

# Counting infinitely by oritatami co-transcriptional folding

Kohei Maruyama · Shinnosuke Seki

Received: date / Accepted: date

**Abstract** A fixed bit-width counter was proposed as a proof-of-concept demonstration of the oritatami model of cotranscriptional folding [Geary et al., Proc. MFCS 2016, LIPIcs 58, 43:1-43:14], and it was embedded into another oritatami system that self-assembles a finite portion of Heighway dragon fractal. In order to expand its applications, we endow this counter with capability to widen bit-width at every encounter with overflow.

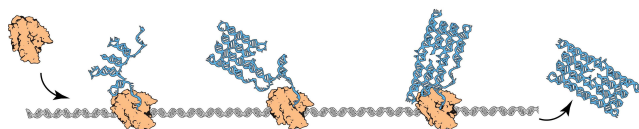
## 1 Introduction

Counting is one of the most essential tasks for computing. Nature has been counting billions of days using molecular “circadian clockwork” which is “as complicated and as beautiful as the wonderful chronometers developed in the 18th century” [?]. Nowadays, developments in molecular self-assembly technology enable us to design molecules to count. Evans has demonstrated a DNA tile self-assembly system that counts accurately *in-vitro* in binary from a programmed initial value until it overflows [?]. In its foundational theory of molecular

An extended abstract on this work was published as a short paper in the proceedings of the 46th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2020, Limassol, Cyprus, January 20-24, 2020), Lecture Notes in Computer Science (LNCS) 12011, pp. 566–575.

K. Maruyama  
The University of Electro-Communications, 1-5-1 Chofugaoka, Chofu, Tokyo, 1828585, Japan

S. Seki  
The University of Electro-Communications, 1-5-1 Chofugaoka, Chofu, Tokyo, 1828585, Japan  
École Normale Supérieure de Lyon, 46 allée d’Italie, 69007, Lyon, France  
E-mail: s.seki@uec.ac.jp



**Fig. 1** RNA origami. RNA polymerase enzyme (orange) synthesizes the temporal copy (blue) of a gene (gray spiral) out of ribonucleotides of four types A, C, G, and U.

self-assembly, such binary counters have been proved versatile, being used to assemble shapes of particular size [?,?], towards self-assembly of fractals [?], and as an infinite scaffold on which Turing machines can be simulated in parallel in the abstract tile-assembly model [?,?], to name a few.

A fixed bit-width (finite) binary counter has been implemented as a proof-of-concept demonstration of the oritatami model of cotranscriptional folding [?]. As shown in Fig. ??, an RNA transcript folds upon itself while being transcribed (synthesized) from its corresponding DNA template strand. Geary, Rothmund, and Andersen programmed a specific RNA rectangular tile structure into a DNA template in such a way that the corresponding RNA transcript *folds cotranscriptionally* into the programmed tile structure with high probability *in-vitro* at room temperatures (*RNA origami*) [?]. An oritatami system folds a transcript of abstract molecules called *beads* of finitely many types over the 2-dimensional triangular lattice cotranscriptionally according to a rule set that specifies which types of molecules are allowed to bind at unit distance. The transcript of the binary counter in [?] is of period 60 as  $\textcircled{0}-\textcircled{1}-\dots-\textcircled{59}-\textcircled{0}-\dots$  and its period is divided semantically into two half-adder (HA) modules  $A = \textcircled{0}-\textcircled{1}-\dots-\textcircled{11}$  and  $C = \textcircled{30}-\textcircled{31}-\dots-\textcircled{41}$  and two structural modules  $B$  and  $D$ , which are sandwiched by half-adder modules along the

transcript as  $ABCD$ . While being folded cotranscriptionally in zigzags, HA modules increment the current counter value  $i$  by 1, which is initialized on a linear *seed* structure, alike the Evans' counter, whereas structural modules  $B$  and  $D$  align HA modules properly and also make a turn at an end of the counter value  $i$ ;  $B$  guides the transcript from a zig to a zag ( $\hookrightarrow$ ) while  $D$  does from a zag to a zig ( $\hookleftarrow$ ). This counter was embedded as a component of an oritatami system to self-assemble an arbitrary finite portion of Highway dragon fractal [?]; it remembers how many turns the dragon has made so far. Its applications are limited, however, by lack of mechanism to detect an overflow, at which its behavior is undefined and it gets nondeterministic. In this paper, we shall endow this counter, or more precisely, its structural module  $B$ , with the capability of overflow detection and bit-width expansion thereof.

The oritatami model has been proven Turing universal in [?], where a universal Turing machine is encoded as a seed and a period of transcript of an oritatami system using 542 bead types via cyclic tag system, and the number of bead types needed for the universality was reduced to almost one-third (183 bead types) recently [?] by an intrinsic simulation of 1-dimensional cellular automata (CA). It is hence no surprise that oritatami systems can count even infinitely. In the read-once-write-once models of computation such as oritatami, abstract tile-assembly model (aTAM) [?,?], and many others in molecular self-assembly, however, what a computation matters is not so much an output obtained conclusively but rather a shape self-assembled in the course; the shape is often the goal of computation in molecular self-assembly. Unless being interlocked geometrically, the output cannot wire to an input of another computation. The versatile systems in oritatami [?,?] hence never discourage us to design a set of singly-functional modules of simple enough shape to be wired, preferably via a common interface. (Note that unless being provided anyhow with random access memory, in molecular self-assembly, the ability to count highly unlikely suffices for Turing universality; c.f. [?].)

The proposed counter is simple in number of bead types used as well as length of its transcript period. It transcribes ①–②– $\cdots$ –③ $\cdot$  repetitively (needless to say, 132 bead types), whereas the period of a transcript to simulate a radius- $r$  CA with  $Q$  states by the simplified Turing universal oritatami system [?] is of length about  $\frac{142}{3}Q_r^2 \log_2 Q_r$ , where  $Q_r = 2^{\lceil \log_2(2Q^{2r+1}) \rceil}$ , whose value cannot be smaller than  $\frac{142}{3} \cdot 4 \geq 189$ , obtained at  $Q = 1$  and  $r = 0$ , and rises significantly in order for CAs to be able to compute as  $\frac{142}{3} \cdot 4 \cdot 16^2 \geq 48469$  at  $Q = 2$  and  $r = 1$ . A periodic transcript is expected to be transcribed from a circular DNA [?] but such a template

DNA sequence gets more costly to be synthesized as it gets longer. The proposed counter folds in zigzags into a logarithmically-widening trapezoid so that it cannot be afforded at any scaling of Highway dragon fractal. It therefore does not truly enable to self-assemble this fractal. The significance of this counter is rather to demonstrate that an oritatami system can detect an overflow and change its phase therein from counting to bit-expansion and back (note that oritatami is not provided with any internal states; see Sect. ??). This phase-transition capability should be applicable, for example, to self-assemble the  $n \times n$  square in oritatami at a scale, starting from an  $O(n)$ -size seed encoding  $n$ , based on the system in aTAM for square self-assembly [?]; such a system would make use of this capability rather to transition from the binary counting phase to the phase to fill the remaining  $n \times (n - \log n)$  rectangle while drawing the diagonal.

