Scaling up Thermodynamically Favoured Scaffolded DNA Computing by Sculpting the Energy Landscape

Joshua Petrack[1]
Hamilton Institute and Department of Computer Science, Maynooth University, Ireland

Constantine G. Evans[1]
Hamilton Institute and Department of Computer Science, Maynooth University, Ireland

Angel Cervera Roldan
Hamilton Institute and Department of Computer Science, Maynooth University, Ireland

Mahboobeh Enayati
Hamilton Institute and Department of Computer Science, Maynooth University, Ireland

Damien Woods
Hamilton Institute and Department of Computer Science, Maynooth University, Ireland

Abstract

Thermodynamically favoured molecular computation offers advantages over more typical out-of-equilibrium computing, including simpler experimental protocols and automatic error correction. But as systems scale to large sizes the number of states increases dramatically, increasing the need for efficiently navigable energy landscapes. We give results in two theoretical models of Scaffolded DNA Computing (SDC), a recently implemented form of thermodynamically favoured DNA computing [Stérin, Eshra et al, bioRχiv 2025]. We show their computational power is characterised by logarithmic space complexity classes, meaning they are expressive at scale. Our first energy landscape result is an exact relation between the probability of target configurations (outputs), temperature and DNA sequence domain length, showing that domain length merely logarithmic in scaffold length is sufficient for the probability of the target configuration to outcompete all off-target structures. We show that even in the presence of imperfect/unequal binding strength scaffold domains we still achieve good kinetics: $O(N^2)$ or $O(N^3)$ expected completion time, depending on model assumptions, and there are even narrow conditions that yield fast $O(N)$-time kinetics. Finally, we address a thorny scaling problem: SDC outputs often have repeated compute domains and any binding energy variances get exaggerated under repetition making errors favourable, but we give a construction that reprograms the energy landscape to convert such a non-isoenergetic system into one with almost perfectly isoenergetic energy plateaus. We also show that systems maintain good (polynomial-time) kinetics, even in the face of a poor (uphill) scaffold energy landscape. These results give a roadmap for scaling up the SDC while highlighting the role kinetics and energy landscape programming could play in thermodynamically-favoured computing more generally.

2012 ACM Subject Classification Applied computing → Chemistry; Applied computing → Physics; Theory of computation → Models of computation

Keywords and phrases Thermodynamically favoured computing, molecular programming, DNA computing, kinetics, computational power, Scaffolded DNA Computer

Digital Object Identifier 10.4230/LIPIcs.DNA.32.7

Supplementary Material *Collection (Source code and data for paper & figures)*: `https://doi.org/10.5281/zenodo.21441279`

[1] Equal contribution

32nd International Conference on DNA Computing and Molecular Programming (DNA 32).
Editors: Dominic Scalise and Robert Schweller; Article No. 7; pp. 7:1–7:22

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Funding Supported by the European Union's (EU) European Innovation Council (EIC) and SMEs Executive Agency (EISMEA), No 101115422, DISCO project, Research Ireland (RI) under grant number 20/FFP-P/8843, and Maynooth University. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the EU, EISMEA or RI. Neither the EU nor the granting authority can be held responsible for them.

Acknowledgements We thank Abeer Eshra, Tristan Stérin & Janet Adio for discussions on SDC scaling and data in Figure 4, and John Reif for conversations on complexity results.

1 Introduction

The idea behind thermodynamically-favoured molecular computing is to mix a soup of molecules, let them find their chemical equilibrium – *et voilà!* – we have the answer to a computation. The notion has roots in the work of Bennett [2, 3], who reasoned about reversible computation at equilibrium, in order to understand the ultimate energy costs in computing [6]. In the DNA computing and molecular programming field, one useful model has been Thermodynamic Binding Networks [7], which were recently implemented [21]. DNA nanostructures have had a thermodynamically-favoured method for some decades: DNA origami [14] works so well precisely because the intended structure is usually favoured and the folding landscape likely has efficient kinetic pathways – an ongoing topic of research [4, 8, 10, 16, 18]. Three key rules underlie DNA origami's energetic favourability: (1) staples in high concentration excess bind to a long low-excess scaffold strand; (2) staples do not bind to each other; and (3) good 3D geometric design. The Scaffolded DNA Computer (SDC) [20] obeys the first rule, but carefully breaks the second, by having short strands bind both to a scaffold and to each other, crucially exchanging information and logic by doing so, see Figure 1. We desire thermodynamically favoured computers be easy to prepare (just mix the molecules) and easy to run (heat up, cool down). The scaffolded principle allows for straightforward implementation of thermodynamically-favoured computing, with multiple tunable system parameters to optimise the energy landscape. Here, we explore several.

The SDC was demonstrated on 10 programs, was shown to be reusable multiple (25) times, and even ran super-fast 16-bit computations in under a minute – some of the fastest nontrivial DNA computations to date [20]. For scaling up, binary Addition of pairs of 25-bit binary numbers was experimentally demonstrated (a 100-bit computation: 50 input bits, 25 carry bits, 25 output bits). However, for a very small fraction of inputs (around 2^{-50}) the system was stuck in an incorrect energy minimum. One problem identified was non-isoenergetic compute and scaffold domains (Figure 1a) conspiring to mess up the scaled up energy landscape in two ways: creation of local minima that mess up the kinetics, and even having erroneous structures unintentionally be the most energetically favoured.

1.1 Sculpting the energy landscape

Here we take the approach of using both program logic, and characterisation of system parameters, to sculpt the energy landscape to have both a high probability of the target output, as well as efficient, mostly downhill, pathways to that target.

For instance, the target output of the BitCopy program (Figure 1a) on a length N scaffold, repeats the same compute domain (either bit `0` or `1`) $N-1$ times, but if the binding strength of `0` is slightly weaker than `1` by some ϵ, then the two target configurations for input `0` and `1` will differ by $\epsilon(N-1)$, a large amplification. In that case, for some large enough N, an incorrect configuration will become the most favourable structure meaning that the energy

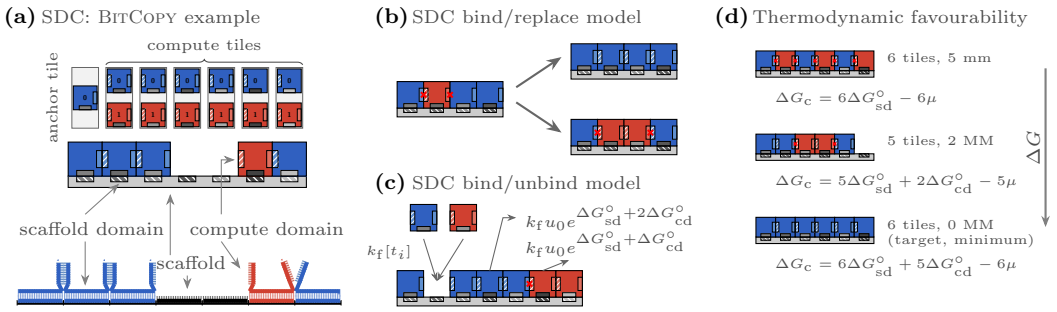

Figure 1 **(a)** In the Scaffolded DNA Computer (SDC) tiles bind to a 1D scaffold. In the BitCopy program, an input bit is encoded in an anchor tile placed at the first scaffold position, then 0/1 tiles compete at each position to faithfully transmit the bit to the right. **(b)** The bind/replace SDC model considers configurations with a tile at every position; a tile is replaced if the replacing tile matches the same number of, or more, compute domains. **(c)** In the bind/unbind SDC model, tiles attach at concentration-dependent rates, and detach at energy-dependent rates. **(d)** The configuration energy depends on the number of tiles and bonds (or mismatches), with the full, no-mismatch target configuration having minimum energy configuration, assuming its bonds are sufficiently strong. See [20] for 10 more programs.

landscape has the "wrong" global minimum as well as being polluted by local minima that slow the system down.[2] The program IsoenergeticBitCopy [20] remedied the situation by reprogramming the energy landscape to be composed of a sequence of energy plateaus, one for each number of compute domain mismatches m, guaranteed to have both the correct global minimum and good kinetic pathways to get there. In Section 6 we prove that many SDC programs can be made isoenergetic in this way.

A second scaling problem comes from having non-isoenergetic scaffold domains: a strong scaffold domain can lock-in an incorrect tile during an anneal, or a weak scaffold domain can mean that two halves of the computation become isolated from each other, slowing the system's natural error-correction processes. Since synthesis of custom-designed long DNA scaffolds is expensive and challenging, and tricks like DNA-base mismatches on biological (M13) scaffolds to even out domain energies have shown limited benefit [20], here, we instead turn to mathematics by proving an asymptotic result showing that good, polynomial time, kinetics can be maintained even in the face of poor scaffold energetics.

Finally, another scaling issue is whether for large enough scaffold length N there are accessible and wide enough regions of experimental assembly conditions (temperature, domain strength, concentrations, etc.) to allow for easy running of the system. We want to retain the in-principle advantage of just heating the system and cooling it down to find the intended correct output – precipitating a careful analysis of the system's phase space.

1.2 Results and paper structure

In Section 2 we define two models: The SDC bind/replace model (Figure 1b) is intended to be simple to mathematically analyse, but ignores physically important properties like domain strength, tile concentration and temperature, key features of the SDC bind/unbind (Figure 1c) model. Sections 3–6 each give results concerned with scaling to larger scaffolds.

[2] Such as 011111: input is bit 0, the structure gains favourability by repeating lots of 1 bonds, and suffers only the cost of a single compute domain mismatch; see Figure 5 for more details.

In Section 3 we find that computational power of SDC systems is exactly characterised by the complexity class NL, or nondeterministic logspace [11] – assuming we allow polynomially many tile types per scaffold position and arbitrary amounts of nondeterminism. However, most SDC programs implemented to date, have a more restrictive left-to-right dynamics which turns out to characterise the complexity class L (deterministic logspace). (Previous work proved that SDCs simulate finite state machines [20], hence we generalise that result.)

Section 4 (Corollary 16) shows that the SDC bind/replace model has expected time for programs like BitCopy of $\leq N^2$ (a small improvement on a previous $O(N^2)$ upperbound [20]), but mainly serves to set the scene for a more dramatic result in Section 6 (Theorem 29) where despite adversarially left-to-right increasing scaffold domain strengths on the scaffold we retain $O(N^2)$ completion time if the strength increase is a constant amount over the entire scaffold length, or polynomial completion time if it is $O(\log N)$ over the entire scaffold.

