

Received December 16, 2021, accepted January 6, 2022, date of publication January 12, 2022, date of current version January 18, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3142329

Theoretical Basis and Implementation Mechanism of the Programming Platform for Ternary Optical Computer

SHUANG LI^{1,2}, ZHEHE WANG^{1,3}, SHUXIN WANG⁴, AND DONGDONG AN^{1,2}

¹College of Information, Mechanical and Electrical Engineering, Shanghai Normal University, Shanghai 200234, China

²Shanghai Engineering Research Center of Intelligent Education and Bigdata, Shanghai Normal University, Shanghai 200234, China

³College of Computer Science and Technology, Hainan Tropical Ocean University, Sanya 572022, China

⁴School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China

Corresponding author: Shuang Li (lishuang@shnu.edu.cn)

This work was supported in part by the National Science Foundation of China under Grant 61866006, Grant 61775139, Grant 61772164, and Grant 62072126; in part by the Shanghai Sailing Program under Grant 21YF1415100 and Grant 21YF1432900; and in part by the University-Level General Research Project of Shanghai Normal University under Grant SK202121.

ABSTRACT This paper presents a method for building a programming platform for ternary optical computer (TOC). Firstly, the reasons why the existing programming platform can not be directly applied to the ternary optical computer are analyzed. Then, the theoretical basis and core technologies for building the programming platform of the ternary optical computer are given, including: building a model—simple structured data type computer to express the application characteristics of the ternary optical computer, building an operation-data file containing data and computational requirements, expanding the operation-data file transfer instructions. According to our proposed theory, the implementation mechanism of the TOC programming platform is constructed. Finally, the effectiveness of the programming platform is verified by experiments. The programming platform simplifies the application of the TOC and bridges the gap between the user and the TOC.


INDEX TERMS Ternary optical computer, programming platform, operation-data file, ternary optical processor.

I. INTRODUCTION

As electronic computers have matured since their inception, scientists have been exploring and developing other types of computers. In particular, considering that the physical characteristics of electrons will limit the development of electronic computer hardware, various non-electronic computers have emerged, such as optical computers [1], [2], quantum computers [3], [4], biological computers [5], etc. Research on ternary optical computers is a unique branch of non-traditional computer research, with France, India, and Iraq currently engaged in many aspects of international research on ternary optical computers. The second and third generation experimental systems of the ternary optical computer, ShangDa11 (for short SD11, means:Shanghai University 2011), developed by the Optical Computer Laboratory of Shanghai University, both have reconfigurable ternary logical

optical processors with thousand-bit data widths and easily scalable bit counts. The completed experiments fully confirm this theory of large number of computer data bits and processor reconfigurability. The first stably operating ternary optical computer - SD16 (Shanghai University 2016), has 192 ternary logic bits in one basic arithmetic module and can construct up to a 36-bit three-step parallel binary adder or a 64-bit one-step parallel binary adder.

Ternary optical computers are named for their processors that use three-state optical signals to represent information and are capable of performing all ternary logic operations [6], [7]. The biggest difference between this processor in application and conventional electronic processors is that it allows the user to reconfigure the computational functions of individual data bits at any time, and a huge number of data bits can be used independently in any group [8]–[11]. In addition, the memory of a ternary optical computer has a data bus width that varies with the total number of processed data bits on the processor-facing side, but the effect of this

The associate editor coordinating the review of this manuscript and approving it for publication was Md Selim Habib .

feature is not seen at the application level [12]. These dictate that current programming environments built on conventional processors cannot produce applications that take advantage of the characteristics of ternary optical processors. Since the functions of the I/O components and other parts of the ternary optical computer are no different from those of the conventional computer, which is the background device for building the new computer, the ternary optical computer follows exactly the conventional device in these aspects and contains a front-end PC for peripheral management, communication management, task management, and human-computer dialogue. Therefore, the programming platform of the ternary optical computer should enable the programmer to accomplish the application of the new features of the ternary optical processor, but also be fully compatible with the traditional programming environment. Therefore, the basic idea of building a ternary optical computer programming platform is to add to the existing programming environment ways for programmers to use the new features of the ternary optical processor. The main difference between the programming platform of the ternary optical computer constructed in this paper and the traditional computer programming platform is that the programmer no longer uses a single instruction to arrange the working order of each part of the processor and the timing and direction of data transmission to match it. Instead, the programmer only requests the calculation and the number of data bits required, and gives all the raw data to the ternary optical computer management program, which controls and allocates the processor's data bit resources and calculates the raw data. This paper introduces the basic theory and implementation mechanism of this new programming platform, and gives application examples to prepare conceptually, theoretically and technically for the application stage of the ternary optical computer.

This paper has the following key contributions:

a. It presents the theoretical basis and core technologies for building a programming platform for ternary optical computers.

b. It constructs the operation-data files containing data and computational requirements, and expands on high-level programming languages.

c. It constructs the implementation mechanism of the ternary optical computer programming platform based on these proposed theories.

The remainder of this paper is as follows. In section II, we introduce the ternary optical processor architecture and the MSD adder. Section III presents the theoretical basis of the ternary optical computer programming platform., including: building a model—simple structured data type computer to express the application characteristics of the ternary optical computer, building an operation-data file containing data and computational requirements, expanding the operation-data file transfer instructions. Section IV constructs a ternary optical computer programming platform based on the theory and techniques proposed in the first half of the paper. Section V verifies the effectiveness of the new programming

platform through experiments. Finally, Section VI draws conclusion to this research paper.

II. RELATED WORK

A. TERNARY OPTICAL PROCESSOR

Light has three stable physical states: dark light state, horizontally polarized light bright state, vertically polarized light bright state, ternary optical computer with these three stable physical light states to represent the information, that is, the source of the “three values”. Due to the non-interference of light rays, multiple rays can cross each other in the same space without interfering with each other, so many signal rays can be arranged together without isolation to form a flat image. Generally, it takes a few pixels on a flat image to make up a single bit of optical information [13], but optical array devices have many pixels, thus creating the characteristic of a large number of bits of optical computer data. The current experimental system of ternary optical processors constructed with liquid crystal dot matrix screens can have 16384 bits of data, which far exceeds the number of data bits of electronic computers [9], [14]–[16].

