Temperature 1 Self-Assembly:
Deterministic Assembly in 3D and Probabilistic Assembly in 2D

Matthew Cook[*]   Yunhui Fu[†]   Robert Schweller[‡]

Abstract

We investigate the power of the Wang tile self-assembly model at temperature 1, a threshold value that permits attachment between any two tiles that share even a single bond. When restricted to deterministic assembly in the plane, no temperature 1 assembly system has been shown to build a shape with a tile complexity smaller than the diameter of the shape. In contrast, we show that temperature 1 self-assembly in 3 dimensions, even when growth is restricted to at most 1 step into the third dimension, is capable of simulating a large class of temperature 2 systems, in turn permitting the simulation of arbitrary Turing machines and the assembly of $n \times n$ squares in near optimal $O(\log n)$ tile complexity. Further, we consider temperature 1 probabilistic assembly in 2D, and show that with a logarithmic scale up of tile complexity and shape scale, the same general class of temperature $\tau = 2$ systems can be simulated, yielding Turing machine simulation and $O(\log^2 n)$ assembly of $n \times n$ squares with high probability. Our results show a sharp contrast in achievable tile complexity at temperature 1 if either growth into the third dimension or a small probability of error are permitted. Motivated by applications in nanotechnology and molecular computing, and the plausibility of implementing 3 dimensional self-assembly systems, our techniques may provide the needed power of temperature 2 systems, while at the same time avoiding the experimental challenges faced by those systems.

1 Introduction

Self-assembly is the process by which simple objects autonomously assemble into an organized structure. This phenomenon is the driving force for the creation of complex biological organisms, and is emerging as a powerful tool for bottom up fabrication of complex nanostructures. One of the most fruitful classes of self-assembly systems is DNA self-assembly. The ability to synthesize DNA strands with specific base sequences permits a highly reliable technique for programming strands to assemble into specific structures. In particular, molecular building blocks or *tiles* can be assembled with distinct bonding domains [14;20]. These DNA tiles can be designed to simulate the theoretical bonding behavior of the *Tile Assembly Model* [28].

In the Tile Assembly Model, the particles of a self-assembly system are modeled by four-sided Wang tiles for 2D assembly, or 6 sided Wang cubes in 3D. Each side of a tile represents a distinct binding domain and has a specific glue type associated with it. Starting from an initial *seed* tile, assembly takes place by attaching copies of different tile types in the system to the growing seed assembly one by one. The attachment is driven by the affinities of the glue types. In particular, a tile type may attach to a growing seed assembly if the total binding strength from all glues abutting the seed assembly exceeds some given parameter called the *temperature*. If the assembly process reaches a point when no more attachments are possible, the produced assembly is denoted *terminal* and is considered the output assembly of the system.

Motivated by bottom-up nanofabrication of complex devices and molecular computing, a number of fundamental problems in the tile assembly model have been considered [5;9;22;24;25;29;36]. A few problems are (1) *shape fabrication*: given a target shape Υ, design a system of tile types that will uniquely assemble into shape Υ that uses as few distinct tile types as possible; 2) *molecular computing* [6;34]: given some input assembly that encodes a description of a computational problem, design a tile system that will read this input and assemble a structure that encodes the solution to the computational problem; (3) *shape replication* [1;30], given a single copy of a preassembled input shape or pattern, efficiently create a number of replicas of the input shape or pattern.

While a great body of work has emerged in recent years considering problems in the tile assembly model, almost all of this work has focussed on temperature 2 assembly in which tiles require 2 separate positive strength glue bonds to attach to the growing seed assembly. This is in contrast to the simpler temperature 1 model which permits attachment given any positive strength bond. It seems that some fundamental increase in computational power and efficiency is achieved by making the step from temperature 1 to temperature 2. In fact, there is no known 2D construction to deterministically assemble a width n

[*]Institute of Neuroinformatics, University of Zurich and ETH Zurich, Switzerland, cook@ini.phys.ethz.ch

[†]Department of Computer Science, University of Texas - Pan American, fuyunhui@gmail.com

[‡]Department of Computer Science, University of Texas - Pan American, schwellerr@cs.panam.edu

Table 1: *In this table we summarize the state of the art in achievable tile complexities and computational power for tile self-assembly in terms of temperature 1 versus temperature 2 assembly, 2-dimensional versus 3-dimensional assembly, and deterministic versus probabilistic assembly. Our contributions are contained in rows 2 and 3.*

	$n \times n$ **Squares**		**Computational Power**
	LB	UB	
Temperature 2, $2D$ Deterministic	$\Theta(\frac{\log n}{\log \log n})$ (see [28])	(see [2])	Universal (see [34])
Temperature 1, $3D$ Deterministic	$\Omega(\frac{\log n}{\log \log n})$ (see [28])	$O(\log n)$ (Thm.4.1)	Universal (Thm.B.1)
Temperature 1, $2D$ Probabilistic	$\Omega(\frac{\log n}{\log \log n})$ (Thm.4.3)	$O(\log^2 n)$ (Thm.4.2)	Time Bounded Turing Simulation (Thm.B.2)
Temperature 1, $2D$ Deterministic	$\Omega(\frac{\log n}{\log \log n})$ (see [28])	$2n - 1$ (see [28])	Unknown

shape in fewer that n distinct tiles types at temperature 1. This is in contrast to efficient temperature 2 systems which assemble large classes of shapes efficiently, including $n \times n$ squares in optimal $\theta(\frac{\log n}{\log \log n})$ tile types. In fact, the ability to simulate universal computation and assemble arbitrary shapes in a number of tile types close to the Kolmogorov complexity of a given shape at temperature 2 in 2D[32] has resulted in limited interest in exploring 3D assembly, as it would appear no substantial power would be gained in the extra dimension.

While temperature 2 assembly yields very efficient, powerful constructions, it is not without its drawbacks. One of the main hurdles preventing large scale implementation of complex self-assembly systems is high error rates. The primary cause of these errors in DNA self-assembly systems stems from the problem of insufficient attachments of tile types[4;8;23;35]. That is, in practice tiles often attach with less than strength 2 bonding despite carefully specified lab settings meant to prevent such insufficient bonds. Inherently, this is a problem specific only to temperature 2 and higher systems. For this reason, development of temperature 1 theory may prove to be of great practical interest. Put another way, if self-assembly is viewed as a model of crystal growth[34], temperature 2 assembly models growth at or near the melting temperature of the crystal. Such a scenario is ideal for growing large, complex crystals, but is extremely hard to maintain in a laboratory setting. In contrast, temperature 1 models crystal growth well below the melting temperature of the crystal, a far too easy scenario to maintain in which arbitrary, uncontrolled growth can occur.

Because of a perceived lack of power, temperature 1 assembly has received little attention compared to the more powerful temperature 2 assembly. In addition, directions such as 3D assembly have not received substantial attention stemming from a perceived lack of ability to increase the functionality of the already powerful temperature 2 systems. Interestingly, we find that both directions are fruitful when considered together; temperature 1 assembly systems in 3D are nearly as powerful as temperature 2 systems, suggesting that both the perception of limited temperature 1 power and the perception of limited value in 3D are not completely accurate.

Our Results. In this paper we show that temperature 1 deterministic tile assembly systems in 3D can simulate a large class of temperature 2 systems we call *zig-zag systems*. We further show that this simulation grants both: (1) near optimal $O(\log n)$ tile type efficiency for the assembly of $n \times n$ squares and (2) universal computational power. Further, in the case of 2D probabilistic assembly, we show similar results hold by achieving $O(\log^2 n)$ efficient square assembly and the ability to efficiently simulate any time bounded Turing machine with arbitrarily small chance of error. The key technique utilized in our constructions is a method of limiting glue attachment by creating geometrical barriers of growth that prevent undesired attachments from propagating. This technique is well known in the field of chemistry as *steric hindrance* or *steric protection*[16–18;33] where a chemical reaction is slowed or stopped by the arrangement of atoms in a molecule. These results show that temperature 1 assembly is not as limited as it appears at first consideration, and perhaps such assemblies warrant more consideration in light of the potential practical advantages of temperature 1 self-assembly in DNA implementations. Our results are summarized in Table 1.

Practical Drawbacks of Temperature 1 Self-Assembly. While temperature 1 assembly avoids many of the practical hurdles limiting temperature 2 assembly, temperature 1 assembly also introduces new issues. In particular, the problem of multiple nucleation, in which tiles begin to grow without the presence of the seed tile, is a more substantial problem at temperature 1. However, further research into temperature 1 assembly may suggest and motivate new techniques to limit such errors. In the specific case of multiple nucleation, we discuss in this paper as future work a few possible directions and techniques for temperature 1 self-assembly to limit such errors, even in a pure temperature 1 assembly model. By fully exploring techniques such as this, and any new techniques that may emerge, temperature 1 self-assembly may emerge as a practical alternative to temperature 2 assembly.

Related Work. Some recent work has been done in the area of proving lower bounds for temperature 1 self-assembly. Doty et al[12] show a limit to the computational power of temperature 1 self-assembly for *pumpable* systems. Munich et al[26] show that temperature 1 assembly of a shape requires at least as many tile types as the diameter of the assembled shape if no mismatched glues are permitted. There are a number of works achieving positive results involving temperature 1 assembly, probabilistic assembly, and steric hindrance techniques. Chandran et al[7] consider the probabilistic assembly of lines with expected length n (at temperature 1) and achieve $O(\log n)$ tile complexity. Kao and Schweller[19] and Doty[13] use a variant of probabilistic self-assembly (at temperature 2) to reduce distinct tile type complexity. Demaine et al[10] and Abel et al[1] utilize steric hindrance to assist in the assembly and replication of shapes over a number of stages.

Paper Layout. In Section 2 we define the Tile Assembly Model. In Section 3 we describe an algorithm to convert a temperature 2 *zig zag* system into an equivalent temperature 1 3D system, or a probabilistic 2D system. In Section 4 we show how temperature 1 systems can efficiently assemble $n \times n$ squares. In Section 5 we show that temperature 1 systems can simulate arbitrary Turing machines. In Section 6 we discuss preliminary experimental simulations. In Section 7 we discuss further research directions.

2 Basics

2.1 Definitions: the Abstract Tile Assembly Model in 2 Dimensions.
To describe the tile self-assembly model, we make the following definitions. A tile type t in the model is a four sided Wang tile in which each tile face is assigned a *glue type* from some alphabet of glues Σ. For a tile type t, let north(t) denote the glue type on the north face of t. Define east(t), south(t), and west(t)

analogously. Each pair of glue types are assigned a non-negative integer bonding strength (0,1, or 2 in this paper) by the *glue function* $G : \Sigma^2 \rightarrow \{0, 1, \ldots\}$. It is assumed that $G(x, y) = G(y, x)$, and there exists a null in Σ such that $\forall x \in \Sigma$, $G(\texttt{null}, \texttt{x}) = 0$. In this paper we assume the glue function is such that $G(x, y) = 0$ when $x \neq y$ and denote $G(x, x)$ by $G(x)$.