This paper is organized as follows. In Section ??, we provide basic notions and notation of oritatami system. The infinite counter shall be explained into detail in Section ?. We conclude this paper in Section ?? with a short discussion.

## 2 Preliminaries

Let  $\Sigma$  be a finite alphabet, whose elements should be regarded as types of abstract molecule, or *beads*. A bead of type  $a \in \Sigma$  is called an  $a$ -bead. By  $\Sigma^*$  and  $\Sigma^\omega$ , we denote the set of finite sequences of beads and that of one-way infinite sequences of beads, respectively. The empty sequence is denoted by  $\lambda$ . Let  $w = b_1 b_2 \cdots b_n \in \Sigma^*$  be a sequence of length  $n$  for some integer  $n$  and bead types  $b_1, \dots, b_n \in \Sigma$ . The *length* of  $w$  is denoted by  $|w|$ , that is,  $|w| = n$ . For two indices  $i, j$  with  $1 \leq i \leq j \leq n$ , we let  $w[i..j]$  refer to the subsequence  $b_i b_{i+1} \cdots b_{j-1} b_j$ ; if  $i = j$ , then  $w[i..i]$  is simplified as  $w[i]$ . For  $k \geq 1$ ,  $w[1..k]$  is called a *prefix* of  $w$ .

Oritatami systems fold their transcript, which is a sequence of beads, over the triangular grid graph  $\mathbb{T} = (V, E)$  cotranscriptionally. A directed path  $P = p_1 p_2 \cdots p_n$  in  $\mathbb{T}$  is a sequence of *pairwise-distinct* points  $p_1, p_2, \dots, p_n \in V$  such that  $\{p_i, p_{i+1}\} \in E$  for all  $1 \leq i < n$ . Its  $i$ -th point is referred to as  $P[i]$ . Now we are ready to abstract RNA single-stranded structures in the name of conformation. A *conformation*  $C$  (over  $\Sigma$ ) is a triple  $(P, w, H)$  of a directed path  $P$  in  $\mathbb{T}$ ,  $w \in \Sigma^*$  of the same length as  $P$ , and a set of (hydrogen) bonds  $H \subseteq \{\{i, j\} \mid 1 \leq i, i+2 \leq j, \{P[i], P[j]\} \in E\}$ . This is to be interpreted as the sequence  $w$  being folded along the path  $P$  in such a manner that its  $i$ -th bead  $w[i]$  is placed at the  $i$ -th point  $P[i]$  and the  $i$ -th and  $j$ -th

beads are bonded (by a hydrogen-bond-based interaction) if and only if  $\{i, j\} \in H$ . The condition  $i + 2 \leq j$  represents the topological restriction that two consecutive beads along the path cannot be bonded. A *rule set*  $R \subseteq \Sigma \times \Sigma$  is a symmetric relation over  $\Sigma$ , that is, for all bead types  $a, b \in \Sigma$ ,  $(a, b) \in R$  implies  $(b, a) \in R$ . A bond  $\{i, j\} \in H$  is *valid with respect to  $R$* , or simply  *$R$ -valid*, if  $(w[i], w[j]) \in R$ . This conformation  $C$  is  *$R$ -valid* if all of its bonds are  $R$ -valid. For an integer  $\alpha \geq 1$ ,  $C$  is of *arity  $\alpha$*  if it contains a bead that forms  $\alpha$  bonds but none of its beads forms more. By  $\mathcal{C}_{\leq \alpha}(\Sigma)$ , we denote the set of all conformations over  $\Sigma$  whose arity is at most  $\alpha$ ; its argument  $\Sigma$  is omitted whenever  $\Sigma$  is clear from the context.

The oritatami system grows conformations by an operation called elongation. Given a rule set  $R$  and an  $R$ -valid conformation  $C_1 = (P, w, H)$ , we say that another conformation  $C_2$  is an elongation of  $C_1$  by a bead  $b \in \Sigma$ , written as  $C_1 \xrightarrow{R}_b C_2$ , if  $C_2 = (Pp, wb, H \cup H')$  for some point  $p \in V$  not along the path  $P$  and set  $H' \subseteq \{\{i, |w| + 1\} \mid 1 \leq i < |w|, \{P[i], p\} \in E, (w[i], b) \in R\}$  of bonds formed by the  $b$ -bead; this set  $H'$  can be empty. Note that  $C_2$  is also  $R$ -valid. This operation is recursively extended to the elongation by a finite sequence of beads as: for any conformation  $C$ ,  $C \xrightarrow{R}_\lambda C$ ; and for a finite sequence of beads  $w \in \Sigma^*$  and a bead  $b \in \Sigma$ , a conformation  $C_1$  is elongated to a conformation  $C_2$  by  $wb$ , written as  $C_1 \xrightarrow{R}_{wb} C_2$ , if there is a conformation  $C'$  that satisfies  $C_1 \xrightarrow{R}_w C'$  and  $C' \xrightarrow{R}_b C_2$ .

An *oritatami system*  $\Xi$  is a tuple  $(\Sigma, R, \delta, \alpha, \sigma, w)$ , where  $\Sigma$  and  $R$  are defined as above, while the other elements are a positive integer  $\delta$  called *delay*, a positive integer  $\alpha$  called *arity*, an initial  $R$ -valid conformation  $\sigma \in \mathcal{C}_{\leq \alpha}(\Sigma)$  called the *seed*, and a (possibly infinite) *transcript*  $w \in \Sigma^* \cup \Sigma^\omega$ , which is to be folded upon the seed by stabilizing beads of  $w$  one at a time so as to minimize energy collaboratively with the succeeding  $\delta - 1$  nascent beads. The energy of a conformation  $C = (P, w, H)$ , denoted by  $\Delta G(C)$ , is defined to be  $-|H|$ ; the more bonds a conformation has, the more stable it gets. The set  $\mathcal{F}(\Xi)$  of conformations *foldable* by the system  $\Xi$  is recursively defined as: the seed  $\sigma$  is in  $\mathcal{F}(\Xi)$ ; and provided that an elongation  $C_i$  of  $\sigma$  by the prefix  $w[1..i]$  be foldable (i.e.,  $C_0 = \sigma$ ), its further elongation  $C_{i+1}$  by the next bead  $w[i + 1]$  is foldable if

$$C_{i+1} \in \arg \min_{C \in \mathcal{C}_{\leq \alpha} \text{ s.t. } C_i \xrightarrow{R}_{w[i+1]} C} \min \left\{ \Delta G(C') \mid C \xrightarrow{R}_{w[i+2..i+k]}^* C', k \leq \delta, C' \in \mathcal{C}_{\leq \alpha} \right\}. \quad (1)$$

Then we say that the bead  $w[i + 1]$  and the bonds it forms are *stabilized* according to  $C_{i+1}$ . Note that an

arity- $\alpha$  oritatami system cannot fold any conformation of arity larger than  $\alpha$ . A conformation foldable by  $\Xi$  is *terminal* if none of its elongations is foldable by  $\Xi$ . The oritatami system  $\Xi$  is *deterministic* if for all  $i \geq 0$ , there exists at most one  $C_{i+1}$  that satisfies (??). A deterministic oritatami system folds into a unique terminal conformation.

