Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2020) 28: 904 – 916
© TÜBİTAK
doi:10.3906/elk-1905-35

# Ternary logical naming convention and application in ternary optical computers

**Yi JIN, Shuang LI***
School of Computer Engineering and Science, Shanghai University, Shanghai, P.R. China

**Received:** 07.05.2019 • **Accepted/Published Online:** 29.08.2019 • **Final Version:** 28.03.2020

**Abstract:** This article introduces the ternary logical naming convention, which was newly discovered in the study of logical units of ternary optical computers (TOCs). First, the design principle and design specification of the ternary logical naming convention are elaborated in detail and several examples are given to illustrate the use of the naming convention. Second, taking the modified signed-digit (MSD) adder of the TOC as an example, the naming convention is applied to build four ternary logical units of the MSD adder, and the implementation method of pipelined addition is introduced. Finally, the correctness of the ternary logical naming convention proposed in this paper and the usability of the adder built according to this convention are illustrated by experiments. The ternary logical naming convention can easily obtain the relationship between the ternary logic transforms, thus judging the characteristics of the logic units of TOCs, which is helpful to promote the in-depth study of TOCs in the field of numerical calculation.

**Key words:** Ternary logical, ternary optical computer, naming convention

## 1. Introduction

Logic is an old and dynamic field of study, which addresses judgment or reasoning and serves as the common foundation of various scientific theories [1]. Among them, discrete-valued logic problems, and especially binary-valued logic problems, are closely related to the application of electronic computers and have become the basis of modern computer-based transaction processing [2–4]. The complexity of the logic system expands exponentially as the number of logical variables increases. For example, a binary logic system has only $2^{(2\times2)} = 16$ kinds of logic operation rules, but a ternary-valued logic system has $3^{(3\times3)} = 19683$ kinds of operation rules. Generally speaking, the n-valued logic system has a total of $n^{(n\times n)}$ operation rules. For binary logic, people have named 16 operating rules with proprietary names, such as the well-known AND, OR, NAND, and XOR. The source of these names should be closely related to the "on, off" states in the circuit, so binary logic was called "switching logic" for a long period of time. For $n$ ($n>2$) value logic, it is obviously impossible to give each operation an exclusive name. Therefore, truth tables are still widely used to express the $n$-valued logic operation rule. This expression method is not only difficult to remember, but also difficult to communicate, and it is not easy to recognize the similarities and differences between different logic operations. On the other hand, after giving $n$ values different characteristics, some logic operations lose their actual value, some logic operations change duality, and some logic operations change other aspects of the property. Therefore, the use of truth tables and value features to indicate logical operations is not only complicated but also difficult to discuss.

In the work of studying the ternary logic optical operator, the author finds that with the ternary optical computer entering the application field, various calculation routines and application algorithms using ternary

*Correspondence: lslsshuang@126.com

logic operations continue to emerge. Therefore, it is very necessary to establish a unique scientific name for each ternary logical operation. This means that a convenient and practical ternary logical operation naming scheme must be developed, which must be satisfied: make each ternary logical operation have a corresponding standard name; Regardless of the symbol used by the user to express a three-valued operation in the actual problem, the logical operation rule can be quickly given by its standard name, thereby avoiding academic confusion; The naming convention must also be able to extend the naming of the $n^{(n\times n)}$ operation rules for $n$-valued logic.

## 2. The naming principle of ternary logical

### 2.1. Definition

The ternary logical rules can always be expressed in a three-row, three-column truth table, as shown in Table 1, where A and B are two independent variables of ternary logical and Y is a ternary logical function. The ranges of A, B, and Y are collections {a, b, c}.

Definition 1. The different restrictions between the three symbols are called the characteristic name of the value (CNV) in the ternary logical transform.

Definition 2. The rules used in logical reasoning for ternary logical transforms are called available rules (ARs).

Definition 3. The name of the transform result obtained by the ternary logical naming convention is the name of the transform rule (NTR).

The CNV is different and the AR is different when inferring. For example, when the three symbols of ternary logic are completely unrelated, the dual rule cannot be used for the ternary logic transform when inferring, but when one symbol is defined as the median and the other two symbols are the dual value, the dual rule can be used. Therefore, it is not only required to write the NTR, but also to write the CNV when naming the ternary logic.

**Table 1**. Example of a ternary logical operation standard truth table.

| A \ B | a | b | c |
|---|---|---|---|
| a | a | c | c |
| b | b | c | a |
| c | a | c | b |

### 2.2. Design and design specifications for ternary logical naming convention

#### 2.2.1. Design of the NTR

The value of the ternary logical standard name is given by the column marker value of "a" or "b" of each row of the standard truth table:

[N1 N2  N3 N4  N5 N6  Missing column mark (MCM)],

where the form contains six numbers N1–N6 and MCM, N1 denotes the "a" column mark of the first row, N2 denotes the "b" column mark of the first row, N3 denotes the "a" column mark of the second row, N4 denotes the "b" column mark of the second row, N5 denotes the "a" column mark of the third row, and N6 represents the "b" column mark of the third row. To distinguish them from ordinary numbers, we put them in square brackets. The rules for the values of these six numbers, the missing column mark, and the missing row mark (MRM) are shown in Table 2.

