ACS OMEGA

http://pubs.acs.org/journal/acsodf

Article

M-State and *N*-Color (*M*−*N* = 1−1, 2−1, and 1−2) Turing Algorithms Demonstrated via DNA Self-Assembly

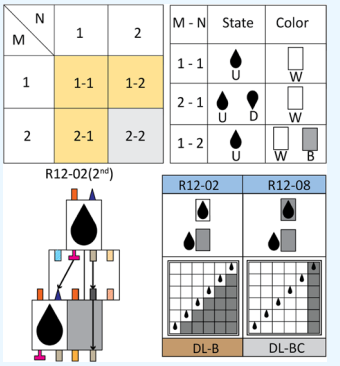
Muhammad Tayyab Raza and Sung Ha Park*

Cite This: *ACS Omega* 2023, 8, 15041−15051

Read Online

ACCESS | Metrics & More | Article Recommendations | Supporting Information

ABSTRACT: The fast and extensive generation of patterns using specific algorithms is a major challenge in the field of DNA algorithmic self-assembly. Turing machines (TMs) are simple computable machines that execute certain algorithms using carefully designed logic gates. We investigate Turing algorithms for the generation of patterns on algorithmic lattices using specific logic gates. Logic gates can be implemented into Turing building blocks. We discuss comprehensive methods for designing Turing building blocks to demonstrate an *M*-state and *N*-color Turing machine (*M*−*N* TM). The *M*-state and *N*-color (*M*−*N* = 1−1, 2−1, and 1−2) TMs generate Turing patterns that can be fabricated via DNA algorithmic self-assembly. The *M*−*N* TMs require two-input and three-output logic gates. We designed the head, tape, and transition rule tiles to demonstrate TMs for the 1−1, 2−1, and 1−2 Turing algorithms. By analyzing the characteristics of the Turing patterns, we classified them into two classes (DL and DR for states grown diagonally to the left and right, respectively) for the 1−1 TM, three for the 2−1 TM, and nine for the 1−2 TM. Among these, six representative Turing patterns generated using rules R11-0 and R11-1 for 1−1 TM, R21-01 and R21-09 for 2−1 TM, and R12-02 and R12-08 for 1−2 TM were constructed with DNA building blocks. Turing patterns on the DNA lattices were visualized by atomic force microscopy. The Turing patterns on the DNA lattices were similar to those simulated patterns. Implementing the Turing algorithms into DNA building blocks, as demonstrated via DNA algorithmic self-assembly, can be extended to a higher order of state and color to generate more complicated patterns, compute arithmetic operations, and solve mathematical functions.



INTRODUCTION

Implementing algorithms at the nanoscale level using proper chemistry requires an efficient computational model and a favorable environment to execute these computations.[1−3] Researchers have begun to analyze these computations based on the efficient programmability of materials in relatively shorter running times.[4−6] Various organic materials and biomaterials have been used for computations; however, among these materials, DNA is a prime candidate owing to the predictable and programmable Watson and Crick base pairing rule and relatively stable molecules.[6] Programmable DNA base sequences are used for computations, and the bit information implemented in a specific base sequence is transferred to the next DNA building block using base pair complementarity.[7,8]

Recently, researchers have started using DNA molecules to execute more efficient computations at the molecular level.[9,10] Using massive parallel computation, DNA can solve various types of mathematical, physical, and biological problems within a relatively short period of time, such as elementary functions, *M*-input and *N*-output logic circuits, arithmetic calculations, iterated Boolean circuits, neural networks, and random walks.[11−19] Algorithmic logic circuits that can be applied in the aforementioned models use a certain number of rules to generate specific patterns.[20,21] The outputs generated using these models can be visualized in the form of specific patterns and fluorescence by scanning probe microscopy and fluorescence spectroscopy, respectively.

In 1936, Turing proposed a computable machine that can execute simple logic circuits (known as a Turing machine (TM)).[22−25] A TM comprises a head and tape to generate certain patterns (Turing patterns). Turing machines provide the output information for the given input information of the head and tape. The information can be stored on the tape and executed by moving the head based on the instructions given by the TM. These instructions (Turing algorithms) provide an appropriate set of given rule algorithms with information on the state of the head [up (U) or down (D)], color of the tape [black (B) and white (W)], and position of the head [left (L) and right (R)]. A TM adopts a two-input, three-output logic gate to compute the Turing algorithm. A Turing algorithm consists of proper instructions of the head state and tape color (as a two-input set) and generates the head state, tape color, and head position (as a three-output set) in the next layer. A

Received: December 17, 2022
Accepted: April 3, 2023
Published: April 18, 2023

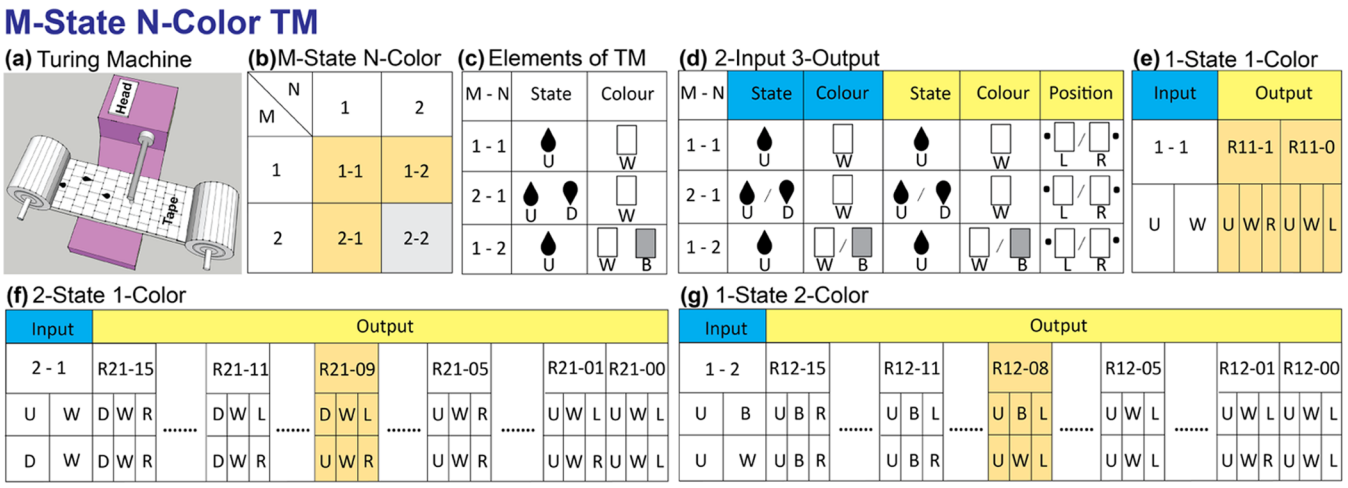


Figure 1. *M*-State and *N*-color Turing machine. (a) Schematic of a TM. A TM comprises a head and tape. (b) 1−1, 2−1, and 1−2 Turing algorithms in an $M \times N$ table. (c) Description of *M*-state and *N*-color TMs for 1−1, 2−1, and 1−2 Turing algorithms. (d) Components of 1−1, 2−1, and 1−2 Turing algorithms in a two-input (blue), three-output (yellow) logic implementation. (e) Two-input, three-output in a 1−1 Turing algorithm. Two rules (R11-0 and R11-1) are available. R11-0 (R11-1) provides the outputs of the UWL (UWR) with an input of UW. (f) Two-input, three-output in a 2−1 Turing algorithm. Sixteen rules are available. For instance, R21-09, which is highlighted in orange, provides the outputs of DWL (UWR) with the inputs of UW (DW). (g) Two-input, three-output in a 1−2 Turing algorithm. Sixteen rules are available. For instance, R12-08, which is highlighted in orange, provides the outputs of UBL (UWL) with the inputs of UB (UW).

