On Algorithmic Self-Assembly of Squares by Co-Transcriptional Folding

Szilárd Zsolt Fazekas
Akita University, Japan

Hwee Kim
Incheon National University, Republic of Korea

Ryuichi Matsuoka
The University of Electro-Communications, Tokyo, Japan

Shinnosuke Seki
The University of Electro-Communications, Tokyo, Japan

Hinano Takeuchi
The University of Electro-Communications, Tokyo, Japan

Abstract

Algorithms play a primary role in programming an orchestrated self-assembly of shapes into molecules. In this paper, we study the algorithmic self-assembly of squares by RNA co-transcriptional folding in its oritatami model. We formalize the square self-assembly problem in oritatami and propose a universal oritatami transcript made of 939 types of abstract molecules (beads) and of period 1294 that folds deterministically and co-transcriptionally at delay 3 and maximum arity into the $n \times n$ square modulo horizontal and vertical scaling factors for all sufficiently large n's after building a $\Theta(\log n)$ width "ruler" that measures n upon the seed of size $\Theta(\log n)$ on which n is encoded in binary.

2012 ACM Subject Classification Theory of computation → Models of computation

Keywords and phrases Algorithmic molecular self-assembly, Co-transcriptional folding, Oritatami system, Self-assembly of squares

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2022.37

Supplementary Material *Software (Source Code)*: `https://github.com/rmatsuoka/oritatami_square`, archived at `swh:1:dir:016668d43c79b95a11aa013450ee472c2c7569cf`

Funding *Szilárd Zsolt Fazekas*: His research is supported by JSPS-KAKENHI Grant-in-Aid for Scientific Research (C) No. 19K11815.
Shinnosuke Seki: His research is supported in part by JSPS-KAKENHI Grant-in-Aid for Scientific Research (B) No. 20H04141 and (C) No. 20K11672.

Acknowledgements We would like to show our sincere gratitude towards Nicolas Schabanel and Ryuhei Uehara for their valuable comments in the development stage of the proposed square self-assembler. We would also like to thank the anonymous reviewers of ISAAC 2022 for their constructive comments, in particular for pointing out that mimicking the aTAM square construction's encoding of n in a base larger than 2 would not be feasible with a periodic transcript. We also want to thank Naoya Iwano and Yu Kihara for proofreading earlier drafts.

1 Introduction

RNA co-transcriptional folding (Figure 1) is a phenomenon in which the RNA single-stranded sequence (*transcript*) folds upon itself into a complex structure while being synthesized (*transcribed*) sequentially, that is, nucleotide by nucleotide, from its DNA template sequence. It plays a significant role in various computations in nature such as gene expression regulation [21] and RNA maturation [13, 18]. Geary, Rothemund, and Andersen have successfully programmed an artificial rectangular tile structure into a transcript, or more precisely into

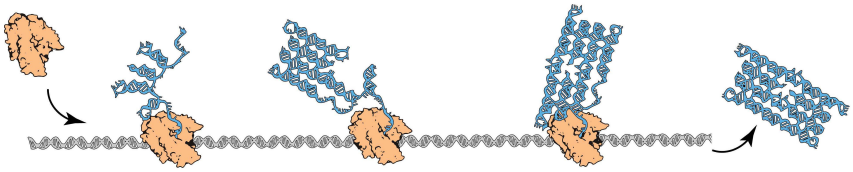

Figure 1 Co-transcriptional folding and RNA origami [8]. An RNA polymerase (colored in orange) scans a double-stranded DNA template (gray) from left to right and synthesizes the corresponding RNA sequence (blue) sequentially by mapping a DNA nucleotide read to the RNA nucleotide according to the rule `A` $\rightarrow$ `U`, `C` $\rightarrow$ `G`, `G` $\rightarrow$ `C`, and `T` $\rightarrow$ `A`. While being thus synthesized, the transcript folds upon itself into an intricate RNA structure.

its DNA template, in such a way that the transcript folds co-transcriptionally into the tile structure *in-vitro* [8]. This "RNA origami" architecture is spared the costly synthesis of RNA (it is much cheaper to synthesize DNA sequences commercially) and has been considerably automated, moduralized, and extended in size and structural diversity, for example, by the RNA Origami Automated Design Software (ROAD) [5]. Nevertheless, a programmed structure is still hardcoded in the sense that as soon as it is modified in size or shape, its transcript must be redesigned from scratch.

Algorithms and computation are fundamental in programming a transcript that folds co-transcriptionally into the shape in the class specified as input. Co-transcriptional folding has proven Turing universal in its abstract *oritatami* model [6]; therefore, any algorithm can be embedded in and guide co-transcriptional folding, though existing Turing universal oritatami systems [7, 16, 17] require much larger numbers of abstract nucleotides than 4 (`A`, `C`, `G`, `U`) as well as an intricate network of interactions among them, compared with the Watson-Crick complementarity `A`-`U` and `C`-`G`. It may however only be a matter of time before the Turing universal oritatami system is simplified enough for *in-vitro* implementation. This computational power has not been exploited yet for folding shapes co-transcriptionally. The oritatami shape builders [4, 9] still hardcode a given shape into a transcript. Masuda, Seki, and Ubukata proposed an oritatami transcript that can fold into an arbitrary finite portion of the Heighway dragon fractal H, known also as the paper folding sequence [12]. H is a sequence of L's and R's (left and right turns) and whether its i-th letter $H[i]$ is L or R can be computed by a 4-state deterministic finite automaton A, which reads the binary representation of i from its least significant bit and outputs L/R assigned to the state finally reached. The proposed transcript is periodic and its period consists of a binary counter to increment the counter value i by 1, a module to simulate A to compute $H[i]$, and a mechanical module to make a turn accordingly, all of which do their job while remembering i. It still hardcodes the bit-width k of the binary counter and cannot fold beyond $H[2^k-1]$ (it is open whether the fractal H can be folded in oritatami, c.f. [10]).

The algorithmic self-assembly of shapes by co-transcriptional folding should be studied more systematically. The assembly of arbitrary size squares is an appropriate first goal in shape assembly, due to both the ubiquity of the shape as building block for larger ones and the opportunity to relate the results to the rich record on square assembly in the DNA abstract Tile Assembly Model (aTAM) [1, 2, 3, 19], where it is considered to be a "benchmark problem" for complexity [15]. The $n \times n$ square S_n is the set of lattice points $\{(x, y) \mid 0 \leq x, y < n\}$. Unlike in the aTAM ([22]), it is challenging or even theoretically impossible to self-assemble a shape without scaling in oritatami [4, 9, 12]. Scaling S_n up by the constant c_w horizontally and by c_h vertically results in the rectangle $R_{c_w n \times c_h n} = \{(x, y) \mid 0 \leq x < c_w n, 0 \leq y < c_h n\}$ by mapping a point (i, j) in S_n to the rectangular region $\{(x, y) \mid c_w i \leq x < c_w(i + 1), c_h j \leq y < c_h(j + 1)\}$.

