Engineering Molecular Self-assembly and Reconfiguration in DNA Nanostructures

Thesis by

Philip Petersen

In Partial Fulfillment of the Requirements for

the degree of

Doctor of Philosophy, Biology

CALIFORNIA INSTITUTE OF TECHNOLOGY

Pasadena, California

2018

Defended on May 22, 2018

Philip F. Petersen
ORCID: 0000-0002-9912-389X

ABSTRACT

Smart electronics have developed ubiquitously to assist people in everything from navigation to health monitoring. The rise of complex electronics relied on rational design of platforms to build ever larger and more complex circuit networks and for frameworks to test those electronics. Biochemical circuits have also seen dramatic advancement in the last two decades within the field of DNA nanotechnology. As with electronics, DNA nanotechnology applied rational design to DNA molecules to build ever more complex biochemical networks that, beyond current electronics, also retain a significant measure of biological compatibility and plasticity akin to many networks of biological origin. Well situated for promising applications in diagnostics and therapeutics, advancing DNA nanotechnology devices will also rely upon larger platforms and testing frameworks.

In roughly the last decade, researchers have been building upon the invention of DNA origami, a technique allowing the robust construction of biomolecular nano-structures capable of precise nanometer positioning of proteins, nanoparticles, and other molecules. DNA circuits have computed on the nanostructures; DNA robots have moved nanoparticles, made choices, and have even sorted cargo on the surface of a nanostructure. The complexity of circuits and devices continues to rise.

In this thesis, we will discuss our contributions to the field of DNA nanotechnology by developing design rules and systematic approaches to controlling nanostructure complex assembly. These rules and approaches allow for the construction of molecular structures with a tunable diversity, large systems approaching the size of bacteria yet retaining nanometer precision, and biological plasticity inspired dynamic systems for arbitrary reconfiguration.

Using a DNA origami tile tailored for array formation with a high continuous surface area, we create a framework inspired from molecular stochasticity for programming DNA array formation and gaining control over diversity of global properties through simple local rules. Three general forms of planar networks, random loops, mazes, and trees, were manipulated

on the micron scale upon the self-assembled DNA arrays. We demonstrate control of several properties of the networks, such as branching rules, growth directions, the proximity between adjacent networks, and size distributions. The large diversity, in principle, allows for a wide, but tunable, testing environment for molecular circuits. By further applying these principles to subunits of finite assemblies, variable components may be mixed with fixed components potentially opening additional applications in high throughput device or drug screening.

Next we turned to expanding the platform size biochemical circuits may be built upon. While DNA origami allows nanometer precise placement, the size remains roughly below 0.05 um^2. Toward making large arbitrarily complex structures with only a set of simple tiles, multi-stage self-assembly has been explored in theory and for small DNA tiles. None were successful experimentally with DNA origami. We developed a strategy for DNA origami: a simple rule set applied recursively in each stage of a hierarchical self-assembly process, and to significantly reduce costs, a constant set of unique DNA strands regardless of size. We also developed a software tool to automatically compile a designed surface pattern into experimental protocols. We experimentally demonstrated DNA origami arrays approaching the size of small bacteria, 0.5 um^2, with several arbitrary patterns, each consisting of 8,704 specifically chosen pixel locations with nanometer precision, including a bacteria sized portrait of a bacteria. The large platform opens the door to more advanced molecular circuits for applications such as diagnostics.

Finally we demonstrated control over the dynamics of DNA origami reconfiguration in tile arrays. In an approach we call DNA tile displacement, we showed that a DNA origami array may have tiles arbitrarily replaced by another tile, including tiles of another shape or surface pattern. We also demonstrated control over the kinetics of tile displacement and performed several general purpose reconfigurations of DNA nanostructures. Examples include sequential reconfiguration, competitive reconfiguration, cooperative reconfiguration, and finally the scalability of multi-step reconfiguration as demonstrated through a fully playable nano-scale biomolecular tic-tac-toe game. The major ramifications

are a plasticity more common to biology than to electronics—molecular platforms with arbitrary patterning that can reconfigure an arbitrary part of the nanostructure in an arbitrary order based on environmental signals. In principle, such reconfiguration can allow advanced circuits with the capacity to adapt to environmental needs or heal damaged components.

PUBLISHED CONTENT AND CONTRIBUTIONS

Philip Petersen*, Grigory Tikhomirov*, and Lulu Qian. "Tile displacement for systematic reconfiguration of DNA origami arrays." In: *Submitted* (2018).

*Equal contribution

P.P., G.T., and L.Q. initiated the project; P.P. and G.T. designed and performed the experiments, and analyzed the data; P.P. developed the software tool; P.P., G.T., and L.Q. wrote the manuscript; L.Q. guided the project.

Grigory Tikhomirov*, Philip Petersen*, and Lulu Qian. "Fractal assembly of micrometre-scale DNA origami arrays with arbitrary patterns". *Nature*, 10.1038/nature24655, 2017.

*Equal contribution

P.P. and G.T. initiated the project, designed and performed the experiments, and analyzed the data; P.P. developed the software tool; G.T., P.P., and L.Q. wrote the manuscript; L.Q. guided the project.

Grigory Tikhomirov*, Philip Petersen*, and Lulu Qian. " Programmable disorder in random DNA tilings". *Nature Nanotechnology*, 10.1038/nnano.2016.256, 2016.

*Equal contribution

P.P. and G.T. performed the experiments and analyzed the data. P.P. performed the simulations and developed the software tools. G.T., P.P., and L.Q. designed the systems and wrote the manuscript. L.Q. initiated and guided the project.

Philip Petersen. "Programming random mazes". *Nature Nanotechnology* 12(3):284, 10.1038/nnano.2017.30, 2017.

P.P. wrote the "In The Classroom" article.

TABLE OF CONTENTS

Abstract ……..iii
Published Content and Contributions……..... vi
Table of Contents……. vii
Chapter I: Introduction 1
1.1 Overview of DNA nanotechnology 2
1.2 DNA origami as a breadboard 5
1.3 Reconfiguration in DNA nanostructures 8
1.4 Summary of the thesis 10
Chapter II: A DNA origami tile 12
2.1 Design considerations for a scaffold path 13
2.2 Calculating single-stranded domains and creating stable seams 16
2.3 Creating inert and interacting edges 21
2.4 Creating a unique pattern 26
Chapter III: Controlling growth and global properties of DNA origami arrays 30
3.1 Growth of unbounded 2D DNA origami arrays 31
3.2 Global properties controlled by programming pattern design on individual tiles 41
3.3 Global properties controlled by programming the tile-tile interactions 48
3.4 Global properties controlled by programming the probabilities of tile choices 52
3.5 Design of 2D DNA Origami arrays with designed sizes 55
3.6 Combinatorial diversity and programmable properties in finite arrays 61
Chapter IV: Fractal assembly of micron-scale DNA origami arrays with arbitrary patterns 64
4.1 Design considerations and rules of Fractal Assembly 66
4.2 Demonstration of Fractal Assembly with DNA Origami tiles 76
4.3 Yield of arrays in relation to number of tiles 83
4.4 Software tool for automated design and experiments 89
Chapter V: Reconfiguration of DNA Nanostructures Using Tile Displacement 92
5.1 A general approach to reconfiguration 94
5.2 Competitive Reconfiguration 99
5.3 Sequential Reconfiguration 104
5.4 Cooperative reconfiguration 107
5.5 Scalability of multi-step reconfiguration 110
Chapter VI: Conclusions 117
Appendix A: Additional diagrams and images 120

Appendix B: DNA Sequences 124
Bibliography 166

Chapter 1

INTRODUCTION

As you read this sentence, a vast network of cells in your head are working together to process the incoming information. However, the stunning complexity of cells contemplating their own complexity is no less rivaled by the impressive diversity of tasks cells perform elsewhere in a human body. Cells cooperate to build the bone scaffolding supporting posture, the push and pull of separate muscles in sync to move a limb, and even to hunt down foreign invader cells in an immune defense. Each individual cell may have vastly different morphological shapes and surface expression of biomolecular components. The list goes on and on. Scientists have worked to understand the complexities of a single cell, asking questions such as how it responds to external environmental cues, how it adapts, how it communicates, and how it replicates. An early step in the field of genetics starts with Gregor Mendel [1] in the mid-1800s raising pea plants and noting the heredity of traits. The hereditary information controlling the vast diversity of traits, as we now know, is from deoxyribonucleic acid (DNA), the central information of life.

It would take until 1953 for the core model of DNA to be created [2] based on available x-ray pattern data of DNA. DNA was a double helix, formed of the chemical bases Thymine (T), Adenine (A), Guanine (G), and Cytosine (C). While DNA can take other conformations, this model well represents most DNA in biological cells [3]. The base A binds to T and the base G binds to C. The specificity of the bindings is due to the A-T connection forming two hydrogen bonds and the G-C connection forming three hydrogen bonds. The stability of the helix is derived from the stacking interactions between the base pairs. The double helix is approximately 2 nm wide and each base pair is 0.34 nm long. There are roughly 10.5 base pairs per turn of the DNA helix. Hence each base pair rotates approximately 360/10.5 degrees with each additional base pair along the length of the double helix. The great structural consistency DNA possesses over other nanomaterials

would lend itself well to nanostructure design forming the field of structural DNA nanotechnology.

Many industries have formed around understanding and controlling DNA. Some seek to develop food crops with new resistances or higher yield, while others analyze a person's genome to find genetic markers of disease. In the last two decades, the emergent field of DNA nanotechnology has taken advantage of DNA molecules to form structural systems [4] and process complex computational functions [5]. Supporting the rise of all these areas of research, there have been technological advancements in DNA synthesis, manipulation, and visualization. The cost of DNA synthesis has decreased several orders of magnitude since the field of DNA nanotechnology started, even allowing for entire gene synthesis [6]. Also, DNA has proved to be a highly compatible material, having been conjugated to a wide range of molecules such as proteins [7], nanoparticles [8,9], or polymers [10]. In terms of visualization, technologies in super resolution microscopy such as DNA PAINT [11] can visualize short individual binding events of DNA strands. Moreover, resolution down to the major and minor groove of DNA has been obtained with atomic force microscopy (AFM) [12]. High-speed AFM may even allow visualization of dynamic molecular events [13]. The supporting technology invites the development of ever more complex DNA structures and devices.

In this thesis, we will discuss our contributions to the field of DNA nanotechnology by programming design rules and demonstrating systematic approaches to controlling nanostructure complex assembly. These rules and approaches allow for the construction of molecular structures with a tunable diversity, large systems approaching the size of bacteria yet retaining nanometer precision, and biological plasticity inspired dynamic systems for arbitrary reconfiguration.

1.1 Overview of DNA nanotechnology

DNA nanotechnology mostly takes advantage of DNA’s information carrying capacity and structure, using DNA for the purposes of nanotechnology rather than as genetic encoding in

cells. This section gives a rough overview of select advancements in the field focusing mostly on structural DNA nanotechnology.

As noted previously, DNA's base pairing rules control how two strands of DNA will bind together to form a well characterized double helix. Beyond base pairing, there has been much investigation into the structural properties [14] and thermodynamic properties [15] of DNA. With artificial DNA synthesis, there is room for rational design of base sequences allowing for the formation of specific target DNA complexes. For example, work by Len Adleman [16] demonstrated rationally designed DNA for computation. By harnessing combinatorial connections of designed DNA molecules, it became possible to compute the solution to a seven-node Hamiltonian path problem. Unique single strands of DNA representing nodes and paths between nodes were mixed together in massive parallelism. Double strand connections between the strands formed combinatorial pathway solutions. While this work was impractical to scale up, it demonstrated the application of combinatorial diversity to DNA strand connections for molecular information processing, and inspired our own exploration of combinatorial diversity in surface patterning (see Chapter 2).

When it comes to forming structures from DNA, Ned Seeman pioneered several early successes. While DNA in nature is mostly double strand unbranched helices, through DNA synthesis, non-natural branched structures could be formed; see Figure 1.1(a). Through sequence design of every DNA strand in a system, even 3-dimensional structures may be formed such as a cube [4]. Others looked at scaling up DNA systems through the formation of arrays from small DNA tiles. Winfree et al. formed small tiles from DNA with short single-strand DNA 'sticky ends' programmed to allow the tiles to come together in large periodic arrays [17], see Figure 1.1(b). Early repeating periodic arrays would lead to more complex algorithmic assembly (see Figure 1.1(c)) opening up the formation of complex computing arrays implementing general frameworks such as cellular automata [18] or specific functions such as binary counting [19].

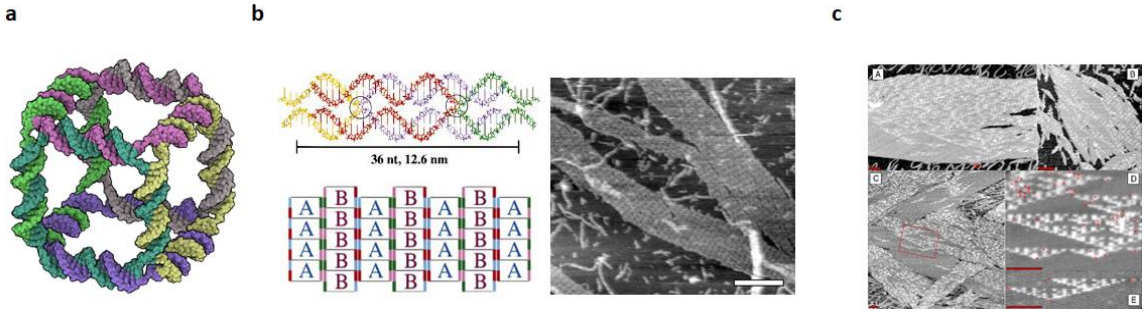


Figure 1.1: **DNA nanostructures**. **a**, DNA designed to form a cube shaped nanostructure [4]. **b**, DNA designed to form a small DNA tile with short single-strand sticky ends. The sticky ends interact to allow periodic array formation [17]. **c**, small DNA tiles with complex sticky ends designed for algorithmic assembly of an array forming Sierpinski gasket fractals [18].

A major step forward in DNA structures took place in 2006. Paul Rothemund invented a new technology which he called DNA origami as it involved folding a long single-stranded DNA into arbitrary shapes [20]. The long single strand was a single-stranded DNA genome of a virus measuring over 7 kilobases long, and, in the context of DNA origami, it is known as scaffold DNA. After routing the scaffold DNA into the desired folding pathway of the target shape, hundreds of shorter synthetic single-strand DNA segments are introduced designed to hold the scaffold DNA in the target conformation, see Figure 1.2(a). The technique of making DNA origami has been so successful and accessible that several groups have formed their own customized structures from scratch [21, 22]. Thinking outside the box, the researchers in Andersen et al. formed their DNA origami into the shape of a box capable of being opened with DNA strand 'keys' [23]. Other researchers with biomedical applications in mind used DNA origami to create a DNA device that can open and close in response to target antigens on a cell surface [24]; see Figure 1.2(b). This device was used to deliver antibody fragments to a natural killer leukemia cell line isolated from a patient with aggressive NK leukemia.

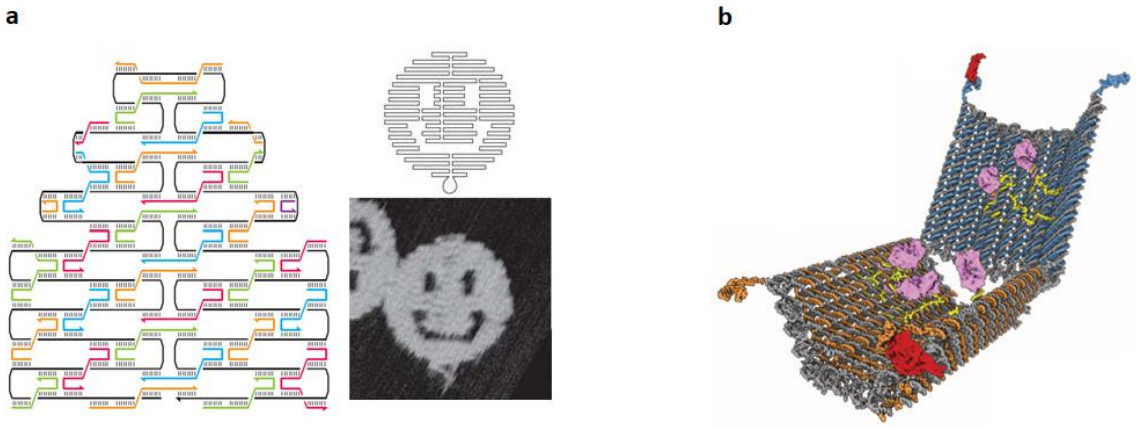


Figure 1.2. **DNA origami nanostructures**. **a**, Iconic Smiley DNA origami from Paul Rothemund's publication of DNA origami. On the left is an example routing path of the long single-strand scaffold strand in black with several colorful short DNA strands providing support [20]. **b**, A DNA origami designed to deliver antibody fragments to a leukemia cell line. The device can open and close in response to antigens on a cell surface [24].

1.2 DNA origami as a breadboard

The structural precision of DNA origami coupled with DNA's compatibility with a wide range of other molecules has been the basis of several molecular devices. This section explores some of the small devices made on DNA origami, and the challenges of scaling up the size of 2D DNA nanostructures to allow greater device complexity.

Similar to how a breadboard is useful in electronics for organizing electronic components on a flat surface, DNA origami can readily provide a flat 2D surface with nanometer precision. Researchers have made good use of the limited space on a single DNA origami to organize all sorts of molecular materials such as proteins [7], nanoparticles[8, 9], or polymers[10]. From those components, a wide range of molecular devices can be formed on DNA origami. In an example of bottom-up fabrication, rather than the top-down approach of electronics, Maune et al. [25] formed a molecular electronic device by arranging two carbon nanotubes on DNA origami to form a transistor. While only a single transistor per DNA origami was fabricated, many transistors were formed in parallel, potentially allowing for construction of complex molecular electronics in a massively parallel fashion should a large enough platform be available.

Other researchers have explored the area of autonomous molecular machines. Fine nanomechanical systems were explored by Hariadi et al. through their investigation of protein-protein interactions [26]. With the capacity to tether muscle's myosin proteins on DNA origami at specific locations, the mechanics of individual protein contributions to collective motion could be investigated. An enzymatic cascade was made by Ke et al., exploiting DNA origami's spatial organization for directional regulation of an enzyme pathway [27]. Researchers have also developed and programmed molecular robots. Molecular robots have developed from just taking several steps on an origami surface to moving materials such as nanoparticles [28]. Robots may also be programmed to perform complex tasks such as sorting cargo species to designated locations [29].

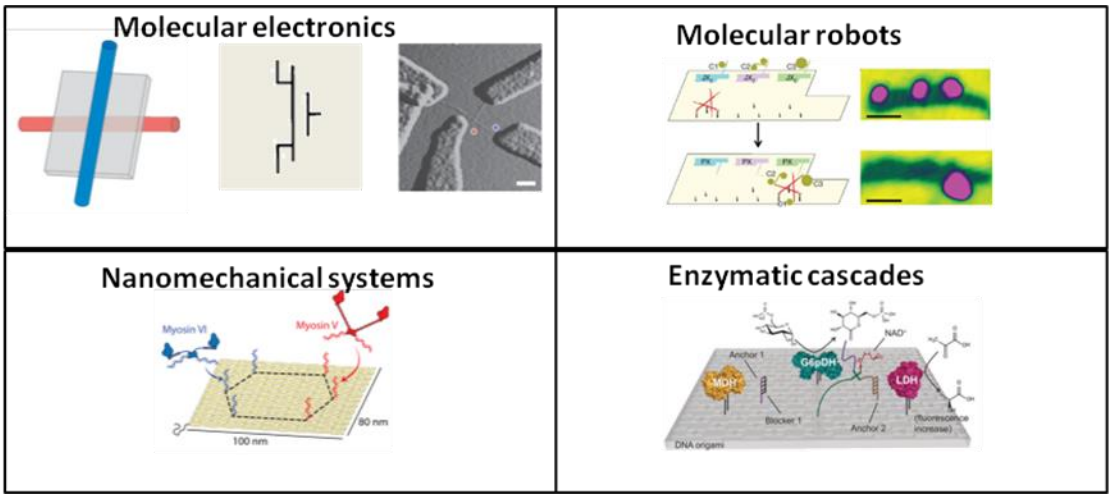


Figure 1.3. **Molecular devices built on DNA origami**. [25-28].

Naturally, devices could be larger and more complex if there was a larger platform. As with small DNA structures, there is interest in scaling up DNA origami. Some researchers took the direct approach. If DNA origami was limited by the length of the scaffold DNA strand, then finding a longer single-stranded DNA strand to act as a new scaffold would make larger DNA origamis. Marchi et al. successfully applied this method to fold DNA origamis several fold larger than the first origamis [30]; however, there has been trouble scaling further with this approach, and the cost of synthetic DNA strands, now numbering in the

thousands to fully connect the larger structure, starts becoming inhibiting. Accordingly, the approach of forming arrays of DNA origami tiles was also explored. Seeman et al. [21] created a DNA origami designed to form arrays, see Figure 1.4. Under specific temperature conditions, it was demonstrated this origami could form periodic arrays of several microns; although, there some drawbacks existed. To maximize locations functional molecules can be attached at, it is favorable to design the origami to have all its staple extension points on the same side of the origami; however, this design required two tiles flipped relative to each other which would put components on opposite sides of the array. A breadboard works best when all components are on the same side of the board able to interact with one another. Also, the tiles used for the large array where a tight plus shape, see Figure 1.4, and thus had limited continuous surface area coverage due to the open corners even in a perfectly formed array. As of this writing, there has been no algorithmic techniques to DNA origami assembly.

While a large periodic array does provide a large surface to build on, each tile in the array loses its unique addressability. Other researcher groups have tried various methods to form finite arrays of tiles where every tile may be uniquely addressed. Woo et al. first made finite arrays with stacking bonds and shape complementary of edges [31]. Rajendran et al. then made nine unique origami tiles in the form of jigsaw pieces [32], see Figure 1.4. Each DNA origami tile piece fit at a particular spot in a three by three finite array. Zhao et al. instead tried organizing separate DNA origami tiles with a scaffold strand [33]. These approaches, while not much different in size compared to the large DNA origami by Marchi et al., did have the additional benefit that a portion of DNA strands were reused. Without as many unique strands, upfront costs for synthetic DNA manufacturing, while still scaling with increased platform size, scaled less extensively as simply forming a single massive DNA origami. Still, the size of finite 2D DNA origami nanostructures was limited. It is worth noting that if the size of a finite array is made large enough, it may be able to integrate with low-cost top-down approaches such as photolithography for scalable device construction.

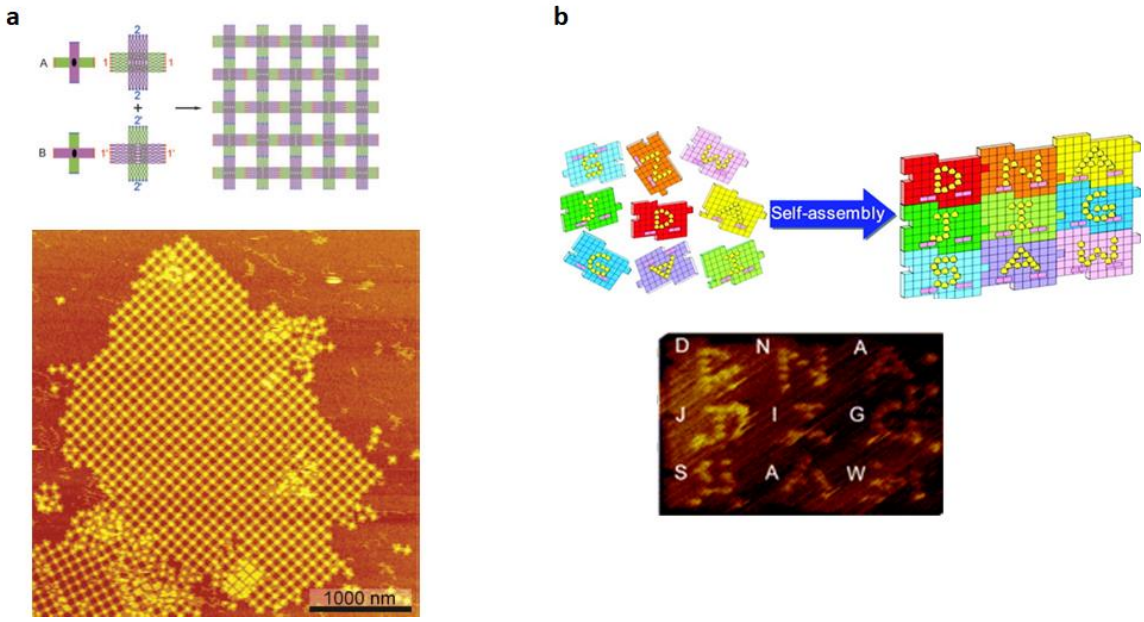


Figure 1.4. **Scaling up DNA origami. a**, periodic array of DNA origami tiles. **b**, three by three array of DNA origami tiles.

1.3 Reconfiguration in DNA nanostructures

Reconfiguration of an existing structure allows for efficient use of resources and also adaptation to environmental stimuli. This section explores the extent of research into DNA nanostructure reconfiguration.

Cells, the quintessential natural molecular machines, undergo significant structural reconfiguration. Immune cells will undergo shape reconfiguration as they chase an invader, pushing and pulling on their membranes [34]. Most other cells will at least undergo pattern reconfiguration by switching their surface receptors as the cells mature or face an environmental stress [35]. Not only are these adaptations efficient for life, this plasticity allows for responsive changes to external stimuli. Still, structural reconfiguration has yet to be thoroughly explored in artificial molecular machines. There have been some dynamic DNA structures. Han et al. [36] formed a DNA Mobius strip able to take a thick and thin conformation. Gerling et al. [37] made a DNA robot-like figure with two hinged arms able to take an open or closed position. Song et al. [38] made a "domino" nanoarray able to pass

an input deformation throughout an array. Others have made openable boxes [23] or drug delivery vehicles [24] that open with the right target. Ultimately, all these systems perform a specific task and lack a generalizable module for arbitrary reconfiguration.

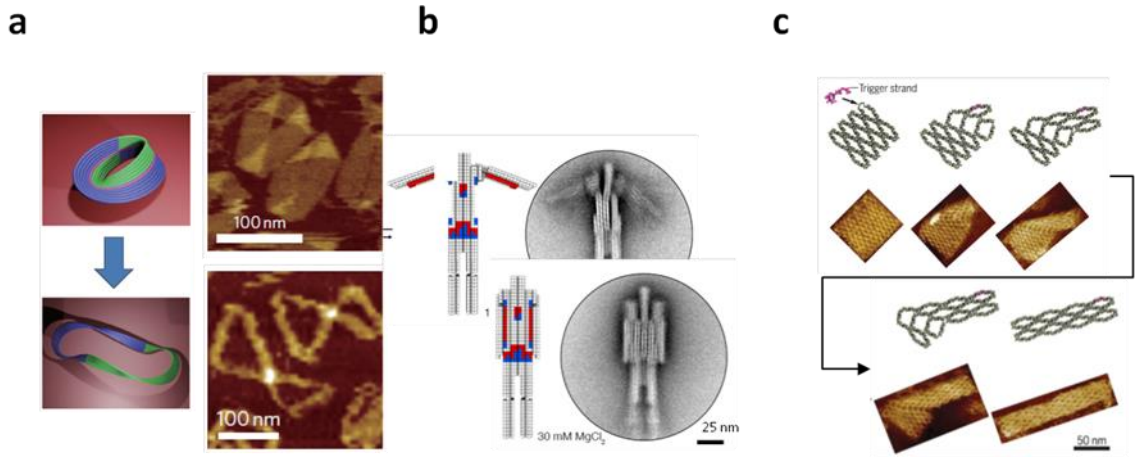


Figure 1.5. **Dynamic DNA structures. a**, a DNA Mobius strip [36]. **b**, a DNA robot-like figure [37]. **c**, a "domino" nanoarray [38].

Stepping away from DNA origami and tile arrays, an important driving force in dynamic DNA nanotechnology has been the capacity to build computational systems. Many of these circuits rely on the ability to tune reaction rates as variable rates allow functions such as analog to digital conversion or thresholds [39]. In essence, small DNA strands can be controlled to interact with one another at rates tunable over several orders of magnitude [40]. Similarly, in programming the pathways of structural reconfiguration, the ability to tune reaction rates would allow for more sophisticated functions such as those that exploit competition. Cells take advantage of competing membrane modifying pathways for their own structural reconfiguration such as when guiding the growth cone of a developing neuron's axon [41]. However, in artificial nanostructures, there has been no such control developed for DNA origami interactions.

1.4 Summary of the thesis

This section presents our main contributions to the field of DNA nanotechnology though unique solutions to the above challenges.

Chapter 2 of this thesis introduces the development of a new DNA origami tile designed specifically to work as an optimized 2D breadboard for larger complex nano-devices and machines. We designed an entirely flat, one-helix thick tile tailored to array formation with a single, non-flipping tile. With interest in the tile forming a breadboard for future devices, we optimized the tile for a high continuous surface area to allow the greatest flexibility in future device designs.

Chapter 3 of this thesis shows the implementation of combinatorial approaches to 2D systems of molecules to allow controlled diversity in network formation. Combinatorial approaches in synthesizing one-dimensional polymer chains have revolutionized chemical synthesis and the selection of functional nucleic acids. We expand these principles to random two-dimensional networks to open new opportunities for fabricating more complex molecular devices on DNA nanostructures. In order to scale up the complexity and diversity of molecular structures and devices, one can control the inherent stochasticity within molecular systems. First, we developed a framework for programming DNA tilings through simple, local rules to obtain control of global patterns. Applying the rules experimentally, we constructed several forms of planar networks on micron-scale self-assembled DNA origami arrays and controlled properties such as growth, network proximity, and size distributions. Furthermore, we demonstrated controlled stochastic self-assembly allowing for combinatorial diversity in subsets of an entire system, potentially useful for applications such as parallel drug screening or parallel testing of molecular robots against diverse operating environments.

Chapter 4 of this thesis demonstrates our robust hierarchical strategy to break the two-dimensional size limitations of DNA origami, and its application to forming micron-scale uniquely-addressable DNA origami arrays. The arrays formed reach the scale of a small

bacteria yet maintain unique addressability. Furthermore, the upfront cost of manufacturing the arrays is fixed regardless of scale, as the strategy employs a constant number of unique structural DNA strands. These arrays may be used for complex organization of diverse molecules and device fabricating with nanometer precision of components over the entire micron scale of the array. Opportunities arise for larger and more sophisticated molecular machines such as DNA robots or DNA circuits. Fundamentally, the simple recursively applied hierarchical strategy may be more widely used to build complex molecular systems with a constant number of simple components not limited to a specific design.

Chapter 5 of this thesis presents a general purpose approach to reconfiguration in DNA nanostructures. Reconfiguration provides efficiency and responsiveness to environmental stimuli. In principle, a reconfiguring device can be modified to a new task, adapt to external cues, or even heal damaged components. We present a general approach called tile displacement, allowing the reconfiguration of DNA nanostructures with arbitrary patterns to reconfigure arbitrary parts of the structure in an arbitrary order. Shapes reconfiguration may also be applied. We further demonstrated control over the kinetics of tile displacement and developed several building blocks for general-purpose reconfigurations of DNA nanostructures. Examples include sequential reconfiguration, competitive reconfiguration, and cooperative reconfiguration. Finally, we explored the scalability of multi-step reconfiguration as demonstrated through a fully playable nano-scale biomolecular tic-tac-toe game, demonstrating a piece by piece arbitrary interchange of a nanostructure's subunits even to the point none of the original species on the nanostructure game board remained. The plasticity may be more common to biology than to engineering, yet the strategy for autonomous reconfiguration in self-assembled DNA nanostructures is now allowing adaptive behaviors in artificial molecular machines.

Chapter 2

A DNA ORIGAMI TILE

Since its conception, the technique of forming DNA origami [1] has been applied to form a myriad set of nano-scale structures. Here, we design a new DNA origami tile tailored to forming arrays with extensive continuous surface area for displaying patterns.

Ever since the first unnatural DNA nanostructures [2], there has been interest in forming arrays. Using small DNA tiles composed of up to just a few DNA strands, researchers have formed finite arrays [3], periodic arrays [4], and even created complex global patterns through the use of algorithmic arrays [5-7]. However, the development of DNA origami has brought an advantageous new tool with its own benefits. An origami tile can be composed of hundreds of unique strands leading to a considerably larger size. Researchers have made good use of the space to organize a wide range of molecules such as proteins [8], nanoparticles [9, 10], or polymers[11]. This high spatial precision of molecular placement holds promise for molecular devices; however, to form and manipulate large complex molecular networks, larger sizes and greater control of tile interactions is necessary.

As a larger structure, the edges of origami tiles can also be larger, allowing for more programmable interactions through multiple sticky ends and geometry. These new parameters creates new challenges and benefits for array formation. Woo et al.[12] explored several origami tile edge design parameters and their effect on tile interactions. They used blunt end helix stacking along with relaxed edges to bring tiles together. However, one of the most successful 2D array designs by Liu et al.[13] took into account several new design considerations. Their origami had two bundles of perpendicular helices forming a plus like shape. While making the origami two helices thick around the tile center, this design created symmetry at its edges with all edge helices parallel with the edge

helices of neighboring tiles. Also, to remove the natural curvature within their tile, a corrugation strategy was implemented where tiles were flipped alternatively in the array.

While the prior origami tile designs were able to form various arrays, it is desirable to have a flat, one-helix thick tile capable of forming arrays without flipped tiles and designed for a high continuous surface area. Such a large, flat surface would function optimally as a breadboard for future devices. We designed such a DNA origami tile that would serve as our platform for engineering array formation and tile-tile dynamics.

2.1 Design considerations for a scaffold path

DNA origami can be formed into a wide range of structures depending on the designed routing of the viral M13 single -stranded DNA known as the scaffold path. This section covers the intended properties of the tile including the capacity to rotate in an array location, simplicity of design, direction of helices at the tile's edges, and the amount of continuous surface area. The pros and cons of multiple potential designs meeting those properties are considered.

One core interest for our DNA origami tile is the ability for a single tile to be able to form large arrays including unbounded array growth and finite arrays. On these arrays, we would be able to implement a useful tool to meet a research aim: Truchet tiling [14, 15] (see Chapter III). For Truchet tiling, each Truchet tile in the array has a rotationally asymmetric pattern that connects to the pattern on its neighboring tiles. The pattern connection allows for controllable large complex global patterns with the right parameters, including each tile's individual rotation in the array (see Chapter III for in depth details and implementation). As rotation is important in the implementation of Truchet tiles, it is desirable for each tile to be able to take several rotated conformations within the array plane at each tile location. Many shapes such as a rectangle, with only two possible rotations in an array, would place limits on the maximum possible rotations. Accordingly, the DNA origami tile should be symmetric on all sides when rotated in a location of the array. Avoiding great complexity in tile shape, a few simple space-filling shapes were

considered such as a triangle, square, and hexagon (Figure 2.1). The square has a comfortable four possible symmetric rotations, better than a triangle with only three. While the hexagon has more possible rotations, in a finite array, the square tile allows the formation of smoother outer edges. Furthermore, square tiles in any array form a standard grid pattern with locations that can be represented with integer (x,y) markers for ease of display, use, calculations, and eventual simulation of systems. For all these reasons, for our main DNA origami tile, we chose to form a symmetric square tile.

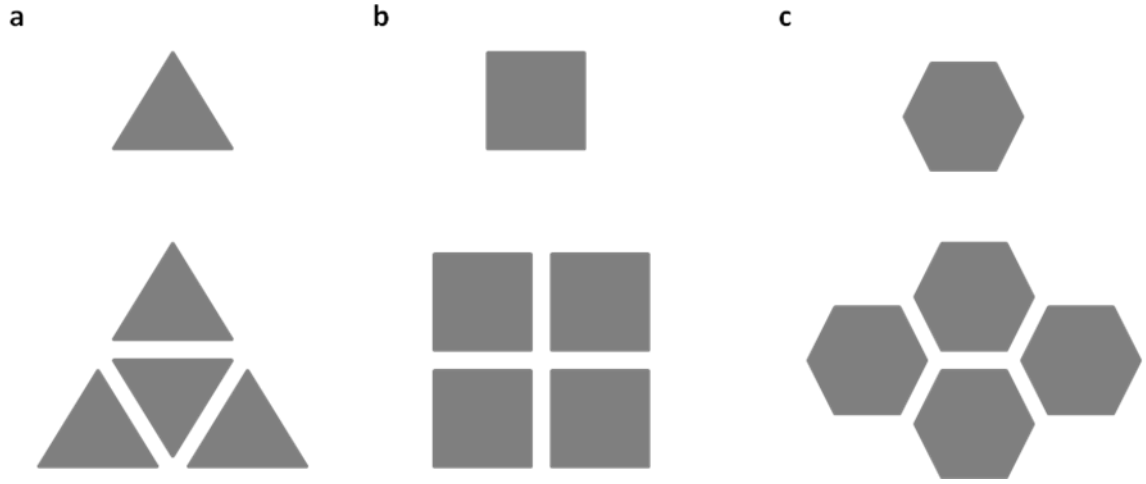


Figure 2.1: **Space-filling regular polygon shapes. a**, A triangle tile and triangles in a mini array. **b**, square tile. c, hexagon tile.

Next, we desire the edges of the square tile to be able to interact with other tiles through the use of weak stacking bonds [12] or short sticky ends (see section 2.3 for edge staple design specifics). Weak interactions between tiles helps reduce kinetics traps during self-assembly, as tiles can disassociate from each other and rearrange themselves into thermodynamically more favorite configurations. A stacking bond requires the blunt ends of DNA double helices to interact; accordingly, DNA helices along an edge of the square tile must be all perpendicular to that edge, see Figure 2.2.

Finally, for the use of the tile as a platform, there must be the ability to freely draw arbitrary patterns upon the surface of the tiles. Pattern pixel locations may be the 5' or 3'

end of staple strands in the DNA origami and are commonly used as attachment sites for proteins, nanoparticles, or other molecules. In order for our tiles to be the potential platform of a wide range of future devices, we desire a flat tile design that maximizes the amount of continuous surface area available on the origami surface, thus avoiding unusable pixel locations. As part of an array forming the global pattern, each tile should be flat and rigid, not only for potential devices built upon the tiles, but also flat structures are useful in encouraging two-dimensional growth for the array formation. With these design criteria in mind, we considered three different scaffold paths; see Figure 2.2:

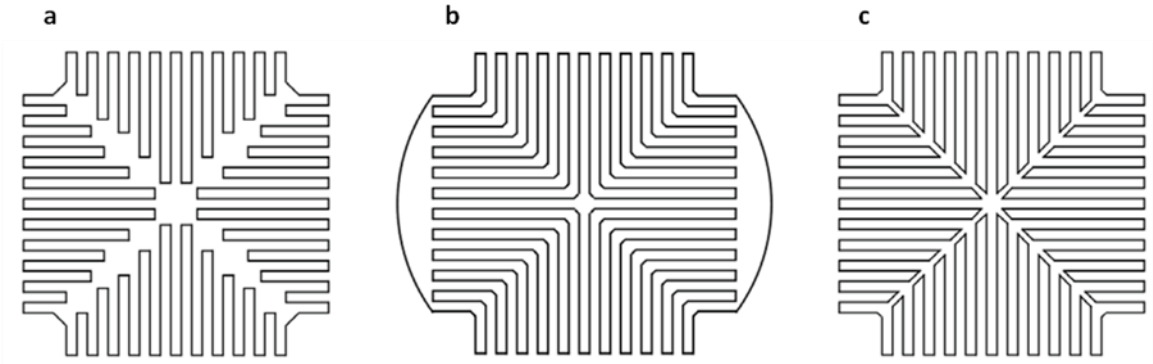


Figure 2.2: **Scaffold paths for a square DNA origami tile. a**, scaffold path fills each of the four triangles within the tile with internal crossovers at the end of each helix row. **b**, scaffold path fills each square corner quadrants within the tile. **c**, scaffold path fills each of the four triangles within the tile with scaffold loops connecting helices within the interior of the tile.

Each of the three designs in Figure 2.2 have their own pros and cons. The first design of the scaffold path considered (Fig. 2.2(a)) routes a path around the square filling each of the four isosceles right triangles within the square sequentially. The tips of each helix row ends with a scaffold crossover. Crossovers are a direct connection between bases on two adjacent DNA double helices. Because DNA is a helix, the common locations where crossovers can geometrically occur from one side of an helix to the other is at integer number of turns plus half a turn of the helix. Accordingly, each internal crossover is an integer plus a half turn away from the exterior edge crossovers. Since the crossover locations are constrained to plus or minus an entire helix turn (10 to 11 base pairs), the interior crossovers tend to leave small holes near the diagonals. If the helices were given an

extra turn to fill those holes, they would then overlap with each other, violating the flat and rigid design criteria. The small holes themselves may cause a lack of rigidity as well, besides also adding to the discontinuous surface area.

The next design (Fig. 2.2(b)) routes a path filling each of the four square shaped corner quadrants sequentially. This design no longer has the small holes in the center as the scaffold does not make any interior crossovers. However, a disadvantage is the need for large scaffold loops connecting the scaffold to adjacent corners of the square. Such large loops may interfere with tile-tile interactions and create a bias. Another disadvantage is the broken symmetry of the edge helices. While the number of helices may be the same, the number of crossover helix pairs varies which could possibly result in unforeseen complications between edges.

Finally, the last and chosen design (Fig. 2.2(c)) routes a path filling each of the four isosceles right triangles within the square sequentially like the first design. The difference in this design is that there are no interior scaffold crossovers. Instead, each interior part of a helix reaches the diagonal and connects to its neighboring helix via a scaffold loop—a calculated number of unpaired nucleotides loosely tethering the two ends together. This design has the advantages of a fully symmetric design with fairly minimal exposed loops of scaffold. It also maximizes the continuous surface area of the tile. Accordingly, we proceed with this design.

2.2 Calculating single-stranded domains and creating stable seams

Single-stranded scaffold loops connect helices along the scaffold path in the interior of the tile. Also, the square tile's diagonals are seams between four isosceles right triangles held together by staple bridges. This section covers the calculations in forming scaffold loops, staple bridges across the seams, and staple design considerations to maintain the square's integrity. See [16 (Supp. S2.2)] for a mathematical workflow.

The single stranded domains of scaffold loops and staple bridges (see Figure 2.3) must be carefully calculated. If the lengths are inaccurate, it could cause budges, strain, or alternatively, a lack of rigidity. Small variations in the strain or twist in a tile can have a deleterious effect on that tile's array formation capacity. In order to properly determine a length we programmed three-dimensional models containing coordinates for every base in every helix within the origami. The length of each base pair in a double helix is set to 0.34 nm and the height of a double helix is set to 2 nm. One base pair is skipped between crossovers in every 48 base pairs to result in an average 10.44 base pairs per helix turn. Adjacent bases thus rotate around the helix at a rate of 360 degrees per 10.44 base pairs. A 150 degree split is assumed from the helix center to the staple and scaffold bases forming the helix. The separation between the four triangles, the width of the seam, was set at 1 nm. It is a distance that is not too close to cause any overlap of base pairs but still keeps the seam tight. 1nm is also the assumed average separation between neighboring helices within the interior of the origami structure. The number of nucleotides needed in a single stranded domain could then be calculated from the Euclidean distances between the start and end bases for the scaffold loops and staple bridges. Each unpaired nucleotide is assumed to be 0.4 nm in a relaxed state. When looking down the length of the seam in the plane of the origami, the staple bridges are staggered. For example, some staple bridges connect from the top side of the origami on one triangle to the bottom side of the origami on the neighboring triangle. The neighboring staple bridge may instead connect from the bottom to the top and so forth along the length of the seam. This staggered set of connections is expected to balance out any strain in the staple bridges.

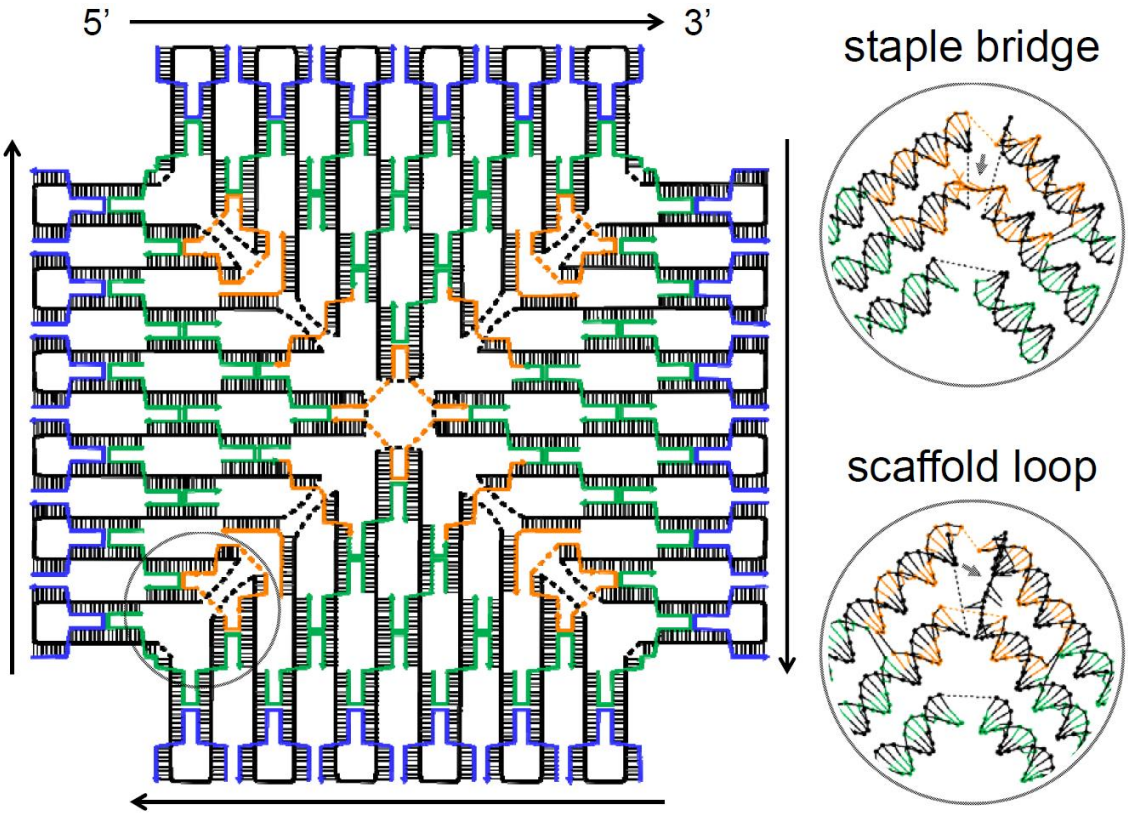


Figure 2.3: **Mini version of the DNA origami with staple bridge and scaffold loop zoom-ins**. Orange represents bridge staples. Black represents scaffold M13. In the mini-origami, the staple bridges connecting neighboring triangles across the seams are dotted orange lines and scaffold loops connecting the interior scaffold between neighboring helices are represented by dotted black lines. Blue represents edge staples—controlling tile-tile interactions (see section 2.3). Green represents internal staples providing pixel locations (see section 2.4).

Applying the model, the number of nucleotides needed in the single stranded domain for each staple bridge from the origami exterior inward was calculated as follows: 7, 5, 4, 7, 6, 4, 7, 3, 5, and 8 nucleotides. Some helices are quite close requiring only 3 nts or around 1.2 nm to bridge the gap, others require more at 8 nts or around 3.2 nm. These numbers are for one seam only, but since the origami has symmetry, the same numbers apply to all seams. The scaffold loops in a single triangle looking down on the origami and going counter-clockwise were calculated as follows: 11, 8, 13, 10, 8, 8, 10, 13, 8, and 11 nucleotides. Each triangle consists of 22 double helices, but note there are only 10 staple bridges in a seam where one would expect 11 connections. This is because the outermost bridge across

the seam is not a staple but rather a scaffold forming a scaffold bridge, see Figure 2.3. The length of the scaffold bridge is calculated at 8 nts; however, the scaffold bridge lengths actually used are 10, 10, 10, and 11 nts for the four seams. The slightly larger values are needed as the full length of the M13 scaffold at 7,249 nts must have a location in the design. Extra nucleotides from the M13 scaffold not used in the main origami structure are stored roughly evenly in the four outer scaffold bridges with the assumption that the staple bridges already provide enough rigidity to compensate for flexible scaffold bridges.

Even with a carefully calculated seam length, the staple bridges have the challenging task of connecting distant portions of the M13 scaffold with a single stranded domain. With so many single stranded domains along the seams, both the staple bridges and the scaffold loops, there is concern that the domains may get tangled or otherwise kinetically trapped during formation harming the flatness of the tile. To investigate the potential issue, we made two designs of staple bridges called "strong-weak" and "strong-strong" as show in Figure 2.4. The strong-weak design, should the bridges have trouble connecting, allow the seams to breath open slightly and reconnect to release any formed tangles. The cost is a weaker seam. The strong-strong design has sufficiently long domains on both adjacent triangles to form a stable connection.

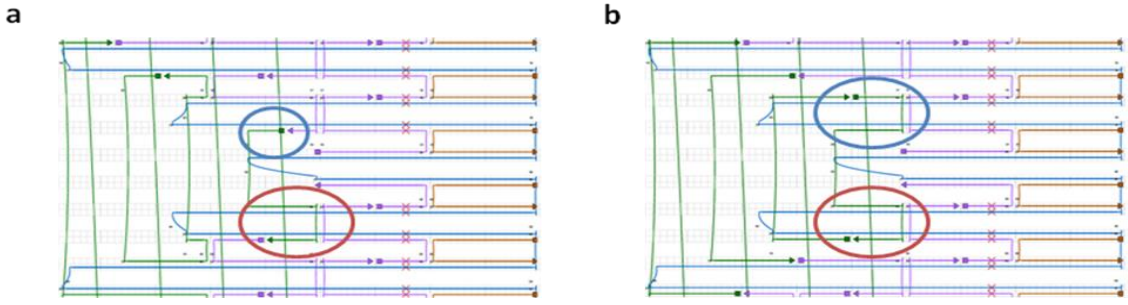


Figure 2.4: **Two Cadnano designs of staple bridges**. **a**, "strong-weak" design. Domains on one side of the seam are much longer than on the other side of the seam. The orange circle shows one longer, and thus stronger, domain on the lower triangle. The blue circle shows that staple bridge's shorter, and thus weaker, domain across the seam on the upper triangle. **b**, "strong-strong" design. Domains are roughly balanced with reasonably long domains on both sides of the seams between adjacent triangles. Note the blue and orange circle cover similar strength domains of the same staple bridge on both sides of the seam.

The structural integrity of the two bridge design was confirmed using atomic force microcopy (AFM). In one set of experiments (Fig. 2.5), the two designs had all their edge staples removed, see Figure 2.3, to provide worse case structural instability. Eventually, to form arrays, a fraction of edge staples will be removed to program distinct interactions between tiles. Thus, we evaluated the structural integrity of the two bridge designs with and without edge staples. In another set, a full set of inert edges (see section 2.3) were used to keep tiles as monomers but structurally complete. Upon AFM imaging, the strong-weak design without edge staples showed extensive ripped open seams demonstrating the weaker nature of the bridges. The strong-strong design mostly remained intact under the same conditions. Also, when imaging with a full set of inert edge staples, both tiles appeared healthy. This result implies that the scaffold loops and staple bridges do not have preferential tangles or at least are able to resolve them even with strong domains on both sides of the seams. Accordingly, we used the "strong-strong" staple bridge design for all future experiments.

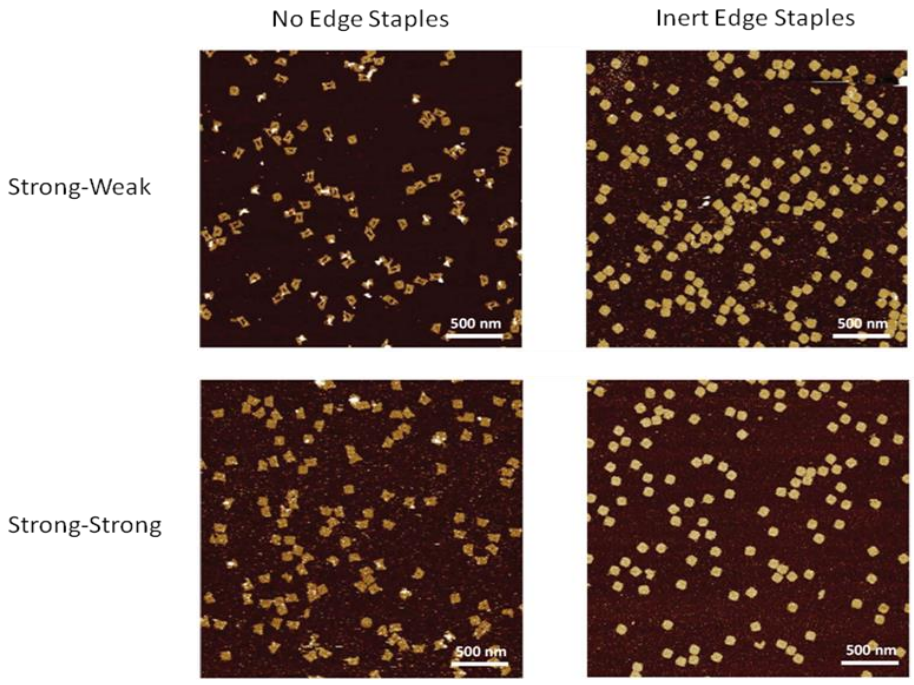


Figure 2.5: **AFM images of strong-weak and strong-strong bridges without edge staples or with a full set of inert edge staples.** Without edge staples, the weaker bridges allowed the tiles to rip apart at the seams. The strong-strong tiles were better able to hold the seams closed on imaging. With inert edges, the tiles were fairly intact in both designs.

2.3 Creating inert and interacting edges

In order to control array formation, control of tile-tile interactions is necessary. This section covers the development of inert edges and the basic design of interacting staples.

Woo et al. [12] explored edges with no staple crossovers at the helix row ends. Having staple crossovers forces the edge nucleotides into the plane interfering with the blunt end's natural capacity to stack. However, that the relaxed edges are able to form the normal 150 degree angle promotes normal stacking. We use the relaxed edge design with no staple crossovers along the tile's edge. Also, each staple binds to two helices for a total of eleven possible edge staple locations per tile side.

Finite arrays require inert edges around the exterior of the array to prevent further tile attachment. Even before that, non-interacting monomer tiles are useful to measure tile integrity. Therefore, a non-interacting inert edge staple would be useful. We investigated four types of edges for their inertness: the absence of edge staples, truncating edge staples leaving free single stranded M13 dangling at the ends, encoding edges with truncated edge staples, and edge staples capped with short hairpins. In the complete absence of edge staples, the tiles did not interact (Fig. 2.5), but there was tile deformation presumably due to the M13 scaffold loops on the edges possessing a non-trivial amount of spurious interactions with each other effectively squeezing an edge. Next, we tried truncated edge staples that may possess less spurious scaffold interactions (see Figure 2.6). All designs, 2, 4, and 6 nucleotide truncations on both the 5' and 3' end of each edge staple, failed to entirely stop tile-tile interactions. With truncations, there was variable amounts of small array formation. With a full set of truncated edges still interacting, we tried encoded the edge with truncated staples of different truncation lengths. First, we used a full set of truncated edge staples with the truncation code of 42462626424, where the eleven numbers represent the eleven edge staples and their respective truncation length. The reasoning is if the 2 nt truncated staples stacked, then the 4 nt and 6 nt staples would be too short to form stacks. The code was designed to ensure that there is no stacking alignment of more than a few stacks even when shifting the edge code along itself. While successfully forming

monomers, the tiles developed worse deformation. When we changed the edge code to _4_2_6_2_4_, where underscores were the absence of an edge staple, tile integrity improved, although there was again an increase in tile-tile interaction.

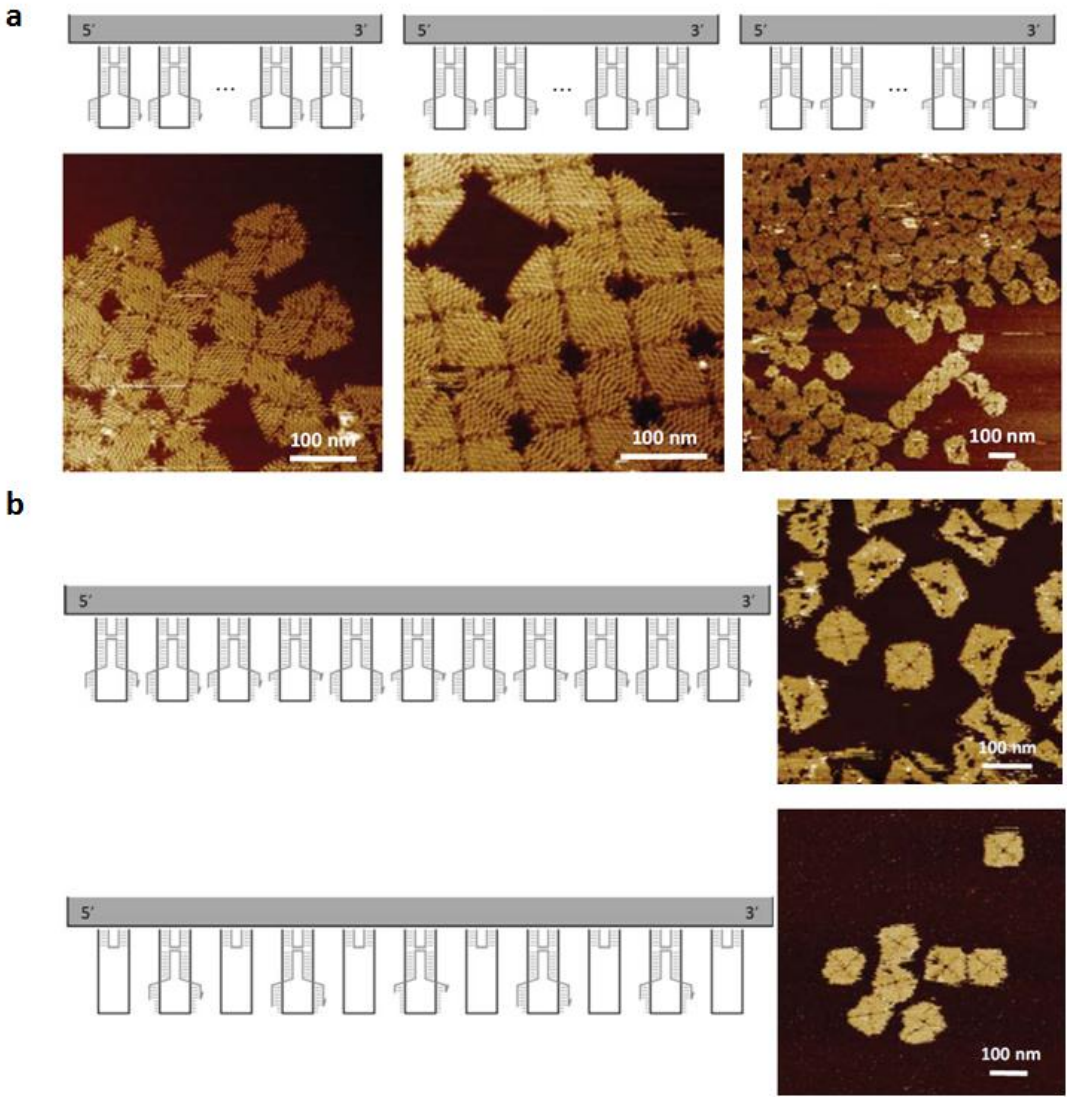


Figure 2.6: **Truncated edges and encoded truncated edges. a**, from left to right: truncation by 2 nts, truncation by 4 nts, and truncation by 6 nts. Each truncation is from the 5' and the 3' end of the staple meaning 4, 8, and 12 nts of free scaffold is formed for each staple design. **b**, encoded truncated edges. The top diagram has an edge code of 42462626424 where the eleven numbers represent the truncation amount of each staple. The bottom diagram has code _4_2_6_2_4_ where underscore is the absence of a staple. While the latter code deformed less, there was still undesired tile-tile interactions.

Moving away from truncated edges, we decided an edge staple might be capped with a hairpin. As a hairpin can close tightly on itself, it can be highly inert unlike scaffold single-stranded domains. We tried four implementations of the hairpin design (see Figure 2.7). The first had a hairpin on the 5' end of an edge staple and a truncation on the 3' end for every edge staple. The second removed every other edge staple along an edge leaving only 5 edge staples. The third had a hairpin on both the 5' end of the edge staple and the 3' end for every edge staple. The fourth design removed every other edge staple along an edge like before but with the double hairpin design. In terms of tile integrity, the double hairpin design was superior and either the full set or spaced layout of edge staples were used in future experiments.

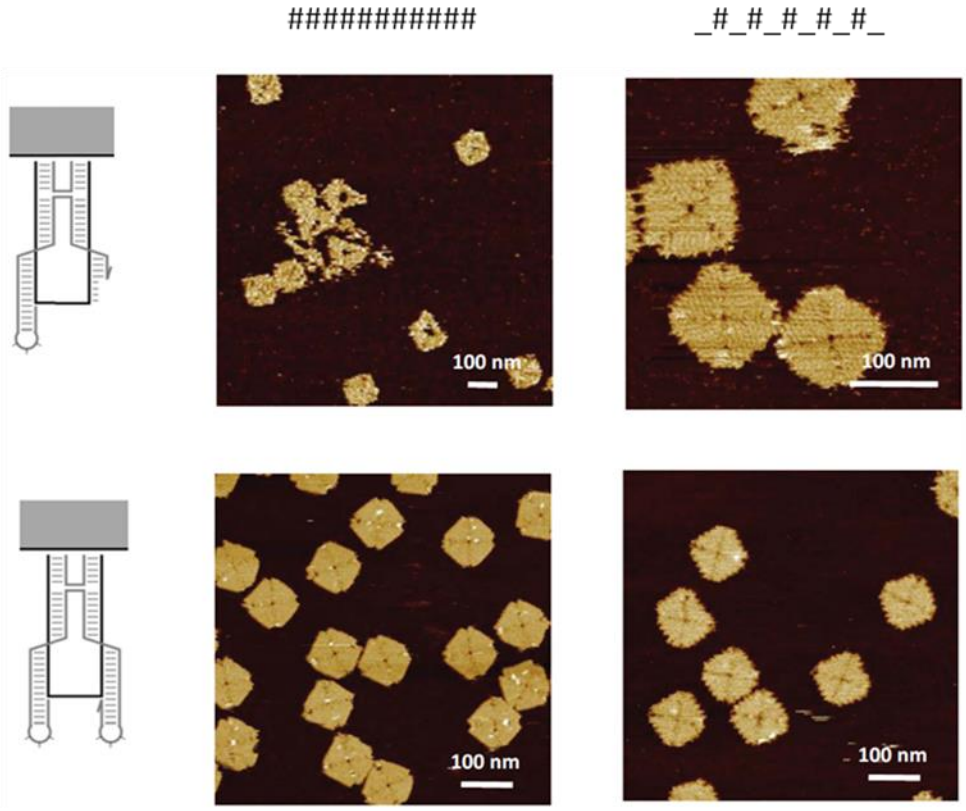


Figure 2.7: **Hairpin capped edge staples**. '#' represents the presence of a hairpin edge staple whereas '_' is the absence of any edge staple at that location. Four designs: 5' hairpin and 3' truncation for each edge staple in either a full edge set (top-left) or spaced layout (top-right), a hairpin on both the 5' and 3' end of edge staples in either a full edge set (bottom-left) or spaced layout (bottom-right).

With inert edge staples able to prevent stacking on desired sides of the tile, there is still a need to have edges that can allow control of tile-tile interactions. In the simplest case, we can include a full set of unmodified edge staples. While similar to the stacking seen in the above truncated edge sets, here each staple will provide a full two stacking bonds with their 5' and 3' ends (Fig. 2.8). While providing an attractive force between two tile edges, a full set of stacking staples lacks significant specificity. Furthermore, the overall edge binding energy would not be tunable with just one edge design. There are two approaches we developed to address the issue of specificity and edge binding energy. First, we can remove edge staples, effectively encoding an edge. With fewer staples, the binding energy decreases as fewer stacking bonds may be made between any two edges. With an encoded edge, the binding energy cost between two aligned interacting edges and misaligned edges can be increased in order to encourage better alignment. Also, by using different edge codes, specificity is improved as maximum binding energy is only achieved when matching edges bind to each other. Second, we can swap one of the stacking ends of an edge staple with either a truncation or a short nucleotide extension ('sticky end'). We predominately truncate only the 3' end of a staple (the "receiver" staple) and extend the 5' end of another staple (the "giver" staple) as this generates enough specificity for most of our purposes. The specificity arises because the 5' sticky end added to one edge staple has a DNA sequence match with the nucleotides truncated from another edge staple. the amount of truncation or extension can vary from one to five nucleotides depending on the application (see individual chapters for the development process of application specific edges). Furthermore, the modified staples can be encoded to tune binding energy or add additional specificity.

