Molecular Computers Built from DNA Components

Sundipta Dharanipragada Rao

A dissertation

submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

University of Washington

2020

Reading Committee:

Georg Seelig, Chair

James Carothers

Sreeram Kannan

Program Authorized to Offer Degree:
Electrical and Computer Engineering

University of Washington

Abstract

Molecular Computers Built from DNA Components

Sundipta Dharanipragada Rao

Chair of the Supervisory Committee:
Georg Seelig
Department of Electrical and Computer Engineering

At the nanoscale, the ability to control spatio-temporal interactions would give us unparalleled power. DNA strand displacement systems seem to be the perfect technology to achieve control over molecular interactions, as they have the major advantage that reactions can be predicted from domain structure alone. Strand displacement technology has resulted in a myriad of dynamic devices with applications for everything from diagnostics [1-3] to biomimetic manufacturing [4, 5]. Our goal is to build DNA strand displacement computers to achieve control over the temporal dynamics of molecules, meaning how they interact in time, and also the spatial dynamics, or how they interact in space. To build these types of molecular computers, we take inspiration from the programmability of silicon computers, which take programming languages as input. As a primitive for a molecular programming language, we look to previous work which

has shown that the behavior of formal chemical reaction networks (CRNs) can be approximated using nicked double stranded DNA (ndsDNA) gates [6]. CRNs are a natural way to think about molecular interactions and have been shown to be Turing complete [7], if used as a programming language. To compose complex or large DNA strand displacement CRNs, it is desirable to compose them from small, well-characterized systems, a strategy that requires quantitatively predictable reaction kinetics. However, parameter estimation of these ndsDNA gates has thus far required fitting reaction rates for every strand displacement occurrence individually and these fitted reaction rates were found to vary over more than an order of magnitude despite toehold sequences designed to have the same length and GC content. With our work on context-independent plasmid-derived gates, high quality fits can be obtained using a single reaction rate constant k for all strand displacement steps, allowing for predictable and composable kinetic behavior. Additionally, the nicked double stranded structure of the gates allows for them to be derived from plasmids as a source for highly pure DNA for use in strand displacement experiments. In our work, we use a cloning strategy that dramatically reduces recombination events, increasing the yield of functional gates that can be used for experiments, and can also be used to control stoichiometry of chemical reactions. Here, we also show that our platform using context-independent plasmid-derived gates can also be used in a spatial setting. We built a travelling wave system by putting a synthetic autocatalytic reaction in a spatial setting, demonstrating the first steps towards an autonomous and synthetic pattern formation system on a centimeter scale based on a DNA strand displacement system.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I have so many people to thank. First and foremost, I'd like to thank my family, who have been my support structure in so many ways. I'm acutely aware that the ability to pursue a graduate degree is a great privilege, one that would not have been afforded to me without my parents. Thank you to both of them, Bhaskar and Krishnaveni Rao, without whom I never would have had the strength or resources to pursue a graduate degree. My mother has always been my unconditional cheerleader and my father has been my academic role model throughout my life. Thanks also to my sister, Nandana Rao. I'm so lucky to have a little sister that's also my best friend, confidant, and roommate.

I'd also like to thank my grandparents, Krishnaveni Tamirisa, Manikyamba Dharanipragada, Apparao Tamirisa, and Rangarao Dharanipragada. Thank you for all your blessings, support, and love throughout.

Graduate school has been such an incredible experience and I owe so much of that to my wonderful advisor, Georg Seelig. Georg cultivates a lab that is incredibly collaborative and supportive of people's intellectual passions. I've loved working in his lab and learning from all the brilliant scientists that have blossomed in this environment, under his mentorship. I'm grateful to leave this lab with Georg not only as an advisor but also as a friend.

Thank you to everyone in and adjacent to the Seelig Lab, including Sherry Chen, Gourab Chatterjee, Ben Groves, Sumit Mukherjee, Paul Sample, Arjun Khakar, Randolph Lopez, Sifang Chen, Charlie Roco, Alex Rosenberg, Sergii Pochekailov, Ban Wang, Yue Zhang, Max Darnell, Alex Baryshev, Johannes Linder, Nick Bogard, Anna Kuchina, Alberto Carignano, Kevin Oishi, Chris Takahashi and David Younger. I know I'll look back on these grad school days as some of the best times of my life, and that's largely due to the amazing people I met here. In particular, I want to thank Yuan Jyue-Chen for being an

incredible mentor and collaborator throughout grad school, without whom I certainly would not be receiving a PhD.