*Example 1 (Glider<sup>1</sup>)* See Fig. ?? for a delay-3 deterministic oritatami system  $\Xi$  to fold a motif called a glider. Its transcript is a repetition of  $a \bullet bb' \bullet a'$  and its rule set  $R$  is  $\{(a, a'), (b, b')\}$ . Its seed is colored in red. The first 3 beads,  $a \bullet b$ , are transcribed and elongate the seed in all possible ways. The  $a$ -bead cannot bind or the second bead is inert according to  $R$ . The third bead,  $b$ , can bind to the  $b'$ -bead in the seed but for that, the  $a$ -bead must be located to the east of the previous  $a'$ -bead; it is thus stabilized there. Then the next bead,  $b'$ , is transcribed. After the three steps, the third bead,  $b$ , is stabilized, along with its bond with the  $b'$ -bead, yielding the rightmost glider of width 3 and height 3 in Fig. ?. In the next 3 steps, the succeeding  $b' \bullet a'$  folds alike and results in a glider of width 4 and height 3. The glider thus proceeds by unit distance per three beads and one bond.

The glider is self-sustaining and has provided a solid scaffold to oritatami systems [?, ?, ?, ?, ?]. Counting infinitely in a zigzag manner requires a self-sustaining structure in order to travel “in a vacuum,” that is, in the absence of the previous zag above, while widening bit-width at an overflow (see Figs. ?? and ??). Thanks to the sparsity of bonds, gliders can be deformed so easily for us to design multifunctional modules for the proposed counter. However, the bond density does not always determine the degree of functional extensibility; indeed, parallelograms, based on which the fixed bit-width counter [?] was designed, have turned out to be highly functionally extensible [?, ?], in spite of relatively larger number of bonds necessary.

### 3 Folding an infinite binary counter

In this section, we describe a delay-3, arity-5 deterministic oritatami system to count infinitely. It employs 132 bead types, and its transcript  $w$  is a repetition of 1-2-3-...-132. Its rule set is given in Sect. ?. The fixed bit-width counter by Geary et al. [?] operates at delay 4 under a different dynamics, but it was modified in [?]

<sup>1</sup> A video to show how a glider folds can be found at <https://www.dailymotion.com/video/x3cdj35>, in which the Turing universal oritatami system by Geary et al. [?] is running at delay 3.

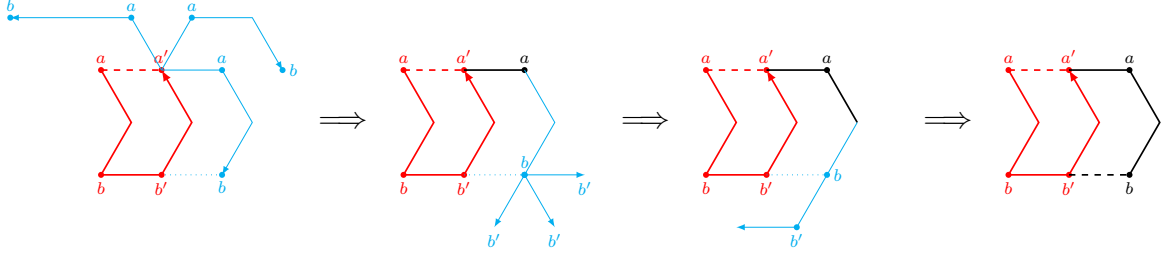


Fig. 2 Folding of a glider motif by a delay-3 deterministic oritatami system.

so as to run at delay 3 and under the more prevailing dynamics (??) in the research of oritatami model.

### 3.1 General idea

Between two consecutive overflows, the proposed system behaves in the same way as the fixed bit-width counter. Its transcript folds in a zigzag manner macroscopically (downward in figures throughout this paper). A zig, folding from right to left, increments the current counter value by 1. The succeeding zag, folding from left to right, formats the incremented counter value for the sake of next zig and copies it downward. Unlike the existing counter, when a zig encounters an overflow, it does not abort but rather extends the bit-width of the current counter value by 1 bit.

The transcript of the proposed counter is periodic. Its period 1-2-3-...-132 is divided semantically into the following four subsequences, called *modules*:

- 1– 30    Format module or F; colored in green
- 31– 66    Left-Turn module or L; blue
- 67– 96    Half-Adder module or H; red
- 97–132    Right-Turn module or R; yellow

The transcript can be hence represented as (FLHR)\* at the modular level. Modules are to play their roles in expected environments by folding into respective conformations which should be pairwise-distinct enough to be distinguishable by other modules transcribed later. Such expected conformations are called a *brick*. For example, module F encounters the four environments shown in Fig. ?? where it takes the four bricks **Fnt**, **Fnb**, **F0**, and **F1**, respectively. Here, by saying (an instance of) a module folds into (or takes) a brick in an environment, what we actually mean is that the rule set is designed so as for the transcript of the module to interact with itself as well as with the environment and fold deterministically into that brick according to the dynamics (??). The whole system is designed to guarantee that each module is transcribed only in one of the environments it expects. This fact is illustrated in

the *brick automaton* in Fig. ??, which describes pairs of an environment and a brick as a vertex and transitions between them, introduced in [?]. Since this automaton is closed, it suffices to test whether for all pairs of an environment and a brick, the brick is folded deterministically in the environment. This test has been done *in-silico* using a simulator developed for this project. This brick automaton and all the certificates can be found at <https://komaruyama.github.io/oritami-infinity-counter/>.

