IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received March 4, 2020, accepted March 23, 2020, date of publication April 1, 2020, date of current version April 16, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2984902

Algorithm on Higher-Order Derivative Based on Ternary Optical Computer

KAI SONG[1,3], QINGQING JIN[1], GONG CHEN[1], LIPING YAN[2,3], YI ZHANG[1], AND XIANCHAO WANG[4]

[1]School of Information Engineering, East China Jiaotong University, Nanchang 330013, China
[2]School of Software Engineering, East China Jiaotong University, Nanchang 330013, China
[3]Department of Computer Science and Engineering, SUNY, University at Buffalo, Buffalo, NY 14260, USA
[4]School of Computer and Information Engineering, Fuyang Normal University, Fuyang 236037, China

Corresponding authors: Kai Song (skpark@163.com) and Liping Yan (csyanliping@163.com)

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 61862023 and Grant 61672006, in part by the Natural Science Foundation of Jiangxi Province under Grant 20181BAB202007, and in part by the Science and Technology Project of Jiangxi Provincial Education Department under Grant GJJ190325.

ABSTRACT As an important tool in the field of mathematics, higher-order derivation problems are widely used in differentials, quantum mechanics, and engineering applications. However, in the electronic computer (EC), due to the existence of the carry in the calculation, the computational efficiency is low when solving the higher-order derivation problem. In response to this problem, the ternary optical computer (TOC) has the advantages of no carry-in and the characteristics of numerous data bits, reconfigurable processors and parallel computing. Solve the higher-order derivation problems with complex operations by constructing multipliers and adders on the TOC platform, and copying multiple composite operator units (COUs). This article introduces the design of the higher-order derivative algorithm based on TOC in detail, the reconfiguration process of the multiplier and adder, and the number of bits of the multiplier and adder required in the implementation is given. Besides, the hardware resources and clock cycles in the operation are analyzed. The feasibility of the implementation scheme is verified by experiments. Compared with the traditional higher-order derivative, the higher-order derivative based on the TOC is superior in time performance, computational efficiency, and processing of complex operations. Due to the limitation of the research stage, the algorithm is only applicable to the function of polynomials, which lays a foundation for the further research of higher-order derivative algorithms, and has certain application significance.

INDEX TERMS Ternary optical computer, higher-order derivative, MSD adder, MSD multiplier, composite operator unit.

I. INTRODUCTION

Internet technology has experienced the rise of the late 20th century, and the rapid development at the beginning of this century. The rapid growth of the number of internet users has led to the growth of data volume, and the large-scale data computing problems brought about by this are compelling [1]. The speed and efficiency of higher-order derivation is critical in practical applications, especially in engineering and scientific computing. For the ordinary electronic computer (EC), the process of higher-order derivative addition, subtraction, multiplication and division is realized by the logic operation of hardware. The addition is the basic operation, the logical relationship is "XOR", the same is 0, the difference is 1, the result of "XOR" is the value of the local sum, and then determine whether to perform the carry according to the operation requirements; subtraction is a complement-addition operation; multiplication operations use shift-addition or a large amount of hardware for logic operations; division is performed by shift-subtraction and performing a complement-addition operation. This process is repeated multiple times to complete the update of the higher-order derivative [2]. It can be seen that the EC is inefficient in dealing with the complicated computing requests.

When the functions are complex, the processing time of higher-order derivatives will increase sharply, which cannot meet the needs of users for quick response. Therefore, people began to construct many CPUs as parallel computing

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

platforms to handle such complex operations [3]. However, since the number of CPUs is not proportional to the processing power obtained, when the number of CPUs exceeds a certain value, the processing power of the parallel computing platform grows slowly. In fact, only 6% to 12% of the energy is actually used to respond to user requests for computing. Most of the energy is used to solve communication and I/O problems, and the energy consumption is huge.

The ternary optical computer (TOC) is a photoelectric hybrid computer. It expresses information through no-light state and two polarization states. The liquid crystal array (LCA) is used to control the polarization direction of the light beam, and the polarizer is used to complete the information processing. From the application point of view, the number of pixels in the LCA is large, which makes it have a large number of processor bits; the calculation function of the processor bits can be reconstructed, and the running time can be reconfigured according to user requirements, which is more advantageous than EC [4]. In addition, TOC's second-generation experimental system already has a reconfigurable optical processor with thousands of data, and the number of bits is easily extended, which can efficiently calculate a large amount of complex data. This paper combines the two characteristics of TOC's numerous data bits and the reconfigurable processor. A ternary optical processor with multiple data bits is used to solve the higher-order derivative problem of complex calculations by constructing multipliers, adders, and replicating multiple composite operator units (COUs), and computational efficiency is analyzed.

II. RESEARCH BACKGROUND

At present, the Modified Signed-Digit (MSD) digital system of TOC has matured, and the design of adders and multipliers has also been implemented, which provides a basis for the algorithm research of higher-order derivatives.

A. MSD DIGITAL SYSTEM

In 1960, in order to solve the problem of fault tolerance during operation, Algirdas Avizienis used redundant symbols to limit the carry process to two binary digits, thereby eliminating the carry chain [5]. Later, Drake proposed the MSD digital system and the specific operation method [44]. For the real number A, it can be simply expressed by MSD as the following formula:

$$A = \sum_{i} a_i 2^i \tag{1}$$

Among them, the a_i has three representations of u, 0, 1, corresponding to the vertical polarization state, the no-light state, and the horizontal polarization state in the TOC, respectively, the liquid crystal and the polarizer are used to achieve the transition between these three states, thereby completing the corresponding operations.

B. ONE-STEP MSD ADDER

Based on the operating method of the MSD digital system, MSD addition uses four logical transformations T (calculating carry value), W (calculating local value), T', W', a parallel algorithm that can be completed without carry [39]. However, each operation in the MSD adder requires 3 photoelectric conversions and 3 data feedbacks, and the overall speed of the addition is still not high. The symmetric MSD encoding can eliminate continuous 1 or continuous u in the data. Two conversion methods of T' and W' can be used to easily implement the two-step MSD number without carry addition, and it is easier to achieve parallelization, but there is still an intermediate process in this method [6].

The one-step non-carry MSD adder is based on the above MSD addition principle and the symmetric MSD encoding technique, which reduces the three steps of the general addition to one step, so that the addition of thousands of bits can be completed in one step. The process is as follows: the coded two numbers are padded with 0 at the end, and the results of adding two data can be obtained by using the one-step MSD addition truth table.

C. MSD MULTIPLICATION

The nature of the multiplication is the addition operation, as is the MSD multiplication. It is an improved addition algorithm based on the MSD addition. Enter two MSD numbers A and B to multiply them, and the product result is set to P, which means the following:

$$P = A \times B = \sum_{i=0}^{n-1} A \times b_i \times 2^i = \sum_{i=0}^{n-1} S_i \times 2^i = \sum_{i=0}^{n-1} p^{(i)} \tag{2}$$

Perform the M operation on the i-th bit in multiplier B and each bit of multiplicand A (the logical operation corresponding to the two one-bit MSD multiplication is called the M operation.). And the result is shifted to the left by i bits, which is called the partial product of the multiplication operation. In the formula (2), $S_{\mathrm{i}} = A \times b_i$ is called partial product, $p^{(i)} = S_i \times 2^i$ is called sum term, and finally the result P of the multiplication operation is obtained.

D. MSD MULTIPLIER OF THE MINIMUM MODULE

In multiplication, two multi-digit multiplications can be split into multiple low-bit multiplications, and then the weights in the original data are restored by adding 0 to the lower bits. Based on the theory, the MSD multiplier of the minimum module (MM) is 4-bits, and two multi-digits in the multiplication operation can be separately split, and the split is performed from the low to the high position with 4 bits as a basic unit. After splitting, each basic unit in the multiplier is multiplied with each basic unit of the multiplicand, complement 0 according to the weight of the two basic units in their original data, and then add them to get two multi-digit multiplication results [1], [7].

A two-input multiplication operation is performed in the MSD multiplier of the ternary optical processor based on the MM. The specific steps are as follows:

Step 1: Reconstructing the number of the MSD multipliers of the MM required to satisfy the multiplication operation according to the number of the input two digits;

Step 2: Data calculation. Input the divided multiplier and multiplicand, and pass them to the MSD multiplier of the MM reconstructed by the processor of the TOC for parallel calculation;

Step 3: According to the weights of the two basic units in their original data, perform a zero-padding operation on the obtained results, and then add them to obtain the product of two multi-digit numbers [38].

III. ALGORITHM DESIGN OF HIGHER-ORDER DERIVATION BASED ON TOC

This article analyzes the higher-order derivatives of polynomial functions, and concludes the laws through summary. The algorithm design of the higher-order derivatives is completed on the TOC platform, and then through the analysis of the actual complex operation process, an improved higher-order derivative is proposed.

A. HIGHER-ORDER DERIVATIVES

Let the function $f(x)$ be expressed as:

$$f(x) = a_1x^{b_1} + a_2x^{b_2} + a_3x^{b_3} + \cdots + a_ix^{b_i} + \cdots + a_nx^{b_n} \tag{3}$$

For the convenience of discussion, $b_1, b_2, \cdots, b_i, \cdots b_n$ are algebraic expressions without x, then the first derivative of the function is:

$$f'(x) = a_1b_1x^{b_1-1} + a_2b_2x^{b_2-1} + a_3b_3x^{b_3-1} + \cdots + a_ib_ix^{b_i-1} + \cdots + a_nb_nx^{b_n-1} \tag{4}$$

The second derivative of $f(x)$ is:

$$\begin{aligned} f''(x) = {} & a_1b_1(b_1-1)x^{b_1-2} + a_2b_2(b_2-1)x^{b_2-1} \\ & + a_3b_3(b_3-1)x^{b_3-1} + \cdots + a_ib_i(b_i-1)x^{b_i-2} \\ & + \cdots + a_nb_n(b_n-1)x^{b_n-2} \end{aligned} \tag{5}$$

According to inductive reasoning, we can get the m-order derivative of $f(x)$ as:

$$\begin{aligned} f^m(x) = {} & a_1b_1(b_1-1)\cdots(b_1-m+1)x^{b_1-m} \\ & + a_2b_2(b_2-1)\cdots(b_2-m+1)x^{b_2-m} \\ & + a_3b_3(b_3-1)\cdots(b_3-m+1)x^{b_3-m} + \cdots \\ & + a_ib_i(b_i-1)\cdots(b_i-m+1)x^{b_i-m} + \cdots \\ & + a_nb_n(b_n-1)\cdots(b_n-m+1)x^{b_n-m} \end{aligned} \tag{6}$$

By analyzing the equation (6), it can be concluded that the general formula of the coefficient in the function $f^m(x)$ is set to c, which is expressed as:

$$c = a_ib_i(b_i-1)(b_i-2)\cdots(b_i-m+1) \tag{7}$$

The general formula composed of the power of x is set to d, which is expressed as:

$$d = x^{b_i-m} \tag{8}$$

The function $f^m(x)$ is a polynomial composed of c and d.

