Optics Communications 470 (2020) 125750

Contents lists available at ScienceDirect

Optics Communications

journal homepage: www.elsevier.com/locate/optcom

Design and implementation of a three-lane CA Traffic Flow model on ternary optical computer

Zhang Sulan [a,b], Shen Yunfu [b,*], Zhao Zheyu [b]

[a] College of Mathematics, Physics and Information Engineering, Jiaxing University, Jiaxing, Zhejiang, 314000, China
[b] School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China

ARTICLE INFO

Keywords:
Cellular automata
Traffic flow model
Ternary optical computer
MSD adder

ABSTRACT

The ternary optical computer (TOC) has the characteristics of numerous bits, bit-wise allocation, bit function reconstruction, parallel calculation, and easy extension of the processor. With the successful running of the adder on a 192-bit prototype system SD16 (an abbreviation of Shanghai Daxue's machine 2016) of the TOC in Shanghai University, the theory and technology of the reconfigurable optoelectronic hybrid processor have matured and the great advances have been made in this field. In this paper, based on cellular automaton, a three-lane traffic flow model is studied and applied on a ternary optical computer. The three-lane traffic flow model is designed and the steps for implementing the model on SD16 are provided in detail. An example of the three-lane traffic flow model is given to verify the correctness of the model and the feasibility of the implementation method. It takes advantage of the numerous data bits, reconfigurable operation bits, and parallel computing of the TOC.

1. Introduction

Cellular automata (CA) is a grid dynamic system in which time, space and state are discrete, spatial interactions and causality relationship in time are local. Each variable takes only a limited number of states. The model meeting these rules can be regarded as a cellular automaton model. It generally does not have a strictly defined physical equation or function form, but consists of a series of rules.

CA was proposed to simulate the self-replication function of biological systems by the father of computers, von Neumann, in the early 1950s [1]. In 1986, M. Cremer and J. Ludwig applied CA to the research of vehicle traffic [2]. The most basic one-dimensional model in the cellular automaton traffic flow model is the 184 CA model [3,4], which uses elements from grid point chain of one-dimension to simulate vehicles on the road. The state of the vehicle at the next moment is completely determined by the state of the vehicle itself and the two grid points before and after the vehicle. In 1992, Nagel K. and Schreckenberg M. proposed a cellular automaton model (NS model), considering the possibility of a car's gradually restricted forward shifting and random reverse direction shifting [5].

In recent years, with the increasing demand for high-performance computing in the social economy, people's enthusiasm for low-power, high-performance computers is becoming more and more intense. It makes researchers pay more and more attention to the research of new type of computers, such as quantum computers [6,7], biological computer [8], optical computer, etc. [9–11].

Optical computers have certain advantages in data width and parallel carry-free addition, etc. Therefore, they have attracted much attention. The research in photoelectric hybrid ternary optical computer (TOC) in Shanghai University is a more promising field. On March 18, 2017, the research team successfully operated a 36 bit MSD adder based on T, W, T′, W′, and T2 transformations (referred to as TM-MSD adder) [12], marking that the theory and technology of the optical reconfigurable photoelectric hybrid processor have matured, and the manufacturing technology has been feasible [9–11,13]. The TOC has more advantageous than traditional electronic computer systems in solving problems that require more resources and repetitive computations.

At present, there are more and more researches on the potential applications for the TOC. The algorithm design of vector matrix multiplication [14,15] is studied, iterative division algorithm for MSD number is designed and implemented [16]. The FFT algorithm is studied by using huge data bits of optical calculations with less clock cycles [17]. The implementation method of the DFT algorithm is studied by using bitwise allocation and reconfigurability of processor bits of the TOC in the literature [18].

From the characteristics that all the cells of a cellular automaton can be calculated in parallel, it is obviously advantageous to apply it to the application research with the TOC.

* Corresponding author.
E-mail address: yfshen@shu.edu.cn (Y. Shen).

https://doi.org/10.1016/j.optcom.2020.125750
Received 18 February 2020; Received in revised form 12 March 2020; Accepted 14 March 2020
Available online 19 March 2020

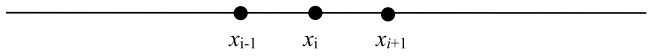

Fig. 1. One-dimension CA.

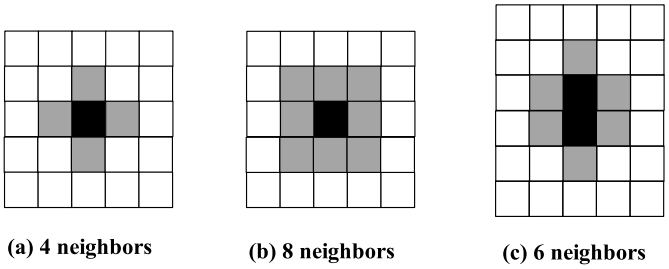

Fig. 2. Typical two-dimension CA.

Table 1
One set of logical transformations of carry-free MSD addition.

S_1				S_2				J_1			J_2			J_3		
b	a			b	a			s_1	s_2		s_1	s_2		j_2	j_1	
	0	1	u		0	1	u		0	u		0	u		0	u
0	0	1	0	0	0	u	u	0	0	u	0	0	1	0	0	u
1	1	1	0	1	u	0	0	1	0	0	1	1	0	1	1	0
u	0	0	u	u	u	0	0	u	0	u	u	u	0	u	u	0

Table 2
Simplified Logical transformations of SJ-MSD adder.

S_1				S_2				J_{12}				J_3		
b	A			b	a			s_1	s_2			j_2	j_1	
	0	1	U		0	1	u		0	V	H		0	u
0	0	1	0	0	0	u	u	0	0	u	1	0	0	u
1	1	1	0	1	u	0	0	1	1	0	0	1	1	0
u	0	0	u	u	u	0	0	u	u	u	0	u	u	0

2. The related theory of CA and the TOC

2.1. Cellular automata

The cellular automaton consists of basic elements and cell states. The basic elements have four parts: cell, cell space, neighbor, and transfer rule (or function). Therefore, the CA can be regarded as consisting of a cell space and a transformation function defined on the space [4,5].