In Section 5, for the more physically accurate SDC bind/unbind model, this quadratic scaling result becomes $O(N^3)$ due to the wait time for tiles to fall off. This in turn comes from a positive scaling result (Corollary 18) showing that compute domain length of merely $O(\log N)$ DNA bases (or strength) is sufficient to have the correct target (output) with constant probability, greater than the total probability of exponentially many other structures. Thus, for a mere factor-N increase in running time, we gain high-probability computing. More generally, we give an exact expression for the probability of the target structure, over all filled structures, as a function of scaffold length and compute domain strength. We also find certain regimes (temperature, domain strength/length) with super-fast, mostly left-to-right, $O(N)$-time kinetics. These theoretical results align with simulation, and in some cases experiment, despite us not yet fully understanding the energetics of our systems.

Finally, in Section 6.1 we prove a generalisation of the IsoenergeticBitCopy result described above: we show that a wide class of SDC programs can be simulated by systems that have almost perfectly isoenergetic configurations; in other words these simulator SDCs have an energy landscape that is composed of a sequence of plateaus that increase in height with the number of mismatches, and that lack kinetic traps. Traversing such a landscape amounts to downhill ratcheting steps, interspersed with flat plateau steps, giving fast dynamics.

1.3 Future work

We leave a number of open questions and directions for future work.

Many of our theoretical kinetics results are shown for BitCopy, an SDC program that is straightforward to analyze while demonstrating much of the system's dynamics. The analysis holds for similar programs with $O(1)$ tiles competing at each scaffold position, or where following the leftmost mismatch resolves errors. Can these results be generalized to programs that have a variable number of tiles per scaffold position, or where pushing the leftmost mismatch to the right results in unresolvable kinetic traps, or other complex pathways through the energy landscape?

In the SDC bind/replace model with scaffold length N, Corollary 16 gives a running time of $\leq N^2$ (for BitCopy and similar programs). We conjecture that the running time from configurations chosen uniformly at random is $\Theta(N^2)$, a proof of which would show that mismatches cancelling each other out does not asymptotically speed up completion time.

Isoenergetic programs: In Section 6.1 we show that with a multiplicative scale factor equal to the number of distinct compute domains, we can compile any program into one with isoenergetic configurations that have a fixed number of mismatches. Can we reduce the scale factor? Can isoenergetic programs be achieved with scale factor 1? Can we further reduce energy differences between configurations with the same number of mismatches?

Temperature anneals: An advantage of thermodynamically-favoured DNA computing, and SDC in particular, is the ability to run the system via a simple thermal anneal: heat it up to melt everything, then cool down at some rate to compute. Previous experimental work showed 4-bit Addition in under a minute, but requiring several hours for 25-bit Addition [20]. What is the theoretically optimal annealing schedule for high-probability computing in terms of scaffold length (or number of input bits)?

Going beyond the SDC-specific work here, is there a more general theory of thermodynamically favoured computation, that embodies both equilibrium and kinetic pathways through a state space defined by the Boltzmann distribution over valid molecular complexes?

2 Definitions

The *domains* are a finite set D, partitioned into scaffold domains D_{sd} and compute domains D_{cd}. A domain $d \in D$ has a unique complement $d^* \in D$ with $(d^*)^* = d$. Complements respect the partition: either $d, d^* \in D_{\text{sd}}$, or $d, d^* \in D_{\text{cd}}$.

▶ **Definition 1** (tile). A *tile* is a triple $t = (l, p, r)$ where $p \in D_{\text{sd}}$ is a scaffold domain and $l, r \in D_{\text{cd}}$ are the left and right compute domains, respectively.

▶ **Definition 2** (SDC). An *SDC* S is a sequence of N nonempty tilesets $S = (T_1, T_2, \ldots T_N)$ where for each i, all tiles in T_i have the same scaffold domain p_i. The SDC is said to have scaffold length N with scaffold domains ${p_1}^*, {p_2}^*, \ldots, {p_N}^*$ at scaffold positions $1, 2, \ldots, N$. If a tileset T_i has size 1, we say that T_i (or position i, or the tile in T_i) is an *anchor*.

Although the previous definition allows for scaffold domains to be repeated, in this paper, and previous work [20], all N scaffold domains are distinct, i.e. $p_i \neq p_j$ when $i \neq j$.

The *configuration space* of S is $S^* = (T_1 \cup \{\emptyset\}) \times (T_2 \cup \{\emptyset\}) \times \ldots \times (T_N \cup \{\emptyset\})$. A *configuration* is an N-tuple $C \in S^*$, indicating a tile $C[i]$, or no tile ($C[i] = \emptyset$), at each position i. In C, if adjacent tiles $t_{i,j} = (l, p_i, r) \in T_i$ and $t_{i+1,j'} = (l', p_{i+1}, r') \in T_{i+1}$ have adjacent complementary compute domains (meaning $r = {l'}^*$), we say the tiles, and their compute domains, *match*. Otherwise, they *mismatch*. A *full configuration* has no empty positions, a *partial configuration* has at least one. Absent tiles are treated as having domains which are not complementary to anything.

One key example of an SDC is BitCopy (Figure 1a), having compute domains $\mathtt{0}$ and $\mathtt{1}$, and their complements $\mathtt{0}^*, \mathtt{1}^*$. The tileset at each position i has two tiles $T_i = \{(\mathtt{0}^*, p_i, \mathtt{0}), (\mathtt{1}^*, p_i, \mathtt{1})\}$, except for the first, which is an anchor (one tile). The only full mismatch-free configuration has all tiles matching the anchor, meaning it acts as a wire.

▶ **Definition 3.** An *SDC model* is a tuple $\mathcal{S} = (S, \xrightarrow{1}, R)$, where S is an SDC, $\xrightarrow{1}$ is a relation on $S^* \times S^*$ defining *one-step reachability* and denoted as $C \xrightarrow{1} C'$, and $R : S^* \times S^* \to \mathbb{R}^{\geq 0}$ is a *transition rate matrix* where $R(C, C') > 0 \iff C \xrightarrow{1} C'$. These rates define a continuous-time Markov chain over S^*.

This definition is intentionally abstract; two concrete models will be instantiated in Definitions 5 and 7. In a model, C' is reachable from C if there is a sequence of steps $C \xrightarrow{1} \ldots \xrightarrow{1} C'$. A *terminal configuration* is one to which no step applies. For computation, target (output) configuration(s) should be reachable from any configuration and terminal; a computation is a sequence of steps from an initial to a terminal configuration. An SDC model is *directed* if it has a unique terminal configuration,[3] and *undirected* otherwise (it reaches multiple terminal

[3] This notion of determinism (directness) comes from work on the abstract tile assembly model [19].

configurations, or none). The *expected time to completion* (respectively, expected steps) from configuration C of a directed SDC model is the expected time (steps) for the Markov chain to reach a terminal configuration from C. When $\xrightarrow{1}$ and R are clear from context, we use S and $\mathcal{S}$ interchangeably. $C \xrightarrow[i]{1} C'$ means $C \xrightarrow{1} C'$ and C, C' differ by a single tile $C[i] \neq C'[i]$.

2.1 SDC bind/replace model

▶ **Definition 4** (SDC bind/replace: mismatch energy)**.** The *mismatch energy* of a configuration C is $G_{\mathrm{MM}}(C) = |\{i \mid 1 \leq i < N \text{ and } t_i, t_{i+1} \text{ mismatch}\}|$; that is, the number of compute domain mismatches in C.[4]

▶ **Definition 5** (SDC bind/replace model)**.** Given an SDC S, the *SDC bind/replace* model $(S, \xrightarrow{1}, R)$ is defined by $C \xrightarrow{1} C'$ if C and C' differ in exactly one position i, $C'[i] \neq \emptyset$, and either $C[i] = \emptyset$ or $G_{\mathrm{MM}}(C') \leq G_{\mathrm{MM}}(C)$. That is, in one step, a tile may bind to an empty position, and a tile may replace another if the replacement binds with equal or fewer mismatches. The transition rates are given by $R(C, C') = 1$ if $C \xrightarrow{1} C'$ and 0 otherwise.

The SDC bind/replace model has a first step where tiles bind to the scaffold, nondeterministically chosen at each position, yielding a full configuration. We ignore partial configurations and instead consider arbitrary full configurations, as this first step is generally fast.

2.2 SDC bind/unbind model

The SDC bind/unbind model includes key physical/chemical details. Throughout the paper, T denotes temperature in Kelvin, and R is the gas constant – a molar equivalent of the Boltzmann constant, in energy/temperature/amount ($\mathrm{R} = 1.987$ cal $\mathrm{K}^{-1}\mathrm{mol}^{-1}$). A tile t has a positive real-valued concentration denoted $[t]$, in units of amount/volume (Molar, or M).

▶ **Definition 6** (SDC bind/unbind: standard binding free energy)**.** Each pair of domains has a *standard free energy of binding* $\Delta G^\circ : \mathbb{R}^+ \times D \times D \to \mathbb{R}$ mapping a temperature and domain pair to a real-valued free energy in energy/amount units (kcal/mol [15]), where more negative is more favourable. We assume $\Delta G^\circ(\mathrm{T}, d, d') = 0$ if $d' \neq d^*$, unless otherwise stated. This energy is calculated with respect to reactants at a reference concentration $u_0 = 1\,\mathrm{M}$; the reference is denoted by the $^\circ$ sign.[5]

▶ **Definition 7** (SDC bind/unbind model)**.** Given an SDC S, the *SDC bind/unbind* model $(S, \xrightarrow{1}, R)$ is defined by $C \xrightarrow{1} C'$ if C and C' differ in exactly one position i, where exactly one of $C[i]$ and $C'[i]$ is $\emptyset$. The transition rates $R(C, C')$ for attachment (when $C[i] = \emptyset$) or detachment (when $C'[i] = \emptyset$) of a tile $t_i = (l_i, p_i, r_i)$ are given by

- $r_{\mathrm{att}} = k_{\mathrm{f}}[t]$,
- $r_{\mathrm{det}} = k_{\mathrm{f}} u_0 \mathrm{e}^{(\Delta G^\circ(\mathrm{T}, p_i, {p_i}^*) + \Delta G^\circ(\mathrm{T}, r_{i-1}, l_i) + \Delta G^\circ(\mathrm{T}, r_i, l_{i+1}))/\mathrm{RT}}$

where k_{f} is an assumed or experimentally-derived rate constant (positive, real-valued, 1/concentration/time, and consistently $10^6\,\mathrm{M}^{-1}\mathrm{s}^{-1}$ in this work).

For notational convenience, we define the *chemical potential* of tile t as $\mu(t) = \mathrm{RT}\ln([t]u_0^{-1})$, implying a binding rate of the form $r_{\mathrm{att}} = k_{\mathrm{f}} u_0 \exp(\mu(t)/\mathrm{RT})$.

[4] This notion shows up in the Ising model, where a mismatch is called a domain. Here, the word domain has a completely different meaning coming from chemistry.