From the above discussion, $f(x)$ is the higher-order derivative formula obtained in the ideal case, namely, b_i is not an algebraic form of x and the value is large. However, in general, b_i is finite, so we must first determine b_i when we want to obtain the higher-order derivative $f^m(x)$. When b_i is larger than m, we can obtain the m-order derivative of $f^m(x)$. When b_i is smaller than m, m-order derivative of $f(x)$ is 0. Taking formula (6) as an example, it can be determined that the relationship between c and b_i is represented by $D(m)$ as follows:

$$D(m) = \begin{cases} 0, & m > b_i/2 \\ a_ib_i(b_i-1)(b_i-2)\cdots(b_i-m+1), & m < b_i/2 \end{cases} \tag{9}$$

It can be seen that in $m > b_i/2$, the general formula c of the coefficient in the function $f(x)$ contains $m+1$ different numbers multiplied, and when m is large, the time cost of calculating the coefficient is large, and the coefficient can be up to n terms. In the case where calculating a single coefficient takes a lot of time, the calculation amount of the n coefficients can be imagined. After the calculation of each coefficient is completed, if you need to calculate the value of the specific point of the higher-order derivative, it is also necessary to multiply each coefficient by the power of the point and then add the products of each item, which increase the amount of data, making the calculation amount complicated and the computational efficiency is low. The solution to this problem is described in detail in the following sections.

B. HIGHER-ORDER DERIVATIVE ON THE TOC

TOC is a "three-value" opto-electric hybrid computer. It uses the redundant numeral system with {0, 1, u} digits in radix 2. There are 19,683 logic transformations, which can calculate more than one thousand data in parallel on one optical processor, or multiple programs can be paralleled on one processor, beyond the computing ability of ECs.

Taking $f^m(x)$ as an example, it can be observed that each coefficient c is multiplied by m+1 data, and there are such n coefficients in $f^m(x)$. Since the TOC has a reconfigurable processor, the multiplier can be constructed based on the date contained in c_i in the formula (10).

$$\begin{aligned} f^m(x) = {} & \underbrace{\overbrace{a_1b_1(b_1-1)\cdots(b_1-m+1)}^{m+1 \text{ multipliers}}}_{\text{coefficient}c_1}\underbrace{x^{b_1-m}}_{d_1} \\ & + \underbrace{\overbrace{a_2b_2(b_2-1)\cdots(b_2-m+1)}^{m+1 \text{ multipliers}}}_{\text{coefficient}c_2}\underbrace{x^{b_2-m}}_{d_2} \\ & + \underbrace{a_3b_3(b_3-1)\cdots(b_3-m+1)}_{\text{coefficient}c_3}\underbrace{x^{b_3-m}}_{d_3} + \cdots \\ & + \underbrace{a_ib_i(b_i-1)\cdots(b_i-m+1)}_{\text{coefficient}c_i}\underbrace{x^{b_i-m}}_{d_i} \\ & + \cdots + \underbrace{a_nb_n(b_n-1)\cdots(b_n-m+1)}_{\text{coefficient}c_n}\underbrace{x^{b_n-m}}_{d_n} \end{aligned} \tag{10}$$

The detailed algorithm steps are as follows:

Step 1: Data preprocessing, which is completed by the TOC's task management software, converts the calculated values into MSD binary data. TOC uses MSD binary parallelism, so the internal data of the TOC is MSD binary data.

Step 2: The derived higher-order derivative formula uses the SZG file to transfer the user-submitted operation request to the TOC. The SZG file is a way for users to interact with the TOC's task management software. For the specific content of SZG file, refer to the literature [40]. With the help of the SZG file, the order of the higher-order derivative and the corresponding coefficient can be passed to the TOC.

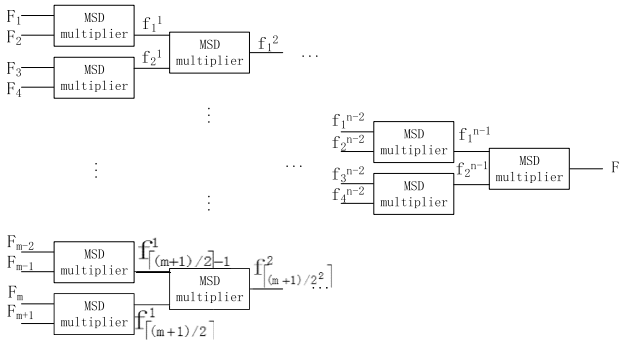


FIGURE 1. The specific reconstruction process of the multiplier.

Step 3: Construct the number of multipliers constituting a single coefficient according to the order m of the higher-order derivative. The construction process of the multiplier is shown in Figure 1, the detailed steps are as follows:

1) When m is an even number, m+1 is an odd number, and the multiplier is converted to an even number by increasing the vertical polarized light u) of the TOC, and $\lceil (m+1)/2 \rceil$ multipliers are constructed ($\lceil\ \rceil$ for upward taking integer);
2) Using the result of the previous step, namely, the total $\lceil (m+1)/2 \rceil f_i$ s as the input of the next column of MSD multipliers, $\lceil (m+1)/4 \rceil$ multipliers are constructed to complete multiplication of $\lceil (m+1)/2 \rceil$ numbers, and $\lceil (m+1)/4 \rceil$ products are obtained after completing the MSD multiplication.
3) Repeat the above steps $t = \lceil \log_2(m+1) \rceil$ times. When the reconstructed MSD multiplier is 1, the reconstruction ends and the operation result is output.

Step 4: The process of reconstructing the multiplier in step 3 is encapsulated into a COU. The number of multipliers and the specific construction process are not recalculated, and the COU is copied from the next data bit according to the number of coefficients c.

C. IMPROVED HIGHER-ORDER DERIVATIVE ALGORITHM BASED ON TOC

In practical applications, there are many complex operations that require high precision and are not suitable for direct operations. The solution is to use Taylor series method to expand the function, generate simple multiplication and addition operations, and then perform higher-order derivative operations, but the efficiency problem is not solved [37]. This section proposes an improved higher-order derivative algorithm for higher-order derivatives in practical applications.

In formula 10, when performing higher-order derivative operations on specific points in $f^m(x)$, the multiplier cannot be constructed by the number of coefficients, but the multiplier should be constructed by the number of exponents. In addition, the adder needs to be constructed according to the number of items n, which is used for the implementation of parallel computing.

Taking $f^m(x)$ as an example, the higher-order derivative of $f^m(a)$ is actually solved. The algorithm steps are as follows:

Steps 1 and 2 are the same as section 3.2.

Step 3: Define the data bit sequence. For more details, please refer to reference [39]. In the actual operation process, according to the user's operation requirements, data bits need to be allocated to prepare for the construction of multipliers and adders below.

Step 4: Construct a multiplier from the raw data. Analyze the number of multipliers for each multiplication operation in $f^m(a)$, and construct the multiplier based on the multiplier contained in the largest exponent b_i, and then avoid the reconstruction time by copying the COU.

Step 5: Analyze the number of addition operations, and construct the adder according to the number of terms n, the detailed steps are as follows:

1) When the number of terms n is an odd number, a zero value is added to make the whole number of terms even. It is necessary to construct $\lceil n/2 \rceil$ adders.
2) Using $\lceil n/2 \rceil$ date in (1) as the input of this layer, construct $n/2^2$ adders so that n_{2i-1} and n_{2i} (i $=$ 1, 2, 3, ... $\lceil$n/2$\rceil$) get $P_i^{(2)}$
3) When k round iterations, $\lceil n/2^k \rceil$ is obtained as an odd number, it is supplemented with a value of 0 to make it an even number $\lceil n/2^k \rceil + 1$, and using this as the next input, which need to construct $\lceil n/2^{k+1} \rceil$ adders so that $P_{2i-1}^{(k)}$ and $P_{2i}^{(k)}$ (i $= 1, 2, 3, \ldots \lceil n/2^k \rceil$) to get $P_i^{(k+1)}$
4) Repeat 3) until iterative $k = \lceil \log_2 n \rceil$ times, the reconstruction ends when the number of reconstructed adders is 1, and finally the result of the higher-order derivative value $f^m(a)$ is obtained.

Step 6: Write step 5 into the form of an SZG file [40], send it to the underlying control software, and integrate it into the reconstruction module of the TOC.

Step 7: Enter multiple pairs of data that need to be calculated. This step is directly completed by the corresponding functional modules in the monitoring system of the TOC, which is not part of the research content of this article.

An improved higher-order derivative calculation routine based on the TOC is shown in Figure 2.

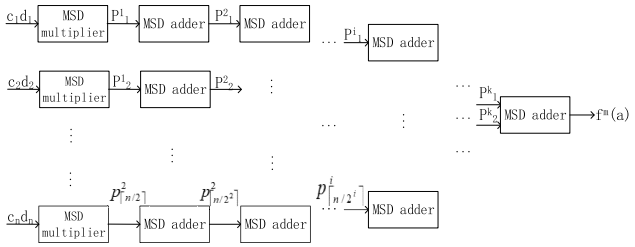


FIGURE 2. Reconstruction process of multiplier and adder.