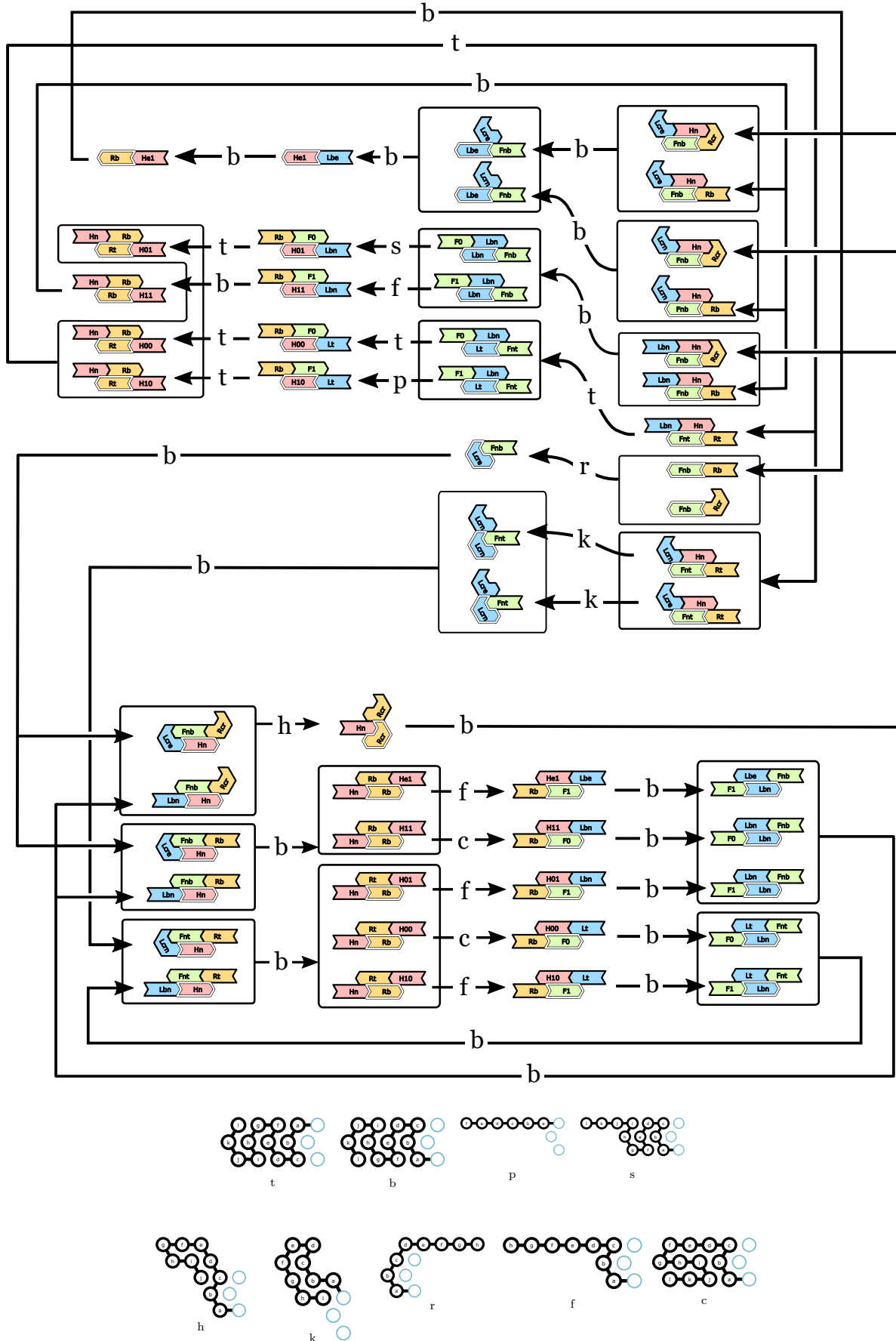
*Seed and Encoding.* An initial counter value is encoded in binary as  $b_{k-1}b_{k-2}\cdots b_1b_0$  on the seed in the following format:

$$64-65-66-\left(\prod_{i=k-1}^0 (w_{Hn}w_{Rb}w_{Fb_i}w_{Lbn})\right)w_{Hn}, \quad (2)$$

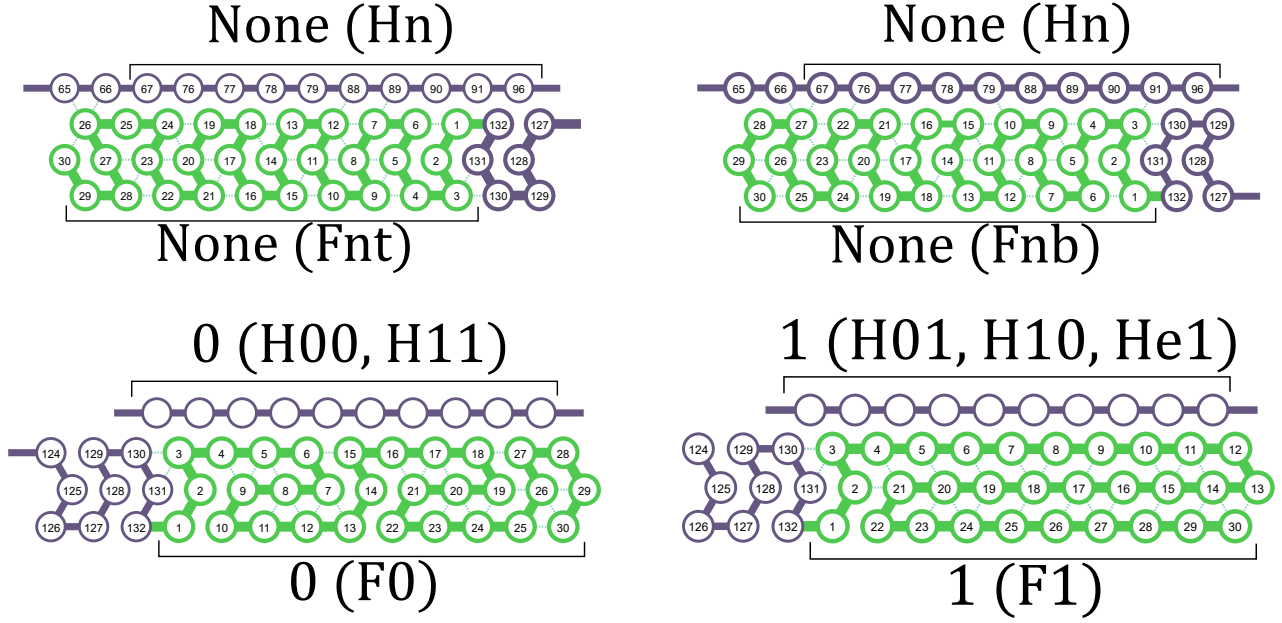
where

$$\begin{aligned} w_{Hn} &= 67-76-77-78-79-88-89-90-91-96, \\ w_{Rb} &= 97-102-103-108-109-114-115-120- \\ &\quad -121-126-127-132, \\ w_{F0} &= 1-10-11-12-13-22-23-24-25-30, \\ w_{F1} &= 1-22-23-24-25-26-27-28-29-30, \\ w_{Lbn} &= 31-36-37-42-43-48-49-54-55-64- \\ &\quad -65-66. \end{aligned}$$

$w_{Hn}$ ,  $w_{Rb}$ ,  $w_{F0}$ ,  $w_{F1}$ , and  $w_{Lbn}$  are sequences of bead types exposed downward by modules H, R, F, L when they fold into bricks **Hn**, **Rb**, **Fb<sub>i</sub>**, **Lbn**, respectively, which can be found in Figs. ??, ??, ??, and ??. The seed consists of the initial value encoded as (??), preceded by the sequence 58–59–60–61–62–63 folded into a glider-shaped signal for left turn, and succeeded by 97–108–109–120–122–123–...–131–132 that folds as the module R folds into a brick **Rcr** (see Fig. ??). The seed is exemplified for  $k = 1$  and  $b_0 = 0$  in Fig. ?? or for  $k = 0$  (non-coding, i.e., initial counter value 0) in Fig. ??, where it is colored in purple.



**Fig. 3** A brick automaton of the proposed infinite binary counter. 9 possible ways of folding the first several beads of a module being transcribed are enumerated at the bottom with labels t, b, ..., and transitions in the brick automaton are labeled with a corresponding way of folding.



**Fig. 4** All the four bricks of module F: The two bricks at the top, **Fnt** and **Fnb**, are for zigs while the others, **F0** and **F1**, are for zags. **Fnb** binds to the zag above so weakly that it can fold even in the absence of such a zag, that is, at an overflow.