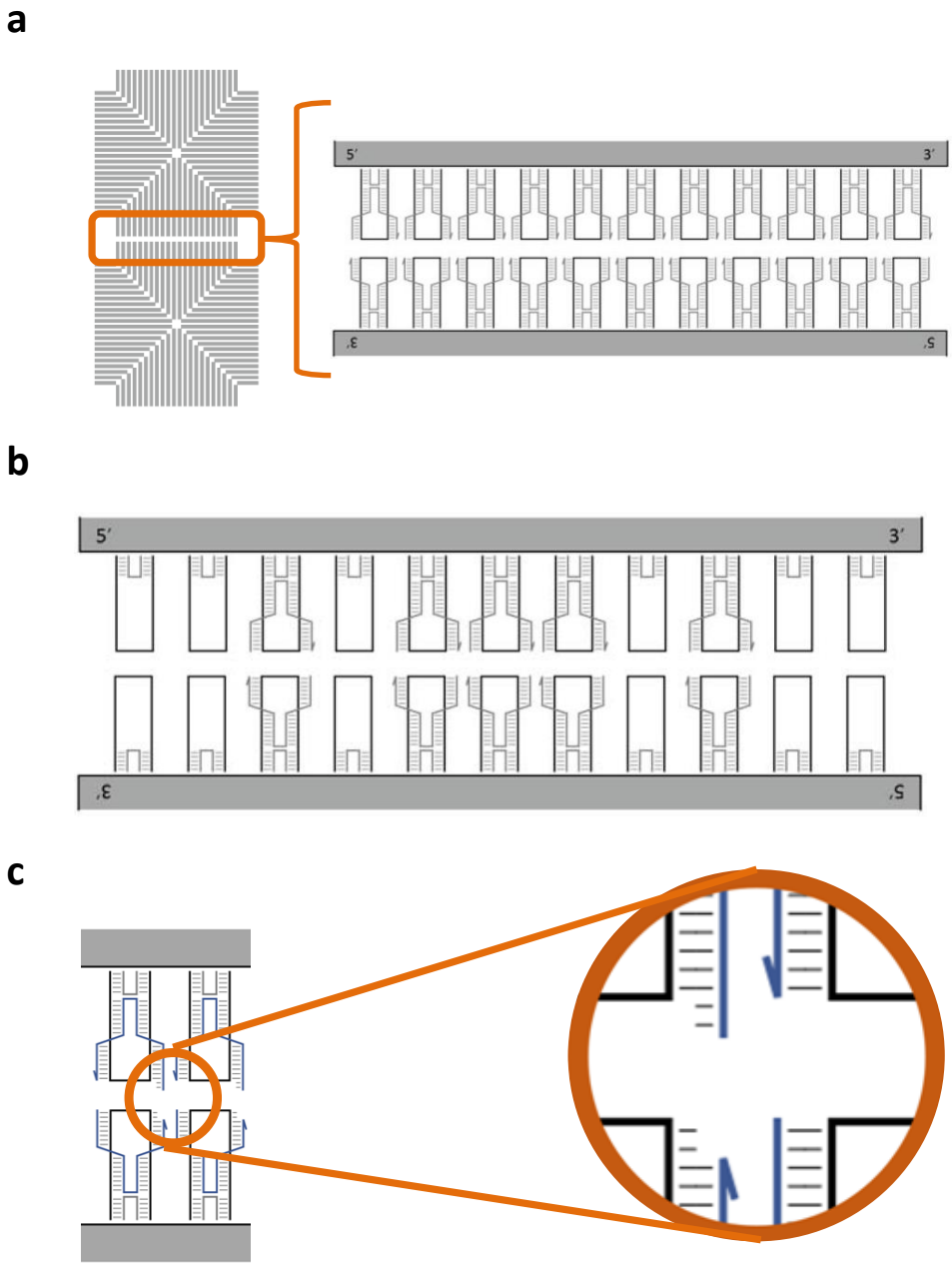


Figure 2.8: **Edge encoding and edge sticky ends. a**, Tile-tile interaction with a full set of unmodified edge staples. All eleven edge staples form two stacking bonds for 22 total stacking bonds. **b**, An example of an encoded edge with unmodified edge staples. Only 5/11 edge staple locations are used resulting in a lower binding energy of just 10 total stacking bonds. A misaligned edge can only form a maximum of 6 stacking bonds. **c**, An example of a sticky end edge staple "giver" (top) and a truncated edge staple "receiver" (bottom). Here the extension and truncation are 2 nucleotides but values from one to five are used. Each staple still forms one stacking bond with its unmodified end.

2.4 Creating a unique pattern

In section 2.1, we laid out a scaffold path that met several important design criteria, one of which is a maximized continuous surface area for the application of arbitrary patterning. To verify if molecules can be placed at precisely defined locations, we needed a method to visualize designed patterns using atomic force microscopy. A simple method is to use extensions of DNA strands to create an increased height of the molecules on the tile surfaces, marking each pixel location in the designed pattern. As different forms of extensions would have different properties in terms of imaging contrast and pixel resolution, we explored several types of surface modifications for their ability to apply simple, yet high-contrast, patterning.

We investigate four design types (Fig. 2.9). The 5' and 3' end of integrated staples within an origami tile are popular locations to attach molecules; although, staple modifications can be made along the staple length. Accordingly, three of our four designs make use of the 3' end of staples internal to our tile with only one design supplying contrast from the middle of the staple strand. Due to the layout of internal staples within the origami tile, the 3' ends of the internal staples form a hexagonal pattern on the same side of the tile. The benefit of this design is that the pattern forms completely on the same side of the tile. An analogy can be made to an electronics breadboard, where electronic components such as chips, wires, resisters, capacitors, etc. can be arbitrarily placed on the same side of a board; albeit, as we are only interested in verifying the placement locations via imaging contrast, it would be more similar to arbitrary attachment of only markers.

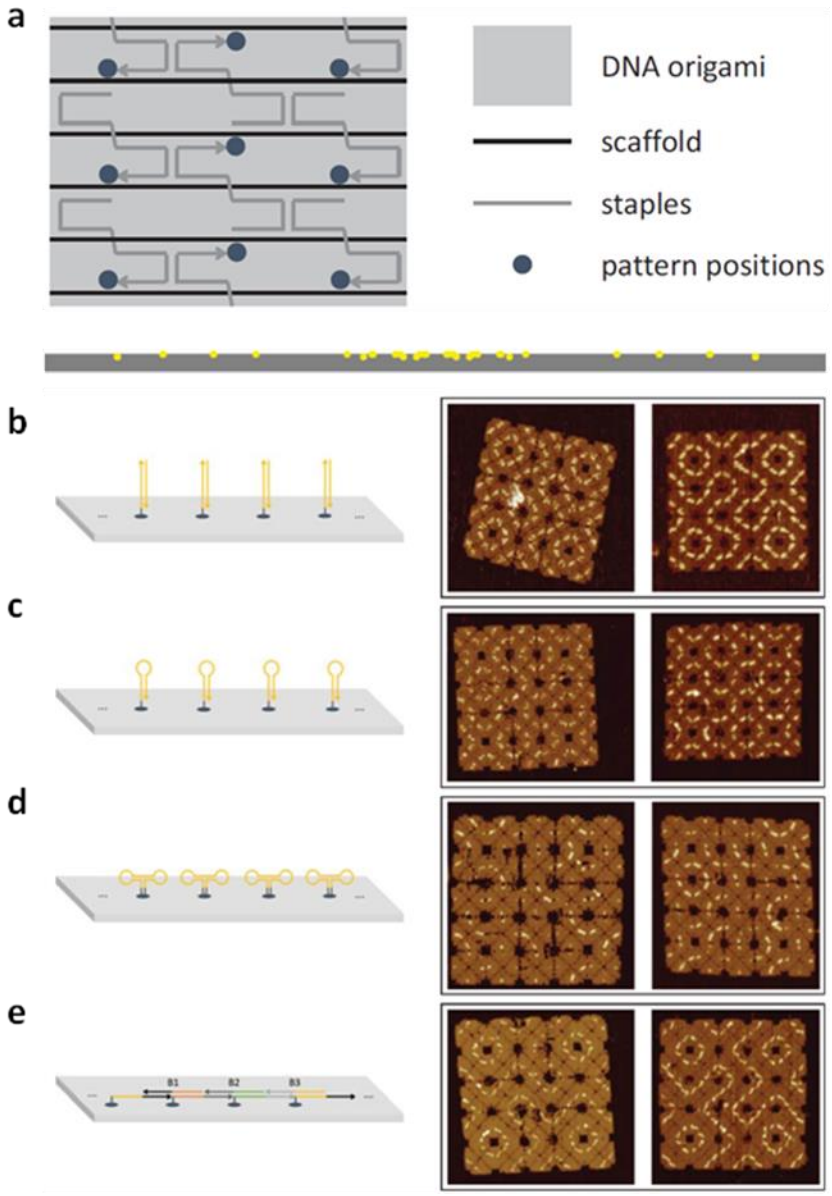


Figure 2.9: **Four staple extension designs for patterning and AFM images of a 5 by 5 tile array with an "arc" pattern. a**, positions of the 3' end of internal staples in a hexagonal pattern and a side view of a tile showing several 3' staple ends all on the same side of an origami tile. These locations are used for (b), (c), and (e). **b**, Extensions are double strands. Internal staples are extended by 20 Ts and are hybridized to a strand of 20 As. The lines in the AFM image are 3 pixels wide with a pixel being an extension location. **c**, Internal staples extended by a hairpin. The lines in the AFM images are 2 pixels wide. **d**, Dumbbell hairpins at the midpoint of each staple [1]. The lines in the AFM image are 2 pixels wide. **e**, Bridged extensions. Internal staples in a line have their 3' end extended by a repeating set of three unique nucleotide sequences. Three 20-nucleotide binder strands are added complementary to parts of two adjacent staple extensions. There are two Ts inserted as spacers between the 3' end of the selected staples and the bridge extensions. The lines in the AFM image are just 1 pixel wide. In (b) and (e), left AFM images have the surface modifications facing up on the imaging surface and right AFM images have the surface modifications facing down.

The first design we tested is a double stranded extension from the 3' end of 'internal' staples (Fig. 2.9(b)), staples not involved with forming bridges or edge interactions. Each selected pixel for contrast, as defined as each useable 3' extension location, was extended by 20 Ts. For imaging, another strand of 20 As would be added and allowed to hybridize to the 20 T extensions, forming a double stranded extension from the tile surface. The extensions are like flexible poles that can move around and this effect is visible on AFM imaging. In Figure 2.9(b), an arc like pattern of lines 3 pixels wide is formed on a 5 by 5 array of tiles. The left AFM image has the double stranded extensions facing upwards and the right AFM image has the double stranded extensions facing downwards—sandwiched between the mica surface and the origami. There is a noticeable change in contrast. When the double strands are facing upwards, the AFM tip likely can push the double strand out of the way everywhere except at the base of the extension where it is fixed. With each extension better localized, the lines of the arc pattern can be seen to be made up of several pixel points. Now if the double strands are facing down, the strands end up stuck between the mica surface and the origami above them. On imaging, the strands are presumably more fixed and the entire strand provides a larger area of contrast. Each pixel covers a larger surface area and connects with neighboring pixels to form an overall more continuous pattern. While we do not specifically control how an origami array lands on a mica surface, it is worth noting that larger origami arrays tend to preferentially land with extensions face down likely due to curvature created by the extensions themselves. Due to the good contrast and extremely simple extension design, we ultimately used this design in all future experiments.

The second design we explored replaces the double stranded extension with a hairpin. The reasoning is that a hairpin naturally has a double stranded domain, and there would be no need to add another strand such as 20 A in order to form contrast for imaging. The results on AFM (Fig. 2.9(c)) were similar in contrast; although, single stranded extensions were deemed to be a simpler design for similar results. The next design we tried was borrowed from a contrast method developed in the first DNA origami research by Rothemund [1]. Instead of modifying the 3' end of a staple, a dumbbell hairpin is added right at the

midpoint of each selected staple (Fig. 2.9(d)). Such a design benefits similarly to the 3' hairpin in that no additional contrast strand is needed; although, staples need more sequence length to accommodate the dumbbell. The last design explored was the most complex but was meant to give the highest resolution contrast even with 1 pixel thick lines. This design aimed to create a bridge-style extension (Fig. 2.9(e)) chain where the contrast is created by a mostly continuous fixed chain of double stranded DNA formed along the pattern lines. 3' ends of staples are extended by 3 unique sequences in sequential order along the length of the line. To connect the unique extensions, 3 binder staples are added. Each 20 nt binder staple is half complementary to the beginning of one 3' extension and the end of an adjacent 3' extension. Therefore, when the binder staple hybridizes, it holds together two neighboring extensions, ultimately forming a chain when multiple binders hybridize. There is a two T spacer between the 3' end of staples and the bridge-style extensions. On AFM, the contrast can be high as expected even despite being only 1 pixel wide. While the contrast is excellent, the design is complex and is geared toward forming lines rather than arbitrary patterns. As we planned to create arbitrary drawings, we opted for the extremely simple first design over the more intricate designs.

With all the basic components forming our tiles determined, we can now move on to using the tiles to engineer the formation of DNA origami arrays and their reconfigurations.

Chapter 3

CONTROLLING GROWTH AND GLOBAL PROPERTIES OF DNA ORIGAMI ARRAYS

Combinatorial approaches for fabricating one-dimensional polymer chains have revolutionized chemical synthesis [1] and the selection of functional nucleic acids [2]. This chapter expand these principles to random two-dimensional networks to open new opportunities for fabricating more complex molecular devices on DNA nanostructures.

Nature adapts through an inherent stochasticity. One has only to look at human's long biologic struggle against viruses to see an example of this in action. Nobel prize winning [3] research showed that developing human immune cells undergo a combinatorial rearrangement of antibody related gene segments. The goal is the formation of novel amino acid sequences in the antigen-binding region of antibodies for targeting a wide range of ever-adapting foreign invaders such as bacteria, parasites, or viruses. Yet we still get sick. Viruses also take advantage of combinatorial diversity to find functional units that have not yet been targeted by the immune system. Every year, people get flu shots yet become sick with the flu. While immunity is developed to the flu, inevitably, illness reoccurs the following year. The resilience of the flu is due to massive combinatorial changes in its surface receptors through random mutations and random switching of receptors among virus strains [4]. Applying these combinatorial methods, so successful in nature, to scientific research has yielded tremendous results.

Combinatorial approaches can be used for many applications. Len Adleman [5] harnessing combinatorial connections of designed DNA molecules to compute the solution to a seven-node Hamiltonian path problem, a computationally intensive task. However, one of the most notable recent advancements with combinatorial chemistry is systematic evolution of ligands by exponential enrichment, known an SELEX. This combinatorial chemistry technique in molecular biology binds nucleic acids of single-stranded DNA or RNA to

target ligands [2,6,7]. The method starts with the synthesis of an enormous combinatorial library of random DNA or RNA sequences. That library is then exposed to a target ligand such as a cancer related protein or other compound. Non-binding sequences are removed and those bound are amplified for further rounds of selection for the very highest affinity sequences. The end result is the ability to make aptamers of very high binding affinity to a wide range of target ligands, including the small molecule ATP [8], the disease causing protein prions [9], cancer stimulated proteins such as Vascular Endothelial Growth Factor [10], and even entire cancer cells [11]. Some aptamers are already approved by the Food and Drug Administration to treat diseases [10,12]. Having an ever expanding library of more functional DNA components to integrate into designs, the field of DNA nanotechnology greatly benefits.

While it is easier to generate a one-dimensional polymer or nucleic acid chain for screening, many natural structures such as cell surfaces or functional devices such as circuits exist in two-dimensions. By expanding the combinatorial principles to another dimension, we generate a framework for programming random DNA tilings in two-dimensions, exhibiting tunable global properties through simple, local rules.

3.1 Growth of unbounded 2D DNA origami arrays

See Chapter 2, section 3 for the development of inert non-interacting edges and the design of interacting edges. This section explores the development of an edge set encouraging large unbounded 2D DNA origami arrays.

The very direct approach to forming arrays from our tile design is to leave all eleven edge staples unmodified and forming two stacking bonds at their 5' and 3' ends for a total of 22 stacking bonds per edge. Arrays up to 500 × 500 nm were self-assembled, see Figure 3.1. Tiles aggregated around the exterior of arrays, and structures showed misalignment between tiles. The misalignment likely arises due to the M13 scaffold sequence dictating the sequences of the edge staples. Stacking bonds are a mix of A-T and G-C base pairs. As G-C stacks have greater energy than A-T stacks [13], edges can misalign and still find

favorable binding conditions. These issues implied the binding energy was too strong and specificity was too low.

In order to reduce binding energy and increase the specificity between edges, we reduced the number of staples in an edge to five. The distribution of the five staples limits misaligned stacking energy to 6 total stacking bonds out of a possible of 10 stacking bonds when perfectly align, remember each staple forms two stacking bonds. Arrays of up to 1 × 1 μm were formed with better alignment but still suffering from some aggregation.

To gain even further control over specificity with low binding energy, we modified the edge staples to either include a 5' sticky end or a 3' truncation. The first edge set we designed had five edge staples evenly distributed along a tile's edge. These edge staples had either a 1 nt sticky end (giver) or a 1nt truncation (receiver). Each edge staple still formed one normal stacking bond, rather than the above two. In order to form arrays, one edge would have staples with 1 nt sticky ends and the edge opposite on the tile would have staples truncated by 1 nt. The sticky ends would be complementary to the M13 scaffold exposed by the 1 nt truncations. As noted with the stacking bonds only tile, the M13 scaffold results in unique sequences on each edge of the square DNA tile. Compared to just encoding specificity in the geometric layout of edges, there is extra specificity of having complementary base pair matching. Unlike the tiles forming arrays with stacking bonds only, tiles with sticky ends and truncations have defined relative orientations. In this design, all tiles have the same relative orientation in the array. Larger arrays of around 2 × 2 μm were grown.

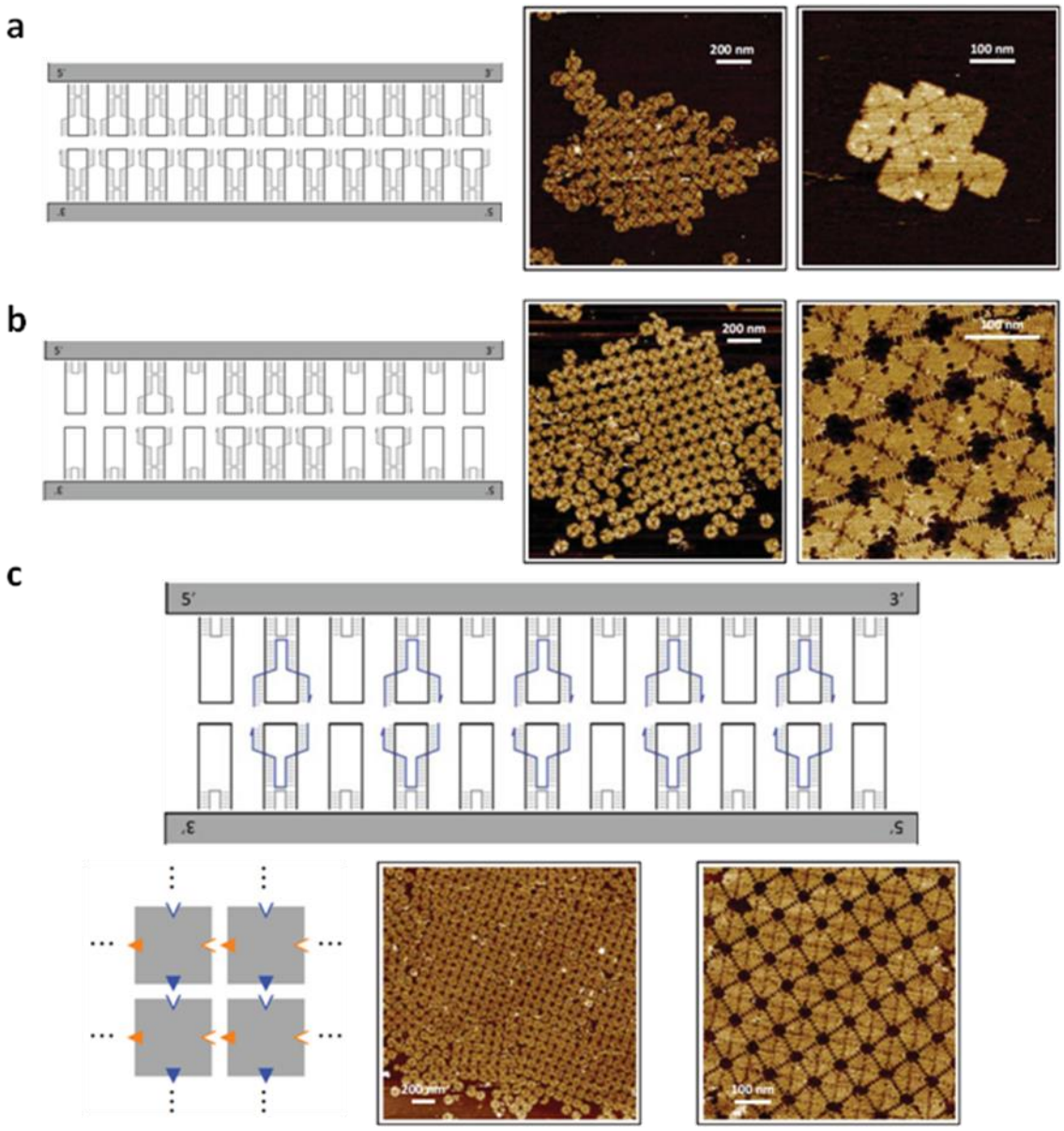


Figure 3.1. **Unbounded arrays of square DNA origami tiles with stacking bonds only, encoded stacking bonds, and 1nt connections. a**, Full set of edge staples forming only stacking bonds. Edge design then a representative array of approximately 500 × 500 nm with a zoom in on a misaligned set of tiles. **b**, Five edge staples forming only stacking bonds. Edge design then a representative array of approximately 1 × 1 μm. There is still some aggregation. The bonds between tiles is visible in the rightmost image clearly showing the 5 pairs of bonds space in the edge design. **c**, five edge staples with a 1nt sticky end and stacking bond. Edge design on top showing the staples are evenly distributed along the edge. On the bottom-left is a tile abstraction showing four tiles in the array. Edges with a 1 nt sticky end extension are shown with a filled triangle and 1 nt truncated edges are indented triangles with colors representing the edge specificity. Representative arrays of around 2 × 2 μm are shown.

While including more sticky end staples would further increase specificity, the binding energy becomes a concern. We tried expanding the specific 1 nt edge staples from just five staples to all eleven staples to determine the consequences of strong binding energy, see Figure 3.2. The previously flat arrays were now forming tubes. Due to the extremely strong edge connections and the flexible seam along the diagonal of our tile, the tubes were resistant to breakage on AFM imaging. We suspect that there was some amount of curvature in our tiles. The curvature allowed the two ends of a growing array to flex together to form a tube. Stronger binding energy promotes tube formation as it forms more stable connections, counterbalancing the entropy cost of restricting a tile into a tube shape. Furthermore, as the bonds are stable at higher temperatures, tubes can form at higher temperatures where the tiles are presumably more flexible.

As we wished to form interconnected patterns on our DNA origami arrays, we needed to test how surface modifications on the origami tiles would affect growth of unbounded arrays. We took the five staple and an eleven staple edge design and place a double arc type pattern on their surfaces consisting of double stranded DNA extensions. Arrays were indeed inhibited. The strong new eleven staple design showed tubes of smaller diameter than with unmodified tiles, presumably implying the surface modification created even greater curvature. Even the five staple design showed ribbon-like structures on AFM imaging, presumably still tubes but with edges too weak to remain stable under AFM imaging conditions. In order to determine which way the tiles were flexing to form tubes, we made a surface design that was asymmetric. This partial arc pattern would create a difference if a tile was face up or face down on an image. Simultaneously, we wished to explore longer sticky ends to see how greater specificity would change array formation. To keep an overall weak binding energy, we only used two edge staples with 4 nt sticky ends. Tubes still formed with this design; however, the partial arc pattern showed the double stranded extensions were on the outside of the tube. This is likely caused by the double-stranded extensions repelling each other. We needed a means to counter the curvature induced by the surface modifications.

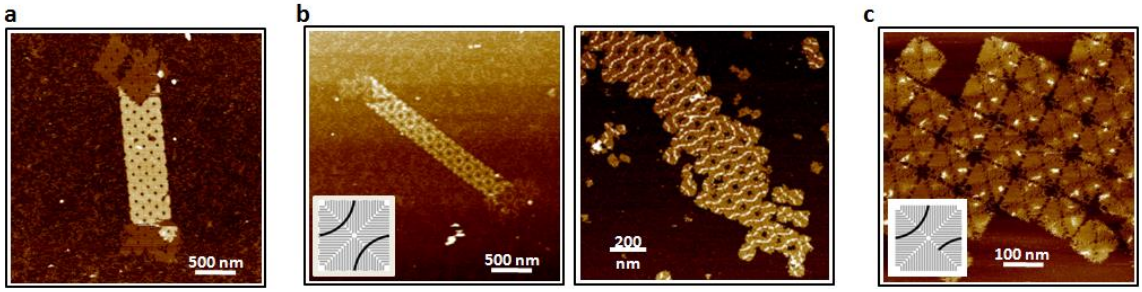


Figure 3.2. **Tube forming unbounded arrays of square DNA origami tiles with strong binding energies and/or surface modifications. a**, Eleven 1 nt sticky end staples. Giving and receiving edges are on opposite sides so all tiles in an array have the same relative orientation. The tile has no surface modifications. The tube is partially opened on the two ends with the double-thick region of the tube represented by a lighter color. The tube flattens along the seams of our tile and remains stable on AFM imaging. Note the large diameter. **b**, left: eleven 1 nt sticky end staples with surface modifications on the tiles. See inset for the modification design on a single tile. The tube has a smaller diameter. right: five 1 nt sticky ends with the same surface modifications. AFM shows a ribbon-like pattern likely due to a tube being unstable and flattening out when imaged by an AFM tip. **c**, Asymmetric partial arc surface pattern with two 4 nt sticky end edge staples per tile edge. Tubes still form, but the face-down surface patterns on AFM imaging imply the pattern is on the outside of the tube.

To counteract curvature, we employed a global curvature correction mechanism, see Figure 3.3. Where tiles all in an array have the same relative orientation, any curvature during self-assembly is likely to accumulate with each additional tile. Thus, instead of having tiles attach in the same orientation, we have each tile attach via a 90 degree rotation. This is similar to the "corrugated" design used in prior cross-shaped origami arrays [14], but avoids having any flipped tiles. Avoiding flipped tiles is important to allow our arrays to be better used as breadboards as we desire all surface modifications to be able to connect to each other on the same side of the tiles. It is also favorable to design all staple extension points on the same side of an origami for maximum surface area usage. Our edge of choice was four staple locations evenly distribution with 2 nt sticky ends. The binding energy is fairly weak yet provides increased specificity for forming arrays. We first tested tiles without surface modification, obtaining arrays. Next we added the arc pattern to the tiles to determine if tubes or arrays would form. The global curvature correction mechanism appeared to work in that large 2D arrays formed. The increased specificity also helped the formation of the 2D arrays. Arrays had more uniformly-orientated domains and were also more crystalline, containing straight edges akin to the facets found in crystals.

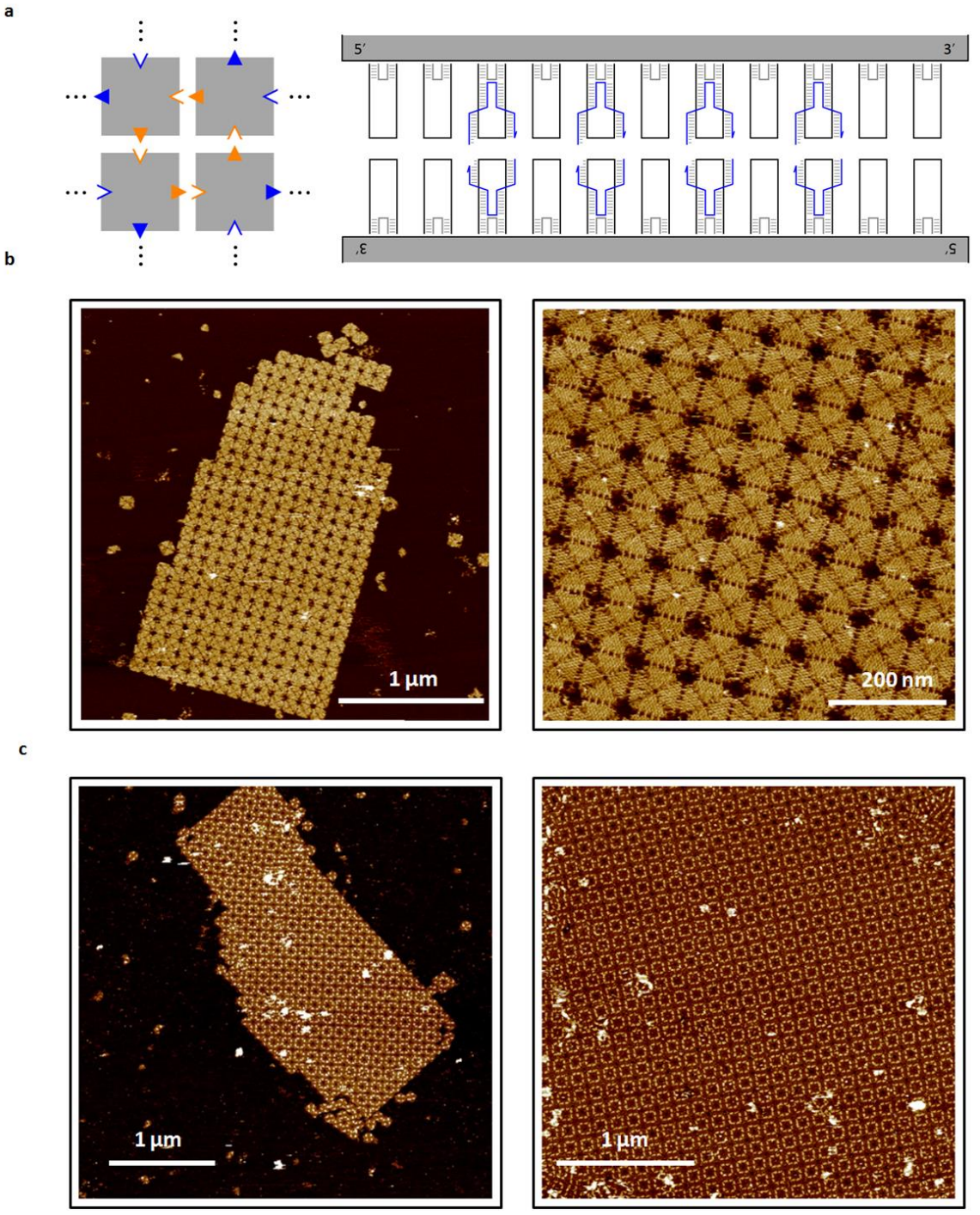


Figure 3.3. **Unbounded arrays of square DNA origami tiles with global curvature correction. a**, left: array abstraction diagram. Each tile is rotated by 90 degrees compared to its neighbors. right: Edge diagram. Four edge staples with 2 nt sticky ends provide specificity yet weak binding. **b**, array formation of unmodified tiles. Array edges are straight as in a crystal. **c**, array formation with arc tiles.

With a design able to support surface modification, we then turned to characterizing and optimizing conditions. First, we experimentally investigated the melting temperature of our edges to better understand at what temperature our arrays form. By centering our temperature annealing schedules around the melting point, we should optimize array growth. At high temperature, origami becomes overly flexible and even unstable, thus the best melting points should be much lower. We added a fluorophore-quencher pair to our edge, overall adding only one additional stacking bond as the fluorophore-quencher pair did not touch. We allowed the tiles to form two by two arrays where the fluorophore and quencher come into proximity when the array is formed resulting in lowered fluorescence intensity (see Figure 3.4). When the arrays melt with increasing temperature, the tiles separate resulting in increased fluorescence intensity. It was determined that the melting point was close to 35 degrees Celsius. We also verified the data with AFM experiments. By mixing pre-formed arrays with two different patterns and heating to different temperatures, we could find the temperature resulting in the tiles mixing—the melting point. We tested 30, 33, 35, 37, and 40 degrees Celsius. 30 was similar to 33. 37 and 40 were similar to 35. Thus, the melting temperature was likely close to 35 degrees Celsius.

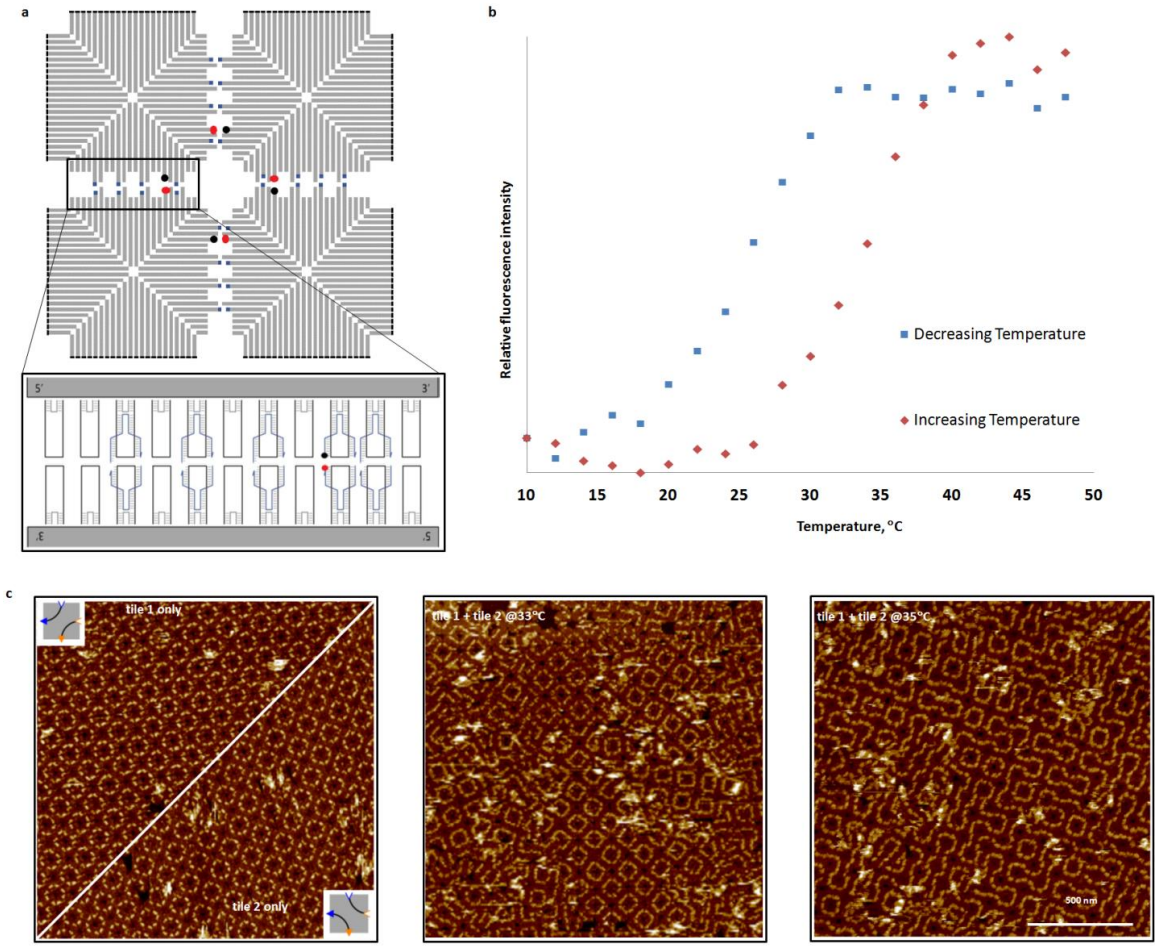


Figure 3.4. **Fluorescence and AFM melting temperature experiment. a**, Tile abstraction of a two by two array with a fluorophore and quencher pair shown as the red and black dot respectively. When the array forms, the pair are in proximity and the fluorescence will be lower. **b**, melting graph showing the relative fluorescence intensity when heating and cooling the sample as measured on a Fluorolog-3 spectrofluorometer with temperature control (Horiba Scientific). Tiles were heating in 2 degree Celsius increments at 5 minutes per increment for equilibration. The fluorophore is ROX, measured with 584 nm excitation and 602 nm emission wavelength. Each data point is the average of three data points taken over three heating and cooling cycles. Tiles are at the concentration of 50 nM in a 500 uL quartz cuvette. **c**, left: Two arrays of origami with a mirrored arc pattern were formed separately. middle: The two tile arrays were mixed together and heated to 33 degrees Celsius. There is marginal mixing with arrays mostly intact as seems from the circle patterns. right: the two arrays heated together to 35 degrees Celsius. The patterns merge together showing random loops rather than circles.

Next, we tried annealing our arrays over different time durations (see Figure 3.5). Multi-day anneals gave large arrays. Varying the annealing time from two days to a week, arrays of up to 10 by 10 microns were formed. This array would contain several thousand tiles; see Figure 3.6a and 3.6b.

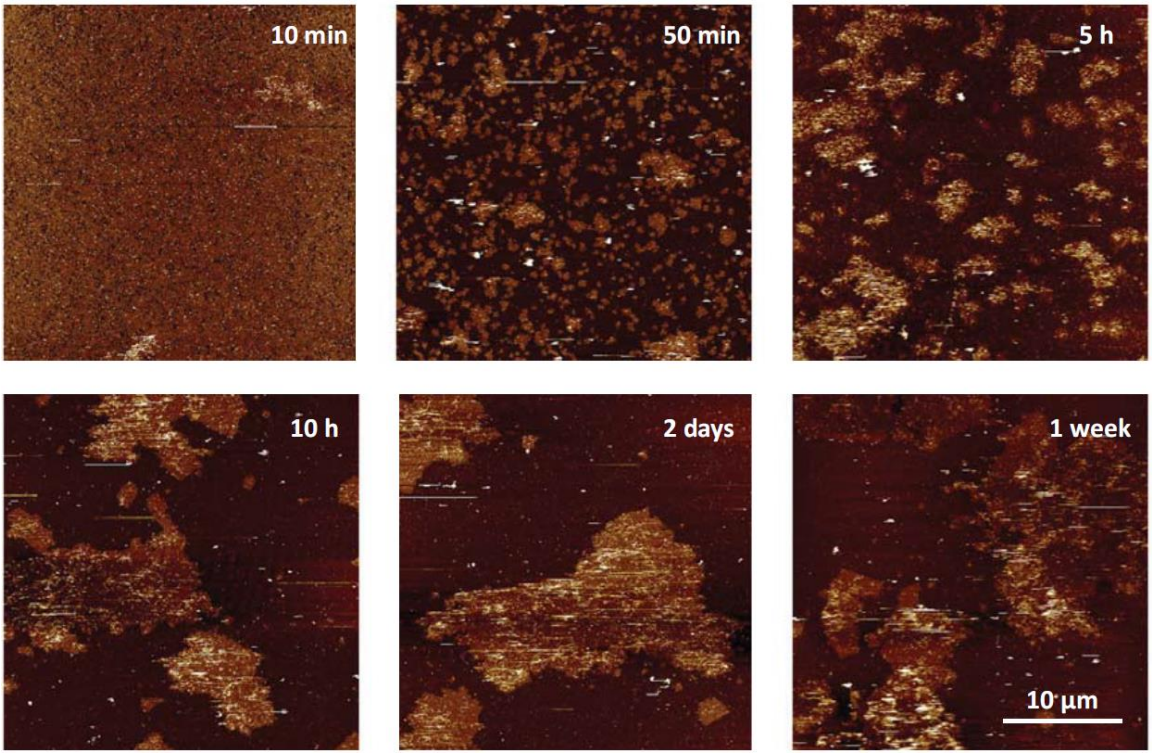


Figure 3.5. **Representative AFM images of unbounded arrays with increasing anneal time.**

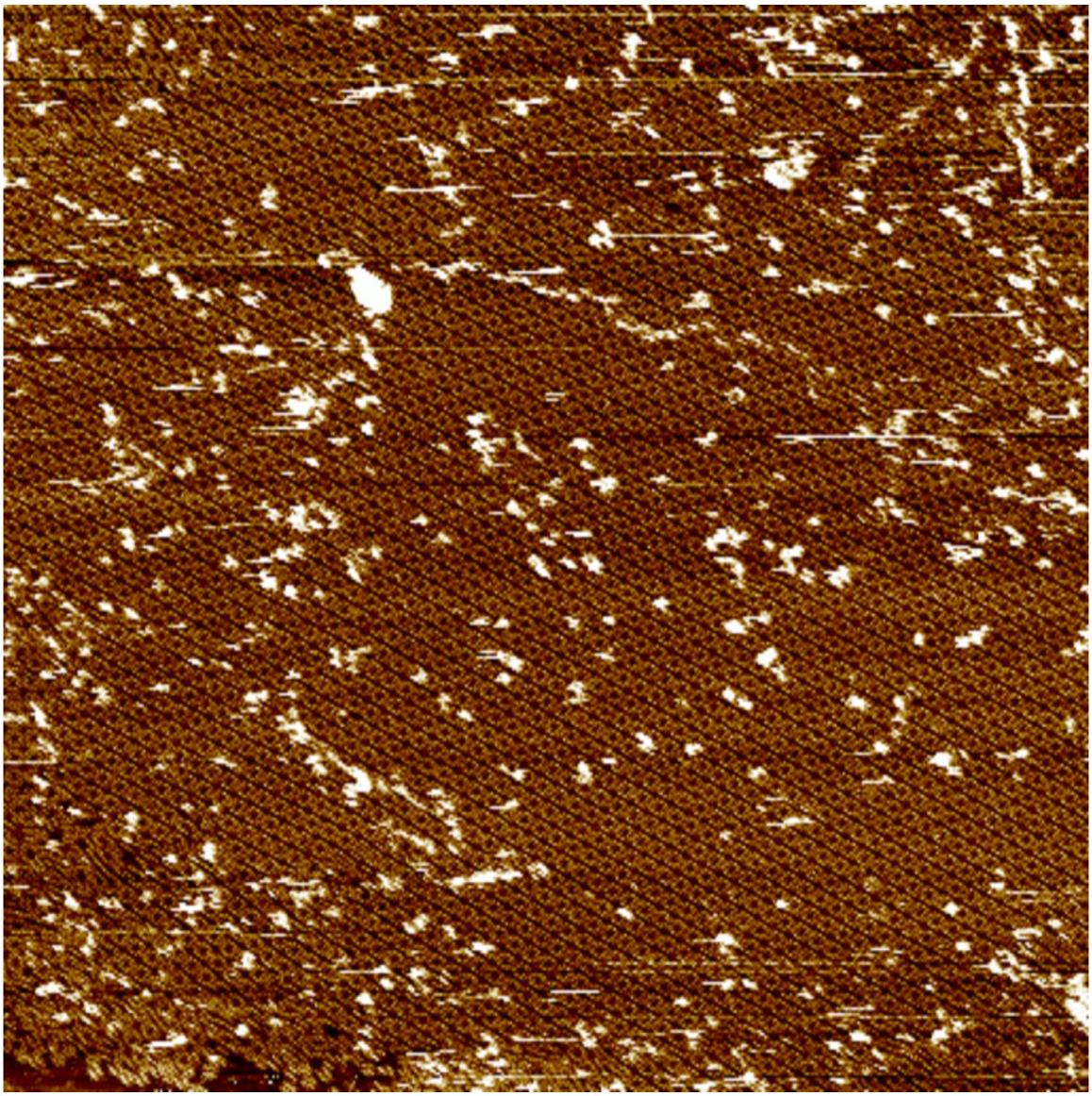


Figure 3.6a. **7 by 7 um AFM image showing part of a crystalline domain of unbounded tiles with double-stranded surface modifications forming circle patterns**.

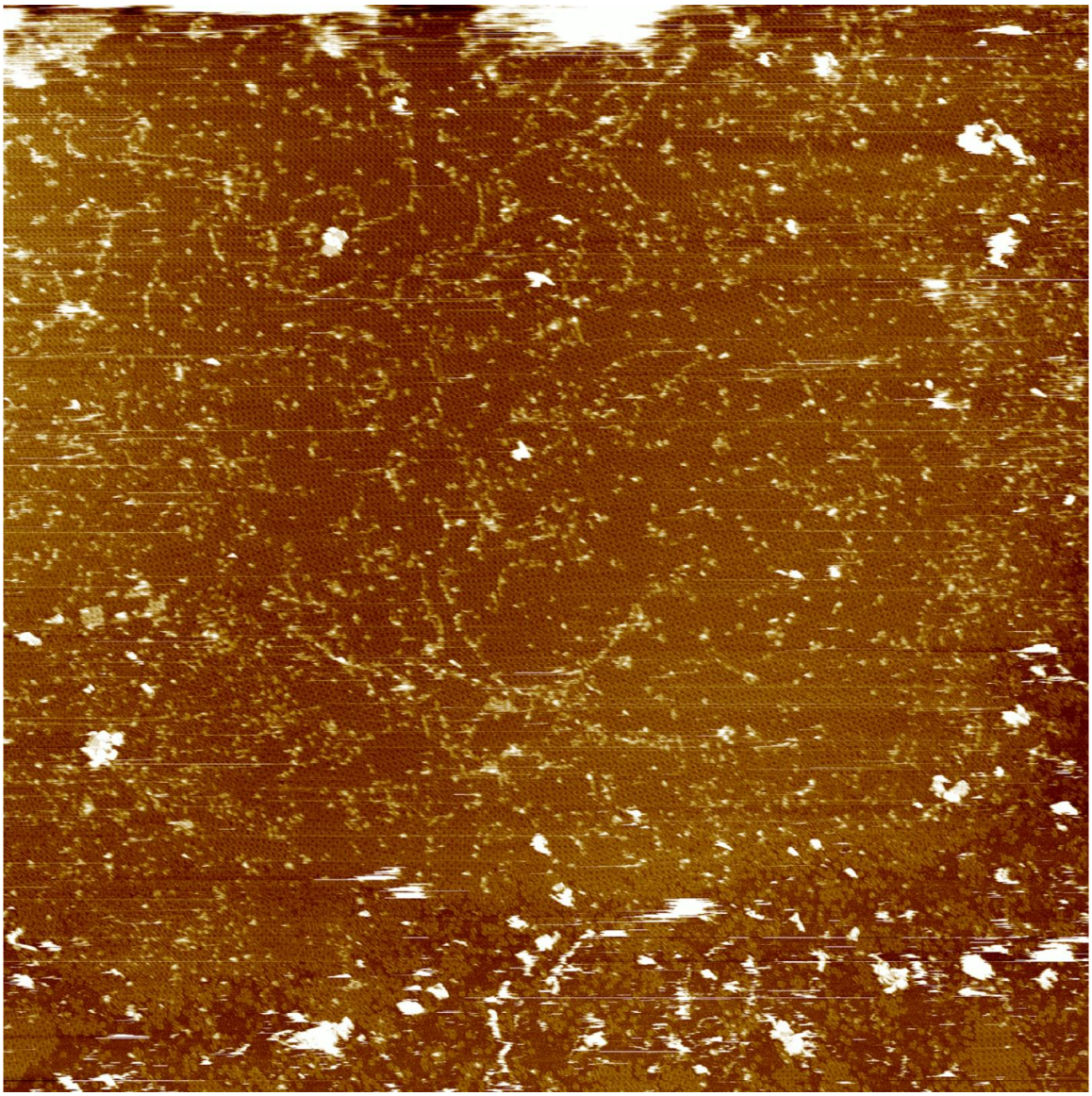


Figure 3.6b: **A 16.2 by 16.2 um AFM image showing part of a crystalline domain of unbounded tiles with double-stranded surface modifications forming circle patterns**. There exists some defects at this scale. Zoom in to see the circle patterns showing the domains.

3.2 Global properties controlled by programming pattern design on individual tiles

With the capacity to grow the largest-to-date 2D arrays of origami tiles, next comes the process of controlling global properties of the arrays. This section explores the first of three levels of control: the design of local tile patterns. Most of these methods can be

implemented using only a single type of edge design. Hence, experimentally robustness is created as an incorrect tile attachment requires four errors, all four sides of the square, to be incorrect to integrate into an array.

The surface design of interest to represent the level of control over global properties is Truchet tiles [15]. A Truchet tile has a rotationally asymmetric pattern that continues into neighboring tiles, resulting in complex patterns. We used two surface pattern designs: a double arc design and a T shaped design (see Figure 3.7). For an array of arcs taking random orientations at each tile location, the lines continue between tiles forming non-branching loops of varying sizes or touch the edge of an array. However, in order to create mazes, branching is needed and can be formed by designing how lines connect within and among tiles. If one looks at the area between the loops on arc tiles, it is analogous to diagonal tiles. A diagonal tile can introduce branches where four tiles meet at their corners with either a three or four-way branched junction. There is also a variable distance between junctions with shorter distances being more likely. If branching points are placed within a tile such as with a T pattern on a tile, four-way junctions are removed, and since every tile center has a junction, the distance between junctions becomes fixed with a separation of a single tile's length. Again, by looking at the area between the lines, new global properties emerge.

Beyond branching, other properties vary. The proximity or degree of interconnection between global array patterns will change depending on the tile pattern design. In arrays where arc/diagonal tiles rotate randomly, adjacent mazes are interwoven. However, in arrays formed from T pattern tiles, large mazes often occupy their own large rectangular areas with tiny mazes occasionally scattered within them. Furthermore, the size distributions of mazes formed between the tile surface pattern designs varies significantly. Simulation results of randomly generated mazes showed the size of the largest mazes on random arrays of T tiles is generally larger with a wider distribution than the largest mazes formed by arc/diagonal tiles; see Figure 3.7(d and e). Looking at the area between the T tile

mazes, smaller mazes were formed. We set out to experimentally demonstrate the simulation results.

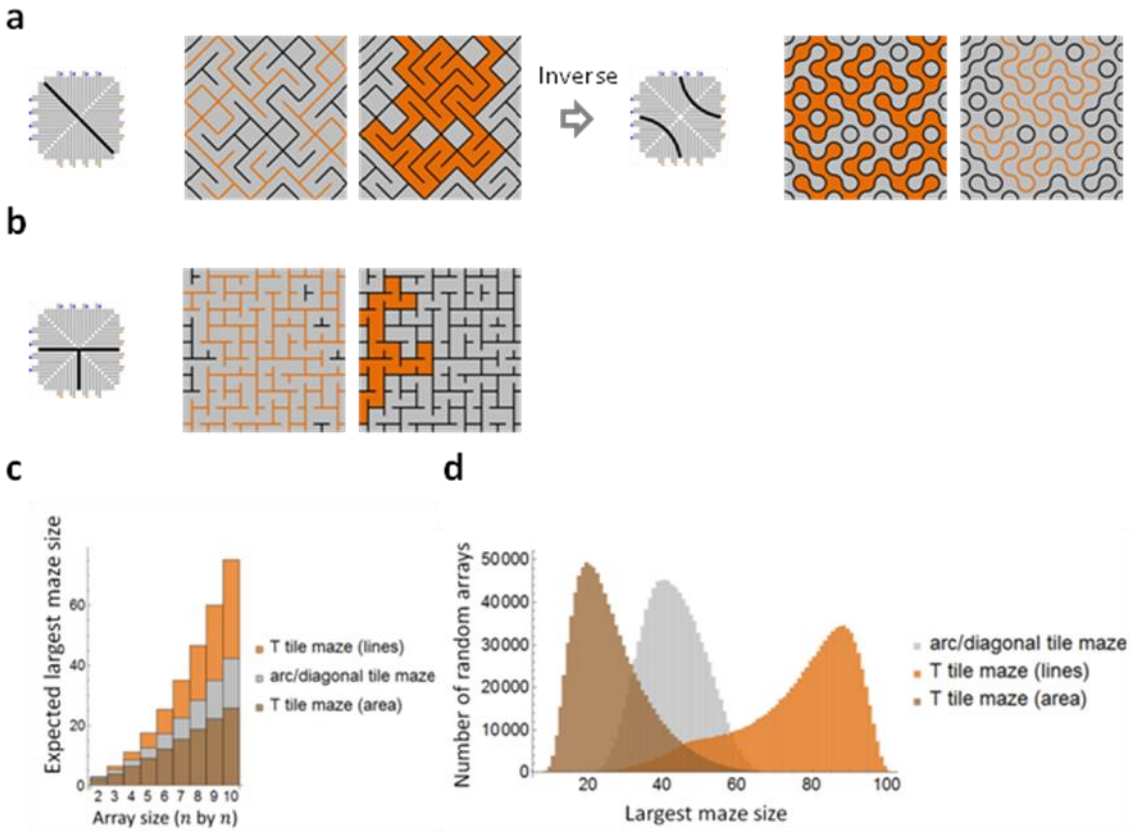


Figure 3.7. **Programming the tile. a**, a diagonal tile maze looking at the maze formed from the line and the area between the line. Note the arc tile design is the inverse of the diagonal tile design. The largest maze is highlighted in orange. **b**, T tile design. **c**, Simulation from ten thousand independently generated mazes showing the average size of the largest maze on random arrays of size 2 by 2 to 10 by 10. **d**, Histogram comparing the largest maze size on 10 by 10 arrays formed from simulations of a million independent trials.

First we desired to prove that we could integrate two tiles with the same edges but different surface patterns into an array with a random choice between the two tiles. We mixed equal amounts of two tiles each with a mirrored version of the double arc as compared to the other with the intent to form loop mazes. We then noted the orientation of each tile and made a layout of two by two local neighborhoods. If tile integration was random, there should be little bias for any two by two local pattern formation; see Figure 3.8. No notable

bias was found, implying tile integration was not significantly affected by the surface modification.

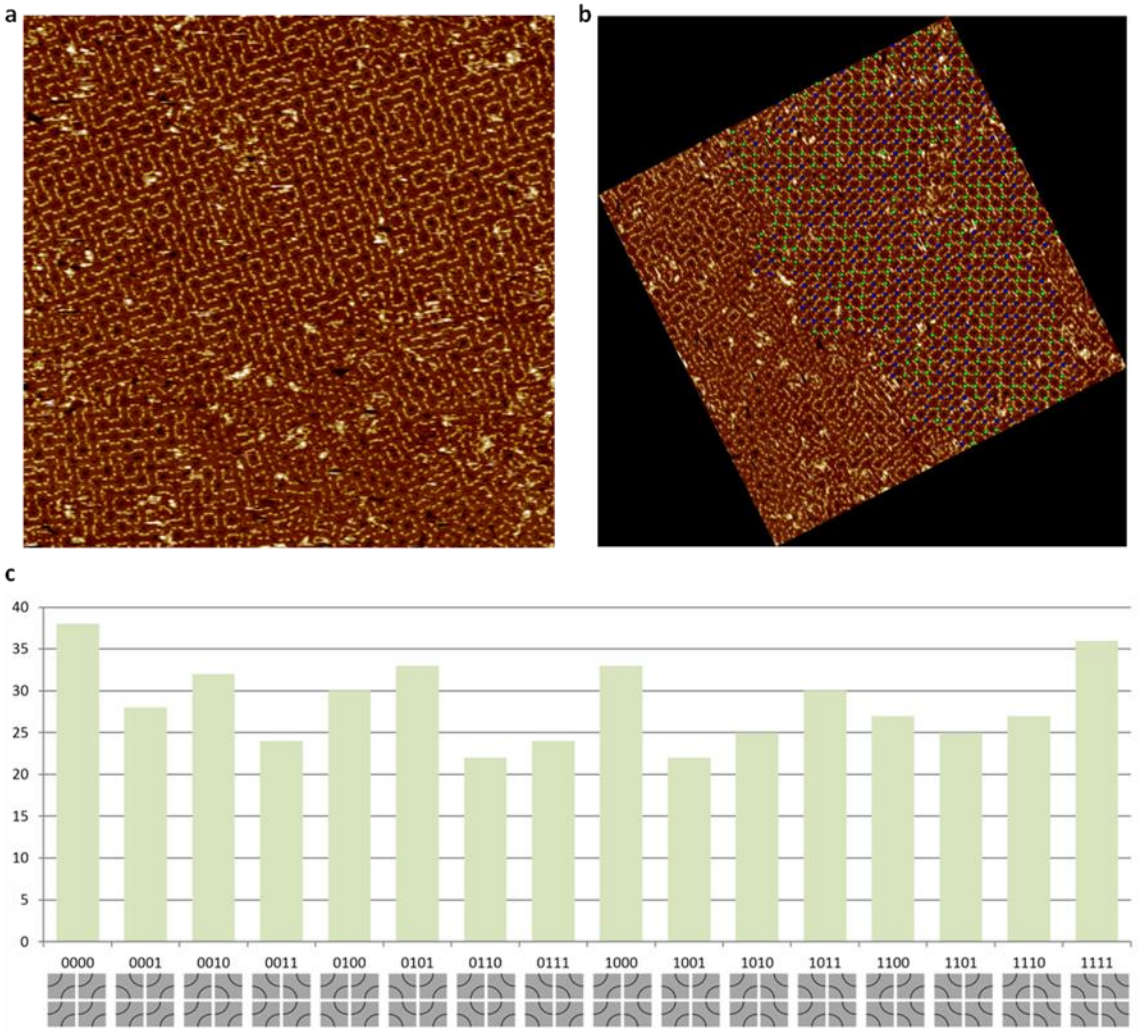


Figure 3.8.**Analysis of tile orientations of arc tiles forming a loop maze pattern on an array. a**, Representative AFM image showing arc tiles in two orientation resulting in a loop pattern. **b**, rotated image with the tile orientation marked with blue or green dots. **c**, Counts of all 16 possible two by two local neighborhoods. n=456 neighborhoods. Each pattern appears 28.5 ± 4.9 times (6.25% ± 1.07%).

We then performed analysis on several separate tile areas of size 10 by 10 by randomly selecting a zoomed in location on large arrays (Figure 3.9). Sizes of the longest loop and largest maze from the area between the loops were simulated and compared to the experimental data for the 10 by 10 tile areas. Experimental data roughly matched the

simulated results within the 95% confidence interval. Mazes showed the expected properties: branching with three and four-way junctions varying from one to six tiles apart, network proximity, and differences in maze sizes. In order to form random T tile arrays, we used four tiles each with the T design rotated by 90 degrees to cover the four possible orientation of a tile. Analysis of random 10 by 10 tile area also matched simulated results within the 95% confidence interval; see Figure 3.10. These mazes showed three-way junctions only one tile apart. While the arcs were interwoven, the T mazes were mostly separated. The largest arc maze was 39.4 ± 4.5 with expected size 42.3, whereas the largest T maze was 77.8 ± 9.5 with expected size of 75.3.

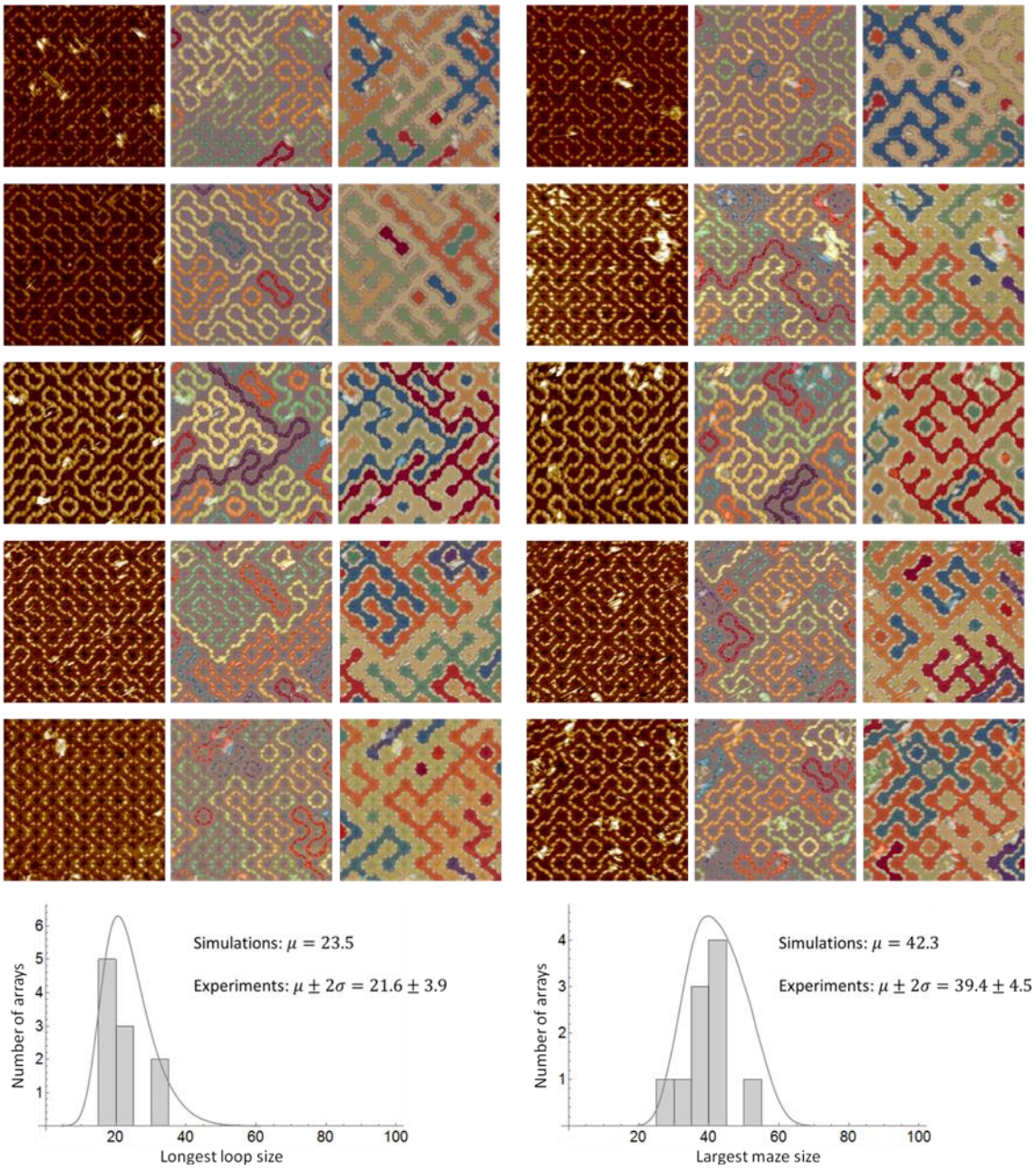


Figure 3.9. **Analysis of arc loops and mazes for 10 by 10 tile area.** Each maze shows the original AFM image on the left. In the middle are the loops with each separate loop colored a distinct color to better distinguish separate loops. On the right are mazes formed from the area between arcs again with separate mazes colored distinctly. At the bottom, the longest loop and largest maze from each of the ten 10 by 10 tile areas are overlaid with the probability density function (scaled to 100) generated from simulations. μ is the mean, σ is the standard error of the mean, and ±2σ corresponds to 95% confidence. The size of a circle is set to 2 with the total size of all loops in a 10 by 10 tile area being 100.