IV. IMPLEMENTATION AND ANALYSIS OF HIGHER-ORDER DERIVATIVE ALGORITHMS FOR TOC

A. EXAMPLE

Let a function be

$$f(x) = 2019x^{2018} + 2018x^{2017} + 2017x^{2016} + 2016x^{2015} \tag{11}$$

After derivation, the derivative formula of the function is as shown in equation (6). To find the 8-order derivative of the function, the multiplier needs to be reconstructed $\lceil \log_2(8+1) \rceil = 4$ times according to the order. The higher-order derivative $f^8(x)$ contains four different coefficients, which are set to A, B, C, and D, respectively. The construction results are shown in Table 1:

TABLE 1. Multipliers reconstruction process of 8-order derivative function.

Coeffici ent	Number of constructions f				Construct the total number of multipliers
	f_i^1 (i =1,2, …, $\lceil \frac{m+1}{2} \rceil$)	f_i^2 (i= 1,2, …, $\lceil \frac{m+1}{2^2} \rceil$)	f_i^3 (i= 1,2, …, $\lceil \frac{m+1}{2^3} \rceil$)	f_i^4 (i =1,2, …, $\lceil \frac{m+1}{2^4} \rceil$)	
A	5	3	2	1	11
B	5	3	2	1	11
C	5	3	2	1	11
D	5	3	2	1	11

It can be seen from Table 1 that each coefficient needs to construct 11 multipliers, and the required multipliers are $11+11+11+11=44$. The multiplier constructed by a single coefficient can be packaged into a COU. After 4 iterations, it is equivalent to constructing 4 COUs [8].

If $f^8(2)$ is found, it is calculated by the improved TOC higher-order derivative algorithm. $f^8(x)$ contains four different coefficients, which are respectively set to A', B', C', D', and the higher-order derivative is determined by the exponent b_i in $f(x)$. Therefore, it is necessary to analyze it separately. The reconstruction results of the multiplier are shown in Table 2.

From the results of the construction of Table 2, the number of multipliers required to construct A', B', C', D' is very small, and A' contains the largest exponent, which constructs the multiplier to avoid the reconstruction time of each coefficient.

The adder is constructed, and the results obtained by the above multipliers are used as the input to the adder. Construct n=4 adders from the number of terms in $f^8(2)$. It is necessary to construct $\lceil \log_2 n \rceil = 2$ times, the first time $p_i^{(1)}$ (i $= 1, 2, \ldots, \lceil$n/2$\rceil$) needs to construct $\lceil 4/2 \rceil = 2$ adders, the second $p_i^{(2)}$ (i $= 1, 2, \ldots,$ n/2^2) takes the sum of the previous iteration as the input of this layer, and needs to construct an adder to get the final higher-order derivative value.

B. DATA BIT ANALYSIS OF MULTIPLIER AND ADDER

The data bit analysis is performed with the higher-order derivative function $f^m(x)$. The total number of constructing multipliers is determined by the number of coefficients, and the general formula constituting a single coefficient is composed of at most $m+1$ data.

If m is an even number, the first layer needs to construct $\lceil (m+1)/2 \rceil$ multipliers, and the second layer needs to construct $\lceil (m+1)/2^2 \rceil$ multipliers; the i-th layer needs to construct $\lceil (m+1)/2^i \rceil$ multipliers, and a total of $\lceil \log_2(m+1) \rceil$ columns need to be constructed, so for a single coefficient, the multiplier that needs to be constructed is [41]

$$\lceil (m+1)/2 \rceil + \left\lceil (m+1)/2^2 \right\rceil + \cdots + \left\lceil (m+1)/2^i \right\rceil + \cdots + 1 = \sum_{i=1}^{\lceil \log_2(m+1) \rceil} \left\lceil (m+1)/2^i \right\rceil.$$

There are n coefficients in $f^m(x)$, then the total number of constructing multipliers is at most

$$n \sum_{i=1}^{\log_2(m+1)} \left\lceil (m+1)/2^i \right\rceil.$$

If m is an odd number, the total number of multipliers that need to be constructed is at most

$$n \sum_{i=1}^{\log_2(m+1)} (m+1)/2^i.$$

The data bit analysis is performed with the higher-order derivative function $f^m(a)$. The maximum exponent is b_i, when b_i is even, the number of multipliers that need to be constructed in the first layer is $\lceil (b_i+1)/2 \rceil$; the second layer needs to construct $\lceil (b_i+1)/2^2 \rceil$ multipliers [7], and the j-th layer needs to construct $\lceil (b_i+1)/2^j \rceil$ multipliers [41]. A total of $\lceil \log_2(b_i+1) \rceil$ columns need to be constructed, and the number of multipliers corresponding to a single coefficient is

$$\sum_{j=1}^{\log_2(b_i+1)} \left\lceil (b_i+1)/2^j \right\rceil.$$

Although the value of b_i is different, the number of multipliers constructed with the maximum exponent b_i, the total number of multipliers that need to be constructed is at most

$$n \sum_{j=1}^{\log_2(b_i+1)} \left\lceil (b_i+1)/2^j \right\rceil.$$

TABLE 2. Number of reconstructed multipliers.

Coefficient	Multipliers $b_i + 1$	Total number of constructions f	Number of constructions f										
			f_i^1	f_i^2	f_i^3	f_i^4	f_i^5	f_i^6	f_i^7	f_i^8	f_i^9	f_i^{10}	f_i^{11}
A'	2018+1=2019	$\lceil \log_2 2019 \rceil = 11$	1010	505	253	127	64	32	16	8	4	2	1
B'	2017+1=2018	$\lceil \log_2 2018 \rceil = 11$	1009	505	253	127	64	32	16	8	4	2	1
C'	2016+1=2017	$\lceil \log_2 2017 \rceil = 11$	1009	505	253	127	64	32	16	8	4	2	1
D'	2015+1=2016	$\lceil \log_2 2016 \rceil = 11$	1008	504	252	126	63	32	16	8	4	2	1

The number of items of $f^m(a)$ is n, When n is an odd number, the first layer constructs $\lceil n/2 \rceil$ adders, the second layer is $\lceil n/2^2 \rceil$ adders, and the i-th layer needs to construct the adder to be $\lceil n/2^i \rceil$, which construct the $\lceil \log_2 n \rceil$ columns. Then the number of adders constructed is at most

$$\sum_{i=1}^{\lceil \log_2 n \rceil} \left\lceil n/2^i \right\rceil.$$

When n is an even number, the number of the constructed adders does not need to be rounded up. To calculate the higher-order derivative function $f^m(a)$, which needs to construct multipliers and adders, the total number is up to

$$\sum_{j=1}^{\lceil \log_2(b_i+1) \rceil} \left\lceil (b_i+1)/2^j \right\rceil + \sum_{i=1}^{\log_2 n} n/2^i.$$

C. ANALYSIS OF THE NUMBER OF LIQUID CRYSTALS

When performing higher-order derivation operations, $f^m(x)$ can be expressed as

$$f^m(x) = c_1 d_1 + c_2 d_2 + \cdots + c_i d_i + \cdots + c_n d_n = \sum_{i=1}^{n} c_i d_i. \quad (12)$$

In the formula 12, $c_1 d_1, c_2 d_2, \ldots, c_n d_n$ can be calculated in parallel, so the calculation time of the function $f^m(x)$ is the calculation time of one real multiplication, and the constructed multiplier does not slow down as the number of bits increases. Compared with EC, TOC has many advantages when dealing with extremely large numbers of data. At the same time, TOC has a large number of bits and is easy to expand [9].

When calculating $f^m(a)$, the multiplication operation in $\sum_{i=1}^{n} c_i d_i$ can be calculated in parallel, but there are n-1 addition operations, so the calculation time of one $\sum_{i=1}^{n} c_i d_i$ is the calculation time of one real multiplication and $n-1$ real addition operations.

The number of multipliers and adders required is analyzed from section IV-B. For the higher-order derivation operation of d-bits in $f^m(a)$, the adder and multiplier are constructed to perform continuous multiply-and-accumulate operations, and the number of liquid crystals is analyzed.

Analyze the number of liquid crystals of a single multiplier. A single multiplier is determined by the number of liquid crystals of the M converter. The number of liquid crystals of a single M converter is d, and the total number of M converter required is

$$n \sum_{j=1}^{\lceil \log_2(b_i+1) \rceil} \left\lceil (b_i+1)/2^j \right\rceil.$$

Then, the total number of liquid crystals of the M converter is at most

$$d \times n \sum_{j=1}^{\lceil \log_2(b_i+1) \rceil} \left\lceil (b_i+1)/2^j \right\rceil.$$

After M transform, the number of data bits input to the MSD adder of the first layer becomes $2d$-1, when sent to each adder, the number of upper layer bits is increased by 2 bits, and the number of data bits p_i input by the MSD adder in the i-th layer is $2d - 1 + 2(i-1)$ [33]. Therefore, the number of MSD adders in different layers is different from each other. Since the one-step MSD adder is used, there is only one conversion in one MSD adding device, the number of bits is p_i, and the required number of liquid crystals is p_i, that is, $2d+2i-3$, the function $f^m(a)$ requires the number of liquid crystals of the MSD adder to be $2d + 2\lceil \log_2 n \rceil - 3$. Therefore, the total number of liquid crystals required when calculating $f^m(a)$ is up to

$$C = d \times n \sum_{i=1}^{\lceil \log_2(b_i+1) \rceil} \left\lceil (b_i+1)/2^j \right\rceil + 2d + 2\lceil \log_2 n \rceil - 3.$$

According to the same analysis method, it can be obtained that the total number of liquid crystals required for $f^m(x)$ is

$$C = d \times n \sum_{i=1}^{\lceil \log_2(m+1) \rceil} \left\lceil (m+1)/2^i \right\rceil.$$

D. CLOCK CYCLE ANALYSIS OF HIGHER-ORDER DERIVATIVE ON TOC

The higher-order derivative $f^m(x)$ is calculated by constructing a multiplier, and the clock cycle is determined by the M converter. The M converter requires one clock cycle. Since each coefficient needs to construct $\lceil \log_2(m+1) \rceil$-layer multipliers, the coefficients are calculated in parallel, and the total clock cycle is T= $\lceil \log_2(m+1) \rceil$. The number of cycles required for the implementation of the higher-order derivative function $f^m(x)$ is only related to the order m. When the order m =1023, the required number of cycles is 10, which effectively solves the problem of higher-order derivatives.