### 3.2 Brick level overview

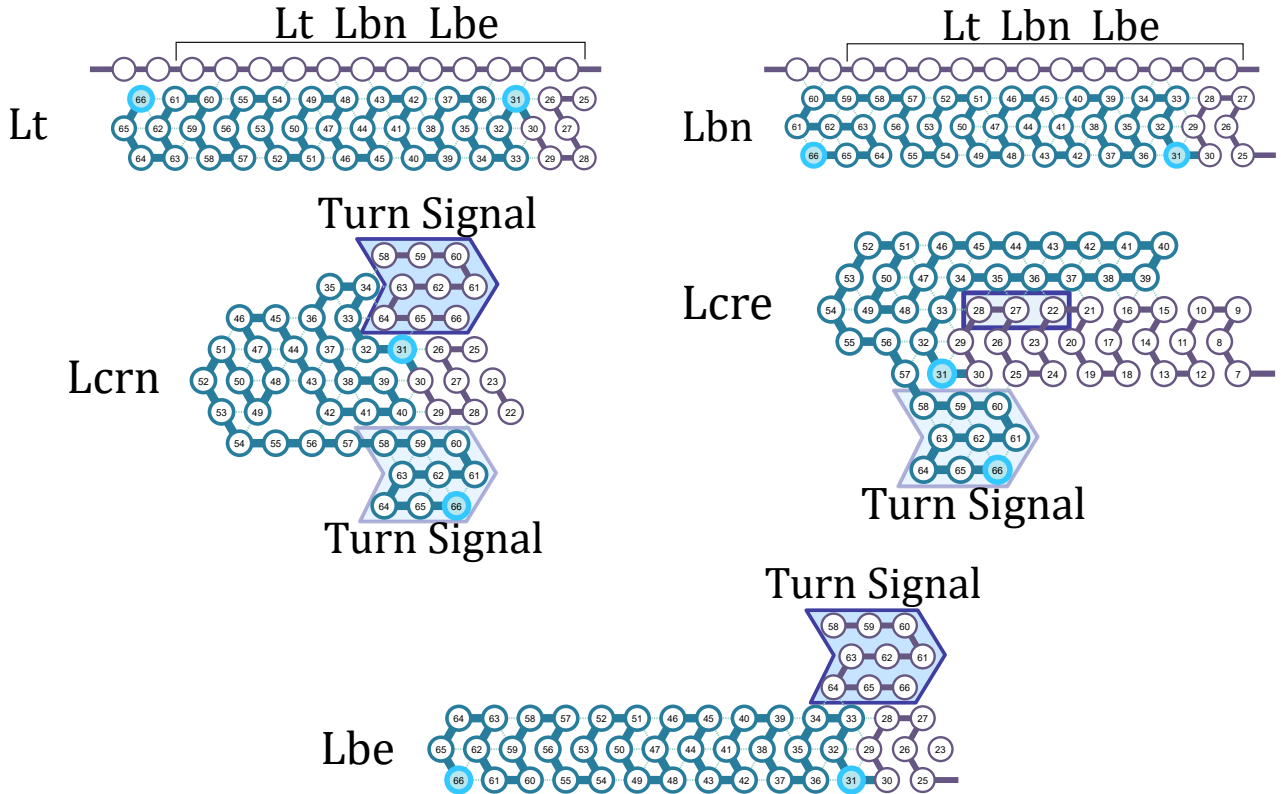
Starting from the seed, this system cyclically transits four phases: zig ( $\leftarrow$ ), left carriage-return ( $\leftarrow$ ), zag ( $\rightarrow$ ), and right carriage-return ( $\rightarrow$ ). The prefix  $(\text{FLHR})^k \text{F}$  of the transcript folds into the first zig (recall that  $k$  is the bit-width of the initial count). In zigs in general, all the instances of modules F and H fold into bricks of width 10 and height 3, while those of L and R fold into bricks of width 12 and height 3. Zigs thus turn out to be a linear structure of height 3. We can inductively observe that the  $i$ -th instance of H in the prefix is transcribed right below  $b_{i-1}$  encoded on the seed in the format (2) so that the H can “read”  $b_{i-1}$ . After the whole prefix has thus folded into the first zig, the next L is transcribed right below Turn Signal, which lets the L fold into a special brick for left carriage-return if the zig ended at the top (this occurs unless  $b_{k-1} = b_{k-2} = \dots = b_0 = 1$ ) (see Fig. ??). We should note that this special brick **Lcre** is provided with another Turn Signal for the sake of next left carriage-return (see Fig. ??). Having been thus carriage-returned, the succeeding subsequence  $(\text{HRFL})^k \text{H}$  of the transcript folds into the first zag. Even in zags, F and H fold into bricks of width 10 and height 3, while L and R fold into bricks of width 12 and height 3. As a result, zags turn out also to be a linear structure of height 3. More importantly, instances of H and F are aligned thus vertically and alternately into columns (see Figs. ??, ??, and ??),  $i$ -th of which from the right propagates the  $(i-1)$ -th bit of the counter value downward. After the whole subse-

quence has folded into the first zag, an instance of  $R$  is transcribed and folded into a special brick **Rcr** for right carriage-return due to the Turn Signal 125-124-123-122, which occurs also at the bottom of **Rcr** (see Fig. ??) for the sake of next right carriage-return. This amounts to one cycle of the phase transition.

### 3.3 Increment of the counter

In a zig, module H plays its primary role as a half-adder and carry transfers through instances of others (F, L, and R) from an instance of H to another for more significant bit. Carry transfers as a height for modules to start. In zigs, modules F, L, and R take the respective two bricks (**Fnt** and **Fnb** for F, **Lt** and **Lbn** for L, and **Rt** and **Rb** for R; see Figs. ??, ??, and ??), both of which start and end at the same height: one at the top while the other at the bottom. A zig is fed with carry by being forced to start at the bottom by the last **Rcr** or the seed. Until an overflow, module H encounters only four environments, which encode input 0 as  $w_{F0}$  or 1 as  $w_{F1}$  and carry or no-carry as of whether the module starts at the bottom or top, where it takes H00, H01, H10, and H11, respectively, as shown in Fig. ?? (**Hxc** is folded when the input is  $x$  and the carry is given if  $c = 1$  or not otherwise).

Let us see how the subsequence  $(\text{FLHR})^k \text{F}$  folds into a zig in order to count up; for  $k = 1$  and the current counter value 0, see Fig. ?? . The zig starts at the bottom, that is, being carried, and the carry transfers



**Fig. 5** All the five bricks of module L: Lt, Lbn, Lcrn, Lcre, and Lbe from top left to bottom right. In zigs, L folds into either Lt or Lbn depending on where it starts, until the transcript reaches the left end, where L folds either into Lcrn if the current counter value has not been overflowed, or into Lbe at an overflow. In the case of overflow, the next L folds into Lcre. In zags, L always folds into Lbn.

through the first instances of F and L in the way just explained toward the first instance of H. This H is thus fed with carry and folds into H01 if the bit encoded above is 0, as illustrated in Fig. ??, or H11 if the bit is rather 1. H01 ends at the top, corresponding to canceling the carry out. This absence of carry transfers through the succeeding modules leftward. As a result, the zig, or more precisely, the last instance of F of the subsequence ends folding at the bottom if this increment causes an overflow (Fig. ??), or at the top otherwise (Fig. ??). An instance of L is to be transcribed next. It folds either into Lcrn for carriage-return unless the counter value is overflowed, or into Lbe at an overflow.

