A Simple Universal Logic Element and Cellular Automata for Reversible Computing

Kenichi Morita

Hiroshima University, Faculty of Engineering,
Higashi-Hiroshima, 739-8527, Japan
morita@ke.sys.hiroshima-u.ac.jp
http://www.ke.sys.hiroshima-u.ac.jp/~morita

Abstract. Reversible computing is a paradigm of computation that reflects physical reversibility, and is considered to be important when designing a logical devices based on microscopic physical law in the near future. In this paper, we focus on a problem how universal computers can be built from primitive elements with very simple reversible rules. We introduce a new reversible logic element called a “rotary element”, and show that any reversible Turing machine can be realized as a circuit composed only of them. Such reversible circuits work in a very different fashion from conventional ones. We also discuss a simple reversible cellular automaton in which a rotary element can be implemented.

1 Introduction

Recently, various computing models that directly reflect laws of Nature have been proposed and investigated. They are quantum computing (e.g., [2,4]), DNA computing (e.g., [11]), reversible computing (e.g., [1,2,3]), and so on. Reversible computing is a model reflecting physical reversibility, and has been known to play an important role when studying inevitable power dissipation in a computing process. Until now, several reversible systems such as reversible Turing machines, reversible cellular automata, and reversible logic circuits have been investigated.

A logic gate is called reversible if its logical function is one-to-one. A reversible logic circuit is a one constructed only of reversible gates. There are “universal” reversible gates in the sense that any logic circuit (even if it is irreversible) can be embedded in a circuit composed only of them. A Fredkin gate [3] and a Toffoli gate [12] are typical universal reversible gates having 3 inputs and 3 outputs.

A rotary element (RE), which is also a reversible logic element, was introduced by Morita, Tojima, and Imai [8]. They proposed a simple model of 2-D reversible cellular automaton called P_4 in which any reversible two-counter machine can be embedded in a finite configuration. They showed that an RE and a position marker as well as several kinds of signal routing (wiring) elements can be implemented in this cellular space, and gave a construction method of a reversible counter machine out of these elements. An RE is a 4-input 4-output reversible element, and has two states. It is a kind of switching element that changes the path of an input signal depending on its state.

© Springer-Verlag Berlin Heidelberg 2001

In this paper, we show that any reversible Turing machine can be realized as a circuit composed only of REs in a systematic manner. In spite of the simplicity of an RE, the circuit obtained here is relatively concise and its operation is easy to be understood. Especially, there is no need to supply a clock signal to this circuit, and it contrasts sharply with a conventional logic circuit. We also discuss a simple reversible cellular automaton called P_3 proposed in [10], in which an RE can be embedded.

2 Preliminaries

2.1 A Reversible Sequential Machine

We first give a definition of a reversible sequential machine (RSM). As we shall see below, a reversible logic circuit composed of REs, as well as an RE itself, can be formulated as an RSM.

A *reversible sequential machine* (RSM) is a system defined by

$$M = (Q, \Sigma, \Gamma, q_1, \delta),$$

where Q is a finite non-empty set of states, Σ and Γ are finite non-empty sets of input and output symbols, respectively, and $q_1 \in Q$ is an initial state. $\delta : Q \times \Sigma \rightarrow Q \times \Gamma$ is a one-to-one mapping called a move function (hence $|\Sigma| \leq |\Gamma|$).

We can see that an RSM is "reversible" in the sense that, from the present state and the output of M, the previous state and the input are determined uniquely. A variation of an RSM $M = (Q, \Sigma, \Gamma, \delta)$, where no initial state is specified, is also called an RSM for convenience.

2.2 A Rotary Element and a Reversible Logic Circuit

A *rotary element* (RE) is a logic element depicted in Fig. 1. It has four input lines $\{n, e, s, w\}$ and four output lines $\{n', e', s', w'\}$, and has two states called H-state and V-state. All the values of inputs and outputs are either 0 or 1, i.e., $(n, e, s, w), (n', e', s', w') \in \{0, 1\}^4$. However, we restrict the input (and output) domain as $\{(0, 0, 0, 0), (1, 0, 0, 0), (0, 1, 0, 0), (0, 0, 1, 0), (0, 0, 0, 1)\}$, i.e., at most one "1" appears as an input (output) at a time. Hence, the operation of an RE is left undefined for the cases that signal 1's are given to two or more input lines. In order to explain its operation, we employ the following intuitive interpretation for it. Signals 1 and 0 are interpreted as existence and non-existence of a particle. An RE has a "rotating bar" to control the moving direction of a particle. When no particle exists, nothing happens on the RE. If a particle comes from a direction parallel to the rotating bar, then it goes out from the output line of the opposite side (i.e., it goes straight ahead) without affecting the direction of the bar (Fig. 2 (a)). On the other hand, if a particle comes from a direction orthogonal to the bar, then it makes a right turn, and rotates the bar by 90 degrees counterclockwise (Fig. 2 (b)).