The cell is the most basic component, distributing as lattice points in discrete 1-dimension, 2-dimension or high dimensional space. It has a discrete and finite state space. Given a transfer function f of the cellular automaton model and the initial values, the changes of the cells at the next moment are related to the states of the cells at the current time and its neighboring positions. The cell states can be in binary forms, such as $(0, 1)$, (live, dead), (black, white), etc., or in ternary forms, or in some finite integer set S or in a certain interval. A set of spatial lattice points of a cell distribution is called as a cell space. The boundary rules of the cellular space are generally periodic, fixed and reflective, and so on. The CA converts with discrete time values. If the time step is $dt = 1$, then $t + 1$ is the next time. $t = 0$ is the initial time.

One-dimension CA is a linear structure, as shown in Fig. 1, and the transformation function at each position is $x_i' = f(x_{i+1}, x_i, x_{i-1})$, which has 2^8 states.

Two-dimension CA is a grid structure, as shown in Fig. 2, and its transfer function is $x_{ij}' = f(N_{ij})$, where N_{ij} is the set of neighbor information of x_{ij}.

The basic operation of the CA model M of scale m × n is as follows:

(1) For some kind of grid division of M, the next state x_{ij}' of each cell x_{ij} depends on the states of some cells adjacent to $x_{ij}, x_{ij}' = f(N_{ij})$, where N_{ij} is a set of neighbor elements ;
(2) The states of all cells can be calculated in parallel;
(3) Evolutionary calculations finish until entering a stable state or the specified step k.

Since the next state of a cell is related to its neighbor states, its transfer rule is defined in the local scope of the space. Therefore, the neighbor rules must be defined firstly.

The transfer rule is a dynamic function that determines the states of the cells at the next moment based on the current states of the cells and their neighbors, that is, a state transfer function. It can be expressed as formula (1):

$$S_i^{t+1} = f(S_i^t, S_N^t) \tag{1}$$

Where S_i^t is the state of the cell i at time t and S_N^t is the state of the neighbor N. Here, f is called as a local map or a local rule of a CA.

Therefore, the CA system M can be represented by a tuple M = (L_d, S, N, f), where L_d is a cell space, S is a finite state set, d is a spatial dimension, and N represents a set of the combination of all cells in the neighborhood (including the central cell), f is the local conversion rule (or function). In this paper, we selects the number of neighbors $3^d - 1$ according to the actual situation.

2.2. The principle of SJ-MSD adder

As early as in 2010, the research team began to consider the problem of parallel carry-free modified signed-digit (MSD) addition. By examining the three-step TW-MSD addition transformation T, W, T_1, W_1, T_2, Prof. Yunfu Shen obtained a set of simpler parallel carry-free three-step MSD addition transformation rules in 2013, as shown in Table 1. In 2016, Shen described the characteristics of the parallel carry-free three-step MSD addition, and gave sets of all the transformation rules that satisfy parallel computing of the MSD addition in three-step thoroughly. These MSD addition rules provide a theoretical basis for the operator selections of the TOC. Some were applied in the design of actual optical computer processors.

The Shen–Jiang modified signed-digit (SJ-MSD) adder is named after Yunfu Shen and Ph.D Jiabao Jiang. It is based on the MSD addition rules shown in Table 1 by combining the sub-tables J_1 and J_2, as shown in sub-table J_{12} of Table 2.

Here, we use the vertical polarized light (V), horizontal polarized light (H) and null light in optical implementation.

SJ-MSD addition consist of 5 ternary logical transformations (SJ transformations) and a set of operation rules (SJ rule). The 5 transformations are in turn the transformation S_1, S_2, J_1, J_2, J_3 shown in Table 1. SJ rule for k-bits MSD numbers a and b is described as follows [19]:

Let a and b be two MSD numbers, $a = a_{k-1} \cdots a_1 a_0, b = b_{k-1} \cdots b_1 b_0$.

Step 1: Implementing S_1transformation on data a and b bit by bit, adding one 0 behind the transformation result s_1' to obtain $(k + 1)$ bits value s_1. At the same time, implementing S_2 transformation on data a and b bit by bit too, adding a 0 in front of the transformation result s_2' to obtain $(k + 1)$ bits value s_2.

Step 2: Implementing J_1 transformation on the low k bits both s_2 and s_1 bit by bit, adding a 0 behind the result j_1' to obtain $(k + 1)$ bits value j_1. At the same time, implementing J_2 transformation on s_2 and s_1 bit by bit, and obtain $(k + 1)$ bits value j_2.

Step 3: Implementing J_3 transformation on j_1 and j_2 bit by bit, obtaining $(k + 1)$ bits value $j_3 = a + b$.

Then s is the sum of a and b.

A SJ-MSD adder in SD16 is constructed according to Table 2 [19]. We omit its details here. In this way, only $4n + 2$ processor bits are needed to complete the addition of the n bits MSD number a and b.

3. Three-lane CA traffic flow model

The multi-lane traffic flow model based cellular automaton evolved from the NS model [20–22]. The three-lane CA traffic road system consists of three parallel and co-directional lanes, representing by lanes 1, 2, and 3, each lane is composed of a one-dimensional discrete lattice chain of length L, and each grid on the lattice chain represents a cell, each cell has two states: 0 or 1, where 0 means null and 1 means that it is occupied by a car. The maximum speed of the vehicle in each lane is V_{max}, and the motion direction of the vehicle on each lane is fixed (such as from south to north, or from left to right). The speed of each vehicle is determined by the interval [0, V_{max}] according to the real situation. We use the following variables in the paper:

$V_{ji}(t)$: the speed of the ith car from the time $t-1$ to t in the jth lane;

$d_{ji}(t)$: the distance between the ith car and the front car at time t on the jth lane;

$x_{ji}(t)$: the position of the ith car on the jth lane at time t;

K_0: the probability of randomly generating a car;

K_1: the acceleration probability of each vehicle;

K_2: the random slowing probability;

K_3: changing lane probability for each lane;

V_h: the driver's desired speed.

