Chapter 15
A Computation in a Cellular Automaton Collider Rule 110

Genaro J. Martínez, Andrew Adamatzky and Harold V. McIntosh

Abstract A cellular automaton collider is a finite state machine build of rings of one-dimensional cellular automata. We show how a computation can be performed on the collider by exploiting interactions between gliders (particles, localisations). The constructions proposed are based on universality of elementary cellular automaton rule 110, cyclic tag systems, supercolliders, and computing on rings.

15.1 Introduction: Rule 110

Elementary cellular automaton (CA) rule 110 is the binary cell state automaton with a local transition function φ of a one-dimensional (1D) CA order ($k = 2, r = 1$) in Wolfram's nomenclature [57], where k is the number of cell states and r the number of neighbours of a cell. We consider periodic boundaries, i.e. first and last cells of a 1D array are neighbours. The local transition function for rule 110 is defined in Table 15.1, the string 01101110 is the number 110 in decimal notation:

G.J. Martínez (✉) · A. Adamatzky
Unconventional Computing Centre, University of the West of England,
Coldharbour Lane, Bristol BS16 1QY, UK
e-mail: genaro.martinez@uwe.ac.uk

A. Adamatzky
e-mail: andrew.adamatzky@uwe.ac.uk

H.V. McIntosh
Departamento de Aplicación de Microcomputadoras,
Universidad Autónoma de Puebla, 49 Poniente 1102, 72000 Puebla, Mexico
e-mail: mcintosh@unam.mx

A. Adamatzky (ed.), *Advances in Unconventional Computing*,
Emergence, Complexity and Computation 22,
DOI 10.1007/978-3-319-33924-5_15

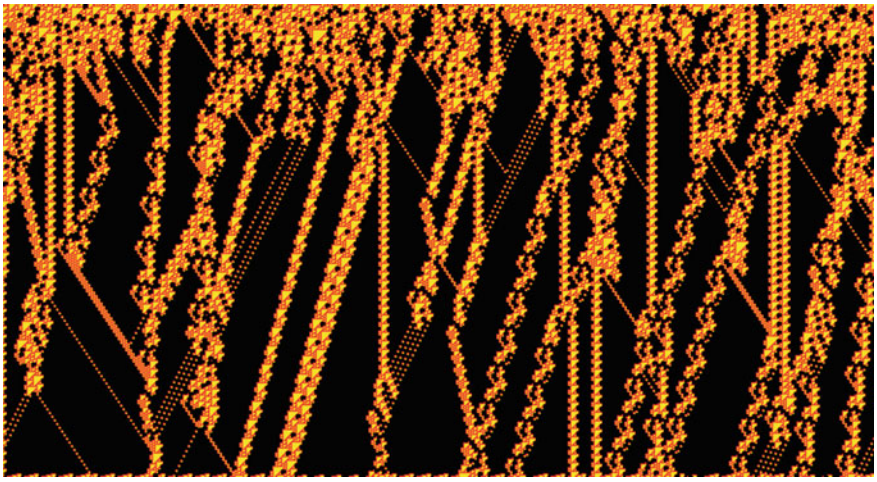

Fig. 15.1 An example of CA rule 110 evolving for 384 time steps from a random configuration, where each cell assigned state '1' with uniformly distributed probability 0.5. The particles are filtered. Time goes down

$$\begin{array}{ll} \varphi(1,1,1) \to 0 & \varphi(0,1,1) \to 1 \\ \varphi(1,1,0) \to 1 & \varphi(0,1,0) \to 1 \\ \varphi(1,0,1) \to 1 & \varphi(0,0,1) \to 1 \\ \varphi(1,0,0) \to 0 & \varphi(0,0,0) \to 0 \end{array} \tag{15.1}$$

A cell in state '0' takes state '1' if both its neighbours are in state '1' or left neighbour is '0' and right neighbour is '1'; otherwise, the call remains in the state '0'. A cell in state '1' takes state '0' if both its neighbours are in state '1', or both its neighbours are in state '0' or it left neighbour is '1' and its right neighbour is '0'. Figure 15.1 shows an evolution of rule 110 from a random initial condition. We can see there travelling localisation: particles or gliders, and some stationary localisations: breathers, oscillators or stationary structures.

15.1.1 System of Particles

A detailed description of particles/gliders discovered in evolutions of CA rule 110 is provided in [32, 36].[1] Further, we refers to a train of n copies of particle A as A^n.

Figure 15.2 shows all known particles, and generators of particles, or glider guns. Each particle has its unique features, e.g. slopes, velocities, periods, contact points, collisions, and phases [33, 35, 37]. A set of particles in rule 110 is defined as:

[1] See also, http://uncomp.uwe.ac.uk/genaro/rule110/glidersRule110.html.

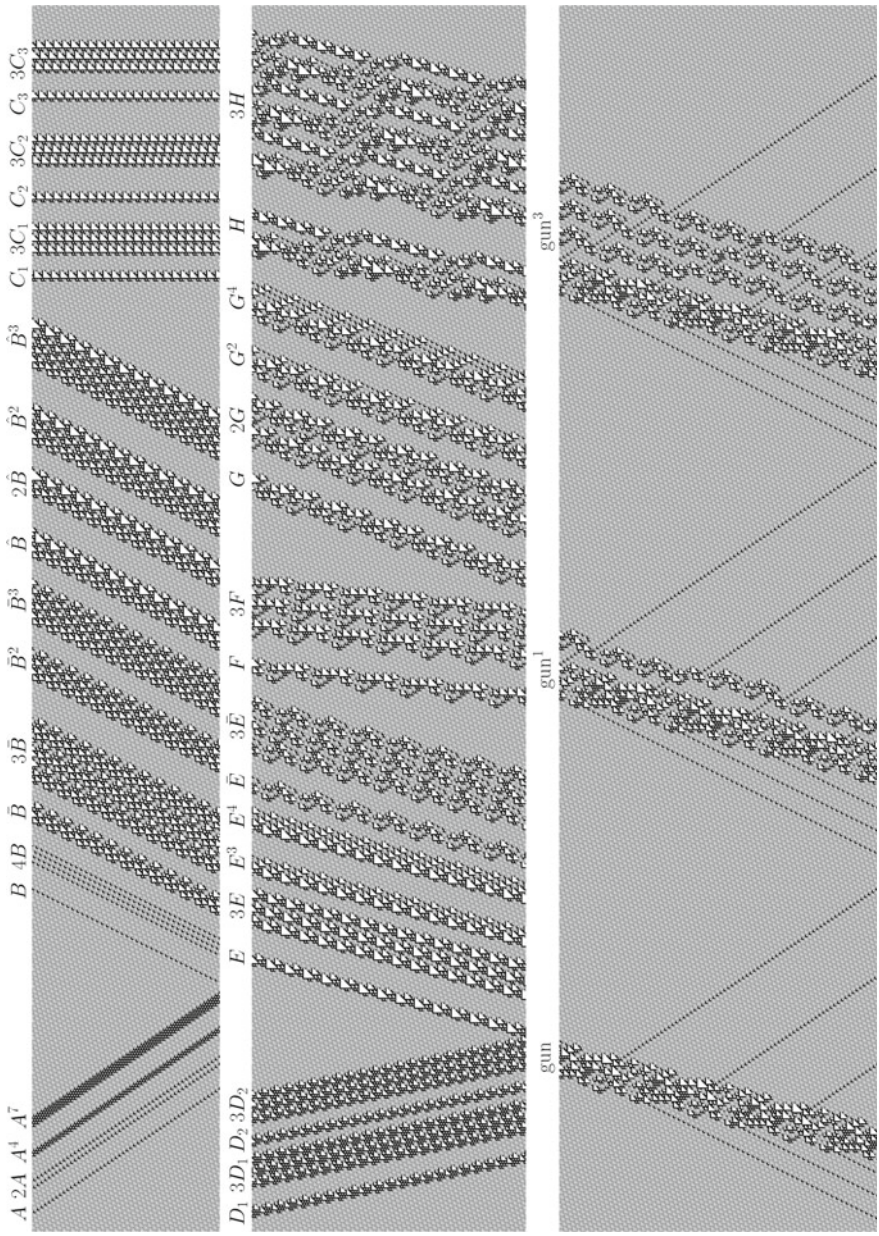

Fig. 15.2 Types of particles discovered in rule 110

Table 15.1 Properties of particles in rule 110

Structure	Margins				Velocity	Lineal volume
	Left – Right					
	ems	*oms*	*ems*	*oms*		
e_r	–	1	–	1	$2/3 \approx 0.666666$	14
e_l	1	–	1	–	$-1/2 = -0.5$	14
A	–	1	–	1	$2/3 \approx 0.666666$	6
B	1	–	1	–	$-2/4 = -0.5$	8
$\bar{B}^n$	3	–	3	–	$-6/12 = -0.5$	22
$\hat{B}^n$	3	–	3	–	$-6/12 = -0.5$	39
C_1	1	1	1	1	$0/7 = 0$	9–23
C_2	1	1	1	1	$0/7 = 0$	17
C_3	1	1	1	1	$0/7 = 0$	11
D_1	1	2	1	2	$2/10 = 0.2$	11–25
D_2	1	2	1	2	$2/10 = 0.2$	19
E^n	3	1	3	1	$-4/15 \approx -0.266666$	19
$\bar{E}$	6	2	6	2	$-8/30 \approx -0.266666$	21
F	6	4	6	4	$-4/36 \approx -0.111111$	15–29
G^n	9	2	9	2	$-14/42 \approx -0.333333$	24–38
H	17	8	17	8	$-18/92 \approx -0.195652$	39–53
Glider gun	15	5	15	5	$-20/77 \approx -0.259740$	27–55

$$\mathcal{G} = \{A, B, \bar{B}^n, \hat{B}^n, C_1, C_2, C_3, D_1, D_2, E^n, \bar{E}, F, G^n, H, gun^n\}.$$

where n means that a structure of the particle can be extendible infinitely, the rest of symbols denote types of particles as shown in Fig. 15.2. Table 15.1 summarizes key features of the particles: column *structure* gives the name of each particle including two more structures: e_r and e_l which represent the slopes of ether pattern (periodic background). The next four columns labeled *margins* indicate the number of periodic margins in each particle: they are useful to recognize contact points for collisions. The margins are partitioned in two types with even values *ems* and odd values *oms*

which are distributed also in two groups: left and right margins. Column v_g indicates a velocity of a particle g, where g belongs to a particle of the set of particles $\mathcal{G}$. A relative velocity is calculated during the particle's displacement on d cells during period p. We indicate three types of a particle propagation via sign of its velocity. A particle travelling to the right has *positive velocity*, a particle travelling to the left has *negative velocity*. Stationary particle has zero velocity. Different velocities of particles allow us to control distances between the particle to obtained programmable reactions between the particles. Typically, larger particles has lower velocity values. No particle can move faster than v_{e_r} or v_{e_l}. Column *lineal volume* shows the minimum and maximum number of necessary cells occupied by the particle.

15.1.2 Particles as Regular Expressions

We represent CA particles as strings. These strings can be calculated using de Bruin diagrams [31, 32, 34, 37, 55] or with the tiles theory [16, 33, 35, 37].[2]

A regular language L_{R110} is based on a set of regular expressions Ψ_{R110} uniquely describing every particle of $\mathcal{G}$. A subset of the set of regular expressions

$$\Psi_{R110} = \bigcup_{i=1}^{p} w_{i,g} \; \forall \; (w_i \in \Sigma^* \wedge g \in \mathcal{G}) \tag{15.2}$$

where $p \geq 3$ is a period, determines the language

$$L_{R110} = \{w | w = w_i w_j \vee w_i + w_j \vee w_i^* \text{ and } w_i, w_j \in \Psi_{R110}\}. \tag{15.3}$$

From these set of strings we can code initial configurations to program collisions between particles [27, 36, 39].

To deriver the regular expressions we use the de Bruijn diagrams [31, 34, 55] as follows. Assume the particle A moves two cells to the right in three time steps (see Table 15.1). The corresponding extended de Bruijn diagram (2-shift, 3-gen) is shown in Fig. 15.3. Cycles in the diagram are periodic sequences uniquely representing each phase of the particle. Diagram in Fig. 15.3 has two cycles: a cycle formed by just a vertex 0 and another large cycle of 26 vertices composed by other nine internal cycles. The sequences or regular expressions determining the phases of the particle A are obtained by following paths through the edges of the diagram. There regular expressions and corresponding paths in Bruijn diagram are shown below.

I. The expression (1110)*: vertices 29, 59, 55, 46 determining A^n particles.

II. The expression (111110)*: vertices 61, 59, 55, 47, 31, 62 defining nA particles with a T_3 tile among each particle.

[2] See a complete set of regular expressions for every particle in rule 110 in http://uncomp.uwe.ac.uk/genaro/rule110/listPhasesR110.txt.

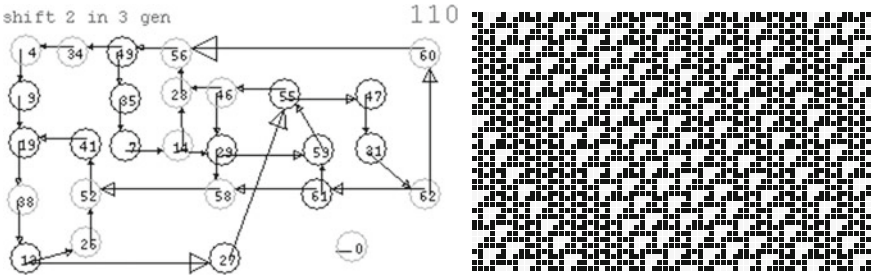

Fig. 15.3 De Bruijn diagram calculating *A* particles (*left*) and space-time configuration of automaton showing locations of periodic sequences produced (*right*)

Table 15.2 Four sets of phases Ph_i in rule 110

Phases level one (Ph_1) → {f_1_1, f_2_1, f_3_1, f_4_1}
Phases level two (Ph_2) → {f_1_2, f_2_2, f_3_2, f_4_2}
Phases level three (Ph_3)→ {f_1_3, f_2_3, f_3_3, f_4_3}
Phases level four (Ph_4) → {f_1_4, f_2_4, f_3_4, f_4_4}

III. The expression (11111000100110)*: vertices 13, 27, 55, 47, 31, 62, 60, 56, 49, 34, 4, 9, 19, 38 describing the periodic background configurations in a specific phase.

Cycle with period 1 (vertex 0) yields a homogeneous evolution in state 0. The evolution space in Fig. 15.3 shows different trains of *A* particles. The initial condition is constructed following some of the seven possible cycles of the de Bruijn diagram or a combination of them. In this way, the number of particles *A* or the number of intermediate tiles T_3 can be selected by moving from one cycle to another.

The alignment of the f_i_1 phases is analysed to determine the whole set of strings for every particle. We describe the form and limits of each particle by tiles. Then a phase is fixed (in our case the phase f_i_1) and a horizontal line is placed in the evolution space bounded by two aligned T_3 tiles. The sequence between both tiles aligned in each of the four levels determines a periodic sequence representing a particular structure in the evolution space of rule 110. All periodic sequences in a specific phase are calculated, enumerating the phases for each particle or non-periodic structure.