The cycle required for the function $f^m(a)$ is determined by the M converter of the multiplier and the one-step MSD adder. Since the multiplier needs to construct $\lceil \log_2(b_i + 1) \rceil$- layers in total, the required number of cycles is $\lceil \log_2(b_i + 1) \rceil$; the one-step MSD adder requires one clock cycle per layer, the $\lceil \log_2 n \rceil$-layer adders require $\lceil \log_2 n \rceil$ clock cycles, and the total number of clock cycles is $T = \lceil \log_2(b_i + 1) \rceil + \lceil \log_2 n \rceil$ [34]. The number of cycles required to calculate $f^m(a)$ is determined by the order m and the number of terms n. For complex functions, the number of terms $n = 128$, the power of the unknown number x is 1023, and each multiplier $d = 16$ bits [18], when the order of $f^m(a)$ is $m = 1023$, the number of cycles required is $10 + 7 = 17$. It has nothing to do with a, so a is even large, and it will not affect the calculation efficiency.

E. RECONFIGURABLE ANALYSIS OF TOC

The higher-order derivative algorithm is mainly based on the reconfigurability of the TOC. The higher-order derivative operation is realized by reconstructing multipliers and adders. How the constructed number is used for the allocation of data bits of the processor is the key. We hope that during the implementation of the higher-order derivative algorithm, the resources of the system can be used as efficiently as possible. From section IV-D, we know that the total period of the multiplier and adder reconstructed by the improved higher-order derivative algorithm is $\mathrm{r} = \lceil \log_2(b_i + 1) \rceil + \lceil \log_2 n \rceil$, and Ny calculations can be completed between the two reconstructions. The total number of data bits is n, and the effective number of data bits is m'. The effective use ratio B of the data bits is

$$B = m' \times Ny/((Ny + \mathrm{r}) \times n) \tag{13}$$

Among them, $m' \times$ Ny is the calculation amount completed by the optical processor after this reconstruction [43]. It can be seen from formula (13) that when Ny is much larger than r, the utilization rate will not be restricted, which indicates the higher-order derivative algorithms are more suitable for tasks with more complex functions and a large amount of calculations.

When Ny is much larger than *r,* the utilization rate is determined by the value of m' / n. The TOC has many data bits, however, in actual higher-order derivative operations, it is rarely possible to perform direct operations by reconstructing multipliers and adders, so at this time m' is very important. It can connect the calculations of different users to make the maximum use of the number of data bits, which also guarantees the reconstruction of multipliers and adders for higher-order derivative algorithms.

V. EXPERIMENT

Aiming at the above-mentioned higher-order derivative algorithm of the TOC, this section will design experiments to verify its feasibility.

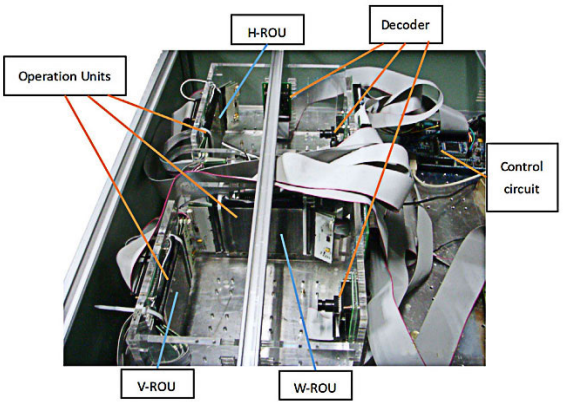


FIGURE 3. Experimental equipment of TOC.

A. EXPERIMENTAL EQUIPMENT

In this paper, the experimental equipment is shown in Figure 3. The processor used is the double-rotator-structure ternary optical processor (DRSTOP), the operation unit of DRSTOP is composed of three functional modules: encoder, operator and decoder. The encoder includes two liquid crystal devices (LCD) and a vertical polarizer. The decoder is composed of a beam splitter, a polarizer and a photoelectric converter [14]. It is responsible for the acquisition and analysis of the LCD output signal. The results of the calculation are obtained by analyzing the image information on the DCS. More detailed information on the experimental equipment of TOC can be found in the literature [2], [3].

The core optical element of TOC is mainly composed of LCD and polarizer. The LCD has 576 pixels arranged in a 24 × 24 array. Each pixel can be individually controlled. The parallel LCA has two interfaces CN1 and CN2, and each interface controls two NT7701 chips, as shown in Figure 4 (a) [16]. In the experiment, the TOC is coded according to the light intensity. When the light intensity reaches the set threshold (bright), the H or V light state is output, and the set threshold (dark) is not reached, the W light state is output. In this study, two adjacent pixels in each row represent one data bit of the optical processor, so there are 576/2 = 288 pixels, and the optical processor bits also become 288, as shown in Figure 4 (b). According to the number of bits of the optical processor, each data bit corresponds to two LCDs, which are called left LCD and right LCD, respectively. If the left LCU is on, the output value is V state; if the right LCU is on, the output value is H state; if the left and right LCUs are not lit, the output is W state; if the left and right LCUs are on, the output value is illegal [17].

B. EXPERIMNETAL SIMULATION

The experiment consists of two parts: EC simulation and TOC processing. The task of the EC is to adjust and encode the user's input data, and the software simulates the experimental operation process. The TOC is composed of the device that

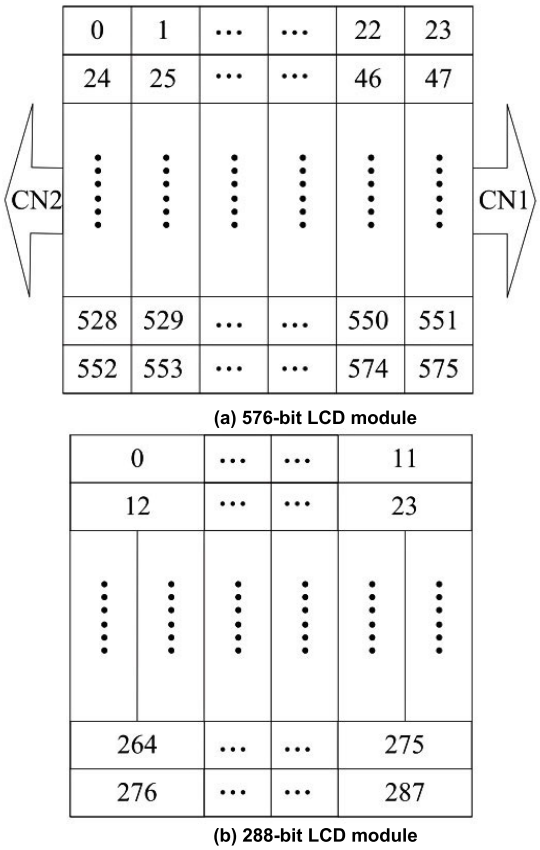


FIGURE 4. LCD module.

can rotate the polarization direction of the light. It can assemble multiple basic arithmetic modules together. It is the core of the experiment and is responsible for the specific operation of the numerical value [13].

1) EXPERIMENTAL PROCESS OF EC

The application-oriented TOC is still under construction, so the experiments of higher-order derivative algorithms need to be completed in conjunction with EC, and the experimental process is shown in Figure 5 (a). The steps are as follows:

1) firstly, the EC is required to process the function input by the user, and using the software to find the higher-order derivative formula of the input function;
2) Use the SZG file (as shown in Figure 6), which is the window between the TOC and the EC, input the order of the higher-order derivative and generate the result file;
3) Preprocess the result file and input it to the TOC for calculation;
4) Determine the number of functions. If there are multiple functions to be processed, return to step (1) and continue to execute the above experimental process;
5) Until the number of functions to be processed is 0, the experiment ends.

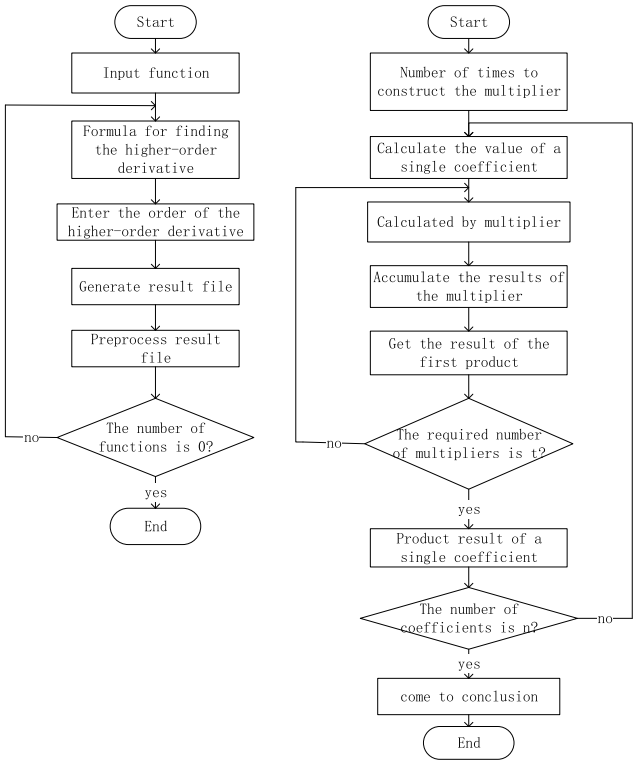


FIGURE 5. The left picture is figure 5(a), which shows the experimental flow chart of the EC, and the right picture is figure 5(b), which is the experimental flow chart of the TOC.