We can define an RE as an RSM M_{RE}. Since the input $(0, 0, 0, 0)$ has no effect on an RE, we omit it from the input alphabet for convenience. Further,

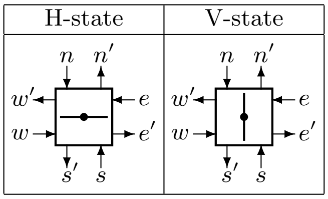


Fig. 1. Two states of a rotary element.

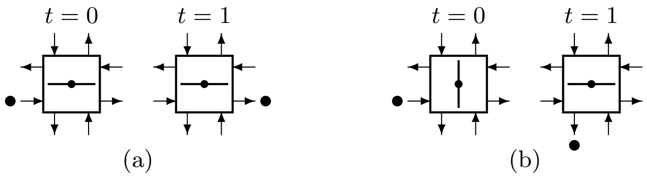


Fig. 2. Operations of a rotary element: (a) the parallel case (i.e., the coming direction of a particle is parallel to the rotating bar), and (b) the orthogonal case.

we denote the sets of input and output alphabets of M_{RE} as $\{n, e, s, w\}$ and $\{n', e', s', w'\}$ instead of $\{(1,0,0,0), (0,1,0,0), (0,0,1,0), (0,0,0,1)\}$.

M_{RE} is defined by

$$M_{\mathrm{RE}} = (\{⊟, ◫\}, \{n, e, s, w\}, \{n', e', s', w'\}, \delta_{\mathrm{RE}}),$$

where the move function δ_{RE} is shown in Table 1 (for instance, if the present state is ⊟ and a particle comes from the input line n, then the state becomes ◫ and a particle goes out from w'). We can see that the operation of an RE is reversible. It has also a bit-conserving property, i.e., the number of 1's is conserved between inputs and outputs, since a particle is neither annihilated nor newly created.

Table 1. The move function δ_{RE} of a rotary element M_{RE}.

Present state	Input			
	n	e	s	w
H-state: ⊟	◫ w'	⊟ w'	◫ e'	⊟ e'
V-state: ◫	◫ s'	⊟ n'	◫ n'	⊟ s'

An *RE-circuit* is a one composed only of REs satisfying the following condition: each output of an RE can be connected at most one input of some other (or may be the same) RE, i.e., "fan-out" of an output is not allowed. It is also easy to formulate each RE-circuit as an RSM.

2.3 Logical Universality of a Rotary Element

It has been shown that a Fredkin gate can be realized by an RE circuit as in in Fig.3 [9]. Since a Fredkin gate is logically universal [3], an RE is also universal. However, in the following, we do not use this construction method.

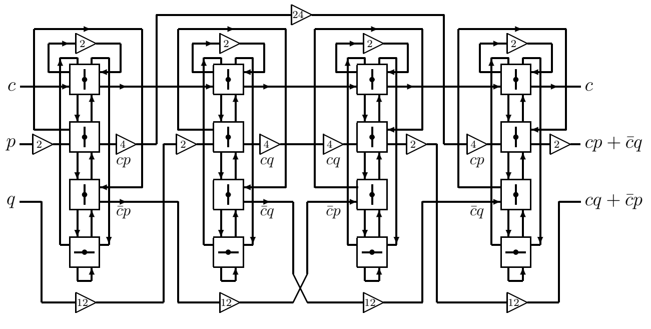


Fig. 3. Realization of a Fredkin gate as an RE circuit [9]. Small triangles are delay elements, where the number written inside of each triangle indicates its delay time. (Note that delay elements can also be implemented by using REs as in Fig.2(a).)

3 A Reversible Turing Machine Composed of REs

3.1 A Reversible Turing Machine

We first define a one-tape Turing machine and its reversible version. We use quadruple formalism [1] of a Turing machine in order to define a reversible one.