Table 15.2 represents disjoint subset of phases, each level contains four phases. Variable f_i indicates the phase of a particle, and the subscript *j* (in the notation f_i_*j*) indicates the selected set Ph_j of regular expressions. Finally, we use the next notation to codify initial conditions by phases as follows:

$$\#_1(\#_2, f_i\_1) \tag{15.4}$$

where $\#_1$ represents a particle according to Cook's classification (Table 15.1) and $\#_2$ is a phase of the particle with period greater than four.

15.2 Universal elementary CA

A concept of universality and self-reproduction in CA was proposed by von Neumann in [54] in his design of a universal constructor in a 2D CA with 29 cell-states. Architectures of universal CA have been simplified by Codd in 1968 [10], Banks in 1971 [7], Smith in 1971 [51], Conway in 1982 [8], Lindgren and Nordahl in 1990 [22], and Cook in 1998 [11].[3] Cook simulated a cyclic tag system, equivalent to a minimal Turing machine, in CA rule 110. In general, computation capacities are explores in complex CA and chaotic CA [40].

15.3 Cyclic Tag Systems

Cyclic tag systems are used by Cook in [11] as a tool to implement computations in rule 110. Cyclic tag systems are modified from tag systems by allowing the system to have the same action of reading a tape in the front and adding characters at its end:

1. Cyclic tag systems have at least two letters in their alphabet ($\mu > 1$).
2. Only the first character is deleted ($\nu = 1$) and its respective sequence is added.
3. In all cases if the machine reads a character zero then the production rule is always null ($0 \to \varepsilon$, where ε represents the empty word).
4. There are k sequences from μ^* which are periodically accessed to specify the current production rule when a nonzero character is taken by the system. Therefore the period of each cycle is determinate by k.

Such cycle determines a partial computation over the tape, although a halt condition is not specified. Let us see some samples of a cyclic tag system working with $\mu = 2$, $k = 3$ and the following production rules: $1 \to 11$, $1 \to 10$ and $1 \to \varepsilon$. To avoid writing a chain when there is no need to add characters, the $\vdash_k$ relation is just indicated. For example, the $00001 \vdash_1 \vdash_2 \vdash_3 \vdash_1 \vdash_2 10$ represents the relations $00001 \vdash_1 0001 \vdash_2 001 \vdash_3 01 \vdash_1 1 \vdash_2 10$. Each relation indicates which exactly sequence μ is selected.

Cyclic tag systems tend to growth quickly which makes it difficult to analyse their behaviour. Morita in [43, 44] demonstrated how to implement a particular halt condition in cyclic tag systems given an output string when the system is halting, and how a partitioned CA can simulate any cyclic tag system, consequently computing all the recursive functions.

Similar to Post's developments with tag systems, Cook determined that for a cyclic tag system with $\mu = 2$, $k = 2$, the productions $1 \to 11$ and $1 \to 10$, and starting evolution with the state 1 on the tape, it is impossible to decide if the process is terminal.

[3] A range of universal CA is listed here http://uncomp.uwe.ac.uk/genaro/Complex_CA_repository.html.

15.4 Cyclic Tag System Working in Rule 110

Let us see how a cyclic tag system operates in rule 110 [58]. We use a cyclic tag system with $\mu = 2$, $k = 2$ and the productions $1 \rightarrow 11$ and $1 \rightarrow 10$, starting its evolution in state 1 on the tape. A fragments of the systems' behaviour is shown below:

$$1 \vdash_1 11 \vdash_2 110 \vdash_1 1011 \vdash_2 01110 \vdash_1\vdash_2 11010 \vdash_1 101011 \vdash_2 0101110 \vdash_1\vdash_2 0111010 \vdash_1\vdash_2 1101010 \vdash_1 10101011 \vdash_2 010101110 \vdash_1\vdash_2 010111010 \vdash_1\vdash_2 011101010 \vdash_1\vdash_2 110101010 \vdash_1 1010101011 \vdash_2 01010101110 \vdash_1\vdash_2 01010111010 \vdash_1\vdash_2 01011101010 \vdash_1\vdash_2 01110101010 \vdash_1\vdash_2 11010101010 \vdash_1 101010101011 \vdash_2 0101010101110 \vdash_1\vdash_2 0101010111010 \vdash_1\vdash_2 01010111010\,10 \vdash_1\vdash_2 0101110101010 \vdash_1\vdash_2 0111010101010 \vdash_1\vdash_2 1101010101010 \vdash_1 10101010101011 \vdash_2 0101010101110 \vdash_1\vdash_2 0101010111010 \vdash_1\vdash_2 01010\ 1011101010 \vdash_1\vdash_2 010101110101010 \vdash_1\vdash_2 010111010101010 \ldots$$

We start with the expression 1(10)*. The cyclic tag systems moves (from the right to the left) and adds a pair of bits. As soon as the expression 1(10)* appears again, a number of relations selected in each interval in such a manner that the expressions grow lineally in order of $f_1 = 2(n + 1)$.

If we take consecutive copies of 1(10)* with their respective intervals determined by the number of j productions (represented as $\vdash_i^j$), we obtain the following sequence: $1 \vdash_i^2 110 \vdash_i^4 11010 \vdash_i^6 1101010 \vdash_i^8 1101010 \vdash_i^{10} 110101010 \vdash_i^{12} 11010101010 \vdash_i^{14} 1101010101010 \vdash_i^{16} \ldots$. There are no states where to '0' appear together.

Further, we show how to interpret particles and their collisions to emulate a cyclic tag system in rule 110. We must use trains of particles to represent data and operators, their reactions, transform and deletion of data on the tape. A schematic diagram, where trains of particles are represented by lines, is shown in Fig. 15.4. The diagram is explained with details in the next sections.

15.4.1 Components Based on Sets of Particles

A construction of the cyclic tag system in rule 110 can be subdivided into three parts (Fig. 15.4). First part is the left periodic part controlled by trains of 4_A^4 particles. This part is static. It controls the production of 0's and 1's. The second part is the center determining the initial value on the tape. The third part is the right, cyclic, part which contains the data to process. It adds or removes data on the tape.

Set of particles 4_A^4

The four trains of A^4 particles are static but their phases change periodically. A key point is to implement these components by defining both distances and phases, because some choices of phases or distances might induce an undesirable reactions between the trains of particles.

Packages defined by particles A^4 have three different phases: f_1_1, f_2_1 and f_3_1. To construct the first train 4_A^4 we must establish the phase of each A^4. Let us assign

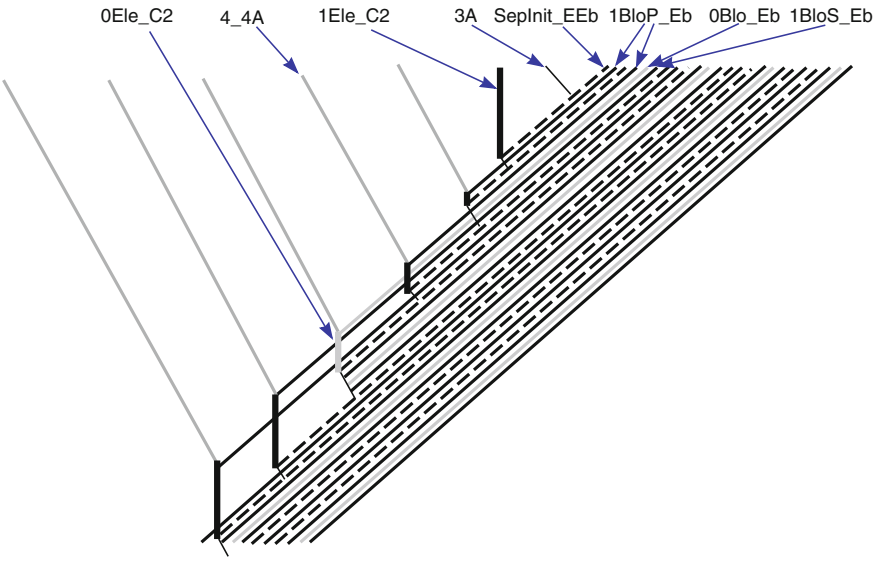

Fig. 15.4 Schematic diagram of a cyclic tag system working in rule 110

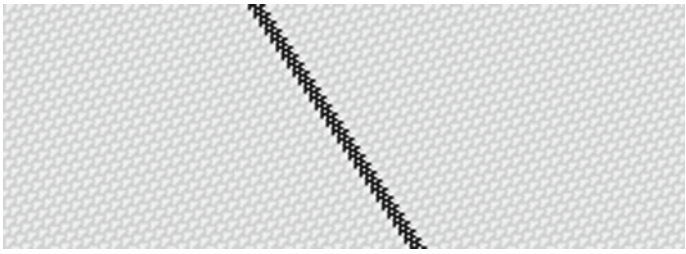

Fig. 15.5 Set of particles 4_A^4

phases as follows:

A^4(f$_3$_1)-27e-A^4(f$_2$_1)-23e-A^4(f$_1$_1)-25e-A^4(f$_3$_1),

see Fig. 15.5. Spaces between each train 4_A^4 are fixed but the phases change. The soliton-like collisions between the particles $\bar{E}$ occur:

{649e-A^4(f$_2$_1)-27e-A^4(f$_1$_1)-23e-A^4(f$_3$_1)-25e-A^4(f$_2$_1)-649e-A^4(f$_1$_1)-27e-A^4(f$_3$_1)-23e-A^4(f$_2$_1)-25e-A^4(f$_1$_1)]-649e-A^4(f$_3$_1)-27e-A^4(f$_2$_1)-23e-A^4(f$_1$_1)-25e-A^4(f$_3$_1)}*

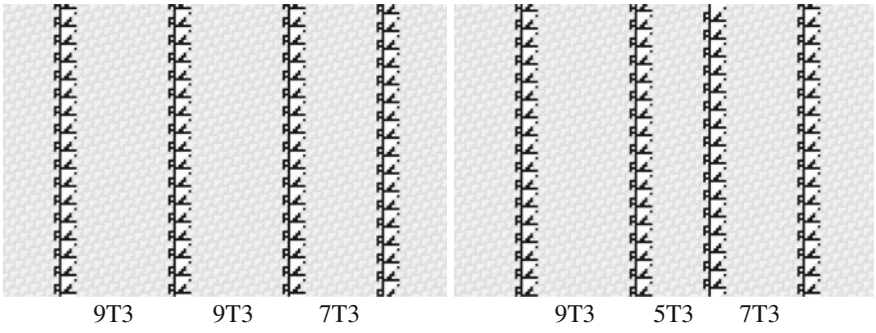

Fig. 15.6 Set of particles 1Ele_C_2 (*left*) and 0Ele_C_2 (*right*)

If for every 4_A^4 we take a phase representing the complete train, we can rename it as:

$$\{649e - 4\_A^4(F_2) - 649e - 4\_A^4(F_1) - 649e - 4\_A^4(F_3)\}*$$

this phase change is important to preserve good reactions coming to the left side of the system.

Set of particles 1Ele_C_2 and 0Ele_C_2

The central part is made of the state '1' written on the tape represented by a train of four C_2 particles. A set of particles 1Ele_C_2 represents '1' and a set of particles 0Ele_C_2 represents '0' on the tape.

The left configurations in Fig. 15.6 shows the set of particles 1Ele_C_2. We should reproduce each set of particles by the phases f_i_1. The phases are coded as follows: C_2(A,f_1_1)-2e-C_2(A,f_1_1)-2e-C_2(A,f_1_1)-e-C_2(B,f_2_1). The first three particles C_2 are in phase (A,f_1_1) and the fourth particle C_2 is in phase (B,f_2_1). The distances between the particles are $9T_3$-$9T_3$-$7T_3$. To determine the distances, we count the number of tiles T_3 between particles. Similarly, we obtain the distances $9T_3$-$5T_3$-$7T_3$ for the particles 0Ele_C_2.

Set of particles 0Blo_$\bar{E}$

The left part stores blocks of data without transformations in trains of E and the particles $\bar{E}$.

The set of particles 0Blo_$\bar{E}$ is formed by 12$\bar{E}$ particles as we can see in Fig. 15.7. There must be an exact phase and distance between each one of the particles, otherwise the whole system will be disturbed.

Set of particles 1BloP_$\bar{E}$ and 1BloS_$\bar{E}$

To write '1's we must use two set of particles—*primary* and *standard*.

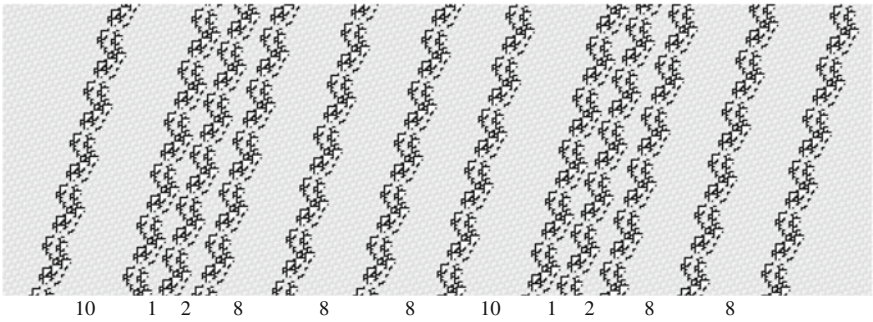

Fig. 15.7 Set of particles 0Blo_$\bar{E}$

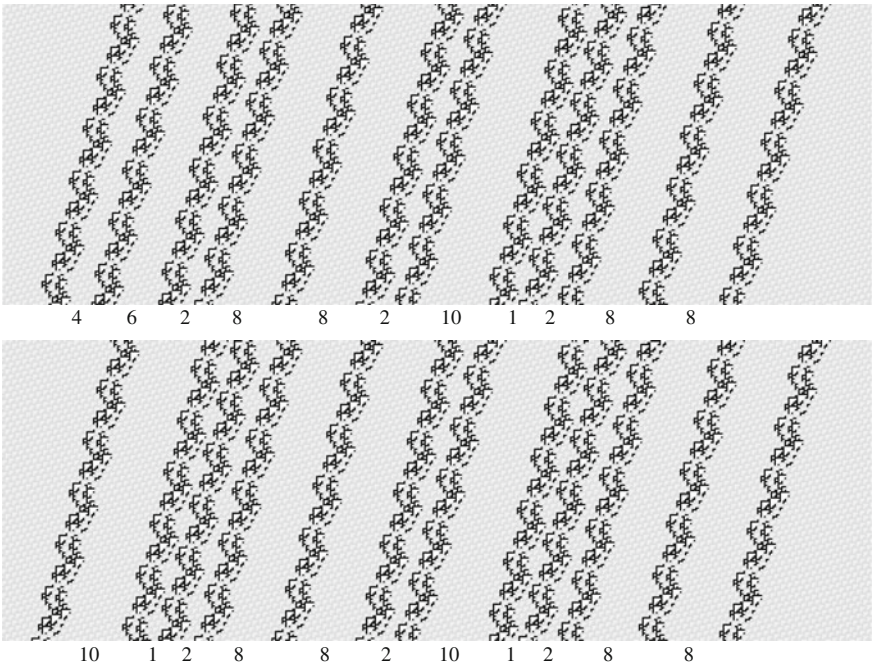

Fig. 15.8 Set of particles 1BloP_$\bar{E}$ (*up*) and 1BloS_$\bar{E}$ (*down*)

They are differences in distance between first two particles $\bar{E}$, as shown in Fig. 15.8. Both blocks produce the same set of particles 1Add_$\bar{E}$. The main reason to use both set of particles is because the CA rule 110 evolves asymmetrically and therefore we need a double set of particles to produce values 1 correctly.