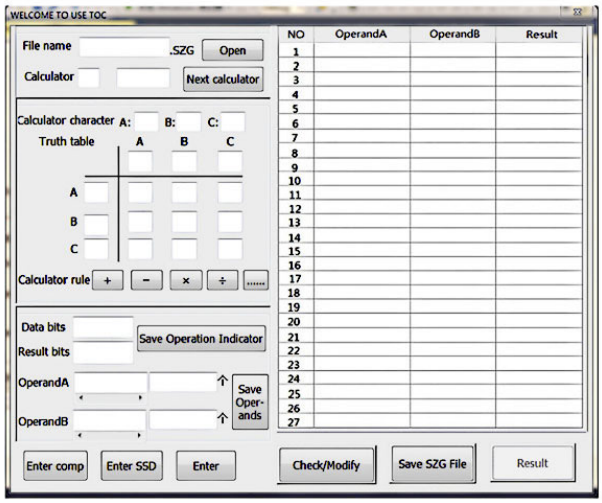


FIGURE 6. SZG file.

2) EXPERIMENTA PROCESS OF TOC

Due to the limitation of the number of liquid crystal layers, the realization of multi-digit multiplication is not easy. The MM of the MSD multiplier of the TOC [1] and the one-step MSD adder were used to verify the higher-order derivative algorithm and its implementation. The experimental flow chart is shown in Figure 5 (b). The detailed experimental process is as follows:

TABLE 3. Construction of the f^1 layer computation channel.

CC1	CC1	CC1	CC1	CC1	CC1	CC1	CC1	CC1	CC1	CC1	CC1
CC1	CC1	CC1	CC1	CC2	CC2	CC2	CC2	CC2	CC2	CC2	CC2
CC2	CC2	CC2	CC2	CC2	CC2	CC2	CC2	CC3	CC3	CC3	CC3
CC3	CC3	CC3	CC3	CC3	CC3	CC3	CC3	CC3	CC3	CC3	CC3
CC4	CC4	CC4	CC4	CC4	CC4	CC4	CC4	CC4	CC4	CC4	CC4
CC4	CC4	CC4	CC4	CC5	CC5	CC5	CC5	CC5	CC5	CC5	CC5
CC5	CC5	CC5	CC5	CC5	CC5	CC5	CC5	CC6	CC6	CC6	CC6
CC6	CC6	CC6	CC6	CC6	CC6	CC6	CC6	CC6	CC6	CC6	CC6
CC7	CC7	CC7	CC7	CC7	CC7	CC7	CC7	CC7	CC7	CC7	CC7
CC7	CC7	CC7	CC7	CC8	CC8	CC8	CC8	CC8	CC8	CC8	CC8
CC8	CC8	CC8	CC8	CC8	CC8	CC8	CC8	CC9	CC9	CC9	CC9
CC9	CC9	CC9	CC9	CC9	CC9	CC9	CC9	CC9	CC9	CC9	CC9
CC10	CC10	CC10	CC10	CC10	CC10	CC10	CC10	CC10	CC10	CC10	CC10
CC10	CC10	CC10	CC10	CC11	CC11	CC11	CC11	CC11	CC11	CC11	CC11
CC11	CC11	CC11	CC11	CC11	CC11	CC11	CC11	CC12	CC12	CC12	CC12
CC12	CC12	CC12	CC12	CC12	CC12	CC12	CC12	CC12	CC12	CC12	CC12
CC13	CC13	CC13	CC13	CC13	CC13	CC13	CC13	CC13	CC13	CC13	CC13
CC13	CC13	CC13	CC13	CC14	CC14	CC14	CC14	CC14	CC14	CC14	CC14
CC14	CC14	CC14	CC14	CC14	CC14	CC14	CC14	CC15	CC15	CC15	CC15
CC15	CC15	CC15	CC15	CC15	CC15	CC15	CC15	CC15	CC15	CC15	CC15
CC16	CC16	CC16	CC16	CC16	CC16	CC16	CC16	CC16	CC16	CC16	CC16
CC16	CC16	CC16	CC16	CC17	CC17	CC17	CC17	CC17	CC17	CC17	CC17
CC17	CC17	CC17	CC17	CC17	CC17	CC17	CC17	CC18	CC18	CC18	CC18
CC18	CC18	CC18	CC18	CC18	CC18	CC18	CC18	CC18	CC18	CC18	CC18

1) Define raw data and input functions. Read a set of raw data from the SZG file, and during the experiment, randomly generate multiple functions for higher-order differentiation;
2) Define variables. Define the relevant variables of the randomly generated function. Define the number of terms of the function as n, the order of the higher-order derivative m, and the number of times required to construct the multiplier is t. This step is the internal communication process of the monitoring software of the TOC, which is implemented by the monitoring software itself, but does not interfere in the implementation;
3) Determine t according to the m, and perform reconstruction on TOC according to the reconstruction scheme of the multiplier.
4) Call the underlying control module and perform calculations through the reconstructed MSD multiplier of the MM;
5) Construct the MSD Adder () function to simulate a one-step MSD adder. This function calls the truth table of one-step MSD addition in turn to operate on each element of the operand, accumulates the values obtained by the MM, and obtains the result of the first reconstruction of the multiplier;
6) Loop iteration. Judging t, if the number of executions is less than t, return to step (4), otherwise continue to the next step;
7) The calculation process of the single coefficient ends;
8) Loop iteration. Judge the number of coefficients required. If it is less than n, return to step (3) to continue execution, otherwise the calculation ends.

C. EXPERIMENTAL TESTS AND RESULTS

This experiment performed higher-order derivative operations on randomly generated polynomial functions. Because this experiment is aimed at more complex functions and the experimental process is more complicated, it is impossible to test the experimental routines in an exhaustive manner. We first show in detail the test of the special function in section IV-A. The number of terms of the known function is $n = 4$, when $m = 3$, construct the multiplier with a single coefficient c_2, and get $t = 3$, and the data entered are as follows:

$$\begin{aligned} a_2 &= 2018_{10} = 1000, 00u0, 01u0_{\text{MSD}}, \\ b_2 &= 2017_{10} = 1000, 00u0, 001u_{\text{MSD}}, \\ b_2 - 1 &= 2016_{10} = 1000, 00u0, 0000_{\text{MSD}}, \\ b_2 - 2 &= 2015_{10} = 1000, 00u0, 000u_{\text{MSD}}, \end{aligned}$$

they are all 12 bits. Since the MM of the MSD multiplier is 4 bits and four sets of operation data need to be set at the same time, the first layer needs to construct 3*3*2 = 18 multipliers for calculation, and the calculation amount of each multiplier is $4 \times 4 = 16$, and the width of the composite operation channel is 18*16 = 288. Therefore, 18 sub-computation channels with a width of 16 are required, respectively CC1, CC2, CC3, CC4, CC5, CC6, CC7, CC8, CC9, CC10, CC11, CC12, CC13, CC14, CC15, CC16, CC17, CC18, the results of the channel construction are shown in Table 3 [18], [21], [22].

Since the reconstructed MSD multiplier is implemented in full parallel, the results of the corresponding 18 MSD multipliers can be output simultaneously. The optical state table corresponding to the outputs are shown in Table 4,

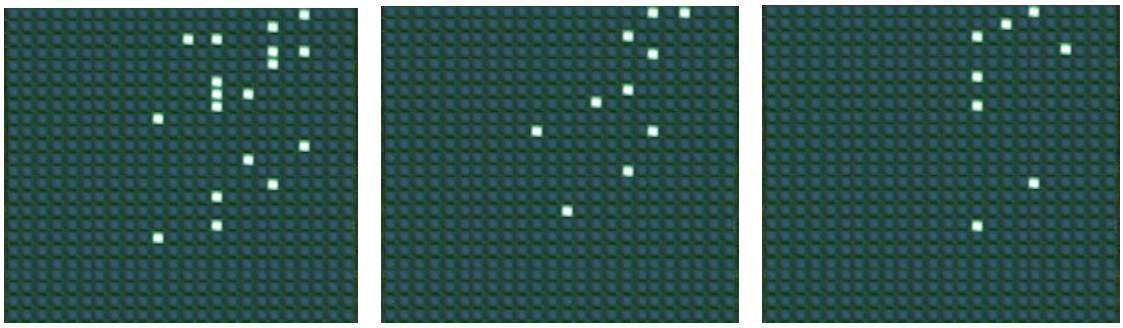


FIGURE 7. The LCD display of the first layer, there are three figures corresponding to the output of the W-ROU, H-ROU, and V-ROU. The following figures have the same meaning as this.

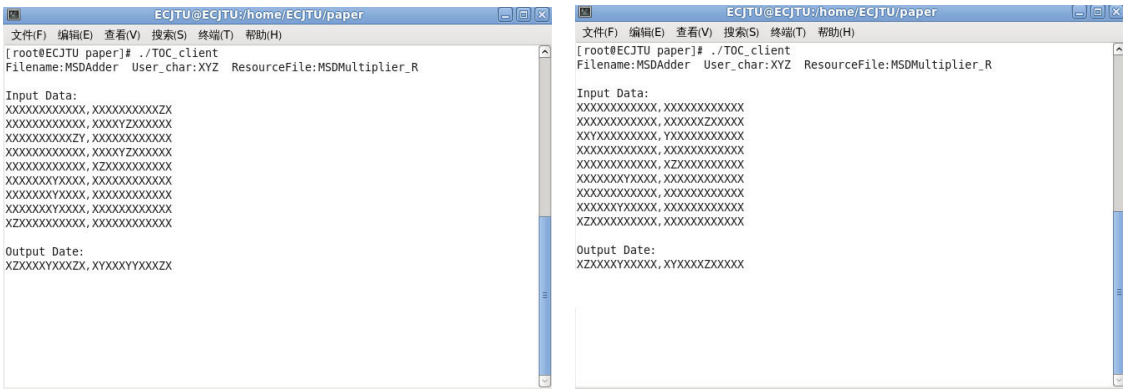


FIGURE 8. The outputs of the software of the first layer. The left picture shows result of a_2b_2, the right picture shows result of $(b_2-1)(b_2-2)$.

TABLE 4. Performance comparison of adders.