I'd also like to thank Neil Dalchau and Andrew Phillips at Microsoft Research, who have been invaluable collaborators, without whom the VisualDSD models would have been impossible. Thanks also to the researchers at center for Game Science for their collaboration on the Nanocrafter game project.

A special shout out to Leandra Brettner, Tileli Amimeur, and Stephanie Berger, whom I've been blessed to consider my friends, colleagues, and role models through my twenties. Academia might have been tougher to navigate without the support of such amazing and driven women.

Certainly, my tenure in graduate school has seen the world struggle with some tumultuous times. There have been no better friends than Sifang Chen, Randolph Lopez, and Chuhern Hwang to help stave off the existential crises that inevitably comes from watching the world continue to reject the advice of experts in favor of ever-more divisive narratives, even as we each proceeded to become experts in our respective fields. Thank you for keeping my faith in humanity alive and pushing me to continue to strive to make the world we live in a better place for us all.

Lastly, I'd like to thank my wonderful life partner, Nicholas Bolten. Possibly the best part of grad school was that it brought us together. Thanks for making me a better scientist and human and making every day of my life full of laughter and love. (ILN)

DEDICATION

This thesis is dedicated to my parents, Bhaskar and Krishnaveni Rao. As I close the door on this chapter in my life, I look to the future with hopeful eyes. I know the skills I've cultivated here will serve me well, as I make my way outside of academia, trying to make the world a better place for us all.

Chapter 1. INTRODUCTION

The fundamental principles of computer science that sparked the computer revolution have allowed us to master colossal electronic systems and software to do amazingly complex tasks. My research investigates how to apply these principles at a biomolecular level. From the operating system controlling cellular metabolism to the code for biological development, biomolecular programming is a natural sub-discipline of computer science and learning how to program biological systems will be a valuable tool in the next century. DNA has emerged as a viable biological engineering material due to the programmability and predictability of Watson-Crick base pairing [8]. In fact, the ability to engineer DNA interactions has the remarkable consequence of allowing us such fine-grain control over the interactions of matter that we can now control interactions happening on the nanoscale.

1.1 DNA Strand Displacement

Most commonly, DNA is thought of as the quaternary code that encodes the "blueprint of life." In the central dogma of molecular biology, DNA is transcribed into RNA, which are then translated into proteins, the macromolecules which carry out many of the most important biological processes. However, at a fundamental level, DNA is simply a molecule. In fact, the predictability and specificity of DNA interactions on a sequence level makes DNA a powerful and versatile molecule for engineering at the nanoscale [9].

A powerful motif in DNA nanotechnology is DNA strand displacement (DSD), an enzyme-free, competitive hybridization reaction, is a powerful primitive for engineering dynamic molecular systems [10]. DNA strand displacement is the process by which two DNA strands with partial or full complementarity hybridize to each other, displacing one of more pre-

hybridized strands in the process. This simple mechanism has been used to engineer a variety of structural and dynamic DNA devices and has opened up a huge space for potential designs.

DNA typically consists of a directional sequence of nucleotides (A, C, T, G), which can either be single-stranded or double-stranded. For representational purposes, strands of DNA as directional lines, with the arrowhead representing the 3' end. We term sequential bases which act as a functional unit as "domains" and typically denote them with an alphanumeric label, allowing us to forgo the sequence level notation of DNA. The complementary strand is often denoted with a * or ' (i.e. domain 3* would be the reverse complement of domain 3).

Toehold-mediated strand displacement is initiated at a "toehold", an overhanging single-stranded domain of DNA, where a complementary single stranded piece of DNA can bind. Toeholds are typically short (6-8 bases long) and binding to the toehold is a fast, reversible process. Hybridization at the toehold allows for the possibility of an incumbent strand to displace a previously hybridized strand via a random walk process called branch migration. This can occur because typically the nucleotides bound at the end of a double-stranded region of DNA can "breathe", meaning they can bind and unbind, allowing for the possibility of an incumbent strand to invade and bind instead. Eventually, a complementary incumbent strand can completely displace a previously bound strand.

The process of DNA strand displacement is thermodynamically driven. It is energetically favorable to have as many bases pairs bound as possible. Additionally, there is no entropic cost to the strand displacement reaction, meaning that the overall number of DNA complexes does not change after the reaction proceeds.