Definition 1. *A* one-tape Turing machine *(TM) is a system defined by*

$$T = (Q, S, q_0, q_f, s_0, \delta),$$

where Q is a non-empty finite set of states, S is a non-empty finite set of symbols, q_0 is an initial state ($q_0 \in Q$), q_f is a final state ($q_f \in Q$), s_0 is a special blank symbol ($s_0 \in S$), and δ is a move relation which is a subset of $(Q \times S \times S \times Q) \cup (Q \times \{/\} \times \{-, 0, +\} \times Q)$. Each element of δ is called a quadruple, and either of the form $[q_r, s, s', q_t] \in (Q \times S \times S \times Q)$ or $[q_r, /, d, q_t] \in (Q \times \{/\} \times \{-, 0, +\} \times Q)$. The symbols "$-$", "$0$", and "$+$" denote "left-shift", "zero-shift", and "right-shift", respectively. $[q_r, s, s', q_t]$ means that if T reads the symbol s in the state q_r, then write s' and go to the state q_t. $[q_r, /, d, q_t]$ means that if T is in the state q_r, then shift the head to the direction d and go to the state q_t.

Let $\alpha_1 = [p_1, b_1, c_1, p'_1]$ and $\alpha_2 = [p_2, b_2, c_2, p'_2]$ be two quadruples in δ. We say α_1 and α_2 overlap in domain *iff*

$$p_1 = p_2 \ \wedge \ [\ b_1 = b_2 \ \vee \ b_1 = / \ \vee \ b_2 = /\ ].$$

We say α_1 and α_2 overlap in range *iff*

$$p'_1 = p'_2 \ \wedge \ [\ c_1 = c_2 \ \vee \ b_1 = / \ \vee \ b_2 = /\ ].$$

A quadruple α is said to be deterministic *(in δ) iff there is no other quadruple in δ with which α overlaps in domain. On the other hand, α is said to be* reversible *(in δ) iff there is no other quadruple in δ with which α overlaps in range. T is called deterministic (reversible, respectively) iff every quadruple in δ is deterministic (reversible). (In what follows, we consider only deterministic Turing machines.)*

Theorem 1. [1] *For any one-tape Turing machine, there is a reversible three-tape Turing machine which simulates the former.*

Theorem 2. [6] *For any one-tape Turing machine, there is a reversible (semi-infinite) one-tape two-symbol Turing machine which simulates the former.*

Theorem 1 shows computation-universality of deterministic reversible three-tape Turing machines. Theorem 2 is useful for giving a simple construction method of a reversible TM out of REs.

3.2 Constructing a Tape Unit

In order to construct a reversible one-tape two-symbol Turing machine out of REs, we first deign a *tape cell module* (TC-module) as an RE-circuit. A TC-module simulates one tape square of a two-symbol reversible TM. It can store a symbol 0 or 1 written on the square and an information whether the tape head is on the square or not. It is formulated as an RSM M_{TC} defined below.

$$\begin{aligned}
&M_{\mathrm{TC}} = (Q_{\mathrm{TC}}, \varSigma_{\mathrm{TC}}, \varGamma_{\mathrm{TC}}, \delta_{\mathrm{TC}})\\
&Q_{\mathrm{TC}} = \{(h,s) \mid h,s \in \{0,1\}\}\\
&\varSigma_{\mathrm{TC}} = \{R, Rc0, Rc1, W, Wc, SR, SRI, SRc, SL, SLI, SLc, E0, E1, Ec\}\\
&\varGamma_{\mathrm{TC}} = \{x' \mid x \in \varSigma_{\mathrm{TC}}\}
\end{aligned}$$

δ_{TC} is defined as follows, where $s \in \{0,1\}$ and $y \in \varSigma_{\mathrm{TC}} - \{SRI, SLI\}$:

$$\begin{aligned}
\delta_{\mathrm{TC}}((0,s),y) &= ((0,s),y') && (1)\\
\delta_{\mathrm{TC}}((0,s),SRI) &= ((1,s),SRc') && (2)\\
\delta_{\mathrm{TC}}((0,s),SLI) &= ((1,s),SLc') && (3)\\
\delta_{\mathrm{TC}}((1,0),R) &= ((1,0),Rc0') && (4)\\
\delta_{\mathrm{TC}}((1,1),R) &= ((1,1),Rc1') && (5)\\
\delta_{\mathrm{TC}}((1,0),W) &= ((1,1),Wc') && (6)\\
\delta_{\mathrm{TC}}((1,1),W) &= ((1,0),Wc') && (7)\\
\delta_{\mathrm{TC}}((1,s),SR) &= ((0,s),SRI') && (8)\\
\delta_{\mathrm{TC}}((1,s),SL) &= ((0,s),SLI') && (9)\\
\delta_{\mathrm{TC}}((1,0),E0) &= ((1,0),Ec') && (10)\\
\delta_{\mathrm{TC}}((1,1),E1) &= ((1,1),Ec') && (11)
\end{aligned}$$

M_{TC} has the state set $\{(h,s) \mid h,s \in \{0,1\}\}$. The state (h,s) represents that the symbol s is written on the tape square, and that the tape head is on this cell (if $h=1$) or not (if $h=0$). There are 14 input and 14 output symbols. In the following construction of an RE-circuit, there are also 14 input and 14

Table 2. Fourteen input lines of an RSM M_{TC}.