	Output data	Optical state
1	XXXXXXXXXXZX	WWWWWWWWWWHW
2	XXXXXXXXYZXX	WWWWWWWWWVHWW
3	XXXXXXZYXXXX	WWWWWWHVWWWW
4	XXXXXXXXXYZX	WWWWWWWWWVHW
5	XXXXXXXXXZXX	WWWWWWWWWHWW
6	XXXXXXXYXXXX	WWWWWWWVWWWW
7	XXXXXXXZYXXX	WWWWWWWHVWWW
8	XXXXXXXYXXXX	WWWWWWWVWWWW
9	XXXXXZXXXXX	WWWWWHWWWWW
10	XXXXXXXXXXX	WWWWWWWWWWW
11	XXXXXXXXXXZX	WWWWWWWWWWHW
12	XXXXXXXXYXXX	WWWWWWWWVWWW
13	XXXXXXXXXXX	WWWWWWWWWWW
14	XXXXXXXXXZXX	WWWWWWWWWHWW
15	XXXXXXXYXXXX	WWWWWWWVWWWW
16	XXXXXXXXXXX	WWWWWWWWWWW
17	XXXXXXXYXXXX	WWWWWWWVWWWW
18	XXXXXZXXXXX	WWWWWHWWWWW

in this experiment, X, Y, and Z correspond to W, V, and H, respectively [19].

Two-pixel bits can realize one data bit, when decoding by the TOC decoder, it is necessary to assign the output result to different row operator unit (ROU) according to the input state of a. When the input of a is W state, the corresponding output will be assigned to the W row operator units (W-ROUs). When the input of a is H state, the corresponding output will be assigned to the H row operator units (H-ROUs). When the input of a is V state, the corresponding output will be assigned to the V row operator units (V-ROUs) [23]–[25], each row corresponds to the output of one multiplier, and the 19-24 line LCDs have no output and shows no-light state, the experimental results are shown in Figure 7 [35].

For the output results, a zero-compensation operation is performed. Then use the obtained result file as input and send it to the simulated one-step adder to get the results of a_2b_2 and $(b_2-1)(b_2-2)$ in the first layer f^1. The results are shown in Figure 8.

The two products obtained in the first layer f^1 are sent to the multiplier as the input of the second layer f^2. According to the same calculation process described above, the 36 multipliers required for the reconstruction of the number of bits, and the results are displayed on the LCD as shown in Figures 9 and 10.

The one-step adder is simulated in the software, and finally the result of the single coefficient is shown in Figure 11.

According to the calculation method of the single coefficient c_2, the results of calculating c_1, c_3, and c_4 are shown in Table 5. The experimental results are compared with the theoretical results, and the calculated values are the same.

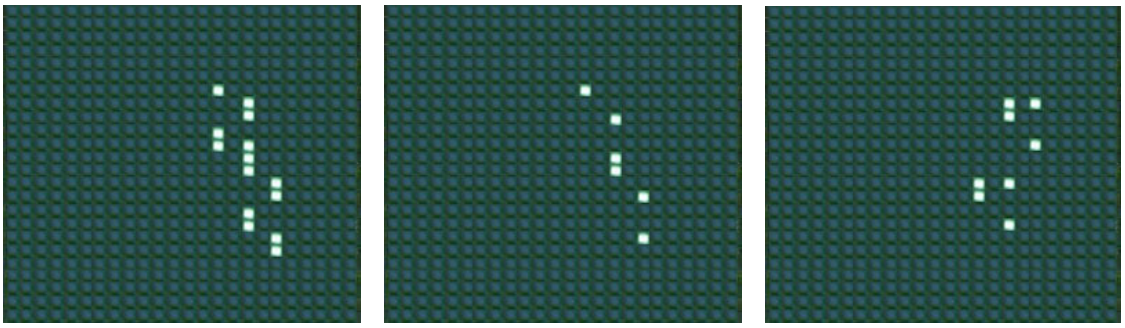


FIGURE 9. LCD display of the ROU of 1-24 MSD multipliers in the second layer.

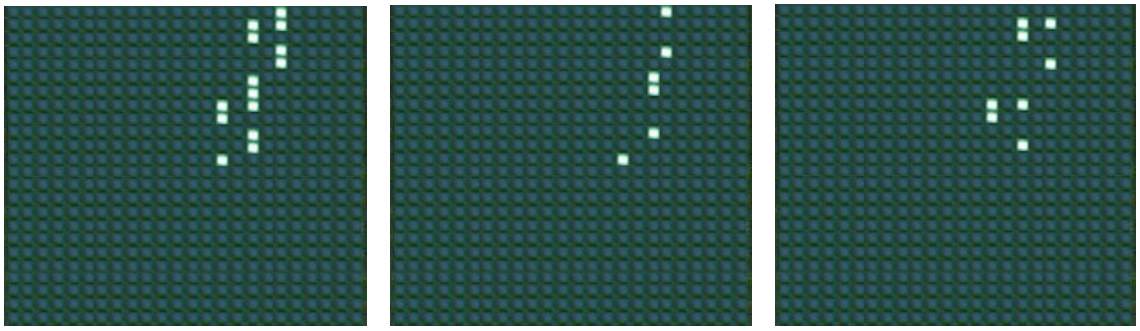


FIGURE 10. LCD display of the ROU of 25-36 MSD multipliers in the second layer.

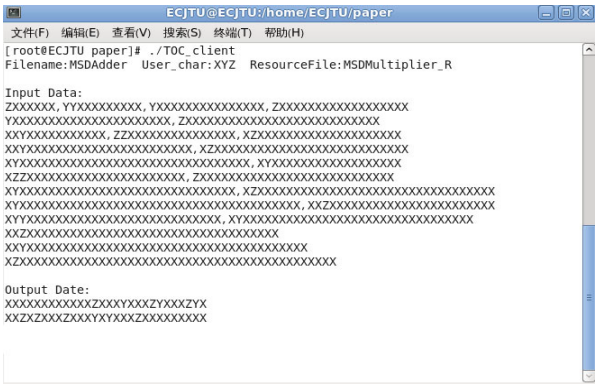


FIGURE 11. Output results of the second layer.

TABLE 5. Experimental results.

	MSD binary system	EC system	Test Results
c_1	100u100010010u0u0010100 0u000u0u0u0u00uu000000	16567382793024	correct
c_2	1000u0001u0011100000010 1u0001000u0u0001000000	16534559845440	correct
c_3	1000u0000001000010000u00 11000000u00uuuu0010011	16501785693120	correct
c_4	100u10000u1u01000000010 000u0100000u0011000000	16469060287680	correct

For the experiment of the improved higher-order derivative algorithm, it is also necessary to construct the adder according to the number of terms of the function, and use the result obtained by the reconstructed multiplier as the input of the adder. The experimental example is shown in Table 6.

In order to ensure the rigor of the functional verification of the higher-order derivative algorithm, according to the above experimental test method, we performed 216 special functions and 10,000 random functions with different orders of higher-order derivative operations. And we obtained 10 216 simulated experimental results that are basically consistent with theoretical values, but due to space limitations, we cannot show them one by one. This experiment can verify the correctness and feasibility of the operation of the higher-order derivative algorithm for polynomial functions by constructing multipliers and adders on the TOC.

VI. ALGORITHM ANALYSIS

A. EFFICIENCY ANALYSIS

On the TOC platform, the algorithm uses the MM of the MSD multiplier (see section V-B) to multiply the data. Compared with the ordinary MSD multiplier, the required operating cycle of the optical processor is shown in Table 7.

In the higher-order derivative algorithm, the multiplier is first constructed to perform operations. The period of the multiplier determines the efficiency of the higher-order derivative algorithm. It can be clearly seen from Table 7 that when the MSD multiplier with the MM is used to perform multiple

TABLE 6. Input and output results of the reconstructed adder.

Input order	TOC input	First layer adder output	Second layer adder output	Theoretical value	result
1	100u100010010u0u0010100 0u000u0u0u0u00uu000000	000000000001000u0000 10100u00001000001u1 u010u0100u00000000	01000u000001000000 100100u0u1u101u10u u0010u000000	66040011466 944	correct
2	1000u0001u0011100000010 1u0001000u0u0001000000				
3	1000u0000001000l0000u00 11000000u00uuuu0010011	000000000001000u0000 00u01u0100000u00u01 u0010000100u000000			
4	100u10000u1u01000000010 000u0100000u0011000000				

TABLE 7. Comparison of operating cycles required for algorithm implementation.

Number of data bits	Required number of operating cycles	
	Reconstructed multiplier for higher-order derivation based on TOC	Existing multiplier
4 bits × 4bits	3	7
8 bits × 2 bits	4	3
8 bits × 4 bits	4	7
8 bits × 8 bits	5	13
16 bits × 2 bits	5	3
16 bits × 4 bits	5	7
16 bits × 8 bits	6	13
16 bits × 16 bits	7	23
40 bits × 2 bits	7	3
40 bits × 4 bits	7	7
40 bits × 8 bits	8	13
40 bits × 16 bits	9	23
40 bits × 40 bits	10	53

data bit operations, the period is greatly reduced, especially when calculating complex functions. As the number of data bits increases, the operation efficiency becomes higher.

In the improved higher-order derivative algorithm, it is also necessary to construct an adder for operation, so the calculation efficiency of the adder also affects the efficiency of the higher-order derivative algorithm. This paper uses the one-step adder. Compared to the general MSD addition, the 3 photoelectric conversions and 3 data feedback steps are reduced to one step, so that thousands of additions can be completed in one step in parallel. As can be seen from IV-D, the number of liquid crystals required for the one-step MSD adder during the calculation process is $2d + 2\lceil \log_2 n \rceil - 3$. For a general MSD adding device, there are five transforms of T, W, T', W', and T, and the number of bits is p_i, p_i+1, p_i+1, p_i+2, and the number of liquid crystals required is $5p_i+4$, that is $10d + 10\lceil \log_2 n \rceil - 11$. The one-step MSD adder requires one clock cycle per layer, the $\lceil \log_2 n \rceil$-layer adders require $\lceil \log_2 n \rceil$ clock cycles, and the general MSD adder requires three clock cycles per layer. The comparison result of the calculation efficiency of the adder is shown in Table 8 [32].

TABLE 8. Performance comparison of adders.