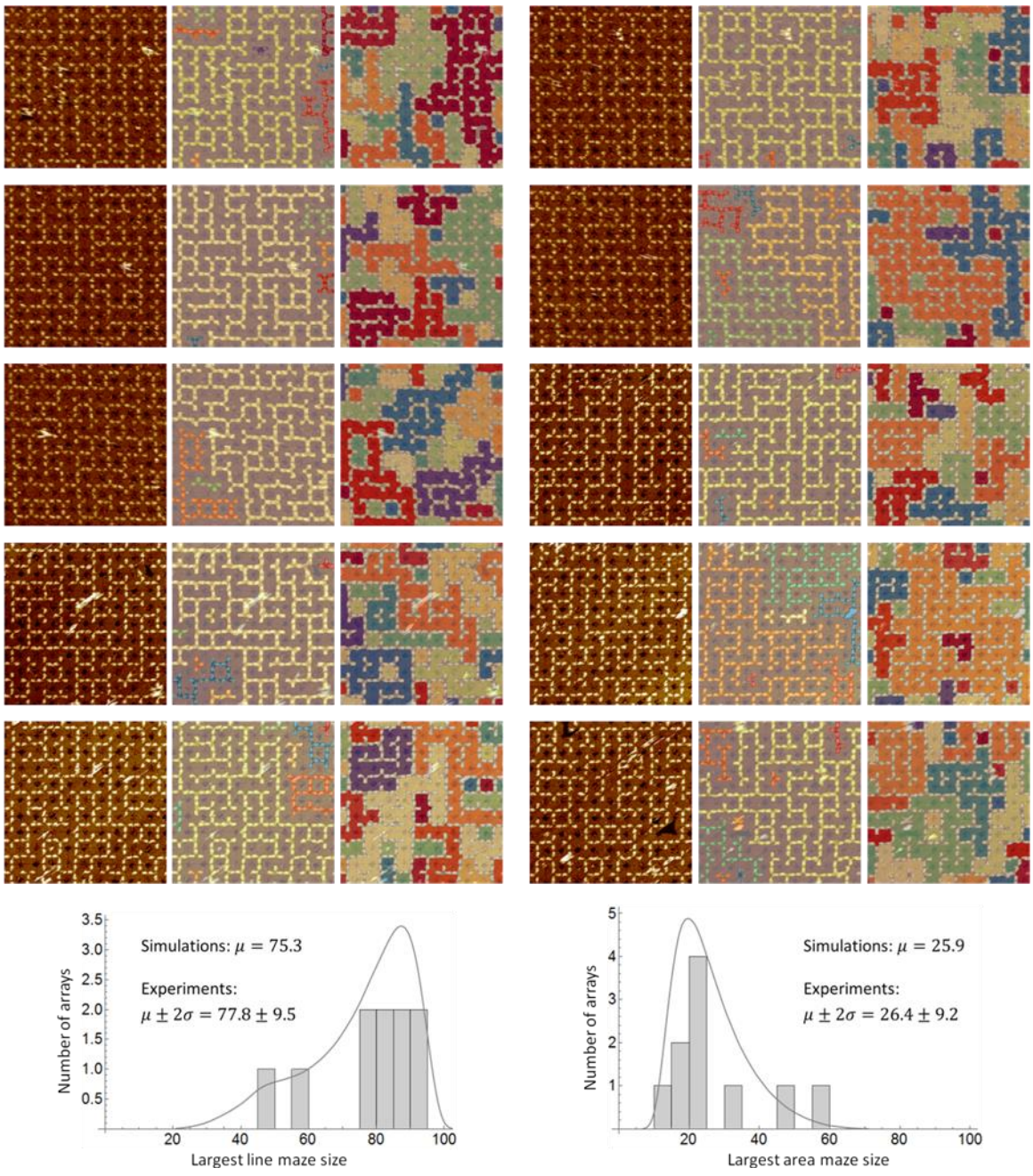


Figure 3.10. **Analysis of T mazes for a 10 by 10 tile area**. Ten 10 by 10 tile areas of T mazes. Each maze shows the original AFM image on the left. In the middle are the line mazes with each separate line colored a distinct color. On the right are mazes formed from the area between the lines again with separate mazes colored distinctly. At the bottom, sizes of the largest line and area maze are overlaid with the probability density function (scaled to 100) generated from simulations. μ is the mean, σ is the standard error of the mean, and ±2σ corresponds to 95% confidence. The size of a T tile is set to 1 with the total size of all mazes being 100.

3.3 Global properties controlled by programming the tile-tile interactions

This section explores the second of three levels of control of global properties, programming the grid through tile-tile interactions.

Previous mazes all contained loops. For example, with T tiles able to take all four orientations at any location in an array, the simplest loops that could form are on a 2 by 2 tile area. This simple loop may also form with only two orientations if those two orientations are 180 degree rotations of each other. Note that there are possible orientations 90 degrees rotated from each other where loops will no longer form (Figure 3.11). Furthermore, the growth direction will be along the diagonal of arrays forming a tree-like pattern considering the corner tile as the root.

The growth direction of the trees may be instead designed so that trees could grow in all directions. This is accomplished by changing the tile orientation grid. The one-direction tree assumed all tiles had the same relative orientation in the grid. However, if the grid had four distinct orientations in each 2 by 2 tile area, a single type of T will always form square loops of size 4, when setting a single T tile to size 1. Any two types of T tiles on this four-orientation grid would either be in phase (there exists alternating choices in adjacent 2 by 2 tile areas to connect loops together) or out of phase (preserving a loop would necessarily break adjacent loops). Setting each tree to a single loop, called the root, trees can be grown in all directions on the four-orientation grid using two T tiles that are out of phase with each other. This can be done with specific 90 degree rotated tiles and 180 degree rotated tiles. While the root will vary in size, larger roots are statistically less likely than a small root. Simulations of random 10 by 10 arrays suggest above 80% of T90 trees (two 90 degree rotated tiles that are out of phase) and above 99% of T180 trees will have a 'square root' on four tiles, the smallest possible loop.

Besides differences in the percentage of smaller roots, T90 and T180 trees will exhibit distinct branching properties. T90 trees will have straight branches with variable lengths, shorter being more likely than longer, whereas T180 trees have only two possible lengths

of straight branches. There are size variations as well. The largest tress on T90 arrays will be expected to be larger than on T180 arrays with a wider distribution.

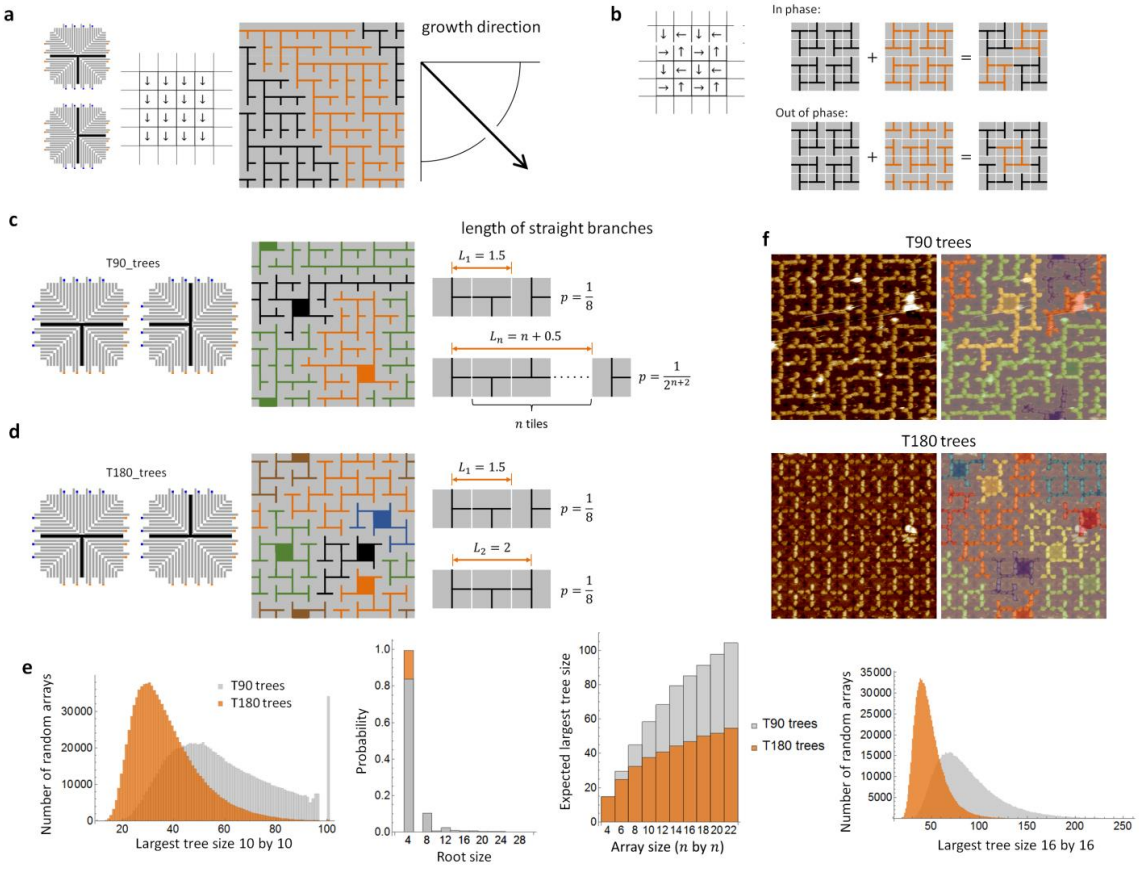


Figure 3.11. **Programming the grid to form tree-like patterns in all directions. a**, A T tiles design on a grid with all tiles having the same relative orientation. Arrows represent a specific orientation of a tile at the given location, not having an effect on the orientation of the pattern on that tile **b**, four-orientation grid with an example of two tiles that are in phase when mixed together and two tiles that are out of phase when mixed together. **c** and **d**, Two T tile designs on the four-orientation grid generating tree-like properties with different branch lengths. For the ease of analysis, we assume each array in on a torus. Each tree is shown in a distinct color for visual clarity. The root is filled with the same color as the branches for visualization. **e**, From left to right. Histogram of the largest tree size on 10 by 10 arrays from simulations with a million independent trials. Probability of a root size in 10 by 10 arrays from simulations with ten thousand independent trails. Average size of the largest trees on random arrays from 4 by 4 to 22 by 22 from simulations of ten thousand independent trials per array size. Histogram of the largest tree size on 16 by 16 arrays from simulations with a million independent trials.

The corrugated edge design implements the four-orientation grid as each tile integrates 90 degrees compared to its neighbor, mimicking the four-orientation grid layout. Using T tiles

in two orientations we construct T90 and T180 arrays. Trees grew in all directions with expected branching patterns. T90 straight branches had variable length straight branches from 1.5 to 8.5, and the T180 trees had straight branch lengths of only 1.5 and 2. Expected size differences between the largest trees were also observed. The largest T90 tree was 54.7 ± 11.7 whereas the largest T180 tree was 33.4 ± 11.9, agreeing with expected sizes of 58.0 and 36.7 from simulations (see Figure 3.12). For greater control, more complex grids may be generated with several tiles with distinct edge designs at the cost of increased design and experimental challenges; see section 3.5.

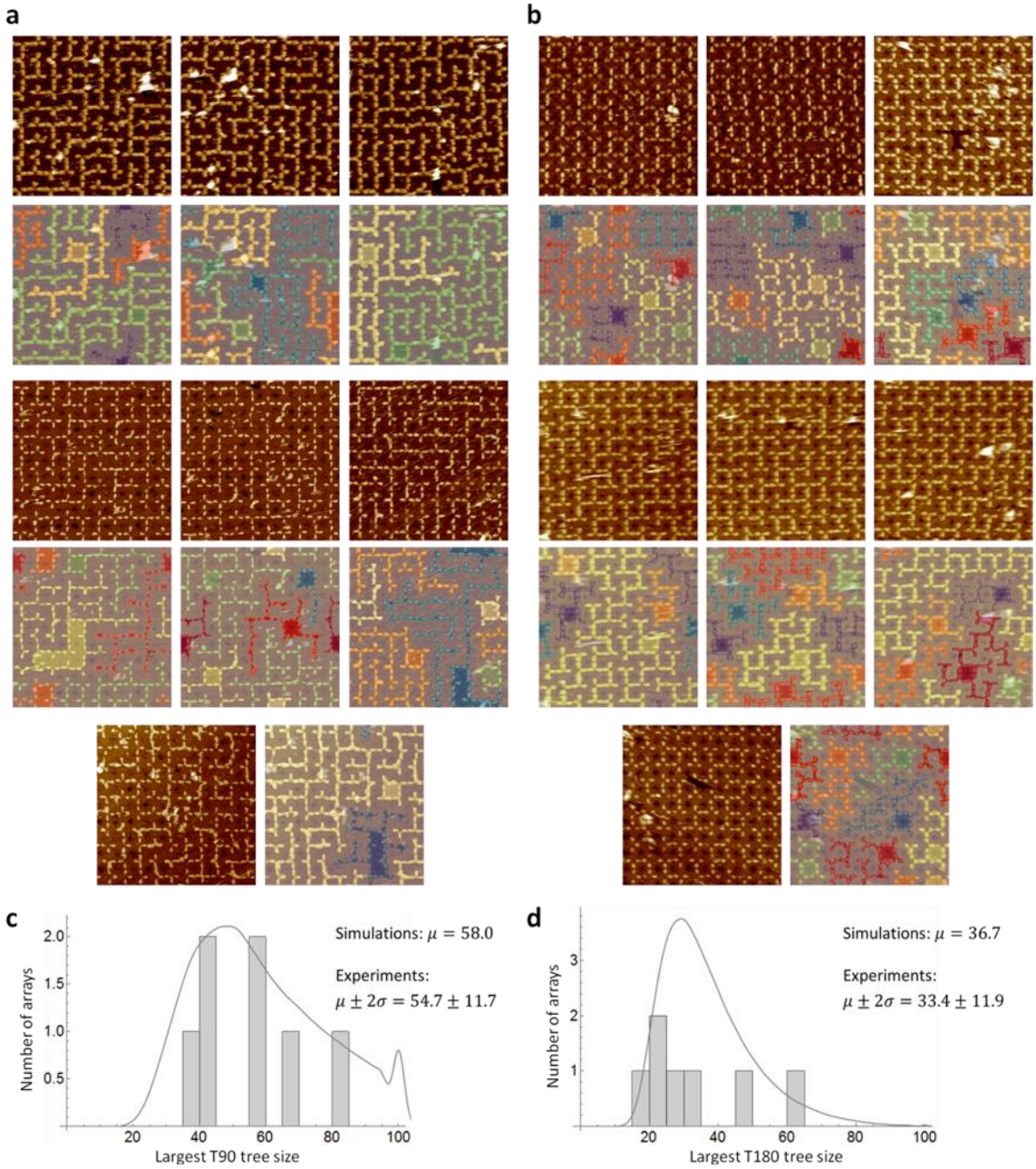


Figure 3.12. **Analysis of T90 and T180 trees on 10 by 10 tile area**. Each maze shows the original AFM image on the top. Below is an image with each distinct maze colored a distinct color and the root of that maze being filled in with the maze's color. **a,** T90 trees. **b**, T180 trees. **c** and **d**, Sizes of the largest maze from each of the ten 10 by 10 tile areas are overlaid with the probability density function (scaled to 100) generated from simulations. μ is the mean, σ is the standard error of the mean, and ±2σ corresponds to 95% confidence. A single T tile is set to a value of 1 and the array has a total value of 100.

3.4 Global properties controlled by programming the probabilities of tile choices

This section explores the last explored level of control of global properties, programming the random choice of tile.

The prior two sections involved equal probability of tiles with different pattern orientations obtained by mixing an equimolar ratio of the tiles. If instead, the tile ratio is tunable, the size distributions of global patterns can be further controlled; see Figure 3.13. In the prior section, T90 trees were growth from two T tile patterns 90 degrees rotated from each other. If the total tile population is represented by 1 and the probability of one tile type is called p, then the other tile's probability is 1-p. When the probability p of a tile is not present at all 0 or the only tile present 1, all trees are just the square root with expected size equal to exactly 4. When p varies between 0 and 1, then the trees grow larger with an expected maximum size at $p = 0.5$.

More tiles with multiple pattern types may be introduced. By tuning their probabilities, we gain control over new features of patterns. A tile with a cross-like pattern mixed with the arc pattern would allow control over the number of crossings in random loops. When $p = 1$, all loops will be circles of the four-orientation grid. As p decreases, the loops become much longer with only a few crossings. As p further decreases, the trend changes, loops become shorter as there are fewer turns. When $p = 0$, every tile will be a crossing, making an array of only straight lines of the array length (see Figure 3.13(d)).

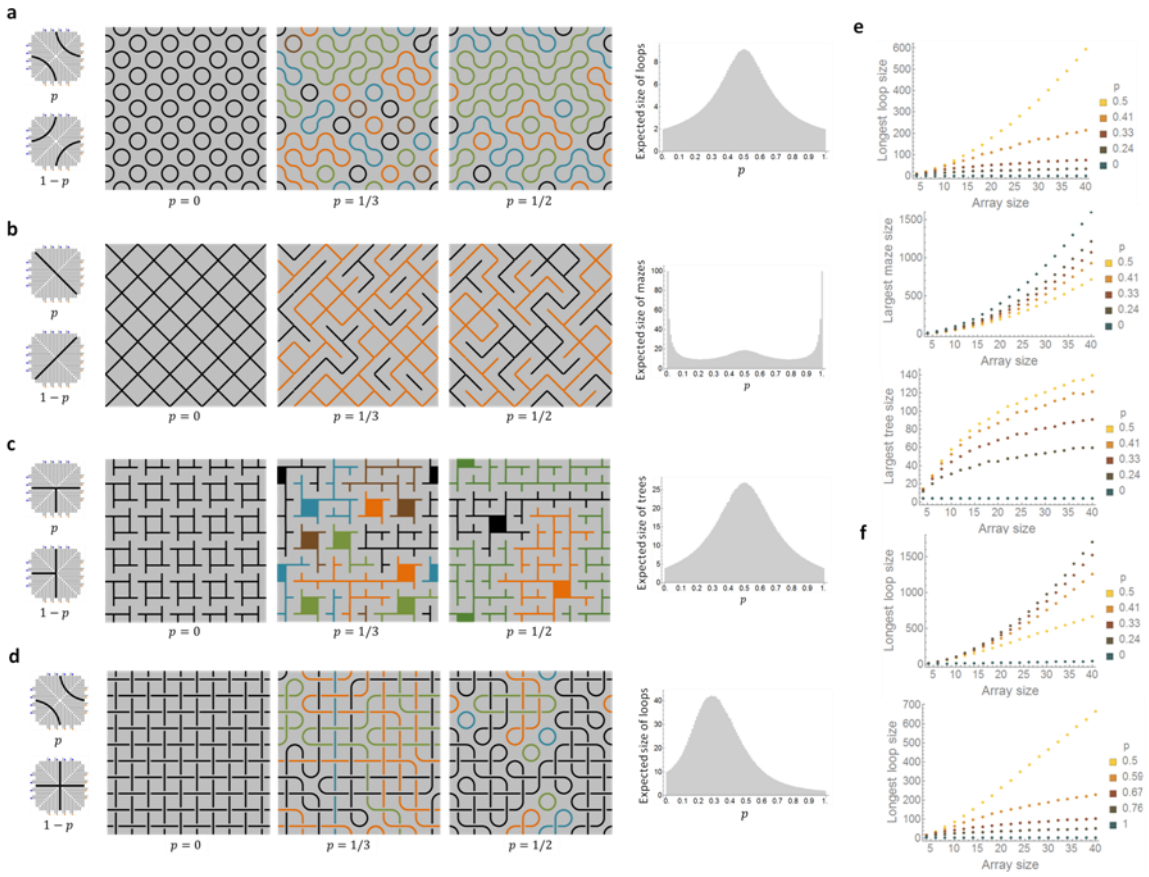


Figure 3.13. **Programming the random choice to control the crossings and size of loops, mazes, and trees. a, b,** and **c,** Random loops mazes and trees with size controlled by the probabilities of two tiles with different pattern orientations. The numerical simulations are formed from ten thousand independent trials for each p from 0 to 1 with 0.01 increments. **d**, Random loops with crossings controlled by the probability ratio of two tiles with distinct patterns. **e**, Average sizes of longest loop, largest maze and tree on array sizes of 4 by 4 to 40 by 40 with p values of 0, 0.24, 0.33, 0.41, and 0.5. Numerical simulations used one thousand independent trials for each array size and each p value. **f**, Average size of the longest loop on random arrays from size 4 by 4 to size 40 by 40 for p values of 0, 0.24, 0.33, 0.41, 0.5, 0.59, 0.67, 0.76, and 1. Numerical simulations used one thousand independent trials for each array size and each p value.

We implement experimentally the random trees (Fig. 3.14) and random loops with arc and cross tiles all mixed at ratios of p or 1-p respectively. The value of p is set to three probabilities: 0, 1/3, and 1/2. When p = 0, all trees are square roots of size 4 as expected. At p = 1/3 or 1/2, larger trees formed at the cost of fewer trees per array. The average tree size was 17.6 ± 5.8 at p = 1/3 compared to its expected size of 16.7. The average tree size was 25.9 ± 8.0 at p = 1/2 compared to its expected size of 26.7. The experiments thus showed the largest tree sizes at p = 1/2 as expected with simulations. For the loops, at p = 0, all

loops were straight lines as expected since every tile was a cross. When the value of p was set to 1/2 and 1/3, it was found that longer and fewer loops existed at p = 1/3. This again agreed with simulations and expectations. The number of crossings was 70.3 ± 7.4 when p = 1/3 with an expected value of 67. The number of crossings was 48.7 ± 4.4 when p = 1/2 with the expected value of 50. This shows the tuning of tile ratios is properly reflected in the experimental arrays. In principle, even further tile patterns could be introduced to form a wider range of tunable pattern properties.

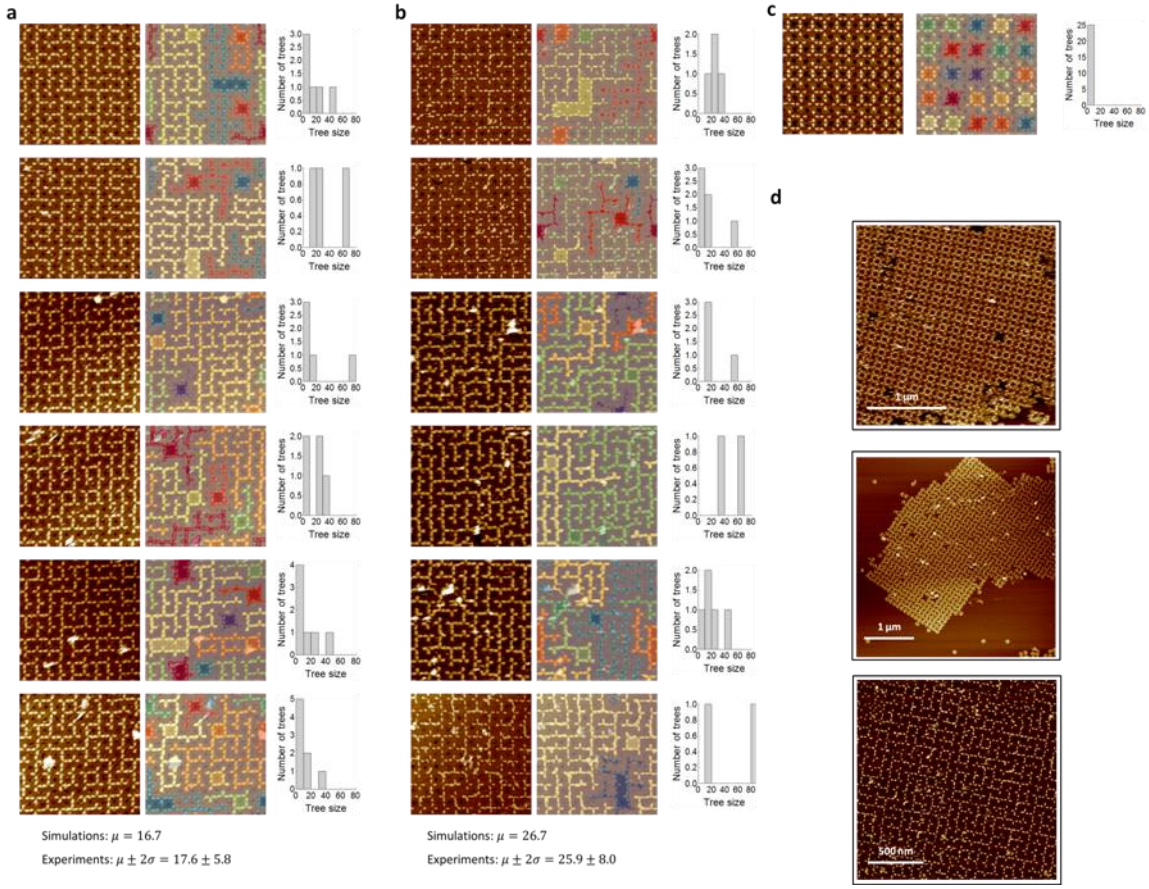


Figure 3.14. **Analysis of T90 trees on 10 by 10 tile areas. a**, p = 1/3. Analysis includes every tree in the tile area. **b**, p = 1/2. **c**, p = 0. Only one tile area is shown as any other accurate tile area would appear identical. **d**, Zoomed out AFM images of tile arrays before cropping out a random 10 by 10 tile area. From top to bottom, p = 0, 1/3, and 1/2 respectively. μ is the average size of all T90 trees, σ is the standard error of the mean, and ±2σ corresponds to 95% confidence.

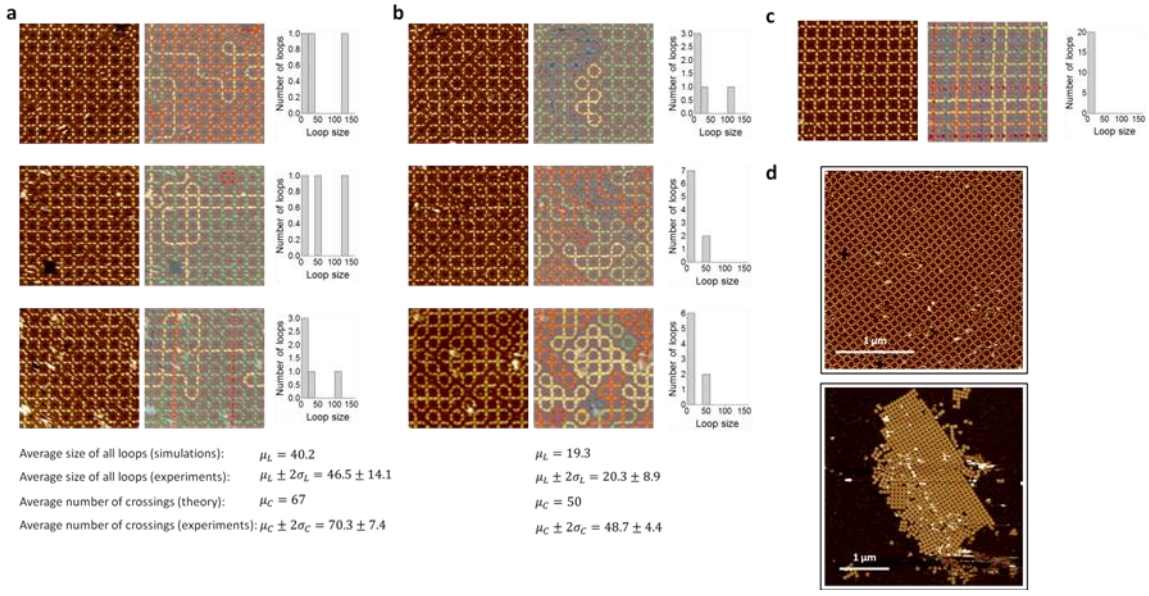


Figure 3.15. **Analysis of loops with crosses on 10 by 10 tile area. a**, p = 1/3. Analysis includes every loop in the tile area. **b**, p = 1/2. **c**, p = 0. Only one tile area is shown as any other accurate tile area would appear identical. **d**, Zoomed out AFM images of tile arrays before cropping out a random 10 by 10 tile area. From top to bottom, p = 0 and 1/3 respectively. μ is the mean, σ is the standard error of the mean, and ±2σ corresponds to 95% confidence.

3.5 Design of 2D DNA Origami arrays with designed sizes

All prior sections involved a single edge design set on multiple tiles using different surface modifications to form the complex patterns with the desired global properties on unbounded arrays. This section looks to a new approach for controlling complexity in global patterns by instead controlling the size of the grid through finite DNA origami arrays. The work in this section motivates the work done in Chapter 4. Chapter 4 explores forming fully addressable finite arrays through a different technique that is much more scalable at the cost of more mixing stages.

When the tiles of an array are all mixed together, a finite array requires multiple types of tiles with distinct edge designs. Each tile may have a specific pattern that is unique to a particular location in the array providing significant additional control of the array's global properties. At this point, we explored two designs for finite array self-assembly. One is meant to encourage assemblies over incomplete arrays. The other allows asymptotically

fewer numbers of distinct edges. The first design is a fully connected design whereas the second design is similar to a comb structure. While the comb structure saves edges by replacing several edges with weak stabilizing interactions, each tile attaching to the array attaches by only one sides during self-assembly. Completing assemblies in not preferring over incomplete ones for the comb design (see Figure 3.16). However, for fully connected structures, the optimal tile connections are made when attaching to complete an assembly versus only making one connection in an incomplete structure. Due to the large number of orthogonal edge sets needed, all arrays were given four-fold rotational symmetry to minimize the number of unique edge sets needed. Still, a fully connected design requires n(n-1)/2 distinct pairs of edges, always more than a comb design.

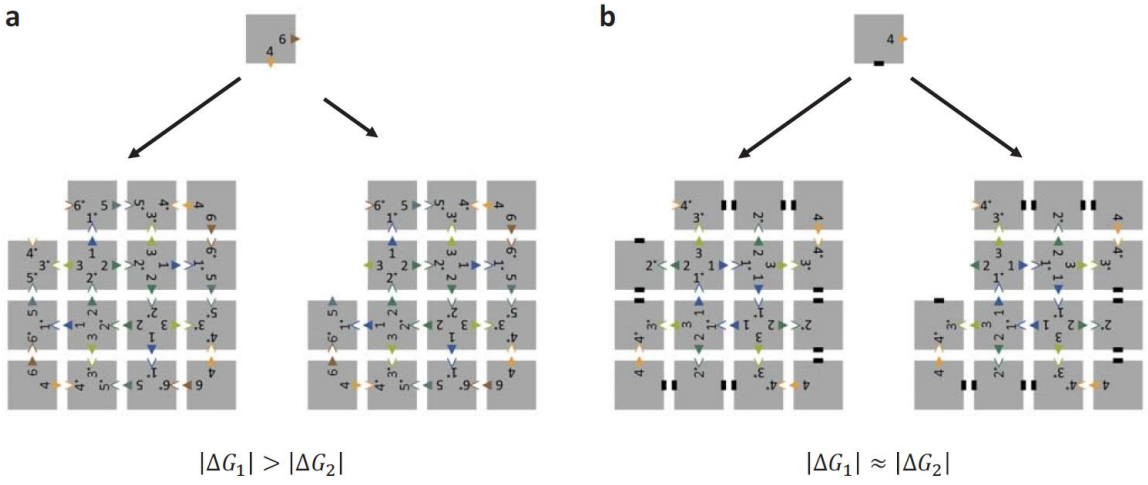


Figure 3.16. **Differences in completing arrays for two finite array designs. a**, fully connected design favoring completing assemblies over incomplete. Filled arrows and indented arrows of the same color are matching edges. **b**, comb design. Double black bars are weak stabilizing edges with only stacking bonds.

Four criteria dictate edge design. First, spurious edge interactions between different copies of the same tile type are to be minimized. Edges with a sticky edge overhang are called giving edges, and edges with a truncation are called receiving edges. While a giving edge maximizes binding energy when it binds to its complementary receiving edge, spurious interactions may occur. The worst is commonly between a giving edge and a non-complementary receiving edge as some sequence similarity of sticky ends is likely. Hence,

we chose individual tiles with either all giving or all receiving edges when possible. Thus, when the tiles are initially formed as monomers, they are less likely to form aggregates.

Second, specific edge interactions are given different binding energies for a design principle we call one-pot staged self-assembly which contributed to the high yield of arrays. By varying the binding energy of specific edge interactions, we can promote sequential stages of self-assembly during the annealing of all tiles mixed together. In this manner, the complex task of all tiles finding their proper matches is broken into multiple stages, reducing the risk of potential spurious interactions.

Third, tile orientations in the final finite array are balanced as much as possible in 2 by 2 tile areas. As with the corrugated method used with the unbounded arrays to form flat arrays despite surface modifications, multiple 90 degree rotations of tiles within the finite array was used.

Fourth, we keep the number of edge codes as low as possible. Edge codes refer to a specific layout of edge staples with a sticky end length (Fig. 3.17). Each edge code has variable sticky end length and layout. Since there is variation in M13 scaffold at different locations, the sticky end sequences to each of the four sides of the square tile are different. Hence, each edge code can provide to four pairs of distinct edge interactions. This design criteria conflicts with the third design criteria to some extent and both are balanced against each other. Accordingly, neither design criteria is fully satisfied completely. Fully giving tiles can also be rotated in their array location, since the sticky end sequences can be extended off of any side of the square. Since receiving edges are dictated by the M13 sequence, these tiles cannot be rotated without considering the edge pairing. We also do not take into account the M13 sequence causing different amount of stacking bond energies depending on which edges interact. It would be possible to consider this, however, it would provide even greater constrictions on the design space.

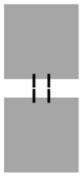
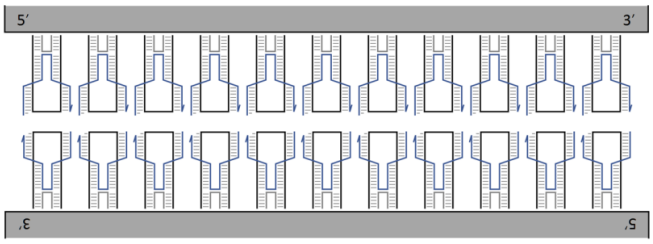
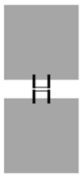
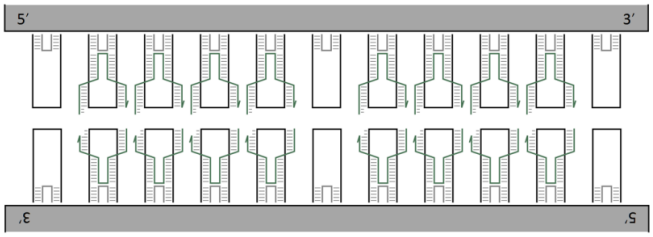
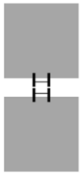
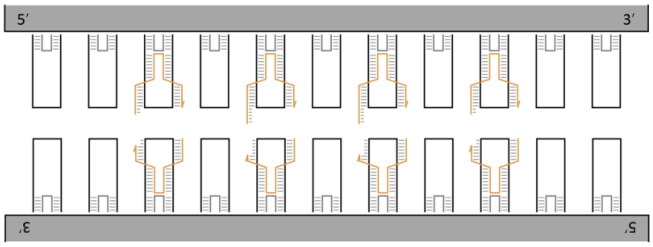
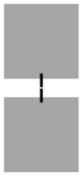
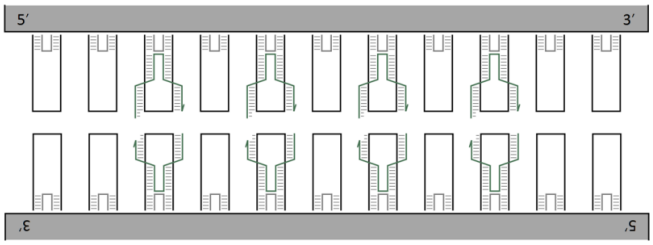
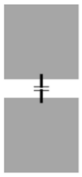
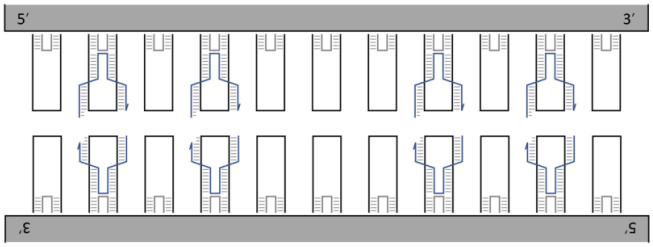
Code #	Abstraction	Edge design	Total stacking bonds
1			33
2			32
3			22
4			16
5			16

Figure 3.17. **Five types of edge codes used in finite arrays.** All staples in an edge code have a specific sticky end length on the 5' end of the staple and truncation on the 3' end. Each edge code may be used at four different locations on a tile due to the M13 sequence variance around an origami tile. Total stacking bonds roughly approximates overall binding energy between the edges and is the sum of all stacking bonds formed from all staples interacting between the edges.

Using these design criteria we design 3 by 3, 4 by 4, and 5 by 5 finite arrays for the fully connected design (Fig. 3.18) and the comb design (Fig. 3.19). Before mixing individual tiles together, we added a set of negation strands with complementary sequences to the edge staples to each tile to prevent any edge staples incorporating into locations on other tiles.

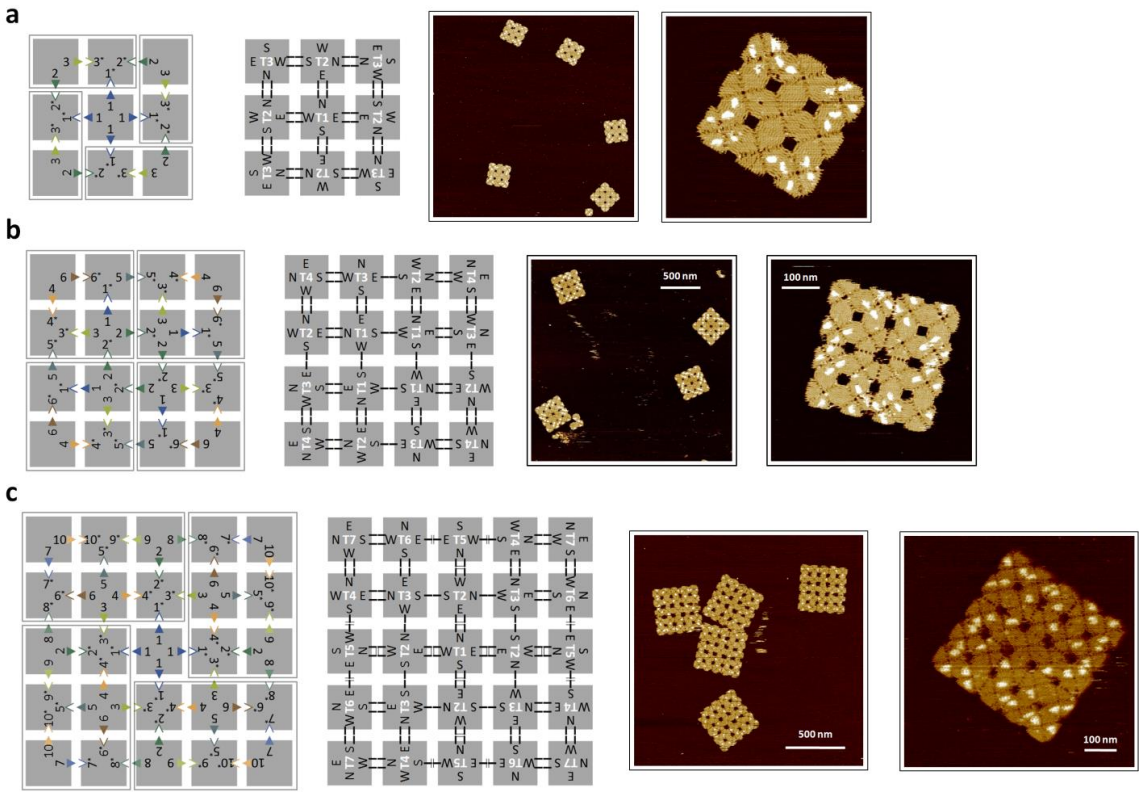


Figure 3.18. **Fully connected finite arrays of square DNA origami tiles. a**, 3 by 3 array. From left to right: abstract design diagram with distinct edge interactions labeled 1/1* to 3/3* where giving and receiving edges are represented by solid triangles and indented triangles respectively. Abstract design diagram with edge code from figure 3.17, and the distinct types of tiles labeled with N, E, S, and W representing unique orientation of each tile. AFM images of a selected zoomed out and zoomed in view. Patterns on arrays are for ensuring complete structures have correctly integrated tiles at each location. **b**, 4 by 4 array. **c**, 5 by 5 array.

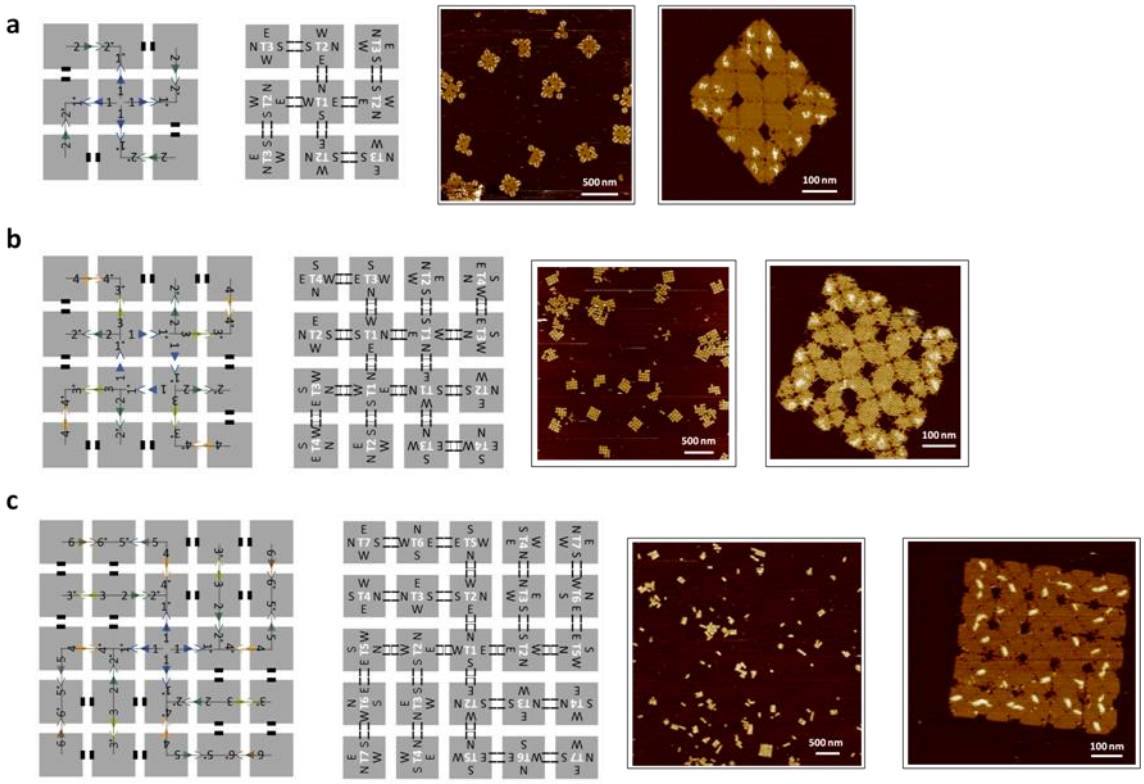


Figure 3.19. **Comb design for finite arrays of square DNA origami tiles**. a, 3 by 3 array. From left to right: abstract design diagram with distinct edge interactions labeled 1/1* to 2/2* where giving and receiving edges are represented by solid triangles and indented triangles respectively, and weak non-specific supporting edges are double black bars. Abstract design diagram with edge code from figure 3.17, and the distinct types of tiles labeled with N, E, S, and W representing unique orientation of each tile. AFM images of a selected zoomed out and zoomed in view. Patterns on arrays are for ensuring complete structures have correctly integrated tiles at each location. **b**, 4 by 4 array. **c**, 5 by 5 array.

Yield was calculated from AFM images with the aid of a software tool to mitigate the risk of overestimating actual yield [16]. If using selected AFM images of around 2 by 2 microns as with other works in the literature, we could find yields of 3 by 3, 4 by 4, and 5 by 5 arrays close to 100%. Instead, we desired to avoid selecting optimal small AFM images for yield calculations, aiming to calculate yield on very large areas of 30 by 30 microns to minimize bias. Yield was calculated as the total pixels in mostly isolated complete assemblies of the designed size in the AFM image divided by the total pixels above a threshold of background. With a large sample size of several thousand to tens of thousands of origami tiles per AFM image, the yield of fully connected 3 by 3, 4 by 4, and 5 by 5

arrays was 15.6%, 15,0%, and 32.4% respectively. The yield of the comb design only gave 8.0%, 6.7%, and 1.3%. This suggests that the fully connected design that favored complete assemblies is crucial for higher yields.

3.6 Combinatorial diversity and programmable properties in finite arrays

This section covers the application of combinatorial rules to control fundamentally new properties enabled by the multiple tile types in finite arrays. Since a tile's edge design dictates its location in the finite array, it becomes possible to control specific pattern configurations at specific locations that differ from neighboring tiles.

For our experiments, we took our high yield 5 by 5 fully connected arrays. These arrays have four-fold symmetry and are made from seven unique tiles in terms of edge design. By using tiles with a double arc design, we made arc mazes of 440 by 440 nm in size. To obtain random mazes, we applied the first rule from section 3.2 for programming the tile. For each of the tile types per distinct edge designs, we included two different tiles with different surface patterns. As with the unbounded arrays forming loops, the two designs were mirror images of double arcs (Fig. 3.20). Theoretically, looking at the mazes from a fixed view point, there exists over thirty million (225 = 33,554,432) distinct mazes that may form in a single test tube. We found the average number of circles to be 1.0 ± 0.6, ageing with the expect value of $(m - 1) \times (n - 1) \times (1/2)^4$ in random m by n arrays of arc tiles [17].

We next demonstrated not only the fixed properties emerging from having a finite array, but the control of pattern configurations at specific locations in the finite array. We created random mazes with a designed entrance and exit (Fig. 3.20). Here, we fixed the exterior tiles in each 5 by 5 array in a determined orientation. The interior tiles remained a $p = 0.5$ random choice between two mirror image arc patterns. Considering rotational symmetry, there are $2^8 = 256$ distinct mazes formed, all with a path from the entrance to the exit. All 5 by 5 arrays extracted from AFM showed this property.

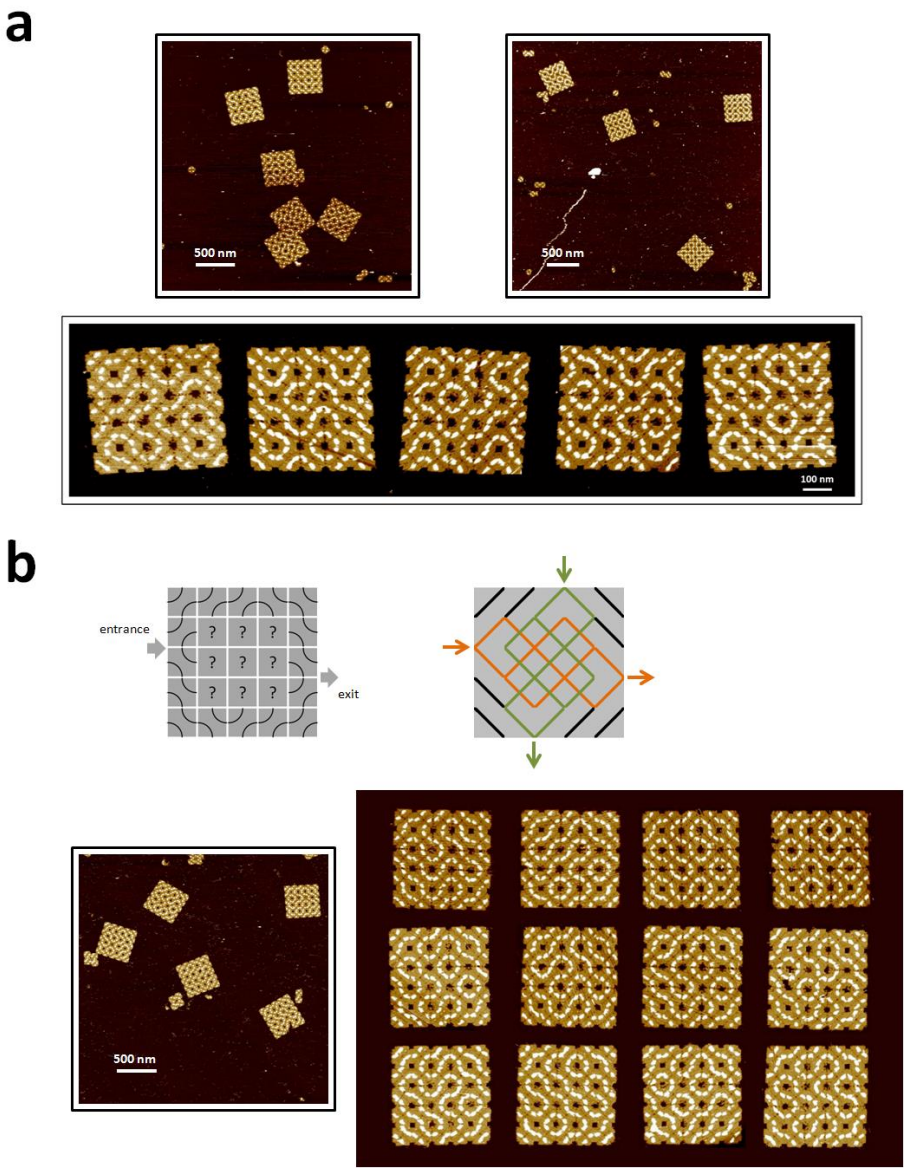


Figure 3.20. **Random loops and mazes on the entire or part of a 5 by 5 finite array. a**, Random loops and mazes on entire 5 by 5 arrays. The bottom image is composed from 5 separate zoomed in AFM images. **b**, Designed mazes with fixed orientation of exterior tiles and random orientation of interior tiles. The right image is composed of 12 separate zoomed in AFM images.

Here we only varied the tile type, taking advantage of the finite nature of the array. The random choice of the tiles may also be programmed for further control. In principle, these

rules can further be applied to other more complex grids, such as the arrays and nanostructures introduced in the following chapters. One can image the pattern being replaced with functional devices: maze lines being replaced with nanoparticles to form waveguides, cross-like patterns being nanotubes forming transistors, or even each tile just holding a random library of potential functional nucleic acids taking SELEX to 2D.

Chapter 4

FRACTAL ASSEMBLY OF MICRON-SCALE DNA ORIGAMI ARRAYS WITH ARBITRARY PATTERNS

The previous chapter focused on controlling global properties and generating tunable diversity of patterns in arrays of DNA origami. Here, we demonstrate a set of hierarchical assembly rules to construct a single target pattern on a micron-scale finite array.

Since the creation of the first DNA origami, there has been interest in scaling up the size of two dimensional DNA origami structures (see Fig 4.0). Marchi et al. [1] took the direct approach to increasing the scale of a DNA origami structure by taking the same principles as the first DNA origami but applied to a significantly longer scaffold strand. With a single scaffold strand for the staples to attach onto, the author's overall DNA structure maintains the benefit that it can still be made in a single mixing step. However, as the number of unique locations to be held together increases due to a longer scaffold, so does the number of unique staple strands needed to form the complete DNA structure. More unique strands means a higher upfront cost and issues with spurious interactions among the distinct strands. Raiendran et al [2] implemented a scaled DNA structure requiring fewer additional unique staples through the use of select recycled staples. The authors split the mixing stages into two steps: in the first, nine individual DNA origami tiles were annealed reusing many staple sequences between the separate tile's interiors; in the second, the nine DNA origami tiles were mixed together, and due to their unique edges, formed a 3 by 3 array. Accordingly, a DNA structure nine times the size of a single DNA origami was formed with only edge staples scaling with the size of the complete structure.

The question arises ─ does the number of unique staples necessarily have to scale with the size of the structure? In Demaine et al. [3], the authors showed that, in theory, it is possible to form arbitrary shapes with only O(1) glues with no scaling of unique staple numbers needed if applied to DNA origami tiles. Experimentally, successful demonstrations have

been rarer. Still, by using multi-stage hierarchical methods, Park et al. [4] made a 4 by 4 array of small DNA tiles recycling edges and interior strands. There has been no demonstration of a hierarchical multi-stage approach with DNA origami.

While prior scaled two dimensional DNA origami structures increased the surface area, they all result in an increase in unique staples required. Furthermore, to reach the scale necessary to interact with the low cost fabrication technique of photolithography [5], an even larger structure is necessary than those structures create thus far. We designed a set of rules called Fractal Assembly (as the technique builds complex structures using only a set of simple rules applied recursively) and applied the rules to DNA origami tiles, demonstrating micron-scale finite DNA structures with arbitrary patterns.

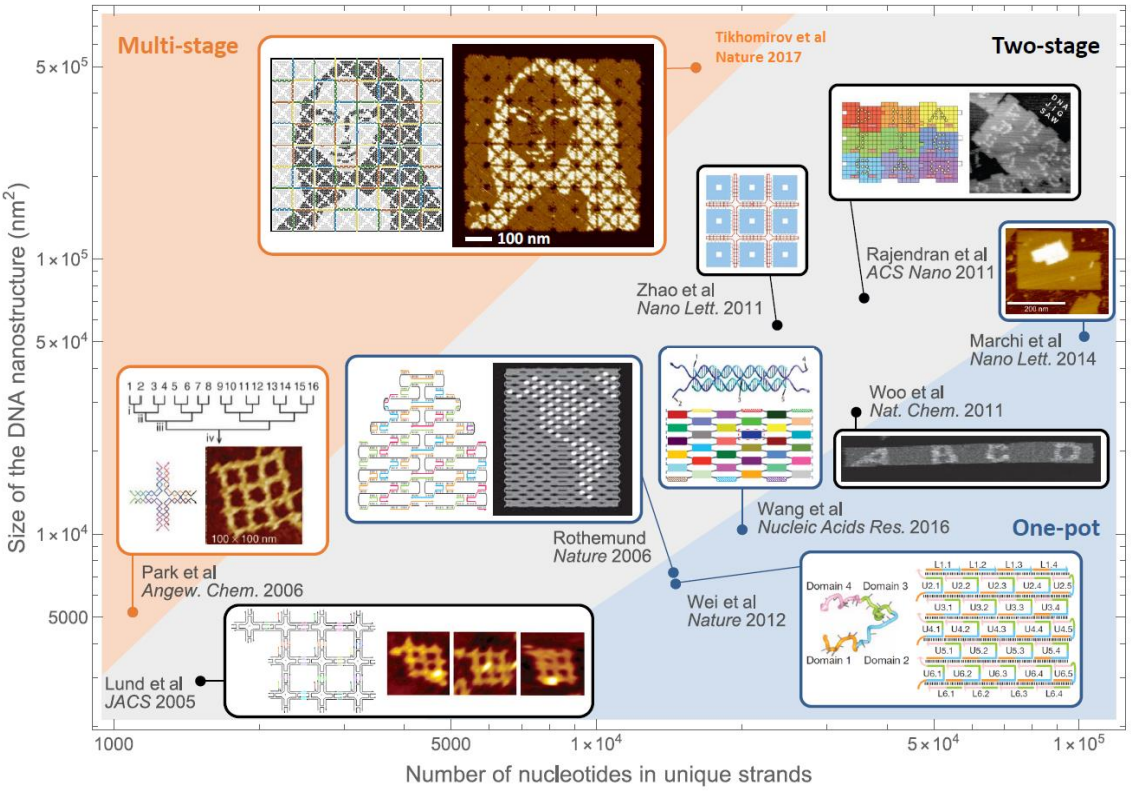


Figure 4.0: **The development of arbitrarily patternable two-dimensional DNA structures**. Log-log plot comparing the size of a DNA structure against the required number of nucleotides in unique strands in that structure.

4.1 Design considerations and rules of Fractal Assembly

In Chapter 3, section 5 we explored creating finite arrays. This section first reviews the limitations of our initial approach followed by the three simple rules of Fractal Assembly designed to address those limitations.

Our finite arrays from Chapter 3 were self-assembled in two stages where first the DNA origami tiles were annealed separately, and then the formed tiles were annealed together. We further introduced the technique we called one-pot staged self-assembly where we used two types of edge codes with strong and weak binding energies to "stage" self-assembly even in one-pot. For example, in our 4 by 4 array, a strong 1 nt sticky end 11 staple edge was used to form four 2 by 2 arrays. Next, a weaker 2 nt sticky end 4 staple edge was used to bring the 2 by 2 arrays together to form the 4 by 4 array. To lower the number of tiles and edges required (allowing only two edge codes to be sufficient), we introduced four-fold rotational symmetry (see top of Figure 4.1). For our higher yield fully connected arrays, the symmetry meant $n^2/4$ distinct types of tiles and $n(n-1)/2$ distinct types of edges were needed for n by n sized arrays. Unfortunately, the rotational symmetry removes unique addressability for the entire array as the same tiles integrate into multiple locations in the array.

In order to allow unique addressability on our arrays, we initially took the above design and removed the four-fold rotational symmetry asepct. To form the new array, we now needed four times the number of distinct types of tiles and edges as before. Since our square tile has four edges made unique by the differing M13 sequence along each edge, each edge code can provide only four pairs of distinct edges. The two edge codes initially used with four-fold symmetry would not provide enough edge pairs now. A total of 24 distinct pairs are required for unique addressability (Fig. 4.1 bottom)—16 to form the 2 by 2 arrays (previously only 4 were needed) and 8 to bring the 2 by 2 arrays together (previously only 2 were needed). Hence, 6 types of edge codes are needed as each code provides four pairs. Using palindromic edge codes as before, we made edges from four staples taken from an eight staple pool out of the eleven possible staple locations (011101110, where 1 represents

a location where a staple could be chosen). With these new 6 edge codes, we formed a uniquely-addressable 4 by 4 array (Fig. 4.1 bottom), although the yield was lower than the array with rotational symmetry. As there is an increase in the number of distinct tiles and edges, there is likely an increased amount of spurious interactions resulting in the reduced yield. Besides this problem, we already pushed the limits of our edge codes, limiting us from forming larger arrays with this method.

Transitioning into more stages for the self-assembly process is a potential solution for (i) reducing spurious interactions by reducing the number of possible reactions at each stage, and (ii) using fewer distinct edges by reusing the same edge interaction for tiles assembling in different test tubes during the same stage. Still, multi-stage self-assembly requires a edge design compatible with the multi-stage annealing protocol. The temperature at any stage must be high enough to melt the spurious interactions in the current stage but not high enough to melt the previous stage's target structures. We took the above uniquely-addressable 4 by 4 array and split its formation into three stages: (1) we annealed the tiles in separate test tubes from 90 to 20 °C, (2) we annealed four 2 by 2 arrays in separate test tubes from 50 to 20 °C, and (3) we annealed the four 2 by 2 arrays together from 30 to 20 °C. The last stage required a very low temperature as the tile edges used four 2 nt sticky end staples which have a melting point close to 35 °C (see Chapter 3, section 2). The low temperature likely was not high enough to melt the spurious interactions, and we obtained poor yield.

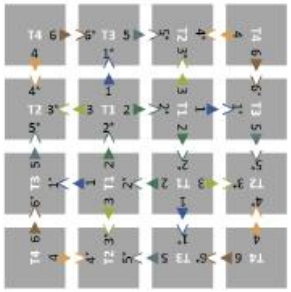
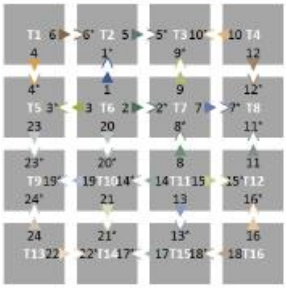
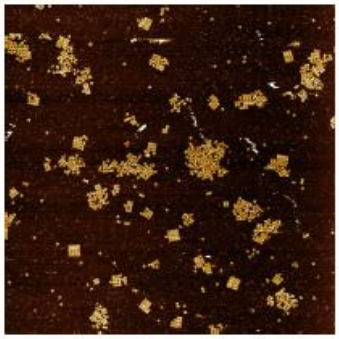
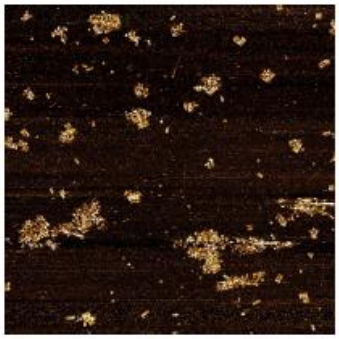
	Design diagram	Annealing protocol	AFM image	Yield
4 by 4 array with a four-fold rotational symmetry Tikomirov et. al. *Nat. Nanotechnol.* 2016		two-stage		15%
4 by 4 array with unique addressability		two-stage		4.7%
		three-stage		0%

Figure 4.1. **Chapter 3, Section 5 finite arrays and failed attempts to expand the approach to uniquely-addressable DNA origami arrays.** AFM images are 10 by 10 microns.

There are three unresolved issues that arise from creating a multi-stage self-assembly strategy in practice that we can see from these failed attempts. (i) Designing larger DNA origami arrays should not rely on increasing numbers of distinct edge strands as the sheer quantity would inevitably limit the size of the structures, (ii) the multi-stage tile design must be compatible with the multi-stage annealing process as mentioned above to break spurious interactions but not melt the prior stage's structures, and (iii) the self-assembly

process should be simple and accordingly the rules for designing arrays of different sizes should be as similar as possible. If the same rules apply across sizes, design principles and experimental conditions learned from smaller structures can be more readily applied to larger structures.

We introduce an assembly technique that is hierarchical and also self-similar that we call fractal assembly. The name fractal was chosen as our technique mimicked fractal formation, the formation of complex structures through a simple algorithm repeating at different scales [6]. The self-similar component critically allows the use of a constant number of unique strands to built structures of increasing sizes, the multi-stage annealing protocol that functions experimentally, and allows applying assembly principles learned from creating smaller structures to larger structures.

For fractal assembly, the formation of the DNA origami tiles individually is considered stage 0. Four square tiles are combined to form a square 2 by 2 array for stage 1 (Fig. 4.2(a)). For each subsequent stage, a larger square array is formed from four smaller square arrays formed in the prior stage. For an n by n array, one starts with n^2 test tubes of individual origami tiles and combines four into one at each stage until all molecules are combined into one last test tube after $\log_2 n$ stages (Fig. 4.2(b)). For a tile with x uniquely-addressable pixels, the final array will have $x \times n^2$ pixels, and each pixel will have a unique address based off the identity of an initial test tube (T_{ij}, $1 \leq i, j \leq n$) and the identity of the pixel on the DNA origami tile in that test tube (Fig. 4.2(c)).

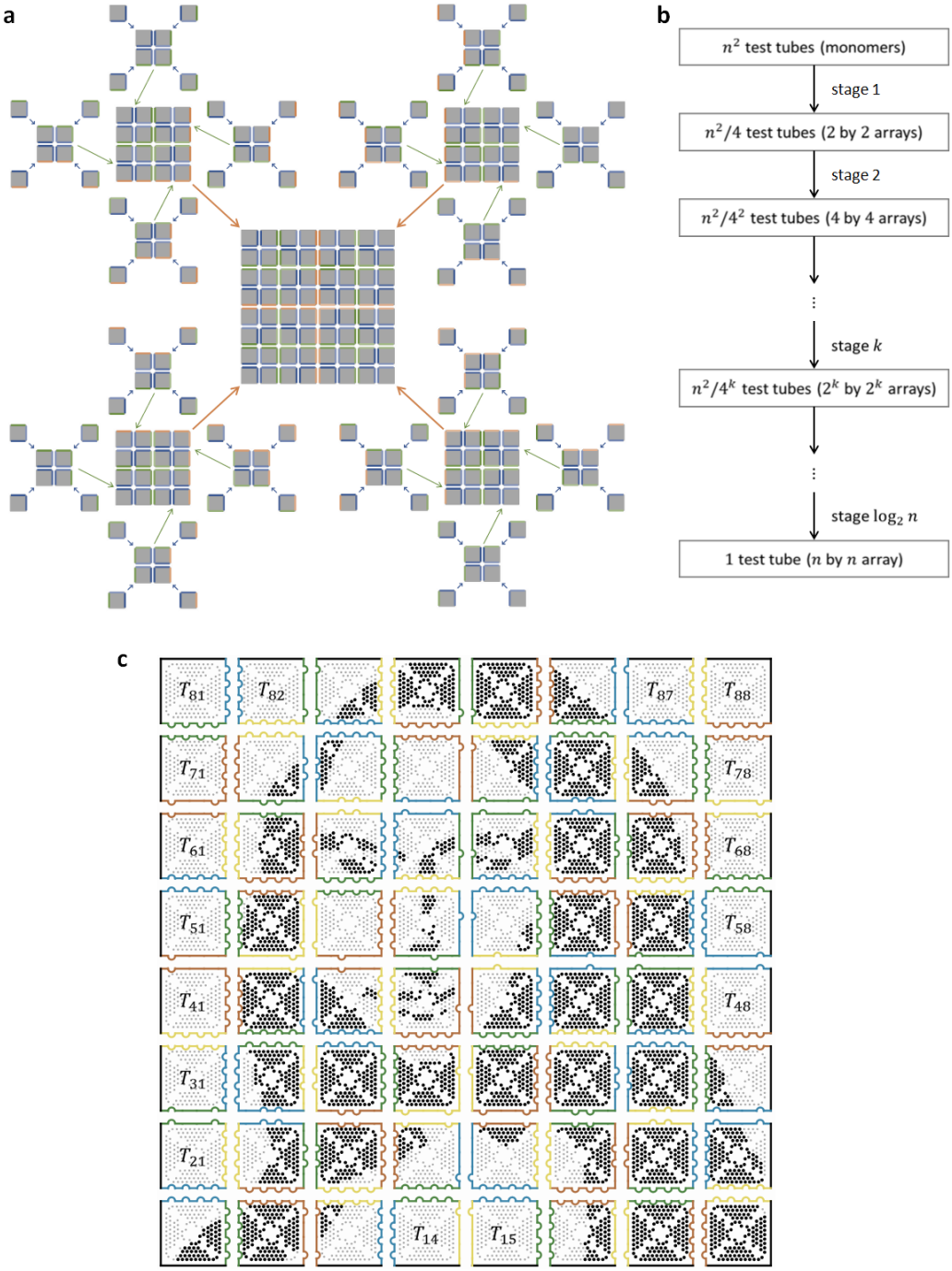


Figure 4.2: **Design of fractal assembly**. **a**, A three-stage self-assembly process forming an 8 by 8 array. Each distinct color dictates the tile-tile or array-array interactions in a particular stage. The shades of the same color are a particular edge interaction within the same stage. **b**, Forming n by n arrays in $\log_2$ n stages. **c**, 8 by 8 array with a Mona Lisa pattern. Indentations and bumps along edges show receiving and giving tile edges that consist of staples with nucleotide truncations and extensions, respectively. The four relative receiving edges of each tile are colored blue, green, orange, and yellow, as receiving edge identities are determined by the M13 sequence. Giving edges are colored based on their complementary receiving edges.