[5] While a reference concentration is the convention in many works [15], another convention uses amount fractions rather than concentrations, and results in different standard free energy values; this convention is notably used by NUPACK [5].

Equilibrium and configuration energy

In the SDC bind/unbind model, rather than defining the free energy of configurations by fiat, we can derive it from the previous definitions (see Appendix A.1 for the proof):

▶ **Lemma 8.** *Given any configuration C in the SDC bind/unbind model with reaction rates from Definition 7, the equilibrium probability of C is* $\text{prob}[C] = \mathrm{e}^{-\Delta G_c(\mathrm{T},C)/\mathrm{RT}} \cdot Z^{-1}$*, where*

$$\Delta G_c(\mathrm{T}, C) = \sum_{i\in\{1,\ldots,N\}} [\Delta G^\circ(\mathrm{T}, p_i, p_i^*) - \mu(t_i)] + \sum_{i\in\{1,\ldots,N-1\}} \Delta G^\circ(\mathrm{T}, r_i, l_{i+1}) \tag{1}$$

is called the configuration energy *of C at temperature* T*, R is the gas constant, $t_i = (l_i, p_i, r_i)$ is the tile at position i, $[t_i]$ is its concentration (if no tile is present $\mu(t_i) = 0$), and $Z = \sum_{C'\in\mathcal{C}} \mathrm{e}^{-\Delta G_c(\mathrm{T},C')/\mathrm{RT}}$ is the* partition function *over the set of all configurations $\mathcal{C}$.*

2.3 Programs, inputs and outputs

In computer science, a problem is an infinite set of input-output pairs. To solve problems with SDCs we need to encode inputs and outputs, and a notion of an infinite set of SDCs to handle an infinite problem; we take inspiration from uniform Boolean circuit families [1, 12].

▶ **Definition 9** (SDC program)**.** Let Σ be a finite alphabet, and let $\mathcal{P}(\mathfrak{T})$ be the power set of all tiles $\mathfrak{T}$. An *SDC program* is a tuple $(f_{\text{tiles}}, f_{\text{in}}, f_{\text{out}})$, where the function $f_{\text{tiles}} : \mathbb{N} \to \mathcal{P}(\mathfrak{T}) \times \mathcal{P}(\mathfrak{T}) \times \ldots \times \mathcal{P}(\mathfrak{T})$ on input $x \in \Sigma^*$ maps its length $|x|$ to an N-tuple of tilesets $f_{\text{tiles}}(|x|) = \tilde{T} = (\tilde{T}_1, \ldots, \tilde{T}_N)$ called the *program*. The function f_{in} encodes an input word by removing some tiles from the program, thus defining the SDC instance $S \stackrel{\text{def}}{=} f_{\text{in}}(x, f_{\text{tiles}}(|x|)) = (T_1, \ldots, T_N)$ where $T_i \subseteq \tilde{T}_i$. Finally, the function f_{out} maps the input length to a tuple of *interpretation functions*: $f_{\text{out}}(|x|) = (f_1, f_2, \ldots, f_N)$, where $f_i : T_i \to \Sigma \cup \{\epsilon\}$ interprets the tile at index i in a full mismatch-free configuration C of S as a, possibly empty, symbol. C is interpreted as a whole by concatenating of these symbols.

An SDC's full mismatch-free configuration(s) are called *target* configuration(s). Note that this definition is chosen for simplicity: it requires the SDC to have at least one target configuration, and focuses on what these configuration(s) are, rather than how we get there kinetically. A language $L \subseteq \Sigma^*$ is deterministically (respectively, nondeterministically) SDC-decided if for all $x \in \Sigma^*$, $x \in L$ if and only if the (respectively, at least one) target configuration(s) of the SDC $S = f_{\text{in}}(x, f_{\text{tiles}}(|x|))$ are interpreted as 1 (the string in Σ^*) by $f_{\text{out}}(|x|)$. The functions $f_{\text{tiles}}, f_{\text{in}}, f_{\text{out}}$ should be "easily computable" otherwise the encoding/decoding scheme could cheat by doing all the computation; here, we assume they are DLOGTIME computable [1], an incredibly weak notion of computability commonly used in uniform circuit complexity [12]. [6]

▶ **Definition 10.** An SDC bind/unbind S is (a) *perfectly compute-domain isoenergetic* if for all temperatures T and all compute domains $d, d' \in D_{\text{cd}}$ in S, $\Delta G^\circ(\mathrm{T}, d, d^*) = \Delta G^\circ(\mathrm{T}, d', d'^*)$, and is (b) *perfectly scaffold-domain isoenergetic* if for all scaffold domains $d, d' \in D_{\text{sd}}$ in S, $\Delta G^\circ(\mathrm{T}, d, d^*) = \Delta G^\circ(\mathrm{T}, d', d'^*)$, and (c) is *perfectly isoenergetic* if both are true.

[6] This encoding assumes that the entire system is described as a binary string. For f_{tiles} there needs to be a DLOGTIME-computable predicate verifying $t \in \tilde{T}_i$, given input $|x| = n$, index i and tile t. We note that these inputs are describable in $O(\log n)$ bits. Likewise, for f_{in}, the predicate takes as input an index $0 \le i < n$ and $O(1)$ bits in defined indices $x_i, \ldots, x_{i+O(1)}$ (the extra $O(1)$ bits are to accommodate putting multiple input bits on a single tile). f_{out} acts on index i, a tile t and bit b. These predicates would be applied a polynomial number of times, in parallel, to verify the uniformity of the entire polynomial size SDC system.

3 Computational power of SDC

Previous work showed that SDC systems simulate finite state machines [20] and that MFE prediction is in logspace (L) [17] for $O(1)$ tiles per scaffold position. Here we show that, computationally, SDC systems are characterised exactly by the complexity class NL: they can solve any problem in that class and no more. The result requires both nondeterminism and a large number of tiles, polynomial in scaffold length N. We also get an L-completeness result for polynomially many tiles, but with a determinism restriction that seems syntactically strong, but many implemented programs satisfy it.

3.1 SDCs are simulated by logarithmic space machines

A Turing machine has a read-only input tape, write-once-only output tape and a read/write work tape. The class of languages decided by deterministic (resp., nondeterministic) Turing machines using workspace $O(\log n)$ on input length n is called deterministic logspace and denoted L, resp. nondeterministic logspace and denoted NL [11, 13].

▶ **Lemma 11.** *Given an SDC S, there is a nondeterministic logspace Turing machine M, whose accepting computations output the set of full mismatch-free configurations of S.*

Proof. M maintains a scaffold position counter $i \in \{1, \ldots, N\}$, on its worktape. Encodings are DLOGTIME computable (Definition 9), hence computable in logspace. First, M places any tile from T_1 on its output tape, and stores the right-side compute domain of that tile on its worktape, which needs only $O(\log |x|)$ bits since there are at most polynomially many domains since DLOGTIME is a subset of P (see Footnote 6). At scaffold position $i \geq 2$, M nondeterministically chooses any tile $t_{i,j}$ from tileset T_i if $t_{i,j}$'s left compute domain matches the right compute domain of the guessed tile at position $i-1$. Then M increments i, iterating the procedure. If $i = N$, and a matching tile was placed at position N, M accepts. Else if M reaches some position $\leq N$ where there is no matching tile to place, it rejects.

Given any mismatch-free configuration C, M outputs C by choosing each of its tiles successively, only ever writing one domain on its worktape; no mismatches implies that each successive tile is always a valid choice. Conversely, by construction, M can never choose a mismatching tile, and so can never output a configuration with a mismatch. ◀

▶ **Lemma 12.** *Let $L \subseteq \{0,1\}^*$, be decided by an SDC program (Definition 9). Then $L \in$ NL.*

The proof is in Appendix A.2.

3.2 Logarithmic space machines are simulated by SDCs

▶ **Lemma 13.** *Let $L \in$ NL. Then there is an SDC program that decides L.*

Proof. We will define the SDC program by describing the behavior of $f_{\text{tiles}}, f_{\text{in}}$ and f_{out} on an input $x \in \{0,1\}^*$ to the nondeterministic logspace Turing machine M_L that decides L. Let $n = |x|$ and consider the configuration graph of M_L on inputs of length n (as a reminder [11, 13]: $G = (V, E)$ is a directed graph with one node c for every possible (including unreachable) configuration c of the *worktape* of M_L on inputs of length n, and given a particular input x, there is an edge from node c to c' iff M_L goes from configuration c to configuration c' in a single step when running on x [11]). G has at most $m = n^{O(1)}$ nodes, since that is the number of configurations of M_L on inputs of length n [11, 13].

The program is given by $f_{\text{tiles}}(n) = (V \times \{i\} \times V)^m$: that is, the compute domains of the SDC are nodes of M_L's configuration graph, and the scaffold length is the number of nodes. On a specific input, $f_{\text{in}}(x)$ restricts these tilesets: for each $i \geq 2$, the set T_i has one

tile (c, i, c') for each directed edge (c, c') in G (which may depend on x). We let T_1 contain a single tile $(c, 1, c)$ where c is the configuration in which M_L is initialised on input x (noting that the left domain does not matter). Finally, f_{out} gives functions $f_i((c, i, c')) = \epsilon$ (the empty symbol) for $i < N$, and $f_N((c, N, c')) = 1$ if c' is an accepting configuration of M_L and ϵ otherwise. It is straightforward, albeit tedious, to verify these encodings are DLOGTIME computable, by iterating over all log-size pieces of the construction. ◀

3.3 An L characterisation

Many SDC programs implemented to date are *directed* (with exactly one target configuration), and have the property that their computation is not "too nondeterministic" (BitCopy, IsoenergeticBitCopy, Addition, Parity, MultiplyBy3, DivBy2, Counter, Rule110 and BalancedBrackets [20]), which we define here and which limits their power to be within the class L.

▶ **Theorem 14.** *Consider the class of SDC programs P where each SDC instance S of P has an anchor in position 1 ($|T_1| = 1$) and is* left-to-right deterministic*: no tileset T in S has two tiles with the same left compute domain, i.e. given $t_i \in T_i$, there is at most one t_{i+1} in T_{i+1} that matches it. The languages decided by such programs are exactly* L*.*

Proof sketch. ($\subseteq$ L) We use the proof from Lemma 12, except that there is only one choice at each step when enumerating mismatch-free configurations, so nondeterminism is not needed.

($\supseteq$ L) Again, the exact proof of Lemma 13 works here: the configuration graph G of any deterministic logspace Turing machine M has outdegree ≤ 1 at each node, so no two tiles in any tileset T_i can have the same left compute domain. ◀

We note that the previous theorem relates to Theorem 3 from Shalaby et al [17], which showed that computing the MFE of an SDC system, in (essentially) our bind/unbind model, is in L for $O(1)$ size tilesets. Here we generalise to polynomial size tilesets, but in bind/replace.