Number of data bits d , Number of function terms n	Number of cycles		Number of LCDs	
	One-step MSD adder	MSD adder	One-step MSD adder	MSD adder
d=2 , n=4	2	6	5	29
d=4 , n=8	3	9	11	59
d=8 , n=16	4	12	21	109
d=16 , n=32	5	15	39	199
d=32 , n=64	6	18	73	369
d=64 , n=128	7	21	139	699
d=128, n=256	8	24	269	1349

It can be seen from Table 8 that the efficiency of the higher-order derivative algorithm in addition operation is much lower than the operation of the general MSD adder in terms of resource consumption, and the calculation cycle is relatively short, especially when the function becomes more complex, and the number of items and the number of data bits of the function increase gradually, the operation efficiency becomes more obvious.

B. COMPARISON OF COMPUTATIONAL EFFICIENCY WITH THE EC

Compare with the same type of EC to implement higher-order differentiation. The EC's shift-add multiplier needs to complete d binary additions when multiplying d-bit data, but each addition will cause a carry delay, and the delay time

is the function of t_e, so the time required to complete an addition is the time function t_e of the data bit d and each bit delay. In the serial carry adder, the total delay time for one addition is $d \times t_e$, and the total time for completing d times is $T_e \approx d^2 \times t_e$ [31].

In this study, when the TOC performs d-bit multiplication, the MSD addition is up to $2d$ times, and the time required for each MSD addition is not affected by the number of bits d, which is only related to the optical device. If the delay time of the optical instrument is t_0, even without using the pipeline, the three-valued logic transformation is at most $6d$ and the total time is at most $T_0 \approx 6d \times t_0$ [12].

Assume the number of data bits $d = 100$, the computing time of the EC is $T_e \approx 10000t_e$, and the time on TOC is $T_0 \approx 600t_0$, and the delay time t_0 of the optical device is much smaller than the delay time t_e of the EC. When $t_0 = 1$ns and $t_e = 0.1$ns, the relationship between the time of the higher-order derivative operation with errors and the value performed by the operation is shown in the Figure 12, where T_0 represents the operation time of TOC and T_e represents the operation time of the EC.

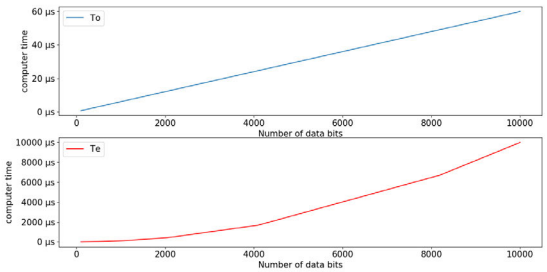


FIGURE 12. The relationship between the number of bits for processing higher-order derivative operations and the running time. The blue line represents T_0 and the red line represents T_e. The following figures have the same meaning as this.

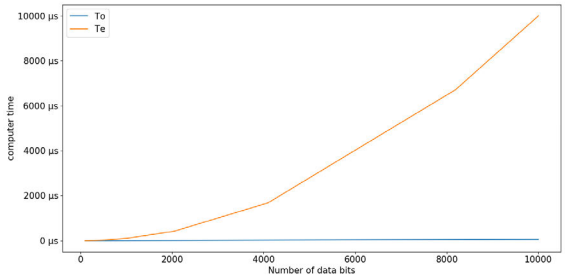


FIGURE 13. Comparison of the efficiency of ECs and TOCs in processing higher-order derivatives.

As can be seen from Figure 12, with the increase of the number of execution bits, higher-order derivatives are realized on TOC, and the time complexity increases linearly, while the computing speed of EC increases by two orders of magnitude compared with TOC. After comparison, it is found that the time complexity of the EC in the operation is $O(n^2)$, and the time complexity of the TOC is O(n). Comparing the higher-order derivative of TOC and EC with different digits in Figure 13, it can be seen that the TOC has a great advantage in processing multiple data bits.

C. AIGORITHM COMPARISON

The higher-order derivative proposed in this paper is based on the TOC platform, which has the characteristics of numerous data bits and great potential for parallel computing. This algorithm takes advantage of the reconfigurability of the optical processor and operates by reconstructing multipliers and adders. The differences between this algorithm and the traditional higher-order derivative method are as follows:

1) For higher-order derivatives in actual complex operations, although the accuracy has been improved, the cost of the calculation has not changed at all [39], [45]. This algorithm uses the redundant number system of TOC [5]. There is no carry delay in the calculation process, which improves the efficiency and also improves the calculation accuracy. In addition, the algorithm can allocate data bits according to the user's calculation requirements. By replicating the COU, the time of reconstruction is avoided, and the cost of calculation is saved.
2) The performance of the algorithm is different due to different basic platforms. The higher-order derivative algorithm in this paper uses the MSD multiplier of the MM and one-step MSD adder to operate on the data. As the number of data bits increases, the period is greatly shortened and the operation is more efficient (see section 6.1). Compared with the general higher-order derivative algorithm [46], the time complexity is O(n). The advantages of this algorithm are more obvious when calculating complex functions.
3) It is undeniable that this algorithm increases the resource consumption to a certain extent when reconstructing the multiplier and adder. However, through the detailed analysis of the required hardware resources (see section IV), it can be seen that the algorithm proposed in this paper greatly shortens the clock cycle, and also controls the amount of required hardware resources within the acceptable range of the TOC platform. It greatly improves the calculation efficiency, which is difficult to achieve compared to the general higher-order derivative method [47].

VII. CONCLUSION

The reconstruction scheme of the multipliers and adders proposed on the TOC can well solve the problem of low computational efficiency of complex operations such as higher-order derivation in ECs, and the experiment proves that the scheme is feasible [36]. The advantages of TOC's numerous data digits, reconfigurable processors, and parallel computing are fully utilized. It provides a new method for solving complex computing problems and is more practical. Since this paper deals with higher-order derivatives, the premise requires that the function has a derivative formula. The function is complex and the amount of calculation is large, and the algorithm is only applicable to polynomials, not any arbitrary function, so there are certain limitations. This is also the problem that

TOC needs to consider and solve in the research of complex operations in the next stage.

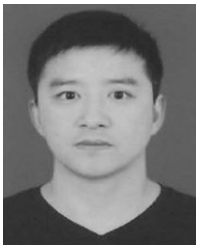
ACKNOWLEDGMENT

The authors would like to express their sincere gratefulness to the TOC Team, School of Computer Engineering and Science, Shanghai University, for providing the optical platform and giving many inspired ideas to the paper.

