Hindawi
VLSI Design
Volume 2018, Article ID 8258397, 8 pages
https://doi.org/10.1155/2018/8258397

Research Article

Novel Verification Method for Timing Optimization Based on DPSO

Chuandong Chen, Rongshan Wei, Shaohao Wang, and Wei Hu

Department of Microelectronics, Fuzhou University, 2 Xue Yuan Road, University Town, Fuzhou, Fujian, China

Correspondence should be addressed to Chuandong Chen; cdchen@fzu.edu.cn

Received 29 September 2017; Revised 30 January 2018; Accepted 18 February 2018; Published 21 March 2018

Academic Editor: Mohamed Masmoudi

Timing optimization for logic circuits is one of the key steps in logic synthesis. Extant research data are mainly proposed based on various intelligence algorithms. Hence, they are neither comparable with timing optimization data collected by the mainstream electronic design automation (EDA) tool nor able to verify the superiority of intelligence algorithms to the EDA tool in terms of optimization ability. To address these shortcomings, a novel verification method is proposed in this study. First, a discrete particle swarm optimization (DPSO) algorithm was applied to optimize the timing of the mixed polarity Reed-Muller (MPRM) logic circuit. Second, the Design Compiler (DC) algorithm was used to optimize the timing of the same MPRM logic circuit through special settings and constraints. Finally, the timing optimization results of the two algorithms were compared based on MCNC benchmark circuits. The timing optimization results obtained using DPSO are compared with those obtained from DC, and DPSO demonstrates an average reduction of 9.7% in the timing delays of critical paths for a number of MCNC benchmark circuits. The proposed verification method directly ascertains whether the intelligence algorithm has a better timing optimization ability than DC.

1. Introduction

Timing optimization for logic circuits is one of the most important requirements during logic synthesis [1–4]. The industry's mainstream electronic design automation (EDA) tool for logic synthesis is Design Compiler (DC) produced by Synopsys®. Timing optimization algorithms for logic circuits have been integrated in DC and have been demonstrated to reduce critical path delays significantly [5, 6]. Figure 1 illustrates the positioning of logic circuits (represented by ellipses) between flip-flops FF1 and FF2 of a pipeline circuit [7, 8], where the logic circuits generally employ multiple inputs and outputs. As shown by the red line in the figure, a typical path, denoted as the critical path, for the process of timing optimization in DC begins from the clock pin of FF1, passes through logic circuits, and finally arrives at the D pin of FF2. Here, the clock pin of FF1 is the startpoint of the critical path, and the D pin of FF2 is the endpoint of the critical path.

To facilitate timing optimization, logic circuits with n inputs and m outputs are transformed to equivalent mixed polarity Reed-Muller (MPRM) [9, 10] circuits, which have 3^n mixed polarities. Abundant experimental data and conclusions are available regarding the timing optimization of an MPRM logic circuit based on intelligence algorithms. However, the experimental results of intelligence algorithms cannot be compared with those of mainstream EDA tools, thus making the verification of the superiority of intelligence algorithms over EDA tools impossible.

To address these problems, this research proposed a new related verification method. The suggested verification method allows for contrast analysis between the experimental results of an intelligence algorithm and the optimization outcomes of an EDA tool, thereby evaluating the optimization ability of the intelligence algorithm.

2. State of the Art

MPRM is a standard form of a logic circuit; the problem of timing-driven logic optimization for MPRM circuits is an nondeterministic polynomial (NP) time complete problem [11]. The discrete particle swarm optimization (DPSO) algorithm [12–15], genetic algorithm (GA) [16, 17], and simulated

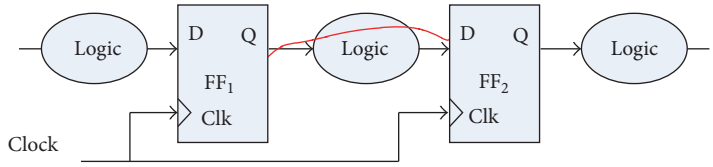

Figure 1: A typical path (red line) for the timing optimization process of a pipeline circuit in the Design Compiler (DC) tool beginning from the clock pin of flip-flop FF_1, passing through logic circuits, and finally arriving at the D pin of flip-flop FF_2.

annealing (SA) [18–20] are commonly employed methods for solving the timing optimization of MPRM logic circuits.