▶ **Definition 1** (Self-assembly of scaled-up square in oritatami). *A deterministic oritatami system* self-assembles the square at scale $c_w \times c_h$ *if for all sufficiently large* n*, starting from a seed that efficiently encodes* n*, its (unique) terminal assembly puts a bead in all rectangular regions of* $R_{c_w n \times c_h n}$*, each of which corresponds to a point in* S_n*, but no bead outside.*

The square self-assembly problem in oritatami naturally arises from Definition 1 as the problem of efficiently programming a deterministic oritatami system that self-assembles the square at as small scale factors as possible. It can be solved by using the compiler from a Turedo to a delay-3 deterministic oritatami system [16]. Turedo is a self-avoiding 2D Turing machine over the triangular grid; that is, its head cannot go back to any (hex) cell already visited (for its computational power, see also [14]). It can intrinsically simulate the square self-assembly system in the aTAM by Rothemund and Winfree [19], which consists of a constant number of tile types exclusively for counting in binary upon an $O(\log n)$-size initial assembly in order to build the $\lceil \log n \rceil \times (n - \lceil \log n \rceil)$ rectangle, for self-assembling a diagonal of S_n "cooperatively," and for filling, or even base conversion as a pre-process to let this aTAM system launch from the optimal $O(\log n / \log \log n)$-size initial assembly [1].

Nevertheless, it is worthwhile to propose an oritatami square self-assembler. Firstly, the oritatami system compiled from Turedo requires almost twice as many bead types as our proposed system (1735 compared with 939) and the period of its transcript is much longer due to the constant hidden in its asymptotic notation $\Theta(|Q|^6 \log |Q|)$, where Q is the alphabet of a simulated Turedo whereas the period of the proposed transcript is 394. The period in particular is directly translated to the size of the circular DNA template, that is, its synthesis cost. In addition, since every period folds into a regular hexagon, the resulting square would be at scale $\Theta\left(|Q|^3 \sqrt{\log |Q|}\right)$ and too bumpy to ignore in practice.

More importantly, the proposed transcript addresses the problem of how to run more than one computation in parallel upon co-transcriptional folding, which is inherently sequential. Solutions to fundamental challenges in oritatami have been almost always given as simple motifs or modules and seem not too far from solutions given by nature. For instance, a pseudoknotted linear structural motif in oritatami called a *glider* (see Figure 2) provided computations with a solid scaffold [7, 11], whereas a tandem of several pseudoknots is stacked coaxially into a long double-helix-like domain in order to support, for example, the tobacco mosaic virus [20]. Gliders can carry 1-bit in the direction of travel and this 1-bit-carrying capability was used, for instance, to wire half-adders in the oritatami binary counter, which is the first system implemented in oritatami [6] and later endowed with the capability to expand the bit-width at an overflow [11]. The counter folds its periodic transcript upon a seed that encodes an initial count in k bits in a zigzag manner so as to arrange half-adders into a 2D array with k half-adders along every zig and zag. The square self-assembler is based on the infinite binary counter and utilizes the overflow detectability rather for making a transition from Phase 1, in which it counts in binary to build a thin rectangle of height $c_h n$, to Phase 2, where it folds along this ruler into zigzags. In order to track a diagonal for halting, it displaces the half-adders along the diagonal by using a novel *elastic glider* module (Section 3). Elastic gliders are functionally upward compatible with gliders. They can shrink while carrying 1-bit. Furthermore, elastic gliders enable the square self-assembler to create and absorb offsets to propagate a 1-bit signal along the diagonal.

The system we propose does not strictly assemble a square in the sense of Definition 1: the shape assembled in Phase 2 is indeed a square satisfying the definition, but we also have the ruler shape from Phase 1 next to it. The square in Phase 2 is "thick" (as defined in aTAM square assembly [2]), that is, both its height and width are linear in n with constant ratio 16, whereas the Phase 1 rectangle results from the binary counting, hence it is thin with width logarithmic in n. In Section 5 we discuss the implications of our construction with regards to the side lengths in more detail.

We invite the interested reader to run the proposed square self-assembler on Schabanel's Simple OS Simulator available for free at:

`http://perso.ens-lyon.fr/nicolas.schabanel/OSsimulator/`

The OS specifications can be downloaded from the link below. We also included a readable listing of the rule set and the transcript in separate files.

`https://github.com/rmatsuoka/oritatami_square`

2 Preliminaries

Let Σ be a finite alphabet of types of abstract molecules, or *beads*. A bead of type $a \in \Sigma$ is called an a-bead. By Σ^* and Σ^ω, we denote the set of finite sequences of beads and that of one-way infinite sequences of beads, respectively, both including the empty sequence λ. Let $w = b_1 b_2 \cdots b_n \in \Sigma^*$ be a sequence of length n for some $n \geq 0$ and bead types $b_1, \ldots, b_n \in \Sigma$. 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.

In oritatami, a transcript, which is a directed sequence of beads, folds over the triangular grid graph $\mathbb{T} = (V, E)$ co-transcriptionally. A directed path $P = p_1 p_2 \cdots p_n$ in $\mathbb{T}$ is a sequence of *pairwise-distinct* points $p_1, p_2, \ldots, p_n \in V$ such that $\{p_i, p_{i+1}\} \in E$ for all $1 \leq i < n$. Its i-th point p_i 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 Σ) is a triple (P, w, H) of a directed path P in $\mathbb{T}$, a sequence of bead types $w \in \Sigma^*$ which is as long 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. 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. A *rule set* $R \subseteq \Sigma \times \Sigma$ is a symmetric relation over Σ, 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. By $\mathcal{C}(\Sigma)$, we denote the set of all conformations over Σ; its argument Σ is omitted whenever Σ is clear from the context.

Conformations grow 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 $H' \subseteq \{\{i, |w|+1\} \mid 1 \leq i \leq |w|, \{P[i], p\} \in E, (w[i], b) \in R\}$ 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* Ξ is a tuple $(\Sigma, R, \delta, \sigma, w)$, where Σ and R are defined as above, δ is a positive integer called *delay*, $w \in \Sigma^* \cup \Sigma^\omega$ is a (possibly infinite) *transcript*, and σ is an R-valid initial conformation called a *seed*, which can be considered not as a part of the system but as an input to the system. The definition usually includes a parameter called arity, which

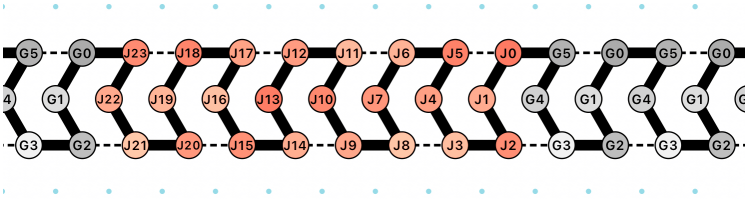

Figure 2 Glider motif in oritatami at delay 3. Here it folds co-transcriptionally from right to left.