Set of particles SepInit_$E\bar{E}$

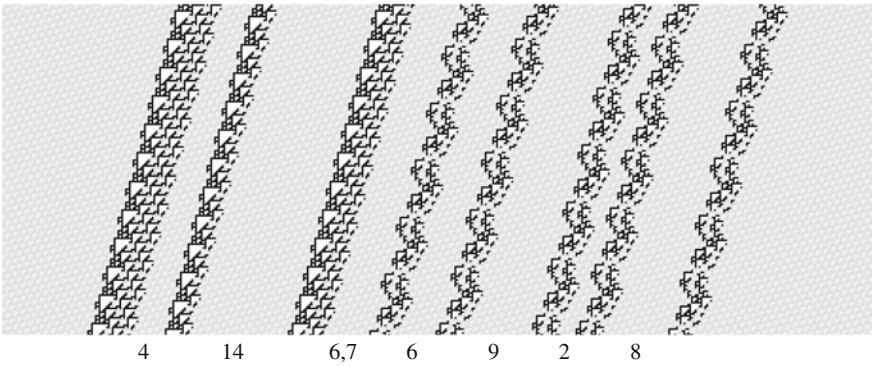

Fig. 15.9 Set of particles SepInit_$E\bar{E}$

A leader component renamed as the set of particles SepInit_$E\bar{E}$ (see Fig. 15.9) is essential to separate trains of data and to determine the incorporation of the data on the tape. Its has a small but detailed code determining which data without transformation would be added or erased from the tape, depending on the value that is coming.

Set of particles 1Add_$\bar{E}$ and 0Add_$\bar{E}$

Figure 15.10 illustrates the set of particles 1Add_$\bar{E}$ and 0Add_$\bar{E}$ produced by two previous different trains of data. A set of particles 1Add_$\bar{E}$ must be generated by the set of particles 1BloP_$\bar{E}$ or 1BloS_$\bar{E}$. This way, both set of particles can produce the same element.

On the other hand, a set of particles 0Add_$\bar{E}$ is generated by a set of particles 0Blo_$\bar{E}$. Nevertheless, we could produce $\bar{E}$ particles modifying their first two distances and preserving them without changing others particles to get a reliable reaction. This is possible if we want to experiment with other combinations of blocks of data.

If a leader set of particles SepInit_$E\bar{E}$ reaches a set of particles 1Ele_$\bar{E}$, it erases this value from the tape and adds a new data that shall be transformed. In other case, if it finds a set of particles 0Ele_$\bar{E}$, then it erases this set of particles from the tape and also erases a set of unchanged data which comes from the right until finding a new leader set of particles. This operation represents the addition of new values from periodic trains of particles coming from the right. Thus a set of particles 1Add_$\bar{E}$ is transformed into 1Ele_$\bar{E}$ colliding against a train of 4_A^4 particles representing a value 1 in the tape, and the set of particles 0Add_$\bar{E}$ is transformed into 0Ele_$\bar{E}$ colliding against a train of 4_A^4 particles representing a value 0 in the tape.

Table 15.3 shows all distances (in numbers of T_3 tiles) for every. We can code the construction of this cyclic tag system across phase representations in three main big sub systems:

left: …-217*e*-4_A^4(F2)-649*e*-4_A^4(F1)-649*e*-4_A^4(F3)-649*e*-4_A^4(F2)-

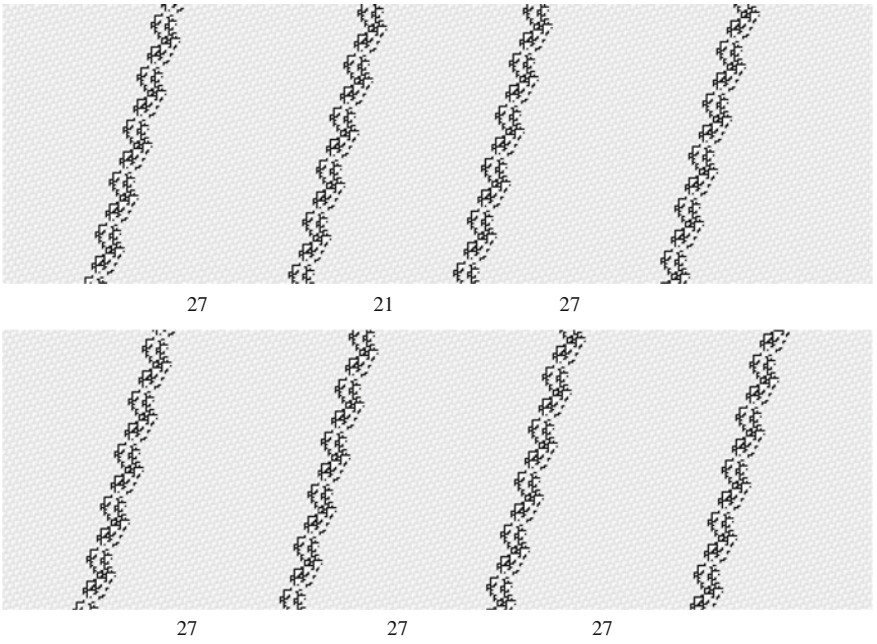

Fig. 15.10 Set of particles 1Add_Eb (*up*) and 0Add_$\bar{E}$ (*down*)

Table 15.3 Distances between sets of particles

Set of particles	Distance
1Ele_C_2	9-9-7
0Ele_C_2	9-5-7
1BloP_$\bar{E}$	4-6-2-8-8-2-10-1-2-8-8
1BloS_$\bar{E}$	10-1-2-8-8-2-10-1-2-8-8
0Blo_$\bar{E}$	10-1-2-8-8-8-10-1-2-8-8
SepInit_$E\bar{E}$	4-14-(6 or 7)-6-9-2-8
1Add_$\bar{E}$	27-21-27
0Add_$\bar{E}$	27-27-27

649e-4_A^4(F1)-649e-4_A^4(F3)-216e-

center: 1Ele_C_2(A,f$_1$_1)-e-A^3(f$_1$_1)-

right: SepInit_$E\bar{E}$(C,f$_3$_1)-1BloP_$\bar{E}$(C,f$_4$_1)-SepInit_$E\bar{E}$(C,f$_3$_1)-
1BloP_$\bar{E}$(C,f$_4$_1)-0Blo_$\bar{E}$(C,f$_4$_1)-1BloS_$\bar{E}$(A,f$_4$_1)-
SepInit_$E\bar{E}$(A,f$_2$_1)(2)-1BloP_$\bar{E}$(F,f$_1$_1)-SepInit_$E\bar{E}$(A,f$_3$_1)(2)-
1BloP_$\bar{E}$(F,f$_1$_1)-0Blo_$\bar{E}$(E,f$_4$_1)-1BloS_$\bar{E}$(C,f$_4$_1)-e-
SepInit_$E\bar{E}$(B,f$_1$_1)(2)-1BloP_$\bar{E}$(F,f$_3$_1)-e-
SepInit_$E\bar{E}$(B,f$_1$_1)(2)-217e-….

The initial conditions in rule 110 are able to generate the serial sequence of bits 1110111 and a separator at the end with two particles. A desired construction is achieved in 57,400 generations and an initial configuration of 56,240 cells. The whole evolution space is 3,228,176,000 cells. See details [38].

15.4.2 Simulating a Cyclic Tag System in Rule 110

The cyclic tag system starts with the value '1' on the tape, see Fig. 15.4. We show a selection of snapshots of the machine working in rule 110 (see details in [38, 47]). We show different sets of particles with coloured labels on the snapshot below.

Figure 15.11 shows the initial stage of the cyclic tag system with the state '1' in the tape. This data is represented by the set of particles 1Ele_C_2. The snapshot shows a central part of the machine and a train of A^3 particles. We can see the first leader in the set of particles SepInit_$E\bar{E}$ coming from the right periodic side.

The first reaction in Fig. 15.11 deletes the state '1' on the tape. The set of particles 1Ele_C_2) and the particles' separator are prepared for next data to be aggregated. If a set of particles 0Ele_C_2 is encountered on the tape then data is not added to the tape until another separator appears. The particles $\bar{E}$ left after the first production are invisible to the system, they do not affect any operations because they cross as solitons, without state modifications, the subsequent set of particles 4_A^4.

In Fig. 15.12 we see a set of particles 1Ele_C_2 constructed from a train of particles 4_A^4. These particles have a very short life because quickly a separator set of particles arrives. This separator erases the particles and prepares new data that would be aggregated to the tape.

Figure 15.13 presents the construction of a set of particles 1Ele_C_2. In this stage of the evolution, we can see how data is aggregated, based on their values, before they cross the tape. Similar reactions can be observed with the set of particles 0Ele_C_2.

Figure 15.14 shows a set of particles 0Ele_C_2 and its roles in the system. At the top, a set of particles 1Add_$\bar{E}$, previously produced by a standard component 1BloS_$\bar{E}$, crosses a set of particles 0Ele_C_2. A leader set of the particles deletes '0' from the tape and all the subsequent incoming data. There are 1BloP_$\bar{E}$, 0Blo_$\bar{E}$ and 1BloS_$\bar{E}$ set of particles in the illustrated sequence. The tile T_{14} is generated in the process. This differences in distances between the particles determine a change of phases which will lead to erasure of particles $\bar{E}$, instead of production of particles C. The reaction $A^3 \rightarrow \bar{E}$ is used to delete the particles.

Production rules in cyclic tag system specify that for the state '0' the first element of the chain must be erased and the other elements are conserved and no data are written on the tape. If the state is '1' the first element of the chain is deleted and 10 or 11 are aggregated depending of the k value. This behaviour is particularly visible when a separator finds 0 or 1 and deletes it from the tape. If the deleted data is '0', a separator does not allow the production of new data. If the deleted data is '1' the separator aggregates new elements 11 or 10, which are modified at later stages of the

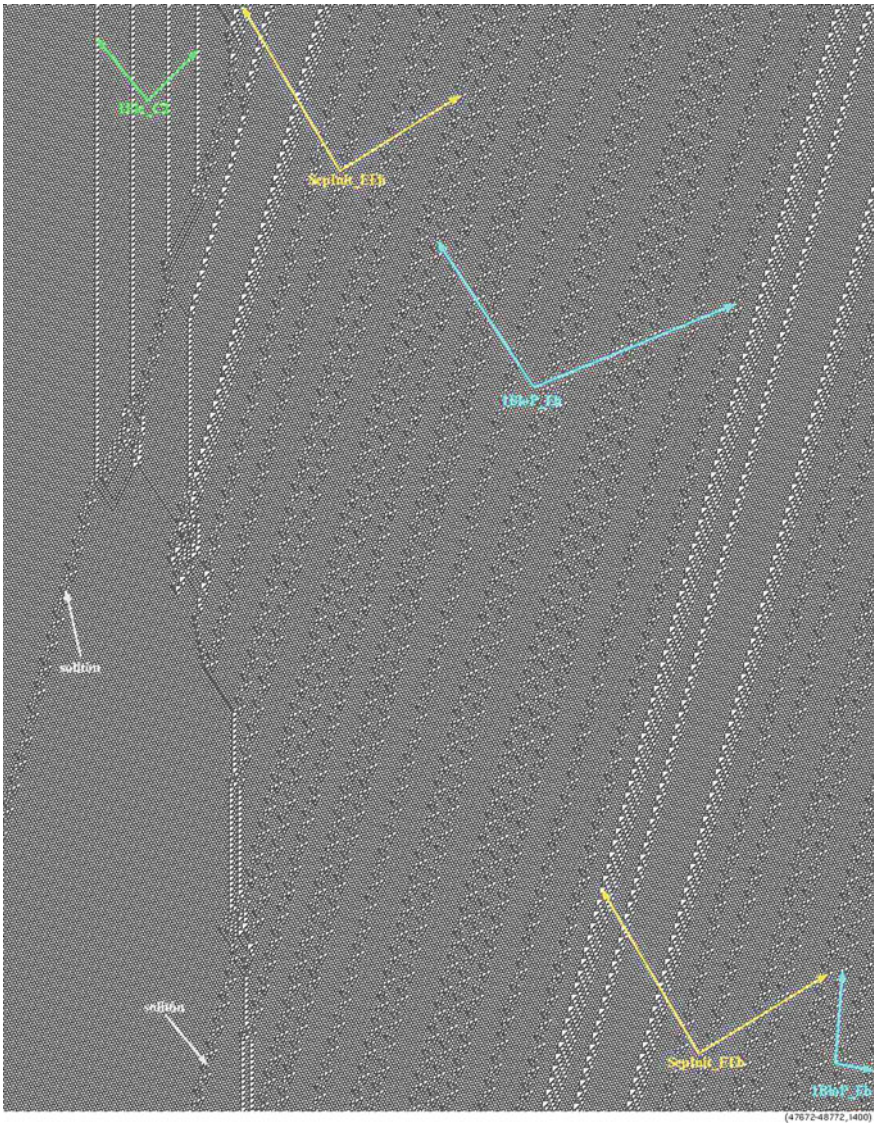

Fig. 15.11 Initial stage of cyclic tag system in rule 110

system's development. Using this procedure, we can calculate up to the sixth '1' of the sequence $011<1>0$ produced by the cyclic tag system.

