, $r2$ and $r3$ are randomly selected that satisfy $r1, r2, r3 \in \{1, 2, \ldots, NP\}$ and $r1 \neq r2 \neq r3 \neq i$.

#### 3) CROSSOVER OPERATOR

The trial vector $u_i$ is generated by crossing the target vector $x_i$ with its mutation vector $v_i$. The widely used binomial crossover is defined as follows:

$$u_{ij} = \begin{cases} v_{ij}, & rand < CR \ or \ j = r \\ x_{ij}, & otherwise \end{cases} \tag{7}$$

where $j$ is the index of the dimensionality $n$, and *rand* is a stochastic number taken from the uniform distribution [0,1]. *CR* is the constant crossover rate in [0,1], and $r$ is the random number in [1,$n$].

#### 4) SELECTION OPERATOR

The fitness value $f(x_i)$ of the target vector $x_i$ is compared with the fitness value $f(u_i)$ of trial vector $u_i$ to create the new population in the next generation. The winner will survive for the next generation. Taking the minimization problem as an example, the selection process is depicted as:

$$x_i^{next} = \begin{cases} u_i, & f(u_i) \geq f(x_i) \\ x_i, & otherwise \end{cases} \tag{8}$$

## III. FAST MINIMIZATION OF FPRMS

In this section, we propose a binary version of DE according to the polarity characteristic of FPRMs, called BDE, to find the optimal solution in binary optimization space. Moreover, based on the BDE, we propose a FPRMs minimization algorithm, called FMA, which uses the BDE to search the best polarity corresponding to minimum FPRM with the fewest products. The overview of the FMA is depicted in Fig.2.

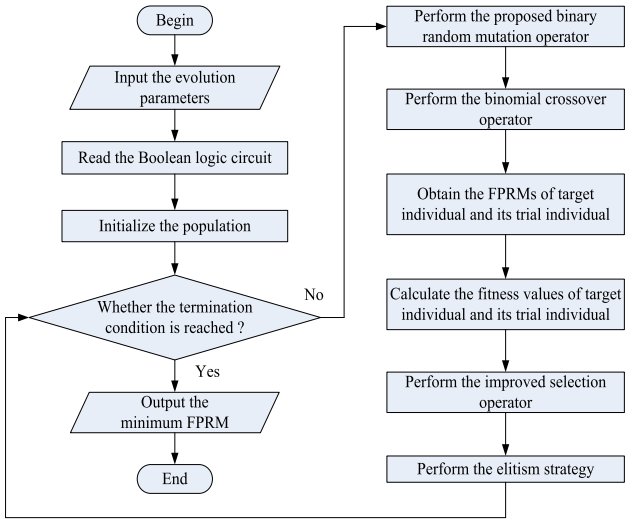


**FIGURE 2. Overview of FMA.**

### A. BINARY DIFFERENTIAL EVOLUTION ALGORITHM

Since the polarity of FPRM can be represented by replacing each variable with 0 or 1 depending on whether the variable is used in true or complement, the polarity optimization of FPRMs is binary-encoded combinational optimization problem. However, the standard DE and most of its improved variants operate in the continuous space, which are not suitable for solving polarity optimization problems of FPRMs. Moreover, the DE has not yet been used as optimizer for RM circuits until now.

In this section, we propose a binary version of DE, called BDE, which enables the DE to operate in binary spaces. The structure of the BDE is similar to the standard DE so that the advantages of DE, such as easy implementation and parameter tuning, are inherited. Moreover, to enhance the global convergence ability, we introduce the elitism strategy to the BDE, namely the worst individual in current population is replaced by the elitism individual. In addition, we propose a binary random mutation operator and an improved selection operator to increase the population diversity and improve the global searching ability.

#### 1) POPULATION INITIALIZATION

In the initialization process, the initial individuals are randomly generated as follows:

$$x_{ij} = \begin{cases} 1, & r < 0.5 \\ 0, & r \geq 0.5 \end{cases} \tag{9}$$

After a random number $r$ between 0 and 1 is generated, the $j$-th element of $i$-th individual $x_{ij}$ is set to 1 (namely, polarity 1) if $r$ is less than 1/2; otherwise, $x_{ij}$ is set to 0 (namely, polarity 0).