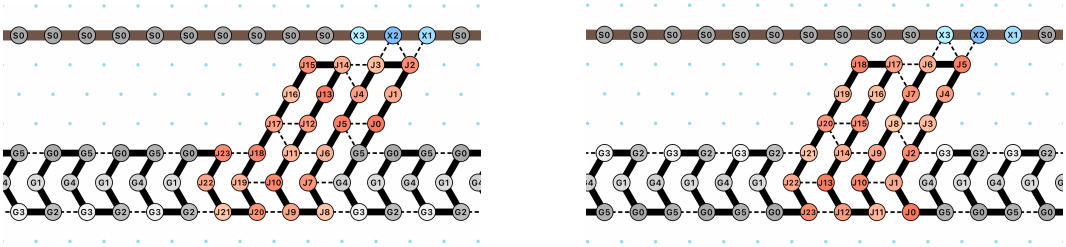

Figure 3 Elastic glider in the shrunk conformations.

is often assumed to be inexhaustible, that is, 6, which is our case as well. We opted to omit the parameter here to simplify the formalism. The transcript is to be synthesized one bead at a step. At the i-th step it has been already stabilized up to $w[i-1]$ as an elongation C_{i-1} of the seed by $w[1..i-1]$ and only the fragment $w[i..i+\delta-1]$ of the most nascent δ-beads - beyond which w has not been transcribed yet - can jiggle to minimize energy collaboratively. The set $\mathcal{F}(\Xi)$ of conformations *foldable* by the system Ξ is recursively defined as: the seed σ is in $\mathcal{F}(\Xi)$; and provided that elongation C_i of Σ by the prefix $w[1\dots i]$ be foldable (i.e., $C_0 = \sigma$), its further elongation C_{i+1} by the next bead $w[i+1]$ is foldable if

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

Then we say that the bead $w[i+1]$ and its bonds are *stabilized* according to C_{i+1} and they will not move or break any more. A conformation foldable by Ξ is *terminal* if none of its elongations is foldable by Ξ. The oritatami system Ξ is *deterministic* if for all $i \ge 0$, there exists at most one C_{i+1} that satisfies (1). A deterministic oritatami system folds into a unique terminal conformation. All the systems presented in this paper are deterministic and we set $\delta = 3$.

In the next section, we illustrate the dynamics (1) by introducing a novel oritatami motif of practical use called an *elastic glider*.

3 Elastic Glider

The square self-assembler which we shall describe in Section 4 has to remember where the diagonal is while counting in binary. As done in the existing oritatami binary counters [6, 11], its periodic transcript folds in a zigzag manner into a 2D array of half-adders such that in each zig (←) the carry-out of a half-adder is wired to the carry-in of the half-adder transcribed next by a self-standing motif called a *glider*, which folds at delay 3 as shown in Figure 2 in the direction indicated by its arrow-like shape (leftward in the figure, for instance). The capability of gliders to carry 1 bit is thus already used up and transferring more bits in the direction of folding is yet to be achieved in the theory of oritatami systems.

The square self-assembler overcomes this limit by displacing all and only half-adders on the diagonal together with some preceding and succeeding modules using a novel *elastic glider* module, colored in orange in Figures 2 and 3. The elastic glider also works at delay 3 and is upward compatible with the standard glider in the sense that besides the conventional "stretched-out" glider conformation (Figure 2), it can shrink in the two ways shown in Figure 3 below the tandem of beads `X3-X2-X1`.

Let us see how an elastic glider shrinks as a demonstration of oritatami dynamics (1) at $\delta = 3$. Initially, the nascent fragment of its first δ beads, `J0-J1-J2`, is attracted by `G3` but pulled upward more strongly by `X2-X1` with two bonds (Fig. 3, left); in fact, folding this fragment in any other way results in at most one bond. As a result, `J0` is stabilized to the northeast of `G5`. These two bonds between `J2` and `X2-X1` also stabilize `J1` and `J2` as shown in Figure 3 (left). The bond `J3-X2` guides the transcript leftward and the fragment `J3-J4-J5` folds back along `J0-J1-J2` just stabilized due to the original glider rule (`J5`, `J0`) as well as a new rule (`J5`, `G5`), which does no harm in the usual glider conformation as when `J5` is transcribed after `J2` is stabilized, `G5` is "too far." The fragment `J6-J7-J8` could fold back along `J3-J4-J5` just stabilized but rather folds downward along `G3-G4-G5` due to the new rules (`J7`, `G4`) and (`J8`, `G3`); observe that beads paired here are too far from each other to bind in the glider conformation. The succeeding `J9-J10-J11` folds as in the usual glider. This is the functional unit of this module and it can shorten the glider by 2. Arbitrarily many, say k, functional units can be repeated and the resulting elastic glider, of $12k$ beads length, yields the offset $2k$ (the case of $k = 2$ is illustrated in Figures 2 and 3); let us call this the *elastic glider for offset* $2k$. Note that the second and subsequent units do not need to be suspended from the ceiling for shrinking because they can rather rely on their shrunk predecessor as `J14` binds to `J3-J4`. With the preceding glider being vertically flipped (`G5` at the bottom), the first unit `J0-J1-`$\cdots$`-J11` folds rather as shown in Figure 3 (right). Compare Figure 3 (left) with (right) and observe that they yield the same offset and the last bead of every unit is at the same height as the preceding `G5`; this means that this module can shrink independently of whether 0 or 1 is propagated.

Offsets can be not only created but absorbed by elastic gliders. Observe in Figure 3 how critical it is to place the tandem `X3-X2-X1` precisely as illustrated relative to the preceding glider. Starting too early (right) or too late (left), this elastic glider would stretch out to the normal glider. This property enables the square self-assembler as shown in Figure 8 to shift half-adders considerably (by 6 points) by sandwiching them by elastic gliders *Pre*, *Post*, D, and *spc* as well as to have this D push the succeeding transcript up to the next *Pre* slightly (by 2 points) to propagate a signal to track the diagonal.