REFERENCES

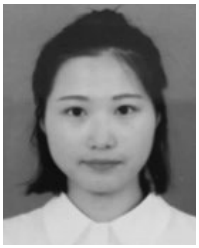
[1] S. Kai, C. Gong, J. Qingqing, Y. Liping, and Z. Yi, "Design of MSD multiplier for ternary optical computer processor based on minimum module," *Opt. Commun.*, vol. 448, pp. 33–42, Oct. 2019.
[2] J. B. Jiang, X. L. Chen, and S. O. Yang, "Hardware implementation of converting ternary optical computer MSD into standard binary data," *J. Nanjing Univ. Sci. Technol.*, vol. 40, no. 3, pp. 278–284, Jun. 2016.
[3] C. Ye, S. Kong, Y. Fu, and J. Peng, "Optical computer based application platform for MSD multiplication," *Opt. Commun.*, vol. 458, Mar. 2020, Art. no. 124814, doi: 10.1016/j.optcom.2019.124814.
[4] M. Kumm, O. Gustafsson, M. Garrido, and P. Zipf, "Optimal single constant multiplication using ternary adders," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 65, no. 7, pp. 928–932, Jul. 2018.
[5] A. P. He, G. B. Feng, and J. L. Zhang, "An asynchronous based booth multiplication," *IET Circuits Dev. Syst.*, vol. 13, no. 1, pp. 73–78, 2018.
[6] W. Jin, Z. H. Wang, and Y. J. Liu, "Ternary optical computer," *J. Nature*, vol. 41, no. 03, pp. 207–218, Mar. 2019.
[7] P. Junjie, F. Youyi, Z. Xiaofeng, K. Shuai, and W. Xinyu, "Implementation of DFT application on ternary optical computer," *Opt. Commun.*, vol. 410, pp. 424–430, Mar. 2018.
[8] Y. Jin, H. Wang, S. Ouyang, Y. Zhou, Y. Shen, J. Peng, and X. Liu, "Principles, structures, and implementation of reconfigurable ternary optical processors," *Sci. China Inf. Sci.*, vol. 54, no. 11, pp. 2236–2246, Nov. 2011.
[9] Y. Jin, "Ternary optical computer principle," *Sci. China F*, vol. 46, no. 2, pp. 145–150, Apr. 2003.
[10] S. Zhang, J. Peng, Y. Shen, and X. Wang, "Programming model and implementation mechanism for ternary optical computer," *Opt. Commun.*, vol. 428, pp. 26–34, Dec. 2018.
[11] Y.-T. Fang, Y.-C. Zhang, and J. Xia, "Reversible unidirectional reflection and absorption of PT-symmetry structure under electro-optical modulation," *Opt. Commun.*, vol. 416, pp. 25–31, Jun. 2018.
[12] S. Li, J. Jiang, Z. Wang, and H. Zhang, "Basic theory and key technology of programming platform of ternary optical computer," *Optik*, vol. 178, pp. 327–336, Feb. 2019.
[13] J. Chase and D. Niyato, "Joint optimization of resource provisioning in cloud computing," *IEEE Trans. Services Comput.*, vol. 10, no. 3, pp. 396–409, May 2017.
[14] X. C. Wang, S. Zhang, M. Zhang, J. Zhao, and X. Niu, "Performance analysis of a ternary optical computer based on M/M/1 queueing system," in *Proc. Int. Conf. Algorithms Archit. Parallel Process.*, vol. 10393. Cham, Switzerland: Springer, 2017, pp. 331–344.
[15] S. Kai and Y. Liping, "Control mechanism of double-rotator-structure ternary optical computer," *Opt. Commun.*, vol. 387, pp. 338–349, Mar. 2017.
[16] Q. Xu, X. Wang, and C. Xu, "Design and implementation of the modified signed digit multiplication routine on a ternary optical computer," *Appl. Opt.*, vol. 56, no. 16, pp. 4661–4669, Jun. 2017.
[17] H. Gao, Y. Jin, and K. Song, "Extension of C language in ternary optical computer," *Shanghai Univ., Nat. Sci.*, vol. 19, no. 3, pp. 280–285, Jun. 2013.
[18] W. Li, S. Ouyang, Y. Jin, Y. Han, and Q. Xu, "Structured data computer—Application characteristics of a ternary optical computer," *Scientia Sinica Informationis*, vol. 46, no. 3, pp. 311–324, Mar. 2016.
[19] P. Aneela, B. Raheela, and D. M. Tayab, "Analysis of booth multiplier based conventional and short word length FIR filter," *Mehran Univ. Res. J. Eng. Technol.*, vol. 37, no. 3, pp. 595–602, 2018.
[20] S. Kai and Y. LiPing, "The symmetric MSD encoder for one-step adder of ternary optical computer," *Opt. Commun.*, vol. 372, pp. 221–228, Aug. 2016.
[21] M. Martinelli, P. Martelli, and A. Fasiello, "A universal compensator for polarization changes induced by non-reciprocal circular birefringence on a retracing beam," *Opt. Commun.*, vol. 366, pp. 119–121, May 2016.
[22] H. Zhang, J. Zhou, and S. Zhang, "Design and implementation of positive and negative discriminator of MSD data for ternary optical processor," *J. Comput. Res. Develop.*, vol. 54, no. 6, pp. 1391–1404, Jun. 2017.
[23] Y. Zhao, C.-X. Shi, K.-C. Kwon, Y.-L. Piao, M.-L. Piao, and N. Kim, "Fast calculation method of computer-generated hologram using a depth camera with point cloud gridding," *Opt. Commun.*, vol. 411, pp. 166–169, Mar. 2018.
[24] M. V. Dolgopolik, "Existence of augmented Lagrange multipliers: Reduction to exact penalty functions and localization principle," *Math. Program.*, vol. 166, nos. 1–2, pp. 297–326, Nov. 2017.
[25] S. Mazahir, O. Hasan, R. Hafiz, and M. Shafique, "Probabilistic error analysis of approximate recursive multipliers," *IEEE Trans. Comput.*, vol. 66, no. 11, pp. 1982–1990, Nov. 2017.
[26] M. D. Jiménez-Gamero and J. C. Pardo-Fernández, "Empirical characteristic function tests for GARCH innovation distribution using multipliers," *J. Stat. Comput. Simul.*, vol. 87, no. 10, pp. 2069–2093, Jul. 2017.
[27] J. Wang, "Critical factors for personal cloud storage adoption in China," *J. Data Inf. Sci.*, vol. 1, no. 2, pp. 60–74, May 2016.
[28] R. Rumipamba-Zambrano, J. Perelló, J. M. Gené, and S. Spadaro, "On the scalability of dynamic flex-Grid/SDM optical core networks," *Comput. Netw.*, vol. 142, pp. 208–222, Sep. 2018.
[29] C.-C. Liu, Y.-T. Chang, C.-C. Yang, and J.-F. Huang, "Packets buffering enhancement with hybrid coding labels for virtual optical memory," *Procedia Comput. Sci.*, vol. 130, pp. 336–343, Jun. 2018.
[30] D. D. Zhdanov, V. A. Galaktionov, A. G. Voloboy, A. D. Zhdanov, A. A. Garbul', I. S. Potemin, and V. G. Sokolov, "Photorealistic rendering of images formed by augmented reality optical systems," *Program. Comput. Softw.*, vol. 44, no. 4, pp. 213–224, Jul. 2018.
[31] J. Lian, S. Hou, X. Sui, F. Xu, and Y. Zheng, "Deblurring retinal optical coherence tomography via a convolutional neural network with anisotropic and double convolution layer," *IET Comput. Vis.*, vol. 12, no. 6, pp. 900–907, Sep. 2018.
[32] S. Venkatachalam and S.-B. Ko, "Design of power and area efficient approximate multipliers," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 5, pp. 1782–1786, May 2017.
[33] M. Osta, A. Ibrahim, L. Seminara, H. Chible, and M. Valle, "Low power approximate multipliers for energy efficient data processing," *J. Low Power Electron.*, vol. 14, no. 1, pp. 110–117, Mar. 2018.
[34] A. Kumar, S. Kumar, and S. K. Raghuwanshi, "Implementation of full-adder and full-subtractor based on electro-optic effect in Mach–Zehnder interferometers," *Opt. Commun.*, vol. 324, pp. 93–107, Aug. 2014.
[35] S. Kai and Y. LiPing, "Reconfigurable ternary optical processor based on row operation unit," *Opt. Commun.*, vol. 350, pp. 6–12, Sep. 2015.
[36] S. Kai, O. Shan, and J. Yi, "Using row operation unit to realize reconfigurable ternary optical processor," in *Proc. IEEE 12th Int. Conf. Comput. Inf. Technol.*, Oct. 2012, pp. 999–1003.
[37] J. Li, J. Li, X. Chen, C. Jia, and W. Lou, "Identity-based encryption with outsourced revocation in cloud computing," *IEEE Trans. Comput.*, vol. 64, no. 2, pp. 425–437, Feb. 2015.
[38] A. Mansour, R. Mesleh, and M. Abaza, "New challenges in wireless & free space optical communications," *Opt. Lasers Eng.*, vol. 89, pp. 95–108, Feb. 2017.
[39] H. M. Oubei, E. Zedini, R. T. ElAfandy, A. Kammoun, M. Abdallah, T. K. Ng, M. Hamdi, M.-S. Alouini, and B. S. Ooi, "Simple statistical channel model for weak temperature-induced turbulence in underwater wireless optical communication systems," *Opt. Lett.*, vol. 42, no. 13, pp. 2455–2458, Jul. 2017.
[40] S. Li, "The initial SZG file generation software for ternary optical computer," *J. Shanghai Univ. Natural Sci.*, vol. 24, no. 02, pp. 181–191, Apr. 2018.
[41] J. Peng, S. Kong, and C. Ye, "A carry-free multiplication implementation method," *IEEE Access*, vol. 7, pp. 85848–85854, 2019.
[42] Z. Ying, Z. Wang, Z. Zhao, S. Dhar, D. Z. Pan, R. Soref, and R. T. Chen, "Silicon microdisk-based full adders for optical computing," *Opt. Lett.*, vol. 43, no. 5, pp. 983–986, Mar. 2018.
[43] Y. Jin and K. Song, "Management of many data bits in ternary optical computers," *Sci. Sinica Inf.*, vol. 43, pp. 361–373, Mar. 2013.
[44] M. Li, "Multi-digit MSD multiplication algorithm on ternary optical computer and operation analysis," *Technol. Univ.*, vol. 35, no. 12, pp. 1020–1025, 2015.
[45] R. Mokhtari and F. Mostajeran, "A high order formula to approximate the Caputo fractional derivative," *Commun. Appl. Math. Comput.*, vol. 2, no. 1, pp. 1–29, Mar. 2020. 10.1007/s42967-019-00023-y.

[46] W. Q. Luo, L. Y. Wang, and Y. S. Xia, "Implementation of algorithms for solving high-order Boolean E partial derivatives of logic functions," *J. Zhejiang Univ., Sci. Ed.*, vol. 45, no. 4, pp. 420–426, 2018.
[47] M. Ramezani, R. Mokhtari, and G. Haase, "Some high order formulae for approximating Caputo fractional derivatives," *Appl. Numer. Math.*, vol. 153, pp. 300–318, Jul. 2020.



KAI SONG received the B.S. and M.S. degrees in computer application technology from East China Jiaotong University, Nanchang, China, in 2002 and 2007, respectively, and the Ph.D. degree in computer application technology from Shanghai University, Shanghai, China, in 2014.

Since 2014, he has been an Associate Professor with the School of Information Engineering, East China Jiaotong University. His research interests include ternary optical computers, parallel computing, and embedded systems.

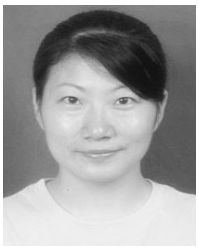


QINGQING JIN received the B.S. degree in Internet of Things engineering from Huainan Normal University, China, in 2018. She is currently pursuing the M.S. degree in computer application technology with East China Jiaotong University, Nanchang, China.

Her research interests include ternary optical computers, embedded systems, and artificial intelligence.

GONG CHEN received the B.S. degree in Internet of Things engineering from East China Jiaotong University, Nanchang, China, in 2017, where he is currently pursuing the M.S. degree in computer application technology.

His research interests include ternary optical computers and embedded systems.



LIPING YAN received the B.S. and M.S. degrees in computer application technology from East China Jiaotong University, Nanchang, China, in 2001 and 2007, respectively, and the Ph.D. degree in computer application technology from Wuhan University, Wuhan, China, in 2018.

Since 2013, she has been an Associate Professor with the School of Software Engineering, East China Jiaotong University. Her research interests include ternary optical computers, intelligent transportation systems, and uncertainty artificial intelligence.

YI ZHANG received the B.S. degree in computer networks engineering from Taiyuan University, Taiyuan, China, in 2018. She is currently pursuing the M.S. degree in computer technology with East China Jiaotong University, Nanchang, China.

Her research interests include ternary optical computers, machine learning, and artificial intelligence.

XIANCHAO WANG received the B.S. degree in applied mathematics from Fuyang Normal University, Fuyang, China, in 1997, the M.S. degree in computer application technology from Northeastern University, Shenyang, China, in 2004, and the Ph.D. degree in computer application technology from Shanghai University, Shanghai, China, in 2011.

He is currently a Professor with Fuyang Normal University. He has coauthored a book, about 30 conference publications, and journal articles. He holds nine patents. His research interests include queuing modeling and theory, computer software, optical computing, complex networks, and big data. He is both a member of the ACM and a Senior Member of the China Computer Federation.