$fd_{j,i}$: the front distance of the vehicle on lane j in corresponding position relative to the vehicle in lane i;.

$bd_{j,i}$, $bV_{j,i}$: the rear distance and the speed of the nearest car on lane j in corresponding position relative to the vehicle in lane i respectively.

In the evolution of the model, each evolution is divided into the following three steps.

(1) Update principles of the speed for each vehicle
Deceleration:

If $V_{ji}(t) \geq d_{ji}(t)$, then $V_{ji}(t+1) = d_{ji}(t) - 1$, according to the probability K_2;
or $V_{ji}(t+1) = d_{ji}(t)$, according to the probability $1 - K_2$.
If $V_{ji}(t) < d_{ji}(t))$ and $V_{ji}(t) = V_{max}$, then $V_{ji}(t+1) = V_{max} - 1$, according to the probability K_2;
or $V_{ji}(t+1) = V_{max}$, according to the probability is $1 - K_2$.

Acceleration:

If $V_{ji}(t) < V_{max}$ then $V_{ji}(t+1) = V_{ji}(t)$, according to the probability K_1;
or $V_{ji}(t+1) = V_{ji}(t) + 1$, according to the probability $1 - K_1$.

(2) Update rules of each vehicle location
$x_{ji}(t+1) = x_{ji}(t) + V_{ji}(t+1)$;

(3) Change lane rules for each vehicle

After the vehicle running in each lane, it will change lanes according to the driving needs. But it must meet the overtaking principle and safety principles.

(1) The changing lane rule of the vehicle in lane 1. When the distance d_{1i} of the current ith car of the first lane is smaller than the expected value V_h of the driver, the changing lane consciousness is generated. If d_{1i} is smaller than the distance $fd_{2,1}$ of the corresponding position of the second lane in front, and the rear distance $bd_{2,1}$ of the second lane in corresponding position is greater than or equal to the speed $bV_{2,1}$ of the nearest neighbor vehicle behind in the second lane, the current vehicle in the first lane is transferred to the second lane with the probability q, keeping its current speed.

(2) The changing lane rule of the vehicle in lane 2. When the distance d_{2i} of the current position of the ith car in the second lane is less than the expected value V_h, the changing lane consciousness is generated.
(1) Turn-left rule: If $d_{2i} < fd_{1,2}$ and $bd_{1,2} > bV_{1,2}$, that is, if d_{2i} is smaller than the distance $fd_{1,2}$, when the rear distance $bd_{1,2}$ of the corresponding position of one lane is greater than or equal to the speed $bV_{1,2}$ of the nearest neighbor vehicle behind on the first lane, the current vehicle on the second lane can be transferred to the first lane with the probability q, keeping its current speed.
(2) Turn-right rule: If $d_{2i} \geq fd_{1,2}$, and $fd_{3,2} > d_{2i}$, $bd_{3,2} > bV_{3,2}$, the current vehicle in the second lane can be transferred to the third lane with the probability q, keeping its current speed.

(3) The changing lane rule of the vehicle in lane 3. When $d_{3i} < V_h$, the changing lane consciousness is generated. If $d_{3i} < fd_{2,3}$, and $bd_{2,3} \geq bV_{2,3}$, the current vehicle in the third lane can be transferred to the second lane with the probability q. After the vehicle turning to the second lane, the vehicle speed remains unchanged.

4. Algorithm design of three-lane traffic flow on the TOC

4.1. Three-lane traffic flow design based on the TOC

Giving a section of road with the length L kilometers, suppose that one kilometer is regarded as one cell and the maximum speed is V_{max}. Denote the position and its speed of the ith car on the jth lane by x_{ji}, and V_{ji} respectively, $0 \leq x_{ji} \leq L, 0 \leq V_{ji} \leq V_{max}$. Let $L' = L + V_{max}$ and $w = \lceil \log_2(L+V_{max}) \rceil$. L' is the maximum length needed to be considered for this CA.

Any position between 0 and L' can be represented by a binary number of w bits.

In the CA traffic flow model, three lanes have 3L grid points, which the position of each grid point is regarded as a cell. The new status of every cell involves at most $2 + V_{max}$ cells, as shown in Fig. 3. The position, speed and displacement of the vehicle need to be calculated. They just need addition. So each car is equipped with an adder of w bits.

Suppose that the ternary optical processor has a total of m bits. According to the calculation process of the SJ-MSD adder, the MSD adder of w bits only needs $4w + 2$ processor bits. Therefore, we can construct $p = \lceil m/(4 * w + 2) \rfloor$ small adders within the m-bit ternary optical processor. That is, the position states of p vehicles can be calculated simultaneously. For the p adders of w bits, the task management software of host computer performs the allocation of processor bits for each adder. Let G be the start position of the first adder in the processor.

Denote $Pos[0] = G$, $Pos[i + 1] = Pos[i] + 4w + 2$, $i = 0, 1, \ldots, p - 2$. Then the start bit of the ith adder is Pos[i].

Suppose that there are num vehicles on the road. Then it will need $\lfloor num/p \rfloor$ calculation to update the information of these vehicles, and each calculation is completed in parallel. The image information calculated of the p cars is stored in a BUF. The updated position information of all the vehicles is obtained after $\lfloor num/p \rfloor$ calculation. The information in the register is transmitted to the upper computer for decoding, and the information of all vehicle positions is obtained.