A *tile system* is an ordered triple $\langle T, s, \tau \rangle$ where T is a set of tiles called the *tileset* of the system, τ is a positive integer called the *temperature* of the system and $s \in T$ is a single tile called the *seed* tile. $|T|$ is referred to as the *tile complexity* of the system. In this paper we only consider temperature $\tau = 1$ and $\tau = 2$ systems.

Define a *configuration* to be a mapping from $\mathbb{Z}^2$ to $T \cup \{\texttt{empty}\}$, where empty is a special tile type that has the null glue on each of its four edges. The *shape* of a configuration is defined as the set of positions (i, j) that do not map to the empty tile. For a configuration C, a tile type $t \in T$ is said to be *attachable* at the position (i, j) if $C(i, j) = \texttt{empty}$ and $G(\text{north}(t), \text{south}(C(i, j + 1))) + G(\text{east}(t), \text{west}(C(i+1, j))) + G(\text{south}(t), \text{north}(C(i, j-1))) + G(\text{west}(t), \text{east}(C(i - 1, j))) \geq \tau$. For configurations C and C' such that $C(x, y) = \texttt{empty}$, $C'(i, j) = C(i, j)$ for all $(i, j) \neq (x, y)$, and $C'(x, y) = t$ for some $t \in T$, define the act of *attaching* tile t to C at position (x, y) as the transformation from configuration C to C'. For a given tile system **T**, if a configuration B can be obtained from a configuration A by the addition of a single tile we write $A \rightarrow_T B$. Further, we denote $A \rightarrow_T$ as the set of all B such that $A \rightarrow_T B$ and $\rightarrow_T^*$ as the transitive closure of $\rightarrow_T$.

Define the *adjacency graph* of a configuration C as follows. Let the set of vertices be the set of coordinates (i, j) such that $C(i, j)$ is not empty. Let there be an edge between vertices (x_1, y_1) and (x_2, y_2) iff $|x_1 - x_2| + |y_1 - y_2| = 1$. We refer to a configuration whose adjacency graph is finite and connected as a *supertile*. For a supertile S, denote the number of non-empty positions (tiles) in the supertile by size(S). We also note that each tile type $t \in T$ can be thought of as denoting the unique supertile that maps position $(0, 0)$ to t and all other positions to empty. Throughout this paper we will informally refer to tiles as being supertiles.

2.2 The Assembly Process
Deterministic Assembly. Assembly takes place by *growing* a supertile starting with tile s at position $(0, 0)$. Any $t \in T$ that is attachable at some position (i, j) may attach and thus increase the size of the supertile. For a given tile system, any supertile that can be obtained by starting with the seed and attaching arbitrary attachable tiles is said to be *produced*. If this process comes to a point at which no tiles in T can be added, the resultant supertile is said to be *terminally* produced. For a given

shape Υ, a tile system Γ *uniquely produces* shape Υ if for each produced supertile A, there exists some terminally produced supertile A' of shape Υ such that $A \rightarrow_T^* A'$. That is, each produced supertile can be grown into a supertile of shape Υ. The *tile complexity* of a shape Υ is the minimum tile set size required to uniquely assemble Υ. For an assembly system which uniquely assembles one supertile, the system is said to be *deterministic*.

Probabilistic Assembly. For non-deterministic assembly systems, we define the probabilistic assembly model to place probability distributions on which tiles attach throughout the assembly process. To study this model we can think of the assembly process as a Markov chain where each producible supertile is a state and transitions occur with non-zero probability from supertile A to each $B \in A \rightarrow_T$. For each $B \in A \rightarrow_T$, let t_B denote the tile added to A to get B. The transition probability from A to B is defined to be

$$\mathrm{TRANS}(A, B) = \frac{1}{|A \rightarrow_T|}.$$

The probability that a tile system T terminally assembles a supertile A is thus defined to be the probability that the Markov chain ends in state A. Further, the probability that a system terminally assembles a shape Υ is the probability that the chain ends in a supertile state of shape Υ.

2.3 Extension to 3 Dimensions.
Extending the model, we can consider assembly in 3 dimensions by considering Wang cubes with added "up" and "down" glues ($\mathrm{up}(t)$ and $\mathrm{down}(t)$), and configurations of tile types as mappings from Z^3 to a tile set T. In this model, assembly begins as before with an initial seed cube (informally we will refer to cubes as tiles) at the origin, with cubes attaching to the north, west, south, east, top, or bottom of already placed tiles if the total strength of attachment from the glue function meets or exceeds the temperature threshold τ.

In this paper we only consider temperature $\tau = 1$ assembly systems in 3D. Further, we only consider systems that assemble tiles within the $z = 0$ or $z = 1$ plane. To depict 3 dimensional tile sets and 3 dimensional assemblies, we introduce some new notation in Figure 1.

2.4 Scaled Simulations.
Our constructions in this paper relate to algorithms that generate temperature 1 tile systems that simulate the assembly of a given temperature 2 tile system.

Informally, a system Υ simulates a system Γ if for each tile type t placed by system Γ, a tile type representing t is placed by Υ at the same position but scaled up by some fixed vertical and horizontal *scale factors*. In more detail, the simulating tileset should
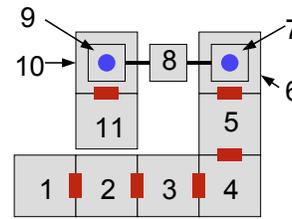

Figure 1: *For 3 dimensional tile assemblies, we create figures with the notation depicted above. First, large tiles denote tiles placed in the $z = 0$ plane, while the smaller squares denote tiles placed in the $z = 1$ plane. The red connectors between bottom tiles denotes some unique glue shared by the tiles in the figure, as do the thin black lines connecting tiles in the top plane. Blue circles denote a unique glue connecting the bottom of the top tile with the top of the tile below it. In this example, the tiles are numbered showing the implied order of attachment of tiles assuming the '1' tile is a seed tile.*

have a special set of tile types R which act as "labels" to denote which tile type from the original set is being represented by a given scaled up "macro block" of tile types. Formally, we define the notion of one tile system simulating another system as follows.

DEFINITION 1. *A tile system $\Upsilon = (T', s', \tau')$ simulates a deterministic tile system $\Gamma = (T, s, \tau)$ at horizontal scale factor $scale_x$, vertical scale factor $scale_y$, and tile complexity factor C if:*

1. *there exists an onto function $SIM : R \subseteq T' \Rightarrow T$ for some subset $R \subseteq T'$ such that for any non empty tile type $\Gamma(i, j)$ at position (i, j) in the terminal assembly of Γ, there exists exactly one tile type r in the set R such that $\Upsilon(k, \ell) = r$ for some k and ℓ such that $i \cdot scale_x \leq k < (i + 1)scale_x$ and $j \cdot scale_y \leq \ell < (j + 1)scale_y$, and $SIM(r) = \Gamma(i, j)$.*

2. *for each position (i, j) such that $\Gamma(i, j) = empty$, it is the case that for each k and ℓ such that $i \cdot scale_x \leq k \leq (i + 1)scale_x$ and $j \cdot scale_y \leq \ell \leq (j + 1)scale_y$, $\Upsilon(k, \ell) = empty$.*

3. $|T'| \leq C|T|$

In the case that the simulating system is a 3D system, all tile positions from the 3D system are projected on the $z = 0$ plane to apply the definition. In the case that Υ is a probabilistic assembly system, the tile placed at a given position by Υ is a random variable, and we are interested in the probability that each tile placed by Γ is correctly simulated by Υ according to some given assignment of representative tile types.

## 3 Simulation of Temperature $\tau = 2$ Zig-Zag Systems at Temperature $\tau = 1$
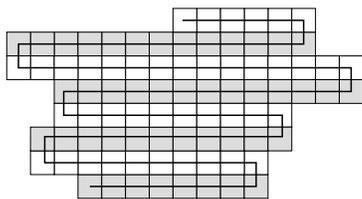
Figure 2: *A zig-zag tile system alternates growth from left to right at each vertical level.*

3.1 Zig-Zag Tile Systems.

Our temperature $\tau = 1$ results, which make use of either the third dimension or probabilistic assembly in 2D, are derived by simulating a large class of deterministic 2D temperature $\tau = 2$ systems we refer to as "zig-zag" systems.

Zig-zag assembly systems. Informally, a zig-zag assembly is an assembly which must place tiles from left to right up to the point at which a first tile is placed into the next row north. At this point growth must grow from right to left. Thus, even rows grow from left to right, and odd rows grow from right to left, as shown in Figure 2. Formally, an assembly system is defined to be a zig-zag system if it has the following two properties:

- The assembly sequence, which specifies the order in which tile types are attached to the assembly along with their position of attachment, is unique. For such systems denote the type of the ith tile to be attached to the assembly as $type(i)$, and denote the coordinate location of attachment of the ith attached tile by $(x(i), y(i))$.

- For each i in the assembly sequence of a zig-zag system, if $y(i-1)$ is even, then either $x(i) = x(i-1) + 1$ and $y(i-1) = y(i)$, or $y(i) = y(i-1) + 1$. For odd $y(i-1)$, either $x(i) = x(i-1) - 1$ and $y(i-1)$, or $y(i) = y(i-1) + 1$.

The key property zig-zag systems have is that it is not necessary for a west or east strength 1 glue to be exposed without the position directly south of the open slot having already been tiled. For any zig-zag system that did expose such a glue, the glue would be redundant and could be removed to achieve the same assembly sequence.

3.2 3D Temperature 1 Simulation of Zig-Zag Tile Systems.

In this section we show that any temperature 2 zig-zag tile system can be simulated efficiently by a temperature 1 3D tile system. The scale and efficiency of the simulation is dependant on the number of distinct strength 1 glues that occur on either the north or south edge of a tile type in the input tile system's tileset T. For a tileset T, let σ_T denote the set of strength 1 glue types that occur on either a north or south face of some tile type in T. The following simulation is possible.

THEOREM 3.1. *For any 2 dimensional temperature 2 zig-zag tile system $\Gamma = \langle T, s, 2 \rangle$ with σ_T denoting the set of distinct strength 1 north/south glue types occuring in T, there exists a 3D temperature 1 tile system that simulates Γ at vertical scale = 4, horizontal scale = $O(\log |\sigma_T|)$, and (total) tile complexity $O(|T| \log |\sigma_T|)$, which constitutes a $O(\log |\sigma_T|)$ tile complexity scale factor.*

Proof. To simulate a temperature 2 zig-zag tileset with a temperature 1 tileset, we replace each tile type in the original system with a collection of tiles designed to assemble a larger *macro tile* at temperature 1. The goal will be to "fake" cooperative temperature 2 attachment of a given tile from the original system by utilizing geometry to force an otherwise non-deterministic growth of tiles to *read* the north input glue of a given macro tile.