**Table 2**. The rule of each marked value in the names of the ternary logic operation rule.

| Numeric value rules | | MCM value rules | | MRM value rules | |
|---|---|---|---|---|---|
| The column number where "a" or "b" appears | Marked value | Missing column | MCM value | Missing row | Replace the missing row with X |
| 1 | 1 | c | -c | 1 | N1 N2 |
| 2 | 2 | b | -b | 2 | N3 N4 |
| 3 | 4 | a | -a | 3 | N5 N6 |
| null | 0 | c and a | -ca | - | - |
| 1 and 2 | 3 | c and b | -cb | - | - |
| 1 and 3 | 5 | a and b | -ab | - | - |
| 2 and 3 | 6 | - | - | - | - |
| 1 and 2 and 3 | 7 | - | - | - | - |

According to Table 2, there are the following constraints between the values of "a" and "b" in the same row:

When "a" is 0, "b" can be any value from 0 to 7, a total of 8 values.

When "a" is 1, "b" can be 0, 2, 4, and 6, a total of 4 values.

When "a" is 2, "b" can be 0, 1, 4, and 5, a total of 4 values.

When "a" is 3, "b" can be 0 and 4, a total of 2 values.

When "a" is 4, "b" can be 0, 1, 2, and 3, a total of 4 values.

When "a" is 5, "b" can be 0 and 2, a total of 2 values.

When "a" is 6, "b" can be 0 and 1, a total of 2 values.

When "a" is 7, "b" can be 0, a total of 1 value.

Therefore, there are 27 cases for each row, and the ternary transform has $27^3 = 19683$.

### 2.2.2. Design of the CNV

The characteristics of three symbols of ternary logic determine the mathematical operations that can be used in the reasoning process. Therefore, a category of ternary transform is delineated and the three symbols are named according to the type of the ternary transform so that the mathematical operation rules can be used in the reasoning process. For example, use "L" to indicate that the characteristic of the value is logic, "C" to indicate that the characteristic of the value is contain, "D" to indicate that the characteristic of the value is digit, etc.

For the "C"-type, it is necessary to give the inclusion relationship of three symbols, and for the "D"-type, the specific values of the three symbols must be given. Then set a pair of parentheses in the CNV, which contains three elements corresponding to three symbols. For example:

L (x1, x2, x3), where x1, x2, and x3 are the three symbols actually used by the user, which represent one of the three events that are not included in each other. This type of transform rule only indicates that when two events are "true" (the value of two inputs), an event is also "true" (the output value of the logical transform), but it does not indicate that other events are “false” or “not sure”. The actual symbol x1 corresponds to the symbol “c”, x2 corresponds to “a”, and x3 corresponds to “b”.

Propositional logic is a common example of the “L”-type. For binary propositional logic, one symbol is 0, which means the proposition is false, and the other symbol is 1, which means the proposition is true. For ternary propositional logic, the third symbols can have multiple definitions, such as semi-true half-false (0.5), or a certain degree of “true” (fuzzy logic), or may have nothing to do with true and false, etc. In this case, x1 is equal to 0, x2 is equal to 1, and x3 gives its definition.

For the “C”-type, there is a “true” inclusion relationship between the three symbols. One symbol that is “true” contains one or two symbols as “true”. If you add the “true inclusion” relationship in parentheses, it can have the following types:

C (x3, x2 <, <x1): “c” corresponds to x3 and has no true contain relation; “a” corresponds to x2 and is contained; “b” corresponds to x1, and true contains “a”.

C (x3, <x2, x1 <): “c” corresponds to x3 and has no true contain relation; “a” corresponds to x2, and true contains “b”; ”b” corresponds to x1 and is contained.

C (x1, x2 <, <x3): “c” corresponds to x1 and has no true contain relation; “a” corresponds to x2 and is contained; “b” corresponds to x3 and contains “a”.

C (<x3, x2 <, x1): “c” corresponds to x3, and true contains “a”; “a” corresponds to x2 and is contained; “b” corresponds to x1 and there is no true contain relation.

C (<x3 <, x2 <, <x1): “c” corresponds to x3, and true contains “a” and is contained; “a” corresponds to x2 and is true contained by “b” and “c”;“b” corresponds to x1, and true contains “c” and “a”.

For the “D”-type, three symbols of the ternary logic represent three numbers; that is, x1, x2, and x3 are represented by three numbers. The “D”-type is a special case of “C”-type when considering numerical values. For example:

D (5, 0.3, 7): in the ternary transform, “c” represents 5, “a” represents 0.3, and “b” represents 7.

It should be noted that the three elements of the CNV are the symbols used in the ternary logic transform, and they are not related to the missing column and missing row in the NTR.