### 3.4 Bit-width expansion at an overflow

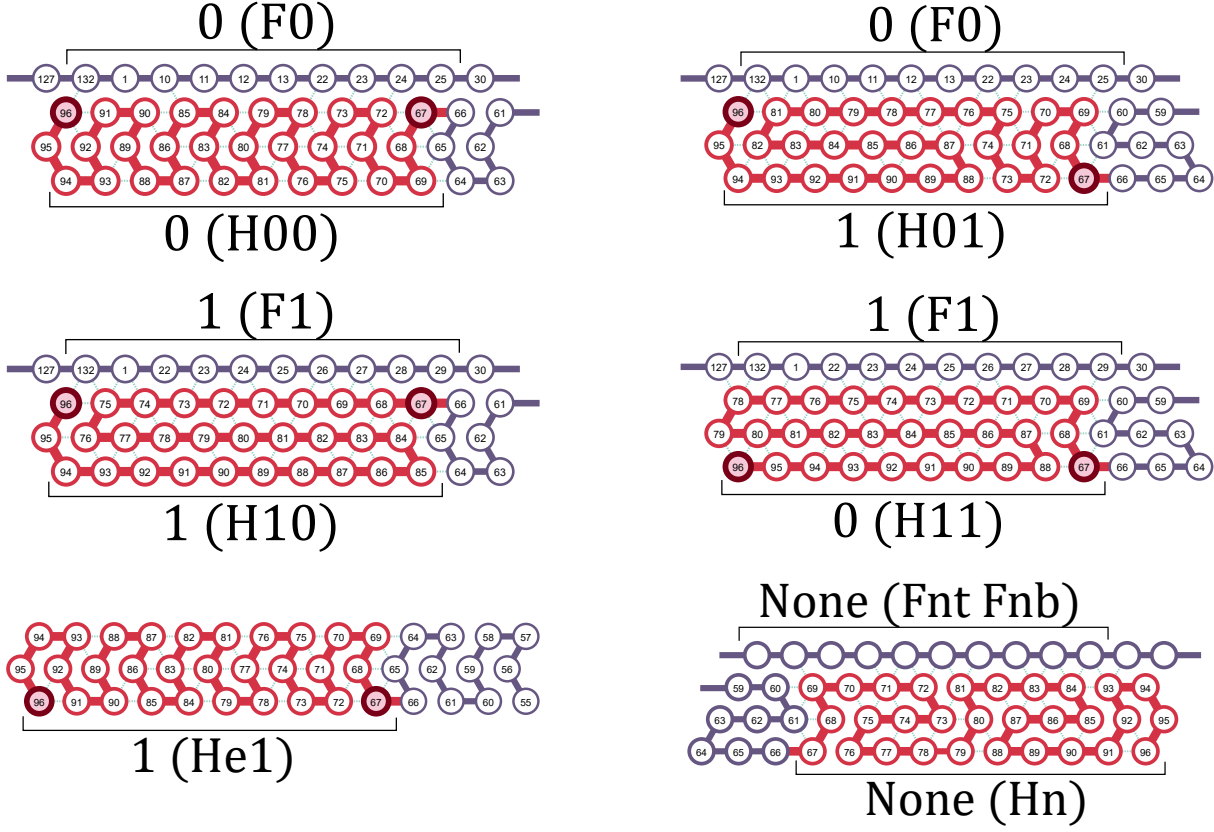
The fixed bit-width counter [?] cannot handle a zig that ends at the bottom, that is, its behavior is undefined, that is, it gets nondeterministic, at an overflow. In contrast, module L in this infinite counter is designed so as to fold into Lbe in this situation in order to keep counting up (Fig. ??). Observe that the dent on Turn Signal made of 58, 63, and 64 is too far for module L, or more precisely, for its beads 33 and 34 to inter-

act with strongly enough to fold into Lcrn. Lbe is a self-sustaining conformation (glider) so that it can fold even in a “void,” which occurs at that very moment. For the same reason, the following instances of H, R, and F fold into self-sustaining conformations He1, Rb, and Fnb, respectively (Figs. ?? and ??). Note that He1 is essentially the same as H00 but exposes the opposite side downward, which will be interpreted as the leading bit 1 after expansion in the next zag. When the next instance of L is transcribed, there is nothing around. Nevertheless, it does not fold into Lbe but folds into Lcre for carriage-return; how? It is guided by interaction between the beads 35, 36 along the transcript of L and the Turn Signal 28-27-22 above Fnb (Figs. ?? and ??). This signal is usually hidden geometrically by the previous zag, and hence, does no harm.

### 3.5 Formatting

The counter value has been successfully incremented but the resulting value is not in the format (??) yet. A subsequence  $(HRFL)^k H$  of the transcript folds into the next zag, where  $k$  is the bit width of the incremented counter value. In this zag, instances of F play





**Fig. 6** All the six bricks of module H: H00, H01, H10, H11, He1, and Hn from top left to bottom right. In zags, H always folds into Hn while in zigs, it folds into one of the other five bricks.

their primary role to format the 1-bit output by module H (recall that instances of H and F are aligned vertically and alternately) so as to expose the incremented counter value in the format (??) for the next zig. Both of the bricks of L for carriage-return, i.e., Lcrn and Lcre, end at the bottom so that zags start at the bottom. All modules start and end at the bottom in zags; note that nothing has to be transferred between modules. That is, instances of H, L, and R fold into Hn, Lbn, and Rb, respectively. Below the brick Hxc, an instance of F folds into Fy, where  $y = (x + c) \bmod 2$ .

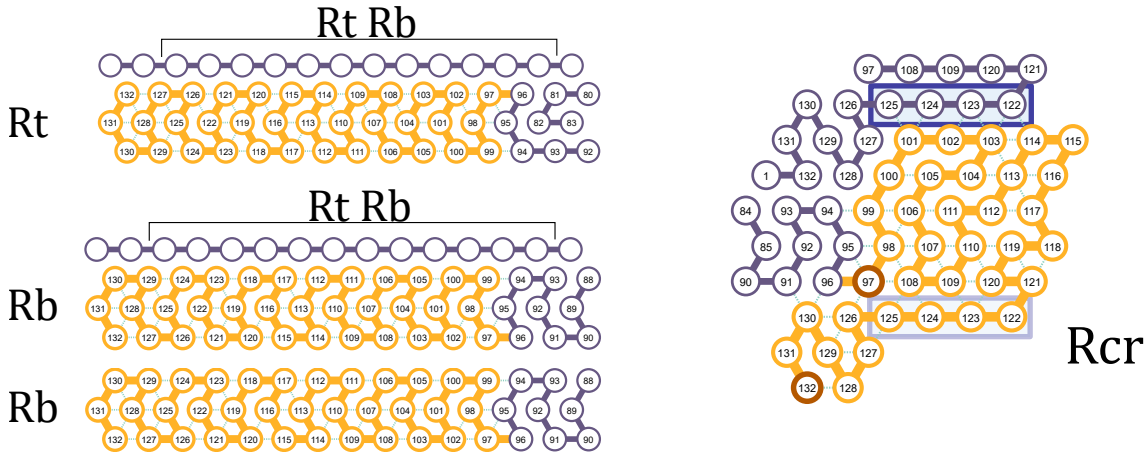
### 3.6 Rule set

The rule set of the proposed counter consists of the following 377 rules:

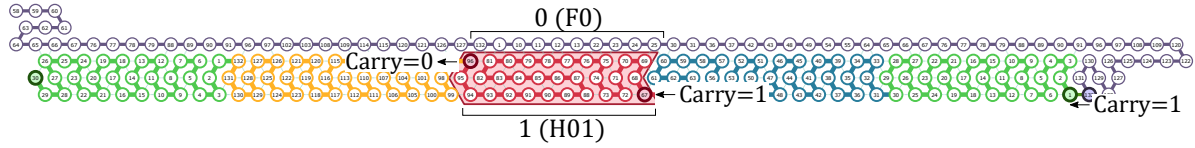
(1,6), (13,72), (29,32), (65,68), (88,93),  
 (1,74), (13,81), (29,33), (65,69), (89,93),  
 (1,75), (14,18), (29,40), (65,84), (91,130),  
 (1,77), (14,29), (29,60), (67,131), (91,96),  
 (1,80), (14,30), (29,69), (67,72), (92,96),  
 (1,81), (15,28), (30,32), (67,84), (93,130),

(1,84), (15,39), (30,33), (67,88), (93,132),  
 (1,93), (15,72), (30,39), (68,72), (94,99),  
 (2,21), (15,76), (30,40), (68,83), (95,97),  
 (3,130), (15,81), (30,60), (68,84), (95,98),  
 (3,131), (15,90), (31,36), (68,87), (96,130),  
 (3,64), (15,91), (31,65), (69,130), (96,132),  
 (3,65), (16,21), (32,35), (69,131), (97,102),  
 (3,84), (16,27), (32,36), (70,75), (97,108),  
 (3,91), (16,38), (32,37), (70,81), (97,126),  
 (3,93), (16,39), (32,38), (70,87), (98,102),  
 (3,95), (16,71), (32,56), (71,74), (98,106),  
 (4,21), (16,72), (32,57), (71,75), (98,107),  
 (4,83), (17,20), (33,35), (71,81), (98,108),  
 (4,84), (17,21), (33,47), (71,86), (99,106),  
 (4,9), (17,26), (33,48), (71,87), (99,127),  
 (5,20), (17,27), (33,61), (72,79), (99,129),  
 (5,8), (17,70), (33,63), (72,80), (100,105),  
 (5,85), (17,88), (33,64), (73,78), (101,104),  
 (5,9), (18,25), (34,39), (73,80), (101,124),  
 (5,90), (18,27), (34,45), (73,81), (101,125),  
 (5,91), (18,67), (34,46), (73,84), (102,123),

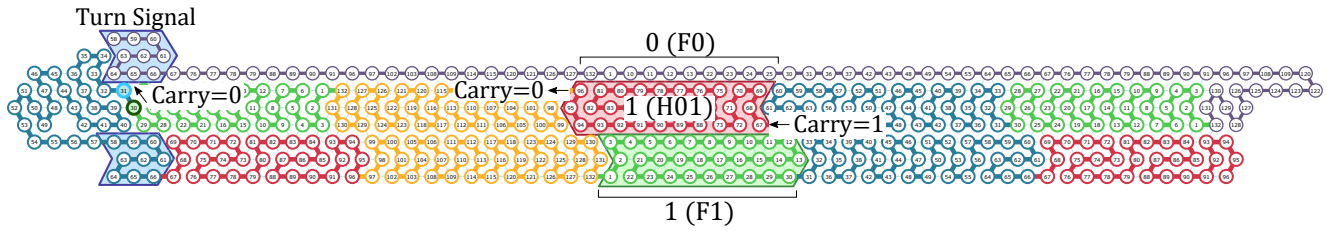




**Fig. 7** All the three bricks of module R: Rt, Rb, and Rcr. In zigs, R folds into Rt or Rb, depending on how high it starts. In zags, R always folds into Rb until the transcript reaches the right end, where R folds into Rcr due to the four beads (boxed). Rcr is provided with this Turn Signal for the next right carriage-return. Note that Rb is not bonded to the environment at all. That is, this module certainly folds into Rb at an overflow.

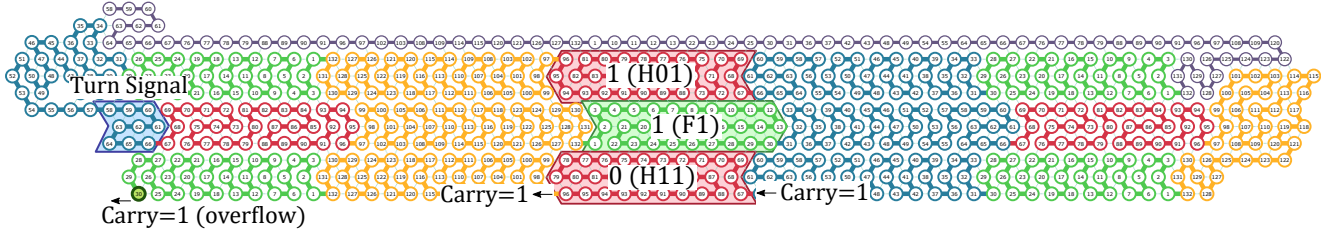


**Fig. 8** The first zig. The initial counter value 0 is encoded below the seed in the format (??) with  $k = 1$  (1-bit width). Being fed with carry, the zig increments the count. Module H outputs 1, or more precisely a sequence of bead types which shall be interpreted as 1 in the next zag and reformatted, as a sum and cancels the carry.

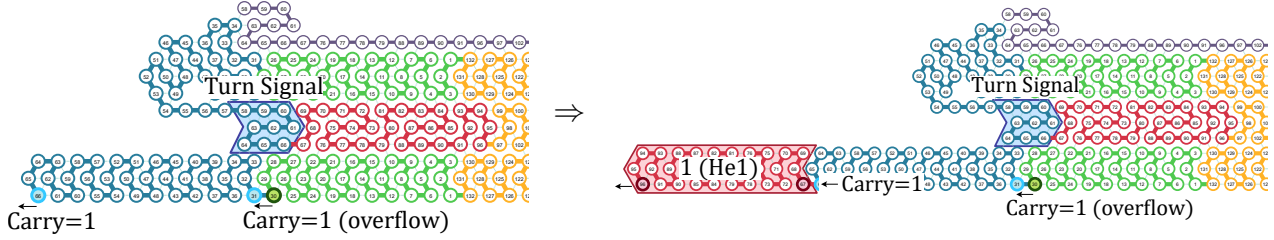


**Fig. 9** The first zag. A Turn Signal at the left end of the seed makes module L folds into the brick Lcrn to turn and initiate the zag. This brick is equipped with another Turn Signal for the sake of the next turn. In a zag, module F reads the output of module H from above and formats it for the sake of the next zig.