Signal Line	Meaning of a Signal
R	Read the symbol at the head position.
$Rc0$	A read operation is completed with the result 0.
$Rc1$	A read operation is completed with the result 1.
W	Write a complementary symbol at the head position (i.e., if the current symbol is 1, then write 0, else write 1).
Wc	A write operation is completed.
SL	Shift the head position to the left.
SLI	Set the head position at the cell immediately to the left.
SLc	A left-shift operation is completed.
SR	Shift the head position to the right.
SRI	Set the head position at the cell immediately to the right.
SRc	A right-shift operation is completed.
$E0$	Reversibly erase the information 0 kept by the finite-state control by referring the symbol 0 at the head position.
$E1$	Reversibly erase the information 1 kept by the finite-state control by referring the symbol 1 at the head position.
Ec	A reversible erasure (i.e., merge) operation is completed.

output lines corresponding to these symbols. The roles of the 14 input lines are shown in Table 2. To each input line $x \in \Sigma_{\mathrm{TC}}$ there corresponds an output line $x' \in \Gamma_{\mathrm{TC}}$ in the one-to-one manner.

M_{TC} acts as follows according to the move function δ_{TC}.

(I) The case $h = 0$: (i) If a signal 1 arrives at the input line $y \in \Sigma_{\mathrm{TC}} - \{SRI, SLI\}$, then it simply goes out from y' without affecting the state of M_{TC} by (1). (ii) If a signal arrives at the line SRI (or SLI, respectively), then the head position is set to this tape cell, and a completion (i.e., response) signal for the shift-right (shift-left) operation goes out from the line SRc' (SLc') by (2) (or (3)).

(II) The case $h = 1$: (i) If a signal 1 arrives at the line R and if $s = 0$ (or $s = 1$, respectively), then a response signal goes out from $Rc0'$ ($Rc1$) by (4) (or (5)), performing a read operation. (ii) If a signal arrives at the line W and if $s = 0$ (or $s = 1$, respectively), then s is set to 1 (or 0) and a response signal goes out from W' by (6) (or (7)), performing an operation of writing a complementary symbol. (iii) If a signal arrives at the line SR (or SL, respectively), then h is set to 0 and a response signal goes out from SRI' (or SLI') by (8) (or (9)). (iv) If a signal arrives at the line $E0$ (or $E1$, respectively) and $s = 0$ ($s = 1$), then a response signal goes out from Ec' by (10) (or (11)), performing a "reversible erasure" of one bit of information by referring the symbol at the head position (usage of this operation is explained later).

Fig. 4 shows a TC-module, a realization of a tape cell M_{TC} as an RE-circuit. In order to explain the operations of a TC-module, it is convenient to consider an *RE-column* shown in Fig. 5 (a). It consists of $k+1$ REs, and has $2k$ input lines and $2k$ output lines ($k \in \{1, 2, \cdots\}$). We assume all REs except the bottom one (indicated by x) are initially set to V-states (the bottom RE may be either H- or

V-state). Further assume a signal 1 is given to at most one input line. Then, an RE-column acts like an RSM shown in Fig. 5 (b), where the input and output alphabets are $\{l_1, \cdots, l_k, r_1, \cdots, r_k\}$ and $\{l'_1, \cdots, l'_k, r'_1, \cdots, r'_k\}$, respectively, and the state set is {⊟, ⎅} which matches that of the bottom RE. Though there are $k+1$ REs, we can consider the RE-column as if a two-state machine. Because all the REs except the bottom one are reset to V-states when a signal 1 goes out from some output line. For example, if it is in the state ⊟ and the input is l_j, then after *some* time steps the state becomes ⎅ and gives an output l'_j. In what follows, we write $x = 1$ (or *marked*) if it is ⊟ , and $x = 0$ (or *unmarked*) if it is ⎅ .