In terms of periodic phases, this cyclic tag system working in rule 110 can be simplified as follows:

left: $\{649e\text{-}4\_A^4(\mathrm{F}\_i)\}^*$, for $1 \leq i \leq 3$ in sequential order

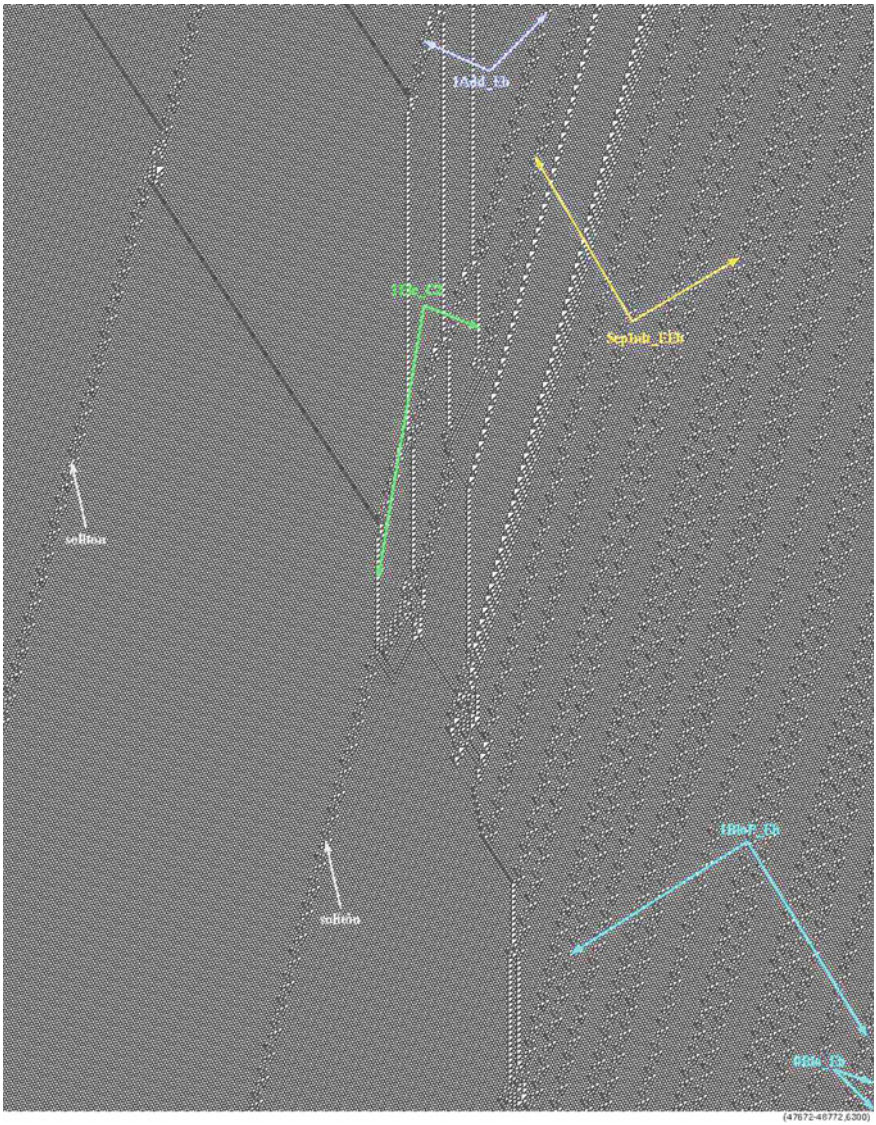

Fig. 15.12 Constructing an element 1Ele_C_2

center: 246e-1Ele_C2(A,f_1_1)-e-A^3(f_1_1)

right: {SepInit_$E\bar{E}$(#,f_i_1)-1BloP_$\bar{E}$(#,f_i_1)-SepInit_$E\bar{E}$(#,f_i_1)-1BloP_$\bar{E}$(#,f_i_1)-0Blo_$\bar{E}$(#,f_i_1)-1BloS_$\bar{E}$(#,f_i_1)}* (where $1 \leq i \leq 4$ and # represents a particular phase).

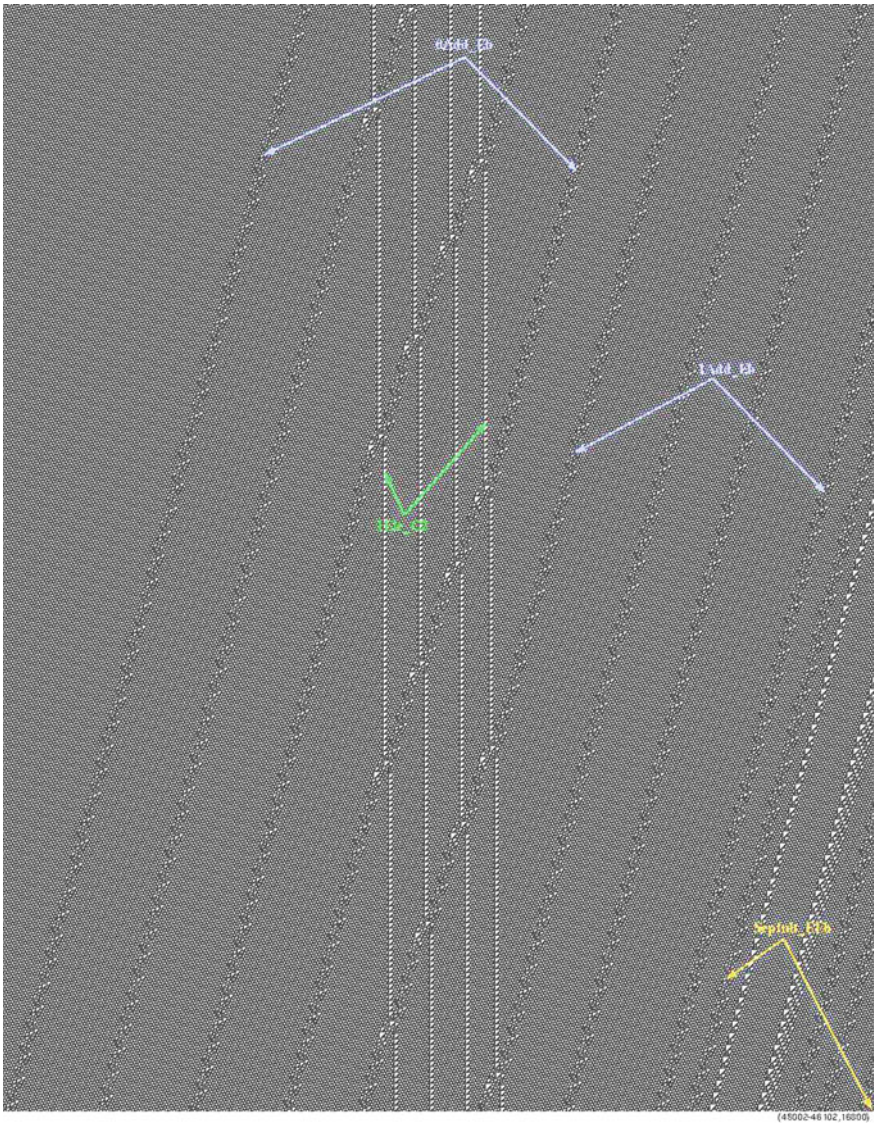

Fig. 15.13 Transformed data crossing the tape of values

These periodic coding will be very useful to design and synchronise three interlinked rings of 1D CA (cyclotrons) to make a 'supercollider'.

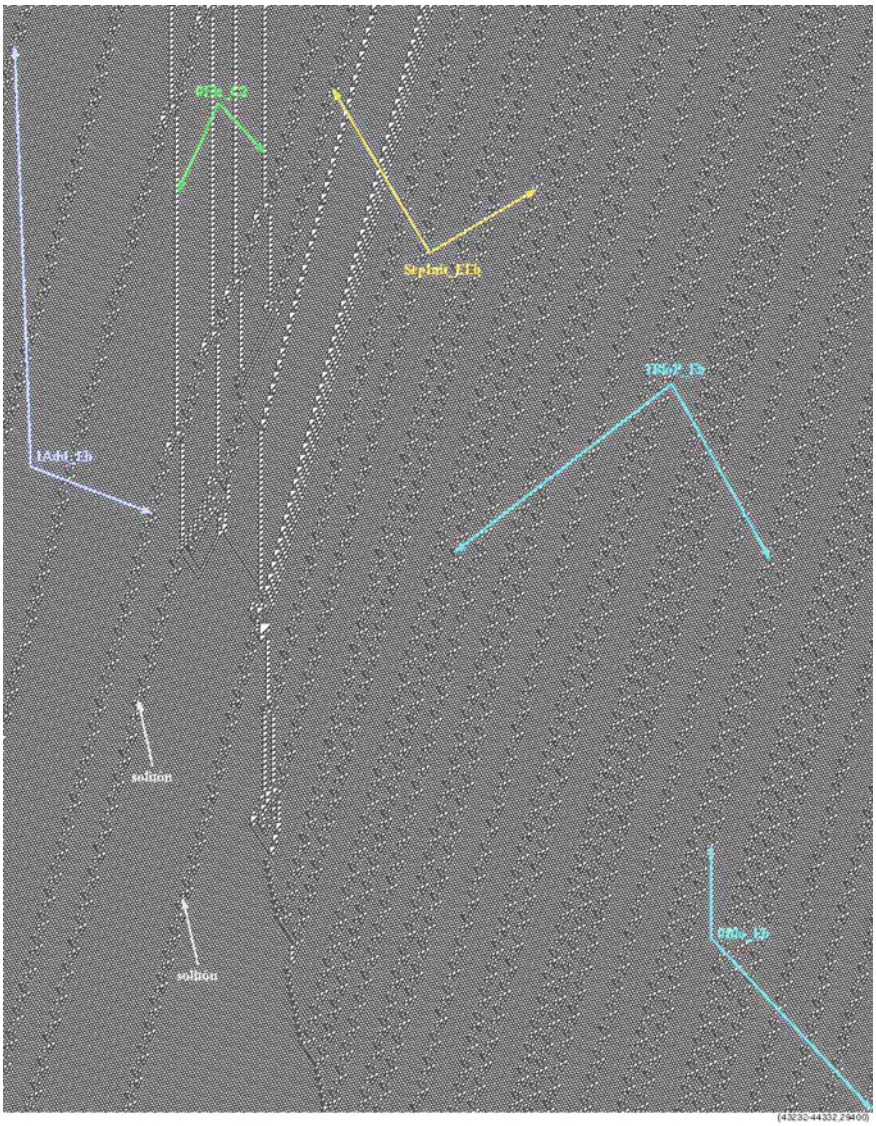

Fig. 15.14 Deleting a set of particles 0Ele_C2

15.5 Cellular Automata Supercollider

In the late 1970s Fredkin and Toffoli proposed a concept of computation based on ballistic interactions between quanta of information that are represented by abstract particles [53]. The Boolean states of logical variables are represented by balls or

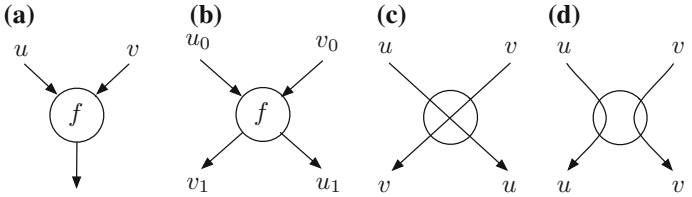

Fig. 15.15 Schemes of ballistic collision between localizations representing logical values of the Boolean variables u and v

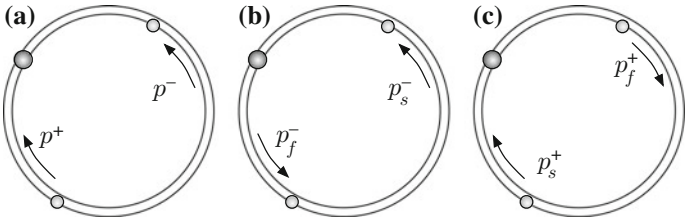

Fig. 15.16 Representation of abstract particles in a 1D CA ring

atoms, which preserve their identity when they collide with each other. Fredkin, Toffoli and Margolus developed a billiard-ball model of computation, with underpinning mechanics of elastically colliding balls and mirrors reflecting the balls' trajectories. Margolus proposed a special class of CA which implements the billiard-ball model [24]. Margolus' partitioned CA exhibited computational universality because they simulated Fredkin gates via collision of soft spheres [25, 26]. Also, we consider previous results about circular machines designed by Arbib, Kudlek, and Rogozhin in [5, 20, 21]. Initial reports about CA collider were published in [28–30].

The following functions with two input arguments u and v can be realised in collisions between two localizations:

- $f(u, v) = c$, fusion (Fig. 15.15a)
- $f(u, v) = u + v$, interaction and subsequent change of state (Fig. 15.15b)
- $f_i(u, v) \mapsto (u, v)$ identity, solitonic collision (Fig. 15.15c);
- $f_r(u, v) \mapsto (v, u)$ reflection, elastic collision (Fig. 15.15d);

To represent Toffoli's supercollider [53] in 1D CA we use the notion of an idealised particle $p \in \mathcal{G}$ (without energy and potential). The particle p is represented by a binary string of cell states.

Figure 15.16 shows two typical scenarios where particles p_f and p_s travel in a CA cyclotron. The first scenario (Fig. 15.16a) shows two particles travelling in opposite directions; these particles collide one with another. Their collision site (contact point) is shown by a dark circle in Fig. 15.16a. The second scenario demonstrates a beam routing where a fast particle p_f eventually catches up with a slow particle p_s at a collision site (Fig. 15.16b). If the particles collide like solitons, then the faster particle p_f simply overtakes the slower particle p_s and continues its motion (Fig. 15.16c).

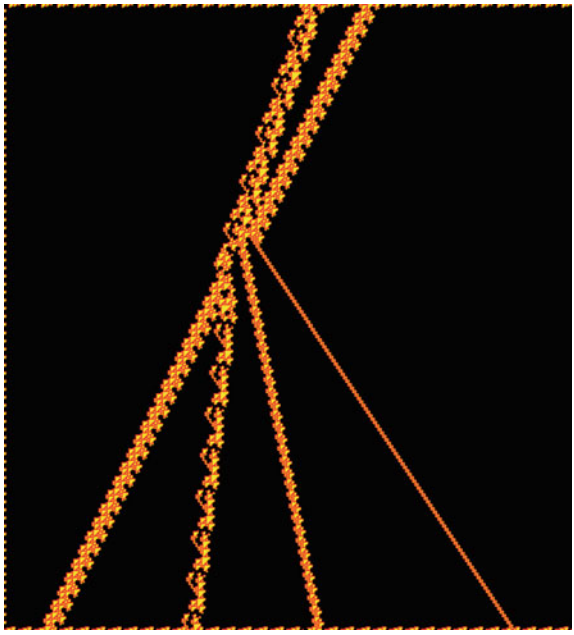

Fig. 15.17 Particle collision in rule 110. Particle $p^-_{\bar{B}}$ collides with particle p^-_G giving rise to three new particles—p^-_F, $p^+_{D_2}$, and $p^+_{A^3}$, and preserving the $p^-_{\bar{B}}$ particle—that are generated as a result of the collision

Typically, we can find all types of particles in complex CA, including particles with positive p^+, negative p^-, and neutral p^0 displacements, and composite particles assembled from elementary localizations. A sample coding and colliding particles is shown in Fig. 15.17, which displays a typical collision between two particles in rule 110. As a result of the collision one particle is split into three different particles (for full details please see [35]). The previous collision positions of particles determines the outcomes of the collision. Particles are represented now with orientation and name of the particle in rule 110 as follows: $p^{+,-,0}_{\mathcal{G}}$.

To represent particles on a given beam routing scheme (see Fig. 15.16), we do not consider the periodic background configuration in rule 110 because essentially this does not affect on collisions. Figure 15.18 displays a 1D configuration where two particles collide repeatedly and interact as solitons so that the identities of the particles are preserved in the collisions. A negative particle p^-_F collides with and overtakes a neutral particle $p^-_{C_1}$. First cyclotron (Fig. 15.18a) presents a whole set of cells in state 1 (dark points) evolving with the periodic background. By applying a filter we can see better these interactions (Fig. 15.18b).[4] Typical space-time configurations of a CA exhibiting a collision between p^-_F and $p^-_{C_1}$ particles are shown in Fig. 15.18c.

[4]Cyclotron evolution was simulated with DDLab software, available at http://www.ddlab.org.