#### 2) FITNESS FUNCTION

In BDE, the fitness function is used to evaluate the quality of individuals. Since the higher the fitness, the better the individual, the fitness functions is defined as follows:

$$fitness(x_i) = 1.0/products(x_i) \tag{10}$$

where $fitness(x_i)$ represents the fitness value of individual $x_i$, and $products(x_i)$ represents the number of products corresponding to individual $x_i$.

#### 3) MUTATION OPERATOR

The use of logical mutation operator proposed in [19] has been widely reported in many studies. The logical mutation operator replaces the subtraction, multiplication and addition operator with XOR, AND and OR operators. It is represented as follows:

$$v_i = x_{r1} \otimes F \otimes (x_{r2} \oplus x_{r3}) \tag{11}$$

In the logical mutation operator, the subtraction of two random selected vectors was represented by XOR operator, the multiplication factor F was replaced by the AND operator, and the OR operator was applied to replace the operation for the addition of two randomly selected vectors. However, the OR operator has a higher probability of production of bit 1 in the evolution process, which could reduce the search diversity of the optimal solution.

To avoid the premature convergence and increase the population diversity, we propose a binary random mutation operator, which is represented as follows:

$$v_{i,j} = \begin{cases} x_{i,j} + (-1)^{x_{i,j}} \cdot \left| x_{r2,j} - x_{r3,j} \right|, & rand \geq 0.5 \\ x_{best,j} + (-1)^{x_{best,j}} \cdot \left| x_{r2,j} - x_{r3,j} \right|, & rand < 0.5 \end{cases} \tag{12}$$

where $v_{i,j}$ represents the $j$-th element of mutation vector $v_i$. $r2, r3 \in \{1, 2, \ldots, NP\}$ and $r2 \neq r3 \neq i$. $x_{best}$ is the optimal individual in current population, and $rand$ is a random number in [0,1].

Since the polarity of FPRMs is taken as 0 or 1, the absolute value of difference vector is only 0 or 1 and the mutated variable is still 0-1 variable. Therefore, the binary random mutation operator satisfies the closure. Moreover, whether or not one dimension variable can be mutated depends on the difference vector. Specifically, the $x_{i,j}$ or $x_{best,j}$ can be mutated (from 0 to 1 or from 1 to 0) when $\left| x_{r2,j} - x_{r3,j} \right| = 1$, which could increase the population diversity, improve the global searching ability and prevent the algorithm trapping into the local optimal solution. In addition, the probability of $x_{r2,j}$ is equal to $x_{r3,j}$ is increasing along with the population evolution. The $x_{i,j}$ or $x_{best,j}$ remain the same when $\left| x_{r2,j} - x_{r3,j} \right| = 0$, which could accelerate the convergence speed.

#### 4) IMPROVED SELECTION OPERATOR

The traditional greed selection operator chooses the winner with higher fitness value from target vector $x_i$ and trial vector $u_i$ for the next generation by comparing their fitness values. However, compared to other individuals of population, the loser may have higher fitness value. Therefore, the loser is ignored will omit better information and reduce the convergence speech. To preserve the better individual for next generation, the improved selection operation is as follows:

$$W = U \cup X \tag{13}$$

where $U$ and $X$ represent the populations consisting of individuals $u_i$ and $x_i (i \in \{1, 2, \ldots, NP\})$, respectively. $W$ contains all the individuals of $U$ and $X$, and the $NP$ better individuals are selected as child population from $W$.

### B. ALGORITHM DESCRIPTION

Based on the above description, the FMA is illustrated in Algorithm 1, in which "$G_{\max}$" represents the maximum number of iteration, "$NP$" represents the population size, and "$D$" represents the dimensionality of decision variable. As shown in Algorithm 1, the FPRMs of target individual and its trial individual are obtained by using the fixed polarity conversion algorithm [11] according to their binary codes. Then, their fitness values are calculated according to their FPRMs and fitness function.

**Algorithm 1** FMA

**Input**: The evolutionary parameters
**Output**: The minimum FPRM
**1**: Read the Boolean logic circuit;
**2**: $g \leftarrow 0$;
**3**: Initialize the population;
**4**: **while** ($g <$ G$_{\max}$)
**5**: **for** $i = 1$ to $NP$ **do**
**6**: Perform proposed binary random mutation operator;
**7**: **for** $j = 1$ to $D$ **do**
**8**: Perform the binomial crossover operator;
**9**: **end for**
**10**: Obtain FPRMs of target and trial individuals;
**11**: Calculate the fitness values of target and trial individuals;
**12**: Perform proposed improved selection operator;
**13**: **end for**
**14**: Perform the elitism strategy;
**15**: $g \leftarrow g + 1$;
**16**: **End**
**17**: Output the minimum FPRM