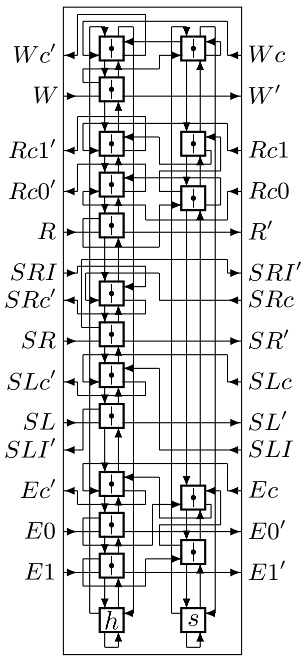
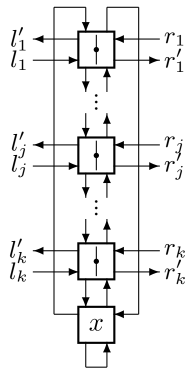


Fig. 4. A TC-module: a realization of a tape cell M_{TC} as an RE-circuit.

A TC-module consists of two RE-columns, i.e., left and right ones which correspond to h and s respectively. We can verify that the TC-module acts as M_{TC} by testing all the cases of inputs and states. For example, consider the case that a signal is given to R. If $h = 0$ then the signal eventually goes out from R' without affecting the states h and s. If $h = 1$, the signal first sets the state h to 0, and then enters the third RE of the right RE-column. There are two sub-cases: $s = 0$ and $s = 1$. If $s = 0$, the signal goes out from the right side of the third RE without affecting s, and enters the fourth RE of the left RE-column. It then sets h to 1, and finally goes out from $Rc0'$. If $s = 1$, the signal first goes out from the left side of the third RE setting s to 0, and enters the second RE of the right



(a)

State x	Input l_j	Input r_j
H-state: ⊟	◫ l'_j	⊟ l'_j
V-state: ◫	◫ r'_j	⊟ r'_j

(b)

Fig. 5. (a) An RE-column, and (b) its move function ($j \in \{1, \cdots, k\}$).

RE-column. It restores both s and h to 1 and finally goes out from $Rc1'$. It is also easy to verify other cases.

An entire circuit for a tape unit can be obtained by placing infinite number of TC-modules in a row, and connecting input and output lines between adjacent TC-modules as shown in the right-half of Fig. 8. From the move function of the TC-module, we can see that by giving a signal 1 to one of the lines $R, W, SL, SR, E0$, or $E1$ of the leftmost TC-module in the semi-infinite array, it can correctly simulate each operation on the tape unit.

3.3 Constructing a Finite-State Control

We now design an RE-circuit that simulates a finite-state control of a given one-tape two-symbol reversible Turing machine T. This circuit is called a *finite-state control module* (FC-module). It is constructed in a similar manner given in [8].

Each quadruple of T performs either a read/write (i.e., R and/or W) operation, or a head-shift (SL or SR) operation to a tape unit. In addition, in order to realize a FC-module as an RE-circuit, we need a "reversible erasure" operation that erases an information kept by the FC-module. This is due to the following reason. When a read operation is performed, the information of the read symbol (0 or 1) is distinguished by some different states of the FC-module of T. If such an information is never erased, then the total amount of the information grows indefinitely, and thus an FC-module needs infinite number of states. Hence, such information should be reversibly erased (i.e., merged) by referring the read symbol itself each time a read operation is performed. This operation is done by giving a signal to the line $E0$ or $E1$ of the leftmost TC-module.

To explain a construction method of an FC-module, we consider a simple example of a one-tape two-symbol reversible Turing machine

$$T_{2n} = (\{q_1, \cdots, q_{16}\}, \{0, 1\}, q_1, q_{16}, 0, \delta_{2n})$$

having the following quadruples as δ_{2n}.

$[q_1, 0,\ 0,\ q_2]$	$[q_4, /, +, q_5]$	$[q_8, /, +, q_9]$	$[q_{12}, /, -, q_{13}]$
$[q_2, /, +, q_3]$	$[q_5, 0,\ 0, q_6]$	$[q_9, 0,\ 1, q_{10}]$	$[q_{13}, 0,\ 0, q_{14}]$
$[q_3, 0,\ 0, q_{16}]$	$[q_5, 1,\ 1, q_4]$	$[q_{10}, /, +, q_{11}]$	$[q_{13}, 1,\ 1, q_{12}]$
$[q_3, 1,\ 0,\ q_4]$	$[q_6, /, +, q_7]$	$[q_{11}, 0,\ 0, q_{12}]$	$[q_{14}, /, -, q_{15}]$
	$[q_7, 0,\ 1, q_8]$		$[q_{15}, 0,\ 1,\ q_2]$
	$[q_7, 1,\ 1, q_6]$		$[q_{15}, 1,\ 1, q_{14}]$

It is easy to verify that T_{2n} is reversible. It computes the function $f(n) = 2n$ for a unary input as shown in Fig. 6.