Consider an arbitrary temperature 2 zig-zag tile system $\Gamma = \langle T, s, 2 \rangle$ where σ_T denotes the alphabet of strength 1 glue types that occur on either the north or south face of some tile type in T. Index each glue $g \in \sigma_T$ from 0 to $|\sigma_T| - 1$. For a given $g \in \sigma_T$, let $b(g)$ denote the index value of g written in binary. We will refer to each glue g by the new name $c_{b(g)}$ when describing our construction.

To prove the theorem, we construct a tileset with $O(\log |\sigma_T|)$ distinct tile types for each tile $t \in T$. In particular, the number of tile types will be at most $|T|(12 \cdot \log |\sigma_T| + 12) = O(|T| \log |\sigma_T|)$.

To assign tile types to the new temperature 1 assembly system, take all west growing tile types that have an east glue of type 'x' for some strength 1 glue 'x'. As a running example, see Figure 3 (a) for such a set of tile types. If there happens to be both east growing and west growing tile types that have glue 'x' as an east glue, first split all such tile types in to two separate tiles, one for each direction.

For the set of west growing tile types with east glue x, a collection of tile types are added to the simulation set as a function of x and the subset of σ_T glues that appear on the south face of the collected tile types. The tile types added are depicted in Figure 3. In the example from Figure 3 there are 4 distinct tile types that share an east x glue type. As these tiles are west growing tile types, the temperature 2 simulation places each of these tiles using the cooperative bonding of glue x and glue c_{111} in the case of the rightmost tile type. The east and south glue types of a west growing tile can be thought of as *input glues*, which uniquely specify which tile type is placed, in turn specifying two *output glues*, glue type a to the west and glue type c_{111} to the north in the case of the rightmost

tile type. At temperature 1, we cannot directly implement this double input function by cooperative bonding as even a single glue type is sufficient to place a tile. Instead, we use glue type to encode the east input, and geometry of previously assembled tiles to encode the south input.

In more detail, the tile types specified in Figure 3 (b) constitute a nondeterministic chain of tiles whose possible assembly paths form a binary tree of depth $\log$ of the number of distinct south glue types observed in the tile set being simulated. In the given example, the tree starts with an input glue $(x, -)$. This glue knows the tile to its east has a west glue of type x, but has no encoding of what glue type is to the south of the macro tile to be placed. This chain of tiles nondeterministically places either a 0 or a 1 tile, which in turn continues growth along two separate possible paths, one denoted by glue type $(x, 0)$, and the other by glue type $(x, 1)$. By explicitly encoding all paths of a binary prefix tree ending with leaves for each of the south glues of the input tile types, the decoding tiles nondeterministically pick exactly one south glue type to pair with the east glue type x, and output this value as a glue specifying which of the 4 tile types should be simulated at this position.

Now, to eliminate the non-determinism in the decoding tiles, we ensure that the geometry of previously placed tiles in the $z = 1$ plane is such that at each possible branching point in the binary tree chain, exactly 1 path is blocked, thus removing the non-determinism in the assembly as depicted in Figure 3 (d). This prebuilt geometry is guaranteed to be in place by the correct placement of the simulated macro tile placed south of the current macro tile. Once the proper tile type to be simulated is decoded, the 2 output values, a and c_{111} in the case of the rightmost tile type of Figure 3 (a), must be propagated west and north respectively. This is accomplished by the collection of tile types depicted in Figure 3 (c). Now that the north and west output glues have been decoded, this macro tile will assemble a geometry of *blocking tiles* to ensure that a tile using this north glue as a south glue input will deterministically decode the correct glue binary string. In particular, pairs of tiles are placed in the plane $z = 2$ for each bit of the output binary string. The pair is placed to locations vertically higher for 1 bits than for 0 bits. The next row of macro tiles will then be able to decode this glue type encoded in geometry by applying a binary tree of decoder tiles similar to those shown in Figure 3 (b).

The complete conversion algorithm from a temperature 2 zig-zag system to a temperature 1 3D system has a large number of special cases. However, the example worked out in this proof sketch gets at the heart of the idea. The fully detailed conversion algorithm for all cases is described in the Appendix in Section C.1. Additionally, a fully detailed example of the conversion of a tempera-

ture $\tau = 2$ unary counter into a 3D temperature $\tau = 1$ system is described in Section A. Further, we have implemented automatic conversion software for any zig-zag system which is available upon request.

3.3 2D Probabilistic Temperature 1 Simulation of Zig-Zag Tile Systems.

In this section we sketch out the simulation of any temperature 2 zig-zag tile system by a 2D probabilistic tile system. The basic idea is similar to the 3D conversion. Each tile type for a given east glue x is used to generate a binary tree of nondeterministic chains of tiles to decode a southern input glue encoded by geometry. However, in 2D, it is not possible to block both branches of the binary tree due to planarity. Therefore, we only block one side or block neither side. The basic idea behind the decoding is depicted in Figure 4.

The key idea is to buffer the length of the geometric blocks encoding bits by a buffering factor of k. In the example from Figure 4, a 2-bit string is encoded by two length $2k$ stretches of tiles (k=4 in this example), where the binary bit value 1 is represented by a dent that is one vertical position higher than the encoding for the binary bit value 0. Whereas in the 3D encoding each bit value is encoded with a fixed two horizontal tile position for encoding, we now have an encoding region of size $2k$ for each bit value.

As in the 3D case, we utilize a set of decoder tiles (the orange tiles in Figure 4) to read the geometry of the north face of the macro tile to the south. Due to planarity, only one possible growth branch is blocked. However, in the case of a 0 bit, the k repetitions of the non-deterministic placement of the branching tile makes it highly likely that at least one downward growth will beat its competing westward growth over k independent trials, if k is large. In the case of 1 bits, the decoder is guaranteed to get the correct answer.

Therefore, we can analyze the probability that a given 0 bit is correctly decoded by bounding the probability of flipping a (biased) coin k times and never getting a single tail. The coin is biased because the south branch of growth must place 3 consecutive tiles before the west branch places a single tile to successfully read the 0 bit, which happens with probability $1/8$[1]. Thus, by making k sufficiently large, we can bound the probability that a given block will make an error by incorrectly interpreting a 0-bit as a 1-bit. For a zig-zag system that makes r tile attachments, we can therefore set k to be large enough such that with high probability a temperature 1 simulation will make all r attachments without error, yielding the following theorem.

---

[1]To make the choice non-biased, multiple copies of the south branching tile types could be included in the tile set, making the 3 consecutive placements happen with equal, or even greater probability than the single tile placement of the west branch.
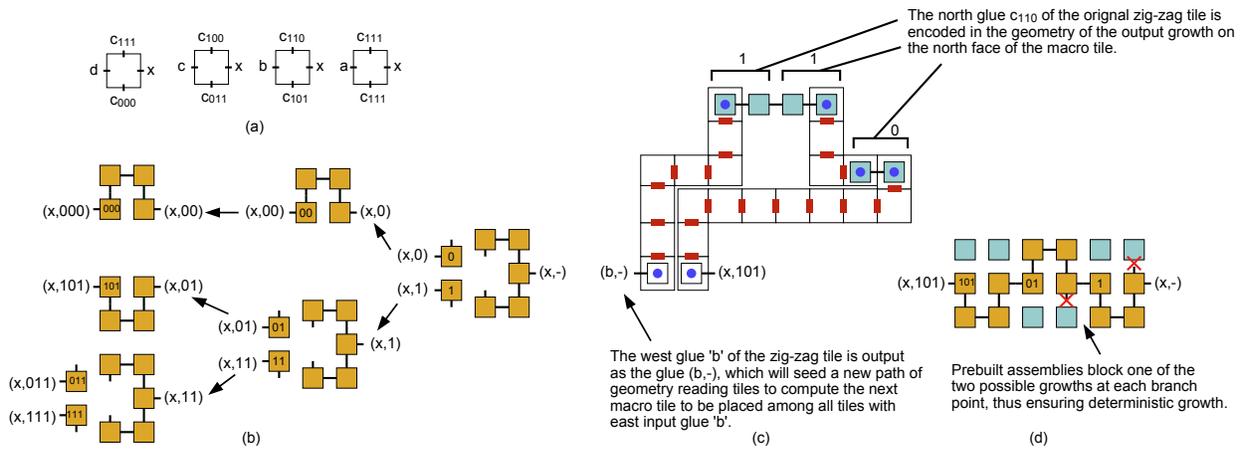
Figure 3: *This tile set depicts how one tile from a collection of input tile types all sharing the same east glue x are simulated by combining strength 1 glue bonds with geometrical blocking to simulate cooperative binding and correctly place the correct macro tile.*

THEOREM 3.2. *For any $\epsilon > 0$ and zig-zag tile system $\Gamma = \langle T, s, 2 \rangle$ whose terminal assembly has size r, there exists a temperature 1, 2D probabilistic tile system that simulates Γ without error with probability at least $1 - \epsilon$. The $scale_x$ of this system is $O(\log \sigma_T \log r + \log \sigma_T \log \frac{1}{\epsilon})$, the $scale_y$ is 4, and the tile complexity is $O(|T| \log \sigma_T \log r + |T| \log \sigma_T \log \frac{1}{\epsilon})$.*

Proof. We first note that the tile complexity and scale factor of the simulation are determined by the values $|T|$ and $|\sigma_T|$, as in the 3D transformation, as well as the choice of the parameter k. In particular, the tile complexity of the system is $O(|T| k \log \sigma_T)$, the horizontal scale factor is $scale_x = O(k \log \sigma_T)$, and the vertical scale factor is $scale_y = 4$. Therefore, to show the result we need to determine what parameter choice for k is needed to guarantee that all r block placements are error free with probability at least $1 - \epsilon$.

For a given $\epsilon > 0$, set k equal to

$$k = \log_{8/7} r + \log_{8/7} \frac{1}{\epsilon}$$

As any given block will make an error with probability at most $\left(\frac{7}{8}\right)^k$, the expected number of total errors over r block placements is

$$E[errors] = r\left(\frac{7}{8}\right)^k = \epsilon$$

Therefore, for $k = \log_{8/7} r + \log_{8/7} \frac{1}{\epsilon}$ the probability of making 1 or more errors is bounded by ϵ, and this value of k achieves the theorem's stated bounds.

4 Efficient Assembly of $n \times n$ Squares at Temperature 1

In this section we show how to apply the zig-zag simulation approach from Section 3 to achieve tile complexity efficient systems that uniquely assemble $n \times n$ squares at temperature $\tau = 1$ either deterministically in 3D or probabilistically in 2D.

4.1 Deterministic Temperature 1 assembly of $n \times n$ Squares in 3D.
We show that the assembly of an $n \times n$ square with a 3D tile system requires at most $O(\log n)$ tile complexity at temperature 1.