The “reconfigurable processor hardware” feature is derived from the construction of the ternary optical logic processor—Decrease-radix design principle [17]. The Decrease-radix design principle states that if the n physical states that represent information contain D states, then $n(n \times n)$ two-input n -value operators without input (borrowing) can be selected from $n \times n \times (n - 1)$ basic base operation units in a fixed step with no more than $n \times (n - 1)$ combinations. where the D -state is the physical superposition with any physical state λ resulting in a physical state that is still a λ -state. The basic operation units is an n -valued logical operator where only one combination of input data makes its output non- D -state and all other combinations of input data make its output D -state [7], [18]. Applying the theory to the ternary optical computer, no more than six of the 18 basic operation units can be combined to form any of the $3^9 = 19,683$ ternary logical operators. Figure 1 shows the optical structure of the ternary optical operation unit. In this structure, the input signal a is input to the main optical path, which consists of two polarizers (P1 and P2) sandwiching a liquid crystal pixel (LC), and the other input signal b is input to the control optical path [19]. There are three types of control optical paths, two semi-reflectors $f1$ and $f2$ and a total reflector F for the input optical signal b beam splitting and the upper split is a vertical polarizer V . Only when b is horizontally polarized light, the photocell $g2$ outputs a high level. There is no polarizer in the lower splitter, as long as b is bright, either horizontally polarized light or vertically polarized light, the photocell $g3$ will output high level. Based on the values of reconfiguration instructions $k2$ and $k3$, the device S will select one of the output signals of $g1$, $g2$ and $g3$ and send the selected signal to the XOR gate Y . When the $k1$ bit of the reconfiguration instruction is 1, Y takes the signal from the inverse S output, and when $k1$ is 0, it does not take the inverse.

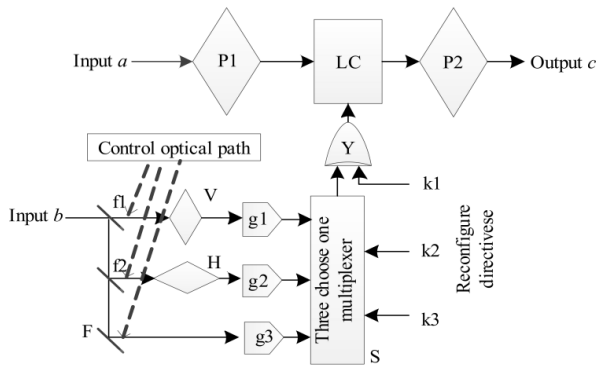


FIGURE 1. Optical structure of the basic computing unit of the ternary optical computer.

The output signal of Y controls the spin characteristics of the liquid crystal LC in the main optical path [20]. According to the polarization direction of polarizers P1 and P2, the main optical path is divided into four types: P1 is a vertical polarizer while P2 is a horizontal polarizer called VH type, P1 is a horizontal polarizer while P2 is a vertical polarizer called HV type, P1 and P2 are both vertical polarizers called VV type, and P1 and P2 are both horizontal polarizers called HH type.

B. MSD ADDER

The MSD parallel adder is based on a redundant representation of binary values in three symbols. Using this redundant representation of values inside the computer can speed up numerical computation by eliminating the feed delay of the adder. In 1961, Avizienis proposed MSD (Modified Signed Digit) digital expression, an addition technique capable of eliminating the rounding delay [21]. Any decimal number A can be represented by an MSD number as follows:

$$A = \sum_i k_i 2^i \tag{1}$$

Among them, the value range of $k_i \in \{u, 0, 1\}$ and i is an integer, the symbol u stands for -1. MSD is also a binary counting. So there can be multiple MSD representations for a decimal number. In the TOC, the three states of light (horizontally polarized light, vertically polarized light and no light) are used to represent 1, 0 or -1. The transition between these three states is realized by the liquid crystal and polarizer to complete the corresponding operation.

MSD addition uses four logical operations, called T, W, T' and W', as shown in Table 1 (a and b are the input data, and T, W, T', W' and M are the output data after the corresponding truth table conversion). An MSD addition operation is completed in 3 steps as follows:

Step 1. T, W operations are performed on input data a and b simultaneously, and the result of T is shifted one place to the left.

Step 2. Perform both T' and W' operations on the result of the first step, and shift the result of T' one place to the left.

TABLE 1. T, W, T', W' and M transformations used by MSD adders.

a	b	T	W	T'	W'	M
$\bar{1}$	$\bar{1}$	$\bar{1}$	0	$\bar{1}$	0	1
$\bar{1}$	0	$\bar{1}$	1	0	$\bar{1}$	0
$\bar{1}$	1	0	0	0	0	$\bar{1}$
0	$\bar{1}$	$\bar{1}$	1	0	$\bar{1}$	0
0	0	0	0	0	0	0
0	1	1	$\bar{1}$	0	1	0
1	$\bar{1}$	0	0	0	0	$\bar{1}$
1	0	1	$\bar{1}$	0	1	0
1	1	1	0	1	0	1

Step 3. The result of the second step is T, and the result is the sum of a and b .

III. THEORETICAL BASIS OF TERNARY OPTICAL COMPUTER PROGRAMMING PLATFORM

A. SIMPLE STRUCTURED DATA TYPE COMPUTER

The number of data bits in a ternary optical computer is on the order of thousands, and the long delay caused by the rounding process in a serial rounding adder is undoubtedly a waste of resources, so it is necessary to construct a full adder with no rounding operation. Ordinary electronic computers have only two values: 0 and 1, so electronic computers cannot use the simple structure and parallel operation of the MSD adder. There are three bases for constructing a ternary optical MSD adder: (1) The ternary optical computer has three stable physical states. (2) The per-bit reconfigurability of the ternary optical logic processor. (3) The large number of data bits of the ternary optical processor. These three foundations dictate that the use of binary MSD adders for ternary optical computers becomes inevitable. The MSD adder extends the ternary optical computer from a logic operator to an optical adder, and the processor can be reconfigured into a composite operator containing a three-valued logic operator, a two-valued logic operator, an adder, and a subtractor.