4 SDC bind/replace: $O(N^2)$ kinetics via the leftmost mismatch

In this section, we analyze how long SDC programs might take to run under the simple SDC bind/replace model. We start with a lemma helping us generalize a result from [20].

▶ **Lemma 15.** *Let $\mathcal{S} = (S, \xrightarrow{1}, R)$ be any SDC model where S is a BitCopy SDC, $\xrightarrow{1}$ is the same as in bind/replace (see Definition 5), and R has the following properties:*

1. *Transition rates are local: If $C \xrightarrow[i]{1} C'$, then $R(C, C')$ does not depend on anything outside the neighborhood $[i-1, i+1]$ in either C or C'.*
2. *More mismatches do not slow a transition: if $C_1 \xrightarrow[i]{1} C_1'$ and $C_2 \xrightarrow[i]{1} C_2'$ make the same replacement (i.e. $C_1[i] = C_2[i]$ and $C_1'[i] = C_2'[i]$), and $C_1[i]$ matches more of its neighbors than $C_2[i]$, then $R(C_1, C_1') \leq R(C_2, C_2')$.*

Then the configuration C_0 with a single mismatch in the leftmost position (i.e., a mismatch with the anchor) has the largest expected time to completion among all full configurations.

Proof sketch. Conceptually, BitCopy cannot finish until its leftmost mismatch disappears. Putting that mismatch far away from the end of the scaffold maximizes the expected time to completion. Having any other mismatches cannot slow down the process because any mismatch that does not disappear before the leftmost mismatch disappears will instead cancel with the leftmost mismatch as it moves to the right, causing the position of the new leftmost mismatch to jump forward. We formalize this in Appendix A.3 using coupling. ◀

This framework demonstrates the advantage of the SDC bind/replace model: we need only think about one particular configuration to analyze its kinetics. Later, in Section 6, we will use this to analyze a more complex version of the model. Here, we characterize the simple version.

▶ **Corollary 16.** *In the SDC bind/replace model on a* BitCopy *SDC with scaffold length N, the expected completion time from* any *configuration is less than N^2.*

Proof. By Lemma 15, we need only consider the configuration with a single mismatch in the leftmost position. By Lemma 28 (a more general result, shown later), this configuration takes $(N-1)^2$ expected steps, and therefore also expected time bounded by this (it has only 1 mismatch, each transition has rate 1, so each step takes expected time at most 1). ◀

5 The SDC bind/unbind model

5.1 Equilibrium in isoenergetic systems

Simple observation of the configuration energy shows that, in a perfectly isoenergetic (Definition 10) SDC in the SDC bind/unbind model, configurations will melt, that is, the empty configuration will be the most probable, at the critical point of $\mu \leq \Delta G^\circ_{\text{sd}} + \Delta G^\circ_{\text{cd}}$, when a tile attaching by one compute domain and one scaffold domain is unfavorable. In conditions where $N\Delta G^\circ_{\text{sd}} - N\mu + (N-1)\Delta G^\circ_{\text{cd}} < 0$, the target configuration (full, mismatch-free) will be the minimum free energy configuration and thus the most probable. However, being the most probable amongst many configurations does not necessarily mean that the target configuration is probable overall. The overall equilibrium probability of the target configuration depends on the full partition function, which, in general, must be calculated algorithmically [17]. Here, we consider analytic approximations.

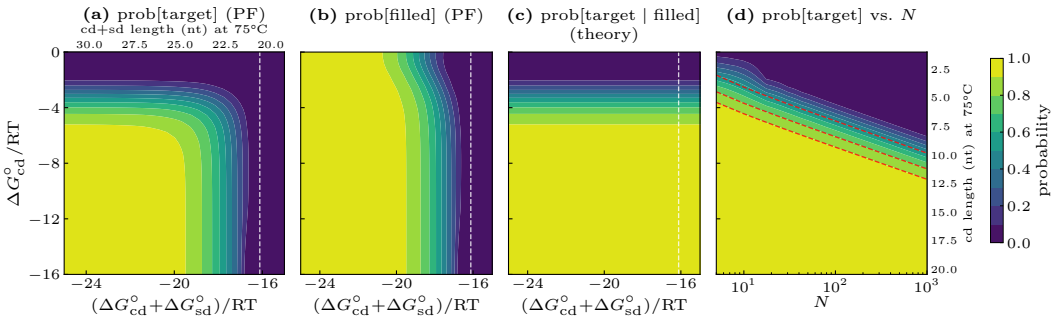

Figure 2 Equilibrium probabilities for BitCopy in the bind/unbind model. **(a)** Probability of the target configuration for $N = 20$, using the full partition function algorithm. Dashed line shows $\Delta G^\circ_{\text{cd}} + \Delta G^\circ_{\text{sd}} - \mu = 0$. **(b)** Probability that all scaffold sites are filled ($N = 20$, using full partition function). **(c)** prob[target | filled] from Theorem 17. **(d)** prob[target] vs. N at fixed $\Delta G^\circ_{\text{sd}}/\text{RT} = -19$; dashed red lines show constant prob[target | filled] from Theorem 17.

▶ **Theorem 17.** *Let S be a bind/unbind* BitCopy *SDC system of length N with all tiles at equal concentration [t]. Conditioned on all scaffold sites being filled, the equilibrium probability of the target configuration is $p = (1 + e^{\Delta G^\circ_{cd}/\text{RT}})^{-(N-1)}$.*

Proof. We take the approach of [20], but seek an exact result for any p. Recall that the first position in S is an anchor, with only one tile, and every other position has two, copying either 0 or 1. Each mismatch causes an increase in free energy of $-\Delta G^\circ_{\text{cd}}$. For this analysis,

we choose to set the free energy of the target state, with no mismatches, to be 0, then a state with m mismatches will differ in free energy from that target state by $-m\Delta G^\circ_{\text{cd}}$. As there are $(N-1)$ possible mismatches and therefore $\binom{N-1}{m}$ states with m mismatches, the partition function (i.e., the sum of all Boltzmann weights) will be $Z = \sum_{m=0}^{N-1} \binom{N-1}{m} e^{+m\Delta G^\circ_{\text{cd}}/\text{RT}}$.

In this regime, as the free energy of the target state is simply 0, the equilibrium probability of the target is $p = 1/Z$. The binomial theorem gives: $p = Z^{-1} = (1 + e^{\Delta G^\circ_{\text{cd}}/\text{RT}})^{-(N-1)}$. ◀

▶ **Corollary 18.** *Conditioned on all N scaffold positions being filled, the ΔG°_{cd} required to maintain a fixed target state probability p at equilibrium scales as $-O(\ln N)$.*

Proof. Rearranging, $\Delta G^\circ_{\text{cd}}/\text{RT} = \ln\left(p^{-1/(N-1)} - 1\right)$. Expand $p^{-1/(N-1)} \approx 1 - \ln p/(N-1)$ for large N, giving $\Delta G^\circ_{\text{cd}}/\text{RT} \approx \ln(-\ln p) - \ln(N-1)$ and noting p is constant wrt N. ◀

An implication of these results is that we may choose some desired $p > 0.5$ and the system and input will be easy to prepare experimentally, since there is no need to prepare particular configuration(s). Indeed components can be mixed haphazardly, heated up (to melt everything) and cooled down to give the target high enough probability.

5.2 Kinetics in isoenergetic systems

5.2.1 Simulations and the phases of growth behaviours

To understand SDC bind/unbind kinetics, we implemented the model in the Rgrow kinetic simulator [9]. We ran simulations of BitCopy at fixed $[t] = 100\,\text{nM}$, measuring "completion time" as the time taken to reach within 5% of the equilibrium target probability across many scaffolds. Simulations showed three distinct regimes, summarized in Figure 3b. At low N, with weak CDs and strong SDs, computation took place mainly via a "walking mismatch" phase: scaffolds quickly filled with potentially mismatching tiles, remained at high occupancy, and evolved primarily through single detachments followed by attachment, approximating the behavior of the SDC bind/replace model. We describe this phase in Section 5.2.2.

With stronger CDs and weaker SDs, however, behavior shifted: the number of mismatches per scaffold was often nearly zero even when the number of scaffolds in the target state was low; while mostly filled, many scaffolds became trapped for long periods of time in an all-incorrect state with an empty input site. We briefly discuss this phase in Remark 21.

The empty input behavior suggested a simple change: if instead of using a completely uniform SD strength, we set the SD at the left-most input location to $\Delta G^\circ_{\text{id}} = \Delta G^\circ_{\text{sd}} + \Delta G^\circ_{\text{cd}}$, then attachment of the input tile would be favorable even without a bond with its neighbor. With this change, for sufficiently strong $\Delta G^\circ_{\text{cd}}$ and weak $\Delta G^\circ_{\text{sd}}$, the system exhibits a "left-to-right growth" phase: computation takes place on an initially empty scaffold, with the target state growing via a biased random walk of attachments and detachments at a growth front of a contiguous region of tiles starting at the left-most input. Completion times are much faster than other phases, and scale more slowly with N; however, as N increases, the energy conditions required for the phase narrow. We investigate this phase in Section 5.2.3.

5.2.2 Walking-mismatch behaviour: $O(N^3)$ for fixed target probability

▶ **Theorem 19.** *In a perfectly isoenergetic (Definition 10) BitCopy SDC bind/unbind system, in conditions where growth proceeds via a mostly-filled configuration where single detachments are quickly filled by attachment events, as ΔG°_{cd} increases in strength, the system behavior will approximate the SDC bind/replace model, with the replacement of a tile attached by one compute domain bond taking an expected time $k_f^{-1} u_0^{-1} e^{-(\Delta G^\circ_{sd} + \Delta G^\circ_{cd})/\text{RT}} + (2k_f[t])^{-1}$.*