We developed three simple rules, recursively applied at all stages in fractal assembly. The first is the “giving and receiving rule,” the second is the “rotation rule,” and the last is the “edge code rule”.

4.1.a The giving/receiving rule

The first rule is the “giving and receiving rule”. In our edge design, desired interactions between DNA origami tiles are formed with nucleotide extensions "sticky ends" (called a giving edge) and truncations exposing the underlying M13 sequence (called a receiving edge). As the M13 scaffold sequence varies for the four edges of the square tile, we obtain four orthogonal receiving edges for any edge code — a particular sticky end length and staple layout along an edge. The rule for assigning giving and receiving edges (Fig. 4.3) is that for each of the four tiles or arrays, the two interactive edges — the participating edges in a particular stage — are either both giving or both receiving. The aim of this rule is to reduce self-aggregation. We previously noted in Chapter 3, section 5 that the spurious interactions between non-complementary giving and receiving edges are worse than those between giving or receiving edges alone.

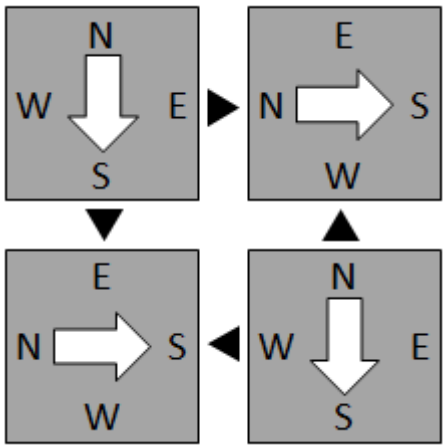


Figure 4.3. **The giving and receiving rule, and the rotation rule**. Each triangle points from a giving edge of a tile (or array) to a receiving edge of another tile (or array). North (N), east (E), south (S), and west (W) relative orientations of the DNA origami tile. Each white arrow points from the north edge to the south edge of a tile or an array.

4.1.b The rotation rule

The second is the “rotation rule”. In Chapter 3, section 2, we investigated the detrimental effect surface modifications such as staple extensions had on the curvature of origami tiles in the formation of two-dimensional arrays. To mitigate the curvature effects and form large arrays, we changed the orientation of tiles relative to their neighbors to inhibit the ability of the curvature to propagating globally.

In our final design, for each assembly of four tiles (or arrays), two tiles (or arrays) have one orientation, while the other two are 90 degrees rotated relative to the first pair (Fig. 4.3). If our prior results applied here, then should the DNA origami tiles have any internal curvature, it will be stopped from propagating globally in larger finite arrays. The specific orientations chosen also allow the four pairs of tile-tile or array-array interactions to be distinct from each other (E>N, N>W, W>S, and S>E). Note the receiving edges are spread once over N, S, E, and W, allowing a single edge code to be applied to all four edge pairs.

Still, we explored the effect of tile orientation in fractal assembly to determine if it was truly necessary for finite arrays formed using fractal assembly. When all tiles had the same orientation in the array, we observed aggregations and no target structures (Fig. 4.4(a)); however, when the tiles were rotated in a 90 degree fashion compared to their neighbors, we obtained a greatly increased amount of incomplete but correctly-formed structures (Fig. 4.4 (b)). The results imply the rotation rule is necessary to keep curvature from propagating in large finite arrays.

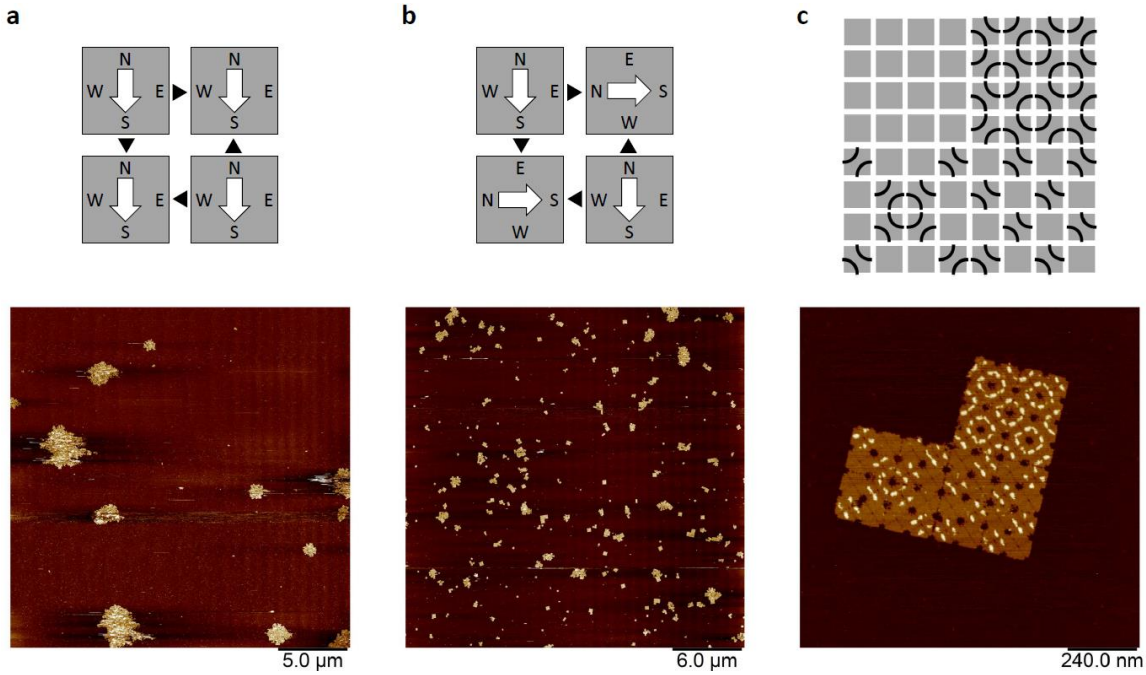


Figure 4.4. **Rotation rule's effect on formation**. **a**, top: design diagram, all tiles or arrays have the same relative orientation. Bottom: AFM image of 8 by 8 fractal assembly arrays. **b**, top: design diagram, each two tiles (or arrays) have the same orientation with the other two being rotated 90 degrees at each assembly stage. Bottom: AFM image of 8 by 8 fractal assembly arrays. **c**, top: design diagram of the 8 by 8 array from (b). Bottom: AFM image of 8 by 8 fractal assembly array from (b) missing one quadrant.

4.1.c The edge code rule

The last rule is the "edge code rule". As shown in Figure 4.5, each edge code consists of eleven 0s and 2s, defining one edge of a tile. Each 0 represents a scaffold loop whereas each 2 represents a staple with a 2 nt sticky end. Our staples each provide one stacking bond along with the short sticky end for specificity. Overall, these edges are meant to provide high enough specificity to drive desire interactions, yet weak enough binding energy to allow tiles to rearrange themselves to avoid kinetic traps during self-assembly. Palindromic codes are used to reduce asymmetric structural fluctuation along the edges of the DNA origami tiles and promote stability of the self-assembled structures. The first stage of a fractal assembly uses an edge code of eight 2s. In the subsequent stages, the size of the array doubles and so does the number of tile edges participating in the desired array interactions. As a solution to the annealing process challenge, we split each edge code from

the previous stage to generate two new edge codes each with half of the 2s from the previous code (Fig. 4.5). While the locations of edge staples will spread out more for each later stage, the total number of edge staples participating in the desired interactions between tiles (or arrays) at any stage is constant. Plus with increased spacings between individual hybridization locations, the total binding energy between tiles or arrays will decrease [7, 8]. Decreased binding energy results in decreased melting temperatures of self-assembled structures at each stage, making an annealing temperature possible for each stage high enough to break spurious interactions yet not high enough to melt self-assembled structures from the prior stage.

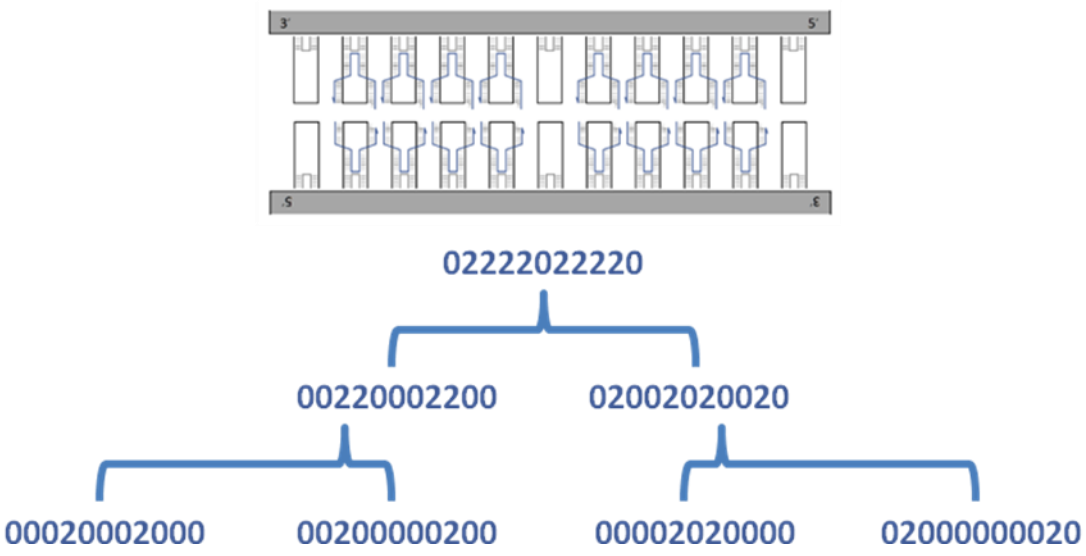


Figure 4.5. **Edge rule**. The edge rule for splitting staples in each stage. 0s represent the absence of a staple (open M13). 2s represent staple that is either truncated or extended by two nucleotides. Edges are split evenly in each stage and are palindromic. Each 2 in the second to fifth digits in each code corresponds to an indentation or bump on a tile edge shown in 4.2(c).

As noted in Figure 4.4(c), we had formed 8 by 8 arrays with one 4 by 4 quadrant missing. The annealing schedule was 55, 45, and 35 ºC to 20 ºC for the three stages respectively. The melting temperature of a 2 by 2 array was measured at 52 ºC (Fig. 4.6), allowing 55 ºC to melt spurious interactions yet not be higher than the melting temperature of average origami structures as monomers [9].

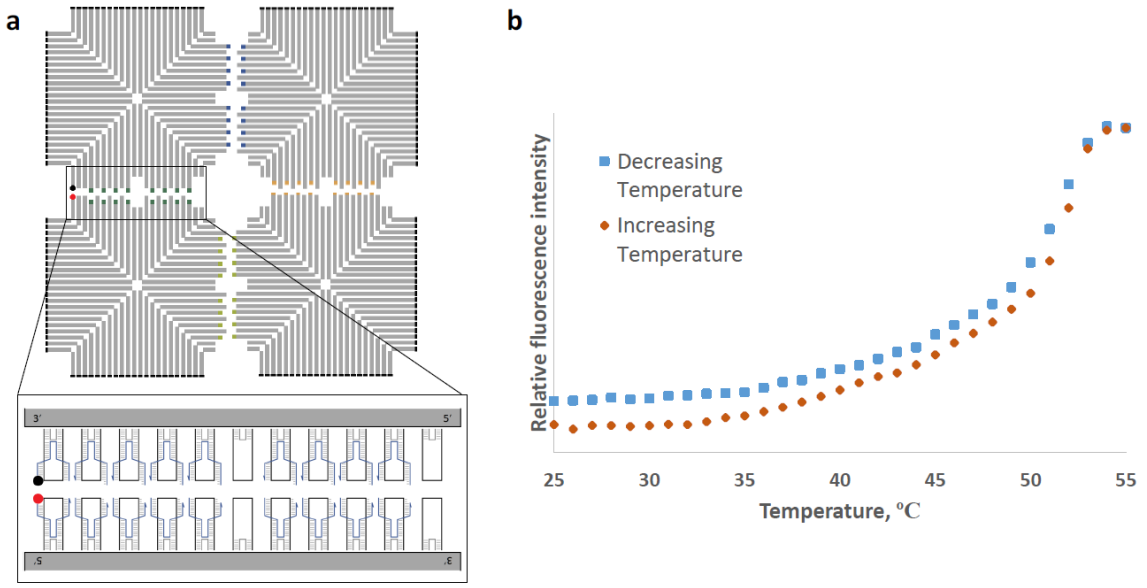


Figure 4.6. **Fluorescence experiments for melting temperature measurement**. **a**, Tile abstraction and edge design for a 2 by 2 array. The red dot (ROX fluorophore) is on a 5' end of an edge staple and is paired with the black dot (Iowa Black RQ quencher) on the 3' end of an edge staple on another tile. When the 2 by 2 array formed, the fluorophore is quenched causing low fluorescence. **b**, Melting curve for heating and cooling of 2 by 2 arrays. Measured on a Mx3005P QPCR system (Agilent Technologies). Sample is heated from 25 to 45 ºC at 5 sec/0.1 ºC, from 45 to 55 ºC at 30 sec/0.1 ºC, held at 55 ºC for 30 sec, cooled from 55 to 45 ºC at 30 sec/0.1 ºC, and finally cooled down 45 to 25 ºC at 5 sec/0.1 ºC. Each data point is an average of all data points at the same degree. Note, the fluorophore/quencher pair adds two stacking bonds to an edge in the 2 by 2 array, likely increasing the melting temperature slightly.

We decided to explore if increasing the annealing temperature would melt more spurious interactions and promote the formation of the missing quadrant (Fig. 4.7). However, the increased temperature of the anneal resulted in final structures that were scrambled. This result indicates that the temperature passed the melting point of the 4 by 4, arrays leading them to melt into 2 by 2 arrays recombining randomly. We continued with the 55, 45, and 35 ºC annealing schedule for further experiments.

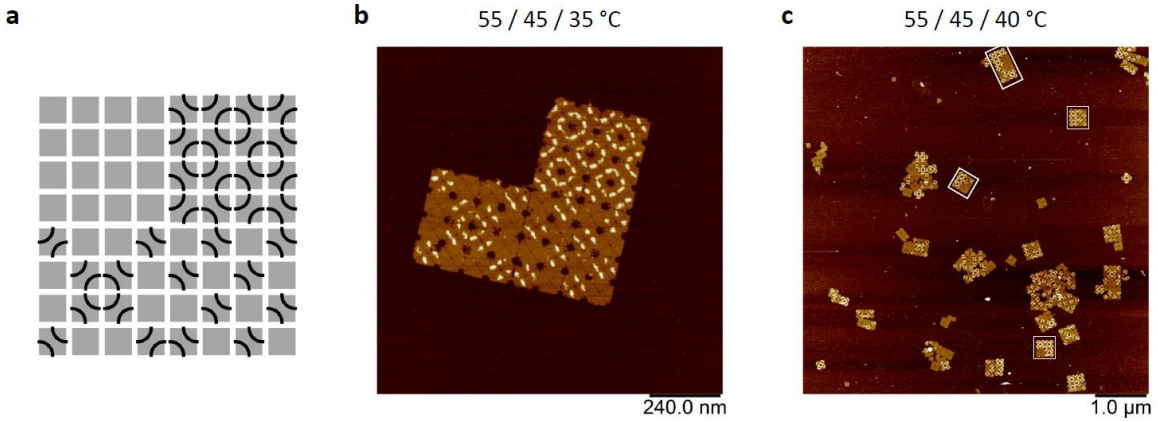


Figure 4.7. **Annealing temperature. a,** Design diagram of an 8 by 8 fractal array. Each quadrant has a unique pattern. **b**, AFM image of an incomplete 8 by 8 array, but with correctly formed quadrants. The annealing schedule was 55, 45, and 35 ºC to 20 ºC for each stage respectively. **c**, AFM image of scrambled and incomplete arrays. The last stage anneal was raised to 40 to 20 ºC. A few scrambled structures are outlined.

It would be in principle possible to reduce the gap between the melting temperatures for the purposes of forming larger arrays. The edges would have to be redesigned to increase their melting point, possibly by adding more edge staples at each further stage.

4.2 Demonstration of Fractal Assembly with DNA Origami tiles

This section describes our structural modifications leading to the implementation of the Fractal Assembly rules on our square DNA origami tile. Using Fractal Assembly, we assembled and demonstrated the arbitrary patterning of two-dimensional arrays up to sixty-four DNA origami tiles, reaching the micron-scale.

With the design rules from the prior section, in principle, DNA origami arrays of any size using a constant set of unique strands becomes possible. After all, a larger array would just use more stages and more tiles that make use of a subset of the edge staples compared to tiles in a smaller array. At any point, desired interactions between tiles or arrays have at least twice as many edge staples involved as competing interactions (Fig. 4.8). This is important to prevent reused edge staples from inhibiting the self-assembly of the desired structures.

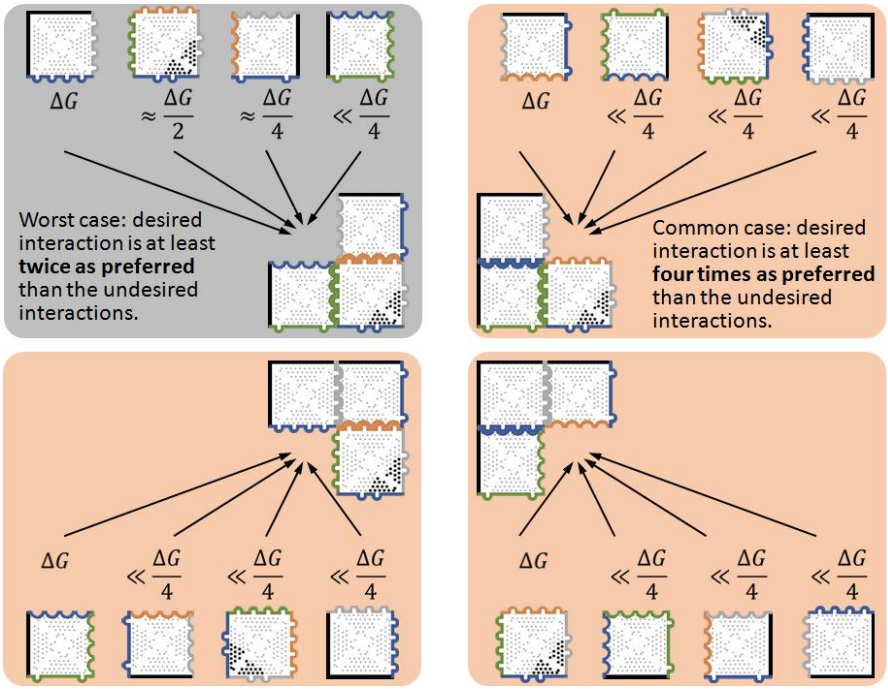


Figure 4.8: **Desired and undesired interaction preference.** The worst case maintains a desired reaction at least twice as preferred as undesired. Normally, the desired is at least four times more preferred. In subsequent stages, this gap between desire and undesired generally increases due to geometric spacing.

The formation yield of one four by four quadrant was still relatively poor, and appeared to have a deformation on AFM imaging (Fig. 4.9(b)). We hypothesized that the poor yield was due to the deformation of tiles caused by imbalanced edges. Tiles near the exterior of the array had inert edges formed from a full set of eleven double-hairpin staples yet some of those tiles also had an interior edge with just four staples. If the adjacent scaffold loops provided an unbalanced level of strain, the difference in the number of edge staples could have caused the deformation. We decided to reduce the number of double-hairpin staples from all eleven locations to just five inert staples evenly spaced along the edge. The change proved to be a success when tested on 4 by 4 arrays, improving the yield of the 4 by 4 arrays from 2.62% to 10.21% (Fig. 4.9 (b and c)).

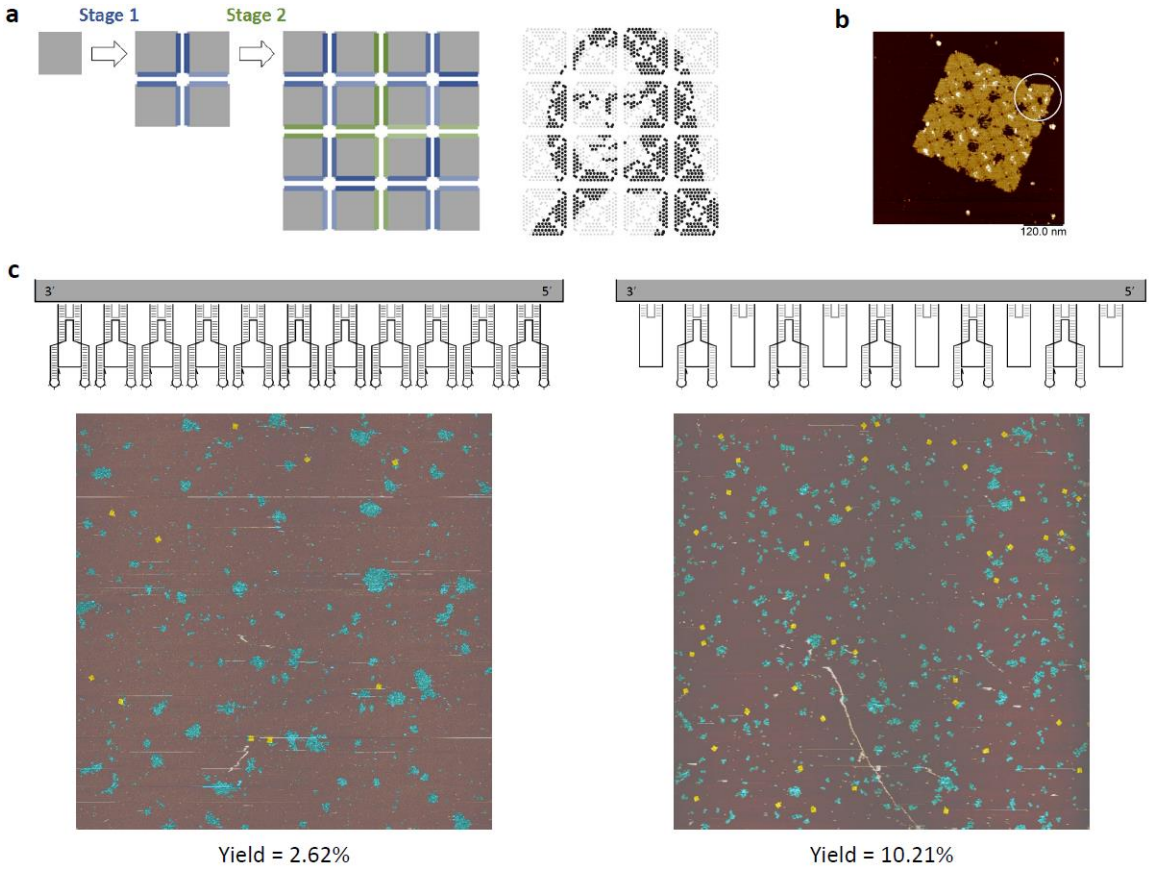


Figure 4.9. **Inert edges. a**, Design diagram for (c). **b**, four by four array from a low yield quadrant showing a deformation. **c**, Edge diagram and AFM images for eleven and five double-hairpin edge staples per edge at the exterior of the array. Correct structures are colored yellow, with the rest colored blue.

We suspected another reason holding back the yield was pipetting accuracy. To address this, we modified our protocol to increase pipetting volumes to better ensure each tile or array in a mix had a more accurate concentration. With the adjusted protocol, our yield increased from 10.21% to 16.11% (Fig. 4.10(a)). At the same time, we were redesigning the bridge staples and interior staples near the seams in our DNA origami design. The new design was meant to supply more pixel points for even greater continuous surface addressability. Perhaps due to break points adding flexibility, the new design fortunately led to an even higher yield of 24.99%.

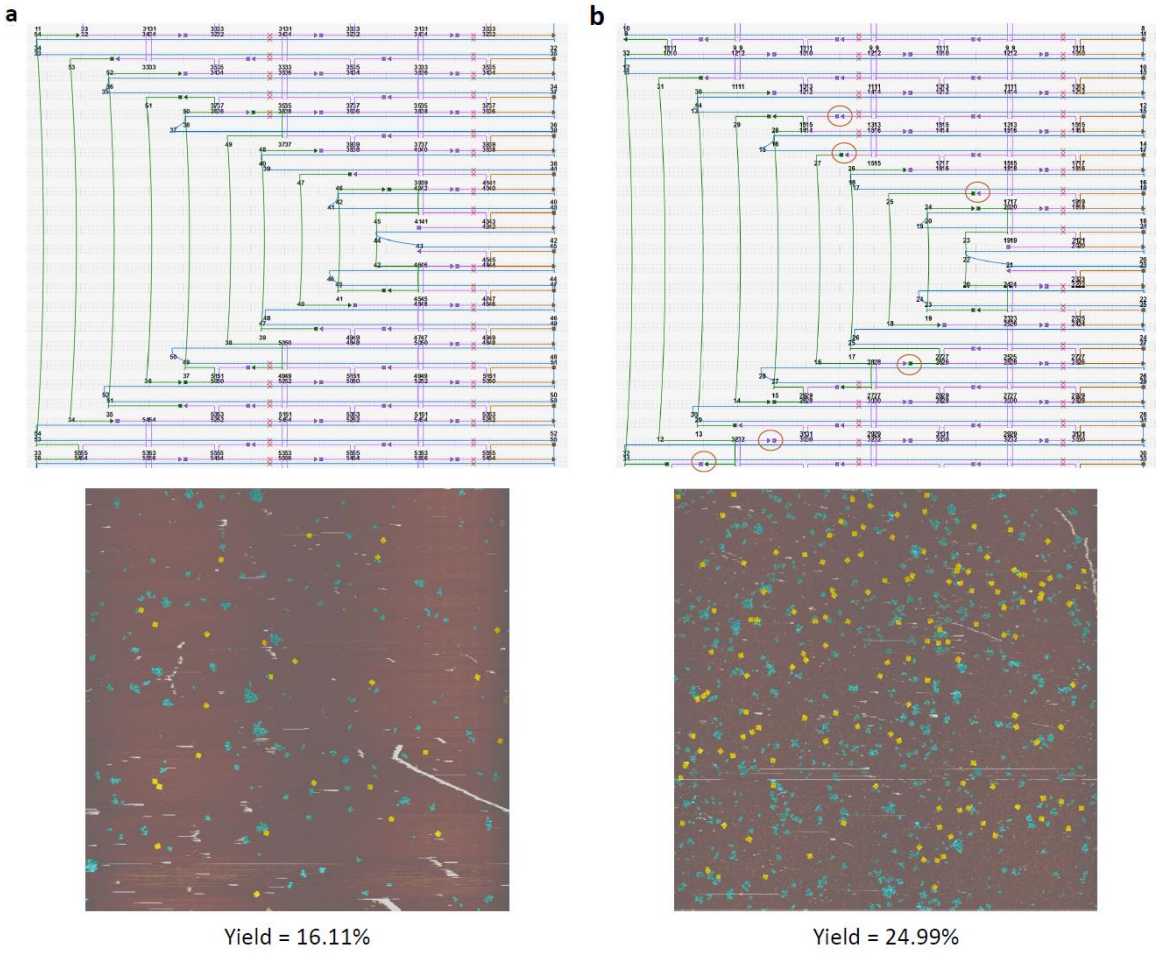


Figure 4.10. **Bridge and interior staple redesign for increased continuous addressable surface area. a**, Original Cadnano [10] diagram with 112 pixels and a formed 4 by 4 array (same design as in 4.9 (c and d)). **b**, New Cadnano design with 136 pixels. Correct structures are colored yellow, with the rest colored blue.

Finally, we moved to test the fractal assembly procedure on an automatic liquid handler. In making just a 4 by 4 array, there are approximately 3,000 pipetting events involved. A human is prone to errors and even correct pipetting actions are likely to have greater inaccuracy than the machine liquid handler. Once a human user writes a program to form the mixing protocol (and properly debugs the program), the liquid handling robot can rapidly and reliably carry out complex mixing procedures with great accuracy. Our yield for the 4 by 4 arrays increased from the 24.99% all the way up to 45.19% (Figure 4.11) when the individual tile mixing protocol was performed by an Echo 525 liquid handler. The

mixing time to form the entire array from scratch also decreased from days to minutes. The human user still combines the tiles together in each stage.

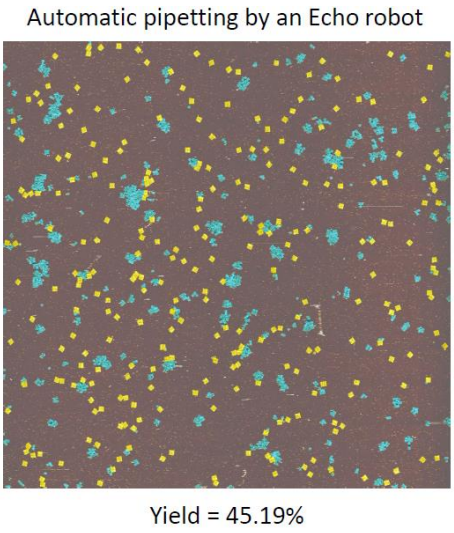


Figure 4.11. **Automatic liquid handler**. Same 4 by 4 design as in 4.10(b). Automatically mixed by an Echo 525 liquid handler. Correct structures are colored yellow, with the rest colored blue.

Another factor when scaling to an 8 by 8 array size is the addition of an additional annealing stage. For forming the 4 by 4 array, we used an annealing schedule four times as long as the first stage for the second stage. The reason was that the concentration of 2 by 2 arrays was necessarily at most a fourth the concentration of the individual tiles as four tiles were mixed together. Extrapolating this rule, the annealing time for the third stage forming an 8 by 8 array would be more than two days. To justify the long time, we attempted to form a 4 by 4 quadrant of the final 8 by 8 array where the first and second stages were annealed for the same duration. The yield of that quadrant indeed dropped from 48.72% to 32.02% (Fig. 4.12). It appeared that the longer anneal time was necessary.

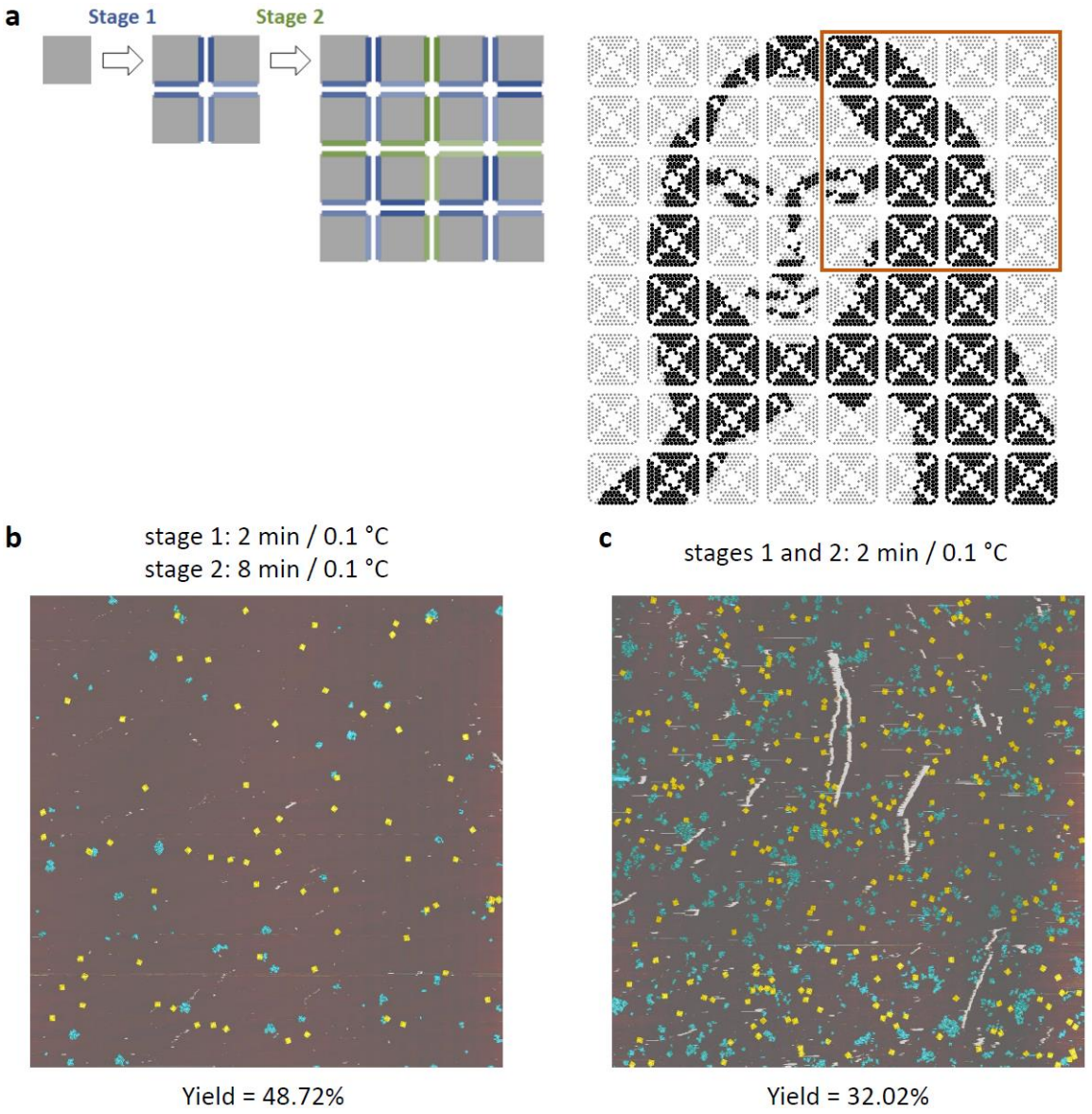


Figure 4.12. **Annealing time per stage. a,** Design diagram of a 4 by 4 quadrant of an 8 by 8 array with a Mona Lisa pattern. **b,** AFM image of the quadrant annealed for 2 minutes per 0.1 ºC in stage 1 and 8 minutes per 0.1 ºC in the stage 2. **c,** both stages were annealed 2 minutes per 0.1 ºC

Throughout the development of fractal assembly, we explored several other potential solutions to the above discussed issues. See [11 (Supplementary 6.1 to 6.9)] for other less successful but informative investigations. Importantly, what we learned from our simpler experiments on 4 by 4 arrays could apply to more complex 8 by 8 arrays due to the fractal nature of our approach — a simple set of rules repeated at different scales.

With the design principles and experimental conditions determined, we proceeded with the 8 by 8 array formation to provide us a full set of 2 by 2, 4 by 4, and 8 by 8 arrays (Fig. 4.13). These arrays were assembled in two to four stages (including stage 0, the annealing of the individual origami tile). Each subsequent stage had a lower annealing temperature range and a longer annealing time duration. Before mixing individual tiles together, we added a set of negation strands with complementary sequences to the edge staples to each tile to prevent any edge staples incorporating into locations on other tiles as we did for the finite arrays in Chapter 3, Section 5.

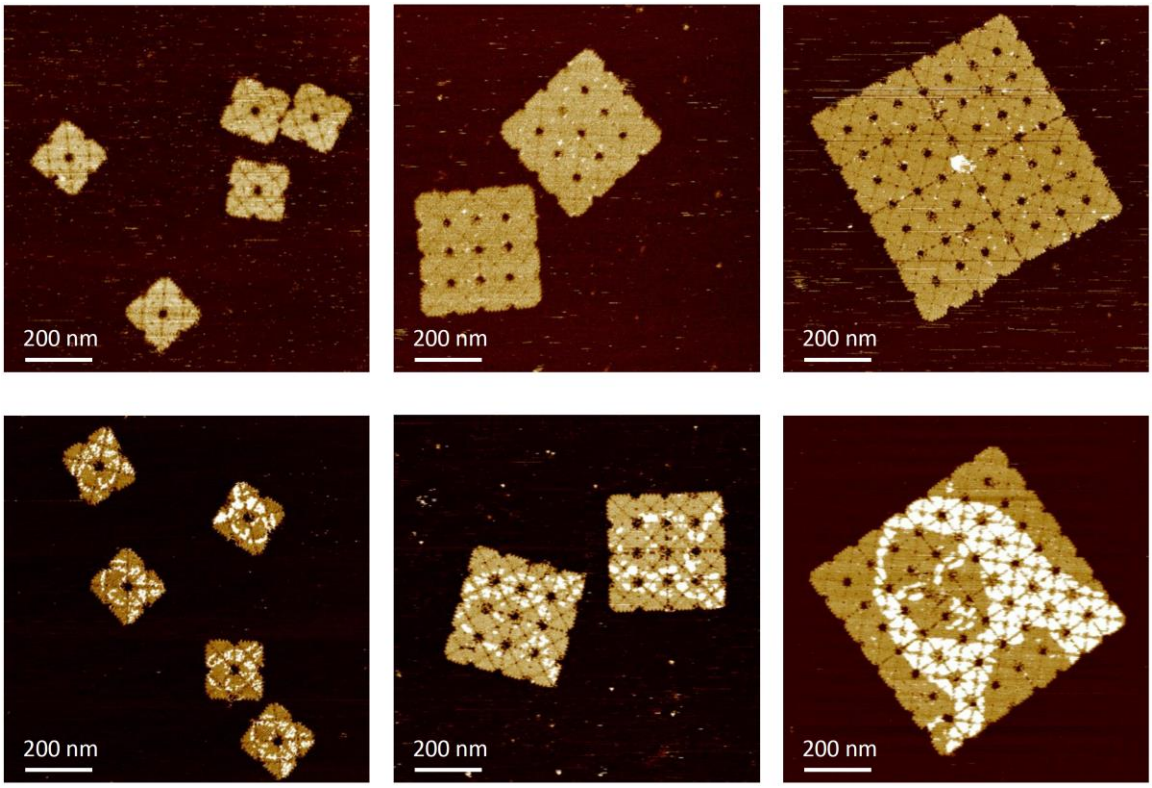


Figure 4.13. **Experimental demonstration of fractal assembly**. AFM images of 2 by 2, 4 by 4, and 8 by 8 arrays with either no surface modifications (top row) or a Mona Lisa pattern (bottom row)

With the arrays formed and each pixel patternable, in principle, the 5' or 3' end of each interior staple can be directly conjugated to, or extended as attachment sites for, diverse molecules including proteins [11], carbon nanotubes [12], polymers [13], metal nanoparticles, and organic dyes [14]. In this design, each interior staple is a uniquely-

addressable pixel that can be either be ON or OFF (with or without modification), which leads to arbitrary binary patterns. These principles could expanded upon to form more sophisticated patterns if multiple types of modifications or extensions are utilized. Our design has 136 readily addressable pixels per tile, which on an 8 by 8 array, allows the selection of 8,704 pixels with nanoscale precision on a micron scale.

4.3 Yield of arrays in relation to number of tiles

In the prior section, yield of assemblies was presented as we explored design principles and experimental conditions in the demonstration of fractal assembly. This section covers the methodology and process of determining yield, and the yield of our final assemblies.

The yield is determined as the total pixels in a complete array of the designed size, divided by the total pixels above a threshold of the background in random 10 by 10 μm images for 2 by 2 arrays and 30 by 30 μm images for 4 by 4 and 8 by 8 arrays. The error was calculated as $p\sqrt{(1-p)}/\sqrt{n}$, where p is the estimated yield and n is the number of complete arrays, treating the yield as a Bernoulli probability. This calculation was software assisted [15] but determinations of correct target structures was a manual determination. Structures cut off by the AFM image borders or structures tightly clustered in an aggregation were not considered correct target structures. Target structures are highlighted in yellow and all pixels above a background threshold are highlighted in blue.

In order to determine the accuracy of our AFM-based yield estimation, we explored whether a bias existed for binding to mica between structures. We were particularly interested in differences based off of size. To test the differences, we mixed 4 by 4 arrays with monomers at an equal concentration of individual tiles (Fig. 4.14). If the monomers bound to the same extent as 4 by 4 arrays, the yield of 4 by 4 arrays would drop in half compared to its yield calculation without monomers present. However, if there is a bias, greater than half would indicate less binding of monomers, and less than half would indicate favored binding of monomers. The AFM images showed the latter, with monomers biased to land. We hypothesize that the smaller monomers diffuse faster and

thus land faster on the mica surface. The result is our yield estimates could be lower (or higher) than the actual yield depending on the off-target structure size distribution. If the off-target structures are smaller incomplete structures, the yield is likely to be an underestimate, whereas larger aggregations would settle more slowly, resulting in an overestimate. Our AFM images show a balance of incomplete structures and aggregated structures, ideally keeping the yield estimate reasonably accurate.

a

	Binding of monomers compared to that of 4 by 4 arrays to mica surface	Yield
4 by 4 array		x
4 by 4 array mixed with monomers at a 1: 1 ratio	less	$> x/2$
	same	$= x/2$
	more	$< x/2$

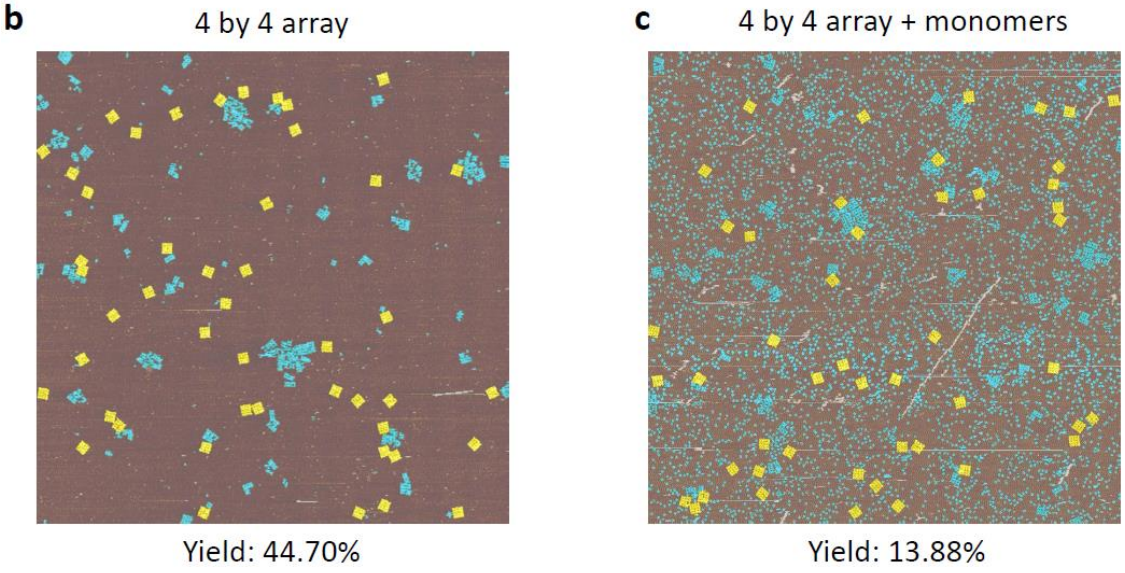


Figure 4.14. **Assessing accuracy of AFM-based yield estimation. a,** Three case scenarios for yield estimation. **b,** Yield of plain 4 by 4 arrays. **c,** Yield of the same plain 4 by 4 arrays mixed with monomers. The yield is less than half.

The basic yield of monomer DNA origami was roughly 97%. For the plain arrays (Fig. 4.15), the yield of two by two arrays was calculated at 92.81 ± 1.74%. The yield of four by

four arrays was calculated at 47.91 ± 1.76%. The yield of the eight by eight arrays was calculated at 1.81 ± 1.27%. The yield decreases quite significantly with increasing size of the array. This is not surprising when compared to other assembly approaches such as in chemical synthesis that involved sequential steps. While Fractal Assembly does not involve single tile attachments, it can be helpful to consider the yield if tiles attached one at a time. In order to match the above yields of Fractal Assembly, each individual tile attachment would need to have nearly a 95% successful attachment rate. This can be calculated from the formula 0.95^{n-1} for an n tile assembly: ~86% for 4 tiles, ~46% for 16 tiles, and ~4% for 64 tiles. Furthermore, unlike in chemical synthesis, we did not perform purification at any stage before proceeding to subsequent stages.

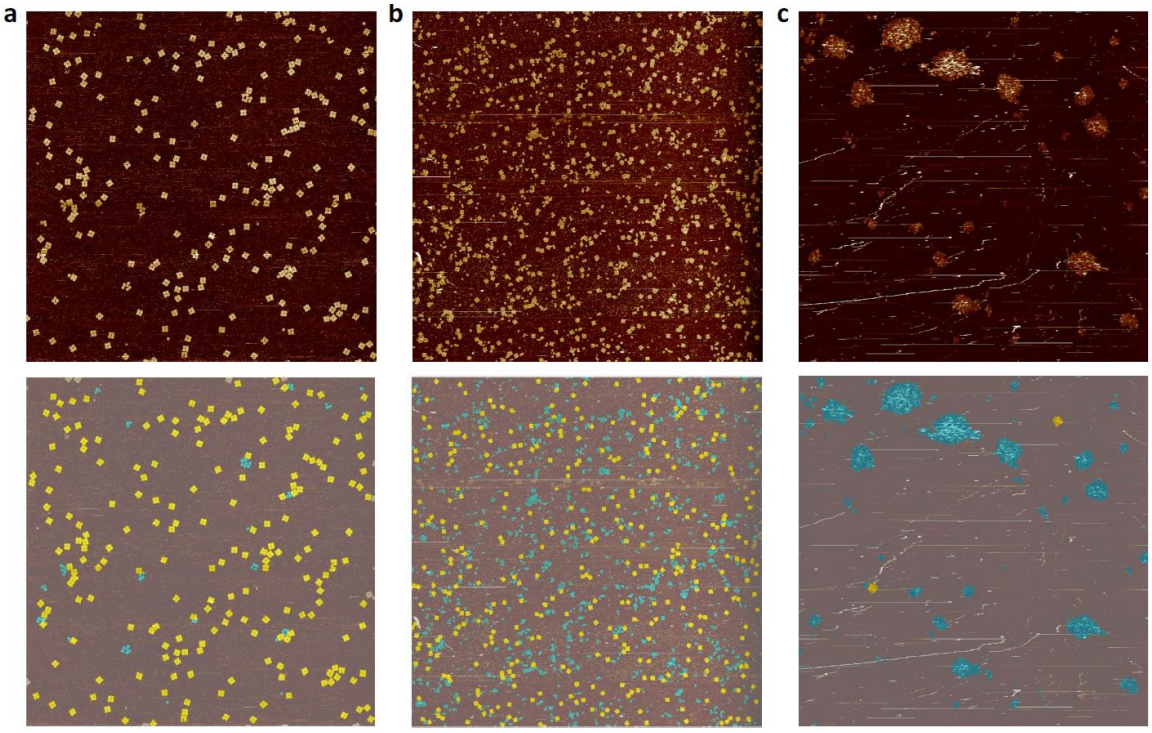


Figure 4.15. **Plain arrays. a,** a 10 by 10 μm image for 2 by 2 arrays and **b and c,** 30 by 30 μm images for 4 by 4 and 8 by 8 arrays respectively. Yield is the total pixels in complete arrays (yellow) divided by the total pixels above the threshold of background (blue + yellow). The error was calculated as $p\sqrt{(1-p)}/\sqrt{n}$, where p is the estimated yield and n is the number of complete arrays, treating the yield as a Bernoulli probability. n = 205, 384, and 2 for plain arrays of sizes 2 by 2, 4 by 4, and 8 by 8, respectively.

To demonstrate our arbitrary capability, we modified a select set of pixels with double-stranded extensions to form a global pattern design of a Mona Lisa. For the arrays with patterns, we calculated yield in the same fashion getting yields of 94.22 ± 1.22%, 41.55 ± 1.26%, and 3.22 ± 1.00% respectively for the two by two, four by four, and eight by eight arrays (Fig. 4.16). These values are remarkably similar to plain tiles without any patterns. This suggests that the Fractal Assembly rules create a robust enough structure to handle particular changes in its surface modification — a good property to have when attaching molecules such as proteins [11], carbon nanotubes [12], or polymers [13] to the origami surface.

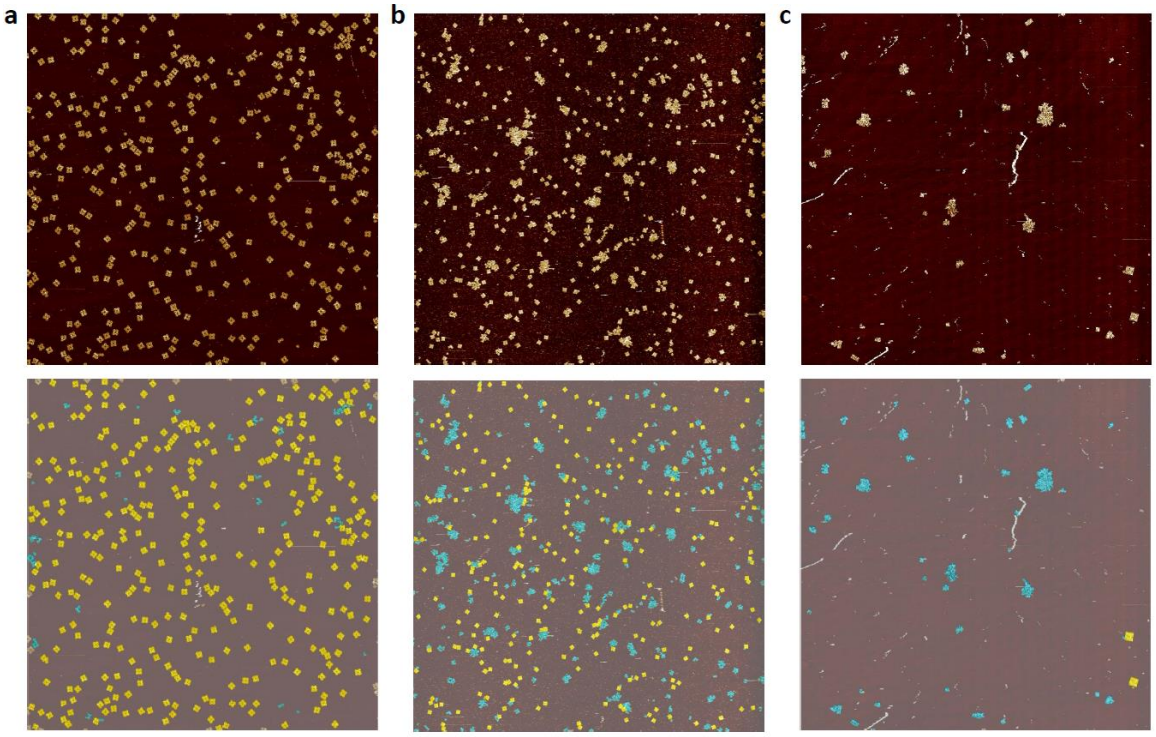
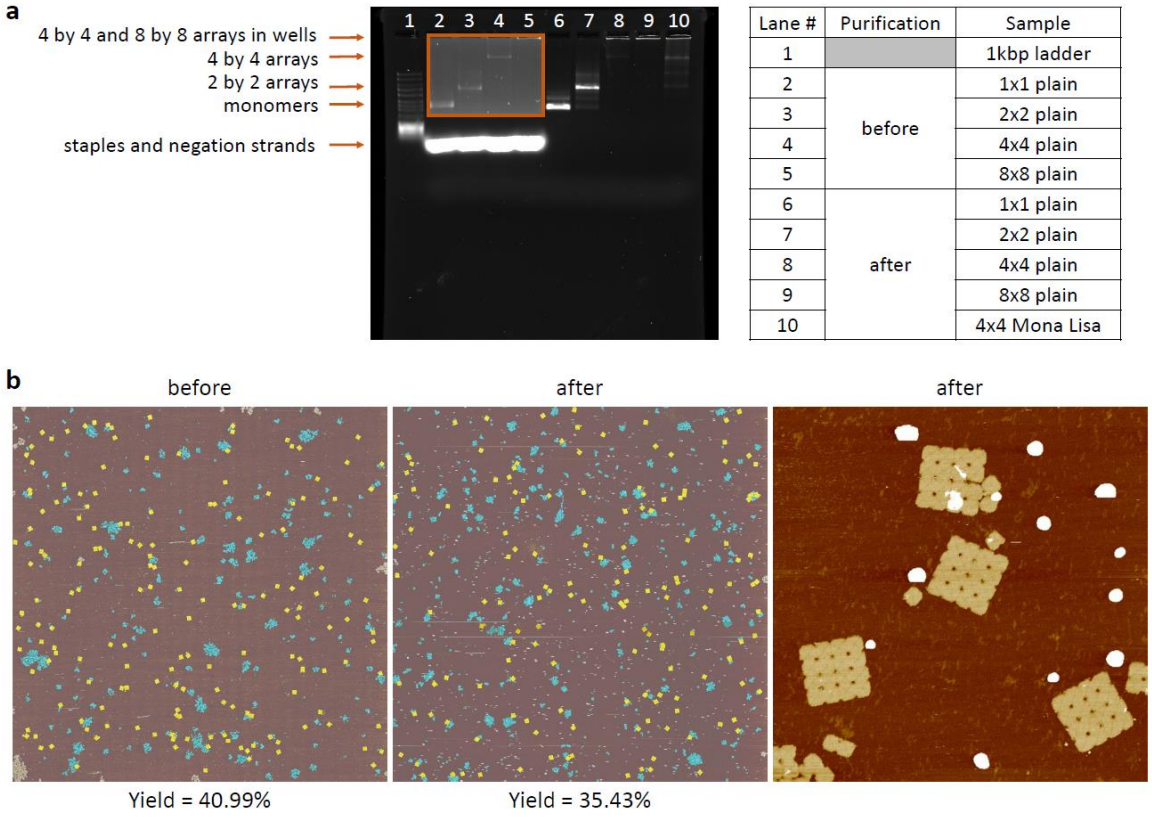


Figure 4.16. **Arrays with a Mona Lisa pattern. a,** a 10 by 10 μm image for 2 by 2 arrays and **b and c,** 30 by 30 μm images for 4 by 4 and 8 by 8 arrays respectively. Yield is the total pixels in complete arrays (yellow) divided by the total pixels above the threshold of background (blue + yellow). The mean was calculated as $p = \sum n_i/(\sum n_i/p_i)$, where p_i is the estimated yield and n_i is the number of complete arrays in each image. The error was calculated as $p\sqrt{(1-p)}/\sqrt{n}$, where p is the estimated yield and n is the number of complete arrays, treating the yield as a Bernoulli probability. n = 346, 633, and 10 for arrays with a pattern for sizes 2 by 2, 4 by 4, and 8 by 8, respectively.

Using the automatic liquid handler, we tested several other patterns on the 8 by 8 array (see section 4.4). The designs had more complex details with finer features. Using these automatic experiments, we made an estimate for the correct tile incorporation rate in four 8 by 8 array patterns (Mona Lisa, rooster, bacteria, and a circuit). For each pattern, we used a sample size of 6 to 38 arrays for each pattern. The rate varied from 99.7 ± 0.09% to 98.96 ± 0.52% depending on the pattern (Mona Lisa pattern had the highest rate and the circuit had the lowest). For the smaller arrays of 4 by 4 and 2 by 2, the rate increased to 99.93 ± 0.07% and 100% with a sample size of 90 and 167 arrays respectively.

There are several aspects of fractal assembly that merit further study. First, while our design has fairly decent continuous surface area, the pixels are not completely continuous in the origami arrays, resulting in periodic holes in the array. Using a different DNA origami with a fully packed pixel layout with fractal assembly would make a smoother breadboard. Second, a deeper understanding of the thermodynamics and kinetics of origami tile self-assembly [7, 16] would allow better control of the assembly process, allowing perhaps larger origami arrays with higher yield.

Third, for many potential application, a separation of incomplete from complete structures is desired. We explored removing excess staples and negation strands from the origami arrays after the fractal assembly process. We experimentally purified 1 by 1 (monomers) to 8 by 8 arrays using 0.5 mL and 100kDa spin filters (Amicon, #UFC510096). Each sample was filtered six times, each time was for 3 minutes at 13,000 relative centrifugal force (RCF). We used gel electrophoresis to analyze the samples before and after purification (Fig. 4.17(a)). The gel showed the successful removal of excess staples and negation strands in all samples. The monomers and 2 by 2 arrays migrated at the same speed before and after purification demonstrating they were still intact. The 4 by 4 and 8 by 8 arrays partially or completely stayed in the wells, likely because they were too large to enter the gel. We AFMed to determine the structural integrity of larger arrays (Fig. 4.17(b)). While the yield decreased, no particular deformation was observed as compared to the unpurified samples.



Lane #	Purification	Sample
1		1kbp ladder
2	before	1x1 plain
3		2x2 plain
4		4x4 plain
5		8x8 plain
6	after	1x1 plain
7		2x2 plain
8		4x4 plain
9		8x8 plain
10		4x4 Mona Lisa

Figure 4.17. **Spin-filter purification. a,** 1 by 1 (monomer) to 8 by 8 arrays before and after purification. 0.5% agarose gel, ran at 80 mV for 2 hours. The contrast was increase in the orange box to show bands at low concentrations. **b,** AFM images of 4 by 4 arrays pre and post purification.

We also investigated strengthening the origami arrays after their formation for potential purification techniques such as glycerol-gradient centrifugation [17] that have been used to separate other multi-origami structures based on their sizes. We accomplished the strengthening by added a 10-fold excess of the full set of 44 edges staples each with two stacking bonds to 4 by 4 arrays (Fig. 4.18). After incubated at room temperature for 1 hour, the staples were observed integrated into the origami tiles. The additional edge staples increased the interactions from 32 and 16 to 38 and 30 stacking bonds respectively (each 2 nt sticky end is counted at 3 stacking bonds). If applied to an 8 by 8 array, the weakest interactions would increase from 8 to 26 stacking bonds.

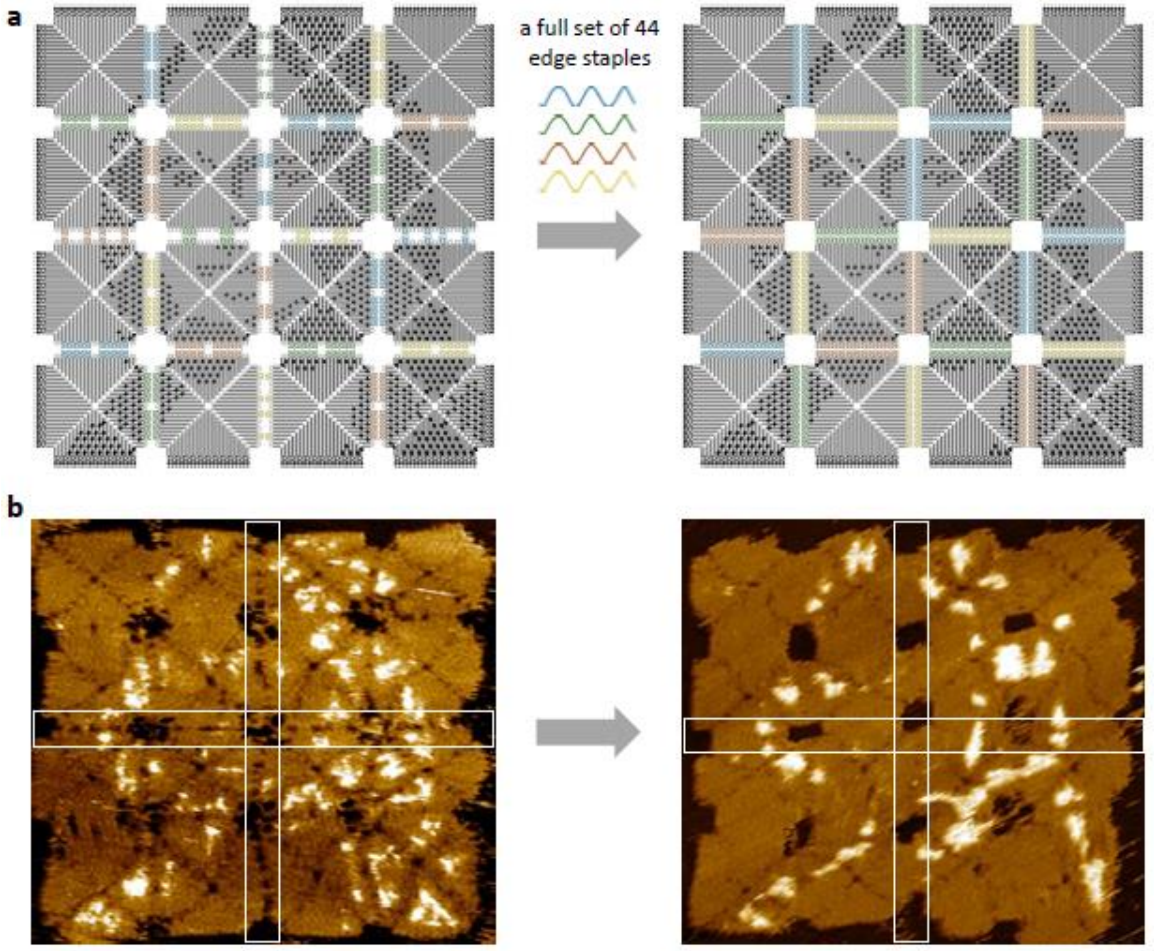


Figure 4.18. **Strengthening origami arrays after fractal assembly a**, design diagram. **b**, AFM of 4 by 4 array before and then after strengthening. The white box highlights the most obvious changes on the edges (4 staples per edge on the left to 11 staples per edge on the right).

4.4 Software tool for automated design and experiments

Forming a single array can take the mixing of hundreds to over ten thousand strands. In order to facilitate adoption of our approach, we decided to increase accessibility to Fractal Assembly through the development of an online software tool called the FracTile Compiler [18].

The compiler simplifies the design steps from creating an arbitrary pattern to the mixing protocols; see Figure 4.19. The process starts with a user selecting the size of the canvas

from a single monomer tile all the way up to our demonstrated eight by eight array. The canvas starts blank, allowing the user to select pixels and draw their desired image. An image may be uploaded, upon which, it is converted to grayscale, and pixels are automatically selected to match the thresholded result. Naturally, the threshold level is adjustable, along with the options to invert, shift, or scale the entire layout. Manual drawing or touchups is always available. All layouts can be saved for future changes. Next, the compiler will automatically convert a pixel layout into a set of tiles depicting the series of stages of Fractal Assembly necessary to create the final array. The giving/receiving, rotation, and edge code rules are all automatically applied recursively to the diagram at all stages. Furthermore, the compiler will generate design diagrams of DNA origami tiles showing the form of every edge staple. Every diagram may be saved, often with different formats.