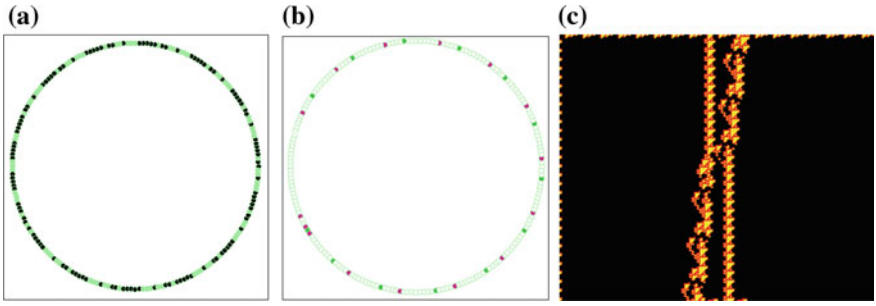

Fig. 15.18 A soliton-type interaction between particles in rule 110: **a**, **b** two steps of beam routing, **c** exact configuration at the time of collision

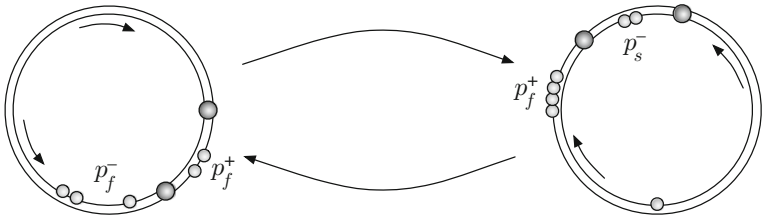

Fig. 15.19 Transition between two beam routing synchronising multiple reactions. When the first set of collisions is done a new beam routing is defined with other set of particles, so that when the second set of collisions is done then first beam returns to its original state

15.6 Beam Routings and Computations

We examine beam routing based on particle-collisions. We will show how the beam routing can be used in designs of computing based-collisions connecting cyclotrons. Figure 15.19 shows a beam routing design, connecting two of beams and then creating a new beam routing diagram where edges represent a change of particles and collisions. In such a transition, new particles emerge and collide to return to the first beam. The particles oscillate between these two beam routing indefinitely.

To understand how dynamics of a double beam differs from a conventional 1D evolution space we provide Fig. 15.20. There we can see multiple collisions between particles from first beam routing and trains particles. Exactly, we have that

$$p_A^+, p_A^+ \leftrightarrow p_{\bar{B}}^-, p_B^-, p_B^-$$

changes to the set of particles derived in the second beam routing:

$$p_{A^4}^+ \leftrightarrow p_E^+, p_{\bar{E}}^+.$$

This oscillation determines two beam routing connected by a transition of collisions as:

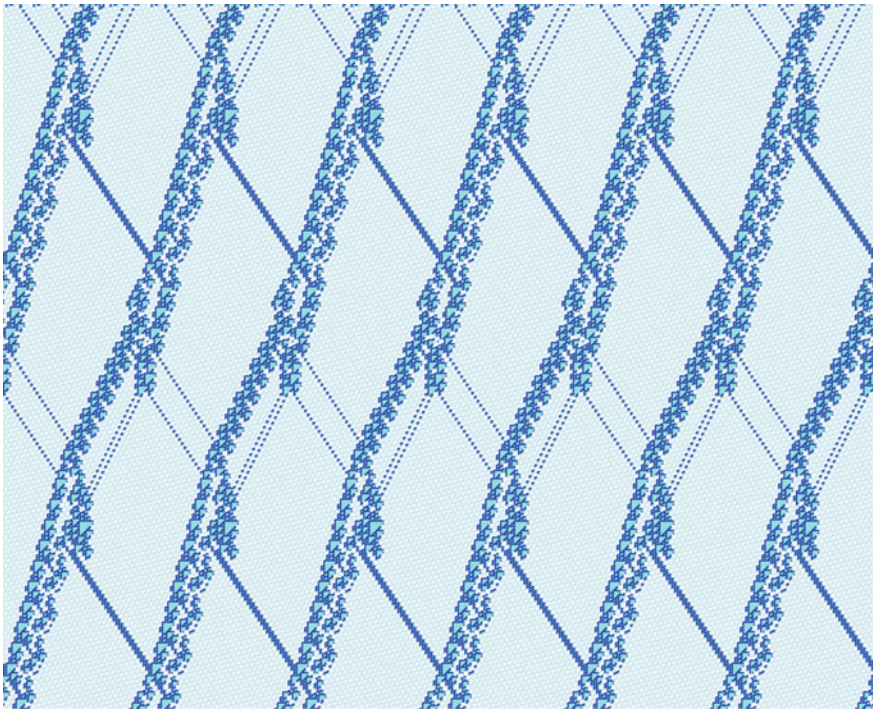

Fig. 15.20 Synchronisation of multiple collisions in rule 110 on a ring of 1,060 cells in 1,027 generations, starting with 50 particles from its initial condition

$$(p_A^+, p_A^+ \leftrightarrow p_{\bar{B}}^-, p_B^-, p_B^-) \rightarrow (p_{A^4}^+ \leftrightarrow p_E^+, p_{\bar{E}}^+), \text{ and}$$

$$(p_{A^4}^+ \leftrightarrow p_E^+, p_{\bar{E}}^+) \rightarrow (p_A^+, p_A^+ \leftrightarrow p_{\bar{B}}^-, p_B^-, p_B^-).$$

We can see that a beam routing representation allows for a design of collisions in cyclotrons. We employ the beam routing to implement the cyclic tag system in the CA rings. A construction of the cyclic tag system in rule 110 consists of three components (as was discussed in Sect. 15.4.2):

- The *left periodic part*, controlled by trains of 4_A^4 particles. This part is static. It controls the production of 0's and 1's.
- The *centre*, determining the initial value in the tape.
- The *right periodic part*, which has the data to process, adding a leader component which determines if data will be added or erased in the tape.

Left periodic part is defined by four trains of A^4 (Fig. 15.21c), trains of A^4 have three phases. The key point is to implement these components defining both distances and phases, because a distinct phase or a distance induces an undesirable reaction.

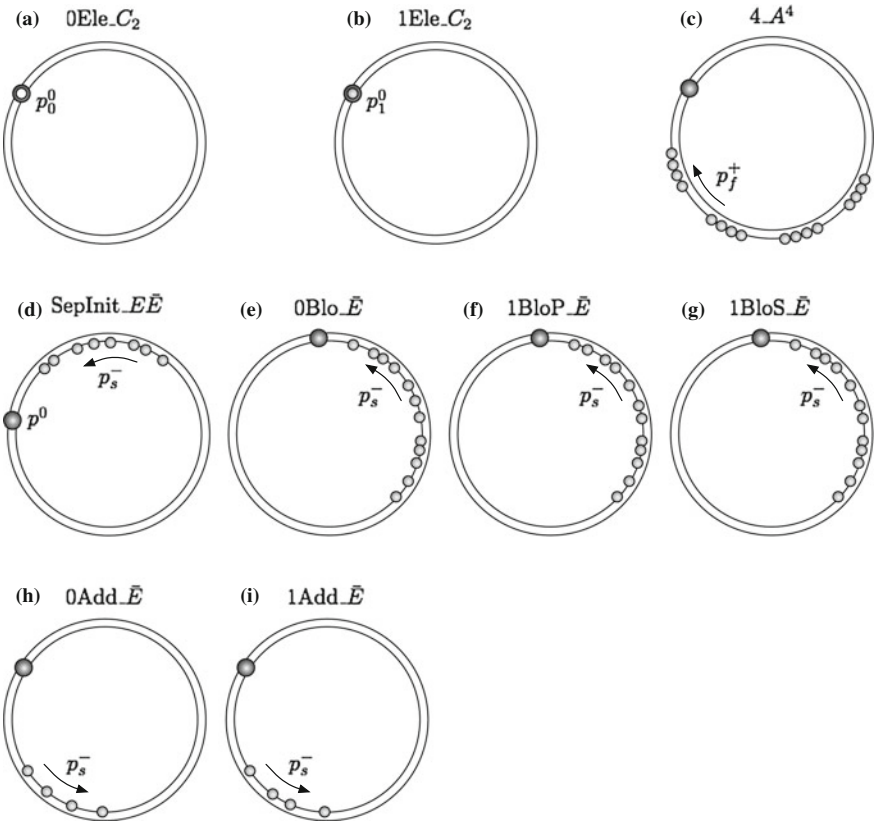

Fig. 15.21 The whole set of beam routing codification representing train of particles, to simulate a cyclic tag system. Each global state represents every component (set of particles) described in Sect. 15.4.1

The *central part* is represented by one value '1' on the tape across a train of four C_2 particles. The component 1Ele_C_2 (Fig. 15.21b) represents '1' and the component 0Ele_C_2 (Fig. 15.21a) represents '0' on the tape. The component 0Blo_$\bar{E}$ is formed by 12$\bar{E}$ particles. The construct includes two components to represent the state '1': 1BloP_$\bar{E}$ (Fig. 15.21f) named *primary* and 1BloS_$\bar{E}$ (Fig. 15.21g) named *standard*. A leader component SepInit_$E\bar{E}$ (Fig. 15.21d) is used to separate trains of data and to determine their incorporation into of the tape.

The components 1Add_$\bar{E}$ (Fig. 15.21i) and 0Add_$\bar{E}$ (Fig. 15.21h) are produced by two previous different trains of data. The component 1Add_$\bar{E}$ must be generated by a block 1BloP_$\bar{E}$ or by 1BloS_$\bar{E}$. This way, both components can yield the same element. The component 0Add_$\bar{E}$ is generated by a component 0Blo_$\bar{E}$ (Fig. 15.21e). For a complete and full description of such reproduction by phases f$_i$_1, see [38].

To get a cyclic tag system emulation in rule 110 by beam routings, we will use connections between beam routings as a finite state machine represented in Fig. 15.22.

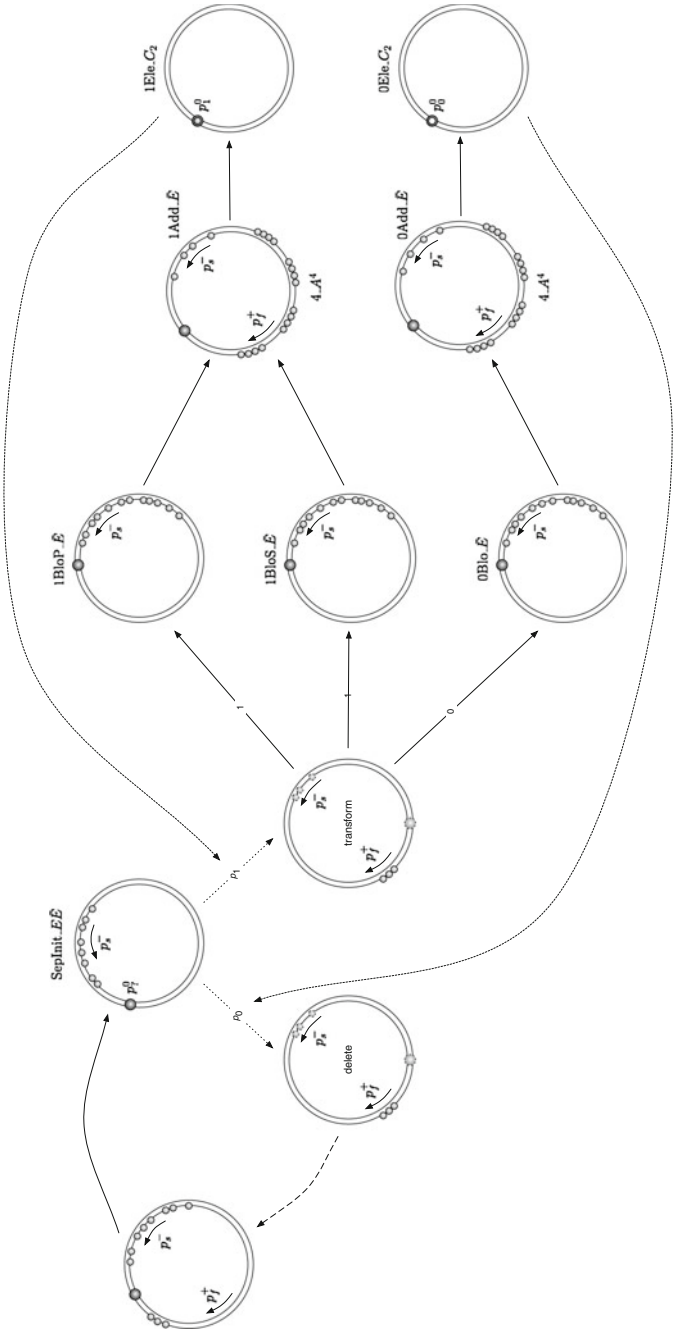

Fig. 15.22 Beam routing finite state machine simulating the cyclic tag system by state of cyclotrons representation

Transitions between beam routings means a change of state (transition function). Initial state is represented by the component 1Ele_C_2. A final state is not specified because it is determined by the state of the computation, i.e., a halt condition. Components 1Ele_C_2 and 0Ele_C_2 are compressed and shown as a dark circle, which represents the point of collision. Both components are made of four C_2 particles being at different distances. When a leader component (SepInit_$E\bar{E}$) is transformed, given previous binary value on the tape, it collides with $p^0_?$ component, i.e., a p^0_1 or p^0_0 element. If $p^0_?$ is '0', then a cascade of collisions starts to *delete* all components that come with three particles successively. If $p^0_?$ is '1' then a cascade of *transformations* dominated by additional particles p^0 is initiated, in order to reach the next leader component. Here, we have more variants because pre-transformed train of particles is encoded into binary values that are then written on the machine tape. If a component of particles is 1BloP_$\bar{E}$ or 1BloS_$\bar{E}$ this means that such a component will be transformed to one 1Add_$\bar{E}$ element. If a component of particles is 0Blo_$\bar{E}$, then such a component will be transformed to 0Add_$\bar{E}$ element. At this stage, when both components are prepared then a binary value is introduced on the tape, a 1Add_$\bar{E}$ element stores a 1 (1Ele_C_2), and a 0Add_$\bar{E}$ element stores a 0 (0Ele_C_2), which eventually will be deleted for the next leader component and starts a new cycle in the cyclic tag system. In bigger spaces these components will be represented just as a point in the evolution space: we describe this representation in the next section.

15.7 Cyclotrons

We use cyclotrons to explore large computational spaces where exact structures of particles are not relevant but only the interactions between the particles. There we can represent the particles as points and trains of particles as sequences of points. A 3D representation is convenient to understand the history of the evolutions, number of particles, positions, and collisions. Figure 15.23 shows a cyclotron evolving from a random initial configuration with 20,000 cells. Three stages are initialised in the evolution and the particles undergo successions of collisions in few first steps of evolution. The evolution is presented in a vertical orientation rotated 90 degrees. The present state shown is a front and its projection in three dimensions unveils the history and the evolution. Following this representation we can design a number of initial conditions to reproduce periodic patterns.[5]

Figure 15.24 shows a basic flip-flop pattern. We synchronise 16 particles $p_F \leftarrow p_B$, the basic collision takes place for two pairs of particles, a p_{D_1} particle and a train of p_{A^2} particles. The distance is determined by a factor of mod 14. A second reaction is synchronised with $p_{D_1} \leftarrow p_{A^2}$ to return back to the initial p_F and p_B particles. All 16 particles are forced in the same phase to guarantee an adequate distance, this distance is fixed in 64 copies of 14 cells (ether). Finally eight collisions are controlled every time simultaneously on an evolution space with 7,464 cells.