### 2.2.3. Design specification for ternary logical naming convention

According to the above analysis, the ternary logical naming design specification is as follows:

(1) Determine the correspondence between ternary logical symbols “a”, “b”, “c” and the actual symbols x1, x2, x3;

(2) Arrange the ternary logic truth table in the order of “a”, “b”, and “c”;

(3) Determine the CNV;

(4) Determine the name of NTR according to Table 1.

## 3. Application of ternary logical naming convention in TOCs

### 3.1. Architecture of TOCs

A TOC uses no intensity light, horizontal polarized light, or vertical polarized light polarization directions of light, as shown in Figure 1. The bit reconfigurable processor of the TOC is a product of the decrease-radix design theory. The main conclusion of the theory is that if the D-state is included in $n(n > 1)$ physical states that are used to represent information, each of $n$-valued logic operators with 2 inputs can be made of no more than $n\times(n\text{-}1)$ basic operation units via a determinate procedure, where there are up to $n\times\ n\times\ (n\text{-}1)$ types of basic operation units. The total number of different 2-input $n$-valued logic operators is known to be $n\ (n\ \times\ n)$. The D-state is a special physics state, which would still generate state $\lambda$ as the result of superimposition with any other physical state $\lambda$. A basic operation unit is an $n$-valued logic operator with the simplest structure and the following feature: only one combination of input values would produce a non-D-state as the output, whereas all the remaining input combinations invariably produce the D-state.

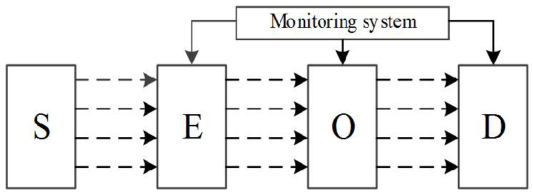

**Figure 1**. The processor structure of TOC. S: Surface light source, E: encoder, O: optical processor, D: decoder.

When applying the decrease-radix design theory to a TOC, there are $3^{(3\times3)}\ =19683$ ternary logic operators and 18 basic operation units. Any ternary logic operation unit can be made of no more than 6 basic operation units. After thorough research, we found there is a uniform structure for the 18 basic operation units of the TOC as displayed in Figure 2. We can see in Figure 2 that there are two optical paths: one is the main light path and the other is the control light path. These two optical paths are composed of a liquid crystal array, and they have different polarizers. The liquid crystal of the main light path is divided into four parts (HH, HV, VH, VV), and the liquid crystal of the control light path is divided into two parts (H, V), as shown in Figure 3. The input signal "a" enters the main optical path, which involves two polarizers (P1 and P2) holding a liquid crystal (LC) to form a sandwich-like structure. Another input signal, "b", enters the control optical path. The differences between basic operation units are that there are two opposite cases for the optical rotation of LC in the static state, four combinations of P1 and P2 polarization directions, and three kinds of control optical paths. The three control optical paths are produced by dividing input signal "b" into three subbeams by two half-reflecting mirrors f1, f2 and mirror F. Because there is a vertical polarizer V in the top branch of the control optical path, phototube g1 outputs high voltage only when "b" is vertical polarized light. Similarly, for a horizontal polarizer H in the middle branch, g2 outputs high voltage only when "b" is horizontal polarized light. For no polarizer in the bottom branch, g3 outputs high voltage when "b" is bright, whether vertical or horizontal polarized light. Device S (three-choose-one multiplexer) selects the right one from the outputs of g1, g2, or g3 according to the reconfigure directive bits k2 and k3 and sends the right signal to XOR gate Y. Y will negate the output signal of S when the reconfigure directive bit k1 is 1, and will not negate when k1 is 0. The output signal of Y controls the optical rotation of the LC in the main optical path, so the output signal "c" is produced from the main optical path under the sway of the control optical path. The main optical path is split into four kinds in accordance with the combinations of P1 and P2 polarization directions. When P1 is a vertical polarizer and P2 is a horizontal polarizer, the main optical path is called VH; when P1 is a horizontal polarizer and P2 is a vertical polarizer, it is HV. By that analogy, P1 and P2 are vertical polarizers as VV, and P1 and P2 are horizontal polarizers as HH.

### 3.2. Comparison example of ternary logic naming convention and traditional truth table

The carry-in process of the addition operation severely delays the working speed of the adder. It was not until 1950 that American-Lithuanian computer scientist A. Avizienis proposed the theory of carry-less addition