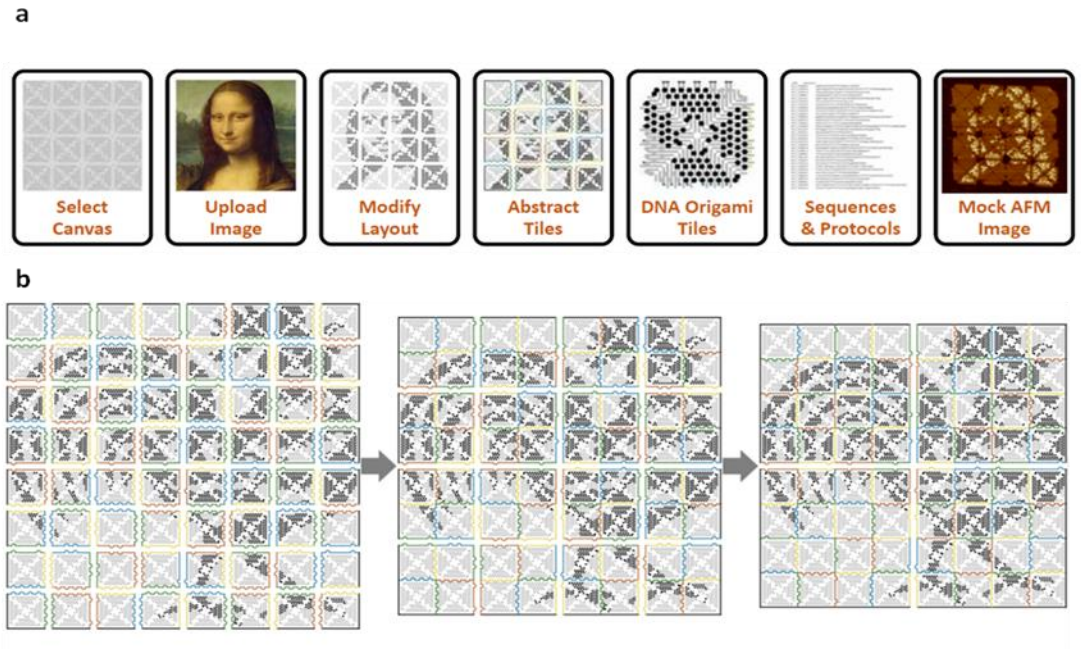


Figure 4.19. **Drawing and display options of the FracTile Compiler. a**, main options available with the software [18]. From left to right: canvas size selection, image uploading, image modification options, display abstract tiles showing the Fractal Assembly process, display detailed DNA origami tiles showing edge configurations, display mixing protocols, show image with mock AFM colors. **b**, Abstract Tile diagram display for a rooster image.

Besides ease of display, the compiler forms the mixing protocols, listed what goes in each test tube, and lists all DNA sequences necessary for those protocols. While a manual mix option is supplied that allows a user to manually pipette all DNA strands, the program supports output of a mixing scheme directly readable by an Echo 525 liquid handler. By inputting the scheme into a liquid handler, the initial mixing step can be performed fully automatically. Even for the eight by eight array, with over ten thousand staples strands, automatic mixing can be completed in around half an hour, with user presence not necessary. The need for extensive initial lab work is thus removed, only requiring artistic efforts in the design of the layout.

In order to demonstrate the generality of patterns and the ease of which design could be implemented, we designed several patterns with an assortment of fine and complex features; see Figure 4.20. Each of these designs were successfully formed experimentally with the designed patterns.

Figure 4.20: **Software aided automatically mixed experiments. a**, Design diagrams displayed in mock AFM colors predictive of the design's appearance on AFM. **b**, The actual AFM images: a Mona Lisa, a bacteria scale bacteria, a rooster, a photoreceptor circuit [19], a DNA robot (drawing), and a chess game.

Chapter 5

RECONFIGURATION OF DNA NANOSTRUCTURES USING TILE DISPLACEMENT

This chapter focuses on a general approach to reconfiguration of DNA origami nanostructures through a fundamentally new concept we call tile displacement.

Cells, the quintessential natural molecular machines, undergo significant structural reconfiguration. Why do cells bother with reconfiguration? Two very general reasons apply. First, it is critical to adapting to the environment and responding to environmental cues. Shape and structural reconfiguration in response to cues is important for many cells to perform their life sustaining tasks in the human body. For example, immune cells will undergo shape reconfiguration as they chase an invader, pushing and pulling on their membranes [1]. It is worth noting that these reconfiguration pathways are genetically broken in some unfortunate individuals, leading to a set of structurally more "static" immune cells unable to properly respond to the presence of foreign invaders, which leads to the health results expected—easy infections [2].

A second reason to reconfigure is efficiency. Most other cells will at least undergo pattern reconfiguration by switching their surface receptors as the cells mature [3] or face an environmental stimuli [4]. In this manner, a single cell might simply transform into whatever type of cell it determines is needed based off of the cues it receives. For example, mammalian stem cells will efficiently differentiate into several kinds of blood cells each performing their own tasks in the body and each with their own unique phenotypic markers on their surface [5]. Furthermore, the location of surface markers can also be very important. Many cells rely on careful signaling complexes to define the plasma membrane domains to which proteins are delivered [6]. Failure to properly reconfigure the correct plasma membrane domain leads to all sorts of diseases such as Bartter syndrome, congenital sucrase-isomaltase deficiency, or familial hypercholesterolamia [6]. Regardless,

in a healthy person, the continuous dynamic surface configuration in cells function impressively. Not only are these adaptations efficient for life, allowing one cell to potentially adapt to a wide range of tasks, but also this plasticity allows for responsive changes to external stimuli such as foreign invaders or biological chemicals.

Still, structural reconfiguration has yet to be thoroughly explored in artificial molecular machines. There have been some dynamic DNA structures (Fig. 5.1). Han et al. [7] formed a DNA Mobius strip able to take a thick and thin conformation. Gerling et al. [8] made a DNA robot-like figure with two hinged arms able to take an open or closed position. Song et al. [9] made a "domino" nanoarray able to pass an input deformation throughout an array. Others have made openable boxes [10] or drug delivery vehicles [11] that open with the right target. Ultimately, all these systems perform a specific task and lack a generalizable module for arbitrary reconfiguration.

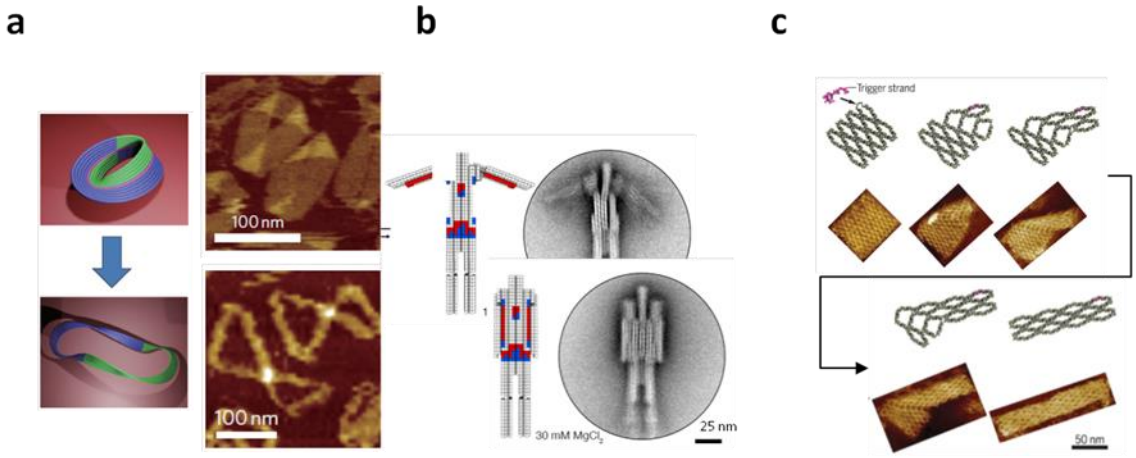


Figure 5.1. **Dynamic DNA structures**. **a**, a DNA Mobius strip [7]. **b**, a DNA robot-like figure [8]. **c**, a "domino" nanoarray [9].

We present a general approach to systematic structural reconfiguration we call tile displacement after the similar system in DNA strand interactions known as strand displacement. Our approach is designed to allow DNA nanostructures with arbitrary patterns to reconfigure arbitrary parts of the structure in an arbitrary order. Shapes

reconfiguration may also be applied. We further demonstrated control over the kinetics of tile displacement and developed several building blocks for general-purpose reconfigurations of DNA nanostructures. Examples include sequential reconfiguration, competitive reconfiguration, and cooperative reconfiguration. Finally, we explored the scalability of multi-step reconfiguration as demonstrated through a fully playable nano-scale biomolecular tic-tac-toe game, demonstrating a piece by piece arbitrary interchange of a nanostructure's subunits even to the point none of the original species on the nanostructure game board remained. In principle, such reconfiguration can allow advanced circuits with the capacity to adapt to environmental needs or heal damaged components. The plasticity may be more common to biology than to engineering, yet the strategy for autonomous reconfiguration in self-assembled DNA nanostructures is now allowing adaptive behaviors in artificial molecular machines.

5.1 A general approach to reconfiguration

This section introduces the process of developing the general approach to reconfiguration we call tile displacement.

In order to better understand control of dynamic behaviors, first we will step away from DNA origami and arrays. We look to the earlier developments in the dynamic DNA nanotechnology field for inspiration. A simple mechanism for controlling dynamic behaviors in DNA circuits [12] and robots [13] is the mechanism of DNA strand displacement (Fig 5.2(a)). A single-stranded DNA invader signal can first bind to a partially double-strand DNA complex by a single-stranded domain known as a toehold. After branch migration in a competing domain, the originally bound strand in the complex is released. Does there exist a similar mechanism for controlling reconfiguration in DNA nanostructures?

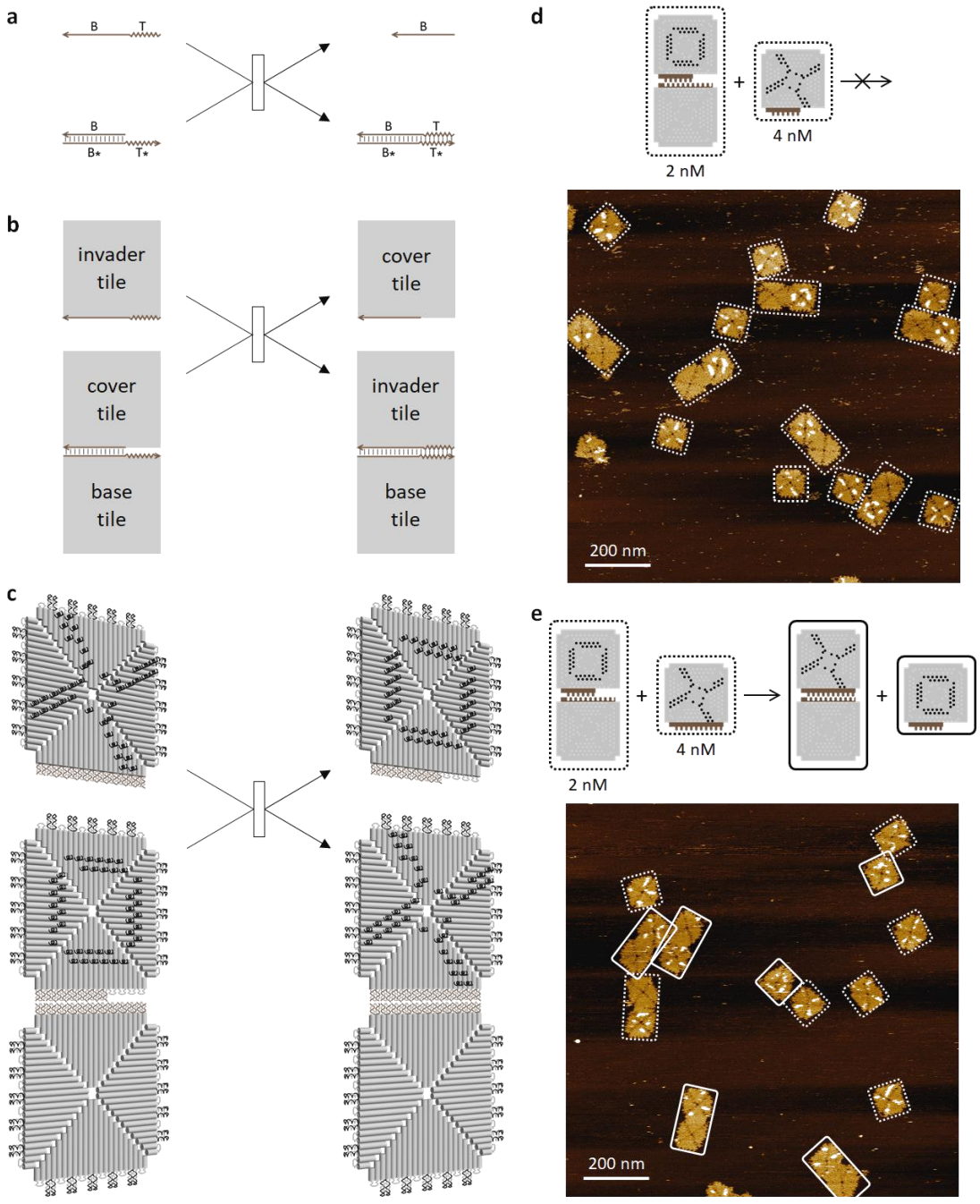


Figure 5.2. **DNA strand displacement as compared to the new mechanism of DNA tile displacement**. **a**, A simple DNA strand displacement. The single-stranding invader binds to the partially double-stranded complex via the complementary toehold T/T* domains, then through branch migration along B, releases the originally bound strand and fully integrates itself into the complex. **b**, abstract diagram of tile displacement with tile positions. **c**, 3D diagram of tiles undergoing the same reaction. **d**, Experimental data of displacement where the invading monomer lacks a toehold. Dotted outlines are of unreacted species. **e**, Experimental data of displacement where the invading monomer has a toehold. Dotted outlines are of unreacted species and solid outlines are of reacted species.

Introducing the concept of tile displacement. Instead of the invader single-stranded DNA segment, there is an invader tile, see Figure 5.2(b), and instead of a partially double-strand DNA complex, there is a partially connected dimer between two tiles. While the tiles may come in many shapes, here shown as a square, they must possess edge binding domains that is analogous to DNA binding domains. The invader tile will have a fully unbound edge, similar to the single-stranded DNA invader. The tile dimer, analogous to the partially double-stranded DNA complex in DNA strand displacement, will be bound together only partially leaving a toehold segment of unbound edge domain. The invader tile will be able to bind to the toehold, and then through some mechanism likely similar to the branch migration in DNA strand displacement, replace the originally bound dimer tile with itself. In Figure 5.2(c), the concept of tile displacement is drawn using square DNA origami tiles as the invader and dimer complex tiles. The invader tile has an X surface modification pattern to distinguish it from the originally bound dimer tile, marked with an O surface modification pattern, that the invader is designed to replace. After replacement, the original plain-O dimer is now reconfigured into a new plain-X dimer.

We experimentally formed the tiles; see Figure 5.2(d and e). Besides having an invading tile being able to replace a target tile, it is important that the tile does not undergo a replacement without a proper toehold domain; otherwise, the process of displacement would not be controllable. Two reactions were thus tested. First, an invading tile missing a complementary toehold region is added to a pre-annealed dimer complex of plain and O surface modified tiles. Second, an invading tile with a complete complementary toehold domain is added to a pre-annealed dimer complex of plain and O surface modified tiles. Upon AFM imaging the samples, the invaders with no toehold existed essentially as monomers as there was little drive for them to integrate into the dimer. This is also evidence that the dimers cannot spontaneously disassociate but must go through the tile displacement mechanism. Indeed, the invaders with toeholds had mostly converted all the dimers into the new plain and X configuration. The toehold concept is thus capable of controlling tile displacement.

While we are mostly interested in reconfiguring surface patterns due to patterning's potential applications to nano-robotics and devices, the shape may also be reconfigured (Fig. 5.3). We tested another simple tile displacement reaction with the invader tile being a triangle shape rather than another square. Hence, after the reconfiguration, the complex changed its shape from a square-square to a triangle-square. As a triangle has different edge angles as compared to a square, the integration of a triangle can change tile growth fronts, and if integrated into an array, may potentially even introduce curvature in three-dimensions.

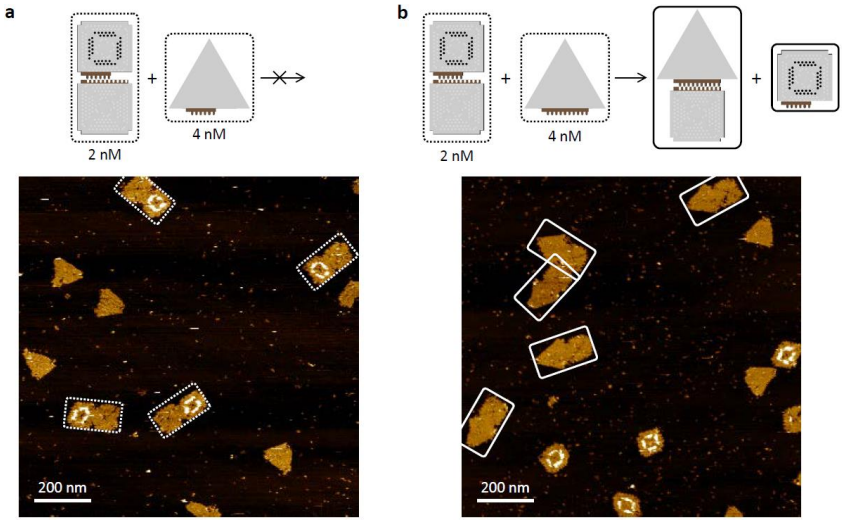


Figure 5.3. **Triangle shaped invader tile reconfigures a square-square dimer complex into a triangle-square dimer complex. a,** Triangle without a proper toehold does not react. **b,** Triangle with a toehold reacts.

When looking at the tile diagrams in Figure 5.2, it is noticeable that the edges of the designed tile and a DNA strand are only analogous in an abstract diagram. While the concept of tile displacement can be applied to a wide range of tile shapes and tile edge connections strategies, we present a method built on the prior chapters for controlling tile-tile interaction (Fig. 5.4). As noted in prior chapters, our tile's edges are composed of helices perpendicular to the edge. We controlled tile-tile interactions through several staples with sticky ends and stacking bonds. While a DNA strand hybridization involves nucleic acid base pairing together through hydrogen bonds, we bring tiles together through staples binding to other staples through sticky ends and stacking bonds. In this manner,

each staple can be in the abstract thought of as a base along a DNA strand, albeit with differing size, geometry considerations, specificity, and binding energy control. The strength of binding at each staple along the edge of a tile can be individually changed by modifying the sticky end length of that staple. The binding strength of a toehold may be also changed by modifying the number of staples interacting. In order to have uncovered domains such as with a complex's original cover tile not covering the toehold domain, we can remove sets of staples entirely. Branch migration domains were kept with 2 nt sticky ends. These domains were encoded as desired by removing select staples in order to increase the number of edge codes as done in Chapter 3, section 5.

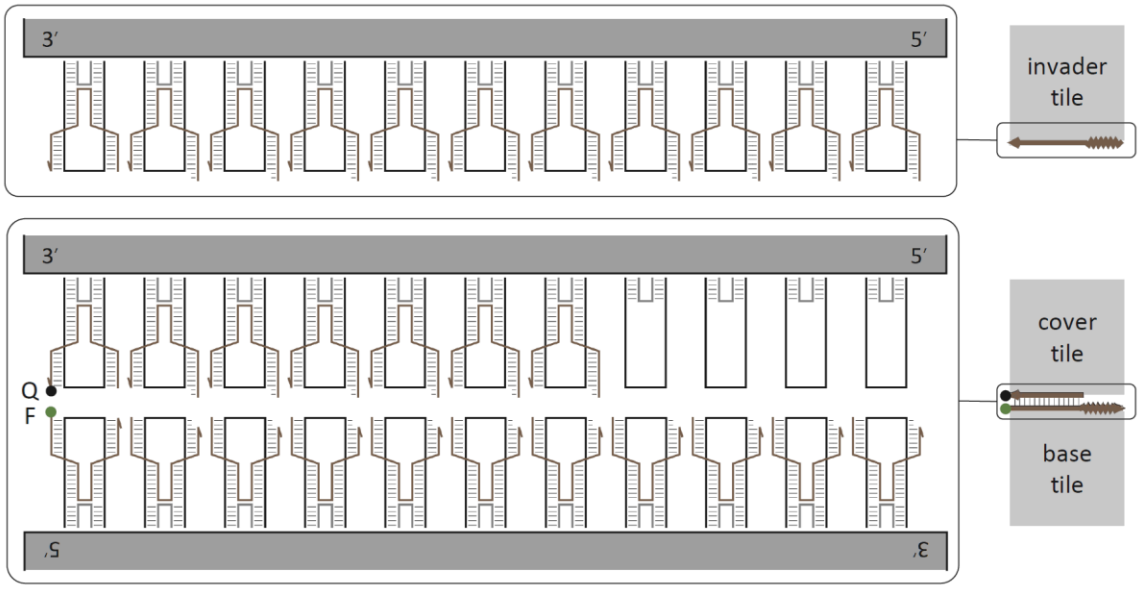


Figure 5.4. **Edge design of tiles**. All domains are formed from staples interacting through sticky ends and stacking bonds. Open toeholds are formed by removing staples in the opposite staple locations within the dimer complex.

It is worth noting the origins of tile displacement, and its behind-the-scenes existence in the formation of static nanostructures. In the prior chapter, we formed finite arrays of square DNA origami tiles. Focusing on the smallest array, a 2 by 2 array, comprising 4 tiles, the yield was estimated at over 92.81 ± 1.74%. The yield is quite high, especially when we noted tiles did not form perfectly from M13 in the first place. If tile connections were irreversible, then once three tiles interacted to form a 3-mer, the only path to forming a

complete structure would be the integration of the correct last monomer. Many incomplete structures would be expected to be leftover near the end of the anneal as they would be unable to react with each other. However, if the incomplete structures were allowed to react with each other, for example two 3-mers reacting with each other to form a complete structure and a dimer, the overall yield would increase markedly. It was results such as this that led us to discover DNA tile displacement.

5.2 Competitive Reconfiguration

This section explores the first of three general purpose reconfiguration of DNA nanostructures, competitive reconfiguration.

An important driving force in dynamic DNA nanotechnology has been the capacity to build computational systems. Many of these circuits rely on the ability to tune reaction rates as variable rates allow functions such as analog to digital conversion or thresholds [12] (Fig. 5.5). In essence, small DNA strands can be controlled to interact with one another at rates tunable over several orders of magnitude [14]. Similarly, in programming the pathways of structural reconfiguration, the ability to tune reaction rates would allow more sophisticated functions such as those that exploit competition. Cells take advantage of competing membrane modifying pathways for their own structural reconfiguration such as when guiding the growth cone of a developing neuron's axon [15]. However, in artificial nanostructures, there has been no such control developed for DNA origami interactions.

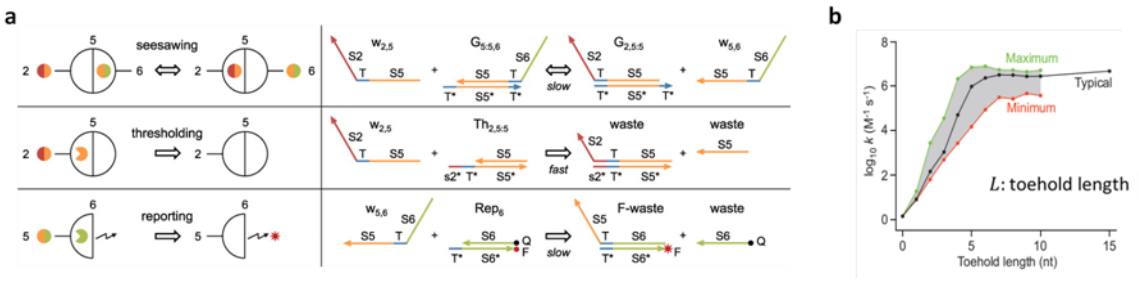
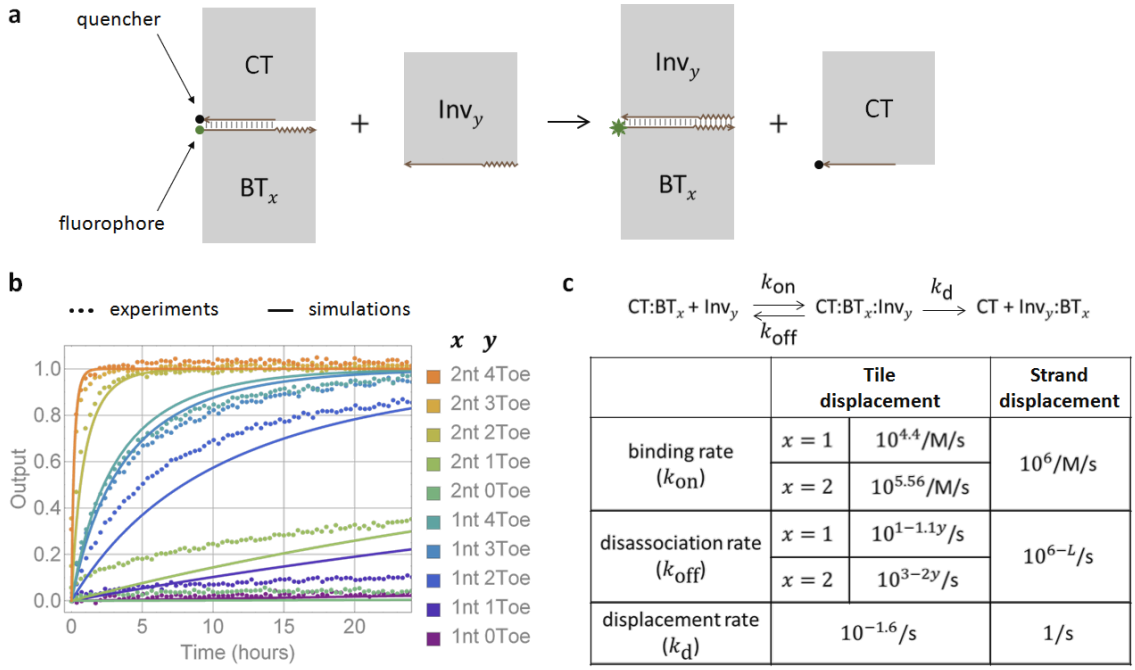


Figure 5.5. **DNA circuits and rate tuning. a**, Simple DNA building blocks using strand displacement and its rates, capable of scaling to large digital logic circuits [15]. **b,** Variable rate of strand displacement over several orders of magnitude.

We explore a range of tile toeholds for their kinetic properties. There are two general strategies we used to modify toehold binding energy. First, we can vary the sticky end length of staples involved in the toehold. We tested lengths of one and two nucleotides in the staple's sticky ends to provide variable binding energy yet still maintain specificity. Second, we can directly vary the number of staples involved in the toehold domain similar to using fewer base pairs in a DNA strand toehold. We vary the number of staples from zero to four. The branch migration domain is entirely 2 nt staples. In order to measure the kinetic rates, we formed a dimer from two tiles we refer to as the Base Tile (BT) and Cover Tile (CT), see Figure 5.6. The CT has a quencher and the BT has a fluorophore arranged such that the fluorophore is quenched while in the dimer state. The BT has the toehold domain with either 1 or 2 nt truncation receivers. If the CT is released, the fluoresence signal would rise. We displace the CT with an Invader (Inv) tile that has variable toeholds. Kinetic curves are shown in Figure 5.6(b) for 1 nt and 2 nt sticky end lengths and from zero to four staples. As expected, 2 nt sticky end curves are generally faster than 1nt sticky end curves with the exception of one staple of 2 nt. Also, decreasing the number of staples decreases the rate of the reaction as expected. At a toehold length of zero, the reaction, while not zero, is a very slow leak. The analysis of the kinetic curves is show in Figure 5.6(c) and is compared to DNA strand displacement. The results might be attributed to the DNA origami's larger size. For example, the displacement rate is lower than strand displacement perhaps due to long range geometric considerations. While the binding rate is somewhat lower, the disassociation rate is also a lower variable of the toehold binding energy. There also appears to be differences between 1 nt and 2 nt sticky ends for kinetics. It is worth pointing out that M13 sequences dictated the edge's sequences and performing kinetic measurements on another two random edges is likely to yield varying specific results; however, the trends in toehold strengths (i.e., more staples increasing binding energy) are presumably similar.



	Tile displacement		Strand displacement
binding rate (k_{on})	$x = 1$	$10^{4.4}$/M/s	10^6/M/s
	$x = 2$	$10^{5.56}$/M/s	
disassociation rate (k_{off})	$x = 1$	$10^{1-1.1y}$/s	10^{6-L}/s
	$x = 2$	10^{3-2y}/s	
displacement rate (k_d)	$10^{-1.6}$/s		1/s

Figure 5.6. **Kinetic measurements of tile toeholds. a**, abstract diagram of tile displacement reaction. There are two types of BT toeholds, one with 1 nt truncations and the other with 2 nt truncations. The Inv toehold varies between one and four staples of 1nt and 2nt sticky ends. When the Inv tile replaces the CT, the quencher is released from the BT resulting in signal rise. **b,** kinetic curves. For invaders of 1 nt and 2 nt sticky ends from zero to four staples. For example, 1nt0Toe is 1 nt sticky end with a zero staple toehold length. **c**, analysis of kinetic data for the interaction of these specific tile edges as compared to strand displacement.

With a means to control kinetics, we design a competitive reconfiguration pathway with tiles, see Figure 5.7. We take advantage of the separation between 1nt and 2nt sticky end kinetics to form two dimers. The dimer for the faster pathway has a four staple 2 nt toehold, and the dimer for the slower pathway has a four staple 1 nt toehold to provide the kinetic separation. The invader tile has a four staple 2 nt sticky end toehold which can bind perfectly with the faster pathway toehold but can also bind less effectively with the slower pathway toehold as it can only bind by 1 nt. As long as the stronger toehold actually results in a faster reaction, then it should out-compete the designed slower reaction. It would not be until the amount of invader tile exceeds the amount of fast pathway dimer that it would particularly react with the slower pathway. In this way, the reaction also works as a threshold. Experimental results roughly agreed with expectations. The Invader tile preferentially resulted in a rise in signal from the faster pathway with only a significant rise

in the slower pathway when more invader was added than there was available fast pathway. We also performed AFM on the sample. We labeled our tiles with unique surface patterns so that we could distinguish each tile on imaging. Thus, we were able to verify that starting and target nanostructures were present in expected ratios. The fast pathway successfully outcompeted the slower pathway for reconfiguration and also demonstrated a threshold reaction.

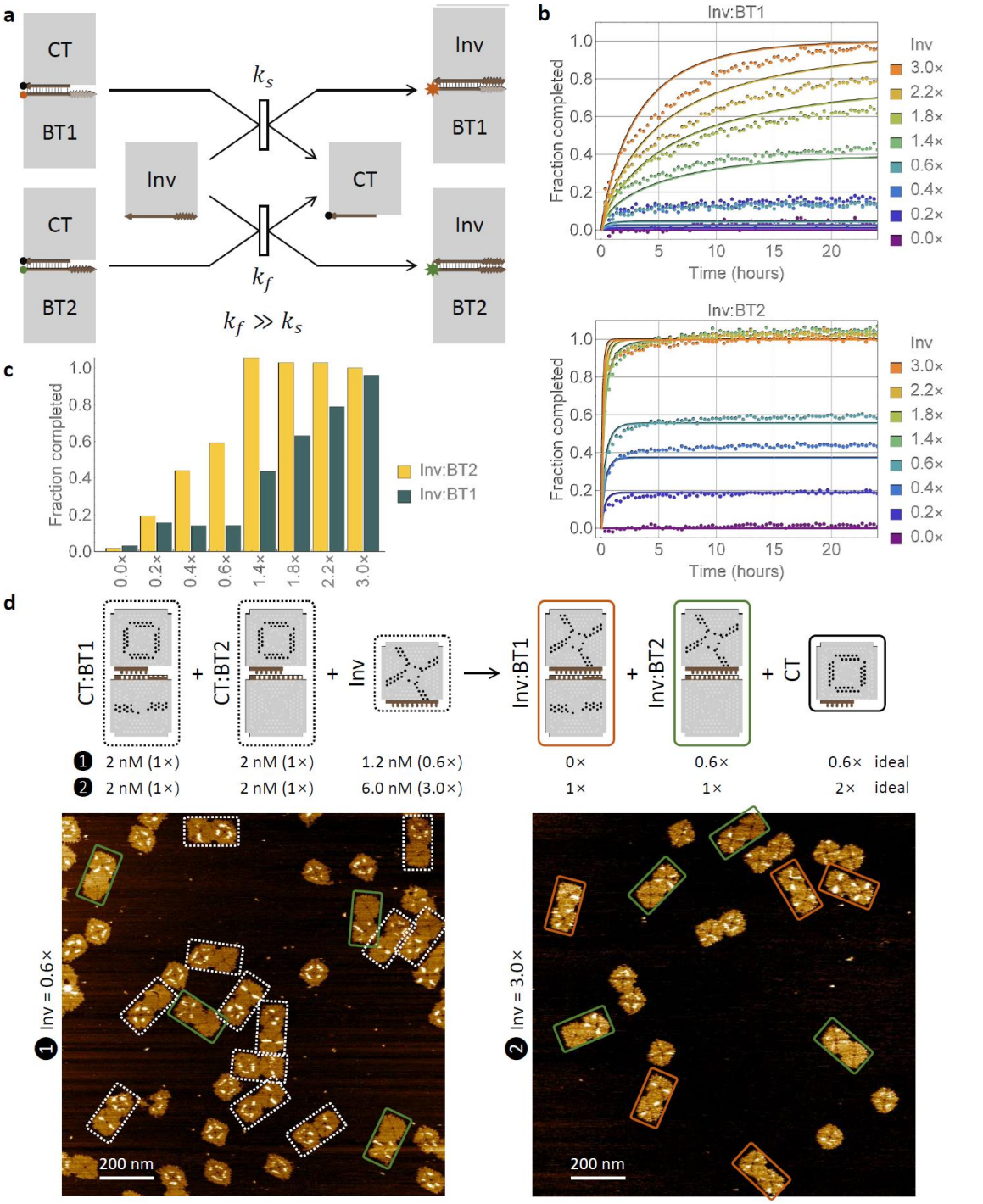


Figure 5.7. **Competitive reconfiguration. a**, tile abstract diagram. Invader tile can react with either dimer CT:BT1 or dimer CT:BT2 with different kinetics. **b**, fluoresence kinetics data for the two dimers at 2nM with variable amounts of invader tile added. **c**, Completion levels **d,** Origami diagrams and AFM images of two invader concentrations. Left: 1.2nM, right: 6nM. Dotted outlines are incomplete structures. Solid colored outlines are complete reactions.

5.3 Sequential Reconfiguration

This section explores the second of three general purpose reconfiguration of DNA nanostructures, sequential reconfiguration.

Biological pathways often work through a sequential series of steps. For example, introns are cut before protein synthesis least wasted non-sense protein be formed. Likewise, the reconfiguration of a complex structures (nano-devices for example) can avoid costly steps until all the support steps have occurred.

We likewise designed a sequential reconfiguration consisting of a size two cascade requiring the second step to occur only if the first step occurred; see Figure 5.8. Here, we also add another dimension to our method; rather than work on a single edge between dimers, we move our geometry to two-dimensions. Our starting unit is now a 2 by 2 array with four edge interactions going vertically or horizontally. Obtaining bulk data is still done with a fluorophore and quencher pair located between two connected tiles. In this sequential reconfiguration, first, a monomer tile does a tile displacement into a corner of the 2 by 2 array. The monomer invades by a strong four staples to drive the reaction forwards. In order to complete the reaction, the first monomer invader must invade two tile edges in two-dimensions. Furthermore, the invasion reaction is not a complete reaction, but rather, the invader is unable to bind to the last three staples. This mismatch is planned as three staples is not a stable domain on its own, and so even though the invader does not displace every single staple, the original tile in the 2 by 2 array can still fall off once all of its connections to the 2 by 2 array but 3 staples are displaced. When the original tile falls off, it leaves behind a toehold of 3 staples suitable for attack by the second step species of our reaction—a dimer. This dimer matches the new toehold that appears only after the first step of the reconfiguration pathway. It binds and performs its own complex displacement. While only displacing in one-dimension, by displacing off half the 2 by 2, it does a displacement across two tile edges joined together. Only when this last step succeeds would we see a fluoresence increase.

We further performed AFM to visualize our reaction. The two un-displaced tiles in the 2 by 2 array were given surface modification patterns of circles. The two tiles to be displaced were given an arc pattern such that the 2 by 2 array appeared to show a frown. The monomer tile contained a single straight line along its midpoint. When it integrated into the 2 by 2, it replaced half the frown with a straight line similar to a smirk. Finally, the dimer contained another arc pattern but upside down. Hence, when the dimer invaded for the last step, it turned the smirk into a smile.

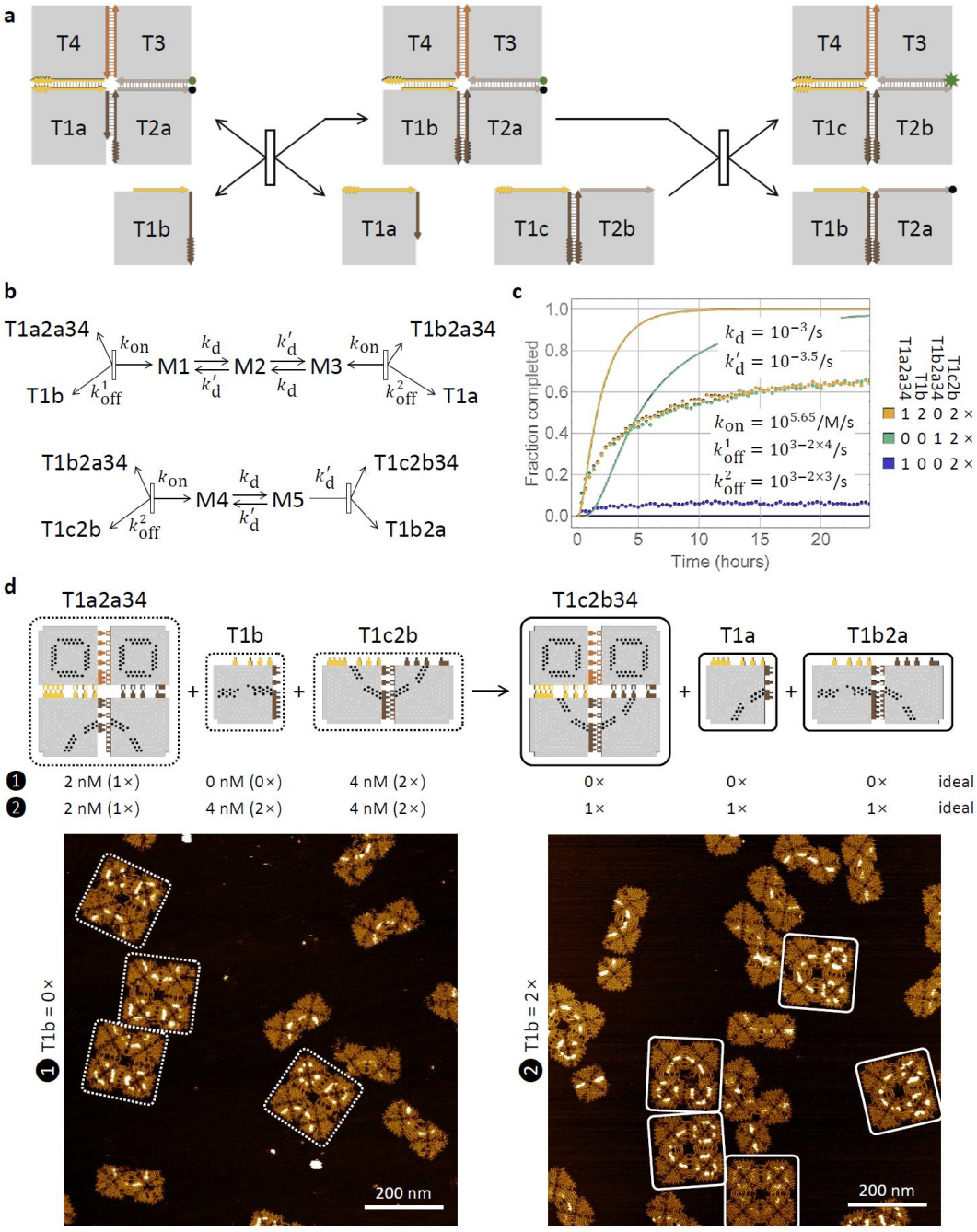


Figure 5.8. **Sequential reconfiguration**. **a**, abstract diagram of a two step sequential reconfiguration. First a monomer tile invader in two-dimensions opening a toehold for a dimer to invade and displace across multiple tiles. **b**, reaction mechanism in (a). **c,** fluorescence data. The bottom blue curve is a two by two array at 2nM plus the second step dimer at 4nM. The green curve is a two by two array with the first step monomer already pre-annealed into the array at 2nM plus the second step dimer at 4nM (kinetics of just the second step reaction). The yellow curve is the two step reaction with a two by two array at 2nM, and monomers and dimers at 4 nM. **d**, Origami species diagrams and AFM images. Dotted outlines are of starting components. Solid outlines are of reacted components. All staples are 2 nt sticky ends. See Appendix A for a detailed design diagram.

The Fluorescence kinetics data gave expected and unexpected results (Fig. 5.8(c)). We wanted to ensure that the second reconfiguration may only occur if the first reconfiguration occurred first. We tested mixing the second step dimer reaction with the starting two by two array without any of the first step monomer. Without the monomer to open the toehold for the dimer, the reaction should not take place. We see this expected result in the blue curve being reasonably flat near zero. We also wanted to see the rate of the second step as compared to the first step. We measured by second step by starting with the preformed product of the first step. This 1-step reaction reacted as shown in the green curve. Putting the whole reaction together, we tested the sequential reconfiguration with both invaders to form the yellow curve. The degree of separation between the two curves was unexpectedly close.

We then looked to the AFM data. The leak reaction of the second step dimer shows extensive frown 2 by 2 arrays even though there are many smile dimers strewn around unreacted. Thus, the dimer could not easily react without the monomer being present. The one-step reaction of the second step shows extensive smile 2 by 2 arrays where the dimer invaded successfully. Likewise, the complete reaction with both steps also shows extensive smile 2 by 2 arrays where the monomer invaded followed by a successful dimer invasion. However, the complete reaction also showed a population of 3-mers. We suspected there was an alternative pathway for the reaction contributing to a faster than expected reaction on par with the 1-step reaction. See Appendix A.4 for the proposed alternative 2-step mechanism.

In the end, we successfully demonstrate sequential tile displacement in two dimensions, opening hidden toeholds, and with domains stretching across multiple tiles.

5.4 Cooperative reconfiguration

This section explores the third of three general purpose reconfiguration of DNA nanostructures, cooperative reconfiguration.

Cooperative reactions are a building block to combining reconfiguration with logic. After all, if Structure A and Structure B both need to be present, it is a logic AND function. While similar to the previous sequential reconfiguration where the second step required the first step to have taken place, allowing both inputs to react initially should increase kinetic rates.

We designed a cooperative reconfiguration system on a two by two array (Fig. 5.9). The two halves of the array are held together by two tile edges. If a single tile were to invade, it can only, at most in one-dimension, displace one of the edges. However, if a second tile were to invade at the same time and both tiles displace one edge, then the combined effect is enough to displace off half of the array. Unlike in the prior section, here the two by two array has two initial toeholds meant for two different invader tiles. Accordingly, either invader alone can invade but should not be able to fully displace off a section of the two-by-two array. Invaders invade by 3 staples with 2 nt sticky ends.

For AFM, we reused the frown-to-smile design from the prior section. The two-by-two array starts as a frown with the two non-displaced tiles in the array being circle patterns for the eyes. Each invader swaps half the frown-into-a-smile with both invaders forming a smile.

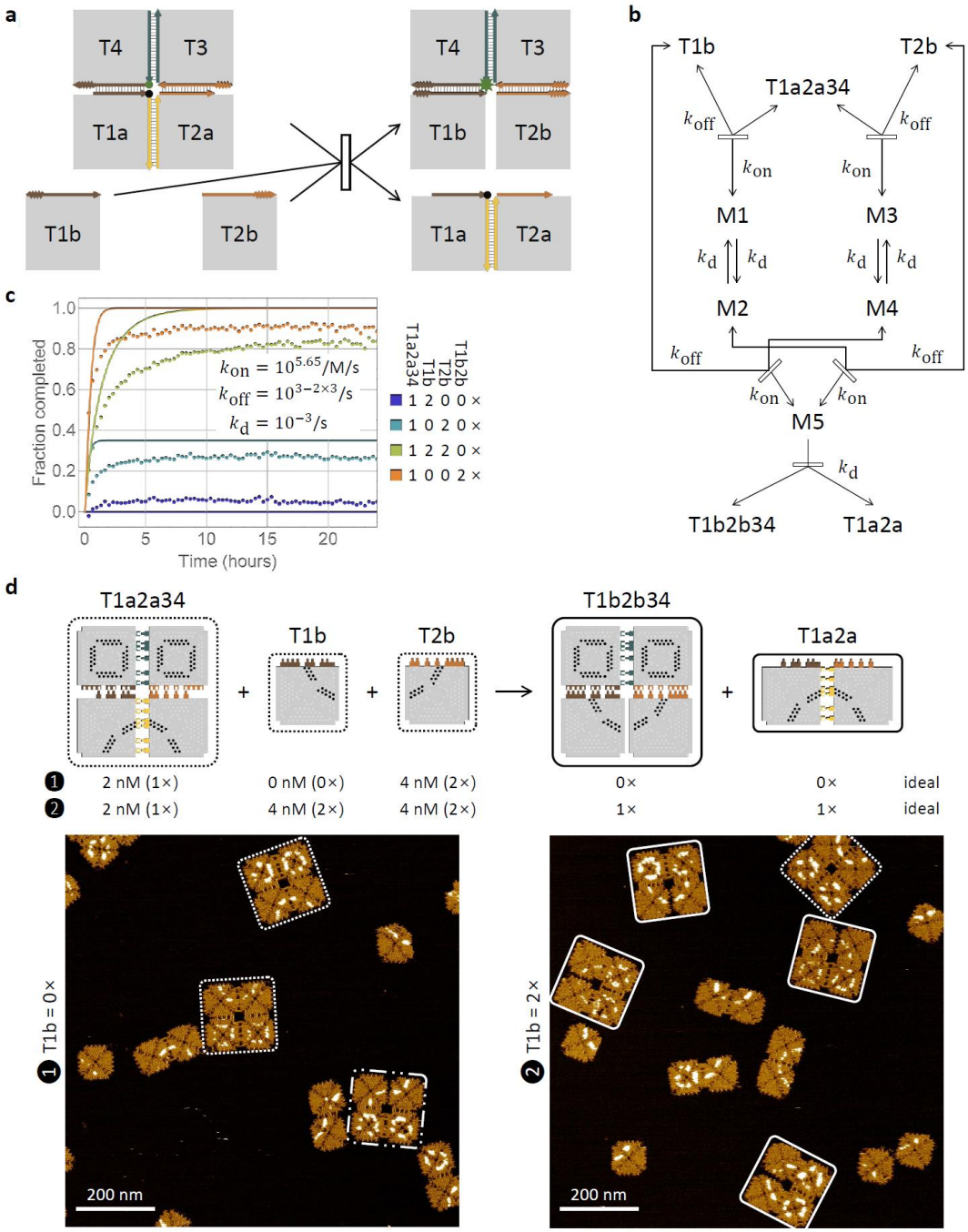


Figure 5.9. **Cooperative reconfiguration**. **a**, abstract diagram of a cooperative reaction. Two invader monomers T1b and T2b have toeholds into the two by two array of 3 staples 2 nt sticky ends. Only if both monomers are present should a dimer be displaceable from the two by two array. **b**, reaction mechanism in (a). **c**, Fluorescence data. Two by two arrays are at 2nM. Blue curves: Leak—tile T2b only is added at 4nM. Green curve: Leak—tile T1b only is added at 4nM. Note there is some increase in signal likely due to the fluorophore quencher pair being located on that tile's side, and the 5-mer with the invader in the structure being somewhat stable. Yellow curve: Both invaders present at 4nM. Orange curve: dimer invader at 4nM. **c**, Origami diagrams and AFM. Left: only one invader present. Right: both invaders present. Dotted outlines are unreacted structures. Solid outlines are reacted structures. See Appendix A for detailed design diagram.

The desired reaction took place when both invaders are present. They both successfully invade to displace off a dimer from the two by two. In principle, adding a single invader monomer is a reversible reaction as it should not be able to complete the displacement alone. However, the invader on the side with the fluorophore quencher pair results in a notable, but incomplete, signal rise. This likely means that the 5-mer structure formed when the invader integrates into the two by two array is somewhat stable. Our tile is not a perfect square and has small missing corners; see the AFM images of the two by two array in Figure 5.9(d). Since the displacement is across the center of the two by two, there is a gap that forms there which is the sum of two square DNA origami tile's missing corners. This means that the 5-mer structure is somewhat stable since, in order for the invader to be knocked off, a substantial remote toehold type reaction must take place across the gap. The T3T4 dimer may also become unstable after the invader binds and spontaneously disassociate. Still, AFM showed the desired reaction did take place forming smiles, and the leak reactions still having frowns.

5.5 Scalability of multi-step reconfiguration

This section explores the composition of our building blocks together into a large scale multi-step reconfiguration.

We demonstrate multi-step reconfiguration of a large three by three DNA origami tile array with arbitrary patterns by reconfiguring arbitrary parts of the structure in an arbitrary order. The three by three array is a large DNA nanostructure containing over 1,200 uniquely addressable pixels, potentially targets for nanoparticles [17-18], proteins [16], or polymers [19] for nano-devices or artificial molecular machine components. In principle, such large-scale reconfiguration might eventually allow advanced circuits with the capacity to adapt to environmental needs or heal damaged components.

For now, we played a massively parallel game of tic-tac-toe on our three by three DNA origami nanostructure game boards. This classic game where players take turns placing an X or an O on the board requires each tile insertion to be at any arbitrary location on the

board. Plus, the order of tile displacements must be capable of taking place in any order as each game may be different depending on a player's whim. Each location can either be turned into an X or an O from an X or O invader tile. Also tiles being able to integrate into any location in an array means corner, edges, and even the center of an array. Tile displacement must truly operate in two-dimensions with geometry. The entire board must be open to reconfiguration.

5.5.a Tic-Tac-Toe design

We first designed a three by three array game board capable of allowing reconfiguration at any of its nine locations. Having a location reconfigured must also not prohibit any other tile from also reconfiguring in neighboring locations. See Figure 5.10 for our design. The board has several open toeholds. These toeholds are all unique and guide invaders to that specific location. Toeholds are 2 staples with 5 nt sticky ends for high specificity in a short domain and varied between 5' and 3' extensions to create more design space. Invaders only invade by one toehold expect for the center location which invades by four toeholds. The corner invader must bind by one toehold and branch migrant across two tile edges in two dimensions. The edge invader must bind by one toehold and further branch migrant across three tile edges in two dimensions. The center tile has one toehold for each of its four edges that it needs to branch migrant across.

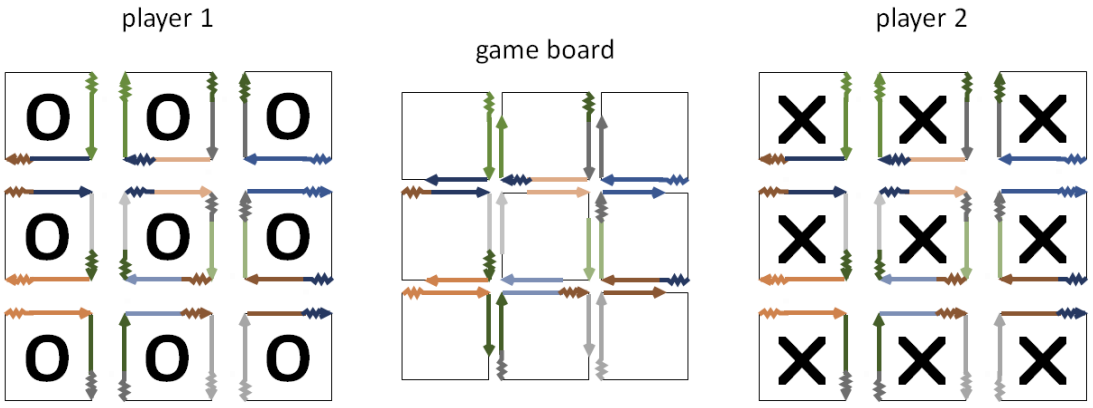


Figure 5.10. **Design of Tic-Tac-Toe game board on a three by three array**. Toeholds are marked with jagged colored lines. Branch migration domain are solid colored lines. See Appendix A for detailed design diagram.

The only difference between invaders between the two players is the surface modification pattern to appear like an O or an X. Otherwise, each player gets 9 tiles, each targeting a different location in the array. Note that while an invader may appear to have multiple toeholds on its edges, only one toehold is used for every tile except the center. For example, the top-left invader tile has a brown and green toehold. It will invade using the brown toehold that matches the toehold on the left-middle tile in the game board. The green toehold is unused in the invasion; however, it maintains the presence of the green toehold in the game board. Thus when the center-top tile invades, it will have the green toehold to invade with regardless of the move order.

Based off of prior result kinetics, we expected it to be harder for an invading tile to cross multiple tile edges. This is of particular concern for the center tile as all four of its edges must be branch migrated. It is for this reason we apply four toeholds to the center tile, one toehold for every edge. We expect such a design to increase the reconfiguration yield of the central tile.

5.5.b Kinetics of each move

Before playing a full game, we explored the kinetics of each general move class: corner, edge, and center. We needed to ensure every type of move was possible in order to have a board that is truly able to arbitrarily reconfigure. We formed all the boards at 4nM and mixed in 8nM of either a corner tile, a edge tile, or the center tile and measured the kinetics as with prior experiments; see Figure 5.11.

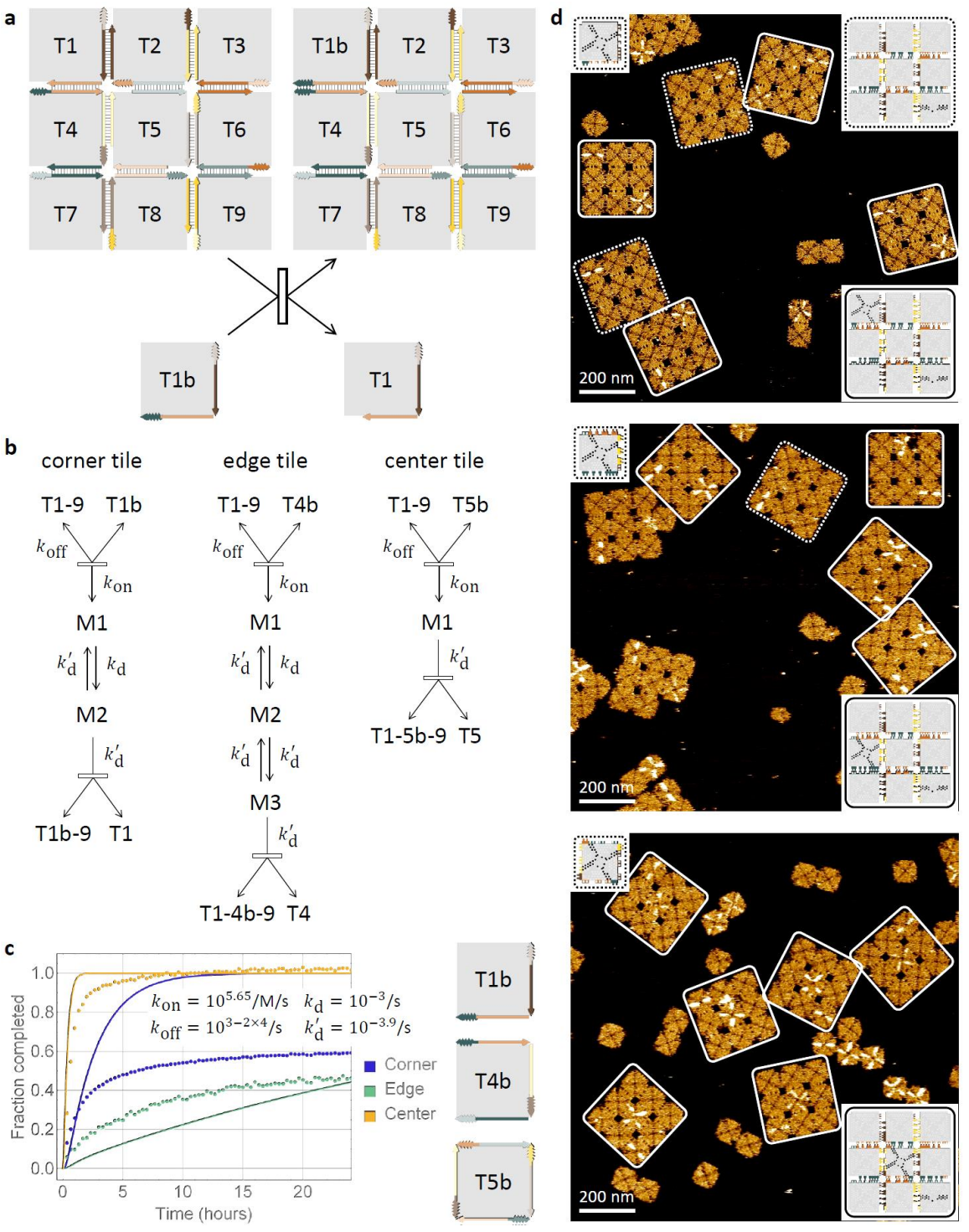


Figure 5.11. Tic-**Tac-Toe kinetics of the corner tile, edge tile, and center tile. a**, Abstract diagram. **b,** Reaction mechanism **c,** Fluorescence data. Green is the edge tile. Blue is the corner tile. Yellow is the center tile. **d**, Origami diagrams and AFM images. Top to bottom: corner, edge, and center tile added respectively. Dotted outlines are of non-displaced game boards. Solid outlines are properly displaced game boards.

The fluorescence data provides interesting but not unexpected results. First, the corner tile reacts more readily and more completely than the edge tile. This is expected as the edge tile has to branch migrant over three tile edges with only one toehold, whereas the corner only has the branch migrant over two tile edges also with only one toehold. The edge and corner appear to reconfigure to an extent the Tic-Tac-Toe game should be possible to play. Second, the center tile reacts nearly perfectly. While the center tile was likely to have several issues as it required displacing over four edges, the addition of a toehold on every displacement edge appeared to solve the problem. Once bound by a toehold, that toehold simply has to encourage the displacement of one edge similar to our initial dimer experiments that reacted rapidly and to a high completion level. The center invading tile is apparently flexible enough to bind to the four toeholds at some point in the displacement process.

Invader tiles also appear to localize to their target location accurately with, for example, no observed center tiles integrating in an edge location. With each of the three classes of moves successful, the game of tic-tac-toe is ready to be played

5.5.c Demonstration of the Tic-Tac-Toe game encompassing all move types

Over the course of several days, games of Tic-Tac-Toe were played. Each day, a player would mix an increasing amount of excess of invader tile of their choice into a game board mix. The samples were allowed to incubate at room temperature overnight to allow that tiles time to reconfigure the game board. The next day, a small amount of sample was taken for AFM imaging to verify the correct displacement. Also, the other player would then have the opportunity to place a move from the opposing side at that point. This continued until a player won; see Figure 5.12.

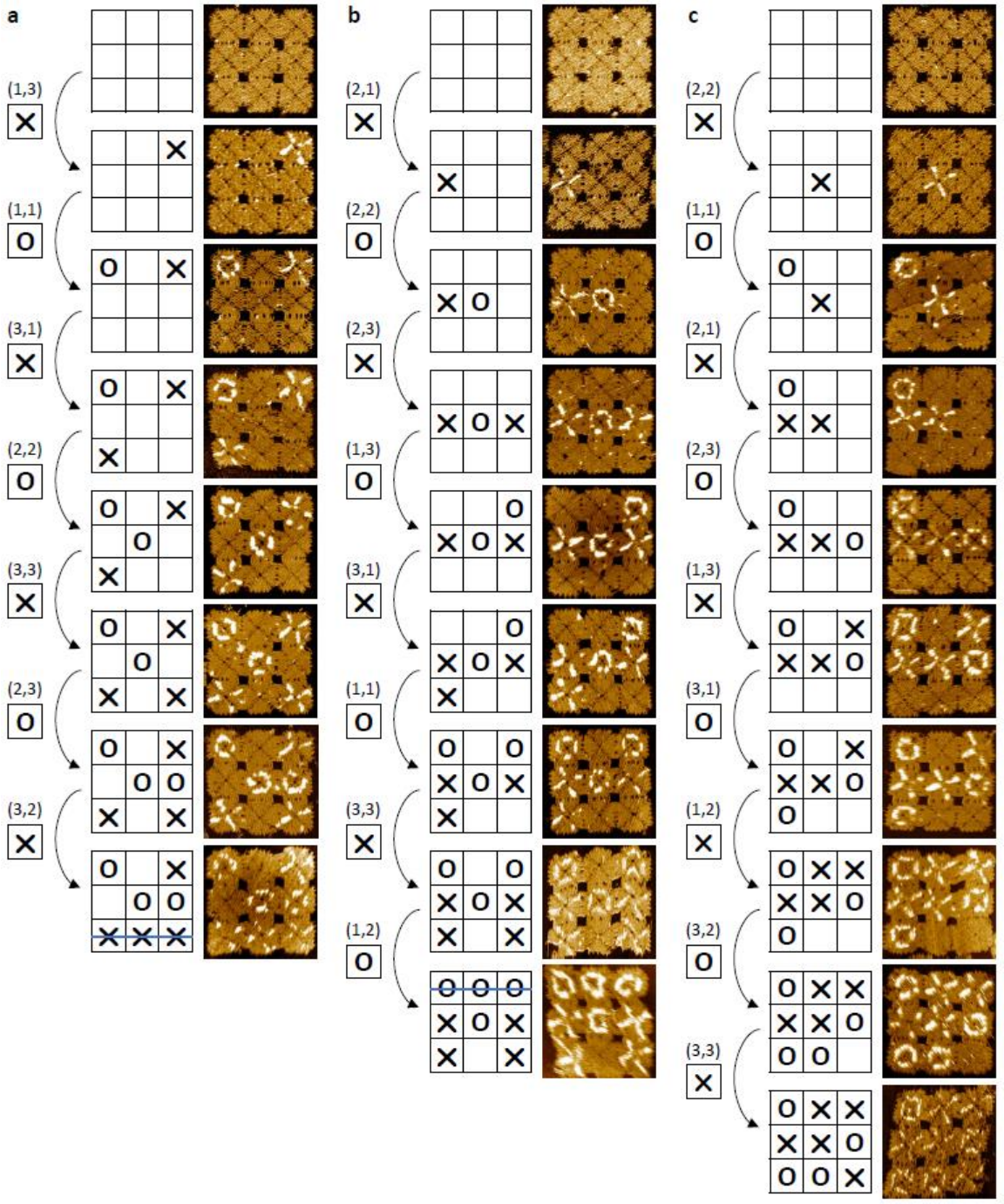


Figure 5.12. **Full games of Tic-Tac-Toe based on tile displacement to reconfigure the game board.** For each game played, the abstract diagram is shown on the left and an AFM image of that move on the right.

As we AFMed the game board after every move, we were able to see the development of the game. By the end, we had successfully replaced 7/9, 8/8, and 9/9 of the entire game board with new custom patterned tiles. Between all the games, tiles integrated into all playable locations. One game even played all nine locations replacing the entire game board. As DNA origami is highly compatible with a wide range of molecular compounds such as proteins [16], nanoparticles [17, 18], or polymers [19], each new tile could potentially bring its own device into the original system in response to environmental cues or even to heal damaged components.

Chapter 6

CONCLUSIONS

In this thesis, we discussed our contributions to the field of DNA nanotechnology by developing design rules and systematic approaches to controlling nanostructure complex assembly. Drawing inspiration from the biology's quintessential molecular machine, the cell, these rules and approaches allow for the construction of artificial molecular structures with tunable diversity, large systems approaching the size of bacteria yet retaining nanometer precision, and biological plasticity inspired dynamic systems for arbitrary reconfiguration.

In Chapter 2 of this thesis, we introduced the development of a new DNA origami tile designed specifically to work as an optimized 2D breadboard for larger complex nano-devices and machines. We designed an entirely flat, one-helix thick tile tailored to array formation with a single, non-flipping tile. With interest in the tile forming a breadboard for future devices, we optimized the tile for a high continuous surface area to allow the greatest flexibility in future device designs.

In Chapter 3, we used the tile to create a framework inspired from molecular stochasticity for programming DNA array formation and gaining control over diversity of global properties through simple local rules. Combinatorial chemistry techniques in synthesizing one-dimensional polymer chains, such as SELEX, have revolutionized chemical synthesis and the selection of functional nucleic acids. We expand these principles to random two-dimensional networks to open new opportunities for fabricating more complex molecular devices on DNA nanostructures. Three general forms of planar networks, random loops, mazes, and trees, were manipulated on the micron scale upon the self-assembled DNA arrays. We demonstrated control of several properties of the networks, such as branching rules, growth directions, the proximity between adjacent networks, and size distributions. The large diversity, in principle, allows for a wide, but tunable, testing environment for

molecular circuits. By further applying these principles to subunits of finite assemblies, variable components may be mixed with fixed components potentially opening additional applications in high throughput device or drug screening over two-dimensions.