Strand displacement reaction kinetics are largely controlled the by length and sequence of the toehold, where the process of DNA strand displacement initiates. Variation in the strength of

the toehold (by varying the length or sequence composition) can impact the reaction rates over a factor of 10^6 [11, 12]. Indeed, DNA strand displacement rates are so sequence dependent that even single base differences can be used to differentiate between different strands [13].

Experimentally, the completion of a DNA strand displacement reaction can be visualized using a “reporter” complex, which is typically made of one strand labeled with a fluorophore and another with a quencher, which suppresses fluorescence when it is close to a fluorophore. When a strand displacement reaction successfully displaces the fluorophore-bound strand from the quencher-bound strand, it results in an observable fluorescent signal [14].

Strand displacement has been used as a primitive to engineer a variety of structural and dynamic DNA devices. DNA strand displacement networks have been used to produce repeating figures [15], self-assembling structures [16-19], logic circuits [20, 21], and walking robots [22, 23], with potential applications such as tissue engineering or repair, diagnostics, therapeutic delivery [2, 24], or even information storage [25, 26]. These DNA circuits can be made of modules that are highly composable. In fact, one of the largest DNA circuits, the square-root circuit implemented with the seesaw DNA motif, uses 74 initial DNA species [27, 28]. Additionally, DNA strand displacement circuits are bio-compatible, allowing for more easy interfacing with biological materials than traditional silicon computers [29]. In fact, several papers have shown that strand displacement systems can be engineered to be functional in cells [30, 31].

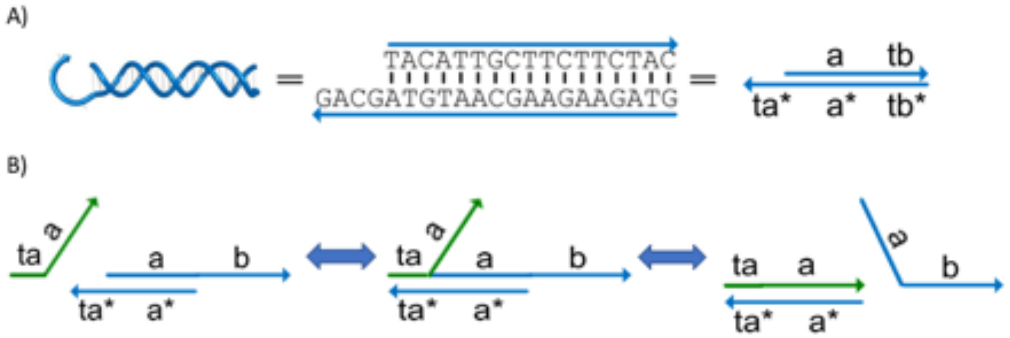

Figure 1.1. DNA Strand displacement.
A) Double- stranded DNA with a single-stranded overhang can be represented as lines drawn with the arrowhead on the 3' end. Groups of functional bases are labeled by letters or numbers with domains denoted with a * being complementary to the non-starred domain (i.e domain 3* is complementary to domain3). The single-stranded overhang is called a toehold. B) Single-stranded DNA with a region complementary to an open toehold can bind and through a random walk branch migration process displace the previously bound strand.

1.2 Chemical Reaction Networks

Chemical reaction networks (CRNs) have long been used as a powerful mathematical tool for analyzing molecular behavior. Recent advances in chemical reaction network theory have proved the strong Turing completeness of these systems [7], demonstrating that CRNs have incredible computational potential aside from their descriptive usage. However, CRNS have not traditionally served as a prescriptive engineering framework for molecular systems because the components required for implementing a desired set of chemical reactions are prohibitively difficult to find and compose in nature.

1.2.1 *Describing CRNS Mathematically*

Chemical reactions are a way to denote the stoichiometric reactions that occur between molecules where one or more molecules react to transform into different molecules. These reactions may be represented by a chemical reaction such as $aA + bB \rightarrow cC + dD$, where a and b denote the number of moles of reactants A and B that react to give c and d moles of the products

C and D. Chemical reactions may be reversible or irreversible and these reactions occur at some reaction rate, k, which is often experimentally determined.