4 Algorithmic Oritatami Square Self-Assembler

The infinite binary counter [11] serves as a basis of the proposed square self-assembler. The transcript of the counter is made of four glider-based modules F (formatter), L (left carriage return: CR), H (half-adder), and R (right CR) which are transcribed in this order periodically infinitely many times; that is, it can be represented as $(FLHR)^*$. All the modules can fold into a glider of even length, which puts the first and last beads of their transcript on the same side (top or bottom; see `J0` and `J23` in Figure 2). We shall explain shortly how this property enables gliders to wire half-adders. Below a seed on which the count is initialized, the transcript folds into zigs ($\leftarrow$) and zags ($\rightarrow$) as:

Figure 4 (Left) The glider-like conformation `Rg` that R takes during zigs and zags; (Right) The right-CR conformation of R `Rcr`.

$$
\begin{array}{llllll}
 & & \text{Seed} & & & \\
\hline
 & L\circlearrowleft & FRHLFRH\leftarrow & \cdots & \leftarrow LFRHLF & \circlearrowright \\
 & & HRFLHRF\rightarrow & \cdots & \rightarrow LHRFLH & \\
 & L\circlearrowleft & FRHLFRH\leftarrow & \cdots & \leftarrow LFRHLF & \circlearrowright R \\
 & & HRFLHRF\rightarrow & \cdots & \rightarrow LHRFLH & \\
L\circlearrowleft & FRH\ L\leftarrow & FRHLFRH\leftarrow & \cdots & \leftarrow LFRHLF & \circlearrowright R
\end{array}
\tag{2}
$$

Each period $HRFL$ folds into a glider-like linear conformation (see Figure 4 for such a conformations that an instance of corresponding right-CR module R takes in the proposed square self-assembler) except at the left end of the counter where an instance of L takes a special CR conformation $\circlearrowleft$ to guide the transcript into the next zag or at the right end where an instance of R does a carriage-return alike. The instance of L behaves thus differently there due to a special configuration of beads exposed below the instance of L above, which has also folded into the left CR conformation. This is also the case for R; the corresponding module of the same name in the proposed square self-assembler admits the two conformations in Figure 4. Observe that a period $HRFL$ in a zag is aligned vertically with another period in the preceding zig so as for their R's are one above the other; see those colored in blue and red in (2). These periods collaboratively function as one half-adder in a way explained shortly so that we collectively call them a *functional unit*. The binary counter thus co-transcriptionally arranges functional units into a 2D array.

Among the two instances of H and those of F in a functional unit, that of H in a zig and that of F below actually play a functional role whereas the other H and F are idled; let us call these active ones its *zig* H and *zag* F, respectively (we shall refer to other modules, say X, of a functional unit of the square self-assembler as the *zig* X or *zag* X). The zig H actually serves as a half-adder. It receives a 1-bit carry-in c_{in} propagated through the zig from the previous H and another 1-bit b from the zag F of the functional unit above, and carries out $c_{\text{in}} \wedge b$ to the next H and outputs c_{in} `xor` b to the zag F below, which formats the output for the sake of the zig H of the functional unit which will be transcribed later below. Instance of the corresponding formatting module F_H of the square self-assembler take one of the two conformations `FH0` and `FH1` in Figure 6 in order to format the outputs 0 and 1 from above, respectively, in a zag. In a zig, they rather fold into the glider-like conformation `FHg` in Figure 7. Other modules also fold into their own glider-like conformation in a zig so that a

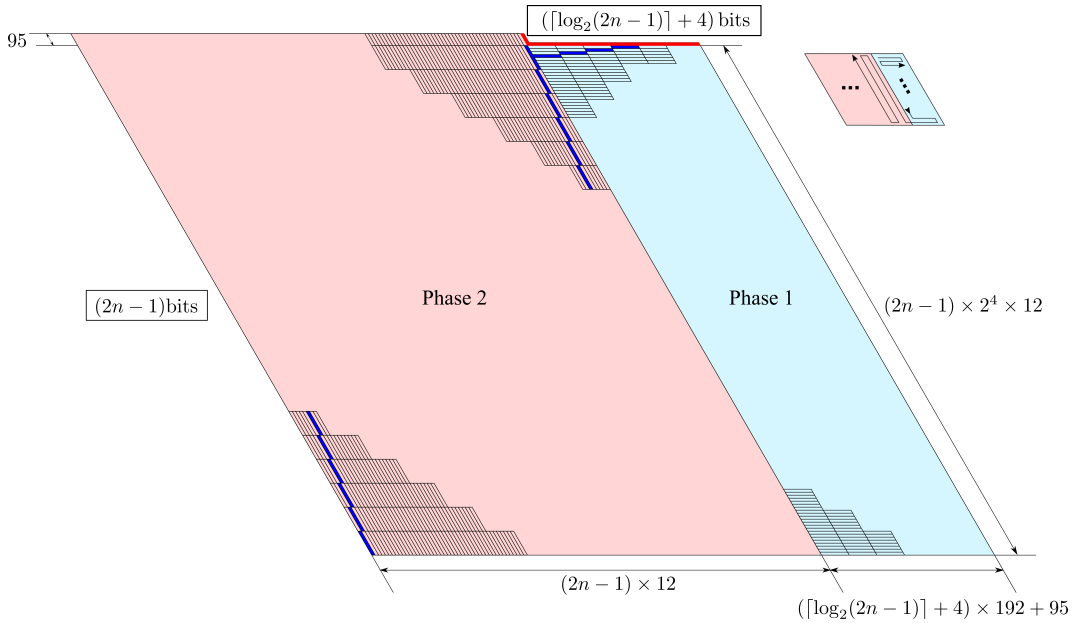

Figure 5 Propagation of a diagonal signal through the 2D array of $(\lceil \log(2n-1) \rceil + 4) \times 16(2n-1)$ functional units in Phase 1 and the 2D array of $(2n-1) \times (2n-1)$ functional units in Phase 2. Thin rectangles of size 192×12 in Phase 1 or 12×192 in Phase 2 are a functional unit and the signal propagates through those colored in dark blue. The seed is colored in red.

carry propagates from the zig H's of a functional unit to that of the next functional unit as a position of the last bead of each module (top for no-carry, bottom for carry). Thus, at an overflow, a zig ends at the bottom so that the CR signal is too far for L being transcribed to do a carriage-return; this instance of L and the succeeding H, R, and F, all fold into gliders to expand the counter bit-width by 1. The square self-assembler does not need bit-width expansion. As overviewed in Figure 5, it rather exploits this overflow detectability rather to have the instance of L guide the transcript from Phase 1, in which a tall rectangle of height $c_h n$ has self-assembled upon an $O(\log n)$-size seed (see Figure 14) by thus counting in binary, to Phase 2, where the zigzag counting continues after the count is reset to 0 along the left long side of the rectangle, by taking the 90-degree right-turn conformation; see the bottom-right instance of L, colored in teal, in Figure 10. Since the overflow till the next right CR, the transcript proceeds along the left CRs northwestward, which have all the instances