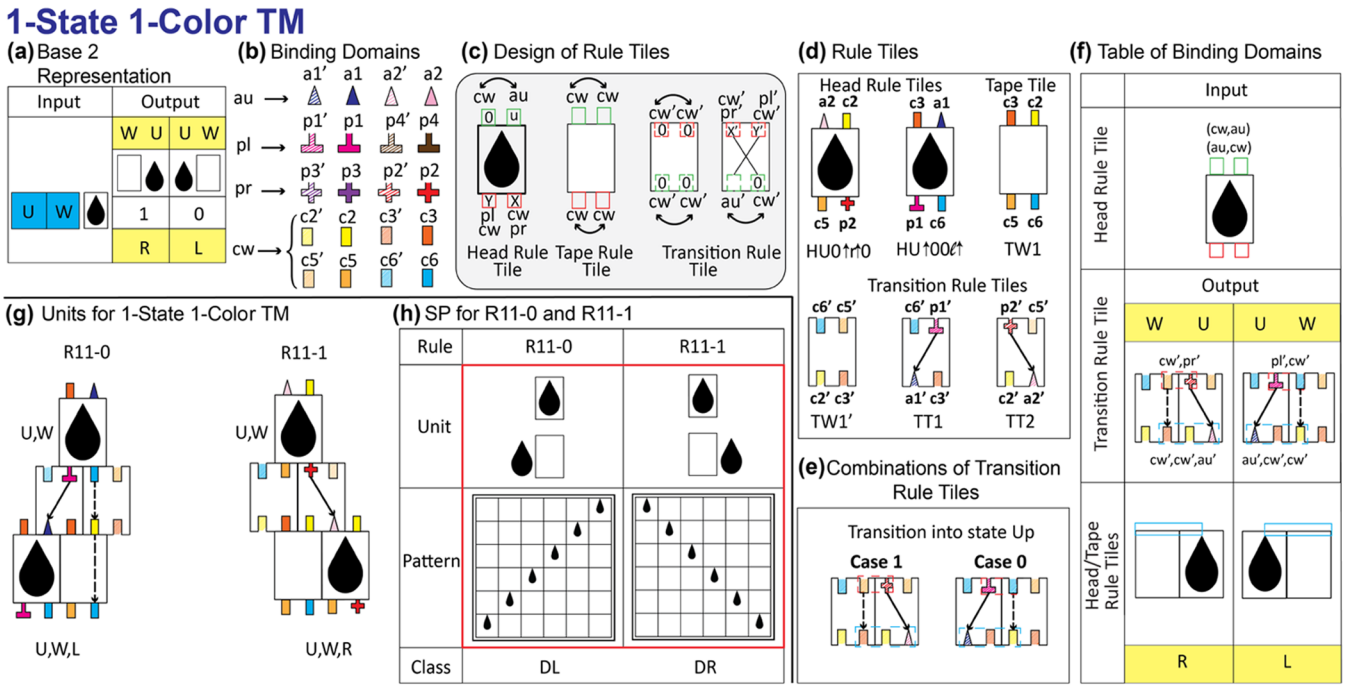


Figure 2. Design of binding domains in abstract building blocks for a 1−1 Turing algorithm. (a) Two-input, three-output with a base 2 representation in a 1−1 Turing algorithm. One set (UW) of a two-input and two sets (UWL and UWR) of a three-output are highlighted in blue and yellow, respectively. (b) Binding domains of the state alignment (a and u denote alignment and up, respectively), position specification (p, l, and r denote position, left, and right, respectively), and color allocation (c and w denote color and white, respectively) with the corresponding geometrical representations. (c) Schematics of the designs of the head, tape, and transition rule tiles. Based on the given input-binding domains, one type of head, one type of tape, and two types of transition rule tiles were designed. (d) Two head, one tape, and three transition rule tiles with specific binding domains. (e) Two possible combinations of transition rule tiles. (f) Table of binding domains for head, tape, and transition rule tiles. Two input-binding domains (cw′ and pr′) in a transition rule tile marked with a dotted red box are complementary to the outputs of the head rule tile marked with red boxes. Similarly, three output-binding domains (cw′, cw′, and au′) in a transition rule tile marked with a dotted cyan line are complementarity to the inputs of the head and tape rule tiles marked with cyan boxes. (g) Units for R11-0 and R11-1. (h) Simulated patterns (SP) for R11-0 (states grown diagonally to the left (DL)) and R11-1 (states grown diagonally to the right (DR)).

Turing algorithm with *M* states and *N* colors (*M*-state, *N*-color) can generate various Turing patterns operated by the Turing rules. The total number of available rules for a given *M*-state, *N*-color TM (*M*−*N* TM), is obtained by $(2 \times M \times N)^{M \cdot N}$. For instance, a 1−2 TM generates 16 different patterns by using $16 = (2 \times 1 \times 2)^{1 \cdot 2}$ available rules.

https://doi.org/10.1021/acsomega.2c08017
ACS Omega 2023, 8, 15041−15051

Here, we discuss the 1−1, 2−1, and 1−2 TMs demonstrated by the two-input, three-output algorithmic self-assembly of DNA. Three different rule tiles, head, tape, and transition rule tiles, were conceived to generate simulated (abstract rectangular building blocks containing proper binding domains) and experimentally obtained (DNA tiles with sticky ends) Turing patterns. By analyzing the characteristics of the simulated Turing patterns, the 1−1, 2−1, and 1−2 TMs are classified into two (DL and DR for states grown diagonally to the left and to the right, respectively), three, and nine classes, respectively (as discussed in the Results and Discussion section). Turing patterns on the DNA lattices were visualized by atomic force microscopy (AFM).

RESULTS AND DISCUSSION

Description of an *M*-State *N*-Color Turing Machine. The 1−1, 2−1, and 1−2 TMs were implemented into two-input and three-output logic gates (Figure 1). Figure 1a shows a Turing machine comprising a head and rolling tape with the symbols of the head state. Figure 1b shows an $M \times N$ table with the 1−1, 2−1, and 1−2 Turing algorithms. The elements of an M−N TM are the head states (U and D) and tape colors (B and W) (Figure 1c). The 1−1 TM consists of head state U and tape color W, the 2−1 TM consists of head state U or D and tape color W, and the 1−2 TM consists of head state U and tape color W or B.

Figure 1d−g shows the components of the 1−1, 2−1, and 1−2 Turing algorithms in two-input (indicated by blue), three-output (yellow) logic implementations. The number of available rules can be determined by arranging the components of each Turing algorithm. For instance, the 1−1 TM containing one two-input set with two three-output sets yields two distinct Turing patterns (i.e., R11-0 and R11-1). For the 2−1 and 1−2 TMs, 16 rules (R21-00−R21-15 for the 2−1 TM and R12-00−R12-15 for the 1−2 TM) are available in each two-input set.

The total number of available rules for each two-input set is obtained by $(2 \times M \times N)^{M \cdot N}$. For instance, the 1−1, 2−1, and 1−2 TMs generate 2 $(=(2 \times 1 \times 1)^{1 \cdot 1})$, 16 $(=(2 \times 2 \times 1)^{2 \cdot 1})$, and 16 $(=(2 \times 1 \times 2)^{1 \cdot 2})$ different rules per two-input set. Notably, two different Turing patterns are available at a given rule for the 2−1 and 1−2 TMs because of the two two-input sets. For example, R21-09 revealed a three-output set of DWL (UWR) with a two-input set of UW [DW]. We named this rule R21-09(1st) [R21-09(2nd)]. Similarly, R12-08(1st) and R12-08(2nd) show two three-output sets of UBL and UWL generated using the two two-input sets of UB and UW, respectively.

1−1 Turing Algorithm in Base 2 Demonstrated Using Abstract Rule Tiles. Figure 2a,b shows the 1−1 Turing algorithm in the base 2 representation and binding domains with geometrical shapes. For a given single-input UW (in blue), two available output sets are labeled, i.e., UWL and UWR (in yellow). Here, the base 2 representation of the 1−1 TM exhibits (U, W) → (U, W, L) and (U, W) → (U, W, R) for 0 (i.e., 0_2) and 1 (i.e., 1_2), respectively. These correspond to rules R11-0 and R11-1, respectively (Figure 2a). Figure 2b and Table S1 in the Supporting Information show geometrical representations of the binding domains for the rule tiles. The information delivery of the state and color in the 1−1 TM requires specific rule tiles that have binding domains containing specific input and output characteristics, such as state alignment up (au), state position specification (pl and pr for left and right positions, respectively), and tape color allocation (cw for white). For the alignment up (↑) state, two specific bindings (a1 and a2) were conceived with the corresponding complementary binding domains (a1′ and a2′). For state position specification, two specific bindings (p1 and p4 for the left position and p2 and p3 for the right position) were introduced with the corresponding complementary binding domains (p1′ and p4′ for the left position and p2′ and p3′ for the right position). Finally, four binding domains (c2, c3, c5, and c6) were introduced to allocate white color with the corresponding complementary binding domains (c2′, c3′, c5′, and c6′).