4.2. Implementation process of three-lane traffic flow on a ternary optical processor

SD16 is a photoelectric hybrid computer, and the structure of computing-data model is shown in Fig. 4 [23]. It can be seen from Fig. 4 that it consists of two parts: an upper computer (a master computer) and a lower computer (a slave computer). The master computer is a PC with a traditional operating system, which is responsible for running task management software and communicating with users. The task management software is responsible for generating the reconstruction information and calculation data information of the slave computer (the optical processor) [24]. The reconstruction information makes the TOC operator to reconstruct the various calculators required by the user,

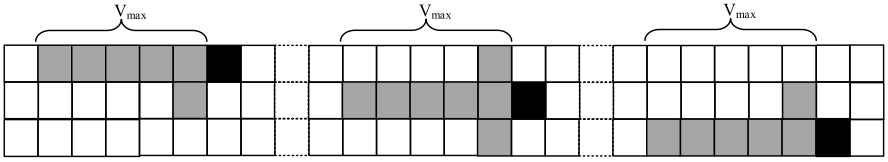

Fig. 3. Three-lane traffic flow Model CA.

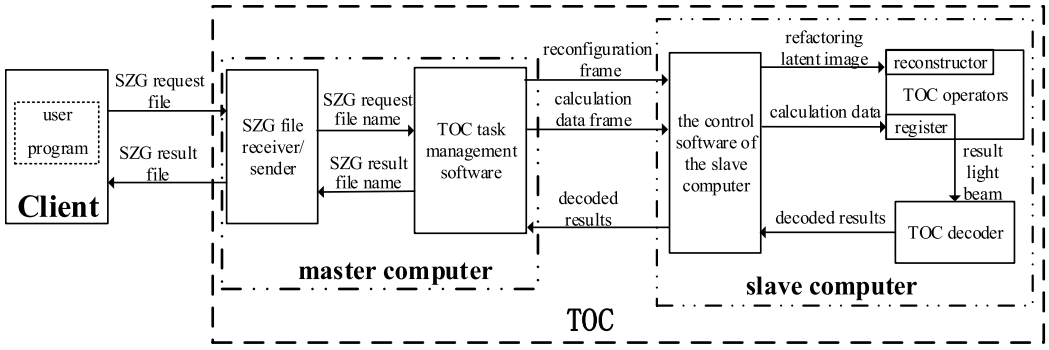

Fig. 4. Computing-data model.

and the calculation data information is used for calculation on the TOC operator. The calculated result beam is sent to the TOC decoder by the slave computer control software to obtain the user's binary result data. The calculation result is returned to the user through the task management software [23–25].

In the three-lane CA traffic flow model, each vehicle has three attributes: acceleration, displacement, and lane change. The master computer is responsible for generating the vehicle grid position information that the salve computer needs to calculate, and sends the grid information to the salve computer for calculation. The salve computer receives the calculation information of each grid point transmitted from the master computer, calculates the speed and displacement of the vehicle, obtains the result information, and returns the result information to the master computer for decoding to obtain the binary result data.

Therefore, only the adder needs to be designed on the TOC, and the lane change is completed by the master computer. The three-lane traffic flow design based on the ternary optical computer, the workflow is as follows:

(1) Constructs the whole road L with the task management software in the master computer;
(2) Determine the bit number w of the required optical adder according to the road length L and the maximum speed V_{max};
(3) Determine the number p of small adders with w bits that can be reconstructed in optical processor by $p = \lceil m/(4 * w + 2) \rceil$;
(4) Allocate processor bits for the p adders in master computer, and complete the reconstruction of these p adders;
(5) Generate a vehicle (car) randomly, and initialize its speed v, the lane and the displacement S. And arrange the vehicle randomly on lane 1, 2 or 3 according to the probability;
(6) Assign an adder to the newly generated vehicle;
(7) Find all the vehicles on the road, and for each vehicle at time t determine the information of distance before and after it, speed and position;
(8) Determine if the vehicle is shifting or not, and determine its acceleration according to the information of distance before and after it, speed and position at time t;
(9) Send the speed and acceleration of each vehicle at time t to the optical processor of the slave computer, and calculate the speed of each vehicle at time $t + 1$;
(10) Send the speed and position information of each vehicle at t+1 to the corresponding adder in the optical processor to calculate and complete the update of the corresponding position information. The position information of the p vehicles can be updated simultaneously, and these information can be stored in the buffer BUF. Repeat the process until all vehicles complete the update of the location information.
(11) Denote the storage information for empty position with null or 0;
(12) Send the image information of 3L positions to the master computer for decoding, and finally generate a binary result of position information.

5. Implementation and analysis of three-lane traffic flow problem on SD16

5.1. Brief introduction to SD16

The TOC uses two polarized light with orthogonal polarization states and a null light state to express information, and uses such device as liquid crystal pixel array to rotate the polarization direction of light in order to perform light state conversion. It can have millions of processor bits to realize various operations.

SD16 is a ternary optical reconfigurable photoelectric hybrid processor developed by Shanghai University in 2016. SD16 consists of two parts: the master computer and the slave computer. The master computer is a traditional electronic computer with 64 bit windows 7 operating system, Intel(R) Core(TM) i7-4790 CPU @ 3.60 GHz, 4 GB. The slave computer is ternary optical processor. The processor bits of the slave computer can be divided into several sections according to the needs of the user, and each section can be used to run different programs independently; each processor bit can be changed its calculation function at any time according to the needs of the user. SD16 can add 64 basic operation modules, one of which has 192 processor bits, reaching 12288 processor bits. At present, each processor bit can be reconstituted into a bit of any three-valued (including two-valued) logical operator, or 4n + 2 processor bits can be used to construct an n-bit parallel carry-free SJ-MSD adder. For a module of 192 bit SD16, it can constitute up to 47 bit three-step SJ-MSD adder. The appearance of the prototype SD16 is shown in Fig. 5.