On the basis of research on discrete ternary particle swarm optimization, Yu et al. [21] proposed a ternary diversity particle swarm optimization (TDPSO). A mathematic mode for area and low power dissipation was built and mixed polarity conversion of XNOR/OR circuits was improved. The algorithm was tested using MCNC benchmark circuits. Experimental results showed that the algorithm significantly outperformed the reported method. However, the experimental results were not compared with the optimization outcomes of the EDA tool. Thus, determining whether this algorithm is better than the EDA tool in terms of timing optimization is impossible.

To improve the efficiency of the polarity optimization of MPRM logic circuits, He et al. [22] proposed an efficient and fast polarity optimization approach (FPOA) considering the polarity conversion sequence. FPOA performed better for a complicated MPRM logic circuit. Nevertheless, experimental data on this algorithm were not compared with optimization outcomes of the EDA tool.

Zhang et al. [23] proposed an innovative niche GA for area optimization of fixed-polarity RM circuits. Experimental results of the MCNC benchmark circuits showed that the proposed algorithm was superior to the traditional GA. However, this study compared niche GA and traditional GA but did not compare the optimization results of the niche GA with those of the EDA tool.

Wang et al. [24] presented a power estimation model for MPRM logic circuits, which accurately and efficiently handled temporal signal correlations during the estimation of average power using lag-one Markov chains. An ordered binary decision diagram-based procedure was used to propagate the temporal correlations from the primary inputs throughout the network. Unfortunately, the experimental data of the model were impossible to compare with experimental data of the mainstream EDA tool.

Bu and Jiang [11] proposed a hybrid multivalued DPSO for the minimization problem of the MPRM logic circuit. Compared with the simulated annealing genetic algorithm (SAGA), this hybrid multivalued DPSO increased the time efficiency of MPRM minimization while achieving comparable optimization results. However, the experimental data were compared only with the optimization results of SAGA and not with the optimization results of the mainstream EDA tool.

Yang and Xu [25] suggested a whole annealing genetic algorithm (WAGA), which was used to search for MPRM functions for obtaining optimal circuit implementation. By combining the global searching ability of a GA and the local searching ability of SA, WAGA could achieve fast convergence. The algorithm was more effective than other GA methods in searching for the best MPRM functions. Nevertheless, experimental results were unable to prove the superiority of WAGA to the EDA tool.

Many studies are available on the timing optimization of MPRM logic circuits based on different intelligence algorithms. Their experimental results were mainly proposed based on intelligence algorithms. No verification method exists for comparing the timing optimization results of an EDA tool and an intelligence algorithm to determine whether an intelligence algorithm is superior to the EDA tool for timing optimization.

Accordingly, a novel verification method was designed in this study. We employ MCNC benchmark circuits in the Berkeley Logic Interchange Format (BLIF) and limit the logic gates that can be mapped in the technology library to two-input AND gates, two-input XOR gates, and inverters. First, the DPSO algorithm is applied for timing optimization of the MCNC benchmark circuits according to the BLIF netlist. Then, the benchmark circuits with BLIF format are transformed to the VHSIC Hardware Description Language (VHDL) format and then compiled in DC. The critical paths are then calculated according to the netlist saved in DC, and the timing optimization results are reported and compared with the results obtained from the DPSO algorithm.

3. The Proposed Method

In this section, a novel verification method is proposed. First, the MPRM circuit and its mathematical model are introduced. Second, the calculation equations for timing delay of a critical path with DPSO and with DC are proposed. Finally, the steps of the DPSO algorithm for timing optimization and the verification process are proposed.