The 1−1 Turing algorithm with four binding domain-embedded rectangular building blocks is demonstrated by designing three rule tiles with the specific characteristics of the head state and tape color (Figure 2c). A head rule tile with head state alignment up information comprises two input-binding domains (which carry information of the head state alignment up marked with U and tape color white marked with 0 in green) and two output-binding domains (which carry information of the head state position and tape color in red). Two input-binding domains, i.e., au and cw, can exchange their locations (indicated by a reversible solid arrow). Similarly, two output-binding domains, i.e., pl and cw (cw and pr), can exchange their locations in Y and X (Figure 2c). The tape rule tiles contain information about the tape color white, which corresponds to the 0-bit information. The tape rule tile was designed based on the color specification of the binding domains. Two input-binding domains, both containing the 0-bit information (white tape rule tiles), copy the same bit information to the corresponding output-binding domains, i.e., (cw, cw → cw, cw). The binding domains of the transition rule tiles in the input sets contain information on the tape color and head state position obtained from the head rule tile placed in the previous tape. The output sets in the binding domains contain information on the head alignment and tape color, which is delivered to the head and tape rule tiles in the next layer. Two types of transition rule tiles were designed: one copies the tape color, e.g., cw′, cw′ → cw′, cw′, and the other delivers the head state position to the head alignment (pl′ → au′ and pr′ → au′).

Figure 2d shows two head, one tape, and three transition rule tiles with specific input- and output-binding domains with geometrical shapes. Input (output)-binding domains in each head rule tile carry the head state alignment (position) and tape color. For instance, the inputs (outputs) of a1 and c2 (c5 and p2) indicate the head state alignment U and tape color W (head state position R and tape color W), respectively. Hence, each head rule tile can be formulated using two output combinations. Figure 2d also shows the tape rule tile, with detailed information on the binding domains. Inputs (c3 and c2) both with W (possessing the 0-bit information) in tape rule tile TW1 deliver the same bit information to the corresponding output-binding domains (c5 and c6). Figure 2d,e shows three individual and two possible combinations of the transition rule tiles, respectively. Transition rule tiles have two types: one copies the tape color, such as TW1′, and the other delivers the state position to the head alignment, such as TT1 and TT2. For instance, TT1 delivers the state position left (p1′) in the input to the state alignment (a1′) in the output. Two combinations (i.e., cases 0 and 1) comprising two transition rule tiles were designed to deliver information acquired from the head rule tile to the head and tape rule tiles.

https://doi.org/10.1021/acsomega.2c08017
ACS Omega 2023, 8, 15041−15051

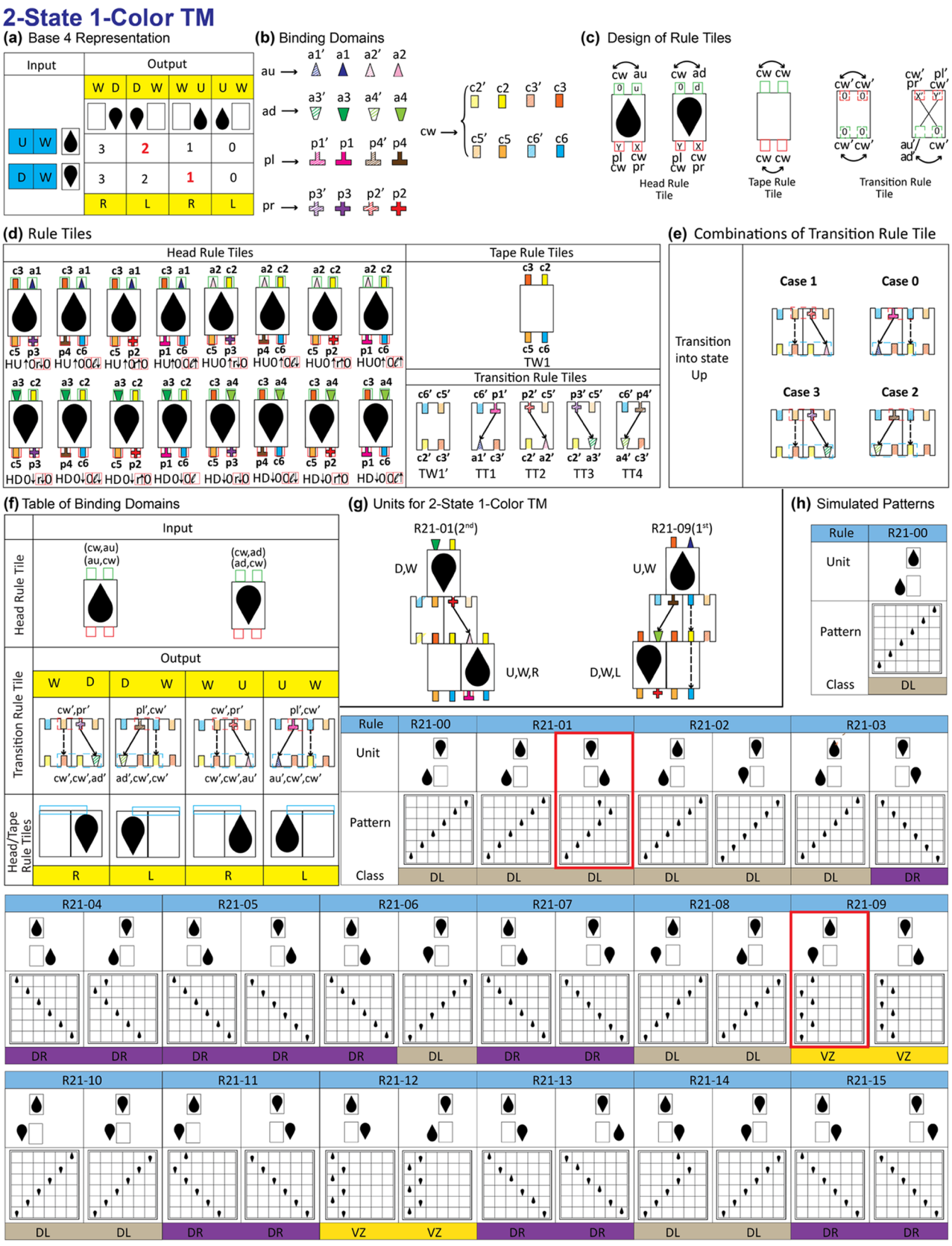


Figure 3. Design of binding domains in abstract building blocks for a 2−1 Turing algorithm. (a) Two-input, three-output with a base 4 representation in a 2−1 Turing algorithm. Two sets (UW and DW) of a two-input and four sets (UWL, UWR, DWL, and DWR) of a three-output are highlighted in blue and yellow, respectively. (b) Binding domains of the state alignment (au and ad), position specification (pl and pr), and color allocation (cw) with the corresponding geometrical representations. (c) Schematics of the design of head, tape, and transition rule tiles. Based on the given input-binding domains, two types of head, one type of tape, and two types of transition rule tiles were designed. (d) Sixteen head, one tape, and five transition rule tiles with specific binding domains. (e) Four possible combinations of transition rule tiles. (f) Table of binding

https://doi.org/10.1021/acsomega.2c08017
ACS Omega 2023, 8, 15041−15051

Figure 3. continued

domains for the head, tape, and transition rule tiles. (g) Units for R21-01 with the two-input of DW (R21-01(2nd)) and R21-09 with the two-input of UW (R21-09(1st)). Each rule has two sets (1st UW and 2nd DW) of the two-input. (h) Simulated patterns (SP) with classification for a 2−1 Turing algorithm. Sixteen rules (each rule produces two Turing patterns) and three pattern classes (DL, DR, and VZ, which indicate states grown diagonally to the left, diagonally to the right, and vertically in a zigzag manner, respectively) are available. The simulated patterns in the red boxes were obtained experimentally.

Case 0 (consisting of TT1 and TW1′) delivers two sets of information: (i) head state position left to head state alignment up and (ii) tape color white to tape color white. Similarly, case 1 (consisting of TW1′ and TT2) delivers information on (i) tape color white to tape color white and (ii) head state position right to head state alignment up. The dashed rectangular red box (cyan) on each combination of transition rule tiles indicates the binding sites of a head rule tile (head and tape) through the complementary binding between unprimed and primed binding domains.