In the slave computer of SD16, we use a liquid crystal board LCD with 576 pixels to serve as a processor, and the pixel bits are arranged in the form as Fig. 6, where three adjacent pixels in the same row form a processor bit. To observe the result data easily, we divide the LCD board

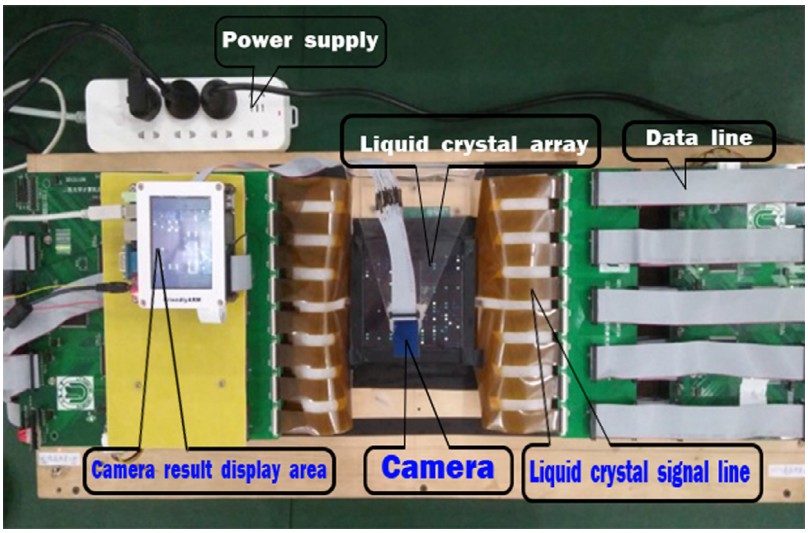

Fig. 5. The prototype SD16 of the TOC.

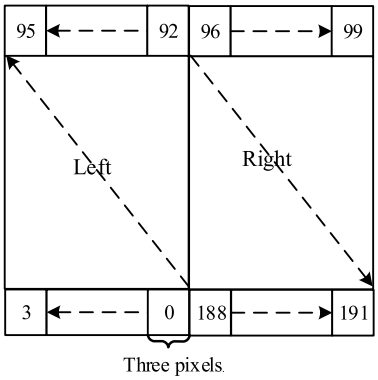

Fig. 6. Liquid crystal divisions of SD16.

into two parts: the left and right parts, and number each processor bit from 0 to 191. The number of each pixel of the LCD is shown in Fig. 6. Each pixel can output no light, horizontally polarized light or vertically polarized light. According to our design for the processor, the three adjacent pixels in the same row are denoted by W (null light), V (vertically polarized light) and H (Horizontally polarized light) respectively. There is only one output (W, V or H) and it cannot have two outputs V and H at the same time for the same processor bit.

5.2. Implementation of an instance on SD16

As an example, suppose the length of a section of highway is L = 80 km and the maximum speed V_{max} is set to 12 units. Hence, the position range of each vehicle is $0 \le x_{ji} \le 80$ and the speed range of each vehicle is $0 \le V_{ji} \le 12$, and $L' = L + V_{max} = 92$ and $w = \lceil \log_2 L' \rceil = 7$. It means that it requires 7 bits to represent the displacement value.

In our SJ-MSD adder, J1 and J2 transformations can be finished in different pixels of the same processor bit [19]. Now each position on the road is regarded as a cell which is corresponding to a small MSD adder with 7 bits. Here, we design all the small SJ-MSD adders with 8 bits. So, each adder needs $4 * 8 + 2 = 34$ processor bits. We can equip with $\lceil 192/34 \rfloor = 5$ adders, which can calculate the position status of 5 vehicles at the same time.

In this example, the processor bit allocation on SD16 is as follows. Let G = 0. Then Pos [0] = 0, Pos[i + 1] = Pos[i] + 34, i = 0, 1, ..., p − 2. So the start position of the ith adder is Pos[i]. The start positions of the five small adders are 0, 34, 68, 102, and 136 respectively. After the allocation of processor bits, the configuration image of processor bits in SD16 is formed according to the specific function of each bit, as shown in Fig. 7. The five districts A, B, C, D, and E in this image are divided according to the corresponding adders. For example, eight AS_1 and AS_2 represent the 8 bit input data S_1 and S_2 of the first adder respectively. Similarly, AJ_1 and AJ_2 indicate the results of AS_1 and AS_2 after the transformation of J_1 and J_2, and AJ_3 is the result of AJ_1 and AJ_2 after $J_3$the transformation, that is, AJ_3 is the sum of AS_1 and AS_2.

5.3. Experimental results and analysis

The experiment is carried out on SD16. Assume that the vehicle is randomly generated in each lane with equal probability K_0. Here, we let $K_0 = 0.9$, $K_1 = 0.7$, $K_2 = 0.5$, $K_3 = 0.3$. And let $V_h = 3$, which means 30 km per hour, and $V_{max} = 5$, which means 50 km per hour. Before the beginning of the system, we construct five 8 bit adders, each of which has two input data: the augend a (the position of the vehicle at time t, $x_{ji}(t)$) and the addend b (the speed of the vehicle at time t, $V_{ji}(t)$). The result in this example represents the displacement value of the corresponding vehicle.

When time t is 0, the first car was randomly generated, and the first adder was assigned to the car at this time. The position $x_{31}(0)$ is 0, and the speed $V_{31}(0)$ is 1. Therefore, for the first vehicle, the augend a is 0, and the addend b is 1. The augends and addends of the other four adders are 0, that is, the input data of the five adders in the first screen are: (1) a is 0, b is 1; (2) a is 0, b is 0; (3) a is 0, b is 0; (4) a is 0, b is 0; (5) a is 0, b is 0. The screenshot of the vehicle running state on the road at the next moment is shown in Fig. 8, and the corresponding calculation result on the TOC is as shown in Fig. 9.

It can be seen from Fig. 7 that the result values are from the 25th to 33rd bits on the liquid crystal board, and the result is displayed from the high to the low. It can be seen from Fig. 8 that the result value of the first adder is $(00000001u)_{MSD} = 1$. Where the augend a is 0, the addend b is 1, 0 + 1 = 1, the calculation result is correct.