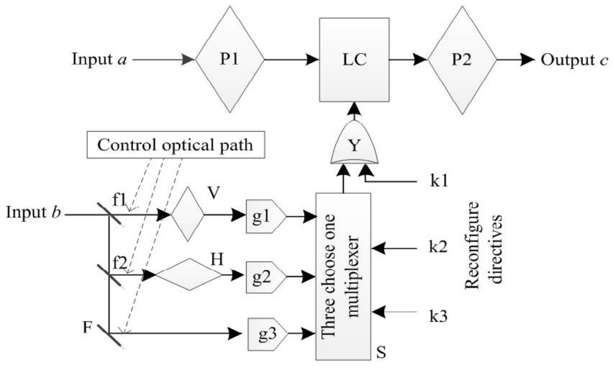

**Figure 2**. Uniform structures of the TOC's basic operation units.

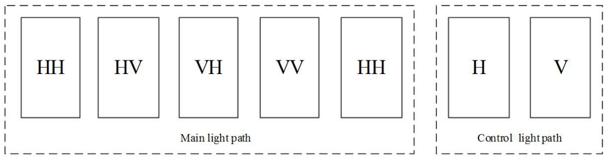

**Figure 3**. Optical processor light path division.

with redundant values and designed the first MSD number parallel adder structure, which brought a bright future for completely solving this thorny problem [5–7]. In 1986, the structure of the MSD adder of Avizienis was transformed into a three-step logic transformation. Each step's transform uses a different ternary logic operation, sharing up to four ternary logic operations and calling these four logical operations T, W, T', and W'. In 2009, Professor Jin discussed the conditions and advantages of using the TW-MSD adder for TOCs in theory [8–16]. In just three years, there have been a variety of adders, such as the improved MSD carry-free adder for TOCs [17], the one-step binary MSD adder for TOCs, [18],modified signed-digit adders [19], and the principle of a one-step MSD adder for TOCs [20]. The TW-MSD adder was implemented on ShangDa-16 (for short, SD16: Shanghai University 2016) in March 2017, and only two months later, the theory that sufficient conditions for a ternary logical transform could constitute an MSD adder was proposed. Ternary operation constituting a sufficient condition for the MSD adder and the ternary optical JS-MSD adder [21], and other series of theories, structures, and objects, were also proposed. It is very difficult to use the truth table of ternary logic to determine the similarities, differences, and characteristics of various MSD adders [22–26]. However, after writing the standard names of the ternary logic operations used by the respective MSD adder, the similarities and differences between the various MSD adders and their respective features are clear at a glance. This not only avoids unnecessary arguments, but more importantly, it provides a simple and effective tool for improving the MSD adder.

For example, the truth table of four ternary logic operations of T, W, T', and W' are shown in Table 3. The table of the ternary logic operation used by the SJ-MSD adder is shown in Table 4. It can be seen that both expressions and comparisons are cumbersome. After changing to the standard name, their CNVs are the same, as both are D (0, -1, 1), and the NTRs are different, as shown in Table 5. For an intuitive representation, "a", "b", and "c" in the ternary logical naming convention are replaced with three physical states of H, V, and D in the ternary optical computer [27–29].

The information expressed in Table 5 is the same as the information expressed in Table 3 and Table 4, but Table 5 is much simpler and easier to write and discuss, and one immediately sees: (1) these two MSD adders are different structures; (2) the output data of T' will be only u in the second column of the second row,

**Table 3**. Ternary logic operation truth table for TW-MSD addition.

| T transform | | | | W transform | | | | T′ transform | | | | W′ transform | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $s$ or $w'$ \ $p$ or $t'$ | u | 0 | 1 | $s$ \ $p$ | u | 0 | 1 | $w$ \ $t$ | u | 0 | 1 | $w$ \ $t$ | u | 0 | 1 |
| u | u | u | 0 | u | 0 | 1 | 0 | u | u | 0 | 0 | u | 0 | u | 0 |
| 0 | u | 0 | 1 | 0 | 1 | 0 | u | 0 | 0 | 0 | 0 | 0 | u | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 | u | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |

**Table 4**. Ternary logic operation truth table for SJ-MSD addition.

| S1 transform | | | | S2 transform | | | | J1/J2 transform | | | | J3 transform | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $b$ \ $a$ | 0 | 1 | u | $b$ \ $a$ | 0 | 1 | u | $s1$ \ $s2$ | 0 | 1 | u(J2) | $j2$ \ $j1$ | 0 | 1 |
| 0 | 0 | 0 | u | 0 | 0 | 1 | 1 | 0 | 0 | u | u | 0 | 0 | u |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | u | 1 | u | 0 |
| u | u | 0 | u | u | 1 | 0 | 0 | u | u | 0 | 0 | u | 0 | 1 |

**Table 5**. Ternary logical operation standard name used for TW-MSD addition and SJ-MSD addition.

| value feature name :D(0, -1, 1). u stands for -1 in the truth table | | | |
|---|---|---|---|
| TW-MSD addition | | SJ-MSD addition | |
| T | [24 30 05] | S1 | [20 30 04] |
| W | [42 01 10] | S2 | [06 01 01] |
| T′ | [00 20 04] | J1/J2 | [60 10 21] |
| W′ | [24 10 01] | J3 | [40 04 10] |

the third column and the third column are 1, and the rest are 0; (3) there is no u value in the output of S2; (4) the input variables of J3 have only two values of 0 and 1.

The following is an example of the operation rule of T, which introduces the detailed steps of using the ternary logic naming convention. For convenience in reading, the T operation truth table in Table 3 is rewritten as the T(1) subtable of Table 6.