## IV. EXPERIMETNS RESULTS

The FMA has been implemented in C language, and the programs were compiled by the GNU C complier. The results were obtained by using a PC with Intel Core i7

**TABLE 2.** Parameters and evolution operators settings for GAFMA, SAGAFMA, and FMA.

| Parameter | GAFMA | SAGAFMA | FMA |
|---|---|---|---|
| The population size | 100 | 100 | 100 |
| The crossover probability | 0.80 | 0.80 | 0.70 |
| The mutation probability | 0.01 | 0.01 | - |
| The selection operator | Roulette wheel selection | Roulette wheel selection | Proposed improved greed selection |
| The crossover operator | One-point crossover | One-point crossover | Binomial crossover |
| The mutation operator | Basic bit mutation | Basic bit mutation | Proposed binary random mutation |
| Termination criterion | There is no improvement on the optimal solution in population over 10 iterations | | |

**TABLE 3.** Comparison of GAFMA, SAGAFMA, and FMA on the accuracy of solutions.

| Name | Input | GAFMA | | | SAGAFMA | | | FMA | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | min | avg | std | min | avg | std | min | avg | std |
| b3 | 3 | 2 | 2 | 0 | 2 | 2 | 0 | 2 | 2 | 0 |
| xor3 | 3 | 3 | 3 | 0 | 3 | 3 | 0 | 3 | 3 | 0 |
| bw | 5 | 8 | 8 | 0 | 8 | 8 | 0 | 8 | 8 | 0 |
| xor5 | 5 | 5 | 5 | 0 | 5 | 5 | 0 | 5 | 5 | 0 |
| m1 | 6 | 3 | 3 | 0 | 3 | 3 | 0 | 3 | 3 | 0 |
| Z5xp1 | 7 | 3 | 3.20 | 0.60 | 3 | 3 | 0 | 3 | 3 | 0 |
| lin | 7 | 58 | 59.30 | 2.10 | 49 | 49 | 0 | 49 | 49 | 0 |
| ex5 | 8 | 12 | 12.40 | 1.20 | 8 | 8 | 0 | 8 | 8 | 0 |
| m3 | 8 | 5 | 5.90 | 1.37 | 5 | 5 | 0 | 5 | 5 | 0 |
| rd84 | 8 | 57 | 60.10 | 3.30 | 54 | 54 | 0 | 54 | 54 | 0 |
| 1020 | 10 | 72 | 75.20 | 6.40 | 72 | 72 | 0 | 72 | 72 | 0 |
| ex1010 | 10 | 56 | 59.60 | 7.20 | 56 | 56 | 0 | 56 | 56 | 0 |
| br1 | 12 | 26 | 28.40 | 2.62 | 23 | 23 | 0 | 23 | 23 | 0 |
| 14_4color | 14 | 25 | 32.00 | 3.69 | 14 | 14 | 0 | 14 | 14 | 0 |
| table3 | 14 | 92 | 104.80 | 9.43 | 75 | 75 | 0 | 75 | 75 | 0 |
| dk48 | 15 | 6 | 10.00 | 1.79 | 6 | 6.40 | 0.80 | 6 | 6 | 0 |
| alcom | 16 | 78 | 86.60 | 4.74 | 71 | 71 | 0 | 71 | 71 | 0 |
| table5 | 17 | 647 | 664.70 | 13.07 | 618 | 621.20 | 1.60 | 618 | 618 | 0 |
| src1 | 18 | 33 | 37.00 | 2.00 | 12 | 13.20 | 1.83 | 12 | 12.20 | 0.60 |
| in2 | 19 | 325 | 355.10 | 10.72 | 277 | 283.30 | 4.12 | 277 | 277 | 0 |
| mark1 | 20 | 420 | 431.40 | 17.41 | 356 | 368.80 | 6.40 | 356 | 356 | 0 |
| mux | 21 | 27 | 28.00 | 2.00 | 23 | 23 | 0 | 23 | 23 | 0 |
| duke2 | 22 | 302 | 303.60 | 3.20 | 261 | 277.10 | 10.54 | 261 | 265.60 | 9.20 |
| cordic | 23 | 3419 | 3531.40 | 59.44 | 2148 | 2172.40 | 15.01 | 2148 | 2149.60 | 3.20 |

3.40GHz with 4G RAM under Linux. We compared the FMA with the GA based FPRMs Minimization Algorithm (GAFMA) [20] and SAGA based FPRMs Minimization Algorithm (SAGAFMA) [21] on 24 randomly selected MCNC benchmark circuits. Moreover, we ran the GAFMA, SAGAFMA, and FMA 10 times on each circuit to reduce the impact of randomness on the results. Table 2 gives the parameters and evolution operators settings for GAFMA, SAGAFMA, and FMA.

