Corrected: Author Correction

https://doi.org/10.1038/s41586-019-1014-9

Diverse and robust molecular algorithms using reprogrammable DNA self-assembly

Damien Woods[1,2,8,9]*, David Doty[1,3,9]*, Cameron Myhrvold[4,5], Joy Hui[1,6], Felix Zhou[1,7], Peng Yin[4,5] & Erik Winfree[1]*

Molecular biology provides an inspiring proof-of-principle that chemical systems can store and process information to direct molecular activities such as the fabrication of complex structures from molecular components. To develop information-based chemistry as a technology for programming matter to function in ways not seen in biological systems, it is necessary to understand how molecular interactions can encode and execute algorithms. The self-assembly of relatively simple units into complex products[1] is particularly well suited for such investigations. Theory that combines mathematical tiling and statistical–mechanical models of molecular crystallization has shown that algorithmic behaviour can be embedded within molecular self-assembly processes[2,3], and this has been experimentally demonstrated using DNA nanotechnology[4] with up to 22 tile types[5–11]. However, many information technologies exhibit a complexity threshold—such as the minimum transistor count needed for a general-purpose computer—beyond which the power of a reprogrammable system increases qualitatively, and it has been unclear whether the biophysics of DNA self-assembly allows that threshold to be exceeded. Here we report the design and experimental validation of a DNA tile set that contains 355 single-stranded tiles and can, through simple tile selection, be reprogrammed to implement a wide variety of 6-bit algorithms. We use this set to construct 21 circuits that execute algorithms including copying, sorting, recognizing palindromes and multiples of 3, random walking, obtaining an unbiased choice from a biased random source, electing a leader, simulating cellular automata, generating deterministic and randomized patterns, and counting to 63, with an overall per-tile error rate of less than 1 in 3,000. These findings suggest that molecular self-assembly could be a reliable algorithmic component within programmable chemical systems. The development of molecular machines that are reprogrammable—at a high level of abstraction and thus without requiring knowledge of the underlying physics—will establish a creative space in which molecular programmers can flourish.

In DNA nanotechnology, an essential building block for programmable self-assembly is the so-called DNA tile. Structural motif variants include the double-crossover tile[12], the triple-crossover tile[5], and the single-stranded tile (SST)[13]. DNA tiles bind through Watson–Crick complementary domains to several neighbours arranged on a one-, two-, or three-dimensional lattice. A small number of DNA tile types—each using the same motif, but with different sequences dictating their neighbourhood relationships—is sufficient to program the self-assembly of large (micrometre-scale) crystalline structures in which each tile type appears periodically[12,13]. DNA origami[14] has also inspired designs composed of hundreds to tens of thousands of distinct tile types that self-assemble into finite-sized 'uniquely addressed' structures, in which each tile type appears in exactly one location and has a fixed set of neighbouring tiles[15–17].

Beyond periodic and uniquely addressed structures, the combinatorial arrangement of distinct molecular units can store information that can be processed by algorithmically driven choices inherent in the self-assembly itself[2]. Such algorithmic self-assembly can generate arbitrarily complex patterns and structures from information encoded in a seed structure[18,19], similar in spirit to genomic DNA encoding a genetic program for the development of an organism. Although unmistakably algorithmic growth—including Sierpinski triangles and binary counter patterns—has been demonstrated experimentally with double-crossover and triple-crossover tiles[5–11], tile sets have so far been small and hard-coded for a specific assembly algorithm. Our goal was to unlock a new level of sophistication by creating a reprogrammable self-assembly system, which requires the scaling up of algorithmic self-assembly to hundreds of tile types, for which SSTs are a natural choice. However, algorithmic self-assembly puts more stringent constraints on system design, sequence design, and biophysical properties of the tiles than does periodic and uniquely addressed self-assembly.

System design begins with an abstract model of computation—the Iterated Boolean Circuit (IBC) model—that is naturally suited to a robust molecular implementation as the growth of SSTs into nanotubes (see Fig. 1a and Supplementary Information section S1). The model considers a locally connected array of Boolean gates that are executed repeatedly, with an n-bit l-layer IBC having $(n-1)l$ gates, each with two input and two output wires, and $2l$ boundary gates, each with one input and one output. The logic function computed by each of the $(n+1)l$ gates is specified by the user. The system computes by repeatedly iterating the circuit layers, eventually reaching a fixed point or a cycle. This is a powerful and general model that can simulate many other models of computation[20], such as Turing machines, general Boolean circuits, cellular automata[21,22], and bounded-width branching programs (Supplementary Information section S1.3). For molecular implementation, we choose $n = 6$ and $l = 1$, which corresponds to 2^{44} possible distinct circuits.

We followed a hierarchy of abstractions (Fig. 1a–e) to arrive at a molecular implementation of the 6-bit 1-layer IBC model, which itself sits at the first level. The second abstraction level is the abstract Tile Assembly Model (aTAM), which describes square monomer tiles whose self-assembly is guided by information-bearing 'glues' on each side[2] (Fig. 1b and Supplementary Information section S2). The aTAM considers the 'cooperative' assembly regime, whereby a tile may attach to a growing assembly only if at least two glues match. The model assumes an excess of each monomer type in solution, so they are never exhausted. Each two-input/two-output gate of the IBC is implemented by four abstract tiles, one for each possible pair of input bits. Glues encode both bit values and vertical gate position within the circuit. An important benefit of the unique addressing of the vertical gate position is that, as suggested by previous theoretical and experimental studies of self-assembled DNA ribbons and tubes, it can result in substantial kinetic barriers to spontaneous nucleation[7,8,13,23,24]. Thus, the model assumes that self-assembly proceeds exclusively from an initial seed assembly representing the input to the circuit, growing a pattern that represents the execution of the circuit and in principle continuing forever. The seed is cylindrical, so that growth forms a tube and each tile attaches by the same number of glues (two) with uniform local geometry.