If a processor is capable of directly computing structured data types, this processor must have many data bits and the data bits must be groupable for independent use to cope with the number of data bits for each component of the structured quantity. At the same time, the computational functions of the individual data bits of this processor must be reconfigurable at runtime as required by the programmer to cope with the different computational requirements of each component of the structural data type. The ternary optical processor satisfies these conditions, and the reconfigured composite processor can compute all components of a simple structural data type simultaneously when executing a single instruction. Therefore, the application characteristics of ternary optical computers are summarized as "simple structural data type computers". Among them, the structural data type is a composite data type customized by the programmer for the

characteristics of the application problem, which contains multiple components, each with different data bits and calculation rules.

The simple model of “simple structural data type computer” is used to characterize the application of the ternary optical computer, which is easy for programmers to accept, understand and apply this new type of computer. More details about structural data type computers can be found in the literature [22].

B. OPERATION-DATA FILE

Ternary optical computer takes several clock cycles to reconfigure the processor, if only a small amount of data is computed after reconfiguring the processor, the time spent reconfiguring the processor is comparable to the time spent computing the data. If a large amount of data is computed after reconfiguring the processor once, the time to reconfigure the processor is negligible.

The most intuitive representation of optical processor utilization is the percentage of data bits effectively used per unit of time on average. It consists of the ratio of the number of data bits effectively used in one operation to the total number of data bits and the ratio of the processor reconstruction time to the total operating time. Let the operator reconfiguration operation takes q computation cycles, N_y computations are completed between reconfigurations, the total number of data bits is n , and the number of efficiently used data bits is m' . Then, the effective use ratio B of data bits is:

$$B = \frac{m' \times N_y}{(N_y + q) \times n} \tag{2}$$

where $m' \times N_y$ is the amount of computation done by the optical processor after one reconstruction. The product of the channel width m of the task and the number of data N is called the computational volume of the task. From equation (2), the utilization rate is constrained by q when N_y is not much larger than q . This indicates that tasks with very little computation are not suitable for running on a ternary optical computer.

Therefore, the ternary optical computer is more suitable for processing the same operation for large amounts of data, which requires the user to send the reconfiguration requirements for the processor and a large amount of initial data to the ternary optical processor at the same time when using the ternary optical computer. In addition, a single computing task rarely uses a processor of thousands of bits, and it is often necessary to share the processor among several computing tasks. Therefore, the calculation rules and raw data are sent to the CPU at program runtime and are not applicable to ternary optical processors. To adapt to the characteristics of the ternary optical processor, it is proposed to organize the computational tasks and raw data into a dedicated file—the operation-data file, also known as *.SZG file (* is a generic character for the file name), and then send the SZG file to the optical processor in the application program in due time to calculate.

Indication section	File mark	SZG file version
		File name
		User's address
		Label amount
	Label 0	Calculation rule 0
		Data bit's number 0
		Data amount 0
		First data address 0
	::	...
	Label n	Calculation rule n
		Data bit's number n
		Data amount n
First data address n		
Data section	Original data 0 or results 0	
	...	
	Original data n or results n	

FIGURE 2. SZG file format.

SZG files are the only way for users to exchange information with ternary optical computers when processing large volumes of data [23], [24]. To ensure that the ternary optical processor is informed of the user's computational intent accurately, the programmer must maintain a consistent format both when constructing the SZG file and when the ternary optical computer parses the SZG file. As shown in Figure 2, the SZG file consists of two parts: the indication section and the data section. The indicator section records the general information and necessary parameters of this SZG file, which is an important basis for processor reconstruction and data bit allocation by the ternary optical processor, and the data section contains all the raw data in the SZG file. We have developed a helper software to help users generate SZG files accurately. This auxiliary software provides a user-friendly input interface, as shown in Figure 3. The user only needs to input the required arithmetic requirements and raw data on this interface, and the software will automatically generate the corresponding SZG files according to the information input by the user and save them to the specified path.

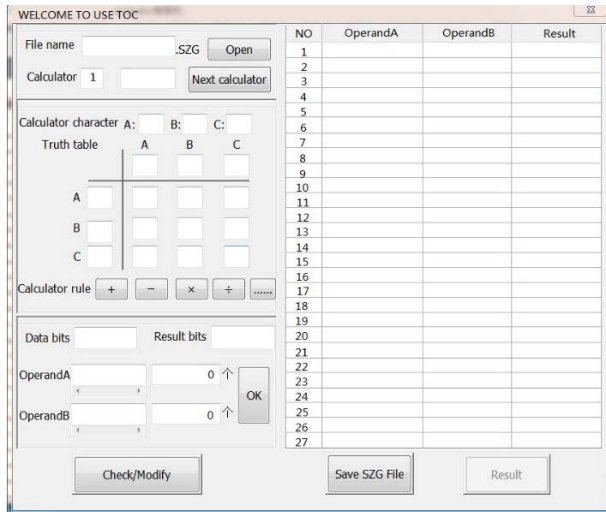


FIGURE 3. SZG file generation assistant software interface.

The SZG file interaction can be summarized as:

Step 1. The programmer generates the SZG file through the auxiliary software, sends the SZG file to the task management software of the ternary optical computer when the application is running, and makes the application wait for all the calculation results to be returned, when the application can handle other tasks.

Step 2. The ternary optical computer task management software parses the received SZG file, assigns the corresponding number of data bits from the ternary optical processor to this SZG file, and reconstructs these data bits into the calculator requested by the SZG file.

Step 3. The task management software sends the raw data in the SZG file to the reconstructed calculator one by one and collects the results until all the data in the SZG file has been calculated.

Step 4. The task management software organizes all calculation results into a result SZG file (-R.SZG file) and returns the -R.SZG file to the corresponding application.

Step 5. The application receives the -R.SZG file and processes the data in the result file for the next step.

In this process, the programmer only generates the SZG file in Step 1 and removes the data from the -R.SZG file in Step 5, while the operations on the ternary optical computer hardware and the underlying software are done by the task management software. Thus, the SZG file obscures the complexity of the ternary optical processor hardware and underlying software for the programmer, and users can more easily use the application features of the ternary optical computer.