Figure 2f shows the rule tile bindings for constructing the units for all rules (R11-0 and R11-1) available in the 1−1 Turing algorithm. Two input-binding domains, (pl′, cw′) and (cw′, pr′), in a set of transition rule tiles marked with a dotted red box are complementary to the outputs of the head rule tile marked with solid red. Similarly, three output-binding domains, (au′, cw′, cw′) and (cw′, cw′, au′), in a set of transition rule tiles marked with a dotted cyan box are complementary to the inputs of the head and tape rule tiles marked with cyan solid boxes. Figure 2g shows two unit sets comprising the head, tape, and transition rule tiles for R11-0: (U, W) → (U, W, L) and R11-1: (U, W) → (U, W, R).

Figure 2h shows the simulated patterns (SP) (which are generated by Mathematica 11) for R11-0 and R11-1. Based on the pattern characteristics, we classify SPs into two classes: (i) head states grown diagonally to the left (DL) and (ii) head states grown diagonally to the right (DR).

2−1 Turing Algorithm in Base 4 Demonstrated Using Abstract Rule Tiles. Figure 3a shows the two-input and three-output of the 2−1 Turing algorithm in the base 4 representation. For the two given sets of two-input, UW and DW (in blue), four available output sets are labeled, i.e., UWL, UWR, DWL, and DWR (in yellow). Using these, 16 ($=4^2$) distinct rules (R21-00−R21-15) can be specified. Here, the rule number in the rule name corresponds to the number in base 4. For instance, 09 ($=2 \times 4^1 + 1 \times 4^0$) in R21-09 corresponds to 21_4 (marked in red in Table), which provides the three-outputs of the DWL for the two-inputs of the UW (1st input) and UWR for the DW (2nd input). The 2−1 Turing algorithm requires specific rule tiles to deliver the information of two states and one color using binding domains (Figure 3b). The rule tiles contain specific input and output characteristics, such as state alignment (au and ad), state position (pl and pr), and tape color (cw).

Figure 3c shows the schematics of the binding domain design of the head, tape, and transition rule tiles. Two types of head rule tiles are required for the two states. A head rule tile with the head state up information comprises two input-binding domains, which contain the information on the state alignment up (u) and tape color white (0) in green, and two output-binding domains, which contain the information on the state position and tape color in red. For the head state down, the head rule tile has two input-binding domains, which contain information on the state alignment down (d) and tape color white (0). The two input-binding domains in both head rule tiles, i.e., au/ad and cw, can exchange their locations (indicated by the reversible solid arrows). Similarly, two output-binding domains, i.e., pl and cw (cw and pr), can exchange their locations in *Y* and *X*. One type of tape rule tile is required. A tape rule tile delivers the tape color white from the input to the output-binding domains. Two types of transition rule tiles were designed: one copies the tape color (such as cw′, cw′ → cw′, cw′) and another delivers the state position to the head alignment (pl′ → au′/ad′ and pr′ → au′/ad′).

Figure 3d shows 16 head, one tape, and five transition rule tiles with specific input and output-binding domains with geometrical shapes. A head rule tile contains the head state (either HU or HD), the input-binding domains of the head alignment (↑ or ↓), the tape color (0), the output-binding domains of the tape color (0), and the head position/state alignment (l↑, l↓, r↑, r↓). The input (output)-binding domains with the state alignment and tape color (state position with the information on the state alignment and tape color) for each head state are specified; in particular, for inputs a1 and a2 (a3 and a4) for u (d) and c2 and c3 for 0, and for outputs p1 and p4 (p2 and p4) for l↑ [r↑] and c5 and c6 for 0. Hence, each head rule tile can be formulated using two output combinations. For instance, HU↑0r↓0 contains the input information on the state alignment up (↑) in a1 and the color white (0) in c3, as well as the output information on the head position right/state alignment down (r↓) in p3 and the tape color white (0) in c5. Similarly, HD0↓r↓0 contains the input information of the tape color white (0) in c2 and the state alignment down (↓) in a3, as well as the output information of the head position right/state alignment down/(r↓) in p3 and the tape color white (0) in c5.

Because of the 1-color TM, a single tape rule tile TW1 is required (which is the same as the tape rule tile for the 1−1 TM). Figure 3d,e shows five individual transition rule tiles and four possible cases (formed with two transition rule tiles), respectively. For the head state up (down) in the following layer, two cases are required, such as (TW1′, TT1) and (TT2, TW1′) ((TW1′, TT4) and (TT3, TW1′)), labeled as cases 0 and 1 (cases 2 and 3), respectively. For instance, case 3, which comprises TT3 and TW1,′ delivers the following information: (i) state position right to state alignment down (indicated by the solid arrow) and (ii) tape color white to tape color white (dotted arrow). Input information received from the head rule tiles and output information delivered to the head/tape rule tiles are indicated by the dotted red and cyan lines, respectively.

Figure 3f shows a list of the rule tile-binding domains for constructing the units for all rules (R21-00 to R21-15) available in the 2−1 Turing algorithm. Two input-binding domains, such as (pl′, cw′) and (cw′, pr′), in the four sets of transition rule tiles are complementary to the outputs of the head rule tile (indicated by two red boxes). Similarly, three output-binding domains, such as (au′/ad′, cw′, cw′) and (cw′, cw′, au′/ad′), in a set of transition rule tiles, marked with