[1]California Institute of Technology, Pasadena, CA, USA. [2]Inria, Paris, France. [3]University of California, Davis, CA, USA. [4]Wyss Institute for Biologically Inspired Engineering, Harvard University, Boston, MA, USA. [5]Department of Systems Biology, Harvard University, Boston, MA, USA. [6]Harvard University, Cambridge, MA, USA. [7]University of Oxford, Oxford, UK. [8]Present address: Maynooth University, Maynooth, Ireland. [9]These authors contributed equally: Damien Woods, David Doty. *e-mail: damien.woods@mu.ie; doty@ucdavis.edu; winfree@caltech.edu

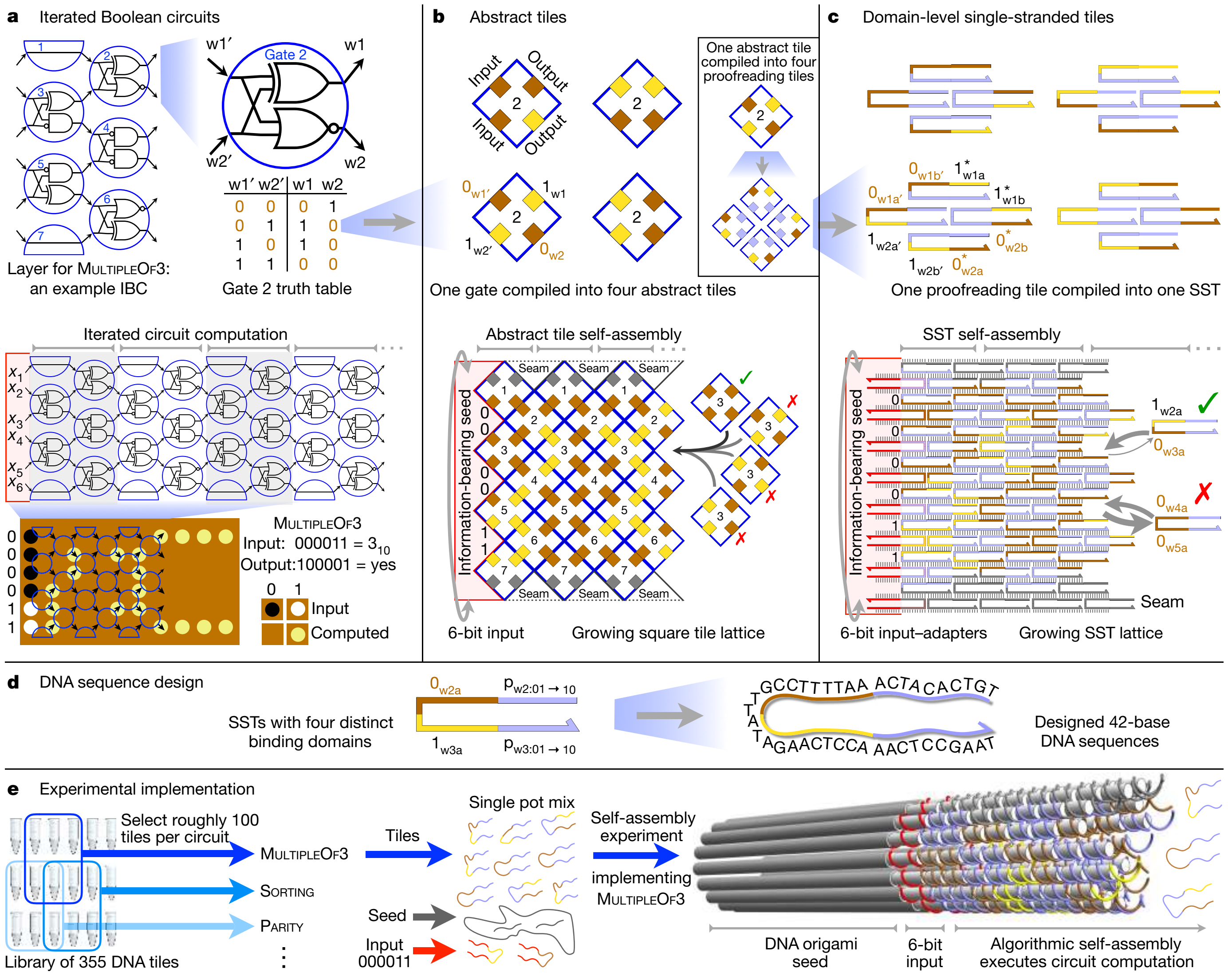
a Iterated Boolean circuits
Layer for MultipleOf3: an example IBC
Gate 2
w1′ w2′ w1 w2
Gate 2 truth table
Iterated circuit computation
MultipleOf3
Input: 000011 = 3_10
Output:100001 = yes
Input
Computed
b Abstract tiles
Input Output
One abstract tile compiled into four proofreading tiles
One gate compiled into four abstract tiles
Abstract tile self-assembly
Seam
Information-bearing seed
6-bit input
Growing square tile lattice
c Domain-level single-stranded tiles
One proofreading tile compiled into one SST
SST self-assembly
6-bit input–adapters
Growing SST lattice
d DNA sequence design
SSTs with four distinct binding domains
Designed 42-base DNA sequences
e Experimental implementation
Select roughly 100 tiles per circuit
MultipleOf3
Sorting
Parity
Library of 355 DNA tiles
Single pot mix
Tiles
Seed
Input 000011
Self-assembly experiment implementing MultipleOf3
DNA origami seed
6-bit input
Algorithmic self-assembly executes circuit computation

Fig. 1 | Abstraction hierarchy for design and implementation of the complete 6-bit IBC tile set. a, Programming a deterministic 6-bit 1-layer circuit. Top, each gate position (1–7) admits either one of four possible one-input/one-output gates, or one of 256 possible two-input/two-output gates. Wires carry input and output signals between (w1–w6) and within (w1′–w6′) layers. To program a randomized circuit, two or more gates are selected for at least one gate position. Bottom, the circuit computes via iteration: we start with the six input bits and append copies of the circuit layer one after another. Below the circuit is an example execution, where a yellow circle denotes a wire value of 1 and its absence denotes 0. **b**, Each two-input gate is compiled to four abstract square tiles, with one tile for each line of the gate's truth table. Each tile glue encodes a bit and a wire. The lattice of abstract tiles grows from a seed and wraps around into a tube via a square seam tile. A green tick and black arrow indicate where a tile may attach correctly, whereas a red cross and grey arrow indicate where tiles would not correctly match, given the logic encoded in the glues. Inset, for error suppression, each abstract tile is converted into four 'proofreading' tiles. **c**, Each proofreading tile is implemented by one DNA SST. Thus, 16 SSTs implement each circuit gate. Yellow and brown domain labels encode bit value (0 or 1), wire position (w1–w6, w1′–w6′), proofreading index (a or b), and Watson–Crick complementarity (∗); blue domain labels are unique to the proofreading block and position within the block. For a tile to attach stably, two domains must match. The thickness of grey arrows indicates the relative rates of tile attachment and detachment events. **d**, The tile set design software automatically generates labels for unique domains within each proofreading block (for example, $p_{w2:01\rightarrow10}$ and $p_{w3:01\rightarrow10}$) while DNA sequence design software assigns a 10- or 11-nucleotide sequence to each distinct domain label. **e**, Experimentally, a deterministic 6-bit circuit is programmed by choosing seven gates, corresponding to selecting 100 DNA tiles from the library of 355. Additional DNA strands encode the 6-bit input and a seed. Algorithmic self-assembly directs the growth of a DNA nanotube according to circuit logic; the input (red) is encoded by DNA strands extending from one end of the seed structure (grey), and the attachment of DNA strands implements the execution of circuit gates.