C. EXPANSION OF THE HIGH-LEVEL PROGRAMMING LANGUAGE FOR TOC

The SZG file translates the programmer's control of the ternary optical processor into the task of transmitting the SZG file to ternary optical computer. The instructions to control the ternary optical processor are thus simplified to transfer SZG file instructions. So we simply package the program module

that transfers the SZG file to the specified target device into an expanded instruction for the current programming language - the transfer SZG file instruction. This expansion command can be used in the application to send the specified SZG file to the ternary optical computer at the right time. The goal of having the few-bit electronic processor where the application is located and the many-bit optical processor where the SZG file calculation task is done work together in one application. Expansion instructions for transferring SZG files have been successfully developed for C, MPI parallel programming language and C++ [25]–[27]. The following is an example of the core technical steps for the expansion of a high-level programming language, using the expansion statements in C as an example.

Step 1. `void SZG_Init()`. Creates the expansion instruction environment for a ternary optical computer. It must be the first expansion instruction that appears in the program and can only appear once in a program.

Step 2. `int SZG(char *path)`. Sends the *.SZG file referred to by path to the ternary optical processor. The return value of this instruction is the code for the success or failure of this send, and the programmer can decide the next action based on this return value.

Step 3. `int SZG_SearchResult(char * path)`. Address pointer path refers to the *.SZG file running status on the ternary optical processor. The return value "0" indicates that the receiving process has obtained the calculation result file *-R.SZG and stored it in the same directory as the *.SZG file. The return value of "1" means that the result file *-R.SZG is not yet available. The return value of "2" indicates that there is an error on the side of the ternary optical processor and the processing of *.SZG files has been stopped, so no result is returned.

Step 4. `void SZG_Suspend()`. Hangs this program until it is woken up when the operation result file returns. In addition, the application must also add the header file "#SZG.h" before proceeding with the above steps.

IV. IMPLEMENTATION MECHANISM OF THE TOC PROGRAMMING PLATFORM

Figure 4 gives a general view of the ternary optical computer programming platform, which consists of two computing cores, the electronic processor and the ternary optical computer (TOP). Based on these two core underlying software, the operating system (OS), the program language of the SZG file transfer instruction is extended, and the SZG file generation software, a complete programming platform is formed. As shown in Figure 4, the ternary optical computer implicitly contains an auxiliary processor with the same status as the ternary optical processor. This auxiliary processor is used to perform various transactional tasks such as memory management, task management of TOC, I/O operation management, SZG file transfer, and ternary memory management for optical processor reconfiguration planning, etc. Together with the TOC task management software, processor reconstruction software and SZG file

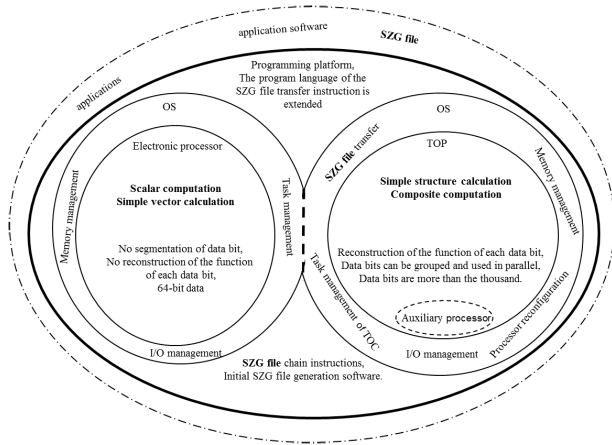


FIGURE 4. View of the TOC programming platform.

transfer software, this constitutes the underlying software system of the TOC.

The basic idea and process of preparing a ternary optical computer application by applying the ternary optical computer programming platform can be described as follows:

Step 1. Modeling. Mathematical description of the application problem according to the structural data type computer.

Step 2. Organize the data. Determine the types of data and the corresponding operation rules. Determine constants, variables, arrays, structures, and enumerations, and determine the rules for calculating each of them. We also need to organize the raw data and the corresponding operation rules into SZG files.

Step 3. Design the program structure. Draw data flow diagram, draw control flow diagram, write pseudo code program, etc.

Step 4. Write the program. Write the instruction sequence. A typical instruction sequence can be described as follows:

Initialization instructions (define and initialize variables, system environment, etc., including establishing the operating

environment for SZG expansion instructions);

Input Data Instruction;

Judgment Status Instruction;

Send the data in pairs to the operator for calculation and obtain the result instruction (e.g. $C = A + B$);

Send the data to a mathematical function routine to calculate and obtain the result instruction (e.g. $\text{Sin}(45, C)$);

The expansion instruction that sends the SZG file to the ternary optical processor for calculation (e.g. $\text{int SZG Send}(\text{char} * \text{path})$);

Waiting for the TOC to send back the calculation results of the expansion command (e.g. $\text{void SZG Suspend}()$);

.....

Judgment result instruction sequence;

Instructions that determine the flow of the program;

Sequence of instructions to output program results;

End of program.

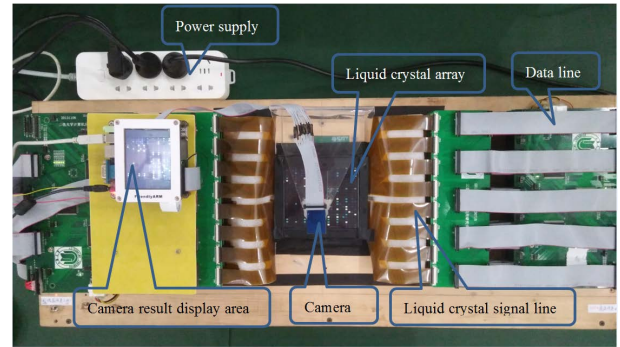


FIGURE 5. Machine #1 of SD16.

Obviously, the four main steps above are the same as the current programming process exactly. Only the concept of “TOC is a simple structural data type computer” is added in the first step. Added the method of “organizing a large amount of raw data into SZG files” in the second step. The fourth step adds the expansion instruction “Send the SZG file to the TOP and wait for the calculation result file to return”. Thus the programming platform of TOC is built on the theoretical basis of the simple structural data type computer concept, SZG files and high-level language expansion instructions proposed in our paper.

In addition, by observing the above steps, the TOC programming environment constructed by this method contains the complete, unaltered electronic computer programming environment. Therefore, in this environment, we can prepare electronic computer applications as usual, and the original electronic computer programs can also run normally intact, moreover, we can prepare electronic computers and ternary optical processors to work together as an application.