According to the law of mass action, the rate of a chemical reaction is proportional to the concentrations of the reactants. That is to say that, given the elementary reaction

$$A + B \xrightarrow{k} C + D$$

the law of mass actions gives us that

$$\frac{\mathrm{d[A]}}{\mathrm{dt}} = \frac{\mathrm{d[B]}}{\mathrm{dt}} = -k[A][B] \tag{1.1}$$

$$\frac{\mathrm{d[C]}}{\mathrm{dt}} = \frac{\mathrm{d[D]}}{\mathrm{dt}} = k[A][B] \tag{1.2}$$

We can use these ordinary differential equations to model the behavior of chemical reaction networks as seen in Figure 1.2.

In theory, chemical reaction networks have been shown to be strong Turing complete [7]. This means that chemical reactions can be used as a prescriptive programming language for molecules, as long as we have the ability to control the composition and interaction of chemicals in our system.

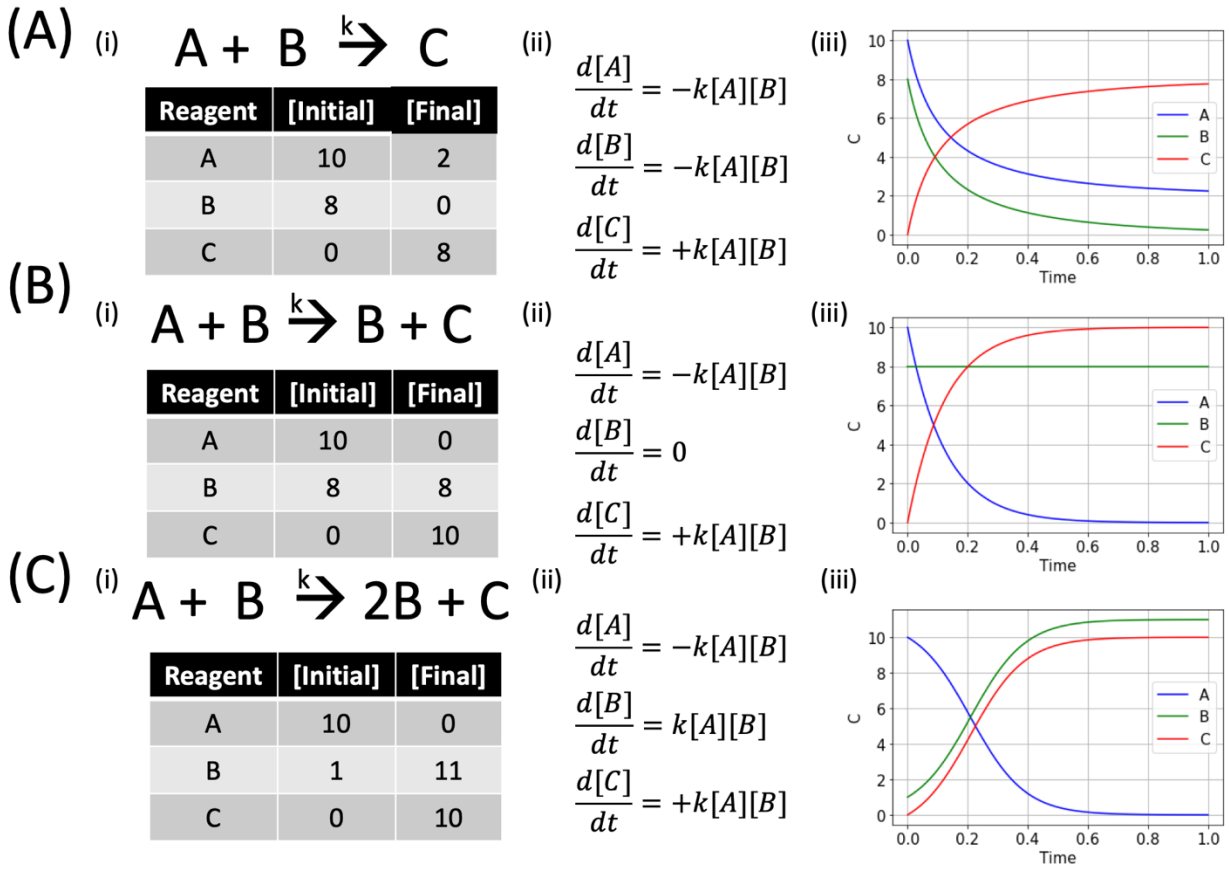

Figure 1.2. Law of Mass Action Describes Behavior of Various Reaction Types.
Panel A) shows a simple bimolecular reaction, panel B) shows a catalytic reaction and panel C) depicts an autocatalytic reaction. Panels i) show the chemical reaction being modeled and the initial and final amounts of each chemical in the reaction. The noncatalytic reaction shows