At the third abstraction level, each abstract tile is converted into four 'proofreading' tiles (Fig. 1b inset and Supplementary Information section S2.3) that, according to a more biophysically realistic self-assembly theory that is supported by experimental studies, should have a substantially lower error rate during assembly than a direct implementation of abstract tiles without proofreading[2,6,8,11,24,25]. This results in a set of 355 tile types that we call the complete 6-bit IBC tile set, because it contains all tiles needed to implement any 6-bit 1-layer IBC; for a particular choice of circuit, only the tiles corresponding to the chosen Boolean gates are used.

The fourth abstraction level defines an SST binding-domain schematic that includes key geometric considerations but excludes DNA sequence and biophysical details (Fig. 1c and Supplementary Information section S3). Each proofreading tile is represented by one SST that is logically partitioned into four domains, one corresponding to each glue. To be compatible with implementing the seed structure as a barrel-shaped DNA origami[8,14,26], we include input–adapter strands that encode the circuit's input bits and are uniquely addressed to the appropriate position on the seed.

Although the intended behaviour is straightforward to describe in terms of SST binding-domain interactions, there are substantial

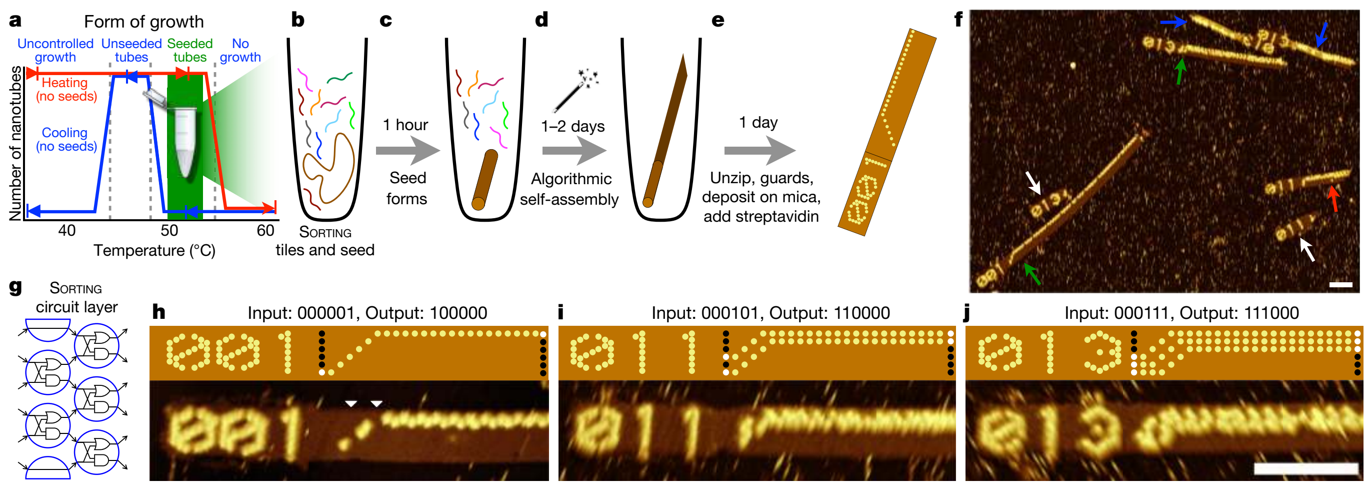

Fig. 2 | Experimental protocol and implementation of the Sorting circuit. a, The blue and red curves conceptually illustrate unseeded experiments that were used to determine the optimal seeded growth temperature range highlighted in green (see Supplementary Information section S5.1 for data) for a concentration of 100 nM per tile type. **b**, One-pot experiment with seed strands, input–adapters, and tiles that implement the Sorting circuit shown in panel **g**. The sample, which contains seed at 1 nM and each tile type at 100 nM, is cooled from 90 °C and held at 50.8 °C. **c**, DNA origami seed forms. **d**, Algorithmic self-assembly occurs. **e**, Samples are prepared for imaging. **f**, Example of a multiplexed AFM image chosen to highlight various properties of the data. Green arrows indicate barcoded 001 and 013 nanotubes with the intended correct growth. The red arrow indicates a single algorithmic error on an otherwise correct 011 nanotube; an erroneous 1 bit attaches and is sorted, adding a third row of 1 bits. Blue arrows show two 013 incompletely unzipped nanotubes. White arrows point to two seed barcodes (013, 011) with little or no growth from seed. **g**, The Sorting circuit layer. **h**–**j**, Simulation (top) and AFM images (bottom) of the Sorting circuit on three different 6-bit inputs. Input bits are illustrated in black (0) and white (1). Two streptavidin labelling errors are highlighted using white triangles in **h**. Scale bars, 100 nm.

biophysical challenges at the fifth abstraction level—that of DNA sequences (Fig. 1d and Supplementary Information section S4). Here we require strands that are capable of desired interactions and not unduly prone to undesired interactions. Two primary concerns are spurious nucleation (the spontaneous formation of nanotubes that are not seeded by origami) and tile-attachment errors (the attachment of tiles despite mismatching domains). Although both concerns informed our design of the abstract tile set, the effectiveness of both the spurious nucleation barrier and the proofreading depends on the uniformity and specificity of glue-directed tile-attachment energies[24,27], which sets up a dual set of challenges for sequence design. Because the SSTs are floppy before being incorporated into the nanotube, ensuring uniformity of energies for tile-binding events requires more than mere domain-binding energy uniformity: domains on the same strand could bind to each other and neighbouring exposed domains on the growth front of a nanotube could bind to each other, both of which must be undone during tile-attachment events; in addition, strands could bind together in solution, resulting in lowered and unequal concentrations of free monomers. Furthermore, enhancing specificity by minimizing the mismatch energies of tiles binding with one correct domain and one incorrect domain entails similar considerations. All of these factors are explicitly accounted for in our sequence-design pipeline. Because existing software for predicting nucleic secondary structure from thermodynamic measurements[28,29] cannot account for the idiosyncratic geometric contexts of SST tube growth, we developed an ad hoc model, building on NUPACK and ViennaRNA, to incorporate the relevant effects. The resulting multi-objective optimization problem was tackled using a stochastic local search algorithm.