### *A. COMPARISON OF GAFMA, SAGAFMA, AND FMA ON THE ACCURACY OF SOLUTIONS*

The comparison of GAFMA, SAGAFMA, and FMA on the accuracy of solutions are listed in Table 3. We took the optimal solution (namely, minimum number of products corresponding to best polarity), average value and standard deviation as experimental data. Column 1 shows the circuit name. Column 2 shows the number of input variables. ''min'', ''avg'' and ''std'' represent the obtained optimal solution,

**TABLE 4.** Comparison of GAFMA, SAGAFMA, and FMA on the average number of iterations and run time.

| Name | Input | GAFMA | | SAGAFMA | | FMA | | Save1(%) | | Save2(%) | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | iteration | time(s) | iteration | time(s) | iteration | time(s) | iteration | time(s) | iteration | time(s) |
| b3 | 3 | 1 | 0.20 | 1 | 0.18 | 1 | 0.03 | 0.00 | 85.00 | 0.00 | 83.33 |
| xor3 | 3 | 1 | 0.19 | 1 | 0.18 | 1 | 0.05 | 0.00 | 73.68 | 0.00 | 72.22 |
| bw | 5 | 6 | 0.96 | 3 | 0.42 | 1 | 0.06 | 83.33 | 93.75 | 66.67 | 85.71 |
| xor5 | 5 | 8 | 0.82 | 4 | 0.57 | 1 | 0.05 | 87.50 | 93.90 | 75.00 | 91.23 |
| m1 | 6 | 12 | 1.18 | 4 | 0.61 | 1 | 0.06 | 91.67 | 94.92 | 75.00 | 90.16 |
| Z5xp1 | 7 | 12 | 1.22 | 5 | 0.63 | 1 | 0.06 | 91.67 | 95.08 | 80.00 | 90.48 |
| lin | 7 | 14 | 7.34 | 6 | 3.70 | 1 | 0.14 | 92.86 | 98.09 | 83.33 | 96.22 |
| ex5 | 8 | 15 | 2.85 | 8 | 1.44 | 2 | 0.37 | 86.67 | 87.02 | 75.00 | 74.31 |
| m3 | 8 | 14 | 2.21 | 6 | 1.28 | 3 | 0.55 | 78.57 | 75.11 | 50.00 | 57.03 |
| rd84 | 8 | 22 | 14.92 | 7 | 5.62 | 4 | 0.62 | 81.82 | 95.84 | 42.86 | 88.97 |
| 1020 | 10 | 24 | 23.03 | 9 | 7.76 | 6 | 4.60 | 75.00 | 80.03 | 33.33 | 40.72 |
| ex1010 | 10 | 20 | 15.78 | 12 | 6.14 | 7 | 4.81 | 65.00 | 69.52 | 41.67 | 21.66 |
| br1 | 12 | 31 | 17.02 | 16 | 9.37 | 8 | 3.44 | 74.19 | 79.79 | 50.00 | 63.29 |
| 14_4color | 14 | 55 | 30.08 | 23 | 15.26 | 10 | 7.05 | 81.82 | 76.56 | 56.52 | 53.80 |
| table3 | 14 | 70 | 37.40 | 27 | 18.70 | 18 | 11.62 | 74.29 | 68.93 | 33.33 | 37.86 |
| dk48 | 15 | 62 | 36.32 | 26 | 16.43 | 20 | 9.59 | 67.74 | 73.60 | 23.08 | 41.63 |
| alcom | 16 | 69 | 67.85 | 31 | 34.41 | 23 | 17.53 | 66.67 | 74.16 | 25.81 | 49.06 |
| table5 | 17 | 81 | 84.99 | 52 | 52.90 | 31 | 22.42 | 61.73 | 73.62 | 40.38 | 57.62 |
| src1 | 18 | 73 | 173.55 | 37 | 139.34 | 29 | 54.63 | 60.27 | 68.52 | 21.62 | 60.79 |
| in2 | 19 | 84 | 158.64 | 41 | 127.88 | 33 | 32.26 | 60.71 | 79.66 | 19.51 | 74.77 |
| mark1 | 20 | 127 | 240.91 | 40 | 195.61 | 30 | 60.75 | 76.38 | 74.78 | 25.00 | 68.94 |
| mux | 21 | 95 | 226.33 | 34 | 174.26 | 27 | 68.90 | 71.58 | 69.56 | 20.59 | 60.46 |
| duke2 | 22 | 116 | 262.72 | 45 | 223.15 | 34 | 75.18 | 70.69 | 71.38 | 24.44 | 66.31 |
| cordic | 23 | 145 | 318.05 | 67 | 257.18 | 42 | 95.84 | 71.03 | 69.87 | 37.31 | 62.73 |