how the behavior of the reaction is stoichiometric; the amount of product created is dependent on the amount of the limiting reagent initially present. In the catalytic reaction, the catalyst is regenerated, so the amount of catalyst does not change while the reagent A is completely converted to the reagent C. In the autocatalytic reaction, the catalyst B is generated as the reaction proceeds to convert all of the initial reactant A to the product C. Panels ii) show the differential equations that describe these reactions. Panels iii) show the dynamic kinetics of the reactions proceeding occurring to the reactions given in panel ii) and the initial conditions given in panel i). All reactions are simulated with an arbitrary reaction rate of k =0.1/Concentration Unit/Time. The dynamics of the reactions show that the catalytic reaction happens much faster than the noncatalytic reaction with the same initial conditions. The autocatalytic reaction proceeds even faster (notice that the initial amount of catalyst is much smaller than in the other two reactions) and displays a typical sigmoidal shape.

1.2.2 *Arbitrary CRNS with DNA Components*

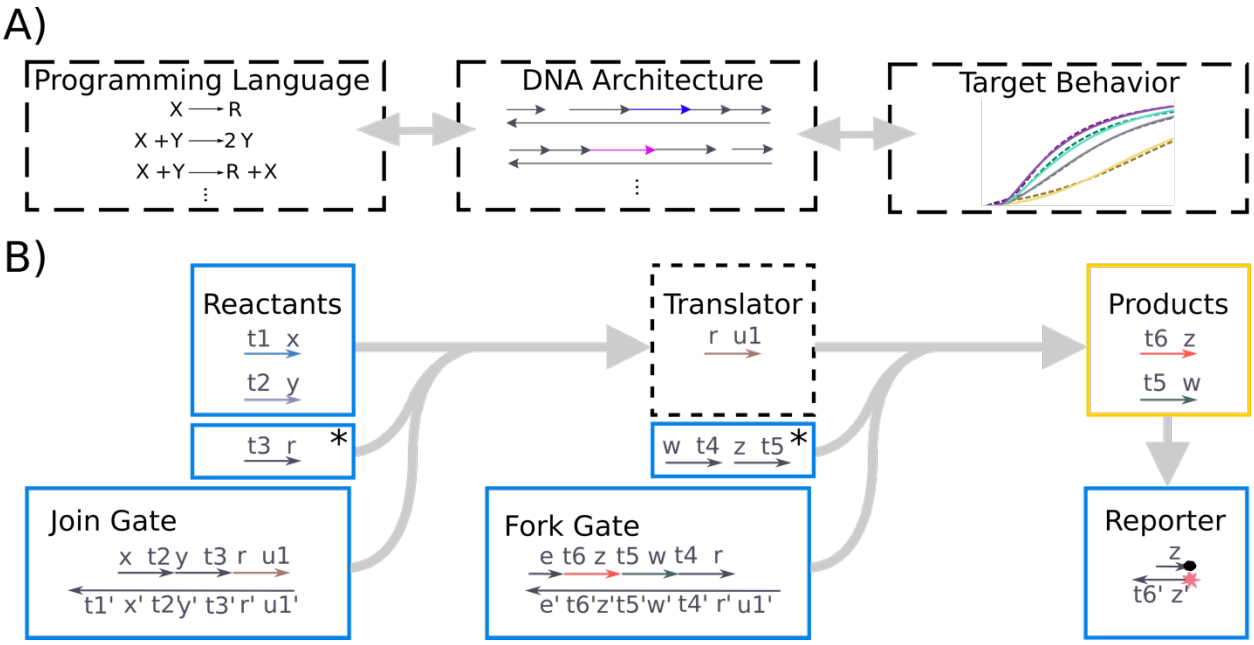

Figure 1.3 Arbitrary CRNs can be built from DNA components.