LEMMA 4.1. *For any n, there exists a 2D, temperature $\tau = 2$ zig-zag tile system, with north/south glue set σ of constant bounded size, that uniquely assembles a $\log n \times n$ rectangle. Further, this rectangle can be designed so that a unique, unused glue appears on the east side of the northeast placed tile (this allows the completed rectangle to seed a new assembly, as utilized in Theorem 4.1).*

Proof. The binary counter described by Rothemund and Winfree[28] is a zig-zag tile set and can easily be modified to have a special glue at the needed position.

THEOREM 4.1. *For any $n \geq 1$, there exists a 3D temperature $\tau = 1$ tile system with tile complexity $O(\log n)$ that uniquely assembles an $n \times n$ square.*

Proof. The construction to assemble an $n \times n$ square uses a system that is not zig-zag itself, but is a combination of 3 zig-zag systems (binary counting systems) that build the west, north and east face of the square, and a single filler tile to fill in the cavity of the square.

The basis of our proof is a construction for a binary counter that is a zig-zag system with a $O(1)$ size north/south glue set. Such a tile system can assemble a $\log n$ by n rectangle using $O(\log n)$ tile types. By Theorem 3.1, this system can be simulated at temperature 1 in 3D to make 1 side of an $n \times n$ square. It is straightforward to design the counter system in such a way that the
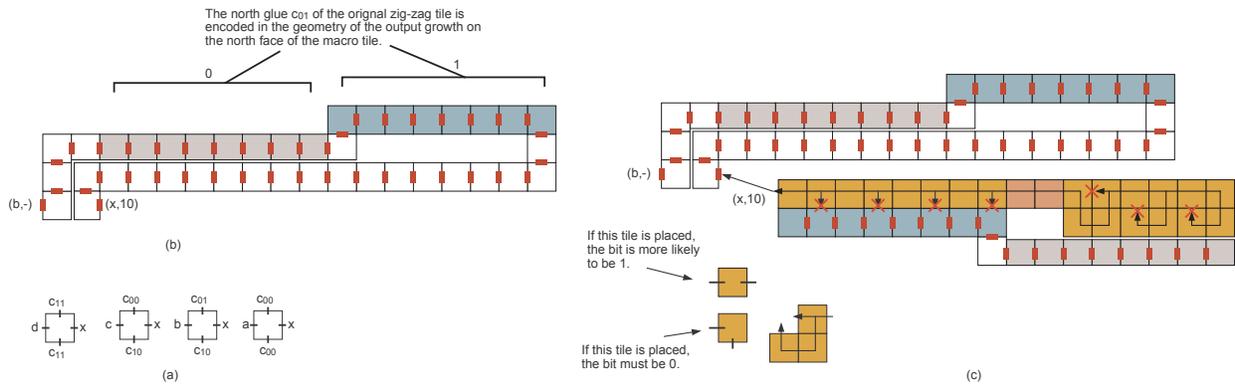
Figure 4: *This tile set depicts the macro tiles used to transform a zig-zag system into a probabilistic 2D assembly system.*

final placed tile seeds a similar binary counter upon the completion of the initial assembly. The second counter then grows east. Upon completion, the final placed tile seeds a final binary counter that grows south, thereby completing the west, north and east faces of the square. The third counter additionally contains a special glue on all exposed leftmost tiles that causes an infinite repeating westward growth consisting of a single tile type. As such tiles cannot be placed until the first counter is completed, this infinite growth will be blocked by the first counter and simply fill in the body of the square.

As the north/south glue set for the counter systems is of constant size, the same $O(\log n)$ tile complexity is achieved for the construction of the square at temperature 1 in 3D by combining Lemma 4.1 with Theorem 3.1.

4.2 Probabilistic Temperature 1 assembly of $n \times n$ Squares in 2D.
In this section we show that for any value ϵ and positive integer n, there exists a 2D tile system with $O(\log^2 n + \log n \log \frac{1}{\epsilon})$ tile types that will assemble an $n \times n$ square with probability at least $1 - \epsilon$. For a fixed constant ϵ, this simplifies to a $O(\log^2 n)$ tile complexity.

THEOREM 4.2. *For any $\epsilon > 0$ and positive integer n, there exists a 2D tile system with $O(\log^2 n + \log n \log \frac{1}{\epsilon})$ tile types that will assemble an $n \times n$ square with probability at least $1 - \epsilon$.*

Proof. The basic approach of this theorem is the same as Theorem 4.1. Consider a zig-zag binary counter construction with $O(1)$ size north/south glue set designed to assemble a $\log n$ by n rectangle using $O(\log n)$ tile types. We can apply Theorem 3.2 to this system to get a temperature 1 probabilistic simulation of the counter. Further, as the counter places at most $O(n \log n)$ tile types before terminating, Theorem 3.2 yields a tile complexity bound of $O(\log^2 n + \log n \log \frac{1}{\epsilon})$ to correctly simulate the counter with probability $1 - \epsilon$. As with the 3D construction, 3 rectangles can be assembled in sequence to build 3 of the

borders of the goal $n \times n$ square, and finally filled with a fill tile type.

4.3 Kolmogorov Lower Bound for Probabilistic Assembly of $n \times n$ Squares.
Rothemund and Winfree[28] showed that the minimum tile complexity for the deterministic self-assembly of an $n \times n$ square is lower bounded by $\Omega(\frac{\log n}{\log \log n})$ for almost all positive integers n. This applies to both 2D and 3D systems. In this section we show that this lower bound also holds for probabilistic systems that assemble an $n \times n$ square with probability greater than $1/2$.

THEOREM 4.3. *For almost all positive integers n, the minimum tile complexity required to assemble an $n \times n$ square with probability greater than 1/2 is $\Omega(\frac{\log n}{\log \log n})$.*

Proof. The Kolmogorov complexity of an integer n with respect to a universal Turing machine U is $K_U(n) = \min|p|$ s.t $U(p) = b_n$ where b_n is the binary representation of n. A straightforward application of the pigeonhole principle yields that $K_U(n) \geq \lceil \log n \rceil - \Delta$ for at least $1 - (\frac{1}{2})^\Delta$ of all n (see[21] for results on Kolmogorov complexity). Thus, for any $\epsilon > 0$, $K_U(n) \geq (1 - \epsilon)\log n = \Omega(\log n)$ for almost all n.

If there exists a fixed size Turing machine that takes as input a tile system and outputs the maximum extent (i.e. width or length) of any shape that is terminally produced by the system with probability more than 1/2, then the $\Omega(\log N)$ bound applies to the number of bits required to represent any such tile system that assembles an $n \times n$ square with probability greater than 1/2, as such a system combined with the fixed size Turing machine would constitute a machine that outputs n (note that the unique output of n depends on the probability of assembly exceeding 1/2 as only one such highly probable assembly can exist in this case). Further, we can represent any tile system $\Gamma = \langle T, s, \tau \rangle$ with $O(|T|\log|n|)$ bits assuming a constant bounded temperature (there are at

most $4|T|$ glues in the system, thus all glues can be labeled uniquely with $O(|T|\log|T|)$ bits). Thus, for constants d and c we have that $d\log n \leq c|T|\log|T|$. If we assume T is the smallest possible tile set to assemble an $n \times n$ square with probability at least 1/2, we know that $c|T|\log|T| \leq |T|\log\log n$ as there is a known (deterministic) construction to build any $n \times n$ square in $O(\log n)$ tile types[28]. Thus, we have that $d\log n \leq |T|\log\log n$ which yields that the smallest number of tile types to assemble a square is $|T| = \Omega(\frac{\log n}{\log\log n})$.

The above argument requires the existence of a fixed size Turing machine/algorithm to output the dimension of any terminally produced shape assembled with probability greater than 1/2. Algorithm 1 below satisfies this criteria.

Algorithm 1: OutputDimension

Input: Tile system $\Gamma = \langle T, s, \tau \rangle$
Output: The dimension of the supertile
        terminally produced by Γ with
        probability greater than 1/2, if it exists
for $k \leftarrow 1$ **to** ∞ **do**
    Iterate over all producible k-tile assemblies of Γ ;
    if *the current assembly X is terminal* **then**
        compute the probability that Γ assembles X ;
        if *the probability exceeds 1/2* **then**
            return *the dimension of X* ;

4.4 Why not $O(\frac{\log n}{\log\log n})$ square building systems at Temperature $\tau = 1$? Temperature $\tau = 2$ assembly systems are able to assemble $n \times n$ squares in $O(\frac{\log n}{\log\log n})$ tile complexity, which meets a corresponding lower bound from Kolmogorov complexity that holds for almost all values n[28]. Further, this can be achieved with zig-zag tile systems. So, why can't we get $O(\frac{\log n}{\log\log n})$ tile complexity for temperature $\tau = 1$ assembly in 3D? One answer is, we don't know that we can't. However, the direct application of the zig-zag transformation to the temperature 2, $O(\frac{\log n}{\log\log n})$ tile complexity system does not work. In particular, the temperature $\tau = 2$ result is achieved by picking an optimal base for a counter, rather than counting in binary. In general, a $k \times n$ rectangle can be assembled by a zig-zag tile system in $O(k + n^{\frac{1}{k}})$ tile complexity by implementing a k digit, base $n^{\frac{1}{k}}$ counter[3]. However, inherent in counting base $n^{\frac{1}{k}}$, the counter uses at least $n^{\frac{1}{k}}$ north/south glue types. Thus, the zig-zag transformation yields a tile complexity of

$$(k + n^{\frac{1}{k}})\log(n^{\frac{1}{k}}) = \Omega(\log n)$$

Therefore, regardless of what base we choose for the counter, we get at least a $\Omega(\log n)$ tile complexity. The same problem exists for the base conversion scheme used in[2]. It is an interesting open question whether or not it is possible to achieve the optimal $O(\frac{\log n}{\log\log n})$ tile complexity bound at temperature $\tau = 1$.

5 Turing Machine Simulation for General Computation

By extending non zig-zag Turing machines from the literature[32] we show how to implement a zig-zag tile set to simulate the behavior of an arbitrary input Turing machine. From Theorem 3.1, we can simulate this system at temperature 1. The details of the construction are given in the Appendix in Section B.
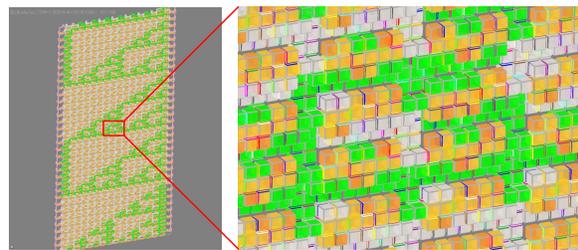

Figure 5: *Snapshot of a Temperature $\tau = 1$ Sierpinski tile system in 3D*

6 Simulation Results

We have developed automatic conversion software to generate temperature 1 3D or 2D systems given any input zig-zag tile set. We have further verified the correctness of our algorithms and converter software by implementing a number of tile systems such as Sierpinski tile sets and binary counters. We have tested the simulations on Caltech's Xgrow simulator[11], the Iowa State University TAS[27], and our own home simulator[15].