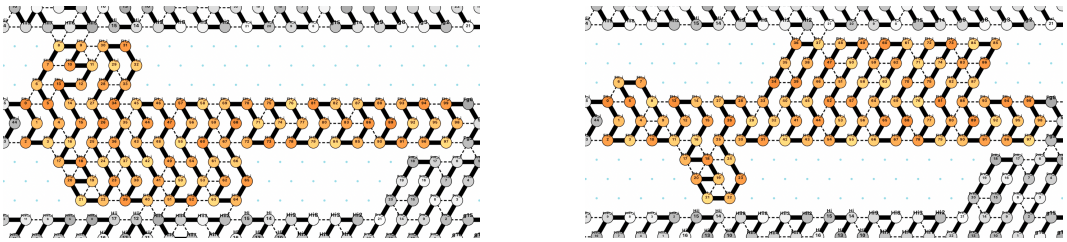

Figure 6 The two conformations `FH0` and `FH1` that an instance of the module F_H can take in a zag. Until the first right CR after the phase transition, all the instances of F_H take `FH0`; in this way, the binary counter is reset to 0 and thus guarantees that it will not be overflown before the square is completed.

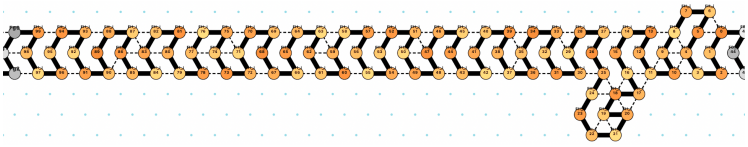

Figure 7 The glider-like conformation `FHg` into which all the instances of F_H folds in a zig.

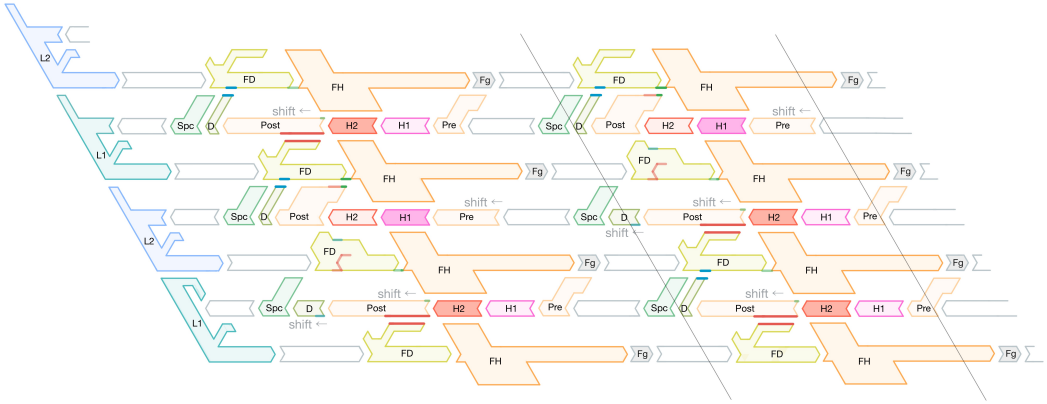

Figure 8 Diagonal signal propagation by elastic gliders *Pre*, *Post*, *D*, and *Spc*. Only along the diagonal, *Pre* is stretched and shrinks the succeeding *Post*. The shrunk *Post* folds F_D below differently, and stretches *D* further below to push the next *Pre* forward so that it cannot shrink.

of F_H take `FH0`; this means that the binary counter is reset to 0 at the beginning of Phase 2 and guarantees that it does not overflow before the diagonal reaches the opposite corner, where the system is supposed to halt.

In order to halt, the self-assembler tracks the left-down diagonal ($\swarrow$) of the square by displacing a functional unit on the diagonal leftward by 2 points and by having this shift be propagated diagonally to displace the next functional unit on the diagonal alike (see Figure 8). Reaching the left end of the counter, this displacement folds an instance of L into a special left-CR conformation shown in Figure 9 (Right), which will remind the transcript in Phase 2 of where the diagonal is; compare the conformations of the yellow module, F_D, to the left of L in Figure 9, depending on whether it is (Left) away from the diagonal or

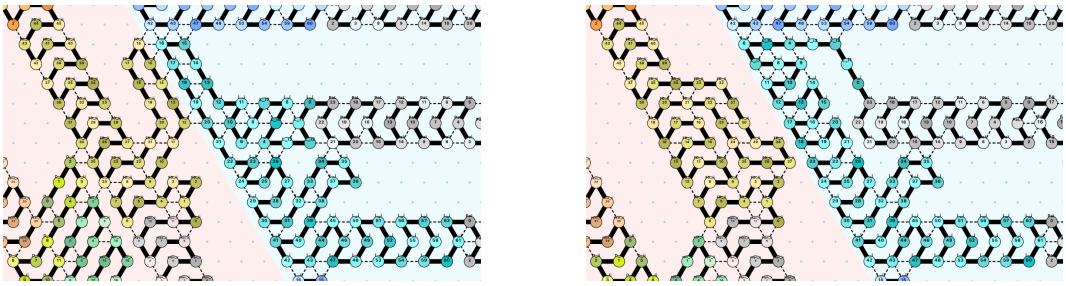

Figure 9 The two carriage-return (CR) conformations of L_1 (colored in teal). Below an instance of L_2 (blue) in a CR conformation, an instance of L_1 does a carriage-return in the way shown left unless the preceding glider is pushed leftward (by the diagonal signal), where it does a carriage-return as shown right in order to remind an instance of F_D (yellow), if transcribed to its left, of where the diagonal is. An instance of L_2 always takes the left conformation for carriage-return.

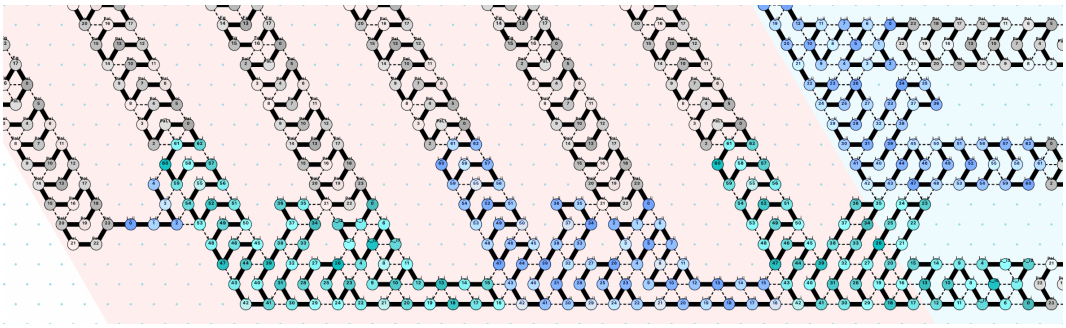

Figure 10 Phase transition by L_1 (colored in teal) and halting by L_2 (blue). The bottom right instance of L_1 starts folding at the bottom, that is, with a carry due to the counter overflow. In this environment, the instance of L_1 takes the phase transition conformation `Lpt`. Afterward, the system keeps counting in binary in Phase 2 until the diagonal signal reaches the bottom edge and pushes an instance of L_2 transcribed there further downward to trap the transcript for halting.