A) The chemical reaction network can be specified in the typical representation. DNA gates that will emulate those reactions can be built and the target behavior can be achieved. These processes are interrelated i.e we can start with a specified target behavior, write down the accompanying CRN and then build it with DNA. B) The emulation of the reaction X + Y → Z + W requires three gates termed join, fork and reporter gates. The join gate binds the reactants through a series of strand displacement reactions. A helper strand initiates a reaction which kicks the translator strand off. The translator strand then binds to the fork gate, initiating a strand displacement cascade that releases the products. Products can then bind to the reporter, releasing a fluorescent output.

In previous work [6, 32], we saw the first experimental evidence that DNA strand displacement cascades could indeed be used to approximate the behavior specified by formal CRNs (see Figure 1.3). Although there are many different ways to implement chemical reactions with DNA, in our work, we use the two-domain architecture originally proposed by Cardelli [33]. One reason for this is that the "gates" required for CRN computation are highly modular, making it

easy to compose larger systems. Additionally, the gates consist primarily of nicked double stranded DNA (ndsDNA), allowing them to be enzymatically derived from highly pure plasmid DNA. Sourcing DNA gates from plasmids significantly improves sequence accuracy and allows for easy storage of the gates as purified plasmid or bacterial glycerol stocks. We can even encode multiple gates onto each plasmid, in order to efficiently use the plasmid "real estate" and increase the yield of functional gates.

In this document, I describe our work solving some of the fundamental problems with previous implementations of chemical reaction networks with DNA strand displacement systems. Mainly, we use *context-independent plasmid derived gates* that use "universal" toeholds to achieve context-independent reaction rates and a novel plasmid design to reduce the rate of recombination and increase yield. We introduce an approach to designing "universal" toeholds using neighboring insulating regions such that when the same design is used in different gate constructs, the reaction rate of each strand displacement step is consistent. Additionally, we solve the problem that occurs when encoding multiple gates, all containing at last some identical sequence elements into a single plasmid, onto a single plasmid which is the increase in frequency of recombination events, particularly loop-out deletions. We introduce a new plasmid design strategy to reduce these recombination events, called inter-marker located ndsDNA (ILN) gate plasmids. These plasmids separate each gate with a genetic part necessary for plasmid viability. We show that this plasmid design dramatically reduces recombination events and thus increases the yield of functional gates that can be used for experiments. With these enhancements, *context-independent plasmid-derived gates* enable predictable and composable kinetic behavior and represent an important step toward composing larger and more complex CRN systems.

1.3 Spatial Pattern Formation

One of the hallmarks of biological systems is the ability to spatially self-organize. The ability to replicate this ability synthetically would give us unparalleled power in the field of synthetic biology and biomaterials. At a fundamental level, pattern formation systems are simply chemical computations. This means, in theory, we should be able to use our chemical programming language to achieve spatial organization.

There are several chemical systems that are capable of pattern formation. Famously, the B-Z oscillator creates spatial wave patterns [34]. Additionally, there are engineered systems that provide examples of self-organizing systems which use RNA or protein based components [35]. However, these systems are less programmable than engineered DNA systems, which, as we have seen, allow for the tuning of reaction rates and building of arbitrary chemical reactions. Thus, we would like to be able to generate spatial patterns using DNA components.

Structural DNA Nanotechnology has already made significant progress in patterning on the nanoscale. Molecular components, typically made from DNA origami [36], are programmed to be able to fit together such that larger structures can be built from the smaller components. This methodology has the building of many intricate structures. However, the structures that can be built are limited by the size of the DNA and the price of the synthesis of DNA, making this technique untenable for attaining patterns on a macroscale.

A feasible method for generating spatial patterns on a centimeter scale is to use reaction-diffusion systems built with DNA components. In these systems, diffusible signals interact with one another to generate complex stationary patterns. The addition of diffusion to the system allows for generation of patterns on a much larger scale.

The approach we use to achieve macroscopic synthetic pattern formation is to use rationally designed reaction-diffusion systems comprised of DNA components. We use our context-independent gates as a platform for engineering these types of synthetic pattern formation systems. In this work, we have taken the first steps toward implementing our ndsDNA CRN strategy into a spatial setting to demonstrate the feasibility of this method. We show that programmable CRNs made with DNA components can be used in reaction-diffusion setting to generate traveling waves.

Chapter 2. DEVELOPING A FRAMEWORK FOR SCIENTIFIC DISCOVERY USING GAME PLAY

2.1 INTRODUCTION

Scientists need to be able to explain their work to the public. We need to be able to share why the research we do is relevant to peoples' lives, not only to be able to receive funding for the important work that we do, but also so the general public can understand how scientific theories are generated and feel comfortable using logical reasoning and scientific discoveries as the basis for rational, data-driven decisions in their daily lives.