https://doi.org/10.1021/acsomega.2c08017
ACS Omega 2023, 8, 15041−15051

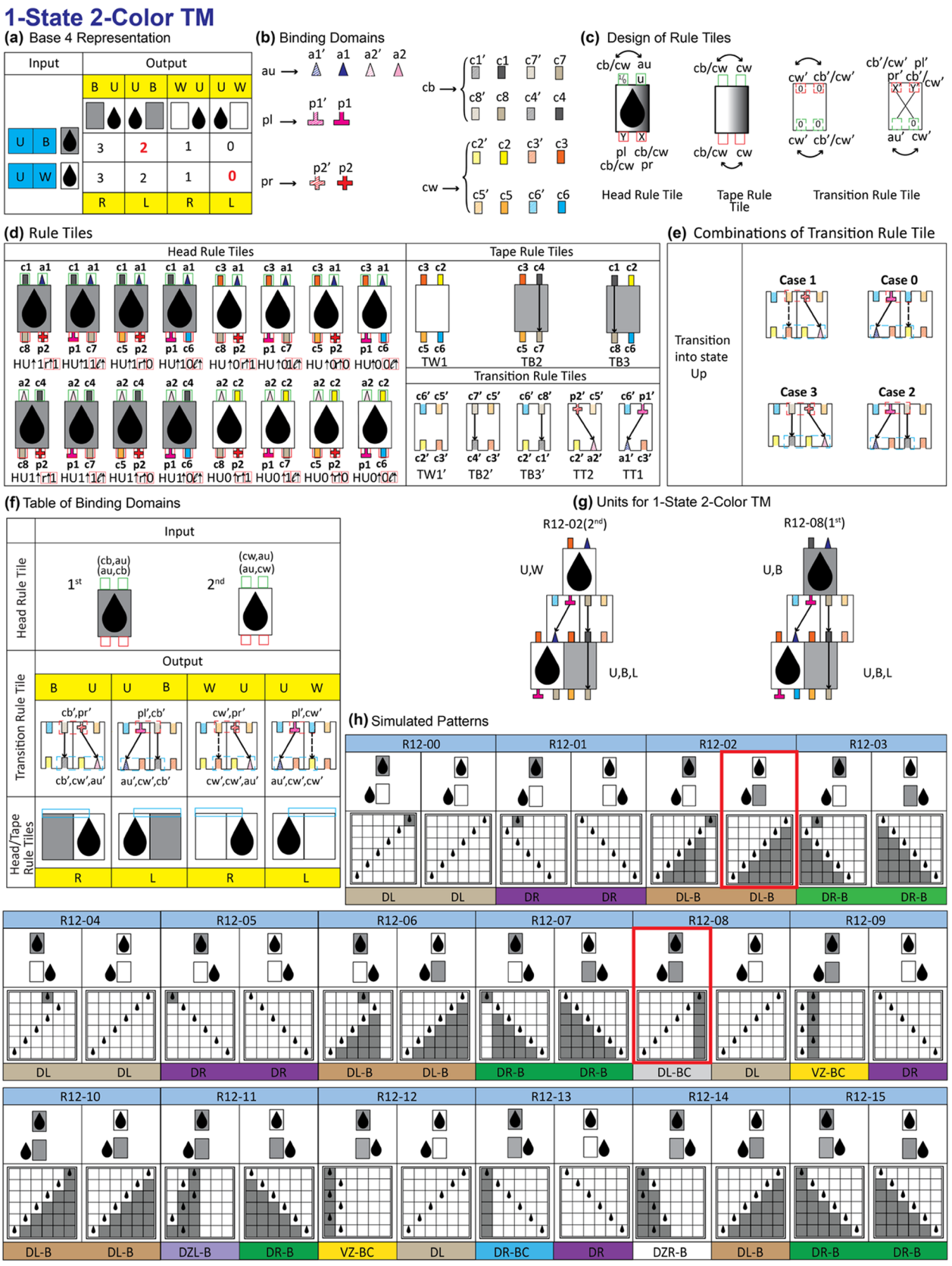


Figure 4. Design of binding domains in abstract building blocks for a 1−2 Turing algorithm. (a) Two-input, three-output with a base 4 representation in a 1−2 Turing algorithm. Two sets (UB and UW) of a two-input and four sets (UWL, UWR, UBL, and UBR) of a three-output are highlighted in blue and yellow, respectively. (b) Binding domains of the state alignment (au), position specification (pl and pr), and color allocation (cb and cw) with the corresponding geometrical representations. (c) Schematics of the design of head, tape, and transition rule tiles. Based on given input-binding domains, two types of head, tape, and transition rule tiles were designed. (d) Sixteen head, three tape, and five transition rule tiles with specific binding domains. (e) Four possible combinations of transition rule tiles. (f) Table of binding domains for head,

https://doi.org/10.1021/acsomega.2c08017
ACS Omega 2023, 8, 15041−15051

Figure 4. continued

tape, and transition rule tiles. (g) Units for R12-02 with the input of UW (R12-02(2nd)) and R12-08 with the two-input of UB (R12-08(1st)). Each rule has two sets (1st UB and 2nd UW) of the two-input. (h) Simulated patterns (SP) with classification for a 1−2 Turing algorithm. Sixteen rules (each rule produces two Turing patterns) and nine pattern classes (such as DL-B, which indicates states grown diagonally to the left and black grown in a triangular shape, and DL-BC, which indicates states grown diagonally to the left and black grown in a line-like shape) are available. The SP in the red boxes was obtained experimentally.

dotted cyan boxes, are complementary to the inputs of the head and tape rule tiles (marked with cyan solid boxes). The units for R21-01(2nd) and R21-09(1st) comprised the head, tape, and transition rule tiles. Every rule in the 2−1 TM has two input sets: UW and DW. For instance, R21-01(2nd) represents the unit initiated by the second input set of DW, and R21-09(1st) shows the unit initiated by the first input set of UW (Figure 3g).

Simulated patterns generated from 16 rules (from R21-00 to R21-15) with two different input sets (i.e., UW and DW) are classified into three different classes (Figure 3h). Classes DL and DR indicate the diagonal left and right movements of the head rule tiles in the simulated patterns, respectively. In contrast, class VZ indicates vertical and zigzag movements of the head rule tiles in a simulated pattern. Although all patterns (except R21-06) generated with the same rules with different input sets show the same type of classes, R21-06 reveals two classes: DR for R21-06(1st) and DL for R21-06(2nd).

1−2 Turing Algorithm in Base 4 Demonstrated Using Abstract Rule Tiles. Figure 4a shows the 1−2 Turing algorithm in a base 4 representation. For the two given input sets UB (1st input) and UW (2nd input) (in blue), four available output sets are labeled, i.e., UWL, UWR, UBL, and UBR (in yellow). Using these, 16 distinct rules ($=4^2$; R12-00−R12-15) can be generated. Here, the rule number in the rule name corresponds to the number in base 4. For instance, 08 ($=2 \times 4^1 + 0 \times 4^0$) in R21-08 corresponds to 20_4 (marked in red in Table), which provides the three-outputs of the UBL for the two-inputs of the UB (1st input) and UWL for the UW (2nd input). The 1−2 Turing algorithm requires specific rule tiles to deliver information on one state and two colors through carefully designed binding domains (Figure 4b). The rule tiles contain specific input and output characteristics, such as the state alignment (such as au), state position (pl and pr), and tape color (cw and cb).

Figure 4c shows schematics of the binding domain design of the head, tape, and transition rule tiles. One type of head rule tile is required for one state. A head rule tile with the head state up information comprises two input-binding domains, which contain the information on the state alignment up (u) and tape color, either white or black (0 or 1), in green, and two output-binding domains, which contain the information on the state position and tape color in red. Two input-binding domains in both head rule tiles, i.e., au and cb/cw, can exchange their locations (indicated by the reversible solid arrows). Similarly, two output-binding domains, i.e., pl and cb/cw (cb/cw and pr), can exchange their locations in *Y* and *X*. The two types of tape rule tiles are required. Tape rule tiles deliver the tape color white/black from the input to the output-binding domains. Two types of transition rule tiles were designed: one copies the tape color, such as (cb′/cw′, cw′ → cb′/cw′, cw′), and another delivers the state position to the head alignment (pl′ → au′ and pr′ → au′).

Figure 4d shows the 16 head, three tape, and five transition rule tiles with specific input- and output-binding domains with geometrical shapes. A head rule tile name contains the head state (HU), the input-binding domains of the head alignment (↑), the tape color (either 0 or 1), the output-binding domains of the tape color (0 or 1), and the head position/state alignment (l↑ and r↑). The input (output)-binding domains containing the state alignment and tape color (state position with the information on the state alignment and tape color) are specified for each head state; in particular, for inputs a1 and a2 for u, c2 and c3 for 0, and c1 and c4 for 1, and for outputs p1 for l↑ and p2 for r↑, c5 and c6 for 0, and c7 and c8 for 1. Hence, each head rule tile can be formulated using two output combinations. For instance, HU↑1r↑1 contains the input information on the state alignment up (↑) in a1 and the color black (1) in c1, as well as the output information on the head position right/state alignment up (r↑) in p2 and the tape color black (1) in c8. Similarly, HU0↑0l↑ contains the input information on the tape color white (0) in c2 and the state alignment up (↑) in a2, as well as the output information on the head position left/state alignment up/(l↑) in p1 and the tape color white (0) in c6.

Figure 4d shows schematics of the tape rule tiles with detailed information on the binding domains. The tape rule tile of TW1 containing two input-binding domains (c2 and c3) with the 0-bit information (tape color white) copies the 0-bit information to the output (c6 and c5). In contrast, TB2 and TB3 containing one input-binding domain with the 1-bit information (c4 and c1) copy the 1-bit information to the output (c7 and c8) in each tile, as indicated by the arrows. Figure 4d,e shows five individual transition rule tiles and four possible cases (formed with two transition rule tiles), respectively. Four combinations of two transition rule tiles are needed: (TW1′, TT1), (TT2, TW1′), (TB2′, TT1), and (TB3′, TT2) labeled as cases 0, 1, 2, and 3, respectively. For instance, case 1, which comprises TT2 and TW1′, delivers the following information: (i) state position right to state alignment up (indicated by the solid arrow) and (ii) tape color white to tape color white (dotted arrow). The input information received from the head rule tiles and the output information delivered to the head/tape rule tiles are indicated by the dotted red and cyan lines, respectively.