We are interested in experimentally comparing the fault tolerance of our temperature 1 probabilistic system with other fault tolerance schemes such as snaked proofreading systems. We have implemented both systems with similar tile complexity constraints to see under what settings either system would begin to make errors. We have done preliminary simulations of the 2D systems in the kTAM with Caltech's Xgrow. Our test case was a simple parity checking tile system[4], and our metric was whether or not the correct parity was computed. Our preliminary tests used the input string "1000". The block size of the snaked system is 6×6, and the length of each bit(K) of probabilistic assembly tiles is 5. The glue strength of the probabilistic assembly tiles are changed to 2.

We used $G_{mc} = 13.6$, $G_{se} = 7.0$. Both tile systems

outputted correct results. But as we increased the G_{se}, the performances of the two systems diverged. The snaked system became unstable, facet errors occurred and the tile mismatche count increased. In contrast, the probabilistic assembly tile set grew faster and the output of the system was still correct. We used the $\tau = \frac{G_{mc}}{G_{se}}$ ranging from 2 to 0.5, and the results show that the 2D probabilistic assembly system is more stable than the snaked system within the parameters we tested.

The comparison of the probabilistic and snaked systems is very preliminary because the two system use different temperatures. We adjusted the glue strength of the probabilistic system to make it comparable with the snaked system. To more accurately compare the two systems, we need to investigate other methods and test sets, both for probabilistic and snaked systems. Exploring to what extent our temperature $\tau = 1$ constructions can be utilized to achieve robustness in self-assembly is an important direction for future work.

7 Further Research Directions

There are a number of further directions relating to the work in this paper. The most glaringly open problems are related to the power of deterministic temperature 1 self-assembly in 2D. For example, it is conjectured that the optimal tile complexity for the assembly of an $n \times n$ square at temperature 1 in 2D is $2n-1$ tile types. While it is straightforward to achieve this value, the highest lower bound known is no better than the small $\Omega(\frac{\log n}{\log \log n})$ lower bound from Kolmogorov complexity. Similarly, little is known about the computational power of this model. It seems that it cannot perform sophisticated computation, but no one has yet been able to prove this.

Another direction is the concern that temperature 1 systems are prone to multiple nucleation errors in which many different growths of tiles spontaneously attach without requiring a seed. One potential solution is to design growths such that assemblies that grow apart from the seed have a high probability of *self-destructing* by yielding growths that block all further growth paths for the assembly. In contrast, properly seeded growths by design could have the self-destruct branches blocked by the seed base they are attached to. To consider such problems it may be fruitful to consider temperature 1 assembly under the *two-handed* assembly model in which there is no seed and large super tiles may attach to one another.

Another direction is the consideration of time complexity. A drawback of zig-zag assembly is a loss of parallelism as there is always only a single unique "next" attachment position. Can the fast assembly systems for temperature $\tau = 2$ be modified to work at temperature $\tau = 1$?

References

[1] Zachary Abel, Nadia Benbernou, Mirela Damian, Erik D. Demaine, Martin L. Demaine, Robin Flatland, Scott Kominers, and Robert Schweller. Shape replication through self-assembly and RNase enzymes. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2010)*, pages 1045–1064, Austin, Texas, January 17–19 2010.

[2] Leonard Adleman, Q. Cheng, A. Goel, and M. Huang. Running time and program size for self-assembled squares. In *Proceedings of the 33nd Annual ACM Symposium on Theory of Computing*, pages 740–748, 2001.

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

[4] Ho-Lin Chen Ashish, Ho lin Chen, and Ashish Goel. Error free self-assembly using error prone tiles. In *In DNA Based Computers 10*, pages 274–283, 2004.

[5] Robert D. Barish, Rebecca Schulman, Paul W. Rothemund, and Erik Winfree. An information-bearing seed for nucleating algorithmic self-assembly. *Proceedings of the National Academy of Sciences*, 106(15):6054–6059, March 2009.

[6] Yuriy Brun. Solving np-complete problems in the tile assembly model. *Theor. Comput. Sci.*, 395(1):31–46, 2008.

[7] Harish Chandran, Nikhil Gopalkrishnan, and John Reif. The tile complexity of linear assemblies. In *ICALP '09: Proceedings of the 36th International Colloquium on Automata, Languages and Programming*, pages 235–253, Berlin, Heidelberg, 2009. Springer-Verlag.

[8] Ho-Lin Chen, Ashish Goel, and Chris Luhrs. Dimension augmentation and combinatorial criteria for efficient error-resistant dna self-assembly. In *SODA '08: Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 409–418, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.

[9] Ho-Lin Chen, Rebecca Schulman, Ashish Goel, and Erik Winfree. Reducing facet nucleation during algorithmic self-assembly. *Nano Letters*, 7(9):2913–2919, September 2007.

[10] E. Demaine, M. Demaine, S. Fekete, M. Ishaque, E. Rafalin, R. Schweller, and D. Souvaine. Staged self-assembly: Nanomanufacture of arbitrary shapes with O(1) glues. In *Proceedings of the 13th International Meeting on DNA Computing*, 2007.

[11] DNA and Natural Algorithms Group. Xgrow simulator. Homepage: http://www.dna.caltech.edu/Xgrow, 2009.

[12] D. Doty, M. Patiz, and S. Summers. Limitations of self-assembly at temperature 1. In *Proceedings of the 15th DIMACS Workshop on DNA Based Computers*, 2008.

[13] David Doty. Randomized self-assembly for exact shapes. In *FOCS '09: Proceedings of the 2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 85–94, Washington, DC, USA, 2009. IEEE Computer Society.

[14] Tsu-Ju Fu and Nadrian C. Seeman. DNA double-crossover molecules. *Biochemistry*, 32:3211–3220, 1993.

[15] Yunhui Fu. Self-assembly simulation system. Homepage: http://zope.cs.panam.edu/selfassembly/projects/sss-project/, 2009.

[16] Kei Goto, Yoko Hinob, Takayuki Kawashima, Masahiro Kaminagab, Emiko Yanob, Gaku Yamamotob, Nozomi Takagic, and Shigeru Nagasec. Synthesis and crystal structure of a stable S-nitrosothiol bearing a novel steric protection group and of the corresponding S-nitrothiol. *Tetrahedron Letters*, 41(44):8479–8483, 2000.

[17] Wilfried Heller and Thomas L. Pugh. "Steric protection" of hydrophobic colloidal particles by adsorption of flexible macromolecules. *Journal of Chemical Physics*, 22(10):1778, 1954.

[18] Wilfried Heller and Thomas L. Pugh. "Steric" stabilization of colloidal solutions by adsorption of flexible macromolecules. *Journal of Polymer Science*, 47(149):203–217, 1960.

[19] Ming-Yang Kao and Robert Schweller. Randomized self-assembly for approximate shapes. In *ICALP '08: Proceedings of the 35th international colloquium on Automata, Languages and Programming, Part I*, pages 370–384, Berlin, Heidelberg, 2008. Springer-Verlag.

[20] T. H. LaBean, H. Yan, J. Kopatsch, F. Liu, E. Winfree, H. J. Reif, and N. C. Seeman. The construction, analysis, ligation and self-assembly of DNA triple crossover complexes. *J. Am. Chem. Soc.*, 122:1848–1860, 2000.

[21] M. Li and P. Vitanyi. *An Introduction to Komogorov Complexity and Its Applications (Second Edition)*. Springer Verlag, New York, 1997.

[22] Furong Liu, Ruojie Sha, and Nadrian C. Seeman. Modifying the surface features of two-dimensional DNA crystals. *Journal of the American Chemical Society*, 121(5):917–922, 1999.

[23] U. Majumder, T. H. Labean, and J. H. Reif. Activatable tiles: Compact, robust programmable assembly and other applications. In *Proceedings of the 15th DIMACS Workshop on DNA Based Computers*, pages 15–25, 2008.

[24] Chengde Mao, Thomas H. LaBean, John H. Reif, and Nadrian C. Seeman. Logical computation using algorithmic self-assembly of DNA triple-crossover molecules. *Nature*, 407(6803):493–6, 2000.

[25] Chengde Mao, Weiqiong Sun, and Nadrian C. Seeman. Designed two-dimensional DNA holliday junction arrays visualized by atomic force microscopy. *Journal of the American Chemical Society*, 121(23):5437–5443, 1999.

[26] Ján Maňuch, Ladislav Stacho, and Christine Stoll. Two lower bounds for self-assemblies at temperature 1. In *SAC '09: Proceedings of the 2009 ACM symposium on Applied Computing*, pages 808–809, New York, NY, USA, 2009. ACM.

[27] Matthew J. Patitz. ISU TAS. Homepage: http://www.cs.iastate.edu/~lnsa/software.html, 2008.

[28] Paul Rothemund and Erik Winfree. The program-size complexity of self-assembled squares. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, pages 459–468, 2000.

[29] Paul W.K. Rothemund, Nick Papadakis, and Erik Winfree. Algorithmic self-assembly of DNA Sierpinski triangles. *PLoS Biology*, 2(12):2041–2053, 2004.

[30] Rebecca Schulman and Erik Winfree. Self-replication and evolution of dna crystals. In *In The 13th European Conference on Artificial Life (ECAL*, pages 734–743. Springer-Verlag, 2005.

[31] Michael Sipser. *Introduction to the Theory of Computation, Second Edition*. Course Technology, 2 edition, February 2005.

[32] David Soloveichik and Erik Winfree. Complexity of self-assembled shapes. In *Tenth International Meeting on DNA Computing*, pages 344–354, 2005.

[33] Leroy G. Wade. *Organic Chemistry*. Prentice Hall, 2nd edition, 1991.

[34] Erik Winfree. *Algorithmic Self-Assembly of DNA*. PhD thesis, California Institute of Technology, Pasadena, 1998.

[35] Erik Winfree and Renat Bekbolatov. Proofreading tile sets: Error correction for algorithmic self-assembly. In *DNA Computers 9 LNCS*, pages 126–144, 2004.

[36] Erik Winfree, Xiaoping Yang, and Nadrian C. Seeman. Design and self-assembly of two-dimensional DNA crystals. In *Proceedings of the 2nd International Meeting on DNA Based Computers*, 1996.

A   A Simple 3D Example

To show how a temperature $\tau = 1$ system can simulate a temperature $\tau = 2$ 2D zig-zag system, in this section we show the explicit 3D tile set to simulate a simple unary counter. The tileset in Figure 6 (a) with seed tile labeled "s" is a system designed to count in unary at temperature 2, starting at value 2, denoted by 2 tiles with label "b" in the initial row of the assembly. At every other row of the assembly the rightmost "b" tile is switched to an "a" tile. Once all tiles are turned to "a" tiles, the counter is designed to stop. While simple, this system demonstrates the cooperative binding that is the key characteristic of temperature $\tau = 2$ systems.