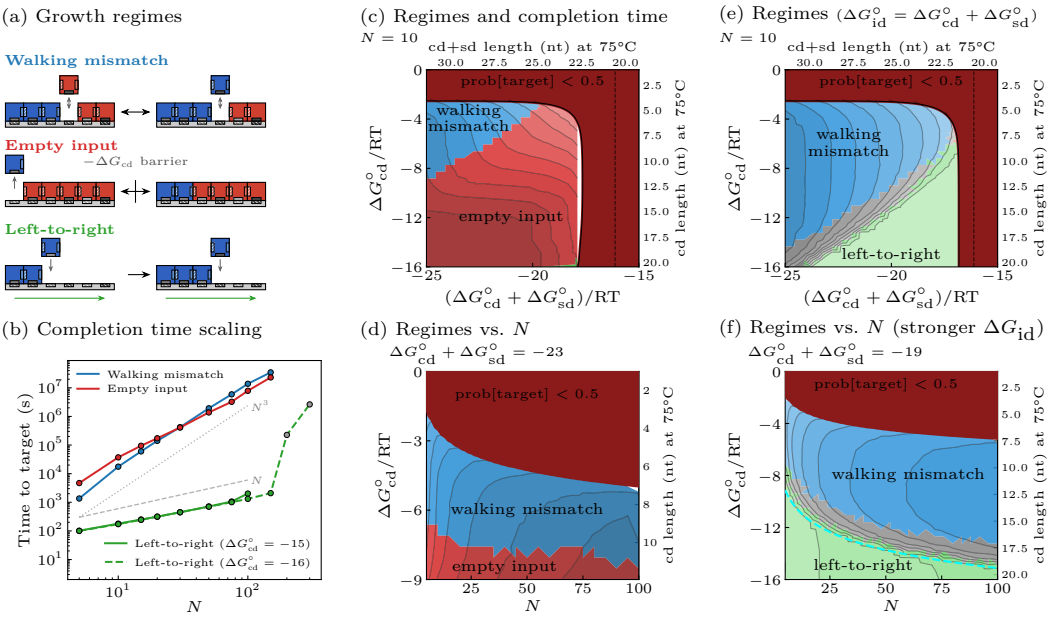

Figure 3 Simulations and theory for kinetic phases of the bind/unbind model. **(a)** Three phases of behavior with prob[target] > 0.5. **(b)** For fixed values of ΔG°_{sd}, and ΔG°_{cd} set to obtain a particular target probability, scaling of completion time with N (time to 5% less than equilibrium target probability), for simulations of BitCopy. **(c)–(d)** Observed behavior and completion time in simulations. Contour lines represent 5-fold changes in completion time. **(e)–(f)** Observed behavior and completion time in simulations when the input SD strength is increased to $\Delta G^\circ_{id} = \Delta G^\circ_{sd} + \Delta G^\circ_{cd}$. Grey regions represent ambiguous/mixed behavior; dashed blue line is $p_{ltr} = 0.95$ from Theorem 22.

Proof. Consider a single site on a full scaffold, where the adjoining sites are both filled. The expected time for the tile to detach will be $k_f^{-1} u_0^{-1} e^{-(\Delta G^\circ_{sd} + b\Delta G^\circ_{cd})/RT}$, where b is the number of compute domain bonds the tile makes with adjoining tiles. Once detached, another tile will attach with expected time $(2k_f[t])^{-1}$ (as there are two possible attachments). If we assume ΔG°_{cd} is strong (very negative), then we may assume that a tile attached by two CD bonds will detach at a negligible rate, and a tile attached by zero will detach immediately. Thus, a tile making two bonds may not be replaced by a detach-attach sequence, a tile attached by zero bonds may be replaced by any other tile, and a tile attached by one bond may be replaced by a tile making one or two bonds. ◀

▶ **Corollary 20.** *In an isoenergetic (Definition 10) BitCopy SDC bind/unbind system, in conditions where a mostly-filled configuration evolves through single detachments that are quickly filled by attachment events, for a fixed probability of the target state at equilibrium and fixed ΔG°_{sd}, with ΔG°_{cd} set to obtain the target probability, the time for a scaffold to reach the target state scales as $O(N^3)$.*

Proof. In SDC bind/replace, the expected time for a scaffold to reach the target state is $O(N^2)$ (Corollary 16). However, to maintain a fixed equilibrium target state probability, $\Delta G^\circ_{cd}/RT \leq \ln(-\ln p) - \ln(N-1)$ (Corollary 18); this means that the expected time per replacement event scales as $O(N)$, and thus the overall expected time for a scaffold to reach the target state scales as $O(N^3)$. ◀

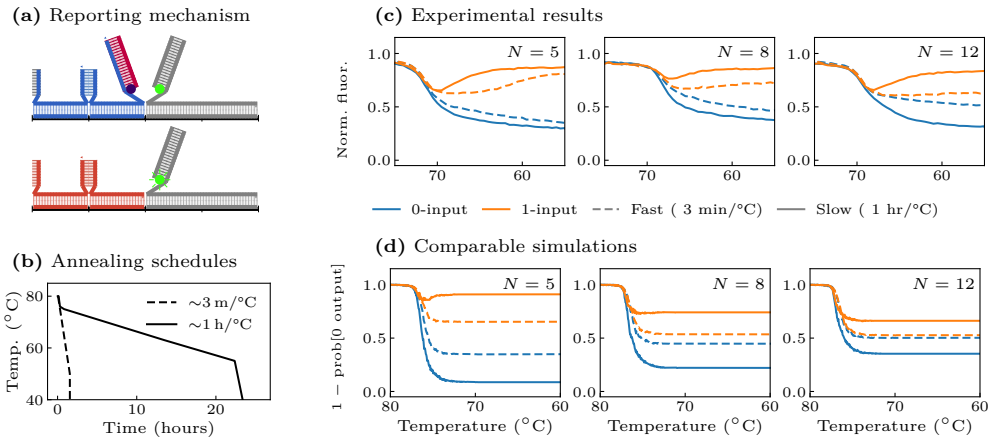

Figure 4 BitCopy simulations and experiments for the bind/unbind model. **(a)** Experiments use a fluorophore/quencher reporter at the last position, with a fluorophore on one of the output tile types. 0-bit output results in a decrease in bulk fluorescence, while 1-bit output (or an empty output) remains fluorescent. **(b)** Two annealing protocols used for experiments. **(c)** Experimental bulk fluorescence during anneals at two rates, for scaffold lengths $N = 5, 8, 12$, of IsoEnergeticBitCopy. Unpublished data courtesy of authors of [20], using extended-length reporting domains along with protocols in [20]. **(d)** Corresponding rgrow kinetic simulations with averaged, uniform domain energies, using nearest-neighbor parameters from [15].

▶ Remark 21. With uniform $\Delta G^\circ_{\text{sd}}$, a region of incorrect tiles next to an empty left-most position allows a kinetic trap that cannot arise elsewhere on the scaffold, as illustrated in Figure 3a. The input tile has only one compute domain, and thus, in this case, will attach by only $\Delta G^\circ_{\text{sd}}$, unfavorably. At this point, in order to form a bond with a matching tile in the next location, the incorrect tile, attached by $\Delta G^\circ_{\text{sd}} + \Delta G^\circ_{\text{cd}}$, would need to detach, also an unfavorable step. This results in a two-step kinetic barrier of $\Delta G^\circ_{\text{sd}} - \mu$ for the first configuration and $-\Delta G^\circ_{\text{cd}}$ for the second. However, it becomes less relevant with increasing N since it occurs only in one scaffold location.

5.2.3 Left-to-right growth: $O(N)$ in shrinking window of conditions

Here we consider *left-to-right growth*, that starts at an anchor tile on the left, grows to the right through a (possibly biased) random walk, but finishes on the right hand side of the scaffold before other tiles can nucleate to the right of the growth front. It is essentially a race of intended growth, against adversarial nucleation. The proof is in Appendix A.4.

▶ **Theorem 22.** *In a BitCopy SDC system in the bind/unbind model with input scaffold domain strength ΔG°_{id} such that $\Delta G^\circ_{id} - \mu < 0$ and scaffold domain strength $\Delta G^\circ_{sd} - \mu \gg 0$, a scaffold will grow via a biased random walk from left to right, with mean growth rate $r_g = k_f\left([t] - u_0 e^{(\Delta G^\circ_{sd} + \Delta G^\circ_{cd})/\text{RT}}\right)$, with probability p_{ltr} such that $\ln p_{ltr} \approx -\frac{1}{2} k_f r_g^{-1} N^2 [t]^2 u_0^{-1} e^{-\Delta G^\circ_{sd}/\text{RT}}$.*

▶ **Corollary 23.** *In the left-to-right regime, the mean time for a scaffold to complete is $N r_g^{-1}$.*

5.2.4 SDC bind/unbind simulations and experimental results

While we have primarily considered the evolution of systems at constant binding energy conditions, experimental implementations of SDC used temperature ramps of varying speed from a high, above-melting temperature to a low temperature where no further changes took

place [20]. Taking ΔG° to be of the form $\Delta H^\circ - \mathrm{T}\Delta S^\circ$, where both terms are temperature-independent, and, while sequence-dependent, strengthen roughly linearly with domain length, a decrease in temperature corresponds with a decrease (strengthening) of both $\Delta G^\circ_{\mathrm{sd}}$ and $\Delta G^\circ_{\mathrm{cd}}$. Below the melting temperature, systems transition through increasing prob[target], but decreasing detachment rates, and thus increasing expected completion time (experiments in left-to-right conditions have not yet been attempted). Computation takes place in the temperature window where both prob[target] and detachment rates are sufficiently high; below this window, on experiment time scales, scaffolds remain "frozen". Optimal temperature schedules to move close to equilibrium as quickly as possible remain an open question.

Experimentally, as shown in Figure 4a, observation of system state uses bulk fluorescence measurements, with the last ("output") tile of one type (0 for BitCopy) labelled with a long domain complementary to a quencher-modified strand, and an additional tile, binding to a corresponding fluorophore-modified strand and with an extended scaffold domain, attached to the right of the output position. Bulk fluorescence, measured on a qPCR machine, decreases when a high percentage of scaffolds have a 0 output tile attached, and remains high if there are a high percentage of scaffolds where the position is empty or has a 1 output tile attached.

Experiments and simulations in Figure 4c-d, show qualitative similarities. In both, the probability of a 1/0 output tile evolves depending on input only within a certain temperature range. An initial fluorescence decrease with input 1 is seen in some slow anneals in experiments and simulations, and as N increases, the annealing time needed for high yield increases.

However, experiments show changing fluorescence over a wider temperature range, a lower melting temperature, and a variance in the temperature of fluorescence changes depending on specific output position. Unlike simulations, where scaffold domains are assumed to be uniform strength, experimental implementations have so far used M13-based scaffolds with widely varying scaffold domain energies [20]; the energetic effects of junctions between bound compute and scaffold domains have also not yet been modelled.

6 Resilience to variable domain strengths

When designing DNA computers, there can be significant inaccuracy or non-ideality in the binding energies of strands from many sources: inaccuracies in our models, imperfections in DNA sequence design, or even using non-designed biological sequences for cost reasons. Here, we study how the SDC model performs when accounting for these discrepancies in scaffold or compute domain strength, particularly when scaling to large scaffold size N.

6.1 Addressing variable compute domain energies by programming an isoenergetic energy landscape

When scaling to larger scaffold length N, non-isoenergetic compute domains are a particularly prominent source of undesirable behaviour, due to DNA sequence design inaccuracies/non-idealities. The fundamental problem is that compute domains typically [20] appear multiple times along the scaffold. Hence, a small difference in compute domain strength may accumulate up to $N-1$ times, and cause unintended structures with mismatches to become more thermodynamically favourable than intended target structures, as shown in Figure 5.