Figure 4f shows a list of the rule tile-binding domains for constructing the units for all rules (R12-00 to R12-15) available in the 1−2 Turing algorithm. Two input-binding domains, such as (pl′, cb′/cw′) and (cb′/cw′, pr′), in the four sets of transition rule tiles are complementary to the outputs of the head rule tile (indicated with two red boxes). Similarly, three output-binding domains, such as (au′, cw′, cb′/cw′) and (cb′/cw′, cw′, au′), in the set of transition rule tiles marked with dotted cyan boxes are complementary to the inputs of the head and tape rule tiles (marked with cyan solid boxes). The units for R12-02(2nd) and R12-08(1st) comprise the head, tape, and transition rule tiles. Every rule in the 1−2 TM has two input sets: UB and UW. For instance, R12-02(2nd) represents the unit initiated by the second input set of UW,

https://doi.org/10.1021/acsomega.2c08017
ACS Omega 2023, 8, 15041−15051

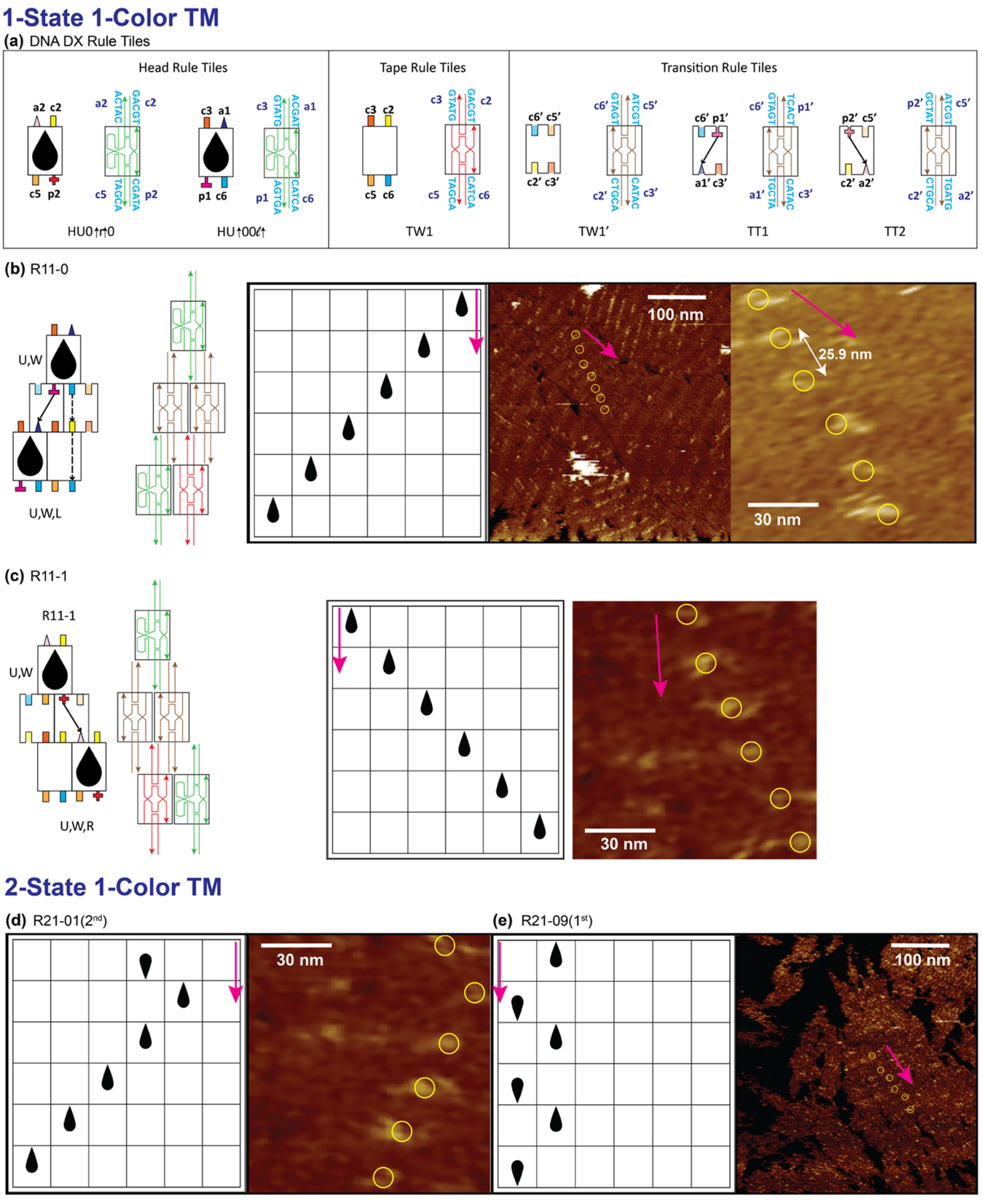


Figure 5. Experimental demonstration of the 1−1 (i.e., R11-0, R11-1) and 2−1 (e.g., R21-01(2nd), R21-09(1st)) Turing patterns via the algorithmic self-assembly of DNA implemented with two-input, three-output logic rules. (a) Head (green), tape (red), and transition (brown) rule tiles represented by DNA DX tiles with corresponding abstract building blocks. (b) Unit with the two-input of UW and AFM images of Turing patterns operated by rule R11-0. (c) Unit, the simulation pattern, and the represented AFM image of Turing patterns operated by rule R11-1. (d, e) Simulation patterns with the two-inputs of DW (i.e., R21-01(2nd)) and UW (R21-09(1st)) with the corresponding AFM images. Magenta arrows in simulation patterns and AFM images indicate the growth direction of the patterns.