Figure 6 parts (b), (c), and (d) show the 3D tiles added to the system for each collection of the original zig-zag system tile set. The tile set is derived from the conversion table detailed in Section C. Finally, the terminal temperature $\tau = 1$ assembly is shown in Figure 7, which can be compared to the final assembly of the original zig-zag system shown in Figure 6 part (a).

B   Constructing a Turing Machine with Zig-Zag Tile Sets at $\tau = 2$ in 2D

In this section we discuss our results for temperature 1 tile assembly systems capable of simulating Turing machines. Our results are based on existing temperature 2 self-assembly systems capable of simulating Turing machines[32]. With straightforward modifications, these constructions can be modified into zig-zag systems, and thus can be simulated at temperature 1 in 3D or in 2D probabilistically. Our first Lemma states that such temperature 2 zig-zag systems exist. For simplicity of bounds, we assume the Turing machine to be simulated is of a constant bounded size (constant bound number of states, alphabet size).

LEMMA B.1. *For a given Turing machine T there exists a temperature 2 zig-zag tile assembly system that simulates T. More precisely, there exists a zig-zag system that, given a seed assembly consisting of a horizontal line with north glues denoting an input string, the assembly will place a unique* accept *tile type if and only if T accepts the input string, and a unique* reject *tile type if and only if T rejects the input string.*

By combining this Lemma with the zig-zag simulation lemma, we get the following results.

THEOREM B.1. *For a given Turing machine T there exists a temperature 1, 3D tile assembly system that simulates T, even when assembly is limited to one step into the third dimension. Thus, 3D temperature 1 self-assembly is universal.*

Proof. This theorem follows from Lemma B.1 and Theorem 3.1.

THEOREM B.2. *For a given Turing machine T there exists a temperature 1, 2D probabilistic tile assembly system that simulates T. In particular, for any $\epsilon > 0$ and machine T that halts after N steps, there exists a temperature 1, 2D system with $O(\log N + \log \frac{1}{\epsilon})$ tile types in terms of N and ϵ that simulates T and guarantees correctness with probability at least $1 - \epsilon$.*

This theorem follows from combining Lemma B.1 and the construction for zig-zag simulation used in Theorem 3.2 with parameter $k = \log \frac{N}{\epsilon}$. As discussed in Section 3.3 the probability of an single error in simulation of a given zig-zag set is $(7/8)^k$. Thus, the expected number of errors for $k = \log_{8/7} \frac{N}{\epsilon}$ becomes $\frac{\epsilon}{N}$, implying that the probability of 1 or more errors is bounded by at most ϵ.

B.1   Details of Zig-Zag Turing Machine Construction. A ***Turing machine*** is a 7-tuple[31], $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, where Q, Σ, Γ are all finite sets and

1. Q is the set of states,

2. Σ is the input alphabet not containing the ***blank symbol*** $\sqcup$,

3. Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$,

4. $\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\}$ is the transition function,

5. $q_0 \in Q$ is the start state,

6. $q_{accept} \in Q$ is the accept state, and
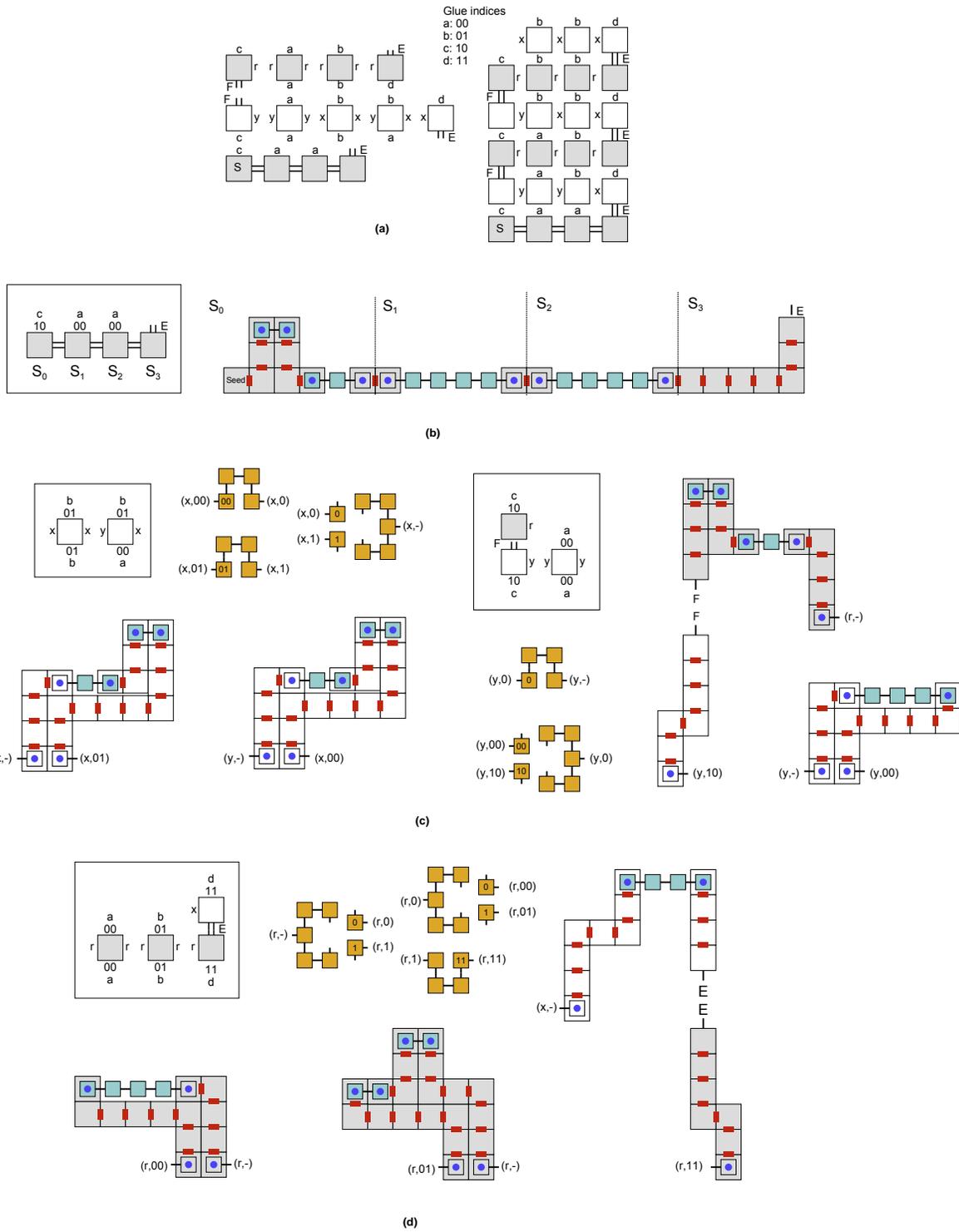
Figure 6: *The tileset in (a) is a zig-zag tileset that swaps the rightmost a to b every other line. In (b), (c), and (d), a collection of macro tiles are provided, according to Theorem 3.1. This set will simulate the given tileset system using 3 dimensions at temperature τ = 1.*
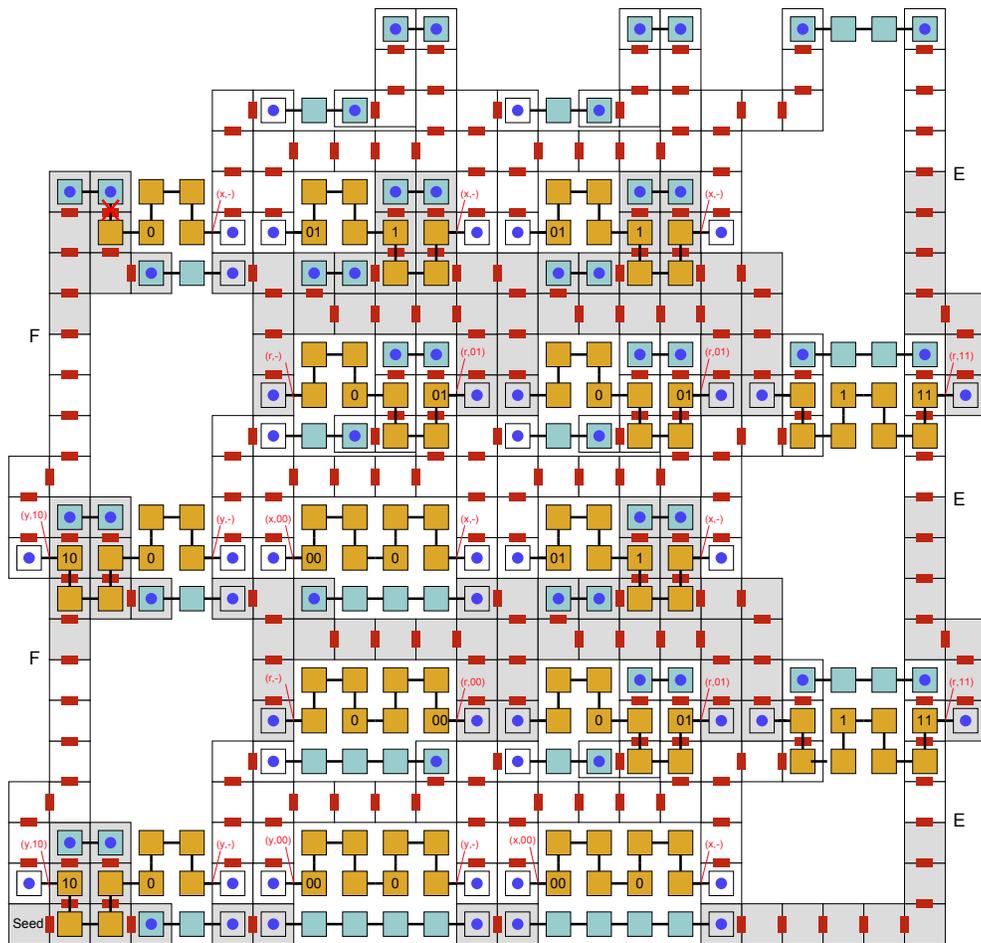
Figure 7: *This is the final assembly from the tileset given in Figure 6*
.

7. $q_{reject} \in Q$ is the reject state, where $q_{accept} \neq q_{reject}$.

All of the glues at the north or south side of a tile make up the alphabet in set For each element of the alphabet Γ, including a corresponding north/south glue unique to that element in the zig-zag tile system. The north glue (g_n) of a tile will serve as the output and g_s as the input in the Turing Machine (TM). The action of placing a tile with a different g_n and g_s is regarded as one write operation while placing one tile with the same g_n and g_s is regarded as one read operation.