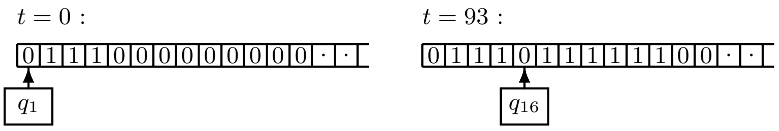
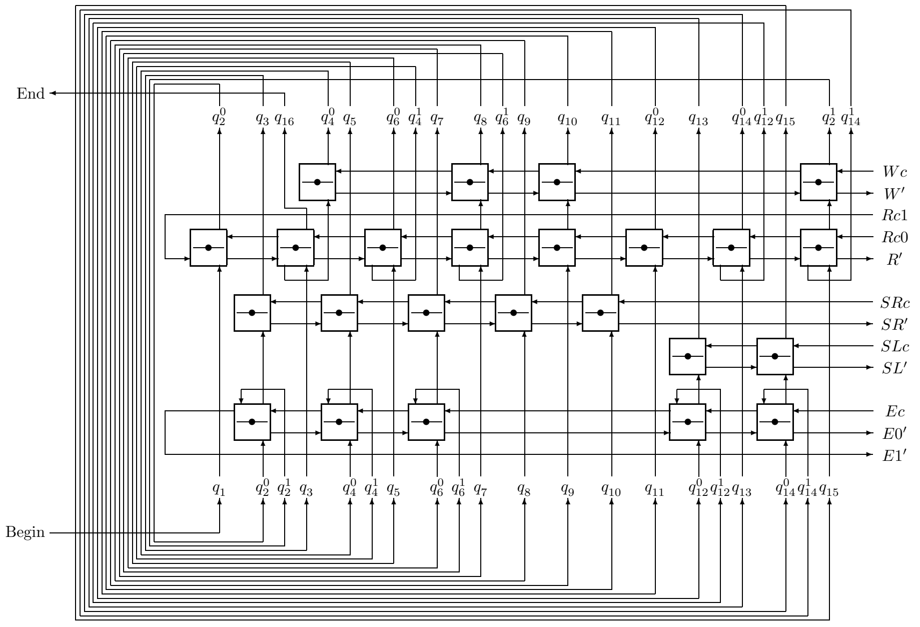


Fig. 6. Computing the function $f(n) = 2n$ by a reversible Turing machine T_{2n}.

An FC-module for T_{2n} is shown in Fig. 7. The quadruples of T_{2n} are executed and controlled by a matrix of REs. Namely, each column corresponds to each state of T_{2n}, and five rows correspond to five operations of write, read, shift-right, shift-left, and reversible erasure. At first, all the REs of the FC-module are in H-states.



End
q_2^0 q_3 q_{16} q_4^0 q_5 q_6^0 q_4^1 q_7 q_8 q_6^1 q_9 q_{10} q_{11} q_{12}^0 q_{13} q_{14}^0 q_{12}^1 q_{15} q_2^1 q_{14}^1
Wc
W'
$Rc1$
$Rc0$
R'
SRc
SR'
SLc
SL'
Ec
$E0'$
$E1'$
q_1 q_2^0 q_2^1 q_3 q_4^0 q_4^1 q_5 q_6^0 q_6^1 q_7 q_8 q_9 q_{10} q_{11} q_{12}^0 q_{12}^1 q_{13} q_{14}^0 q_{14}^1 q_{15}
Begin

Fig. 7. The finite-state control module (FC-module) for T_{2n}.

The FC-module executes an operation as follows. First, consider the case of a shift-right operation. If the present state of T_{2n} is, e.g., q_8, then a signal 1 is put on the line q_8 in the lower part of Fig. 7. The signal goes upward to the RE on the row of a shift-right operation. After changing the RE to the V-state, the signal turns right and goes out from SR'. If a signal returns back from SRc, it restores the state of the RE to the H-state, and goes upward on the column.

Next, consider the case of a read operation. For example, if the present state is q_{13}, a signal 1 goes upward along the line up to the RE on the row of a read operation. Changing the RE to the V-state as in the previous case, it turns right and goes out from R'. A signal will return back from $Rc0$ or $Rc1$. It then restores the state of the RE, and goes upward taking a different path depending on the completion signal $Rc0$ or $Rc1$.

Generally, just after a read operation, a reversible erasure operation should be performed. It is in fact an inverse of a read operation. The bottom row of REs shown in Fig. 7 realizes this operation.