average value and standard deviation respectively, which were taken after running each approach on each test circuit ten times.

From the Table 3, we can find that for the circuits with fewer variables (such as b3, xor3 and xor5), GAFMA was able to find the optimal solution each time. However, for the circuits that have more than 10 variables (such as br1, table5 and duke2), the GAFMA could hardly obtain the optimal solution. Moreover, for some circuits (such as table5, mark1 and cordic), the corresponding average values and standard deviations are large, which illustrates that the GAFMA is not appropriate to solve the large-scale polarity optimization problem due to its weak local searching ability. Additionally, we can also find that the SAGAFMA and FMA have the same searching performance for the circuits that have less than 15 variables. However, for some circuits that have more than 15 variables (such as src1, mark1 and cordic), although the SAGAFMA can also find the optimal solutions, the corresponding average values and standard deviations are larger than that of FMA. In addition, we can find that for the 24 test circuits, there are 21 circuits whose minimum FPRMs can be obtained by FMA each time (namely, the corresponding standard deviation is 0) and there are 17 circuits whose minimum FPRMs can be obtained by SAGAFMA each time. Therefore, from the comparison experiment we can come to the conclusion that the FMA outperforms GAFMA and SAGAFMA on almost all the test circuits in terms of the accuracy of solutions.

### B. COMPARISON OF GAFMA, SAGAFMA, AND FMA ON THE AVERAGE NUMBER OF ITERATIONS AND RUN TIME

The comparison of GAFMA, SAGAFMA, and FMA on the average number of iterations and run time (in CPU seconds) over 10 independent run are listed in Table 4. Columns 3 and 4 show the average number of iterations and run time of GAFMA. Columns 5 and 6 show the average number of iterations and run time of SAGAFMA. Columns 7 and 8 show the average number of iterations and run time of FMA.