**Table 6**. Naming of T logical transforms.

| T(1) | | | | T(2) | | | | T(3) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $s$ \ $p$ | u | 0 | 1 | $s$ \ $p$ | H | D | V | $b$ \ $a$ | D | H | V |
| u | u | u | 0 | H | H | H | D | D | D | H | V |
| 0 | u | 0 | 1 | D | H | D | V | H | H | H | D |
| 1 | 0 | 1 | 1 | V | D | V | V | V | V | D | V |

Step 1: Replace 0, u, and 1 in the T(1) subtable with symbols D, H, and V, respectively, and obtain the T(2) subtable of Table 6;

Step 2: Rewrite the T(2) subtable according to the order of D, H, and V to obtain the T(3) subtable; the T(3) subtable is the standard truth table of the T operation;

Step 3: Determine the identification feature name D(0, u, 1) or D(0, -1, 1); see line 1 of Table 5;

Step 4: Give the identification values of the columns of the H and V symbols of each row in the T (3) subtable.

In the first row, H is in the second column and V is in the third column, so the identifier of the first row is 24. In the second row, H is in columns 1 and 2, and there is no V, so the identifier of the second row is 30. In the third row, there is no H and V is in columns 1 and 3, so the identifier of the third row is 05.

Combine the three lines of identification and write the NTR: [24 30 05]. The above four steps are used for the other ternary logic operations in Table 3 and Table 4, respectively, and the respective NTRs given in Table 5 are obtained.

## 4. Experiment and analysis

### 4.1. Experimental preparation

This experiment uses the No. 1 machine of the TOC prototype system SD16; the shape is shown in Figure 4. Among them, the liquid crystal (LC) array of the TOP (black area with bright spots in the middle) has 576 pixels arranged in a 24 $\times$ 24 array. The three adjacent pixels in each line form one bit of the optical processor, so the experimental device has a total of 192 processor bits, meaning that 192 bits of data can be processed in parallel [30]. The LC arrangement is shown in Figure 5. The LCD is divided into two parts, and the processor bit number of each part is shown by the arrow in the figure. The three pixels in each processor bit in the left half are D, V, and H from left to right, and the right half is H, V, and D; that is, the pixels on the left and right are arranged in mirror image.

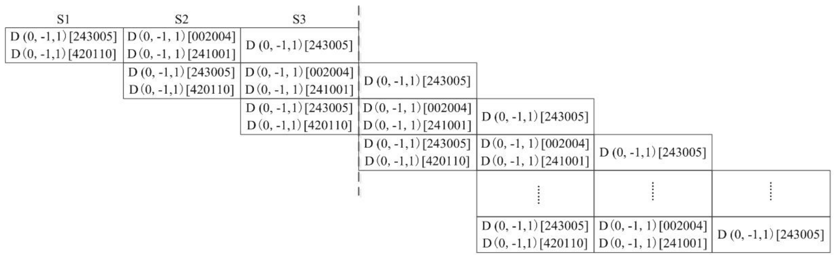

**Figure 4**. Shape of SD16's No. 1 machine.

### 4.2. Experimental cases

We need to compute $f1 = a + b$, $f2 = c + d$, and $f3 = e + g$; the number of bits of the independent variables $a$, $b$, $c$, $d$, $e$, and $g$ are 14, 14, 9, 9, 12, and 12, respectively. $f1 \sim f3$ each contain 6 sets of data, as shown in Table 7.

### 4.3. Experimental procedure

The ternary optical computer research team developed a programming platform for TOCs [31–35]. We applied the programming platform to give the main experimental steps:

(1) The ternary logical standard names of the three adders of $f1 \sim f3$ are organized into a reconstructed frame to send to the TOP;