One approach would be good DNA sequence design: to scale up we would design more isoenergetic compute domain DNA sequences. The number of sequences increases exponentially with length and by Corollary 18 we expect compute domains to be of length $O(\log N)$ anyway, so the approach seems reasonable at first. However, redesigning sequences is not always desirable: perhaps the lab has a known "good" set of compute domains and

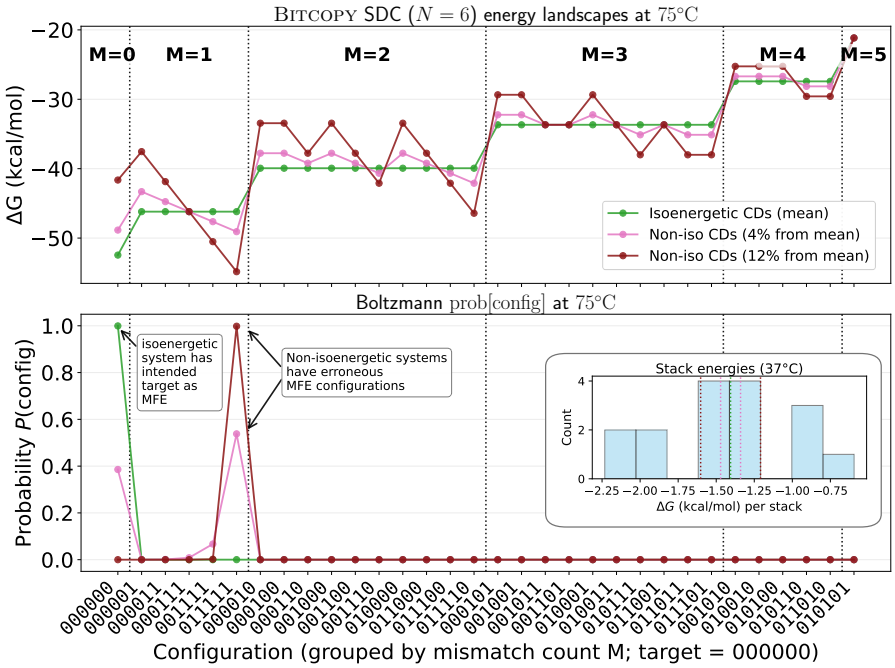

Figure 5 Energy landscapes of BitCopy SDC systems, with input bit `0` and scaffold length $N = 6$, that are identical except for differences in compute domain energies. Each bit represents a tile, the correct target is `000000`. Inset histogram: Distribution of nearest-neighbour stack energies at 37°C, for the 16 stacks with 4 DNA bases [15]. Free energy $\Delta G_c(75°\text{C}, C)$ (top panel) and Boltzmann probability prob$[C]$ (bottom) at equilibrium, for isoenergetic compute domains (green) and non-isoenergetic domains at 4% and 12% from the mean stack energy (pink and dark red; in each case `0-0` binding domains are weaker (more positive) than `1-1`). Arrows mark probability maxima on non-iso curves (erroneous MFE configurations). Compute domains are of length 12, and scaffold domains are isoenergetic and of length 24 bases, as used in previous experimental work [20].

we'd like to repurpose them for a larger system. Also, inaccuracies in the nearest neighbour model of DNA binding and in our ability to assay structures' energies mean that perfectly isoenergetic design is a huge challenge.

In this section, we take a more systematic approach. We show it is possible to sculpt SDC energy landscapes using the logic of SDC tilesets (programs). In previous work [20], a program called IsoenergeticBitCopy was experimentally implemented on a length $N = 25$ scaffold. The program is designed so that both of its two target configurations are approximately isoenergetic, as are configurations with the same number of compute domain mismatches, which in turn makes the energy landscape close to the ideal (green plateaus in Figure 5). IsoenergeticBitCopy outperformed another non-isoenergetic program, Addition, for $N = 25$. That work left open the interesting problem of whether Addition, or other SDC programs that repeat the same tile logic at each scaffold position, could be made to be isoenergetic. Theorem 25 gives a positive answer.

An implication of an SDC being perfectly compute-domain isoenergetic (Definition 10) is that for all m, $1 \leq m \leq N - 1$, there is a constant $c_m \in \mathbb{R}$ such that all full configurations C with m mismatches have identical free energy $c_m = \Delta G_c(\text{T}, C)$, assuming fixed temperature T and that mismatches have the same, e.g. 0, free energy. Here, we loosen things a little to state our main theorem of this section, which is proved in Appendix A.5.

▶ **Definition 24.** For bind/unbind SDC S, let $\epsilon = \max_{d,d' \in D_{\text{cd}}} \Delta G^{\circ}_{\text{cd}}(\text{T}, d, d^*) - \Delta G^{\circ}_{\text{cd}}(\text{T}, d', d'^*)$ be the largest difference between compute domain binding energies. S is *almost compute-domain isoenergetic* if there is a constant c independent of N such that all full configurations with m mismatches have free energy within additive $c(m+1)\epsilon$ of each other.

▶ **Theorem 25.** *Given an SDC bind/unbind system S with scaffold length N, let D_R be the set of distinct compute domains appearing on the right sides of tiles in S, i.e., $D_R = \{r \mid \exists l, i : (l, i, r) \in T_i\}$. Assume that this value is the same for the left sides of tiles, i.e. that all compute domains appearing on tiles are capable of binding. Then there is an almost compute-domain isoenergetic SDC $\overline{S}$ that simulates S with scale factor $k = |D_R|$ (Definitions 24 and 30).*

6.2 Kinetics under adversarial scaffold domain strengths

Here, we generalise the bind/replace model, and prove formal guarantees about the scaling of SDC kinetics under varying scaffold domain strengths. Proofs are in Appendix A.6.

▶ **Definition 26** (energetic SDC bind/replace model)**.** Given an SDC S, the *energetic SDC bind/replace* model $(S, \xrightarrow{1}, R)$ is defined with the same reachability $\xrightarrow{1}$ as bind/replace (Definition 5). However, the replacement rate $R(C, C')$ for $C \xrightarrow{1} C'$ where $C'[i]$ replaces $C[i] = (l_i, p_i, r_i) \neq \emptyset$ is similar to the detachment rate from the bind/unbind model: $R(C, C') = \frac{k_f u_0}{t'} \text{e}^{(\Delta G^{\circ}(\text{T}, p_i, {p_i}^*) + \Delta G^{\circ}(\text{T}, r_{i-1}, l_i) + \Delta G^{\circ}(\text{T}, r_i, l_{i+1}))/\text{RT}}$, where t' is the number of tiles that could have replaced $C[i]$, and other parameters are as in Definition 7. When $C[i] = \emptyset$ (for binding), we maintain the bind/replace convention that $R(C, C') = 1$.

This model captures some of the nuance of having binding domains with differing energies. SDC programs typically do computation from left to right, so it is desirable for good kinetics to have stronger scaffold domains closer to the left. Here we study what happens in the opposite, somewhat adversarial, regime, where scaffold domains become stronger to the right. We first show how to translate physical domain energies into mathematical probabilities.

▶ **Lemma 27.** *In the biased SDC bind/replace model, for a configuration with a single mismatch between tiles t_i and t_{i+1} that can be replaced, the probability that the next replacement step replaces t_{i+1} is given by $\left(1 + e^{\Delta G_l - \Delta G_r}\right)^{-1}$, where $\Delta G_l = \left(\Delta G^{\circ}(\text{T}, p_i, {p_i}^*) + \Delta G^{\circ}(\text{T}, r_{i-1}, l_i)\right)/RT$ and $\Delta G_r = \left(\Delta G^{\circ}(\text{T}, p_{i+1}, {p_{i+1}}^*) + \Delta G^{\circ}(\text{T}, r_{i+1}, l_{i+2})\right)/RT$ give the total binding strength (one scaffold domain and possibly one compute domain) of t_i and t_{i+1}.*

By focusing on the probability of moving forward in a given step, we can analyze simpler, more purely mathematical processes, and use these results to reason about SDC runtimes.

▶ **Lemma 28.** *Consider the following biased random walk (representing the movement of the last mismatch in BitCopy): There are $n = N - 1$ possible locations the mismatch can occupy, numbered from 0 to $n - 1$. It starts in the leftmost position (position 0). Each step, the mismatch moves to an adjacent position. If the mismatch is in position 0, it always moves right. Otherwise, it moves right with probability p and left with probability $1 - p$, until it moves right from position $n - 1$, ending the process at position n (representing the mismatch moving off the end of the scaffold). The expected number of steps s that this process takes is given by $s = n^2$ if $p = 0.5$ and $s = \frac{2p(1-p)}{(2p-1)^2} \cdot \left(\left(p^{-1} - 1\right)^n - 1\right) + \frac{n}{2p-1}$ otherwise.*

The following theorem tells us that the system retains its speed even in the face of domain energies that increase along the entire scaffold (leading to backward bias). Even if this total energy differential increases logarithmically, the kinetics still operate in polynomial time. Corollary 18 shows that we only need domains of length $O(\ln N)$ for good thermodynamic properties, and $O(\ln N)$-length domains cannot differ in energy by more than $O(\ln N)$.

▶ **Theorem 29.** *Fix some temperature* T*, and let* $\mathcal{S} = (S, \xrightarrow{1}, R)$ *be a biased SDC bind/replace model where* S *is a* BitCopy *SDC of length* N*. Let compute domains have the same free energy (i.e.,* $\Delta G^\circ_{cd}(\mathrm{T}, 0, 0^*) = \Delta G^\circ_{cd}(\mathrm{T}, 1, 1^*)$*), and scaffold domains become stronger linearly from left to right: that is, if* p_j *is the scaffold domain in position* j*, then* $\Delta G^\circ_{sd}(\mathrm{T}, p_j, {p_j}^*) = a - bj$ *for some constants* a, b *with* $b > 0$*. Then the expected time to completion for* $\mathcal{S}$ *from any full configuration is* $O(N^2)$ *if* $b \in O\left(\frac{1}{N}\right)$*, and is polynomial in* N *if* $b \in O\left(\frac{\ln N}{N}\right)$*.*

References

1 David A Mix Barrington, Neil Immerman, and Howard Straubing. On uniformity within NC^1. *Journal of Computer and System Sciences*, 41(3):274–306, 1990. doi:10.1016/0022-0000(90)90022-d.

2 Charles H. Bennett. Logical reversibility of computation. *IBM Journal of Research and Development*, 17(6):525–532, 1973. doi:10.1147/rd.176.0525.

3 Charles H. Bennett. The thermodynamics of computation—a review. *International Journal of Theoretical Physics*, 21(12):905–940, 1982. doi:10.1007/bf02084158.