3.1. MPRM Mathematic Model. The Boolean expression of n-input combinational logic circuit is

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

where $\sum$ is the logical OR operation of a set of variables; a_i is the minterm coefficient; i is the minterm ordinal, which can be expressed as $(i_{n-1}, i_{n-2}, \ldots, i_0)$; m_i is the minterm, which can be expressed as $(x_{n-1}x_{n-2}\cdots x_j\cdots x_0)$. The relationship

Table 1: Values of the MPRM expansion term x_j.

j_i	0	0	0	1	1	1
p_i	0	1	2	0	1	2
x_i	1	1	$\overline{x}_i$	x_i	$\overline{x}_i$	x_i

between x_k and its negation $\overline{x}_k$, collectively denoted as $\dot{x}_k$, and i_k, is given as follows:

$$\dot{x} = \begin{cases} x_k, & i_k = 1 \\ \overline{x}_k, & i_k = 0, \end{cases} \quad 0 \le k \le n-1. \tag{2}$$

An MPRM circuit with n inputs has 3^n expansions, which can be expressed as follows:

$$f^p(x_{n-1}, x_{n-2}, \ldots, x_0) = \oplus \sum_{j=0}^{2^n-1} \left(b_j \pi_j\right). \tag{3}$$

Here, $(x_{n-1}, x_{n-2}, \ldots, x_0)$ is the input of the MPRM circuit. p $(0 \le p \le 3^n - 1)$ represents the polarity value of the MPRM and could be expressed as $(p_{n-1}p_{n-2}\cdots p_i \cdots p_0)$, where $p_i \in \{0, 1, 2\}$. It is a ternary system. $\oplus\sum$ is XOR; $b_j = [b_{0,j}, b_{1,j}, \ldots, b_{m-1,j}]^T$ is the jth coefficient vector of MPRM; and $\pi_j = (x_{n-1}x_{n-2}\cdots x_j \cdots x_0)$ is the jth product item of the MPRM. Values of x_j in term "AND" are shown in Table 1.

3.2. Timing Model. The single logic path among all paths that has the longest timing delay from the startpoint to the endpoint is denoted as the critical path. The timing optimization developed in this paper for logic circuits regards the critical path as the basic optimization target. Here, the startpoint of a logic path is taken as the clock pin of the front flip-flop, the endpoint is the input pin of the next flip-flop, and the circuits between the startpoint and the endpoint consist of logic gates.

Figure 2 illustrates the combinational logic circuits with n inputs and m outputs employed in the present study, where the inputs are $(x_n, x_{n-1}, \ldots, x_1)$ and Y_k is one of the outputs. The logic circuit with endpoint Y_k can be expressed by (3). This combinational logic circuit can be decomposed into a logical network comprising two-input AND gates, two-input XOR gates, and inverters. When the logic of the MPRM circuit has been optimized using the DPSO algorithm, which ignores the timing delay for interconnected wires between the pins of logic gates, the corresponding maximum timing delay $T_{\max}$ can be expressed as

$$T_{\max} = \max\left(T_i\right), \tag{4}$$

where T_i represents the number of logic gates from any input to any output in the combinational logic circuits. For comparison, the same logic circuits are synthesized by DC, which also ignores the timing delay for interconnected wires between the pins of logic gates, and the logic gates that can be mapped to netlist in DC are limited to two-input AND gates, two-input XOR gates, and inverters. Finally, the logic circuit timing is constrained for optimization and mapped to the technology library in DC, and every logic gate in the critical path is defined as a timing delay unit. Thus, the maximum timing delay of the critical path obtained from DC can be defined as follows:

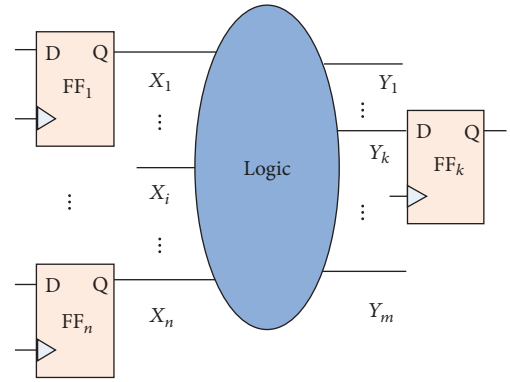

Figure 2: Combinational logic circuit with n inputs and m outputs, where the inputs are $(x_n, x_{n-1}, \ldots, x_1)$ and Y_k is one of the outputs.

$$T'_{\max} = W, \tag{5}$$

where W represents the total number of logic gates from the startpoint to the endpoint of the critical path, which can be calculated according to timing reports of the critical path generated by DC.