[5]The simulations are done in *Discrete Dynamics Lab* (DDLab, http://www.ddlab.org/) [59].

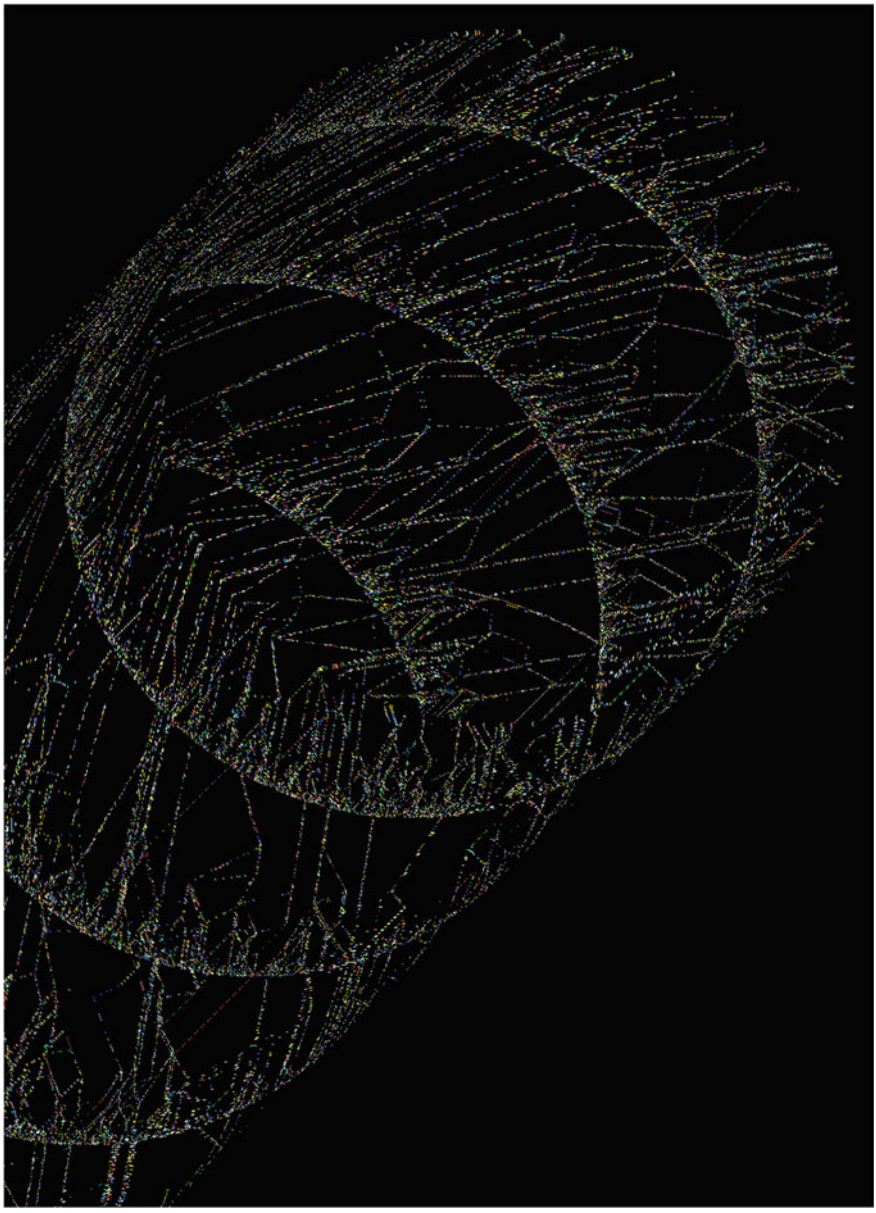

Fig. 15.23 ECA rule 110 particles traveling and colliding inside a cyclotron in a evolution space of 20,000 cells. A filter is selected for a better view of particles, each cyclotron initial stage in the history (three dimensional projection) is restarted randomly to illustrate the complex dynamics and variety of particles and collisions

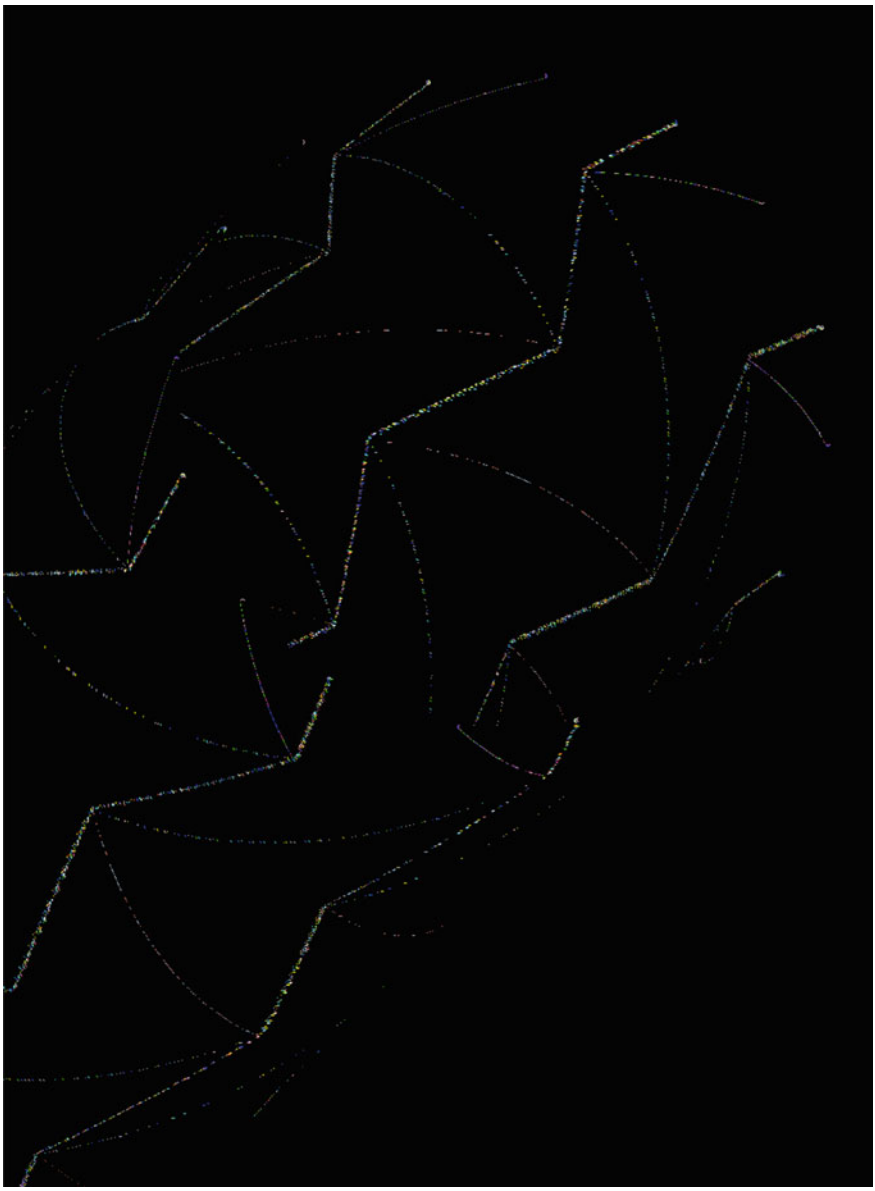

Fig. 15.24 Basic flip-flop oscillator implemented in a cyclotron with 7,464 cells in 25,000 generations. 16 particles $p_F \leftarrow p_B$ were coded

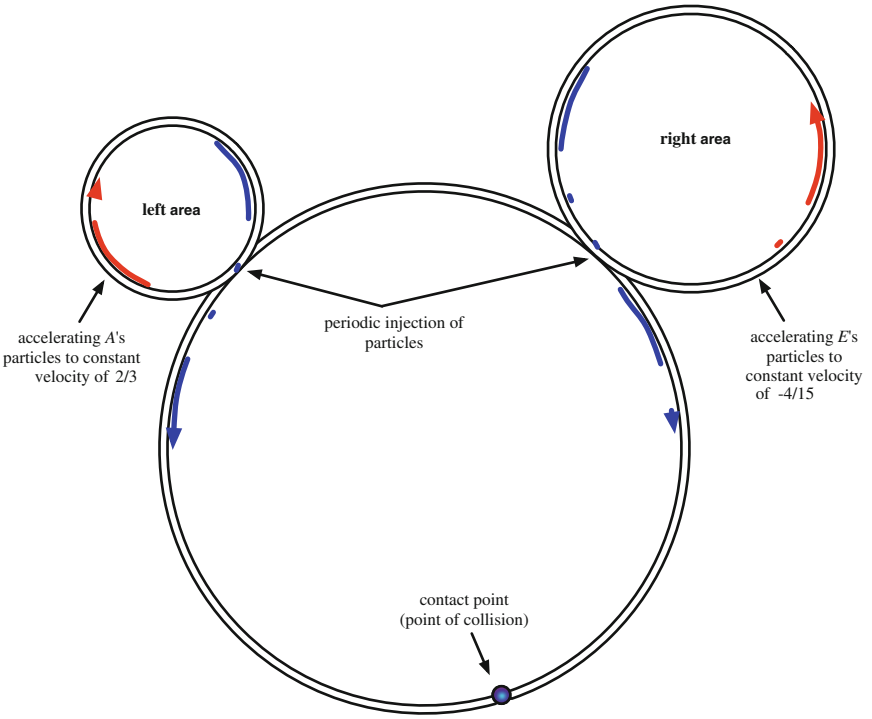

Fig. 15.25 Collider diagram

15.8 Collider Computing

A cyclic tag system consists of three main components. Each stage of computation can be represented with a cyclotron. A synchronisation of these cyclotrons injects beams of particles to a central main collider to obtain the collisions that will simulate a computation. The periodic representations of left and right cyclotrons are fixed. Diagram in Fig. 15.25 shows the dynamics of particles in a collider.

Left part Periodic area handle beams of three trains of four p_{A^4} particles, travelling from the left side with a constant velocity of $2/3$. This ring has 30,640 cells, the minimum interval between trains of particles is 649 copies of ether. Each beam of p_{A^4} can have three possible phases. The sequence of phases is periodic and fixed sequentially: $\{649e\text{-}4A^4(\mathrm{F}_i)\}^*$, for $1 \leq i \leq 3$ (Fig. 15.25 left area). Figure 15.26 shows a simulation of these periodic beams of $4p_{A^4(F_i)}$ particles.

Right part Periodic area handle beams of six trains of $12E$'s particles $(p_{E^n}, p_{\bar{E}})$, travelling from the right side with a constant velocity of $-4/15$. There are 12 particles related to a perfect square with $13{,}500^2$ possibilities to arrange inputs into the main collider. Interval between 12 particles is mod 14. Figure 15.27

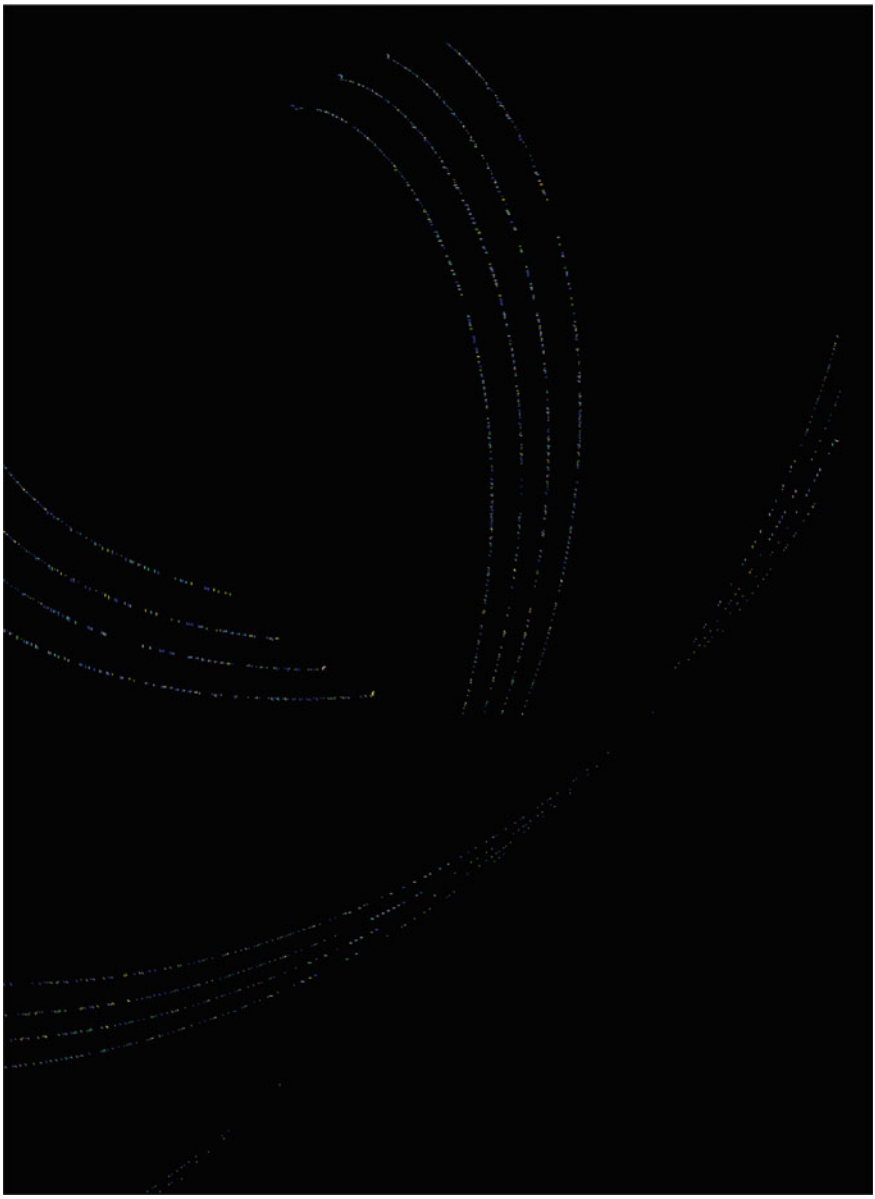

Fig. 15.26 Three beams of $4p_{A^4(F_i)}$ particles. Simulation is displayed in a vertical position to get a better view of particles' trajectories