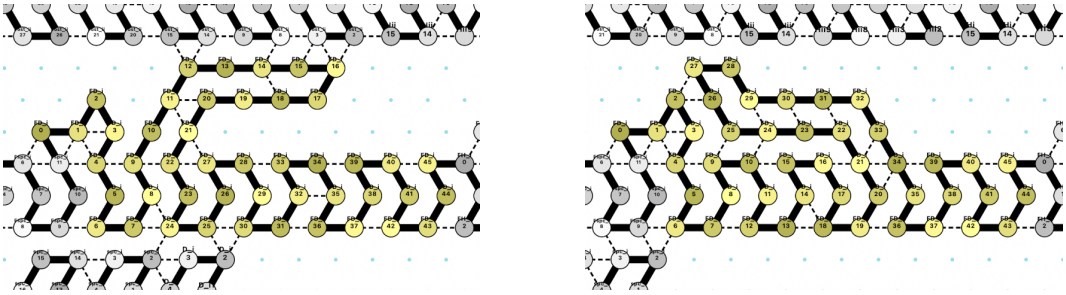

Figure 11 Two conformations `FDf` and `FDt` that F_D takes in a zag when being (Left) away from the diagonal and (Right) on the diagonal, respectively. In a zig, F_D always folds into `FDt` or upside-down.

(Right) on it (see also Figure 11). Afterward, the diagonal signal propagates in Phase 2 until it reaches the left end (at the bottom in figures), where an instance of L traps the transcript inside the region enclosed by the previous zag and zig to halt the system.

As described so far, L must play the two extra functional roles at the left edge of the counter in this square self-assembler: of letting the diagonal signal pass through the phase boundary and of stopping it at the end of Phase 2. It is highly non-trivial to propagate a diagonal signal differently in Phase 1 and in Phase 2, and hence, we could not help but rather introduce two types of L. We indeed replace every other instance of L along the transcript with its "doppelgänger," which is implemented like L but from bead types not used in L. The period is thus doubled, and one period now contributes to two functional units. Observe in (2) that every zig or zag involves an even number of L's as suggested in (2). Therefore,

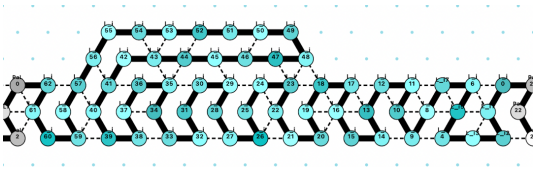

Figure 12 The glider-like conformation `Lg` that L_1 and L_2 always fold into in the middle of zigs and zags.

Table 1 The half-period of the transcript of the square self-assembler at modular level.

	Spacer
Pre:	Elastic glider for offset 4
H_1, H_2:	Half-adder and its doppelgänger (Figure 13)
Post:	Elastic glider for offset 6
D:	Diagonal tracker (Elastic glider for offset 2)
Spc:	Elastic glider for offset 4
	Spacer
R:	Right turner (Figure 4)
	Spacer
F_D:	Formatter for D (Figure 11)
F_H:	Formatter for H (Figures 6 and 7)
F_g:	(Conventional) glider
	Spacer
L_1 or L_2:	Left turner and its doppelgänger (Figures 9, 10, and 12)

along the left end of the counter thus modified, the two types of left-turners, say L_1 and L_2 (colored in teal and in blue, respectively), occur alternately one below another in the CR conformation. Two instances of L_1 still get one above the other inside the counter and so do those of L_2, where introducing an "intra-modular" rule to fold an instance of L_1 in some specific manner may cause an "inter-modular" interference and vice versa. Since this problem cannot be solved no matter how many types of doppelgängers are available, we rather keep a zig away from both the zags above and below by 3 points. Note that both L_1 and L_2 fold into the glider-like conformation in Figure 12 in the middle of zigs and zags.

Based on this modified counter, we now complete the design of the square self-assembler. Its transcript is obtained by further inserting elastic gliders for diagonal signal propagation, which shall be described in detail in Section 4.1, as well as spacers into the half-period as described in Table 1. The spacers, which always fold into a glider, make sure that the ratio of the height of one zigzag, which is 12 due to the space between zigs and zags, to the length of the linear structure into which the half-period folds is a power of 2; here 16 is the smallest possible ratio to accommodate all the functional modules. As a result, each of the two half-periods of a functional unit folds into a glider-like linear structure of width $12 \times 16 = 192$ and of height 6 except at the left and right end of the counter. Though these two structures are not perfectly aligned vertically, it is more convenient for our argument to assume that a functional unit folds into a rectangle of width 192 and height 12 as abstracted in Figure 5. Indeed, this assumption does not impair any mathematical rigor because it is just up to us which bead to designate as an origin of a half-period.

4.1 Diagonal signal propagation for halt

Figure 8 illustrates how instances of the modules *Pre*, H_1, H_2, *Post*, D, F_D, and F_g collaborate for propagating a diagonal signal from top right to bottom left through Phases 1 and 2.

In a zig, the half-period of the functional unit along the diagonal is pushed 2 points leftward by the previous half-period, or more precisely, by its zig D. Its zig *Pre* thus gets too far from an instance of F_g above to be suspended for shrinking. It hence rather stretches out and amplifies the offset from 2 to 6. The resulting offset 6 pushes H_1H_2 leftward but H_1 is interlocked to the counter circuit in substitution for H_2. The succeeding instance of *Post* fully absorbs the offset by shrinking. The glider conformation of *Post* is provided with a

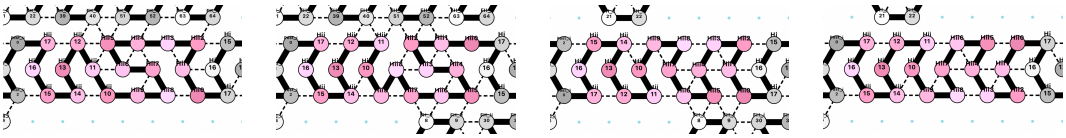

Figure 13 The four conformations of H_1 and of H_2: from left, `H0+c`, `H0+n`, `H1+c`, and `H1+n`. A carry-in c_{in} is given as whether the previous glider ends at the top (no-carry) or bottom (carry) whereas the input 1-bit b is fed from above as whether an instance of F_H just above ($b = 0$) or far above ($b = 1$).