V. EXPERIMENT AND ANALYSIS

A. EXPERIMENT ENVIRONMENT

This experiment uses a TOC prototype system SD16 of No. 1, whose shape is shown in Figure 5. Among them, the TOP (the black area with the bright spot in the middle) has 576 pixels in the LCD array, arranged in a 24×24 array. Three adjacent pixels in each row constitute one bit of the optical processor, so this experimental device has a total of 192 processor bits, meaning that 192 bits of data can be processed in parallel [28]. The liquid crystal is arranged as shown in Figure 6, the liquid crystal is divided into two parts, with the processor bits in each part numbered as shown by the arrows in Figure 6. Co-working electronic computer hardware configuration: CPU: Intel(R) Core(TM) i5 430 M 2.26 GHz. RAM: 4 GB. OS: 64-bit windows 8. Expanded programming language for SZG file transfer instructions: C.

B. TEST CASES AND RESULTS ANALYSIS

In order to test the correctness of the TOC programming platform, we will implement four functions in our program. For example, calculate $f_1 = a + b, f_2 = c - d, f_3 = e \wedge g$ and $f_4 = h \vee i$. Its independent variables a, b, c, d, e, g, h , and

TABLE 2. Test cases.

Item	$f1$	$f2$	$f3$	$f4$
1	$a(u10100101u10u1010)_M$ $b(101u10u10100001u0)_M$	$c(1u1001u00011)_M$ $d(0101u10u101u)_M$	$e(1101010010111011010)_M$ $g(1011110001010000100)_M$	$h(11010100101010010101)_M$ $i(01011101011010000100)_M$
2	$a(u11100101u10u1010)_M$ $b(111u10u10100001u0)_M$	$c(0uu1u0100110)_M$ $d(11001u00011)_M$	$e(0011011001001010011)_M$ $g(1001010010101001010)_M$	$h(0101000110101111010010011)_M$ $i(1100011110010101001010111)_M$
3	$a(u11u00101u10u1010)_M$ $b(111uu0u10100001u0)_M$	$c(110u1u000111)_M$ $d(111u1uu101uu)_M$	$e(1111010010101001010)_M$ $g(1010100001010000110)_M$	$h(110111001011101101010101)_M$ $i(11011111011010000100)_M$
4	$a(u11u00111u10u1010)_M$ $b(111uu0u11100001u0)_M$	$c(110u1u010111)_M$ $d(111u1uu111uu)_M$	$e(1001010010111011010)_M$ $g(1110101001010000110)_M$	$h(11111100101010010101)_M$ $i(11011111011010000110)_M$
5	$a(u11u00111u10u1110)_M$ $b(111uu0u11100001u1)_M$	$c(1110u1u0101u)_M$ $d(111u1uu111u1)_M$	$e(1110100101110u1010)_M$ $g(1010101001010000110)_M$	$h(1011110010111011010101)_M$ $i(1011110001010000101)_M$
6	$a(u11u00111uu0u1110)_M$ $b(111uu0u1110u001u1)_M$	$c(111uu1u0101u)_M$ $d(111u1uu1u1u1)_M$	$e(0101010010111011010)_M$ $g(1010101101010000110)_M$	$h(10111100101010010101)_M$ $i(10011100001010000100)_M$
7	$a(u11u00111uu0u111u)_M$ $b(111uu0u1110uu01u1)_M$	$c(101uu1u0101u)_M$ $d(111u1uu1u0u1)_M$	$e(1001010010111001010)_M$ $g(1010100001010000110)_M$	$h(10111100101010010100)_M$ $i(11011101001010000100)_M$
8	$a(u11u00111uu0u11uu)_M$ $b(uu1uu0u1110uu01u1)_M$	$c(1uuuu1u0101u)_M$ $d(1uuu1uu1u0u1)_M$	$e(1001010010111001010)_M$ $g(1010100001010100110)_M$	$h(11111100101010010100)_M$ $i(10011101001010000100)_M$

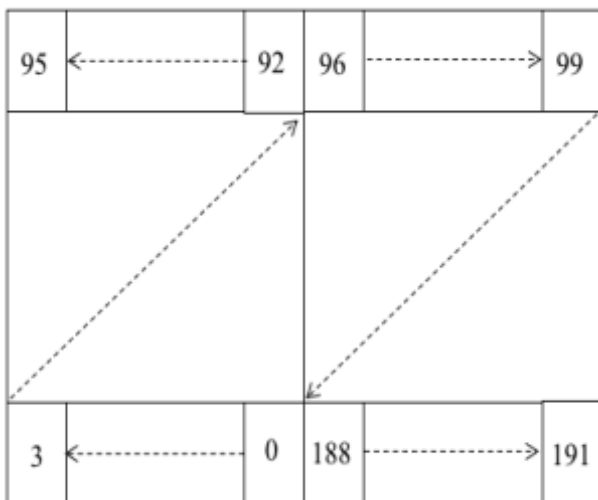


FIGURE 6. Liquid crystal arrangement.

have 17, 17, 12, 12, 19, 19, 20, 20 bits, and each variable has 10,000 data. Given the length of the paper, the first eight sets of MSD numbers for the four functions are given in Table 2, and the subscripts are labeled with M .

In a ternary optical computer, the data of $f1$, $f2$, $f3$ and $f4$ are first formed into simple structural data types. We can represent this simple structured data type by A . It has four subtypes (terms), namely: 17-bit floating-point type $A(1)$, 12-bit floating-point type $A(2)$, 19-bit logical type $A(3)$ and 20-bit logical type $A(4)$. The SZG file is then generated, and the SZG file format is instantiated as follows:

File name: Ls . Total number of operation marks: 4. Calculation rules for operation marker 1 ($f1$): +, number of data bits: 17, number of data: 10000. Calculation rules for operation marker 2 ($f2$): -, number of data bits: 12, number of data: 10000. Calculation rules for operation marker 3 ($f3$): \vee , number of data bits: 19, number of data: 10000. Calculation rule for operation mark 4($f4$): \wedge , number of data bits: 20, number of data: 10000. Then, the $Ls.SZG$ file is

TABLE 3. Allocation table for adder processor bits.