Basically, the tiles growing from the west direction will be used to perform the main calculation, and the tiles growing from the east direction will be used to copy the glues from south to north.
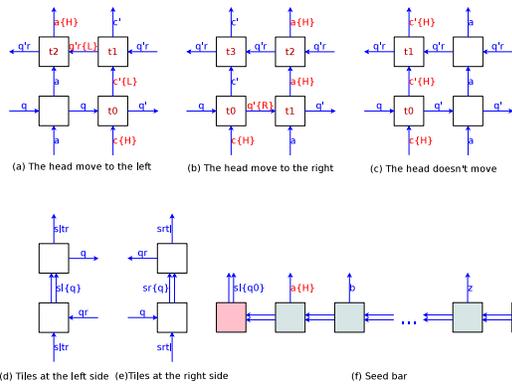
Figure 8: **Zig-Zag Tile Structure for Contructing Turing Machine**. **(a)** $\delta(q, c) = (q', c', L)$; **(b)** $\delta(q, c) = (q', c', R)$; **(c)** $\delta(q, c) = (q', c', \phi)$; **(d)** *Left side*; **(e)** *Right side*; **(f)** *Seed bar: q_0 and input sequence $a, b, \ldots, z$*

First, consider the equation $\delta(q, c) = (q', c', R)$. The head of the TM is located at the c position and the current state is q. The character at the current head position is then changed to c', the current state is changed to q', and the head is moved to the right (R). Three types of the tiles, which are indicated by t_0, t_1, and t_2 in Figure 8b, are used to simulate this process. The glue notation at the south of the tile t_0 means that the head is at the column of tile t_0 and input alphabet is c. The glue notation (q) at the west of the tile t_0 means that the current state is q. The glues at the north and east indicate the output and next state separately. The east glue $(q'\{R\})$ of the tile t_0 indicates the next state (q') and carries the extra information to notify the next tile that the head is coming.

The equation $\delta(q, c) = (q', c', L)$ indicates that the head will move to the left. Figure 8a shows how to move the head to the left. Again, the head is at the column of tile t_0, the current state is q, and input is c. Tile t_0 then outputs north glue $c'\{L\}$, and east glue q'. The tile t_1 will translate the glue information $(c'\{L\})$ to state information

$(q'r\{L\})$ which contains the command to move left. At last, the tile t_2 is added to combine the header notation(H) with the glue of the current column.

If the head doesn't move after reading one character from the tape, the corresponding tile is t_0 shown in Figure 8c. Figure 8d and Figure 8e show the tiles at the left and right sides of the zig-zag structure separately. Figure 8f shows the seed bar, which initializes the Turing Machine, including the start state q_0, the input sequence $(a, b, \ldots, z)$ etc.

B.2 Complexity Analysis In the case of a $\tau = 2$, 2D zig-zag tile system, the number of the tile types for state transfer (t_0 in the Figure 8) is $|\delta|$. And the number of the auxiliary tile types for state transfer (t_1 and t_2 in Figure 8a, 8b) is $|\Sigma| \times |Q| \times 2$ (move left) and $|\Sigma| \times |Q| \times 2$ (move right). The number of tile t_1 in Figure 8c does not need be included because it is counted in the auxiliary tile types (t_2 in Figure 8b). The other tile types are the tiles for transferring alphabet information, which have a count of $|Q| \times |\Sigma| \times 2$ (2 indicates the east direction and west direction of the growth).

Some special types of tiles in both sides of the zig-zag structure are needed to change the growth direction, costing $4|Q|$ tiles. If the length of the tape is n, then the number of the tile types for constructing the seed bar will cost $n + 2$ tiles (See Figure 8f).

The total number of tile types is $|T| = |\delta| + |\Sigma| \times |Q| \times 2 + |\Sigma| \times |Q| \times 2 + |Q| \times |\Sigma| \times 2 + 4|Q| + n + 2 = |\delta| + 6|Q||\Sigma| + 4|Q| + n + 2$. This yields an asymptotic upper bound of $O(|\delta| + |Q||\Sigma| + n)$ tile types. Considering that $|Q| \leq |\delta| \leq |Q|^2|\Sigma|^2$ and $n \leq |\Sigma|$, the complexity of tile types is $O(|\delta| + |Q||\Sigma| + n) = O(|\delta|)$.

Each of the state transfers costs two lines of the zig-zag structure (east direction line and west direction line above). Given an input tape and start state, if the number of the state transfers to be passed before it stops is r, then the lines of the tile structure will be $2r$. So the space used in calculation is $O(nr)$.

In the case of a $\tau = 1$ 3D zig-zag tile system, the length of each mapped tile depends on the length of binary encoding code of glues. If the number of glues of the 2D zig-zag tiles is G, then $G = O(|\delta|)$. The complexity of tile types in 3D is $O(|\delta| \log G) = O(|\delta| \log |\delta|)$, the complexity of space is $O(nr \log G) = O(nr \log |\delta|)$.

In the case of a $\tau = 1$ 2D probabilistic zig-zag tile system, the length of each mapped tile depend on the parameter $K = O(\log r + \log \frac{1}{\epsilon})$ and the length of binary encoding code of glues. The parameter K of the $\tau = 1$ 2D probabilistic zig-zag tile system is a constant that depends on the probability of correctly achieving a result. The space complexity is $O(nrK \log G) = O(nr \log |\delta|)$, and the number of tile types is $O(|\delta| \log |\delta|)$.

584

Table 2: *The Complexity of the Zig-Zag Turing Machine*

	$\tau = 2$ **2D**	$\tau = 1$ **3D**	$\tau = 1$ **2D Prob.**
Space	$O(nr)$	$O(nr \log |\delta|)$	$O(nr \log |\delta|)$
Tiles	$O(|\delta|)$	$O(|\delta| \log |\delta|)$	$O(|\delta| \log |\delta|)$

$|\delta|$: the number of state functions;

n: the length of the tape (size of alphabet);

r: the number of the state transfers to be passed before stopping;

K: the parameter of probabilistic zig-zag tile systems;

C   Convert an Arbitrary Zig-Zag Tile Set from $\tau = 2$ to $\tau = 1$

C.1   The Conversion Table   The conversion algorithm categorizes each type of tile in a Zig-Zag system into a particular category. For each category, the tile types that are added to a temperature $\tau = 1$ 3D system or a temperature $\tau = 2$ probabilistic system for a successful simulation are detailed in Table 3.

Table 3: Zig-Zag Tile Set Mapping
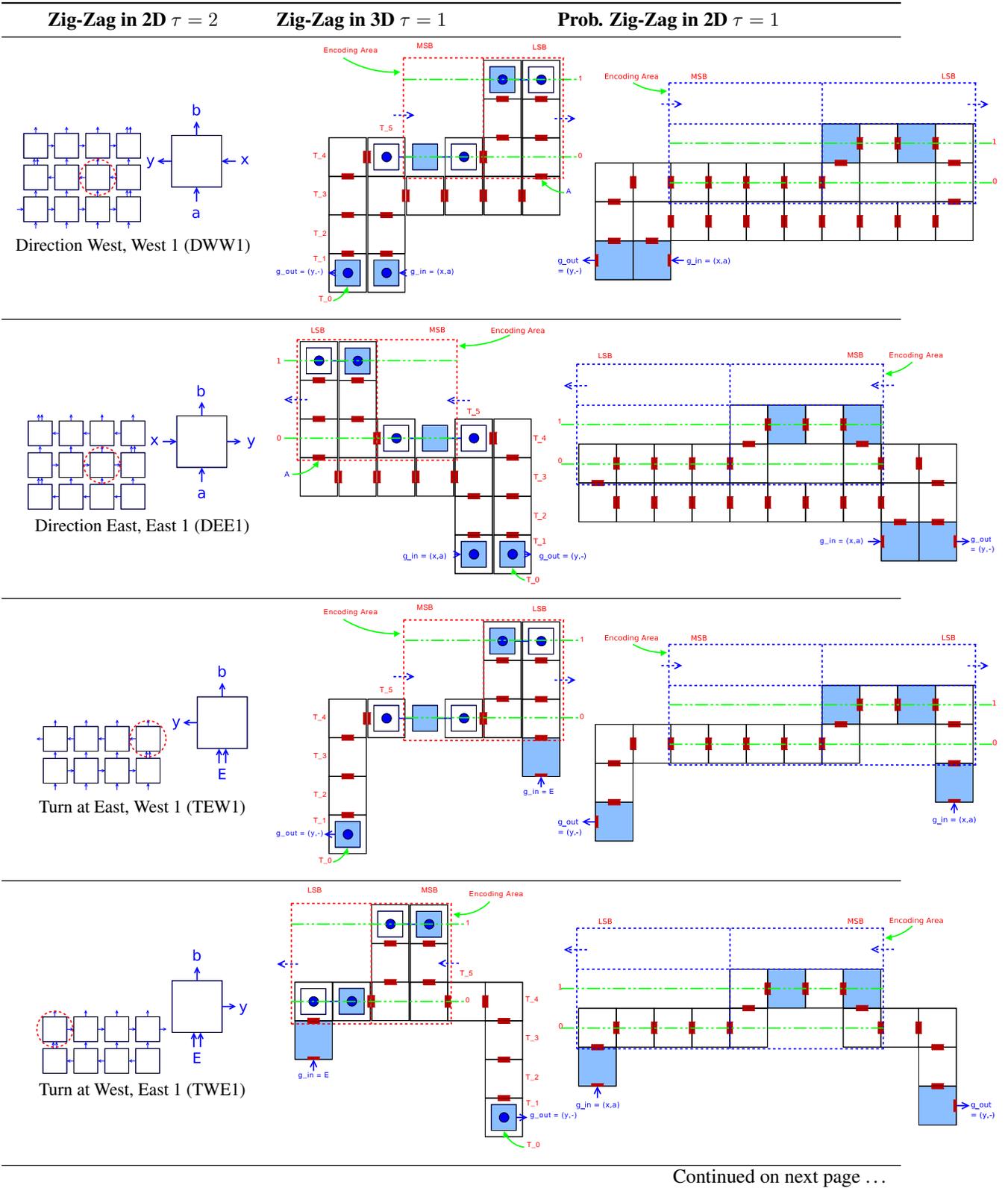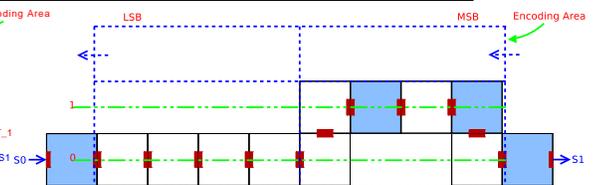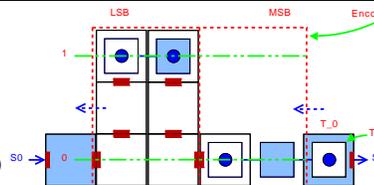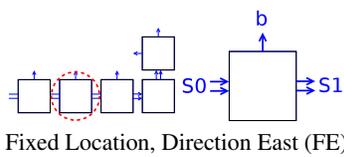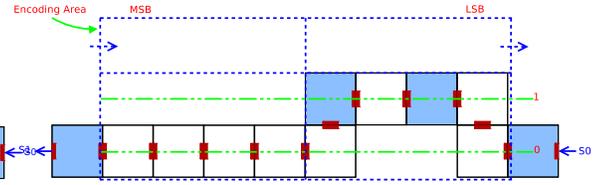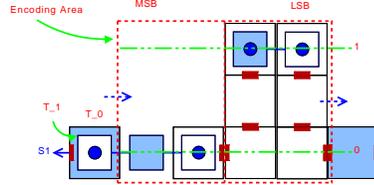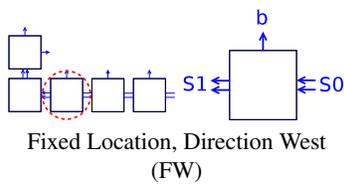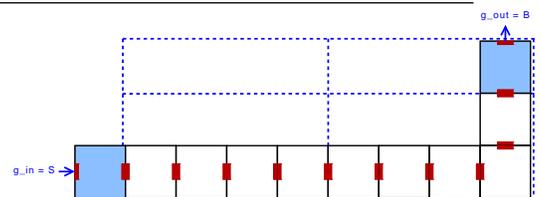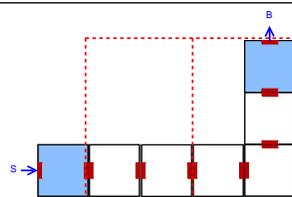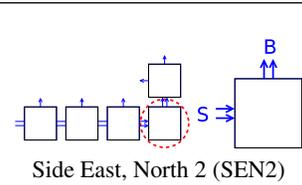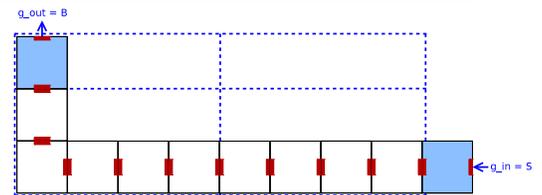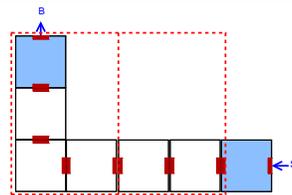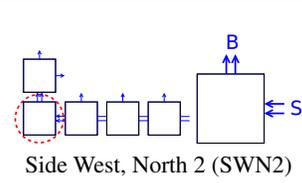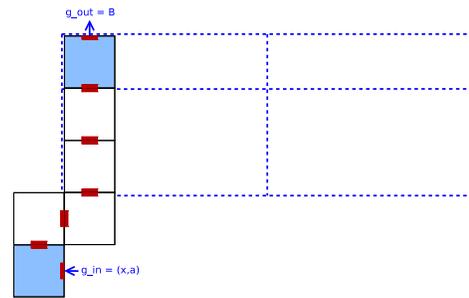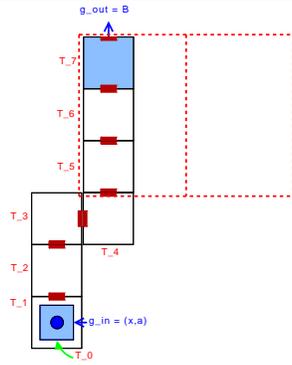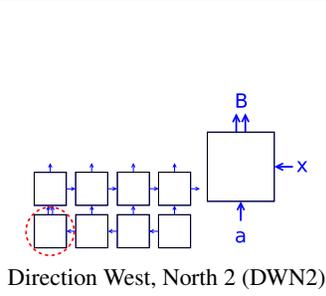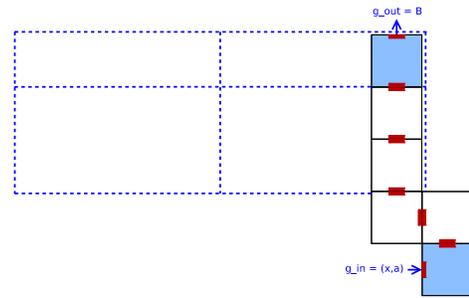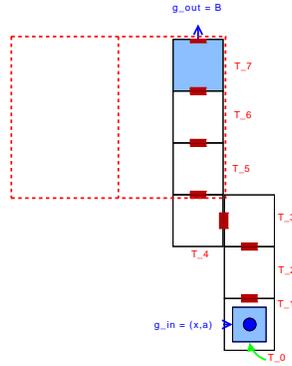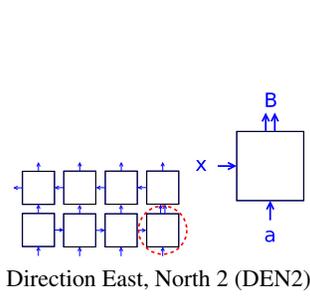
| **Zig-Zag in 2D** $\tau = 2$	**Zig-Zag in 3D** $\tau = 1$	**Prob. Zig-Zag in 2D** $\tau = 1$
Direction West, West 1 (DWW1)		
Direction East, East 1 (DEE1)		
Turn at East, West 1 (TEW1)		
Turn at West, East 1 (TWE1)		