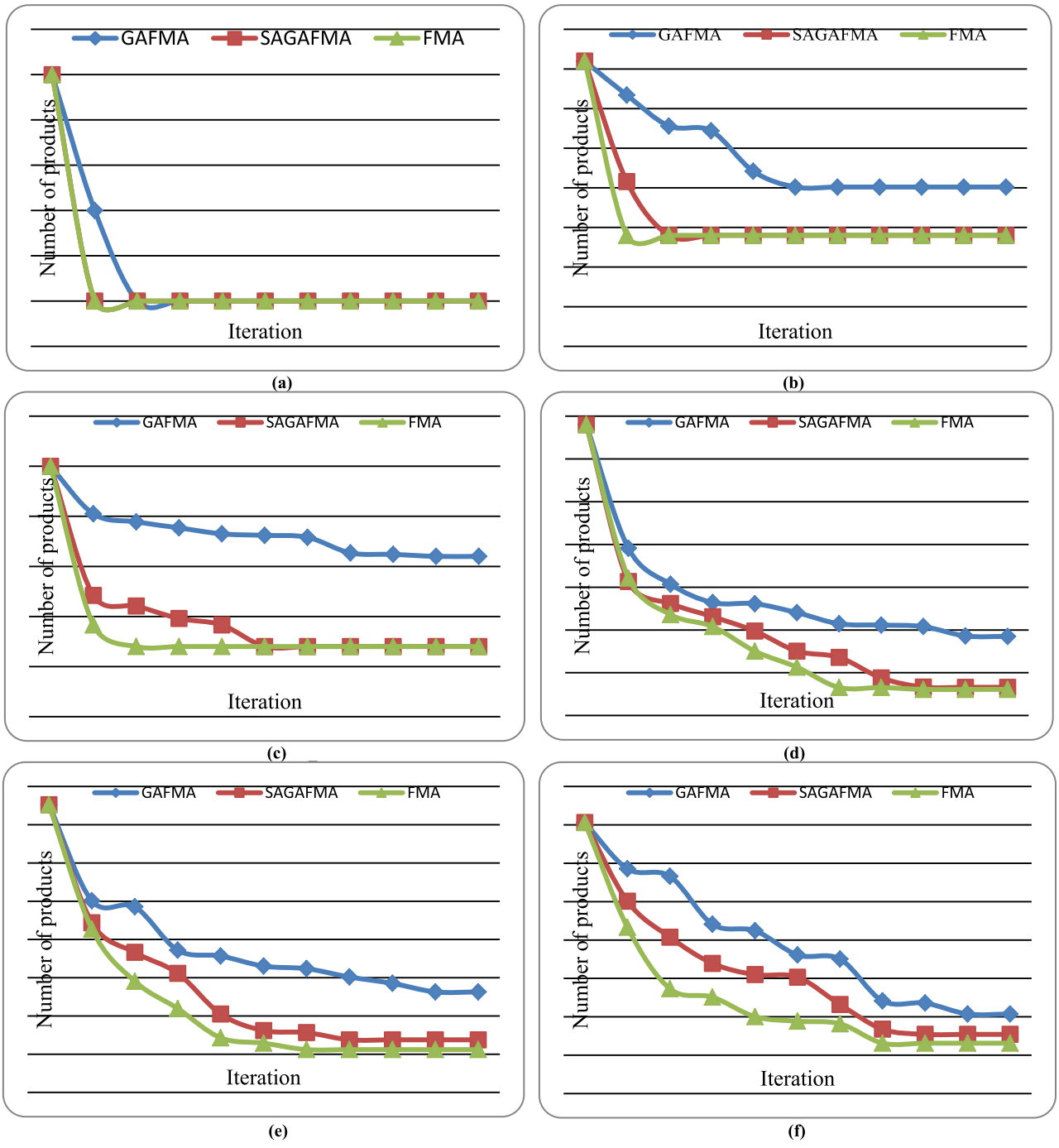


**FIGURE 3.** Convergence comparison of GAFMA, SAGAFMA, and FMA. (a) bw. (b) rd84. (c) 14_4color. (d) src1. (e) mark1. (f) duke2.

Columns 9 and 10 denote the percentage of the number of iterations and run time saved by FMA compared to GAFMA, which are defined as:

$$Save1_{iteration} = \frac{GAFMA_{iteration} - FMA_{iteration}}{GAFMA_{iteration}} \times 100\%0 \tag{14}$$

$$Save1_{time} = \frac{GAFMA_{time} - FMA_{time}}{GAFMA_{time}} \times 100\% \tag{15}$$

Columns 11 and 12 denote the percentage of the number of iterations and run time saved by FMA compared to SAGAFMA, which are defined as:

$$Save2_{iteration} = \frac{SAGAFMA_{iteration} - FMA_{iteration}}{SAGAFMA_{iteration}} \times 100\% \tag{16}$$

$$Save2_{time} = \frac{SAGAFMA_{time} - FMA_{time}}{SAGAFMA_{time}} \times 100\% \tag{17}$$

From the Table 4, it is clear that the FMA outperforms the GAFMA and SAGAFMA in improving the minimization efficiency of FPRMs. Compared to the GAFMA, the greatest improvement in the number of iterations and run

time, which were made by FMA, are 92.86% and 98.09%, respectively. Moreover, compared to the SAGAFMA, the greatest improvement in the number of iterations and run time, which were made by FMA, are 83.33% and 96.22%, respectively. Therefore, it is concluded that compared to the GAFMA and SAGAFMA, the FMA has higher minimization efficiency both for the small-scale circuits and large-scale circuits.

### C. COMPARISON OF GAFMA, SAGAFMA, AND FMA ON THE CONVERGENCE

In order to get a comprehensive analysis on the convergence of GAFMA, SAGAFMA, and FMA, we selected 6 representative MCNC benchmark circuits, namely, two small-scale circuits bw and rd84, two medium-sized circuits 14_4color and src1, and two large-scale circuits mark1 and duke2. Fig.3 shows the convergence comparison of GAFMA, SAGAFMA, and FMA in every 5 iterations, in which each algorithm was run 10 times on each circuit and the average values were used as the experimental data. Moreover, the horizontal axis shows the number of iterations, and the vertical axis shows the number of products.