	T	W	T'	W'	T_2
$f1$	0-16	17-33	34-51	52-69	70-88
$f2$	89-100	101-112	113-125	126-138	139-152

TABLE 4. Allocation of processor bits for logical operators.

$f3$ (Logic AND)	$f4$ (Logic OR)
153-171	172-191

fed into the ternary optical computer via the C expansion command.

After the ternary optical computer receives the $Ls.SZG$ file, the task management software of the optical processor reconfigures the composite processor based on the data bit information given in the $Ls.SZG$ file. The first 89 bits of the complex processor are constructed as an adder, 89 to 152 bits as a subtractor, the next 19 bits as an “AND” operator, and the last 20 bits as an “OR” operator. Among them, corresponding to the 192 processor bits of the optical processor, the allocation of processor bits for the 5 transforms of the adder corresponding to $f1$ and $f2$ is shown in Table 3, and the allocation of processor bits for $f3$ and $f4$ is shown in Table 4. Then the first pair of raw data of $A(1)$, $A(2)$, $A(3)$ and $A(4)$ is fed into the reconstructed simple structured data type processor. Run the calculation instruction once to get a set of $f1$, $f2$, $f3$ and $f4$ results, and so on. 10,000 times in parallel can be executed to complete the calculation of all 40,000 pairs of data and get the final result.

Figure 7 shows the theoretical output results for the first 8 groups of the 4 functions in the experimental use case. Figure 8 shows the resultant images of the optical processor output corresponding to the theoretical results. Table 5 shows the theoretical results in the form of MSD. where, in Figure 8, the high luminance points are pixels that output vertically polarized light (V state), the low luminance points are

TABLE 5. Theoretical values of experimental results expressed in MSD.

Item	$f1$	$f2$	$f3$	$f4$
1	$(0010u1u010u1u000u00)_M$	$(01u1u10u10u100)_M$	$(1111110011111011110)_M$	$(01010100001010000100)_M$
2	$(001101u010u1u000u00)_M$	$(011100u01u1u00)_M$	$(1111110011111001110)_M$	$(10010100001010000100)_M$
3	$(0011uu1010u1u000u00)_M$	$(011001u00u1000)_M$	$(1111110011111001110)_M$	$(11011100001010000100)_M$
4	$(0011uu101101u000u00)_M$	$(011001u0010000)_M$	$(1111111011111011110)_M$	$(11011100001010000100)_M$
5	$(0011uu101101u00001u)_M$	$(0111u10u101u00)_M$	$(1111111011111001110)_M$	$(10111100001000000101)_M$
6	$(0011uu10110u010001u)_M$	$(0111u00u1u1u00)_M$	$(1111111111111011110)_M$	$(10011100001010000100)_M$
7	$(0011uu10110u0010000)_M$	$(0110u00u1u0000)_M$	$(1011110011111001110)_M$	$(10011100001010000100)_M$
8	$(0u01uu10110u0010u00)_M$	$(01uuu00u1u0000)_M$	$(1011110011111011110)_M$	$(10011100001010000100)_M$

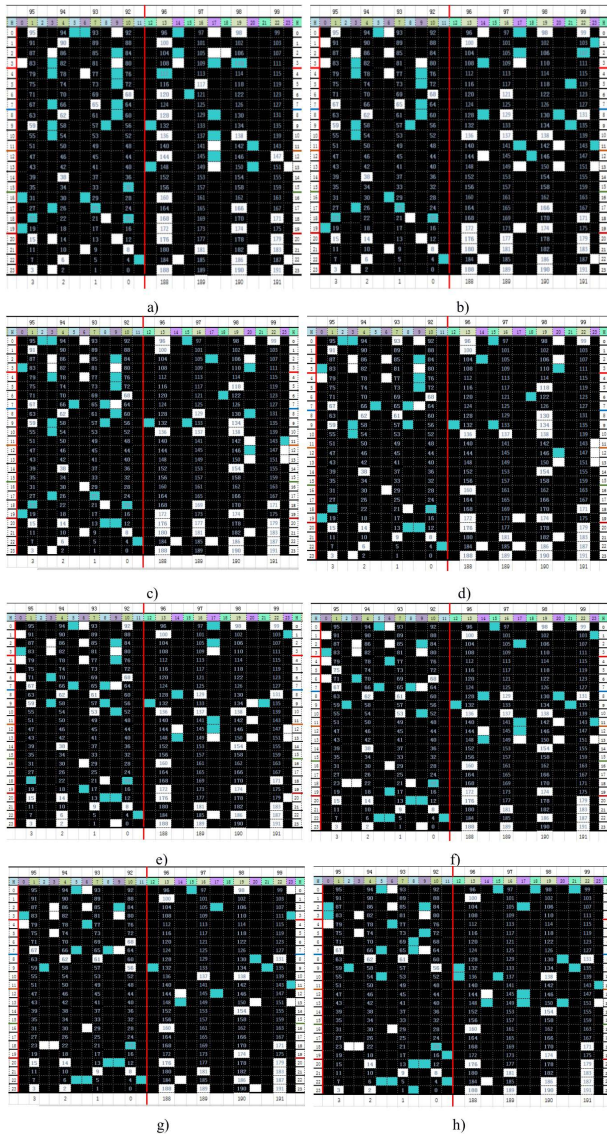


FIGURE 7. Theoretical output results. a)-h) are the theoretical outputs for each group of four functions of the test case.

pixels that output horizontally polarized light (H state), and the rest of the points are pixels that output no light state (W state).