3.3. DPSO Algorithm. Particle swarm optimization (PSO) is a global optimal search algorithm based on swarm intelligence theory. Because of its simple process, few parameters, good convergence, and robustness, PSO has been commonly employed for addressing optimization problems. DPSO is an effective endmember extraction algorithm based on PSO with the advantage of more rapid convergence.

In the DPSO algorithm, the position and velocity of each particle in a swarm are initialized randomly in the solution space. Assuming that the total number of particles is m in an n-dimensional search space, the position of the ith particle can be expressed as $x_i = (x_{i,0}, x_{i,1}, \ldots, x_{i,j}, \ldots, x_{i,n-1})$, and its flying velocity can be depicted as $v_i = (v_{i,0}, v_{i,1}, \ldots, v_{i,j}, \ldots, v_{i,n-1})$. In addition, the optimal position of particle i is expressed as $pbest_i = (pbest_{i,0}, pbest_{i,1}, \ldots, pbest_{i,j}, \ldots, pbest_{i,n-1})$, and the optimal location of the swarm is expressed as $gbest_i = (gbest_0, gbest_1, \ldots, gbest_j, \ldots, gbest_{n-1})$. In addition, the update equations for the velocity and position of particle i at the $(t+1)$th iteration can be expressed for the jth dimension as follows:

$$v_{i,j}(t+1) = w \cdot v_{i,j}(t) + c_1 \cdot r_1\left(pbest_{i,j} - x_{i,j}(t)\right) + c_2 \cdot r_2 \cdot \left(gbest_j - x_{i,j}(t)\right) \tag{6}$$

$$x_{i,j}(t+1) = \text{round}\left(\frac{M}{1 + \exp\left(-v_{i,j}(t+1)\right)} + (M-1) \cdot k \cdot \text{random}(\,)\right), \tag{7}$$

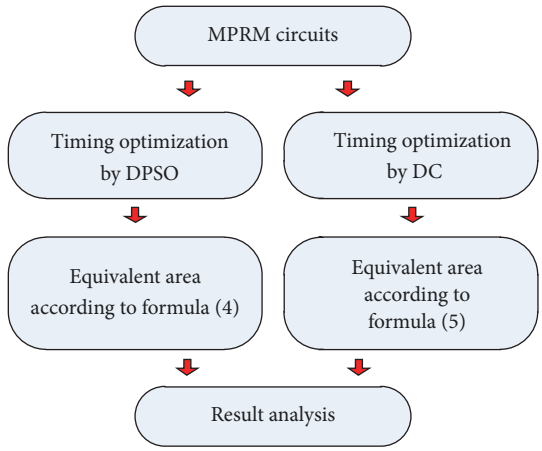

Figure 3: Verification process.

where w is the inertial weight at the tth iteration, c_1 and c_2 are the acceleration factors for adjusting *pbest* and *gbest*, r_1 and r_2 are random numbers in $[0, 1]$, round$(\cdot)$ is the integer operation function, the value of M is 3 [21], k is a constant, and random$(\cdot)$ is the standard normal distribution function. The steps of the DPSO algorithm are given as follows.

Step 1. Initialize the parameters of DPSO.

Step 2. Read the BLIF logical netlist and perform polarity conversion.

Step 3. Update $v_{i,j}$ and $x_{i,j}$ according to (6) and (7), respectively, and let $j = j + 1$.

Step 4. If $j < n$, go to Step 3; otherwise, let $i = i + 1$ and go to Step 5.

Step 5. If $i < Q$, where Q is the size of the particle swarm, go to Step 3; otherwise, go to Step 6.

Step 6. Output the optimal MPRM circuit result.

3.4. Verification Method. To compare the timing optimization of the MPRM combinational logic circuit with that of the DPSO and DC algorithms, a verification method was designed. The verification process is shown in Figure 3. MCNC benchmark circuits were used as the MPRM circuit in the verification process. First, the DPSO algorithm was used for timing optimization of the MCNC benchmark circuit, and an equivalent timing was calculated according to (4). Second, the format of the MCNC benchmark circuit was modified. A DC logic synthesis tool was used to restrict conditions such that the available logic gate types were completely consistent with the MPRM. Timing constraint was executed strictly, and compiling was performed. The number of equivalent logic gates was calculated according to the timing optimization results of the DC logic synthesis and (5). Finally, the timing optimization results of the DC logic synthesis and the DPSO were compared.