In order to establish the experimental conditions suitable for seeded growth, we performed a systematic evaluation of spontaneous nucleation for SST nanotubes of circumference 8–16 helices in the absence of a seed (Fig. 2a and Supplementary Information section S5.1). SST strands were quickly cooled (at a rate of 1 °C per minute, in Tris acetate EDTA (TAE) buffer with 12.5 mM Mg^{2+}) from 90 °C to a target growth temperature, where they were held for constant-temperature growth for at least a day. Similarly, preformed nanotubes were quickly heated to a target temperature and held there for at least a day. Using fluorescence microscopy, we identified a tile concentration and temperature range, just below the nanotube's melting temperature, at which spontaneous nucleation can be avoided for more than 24 hours—sufficient time to allow algorithmic growth triggered by a seed. We chose 16-helix nanotubes for further study as they provide the most space for storing information.

To investigate additional requirements for algorithmic self-assembly with SSTs, we designed a tile set that copies 4 bits from layer to layer (similar to the Copy circuit described later, but hard-coded for this function; see Supplementary Information section S5.2), with input specified by a DNA origami seed. Because of the small size of the SST lattice unit cell (3 nm × 7 nm), we used atomic force microscopy (AFM) to visualize information propagating through individual nanotubes; this necessitated several modifications to the design. To open up nanotubes so they could lie flat on the mica imaging surface, with their insides facing up, we used 'unzipping' strands that remove the seam strands by strand displacement[30]. We then used 'guard' strands[9] to shut down the assembly process by inactivating any remaining tiles in solution, and added streptavidin to bind to the tile strands that represent the bit 1, which we had modified to contain an upward-facing biotin molecule (which binds strongly to streptavidin); this resulted in a clear topographic marker for AFM. Initial experiments suggested that the biotin modification weakened the binding energy of domains in which they appeared, and thus rendered the ideal growth temperature dependent on the particular bit sequence being copied. Because algorithmic self-assembly may involve different bit sequences over the course of a computation and thus requires a single ideal growth temperature for all sequences, we modified the sequence-design pipeline by subtracting 1.1 kcal per mol from the predicted binding strength of biotin-modified domains. This resulted in the selection of sequences that, without the biotin, would bind more strongly than those of non-biotin-modified domains. The redesigned 4-bit copy sequences resulted in excellent growth for sequences with a variety of bit patterns. In a final test, we investigated whether algorithmic tile sets could be rendered more efficient by reusing the same tile type on different rows where permitted logically, but our lack of success underlined the importance of the unique addressing of the vertical gate position. These preliminary investigations, and the resulting design decisions for the complete 6-bit IBC tile set, are detailed in Supplementary Information sections S5.3–S5.6.

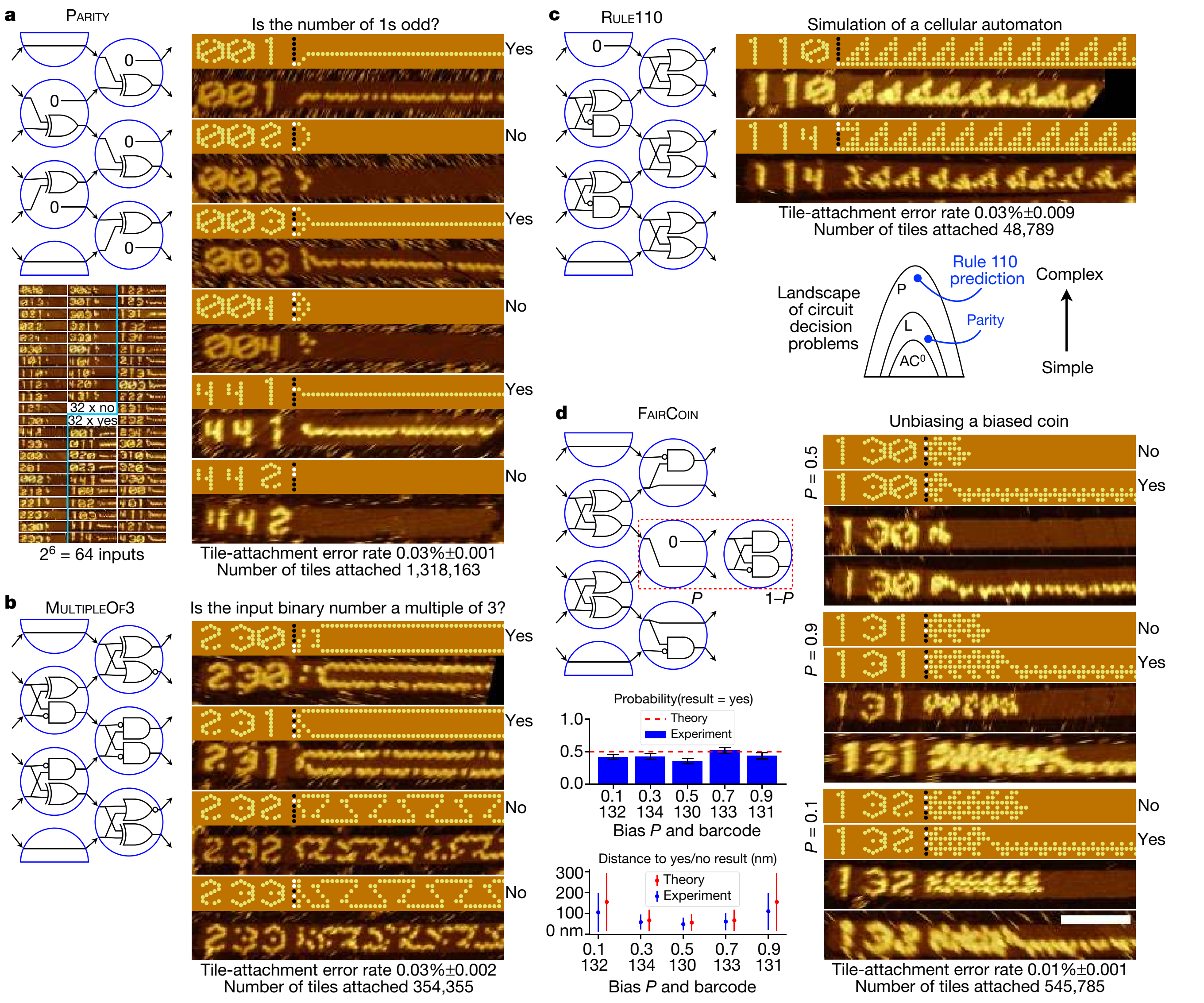
a
Parity
Is the number of 1s odd?
Yes
No
Yes
No
Yes
No
32 x no
32 x yes
2^6 = 64 inputs
Tile-attachment error rate 0.03%±0.001
Number of tiles attached 1,318,163
b
MultipleOf3
Is the input binary number a multiple of 3?
Yes
Yes
No
No
Tile-attachment error rate 0.03%±0.002
Number of tiles attached 354,355
c
Rule110
Simulation of a cellular automaton
Tile-attachment error rate 0.03%±0.009
Number of tiles attached 48,789
Landscape of circuit decision problems
P
L
AC⁰
Rule 110 prediction
Parity
Complex
Simple
d
FairCoin
P
1–P
Unbiasing a biased coin
P = 0.5
P = 0.9
P = 0.1
No
Yes
Probability(result = yes)
Theory
Experiment
1.0
0.5
0.0
0.1 0.3 0.5 0.7 0.9
132 134 130 133 131
Bias P and barcode
Distance to yes/no result (nm)
300
200
100
0 nm
Tile-attachment error rate 0.01%±0.001
Number of tiles attached 545,785