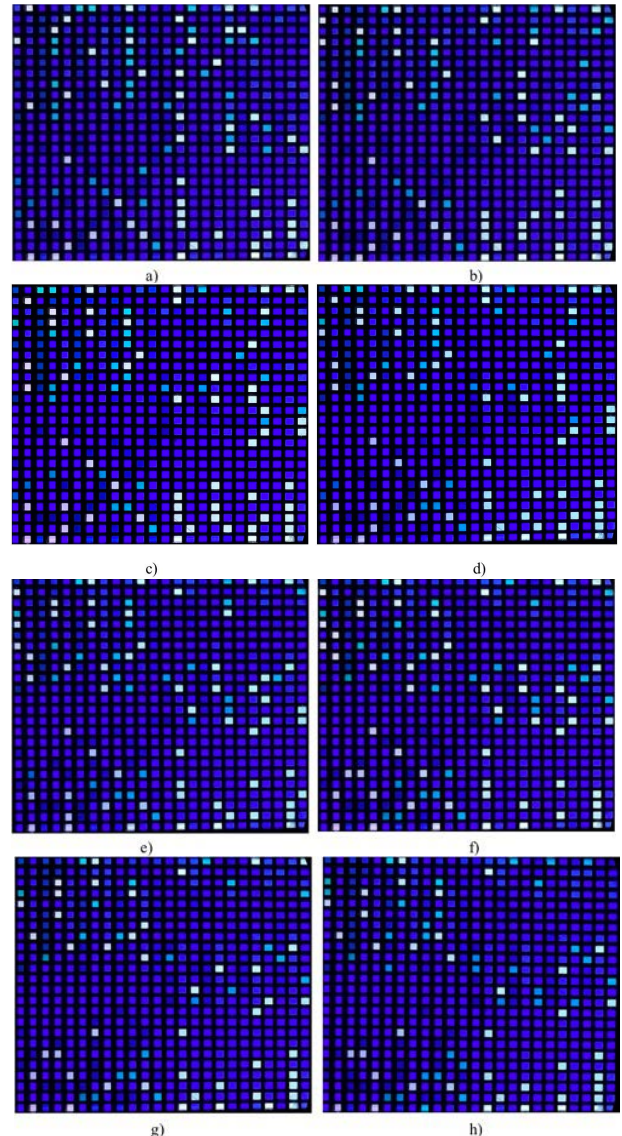


FIGURE 8. Optical processor output results. a)-h) are the optical processor outputs for each group of four functions of the test case.

Taking the first set of results of the addition operation as an example, it can be seen from Table 3 that bits 70-88 of the operator are the result area, which corresponds to the

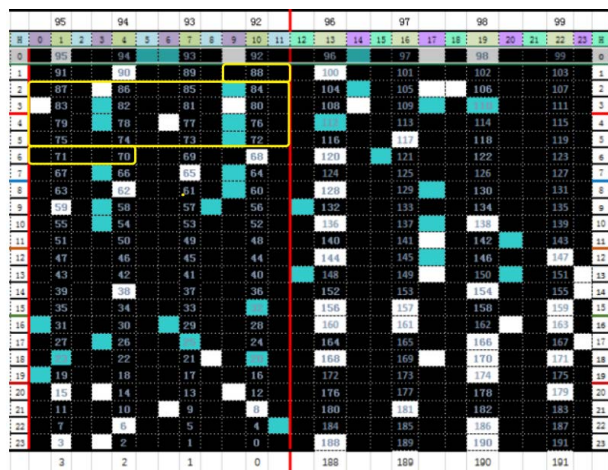


FIGURE 9. The first set of results of the addition operation.

area marked in yellow in Figure 9. White represents vertical light and is represented by 1, blue represents horizontal light and is represented by u , black represents no light and is represented by 0. According to the decoding principle of the ternary optical computer, the number of liquid crystal bits is from left to right and from top to bottom, and every 3 liquid crystal pixels represent one processor bit, in the order of $(0010u1u010u1u000u00)_M = (54076)_D$, which is consistent with the theoretical results. We analyzed the rest of the results and obtained all accurate results. Experiments show that the SZG file generated by the electronic computer can be correctly parsed by the ternary optical processor, that the high-level language expansion instructions can deliver the SZG file to the ternary optical processor correctly, and that the ternary optical processor can compute data of a structural data type correctly. Therefore, the programming platform proposed in this paper can make use of the computational features of the ternary optical computer while allowing programmers to maintain their traditional programming habits and build a bridge between the user and the ternary optical computer.

VI. CONCLUSION

The ternary optical computer provides us with a massively data-parallel computing tool through three major application features: the large number of data bits, the fact that the data bus can be used independently in arbitrary groups, and the fact that the computational function of each data bit can be reconfigured at runtime. However, current programming platforms are built around traditional processors such as electronic CPU, GPU or MIC, and are unable to produce applications that take advantage of the strengths of ternary optical computers. The theory and techniques presented in this paper give a basic scheme for constructing programming environments for new computer systems. The new programming environment is not only fully compatible with traditional programs, but also maintains the programmer’s work and thinking habits very well, making it easy for

programmers to accept and develop new programs that work with traditional processors and new processors. Meanwhile, when other new computers are to enter the application stage, the corresponding programming models can be constructed by drawing on the ideas we propose in this paper.

DISCLOSURES

The authors declare that there are no conflicts of interest regarding the publication of this paper.

ACKNOWLEDGMENT

This work was supported by the c (61866006, 61775139, 61772164, 62072126), the Shanghai Sailing Program (21YF1415100, 21YF1432900), and University-level general research project of Shanghai Normal University(SK202121).