Appropriately connecting the vertical lines in the upper part of the figure to the ones in the lower part, state transition of T_{2n} is also realized. Further, the input line "Begin" is connected to the line of the initial state q_1, and the line of the final state q_{16} is to the output line "End". (From the method given in [6] we can assume that an initial state of a constructed reversible TM does not appear as the fourth element of a quadruple (hence it appears only at time 0)).

3.4 An RE-circuit Realizing an RTM

By connecting an FC-module with the tape module appropriately, we can obtain the whole circuit of a given reversible Turing machine. Fig. 8 shows the RE-circuit for T_{2n}. By giving a signal 1 to the "Begin" input it starts to compute.

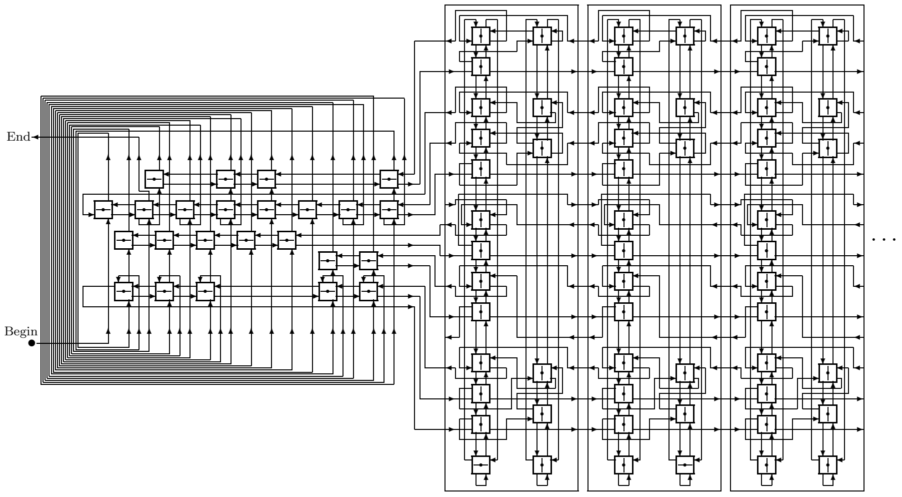


Fig. 8. The whole RE-circuit realizing T_{2n}.

4 A Simple Universal Reversible Cellular Automaton

In [10], a 3^4-state reversible partitioned cellular automaton (RPCA) P_3 was proposed, and it was shown that any reversible counter machine can be embedded in the P_3 space. This improves the previous result in [8] on the number of states. Each cell of P_3 has four parts, and each part has a state set $\{0,1,2\}^4$ (0, 1 and 2 are indicated by a blank, ○ and ●). Its local transition function is shown in Fig. 9. Reversibility of P_3 is verified by checking there is no pair of distinct rules having the same right-hand side. Fig. 10 shows an RE embedded in P_3 space, where a single ● acts as a signal. Hence, the function of an RE can be decomposed into much simpler reversible local rules of P_3.

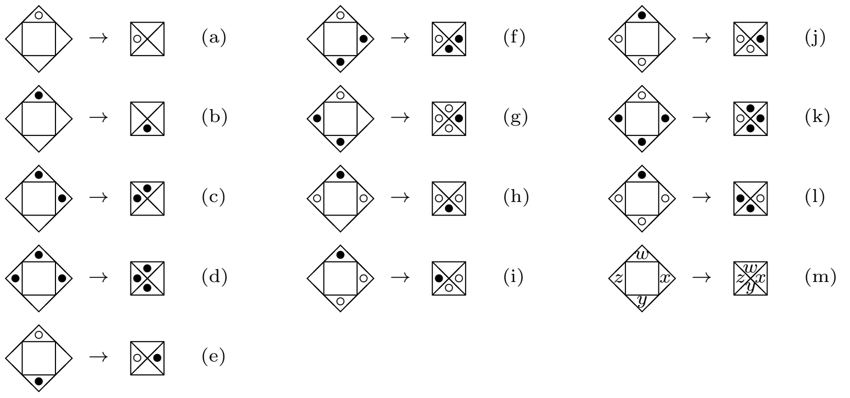


Fig. 9. The set of 13 rule schemes (representing 81 rules) of the rotation-symmetric 3^4-state RPCA P_3. Each rule scheme of (a)–(l) stands for four rules obtained by rotating the left- and right-hand sides of it by 0, 90, 180 and 270 degrees. The rule scheme (m) represents 33 rules not specified by (a)–(l) ($w, x, y, z \in$ {blank, ○, ●}).