sequence of bead types along both sides that attracts an instance of F_D below in the next zag so as for the instance to take the conformation `FDf` as shown in Figure 11 (Left). Both of these "signals" are however convoluted inside when an instance of *Post* shrinks and let an instance of F_D below fold freely into `FDt` shown in Figure 11 (Right). The zig D of the functional unit below stretches out if this zag F_D takes `FDt`, or it shrinks otherwise. These conformations result in the offset 2 and it is propagated leftward through the zig up to the next *Pre*, which is the zig *Pre* of the next functional unit on the diagonal. The signal has successfully propagated along the diagonal. Reaching the left end of the counter instead and moreover an instance of L_1 is being transcribed, the offset 2 has the instance of L_1 fold into `Lcrd`, one of the two possible CR conformations of L_1, as shown in Figure 9. If an instance of L_2 is being transcribed there instead, the offset guides the transcript inside the enclosed region to halt the system, as shown in Figure 10.

Note that one half-period is provided with two half-adder modules H_1 and H_2, which are a doppelgänger of each other, made of disjoint bead type sets. Both of them are indispensable as long as the diagonal signal is propagated as just described due to the slide of the whole half-period. Without any offset, H_2 interlocks with the instance of F_H above and H_1 is idle. The offset 2 must be amplified to let H_1 interlock instead because half-adder modules have not been implemented so compactly yet [6, 11, 12]. These half-adder modules are implemented so as to admit the four conformations of width 6 shown in Figure 13. The offset 2 is hence amplified to 6 by *Pre*. If we implemented D as an elastic glider with offset 6, then *Pre* would be unnecessary. However, L_1 cannot handle offset 6, so we amplify the offset temporarily by *Pre* and then absorb it by *Post*, so the offset 6 never reaches L_1.

Last but not the least, the module *Spc*, which is an elastic glider for offset 4, makes sure that all the zigs and zags are exactly the same length even with all these expansion and contraction mechanisms by shrinking in a zig and expanding in a zag. It turned out that for this purpose it suffices for its instances to always shrink in a zig and stretch out into the glider in a zag. This module guarantees more strongly that all the half-periods except those in a zig along the diagonal or the sub-diagonal fold the same in width; those in a zig along the sub-diagonal are longer than usual by 2 but this is immediately cancelled out by the succeeding half-period, which is on the diagonal and shorter by 2.

4.2 Seed, Phase Boundary, and Scaling

One functional unit is of width 12 and of height 12×2^4 in Phase 2 so that it spans 16 zigzags, that is, functional units in Phase 1. See Figure 5. We set the initial value of the counter as $c_0 = \left(2^{\lceil \log(2n-1) \rceil} - (2n-1)\right) \times 2^4$ and encode its binary representation $b_{k-1}b_{k-2}\cdots b_1 b_0$ with $k = \lceil \log(2n-1) \rceil + 4$ and $b_0, b_1, \ldots, b_{k-1} \in \{0, 1\}$ upon the seed as this value is encoded below the zag; see Figure 14. Starting from this seed, the transcript folds into $(2n-1) \times 2^4$ zigzags until the counter overflows (we shall explain why not n but $2n-1$ shortly). Recall

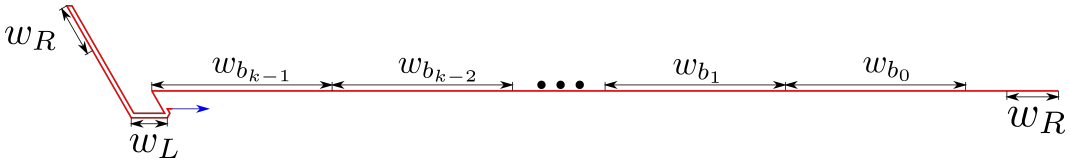

Figure 14 Encoding of an integer represented in binary as $b_{k-1}b_{k-2}\cdots b_1b_0$ upon the seed. w_L and w_R are the configurations of beads exposed at the bottom of the left CR configuration(s) of L_1 and that of R, respectively.

that in its CR conformation, L_1 lets the diagonal signal pass through while L_2 stops it. Thus, the diagonal signal must be launched properly so as to hit an instance of L_1 at the left edge of the counter in Phase 1. More restrictively, it must hit the 4th CR from above of the 16 zigzags that is spanned by a functional unit of Phase 2 so that the zag F_D can read the signal. Thus, the very first left-CR of the counting must be done by an instance of L_2. The final zig in Phase 1 ends with an instance of L_1, which detects the overflow and takes the phase transition conformation `Lpt` as shown in Figure 10. In Phase 2, the diagonal signal must hit the left CR of L_2 to halt. Since phase transition has been made by L_1, the number of zigzags in Phase 2 must be odd. To make sure that happens, we need an even number of functional units below the one which starts the diagonal signal of Phase 2. Therefore, we bundle every 32 zigzags in Phase 1 from the bottom, map the corresponding 2 functional units (one period) in Phase 2 to one lattice point in the $n \times n$ square S_n, and have the diagonal signal hit the 20th CR from above of such a bundle. We bundle the first 16 actual zigzags with 16 imaginary zigzags represented by the seed and hence the diagonal signal hits to its "20th" CR (actually 4th) by launching the signal from the 4th most significant bit of c_0 upon the seed.

We could, in fact, start the diagonal signal of Phase 2 directly from the seed making the construction perhaps somewhat simpler by avoiding signal passing between the phases. However, this would not preclude having to combine the binary counting with diagonal signal propagation as we only have one transcript which has to take care of both phases. Moreover, the present construction is "ready" for generalization: if one can implement a slowdown of the diagonal signal, they then can start it from the top-right corner of Phase 1 and have the whole assembly conform to Definition 1.

5 Conclusions

As explained earlier, the size of one functional unit is 192×12 and this means that the difference between the width and height of the Phase 2 rectangle is linear in n with a factor $2 \cdot (192 - 12)$. By this it is clear that Phase 1 cannot make up for the difference if we want to maintain the efficiency of having a $o(n)$ sized seed in our current construction.