Continued on next page . . .

Zig-Zag in 2D $\tau = 2$	Zig-Zag in 3D $\tau = 1$	Prob. Zig-Zag in 2D $\tau = 1$

Direction East, North 2 (DEN2)

Direction West, North 2 (DWN2)

Side West, North 2 (SWN2)

Side East, North 2 (SEN2)

Fixed Location, Direction West (FW)

Fixed Location, Direction East (FE)

Continued on next page . . .
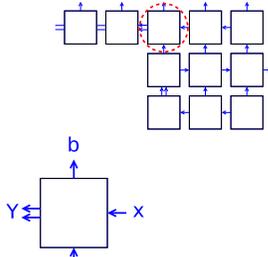
Zig-Zag in 2D $\tau = 2$	Zig-Zag in 3D $\tau = 1$	Prob. Zig-Zag in 2D $\tau = 1$
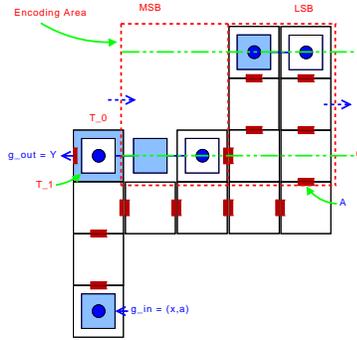
Direction West, West 2 (DWW2)
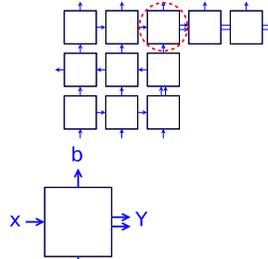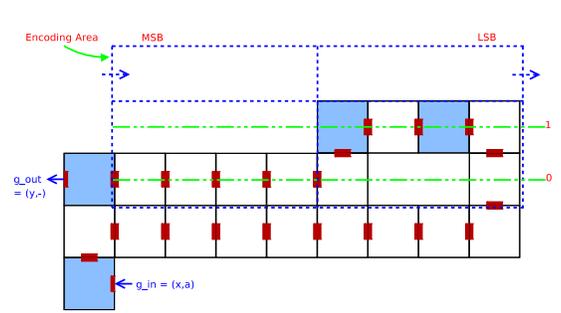
Direction East, East 2 (DEE2)
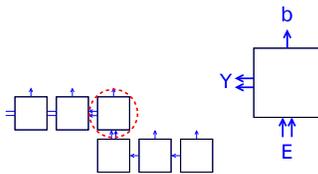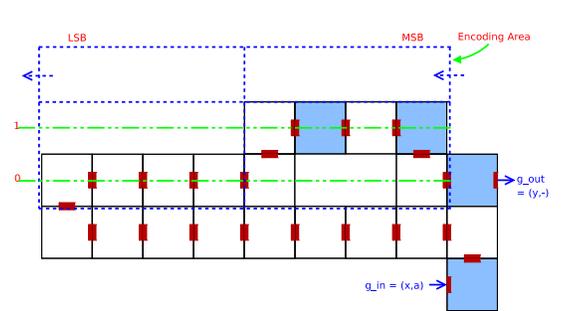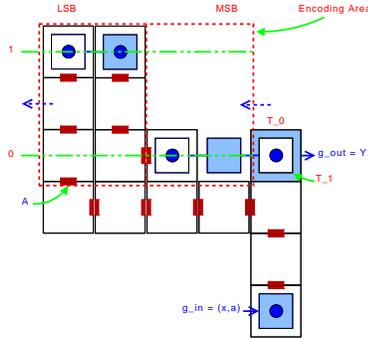
Turn at West, West 2 (TWW2)
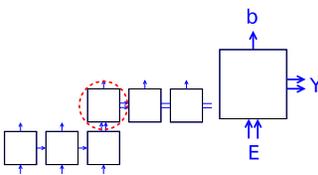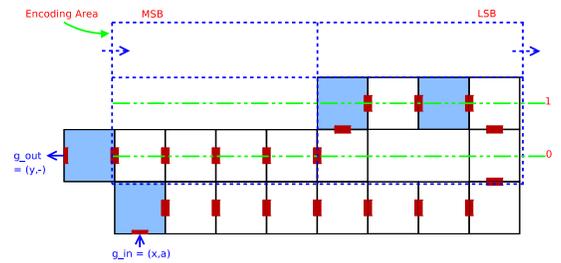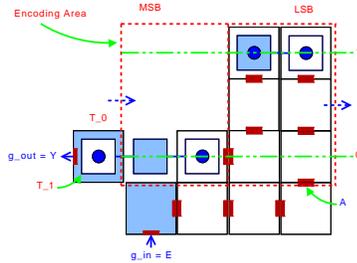
Turn at East, East 2 (TEE2)
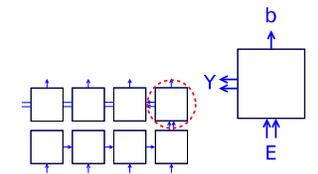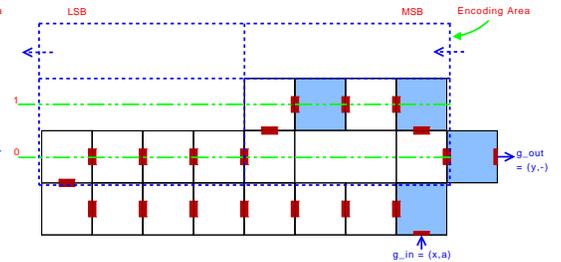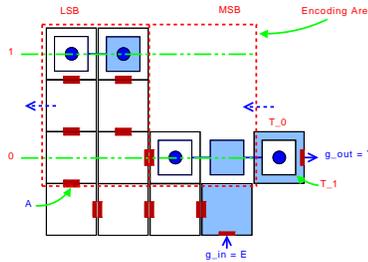
Turn at East, West 2 (TEW2)

Continued on next page . . .

Zig-Zag in 2D $\tau = 2$	**Zig-Zag in 3D** $\tau = 1$	**Prob. Zig-Zag in 2D** $\tau = 1$

Turn at West, East 2 (TWE2)