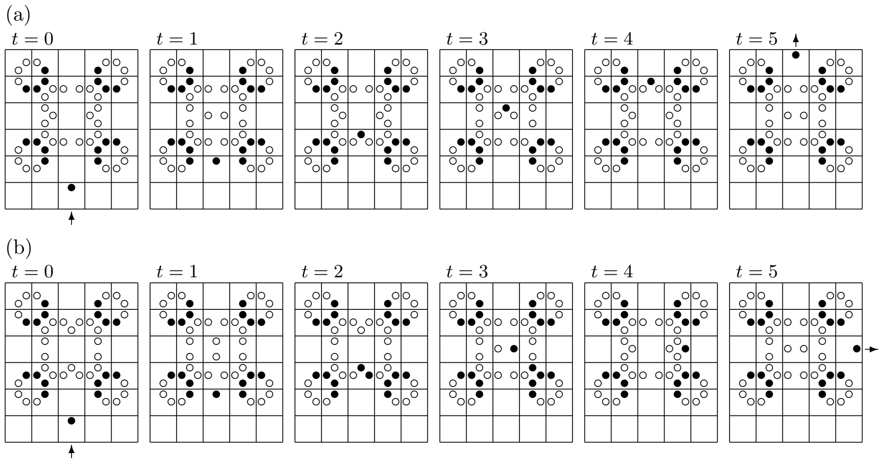


Fig. 10. An RE in the P_3 space: (a) the parallel case, and (b) the orthogonal case.

5 Concluding Remarks

In this paper, we showed that any reversible Turing machine can be realized as a circuit composed only of rotary elements. If we use logic gates such as AND, OR, Fredkin gate, or Toffoli gate, some synchronization mechanism should be provided to make two or more incoming signals interact properly. In a conventional circuit, it is solved by giving a clock signal. However, in the case of a rotary element, there is only one incoming signal to each element, which interacts with rotating bar of the element. Therefore it acts in a somewhat asynchronous manner, and thus there is no need to provide a clock signal. Hence, construction of a whole circuit becomes relatively simple.

There have been known several simple universal models of reversible cellular automata (e.g., [5,7,8,10]). For example, a Fredkin gate can be embedded in the cellular spaces in the references [5] and [7]. Though these models have very small number of states, they need an infinite configuaraion to realize a universal computer. On the other hand, in the models proposed in [8,10], a reversible counter machine can be realized as a finite configuration by using rotary elements and some other elements implemented in these cellular spaces. It is left for the future study whether there are still other methods to implement a universal computer in a simple reversible cellular space, and whether further simplification of a rotary element is possible.

References

1. Bennett, C.H., Logical reversibility of computation, *IBM J. Res. Dev.*, **17**, 525–532 (1973).
2. Feynman, R.P., *Feynman Lectures on Computation* (eds., A.J.G. Hey and R.W. Allen), Perseus Books, Reading, Massachusetts (1996).
3. Fredkin, E. and Toffoli, T., Conservative logic, *Int. J. Theoret. Phys.*, **21**, 219–253 (1982).
4. Gruska, J., *Quantum Computing*, McGraw-Hill, London (1999).
5. Margolus, N., Physics-like model of computation, *Physica*, **10D**, 81–95 (1984).
6. Morita, K., Shirasaki, A. and Gono, Y., A 1-tape 2-symbol reversible Turing machine, *Trans. IEICE Japan*, **E-72**, 223–228 (1989).
7. Morita, K., and Ueno, S., Computation-universal models of two-dimensional 16-state reversible cellular automata, *IEICE Trans. Inf. & Syst.*, **E75-D**, 141–147 (1992).
8. Morita, K., Tojima, Y. and Imai, K., A simple computer embedded in a reversible and number-conserving two-dimensional cellular space, *Multiple-Valued Logic*, (in press). Movie: `http://www.ke.sys.hiroshima-u.ac.jp/~morita/p4/`
9. Morita, K., A new universal logic element for reversible computing, *Technical Report of IEICE Japan*, COMP99-94 (2000).
10. Morita, K. and Ogiro T., Embedding a counter machine in a simple reversible 2-D cellular space, *Proc. Int. Workshop on Cellular Automata*, Osaka, 30–31 (2000). Movie: `http://www.ke.sys.hiroshima-u.ac.jp/~morita/p3/`
11. Păun, Gh., Rozenberg, G. and Salomaa, A., *DNA Computing*, Springer-Verlag, Berlin (1998).
12. Toffoli, T., Reversible computing, in *Automata, Languages and Programming*, Springer-Verlag, LNCS-85, 632–644 (1980).