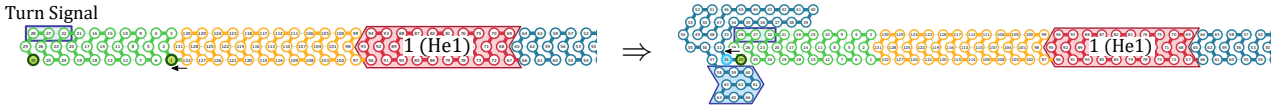
(6,15),	(18,69),	(34,47),	(73,88),	(103,108),	(8,12),	(22,36),	(38,43),	(77,81),	(109,114),
(6,19),	(18,70),	(34,58),	(74,77),	(103,113),	(8,18),	(22,75),	(40,45),	(77,86),	(109,120),
(6,81),	(18,71),	(34,63),	(74,78),	(103,114),	(8,73),	(22,76),	(40,58),	(77,87),	(109,123),
(6,82),	(18,72),	(34,64),	(74,83),	(103,122),	(8,78),	(22,78),	(41,43),	(77,92),	(110,113),
(6,83),	(18,88),	(35,39),	(74,84),	(103,123),	(8,87),	(23,26),	(41,44),	(77,93),	(110,114),
(6,84),	(19,24),	(36,43),	(74,87),	(104,107),	(9,17),	(23,27),	(41,45),	(78,127),	(110,119),
(6,91),	(19,26),	(36,44),	(75,132),	(104,108),	(9,72),	(23,28),	(41,57),	(78,132),	(110,120),
(6,92),	(19,71),	(36,45),	(75,83),	(104,113),	(9,73),	(23,73),	(42,54),	(78,99),	(111,117),
(7,12),	(19,81),	(36,60),	(75,96),	(104,115),	(9,83),	(23,74),	(42,56),	(79,84),	(112,117),
(7,13),	(20,23),	(36,64),	(76,81),	(106,111),	(9,86),	(23,75),	(43,48),	(79,88),	(113,116),
(7,18),	(20,24),	(37,42),	(76,87),	(107,109),	(9,87),	(24,71),	(44,47),	(79,90),	(113,117),
(7,83),	(21,37),	(37,43),	(76,93),	(107,110),	(10,15),	(24,72),	(44,48),	(79,96),	(114,122),
(7,89),	(22,27),	(38,41),	(76,95),	(108,124),	(10,67),	(25,30),	(45,51),	(79,97),	(115,120),
(8,11),	(22,28),	(38,42),	(77,80),	(108,125),	(10,79),	(25,60),	(46,51),	(79,98),	(116,120),



**Fig. 10** Overflow. When the carry has not been canceled out until the end of a zig, the Turn Signal is too far for the upcoming module, which is L, to be folded into the brick Lcrn for turn.



**Fig. 11** (Left) Starting from the bottom, the Turn Signal above is too far for this L to fold into Lcrn. It rather folds into Lbe and initiates bit expansion. (Right) Without anything around, the succeeding H folds into a glider (brick He1).



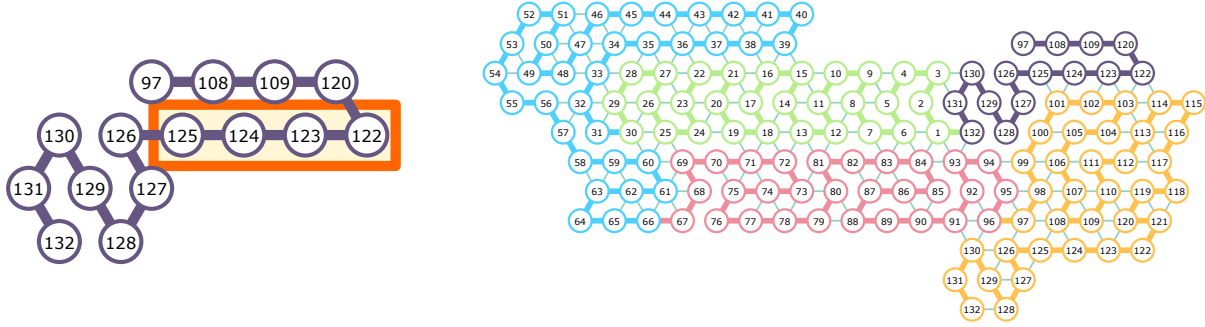
**Fig. 12** The succeeding R and F also fold into respective glider-like bricks. (Left) This brick of F (Fnb) exposes Turn Signal 28-27-22 (boxed), which is usually “hidden” under the previous zag. (Right) The exposed 28-27-22 (boxed) triggers the folding of next L into a special brick (Lcre) for left carriage-return.

(10,81),	(25,69),	(47,49),	(80,83),	(118,121),
(10,85),	(25,73),	(47,50),	(80,84),	(118,123),
(11,14),	(26,29),	(47,51),	(80,88),	(119,121),
(11,15),	(26,30),	(48,50),	(80,89),	(119,122),
(11,64),	(26,31),	(49,53),	(80,95),	(119,123),
(11,66),	(26,65),	(49,54),	(80,96),	(120,123),
(11,78),	(26,66),	(52,57),	(81,132),	(121,126),
(12,33),	(26,69),	(55,60),	(81,89),	(122,126),
(12,61),	(26,70),	(58,63),	(81,96),	(124,129),
(12,63),	(26,71),	(59,62),	(82,87),	(125,127),
(12,64),	(26,72),	(59,63),	(82,93),	(125,128),
(12,65),	(27,35),	(60,66),	(82,94),	(125,129),
(12,66),	(27,36),	(60,69),	(82,95),	(126,129),
(12,77),	(27,64),	(61,66),	(83,86),	(126,130),
(12,78),	(27,66),	(61,67),	(83,87),	(127,129),
(12,81),	(27,67),	(61,68),	(83,92),	(127,132),
(12,88),	(27,68),	(61,69),	(83,93),	(128,130),
(13,18),	(27,69),	(62,65),	(84,131),	(128,131),
(13,30),	(28,33),	(62,66),	(84,93),	(128,132),
(13,31),	(28,35),	(64,69),	(85,130),	
(13,32),	(28,60),	(64,85),	(85,90),	
(13,33),	(29,31),	(65,67),	(86,90),	

## 4 Conclusion

In this paper, we have expanded the existing finite binary counter by equipping it with a function of bit expansion at an overflow. This capability of handling overflows enables the system to launch at a simplest possible seed that encodes nothing but a Turn Signal for the first right carriage-return as long as the system starts counting from 0 (see Fig. ??).

This infinite counter could be used to simulate countably many Turing machines in parallel for proving the undecidability of problems in oritatami as done in [?, ?] for aTAM. However, the real significance of this counter in practice is that it can be embedded into a large-scale system as a counting module that can transition at an overflow into another phase, which does not have to do bit-expansion. A two-phase oritatami system to self-assemble the  $n \times n$  square is under construction; starting from an  $O(\log n)$ -size seed on which an initial counter value is encoded, it first counts up to  $2^{\log n}$  while drawing the diagonal, and at an overflow, it changes the direction of counting at 90 degrees and keeps just drawing diagonally until the diagonal reaches the right end



**Fig. 13** Counting from a non-coding seed. (Left) The simplest-possible seed that consists only of the boxed right Turn Signal. (Right) Starting from this non-coding seed, the first period results in the counter value 1 in the 1-bit.

of the counter. It is preferable to optimize this counter in these applications, for example, by reducing bead types, shortening the period, or simplifying its rule set, though such optimization problems are computationally hard in general [?,?].

**Acknowledgements** This work is supported in part by KAKENHI Grant-in-Aid for Challenging Research (Exploratory) No. 18K19779, for Scientific Research (B) No. 20H04141, and Scientific Research (C) No. 20K11672, and JST Program to Disseminate Tenure Tracking System No. 6F36, both granted to S. S.