As a graduate student I thought about this dilemma quite a bit. One way to start to break down the walls of the ivory tower is to make your work more accessible. In this vain, I worked on a scientific discovery game, called Nanocrafter. The idea behind this game is to abstract away the sequence level design aspect of DNA strand displacement systems and to translate them to a visual programming language, which is usable by game players without any expertise in the field of DNA Nanotechnology. The goal of this project is two-fold; first, to create a game which can educate novices to the field about DNA strand displacement, and second, to create a platform where gamers can contribute novel ideas for device designs to the scientific community.

Indeed, game play has been shown to be a viable method for scientific discovery. Foldit, a protein folding game, provides players with protein structure problems and uses a well-known protein folding program, Rosetta, to help users minimize an optimization function. Players of this game have solved the structure of the M-PMV retroviral protease and have also recreated a typical protein folding algorithm through gameplay [37]. EteRNA, an RNA folding game, allows players to design RNA sequences that fold up into a target shape [38]. Players can come up with solutions to real RNA design problems and these solutions can then be tested experimentally in

the lab, with players being able to access these experimental results. Other biological game applications include genetic sequence alignment [39], and mapping neurons [40].Games have also been built around problems in computer science and physics, including graph theory [41], software verification [42], and quantum computing [43].

Additionally, many existing entertainment games allow players to build solutions to open-ended solutions to puzzles, or even give players a "sandbox" to build whatever they can come up with. These games often allow players to share their solutions, or even new puzzles, with other members of the game's community. Perhaps the most successful of these has been Minecraft [44], which, without specific puzzles or solutions, allows players to build structures, devices, and computation in a voxel world. SpaceChem [45] and Infinifactory [46] allow players to build assembly-line like mechanisms and compare the efficiency of their solutions to those produced by other players. Games like Lightbot [47] and RoboZZle [48] give players basic interfaces to programming constructs in order to come up with logical, automated solutions to puzzles. Scribblenauts lets players summon from a vast array of objects and characters, combining them in new ways, to solve in-game puzzles [49]. However, in these games the creations produced by players are generally self-contained to the game, as the games are not designed to address specific real-world problems.

Along with a team of programmers and artists at the Center for Game Science at the University of Washington, we created Nanocrafter, an online DNA strand displacement video game [50]. The idea behind Nanocrafter is to abstract DNA strand displacement systems into a visual programming language, similar to Scratch, so that players of all ages can program DSD reactions using the flash user interface [51]. Unlike other scientific discovery games, the goal of the game is not to solve a specific structure problem, but instead to educate players on the basic

mechanisms of DSD and then harness the creativity of game-players to build novel devices on the nanoscale. The game presents real-world nanotechnology challenges to players and allows them to create designs that could eventually be built and tested in the wet lab.

2.2 A Visual Programming Language for DSD

The game's core mechanic is a simulation of DNA strand displacement [8, 30, 52, 53]. The basic DNA element visualized in the game is the domain, a short sequence of nucleic acids. For each domain, there are four main properties to visualize: the sequence (which string of bases it represents), the directionality (which end is the "head" and which is the "tail", corresponding to the 5' end and 3' end of DNA molecules), the complementarity (whether the strand represents a base sequence or its antiparallel counterpart), and the length (long or short). These properties define which domains can bond with each other. In the game, each individual block piece represents a domain.

Many potential domain representations were iterated over during development. For clarity, particularly at multiple levels of zoom, we chose a simple, angled chevron shape for the main body of domains. This domain representation abstracts away many of the low-level details of the DNA complexes that are represented, such as the exact sequence of bases. It also does not look like the familiar helical DNA representation and this may help to dispel players of any preconceived notions of the game being related to genetics.