Fig. 3 | Reprogramming IBCs. a, Parity circuit, which decides whether the 6-bit input contains an odd number of 1 bits. Input bits 0 and 1 are shown as black and white circles, respectively. The circuit's gates compute an exclusive OR (XOR) of their 2-bit inputs, sending the result towards one of the centre wires; if a single 1 bit survives, it is copied along that wire, outputting 000100 forever (making a horizontal stripe, which is interpreted as 'yes'). Otherwise, 000000 is copied forever ('no'). **b**, The MultipleOf3 circuit creates one of two repeating patterns depending on whether the input is the binary expansion of a multiple of 3. Seeds 230 and 231 represent inputs $000011_2 = 3$ and $010101_2 = 21$, both multiples of 3, unlike seeds 232 ($010000_2 = 16$) and 233 ($110101_2 = 53$). **c**, The Rule110 circuit simulates three-cell instances of the cellular automaton 'rule 110'. The Venn diagram positions problems solved by IBCs in the computational complexity hierarchy[20] of problems solved by polynomial time Turing machines (P), logarithmic space Turing machines (L), or constant depth circuits (AC^0). **d**, The randomized circuit FairCoin has a choice of two heads/tails gates that encode a biased coin (red dashed rectangle), and its output is an unbiased 'yes'/'no' outcome, encoded respectively as stripe/no stripe. Plots show the analysis of 643 nanotubes, as compared with theory, for five coin biases. In the inset probability plot, the shown value is the sample mean, and error bars are standard errors of the mean; in the inset distance plot, the centre value is the sample mean, and error bars are the standard deviation for an individual trial. Scale bar, 100 nm.

Tests of the complete 6-bit IBC tile set again indicated a narrow temperature range that permits robust algorithmic self-assembly with minimal spurious nucleation for all bit patterns. In a typical experiment (see Fig. 2 and Supplementary Information section S6), origami seed strands, input–adapters and SSTs (all unpurified except for the M13 origami scaffold and biotin-labelled tile strands) for any given circuit were cooled from 90 °C at a rate of 1 °C per minute and held at 50.7 °C or 50.8 °C for at least a day, followed by the addition of unzipping strands and then guard strands. Multiple samples (each with a unique origami barcode) were mixed and deposited on mica. Finally, excess strands were washed away and streptavidin added for final read-out by AFM (Supplementary Information section S5.5.2). Although streptavidin labelling was seldom complete, most algorithmic patterns have the fortunate property that missing labels can be readily distinguished from erroneous assembly on the basis of consistency with subsequent steps, allowing systematic AFM image analysis (Supplementary Information section S7).

The Sorting circuit, which implements an 'odd–even transposition' sort, is especially suitable for analysis because its computation is easy to understand, yet nontrivial (Fig. 2f–j). Each two-input gate in the circuit layer flips pairs of bits that are out of order; the state stabilizes with all 1s at the top after they have passed through three layers (that is, six gates). Figure 2h–j shows three nanotubes, grown from three distinct origami seeds, that correctly sort three respective inputs. As shown in Fig. 2f, some origami seeds failed to grow nanotubes (white arrows; 63.3% of 1,299 analysed); some nanotubes failed to unzip and could not be read