4 Frits Dannenberg, Katherine E Dunn, Jonathan Bath, Marta Kwiatkowska, Andrew J Turberfield, and Thomas E Ouldridge. Modelling DNA origami self-assembly at the domain level. *The Journal of chemical physics*, 143(16), 2015. doi:10.1063/1.4933426.

5 Robert M Dirks, Justin S Bois, Joseph M Schaeffer, Erik Winfree, and Niles A Pierce. Thermodynamic analysis of interacting nucleic acid strands. *SIAM Review*, 49(1):65–88, 2007. doi:10.1137/060651100.

6 David Doty, Niels Kornerup, Austin Luchsinger, Leo Orshansky, David Soloveichik, and Damien Woods. Harvesting Brownian motion: Zero energy computational sampling, 2023. Arxiv preprint: arXiv:2309.06957. doi:10.48550/arXiv.2309.06957.

7 David Doty, Trent A Rogers, David Soloveichik, Chris Thachuk, and Damien Woods. Thermodynamic binding networks. In *DNA23: The 23rd International Conference on DNA Computing and Molecular Programming*, volume 10467 of *LNCS*, pages 249–266. Springer, 2017. doi:10.1007/978-3-319-66799-7_16.

8 Katherine E Dunn, Frits Dannenberg, Thomas E Ouldridge, Marta Kwiatkowska, Andrew J Turberfield, and Jonathan Bath. Guiding the folding pathway of DNA origami. *Nature*, 525(7567):82–86, 2015. doi:10.1038/nature14860.

9 Constantine Evans and Angel Cervera Roldan. Rgrow tile self-assembly simulator, December 2025. doi:10.5281/zenodo.15270539.

10 Jonathan Lee Tin Wah, Christophe David, Sergii Rudiuk, Damien Baigl, and André Estevez-Torres. Observing and controlling the folding pathway of DNA origami at the nanoscale. *ACS nano*, 10(2):1978–1987, 2016. doi:10.1021/acsnano.5b05972.

11 Cristopher Moore and Stephan Mertens. *The Nature of Computation*. Oxford University Press, 2011. doi:10.1093/acprof:oso/9780199233212.001.0001.

12 Niall Murphy and Damien Woods. AND and/or OR: Uniform polynomial-size circuits. In *MCU 2013: Machines, Computations and Universality. Electronic Proceedings in Theoretical Computer Science (EPTCS)*, volume 128, pages 150–166, 2013. doi:10.4204/EPTCS.128.20.

13 Christos H Papadimitriou. *Computational Complexity*. Addison Wesley, 1994.

14 Paul WK Rothemund. Folding DNA to create nanoscale shapes and patterns. *Nature*, 440(7082):297–302, 2006. doi:10.1038/nature04586.

15 John SantaLucia Jr and Donald Hicks. The thermodynamics of DNA structural motifs. *Annu. Rev. Biophys. Biomol. Struct.*, 33:415–440, 2004. doi:10.1146/annurev.biophys.32.110601.141800.

16 Fabian Schneider, Natalie Möritz, and Hendrik Dietz. The sequence of events during folding of a DNA origami. *Science advances*, 5(5):eaaw1412, 2019. doi:10.1126/sciadv.aaw1412.

17 Ahmed Shalaby, Chris Thachuk, and Damien Woods. Minimum free energy, partition function and kinetics simulation algorithms for a multistranded scaffolded dna computer. In *DNA 29: Proceedings of the 29th International Meeting on DNA Computing and Molecular Programming*, volume 276 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 1:1–1:22, 2023. doi:10.4230/LIPIcs.DNA.29.1.

18 Jean-Philippe J Sobczak, Thomas G Martin, Thomas Gerling, and Hendrik Dietz. Rapid folding of DNA into nanoscale shapes at constant temperature. *Science*, 338(6113):1458–1461, 2012. doi:10.1126/science.1229919.

19 David Soloveichik and Erik Winfree. Complexity of self-assembled shapes. *SIAM Journal on Computing*, 36(6):1544–1569, 2007. doi:10.1137/s0097539704446712.

20 Tristan Stérin, Abeer Eshra, Janet Adio, Constantine Evans, and Damien Woods. A thermodynamically favoured molecular computer: Robust, fast, renewable, scalable, 2025. bioRχiv preprint: bioRxiv doi.org/10.1101/2025.07.16.664196.

21 Boya Wang, Cameron Chalk, David Doty, and David Soloveichik. Molecular computation at equilibrium via programmable entropy. *Science Advances*, 12(2):eadx3969, 2026. doi:10.1126/sciadv.adx3969.

22 Erik Winfree. Simulations of computing by self-assembly. Technical Report CS-TR:1998.22, Caltech, May 1998. URL: https://www.dna.caltech.edu/Papers/simulation.pdf.

A Appendix

A.1 Proof of Lemma 8

Proof of Lemma 8. In an SDC bind/unbind system, every tile attachment event $C \xrightarrow{1} C'$ of tile t_i has a corresponding reverse (detachment) event $C' \xrightarrow{1} C$ of tile t_i, and there are no other events in the system. The system is in chemical equilibrium, that is, the concentration of every configuration is constant over time, when every such bind/unbind pair of events has equal forward and reverse rates, i.e. $\text{prob}[C] r_{\text{att}} = \text{prob}[C'] r_{\text{det}}$ for such a triple t_i, C, C':

$$\text{prob}[C'] = \text{prob}[C] \text{e}^{-(\Delta G^\circ(\text{T}, s_i, s_i^*) - \mu(t_i) + \Delta G^\circ(\text{T}, r_{i-1}, l_i) + \Delta G^\circ(\text{T}, r_i, l_{i+1}))/\text{RT}} . \tag{2}$$

Furthermore, any configuration C is reachable from the empty configuration by $\leq N$ tile additions, at most one per scaffold position. Thus the equilibrium probability of C is the equilibrium probability of $C_\emptyset$ times the product of the Equation (2) weight from each step.

Applying Equation (2), starting with $C = C_\emptyset$ one tile bind step at a time, implies that the energy terms involved for each tile binding event are summed: Each additional tile binding contributes a free energy term of $\Delta G^\circ(\text{T}, s_i, s_i^*) - \mu(t)$, and each matching compute domain (binding) between adjacent tiles contributes a free energy term of $\Delta G^\circ(\text{T}, r_i, l_{i+1})$ (each bond contributes exactly once). These N binding event terms sum to give $\Delta G_{\text{c}}(\text{T}, C)$ in Equation (1). Thus, the equilibrium probability of an assembly C is $\text{prob}[C] = \text{prob}[C_\emptyset] e^{-\Delta G_{\text{c}}(\text{T}, C)/\text{RT}}$.

Since configuration probabilities should sum to 1, we set $\sum_{C \in \mathcal{C}} \text{prob}[C] = 1$, hence $\text{prob}[C_\emptyset] \sum_{C \in \mathcal{C}} e^{-\Delta G_{\text{c}}(\text{T}, C)/\text{RT}} = \text{prob}[C_\emptyset] Z = 1$, and $\text{prob}[C_\emptyset] = Z^{-1}$, yielding the claim: $\text{prob}[C] = \text{e}^{-\Delta G_{\text{c}}(\text{T}, C)/\text{RT}} \cdot Z^{-1}$. ◀

A.2 Proof of Lemma 12

Proof of Lemma 12. We describe a nondeterministic logspace machine M on input $x \in \{0,1\}^*$, simulates the SDC to decide x. Given $x \in \{0,1\}^*$, M first generates an SDC instance via the SDC program's DLOGTIME-computable encoding functions. M does not explicitly write out the SDC as it does not have enough workspace, however using a well-known procedure [13] it can generate $O(\log |x|)$-size parts of the SDC on the fly any

time they are needed. Since $N = |x|^{O(1)}$ by DLOGTIME-computable encodings, we get that $O(\log |x|) = O(\log N)$, and hence individual domains and tiles can be generated in this way, which is all that is required to run the construction in the proof of Lemma 11. Via Lemma 11, the nondeterministic logspace machine is capable of outputting full, mismatch-free output configurations. However, instead of outputting the full configuration it merely outputs the decoded output (via the uniformity condition), which in this case is a single bit: 1 if $x \in L$ and 0 otherwise. ◀

A.3 Proof of Lemma 15

Proof. Let C be any full configuration. Since mismatch energy never increases (by definition of $\xrightarrow{1}$), starting from C_0 guarantees exactly one mismatch until completion. We can couple the evolution of C and C_0 (using these names from here on to refer to these evolving configurations) as follows: Whenever the mismatch in C_0 is further left than the leftmost mismatch of C, we pause C and let C_0 run until they are in the same position (which must eventually happen). Whenever the two trajectories have their leftmost mismatch at the same index, we force the two trajectories to move this mismatch in tandem: specifically, suppose the mismatch is between tiles t_i and t_{i+1}. As soon as C replaces either of these tiles (switching its domains from 0 to 1 or 1 to 0), C_0 replaces the same tile. Because of the conditions placed on R, these replacement rates from C must be the same or higher than those of C_0: the replacement of t_i has the same rate by the first condition, and the replacement of t_{i+1} has the same or a faster rate by both conditions (t_{i+1} and t_{i+2} may possibly mismatch in C, but they do not in C_0). This continues until no mismatches remain.

The resulting trajectory of C_0 must take time at least that of C, as it has all the same replacement steps (possibly some extra). From the perspective of C_0, the trajectory was either sampled accurately or sped up: the mismatch may have moved more quickly than it would have and may have been more likely to move right, both of which result in a faster trajectory. It follows that the expected time from C_0 is at least as high as that from C. ◀

A.4 Proof of Theorem 22

Proof of Theorem 22. Whether a scaffold will grow by a left-to-right random walk depends on two competing processes: the random walk of attachments by and detachments at a growth front by 1CD + 1SD, and the probability of a stable region of tiles "nucleating" at some other point before it is covered by the left-to-right growth.

Left-to-right growth, given the $\Delta G^\circ_{\text{id}} - \mu < 0$ assumption, will have no barrier to starting. It will then proceed via individual attachments at rate $k_f[t]$, and detachments at rate $k_f e^{(\Delta G^\circ_{\text{sd}} + \Delta G^\circ_{\text{cd}})/\text{RT}}$, resulting in a mean growth rate of $r_g = k_f \left([t] - u_0 e^{(\Delta G^\circ_{\text{sd}} + \Delta G^\circ_{\text{cd}})/\text{RT}}\right)$. Here, we ignore detachments at points other than the growth front, which would occur at the lower rate of $k_f e^{(\Delta G^\circ_{\text{sd}} + 2\Delta G^\circ_{\text{cd}})/\text{RT}}$, and would be very likely to be quickly filled in again.