The DPSO algorithm was implemented using the C++ language and compiled using the g++ compiler under Linux. The timing of the MCNC benchmark circuit was optimized through the DPSO algorithm. For DC recognition and timing optimization of the MCNC benchmark circuit, the said circuit was transformed into the VHDL format. Each optimization result was tested independently 20 times. The DC process is described in detail as follows.

Step 1. MCNC benchmark circuits in BLIF format are translated into logic circuits in VHDL format.

Step 2. In the DC environment, the logic circuits in VHDL format are constrained for timing optimization and are constrained to generate the logic netlist that contains only two-input AND gates, two-input XOR gates, and inverters.

Step 3. According to the timing optimization results, the startpoint and endpoint of the critical path are reported, the number of logical gates on the critical path is added up, and the value of $T'_{\max}$ is calculated with (5) while ignoring the timing delay of wires between logic gates.

Step 4. Repeat Steps 1–3 above 20 times for each MCNC benchmark circuit.

4. Result Analysis and Discussion

The DPSO algorithm was used for the timing optimization of 17 MPRM logic circuits. The same combinational logic circuit was operated independently 20 times, and the timing of the optimized combinational logic circuit was calculated according to (4). Then, the DC algorithm was used for timing constraint and for setting the same MCNC benchmark circuits to ensure that the logic gates covered in the netlist after logic synthesis are completely consistent with the logic gates in the MPRM circuits. Timing constraint and optimization were performed using the DC algorithm. The timing of the combinational logic circuits was calculated according to (5). Finally, the optimized timing of the combinational logic circuits obtained by the two algorithms was compared to verify the validity of the DPSO algorithm.

As one of the MCNC benchmark circuits, "alu2" has 10 input ports and six output ports. In accordance with the

Table 2: The average (Avg) timing delays obtained for MCNC benchmark circuits using three intelligent algorithms.

Circuits	I/O*	TDPSO Avg/s	SADPSO Avg/s	WAGA Avg/s	DPSO Avg/s
alu2	10/8	44	42	45	44
alu4	14/8	44	43	47	45
b9	41/21	9	9	9	9
cm85a	11/3	12	12	12	12
comp	32/3	23	22	23	23
count	36/16	22	21	22	22
dalu	75/16	25	25	26	26
k2	45/45	23	22	23	23
my_adder	33/17	51	49	53	51
pcler8	27/17	13	13	13	13
pcle	19/9	10	10	10	10
pm1	16/13	7	7	7	7
t481	16/1	20	20	20	20
terml	34/10	15	15	15	15
too_large	39/3	28	27	29	29
ttt2	25/21	12	12	12	12
vda	17/40	17	17	17	17

*The total number of inputs and outputs.

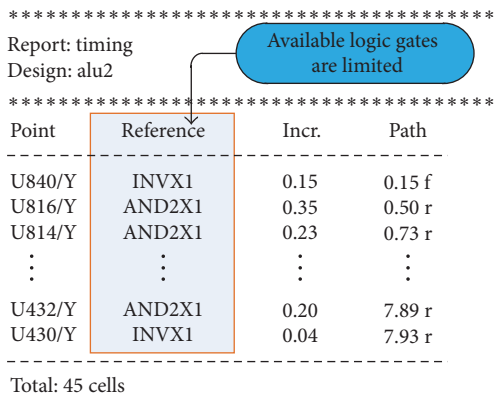

Figure 4: Timing optimization after restriction of available logic gates.

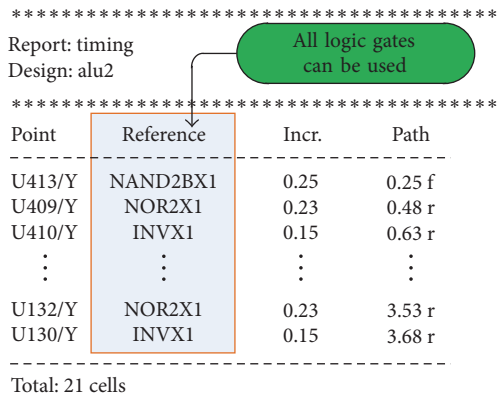

Figure 5: Timing optimization without restriction on logic gates.

features of MPRM logic circuits, the available logic gates were restricted and timing optimization was performed using the DC algorithm. The timing report of the critical path is shown in Figure 4. Dotted lines reflect the type of the used logic gates and the available logic gates that were restricted into the AND gate, phase inverter, and the XOR gate. According to (4), the equivalent timing delay of "alu2" is 45 seconds.