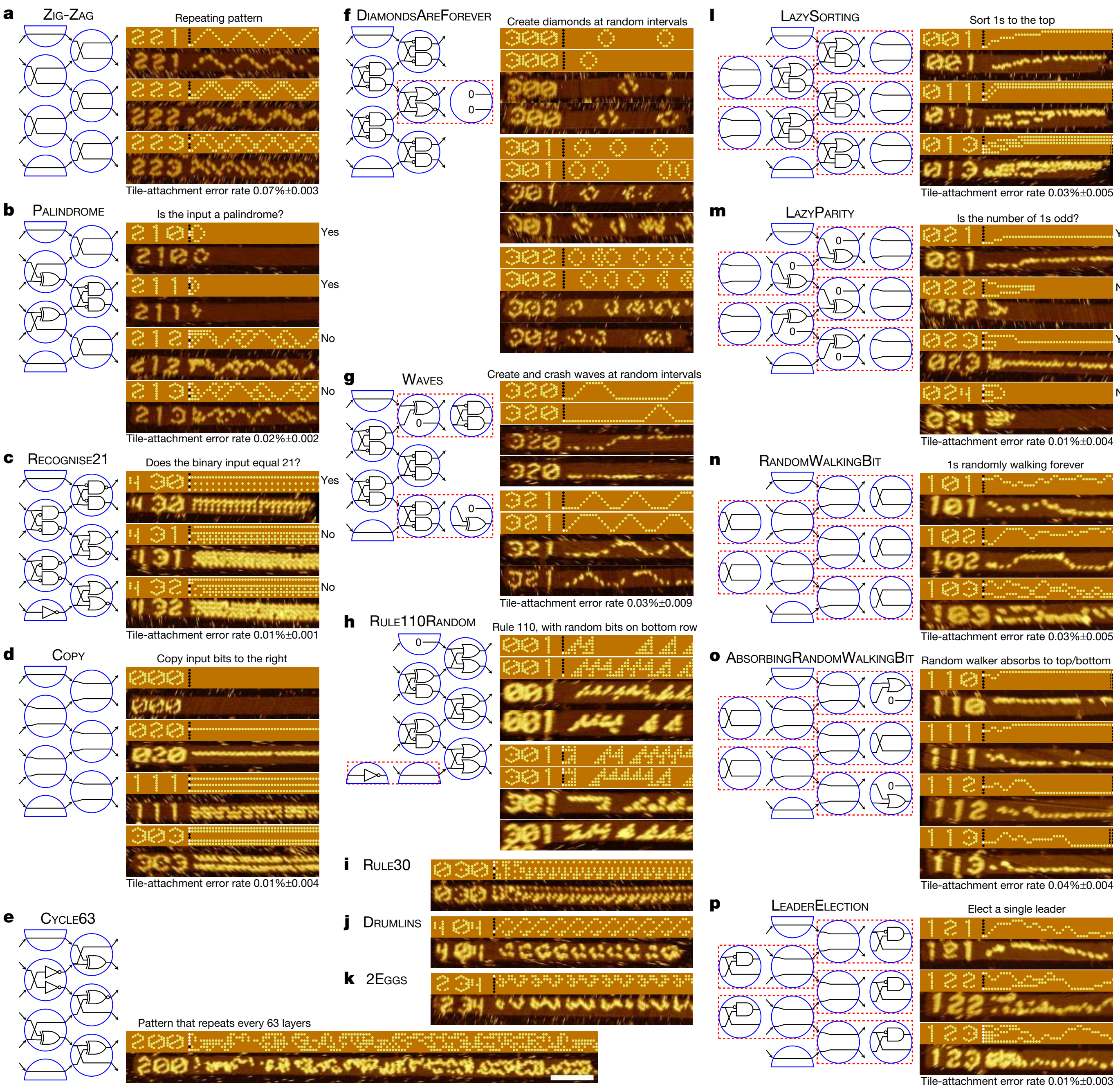
a ZIG-ZAG
Repeating pattern
Tile-attachment error rate 0.07%±0.003
b PALINDROME
Is the input a palindrome?
Yes
Yes
No
No
Tile-attachment error rate 0.02%±0.002
c RECOGNISE21
Does the binary input equal 21?
Yes
No
No
Tile-attachment error rate 0.01%±0.001
d COPY
Copy input bits to the right
Tile-attachment error rate 0.01%±0.004
e CYCLE63
Pattern that repeats every 63 layers
f DIAMONDSAREFOREVER
Create diamonds at random intervals
g WAVES
Create and crash waves at random intervals
Tile-attachment error rate 0.03%±0.009
h RULE110RANDOM
Rule 110, with random bits on bottom row
i RULE30
j DRUMLINS
k 2EGGS
l LAZYSORTING
Sort 1s to the top
Tile-attachment error rate 0.03%±0.005
m LAZYPARITY
Is the number of 1s odd?
Yes
No
Yes
No
Tile-attachment error rate 0.01%±0.004
n RANDOMWALKINGBIT
1s randomly walking forever
Tile-attachment error rate 0.03%±0.005
o ABSORBINGRANDOMWALKINGBIT
Random walker absorbs to top/bottom
Tile-attachment error rate 0.04%±0.004
p LEADERELECTION
Elect a single leader
Tile-attachment error rate 0.01%±0.003

Fig. 4 | Testing of the complete 6-bit IBC tile set. a–p, A further 16 circuits that span a range of deterministic and randomized algorithmic behaviours. The circuits in **a**–**e** and **i**–**k** are deterministic; the remainder are randomized. For the randomized circuits in **f** and **g**, different seed barcodes indicate different gate probabilities, rather than different input bits. The simulations for the randomized circuits intentionally do not match the experimental images to emphasize the possibility of different random choices during execution. See Supplementary Information sections S8.1–S8.21 for details, including AFM images of additional seeds. Scale bar, 100 nm.

(blue arrows); and some nanotubes had logical errors that flipped bits and thus were propagated into subsequent layers (red arrow). By identifying logical errors and measuring the lengths of opened nanotubes, we estimated the tile-attachment error rate for the SORTING circuit to be 0.03% (77 errors out of an estimated 269,028 tile attachments).

Using our abstraction hierarchy, the specification of a deterministic circuit can be compiled into a subset of 100 strands from the 355 in the complete 6-bit IBC tile set (Fig. 1e). By programming different IBCs and compiling them into DNA strands, we specified and grew DNA nanotubes that performed different computations (see Supplementary Information sections S8.1–S8.21 for details about each implemented circuit). The PARITY circuit (Fig. 3a) moves 1 bits to the centre and has them cancel each other out if they meet. The circuit stabilizes with a single 1 in the centre for all inputs that have an odd number of 1s, and stabilizes with all 0s for inputs that have an even number of 1s. We tested the PARITY circuit on all 64 possible 6-bit inputs, and all computed correctly, with a tile-attachment error rate of 0.03% per tile. The MULTIPLEOF3 circuit was tested on four inputs and computed with similar accuracy (Fig. 3b). We simulated the ‘rule 110’ elementary cellular automaton with the circuit RULE110 (Fig. 3c). Although only three cells were simulated here, when generalized to allow more cells RULE110 becomes efficiently computationally universal, meaning that it can simulate any algorithm[21] with only polynomial-time overhead[22].

Beyond deterministic computation, the IBC model and its molecular implementation can perform randomized computation. By mixing different tile types that have identical input glues (resulting in more than 100 tile types in total), we can implement randomized gates whose output probabilities depend on the relative concentrations of tile types. The FairCoin circuit has a randomized gate implementing a biased coin (Fig. 3d). Following previous algorithmic tile assembly constructions[31], the circuit implements von Neumann's procedure for using a biased coin to simulate a fair coin with exact 50/50 odds, reporting the fair coin result as the presence or absence of a stripe. This is accomplished by flipping the biased coin twice, repeating the procedure if the tosses agree, and reporting the second result if they differ. The circuit was tested over a 100-fold range of tile-concentration ratios, with the expected result of near 50% outcome probabilities (that is, about half of the ribbons had a stripe), but with decision times that (as expected) increase for an increasingly biased source of randomness.

To more comprehensively exhibit the computational capabilities of the complete 6-bit IBC tile set, we devised an additional 16 circuits spanning a range of deterministic and randomized algorithmic behaviours (Fig. 4). We further tested for correctness: all 355 tiles were used at some point, and 15 of the 21 circuits were tested on enough inputs that all of the circuit's tiles were used. No tiles had to be redesigned and no tuning of experimental conditions was required for any of the 21 circuits; every circuit worked on the first try. Tile-attachment error rates varied between 0.01% and 0.07% per circuit, comparable with the best error rates reported previously (0.017% ± 0.013)[9]. Averaged over all 21 circuits, 61.1% of origami seeds grew nanotubes and the overall tile-attachment error rate was 0.03% ± 0.0008 (1,419 observed errors out of an estimated 4,600,351 tile attachments).