Formation of stable regions at other points requires a nucleation process, with an initially unfavorable attachment. However, the critical nucleus, or least-favored state, is a single tile (attached by $\Delta G^\circ_{\text{sd}}$), and all subsequent attachments are favorable. Thus, nucleation can be treated as similar to the kinetic trapping model of errors in DNA tile self-assembly [22]: the rate of a tile initially attaching at a given site is $k_f[t]$, and the probability of an additional tile attaching on either side before the initial tile detaches is $2[t]/(2[t] + u_0 e^{\Delta G^\circ_{\text{sd}}/\text{RT}})$, as we are in a regime where $[t] \ll u_0 e^{\Delta G^\circ_{\text{sd}}/\text{RT}}$, this probability is approximately $2[t]u_0^{-1} e^{-\Delta G^\circ_{\text{sd}}/\text{RT}}$, and thus, the rate of nucleation at a site is $r_{\text{nuc}} \approx k_f 2[t]^2 u_0^{-1} e^{-\Delta G^\circ_{\text{sd}}/\text{RT}}$.

There are N sites in the system, and the left-to-right growth will complete in mean time Nr_{g}^{-1}, meaning that the total sites and time available for nucleation is $N^2/(2r_{\mathrm{g}})$. The probability that no nucleation takes place during the left-to-right growth is thus $p_{\mathrm{ltr}} = \exp(-r_{\mathrm{nuc}}N^2/(2r_{\mathrm{g}}))$, and thus $\ln p_{\mathrm{ltr}} \approx -\frac{1}{2}k_{\mathrm{f}}r_{\mathrm{g}}^{-1}N^2[t]^2u_0^{-1}e^{-\Delta G^\circ_{\mathrm{sd}}/\mathrm{RT}}$. ◀

A.5 Definition 30 and Proof of Theorem 25

▶ **Definition 30.** Let S be an SDC with scaffold length N. Given another SDC $\overline{S}$ using the same compute domains, we say that $\overline{S}$ *simulates* S at scale factor $k \in \{1, 2, \ldots\}$ under some reachability function $\xrightarrow{1}$ if it has scaffold length kN and every configuration C of S has a corresponding configuration $\overline{C}$ of $\overline{S}$ that represents it as follows: the tile $t = C[i]$ at position i in S is represented by the tile $\overline{C}[ki]$ at position ki in $\overline{S}$ via some representation function $f_i : \overline{T}_{ki} \cup \{\emptyset\} \to T_i \cup \{\emptyset\}$ (i.e., $f_i(\overline{C}[ki]) = C[i]$) where $f_i(\emptyset) = \emptyset$ (and no other tile maps to $\emptyset$). We require that $\overline{C}$ has a mismatch between $\overline{C}[ki]$ and $\overline{C}[ki+1]$ if and only if C has a mismatch between $C[i]$ and $C[i+1]$, with no other mismatches at any other positions in $\overline{C}$ except for those between two adjacent $\emptyset$s. Further, we require that $C \xrightarrow{1} C'$ implies $\overline{C} \xrightarrow{k} \overline{C'}$.

Proof of Theorem 25. Each scaffold position $i \geq 1$ in S is simulated by a block of k scaffold positions $k(i-1)+1, \ldots, ki$ in $\overline{S}$, where $k = |D_R|$. We call this set of indices "block i of $\overline{S}$". Fix some cyclic permutation σ on D_R. There are two types of tilesets in each block of $\overline{S}$. The tileset at the last index ki of block i is given by $\overline{T}_{ki} = \{(\sigma^{-1}(l^*)^*, ki, r) \mid (l, i, r) \in T_i\}$. Each other tileset at *any* other position j in block i (i.e., with $k(i-1) < j < ki$) is given by $\overline{T}_j = \{(r^*, j, \sigma(r)) \mid r \in D_R\}$. The representation f_i, for all i, is bijective and maps the tiles based on the tileset construction – that is, $f_i((\sigma^{-1}(l), ki, r)) = (l, i, r)$. Given a full configuration C of S, the representative configuration $\overline{C}$ of $\overline{S}$ is the unique configuration of $\overline{C}$ such that $f_i(\overline{C}[ki]) = C[i]$ for each i, and no mismatches occur anywhere except possibly between tiles ki and $ki+1$ for any value(s) of i. This exists uniquely, as after fixing each tile at each index ki, there is always a unique choice of tile to place to the immediate left of each position ki because σ is a permutation. This gives a unique way from each position ki to place tiles leftward until reaching position $k(i-1)$. If C is a partial configuration (containing one or more $\emptyset$s), $\overline{C}$ is constructed in the same way, except the tiles within the block whose rightmost tile maps to $\emptyset$ are *all* $\emptyset$.

In any configuration $\overline{C}$ representing a full configuration C, we observe that, starting from the left of each block and moving rightward, each tile has a distinct left domain, with each domain appearing exactly once. The leftmost tile t_j in any block is given by $(r^*, j, \sigma(r))$ for some $r \in D_R$. As there are no mismatches within any block, the next tile must be $(\sigma(r)^*, j+1, \sigma(\sigma(r)^*)^*) = (\sigma(r)^*, j+1, \sigma^2(r))$. By straightforward induction, the $(k-1)$st tile has right glue $\sigma^{k-1}(r) = \sigma^{-1}(r)$, as σ is a permutation.

Any full configuration of $\overline{S}$ looks like this, so it follows that the sum of the left compute domain energies within each block are the same across all blocks. Therefore, all target configurations of $\overline{S}$ have almost the same free energy, differing only by which compute domain is the unbound leftmost compute domain of the leftmost block. With ϵ as in Definition 24, this implies that these configurations are all within ϵ configuration energy of each other, satisfying Definition 24 for $m = 0$ mismatches. For $m > 0$, each configuration contains at most m blocks that have at least one mismatch; all other blocks contribute the same amount of energy in each. These configurations therefore differ by at most km compute domain bonds, and therefore by total energy at most $km\epsilon$, satisfying Definition 24.

To show that Definition 30 is satisfied, all requirements are true by construction of $\overline{C}$; reachability is straightforward, as $C \xrightarrow{1} C'$ implies that $\overline{C}$ and $\overline{C'}$ differ by exactly k tiles, which are all empty in one and all full in the other, implying that k steps is sufficient to get from one to the other. ◀

We remark that this construction and argument work equally well (though more tediously, and with more complex definitions) by having each scaffold position i be represented by a block of size $|\{r|(l,i,r) \in T_i\}|$, that is, by using a cyclic permutation that cycles through only the right domains of tiles in T_i. This may lead to a much smaller scale factor if compute domains are not frequently reused.

A.6 Proofs for Section 6.2: Lemmas 27 and 28 and Theorem 29

Proof of Lemma 27. Given exponential random variables with rates λ_l and λ_r (describing total tile replacement rates), the probability that a given value sampled from the latter is smaller than a given value sampled from the former is well known to be given by $\frac{\lambda_r}{\lambda_l+\lambda_r}$. The total replacement rate for each tile has one contribution per tile that can replace it (given by Definition 26), and each contribution is divided by the total number of tiles. Therefore, we can substitute the replacement rates from Definition 26 for λ_r and λ_l, and after cancelling constants and removing the irrelevant contribution from the mismatch between t_i and t_{i+1}, it follows that the probability we seek is given by $e^{\Delta G_r} / \left(e^{\Delta G_l} + e^{\Delta G_r}\right) = 1/\left(1 + e^{\Delta G_l - \Delta G_r}\right)$. ◀

Proof of Lemma 28. Let $E(i)$ be the expected number of steps when the mismatch is at position i. We seek the value of $s = E(0)$. At each position, the mismatch moves one step and then occupies (and has the expected value of) a new position. Therefore, $E(i) = 1 + p \cdot E(i+1) + (1-p) \cdot E(i-1)$ for all $0 < i < n$ (where $E(n) = 0$, as the process is finished then).

Let $a_i = s - E(i)$. Then $a_0 = 0$, and $a_1 = a_0 + 1 = 1$ because, from the leftmost position, the mismatch can only move right. Further, as $E(n) = 0$, $a_n = s$, so we must find a_n. Rewriting and re-indexing the above equation in terms of a_i and rearranging it gives a recurrence relation for a_i: $a_i = p^{-1} + p^{-1}a_{i-1} + \left(1 - p^{-1}\right) a_{i-2}$. We solve this recurrence relation in a standard way: we first homogenize it and find that its characteristic polynomial is $f(\lambda) = \lambda^3 + \left(-1 - p^{-1}\right)\lambda^2 + \left(2p^{-1} - 1\right)\lambda + \left(1 - p^{-1}\right)$, with roots 1, 1, and $p^{-1} - 1$.

When $p = \frac{1}{2}$, the multiplicity 3 root at 1 implies that any solution has the form $a_n = c_0 + c_1 n + c_2 n^2$, and plugging in initial conditions $a_0 = 0, a_1 = 1$ and $a_2 = \frac{2}{p}$ gives $a_n = n^2$. When $p \neq \frac{1}{2}$, the roots of $f(\lambda)$ imply solutions have the form $a_n = c_0 + c_1 n + c_2 \left(p^{-1} - 1\right)^n$. Plugging in initial conditions again yields the given solution. ◀

Proof of Theorem 29. As we wish for an upper bound, by Lemma 15, we need only consider the configuration with one mismatch in the leftmost position. Because compute domains are isoenergetic and scaffold domains get linearly stronger, Lemma 27 says that the probability of a mismatch stepping forward will be the same at each scaffold position, and is given by $p = \frac{1}{1+e^b}$. Therefore, since $b > 0$, $p \neq 0.5$ and we can plug it in to the expression in Lemma 28 to obtain an upper bound on the expected number of steps (which is exact in the case of one mismatch in the leftmost position). This is sufficient for an asymptotic bound on the expected time as well: when $b \in O(\frac{1}{N})$, the strongest domain has constant strength (so each step takes constant expected time), and when $b \in O(\frac{\ln N}{N})$ it has strength $O(\ln N)$ (so each step takes at most polynomial expected time). Note in this expression that $n = N - 1$, as a scaffold of size N has this many potential mismatch positions. After plugging in p and simplifying, this expression becomes

$$\frac{1+e^b}{1-e^b}(N-1) + \frac{2e^b}{(1-e^b)^2} \cdot \left(e^{b(N-1)} - 1\right).$$

As we're looking for an upper bound, we ignore the first (negative) term, and we may replace the denominator of the second term with b^2 because $e^b - 1 > b$ for $b > 0$. We may also remove the final -1. This simplifies the upper bound further to $\frac{2e^{bN}}{b^2}$. When $b = c/N$ for some constant c, this becomes $\frac{2e^c}{c^2}N^2 \in O(N^2)$. When $b = \frac{c \ln N}{N}$, it becomes $\frac{2N^{c+2}}{\ln^2 N} \in O(N^{c+2})$. ◀