Next, Chapter 4 of this thesis demonstrates our robust hierarchical strategy to break the two-dimensional size limitations of DNA origami, and its application to forming micron-scale uniquely-addressable DNA origami arrays.. While DNA origami allows nanometer precise placement, the size remains roughly below 0.05 um^2. Our strategy relied upon a simple rule set applied recursively in each stage of a hierarchical self-assembly process. Furthermore, the upfront cost of manufacturing the arrays is fixed regardless of scale, as the strategy employs a constant number of unique structural DNA strands. We also developed a software tool to automatically compile a designed surface pattern into experimental protocols. We experimentally demonstrated DNA origami arrays approaching the size of small bacteria, 0.5 um^2, with several arbitrary patterns, each consisting of 8,704 specifically chosen pixel locations with nanometer precision, including a bacteria sized portrait of a bacteria. These arrays may be used for complex organization of diverse molecules and device fabricating with nanometer precision of components over the entire micron scale of the array. Opportunities arise for larger and more sophisticated molecular machines such as DNA robots or DNA circuits. Fundamentally, the simple recursively applied hierarchical strategy may be more widely used to build complex molecular systems with a constant number of simple components not limited to a specific design.

Finally, in Chapter 5, we present a general purpose approach to reconfiguration in DNA nanostructures. Reconfiguration in biological cells provides efficiency and responsiveness to environmental stimuli. In principle, a reconfiguring device can be modified to a new task, adapt to external cues, or even heal damaged components. In an approach we call DNA tile displacement, we showed that a DNA origami array may have tiles arbitrarily replaced by another tile, including tiles of another shape or surface pattern. We also demonstrated control over the kinetics of tile displacement and performed several general purpose reconfigurations of DNA nanostructures. Examples include sequential

reconfiguration, competitive reconfiguration, cooperative reconfiguration, and finally the scalability of multi-step reconfiguration as demonstrated through a fully playable nano-scale biomolecular tic-tac-toe game. The major ramifications is a plasticity more common to biology than to electronics—molecular platforms with arbitrary patterning that can reconfigure an arbitrary part of the nanostructure in an arbitrary order based on environmental signals.

We have demonstrated tunable combinatorial diversity, precise nanometer addressability on the scale of small bacteria, and even a framework toward adaptive behaviors in artificial molecular machines. Eventually, we may see engineered structures that qualify as life, rivaling cells. Our strategies take us closer.

APPENDIX A: ADDITIONAL DIAGRAMS AND IMAGES

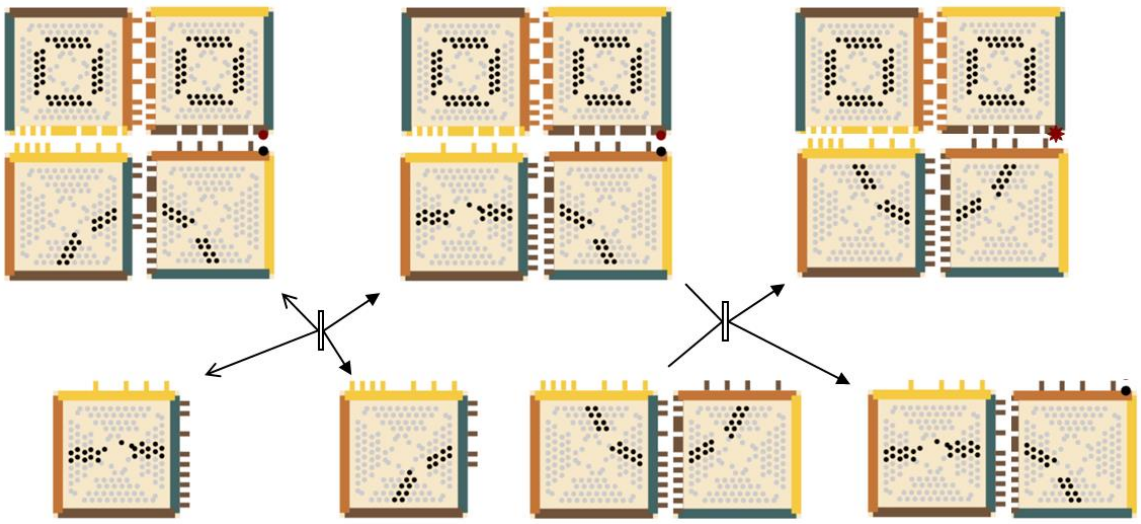


Figure A.1. **Specific edge and pattern design of sequential reconfiguration.** Tile orientation N, E, S, and W are represented by the green, brown, orange, and yellow bars respectively. Giving staples with a 5' 2 nt sticky end and 3' stacking bond are represented by rectangular protrusions from the colored bars. The color of the protrusion represents the complementary edge (N, E, S, or W) of that staple sticky end. Receiving staples 5' stacking bonds and 3' 2 nt truncations are rectangular indentations in the colored bar. Their sequence is determined by the M13 sequence on that edge. A Rox fluorophore is represented by a red dot and is bound to the 5' end of a staple with an additional 3' stacking bond. A Iowa black quencher is represented by a black dot along a tile's edge and is bound to the 3' end of a staple with a 5' stacking bond. 136 pixel locations are represented by gray dots (OFF pixels) or black dots (ON pixels) within the tile's interior. ON pixels have a double stranded extension at that location.

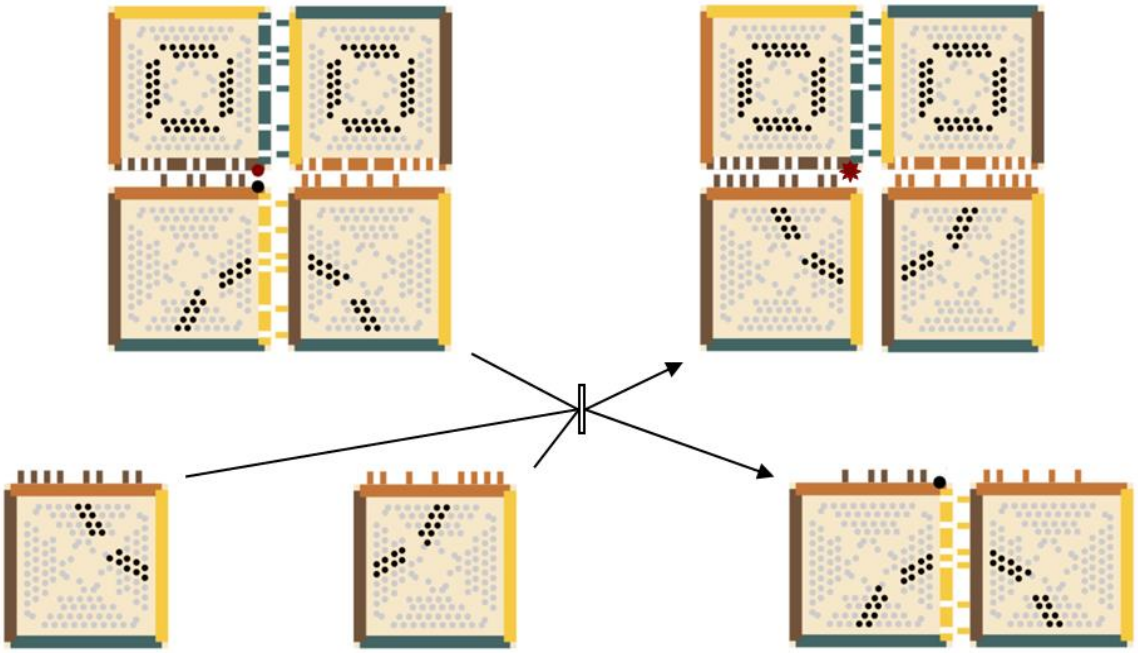


Figure A.2. **Specific edge and pattern design of cooperative reconfiguration.** Tile orientation N, E, S, and W are represented by the green, brown, orange, and yellow bars respectively. Giving staples with a 5' 2 nt sticky end and 3' stacking bond are represented by rectangular protrusions from the colored bars. The color of the protrusion represents the complementary edge (N, E, S, or W) of that staple sticky end. Receiving staples 5' stacking bonds and 3' 2 nt truncations are rectangular indentations in the colored bar. Their sequence is determined by the M13 sequence on that edge. A Rox fluorophore is represented by a red dot and is bound to the 5' end of a staple with an additional 3' stacking bond. A Iowa black quencher is represented by a black dot along a tile's edge and is bound to the 3' end of a staple with a 5' stacking bond. 136 pixel locations are represented by gray dots (OFF pixels) or black dots (ON pixels) within the tile's interior. ON pixels have a double stranded extension at that location.

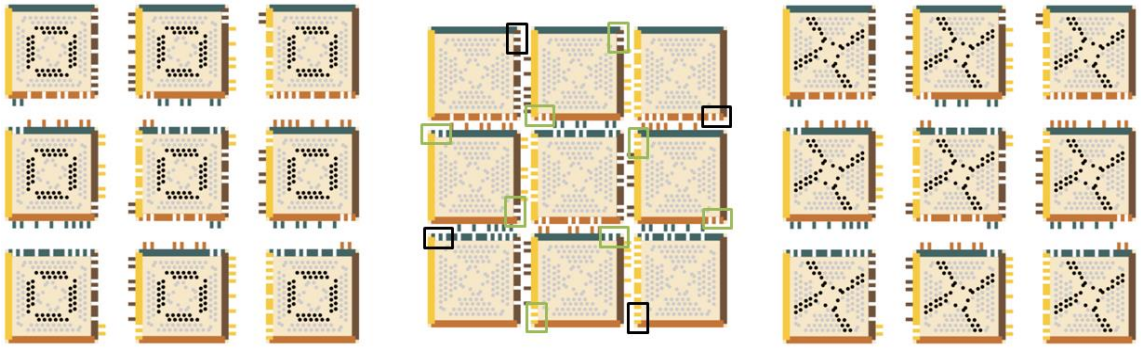


Figure A.3. **Specific edge and pattern design of Tic-Tac-Toe Game.** Tile orientation N, E, S, and W are represented by the green, brown, orange, and yellow bars respectively. In the middle is the game board with toehold regions boxed. On the right and left are the two players X and O. The tiles of the two players are identical in terms of edge design. The toehold receiver staples boxed (and their respective givers from the player O and X set) are two 5 nt staples. The green boxes use 3' trunctions and 5' stacking bonds with their matching giver on the invading tiles being 5' extensions and 3' stacking bonds. The black boxes use 5' trunctions and 3' stacking bonds with their matching giver on the invading tiles being 3' extensions and 5' stacking bonds. By switching the sticky end side of the staple, we are able to have multiple toeholds of the same length at the same location on a particular edge yet maintain orthogonality. For all other staples: giving staples with a 5' 2 nt sticky end and 3' stacking bond are represented by rectangular protrusions from the colored bars. The color of the protrusion represents the complementary edge (N, E, S, or W) of that staple sticky end. Receiving staples 5' stacking bonds and 3' 2 nt truncations are rectangular indentations in the colored bar. Their sequence is determined by the M13 sequence on that edge. 136 pixel locations are represented by gray dots (OFF pixels) or black dots (ON pixels) within the tile's interior. ON pixels have a double stranded extension at that location.

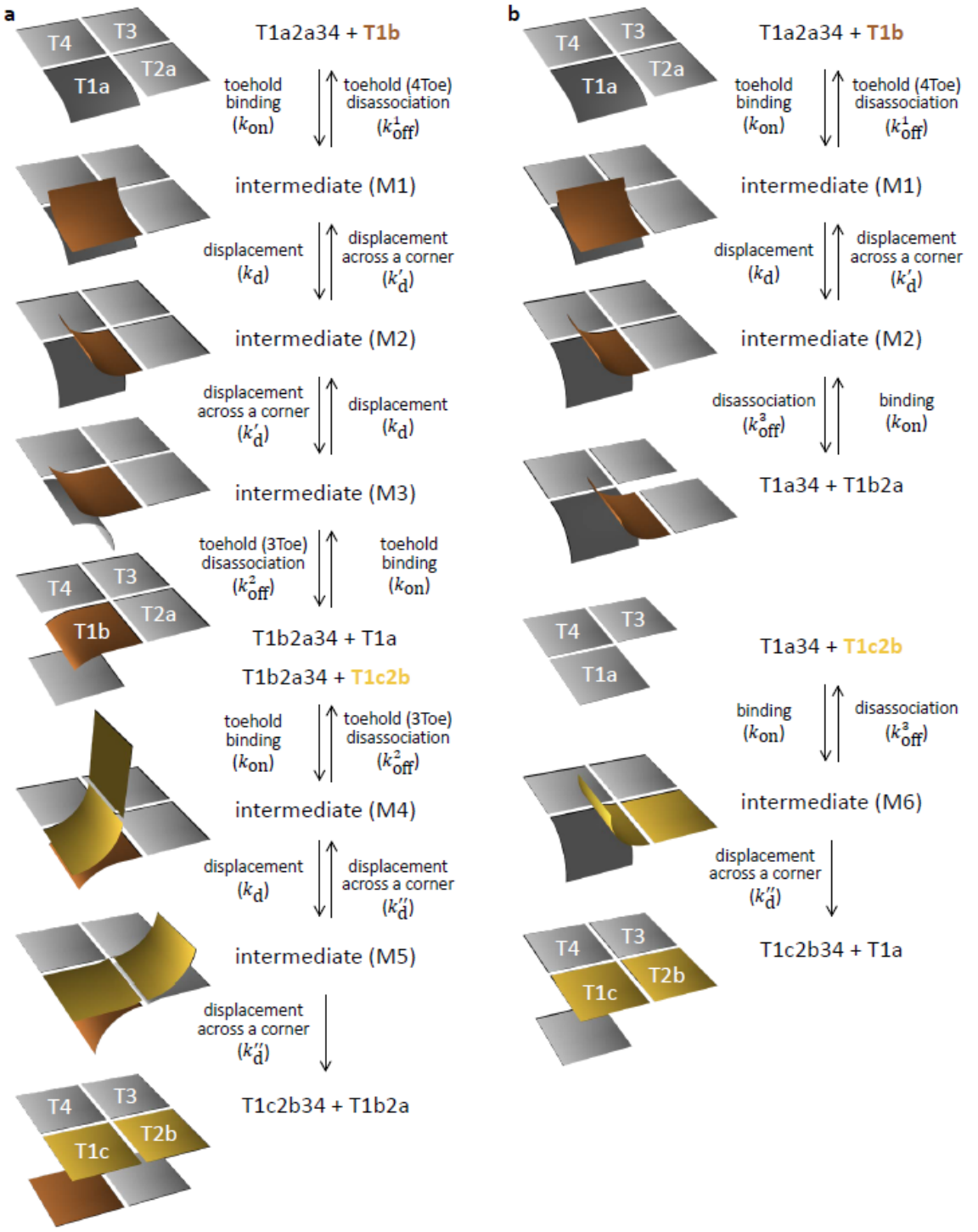


Figure A.4. **Alternative sequential reconfiguration pathway. a,** Designed pathway. **b,** Alternative pathway.

APPENDIX B: DNA SEQUENCES

Chapter 3: all staple sequences and Cadnano diagram for all designs presented are located on pages 79-137 in the supplement of prior work [1].

Chapter 4: all staple sequences and Cadnano diagram for all designs presented are located on pages 33-47 in the supplement of prior work [2].

Chapter 5: Non-edge staples and Cadnano diagram for all designs presented are located on pages 33-38 in the supplement of prior work [2]. Edge staples for each design reproduced below.

Edge staple name nomenclature:

Double hair pin inert edge staples are named in the format "Edg_T[**x]**R[**yy]**C7_DHP" where x is 1 to 4 representing the tile edge N, E, S, or W respectively, and yy is 00, 02, 04,..., 20 (eleven possible numbers) representing the eleven possible staple position along an edge in a clockwise direction looking down on a tile.

Sticky end staples are named in the format "[**z]**b[**C2][w]**T**[x]**C7R**[yy]**". z is the length of the sticky end in nucleotides (1,2 or 5). If 'C2' is present in the name, it means the sticky end is extended from the 3' end of the edge staple with a 5' stacking bond (receiver is vice versa and truncated); otherwise, the sticky end is extended from the 5' end of the edge staple with a 3' stacking bond (receiver is vice versa and truncated). If w is 'R' then the staple is a receiving staple. If w is "G[x]" then it is a giver complentary to edge x where x can be one to four representing the tile edge N, E, S, or W respectively. x is 1 to 4 representing the tile edge N, E, S, or W respectively, and yy is 00, 02, 04,..., 20 (eleven possible numbers), representing the eleven possible staple position along an edge in a clockwise direction looking down on a tile.

Special cases: names starting with "F" are Rox fluorophore strands. Names starting with TYE are TYE563 fluorophore strands. Names starting with Q are Iowa black quencher strands. Names ending in 'pipi' are staples forming two stacking bonds.

Table S1: Edge staples in the 1 nt and 2 nt kinetic measurements

Tile	Edge Staple Name	Edge Sequence
Base2nt	Edg_T1R02C7_DHP	GTGTCGTAGACACTCCCAATTCTGCGAACCCATATAAC AGTTGATGTGTCGTAGACAC
	Edg_T1R06C7_DHP	GTGTCGTAGACACCCATAAATCAAAAATCCAGAAAACG AGAATGAGTGTCGTAGACAC
	Edg_T1R10C7_DHP	GTGTCGTAGACACGAAACACCAGAACGAGAGGCTTGCC CTGACGAGTGTCGTAGACAC
	Edg_T1R14C7_DHP	GTGTCGTAGACACGAACGAGGGTAGCAACGCGAAAGAC AGCATCGGTGTCGTAGACAC
	Edg_T1R18C7_DHP	GTGTCGTAGACACGGGATTTTGCTAAACAAATGAATTT TCTGTATGTGTCGTAGACAC
	Frox_T2_R00	/5Rox_N/AGCCACCACCCTCATTGAACCGCCACCCTC AG
	2bRT2C7R02	GAGAGGGTTGATATAAGCGGATAAGTGCCG
	2bRT2C7R04	GTATAAACAGTTAATGTTGAGTAACAGTGC
	2bRT2C7R06	GCAGGTCAGACGATTGTTGACAGGAGGTTG
	2bRT2C7R08	TAGCGCGTTTTCATCGCTTTAGCGTCAGAC
	2bRT2C7R10	GCGCCAAAGACAAAAGTTCATATGGTTTAC
	2bRT2C7R12	CCGAAGCCCTTTTTAAAGCAATAGCTATCT
	2bRT2C7R14	TTTTTTGTTTAACGTCTCCAAATAAGAAAC
	2bRT2C7R16	AACCTCCCGACTTGCGGCGAGGCGTTTTAG
	2bRT2C7R18	TAAACCAAGTACCGCATTCCAAGAACGGGT
	2bRT2C7R20	AGATAAGTCCTGAACACCTGTTTATCAACA
	Edg_T3R02C7_DHP	GTGTCGTAGACACAGTAGGGCTTAATTGAAAAGCCAAC GCTCAACGTGTCGTAGACAC
	Edg_T3R06C7_DHP	GTGTCGTAGACACAGTCAATAGTGAATTTTTAAGACGC TGAGAAGGTGTCGTAGACAC
	Edg_T3R10C7_DHP	GTGTCGTAGACACCAATATAATCCTGATTGATGATGGC AATTCATGTGTCGTAGACAC
	Edg_T3R14C7_DHP	GTGTCGTAGACACACATCGCCATTAAAAAAACTGATAG CCCTAAAGTGTCGTAGACAC
	Edg_T3R18C7_DHP	GTGTCGTAGACACTTGATTAGTAATAACATTGTAGCAA TACTTCTGTGTCGTAGACAC
	Edg_T4R02C7_DHP	GTGTCGTAGACACCGGGCGCTAGGGCGCTAAGAAAGCG AAAGGAGGTGTCGTAGACAC
	Edg_T4R06C7_DHP	GTGTCGTAGACACATCCTGTTTGATGGTGGCCCCAGCA

		GGCGAAAGTGTCGTAGACAC
	Edg_T4R10C7_DHP	GTGTCGTAGACACGTAACGCCAGGGTTTTAAGGCGATTAAGTTGGGTGTCGTAGACAC
	Edg_T4R14C7_DHP	GTGTCGTAGACACTTTAAATTGTAAACGTATTGTATAAGCAAATAGTGTCGTAGACAC
	Edg_T4R18C7_DHP	GTGTCGTAGACACAAATTTTTAGAACCCTTTCAACGCAAGGATAAGTGTCGTAGACAC
Cover	Edg_T1R02C7_DHP	GTGTCGTAGACACTCCCAATTCTGCGAACCCATATAACAGTTGATGTGTCGTAGACAC
	Edg_T1R06C7_DHP	GTGTCGTAGACACCCATAAATCAAAAATCCAGAAAACGAGAATGAGTGTCGTAGACAC
	Edg_T1R10C7_DHP	GTGTCGTAGACACGAAACACCAGAACGAGAGGCTTGCCCTGACGAGTGTCGTAGACAC
	Edg_T1R14C7_DHP	GTGTCGTAGACACGAACGAGGGTAGCAACGCGAAAGACAGCATCGGTGTCGTAGACAC
	Edg_T1R18C7_DHP	GTGTCGTAGACACGGGATTTTGCTAAACAAATGAATTTTCTGTATGTGTCGTAGACAC
	Edg_T2R02C7_DHP	GTGTCGTAGACACGAGAGGGTTGATATAAGCGGATAAGTGCCGTCGTGTCGTAGACAC
	Edg_T2R06C7_DHP	GTGTCGTAGACACGCAGGTCAGACGATTGTTGACAGGAGGTTGAGGTGTCGTAGACAC
	Edg_T2R10C7_DHP	GTGTCGTAGACACGCGCCAAAGACAAAAGTTCATATGGTTTACCAGTGTCGTAGACAC
	Edg_T2R14C7_DHP	GTGTCGTAGACACTTTTTTGTTTAACGTCTCCAAATAAGAAACGAGTGTCGTAGACAC
	Edg_T2R18C7_DHP	GTGTCGTAGACACTAAACCAAGTACCGCATTCCAAGAACGGGTATGTGTCGTAGACAC
	2G2T3C7R08	TATGAGCAAAAGAAGATGATTCATTTCAATTACC
	2G2T3C7R10	CACAATATAATCCTGATTGATGATGGCAATTCAT
	2G2T3C7R12	TGGTTATCTAAAATATCTAAAGGAATTGAGGAAG
	2G2T3C7R14	AGACATCGCCATTAAAAAAACTGATAGCCCTAAA
	2G2T3C7R16	CCTCGTCTGAAATGGATTACATTTTGACGCTCAA
	2G2T3C7R18	TCTTGATTAGTAATAACATTGTAGCAATACTTCT
	Q_T3_R20	AGGAACGGTACGCCAGTAAAGGGATTTTAGAC/3IAbRQ/
	Edg_T4R02C7_DHP	GTGTCGTAGACACCGGGCGCTAGGGCGCTAAGAAAGCGAAAGGAGGTGTCGTAGACAC
	Edg_T4R06C7_DHP	GTGTCGTAGACACATCCTGTTTGATGGTGGCCCCAGCAGGCGAAAGTGTCGTAGACAC
	Edg_T4R10C7_DHP	GTGTCGTAGACACGTAACGCCAGGGTTTTAAGGCGATTAAGTTGGGTGTCGTAGACAC
	Edg_T4R14C7_DHP	GTGTCGTAGACACTTTAAATTGTAAACGTATTGTATAAGCAAATAGTGTCGTAGACAC
	Edg_T4R18C7_DHP	GTGTCGTAGACACAAATTTTTAGAACCCTTTCAACGCAAGGATAAGTGTCGTAGACAC
2nt-4th	Edg_T1R02C7_DHP	GTGTCGTAGACACTCCCAATTCTGCGAACCCATATAACAGTTGATGTGTCGTAGACAC

	Edg_T1R06C7_DHP	GTGTCGTAGACACCCATAAATCAAAAATCCAGAAAACG AGAATGAGTGTCGTAGACAC
	Edg_T1R10C7_DHP	GTGTCGTAGACACGAAACACCAGAACGAGAGGCTTGCC CTGACGAGTGTCGTAGACAC
	Edg_T1R14C7_DHP	GTGTCGTAGACACGAACGAGGGTAGCAACGCGAAAGAC AGCATCGGTGTCGTAGACAC
	Edg_T1R18C7_DHP	GTGTCGTAGACACGGGATTTTGCTAAACAAATGAATTT TCTGTATGTGTCGTAGACAC
	Edg_T2R02C7_DHP	GTGTCGTAGACACGAGAGGGTTGATATAAGCGGATAAG TGCCGTCGTGTCGTAGACAC
	Edg_T2R06C7_DHP	GTGTCGTAGACACGCAGGTCAGACGATTGTTGACAGGA GGTTGAGGTGTCGTAGACAC
	Edg_T2R10C7_DHP	GTGTCGTAGACACGCGCCAAAGACAAAAGTTCATATGG TTTACCAGTGTCGTAGACAC
	Edg_T2R14C7_DHP	GTGTCGTAGACACTTTTTTGTTTAACGTCTCCAAATAA GAAACGAGTGTCGTAGACAC
	Edg_T2R18C7_DHP	GTGTCGTAGACACTAAACCAAGTACCGCATTCCAAGAA CGGGTATGTGTCGTAGACAC
	2G2T3C7R00	ATGTAAAGTAATTCTGTCAAAGTACCGACAAAAG
	2G2T3C7R02	ATAGTAGGGCTTAATTGAAAAGCCAACGCTCAAC
	2G2T3C7R04	CGAATGGTTTGAAATACCCTTCTGACCTAAATTT
	2G2T3C7R06	GAAGTCAATAGTGAATTTTTAAGACGCTGAGAAG
	2G2T3C7R08	TATGAGCAAAAGAAGATGATTCATTTCAATTACC
	2G2T3C7R10	CACAATATAATCCTGATTGATGATGGCAATTCAT
	2G2T3C7R12	TGGTTATCTAAAATATCTAAAGGAATTGAGGAAG
	2G2T3C7R14	AGACATCGCCATTAAAAAAACTGATAGCCCTAAA
	2G2T3C7R16	CCTCGTCTGAAATGGATTACATTTTGACGCTCAA
	2G2T3C7R18	TCTTGATTAGTAATAACATTGTAGCAATACTTCT
	Edg_T3C7R20_pipi	AGGAACGGTACGCCAGTAAAGGGATTTTAGAC
	Edg_T4R02C7_DHP	GTGTCGTAGACACCGGGCGCTAGGGCGCTAAGAAAGCG AAAGGAGGTGTCGTAGACAC
	Edg_T4R06C7_DHP	GTGTCGTAGACACATCCTGTTTGATGGTGGCCCCAGCA GGCGAAAGTGTCGTAGACAC
	Edg_T4R10C7_DHP	GTGTCGTAGACACGTAACGCCAGGGTTTTAAGGCGATT AAGTTGGGTGTCGTAGACAC
	Edg_T4R14C7_DHP	GTGTCGTAGACACTTTAAATTGTAAACGTATTGTATAA GCAAATAGTGTCGTAGACAC
	Edg_T4R18C7_DHP	GTGTCGTAGACACAAATTTTTAGAACCCTTTCAACGCA AGGATAAGTGTCGTAGACAC
Base1nt	Edg_T1R02C7_DHP	GTGTCGTAGACACTCCCAATTCTGCGAACCCATATAAC AGTTGATGTGTCGTAGACAC
	Edg_T1R06C7_DHP	GTGTCGTAGACACCCATAAATCAAAAATCCAGAAAACG AGAATGAGTGTCGTAGACAC
	Edg_T1R10C7_DHP	GTGTCGTAGACACGAAACACCAGAACGAGAGGCTTGCC CTGACGAGTGTCGTAGACAC
	Edg_T1R14C7_DHP	GTGTCGTAGACACGAACGAGGGTAGCAACGCGAAAGAC

		AGCATCGGTGTCGTAGACAC
	Edg_T1R18C7_DHP	GTGTCGTAGACACGGGATTTTGCTAAACAAATGAATTTTCTGTATGTGTCGTAGACAC
	Frox_T2_R00	/5Rox_N/AGCCACCACCCTCATTGAACCGCCACCCTCAG
	2bRT2C7R02	GAGAGGGTTGATATAAGCGGATAAGTGCCG
	2bRT2C7R04	GTATAAACAGTTAATGTTGAGTAACAGTGC
	2bRT2C7R06	GCAGGTCAGACGATTGTTGACAGGAGGTTG
	2bRT2C7R08	TAGCGCGTTTTCATCGCTTTAGCGTCAGAC
	2bRT2C7R10	GCGCCAAAGACAAAAGTTCATATGGTTTAC
	2bRT2C7R12	CCGAAGCCCTTTTTAAAGCAATAGCTATCT
	1bRT2C7R14	TTTTTTGTTTAACGTCTCCAAATAAGAAACG
	1bRT2C7R16	AACCTCCCGACTTGCGGCGAGGCGTTTTAGC
	1bRT2C7R18	TAAACCAAGTACCGCATTCCAAGAACGGGTA
	1bRT2C7R20	AGATAAGTCCTGAACACCTGTTTATCAACAA
	Edg_T3R02C7_DHP	GTGTCGTAGACACAGTAGGGCTTAATTGAAAAGCCAACGCTCAACGTGTCGTAGACAC
	Edg_T3R06C7_DHP	GTGTCGTAGACACAGTCAATAGTGAATTTTTAAGACGCTGAGAAGGTGTCGTAGACAC
	Edg_T3R10C7_DHP	GTGTCGTAGACACCAATATAATCCTGATTGATGATGGCAATTCATGTGTCGTAGACAC
	Edg_T3R14C7_DHP	GTGTCGTAGACACACATCGCCATTAAAAAAACTGATAGCCCTAAAGTGTCGTAGACAC
	Edg_T3R18C7_DHP	GTGTCGTAGACACTTGATTAGTAATAACATTGTAGCAATACTTCTGTGTCGTAGACAC
	Edg_T4R02C7_DHP	GTGTCGTAGACACCGGGCGCTAGGGCGCTAAGAAAGCGAAAGGAGGTGTCGTAGACAC
	Edg_T4R06C7_DHP	GTGTCGTAGACACATCCTGTTTGATGGTGGCCCCAGCAGGCGAAAGTGTCGTAGACAC
	Edg_T4R10C7_DHP	GTGTCGTAGACACGTAACGCCAGGGTTTTAAGGCGATTAAGTTGGGTGTCGTAGACAC
	Edg_T4R14C7_DHP	GTGTCGTAGACACTTTAAATTGTAAACGTATTGTATAAGCAAATAGTGTCGTAGACAC
	Edg_T4R18C7_DHP	GTGTCGTAGACACAAATTTTTAGAACCCTTTCAACGCAAGGATAAGTGTCGTAGACAC
0Toe	Edg_T1R02C7_DHP	GTGTCGTAGACACTCCCAATTCTGCGAACCCATATAACAGTTGATGTGTCGTAGACAC
	Edg_T1R06C7_DHP	GTGTCGTAGACACCCATAAATCAAAAATCCAGAAAACGAGAATGAGTGTCGTAGACAC
	Edg_T1R10C7_DHP	GTGTCGTAGACACGAAACACCAGAACGAGAGGCTTGCCCTGACGAGTGTCGTAGACAC
	Edg_T1R14C7_DHP	GTGTCGTAGACACGAACGAGGGTAGCAACGCGAAAGACAGCATCGGTGTCGTAGACAC
	Edg_T1R18C7_DHP	GTGTCGTAGACACGGGATTTTGCTAAACAAATGAATTTTCTGTATGTGTCGTAGACAC
	Edg_T2R02C7_DHP	GTGTCGTAGACACGAGAGGGTTGATATAAGCGGATAAG

		TGCCGTCGTGTCGTAGACAC
	Edg_T2R06C7_DHP	GTGTCGTAGACACGCAGGTCAGACGATTGTTGACAGGA GGTTGAGGTGTCGTAGACAC
	Edg_T2R10C7_DHP	GTGTCGTAGACACGCGCCAAAGACAAAAGTTCATATGG TTTACCAGTGTCGTAGACAC
	Edg_T2R14C7_DHP	GTGTCGTAGACACTTTTTTGTTTAACGTCTCCAAATAA GAAACGAGTGTCGTAGACAC
	Edg_T2R18C7_DHP	GTGTCGTAGACACTAAACCAAGTACCGCATTCCAAGAA CGGGTATGTGTCGTAGACAC
	2G2T3C7R08	TATGAGCAAAAGAAGATGATTCATTTCAATTACC
	2G2T3C7R10	CACAATATAATCCTGATTGATGATGGCAATTCAT
	2G2T3C7R12	TGGTTATCTAAAATATCTAAAGGAATTGAGGAAG
	2G2T3C7R14	AGACATCGCCATTAAAAAAACTGATAGCCCTAAA
	2G2T3C7R16	CCTCGTCTGAAATGGATTACATTTTGACGCTCAA
	2G2T3C7R18	TCTTGATTAGTAATAACATTGTAGCAATACTTCT
	Edg_T3C7R20_pipi	AGGAACGGTACGCCAGTAAAGGGATTTTAGAC
	Edg_T4R02C7_DHP	GTGTCGTAGACACCGGGCGCTAGGGCGCTAAGAAAGCG AAAGGAGGTGTCGTAGACAC
	Edg_T4R06C7_DHP	GTGTCGTAGACACATCCTGTTTGATGGTGGCCCCAGCA GGCGAAAGTGTCGTAGACAC
	Edg_T4R10C7_DHP	GTGTCGTAGACACGTAACGCCAGGGTTTTAAGGCGATT AAGTTGGGTGTCGTAGACAC
	Edg_T4R14C7_DHP	GTGTCGTAGACACTTTAAATTGTAAACGTATTGTATAA GCAAATAGTGTCGTAGACAC
	Edg_T4R18C7_DHP	GTGTCGTAGACACAAATTTTTAGAACCCTTTCAACGCA AGGATAAGTGTCGTAGACAC
2nt-3th	Edg_T1R02C7_DHP	GTGTCGTAGACACTCCCAATTCTGCGAACCCATATAAC AGTTGATGTGTCGTAGACAC
	Edg_T1R06C7_DHP	GTGTCGTAGACACCCATAAATCAAAAATCCAGAAAACG AGAATGAGTGTCGTAGACAC
	Edg_T1R10C7_DHP	GTGTCGTAGACACGAAACACCAGAACGAGAGGCTTGCC CTGACGAGTGTCGTAGACAC
	Edg_T1R14C7_DHP	GTGTCGTAGACACGAACGAGGGTAGCAACGCGAAAGAC AGCATCGGTGTCGTAGACAC
	Edg_T1R18C7_DHP	GTGTCGTAGACACGGGATTTTGCTAAACAAATGAATTT TCTGTATGTGTCGTAGACAC
	Edg_T2R02C7_DHP	GTGTCGTAGACACGAGAGGGTTGATATAAGCGGATAAG TGCCGTCGTGTCGTAGACAC
	Edg_T2R06C7_DHP	GTGTCGTAGACACGCAGGTCAGACGATTGTTGACAGGA GGTTGAGGTGTCGTAGACAC
	Edg_T2R10C7_DHP	GTGTCGTAGACACGCGCCAAAGACAAAAGTTCATATGG TTTACCAGTGTCGTAGACAC
	Edg_T2R14C7_DHP	GTGTCGTAGACACTTTTTTGTTTAACGTCTCCAAATAA GAAACGAGTGTCGTAGACAC
	Edg_T2R18C7_DHP	GTGTCGTAGACACTAAACCAAGTACCGCATTCCAAGAA CGGGTATGTGTCGTAGACAC

	2G2T3C7R02	ATAGTAGGGCTTAATTGAAAAGCCAACGCTCAAC
	2G2T3C7R04	CGAATGGTTTGAAATACCCTTCTGACCTAAATTT
	2G2T3C7R06	GAAGTCAATAGTGAATTTTTAAGACGCTGAGAAG
	2G2T3C7R08	TATGAGCAAAAGAAGATGATTCATTTCAATTACC
	2G2T3C7R10	CACAATATAATCCTGATTGATGATGGCAATTCAT
	2G2T3C7R12	TGGTTATCTAAAATATCTAAAGGAATTGAGGAAG
	2G2T3C7R14	AGACATCGCCATTAAAAAAACTGATAGCCCTAAA
	2G2T3C7R16	CCTCGTCTGAAATGGATTACATTTTGACGCTCAA
	2G2T3C7R18	TCTTGATTAGTAATAACATTGTAGCAATACTTCT
	Edg_T3C7R20_pipi	AGGAACGGTACGCCAGTAAAGGGATTTTAGAC
	Edg_T4R02C7_DHP	GTGTCGTAGACACCGGGCGCTAGGGCGCTAAGAAAGCG AAAGGAGGTGTCGTAGACAC
	Edg_T4R06C7_DHP	GTGTCGTAGACACATCCTGTTTGATGGTGGCCCCAGCA GGCGAAAGTGTCGTAGACAC
	Edg_T4R10C7_DHP	GTGTCGTAGACACGTAACGCCAGGGTTTTAAGGCGATT AAGTTGGGTGTCGTAGACAC
	Edg_T4R14C7_DHP	GTGTCGTAGACACTTTAAATTGTAAACGTATTGTATAA GCAAATAGTGTCGTAGACAC
	Edg_T4R18C7_DHP	GTGTCGTAGACACAAATTTTTAGAACCCTTTCAACGCA AGGATAAGTGTCGTAGACAC
2nt-2th	Edg_T1R02C7_DHP	GTGTCGTAGACACTCCCAATTCTGCGAACCCATATAAC AGTTGATGTGTCGTAGACAC
	Edg_T1R06C7_DHP	GTGTCGTAGACACCCATAAATCAAAAATCCAGAAAACG AGAATGAGTGTCGTAGACAC
	Edg_T1R10C7_DHP	GTGTCGTAGACACGAAACACCAGAACGAGAGGCTTGCC CTGACGAGTGTCGTAGACAC
	Edg_T1R14C7_DHP	GTGTCGTAGACACGAACGAGGGTAGCAACGCGAAAGAC AGCATCGGTGTCGTAGACAC
	Edg_T1R18C7_DHP	GTGTCGTAGACACGGGATTTTGCTAAACAAATGAATTT TCTGTATGTGTCGTAGACAC
	Edg_T2R02C7_DHP	GTGTCGTAGACACGAGAGGGTTGATATAAGCGGATAAG TGCCGTCGTGTCGTAGACAC
	Edg_T2R06C7_DHP	GTGTCGTAGACACGCAGGTCAGACGATTGTTGACAGGA GGTTGAGGTGTCGTAGACAC
	Edg_T2R10C7_DHP	GTGTCGTAGACACGCGCCAAAGACAAAAGTTCATATGG TTTACCAGTGTCGTAGACAC
	Edg_T2R14C7_DHP	GTGTCGTAGACACTTTTTTGTTTAACGTCTCCAAATAA GAAACGAGTGTCGTAGACAC
	Edg_T2R18C7_DHP	GTGTCGTAGACACTAAACCAAGTACCGCATTCCAAGAA CGGGTATGTGTCGTAGACAC
	2G2T3C7R04	CGAATGGTTTGAAATACCCTTCTGACCTAAATTT
	2G2T3C7R06	GAAGTCAATAGTGAATTTTTAAGACGCTGAGAAG
	2G2T3C7R08	TATGAGCAAAAGAAGATGATTCATTTCAATTACC
	2G2T3C7R10	CACAATATAATCCTGATTGATGATGGCAATTCAT
	2G2T3C7R12	TGGTTATCTAAAATATCTAAAGGAATTGAGGAAG

	2G2T3C7R14	AGACATCGCCATTAAAAAAACTGATAGCCCTAAA
	2G2T3C7R16	CCTCGTCTGAAATGGATTACATTTTGACGCTCAA
	2G2T3C7R18	TCTTGATTAGTAATAACATTGTAGCAATACTTCT
	Edg_T3C7R20_pipi	AGGAACGGTACGCCAGTAAAGGGATTTTAGAC
	Edg_T4R02C7_DHP	GTGTCGTAGACACCGGGCGCTAGGGCGCTAAGAAAGCG AAAGGAGGTGTCGTAGACAC
	Edg_T4R06C7_DHP	GTGTCGTAGACACATCCTGTTTGATGGTGGCCCCAGCA GGCGAAAGTGTCGTAGACAC
	Edg_T4R10C7_DHP	GTGTCGTAGACACGTAACGCCAGGGTTTTAAGGCGATT AAGTTGGGTGTCGTAGACAC
	Edg_T4R14C7_DHP	GTGTCGTAGACACTTTAAATTGTAAACGTATTGTATAA GCAAATAGTGTCGTAGACAC
	Edg_T4R18C7_DHP	GTGTCGTAGACACAAATTTTTAGAACCCTTTCAACGCA AGGATAAGTGTCGTAGACAC
2nt-1th	Edg_T1R02C7_DHP	GTGTCGTAGACACTCCCAATTCTGCGAACCCATATAAC AGTTGATGTGTCGTAGACAC
	Edg_T1R06C7_DHP	GTGTCGTAGACACCCATAAATCAAAAATCCAGAAAACG AGAATGAGTGTCGTAGACAC
	Edg_T1R10C7_DHP	GTGTCGTAGACACGAAACACCAGAACGAGAGGCTTGCC CTGACGAGTGTCGTAGACAC
	Edg_T1R14C7_DHP	GTGTCGTAGACACGAACGAGGGTAGCAACGCGAAAGAC AGCATCGGTGTCGTAGACAC
	Edg_T1R18C7_DHP	GTGTCGTAGACACGGGATTTTGCTAAACAAATGAATTT TCTGTATGTGTCGTAGACAC
	Edg_T2R02C7_DHP	GTGTCGTAGACACGAGAGGGTTGATATAAGCGGATAAG TGCCGTCGTGTCGTAGACAC
	Edg_T2R06C7_DHP	GTGTCGTAGACACGCAGGTCAGACGATTGTTGACAGGA GGTTGAGGTGTCGTAGACAC
	Edg_T2R10C7_DHP	GTGTCGTAGACACGCGCCAAAGACAAAAGTTCATATGG TTTACCAGTGTCGTAGACAC
	Edg_T2R14C7_DHP	GTGTCGTAGACACTTTTTTGTTTAACGTCTCCAAATAA GAAACGAGTGTCGTAGACAC
	Edg_T2R18C7_DHP	GTGTCGTAGACACTAAACCAAGTACCGCATTCCAAGAA CGGGTATGTGTCGTAGACAC
	2G2T3C7R06	GAAGTCAATAGTGAATTTTTAAGACGCTGAGAAG
	2G2T3C7R08	TATGAGCAAAAGAAGATGATTCATTTCAATTACC
	2G2T3C7R10	CACAATATAATCCTGATTGATGATGGCAATTCAT
	2G2T3C7R12	TGGTTATCTAAAATATCTAAAGGAATTGAGGAAG
	2G2T3C7R14	AGACATCGCCATTAAAAAAACTGATAGCCCTAAA
	2G2T3C7R16	CCTCGTCTGAAATGGATTACATTTTGACGCTCAA
	2G2T3C7R18	TCTTGATTAGTAATAACATTGTAGCAATACTTCT
	Edg_T3C7R20_pipi	AGGAACGGTACGCCAGTAAAGGGATTTTAGAC
	Edg_T4R02C7_DHP	GTGTCGTAGACACCGGGCGCTAGGGCGCTAAGAAAGCG AAAGGAGGTGTCGTAGACAC
	Edg_T4R06C7_DHP	GTGTCGTAGACACATCCTGTTTGATGGTGGCCCCAGCA GGCGAAAGTGTCGTAGACAC

	Edg_T4R10C7_DHP	GTGTCGTAGACACGTAACGCCAGGGTTTTAAGGCGATT AAGTTGGGTGTCGTAGACAC
	Edg_T4R14C7_DHP	GTGTCGTAGACACTTTAAATTGTAAACGTATTGTATAA GCAAATAGTGTCGTAGACAC
	Edg_T4R18C7_DHP	GTGTCGTAGACACAAATTTTTAGAACCCTTTCAACGCA AGGATAAGTGTCGTAGACAC
1nt-4th	Edg_T1R02C7_DHP	GTGTCGTAGACACTCCCAATTCTGCGAACCCATATAAC AGTTGATGTGTCGTAGACAC
	Edg_T1R06C7_DHP	GTGTCGTAGACACCCATAAATCAAAAATCCAGAAAACG AGAATGAGTGTCGTAGACAC
	Edg_T1R10C7_DHP	GTGTCGTAGACACGAAACACCAGAACGAGAGGCTTGCC CTGACGAGTGTCGTAGACAC
	Edg_T1R14C7_DHP	GTGTCGTAGACACGAACGAGGGTAGCAACGCGAAAGAC AGCATCGGTGTCGTAGACAC
	Edg_T1R18C7_DHP	GTGTCGTAGACACGGGATTTTGCTAAACAAATGAATTT TCTGTATGTGTCGTAGACAC
	Edg_T2R02C7_DHP	GTGTCGTAGACACGAGAGGGTTGATATAAGCGGATAAG TGCCGTCGTGTCGTAGACAC
	Edg_T2R06C7_DHP	GTGTCGTAGACACGCAGGTCAGACGATTGTTGACAGGA GGTTGAGGTGTCGTAGACAC
	Edg_T2R10C7_DHP	GTGTCGTAGACACGCGCCAAAGACAAAAGTTCATATGG TTTACCAGTGTCGTAGACAC
	Edg_T2R14C7_DHP	GTGTCGTAGACACTTTTTTGTTTAACGTCTCCAAATAA GAAACGAGTGTCGTAGACAC
	Edg_T2R18C7_DHP	GTGTCGTAGACACTAAACCAAGTACCGCATTCCAAGAA CGGGTATGTGTCGTAGACAC
	1bG2T3C7R00	TGTAAAGTAATTCTGTCAAAGTACCGACAAAAG
	1bG2T3C7R02	TAGTAGGGCTTAATTGAAAAGCCAACGCTCAAC
	1bG2T3C7R04	GAATGGTTTGAAATACCCTTCTGACCTAAATTT
	1bG2T3C7R06	AAGTCAATAGTGAATTTTTAAGACGCTGAGAAG
	2G2T3C7R08	TATGAGCAAAAGAAGATGATTCATTTCAATTACC
	2G2T3C7R10	CACAATATAATCCTGATTGATGATGGCAATTCAT
	2G2T3C7R12	TGGTTATCTAAAATATCTAAAGGAATTGAGGAAG
	2G2T3C7R14	AGACATCGCCATTAAAAAAACTGATAGCCCTAAA
	2G2T3C7R16	CCTCGTCTGAAATGGATTACATTTTGACGCTCAA
	2G2T3C7R18	TCTTGATTAGTAATAACATTGTAGCAATACTTCT
	Edg_T3C7R20_pipi	AGGAACGGTACGCCAGTAAAGGGATTTTAGAC
	Edg_T4R02C7_DHP	GTGTCGTAGACACCGGGCGCTAGGGCGCTAAGAAAGCG AAAGGAGGTGTCGTAGACAC
	Edg_T4R06C7_DHP	GTGTCGTAGACACATCCTGTTTGATGGTGGCCCCAGCA GGCGAAAGTGTCGTAGACAC
	Edg_T4R10C7_DHP	GTGTCGTAGACACGTAACGCCAGGGTTTTAAGGCGATT AAGTTGGGTGTCGTAGACAC
	Edg_T4R14C7_DHP	GTGTCGTAGACACTTTAAATTGTAAACGTATTGTATAA GCAAATAGTGTCGTAGACAC
	Edg_T4R18C7_DHP	GTGTCGTAGACACAAATTTTTAGAACCCTTTCAACGCA

		AGGATAAGTGTCGTAGACAC
1nt-3th	Edg_T1R02C7_DHP	GTGTCGTAGACACTCCCAATTCTGCGAACCCATATAAC AGTTGATGTGTCGTAGACAC
	Edg_T1R06C7_DHP	GTGTCGTAGACACCCATAAATCAAAAATCCAGAAAACG AGAATGAGTGTCGTAGACAC
	Edg_T1R10C7_DHP	GTGTCGTAGACACGAAACACCAGAACGAGAGGCTTGCC CTGACGAGTGTCGTAGACAC
	Edg_T1R14C7_DHP	GTGTCGTAGACACGAACGAGGGTAGCAACGCGAAAGAC AGCATCGGTGTCGTAGACAC
	Edg_T1R18C7_DHP	GTGTCGTAGACACGGGATTTTGCTAAACAAATGAATTT TCTGTATGTGTCGTAGACAC
	Edg_T2R02C7_DHP	GTGTCGTAGACACGAGAGGGTTGATATAAGCGGATAAG TGCCGTCGTGTCGTAGACAC
	Edg_T2R06C7_DHP	GTGTCGTAGACACGCAGGTCAGACGATTGTTGACAGGA GGTTGAGGTGTCGTAGACAC
	Edg_T2R10C7_DHP	GTGTCGTAGACACGCGCCAAAGACAAAAGTTCATATGG TTTACCAGTGTCGTAGACAC
	Edg_T2R14C7_DHP	GTGTCGTAGACACTTTTTTGTTTAACGTCTCCAAATAA GAAACGAGTGTCGTAGACAC
	Edg_T2R18C7_DHP	GTGTCGTAGACACTAAACCAAGTACCGCATTCCAAGAA CGGGTATGTGTCGTAGACAC
	1bG2T3C7R02	TAGTAGGGCTTAATTGAAAAGCCAACGCTCAAC
	1bG2T3C7R04	GAATGGTTTGAAATACCCTTCTGACCTAAATTT
	1bG2T3C7R06	AAGTCAATAGTGAATTTTTAAGACGCTGAGAAG
	2G2T3C7R08	TATGAGCAAAAGAAGATGATTCATTTCAATTACC
	2G2T3C7R10	CACAATATAATCCTGATTGATGATGGCAATTCAT
	2G2T3C7R12	TGGTTATCTAAAATATCTAAAGGAATTGAGGAAG
	2G2T3C7R14	AGACATCGCCATTAAAAAAACTGATAGCCCTAAA
	2G2T3C7R16	CCTCGTCTGAAATGGATTACATTTTGACGCTCAA
	2G2T3C7R18	TCTTGATTAGTAATAACATTGTAGCAATACTTCT
	Edg_T3C7R20_pipi	AGGAACGGTACGCCAGTAAAGGGATTTTAGAC
	Edg_T4R02C7_DHP	GTGTCGTAGACACCGGGCGCTAGGGCGCTAAGAAAGCG AAAGGAGGTGTCGTAGACAC
	Edg_T4R06C7_DHP	GTGTCGTAGACACATCCTGTTTGATGGTGGCCCCAGCA GGCGAAAGTGTCGTAGACAC
	Edg_T4R10C7_DHP	GTGTCGTAGACACGTAACGCCAGGGTTTTAAGGCGATT AAGTTGGGTGTCGTAGACAC
	Edg_T4R14C7_DHP	GTGTCGTAGACACTTTAAATTGTAAACGTATTGTATAA GCAAATAGTGTCGTAGACAC
	Edg_T4R18C7_DHP	GTGTCGTAGACACAAATTTTTAGAACCCTTTCAACGCA AGGATAAGTGTCGTAGACAC
1nt-2th	Edg_T1R02C7_DHP	GTGTCGTAGACACTCCCAATTCTGCGAACCCATATAAC AGTTGATGTGTCGTAGACAC
	Edg_T1R06C7_DHP	GTGTCGTAGACACCCATAAATCAAAAATCCAGAAAACG AGAATGAGTGTCGTAGACAC
	Edg_T1R10C7_DHP	GTGTCGTAGACACGAAACACCAGAACGAGAGGCTTGCC

		CTGACGAGTGTCGTAGACAC
	Edg_T1R14C7_DHP	GTGTCGTAGACACGAACGAGGGTAGCAACGCGAAAGAC AGCATCGGTGTCGTAGACAC
	Edg_T1R18C7_DHP	GTGTCGTAGACACGGGATTTTGCTAAACAAATGAATTT TCTGTATGTGTCGTAGACAC
	Edg_T2R02C7_DHP	GTGTCGTAGACACGAGAGGGTTGATATAAGCGGATAAG TGCCGTCGTGTCGTAGACAC
	Edg_T2R06C7_DHP	GTGTCGTAGACACGCAGGTCAGACGATTGTTGACAGGA GGTTGAGGTGTCGTAGACAC
	Edg_T2R10C7_DHP	GTGTCGTAGACACGCGCCAAAGACAAAAGTTCATATGG TTTACCAGTGTCGTAGACAC
	Edg_T2R14C7_DHP	GTGTCGTAGACACTTTTTTGTTTAACGTCTCCAAATAA GAAACGAGTGTCGTAGACAC
	Edg_T2R18C7_DHP	GTGTCGTAGACACTAAACCAAGTACCGCATTCCAAGAA CGGGTATGTGTCGTAGACAC
	1bG2T3C7R04	GAATGGTTTGAAATACCCTTCTGACCTAAATTT
	1bG2T3C7R06	AAGTCAATAGTGAATTTTTAAGACGCTGAGAAG
	2G2T3C7R08	TATGAGCAAAAGAAGATGATTCATTTCAATTACC
	2G2T3C7R10	CACAATATAATCCTGATTGATGATGGCAATTCAT
	2G2T3C7R12	TGGTTATCTAAAATATCTAAAGGAATTGAGGAAG
	2G2T3C7R14	AGACATCGCCATTAAAAAAACTGATAGCCCTAAA
	2G2T3C7R16	CCTCGTCTGAAATGGATTACATTTTGACGCTCAA
	2G2T3C7R18	TCTTGATTAGTAATAACATTGTAGCAATACTTCT
	Edg_T3C7R20_pipi	AGGAACGGTACGCCAGTAAAGGGATTTTAGAC
	Edg_T4R02C7_DHP	GTGTCGTAGACACCGGGCGCTAGGGCGCTAAGAAAGCG AAAGGAGGTGTCGTAGACAC
	Edg_T4R06C7_DHP	GTGTCGTAGACACATCCTGTTTGATGGTGGCCCCAGCA GGCGAAAGTGTCGTAGACAC
	Edg_T4R10C7_DHP	GTGTCGTAGACACGTAACGCCAGGGTTTTAAGGCGATT AAGTTGGGTGTCGTAGACAC
	Edg_T4R14C7_DHP	GTGTCGTAGACACTTTAAATTGTAAACGTATTGTATAA GCAAATAGTGTCGTAGACAC
	Edg_T4R18C7_DHP	GTGTCGTAGACACAAATTTTTAGAACCCTTTCAACGCA AGGATAAGTGTCGTAGACAC
1nt-1th	Edg_T1R02C7_DHP	GTGTCGTAGACACTCCCAATTCTGCGAACCCATATAAC AGTTGATGTGTCGTAGACAC
	Edg_T1R06C7_DHP	GTGTCGTAGACACCCATAAATCAAAAATCCAGAAAACG AGAATGAGTGTCGTAGACAC
	Edg_T1R10C7_DHP	GTGTCGTAGACACGAAACACCAGAACGAGAGGCTTGCC CTGACGAGTGTCGTAGACAC
	Edg_T1R14C7_DHP	GTGTCGTAGACACGAACGAGGGTAGCAACGCGAAAGAC AGCATCGGTGTCGTAGACAC
	Edg_T1R18C7_DHP	GTGTCGTAGACACGGGATTTTGCTAAACAAATGAATTT TCTGTATGTGTCGTAGACAC
	Edg_T2R02C7_DHP	GTGTCGTAGACACGAGAGGGTTGATATAAGCGGATAAG TGCCGTCGTGTCGTAGACAC

	Edg_T2R06C7_DHP	GTGTCGTAGACACGCAGGTCAGACGATTGTTGACAGGA GGTTGAGGTGTCGTAGACAC
	Edg_T2R10C7_DHP	GTGTCGTAGACACGCGCCAAAGACAAAAGTTCATATGG TTTACCAGTGTCGTAGACAC
	Edg_T2R14C7_DHP	GTGTCGTAGACACTTTTTTGTTTAACGTCTCCAAATAA GAAACGAGTGTCGTAGACAC
	Edg_T2R18C7_DHP	GTGTCGTAGACACTAAACCAAGTACCGCATTCCAAGAA CGGGTATGTGTCGTAGACAC
	1bG2T3C7R06	AAGTCAATAGTGAATTTTTAAGACGCTGAGAAG
	2G2T3C7R08	TATGAGCAAAAGAAGATGATTCATTTCAATTACC
	2G2T3C7R10	CACAATATAATCCTGATTGATGATGGCAATTCAT
	2G2T3C7R12	TGGTTATCTAAAATATCTAAAGGAATTGAGGAAG
	2G2T3C7R14	AGACATCGCCATTAAAAAAACTGATAGCCCTAAA
	2G2T3C7R16	CCTCGTCTGAAATGGATTACATTTTGACGCTCAA
	2G2T3C7R18	TCTTGATTAGTAATAACATTGTAGCAATACTTCT
	Edg_T3C7R20_pipi	AGGAACGGTACGCCAGTAAAGGGATTTTAGAC
	Edg_T4R02C7_DHP	GTGTCGTAGACACCGGGCGCTAGGGCGCTAAGAAAGCG AAAGGAGGTGTCGTAGACAC
	Edg_T4R06C7_DHP	GTGTCGTAGACACATCCTGTTTGATGGTGGCCCCAGCA GGCGAAAGTGTCGTAGACAC
	Edg_T4R10C7_DHP	GTGTCGTAGACACGTAACGCCAGGGTTTTAAGGCGATT AAGTTGGGTGTCGTAGACAC
	Edg_T4R14C7_DHP	GTGTCGTAGACACTTTAAATTGTAAACGTATTGTATAA GCAAATAGTGTCGTAGACAC
	Edg_T4R18C7_DHP	GTGTCGTAGACACAAATTTTTAGAACCCTTTCAACGCA AGGATAAGTGTCGTAGACAC

Table S2: Edge staples in competitive reconfiguration measurements

Tile	Edge	Edge Seq
Base2nt Rox	Edg_T1R02C7_DHP	GTGTCGTAGACACTCCCAATTCTGCGAACCCATATAACA GTTGATGTGTCGTAGACAC
	Edg_T1R06C7_DHP	GTGTCGTAGACACCCATAAATCAAAAATCCAGAAAACGA GAATGAGTGTCGTAGACAC
	Edg_T1R10C7_DHP	GTGTCGTAGACACGAAACACCAGAACGAGAGGCTTGCCC TGACGAGTGTCGTAGACAC
	Edg_T1R14C7_DHP	GTGTCGTAGACACGAACGAGGGTAGCAACGCGAAAGACA GCATCGGTGTCGTAGACAC
	Edg_T1R18C7_DHP	GTGTCGTAGACACGGGATTTTGCTAAACAAATGAATTTT CTGTATGTGTCGTAGACAC
	F_rox_v1_T2_R00	/5Rox_N/AGCCACCACCCTCATTGAACCGCCACCCTCA G
	2bRT2C7R02	GAGAGGGTTGATATAAGCGGATAAGTGCCG

	2bRT2C7R04	GTATAAACAGTTAATGTTGAGTAACAGTGC
	2bRT2C7R06	GCAGGTCAGACGATTGTTGACAGGAGGTTG
	2bRT2C7R08	TAGCGCGTTTTCATCGCTTTAGCGTCAGAC
	2bRT2C7R10	GCGCCAAAGACAAAAGTTCATATGGTTTAC
	2bRT2C7R12	CCGAAGCCCTTTTTAAAGCAATAGCTATCT
	2bRT2C7R14	TTTTTTGTTTAACGTCTCCAAATAAGAAAC
	2bRT2C7R16	AACCTCCCGACTTGCGGCGAGGCGTTTTAG
	2bRT2C7R18	TAAACCAAGTACCGCATTCCAAGAACGGGT
	2bRT2C7R20	AGATAAGTCCTGAACACCTGTTTATCAACA
	Edg_T3R02C7_DHP	GTGTCGTAGACACAGTAGGGCTTAATTGAAAAGCCAACG CTCAACGTGTCGTAGACAC
	Edg_T3R06C7_DHP	GTGTCGTAGACACAGTCAATAGTGAATTTTTAAGACGCT GAGAAGGTGTCGTAGACAC
	Edg_T3R10C7_DHP	GTGTCGTAGACACCAATATAATCCTGATTGATGATGGCA ATTCATGTGTCGTAGACAC
	Edg_T3R14C7_DHP	GTGTCGTAGACACACATCGCCATTAAAAAAACTGATAGC CCTAAAGTGTCGTAGACAC
	Edg_T3R18C7_DHP	GTGTCGTAGACACTTGATTAGTAATAACATTGTAGCAAT ACTTCTGTGTCGTAGACAC
	Edg_T4R02C7_DHP	GTGTCGTAGACACCGGGCGCTAGGGCGCTAAGAAAGCGA AAGGAGGTGTCGTAGACAC
	Edg_T4R06C7_DHP	GTGTCGTAGACACATCCTGTTTGATGGTGGCCCCAGCAG GCGAAAGTGTCGTAGACAC
	Edg_T4R10C7_DHP	GTGTCGTAGACACGTAACGCCAGGGTTTTAAGGCGATTA AGTTGGGTGTCGTAGACAC
	Edg_T4R14C7_DHP	GTGTCGTAGACACTTTAAATTGTAAACGTATTGTATAAG CAAATAGTGTCGTAGACAC
	Edg_T4R18C7_DHP	GTGTCGTAGACACAAATTTTTAGAACCCTTTCAACGCAA GGATAAGTGTCGTAGACAC
Cover_t hres	Edg_T1R02C7_DHP	GTGTCGTAGACACTCCCAATTCTGCGAACCCATATAACA GTTGATGTGTCGTAGACAC
	Edg_T1R06C7_DHP	GTGTCGTAGACACCCATAAATCAAAAATCCAGAAAACGA GAATGAGTGTCGTAGACAC
	Edg_T1R10C7_DHP	GTGTCGTAGACACGAAACACCAGAACGAGAGGCTTGCCC TGACGAGTGTCGTAGACAC
	Edg_T1R14C7_DHP	GTGTCGTAGACACGAACGAGGGTAGCAACGCGAAAGACA GCATCGGTGTCGTAGACAC
	Edg_T1R18C7_DHP	GTGTCGTAGACACGGGATTTTGCTAAACAAATGAATTTT CTGTATGTGTCGTAGACAC
	Edg_T2R02C7_DHP	GTGTCGTAGACACGAGAGGGTTGATATAAGCGGATAAGT GCCGTCGTGTCGTAGACAC
	Edg_T2R06C7_DHP	GTGTCGTAGACACGCAGGTCAGACGATTGTTGACAGGAG GTTGAGGTGTCGTAGACAC
	Edg_T2R10C7_DHP	GTGTCGTAGACACGCGCCAAAGACAAAAGTTCATATGGT TTACCAGTGTCGTAGACAC
	Edg_T2R14C7_DHP	GTGTCGTAGACACTTTTTTGTTTAACGTCTCCAAATAAG