As can be seen from the Fig.3, the convergence speed of GAFMA is much slower than that of SAGAFMA and FMA, and the GAFMA could not converge to the global optimal solutions except for the small-scale test circuit bw. In addition, it can be seen that the convergence curves of SAGAFMA and FMA have similar changing tendency. However, the convergence speed of FMA is much faster than that of SAGAFMA. Specifically, the FMA converges in 4 iterations while SAGAFMA needs 7 iterations on rd84; the FMA converges in 10 iterations while SAGAFMA needs 23 iterations on 14_4color; the FMA converges in 29 iterations while SAGAFMA needs 37 iterations on src1; the FMA converges in 30 iterations while SAGAFMA needs 40 iterations on mark1; the FMA converges in 34 iterations while SAGAFMA needs 45 iterations on duke2. Moreover, it is worthy of noting that compared to the SAGAFMA, the FMA can derive either the same or better solutions.

## V. CONCLUSION

Minimization of FPRMs is a computationally hard problem, because the polarity optimization space increases exponentially with the increase of the number of input variables. In this paper, we propose a fast minimization algorithm of FPRMs, called FMA, which uses proposed BDE to search the best polarity corresponding to minimum FPRM with the fewest products. The experimental results over MCNC benchmark circuits demonstrate that the FMA performs superior to, or at least comparable to, the GA based and SAGA based FPRMs minimization algorithms in terms of the accuracy of solutions and solving efficiency, and confirm the application of BDE as a promising tool for solving the polarity optimization problem of FPRMs. In future, we will study an efficient ternary-encoded DE algorithm to minimize mixed polarity RM expressions.