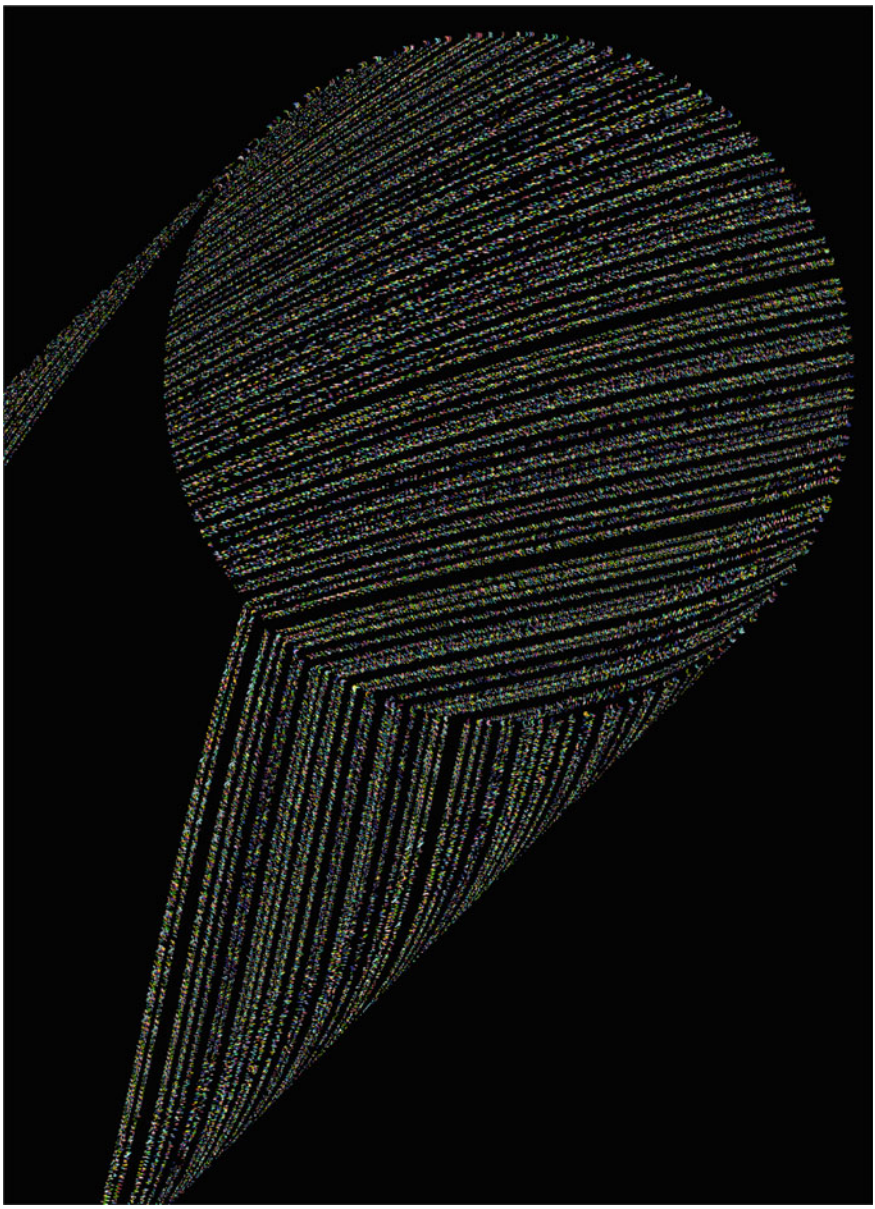

Fig. 15.27 A beam composed of six $12p_{Es}$ particles. Simulation is shown in a vertical position to get a better view of particles' trajectories. Interval between first and last particles can be any number mod 14

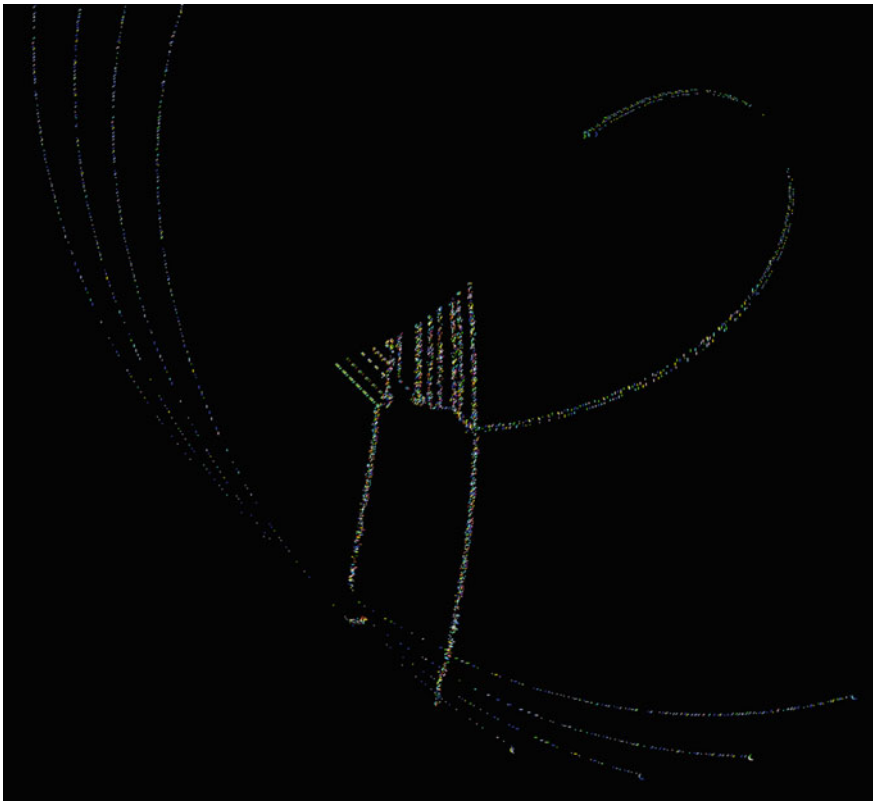

Fig. 15.28 First stage of collisions of the cyclic tag system. Solitonic interactions take place $4p_{A^4(F_3)}$ and two $p_{\bar{E}}$ particles. First symbol '1' on the type is deleted (*center*). The first separator is read and deleted

shows the whole set of 72 p_{Es} particles. The set contains leaders and separator components, and beams of particles that introduce '0's and '1's on the tape.

Center Initial state of particles starts with a '1' on the type of the cyclic tag system. Figure 15.28 shows the first stage of the collider. The system start with one '1' in the type (four vertical p_{Cs} particles), they are static particles that wait for the first beam of p_{Es} particles to arrive at the right side to delete this input and decode the next inputs. In this process two solitons emerge, but they do not affect the system and the first beam of $4p_{A^4(F_3)}$ particles without changing their states.

Figure 15.29 shows how a second symbol '1' is introduced in the collider. A leader component is deleted and the second binary data is prepared to collide later with the first beam of $4p_{A^4(F_3)}$ particles. Finally, the second '1' is represented for the vertical particles, as shown at the bottom of Fig. 15.29.

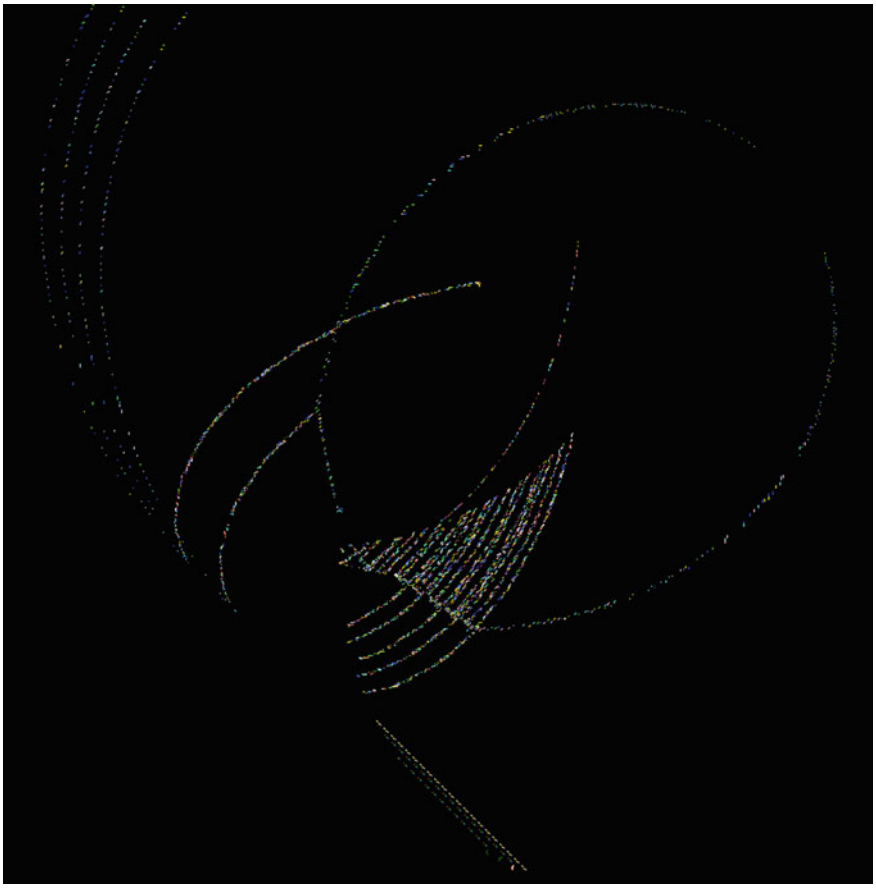

Fig. 15.29 This snapshot shows when a '1' is introduced in the type. A second beam of 12 $p_{\bar{E}}$ particles is coming to leave just spaced four $p_{\bar{E}}$ particles, these particles collide with one of $4p_{A^4(F_3)}$ particles. The result is four $4p_{Cs}$ particles at the bottom of the simulation that represent one '1' in the cyclic tag system type

Figure 15.30 shows how further symbols '0' and '1' are introduced in the system. They are coded with $p_{\bar{Es}}$ particles. Before the current '1' is introduced with $4p_{A^4(F_3)}$ particles, the next set of $4p_{A^4(F_3)}$ particles is prepared in advance.
Figure 15.31 shows the largest stage of the collider's working. A second beam of $4p_{A^4(F_1)}$ arrives. More beams of p_{Es} particles are introduced. Figure 15.32 displays a full cycle of beams of p_A and p_{Es} particles. All operations are performed at least once. The next set of particles is ready to continue with the next stage of the computation.

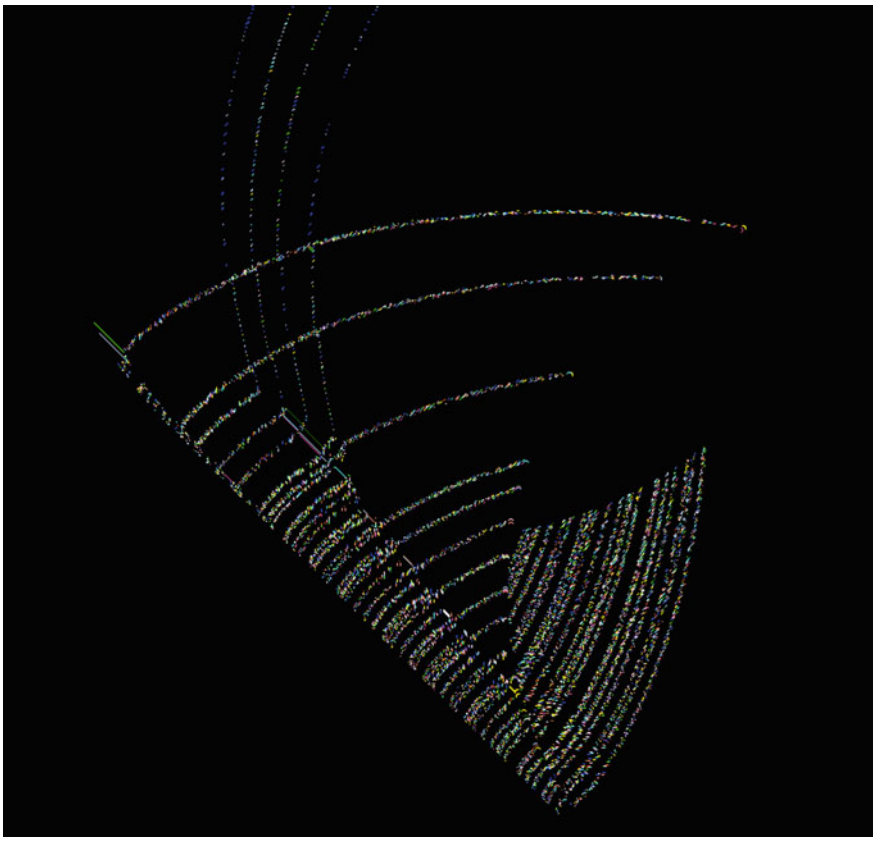

Fig. 15.30 This snapshot shows how a sequence of values '0' and '1' is precoded. You can see sequences of '0's and '1's, and p_{Es} particles travelling to from the *left* to the *right*

15.9 Discussion

The CA collider is a viable prototype of a collision-based computing device. It well compliments existing models of computing circuits based on particle collisions [15, 18, 23, 42, 45, 56, 60]. How complex is our design? With regarding to time complexity, rule simulates Turing machine a polynomial time and any step of rule 110 can be predicted in a polynomial time [46]. As to space complexity, left cyclotron in the collider is made of 30,640 cells and the right cyclotron of 5,864 cells. The main collider should have 61,280 cells to implement a full set of reactions; however, it is possible to reduce the number of cells in the main collider, because the first train of $4p_{A^4(F_i)}$ particles needs just 10,218 cells; and subsequent trains can be prepared while initial data are processed. Thus, the simulated collider have just thousands of cells not millions. The space complexity of the implemented cyclic tag systems has been reduced substantially [11, 12, 58].

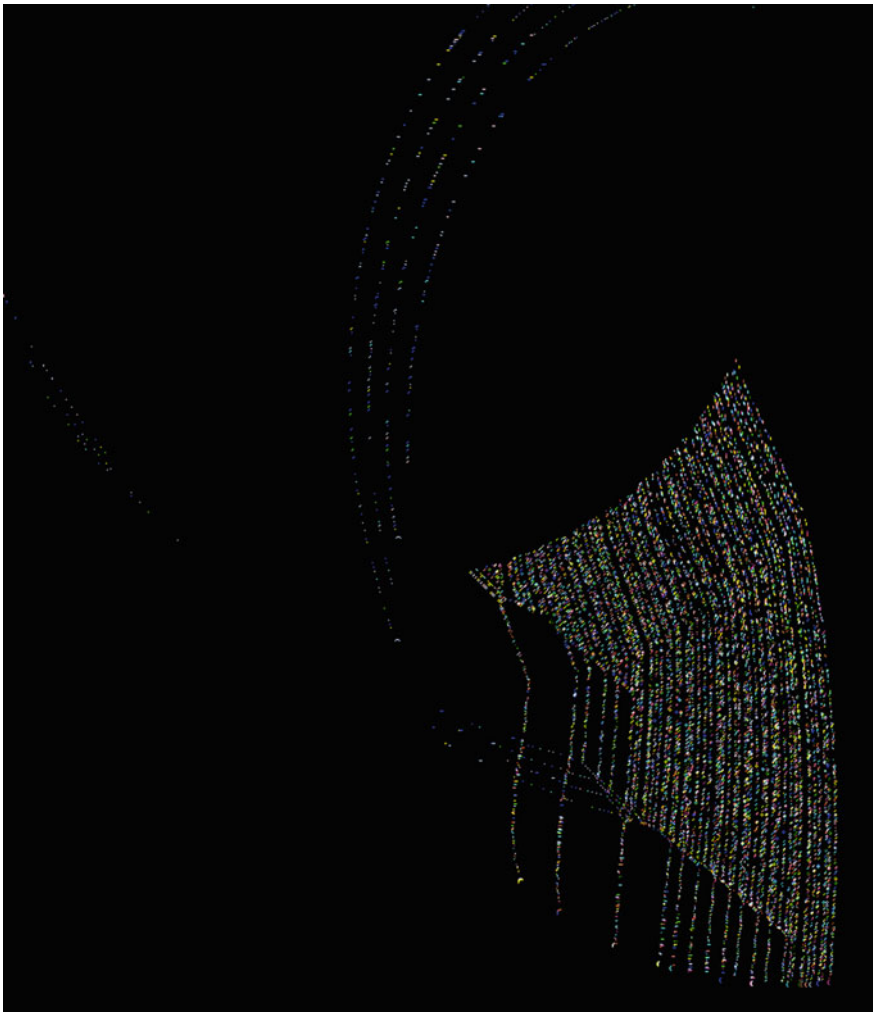

Fig. 15.31 This snapshot shows from another angle how binary values are introduced in the cyclic tag system. We can also see how a number of values are prepared to collide with beams of $4p_{A^4(F_i)}$ particles at the end of simulation

What are chances of implementing the CA collider model in physical substrates? A particle, or gliders, is a key component of the collider. The glider is a finite-state machine implementation of a propagation localisation. A solitary wave, or an impulse, propagating in a polymer chain could be a phenomenologically suitable analog of the glider. A wide range of polymer chains, both inorganic and organic, support solitons [1–3, 6, 9, 13, 14, 17, 19, 50, 52]. We believe actin filaments could make the most suitable substrate for implementation of a cyclic tag system via linked rings of CA colliders.