The rectangle that the proposed system folds into is too thick to be self-assemblable by merely counting in binary, unless we trade seed size for scale and program size complexity by simply using a binary counter with a linear-sized seed and a suitably implemented halting module, which terminates on overflow. However, it is not thick enough yet even with Phase 2, as its aspect width-to-height ratio is asymptotically 16. The periods have been lengthened by spacers in order to make the aspect ratio of the rectangle be a power of 2, so that we can sync the phase bottoms. It should be constructive and resource-friendly to utilize the idle lot, where spacers are, for example, by embedding a yet-to-be-implemented mechanism to decelerate the diagonal signal exponentially, which would allow a Phase 1 like construction

throughout. Even if one could find a module which slows the diagonal signal by a fixed amount, inserting multiple copies of it might be enough to sync the vertical and horizontal propagation speeds, which would make the Phase 2 rectangle closer to, or exactly a square.

Finally, perhaps the proposed system could be programmed more efficiently by encoding n in a larger base than 2 and implementing a mechanism to decode it into its binary representation, as done for optimal program-size self-assembly of squares in aTAM [1]. The aTAM method uses tiles tailored for each n, so one would need a transcript with a prefix which decodes the large base encoding of n and a succeeding periodic part which assembles the square as proposed here. Therefore, instead of a purely periodic transcript, the optimal system would require an ultimately periodic one. At this moment it is not clear whether such a transcript is feasible.

References

1 Leonard Adleman, Qi Cheng, Ashish Goel, and Ming-Deh Huang. Running time and program size for self-assembled squares. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC 2001)*, pages 740–748. ACM, 2001.

2 Gagan Aggarwal, Qi Cheng, Michael H. Goldwasser, Ming-Yang Kao, Pablo Moisset de Espanes, and Robert T. Schweller. Complexities for generalized models of self-assembly. *SIAM Journal on Computing*, 34(6):1493–1515, 2005.

3 Florent Becker, Eric Rémila, and Nicolas Schabanel. Time optimal self-assembly for 2D and 3D shapes: The case of squares and cubes. In *Proceedings of the 14th International Meeting on DNA Computing (DNA 14)*, volume 5347 of *LNCS*, pages 144–155. Springer, 2008.

4 Erik D. Demaine, Jacob Hendricks, Meagan Olsen, Matthew J. Patitz, Trent A. Rogers, Nicolas Schabanel, Shinnosuke Seki, and Hadley Thomas. Know when to fold 'em: Self-assembly of shapes by folding in oritatami. In *Proceedings of the 24th International Conference on DNA Computing and Molecular Programming (DNA 24)*, volume 11145 of *LNCS*, pages 19–36. Springer, 2018.

5 Cody Geary, Guido Grossi, Ewan K. S. McRae, Paul W. K. Rothemund, and Ebbe S. Andersen. RNA origami design tools enable cotranscriptional folding of kilobase-sized nanoscaffolds. *Nature Chemistry*, 13:549–558, 2021.

6 Cody Geary, Pierre-Étienne Meunier, Nicolas Schabanel, and Shinnosuke Seki. Programming biomolecules that fold greedily during transcription. In *Proceedings of the 41st International Symposium on Mathematical Foundations of Computer Science (MFCS 2016)*, volume 58 of *LIPIcs*, pages 43:1–43:14, 2016.

7 Cody Geary, Pierre-Étienne Meunier, Nicolas Schabanel, and Shinnosuke Seki. Proving the turing universality of oritatami co-transcriptional folding. In *Proceedings of the 29th International Symposium on Algorithms and Computation (ISAAC 2018)*, volume 123 of *LIPIcs*, pages 23:1–23:13, 2018.

8 Cody Geary, Paul W. K. Rothemund, and Ebbe S. Andersen. A single-stranded architecture for cotranscriptional folding of RNA structures. *Science*, 345(6198):799–804, 2014.

9 Yo-Sub Han and Hwee Kim. Construction of geometric structure by oritatami system. In *Proceedings of the 24th International Conference on DNA Computing and Molecular Programming (DNA 24)*, volume 11145 of *LNCS*, pages 173–188, 2018.

10 Yo-Sub Han and Hwee Kim. Impossibility of strict assembly of infinite fractals by oritatami. *Natural Computing*, 20(4):691–701, 2021.

11 Kohei Maruyama and Shinnosuke Seki. Counting infinitely by oritatami co-transcriptional folding. *Natural Computing*, 20(2):329–340, 2021.

12 Yusei Masuda, Shinnosuke Seki, and Yuki Ubukata. Towards the algorithmic molecular self-assembly of fractals by cotranscriptional folding. In *Proceedings of the 23rd International Conference on Implementation and Application of Automata (CIAA 2018)*, volume 10977 of *LNCS*, pages 261–273, 2018.

13 Evan C. Merkhofer, Peter Hu, and Tracy L. Johnson. Introduction to cotranscriptional RNA folding. In *Methods in Moleclar Biology*, volume 1126, pages 83–96. Springer, 2014.

14 Samuel Nalin and Guillaume Theyssier. On turedo hierarchies and intrinsic universality. In *Proceedings of the 28th International Conference on DNA Computing and Molecular Programming (DNA 28)*, volume 238 of *LIPIcs*, pages 6:1–6:18, 2022.

15 Matthew J. Patitz. An introduction to tile-based self-assembly and a survey of recent results. *Natural Computing*, 13(2):195–224, 2014.

16 Daria Pchelina, Nicolas Schabanel, Shinnosuke Seki, and Guillaume Theyssier. Oritatami systems assemble shapes no less complex than tile assembly model (aTAM). In *Proceedings of the 39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022)*, volume 219 of *LIPIcs*, pages 51:1–51:23, 2022.

17 Daria Pchelina, Nicolas Schabanel, Shinnosuke Seki, and Yuki Ubukata. Simple intrinsic simulation of cellular automata in oritatami molecular folding model. In *Proceedings of the 14th Latin American Symposium on Theoretical Informaitcs (LATIN 2020)*, volume 12118 of *LNCS*, pages 425–436, 2020.

18 Roberto Perales and David Bentley. "Co-transcriptionality" – the transcription elongation complex as a nexus for nuclear transactions. *Molecular Cell*, 36(2):178–191, 2009.

19 Paul W. K. Rothemund and Erik Winfree. The program-size complexity of self-assembled squares (extended abstracts). In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC 2000)*, pages 459–468. ACM, 2000.

20 Alex van Belkum, Jan Pieter Abrahams, Gornelis W. A. Pleij, and Leender Bosch. Five pseudoknots are present at the 204 nucleotides long 3' noncoding region of tobacco mosaic virus RNA. *Nucleic Acids Research*, 13(1):7673–7686, 1985.

21 Kyle E. Watters, Eric J. Strobel, Angela M. Yu, John T. Lis, and Julius B. Lucks. Cotranscriptional folding of a riboswitch at nucleotide resolution. *Nature Structural and Molecular Biology*, 23(12):1124–1131, 2016.

22 Erik Winfree. *Algorithmic Self-Assembly of DNA*. PhD thesis, California Institute of Technology, 1998.