When time t is 1, the second car was randomly generated, at which time the second adder was assigned to the car. The position $X_{11}(1)$ of the vehicle 2 is 0, and the speed $V_{11}(1)$ is 2, which means that the augend a is 0 and the addend b is 2 for the adder of the second car. At the moment, for the first vehicle, the position $X_{31}(1)$ is 1, the speed $V_{31}(1)$ is 2. The augend and addend of the other three adder are all. That is, the input data a of the five adders at the second time are 1, 0, 0, 0, 0 respectively, the input data b is 2, 2, 0, 0, 0 respectively. The screenshot of electronic computer at the next moment is shown

W	V	H	W	V	H	W	V	H	W	V	H	H	V	W	H	V	W	H	V	W	H	V	W
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
	95	CJ_3		94	CJ_3		93	CJ_3	CJ_2	92	CJ_1		96	CJ_3		97	CJ_3		98	CJ_3		99	CJ_3
CJ_2	91	CJ_1	CJ_2	90	CJ_1	CJ_2	89	CJ_1	CJ_2	88	CJ_1		100	CJ_3		101	CJ_3		102	DS_1		103	DS_1
CJ_2	87	CJ_1	CJ_2	86	CJ_1	CJ_2	85	CJ_1	CJ_2	84	CJ_1		104	DS_1		105	DS_1		106	DS_1		107	DS_1
	83	CS_2		82	CS_2		81	CS_2		80	CS_2		108	DS_1		109	DS_1		110	DS_2		111	DS_2
	79	CS_2		78	CS_2		77	CS_2		76	CS_2		112	DS_2		113	DS_2		114	DS_2		115	DS_2
	75	CS_1		74	CS_1		73	CS_1		72	CS_1		116	DS_2		117	DS_2	DJ_1	118	DJ_2	DJ_1	119	DJ_2
	71	CS_1		70	CS_1		69	CS_1		68	CS_1	DJ_1	120	DJ_2	DJ_1	121	DJ_2	DJ_1	122	DJ_2	DJ_1	123	DJ_2
	67	BJ_3		66	BJ_3		65	BJ_3		64	BJ_3	DJ_1	124	DJ_2	DJ_1	125	DJ_2	DJ_1	126	DJ_2		127	DJ_3
	63	BJ_3		62	BJ_3		61	BJ_3		60	BJ_3		128	DJ_3		129	DJ_3		130	DJ_3		131	DJ_3
	59	BJ_3	BJ_2	58	BJ_1	BJ_2	57	BJ_1	BJ_2	56	BJ_1		132	DJ_3		133	DJ_3		134	DJ_3		135	DJ_3
BJ_2	55	BJ_1	BJ_2	54	BJ_1	BJ_2	53	BJ_1	BJ_2	52	BJ_1		136	ES_1		137	ES_1		138	ES_1		139	ES_1
BJ_2	51	BJ_1	BJ_2	50	BJ_1		49	BS_2		48	BS_2		140	ES_1		141	ES_1		142	ES_1		143	ES_1
	47	BS_2		46	BS_2		45	BS_2		44	BS_2		144	ES_2		145	ES_2		146	ES_2		147	ES_2
	43	BS_2		42	BS_2		41	BS_1		40	BS_1		148	ES_2		149	ES_2		150	ES_2		151	ES_2
	39	BS_1		38	BS_1		37	BS_1		36	BS_1	EJ_1	152	EJ_2	EJ_1	153	EJ_2	EJ_1	154	EJ_2	EJ_1	155	EJ_2
	35	BS_1		34	BS_1		33	AJ_3		32	AJ_3	EJ_1	156	EJ_2	EJ_1	157	EJ_2	EJ_1	158	EJ_2	EJ_1	159	EJ_2
	31	AJ_3		30	AJ_3		29	AJ_3		28	AJ_3	EJ_1	160	EJ_2		161	EJ_3		162	EJ_3		163	EJ_3
	27	AJ_3		26	AJ_3		25	AJ_3	AJ_2	24	AJ_1		164	EJ_3		165	EJ_3		166	EJ_3		167	EJ_3
AJ_2	23	AJ_1	AJ_2	22	AJ_1	AJ_2	21	AJ_1	AJ_2	20	AJ_1		168	EJ_3		169	EJ_3		170			171	
AJ_2	19	AJ_1	AJ_2	18	AJ_1	AJ_2	17	AJ_1	AJ_2	16	AJ_1		172			173			174			175	
	15	AS_2		14	AS_2		13	AS_2		12	AS_2		176			177			178			179	
	11	AS_2		10	AS_2		9	AS_2		8	AS_2		180			181			182			183	
	7	AS_1		6	AS_1		5	AS_1		4	AS_1		184			185			186			187	
	3	AS_1		2	AS_1		1	AS_1		0	AS_1		188			189			190			191	

Fig. 7. Configuration image of processor bits in SD16.

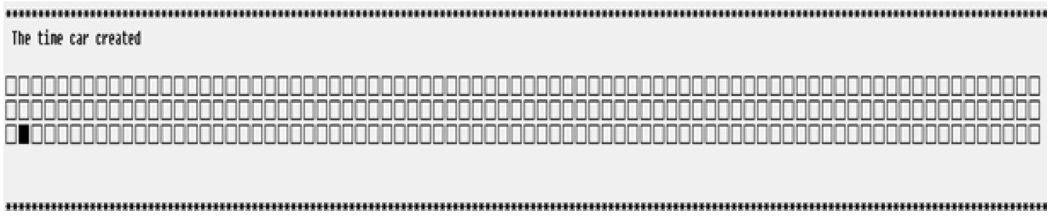

Fig. 8. Vehicle running status at the first time on electronic computer.

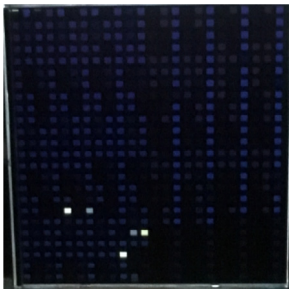

Fig. 9. Vehicle running status at the first time on the TOC.