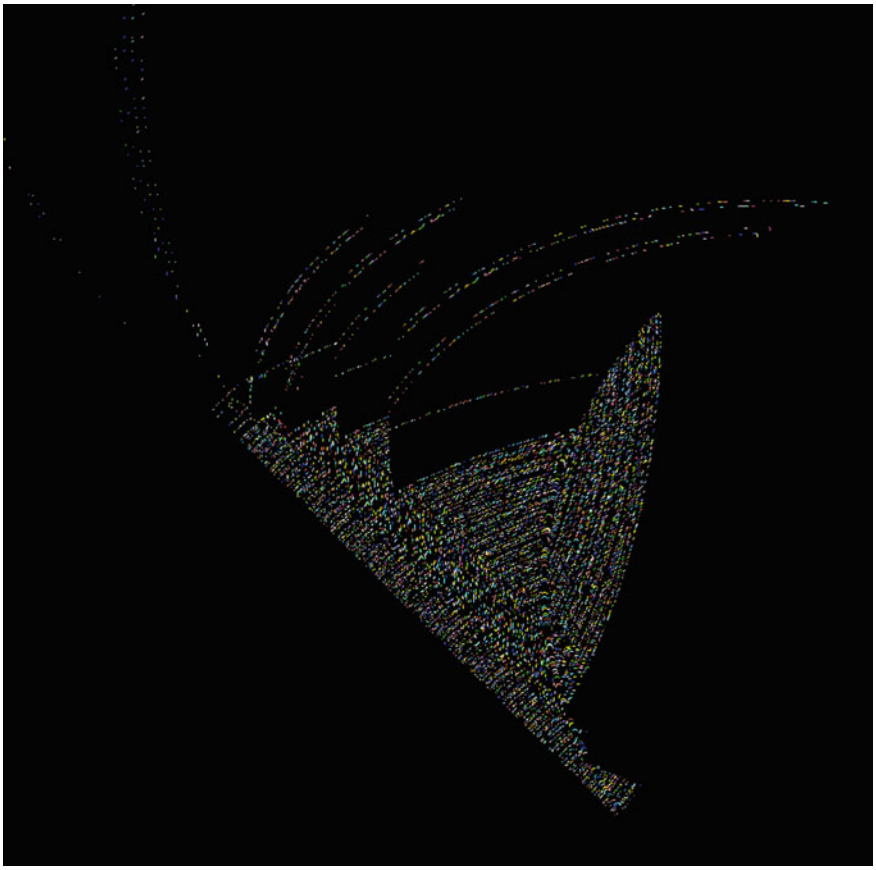

Fig. 15.32 This evolution displays a full cycle of beams of p_A and p_{Es} particles. In this snapshot we can see all necessary operations in the cyclic tag system: input values, deleting block of values, particles like solitons, and the next stage of the collider

An actin filament is a double spiral helix of globular protein units. Not only actin is a key element of a cell skeleton, and is responsible for a cell's motility, but actin networks is a sensorial, information processing and decision making system of cells. In [4] we proposed a model of actin filaments as two chains of one-dimensional binary-state semi-totalistic automaton arrays. We show that a rich family of travelling localisations is observed in automaton model of actin, and many of the localisation observed behave similarly to gliders in CA rule 110. The finite state machine model has been further extended to a quantum cellular automata model in [48]. We have shown that quantum actin automata can perform basic operations of Boolean logic, and implemented a binary adder. To bring more 'physical' meaning in our actin-computing concept we also employed the electrical properties of imitated actin filaments—resistance, capacitance, inductance — and found that it is possible to

implement logical gates via interacting voltage impulses [49]; voltage impulses in non-linear transmission wires are analogs of gliders in 1D CA. Clearly, having just actin is not enough: we must couple rings together, arrange physical initiation of solitons and their detection, and solve myriad of other experimental laboratory problems. That will be a scope of further studies.

References

1. Adamatzky, A.: Computing in Nonlinear Media and Automata Collectives. Institute of Physics Publishing, Bristol (2001)
2. Adamatzky, A. (ed.): Collision-Based Computing. Springer, London (2002)
3. Adamatzky, A.: Unconventional Computing. Human Brain Project Magazine (2015)
4. Adamatzky, A., Mayne, R.: Actin automata: phenomenology and localizations. Int. J. Bifurc. Chaos **25**(02), 1550030 (2015)
5. Arbib, M.A.: Theories of Abstract Automata. Prentice-Hall Series in Automatic Computation, Michigan (1969)
6. Bandyopadhyay, A., Pati, R., Sahu, S., Peper, F., Fujita, D.: Massively parallel computing on an organic molecular layer. Nat. Phys. **6**, 369–375 (2010)
7. Banks, E.R.: Information and transmission in cellular automata. PhD Dissertation. Massachusetts Institute of Technology, Cambridge (1971)
8. Berlekamp, E.R., Conway, J.H., Guy, R.K.: Winning Ways for your Mathematical Plays, vol. 2, Chap. 25, Academic Press, Cambridge (1982)
9. Bredas, J.L., Street, G.B.: Polarons, bipolarons, and solitons in conducting polymers. Acc. Chem. Res. **18**(10), 309–315 (1985)
10. Codd, E.F.: Cellular Automata. Academic Press, Inc., New York (1968)
11. Cook, M.: Universality in elementary cellular automata. Complex Syst. **15**(1), 1–40 (2004)
12. Cook, M.: A concrete view of Rule 110 computation. In: Neary, T., Woods, D., Seda, A.K., Murphy, N. (eds.) The Complexity of Simple Programs, pp. 31–55 (2008)
13. Davydov, A.S.: Solitons and energy transfer along protein molecules. J. Theor. Biol. **66**(2), 379–387 (1977)
14. Davydov, A.S.: Solitons in Molecular Systems. Springer, Heidelberg (1990)
15. Fredkin, E., Toffoli, T.: Design principles for achieving high-performance submicron digital technologies. In: Adamatzky, A. (ed.) Collision-Based Computing, pp. 27–46. Springer, London (2002)
16. Grünbaum, B., Shephard, G.C.: Tilings and Patterns. W. H. Freeman, New York (1986)
17. Heeger, A.J., Kivelson, S., Schrieffer, J.R., Su, W.P.: Solitons in conducting polymers. Rev. Mod. Phys. **60**(3), 781 (1988)
18. Hey, A.J.G.: Feynman and computation: exploring the limits of computers. Perseus Books, New York (1998)
19. Jakubowski, M.H., Steiglitz, K., Squier, R.: Computing with solitons: a review and prospectus. Multiple-Valued Logic **6**(5–6), 439–462 (2001)
20. Kudlek, M., Rogozhin, Y.: New small universal post machine. Lect. Notes Comput. Sci. **2138**, 217–227 (2001)
21. Kudlek, M., Rogozhin, Y.: Small universal circular post machine. Comput. Sci. J. Moldova. **9**(25), 34–52 (2001)
22. Lindgren, K., Nordahl, M.G.: Universal computation in simple one-dimensional cellular automata. Complex Syst. **4**, 229–318 (1990)
23. Lu, Y., Sato, Y., Amari, S.: Traveling bumps and their collisions in a two-dimensional neural field. Neural Comput. **23**(5), 1248–1260 (2011)
24. Margolus, N.H.: Physics-like models of computation. Physica D. **10**(1–2), 81–95 (1984)

25. Margolus, N.H.: Crystalline computation, In: Hey, A.J.G. (ed.) Feynman and computation: exploring the limits of computers, pp. 267–305. Perseus Books, New York (1998)
26. Margolus, N.H.: Universal cellular automata based on the collisions of soft spheres. In: Adamatzky, A. (ed.) Collision-Based Computing, pp. 107–134. Springer, London (2002)
27. Martínez, G.J., Adamatzky, A., Chen, F., Chua, L.: On soliton collisions between localizations in complex elementary cellular automata: rules 54 and 110 and beyond. Complex Syst. **21**(2), 117–142 (2012)
28. Martínez, G.J., Adamatzky, A., McIntosh, H.V.: Computing on rings. In: Zenil, H. (ed.) A Computable Universe: Understanding and Exploring Nature as Computation, pp. 283–302. World Scientific, Singapore (2012)
29. Martínez, G.J., Adamatzky, A., McIntosh, H.V.: Computing with virtual cellular automata collider. In: IEEE Proceedings of Science and Information Conference, pp. 62–68. London (2015). doi:10.1109/SAI.2015.7237127
30. Martínez, G.J., Adamatzky, A., Stephens, C.R., Hoeflich, A.: Cellular automaton supercolliders. Int. J. Mod. Phys. C. **22**(4), 419–439 (2011)
31. McIntosh, H.V.: Linear Cellular Automata Via de Bruijn diagrams, http://delta.cs.cinvestav.mx/~mcintosh/cellularautomata/Papers_files/debruijn.pdf. Cited 10 August 1991
32. McIntosh, H.V.: Rule 110 as it Relates to the Presence of Gliders, http://delta.cs.cinvestav.mx/~mcintosh/comun/RULE110W/RULE110.html. Cited 14 May 2001
33. McIntosh, H.V.: A Concordance for Rule 110, http://delta.cs.cinvestav.mx/~mcintosh/cellularautomata/Papers_files/ccord.pdf. Cited 14 May 2002
34. McIntosh, H.V.: One Dimensional Cellular Automata. Luniver Press, Bristol (2009)
35. Martínez, G.J., McIntosh, H.V.: ATLAS: Collisions of Gliders like Phases of Ether in Rule 110, http://uncomp.uwe.ac.uk/genaro/Papers/Papers_on_CA_files/ATLAS/bookcollisions.html. Cited 14 August 2001
36. Martínez, G.J., McIntosh, H.V., Seck, J.C.S.T.: Gliders in Rule 110. Int. J. Unconv. Comput. **2**(1), 1–49 (2006)
37. Martínez, G.J., McIntosh, H.V., Mora, J.C.S.T., Vergara, S.V.C.: Determining a regular language by glider-based structures called phases fi_1 in Rule 110. J. Cell. Automata **3**(3), 231–270 (2008)
38. Martínez, G.J., McIntosh, H.V., Mora, J.C.S.T., Vergara, S.V.C.: Reproducing the cyclic tag system developed by Matthew Cook with Rule 110 using the phases f_1_1. J. Cell. Automata **6**(2–3), 121–161 (2011)
39. Martínez, G.J., McIntosh, H.V., Mora, J.C.S.T., Vergara, S.V.C.: Rule 110 objects and other collision-based constructions. J. Cell. Automata **2**(3), 219–242 (2007)
40. Martínez, G.J., Seck-Tuoh-Mora, J.C., Zenil, H.: Computation and Universality: Class IV versus Class III Cellular Automata. J. Cell. Automata **7**(5–6), 393–430 (2013)
41. Mills, J.W.: The nature of the extended analog computer. Physica D. **237**, 1235–1256 (2008)
42. Minsky, M.: Computation: Finite and Infinite Machines. Prentice Hall, Upper Saddle River (1967)
43. Morita, K.: Simple universal one-dimensional reversible cellular automata. J. Cell. Automata **2**, 159–166 (2007)
44. Morita, K.: Simulating reversible Turing machines and cyclic tag systems by one-dimensional reversible cellular automata. Theor. Comput. Sci. **412**, 3856–3865 (2011)
45. Margolus, N., Toffoli, T., Vichniac, G.: Cellular-automata supercomputers for fluid dynamics modeling. Phys. Rev. Lett. **56**(16), 1694–1696 (1986)
46. Neary, T., Woods, D.: P-completeness of cellular automaton Rule 110. Lect. Notes Comput. Sci. **4051**, 132–143 (2006)
47. Ninagawa, S., Martínez, G.J.: Compression-based analysis of cyclic tag system emulated by Rule 110. J. Cell. Automata **9**(1), 23–35 (2014)
48. Siccardi, S., Adamatzky, A.: Actin quantum automata: communication and computation in molecular networks. Nano Commun. Netw. **6**(1), 15–27 (2015)
49. Siccardi, S., Tuszynski, J. A., Adamatzky, A.: Boolean gates on actin filaments. Phys. Lett. A (2015)

50. Scott, A.C.: Dynamics of Davydov solitons. Phys. Rev. A **26**(1), 578 (1982)
51. Smith III, A.R.: Simple computation-universal cellular spaces. J. Assoc. Comput. Mach. **18**, 339–353 (1971)
52. Toffoli, T.: Non-conventional computers. In: Webster, J. (ed.) Encyclopedia of Electrical and Electronics Engineering, vol. 14, pp. 455–471. Wiley, New York (1998)
53. Toffoli, T.: Symbol super colliders. In: Adamatzky, A. (ed.) Collision-Based Computing, pp. 1–23. Springer, London (2002)
54. von Neumann, J.: Theory of Self-reproducing Automata (edited and completed by A.W. Burks), University of Illinois Press, Urbana and London (1966)
55. Voorhees, B.H.: Computational analysis of one-dimensional cellular automata. In: World Scientific Series on Nonlinear Science, Series A, vol. 15. World Scientific, Singapore (1996)
56. Wolfram, S.: Cellular automata supercomputing. In: Wilhelmson, R.B. (ed.) High Speed Computing: Scientific Applications and Algorithm Design. pp. 40–48. University of Illinois Press, Champaign (1988)
57. Wolfram, S.: Cellular Automata and Complexity. Addison-Wesley Publishing Company, Colorado (1994)
58. Wolfram, S.: A New Kind of Science. Wolfram Media Inc, Champaign (2002)
59. Wuensche, A.: Exploring Discrete Dynamics. Luniver Press, Bristol (2011)
60. Zenil, H. (ed.): A Computable Universe. World Scientific Press, Singapore (2012)