Given that every circuit performed as expected, we conclude that every tile performed as designed and that there were no systematic design flaws. Although the seeding fraction merits improvement, the tile-attachment error rates are consistent, within a factor of two, with theoretical estimates from physical principles[24] (Supplementary Information section S7.6), suggesting that they are near optimal for our setup. This was achieved even though—in contrast with previous demonstrations of algorithmic self-assembly—we used cheap unpurified DNA molecules wherever possible, introducing additional sources of error owing to incompletely synthesized tiles. We attribute our success to three key principles: effective sequence design; use of an abstraction hierarchy to manage design and experiment complexities; and the reprogrammability of the tile set.

In contrast with our emphasis on sequence design, previous SST work successfully used random or lightly designed DNA sequences[15,17]. Coarse-grained models have even suggested that this is preferable for uniquely addressed self-assembly, predicting better performance from random sequences where stronger-binding tiles assemble first to act as incidental seeds that allow growth nearer to equilibrium[32]. But with an explicit seed added, as required for algorithmic self-assembly, simulation and theory predict that sequences designed to bind with uniform energies and high specificity will assemble with fewer errors[24,33] (Supplementary Information section S4.1). The resultant sequence-design constraints limit the scalability of algorithmic self-assembly, in a manner that depends on the choice of tile motif. For example, previous algorithmic self-assembly[5–11] used double-crossover or triple-crossover tiles with a rigid core (for example, 64 base pairs in two parallel helices for double-crossover tiles) that can be obtained with many different sequences, and assembled via four short (for example, 5-nucleotide) binding domains for which it may be difficult to design more than 100 distinct high-quality binding sequences[27]. Although strand floppiness complicates SST sequence design, the longer binding-domain length permits a greater number of domain sequences (412 here, which we do not believe to be the limit).

Our automated compiler was essential for designing an algorithmic tile set with several hundred tile types. The compiler follows a well-defined abstraction hierarchy from circuit to sequences, facilitating automated verification of each compilation step, seamless incorporation of proofreading for assembly-error correction, and systematic generation of experimental protocols. This separates the concerns of the programmer, who may focus attention at the circuit level, from those of the experimenter, who may focus attention on executing efficient and parallel protocols—helpful even when both are the same person but at different times.

The reprogrammability of our tile set was essential for demonstrating a diverse range of circuits. We knew of only three interesting circuits when ordering strands for the complete 6-bit IBC tile set, with the other 18 designed or discovered later on in instances of 'programming while at the bench'. Although requiring further scale-up, a more advanced conception of reprogrammability is theoretically possible in which a fixed tile set is 'universal' for both computation[2] and construction in the sense that information in a seed[18,19] can specify an algorithm that directs the assembly of an arbitrary structure—with precise control of patterns, shapes and growth pathways.

Algorithmic self-assembly should also be possible using other types of molecules[1], including RNA and proteins. With a sufficiently accurate biophysical model for such polymers, a compiler that uses a hierarchical stack of abstractions should be able to systematically design sets of molecules that process information during self-assembly. Beyond the engineering potential, such concrete implementations and illustrations of molecular self-assembly algorithms should provide new insight into the design space that biological systems explore. First consider an alternative perspective on our work: uniquely addressed SST structures[15,17] have been described as a structural 'molecular canvas', in that a single multipurpose tile set can be 'carved' by the artist to create almost any shape simply by leaving out the tiles that are not needed for that shape, because each tile is used in a unique position. Generalizing this notion, our multipurpose tile set can be seen as an 'algorithmic molecular canvas', in the sense that growth using all 355 tiles corresponds to the execution of an IBC in which every gate position is fully randomized, and by leaving out tiles an algorithm can be 'carved' by the programmer to yield more and more deterministic behaviour towards some desired function. Thus, there is a continuous space of probabilistic self-assembly algorithms that can be tuned—in fact programmed—by adjusting tile concentrations without changing the tiles themselves. Considering, by analogy, a finite set of protein monomers with fixed self-assembly interactions, we can now see that by simply tuning their level of genetic expression, evolution can smoothly explore a sophisticated space of self-assembly behaviours. Going back further, self-assembly could have provided a simple source of algorithmic potential during the origin and early evolution of life[9,34]. By establishing that an algorithmic molecular canvas can be as easy to manipulate as a structural molecular canvas, whether by design or by nature, our work suggests that molecular engineering and molecular science are entering the algorithmic era.

Data availability

The AFM data generated and analysed here are available from the authors' website (http://www.dna.caltech.edu/SupplementaryMaterial/Algorithmic_SST/), as is the Python code for data analysis and DNA sequence design.

Online content

Any methods, additional references, Nature Research reporting summaries, source data, statements of data availability and associated accession codes are available at https://doi.org/10.1038/s41586-019-1014-9.

Received: 22 May 2018; Accepted: 7 January 2019;
Published online 20 March 2019.

1. Whitesides, G. M. & Grzybowski, B. Self-assembly at all scales. *Science* **295**, 2418–2421 (2002).
2. Winfree, E. *Simulations of Computing by Self-Assembly. Technical Report CaltechCSTR:1998.22* (California Institute of Technology, 1998).
3. Doty, D. Theory of algorithmic self-assembly. *Commun. ACM* **55**, 78–88 (2012).
4. Seeman, N. C. & Sleiman, H. F. DNA nanotechnology. *Nat. Rev. Mater.* **3**, 17068 (2017).
5. Mao, C., LaBean, T. H., Reif, J. H. & Seeman, N. C. Logical computation using algorithmic self-assembly of DNA triple-crossover molecules. *Nature* **407**, 493–496 (2000); erratum **408**, 750 (2000).