The circuit after normal timing optimization by DC without restriction on the available logic gates in alu2 is shown in Figure 5. Dotted lines show the types of the used logic gates. Any logic gate in the standard cell library can be used. According to the netlist after the logic synthesis of alu2 and (5), the equivalent timing delay of alu2 is 170 seconds. A comparison of the results in Figures 4 and 5 revealed that the equivalent timing delay in alu2 increased sharply after the available logic gates were restricted. This trend occurred because, during the timing optimization in DC after the restriction, the logic gates that can be used were restricted into the two-input AND gate, phase inverter, and two-input XOR gate according to the MPRM circuit. Therefore, more logic gates were needed to form the logic gate. Without a restriction on the logic gates, DC will use any logic gate to optimize the circuit according to the needs of the logic circuit, thus generating a logic circuit with a small number of equivalent timing delays.

Comparisons between the timing optimization results obtained using the TDPSO algorithm [21], SADPSO algorithm [11], WAGA algorithm [25], and DPSO algorithm are listed in Table 2.

Here, the value of "I/O" represents the total number of inputs and outputs of the logic circuits, "Circuits" represents the names of MCNC benchmark circuits, and "Avg" represents the average value of equipment timing delay. The

Table 3: Experimental results indicative of the input port (startpoint) and the output port (endpoint) of the critical path and the minimum (Min) and average (Avg) timing delays obtained for MCNC benchmark circuits using the DPSO algorithm and DC.

Circuits	I/O*	Critical path		DPSO		DC	
		Startpoint	Endpoint	Min/s	Avg/s	Min/s	Avg/s
alu2	10/8	h	o	42	44	46	46
alu4	14/8	i	r	43	45	47	47
b9	41/21	b0	z0	9	9	10	10
cm85a	11/3	d	l	11	12	13	13
comp	32/3	a0	g0	21	23	25	25
count	36/16	r	z0	20	22	24	24
dalu	75/16	muse12	o14	21	26	26	26
k2	45/45	d0	e2	21	23	24	24
my_adder	33/17	f0	x0	47	51	55	55
pcler8	27/17	t	r0	12	13	14	14
pcle	19/9	l	b0	9	10	11	11
pm1	16/13	c	c1	7	7	8	8
t481	16/1	v4	v16_0	20	20	21	21
terml	34/10	d0	s0	14	15	17	17
too_large	39/3	y	p0	28	29	30	30
ttt2	25/21	m	q0	12	12	14	14
vda	17/40	l	f0	15	17	17	17

*The total number of inputs and outputs.

same combinational logic circuit was operated independently 20 times, the average timing delays obtained for MCNC benchmark circuits using four intelligent algorithms are listed in Table 2, and the average equivalent timing delay generated by the four algorithms is generally close.

Comparisons between the timing optimization results obtained using the DPSO algorithm and DC are listed in Table 3.

Here, the value of "I/O" represents the total number of inputs and outputs of the logic circuits, "Circuits" represents the names of MCNC benchmark circuits, "startpoint" is the input port name of the critical path, "endpoint" is the output port name of the critical path, "Min" represents the minimum value of equipment timing delay, and "Avg" represents the average value of equipment timing delay. We note from Table 3 that the startpoints and endpoints of all critical paths obtained by DC are equivalent to those obtained by the DPSO algorithm, which demonstrates that the proposed algorithm has the same effect as DC for identifying critical paths.

The consistency of the timing optimization provided by the DC tool is demonstrated by the identical "Min" and "Avg" values obtained for the timing delays of each circuit, indicating that DC obtained equivalent values of $T'_{\max}$ for all 20 trials of each logic circuit. However, we note that the DPSO algorithm generally provides slightly different "Min" and "Avg" values of $T_{\max}$. This finding indicates that both DPSO and DC algorithms have unstable factors for the timing optimization of the MPRM combinational logic circuit.