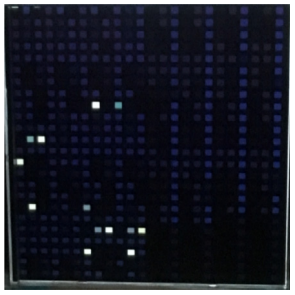

Fig. 11. Vehicle running status at the second time on the TOC.

in Fig. 10, and the corresponding calculation result on the TOC is as shown in Fig. 11.

It can be seen that the result values are from the 59th to 67th bits on the liquid crystal board, and the result is displayed from the

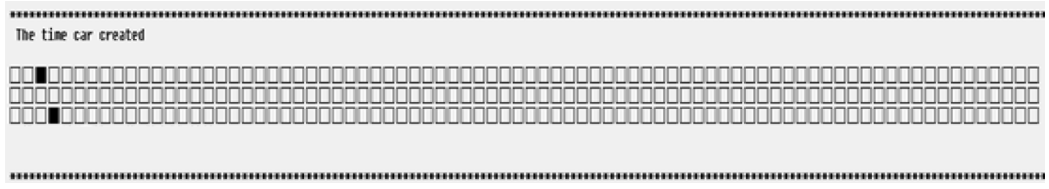

Fig. 10. Vehicle running status at the second time on electronic computer.

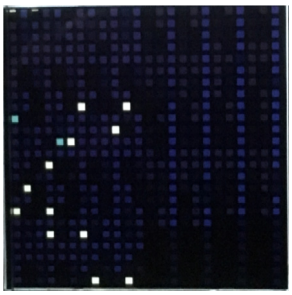

Fig. 12. Vehicle running status at the third time on the TOC.

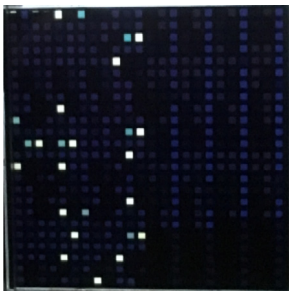

Fig. 13. Vehicle running status on the fourth time on the TOC.

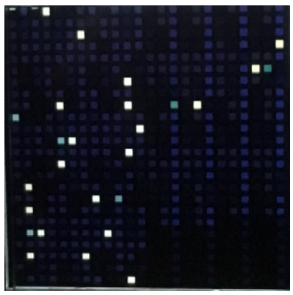

Fig. 14. Vehicle running status on the fifth time on the TOC.

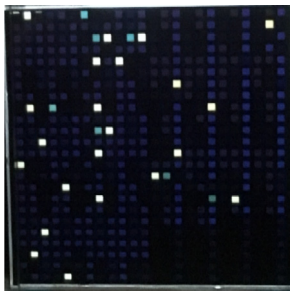

Fig. 15. Vehicle running status on the sixth time on the TOC.

high to the low. It can be seen from Fig. 10 that the result value is $(0000001u0)_{MSD} = 2$. At this time, the calculation result of the first adder is $(00000010u)_{MSD} = 3$, and the calculation result is correct.

When time t is 2, no new vehicle appears, at which time the vehicle 1 and the vehicle 2 are also corresponding to the adder 1 and the adder 2. The position $X_{11}(1)$ of the vehicle 2 is 2 at time 1, the speed $V_{11}(1)$ is 3. At this time, the augend of the adder for the second vehicle is 2, the addend is 3. The augend of the adder for the first vehicle is 3, $V_{31}(1)$ is 3 after the speed being accelerated. The augends and addends of the other three adders are all 0. That is, at this time, the input data a of the five adders at the third time are 3, 2, 0, 0, 0 respectively, the input data b is 3, 3, 0, 0, 0 respectively. The screenshot on liquid crystal board at the next moment is shown in Fig. 12.

As can be seen from Fig. 12, the result of the second adder is $(0000011u)_{MSD} = 5$. The result of the first adder is $(000000110)_{MSD} = 6$, and the calculation result is correct.

When time t is 3, the third car is randomly generated, the displacement a is 0, the speed b is 1. At this time, it is assigned a third adder. Similarly, the input data a of the five adders are 6, 5, 0, 0, 0 respectively, the input data b is 3, 2, 1, 0, 0 respectively. The screenshot on liquid crystal board at the next moment is shown in Fig. 13.

It can be seen that the result values of the third adder are from the 93rd to 101st bits on the liquid crystal board. As can be seen from Fig. 13, the results of the third adder is $(00000001u)_{MSD} = 1$, the results of the second adder is $(00000100u)_{MSD} = 7$, and the results of the first adder is $(00000101u)_{MSD} = 9$. These calculation values are correct.

When time t is 4, the fourth car is randomly generated, the displacement a is 0, and the speed b is 2. At that moment, it is assigned the 4th adder. Similarly, the input data a of the five adders are 9, 7, 1, 0, 0 respectively, the input data b is 3, 2, 1, 2, 0 respectively. The screenshot on the TOC at the next moment is shown in Fig. 14.

It can be seen that the result values of the third adder are from the 127th to 135th bits on the liquid crystal board. As can be seen from Fig. 14, the results of the fourth adder is $(0000001u0)_{MSD} = 2$, the results of the third adder is $(000000010)_{MSD} = 2$, the results of the second adder is $(00000101u)_{MSD} = 9$, and the results of the first adder is $(00001u100)_{MSD} = 12$. These calculation values are correct.

When time t is 5, the fifth car is randomly generated, the displacement a is 0, the speed b is 1, and the fifth adder is assigned to it at this time. Similarly, the input data a of the five adders are 12, 9, 2, 2, 0 respectively, the input data b is 4, 3, 1, 2, 1 respectively. The calculation result screenshot on the TOC at the next moment is shown in Fig. 15.

It can be seen that the result values of the third adder are from the 161st to 169th bits on the liquid crystal board. As can be seen from Fig. 15, the results of the fifth adder is $(00000001u)_{MSD} = 1$, the results of the fourth adder is $(000000100)_{MSD} = 4$, the results of the third adder is $(00000010u)_{MSD} = 3$, the results of the second adder is $(00001u100)_{MSD} = 12$, and the results of the first adder is $(000010000)_{MSD} = 16$. These calculation values are consistent with theoretical calculations.