## REFERENCES

[1] S. Shirinzadeh, M. Soeken, P.-E. Gaillardon, and R. Drechsler, "Logic synthesis for RRAM-based in-memory computing," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 37, no. 7, pp. 1422–1435, Jul. 2018.
[2] Q. Zhang, P. Wang, J. Hu, and H. Zhang, "Cube-based synthesis of ESOPs for large functions," *Chin. J. Electron.*, vol. 27, no. 3, pp. 527–534, Jan. 2018.
[3] S. D. Kumar, H. Thapliyal, and A. Mohammad, "FinSAL: FinFET-based secure adiabatic logic for energy-efficient and DPA resistant IoT devices," *IEEE Trans. Comput-Aided Des. Integr. Circuits Syst.*, vol. 37, no. 1, pp. 110–122, Jan. 2018.
[4] C. Chen, B. Lin, and M. Zhu, "Verification method for area optimization of mixed-polarity Reed-Müller logic circuits," *J. Eng. Sci. Technol. Rev.*, vol. 11, no. 1, pp. 28–34, 2018.
[5] R. Ueno, N. Homma, T. Aoki, and S. Morioka, "Hierarchical formal verification combining algebraic transformation with PPRM expansion and its application to masked cryptographic processors," *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, vol. E100-A, no. 7, pp. 1396–1408, 2017.
[6] H.-H. Huang, H. Cheng, C. Chu, and P. A. Beerel, "Area optimization of timing resilient designs using resynthesis," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 37, no. 6, pp. 1197–1210, Jun. 2018.
[7] X. Wang, Y. Lu, Y. Zhang, Z. Zhao, T. Xia, and L. Xiao, "Power optimization in logic synthesis for mixed polarity Reed-Müller logic circuits," *Comput. J.*, vol. 58, no. 6, pp. 1307–1313, 2014.
[8] X. Wang, M. Li, Z. He, W. Wang, C. Zhou, and Z. Zhao, "PAOA: A power and area optimization approach of Reed-Müller logic circuits," in *Proc. IEEE Int. Midwest Symp. Circuits Syst.*, Aug. 2017, pp. 1394–1397.
[9] D. Debnath and T. Sasao, "Exact minimization of FPRMs for incompletely specified functions by using MTBDDs," *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, vol. E88-A, no. 12, pp. 3332–3341, 2005.
[10] D. Debnath and T. Sasao, "Exact minimization of fixed polarity Reed-Müller expressions for incompletely specified functions," in *Proc. Asia South Pacific Design Automat. Conf.*, Jan. 2000, pp. 247–252.
[11] T. Qu, L. Wang, and W. Luo, "A novel method for large ISFPRM function optimization," *ACTA Electronica Sinica*, vol. 46, no. 5, pp. 1101–1106, 2018.
[12] D. Apangshu and P. S. Nath, "Shared Reed-Müller decision diagram based thermal-aware AND-XOR decomposition of logic circuits," *VLSI Des.*, vol. 2016, Mar. 2016, Art. no. 3191286.
[13] D. Apangshu and P. S. Nath, "Thermal aware FPRM based AND-XOR network synthesis of logic circuits," in *Proc. IEEE Int. Conf. Recent Trends Inf. Syst.*, Jul. 2015, pp. 497–502.
[14] H. H. Zhang, P. Wang, and X. Gu, "Area optimization of fixed-polarity Reed-Müller circuits based on niche genetic algorithm," *Chin. J. Electron.*, vol. 20, no. 1, pp. 27–30, 2011.
[15] S. Chattopadhyay, S. Roy, and P. P. Chaudhuri, "Synthesis of highly testable fixed-polarity AND-XOR canonical networks—A genetic algorithm-based approach," *IEEE Trans. Comput*, vol. 45, no. 4, pp. 487–490, Apr. 1996.
[16] X. Qiu, J.-X. Xu, Y. Xu, and K. C. Tan, "A new differential evolution algorithm for minimax optimization in robust design," *IEEE Trans. Cybern.*, vol. 48, no. 5, pp. 1355–1368, May 2018.
[17] C.-F. Chang, J.-J. Wong, J.-P. Chiou, and C.-T. Su, "Robust searching hybrid differential evolution method for optimal reactive power planning in large-scale distribution systems," *Elect. Power Syst. Res.*, vol. 77, nos. 5–6, pp. 430–437, 2007.
[18] C.-Y. Wu and K.-Y. Tseng, "Topology optimization of structures using modified binary differential evolution," *Struct. Multidisciplinary Optim.*, vol. 42, pp. 939–953, Dec. 2010.
[19] X. He and L. Han, "A novel binary differential evolution algorithm based on artificial immune system," in *Proc. IEEE Congr. Evol. Comput.*, Sep. 2007, pp. 2267–2272.
[20] R. Drechsler, B. Becker, and N. Drechsler, "Genetic algorithm for minimisation of fixed polarity Reed-Müller expressions," *IEE Proc.-Comput. Digit. Techn.*, vol. 147, no. 5, pp. 349–353, Oct. 2000.
[21] P. Wang, H. Li, and Z. Wang, "MPRM expressions minimization based on simulated annealing genetic algorithm," in *Proc. IEEE Int. Conf. Intell. Syst. Knowl. Eng.*, Nov. 2010, pp. 261–265.
[22] R. Storn and K. Price, "Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, 1997.

**ZHENXUE HE** received the Ph.D. degree in computer architecture from Beihang University, Beijing, China, in 2018. He is currently a Full Associate Professor with Hebei Agricultural University. He has authored or co-authored over 20 papers in peer-reviewed journals and proceedings. His research interests include low-power integrated circuit design and optimization, multiple-valued logic circuits, and intelligent algorithm. He is a member of the China Computer Federation.

**LIMIN XIAO** is currently a Professor with the School of Computer Science and Engineering, Beihang University, China. His main research areas include computer architecture, computer system software, high-performance computing, virtualization, and cloud computing. He is a Senior Member of the China Computer Federation.

**ZHISHENG HUO** received the M.S. degree from the College of Computer Science, Shenyang Aerospace University, Shenyang, China, in 2012, and the Ph.D. degree in computer architecture from Beihang University, Beijing, China, in 2018, where he currently holds a Postdoctoral position with the School of Computer Science and Engineering. His research focuses on big data storage and distributed storage systems.

**TAO WANG** was born in Heyang, Shaanxi, China, in 1986. He is currently pursuing the Ph.D. degree in microelectronics and solid electronics with Beihang University, Beijing, China. He has been with the School of Electronic and Information Engineering, Beihang University, since 2014. His current research interests include data fusion, target localization and tracking, and space-sky information networks.

**XIANG WANG** is currently a Professor with the School of Electronic and Information Engineering, Beihang University, China. His main research areas include very large-scale integration, micro-nano systems, genetic circuits, and aerospace information networks. He is a Senior Member of the Chinese Institute of Electronics and the Chinese Society of Micro-Nano Technology.