REFERENCES

- [1] D. Xiaoyi and L. Xueshen, “The advance in optical electronics devices,” (in Chinese), *J. Optoelectron. Laser*, vol. 1, no. 1, pp. 12–23, 1990.
- [2] A. K. Cherri, M. K. Habib, and M. S. Alam, “Optoelectronics recoded ternary signed-digit adder using optical correlators,” in *Proc. Aerosp. Electron. Conf.*, 1997, pp. 495–502.
- [3] J. Cai, A. Miyake, W. Dür, and H. J. Briegel, “Universal quantum computer from a quantum magnet,” *Phys. Rev. A, Gen. Phys.*, vol. 82, no. 5, pp. 052309.1–052309.5, Nov. 2010.
- [4] T. D. Ladd, F. Jelezko, and R. Laflamme, “Quantum computers,” *Nature*, vol. 464, no. 7285, pp. 45–53, 2010.
- [5] S. Tyll, E. Budinger, and T. Noesselt, “Thalamic influences on multi-sensory integration,” *Commun. Integr. Biol.*, vol. 4, no. 4, pp. 378–381, Jul. 2011.
- [6] J. Yi, H. Huacan, and L. Yangtian, “Ternary optical computer architecture,” *Phys. Scripta*, vol. 118, pp. 98–101, 2005.
- [7] J. Yan, Y. Jin, and K. Zuo, “Decrease-radix design principle for carrying/borrowing free multi-valued and application in ternary optical computer,” (in Chinese), *Sci. China F, Inf. Sci.*, vol. 51, no. 10, pp. 1415–1426, 2008.
- [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,” (in Chinese), *Sci. China Inf. Sci.*, vol. 54, no. 11, pp. 2236–2246, Nov. 2011.
- [9] Y. Jin, S. Ouyang, K. Song, Y. F. Shen, J. J. Peng, and X. M. Liu, “Management of many data bits in ternary optical computers,” *Scientia Sinica Inf. Sci.*, vol. 43, no. 3, pp. 361–373, 2013.
- [10] J. Peng, R. Shen, Y. Jin, Y. Shen, and S. Luo, “Design and implementation of modified signed-digit adder,” *IEEE Trans. Comput.*, vol. 63, no. 5, pp. 1134–1143, May 2014.
- [11] J. Peng, Y. Shen, S. Ouyang, X. Liu, W. Li, and Y. Jin, “Structure and theory of dual-space storage for ternary optical computer,” *Scientia Sinica Inf.*, vol. 46, no. 6, pp. 743–762, Jun. 2016.
- [12] Y. Jin, Y. Shen, J. Peng, S. Xu, G. Ding, D. Yue, and H. You, “Principles and construction of MSD adder in ternary optical computer,” *Sci. China F, Inf. Sci.*, vol. 53, no. 11, pp. 2159–2168, Nov. 2010.
- [13] J. Yi, H. Huacan, and A. Lirong, “Lane of parallel through carry in ternary optical adder,” *Sci. China F, Inf. Sci.*, vol. 48, no. 1, pp. 107–116, 2005.
- [14] Y. Jin, “Management strategy of data bits in ternary optical computer,” (in Chinese), *J. Shanghai Univ., Nature Sci.*, vol. 13, no. 5, pp. 519–523, 2007.
- [15] Y. Jin, X. C. Wang, and J. J. Peng, “Conceptual structure of ternary optical computer and high performance computer merger,” *High Perform. Comput. Tech.*, vol. 6, pp. 1–4, 2010.
- [16] Y. Shen, B. Jiang, J. Peng, Y. Jin, O. Shan, and J. Peng, “Principle and design of ternary optical accumulator implementing M-k-B addition,” *Proc. SPIE*, vol. 53, no. 9, pp. 95–108, 2014.
- [17] X. Wang, J. Peng, M. Li, Z. Shen, and O. Shan, “Carry-free vector-matrix multiplication on a dynamically reconfigurable optical platform,” *Appl. Opt.*, vol. 49, no. 12, pp. 2352–2362, 2010.
- [18] J. Y. Yan, Y. Jin, and K. Z. Zuo, “No carry, no borrow n-value operate unit,” China Patent 2007 100 411 441, Oct. 28, 2009.
- [19] Y. Jin, S. Ouyang, and J. J. Peng, “Reconfigurable tri-value optical processor,” China Patent 2010 105 841 293, May 2, 2012.

- [20] J. Peng, L. Teng, and Y. Jin, "Realization of a tri-valued programmable cellular automata with ternary optical computer," *Int. J. Numer. Anal.*, vol. 2, no. 9, pp. 304–311, 2012.
- [21] A. Avizienis, "Signed-digit number representations for fast parallel arithmetic," *IRE Trans. Electron. Comput.*, vol. 10, no. 3, pp. 389–400, 1961.
- [22] W. Li, S. Ouyang, Y. Jin, Y. Han, and Q. Xu, "Structured data computer—Application characteristics of a ternary optical computer," *Scientia Sinica Inf.*, vol. 46, no. 3, pp. 311–324, Mar. 2016.
- [23] S. Zhang, Y. Han, Y. Shen, and Y. Jin, "Principle of a computing request file of ternary optical computers," in *Proc. 3rd Int. Conf. High Perform. Comput. Appl.*, vol. 7, 2015, pp. 150–157.
- [24] S. Li and Y. Jin, "Simple structured data initial SZG file's generation software design and implementation," in *Proc. 3rd Int. Conf. Wireless Commun. Sensor Netw. (WCSN)*, 2017, pp. 383–388.
- [25] H. Gao, Y. Jin, and K. Song, "Extension of C language in ternary optical computer," (in Chinese), *J. Shanghai Univ., Nature Sci.*, vol. 19, no. 3, pp. 280–285, 2013.
- [26] Q. Zhang, Y. Jin, K. Song, and H. Gao, "MPI programming based on ternary optical in supercomputer," (in Chinese), *J. Shanghai Univ., Nature Sci.*, vol. 20, no. 2, pp. 180–189, 2014.
- [27] S. Li, *Theory and Design of a Three-Value Computer Programming Platform*. Shanghai, China: Shanghai Univ., 2019.
- [28] J. B. Jiang, X. F. Zhang, Y. F. Shen, and S. Ouyang, "Design and implementation of SJ-MSD adder in ternary optical computer," (in Chinese), *Acta Electron. Sinica*, vol. 49, no. 2, pp. 275–285, 2021.



SHUANG LI was born in 1988. She received the Ph.D. degree from the School of Computer Engineering and Science, Shanghai University, Shanghai, in 2019. She is currently a Lecturer with the College of Information, Mechanical and Electrical Engineering, Shanghai Normal University, China. Her research interests include parallel computing, swarm intelligence systems, and ternary optical computer.



ZHEHE WANG was born in 1980. He is currently pursuing the Ph.D. degree with the School of Computer Engineering and Science, Shanghai University. He is currently a Lecturer with Hainan Tropical Ocean University. His main research interests include storage architecture, file systems, embedded systems, and ternary optical computers.



SHUXIN WANG was born in 1996. She is currently pursuing the degree with the School of Computer Engineering and Science, Shanghai University. Her research interest includes ternary optical computer.



DONGDONG AN received the Ph.D. degree in software engineering from East China Normal University, Shanghai, China, in 2020. She is currently a Lecturer with Shanghai Normal University, Shanghai. From 2016 to 2018, she got a scholarship from China Scholarship Council (CSC) to work as a joint Ph.D. Student in KAIROS Team in the French Institute for Research in Computer Science and Automation (INRIA), France. Her research interests include model-driven architecture, machine learning, formal methods, and statistical model checking techniques.

...