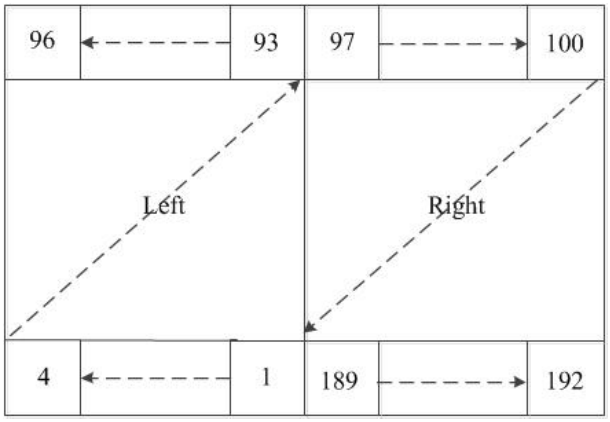

**Figure 5**. Liquid crystal division.

**Table 7**. Experimental data.

| Item | $f1$ | $f2$ | $f3$ |
|---|---|---|---|
| 1 | $a(0u01111u110111)_M$ | $c(11u11u1u1)_M$ | $e(1011u110u110)_M$ |
| | $b(01u01u11101uu1)_M$ | $d(u11101u1u)_M$ | $g(10u1u1u00110)_M$ |
| 2 | $a(01010110001011)_M$ | $c(000000000)_M$ | $e(0uu1100u0110)_M$ |
| | $b(11011101010101)_M$ | $d(000000000)_M$ | $e(g(0u10u10u1110)_M$ |
| 3 | $a(0u01101u110111)_M$ | $c(uuuuuuuuu)_M$ | $e(10uu1u0u1010)_M$ |
| | $b(01u01u11101u01)$M | $d(uuuuuuuuu)_M$ | $g(110uu100u111)_M$ |
| 4 | $a(11111111111111)_M$ | $c(10uu1u0u1)_M$ | $e(10uu1u0u1010)_M$ |
| | $b(11111111111111)_M$ | $d(u10u11u00)_M$ | $g(u10u11u00u1u)_M$ |
| 5 | $a(uuuuuuuuuuuuuu)_M$ | $c(1uu101u01)_M$ | $e(011010011110)_M$ |
| | $b(11111111111111)_M$ | $d(01uu1u010)_M$ | $g(110110010101)_M$ |
| 6 | $a(00000000000000)_M$ | $c(0uu1100u0)_M$ | $e(011000001111)_M$ |
| | $b(00000000000000)_M$ | $d(0u10u10u1)_M$ | $g(111001011010)_M$ |

(2) The TOP reconstructs the corresponding ternary logic calculators (the processor allocation is shown in Table 8);

(3) Original data $b$, $d$, and $g$ are organized into data frames and sent to the main optical path coding area;

(4) Original data $a$, $c$, and $e$ are organized into data frames and sent to the control optical path coding area;

(5) The decoder is started to obtain output light information of all bits of the processor and is converted into corresponding electrical information to form output data of the processor.

The main content of the experiment is to verify the correctness of the ternary logical naming convention on the TOC; the TOP can reconstruct the corresponding ternary calculators according to the ternary logical standard name accurately and these calculators are correct.

**Table 8**. The processor bit allocation for the adder.

| | D (0,-1,1)[243005] | D(0,-1,1)[420110] | D(0,-1,1)[002004] | D(0,-1,1)[241001] | D (0,-1,1)[243005] |
|---|---|---|---|---|---|
| $f1$ | 1-14 | 15-28 | 29-43 | 44-58 | 59-74 |
| $f2$ | 75-83 | 84-92 | 93-102 | 103 -112 | 113-123 |
| $f3$ | 124-135 | 136-147 | 148-160 | 161-173 | 174-187 |

## 4.4. Experimental results and analysis

The theoretical values of the experimental examples are shown in Table 9. The images output by the optical processor during the experiment are consistent with the theoretical values. This article takes the first set of experimental data of $f1$ to illustrate: Figure 6 is the output of the first set of experimental examples of $f1$ and the corresponding theoretical analysis diagram is shown in Figure 7, where the brightest point is the vertically polarized light (V light state, expressed by 1), the second bright spot is the horizontally polarized light (H light state, expressed by u), and the other is the no-light state (expressed by 0). The blue part in Figure 8 is the first set of experimental results for $f1$. In units of rows, from right to left, from bottom to top, every three adjacent pixels are one bit and we can read a string: 00000010110u010, which is the resulting value expressed in MSD form. Then we apply the method of converting MSD to a decimal number and convert $(00000010110u010)_M$ to $(346)_D$, which is consistent with the theoretical results. Experiments show that the ternary logical naming convention proposed in this paper can be applied to the TOC, and reconstructed logical calculators can calculate the data accurately.

**Table 9**. Theoretical results of experimental use cases.

| Item | $f1$ | $f2$ | $f3$ |
|---|---|---|---|
| 1 | $(00000010110u010)_M$ | $(01011u0000)_M$ | $(01001u10000100)_M$ |
| 2 | $(100110011100000)_M$ | $(0000000000)_M$ | $(00u01u1uu10100)_M$ |
| 3 | $(000000010011100)_M$ | $(uuuuuuuuu0)_M$ | $(01uu1u10u0011u)_M$ |
| 4 | $(111111111111110)_M$ | $(0000011011)_M$ | $(0000010u1u10u1)_M$ |
| 5 | $(000000000000000)_M$ | $(0010001111)_M$ | $(01010000110011)_M$ |
| 6 | $(000000000000000)_M$ | $(00uu0uu0uu)_M$ | $(01010001101001)_M$ |

## 5. Conclusion

This paper proposes a design method for a ternary logical naming convention: combining the CNV and NTR to indicate the ternary logical operation rule. The design principle and design specification are introduced in detail, and then the MSD adder is taken as an example to illustrate the application of the naming convention on TOCs. Through the test experiment, it is verified that the TOP can reconstruct the corresponding logic calculators according to the naming convention and calculate correctly. Compared with traditional ternary logic truth tables, the method proposed in this paper is simple to write and easy to identify. As scholars continue to delve into the MSD adder of TOCs, a large number of ternary logical transforms are used. Applying this naming method can reduce the repetitive work during the research process. It is also preparing for TOCs to enter the field of numerical calculation.