https://doi.org/10.1021/acsomega.2c08017
ACS Omega 2023, 8, 15041−15051

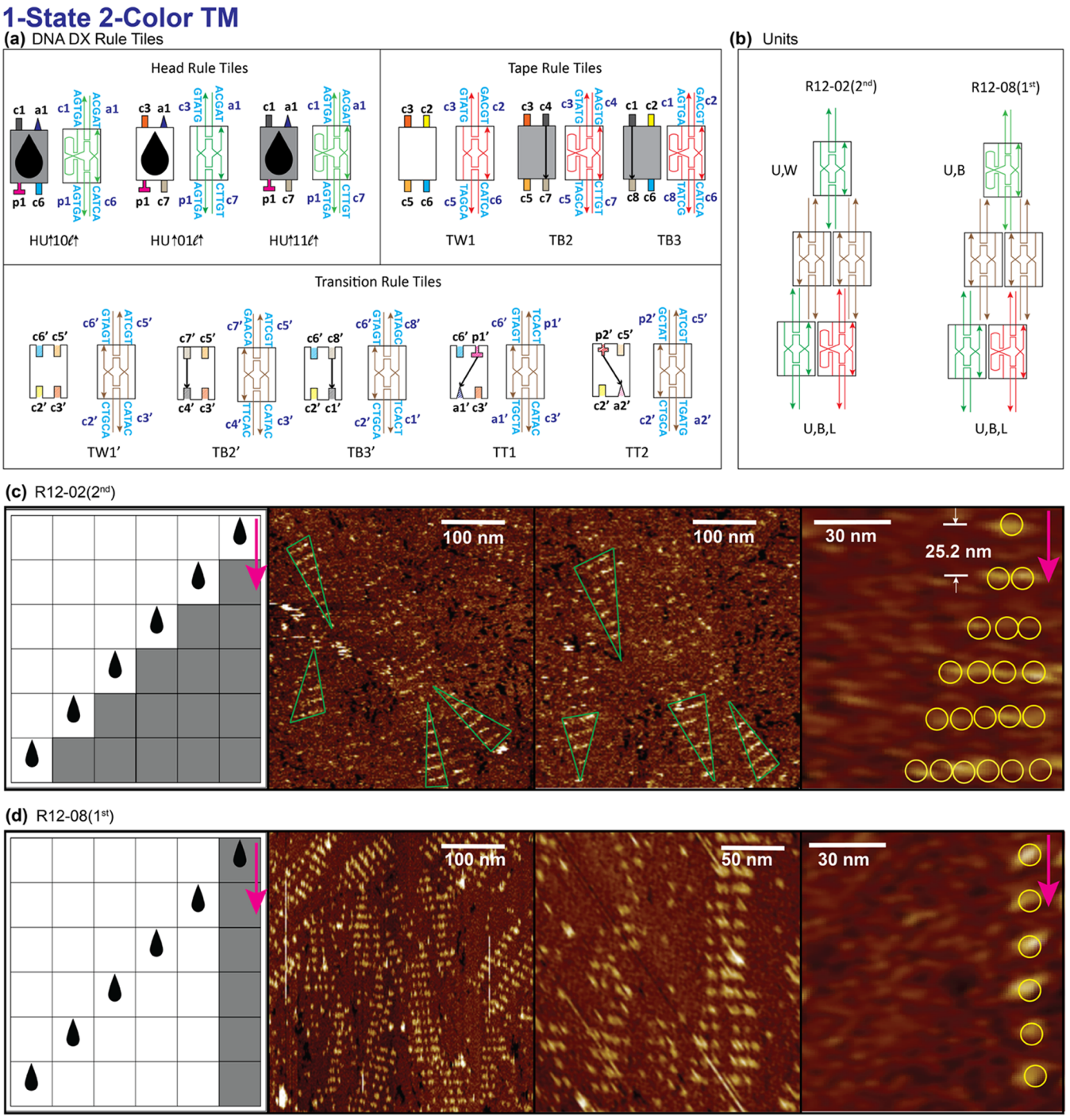


Figure 6. Experimental demonstration of the 1−2 Turing patterns via the algorithmic self-assembly of DNA implemented with two-input, three-output logic rules (e.g., R12-02(2nd) and R12-08(1st)). (a) Head, tape, and transition rule tiles represented by abstract building blocks with the corresponding DNA DX tiles. (b) Units with the two-inputs of UW and UB operated by R12-02(2nd) and R12-08(1st). (c) Simulation pattern and represented AFM images of Turing patterns operated by rule R12-02(2nd). Triangular patterns formed by hairpin-embedded DX tiles are marked with green triangles. (d) Simulation pattern and represented AFM images of Turing patterns operated by rule R12-08(1st).

and R12-08(1st) shows the unit initiated by the first input set of UB (Figure 4g).

Using the characteristics of the simulated patterns, we classified them into nine classes (Figure 4h). The class name contains information on (i) the growth direction of the head state (DL/DR, VZ, DZL/DZR) and (ii) the configuration of the black tape (B, BC). Here, DL/DR indicates head states grown diagonally to the left/right, VZ indicates head states grown vertically in a zigzag manner, and DZL/DZR represents head states grown diagonally in a zigzag manner to the left/right. The configurations of black tapes B and BC indicate black growing in a triangular and a line-like shape, respectively. For instance, DL-B indicates states grown diagonally to the left and black grown in a triangular shape, and DL-BC indicates states grown diagonally to the left and black grown in a line-like shape, respectively. Although the simulated patterns are generated with the same rules, some of the rules show different

https://doi.org/10.1021/acsomega.2c08017
ACS Omega 2023, 8, 15041−15051

pattern class patterns with different input sets (e.g., VZ-BC obtained from UB and DR from UW for R12-09).

Experimental Demonstration of 1−1 and 2−1 Turing Algorithms Using DNA DX Tiles. For the experimental demonstration of the 1−1 and 2−1 TMs, rectangular DNA double-crossover (DX) tiles were used (Figure 5a). A DNA DX tile comprises two parallel duplexes connected with two crossover junctions with dimensions of 12.6 nm × 4.0 nm (length × width). To visualize Turing patterns on DNA lattices, one-full-turn protruding DNA hairpins were decorated on head rule tiles (which contain the information on head state up for the 1−1 TM and head state up or down for the 2−1 TM). For instance, two hairpin structures were observed in the DNA DX representation of head rule tiles (such as HU0↑r↑0 and HU↑00*l*↑ for the 1−1 TM). In addition, five-nucleotide-length sticky-end domains with the corresponding binding domains are indicated in the DNA DX representation (Table S2 in the Supporting Information). In addition, Figure 5a shows DNA DX representations with the corresponding abstract building blocks of tape and transition rule tiles (which do not possess DNA hairpin structures) (also see Figures S1−S3, Tables S3, and S4 in the Supporting Information).

Figure 5b shows a unit (with the two-input of the UW and three-output of the UWL), a simulation pattern, and represented AFM images of DNA lattices operated by rule R11-0. The unit for all rules (such as R11-0 and R21-01) consists of five rule tiles (comprising three layers: an input head rule tile in the first layer, two transition rule tiles in the second, and output head and tape rule tiles in the third). An abstract representation of the unit indicates information on the head alignment, tape color, and head state position, and a DNA DX representation of the unit provides the type of rule tile by color (i.e., head with green, tape with red, and transition with brown). Head rule tiles with protruding hairpins in the DNA lattices in the AFM images are marked with yellow circles to visualize the Turing patterns. We noticed that the experimentally observed Turing patterns for R11-0 (i.e., head states grown diagonally to the left) are in good agreement with the simulation patterns. The average distance between the nearest hairpins in the AFM images was ∼25.9 nm (as indicated by AFM). Figure 5c shows a unit (with the two-input of the UW and three-output of UWR), a simulation pattern, and represented AFM images of the DNA lattices operated by rule R11-1. Similar to R11-0, the experimentally observed Turing patterns for R11-1 (i.e., head states grown diagonally to the right) are in good agreement with the simulation patterns.

For the 2−1 TM, two rules, R21-01(2nd) and R21-09(1st), were demonstrated via DNA DX tiles (Figure 5d,e). The units of R21-01 initiated with the second two-input of the DW, and R21-09 initiated with the first two-input of the UW, as shown in Figure 3g. The AFM images for R21-01(2nd) (head states grown diagonally to the right) and R21-09(1st) (head states grown vertically in a zigzag manner) were similar to the simulation patterns.

Experimental Demonstration of the 1−2 Turing Algorithm Using DNA DX Tiles. Figure 6a shows the 1−2 TM, the head, tape, and transition rule tiles represented by the DNA DX tiles and abstract building blocks. To visualize the Turing patterns on the DNA lattices, protruding one-full-turn DNA hairpins were decorated on the head and tape rule tiles with black tape. Consequently, hairpin-decorated DNA DX tiles have input-binding domains in the head and tape rule tiles, either c1 or c4. For instance, hairpin structures in the DNA DX representation of the head rule tiles (such as HU↑10*l*↑ and HU↑11*l*↑) and tape rule tiles (such as TB2 and TB3) were determined. In addition, five-nucleotide-length sticky-end domains with corresponding binding domains were indicated by the DNA DX tiles. Units operated by two rules, R12-02(2nd) and R12-08(1st), were designed to demonstrate Turing patterns experimentally. Figure 6b shows the units of R12-02 initiated with the second two-input of UW and R12-08 initiated with the first two-input of UB.

The simulation patterns in Figure 6c show head states grown diagonally to the left and black grown in a triangle shape (DL-B) for R12-02(2nd), and Figure 6d shows head states grown diagonally to the left and black grown in a line-like shape (DL-BC) for R12-08(1st). The DNA patterns exhibit black grown in a triangular shape for R12-02(2nd) and black grown in a line-like shape for R12-08(1st) because of hairpins placed in the head and tape rule tiles containing information of black tape not placed in the presence of the head states. We noticed various sizes of triangles (marked with green triangles) in AFM images for R12-02(2nd) and lines in AFM images for R12-08(1st), which were similar to the simulation patterns.

CONCLUSIONS

In conclusion, we introduced the 1−1, 2−1, and 1−2 Turing algorithms and demonstrated them using the two-input, three-output algorithmic self-assembly of DNA. *M*−*N* TMs via the algorithmic assembly of DNA were developed by designing three types of rule tiles: head, tape, and transition. The 1−1, 2−1, and 1−2 TM algorithmic self-assemblies were demonstrated by experiments on DNA lattices and simulations. We discussed six representative Turing patterns that were generated using rules R11-0 and R11-1 for the 1−1 TM, R21-01 and R21-09 for the 2−1 TM, and R12-02 and R12-08 for the 1−2 TM. By analyzing the characteristics of the Turing patterns, the 1−1, 2−1, and 1−2 TMs were classified into two (DL and DR), three (DL, DR, and VZ), and nine classes, respectively. We noticed that the experimentally obtained Turing patterns in the AFM images were in good agreement with the simulation data.