For the 6th, 7th, . . . , cars that are randomly generated later, the new position information is calculated on the TOC according to the similar way above for every 5 cars, and the liquid crystal board will have multi-screen output. From the experimental results, the calculation result of each screen of the TOC are consistent with the theoretical calculation values.

6. Conclusion

In this paper, a three-lane CA traffic flow model is designed and implemented in ternary optical computer SD16 using the characteristics of many processor bits, reconfigurability of processor bit and parallel computing of each processor bit. The specific implementation steps of the model on the TOC are introduced in detail. In this experiment, many vehicles are randomly generated, and their positions are determined by the five MSD adders in parallel. It shows that the system can run successfully and the results are the same as on electronic computer. The experiment verifies the correctness of the proposed model and the feasibility of the implementation method. It shows that the application of the cellular automaton on the TOC has obvious advantages.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Thank Prof. Jin Yi, Prof. Peng Junjie, Ouyang Shan and all members of TOC team for their kind help and valuable discussions in preparing the paper.

This work is supported by National Key R&D Program of China (No. 2017YFE0117500), and National Natural Science Foundation of China (No. 61572305, 61672006).

References

[1] John von Neumann, Design of computers theory of automata and numerical analysis, in: A.H. Taub (Ed.), John Von Neumann: Collected Works, Pergamon Press, Oxford, 1963.
[2] M. Cremer, J. Ludwig, A fast simulation model for traffic flow on the basis of Boolean operations, Math. Comput. Simulation 28 (4) (1986) 297–303.
[3] S. Wolfram, Statistical mechanics of cellular automata, Rev. Modern Phys. 55 (1983) 601–644.
[4] S. Wolfram, Theory and applications of cellular automata, in: Theory and Applications of Cellular Automata, World scientific, 1986, pp. 1346–1357.
[5] K. Nagel, Schreckenberg M., A cellular automaton model for freeway traffic, J. Physique (2) (1992) 2221–2229 (In France).
[6] X. Zhang, H. Li, K. Wang, et al., Quantum computation based on semiconductor quantum dots, Sci. China Inf. Sci. 47 (10) (2017) 1255–1276.
[7] Philip Ball, The Era of Quantum Computing Is Here, Outlook: Cloudy, Quanta Magazine, 2018, https://www.quantamagazine.org/the-era-of-quantum-computing-is-here-outlook-cloudy-20180124/, [OL].
[8] J. Xu, Probe machine, IEEE Trans. Neural Netw. Learn. Syst. 27 (7) (2016) 1405–1416.
[9] Y. Jin, H. He, Y. Lü, The basic principle of ternary optical computer, Sci. China E 33 (1) (2003) 111–115, 2.
[10] Y. Jin, H. He, Y. L.V., Ternary optical computer architecture, Phys. Scr. (T118) (2005) 98–101, 7.
[11] Y. Jin, H. He, Y. Lü, Ternary optical computer principle, Sci. China Inf. Sci. 46 (2) (2003) 145–150.
[12] Y. Jin, Y. Shen, J. Peng, S. Xu, et al., Principles and construction of MSD adder in ternary optical computer, Sci. China F 53 (11) (2010) 2159–2168, 11.
[13] Y. Jin, Draw near optical computer, J. Shanghai Univ. Nat. Sci. 17 (4) (2011) 401–411, 8.
[14] Y. Jin, X. Wang, J. Peng, M. Li, Z. Shen, S. Ouyang, Vector-matrix multiplication in ternary optimal computer, Int. J. Numer. Anal. Model. 9 (2) (2012) 401–409.
[15] J. Peng, M. Li, S. Ouyang, X. Wang, Z. Shen, Carry-free vector-matrix multiplication on a dynamically reconfigurable optical platform, Appl. Opt. 49 (12) (2010) 2352–2362.
[16] J. Jiang, Y. Shen, et al., Design and implementation of parallel SRT integer divider in ternary optical computer (in Chinese), Sci. Sin. Inform. (2020) http://dx.doi.org/10.1360/SSI-2019-0240, (in press).
[17] J. Peng, X. Wei, X. Zhang, et.al, Implementation of parallel FFT algorithm on ternary optical computer, Sci. China Inf. 47 (7) (2017) 846–862.
[18] J. Peng, Y. Fu, X. Zhang, et.al, Implementation of DFT application on ternary optical computer, Opt. Commun. 410 (2018) 424–430.
[19] J. Jiang, X. Zhang, Y. Shen, et.al, Design and implementation of SJ-MSD adder in ternary optical computer, J. Electron. (2019) in press, Manuscript number: C180572 (In Chinese).
[20] M. Rickert, K. Nagel, M. Schreckenberg, et al., Two lane traffic simulation using cellular automata, Physica A 231 (1996) 534–553.
[21] P. Wagner, N. Kai, D.E. Wolf, Realistic multi-lane traffic rules for cellular automata, Physica A 234 (34) (1997) 687–698.
[22] A. Ebersbach, J. Schneider, I. Morgenstern, Simulating traffic on German highways based on the Nagel-Scherckenberg-model, Internat. J. Modern Phys. C 12 (7) (2001) 1081–1089.
[23] S. Zhang, J. Peng, Y. Shen, et.al, Programming model and implementation mechanism for ternary optical computer, Opt. Commun. 428 (2018) 26–34.
[24] S. Zhang, Y. Jin, Y. Shen, J. Peng, et al., Overview of the task management system of ternary optical computer, in: The 2016 IEEE Cyber Science and Technology Congress, CyberSciTech 2016. 2016, pp. 132–135.
[25] Y. Jin, S. Zhang, S. Li, et al., Computing - data file — The key technology of applying ternary optical computer, J. Shanghai Jiaotong Univ. Nat. Sci. 53 (5) (2019) 584–592.