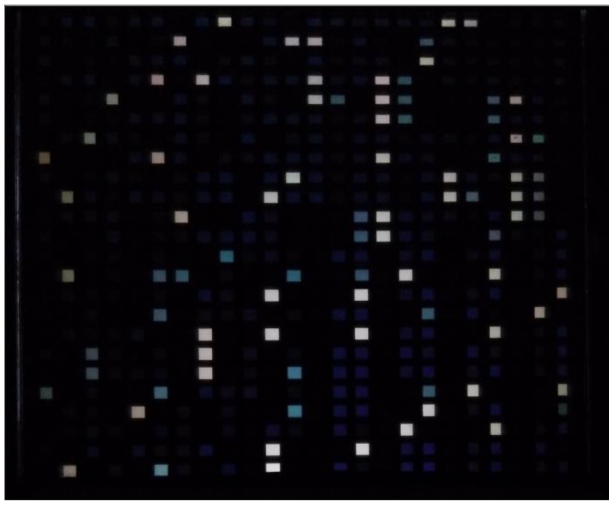
**Figure 6**. Results of the first set of test cases for $f1$.

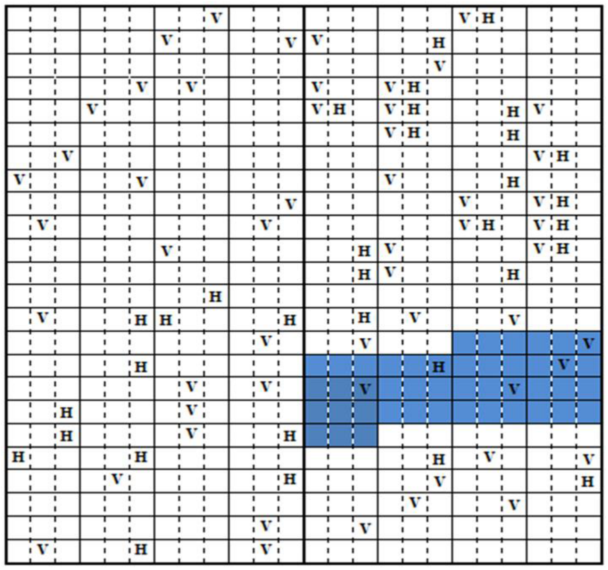
**Figure 7**. Theoretical output diagram.

With the rapid development of artificial intelligence theory, technology, and applications, the complexity of the problems people want to solve is dramatically increasing and the logical characteristics of practical problems have long exceeded the scope of binary logic. New logic concepts such as fuzzy logic, uncertainty reasoning, and rough set logic are also increasing, so multivalued logic has become a very hot topic of discussion. However, the way to express multivalued logic rules has relied on truth tables, which has brought difficulties to the characteristics of academic communication and judgment logic problems. The design principle and usage specification of the ternary logical naming convention proposed in our paper can be directly extended to any multivalued logic operations to form a standard name for various multivalued logic operations.

## Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No. 61572305, 61073049), the Innovation Program of the Shanghai Municipal Education Commission (Grant No. 13ZZ074), the Innovation

Foundation of the Second Institute of China Aerospace Science and Technology Corporation, the Natural Science Foundation of Shanghai (Grant No. 15ZR1415400), and the Special Research Program of Shanghai (Grant No. 15700500400).

## References

[1] Zadehl LA. Outline of a new approach to the analysis of complex systems and decision processes. IEEE Transactions on Systems, Man and Cybernetics 1973; SMC-3 (1): 28-44.

[2] Xu Y, Zhang YW, Wang H. A conflict-eliminating approach for emergency group decision of unconventional incidents. Knowledge-Based Systems 2015; 83: 92-104.

[3] Ben-Arieh D, Chen Z. Linguistic group decision-making: opinion aggregation and measures of consensus. Fuzzy Optimization & Decision Making 2006; 5 (4): 371-386.

[4] Herrera-Viedma E, Alonso S, Chiclana F. A consensus model for group decision making with incomplete fuzzy preference relations. IEEE Transactions on Fuzzy Systems 2007; 15 (5): 863-877.

[5] Avizienis A. Signed digit number representation for fast parallel arithmetic. IRE Transactions on Electronic Computers EC 1961; 10 (3): 389-400.

[6] Drake BL, Bocker RP, Lasher ME, Patterson RH, Miceli WJ. Photonic computing using the modified signed digit number representation. Optical Engineering 1986; 25 (1): 38-43.

[7] Bocker RP, Drake BL, Lasher ME, Henderson TB. Modified signed-digit addition and subtraction using optical symbolic substitution. Applied Optics 1986; 25 (15): 2456-2457.

[8] Cherri AK, Karim MA. Modified signed-digit arithmetic using an efficient symbolic substitution. Applied Optics 1988; 27 (18): 3824-3827.

[9] Casasent D, Woodford P. Symbolic substitution modified signed-digit optical adder. Applied Optics 1994; 33 (8): 1498-1506.