METHODS

DNA Lattice Annealing. High-performance liquid chromatography (HPLC)-purified synthetic oligonucleotides (Integrated DNA Technologies, Iowa) were used. DNA lattices containing the Turing patterns were constructed using a two-step annealing method. In the first step, individual DX tiles (i.e., head, tape, and transition rule tiles) were formed by mixing a stoichiometric quantity of each strand in a 1× TAE/Mg^{2+} buffer (Tris-acetate-EDTA: 40 mM Tris, 1 mM EDTA (pH 8.0), 12.5 mM of magnesium acetate). Test tubes containing DNA samples were placed in a Styrofoam box with 2 L of boiled water and gradually cooled from 95 to 25 °C to facilitate the hybridization process. The final DX tile concentration was 1 μM. For the second step, the annealed head, tape, and transition tiles were added to a new test tube. Test tubes containing rule tiles were maintained in a Styrofoam box containing 2 L of water at 40 °C gradually cooled from 40 to 25 °C to further facilitate hybridization. A DX tile concentration of 100 nM was achieved in the DNA lattice (see Figures 5 and 6).

https://doi.org/10.1021/acsomega.2c08017
ACS Omega 2023, 8, 15041−15051

AFM Imaging. AFM imaging was performed in the fluid ScanAsyst mode with a 1× TAE/Mg^{2+} buffer. Five microliters of an annealed DNA sample were pipetted onto a mica substrate (5 mm × 5 mm), followed by incubation for 30 s. The buffer (40 μL) was then pipetted onto a mica substrate, and another 20 μL of buffer was pipetted onto an oxide-sharpened silicon nitride AFM tip (Veeco Inc., California). AFM images were obtained using a Digital Instruments Nanoscope V8 (Bruker, Massachusetts) (Figures 5 and 6).

■ ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge at https://pubs.acs.org/doi/10.1021/acsomega.2c08017.

Design of the rectangular Turing building blocks and corresponding double-crossover (DX) DNA tiles for the demonstration of the 1−1, 2−1, and 1−2 TMs; sticky-end names; corresponding DNA base sequences used in the six rules (1−1 TM (e.g., R11-0 and R11-1), 2−1 TM (e.g., R21-01(2^{nd}) and R21-09(1^{st})), and 1−2 TM (e.g., R12-02(2^{nd}) and R12-08(1^{st}))); list of rules and corresponding head, tape, and transition rule tiles; and DNA base sequences of strands (PDF)

■ AUTHOR INFORMATION

Corresponding Author

Sung Ha Park − *Department of Physics and Sungkyunkwan Advanced Institute of Nanotechnology (SAINT), Sungkyunkwan University, Suwon 16419, Republic of Korea;* orcid.org/0000-0002-0256-3363; Email: sunghapark@skku.edu

Author

Muhammad Tayyab Raza − *Department of Physics and Sungkyunkwan Advanced Institute of Nanotechnology (SAINT), Sungkyunkwan University, Suwon 16419, Republic of Korea*

Complete contact information is available at: https://pubs.acs.org/10.1021/acsomega.2c08017

Funding

This study was supported by the National Research Foundation (NRF) of Korea (2021R1A2C1005279).

Notes

The authors declare no competing financial interest.

■ REFERENCES

(1) Benenson, Y.; Paz-Elizur, T.; Adar, R.; et al. Programmable and autonomous computing machine made of biomolecules. *Nature* **2001**, *414*, 430−434.

(2) Draper, T. C.; Dueñas-Díez, M.; Pérez-Mercader, J. Exploring the Symbol Processing "Time Interval" Parametric Constraint in a Belousov−Zhabotinsky Operated Chemical Turing Machine. *RSC Adv.* **2021**, *11*, 23151−23160.

(3) Dueñas-Díez, M.; Pérez-Mercader, J. How Chemistry Computes: Language Recognition by Non-Biochemical Chemical Automata. From Finite Automata to Turing Machines. *iScience* **2019**, *19*, 514−526.

(4) Varghese, S.; Elemans, J. A. A. W.; et al. Molecular computing: paths to chemical Turing machines. *Chem. Sci* **2015**, *6*, 6050−6058.

(5) Regev, A.; Shapiro, E. Cellular abstractions: Cells as computation. *Nature* **2002**, *419*, No. 343.

(6) Watson, J. D.; Crick, F. H. C. Molecular Structure of Nucleic Acids: A Structure for Deoxyribose Nucleic Acid. *Nature* **1953**, *171*, 737−738.

(7) Acuna, G. P.; Möller, F.; Holzmeister, P.; Beater, S.; Lalkens, B.; Tinnefeld, P. Fluorescence Enhancement at Docking Sites of DNA Directed Self-Assembled Nanoantennas. *Science* **2012**, *338*, 506−510.

(8) Wang, P.; Gaitanaros, S.; Lee, S.; Bathe, M.; Shih, W. M.; Ke, Y. Programming Self-Assembly of DNA Origami Honeycomb Two Dimensional Lattices and Plasmonic Metamaterials. *J. Am. Chem. Soc.* **2016**, *138*, 7733−7740.

(9) Adleman, L. M. Molecular Computation of Solutions to Combinatorial Problems. *Science* **1994**, *266*, 1021−1024.

(10) Braich, R. S.; Chelyapov, N.; Johnson, C.; Rothemund, P. W. K.; Adleman, L. Solution of a 20-Variable 3-SAT Problem on a DNA Computer. *Science* **2002**, *296*, 499−502.

(11) Mao, C.; LaBean, T. H.; Reif, J. H.; Seeman, N. C. Logical Computation Using Algorithmic Self-Assembly of DNA Triple Crossover Molecules. *Nature* **2000**, *407*, 493−496.

(12) Raza, M. T.; Tandon, A.; Park, S.; Lee, S.; Nguyen, T. B. N.; Vu, T. H. N.; Jo, S.; Nam, Y.; Jeon, S.; Jeong, J. H.; Park, S. H. Demonstration of elementary functions via DNA algorithmic self-assembly. *Nanoscale* **2021**, *13*, 19376−19384.

(13) Cho, H.; Mitta, S. B.; Song, Y.; Son, J.; Park, S.; Ha, T. H.; Park, S. H. 3-Input/1-Output Logic Implementation Demonstrated by DNA Algorithmic Self-Assembly. *ACS Nano* **2018**, *12*, 4369−4377.

(14) Cheng, Z. Computation of Multiplicative Inversion and Division in GF(2n) by Self-Assembly of DNA Tiles. *J. Comput. Theor. Nanosci.* **2012**, *9*, 336−346.

(15) Tandon, A.; Song, Y.; Mitta, S. B.; Yoo, S.; Park, S.; Lee, S.; Raza, M. T.; Ha, T. H.; Park, S. H. Demonstration of Arithmetic Calculations by Tile-Based Algorithmic Self-Assembly. *ACS Nano* **2020**, *14*, 5260−5267.

(16) Margulies, D.; Melman, G.; Shanzer, A. A Molecular Full-Adder and Full-Subtractor, an Additional Step toward a Moleculator. *J. Am. Chem. Soc.* **2006**, *128*, 4865−4871.

(17) Woods, D.; Doty, D.; Myhrvold, C.; Hui, J.; Zhou, F.; Yin, P.; Winfree, E. Diverse and Robust Molecular Algorithms Using Reprogrammable DNA Self- Assembly. *Nature* **2019**, *567*, 366−372.

(18) Qian, L.; Winfree, E. Scaling Up Digital Circuit Computation with DNA Strand Displacement Cascades. *Science* **2011**, *332*, 1196−1201.

(19) Raza, M. T.; Tandon, A.; Son, J.; Park, S.; Lee, S.; Cho, H.; Ha, T. H.; Park, S. H. Construction of One-Dimensional Random Walk Lattices using DNA Algorithmic Self-Assembly. *AIP Adv.* **2020**, *10*, No. 065229.

(20) Lin, C.-H.; Cheng, H.-P.; Yang, C.-B.; Yang, C.-N. Solving Satisfiability Problems Using a Novel Microarray-Based DNA Computer. *Biosystems* **2007**, *90*, 242−252.

(21) Cho, H.; Mitta, S. B.; Song, Y.; Son, J.; Park, S.; Ha, T. H.; Park, S. H. 3-Input/1-Output Logic Implementation Demonstrated by DNA Algorithmic Self-Assembly. *ACS Nano* **2018**, *12*, 4369−4377.

(22) Turing, A. M. On computable numbers, with an application to the Entcheidungproblem. *Proc. Lond. Math. Soc. II Ser.* **1936**, *42*, 230−265.

(23) Turing, A. M. The chemical basis of morphogenesis. *Philos. Trans. R. Soc. London, Ser. B* **1952**, *237*, 37−72.

(24) Wolfram, S. Statistical Mechanics of Cellular Automata. *Rev. Mod. Phys.* **1983**, *55*, 601−644.

(25) Wolfram, S. Universality and Complexity in Cellular Automata. *Phys. D* **1984**, *10*, 1−35.