6. Rothemund, P. W. K., Papadakis, N. & Winfree, E. Algorithmic self-assembly of DNA Sierpinski triangles. *PLoS Biol.* **2**, e424 (2004).
7. Schulman, R. & Winfree, E. Synthesis of crystals with a programmable kinetic barrier to nucleation. *Proc. Natl Acad. Sci. USA* **104**, 15236–15241 (2007).
8. Barish, R. D., Schulman, R., Rothemund, P. W. K. & Winfree, E. An information-bearing seed for nucleating algorithmic self-assembly. *Proc. Natl Acad. Sci. USA* **106**, 6054–6059 (2009).
9. Schulman, R., Yurke, B. & Winfree, E. Robust self-replication of combinatorial information via crystal growth and scission. *Proc. Natl Acad. Sci. USA* **109**, 6405–6410 (2012).
10. Evans, C. G. *Crystals that Count! Physical Principles and Experimental Investigations of DNA Tile Self-Assembly*. PhD thesis, Caltech (2014).
11. Schulman, R., Wright, C. & Winfree, E. Increasing redundancy exponentially reduces error rates during algorithmic self-assembly. *ACS Nano* **9**, 5760–5771 (2015).
12. Winfree, E., Liu, F., Wenzler, L. A. & Seeman, N. C. Design and self-assembly of two-dimensional DNA crystals. *Nature* **394**, 539–544 (1998).
13. Yin, P. et al. Programming DNA tube circumferences. *Science* **321**, 824–826 (2008).
14. Rothemund, P. W. K. Folding DNA to create nanoscale shapes and patterns. *Nature* **440**, 297–302 (2006).
15. Wei, B., Dai, M. & Yin, P. Complex shapes self-assembled from single-stranded DNA tiles. *Nature* **485**, 623–626 (2012).
16. Wang, W. et al. Self-assembly of fully addressable DNA nanostructures from double crossover tiles. *Nucleic Acids Res.* **44**, 7989–7996 (2016).
17. Ong, L. L. et al. Programmable self-assembly of three-dimensional nanostructures from 10,000 unique components. *Nature* **552**, 72–77 (2017).
18. Rothemund, P. W. K. & Winfree, E. The program-size complexity of self-assembled squares. In *STOC: Proceedings of the 32nd Annual ACM Symposium on Theory of Computing* (eds Yao, F. & Luks, E.) 459–468 (Association for Computing Machinery, 2000).
19. Soloveichik, D. & Winfree, E. Complexity of self-assembled shapes. *SIAM J. Comput.* **36**, 1544–1569 (2007).
20. Moore, C. & Mertens, S. *The Nature of Computation* (Oxford Univ. Press, Oxford, 2011).
21. Cook, M. Universality in elementary cellular automata. *Complex Syst.* **15**, 1–40 (2004).
22. Neary, T. & Woods, D. P-completeness of cellular automaton Rule 110. In *ICALP 2006: International Colloquium on Automata, Languages, and Programming* (eds Bugliesi, M., Preneel, B., Sassone, V. & Wegener, I.) 132–143 (Springer, 2006).
23. Schulman, R. & Winfree, E. Programmable control of nucleation for algorithmic self-assembly. *SIAM J. Comput.* **39**, 1581–1616 (2009).
24. Evans, C. G. & Winfree, E. Physical principles for DNA tile self-assembly. *Chem. Soc. Rev.* **46**, 3808–3829 (2017).
25. Winfree, E. & Bekbolatov, R. Proofreading tile sets: error correction for algorithmic self-assembly. In *DNA9: Proceedings of the 9th International Conference on DNA Computing* (eds Chen, J. & Reif, J.) 126–144 (Springer, 2004).
26. Mohammed, A. M. & Schulman, R. Directing self-assembly of DNA nanotubes using programmable seeds. *Nano Lett.* **13**, 4006–4013 (2013).
27. Evans, C. G. & Winfree, E. DNA sticky end design and assignment for robust algorithmic self-assembly. In *DNA19: Proceedings of the 19th International Conference on DNA Computing and Molecular Programming* (Eds Soloveichik, D. & Yurke, B.) 61–75 (Springer, 2013).
28. Zadeh, J. N. et al. NUPACK: analysis and design of nucleic acid systems. *J. Comput. Chem.* **32**, 170–173 (2011).
29. Lorenz, R. et al. ViennaRNA Package 2.0. *Algorithms Mol. Biol.* **6**, 26 (2011).
30. Wei, B., Luvena, L. L., Ong, J., Jaffe, A. S. & Yin, P. Complex reconfiguration of DNA nanostructures. *Angew. Chem. Int. Ed.* **126**, 7605–7609 (2014).
31. Chalk, C., Fu, B., Martinez, E., Schweller, R. & Wylie, T. Concentration independent random number generation in tile self-assembly. *Theor. Comput. Sci.* **667**, 1–15 (2017).
32. Jacobs, W. M., Reinhardt, A. & Frenkel, D. Rational design of self-assembly pathways for complex multicomponent structures. *Proc. Natl Acad. Sci. USA* **112**, 6313–6318 (2015).
33. Hedges, L. O., Mannige, R. V. & Whitelam, S. Growth of equilibrium structures built from a large number of distinct component types. *Soft Matter* **10**, 6404–6416 (2014).
34. Cairns-Smith, A. G. *Genetic Takeover and the Mineral Origins of Life* (Cambridge Univ. Press, Cambridge, 1982).

Acknowledgements We thank C. Evans, A. Gopinath, B. Wei, C. Geary, R. Schulman, S. Woo, P. Rothemund and Y. Rondelez for experimental advice; R. Barish and R. Hariadi for contributing to preliminary designs for algorithmic self-assembly by SST; C. Moore, T. Stérin, C. Thachuk, P.-É. Meunier and C. Geary for discussions on theory; and L. Qian, G. Tikhomirov and P. Petersen for AFM usage. This work was supported by National Science Foundation (NSF) grants CCF-1162589 (to E.W., D.D. and D.W.), CCF-1162459 (to P.Y.), CCF-1219274 (to D.W. and D.D.), CCF-1619343 (to D.D.), CCF-0832824 and CCF-1317694 (Expeditions in Computing, to E.W.) and CCF-1317291 (Expeditions in Computing, to P.Y.), and by NASA grant NNX13AJ56G (to D.W.). C.M. was funded by the Fannie and John Hertz Foundation. F.Z. and J.H. received support from the Caltech Summer Undergraduate Research Fellowship program.

Author contributions D.W., D.D., E.W. and P.Y. conceived the study. D.W., D.D. and E.W. designed the circuits and wrote the manuscript. D.W. and D.D. carried out all data analysis and experiments reported except for the nanotube nucleation/melt experiments (which were performed by J.H. and D.W.) and the unzipping and other early experimental protocols (performed by F.Z., C.M. and D.D.).

Competing interests D.W., D.D., J.H., F.Z. and E.W. declare that they have no competing interests. P.Y. and C.M. declare competing interests: they are both listed as inventors on pending and issued patents on single-stranded tiles; and P.Y. is a co-founder of Ultivue Inc. and NuProbe Global.

Additional information
Supplementary information is available for this paper at https://doi.org/10.1038/s41586-019-1014-9.
Reprints and permissions information is available at http://www.nature.com/reprints.
Correspondence and requests for materials should be addressed to D.W., D.D. or E.W.
Publisher's note: Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.