		AAACGAGTGTCGTAGACAC
	Edg_T2R18C7_DHP	GTGTCGTAGACACTAAACCAAGTACCGCATTCCAAGAAC GGGTATGTGTCGTAGACAC
	2G2T3C7R08	TATGAGCAAAAGAAGATGATTCATTTCAATTACC
	2G2T3C7R10	CACAATATAATCCTGATTGATGATGGCAATTCAT
	2G2T3C7R12	TGGTTATCTAAAATATCTAAAGGAATTGAGGAAG
	2G2T3C7R14	AGACATCGCCATTAAAAAAACTGATAGCCCTAAA
	2G2T3C7R16	CCTCGTCTGAAATGGATTACATTTTGACGCTCAA
	2G2T3C7R18	TCTTGATTAGTAATAACATTGTAGCAATACTTCT
	Q_T3_R20	AGGAACGGTACGCCAGTAAAGGGATTTTAGAC/3IAbRQ /
	Edg_T4R02C7_DHP	GTGTCGTAGACACCGGGCGCTAGGGCGCTAAGAAAGCGA AAGGAGGTGTCGTAGACAC
	Edg_T4R06C7_DHP	GTGTCGTAGACACATCCTGTTTGATGGTGGCCCCAGCAG GCGAAAGTGTCGTAGACAC
	Edg_T4R10C7_DHP	GTGTCGTAGACACGTAACGCCAGGGTTTTAAGGCGATTA AGTTGGGTGTCGTAGACAC
	Edg_T4R14C7_DHP	GTGTCGTAGACACTTTAAATTGTAAACGTATTGTATAAG CAAATAGTGTCGTAGACAC
	Edg_T4R18C7_DHP	GTGTCGTAGACACAAATTTTTAGAACCCTTTCAACGCAA GGATAAGTGTCGTAGACAC
Trig	Edg_T1R02C7_DHP	GTGTCGTAGACACTCCCAATTCTGCGAACCCATATAACA GTTGATGTGTCGTAGACAC
	Edg_T1R06C7_DHP	GTGTCGTAGACACCCATAAATCAAAAATCCAGAAAACGA GAATGAGTGTCGTAGACAC
	Edg_T1R10C7_DHP	GTGTCGTAGACACGAAACACCAGAACGAGAGGCTTGCCC TGACGAGTGTCGTAGACAC
	Edg_T1R14C7_DHP	GTGTCGTAGACACGAACGAGGGTAGCAACGCGAAAGACA GCATCGGTGTCGTAGACAC
	Edg_T1R18C7_DHP	GTGTCGTAGACACGGGATTTTGCTAAACAAATGAATTTT CTGTATGTGTCGTAGACAC
	Edg_T2R02C7_DHP	GTGTCGTAGACACGAGAGGGTTGATATAAGCGGATAAGT GCCGTCGTGTCGTAGACAC
	Edg_T2R06C7_DHP	GTGTCGTAGACACGCAGGTCAGACGATTGTTGACAGGAG GTTGAGGTGTCGTAGACAC
	Edg_T2R10C7_DHP	GTGTCGTAGACACGCGCCAAAGACAAAAGTTCATATGGT TTACCAGTGTCGTAGACAC
	Edg_T2R14C7_DHP	GTGTCGTAGACACTTTTTTGTTTAACGTCTCCAAATAAG AAACGAGTGTCGTAGACAC
	Edg_T2R18C7_DHP	GTGTCGTAGACACTAAACCAAGTACCGCATTCCAAGAAC GGGTATGTGTCGTAGACAC
	2G2T3C7R00	ATGTAAAGTAATTCTGTCAAAGTACCGACAAAAG
	2G2T3C7R02	ATAGTAGGGCTTAATTGAAAAGCCAACGCTCAAC
	2G2T3C7R04	CGAATGGTTTGAAATACCCTTCTGACCTAAATTT
	2G2T3C7R06	GAAGTCAATAGTGAATTTTTAAGACGCTGAGAAG
	2G2T3C7R08	TATGAGCAAAAGAAGATGATTCATTTCAATTACC

	2G2T3C7R10	CACAATATAATCCTGATTGATGATGGCAATTCAT
	2G2T3C7R12	TGGTTATCTAAAATATCTAAAGGAATTGAGGAAG
	2G2T3C7R14	AGACATCGCCATTAAAAAAACTGATAGCCCTAAA
	2G2T3C7R16	CCTCGTCTGAAATGGATTACATTTTGACGCTCAA
	2G2T3C7R18	TCTTGATTAGTAATAACATTGTAGCAATACTTCT
	Edg_T3C7R20_pip i	AGGAACGGTACGCCAGTAAAGGGATTTTAGAC
	Edg_T4R02C7_DHP	GTGTCGTAGACACCGGGCGCTAGGGCGCTAAGAAAGCGA AAGGAGGTGTCGTAGACAC
	Edg_T4R06C7_DHP	GTGTCGTAGACACATCCTGTTTGATGGTGGCCCCAGCAG GCGAAAGTGTCGTAGACAC
	Edg_T4R10C7_DHP	GTGTCGTAGACACGTAACGCCAGGGTTTTAAGGCGATTA AGTTGGGTGTCGTAGACAC
	Edg_T4R14C7_DHP	GTGTCGTAGACACTTTAAATTGTAAACGTATTGTATAAG CAAATAGTGTCGTAGACAC
	Edg_T4R18C7_DHP	GTGTCGTAGACACAAATTTTTAGAACCCTTTCAACGCAA GGATAAGTGTCGTAGACAC
Base1nt TYE563	Edg_T1R02C7_DHP	GTGTCGTAGACACTCCCAATTCTGCGAACCCATATAACA GTTGATGTGTCGTAGACAC
	Edg_T1R06C7_DHP	GTGTCGTAGACACCCATAAATCAAAAATCCAGAAAACGA GAATGAGTGTCGTAGACAC
	Edg_T1R10C7_DHP	GTGTCGTAGACACGAAACACCAGAACGAGAGGCTTGCCC TGACGAGTGTCGTAGACAC
	Edg_T1R14C7_DHP	GTGTCGTAGACACGAACGAGGGTAGCAACGCGAAAGACA GCATCGGTGTCGTAGACAC
	Edg_T1R18C7_DHP	GTGTCGTAGACACGGGATTTTGCTAAACAAATGAATTTT CTGTATGTGTCGTAGACAC
	TYE_F_T2R00	/5TYE563/AGCCACCACCCTCATTGAACCGCCACCCTC AG
	2bRT2C7R02	GAGAGGGTTGATATAAGCGGATAAGTGCCG
	2bRT2C7R04	GTATAAACAGTTAATGTTGAGTAACAGTGC
	2bRT2C7R06	GCAGGTCAGACGATTGTTGACAGGAGGTTG
	2bRT2C7R08	TAGCGCGTTTTCATCGCTTTAGCGTCAGAC
	2bRT2C7R10	GCGCCAAAGACAAAAGTTCATATGGTTTAC
	2bRT2C7R12	CCGAAGCCCTTTTTAAAGCAATAGCTATCT
	1bRT2C7R14	TTTTTTGTTTAACGTCTCCAAATAAGAAACG
	1bRT2C7R16	AACCTCCCGACTTGCGGCGAGGCGTTTTAGC
	1bRT2C7R18	TAAACCAAGTACCGCATTCCAAGAACGGGTA
	1bRT2C7R20	AGATAAGTCCTGAACACCTGTTTATCAACAA
	Edg_T3R02C7_DHP	GTGTCGTAGACACAGTAGGGCTTAATTGAAAAGCCAACG CTCAACGTGTCGTAGACAC
	Edg_T3R06C7_DHP	GTGTCGTAGACACAGTCAATAGTGAATTTTTAAGACGCT GAGAAGGTGTCGTAGACAC
	Edg_T3R10C7_DHP	GTGTCGTAGACACCAATATAATCCTGATTGATGATGGCA ATTCATGTGTCGTAGACAC

	Edg_T3R14C7_DHP	GTGTCGTAGACACACATCGCCATTAAAAAAACTGATAGC CCTAAAGTGTCGTAGACAC
	Edg_T3R18C7_DHP	GTGTCGTAGACACTTGATTAGTAATAACATTGTAGCAAT ACTTCTGTGTCGTAGACAC
	Edg_T4R02C7_DHP	GTGTCGTAGACACCGGGCGCTAGGGCGCTAAGAAAGCGA AAGGAGGTGTCGTAGACAC
	Edg_T4R06C7_DHP	GTGTCGTAGACACATCCTGTTTGATGGTGGCCCCAGCAG GCGAAAGTGTCGTAGACAC
	Edg_T4R10C7_DHP	GTGTCGTAGACACGTAACGCCAGGGTTTTAAGGCGATTA AGTTGGGTGTCGTAGACAC
	Edg_T4R14C7_DHP	GTGTCGTAGACACTTTAAATTGTAAACGTATTGTATAAG CAAATAGTGTCGTAGACAC
	Edg_T4R18C7_DHP	GTGTCGTAGACACAAATTTTTAGAACCCTTTCAACGCAA GGATAAGTGTCGTAGACAC

Table S3: Edge staples in sequential reconfiguration measurements

Tile	Edge	Edge Seq
T1	2G2T1C7R00	ATGGTGGCATCAATTCTAGGGCGCGAGCTGAAAA
	2G2T1C7R02	ATTCCCAATTCTGCGAACCCATATAACAGTTGAT
	2G2T1C7R06	GACCATAAATCAAAAATCCAGAAAACGAGAATGA
	2G2T1C7R10	CAGAAACACCAGAACGAGAGGCTTGCCCTGACGA
	2G2T1C7R12	TGCTGATAAATTGTGTCGAGATTTGTATCATCGC
	2G2T1C7R14	AGGAACGAGGGTAGCAACGCGAAAGACAGCATCG
	2G2T1C7R16	CCGGTTTATCAGCTTGCTAGCCTTTAATTGTATC
	2G2T1C7R18	TCGGGATTTTGCTAAACAAATGAATTTTCTGTAT
	2G2T1C7R20	AGACAAACTACAACGCCTGAGTTTCGTCACCAGT
	Edg_T2R02C7_DHP	GTGTCGTAGACACGAGAGGGTTGATATAAGCGGATAAGTGCC GTCGTGTCGTAGACAC
	Edg_T2R06C7_DHP	GTGTCGTAGACACGCAGGTCAGACGATTGTTGACAGGAGGTT GAGGTGTCGTAGACAC
	Edg_T2R10C7_DHP	GTGTCGTAGACACGCGCCAAAGACAAAAGTTCATATGGTTTA CCAGTGTCGTAGACAC
	Edg_T2R14C7_DHP	GTGTCGTAGACACTTTTTTGTTTAACGTCTCCAAATAAGAAA CGAGTGTCGTAGACAC
	Edg_T2R18C7_DHP	GTGTCGTAGACACTAAACCAAGTACCGCATTCCAAGAACGGG TATGTGTCGTAGACAC
	Edg_T3R02C7_DHP	GTGTCGTAGACACAGTAGGGCTTAATTGAAAAGCCAACGCTC AACGTGTCGTAGACAC
	Edg_T3R06C7_DHP	GTGTCGTAGACACAGTCAATAGTGAATTTTTAAGACGCTGAG AAGGTGTCGTAGACAC

	Edg_T3R10C7_DHP	GTGTCGTAGACACCAATATAATCCTGATTGATGATGGCAATT CATGTGTCGTAGACAC
	Edg_T3R14C7_DHP	GTGTCGTAGACACACATCGCCATTAAAAAAACTGATAGCCCT AAAGTGTCGTAGACAC
	Edg_T3R18C7_DHP	GTGTCGTAGACACTTGATTAGTAATAACATTGTAGCAATACT TCTGTGTCGTAGACAC
	2G4T4C7R00	ATGAGCACGTATAACGTGCTATGGTTGCTTTGAC
	2G4T4C7R02	AACGGGCGCTAGGGCGCTAAGAAAGCGAAAGGAG
	2G4T4C7R04	ATATCACCCAAATCAAGTGCCCACTACGTGAACC
	2G4T4C7R06	TAATCCTGTTTGATGGTGGCCCCAGCAGGCGAAA
	2G4T4C7R12	GCCGTTGGTGTAGATGGGGTAATGGGATAGGTCA
	2G4T4C7R16	CCGCCGGAGAGGGTAGCTTAGCTGATAAATTAAT
	2G4T4C7R20	ACTAAGCAATAAAGCCTCAAAGAATTAGCAAAAT
T1b	2G2T1C7R00	ATGGTGGCATCAATTCTAGGGCGCGAGCTGAAAA
	2G2T1C7R02	ATTCCCAATTCTGCGAACCCATATAACAGTTGAT
	2G2T1C7R06	GACCATAAATCAAAAATCCAGAAAACGAGAATGA
	2G2T1C7R10	CAGAAACACCAGAACGAGAGGCTTGCCCTGACGA
	2G2T1C7R12	TGCTGATAAATTGTGTCGAGATTTGTATCATCGC
	2G2T1C7R14	AGGAACGAGGGTAGCAACGCGAAAGACAGCATCG
	2G2T1C7R16	CCGGTTTATCAGCTTGCTAGCCTTTAATTGTATC
	2G2T1C7R18	TCGGGATTTTGCTAAACAAATGAATTTTCTGTAT
	2G2T1C7R20	AGACAAACTACAACGCCTGAGTTTCGTCACCAGT
	Edg_T2R02C7_DHP	GTGTCGTAGACACGAGAGGGTTGATATAAGCGGATAAGTGCC GTCGTGTCGTAGACAC
	Edg_T2R06C7_DHP	GTGTCGTAGACACGCAGGTCAGACGATTGTTGACAGGAGGTT GAGGTGTCGTAGACAC
	Edg_T2R10C7_DHP	GTGTCGTAGACACGCGCCAAAGACAAAAGTTCATATGGTTTA CCAGTGTCGTAGACAC
	Edg_T2R14C7_DHP	GTGTCGTAGACACTTTTTTGTTTAACGTCTCCAAATAAGAAA CGAGTGTCGTAGACAC
	Edg_T2R18C7_DHP	GTGTCGTAGACACTAAACCAAGTACCGCATTCCAAGAACGGG TATGTGTCGTAGACAC
	Edg_T3R02C7_DHP	GTGTCGTAGACACAGTAGGGCTTAATTGAAAAGCCAACGCTC AACGTGTCGTAGACAC
	Edg_T3R06C7_DHP	GTGTCGTAGACACAGTCAATAGTGAATTTTTAAGACGCTGAG AAGGTGTCGTAGACAC
	Edg_T3R10C7_DHP	GTGTCGTAGACACCAATATAATCCTGATTGATGATGGCAATT CATGTGTCGTAGACAC
	Edg_T3R14C7_DHP	GTGTCGTAGACACACATCGCCATTAAAAAAACTGATAGCCCT AAAGTGTCGTAGACAC
	Edg_T3R18C7_DHP	GTGTCGTAGACACTTGATTAGTAATAACATTGTAGCAATACT TCTGTGTCGTAGACAC
	2G4T4C7R06	TAATCCTGTTTGATGGTGGCCCCAGCAGGCGAAA
	2G4T4C7R12	GCCGTTGGTGTAGATGGGGTAATGGGATAGGTCA

	2G4T4C7R16	CCGCCGGAGAGGGTAGCTTAGCTGATAAATTAAT
	2G4T4C7R20	ACTAAGCAATAAAGCCTCAAAGAATTAGCAAAAT
T1a	2G2T1C7R00	ATGGTGGCATCAATTCTAGGGCGCGAGCTGAAAA
	2G2T1C7R02	ATTCCCAATTCTGCGAACCCATATAACAGTTGAT
	2G2T1C7R06	GACCATAAATCAAAAATCCAGAAAACGAGAATGA
	2G2T1C7R10	CAGAAACACCAGAACGAGAGGCTTGCCCTGACGA
	2G2T1C7R12	TGCTGATAAATTGTGTCGAGATTTGTATCATCGC
	Edg_T2R02C7_DHP	GTGTCGTAGACACGAGAGGGTTGATATAAGCGGATAAGTGCC GTCGTGTCGTAGACAC
	Edg_T2R06C7_DHP	GTGTCGTAGACACGCAGGTCAGACGATTGTTGACAGGAGGTT GAGGTGTCGTAGACAC
	Edg_T2R10C7_DHP	GTGTCGTAGACACGCGCCAAAGACAAAAGTTCATATGGTTTA CCAGTGTCGTAGACAC
	Edg_T2R14C7_DHP	GTGTCGTAGACACTTTTTTGTTTAACGTCTCCAAATAAGAAA CGAGTGTCGTAGACAC
	Edg_T2R18C7_DHP	GTGTCGTAGACACTAAACCAAGTACCGCATTCCAAGAACGGG TATGTGTCGTAGACAC
	Edg_T3R02C7_DHP	GTGTCGTAGACACAGTAGGGCTTAATTGAAAAGCCAACGCTC AACGTGTCGTAGACAC
	Edg_T3R06C7_DHP	GTGTCGTAGACACAGTCAATAGTGAATTTTTAAGACGCTGAG AAGGTGTCGTAGACAC
	Edg_T3R10C7_DHP	GTGTCGTAGACACCAATATAATCCTGATTGATGATGGCAATT CATGTGTCGTAGACAC
	Edg_T3R14C7_DHP	GTGTCGTAGACACACATCGCCATTAAAAAAACTGATAGCCCT AAAGTGTCGTAGACAC
	Edg_T3R18C7_DHP	GTGTCGTAGACACTTGATTAGTAATAACATTGTAGCAATACT TCTGTGTCGTAGACAC
	2G4T4C7R00	ATGAGCACGTATAACGTGCTATGGTTGCTTTGAC
	2G4T4C7R02	AACGGGCGCTAGGGCGCTAAGAAAGCGAAAGGAG
	2G4T4C7R04	ATATCACCCAAATCAAGTGCCCACTACGTGAACC
	2G4T4C7R06	TAATCCTGTTTGATGGTGGCCCCAGCAGGCGAAA
	2G4T4C7R12	GCCGTTGGTGTAGATGGGGTAATGGGATAGGTCA
	2G4T4C7R16	CCGCCGGAGAGGGTAGCTTAGCTGATAAATTAAT
	2G4T4C7R20	ACTAAGCAATAAAGCCTCAAAGAATTAGCAAAAT
T2	Edg_T1R02C7_DHP	GTGTCGTAGACACTCCCAATTCTGCGAACCCATATAACAGTT GATGTGTCGTAGACAC
	Edg_T1R06C7_DHP	GTGTCGTAGACACCCATAAATCAAAAATCCAGAAAACGAGAA TGAGTGTCGTAGACAC
	Edg_T1R10C7_DHP	GTGTCGTAGACACGAAACACCAGAACGAGAGGCTTGCCCTGA CGAGTGTCGTAGACAC
	Edg_T1R14C7_DHP	GTGTCGTAGACACGAACGAGGGTAGCAACGCGAAAGACAGCA TCGGTGTCGTAGACAC
	Edg_T1R18C7_DHP	GTGTCGTAGACACGGGATTTTGCTAAACAAATGAATTTTCTG TATGTGTCGTAGACAC
	2bRT2C7R00	AGCCACCACCCTCATTGAACCGCCACCCTC

	2bRT2C7R02	GAGAGGGTTGATATAAGCGGATAAGTGCCG
	2bRT2C7R04	GTATAAACAGTTAATGTTGAGTAACAGTGC
	2bRT2C7R06	GCAGGTCAGACGATTGTTGACAGGAGGTTG
	2bRT2C7R08	TAGCGCGTTTTCATCGCTTTAGCGTCAGAC
	2bRT2C7R10	GCGCCAAAGACAAAAGTTCATATGGTTTAC
	2bRT2C7R14	TTTTTTGTTTAACGTCTCCAAATAAGAAAC
	2bRT2C7R18	TAAACCAAGTACCGCATTCCAAGAACGGGT
	2bRT2C7R20	AGATAAGTCCTGAACACCTGTTTATCAACA
	2G2T3C7R04	CGAATGGTTTGAAATACCCTTCTGACCTAAATTT
	2G2T3C7R08	TATGAGCAAAAGAAGATGATTCATTTCAATTACC
	2G2T3C7R12	TGGTTATCTAAAATATCTAAAGGAATTGAGGAAG
	2G2T3C7R18	TCTTGATTAGTAATAACATTGTAGCAATACTTCT
	Q_T3_R20	AGGAACGGTACGCCAGTAAAGGGATTTTAGAC/3IAbRQ/
	Edg_T4R02C7_DHP	GTGTCGTAGACACCGGGCGCTAGGGCGCTAAGAAAGCGAAAG GAGGTGTCGTAGACAC
	Edg_T4R06C7_DHP	GTGTCGTAGACACATCCTGTTTGATGGTGGCCCCAGCAGGCG AAAGTGTCGTAGACAC
	Edg_T4R10C7_DHP	GTGTCGTAGACACGTAACGCCAGGGTTTTAAGGCGATTAAGT TGGGTGTCGTAGACAC
	Edg_T4R14C7_DHP	GTGTCGTAGACACTTTAAATTGTAAACGTATTGTATAAGCAA ATAGTGTCGTAGACAC
	Edg_T4R18C7_DHP	GTGTCGTAGACACAAATTTTTAGAACCCTTTCAACGCAAGGA TAAGTGTCGTAGACAC
T3	Edg_T1R02C7_DHP	GTGTCGTAGACACTCCCAATTCTGCGAACCCATATAACAGTT GATGTGTCGTAGACAC
	Edg_T1R06C7_DHP	GTGTCGTAGACACCCATAAATCAAAAATCCAGAAAACGAGAA TGAGTGTCGTAGACAC
	Edg_T1R10C7_DHP	GTGTCGTAGACACGAAACACCAGAACGAGAGGCTTGCCCTGA CGAGTGTCGTAGACAC
	Edg_T1R14C7_DHP	GTGTCGTAGACACGAACGAGGGTAGCAACGCGAAAGACAGCA TCGGTGTCGTAGACAC
	Edg_T1R18C7_DHP	GTGTCGTAGACACGGGATTTTGCTAAACAAATGAATTTTCTG TATGTGTCGTAGACAC
	Edg_T2R02C7_DHP	GTGTCGTAGACACGAGAGGGTTGATATAAGCGGATAAGTGCC GTCGTGTCGTAGACAC
	Edg_T2R06C7_DHP	GTGTCGTAGACACGCAGGTCAGACGATTGTTGACAGGAGGTT GAGGTGTCGTAGACAC
	Edg_T2R10C7_DHP	GTGTCGTAGACACGCGCCAAAGACAAAAGTTCATATGGTTTA CCAGTGTCGTAGACAC
	Edg_T2R14C7_DHP	GTGTCGTAGACACTTTTTTGTTTAACGTCTCCAAATAAGAAA CGAGTGTCGTAGACAC
	Edg_T2R18C7_DHP	GTGTCGTAGACACTAAACCAAGTACCGCATTCCAAGAACGGG TATGTGTCGTAGACAC
	2G3T3C7R00	ACGTAAAGTAATTCTGTCAAAGTACCGACAAAAG
	2G3T3C7R04	AAAATGGTTTGAAATACCCTTCTGACCTAAATTT

	2G3T3C7R08	AGTGAGCAAAAGAAGATGATTCATTTCAATTACC
	2G3T3C7R12	CCGTTATCTAAAATATCTAAAGGAATTGAGGAAG
	2G3T3C7R16	TTTCGTCTGAAATGGATTACATTTTGACGCTCAA
	2G3T3C7R18	ACTTGATTAGTAATAACATTGTAGCAATACTTCT
	2G3T3C7R20	AGAGGAACGGTACGCCAGTAAAGGGATTTTAGAC
	2bRT4C7R00	GAGCACGTATAACGTGCTATGGTTGCTTTG
	2bRT4C7R04	ATCACCCAAATCAAGTGCCCACTACGTGAA
	2bRT4C7R08	GCTCACTGCCCGCTTTACATTAATTGCGTT
	2bRT4C7R14	TTTAAATTGTAAACGTATTGTATAAGCAAA
	2bRT4C7R16	GCCGGAGAGGGTAGCTTAGCTGATAAATTA
	2bRT4C7R18	AAATTTTTAGAACCCTTTCAACGCAAGGAT
	2bRT4C7R20	TAAGCAATAAAGCCTCAAAGAATTAGCAAA
T4	Edg_T1R02C7_DHP	GTGTCGTAGACACTCCCAATTCTGCGAACCCATATAACAGTT GATGTGTCGTAGACAC
	Edg_T1R06C7_DHP	GTGTCGTAGACACCCATAAATCAAAAATCCAGAAAACGAGAA TGAGTGTCGTAGACAC
	Edg_T1R10C7_DHP	GTGTCGTAGACACGAAACACCAGAACGAGAGGCTTGCCCTGA CGAGTGTCGTAGACAC
	Edg_T1R14C7_DHP	GTGTCGTAGACACGAACGAGGGTAGCAACGCGAAAGACAGCA TCGGTGTCGTAGACAC
	Edg_T1R18C7_DHP	GTGTCGTAGACACGGGATTTTGCTAAACAAATGAATTTTCTG TATGTGTCGTAGACAC
	F_rox_v1_T2_R00	/5Rox_N/AGCCACCACCCTCATTGAACCGCCACCCTCAG
	2bRT2C7R02	GAGAGGGTTGATATAAGCGGATAAGTGCCG
	2bRT2C7R08	TAGCGCGTTTTCATCGCTTTAGCGTCAGAC
	2bRT2C7R12	CCGAAGCCCTTTTTAAAGCAATAGCTATCT
	2bRT2C7R16	AACCTCCCGACTTGCGGCGAGGCGTTTTAG
	2bRT3C7R00	GTAAAGTAATTCTGTCAAAGTACCGACAAA
	2bRT3C7R02	AGTAGGGCTTAATTGAAAAGCCAACGCTCA
	2bRT3C7R04	AATGGTTTGAAATACCCTTCTGACCTAAAT
	2bRT3C7R08	TGAGCAAAAGAAGATGATTCATTTCAATTA
	2bRT3C7R12	GTTATCTAAAATATCTAAAGGAATTGAGGA
	2bRT3C7R16	TCGTCTGAAATGGATTACATTTTGACGCTC
	2bRT3C7R20	AGGAACGGTACGCCAGTAAAGGGATTTTAG
	Edg_T4R02C7_DHP	GTGTCGTAGACACCGGGCGCTAGGGCGCTAAGAAAGCGAAAG GAGGTGTCGTAGACAC
	Edg_T4R06C7_DHP	GTGTCGTAGACACATCCTGTTTGATGGTGGCCCCAGCAGGCG AAAGTGTCGTAGACAC
	Edg_T4R10C7_DHP	GTGTCGTAGACACGTAACGCCAGGGTTTTAAGGCGATTAAGT TGGGTGTCGTAGACAC
	Edg_T4R14C7_DHP	GTGTCGTAGACACTTTAAATTGTAAACGTATTGTATAAGCAA ATAGTGTCGTAGACAC
	Edg_T4R18C7_DHP	GTGTCGTAGACACAAATTTTTAGAACCCTTTCAACGCAAGGA

		TAAGTGTCGTAGACAC
T2_2	Edg_T1R02C7_DHP	GTGTCGTAGACACTCCCAATTCTGCGAACCCATATAACAGTTGATGTGTCGTAGACAC
	Edg_T1R06C7_DHP	GTGTCGTAGACACCCATAAATCAAAAATCCAGAAAACGAGAATGAGTGTCGTAGACAC
	Edg_T1R10C7_DHP	GTGTCGTAGACACGAAACACCAGAACGAGAGGCTTGCCCTGACGAGTGTCGTAGACAC
	Edg_T1R14C7_DHP	GTGTCGTAGACACGAACGAGGGTAGCAACGCGAAAGACAGCATCGGTGTCGTAGACAC
	Edg_T1R18C7_DHP	GTGTCGTAGACACGGGATTTTGCTAAACAAATGAATTTTCTGTATGTGTCGTAGACAC
	2bRT2C7R00	AGCCACCACCCTCATTGAACCGCCACCCTC
	2bRT2C7R02	GAGAGGGTTGATATAAGCGGATAAGTGCCG
	2bRT2C7R04	GTATAAACAGTTAATGTTGAGTAACAGTGC
	2bRT2C7R06	GCAGGTCAGACGATTGTTGACAGGAGGTTG
	2bRT2C7R08	TAGCGCGTTTTCATCGCTTTAGCGTCAGAC
	2bRT2C7R10	GCGCCAAAGACAAAAGTTCATATGGTTTAC
	2bRT2C7R14	TTTTTTGTTTAACGTCTCCAAATAAGAAAC
	2bRT2C7R18	TAAACCAAGTACCGCATTCCAAGAACGGGT
	2bRT2C7R20	AGATAAGTCCTGAACACCTGTTTATCAACA
	2G2T3C7R04	CGAATGGTTTGAAATACCCTTCTGACCTAAATTT
	2G2T3C7R08	TATGAGCAAAAGAAGATGATTCATTTCAATTACC
	2G2T3C7R12	TGGTTATCTAAAATATCTAAAGGAATTGAGGAAG
	2G2T3C7R18	TCTTGATTAGTAATAACATTGTAGCAATACTTCT
	Edg_T3C7R20_pipi	AGGAACGGTACGCCAGTAAAGGGATTTTAGAC
	Edg_T4R02C7_DHP	GTGTCGTAGACACCGGGCGCTAGGGCGCTAAGAAAGCGAAAGGAGGTGTCGTAGACAC
	Edg_T4R06C7_DHP	GTGTCGTAGACACATCCTGTTTGATGGTGGCCCCAGCAGGCGAAAGTGTCGTAGACAC
	Edg_T4R10C7_DHP	GTGTCGTAGACACGTAACGCCAGGGTTTTAAGGCGATTAAGTTGGGTGTCGTAGACAC
	Edg_T4R14C7_DHP	GTGTCGTAGACACTTTAAATTGTAAACGTATTGTATAAGCAAATAGTGTCGTAGACAC
	Edg_T4R18C7_DHP	GTGTCGTAGACACAAATTTTTAGAACCCTTTCAACGCAAGGATAAGTGTCGTAGACAC

Table S4: Edge staples in cooperative reconfiguration measurements

Tile	Edge	Edge Seq
Cover1	Edg_T1R02C7_DHP	GTGTCGTAGACACTCCCAATTCTGCGAACCCATATAA CAGTTGATGTGTCGTAGACAC
	Edg_T1R06C7_DHP	GTGTCGTAGACACCCATAAATCAAAAATCCAGAAAAC GAGAATGAGTGTCGTAGACAC
	Edg_T1R10C7_DHP	GTGTCGTAGACACGAAACACCAGAACGAGAGGCTTGC CCTGACGAGTGTCGTAGACAC
	Edg_T1R14C7_DHP	GTGTCGTAGACACGAACGAGGGTAGCAACGCGAAAGA CAGCATCGGTGTCGTAGACAC
	Edg_T1R18C7_DHP	GTGTCGTAGACACGGGATTTTGCTAAACAAATGAATT TTCTGTATGTGTCGTAGACAC
	2G4T2C7R00	ATAGCCACCACCCTCATTGAACCGCCACCCTCAG
	2G4T2C7R04	ATGTATAAACAGTTAATGTTGAGTAACAGTGCCC
	2G4T2C7R10	GGGCGCCAAAGACAAAAGTTCATATGGTTTACCA
	2G4T2C7R12	GCCCGAAGCCCTTTTTAAAGCAATAGCTATCTTA
	2G4T2C7R16	CCAACCTCCCGACTTGCGGCGAGGCGTTTTAGCG
	2G4T2C7R20	ACAGATAAGTCCTGAACACCTGTTTATCAACAAT
	2G3T3C7R00	ACGTAAAGTAATTCTGTCAAAGTACCGACAAAAG
	2G3T3C7R02	CTAGTAGGGCTTAATTGAAAAGCCAACGCTCAAC
	2G3T3C7R06	AAAGTCAATAGTGAATTTTTAAGACGCTGAGAAG
	2G3T3C7R10	ATCAATATAATCCTGATTGATGATGGCAATTCAT
	2G3T3C7R14	AGACATCGCCATTAAAAAAACTGATAGCCCTAAA
	Edg_T4R02C7_DHP	GTGTCGTAGACACCGGGCGCTAGGGCGCTAAGAAAGC GAAAGGAGGTGTCGTAGACAC
	Edg_T4R06C7_DHP	GTGTCGTAGACACATCCTGTTTGATGGTGGCCCCAGC AGGCGAAAGTGTCGTAGACAC
	Edg_T4R10C7_DHP	GTGTCGTAGACACGTAACGCCAGGGTTTTAAGGCGAT TAAGTTGGGTGTCGTAGACAC
	Edg_T4R14C7_DHP	GTGTCGTAGACACTTTAAATTGTAAACGTATTGTATA AGCAAATAGTGTCGTAGACAC
	Edg_T4R18C7_DHP	GTGTCGTAGACACAAATTTTTAGAACCCTTTCAACGC AAGGATAAGTGTCGTAGACAC
Cover2	Edg_T1R02C7_DHP	GTGTCGTAGACACTCCCAATTCTGCGAACCCATATAA CAGTTGATGTGTCGTAGACAC
	Edg_T1R06C7_DHP	GTGTCGTAGACACCCATAAATCAAAAATCCAGAAAAC GAGAATGAGTGTCGTAGACAC
	Edg_T1R10C7_DHP	GTGTCGTAGACACGAAACACCAGAACGAGAGGCTTGC CCTGACGAGTGTCGTAGACAC
	Edg_T1R14C7_DHP	GTGTCGTAGACACGAACGAGGGTAGCAACGCGAAAGA CAGCATCGGTGTCGTAGACAC
	Edg_T1R18C7_DHP	GTGTCGTAGACACGGGATTTTGCTAAACAAATGAATT TTCTGTATGTGTCGTAGACAC
	Edg_T2R02C7_DHP	GTGTCGTAGACACGAGAGGGTTGATATAAGCGGATAA

		GTGCCGTCGTGTCGTAGACAC
	Edg_T2R06C7_DHP	GTGTCGTAGACACGCAGGTCAGACGATTGTTGACAGG AGGTTGAGGTGTCGTAGACAC
	Edg_T2R10C7_DHP	GTGTCGTAGACACGCGCCAAAGACAAAAGTTCATATG GTTTACCAGTGTCGTAGACAC
	Edg_T2R14C7_DHP	GTGTCGTAGACACTTTTTTGTTTAACGTCTCCAAATA AGAAACGAGTGTCGTAGACAC
	Edg_T2R18C7_DHP	GTGTCGTAGACACTAAACCAAGTACCGCATTCCAAGA ACGGGTATGTGTCGTAGACAC
	2G2T3C7R06	GAAGTCAATAGTGAATTTTTAAGACGCTGAGAAG
	2G2T3C7R10	CACAATATAATCCTGATTGATGATGGCAATTCAT
	2G2T3C7R12	TGGTTATCTAAAATATCTAAAGGAATTGAGGAAG
	2G2T3C7R16	CCTCGTCTGAAATGGATTACATTTTGACGCTCAA
	2G2T3C7R18	TCTTGATTAGTAATAACATTGTAGCAATACTTCT
	Q_T3_R20	AGGAACGGTACGCCAGTAAAGGGATTTTAGAC/3IAb RQ/
	2bRT4C7R00	GAGCACGTATAACGTGCTATGGTTGCTTTG
	2bRT4C7R04	ATCACCCAAATCAAGTGCCCACTACGTGAA
	2bRT4C7R08	GCTCACTGCCCGCTTTACATTAATTGCGTT
	2bRT4C7R10	GTAACGCCAGGGTTTTAAGGCGATTAAGTT
	2bRT4C7R16	GCCGGAGAGGGTAGCTTAGCTGATAAATTA
	2bRT4C7R20	TAAGCAATAAAGCCTCAAAGAATTAGCAAA
Base1	Edg_T1R02C7_DHP	GTGTCGTAGACACTCCCAATTCTGCGAACCCATATAA CAGTTGATGTGTCGTAGACAC
	Edg_T1R06C7_DHP	GTGTCGTAGACACCCATAAATCAAAAATCCAGAAAAC GAGAATGAGTGTCGTAGACAC
	Edg_T1R10C7_DHP	GTGTCGTAGACACGAAACACCAGAACGAGAGGCTTGC CCTGACGAGTGTCGTAGACAC
	Edg_T1R14C7_DHP	GTGTCGTAGACACGAACGAGGGTAGCAACGCGAAAGA CAGCATCGGTGTCGTAGACAC
	Edg_T1R18C7_DHP	GTGTCGTAGACACGGGATTTTGCTAAACAAATGAATT TTCTGTATGTGTCGTAGACAC
	Edg_T2R02C7_DHP	GTGTCGTAGACACGAGAGGGTTGATATAAGCGGATAA GTGCCGTCGTGTCGTAGACAC
	Edg_T2R06C7_DHP	GTGTCGTAGACACGCAGGTCAGACGATTGTTGACAGG AGGTTGAGGTGTCGTAGACAC
	Edg_T2R10C7_DHP	GTGTCGTAGACACGCGCCAAAGACAAAAGTTCATATG GTTTACCAGTGTCGTAGACAC
	Edg_T2R14C7_DHP	GTGTCGTAGACACTTTTTTGTTTAACGTCTCCAAATA AGAAACGAGTGTCGTAGACAC
	Edg_T2R18C7_DHP	GTGTCGTAGACACTAAACCAAGTACCGCATTCCAAGA ACGGGTATGTGTCGTAGACAC
	2bRT3C7R00	GTAAAGTAATTCTGTCAAAGTACCGACAAA
	2bRT3C7R02	AGTAGGGCTTAATTGAAAAGCCAACGCTCA
	2bRT3C7R04	AATGGTTTGAAATACCCTTCTGACCTAAAT

	2bRT3C7R06	AGTCAATAGTGAATTTTTAAGACGCTGAGA
	2bRT3C7R10	CAATATAATCCTGATTGATGATGGCAATTC
	2bRT3C7R14	ACATCGCCATTAAAAAAACTGATAGCCCTA
	2bRT3C7R18	TTGATTAGTAATAACATTGTAGCAATACTT
	2bRT3C7R20	AGGAACGGTACGCCAGTAAAGGGATTTTAG
	2G1T4C7R00	GTGAGCACGTATAACGTGCTATGGTTGCTTTGAC
	2G1T4C7R04	TCATCACCCAAATCAAGTGCCCACTACGTGAACC
	2G1T4C7R10	GAGTAACGCCAGGGTTTTAAGGCGATTAAGTTGG
	2G1T4C7R14	GATTTAAATTGTAAACGTATTGTATAAGCAAATA
	2G1T4C7R16	TAGCCGGAGAGGGTAGCTTAGCTGATAAATTAAT
	2G1T4C7R20	AATAAGCAATAAAGCCTCAAAGAATTAGCAAAAT
Base2	2bRT1C7R00	GGTGGCATCAATTCTAGGGCGCGAGCTGAA
	2bRT1C7R04	ATTGCTCCTTTTGATATTAGAGAGTACCTT
	2bRT1C7R06	CCATAAATCAAAAATCCAGAAAACGAGAAT
	2bRT1C7R10	GAAACACCAGAACGAGAGGCTTGCCCTGAC
	2bRT1C7R16	GGTTTATCAGCTTGCTAGCCTTTAATTGTA
	2bRT1C7R20	ACAAACTACAACGCCTGAGTTTCGTCACCA
	F_rox_v1_T2_R00	/5Rox_N/AGCCACCACCCTCATTGAACCGCCACCCT CAG
	2bRT2C7R02	GAGAGGGTTGATATAAGCGGATAAGTGCCG
	2bRT2C7R04	GTATAAACAGTTAATGTTGAGTAACAGTGC
	2bRT2C7R08	TAGCGCGTTTTCATCGCTTTAGCGTCAGAC
	2bRT2C7R10	GCGCCAAAGACAAAAGTTCATATGGTTTAC
	2bRT2C7R14	TTTTTTGTTTAACGTCTCCAAATAAGAAAC
	2bRT2C7R16	AACCTCCCGACTTGCGGCGAGGCGTTTTAG
	2bRT2C7R18	TAAACCAAGTACCGCATTCCAAGAACGGGT
	2bRT2C7R20	AGATAAGTCCTGAACACCTGTTTATCAACA
	Edg_T3R02C7_DHP	GTGTCGTAGACACAGTAGGGCTTAATTGAAAAGCCAA CGCTCAACGTGTCGTAGACAC
	Edg_T3R06C7_DHP	GTGTCGTAGACACAGTCAATAGTGAATTTTTAAGACG CTGAGAAGGTGTCGTAGACAC
	Edg_T3R10C7_DHP	GTGTCGTAGACACCAATATAATCCTGATTGATGATGG CAATTCATGTGTCGTAGACAC
	Edg_T3R14C7_DHP	GTGTCGTAGACACACATCGCCATTAAAAAAACTGATA GCCCTAAAGTGTCGTAGACAC
	Edg_T3R18C7_DHP	GTGTCGTAGACACTTGATTAGTAATAACATTGTAGCA ATACTTCTGTGTCGTAGACAC
	Edg_T4R02C7_DHP	GTGTCGTAGACACCGGGCGCTAGGGCGCTAAGAAAGC GAAAGGAGGTGTCGTAGACAC
	Edg_T4R06C7_DHP	GTGTCGTAGACACATCCTGTTTGATGGTGGCCCCAGC AGGCGAAAGTGTCGTAGACAC
	Edg_T4R10C7_DHP	GTGTCGTAGACACGTAACGCCAGGGTTTTAAGGCGAT TAAGTTGGGTGTCGTAGACAC

	Edg_T4R14C7_DHP	GTGTCGTAGACACTTTAAATTGTAAACGTATTGTATA AGCAAATAGTGTCGTAGACAC
	Edg_T4R18C7_DHP	GTGTCGTAGACACAAATTTTTAGAACCCTTTCAACGC AAGGATAAGTGTCGTAGACAC
Inv1	Edg_T1R02C7_DHP	GTGTCGTAGACACTCCCAATTCTGCGAACCCATATAA CAGTTGATGTGTCGTAGACAC
	Edg_T1R06C7_DHP	GTGTCGTAGACACCCATAAATCAAAAATCCAGAAAAC GAGAATGAGTGTCGTAGACAC
	Edg_T1R10C7_DHP	GTGTCGTAGACACGAAACACCAGAACGAGAGGCTTGC CCTGACGAGTGTCGTAGACAC
	Edg_T1R14C7_DHP	GTGTCGTAGACACGAACGAGGGTAGCAACGCGAAAGA CAGCATCGGTGTCGTAGACAC
	Edg_T1R18C7_DHP	GTGTCGTAGACACGGGATTTTGCTAAACAAATGAATT TTCTGTATGTGTCGTAGACAC
	Edg_T2R02C7_DHP	GTGTCGTAGACACGAGAGGGTTGATATAAGCGGATAA GTGCCGTCGTGTCGTAGACAC
	Edg_T2R06C7_DHP	GTGTCGTAGACACGCAGGTCAGACGATTGTTGACAGG AGGTTGAGGTGTCGTAGACAC
	Edg_T2R10C7_DHP	GTGTCGTAGACACGCGCCAAAGACAAAAGTTCATATG GTTTACCAGTGTCGTAGACAC
	Edg_T2R14C7_DHP	GTGTCGTAGACACTTTTTTGTTTAACGTCTCCAAATA AGAAACGAGTGTCGTAGACAC
	Edg_T2R18C7_DHP	GTGTCGTAGACACTAAACCAAGTACCGCATTCCAAGA ACGGGTATGTGTCGTAGACAC
	2G3T3C7R00	ACGTAAAGTAATTCTGTCAAAGTACCGACAAAAG
	2G3T3C7R02	CTAGTAGGGCTTAATTGAAAAGCCAACGCTCAAC
	2G3T3C7R06	AAAGTCAATAGTGAATTTTTAAGACGCTGAGAAG
	2G3T3C7R10	ATCAATATAATCCTGATTGATGATGGCAATTCAT
	2G3T3C7R14	AGACATCGCCATTAAAAAAACTGATAGCCCTAAA
	2G3T3C7R16	TTTCGTCTGAAATGGATTACATTTTGACGCTCAA
	2G3T3C7R18	ACTTGATTAGTAATAACATTGTAGCAATACTTCT
	2G3T3C7R20	AGAGGAACGGTACGCCAGTAAAGGGATTTTAGAC
	Edg_T4R02C7_DHP	GTGTCGTAGACACCGGGCGCTAGGGCGCTAAGAAAGC GAAAGGAGGTGTCGTAGACAC
	Edg_T4R06C7_DHP	GTGTCGTAGACACATCCTGTTTGATGGTGGCCCCAGC AGGCGAAAGTGTCGTAGACAC
	Edg_T4R10C7_DHP	GTGTCGTAGACACGTAACGCCAGGGTTTTAAGGCGAT TAAGTTGGGTGTCGTAGACAC
	Edg_T4R14C7_DHP	GTGTCGTAGACACTTTAAATTGTAAACGTATTGTATA AGCAAATAGTGTCGTAGACAC
	Edg_T4R18C7_DHP	GTGTCGTAGACACAAATTTTTAGAACCCTTTCAACGC AAGGATAAGTGTCGTAGACAC
Inv2	Edg_T1R02C7_DHP	GTGTCGTAGACACTCCCAATTCTGCGAACCCATATAA CAGTTGATGTGTCGTAGACAC
	Edg_T1R06C7_DHP	GTGTCGTAGACACCCATAAATCAAAAATCCAGAAAAC GAGAATGAGTGTCGTAGACAC

	Edg_T1R10C7_DHP	GTGTCGTAGACACGAAACACCAGAACGAGAGGCTTGC CCTGACGAGTGTCGTAGACAC
	Edg_T1R14C7_DHP	GTGTCGTAGACACGAACGAGGGTAGCAACGCGAAAGA CAGCATCGGTGTCGTAGACAC
	Edg_T1R18C7_DHP	GTGTCGTAGACACGGGATTTTGCTAAACAAATGAATT TTCTGTATGTGTCGTAGACAC
	Edg_T2R02C7_DHP	GTGTCGTAGACACGAGAGGGTTGATATAAGCGGATAA GTGCCGTCGTGTCGTAGACAC
	Edg_T2R06C7_DHP	GTGTCGTAGACACGCAGGTCAGACGATTGTTGACAGG AGGTTGAGGTGTCGTAGACAC
	Edg_T2R10C7_DHP	GTGTCGTAGACACGCGCCAAAGACAAAAGTTCATATG GTTTACCAGTGTCGTAGACAC
	Edg_T2R14C7_DHP	GTGTCGTAGACACTTTTTTGTTTAACGTCTCCAAATA AGAAACGAGTGTCGTAGACAC
	Edg_T2R18C7_DHP	GTGTCGTAGACACTAAACCAAGTACCGCATTCCAAGA ACGGGTATGTGTCGTAGACAC
	2G2T3C7R00	ATGTAAAGTAATTCTGTCAAAGTACCGACAAAAG
	2G2T3C7R02	ATAGTAGGGCTTAATTGAAAAGCCAACGCTCAAC
	2G2T3C7R04	CGAATGGTTTGAAATACCCTTCTGACCTAAATTT
	2G2T3C7R06	GAAGTCAATAGTGAATTTTTAAGACGCTGAGAAG
	2G2T3C7R10	CACAATATAATCCTGATTGATGATGGCAATTCAT
	2G2T3C7R12	TGGTTATCTAAAATATCTAAAGGAATTGAGGAAG
	2G2T3C7R16	CCTCGTCTGAAATGGATTACATTTTGACGCTCAA
	2G2T3C7R18	TCTTGATTAGTAATAACATTGTAGCAATACTTCT
	Edg_T3C7R20_pip i	AGGAACGGTACGCCAGTAAAGGGATTTTAGAC
	Edg_T4R02C7_DHP	GTGTCGTAGACACCGGGCGCTAGGGCGCTAAGAAAGC GAAAGGAGGTGTCGTAGACAC
	Edg_T4R06C7_DHP	GTGTCGTAGACACATCCTGTTTGATGGTGGCCCCAGC AGGCGAAAGTGTCGTAGACAC
	Edg_T4R10C7_DHP	GTGTCGTAGACACGTAACGCCAGGGTTTTAAGGCGAT TAAGTTGGGTGTCGTAGACAC
	Edg_T4R14C7_DHP	GTGTCGTAGACACTTTAAATTGTAAACGTATTGTATA AGCAAATAGTGTCGTAGACAC
	Edg_T4R18C7_DHP	GTGTCGTAGACACAAATTTTTAGAACCCTTTCAACGC AAGGATAAGTGTCGTAGACAC
Inv1_Trig	Edg_T1R02C7_DHP	GTGTCGTAGACACTCCCAATTCTGCGAACCCATATAA CAGTTGATGTGTCGTAGACAC
	Edg_T1R06C7_DHP	GTGTCGTAGACACCCATAAATCAAAAATCCAGAAAAC GAGAATGAGTGTCGTAGACAC
	Edg_T1R10C7_DHP	GTGTCGTAGACACGAAACACCAGAACGAGAGGCTTGC CCTGACGAGTGTCGTAGACAC
	Edg_T1R14C7_DHP	GTGTCGTAGACACGAACGAGGGTAGCAACGCGAAAGA CAGCATCGGTGTCGTAGACAC
	Edg_T1R18C7_DHP	GTGTCGTAGACACGGGATTTTGCTAAACAAATGAATT TTCTGTATGTGTCGTAGACAC

	2G4T2C7R00	ATAGCCACCACCCTCATTGAACCGCCACCCTCAG
	2G4T2C7R04	ATGTATAAACAGTTAATGTTGAGTAACAGTGCCC
	2G4T2C7R10	GGGCGCCAAAGACAAAAGTTCATATGGTTTACCA
	2G4T2C7R12	GCCCGAAGCCCTTTTTAAAGCAATAGCTATCTTA
	2G4T2C7R16	CCAACCTCCCGACTTGCGGCGAGGCGTTTTAGCG
	2G4T2C7R20	ACAGATAAGTCCTGAACACCTGTTTATCAACAAT
	2G3T3C7R00	ACGTAAAGTAATTCTGTCAAAGTACCGACAAAAG
	2G3T3C7R02	CTAGTAGGGCTTAATTGAAAAGCCAACGCTCAAC
	2G3T3C7R06	AAAGTCAATAGTGAATTTTTAAGACGCTGAGAAG
	2G3T3C7R10	ATCAATATAATCCTGATTGATGATGGCAATTCAT
	2G3T3C7R14	AGACATCGCCATTAAAAAAACTGATAGCCCTAAA
	2G3T3C7R16	TTTCGTCTGAAATGGATTACATTTTGACGCTCAA
	2G3T3C7R18	ACTTGATTAGTAATAACATTGTAGCAATACTTCT
	2G3T3C7R20	AGAGGAACGGTACGCCAGTAAAGGGATTTTAGAC
	Edg_T4R02C7_DHP	GTGTCGTAGACACCGGGCGCTAGGGCGCTAAGAAAGC GAAAGGAGGTGTCGTAGACAC
	Edg_T4R06C7_DHP	GTGTCGTAGACACATCCTGTTTGATGGTGGCCCCAGC AGGCGAAAGTGTCGTAGACAC
	Edg_T4R10C7_DHP	GTGTCGTAGACACGTAACGCCAGGGTTTTAAGGCGAT TAAGTTGGGTGTCGTAGACAC
	Edg_T4R14C7_DHP	GTGTCGTAGACACTTTAAATTGTAAACGTATTGTATA AGCAAATAGTGTCGTAGACAC
	Edg_T4R18C7_DHP	GTGTCGTAGACACAAATTTTTAGAACCCTTTCAACGC AAGGATAAGTGTCGTAGACAC
Inv2_Trig	Edg_T1R02C7_DHP	GTGTCGTAGACACTCCCAATTCTGCGAACCCATATAA CAGTTGATGTGTCGTAGACAC
	Edg_T1R06C7_DHP	GTGTCGTAGACACCCATAAATCAAAAATCCAGAAAAC GAGAATGAGTGTCGTAGACAC
	Edg_T1R10C7_DHP	GTGTCGTAGACACGAAACACCAGAACGAGAGGCTTGC CCTGACGAGTGTCGTAGACAC
	Edg_T1R14C7_DHP	GTGTCGTAGACACGAACGAGGGTAGCAACGCGAAAGA CAGCATCGGTGTCGTAGACAC
	Edg_T1R18C7_DHP	GTGTCGTAGACACGGGATTTTGCTAAACAAATGAATT TTCTGTATGTGTCGTAGACAC
	Edg_T2R02C7_DHP	GTGTCGTAGACACGAGAGGGTTGATATAAGCGGATAA GTGCCGTCGTGTCGTAGACAC
	Edg_T2R06C7_DHP	GTGTCGTAGACACGCAGGTCAGACGATTGTTGACAGG AGGTTGAGGTGTCGTAGACAC
	Edg_T2R10C7_DHP	GTGTCGTAGACACGCGCCAAAGACAAAAGTTCATATG GTTTACCAGTGTCGTAGACAC
	Edg_T2R14C7_DHP	GTGTCGTAGACACTTTTTTGTTTAACGTCTCCAAATA AGAAACGAGTGTCGTAGACAC
	Edg_T2R18C7_DHP	GTGTCGTAGACACTAAACCAAGTACCGCATTCCAAGA ACGGGTATGTGTCGTAGACAC
	2G2T3C7R00	ATGTAAAGTAATTCTGTCAAAGTACCGACAAAAG

	2G2T3C7R02	ATAGTAGGGCTTAATTGAAAAGCCAACGCTCAAC
	2G2T3C7R04	CGAATGGTTTGAAATACCCTTCTGACCTAAATTT
	2G2T3C7R06	GAAGTCAATAGTGAATTTTTAAGACGCTGAGAAG
	2G2T3C7R10	CACAATATAATCCTGATTGATGATGGCAATTCAT
	2G2T3C7R12	TGGTTATCTAAAATATCTAAAGGAATTGAGGAAG
	2G2T3C7R16	CCTCGTCTGAAATGGATTACATTTTGACGCTCAA
	2G2T3C7R18	TCTTGATTAGTAATAACATTGTAGCAATACTTCT
	Edg_T3C7R20_pip i	AGGAACGGTACGCCAGTAAAGGGATTTTAGAC
	2bRT4C7R00	GAGCACGTATAACGTGCTATGGTTGCTTTG
	2bRT4C7R04	ATCACCCAAATCAAGTGCCCACTACGTGAA
	2bRT4C7R08	GCTCACTGCCCGCTTTACATTAATTGCGTT
	2bRT4C7R10	GTAACGCCAGGGTTTTAAGGCGATTAAGTT
	2bRT4C7R16	GCCGGAGAGGGTAGCTTAGCTGATAAATTA
	2bRT4C7R20	TAAGCAATAAAGCCTCAAAGAATTAGCAAA

Table S5: Edge staples in scalable multi-step (Tic-Tac_Toe) reconfiguration measurements

Tile	Edge	Edge Seq
T1_TTT	Edg_T1R02C7_DHP	GTGTCGTAGACACTCCCAATTCTGCGAACCCATATAAC AGTTGATGTGTCGTAGACAC
	Edg_T1R06C7_DHP	GTGTCGTAGACACCCATAAATCAAAAATCCAGAAAACG AGAATGAGTGTCGTAGACAC
	Edg_T1R10C7_DHP	GTGTCGTAGACACGAAACACCAGAACGAGAGGCTTGCC CTGACGAGTGTCGTAGACAC
	Edg_T1R14C7_DHP	GTGTCGTAGACACGAACGAGGGTAGCAACGCGAAAGAC AGCATCGGTGTCGTAGACAC
	Edg_T1R18C7_DHP	GTGTCGTAGACACGGGATTTTGCTAAACAAATGAATTT TCTGTATGTGTCGTAGACAC
	5bC2RT2R00	CCACCCTCATTGAACCGCCACCCTCAG
	5bC2RT2R02	GGTTGATATAAGCGGATAAGTGCCGTC
	Q_T2R06	/5IAbRQ/GCAGGTCAGACGATTGTTGACAGGAGGTTG
	2bRT2C7R10	GCGCCAAAGACAAAAGTTCATATGGTTTAC
	2bRT2C7R14	TTTTTTGTTTAACGTCTCCAAATAAGAAAC
	2bRT2C7R16	AACCTCCCGACTTGCGGCGAGGCGTTTTAG
	2bRT2C7R18	TAAACCAAGTACCGCATTCCAAGAACGGGT
	2bRT2C7R20	AGATAAGTCCTGAACACCTGTTTATCAACA
	2bRT3C7R00	GTAAAGTAATTCTGTCAAAGTACCGACAAA
	2bRT3C7R02	AGTAGGGCTTAATTGAAAAGCCAACGCTCA
	2bRT3C7R06	AGTCAATAGTGAATTTTTAAGACGCTGAGA

	2bRT3C7R08	TGAGCAAAAGAAGATGATTCATTTCAATTA
	2bRT3C7R12	GTTATCTAAAATATCTAAAGGAATTGAGGA
	2bRT3C7R16	TCGTCTGAAATGGATTACATTTTGACGCTC
	Edg_T4R02C7_DHP	GTGTCGTAGACACCGGGCGCTAGGGCGCTAAGAAAGCG AAAGGAGGTGTCGTAGACAC
	Edg_T4R06C7_DHP	GTGTCGTAGACACATCCTGTTTGATGGTGGCCCCAGCA GGCGAAAGTGTCGTAGACAC
	Edg_T4R10C7_DHP	GTGTCGTAGACACGTAACGCCAGGGTTTTAAGGCGATT AAGTTGGGTGTCGTAGACAC
	Edg_T4R14C7_DHP	GTGTCGTAGACACTTTAAATTGTAAACGTATTGTATAA GCAAATAGTGTCGTAGACAC
	Edg_T4R18C7_DHP	GTGTCGTAGACACAAATTTTTAGAACCCTTTCAACGCA AGGATAAGTGTCGTAGACAC
T2_TTT	Edg_T1R02C7_DHP	GTGTCGTAGACACTCCCAATTCTGCGAACCCATATAAC AGTTGATGTGTCGTAGACAC
	Edg_T1R06C7_DHP	GTGTCGTAGACACCCATAAATCAAAAATCCAGAAAACG AGAATGAGTGTCGTAGACAC
	Edg_T1R10C7_DHP	GTGTCGTAGACACGAAACACCAGAACGAGAGGCTTGCC CTGACGAGTGTCGTAGACAC
	Edg_T1R14C7_DHP	GTGTCGTAGACACGAACGAGGGTAGCAACGCGAAAGAC AGCATCGGTGTCGTAGACAC
	Edg_T1R18C7_DHP	GTGTCGTAGACACGGGATTTTGCTAAACAAATGAATTT TCTGTATGTGTCGTAGACAC
	5bRT2C7R00	AGCCACCACCCTCATTGAACCGCCACC
	5bRT2C7R02	GAGAGGGTTGATATAAGCGGATAAGTG
	2G4T2C7R04	ATGTATAAACAGTTAATGTTGAGTAACAGTGCCC
	2G4T2C7R08	CATAGCGCGTTTTCATCGCTTTAGCGTCAGACTG
	2G4T2C7R12	GCCCGAAGCCCTTTTTAAAGCAATAGCTATCTTA
	2G4T2C7R14	AATTTTTTGTTTAACGTCTCCAAATAAGAAACGA
	2G4T2C7R18	AGTAAACCAAGTACCGCATTCCAAGAACGGGTAT
	2G4T2C7R20	ACAGATAAGTCCTGAACACCTGTTTATCAACAAT
	2G1T3C7R00	GTGTAAAGTAATTCTGTCAAAGTACCGACAAAAG
	2G1T3C7R02	ATAGTAGGGCTTAATTGAAAAGCCAACGCTCAAC
	2G1T3C7R08	GCTGAGCAAAAGAAGATGATTCATTTCAATTACC
	2G1T3C7R10	GACAATATAATCCTGATTGATGATGGCAATTCAT
	2G1T3C7R14	GAACATCGCCATTAAAAAAACTGATAGCCCTAAA
	2G1T3C7R16	TATCGTCTGAAATGGATTACATTTTGACGCTCAA
	5bRT3C7R18	TTGATTAGTAATAACATTGTAGCAATA
	5bRT3C7R20	AGGAACGGTACGCCAGTAAAGGGATTT
	2G2T4C7R00	ATGAGCACGTATAACGTGCTATGGTTGCTTTGAC
	2G2T4C7R02	ATCGGGCGCTAGGGCGCTAAGAAAGCGAAAGGAG
	2G2T4C7R04	CGATCACCCAAATCAAGTGCCCACTACGTGAACC
	2G2T4C7R06	GAATCCTGTTTGATGGTGGCCCCAGCAGGCGAAA

	2G2T4C7R10	CAGTAACGCCAGGGTTTTAAGGCGATTAAGTTGG
	2G2T4C7R14	AGTTTAAATTGTAAACGTATTGTATAAGCAAATA
T3_TTT	Edg_T1R02C7_DHP	GTGTCGTAGACACTCCCAATTCTGCGAACCCATATAAC AGTTGATGTGTCGTAGACAC
	Edg_T1R06C7_DHP	GTGTCGTAGACACCCATAAATCAAAAATCCAGAAAACG AGAATGAGTGTCGTAGACAC
	Edg_T1R10C7_DHP	GTGTCGTAGACACGAAACACCAGAACGAGAGGCTTGCC CTGACGAGTGTCGTAGACAC
	Edg_T1R14C7_DHP	GTGTCGTAGACACGAACGAGGGTAGCAACGCGAAAGAC AGCATCGGTGTCGTAGACAC
	Edg_T1R18C7_DHP	GTGTCGTAGACACGGGATTTTGCTAAACAAATGAATTT TCTGTATGTGTCGTAGACAC
	Edg_T2R02C7_DHP	GTGTCGTAGACACGAGAGGGTTGATATAAGCGGATAAG TGCCGTCGTGTCGTAGACAC
	Edg_T2R06C7_DHP	GTGTCGTAGACACGCAGGTCAGACGATTGTTGACAGGA GGTTGAGGTGTCGTAGACAC
	Edg_T2R10C7_DHP	GTGTCGTAGACACGCGCCAAAGACAAAAGTTCATATGG TTTACCAGTGTCGTAGACAC
	Edg_T2R14C7_DHP	GTGTCGTAGACACTTTTTTGTTTAACGTCTCCAAATAA GAAACGAGTGTCGTAGACAC
	Edg_T2R18C7_DHP	GTGTCGTAGACACTAAACCAAGTACCGCATTCCAAGAA CGGGTATGTGTCGTAGACAC
	5bC2RT3R00	GTAATTCTGTCAAAGTACCGACAAAAG
	5bC2RT3R02	GGCTTAATTGAAAAGCCAACGCTCAAC
	2bRT3C7R06	AGTCAATAGTGAATTTTTAAGACGCTGAGA
	2bRT3C7R10	CAATATAATCCTGATTGATGATGGCAATTC
	2bRT3C7R14	ACATCGCCATTAAAAAAACTGATAGCCCTA
	2bRT3C7R16	TCGTCTGAAATGGATTACATTTTGACGCTC
	2bRT3C7R18	TTGATTAGTAATAACATTGTAGCAATACTT
	2bRT3C7R20	AGGAACGGTACGCCAGTAAAGGGATTTTAG
	2bRT4C7R00	GAGCACGTATAACGTGCTATGGTTGCTTTG
	2bRT4C7R02	CGGGCGCTAGGGCGCTAAGAAAGCGAAAGG
	2bRT4C7R06	ATCCTGTTTGATGGTGGCCCCAGCAGGCGA
	2bRT4C7R08	GCTCACTGCCCGCTTTACATTAATTGCGTT
	2bRT4C7R12	CGTTGGTGTAGATGGGGTAATGGGATAGGT
	2bRT4C7R16	GCCGGAGAGGGTAGCTTAGCTGATAAATTA
T4_TTT	5bRT1C7R00	GGTGGCATCAATTCTAGGGCGCGAGCT
	5bRT1C7R02	TCCCAATTCTGCGAACCCATATAACAG
	Q_T1R04	/5IAbRQ/AAATTGCTCCTTTTGATATTAGAGAGTACC TTTA
	2G3T1C7R08	AGCGAGGCATAGTAAGAGACGCCAAAAGGAATTA
	2G3T1C7R12	CCCTGATAAATTGTGTCGAGATTTGTATCATCGC
	2G3T1C7R14	AGGAACGAGGGTAGCAACGCGAAAGACAGCATCG

	2G3T1C7R18	ACGGGATTTTGCTAAACAAATGAATTTTCTGTAT
	2G3T1C7R20	AGACAAACTACAACGCCTGAGTTTCGTCACCAGT
	2G4T2C7R00	ATAGCCACCACCCTCATTGAACCGCCACCCTCAG
	2G4T2C7R02	AAGAGAGGGTTGATATAAGCGGATAAGTGCCGTC
	2G4T2C7R08	CATAGCGCGTTTTCATCGCTTTAGCGTCAGACTG
	2G4T2C7R10	GGGCGCCAAAGACAAAAGTTCATATGGTTTACCA
	2G4T2C7R14	AATTTTTTGTTTAACGTCTCCAAATAAGAAACGA
	2G4T2C7R16	CCAACCTCCCGACTTGCGGCGAGGCGTTTTAGCG
	5bRT2C7R18	TAAACCAAGTACCGCATTCCAAGAACG
	5bRT2C7R20	AGATAAGTCCTGAACACCTGTTTATCA
	2G1T3C7R00	GTGTAAAGTAATTCTGTCAAAGTACCGACAAAAG
	2G1T3C7R02	ATAGTAGGGCTTAATTGAAAAGCCAACGCTCAAC
	2G1T3C7R04	TCAATGGTTTGAAATACCCTTCTGACCTAAATTT
	2G1T3C7R06	CGAGTCAATAGTGAATTTTTAAGACGCTGAGAAG
	2G1T3C7R10	GACAATATAATCCTGATTGATGATGGCAATTCAT
	2G1T3C7R14	GAACATCGCCATTAAAAAAACTGATAGCCCTAAA
	Edg_T4R02C7_DHP	GTGTCGTAGACACCGGGCGCTAGGGCGCTAAGAAAGCG AAAGGAGGTGTCGTAGACAC
	Edg_T4R06C7_DHP	GTGTCGTAGACACATCCTGTTTGATGGTGGCCCCAGCA GGCGAAAGTGTCGTAGACAC
	Edg_T4R10C7_DHP	GTGTCGTAGACACGTAACGCCAGGGTTTTAAGGCGATT AAGTTGGGTGTCGTAGACAC
	Edg_T4R14C7_DHP	GTGTCGTAGACACTTTAAATTGTAAACGTATTGTATAA GCAAATAGTGTCGTAGACAC
	Edg_T4R18C7_DHP	GTGTCGTAGACACAAATTTTTAGAACCCTTTCAACGCA AGGATAAGTGTCGTAGACAC
T5_TTT	2bRT1C7R04	ATTGCTCCTTTTGATATTAGAGAGTACCTT
	2bRT1C7R06	CCATAAATCAAAAATCCAGAAAACGAGAAT
	2bRT1C7R10	GAAACACCAGAACGAGAGGCTTGCCCTGAC
	2bRT1C7R12	CTGATAAATTGTGTCGAGATTTGTATCATC
	2bRT1C7R18	GGGATTTTGCTAAACAAATGAATTTTCTGT
	2bRT1C7R20	ACAAACTACAACGCCTGAGTTTCGTCACCA
	2bRT2C7R04	GTATAAACAGTTAATGTTGAGTAACAGTGC
	2bRT2C7R06	GCAGGTCAGACGATTGTTGACAGGAGGTTG
	2bRT2C7R10	GCGCCAAAGACAAAAGTTCATATGGTTTAC
	2bRT2C7R12	CCGAAGCCCTTTTTAAAGCAATAGCTATCT
	2bRT2C7R18	TAAACCAAGTACCGCATTCCAAGAACGGGT
	2bRT2C7R20	AGATAAGTCCTGAACACCTGTTTATCAACA
	2bRT3C7R04	AATGGTTTGAAATACCCTTCTGACCTAAAT
	2bRT3C7R06	AGTCAATAGTGAATTTTTAAGACGCTGAGA
	2bRT3C7R10	CAATATAATCCTGATTGATGATGGCAATTC