There are certain differences existing between the two algorithms in "Min" and "Avg" after timing optimization of the MPRM logic circuits. As shown in Figure 6, the average timing delay of the critical path obtained by the DPSO

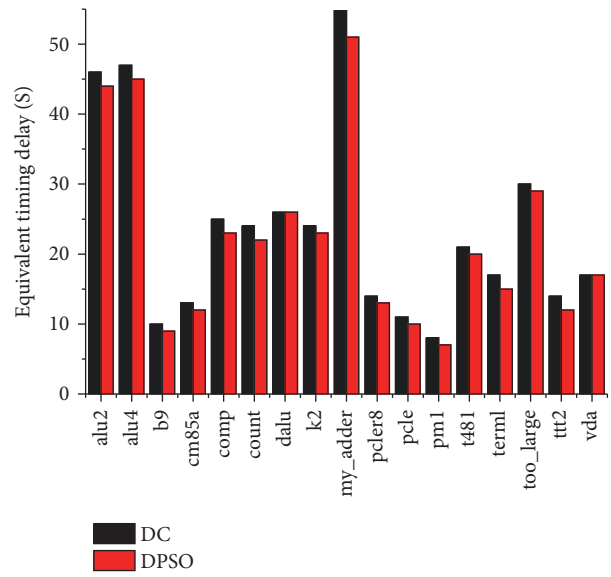

Figure 6: The average timing delay obtained by the DPSO algorithm and DC.

algorithm is less than those obtained by DC for all MCNC benchmark circuits considered. DPSO provides an overall average reduction of 9.7% in the timing delays of critical paths, which sufficiently verifies that the DPSO algorithm has a good timing optimization effect for logic circuits between adjacent flip-flops. This outcome suggests that the intelligence algorithm is better than the DC algorithm in terms of timing optimization of MPRM combinational logic circuits.

5. Conclusion

To verify that an intelligence algorithm is better than the DC algorithm in the timing optimization of MPRM logic circuits, a novel verification method was proposed in this study. First, the DPSO algorithm was used to optimize the timing of the MPRM logic circuits. The equivalent timing delay after timing optimization was calculated. Second, timing constraints and optimization of the MPRM logic circuit were performed using the DC algorithm, and the equivalent timing delay was calculated according to optimization results. Finally, the optimization results of the MCNC benchmark circuits using the two algorithms were analyzed and compared. The proposed verification method is feasible and effective. In this study, MCNC benchmark circuits were used in the verification experiment. The timing optimization results of the two algorithms were converted into the equivalent timing delay. The suggested verification method can determine which algorithm has a better timing optimization ability directly by comparing the experimental data of the two algorithms.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors would like to gratefully acknowledge the financial support of the National Natural Science Foundation of China (Grant no. 61404030).

References

[1] M. Wainberg and V. Betz, "Robust Optimization of Multiple Timing Constraints," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 12, pp. 1942–1953, 2015.

[2] P. Liu, Z. You, J. Kuang, Z. Hu, and W. Wang, "Logic operation-based DFT method and 1R memristive crossbar march-like test algorithm," *IEICE Electronics Express*, vol. 12, no. 23, 2015.

[3] V. V. Shende, A. K. Prasad, I. L. Markov, and J. P. Hayes, "Synthesis of reversible logic circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 6, pp. 710–722, 2003.

[4] J. Cortadella, M. Galceran-Oms, M. Kishinevsky, and S. S. Sapatnekar, "RTL Synthesis: From Logic Synthesis to Automatic Pipelining," *Proceedings of the IEEE*, vol. 103, no. 11, pp. 2061–2075, 2015.

[5] X. Lou, Y. J. Yu, and P. K. Meher, "Fine-grained critical path analysis and optimization for area-time efficient realization of multiple constant multiplications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, no. 3, pp. 863–872, 2015.

[6] S. Abolmaali, N. Mansouri-Ghiasi, M. Kamal, A. Afzali-Kusha, and M. Pedram, "Efficient Critical Path Identification Based on Viability Analysis Method Considering Process Variations," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 9, pp. 2668–2672, 2017.