[10] Cherri AK, Alam MS. Recoded and nonrecoded trinary signed-digit adders and multipliers with redundant-bit representations. Applied Optics 1998; 37 (20): 4405-4418.

[11] Fyath RS, Alsaffar AAW, Alam MS. Optical two-step modified signed-digit addition based on binary logic gates. Optics Communications 2002; 208 (4): 263-273.

[12] Alam MS. Parallel optical computing using recoded trinary signed-digit numbers. Applied Optics 1994; 20: 4392-4397.

[13] Alam MS, Karim MA, Awwal AAS, Westerkamp JJ. Optical processing based on conditional higher-order trinary modified signed-digit symbolic substitution. Applied Optics 1992; 31 (26): 5614-5621.

[14] Huang HX, Masahide I, Toyohiko Y. Classified one-step modified signed-digit arithmetic and its optical implementation. Optical Engineering 1996; 35 (4): 1134-1140.

[15] Alam MS. Parallel optical computing using recoded trinary signed-digit numbers. Applied Optics 1994; 33 (20): 4392-4397.

[16] Jin Y, Shen YF, Peng JJ, Xu SY, Ding GT et al. Principles and construction of MSD adder in ternary optical computer. Science China Information Sciences 2010; 53 (11): 2159-2168.

[17] Pan L, Shen YF. Implementation and principle of improved MSD carry-free adder for ternary optical computer. Computer Science 2011; 38 (12): 293-296.

[18] Shen YF, Pan L, Jin Y, Peng JJ, Jiang B. One-step binary MSD adder for ternary optical computer. Science China Information Sciences 2012; 42 (7): 869-881.

[19] Peng JJ, Shen R, Jin Y, Sheng YF, Luo S. Design and implementation of modified signed-digit adder. IEEE Transactions on Computers 2014; 63 (5): 1134-1143.

[20] Shen YF, Pan L. Principle of a one-step MSD adder for a ternary optical computer. Science China Information Sciences 2014; 57 (1): 1-10.

[21] Jiang JB, Zhang XF, Shen YF, Ouyang S, Zhou SQ et al. Design and implementation of SJ-MSD adder in ternary optical computer. Acta Electronica Sinica (in press) (in Chinese).

[22] Ouyang S, Peng JJ, Jin Y, Shen YF, Liu XM et al. Structure and theory of dual-space storage for ternary optical computer. Science China Information Sciences 2016; 46 (6): 743-762.

[23] Jin Y, Ouyang S, Song K, Shen YF, Peng JJ et al. Management of many data bits in ternary optical computers. Scientia Sinica Informationis 2013; 43 (3): 361-373.

[24] Yan JY, Jin Y, Zuo KZ. Decrease-radix design principle for carrying/borrowing free multi-valued and application in ternary optical computer. Scientia Sinica Informationis 2008; 51 (10): 1415-1426.

[25] Peng JJ, Shen R, Jin Y, Shen YF, Luo S. Design and implementation of modified signed-digit adder. IEEE Transactions on Computers 2014; 63 (5): 1134-1143.

[26] Wang HJ, Jin Y, Ouyang S. Design and implementation of a 1-bit reconfigurable ternary optical processor. Chinese Journal of Computers 2014; 37 (7): 1500-1507 (in Chinese).

[27] Jin Y, He HC, Lu YT. Ternary optical computer architecture. Physica Scripta 2005; T118: 98-101 (in Chinese).

[28] Jin Y, He HC, Ai LR. Lane of parallel through carry in ternary optical adder. Science in China Series F-Information Sciences 2005; 48 (1): 107-116 (in Chinese).

[29] Yi J, Wang HJ, Ouyang S, Zhou Y, Shen YF et al. Principles, structures, and implementation of reconfigurable ternary optical processor. Science China Information Sciences 2011; 54 (11): 2236-2246.

[30] Jin Y, Ouyang S, Song K, Shen YF, Peng JJ et al. Management of many data bits in ternary optical computers. Scientia Sinica Information Sciences 2013; 43 (3): 361-373.

[31] Li S, Jiang JB, Wang ZH, Zhang HH. Basic theory and key technology of programming platform of ternary optical computer. International Journal for Light and Electron Optics 2018; 178: 327-336.

[32] Li S, Jin Y. Simple structured data initial SZG files generation software design and implementation. International Conference of Wireless Communication Sensory Networks 2016; 44: 383-388.

[33] Gao H, Jin Y, Song K. Extension of C language in ternary optical computer. Journal of Shanghai University (Nature Science) 2013; 19 (3): 280-285 (in Chinese).

[34] Zhang Q, Jin Y, Song K, Gao H. MPI programming based on ternary optical in supercomputer. Journal of Shanghai University (Nature Science) 2014; 20 (2): 180-189 (in Chinese).

[35] Li S, Jin Y, Liu YJ, Zhou SQ. Initial SZG file generation software of the ternary optical computer. Journal of Shanghai University (Nature Science) 2018; 24 (2): 181-191 (in Chinese).