	2bRT3C7R12	GTTATCTAAAATATCTAAAGGAATTGAGGA
	2bRT3C7R18	TTGATTAGTAATAACATTGTAGCAATACTT
	2bRT3C7R20	AGGAACGGTACGCCAGTAAAGGGATTTTAG
	2bRT4C7R04	ATCACCCAAATCAAGTGCCCACTACGTGAA
	2bRT4C7R06	ATCCTGTTTGATGGTGGCCCCAGCAGGCGA
	2bRT4C7R10	GTAACGCCAGGGTTTTAAGGCGATTAAGTT
	2bRT4C7R12	CGTTGGTGTAGATGGGGTAATGGGATAGGT
	2bRT4C7R18	AAATTTTTAGAACCCTTTCAACGCAAGGAT
	Q_T4R20	TAAGCAATAAAGCCTCAAAGAATTAGCAAA/3IAbRQSp/
T6_TTT	2G3T1C7R00	ACGGTGGCATCAATTCTAGGGCGCGAGCTGAAAA
	2G3T1C7R02	CTTCCCAATTCTGCGAACCCATATAACAGTTGAT
	2G3T1C7R04	AAATTGCTCCTTTTGATATTAGAGAGTACCTTTA
	2G3T1C7R06	AACCATAAATCAAAAATCCAGAAAACGAGAATGA
	2G3T1C7R10	ATGAAACACCAGAACGAGAGGCTTGCCCTGACGA
	2G3T1C7R14	AGGAACGAGGGTAGCAACGCGAAAGACAGCATCG
	Edg_T2R02C7_DHP	GTGTCGTAGACACGAGAGGGTTGATATAAGCGGATAAGTGCCGTCGTGTCGTAGACAC
	Edg_T2R06C7_DHP	GTGTCGTAGACACGCAGGTCAGACGATTGTTGACAGGAGGTTGAGGTGTCGTAGACAC
	Edg_T2R10C7_DHP	GTGTCGTAGACACGCGCCAAAGACAAAAGTTCATATGGTTTACCAGTGTCGTAGACAC
	Edg_T2R14C7_DHP	GTGTCGTAGACACTTTTTTGTTTAACGTCTCCAAATAAGAAACGAGTGTCGTAGACAC
	Edg_T2R18C7_DHP	GTGTCGTAGACACTAAACCAAGTACCGCATTCCAAGAACGGGTATGTGTCGTAGACAC
	5bRT3C7R00	GTAAAGTAATTCTGTCAAAGTACCGAC
	5bRT3C7R02	AGTAGGGCTTAATTGAAAAGCCAACGC
	2G1T3C7R04	TCAATGGTTTGAAATACCCTTCTGACCTAAATTT
	2G1T3C7R08	GCTGAGCAAAAGAAGATGATTCATTTCAATTACC
	2G1T3C7R12	TAGTTATCTAAAATATCTAAAGGAATTGAGGAAG
	2G1T3C7R14	GAACATCGCCATTAAAAAAACTGATAGCCCTAAA
	2G1T3C7R18	ATTTGATTAGTAATAACATTGTAGCAATACTTCT
	2G1T3C7R20	AAAGGAACGGTACGCCAGTAAAGGGATTTTAGAC
	2G2T4C7R00	ATGAGCACGTATAACGTGCTATGGTTGCTTTGAC
	2G2T4C7R02	ATCGGGCGCTAGGGCGCTAAGAAAGCGAAAGGAG
	2G2T4C7R08	TAGCTCACTGCCCGCTTTACATTAATTGCGTTGC
	2G2T4C7R10	CAGTAACGCCAGGGTTTTAAGGCGATTAAGTTGG
	2G2T4C7R14	AGTTTAAATTGTAAACGTATTGTATAAGCAAATA
	2G2T4C7R16	CCGCCGGAGAGGGTAGCTTAGCTGATAAATTAAT
	5bRT4C7R18	AAATTTTTAGAACCCTTTCAACGCAAG
	5bRT4C7R20	TAAGCAATAAAGCCTCAAAGAATTAGC

T7_TTT	5bC2RT1R00	CATCAATTCTAGGGCGCGAGCTGAAAA
	5bC2RT1R02	ATTCTGCGAACCCATATAACAGTTGAT
	2bRT1C7R06	CCATAAATCAAAAATCCAGAAAACGAGAAT
	2bRT1C7R10	GAAACACCAGAACGAGAGGCTTGCCCTGAC
	2bRT1C7R14	GAACGAGGGTAGCAACGCGAAAGACAGCAT
	2bRT1C7R16	GGTTTATCAGCTTGCTAGCCTTTAATTGTA
	2bRT1C7R18	GGGATTTTGCTAAACAAATGAATTTTCTGT
	2bRT1C7R20	ACAAACTACAACGCCTGAGTTTCGTCACCA
	2bRT2C7R00	AGCCACCACCCTCATTGAACCGCCACCCTC
	2bRT2C7R02	GAGAGGGTTGATATAAGCGGATAAGTGCCG
	2bRT2C7R06	GCAGGTCAGACGATTGTTGACAGGAGGTTG
	2bRT2C7R08	TAGCGCGTTTTCATCGCTTTAGCGTCAGAC
	2bRT2C7R12	CCGAAGCCCTTTTTAAAGCAATAGCTATCT
	2bRT2C7R16	AACCTCCCGACTTGCGGCGAGGCGTTTTAG
	Edg_T3R02C7_DHP	GTGTCGTAGACACAGTAGGGCTTAATTGAAAAGCCAAC GCTCAACGTGTCGTAGACAC
	Edg_T3R06C7_DHP	GTGTCGTAGACACAGTCAATAGTGAATTTTTAAGACGC TGAGAAGGTGTCGTAGACAC
	Edg_T3R10C7_DHP	GTGTCGTAGACACCAATATAATCCTGATTGATGATGGC AATTCATGTGTCGTAGACAC
	Edg_T3R14C7_DHP	GTGTCGTAGACACACATCGCCATTAAAAAAACTGATAG CCCTAAAGTGTCGTAGACAC
	Edg_T3R18C7_DHP	GTGTCGTAGACACTTGATTAGTAATAACATTGTAGCAA TACTTCTGTGTCGTAGACAC
	Edg_T4R02C7_DHP	GTGTCGTAGACACCGGGCGCTAGGGCGCTAAGAAAGCG AAAGGAGGTGTCGTAGACAC
	Edg_T4R06C7_DHP	GTGTCGTAGACACATCCTGTTTGATGGTGGCCCCAGCA GGCGAAAGTGTCGTAGACAC
	Edg_T4R10C7_DHP	GTGTCGTAGACACGTAACGCCAGGGTTTTAAGGCGATT AAGTTGGGTGTCGTAGACAC
	Edg_T4R14C7_DHP	GTGTCGTAGACACTTTAAATTGTAAACGTATTGTATAA GCAAATAGTGTCGTAGACAC
	Edg_T4R18C7_DHP	GTGTCGTAGACACAAATTTTTAGAACCCTTTCAACGCA AGGATAAGTGTCGTAGACAC
T8_TTT	2G3T1C7R00	ACGGTGGCATCAATTCTAGGGCGCGAGCTGAAAA
	2G3T1C7R02	CTTCCCAATTCTGCGAACCCATATAACAGTTGAT
	2G3T1C7R08	AGCGAGGCATAGTAAGAGACGCCAAAAGGAATTA
	2G3T1C7R10	ATGAAACACCAGAACGAGAGGCTTGCCCTGACGA
	2G3T1C7R14	AGGAACGAGGGTAGCAACGCGAAAGACAGCATCG
	2G3T1C7R16	TTGGTTTATCAGCTTGCTAGCCTTTAATTGTATC
	5bRT1C7R18	GGGATTTTGCTAAACAAATGAATTTTC
	5bRT1C7R20	ACAAACTACAACGCCTGAGTTTCGTCA
	2G4T2C7R00	ATAGCCACCACCCTCATTGAACCGCCACCCTCAG

	2G4T2C7R02	AAGAGAGGGTTGATATAAGCGGATAAGTGCCGTC
	2G4T2C7R04	ATGTATAAACAGTTAATGTTGAGTAACAGTGCCC
	2G4T2C7R06	TAGCAGGTCAGACGATTGTTGACAGGAGGTTGAG
	2G4T2C7R10	GGGCGCCAAAGACAAAAGTTCATATGGTTTACCA
	2G4T2C7R14	AATTTTTTGTTTAACGTCTCCAAATAAGAAACGA
	Edg_T3R02C7_DHP	GTGTCGTAGACACAGTAGGGCTTAATTGAAAAGCCAAC GCTCAACGTGTCGTAGACAC
	Edg_T3R06C7_DHP	GTGTCGTAGACACAGTCAATAGTGAATTTTTAAGACGC TGAGAAGGTGTCGTAGACAC
	Edg_T3R10C7_DHP	GTGTCGTAGACACCAATATAATCCTGATTGATGATGGC AATTCATGTGTCGTAGACAC
	Edg_T3R14C7_DHP	GTGTCGTAGACACACATCGCCATTAAAAAAACTGATAG CCCTAAAGTGTCGTAGACAC
	Edg_T3R18C7_DHP	GTGTCGTAGACACTTGATTAGTAATAACATTGTAGCAA TACTTCTGTGTCGTAGACAC
	5bRT4C7R00	GAGCACGTATAACGTGCTATGGTTGCT
	5bRT4C7R02	CGGGCGCTAGGGCGCTAAGAAAGCGAA
	2G2T4C7R04	CGATCACCCAAATCAAGTGCCCACTACGTGAACC
	2G2T4C7R08	TAGCTCACTGCCCGCTTTACATTAATTGCGTTGC
	2G2T4C7R12	TGCGTTGGTGTAGATGGGGTAATGGGATAGGTCA
	2G2T4C7R14	AGTTTAAATTGTAAACGTATTGTATAAGCAAATA
	2G2T4C7R18	TCAAATTTTTAGAACCCTTTCAACGCAAGGATAA
	2G2T4C7R20	AGTAAGCAATAAAGCCTCAAAGAATTAGCAAAAT
T9_TTT	2bRT1C7R00	GGTGGCATCAATTCTAGGGCGCGAGCTGAA
	2bRT1C7R02	TCCCAATTCTGCGAACCCATATAACAGTTG
	2bRT1C7R06	CCATAAATCAAAAATCCAGAAAACGAGAAT
	2bRT1C7R08	CGAGGCATAGTAAGAGACGCCAAAAGGAAT
	2bRT1C7R12	CTGATAAATTGTGTCGAGATTTGTATCATC
	2bRT1C7R16	GGTTTATCAGCTTGCTAGCCTTTAATTGTA
	Edg_T2R02C7_DHP	GTGTCGTAGACACGAGAGGGTTGATATAAGCGGATAAG TGCCGTCGTGTCGTAGACAC
	Edg_T2R06C7_DHP	GTGTCGTAGACACGCAGGTCAGACGATTGTTGACAGGA GGTTGAGGTGTCGTAGACAC
	Edg_T2R10C7_DHP	GTGTCGTAGACACGCGCCAAAGACAAAAGTTCATATGG TTTACCAGTGTCGTAGACAC
	Edg_T2R14C7_DHP	GTGTCGTAGACACTTTTTTGTTTAACGTCTCCAAATAA GAAACGAGTGTCGTAGACAC
	Edg_T2R18C7_DHP	GTGTCGTAGACACTAAACCAAGTACCGCATTCCAAGAA CGGGTATGTGTCGTAGACAC
	Edg_T3R02C7_DHP	GTGTCGTAGACACAGTAGGGCTTAATTGAAAAGCCAAC GCTCAACGTGTCGTAGACAC
	Edg_T3R06C7_DHP	GTGTCGTAGACACAGTCAATAGTGAATTTTTAAGACGC TGAGAAGGTGTCGTAGACAC
	Edg_T3R10C7_DHP	GTGTCGTAGACACCAATATAATCCTGATTGATGATGGC

		AATTCATGTGTCGTAGACAC
	Edg_T3R14C7_DHP	GTGTCGTAGACACACATCGCCATTAAAAAAACTGATAG CCCTAAAGTGTCGTAGACAC
	Edg_T3R18C7_DHP	GTGTCGTAGACACTTGATTAGTAATAACATTGTAGCAA TACTTCTGTGTCGTAGACAC
	5bC2RT4R00	CGTATAACGTGCTATGGTTGCTTTGAC
	5bC2RT4R02	GCTAGGGCGCTAAGAAAGCGAAAGGAG
	2bRT4C7R06	ATCCTGTTTGATGGTGGCCCCAGCAGGCGA
	2bRT4C7R10	GTAACGCCAGGGTTTTAAGGCGATTAAGTT
	2bRT4C7R14	TTTAAATTGTAAACGTATTGTATAAGCAAA
	2bRT4C7R16	GCCGGAGAGGGTAGCTTAGCTGATAAATTA
	2bRT4C7R18	AAATTTTTAGAACCCTTTCAACGCAAGGAT
	2bRT4C7R20	TAAGCAATAAAGCCTCAAAGAATTAGCAAA
T2_TTT_X	Edg_T1R02C7_DHP	GTGTCGTAGACACTCCCAATTCTGCGAACCCATATAAC AGTTGATGTGTCGTAGACAC
	Edg_T1R06C7_DHP	GTGTCGTAGACACCCATAAATCAAAAATCCAGAAAACG AGAATGAGTGTCGTAGACAC
	Edg_T1R10C7_DHP	GTGTCGTAGACACGAAACACCAGAACGAGAGGCTTGCC CTGACGAGTGTCGTAGACAC
	Edg_T1R14C7_DHP	GTGTCGTAGACACGAACGAGGGTAGCAACGCGAAAGAC AGCATCGGTGTCGTAGACAC
	Edg_T1R18C7_DHP	GTGTCGTAGACACGGGATTTTGCTAAACAAATGAATTT TCTGTATGTGTCGTAGACAC
	5bRT2C7R00	AGCCACCACCCTCATTGAACCGCCACC
	5bRT2C7R02	GAGAGGGTTGATATAAGCGGATAAGTG
	2G4T2C7R04	ATGTATAAACAGTTAATGTTGAGTAACAGTGCCC
	2G4T2C7R08	CATAGCGCGTTTTCATCGCTTTAGCGTCAGACTG
	2G4T2C7R12	GCCCGAAGCCCTTTTTAAAGCAATAGCTATCTTA
	2G4T2C7R14	AATTTTTTGTTTAACGTCTCCAAATAAGAAACGA
	2G4T2C7R18	AGTAAACCAAGTACCGCATTCCAAGAACGGGTAT
	2G4T2C7R20	ACAGATAAGTCCTGAACACCTGTTTATCAACAAT
	2G1T3C7R00	GTGTAAAGTAATTCTGTCAAAGTACCGACAAAAG
	2G1T3C7R02	ATAGTAGGGCTTAATTGAAAAGCCAACGCTCAAC
	2G1T3C7R08	GCTGAGCAAAAGAAGATGATTCATTTCAATTACC
	2G1T3C7R10	GACAATATAATCCTGATTGATGATGGCAATTCAT
	2G1T3C7R14	GAACATCGCCATTAAAAAAACTGATAGCCCTAAA
	2G1T3C7R16	TATCGTCTGAAATGGATTACATTTTGACGCTCAA
	5bRT3C7R18	TTGATTAGTAATAACATTGTAGCAATA
	5bRT3C7R20	AGGAACGGTACGCCAGTAAAGGGATTT
	2G2T4C7R00	ATGAGCACGTATAACGTGCTATGGTTGCTTTGAC
	2G2T4C7R02	ATCGGGCGCTAGGGCGCTAAGAAAGCGAAAGGAG
	2G2T4C7R04	CGATCACCCAAATCAAGTGCCCACTACGTGAACC

	2G2T4C7R06	GAATCCTGTTTGATGGTGGCCCCAGCAGGCGAAA
	2G2T4C7R10	CAGTAACGCCAGGGTTTTAAGGCGATTAAGTTGG
	2G2T4C7R14	AGTTTAAATTGTAAACGTATTGTATAAGCAAATA
	5bC2G2T4R18	AAATTTTTAGAACCCTTTCAACGCAAGGATAAGAGAG
	5bC2G2T4R20	TAAGCAATAAAGCCTCAAAGAATTAGCAAAATAGCCA
T3_TTT_X	Edg_T1R02C7_DHP	GTGTCGTAGACACTCCCAATTCTGCGAACCCATATAAC AGTTGATGTGTCGTAGACAC
	Edg_T1R06C7_DHP	GTGTCGTAGACACCCATAAATCAAAAATCCAGAAAACG AGAATGAGTGTCGTAGACAC
	Edg_T1R10C7_DHP	GTGTCGTAGACACGAAACACCAGAACGAGAGGCTTGCC CTGACGAGTGTCGTAGACAC
	Edg_T1R14C7_DHP	GTGTCGTAGACACGAACGAGGGTAGCAACGCGAAAGAC AGCATCGGTGTCGTAGACAC
	Edg_T1R18C7_DHP	GTGTCGTAGACACGGGATTTTGCTAAACAAATGAATTT TCTGTATGTGTCGTAGACAC
	Edg_T2R02C7_DHP	GTGTCGTAGACACGAGAGGGTTGATATAAGCGGATAAG TGCCGTCGTGTCGTAGACAC
	Edg_T2R06C7_DHP	GTGTCGTAGACACGCAGGTCAGACGATTGTTGACAGGA GGTTGAGGTGTCGTAGACAC
	Edg_T2R10C7_DHP	GTGTCGTAGACACGCGCCAAAGACAAAAGTTCATATGG TTTACCAGTGTCGTAGACAC
	Edg_T2R14C7_DHP	GTGTCGTAGACACTTTTTTGTTTAACGTCTCCAAATAA GAAACGAGTGTCGTAGACAC
	Edg_T2R18C7_DHP	GTGTCGTAGACACTAAACCAAGTACCGCATTCCAAGAA CGGGTATGTGTCGTAGACAC
	5bC2RT3R00	GTAATTCTGTCAAAGTACCGACAAAAG
	5bC2RT3R02	GGCTTAATTGAAAAGCCAACGCTCAAC
	2bRT3C7R06	AGTCAATAGTGAATTTTTAAGACGCTGAGA
	2bRT3C7R10	CAATATAATCCTGATTGATGATGGCAATTC
	2bRT3C7R14	ACATCGCCATTAAAAAAACTGATAGCCCTA
	2bRT3C7R16	TCGTCTGAAATGGATTACATTTTGACGCTC
	2bRT3C7R18	TTGATTAGTAATAACATTGTAGCAATACTT
	2bRT3C7R20	AGGAACGGTACGCCAGTAAAGGGATTTTAG
	2bRT4C7R00	GAGCACGTATAACGTGCTATGGTTGCTTTG
	2bRT4C7R02	CGGGCGCTAGGGCGCTAAGAAAGCGAAAGG
	2bRT4C7R06	ATCCTGTTTGATGGTGGCCCCAGCAGGCGA
	2bRT4C7R08	GCTCACTGCCCGCTTTACATTAATTGCGTT
	2bRT4C7R12	CGTTGGTGTAGATGGGGTAATGGGATAGGT
	2bRT4C7R16	GCCGGAGAGGGTAGCTTAGCTGATAAATTA
	5bG2T4C7R18	CCGTCAAATTTTTAGAACCCTTTCAACGCAAGGATAA
	5bG2T4C7R20	CTCAGTAAGCAATAAAGCCTCAAAGAATTAGCAAAAT
T4_TTT_X	5bRT1C7R00	GGTGGCATCAATTCTAGGGCGCGAGCT
	5bRT1C7R02	TCCCAATTCTGCGAACCCATATAACAG

	2G3T1C7R04	AAATTGCTCCTTTTGATATTAGAGAGTACCTTTA
	2G3T1C7R08	AGCGAGGCATAGTAAGAGACGCCAAAAGGAATTA
	2G3T1C7R12	CCCTGATAAATTGTGTCGAGATTTGTATCATCGC
	2G3T1C7R14	AGGAACGAGGGTAGCAACGCGAAAGACAGCATCG
	2G3T1C7R18	ACGGGATTTTGCTAAACAAATGAATTTTCTGTAT
	2G3T1C7R20	AGACAAACTACAACGCCTGAGTTTCGTCACCAGT
	2G4T2C7R00	ATAGCCACCACCCTCATTGAACCGCCACCCTCAG
	2G4T2C7R02	AAGAGAGGGTTGATATAAGCGGATAAGTGCCGTC
	2G4T2C7R08	CATAGCGCGTTTTCATCGCTTTAGCGTCAGACTG
	2G4T2C7R10	GGGCGCCAAAGACAAAAGTTCATATGGTTTACCA
	2G4T2C7R14	AATTTTTTGTTTAACGTCTCCAAATAAGAAACGA
	2G4T2C7R16	CCAACCTCCCGACTTGCGGCGAGGCGTTTTAGCG
	5bRT2C7R18	TAAACCAAGTACCGCATTCCAAGAACG
	5bRT2C7R20	AGATAAGTCCTGAACACCTGTTTATCA
	2G1T3C7R00	GTGTAAAGTAATTCTGTCAAAGTACCGACAAAAG
	2G1T3C7R02	ATAGTAGGGCTTAATTGAAAAGCCAACGCTCAAC
	2G1T3C7R04	TCAATGGTTTGAAATACCCTTCTGACCTAAATTT
	2G1T3C7R06	CGAGTCAATAGTGAATTTTTAAGACGCTGAGAAG
	2G1T3C7R10	GACAATATAATCCTGATTGATGATGGCAATTCAT
	2G1T3C7R14	GAACATCGCCATTAAAAAAACTGATAGCCCTAAA
	5bC2G1T3R18	TTGATTAGTAATAACATTGTAGCAATACTTCTTCCCA
	5bC2G1T3R20	AGGAACGGTACGCCAGTAAAGGGATTTTAGACGGTGG
	Edg_T4R02C7_DHP	GTGTCGTAGACACCGGGCGCTAGGGCGCTAAGAAAGCG AAAGGAGGTGTCGTAGACAC
	Edg_T4R06C7_DHP	GTGTCGTAGACACATCCTGTTTGATGGTGGCCCCAGCA GGCGAAAGTGTCGTAGACAC
	Edg_T4R10C7_DHP	GTGTCGTAGACACGTAACGCCAGGGTTTTAAGGCGATT AAGTTGGGTGTCGTAGACAC
	Edg_T4R14C7_DHP	GTGTCGTAGACACTTTAAATTGTAAACGTATTGTATAA GCAAATAGTGTCGTAGACAC
	Edg_T4R18C7_DHP	GTGTCGTAGACACAAATTTTTAGAACCCTTTCAACGCA AGGATAAGTGTCGTAGACAC
T5_TTT_X	5bG3T1C7R00	TAGACGGTGGCATCAATTCTAGGGCGCGAGCTGAAAA
	5bG3T1C7R02	CTTCTTCCCAATTCTGCGAACCCATATAACAGTTGAT
	2bRT1C7R04	ATTGCTCCTTTTGATATTAGAGAGTACCTT
	2bRT1C7R06	CCATAAATCAAAAATCCAGAAAACGAGAAT
	2bRT1C7R10	GAAACACCAGAACGAGAGGCTTGCCCTGAC
	2bRT1C7R12	CTGATAAATTGTGTCGAGATTTGTATCATC
	2bRT1C7R18	GGGATTTTGCTAAACAAATGAATTTTCTGT
	2bRT1C7R20	ACAAACTACAACGCCTGAGTTTCGTCACCA
	5bG4T2C7R00	AAAATAGCCACCACCCTCATTGAACCGCCACCCTCAG

	5bG4T2C7R02	GATAAGAGAGGGTTGATATAAGCGGATAAGTGCCGTC
	2bRT2C7R04	GTATAAACAGTTAATGTTGAGTAACAGTGC
	2bRT2C7R06	GCAGGTCAGACGATTGTTGACAGGAGGTTG
	2bRT2C7R10	GCGCCAAAGACAAAAGTTCATATGGTTTAC
	2bRT2C7R12	CCGAAGCCCTTTTTAAAGCAATAGCTATCT
	2bRT2C7R18	TAAACCAAGTACCGCATTCCAAGAACGGGT
	2bRT2C7R20	AGATAAGTCCTGAACACCTGTTTATCAACA
	5bG1T3C7R00	CCAGTGTAAAGTAATTCTGTCAAAGTACCGACAAAAG
	5bG1T3C7R02	TGTATAGTAGGGCTTAATTGAAAAGCCAACGCTCAAC
	2bRT3C7R04	AATGGTTTGAAATACCCTTCTGACCTAAAT
	2bRT3C7R06	AGTCAATAGTGAATTTTTAAGACGCTGAGA
	2bRT3C7R10	CAATATAATCCTGATTGATGATGGCAATTC
	2bRT3C7R12	GTTATCTAAAATATCTAAAGGAATTGAGGA
	2bRT3C7R18	TTGATTAGTAATAACATTGTAGCAATACTT
	2bRT3C7R20	AGGAACGGTACGCCAGTAAAGGGATTTTAG
	5bG2T4C7R00	ACAATGAGCACGTATAACGTGCTATGGTTGCTTTGAC
	5bG2T4C7R02	GGTATCGGGCGCTAGGGCGCTAAGAAAGCGAAAGGAG
	2bRT4C7R04	ATCACCCAAATCAAGTGCCCACTACGTGAA
	2bRT4C7R06	ATCCTGTTTGATGGTGGCCCCAGCAGGCGA
	2bRT4C7R10	GTAACGCCAGGGTTTTAAGGCGATTAAGTT
	2bRT4C7R12	CGTTGGTGTAGATGGGGTAATGGGATAGGT
	2bRT4C7R18	AAATTTTTAGAACCCTTTCAACGCAAGGAT
	2bRT4C7R20	TAAGCAATAAAGCCTCAAAGAATTAGCAAA
T6_TTT_X	2G3T1C7R00	ACGGTGGCATCAATTCTAGGGCGCGAGCTGAAAA
	2G3T1C7R02	CTTCCCAATTCTGCGAACCCATATAACAGTTGAT
	2G3T1C7R04	AAATTGCTCCTTTTGATATTAGAGAGTACCTTTA
	2G3T1C7R06	AACCATAAATCAAAAATCCAGAAAACGAGAATGA
	2G3T1C7R10	ATGAAACACCAGAACGAGAGGCTTGCCCTGACGA
	2G3T1C7R14	AGGAACGAGGGTAGCAACGCGAAAGACAGCATCG
	5bC2G3T1R18	GGGATTTTGCTAAACAAATGAATTTTCTGTATAGTAG
	5bC2G3T1R20	ACAAACTACAACGCCTGAGTTTCGTCACCAGTGTAAA
	Edg_T2R02C7_DHP	GTGTCGTAGACACGAGAGGGTTGATATAAGCGGATAAG TGCCGTCGTGTCGTAGACAC
	Edg_T2R06C7_DHP	GTGTCGTAGACACGCAGGTCAGACGATTGTTGACAGGA GGTTGAGGTGTCGTAGACAC
	Edg_T2R10C7_DHP	GTGTCGTAGACACGCGCCAAAGACAAAAGTTCATATGG TTTACCAGTGTCGTAGACAC
	Edg_T2R14C7_DHP	GTGTCGTAGACACTTTTTTGTTTAACGTCTCCAAATAA GAAACGAGTGTCGTAGACAC
	Edg_T2R18C7_DHP	GTGTCGTAGACACTAAACCAAGTACCGCATTCCAAGAA CGGGTATGTGTCGTAGACAC

	5bRT3C7R00	GTAAAGTAATTCTGTCAAAGTACCGAC
	5bRT3C7R02	AGTAGGGCTTAATTGAAAAGCCAACGC
	2G1T3C7R04	TCAATGGTTTGAAATACCCTTCTGACCTAAATTT
	2G1T3C7R08	GCTGAGCAAAAGAAGATGATTCATTTCAATTACC
	2G1T3C7R12	TAGTTATCTAAAATATCTAAAGGAATTGAGGAAG
	2G1T3C7R14	GAACATCGCCATTAAAAAAACTGATAGCCCTAAA
	2G1T3C7R18	ATTTGATTAGTAATAACATTGTAGCAATACTTCT
	2G1T3C7R20	AAAGGAACGGTACGCCAGTAAAGGGATTTTAGAC
	2G2T4C7R00	ATGAGCACGTATAACGTGCTATGGTTGCTTTGAC
	2G2T4C7R02	ATCGGGCGCTAGGGCGCTAAGAAAGCGAAAGGAG
	2G2T4C7R08	TAGCTCACTGCCCGCTTTACATTAATTGCGTTGC
	2G2T4C7R10	CAGTAACGCCAGGGTTTTAAGGCGATTAAGTTGG
	2G2T4C7R14	AGTTTAAATTGTAAACGTATTGTATAAGCAAATA
	2G2T4C7R16	CCGCCGGAGAGGGTAGCTTAGCTGATAAATTAAT
	5bRT4C7R18	AAATTTTTAGAACCCTTTCAACGCAAG
	5bRT4C7R20	TAAGCAATAAAGCCTCAAAGAATTAGC
T7_TTT_X	5bC2RT1R00	CATCAATTCTAGGGCGCGAGCTGAAAA
	5bC2RT1R02	ATTCTGCGAACCCATATAACAGTTGAT
	2bRT1C7R06	CCATAAATCAAAAATCCAGAAAACGAGAAT
	2bRT1C7R10	GAAACACCAGAACGAGAGGCTTGCCCTGAC
	2bRT1C7R14	GAACGAGGGTAGCAACGCGAAAGACAGCAT
	2bRT1C7R16	GGTTTATCAGCTTGCTAGCCTTTAATTGTA
	2bRT1C7R18	GGGATTTTGCTAAACAAATGAATTTTCTGT
	2bRT1C7R20	ACAAACTACAACGCCTGAGTTTCGTCACCA
	2bRT2C7R00	AGCCACCACCCTCATTGAACCGCCACCCTC
	2bRT2C7R02	GAGAGGGTTGATATAAGCGGATAAGTGCCG
	2bRT2C7R06	GCAGGTCAGACGATTGTTGACAGGAGGTTG
	2bRT2C7R08	TAGCGCGTTTTCATCGCTTTAGCGTCAGAC
	2bRT2C7R12	CCGAAGCCCTTTTTAAAGCAATAGCTATCT
	2bRT2C7R16	AACCTCCCGACTTGCGGCGAGGCGTTTTAG
	5bG4T2C7R18	AGGAGTAAACCAAGTACCGCATTCCAAGAACGGGTAT
	5bG4T2C7R20	TTGACAGATAAGTCCTGAACACCTGTTTATCAACAAT
	Edg_T3R02C7_DHP	GTGTCGTAGACACAGTAGGGCTTAATTGAAAAGCCAAC GCTCAACGTGTCGTAGACAC
	Edg_T3R06C7_DHP	GTGTCGTAGACACAGTCAATAGTGAATTTTTAAGACGC TGAGAAGGTGTCGTAGACAC
	Edg_T3R10C7_DHP	GTGTCGTAGACACCAATATAATCCTGATTGATGATGGC AATTCATGTGTCGTAGACAC
	Edg_T3R14C7_DHP	GTGTCGTAGACACACATCGCCATTAAAAAAACTGATAG CCCTAAAGTGTCGTAGACAC
	Edg_T3R18C7_DHP	GTGTCGTAGACACTTGATTAGTAATAACATTGTAGCAA

		TACTTCTGTGTCGTAGACAC
	Edg_T4R02C7_DHP	GTGTCGTAGACACCGGGCGCTAGGGCGCTAAGAAAGCG AAAGGAGGTGTCGTAGACAC
	Edg_T4R06C7_DHP	GTGTCGTAGACACATCCTGTTTGATGGTGGCCCCAGCA GGCGAAAGTGTCGTAGACAC
	Edg_T4R10C7_DHP	GTGTCGTAGACACGTAACGCCAGGGTTTTAAGGCGATT AAGTTGGGTGTCGTAGACAC
	Edg_T4R14C7_DHP	GTGTCGTAGACACTTTAAATTGTAAACGTATTGTATAA GCAAATAGTGTCGTAGACAC
	Edg_T4R18C7_DHP	GTGTCGTAGACACAAATTTTTAGAACCCTTTCAACGCA AGGATAAGTGTCGTAGACAC
T8_TTT_X	2G3T1C7R00	ACGGTGGCATCAATTCTAGGGCGCGAGCTGAAAA
	2G3T1C7R02	CTTCCCAATTCTGCGAACCCATATAACAGTTGAT
	2G3T1C7R08	AGCGAGGCATAGTAAGAGACGCCAAAAGGAATTA
	2G3T1C7R10	ATGAAACACCAGAACGAGAGGCTTGCCCTGACGA
	2G3T1C7R14	AGGAACGAGGGTAGCAACGCGAAAGACAGCATCG
	2G3T1C7R16	TTGGTTTATCAGCTTGCTAGCCTTTAATTGTATC
	5bRT1C7R18	GGGATTTTGCTAAACAAATGAATTTTC
	5bRT1C7R20	ACAAACTACAACGCCTGAGTTTCGTCA
	2G4T2C7R00	ATAGCCACCACCCTCATTGAACCGCCACCCTCAG
	2G4T2C7R02	AAGAGAGGGTTGATATAAGCGGATAAGTGCCGTC
	2G4T2C7R04	ATGTATAAACAGTTAATGTTGAGTAACAGTGCCC
	2G4T2C7R06	TAGCAGGTCAGACGATTGTTGACAGGAGGTTGAG
	2G4T2C7R10	GGGCGCCAAAGACAAAAGTTCATATGGTTTACCA
	2G4T2C7R14	AATTTTTTGTTTAACGTCTCCAAATAAGAAACGA
	5bC2G4T2R18	TAAACCAAGTACCGCATTCCAAGAACGGGTATCGGGC
	5bC2G4T2R20	AGATAAGTCCTGAACACCTGTTTATCAACAATGAGCA
	Edg_T3R02C7_DHP	GTGTCGTAGACACAGTAGGGCTTAATTGAAAAGCCAAC GCTCAACGTGTCGTAGACAC
	Edg_T3R06C7_DHP	GTGTCGTAGACACAGTCAATAGTGAATTTTTAAGACGC TGAGAAGGTGTCGTAGACAC
	Edg_T3R10C7_DHP	GTGTCGTAGACACCAATATAATCCTGATTGATGATGGC AATTCATGTGTCGTAGACAC
	Edg_T3R14C7_DHP	GTGTCGTAGACACACATCGCCATTAAAAAAACTGATAG CCCTAAAGTGTCGTAGACAC
	Edg_T3R18C7_DHP	GTGTCGTAGACACTTGATTAGTAATAACATTGTAGCAA TACTTCTGTGTCGTAGACAC
	5bRT4C7R00	GAGCACGTATAACGTGCTATGGTTGCT
	5bRT4C7R02	CGGGCGCTAGGGCGCTAAGAAAGCGAA
	2G2T4C7R04	CGATCACCCAAATCAAGTGCCCACTACGTGAACC
	2G2T4C7R08	TAGCTCACTGCCCGCTTTACATTAATTGCGTTGC
	2G2T4C7R12	TGCGTTGGTGTAGATGGGGTAATGGGATAGGTCA
	2G2T4C7R14	AGTTTAAATTGTAAACGTATTGTATAAGCAAATA

	2G2T4C7R18	TCAAATTTTTAGAACCCTTTCAACGCAAGGATAA
	2G2T4C7R20	AGTAAGCAATAAAGCCTCAAAGAATTAGCAAAAT
T9_TTT_X	2bRT1C7R00	GGTGGCATCAATTCTAGGGCGCGAGCTGAA
	2bRT1C7R02	TCCCAATTCTGCGAACCCATATAACAGTTG
	2bRT1C7R06	CCATAAATCAAAAATCCAGAAAACGAGAAT
	2bRT1C7R08	CGAGGCATAGTAAGAGACGCCAAAAGGAAT
	2bRT1C7R12	CTGATAAATTGTGTCGAGATTTGTATCATC
	2bRT1C7R16	GGTTTATCAGCTTGCTAGCCTTTAATTGTA
	5bG3T1C7R18	TCAACGGGATTTTGCTAAACAAATGAATTTTCTGTAT
	5bG3T1C7R20	AAAAGACAAACTACAACGCCTGAGTTTCGTCACCAGT
	Edg_T2R02C7_DHP	GTGTCGTAGACACGAGAGGGTTGATATAAGCGGATAAG TGCCGTCGTGTCGTAGACAC
	Edg_T2R06C7_DHP	GTGTCGTAGACACGCAGGTCAGACGATTGTTGACAGGA GGTTGAGGTGTCGTAGACAC
	Edg_T2R10C7_DHP	GTGTCGTAGACACGCGCCAAAGACAAAAGTTCATATGG TTTACCAGTGTCGTAGACAC
	Edg_T2R14C7_DHP	GTGTCGTAGACACTTTTTTGTTTAACGTCTCCAAATAA GAAACGAGTGTCGTAGACAC
	Edg_T2R18C7_DHP	GTGTCGTAGACACTAAACCAAGTACCGCATTCCAAGAA CGGGTATGTGTCGTAGACAC
	Edg_T3R02C7_DHP	GTGTCGTAGACACAGTAGGGCTTAATTGAAAAGCCAAC GCTCAACGTGTCGTAGACAC
	Edg_T3R06C7_DHP	GTGTCGTAGACACAGTCAATAGTGAATTTTTAAGACGC TGAGAAGGTGTCGTAGACAC
	Edg_T3R10C7_DHP	GTGTCGTAGACACCAATATAATCCTGATTGATGATGGC AATTCATGTGTCGTAGACAC
	Edg_T3R14C7_DHP	GTGTCGTAGACACACATCGCCATTAAAAAAACTGATAG CCCTAAAGTGTCGTAGACAC
	Edg_T3R18C7_DHP	GTGTCGTAGACACTTGATTAGTAATAACATTGTAGCAA TACTTCTGTGTCGTAGACAC
	5bC2RT4R00	CGTATAACGTGCTATGGTTGCTTTGAC
	5bC2RT4R02	GCTAGGGCGCTAAGAAAGCGAAAGGAG
	2bRT4C7R06	ATCCTGTTTGATGGTGGCCCCAGCAGGCGA
	2bRT4C7R10	GTAACGCCAGGGTTTTAAGGCGATTAAGTT
	2bRT4C7R14	TTTAAATTGTAAACGTATTGTATAAGCAAA
	2bRT4C7R16	GCCGGAGAGGGTAGCTTAGCTGATAAATTA
	2bRT4C7R18	AAATTTTTAGAACCCTTTCAACGCAAGGAT
	2bRT4C7R20	TAAGCAATAAAGCCTCAAAGAATTAGCAAA
T1_TTT_O	Edg_T1R02C7_DHP	GTGTCGTAGACACTCCCAATTCTGCGAACCCATATAAC AGTTGATGTGTCGTAGACAC
	Edg_T1R06C7_DHP	GTGTCGTAGACACCCATAAATCAAAAATCCAGAAAACG AGAATGAGTGTCGTAGACAC
	Edg_T1R10C7_DHP	GTGTCGTAGACACGAAACACCAGAACGAGAGGCTTGCC CTGACGAGTGTCGTAGACAC

	Edg_T1R14C7_DHP	GTGTCGTAGACACGAACGAGGGTAGCAACGCGAAAGAC AGCATCGGTGTCGTAGACAC
	Edg_T1R18C7_DHP	GTGTCGTAGACACGGGATTTTGCTAAACAAATGAATTT TCTGTATGTGTCGTAGACAC
	5bC2RT2R00	CCACCCTCATTGAACCGCCACCCTCAG
	5bC2RT2R02	GGTTGATATAAGCGGATAAGTGCCGTC
	2bRT2C7R06	GCAGGTCAGACGATTGTTGACAGGAGGTTG
	2bRT2C7R10	GCGCCAAAGACAAAAGTTCATATGGTTTAC
	2bRT2C7R14	TTTTTTGTTTAACGTCTCCAAATAAGAAAC
	2bRT2C7R16	AACCTCCCGACTTGCGGCGAGGCGTTTTAG
	2bRT2C7R18	TAAACCAAGTACCGCATTCCAAGAACGGGT
	2bRT2C7R20	AGATAAGTCCTGAACACCTGTTTATCAACA
	2bRT3C7R00	GTAAAGTAATTCTGTCAAAGTACCGACAAA
	2bRT3C7R02	AGTAGGGCTTAATTGAAAAGCCAACGCTCA
	2bRT3C7R06	AGTCAATAGTGAATTTTTAAGACGCTGAGA
	2bRT3C7R08	TGAGCAAAAGAAGATGATTCATTTCAATTA
	2bRT3C7R12	GTTATCTAAAATATCTAAAGGAATTGAGGA
	2bRT3C7R16	TCGTCTGAAATGGATTACATTTTGACGCTC
	5bG1T3C7R18	TTGATTTGATTAGTAATAACATTGTAGCAATACTTCT
	5bG1T3C7R20	GAAAAAGGAACGGTACGCCAGTAAAGGGATTTTAGAC
	Edg_T4R02C7_DHP	GTGTCGTAGACACCGGGCGCTAGGGCGCTAAGAAAGCG AAAGGAGGTGTCGTAGACAC
	Edg_T4R06C7_DHP	GTGTCGTAGACACATCCTGTTTGATGGTGGCCCCAGCA GGCGAAAGTGTCGTAGACAC
	Edg_T4R10C7_DHP	GTGTCGTAGACACGTAACGCCAGGGTTTTAAGGCGATT AAGTTGGGTGTCGTAGACAC
	Edg_T4R14C7_DHP	GTGTCGTAGACACTTTAAATTGTAAACGTATTGTATAA GCAAATAGTGTCGTAGACAC
	Edg_T4R18C7_DHP	GTGTCGTAGACACAAATTTTTAGAACCCTTTCAACGCA AGGATAAGTGTCGTAGACAC

Note: All invading X tiles used are shown (T2-9). Only T1 from the O set is shown for edge sequence completeness.

BIBLIOGRAPHY

Chapter I: Introduction

[1] Mendel, J. G. (1866). "Versuche über Pflanzenhybriden", Verhandlungen des naturforschenden Vereines in Brünn, Bd. IV für das Jahr, 1865, Abhandlungen: 3–47

[2] James D Watson and Francis HC Crick. “The structure of DNA”. In: Cold Spring Harbor symposia on quantitative biology. Vol. 18. Cold Spring Harbor Laboratory Press. 1953, pp. 123–131.

[3] Timothy J Richmond and Curt A Davey. “The structure of DNA in the nucleosome core”. In: Nature 423.6936 (2003), pp. 145–150.

[4] Chen JH, Seeman NC. Synthesis from DNA of a molecule with the connectivity of a cube. Nature. 1991;350:631–633.

[5] D. Yu. Zhang and G. Seelig. “Dynamic DNA nanotechnology using strand displacement reactions”. In: Nature Chemistry 3.2 (2011), pp. 103–113.

[6] Robert Carlson. “The changing economics of DNA synthesis”. In: Nature biotechnology 27.12 (2009), p. 1091.

[7] Rinker, S., Ke, Y., Liu, Y., Chhabra, R. & Yan, H. Self assembled DNA nanostructures for distance-dependent multivalent ligand-protein binding. Nature Nanotechnology 3, 418-532 422 (2008).

[8] Pal, S., Deng, Z., Ding, B., Yan, H. & Liu, Y. DNA-origami-directed self-assembly of discrete silver-nanoparticle architectures. Angewandte Chemie 122, 2760-2764 (2010).

[9] Pal, S. et al. DNA directed self-assembly of anisotropic plasmonic nanostructures. Journal of the American Chemical Society 133, 17606-17609 (2011).

[10] Knudsen, J. B. et al. Routing of individual polymers in designed patterns. Nature Nanotechnology 10, 892-898 (2015).

[11] Ryosuke Iinuma et al. “Polyhedra self-assembled from DNA tripods and characterized with 3D DNA-PAINT”. In: science 344.6179 (2014), pp. 65–69.

[12] Grigory Tikhomirov, Philip Petersen, and Lulu Qian. Programmable disorder in random DNA tilings. Nature Nanotechnology, 2016.

[13] Toshio Ando. “High-speed atomic force microscopy coming of age”. In: Nanotechnology 23.6 (2012), p. 062001.

[14] Paul J Hagerman. “Flexibility of DNA”. In: Annual review of biophysics and biophysical chemistry 17.1 (1988), pp. 265–286.

[15] John SantaLucia Jr and Donald Hicks. “The thermodynamics of DNA structural motifs”. In: Annu. Rev. Biophys. Biomol. Struct. 33 (2004), pp. 415–440.

[16] Leonard M Adleman. “Molecular computation of solutions to combinatorial problems”. In: Nature 369 (1994), p. 40.

[17] Winfree E, Liu F, Wenzler LA, Seeman NC. Design and self-assembly of two-dimensional DNA crystals. Nature. 1998;394:539–544.

[18] Rothemund, Paul W. K.; Papadakis, Nick; Winfree, Erik (December 2004). "Algorithmic self-assembly of DNA Sierpinski triangles". PLoS Biology. 2 (12): 2041–2053.
[19] Barish, Robert D.; Rothemund, Paul W. K.; Winfree, Erik (December 2005). "Two computational primitives for algorithmic self-assembly: copying and counting". Nano Letters. 5 (12): 2586–2592.
[20] Rothemund PW. Folding DNA to create nanoscale shapes and patterns. Nature. 2006;440:297–302.
[21] Liu W, Zhong H, Wang R, Seeman NC. Crystalline two-dimensional DNA-origami arrays. Angew. Chem. Int. Ed. 2011;50:264–267.
[22] Rajendran A, et al. Programmed two-dimensional self-assembly of multiple DNA origami jigsaw pieces. ACS Nano. 2011;5:665–671.
[23] E. S. Andersen et al., Self-assembly of a nanoscale DNA box with a controllable lid. Nature 459, 73 (2009). doi:10.1038/nature07971 pmid:19424153
[24] Douglas SM, Bachelet I, Church GM. A Logic-Gated Nanorobot for Targeted Transport of Molecular Payloads. Science. 2012;335:831–834.
[25] Maune, H. T. et al. Self-assembly of carbon nanotubes into two-dimensional geometries using DNA origami templates. Nature Nanotech. 5, 61–66 (2010).
[26] Hariadi, R. F., Cale, M. & Sivaramakrishnan, S. Myosin lever arm directs collective motion on cellular actin network. Proc. Natl Acad. Sci. USA 111, 4091–4096 (2014).
[27] Ke G, et al. Directional Regulation of Enzyme Pathways through the Control of Substrate Channeling on a DNA Origami Scaffold. Angew Chem Int Ed Engl. 2016.
[28] Gu, H., Chao, J., Xiao, S. J. & Seeman, N. C. A proximity-based programmable DNA nanoscale assembly line. Nature 465, 202–205 (2010).
[29] Thubagere, A. J. et al. A cargo-sorting DNA robot. Science 357, eaan6558 (2017)
[30] A. N. Marchi, I. Saaem, B. N. Vogen, S. Brown, T. H. LaBean, Toward larger DNA origami. Nano Lett. 14, 5740–5747 (2014). doi:10.1021/nl502626spmid:25179827
[31] Woo, S. & Rothemund, P. W. Programmable molecular recognition based on the geometry of DNA nanostructures. Nature Chemistry 3, 620-627 (2011).
[32] Rajendran, A. et al. Programmed two-dimensional self-assembly of multiple DNA origami jigsaw pieces. ACS Nano 5, 665–671 (2011).
[33] Zhao, Z., Liu, Y. & Yan, H. Organizing DNA origami tiles into larger structures using pre-formed scaffold frames. Nano Lett. http://dx.doi.org/10.1021/nl201603a (2011).
[34] Friedl, P., S. Borgmann, and E.B. Bröcker. 2001. Leukocyte crawling through extracellular matrix and the Dictyostelium paradigm of movement: lessons from a social ameba. J. Leukoc. Biol. 70:491–509.
[35] Shah, MM, Hammond, RS, Hoffman, DA (2010). Dendritic ion channel trafficking and plasticity. Trends Neurosci 33:307–316.
[36] Han, D., Pal, S., Liu, Y., Yan, H. Folding and cutting DNA into reconfigurable topological nanostructures. Nature Nanotech. 5, 712–717 (2010).
[37] Gerling, T., Wagebauer, K. F., Neuner, A. M. & Dietz, H. Dynamic DNA devices and assemblies formed by shape complementary, non-base pairing 3D components. Science 347, 1446–1452 (2015).
[38] Song J, Li Z, Wang P, Meyer T, Mao C and Ke Y 2017 Reconfiguration of DNA molecular arrays driven by information relay Science 357 eaan3377

[39] L. Qian, E. Winfree. Scaling up digital circuit computation with DNA strand displacement cascades. Science 332, 1196 (2011).
[40] D. Yu. Zhang and G. Seelig. “Dynamic DNA nanotechnology using strand displacement reactions”. In: Nature Chemistry 3.2 (2011), pp. 103–113.
[41] Tojima T, Itofusa R, Kamiguchi H. Steering neuronal growth cones by shifting the imbalance between exocytosis and endocytosis. J Neurosci. 2014 May 21;34(21):7165-78

Chapter II: A DNA origami tile

[1] Rothemund, P. W. K. Folding DNA to create nanoscale shapes and patterns. Nature 440, 297–302 (2006).
[2] Seeman, N. C. Nucleic acid junctions and lattices. Journal of Theoretical Biology 99, 237-247 (1982).
[3] Park SH, et al. Finite-size, fully addressable DNA tile lattices formed by hierarchical assembly procedures. Angew. Chem. Int. Ed. 2006;45:735–739.
[4] Winfree, E., Liu, F., Wenzler, L. A. & Seeman, N. C. Design and self-assembly of two-dimensional DNA crystals. Nature 394, 539-544 (1998).
[5] Rothemund, P. W. K., Papadakis, N. & Winfree, E. Algorithmic self-assembly of DNA Sierpinski triangles. PLoS Biology 2, e424 (2004).
[6] Barish, R. D., Schulman, R., Rothemund, P. W. K. & Winfree, E. An information-bearing seed for nucleating algorithmic self-assembly. Proceedings of the National Academy of Sciences 106, 6054-6059 (2009).
[7] Schulman, R., Yurke, B. & Winfree, E. Robust self-replication of combinatorial information via crystal growth and scission. Proceedings of the National Academy of Sciences 109, 6405-6410 (2012).
[8] Rinker, S., Ke, Y., Liu, Y., Chhabra, R. & Yan, H. Self assembled DNA nanostructures for distance-dependent multivalent ligand-protein binding. Nature Nanotechnology 3, 418-532 422 (2008).
[9] Pal, S., Deng, Z., Ding, B., Yan, H. & Liu, Y. DNA-origami-directed self-assembly of discrete silver-nanoparticle architectures. Angewandte Chemie 122, 2760-2764 (2010).
[10] Pal, S. et al. DNA directed self-assembly of anisotropic plasmonic nanostructures. Journal of the American Chemical Society 133, 17606-17609 (2011).
[11] Knudsen, J. B. et al. Routing of individual polymers in designed patterns. Nature Nanotechnology 10, 892-898 (2015).
[12] Woo, S. & Rothemund, P. W. K. Programmable molecular recognition based on the geometry of DNA nanostructures. Nature Chemistry 3, 620-627 (2011).
[13] Liu, W., Zhong, H., Wang, R. & Seeman, N. C. Crystalline two-dimensional DNA-origami arrays. Angewandte Chemie 123, 278-281 (2011).
[14] Truchet, S. Mémoire sur les combinaisons. Mémoires de l'Académie Royale des Sciences 1704, 363-372 (1704).
[15] Smith, C. S. & Boucher, P. The tiling patterns of Sebastien Truchet and the topology of structural hierarchy. Leonardo 20, 373-385 (1987).
[16] Grigory Tikhomirov, Philip Petersen, and Lulu Qian. Programmable disorder in random DNA tilings. Nature Nanotechnology, 2016.

Chapter III: Controlling growth and global properties

[1] Gordon, E. M., Barrett, R. W., Dower, W. J., Fodor, S. P. A. & Gallop, M. A. Applications of combinatorial technologies to drug discovery. 2. combinatorial organic synthesis, library screening strategies, and future directions. Journal of Medicinal Chemistry 37, 1385-1401 (1994).
[2] Ellington, A. D. & Szostak, J. W. In vitro selection of RNA molecules that bind specific ligands. Nature 346, 818-822 670 (1990).
[3] The Nobel Prize in Physiology or Medicine 1987". nobelprize.org. Retrieved 26 December 2014.
[4] Hay AJ, Gregory V, Douglas AR, Lin YP (December 2001). "The evolution of human influenza viruses". Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences. 356 (1416): 1861–70.
[5] Leonard M Adleman. “Molecular computation of solutions to combinatorial problems”. In: Nature 369 (1994), p. 40.
[6] Oliphant, AR; Brandl, CJ; Struhl, K (1989). "Defining the sequence specificity of DNA-binding proteins by selecting binding sites from random-sequence oligonucleotides: analysis of yeast GCN4 proteins". Mol. Cell. Biol. 9 (7): 2944–2949.
[7] Tuerk, C; Gold, L (1990). "Systematic evolution of ligands by exponential enrichment: RNA ligands to bacteriophage T4 DNA polymerase". Science. 249 (4968): 505–510.
[8] Dieckmann, T; Suzuki, E; Nakamura, GK; Feigon, J (1996). "Solution structure of an ATP-binding RNA aptamer reveals a novel fold". RNA. 2 (7): 628–40.
[9] Mercey, R; Lantier, I; Maurel, MC; Grosclaude, J; Lantier, F; Marc, D (2006). "Fast, reversible interaction of prion protein with RNA aptamers containing specific sequence patterns". Arch. Virol. 151 (11): 2197–214.
[10] Ulrich, H; Trujillo, CA; Nery, AA; Alves, JM; Majumder, P; Resende, RR; Martins, AH (2006). "DNA and RNA aptamers: from tools for basic research towards therapeutic applications". Comb. Chem. High Throughput Screen. 9 (8): 619–32.
[11] Daniels, Dion A.; Chen, Hang; Hicke, Brian J.; Swiderek, Kristine M.; Gold, Larry (2003-12-23). "A tenascin-C aptamer identified by tumor cell SELEX: Systematic evolution of ligands by exponential enrichment". Proceedings of the National Academy of Sciences. 100 (26): 15416–15421.
[12] Vavvas, D; D'Amico, DJ (2006). "Pegaptanib (Macugen): treating neovascular age-related macular degeneration and current role in clinical practice". Ophthalmol. Clin. North Am. 19 (3): 353–60.
[13] Protozanova, E., Yakovchuk, P. & Frank-Kamenetskii, M. D. Stacked–unstacked equilibrium at the nick site of DNA. Journal of Molecular Biology 342, 775–785 (2004).
[14] Liu, W., Zhong, H., Wang, R. & Seeman, N. C. Crystalline two-dimensional DNA-origami arrays. Angewandte Chemie 123, 278-281 (2011).
[15] Truchet, S. Memoire sur les combinaisons. Memoires de l'Academie Royale des Sciences 1704, 363-372 (1704).
[16] Philip Petersen. Yield Calculator. http://qianlab.caltech.edu/YieldCalculator, 2016.
[17] Browne, C. Truchet curves and surfaces. Computers & Graphics 32, 268-281 (2008).

Chapter IV: Fractal assembly of micron-scale DNA origami arrays

[1] Marchi, A. N., Saaem, I., Vogen, B. N., Brown, S. & LaBean, T. H. Toward larger DNA origami. Nano Letters 14, 5740– 5747 (2014).
[2] Rajendran, A., Endo, M., Katsuda, Y., Hidaka, K. & Sugiyama, H. Programmed two-dimensional self-assembly of multiple DNA origami jigsaw pieces. Acs Nano 5, 665–671 (2011).
[3] Erik D. Demaine, Martin L. Demaine, Sándor P. Fekete, Mashhood Ishaque, Eynat Rafalin, Robert T. Schweller, and Diane L. Souvaine, Staged self-assembly: Nanomanufacture of arbitrary shapes with O(1) glues, Natural Computing 7 (2008), no. 3, 347-370, Preliminary version appeared in DNA 13.
[4] Park, S. H. et al. Finite-size, fully addressable DNA tile lattices formed by hierarchical assembly procedures. Angewandte Chemie 118, 749–753 (2006).
[5] Xia, Y. & Whitesides, G. M. Soft lithography. Annual Review of Materials Science 28, 153–184 (1998).
[6] Mandelbrot, B. B. The fractal geometry of nature (W. H. Freeman and Company, New York, 1982).
[7] Woo, S. & Rothemund, P. W. Programmable molecular recognition based on the geometry of DNA nanostructures. Nature Chemistry 3, 620–627 (2011).
[8] Nangreave, J., Yan, H. & Liu, Y. Studies of thermal stability of multivalent DNA hybridization in a nanostructured system. Biophysical Journal 97, 563–571 (2009).
[9] Castro, C. E. et al. A primer to scaffolded DNA origami. Nature Methods 8, 221–229 (2011).
[10] Shawn M Douglas, Adam H Marblestone, Surat Teerapittayanon, Alejandro Vazquez, George M Church, and William M Shih. Rapid prototyping of 3D DNA-origami shapes with caDNAno. Nucleic Acids Research, 37(15):5001{5006, 2009.
[11] Rinker, S., Ke, Y., Liu, Y., Chhabra, R. & Yan, H. Self- assembled DNA nanostructures for distance-dependent multi- valent ligand-protein binding. Nature Nanotechnology 3, 418– 422 (2008).
[12] Maune, H. T. et al. Self-assembly of carbon nanotubes into two-dimensional geometries using DNA origami templates. Nature Nanotechnology 5, 61–66 (2010).
[13] Knudsen, J. B. et al. Routing of individual polymers in de- signed patterns. Nature Nanotechnology 10, 892–898 (2015).
[14] Schreiber, R. et al. Hierarchical assembly of metal nanoparticles, quantum dots and organic dyes using DNA origami scaffolds. Nature Nanotechnology 9, 74–78 (2014).
[15] Philip Petersen. Yield Calculator. http://qianlab.caltech.edu/YieldCalculator, 2016.
[16] Zenk, J., Tuntivate, C. & Schulman, R. Kinetics and thermo- dynamics of Watson–Crick base pairing driven DNA origami dimerization. Journal of the American Chemical Society 138, 3346–3354 (2016).
[17] Lin, C., Perrault, S. D., Kwak, M., Graf, F. & Shih, W. M. Purification of DNA-origami nanostructures by rate-zonal centrifugation. Nucleic Acids Research 41, e40–e40 (2012).
[18] Petersen, P. FracTile Compiler. http://qianlab.caltech. edu/FracTileCompiler (2017). Currently supported web browsers: Firefox and Chrome.

[19] Delbrück, T. & Mead, C. Analog VLSI phototransduction by continuous-time, adaptive, logarithmic photoreceptor circuits. Vision Chips: Implementing vision algorithms with analog VLSI circuits 139–161 (1995).

Chapter V: Reconfiguration of DNA Nanostructures

[1] Friedl, P., S. Borgmann, and E.B. Bröcker. 2001. Leukocyte crawling through extracellular matrix and the Dictyostelium paradigm of movement: lessons from a social ameba. J. Leukoc. Biol. 70:491–509.
[2] Boxer LA, Hedley-Whyte ET, Stossel TP. Neutrophil action dysfunction and abnormal neutrophil behavior. N Engl J Med. 1974 Nov 21;291(21):1093–1099.
[3] Berrington, J. E.; Barge, D.; Fenton, A. C.; Cant, A. J.; Spickett, G. P. (May 2005). "Lymphocyte subsets in term and significantly preterm UK infants in the first year of life analysed by single platform flow cytometry". Clinical Experimental Immunology. 140 (2): 289–92.
[4] Aderem A, Underhill DM (1999). "Mechanisms of phagocytosis in macrophages". Annu. Rev. Immunol. 17: 593–623.
[5] Berrington, J. E.; Barge, D.; Fenton, A. C.; Cant, A. J.; Spickett, G. P. (May 2005). "Lymphocyte subsets in term and significantly preterm UK infants in the first year of life analysed by single platform flow cytometry". Clinical Experimental Immunology. 140 (2): 289–92.
[6] Mellman I, Nelson WJ. Coordinated protein sorting, targeting and distribution in polarized cells. Nat. Rev. Mol. Cell Biol. 2008;9:833–845. doi: 10.1038/nrm2525.
[7] Han, D., Pal, S., Liu, Y., Yan, H. Folding and cutting DNA into reconfigurable topological nanostructures. Nature Nanotech. 5, 712–717 (2010).
[8] Gerling, T., Wagebauer, K. F., Neuner, A. M. & Dietz, H. Dynamic DNA devices and assemblies formed by shape complementary, non-base pairing 3D components. Science 347, 1446–1452 (2015).
[9] Song J, Li Z, Wang P, Meyer T, Mao C and Ke Y 2017 Reconfiguration of DNA molecular arrays driven by information relay Science 357 eaan3377
[10] E. S. Andersen et al., Self-assembly of a nanoscale DNA box with a controllable lid. Nature 459, 73 (2009). doi:10.1038/nature07971 pmid:19424153
[11] Douglas SM, Bachelet I, Church GM. A Logic-Gated Nanorobot for Targeted Transport of Molecular Payloads. Science. 2012;335:831–834.
[12] L. Qian, E. Winfree. Scaling up digital circuit computation with DNA strand displacement cascades. Science 332, 1196 (2011).
[13] Thubagere, A. J. et al. A cargo-sorting DNA robot. Science 357, eaan6558 (2017)
[14] D. Yu. Zhang and G. Seelig. “Dynamic DNA nanotechnology using stranddisplacement reactions”. In: Nature Chemistry 3.2 (2011), pp. 103–113.
[15] Tojima T, Itofusa R, Kamiguchi H. Steering neuronal growth cones by shifting the imbalance between exocytosis and endocytosis. J Neurosci. 2014 May 21;34(21):716
[16] Rinker, S., Ke, Y., Liu, Y., Chhabra, R. & Yan, H. Self assembled DNA nanostructures for distance-dependent multivalent ligand-protein binding. Nature Nanotechnology 3, 418-532 422 (2008).

[17] Pal, S., Deng, Z., Ding, B., Yan, H. & Liu, Y. DNA-origami-directed self-assembly of discrete silver-nanoparticle architectures. Angewandte Chemie 122, 2760-2764 (2010).
[18] Pal, S. et al. DNA directed self-assembly of anisotropic plasmonic nanostructures. Journal of the American Chemical Society 133, 17606-17609 (2011).
[19] Knudsen, J. B. et al. Routing of individual polymers in designed patterns. Nature Nanotechnology 10, 892-898 (2015).

Appendix

[1] Grigory Tikhomirov, Philip Petersen, and Lulu Qian. "Programmable disorder in random DNA tilings". *Nature Nanotechnology*, 10.1038/nnano.2016.256, 2016.

[2] Grigory Tikhomirov, Philip Petersen, and Lulu Qian. "Fractal assembly of micrometre-scale DNA origami arrays with arbitrary patterns". *Nature*, 10.1038/nature24655, 2017.