[7] M. Iizuka, N. Hamada, and H. Saito, "An ASIC design support tool set for non-pipelined asynchronous circuits with bundled-data implementation," *IEICE Transactions on Electronics*, vol. E96-C, no. 4, pp. 482–491, 2013.

[8] J. Núñez, M. J. Avedillo, and J. M. Quintana, "Experimental validation of a two-phase clock scheme for fine-grained pipelined circuits based on monostable to bistable logic elements," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 10, pp. 2238–2242, 2014.

[9] Z.-X. He, L.-M. Xiao, L. Ruan et al., "A power and area optimization approach of mixed polarity Reed-Muller expression for incompletely specified Boolean functions," *Journal of Computer Science and Technology*, vol. 32, no. 2, pp. 297–311, 2017.

[10] W. Sun and J. Hou, "A MPRM-based approach for fault diagnosis against outliers," *Neurocomputing*, vol. 190, pp. 147–154, 2016.

[11] D.-L. Bu and J.-H. Jiang, "Hybrid multi-valued discrete particle swarm optimization algorithm for mixed-polarity reed-muller minimization," *Dianzi Yu Xinxi Xuebao/Journal of Electronics and Information Technology*, vol. 35, no. 2, pp. 361–367, 2013.

[12] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks (ICNN '95)*, vol. 4, pp. 1942–1948, Perth, Western Australia, 1995.

[13] M. Fakhfakh, E. Tlelo-Cuautle, and P. Siarry, "Preface," *Computational Intelligence in Digital and Network Designs and Applications*, pp. 1–245, 2015.

[14] X. Wu, W. Wang, Y. Xu, and J. Yuan, "A new signal injection method with PSO for multi-carrier predistortion," *IEICE Electronics Express*, vol. 10, no. 19, 2013.

[15] H. Shayeghi, M. Mahdavi, and A. Bagheri, "An improved DPSO with mutation based on similarity algorithm for optimization of transmission lines loading," *Energy Conversion and Management*, vol. 51, no. 12, pp. 2715–2723, 2010.

[16] D. E. Goldberg, "Genetic algorithms in search, optimization, and machine learning," *Choice Reviews Online*, vol. 27, no. 02, pp. 27-0936–27-0936, 1989.

[17] V. Roberge, M. Tarbouchi, and G. Labonte, "Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 132–141, 2013.

[18] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[19] M. Dai, D. Tang, A. Giret, M. A. Salido, and W. D. Li, "Energy-efficient scheduling for a flexible flow shop using an improved genetic-simulated annealing algorithm," *Robotics and Computer-Integrated Manufacturing*, vol. 29, no. 5, pp. 418–429, 2013.

[20] J. Chen, W. Zhu, and M. M. Ali, "A hybrid simulated annealing algorithm for nonslicing VLSI floorplanning," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 41, no. 4, pp. 544–553, 2011.

[21] H.-Z. Yu, P.-J. Wang, H.-H. Zhang, and K. Wan, "Optimization of MPRM Circuits Based on Ternary Diversity Particle Swarm Optimization," *Tien Tzu Hsueh Pao/Acta Electronica Sinica*, vol. 45, no. 7, pp. 1601–1607, 2017.

[22] Z. He, L. Xiao, F. Gu et al., "An efficient and fast polarity optimization approach for mixed polarity Reed-Muller logic circuits," *Frontiers of Computer Science*, vol. 11, no. 4, pp. 728–742, 2017.

[23] H. Zhang, P. Wang, and X. Gu, "Area optimization of fixed-polarity Reed-Muller circuits based on niche genetic algorithm," *Journal of Electronics*, vol. 20, no. 1, pp. 27–30, 2011.

[24] X. Wang, Y. Lu, Y. Zhang, Z. Zhao, T. Xia, and L. Xiao, "Power optimization in logic synthesis for mixed polarity reed-muller logic circuits," *The Computer Journal*, vol. 58, no. 6, pp. 1306–1313, 2014.

[25] M. Yang and H. Xu, "Optimization of mixed polarity Reed-Muller expressions based on whole annealing genetic algorithm," in *Proceedings of the 2011 IEEE 9th International Conference on ASIC, ASICON 2011*, pp. 401–404, China, October 2011.