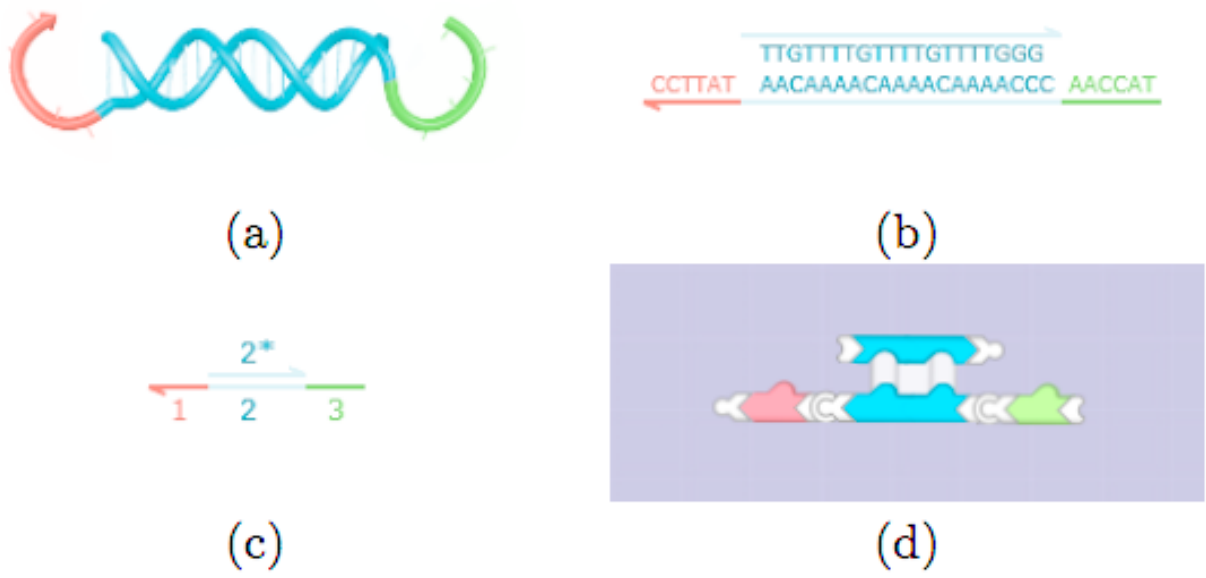

Figure 2.1. More Representations of DNA.
a) The traditional double-helix representation of DNA, b) a sequence level description of the DNA strands, c) a domain representation that abstracts sequences into functional groups (domains) that are given alphanumeric values, with complementary domains denoted with a * d) Nanocrafter's representation of DNA domains, showing the same level of abstraction as c, but aesthetically looking like a puzzle piece.

The sequence uniqueness of a domain is denoted by its color. This is similar to the alphanumeric annotation used by DNA Nanotechnologists when describing their DNA structures. While domains are easier to visualize in a solid color, even more sequences can be used by allowing domains to be multiple colors. Directionality and complementarity are represented by the domain's shape; the chevrons on a domain point towards its head, and a domain with bonding points that point outward is complementary to a domain with bonding points that point inward. These bonding points are simple curves as more complex puzzle-piece like bonding points were difficult to use at low zoom levels. Bonding points are angled slightly to reinforce directionality. In addition to their length differences, long domains have two bonding points and short domains have one, and their colors are drawn from disjoint sets.

Domains can be connected together, head to tail, to create longer strands. Domains that are the same color and length can also be hybridized to complimentary domains, referred to in the game as bonded. Once bonded, they will point in opposite directions, as would the DNA strands comprising a double helix. These representations can be seen in Figure 2.1. Connection points (representing either covalent bonds when domains are chained together, or hydrogen bonds when they are bonded) are shown as area external to the color-filled regions of domains, which allows them to easily be seen if two domains are connected or bonded.

In the game, a group of domains that is connected or bonded together is referred to as a plex (short for complex) and the collection of all the plexes is referred to as an invention. Players assemble inventions by snapping domains together and then run the simulator to observe the results of their inventions in action. As is standard in the field, Nanocrafter represents two different lengths of domains, long and short, that differ in their bonding stability. Long domains can, by themselves, maintain stable hydrogen bonds with their complement. Short domains (representing toeholds) will stay bonded only if they are part of a longer bonded strand, otherwise they will fall apart.

If there is more than one copy of a plex, all the copies are collapsed into a single instance of the plex with a noted concentration. We represent concentrations for each plex as stacks. Plexes with a concentration c greater than 1 will have $\lceil \log_{10}(c) \rceil + 1$ copies of the plex stacked on top of each other. An example of this can be seen can be seen in Figure 2.6.

Abstracting DNA Nanotechnology constructions in this way allows players to build complexes using the visual abstraction, rather than worrying about the sequence level design of their devices. This abstraction is a valid and useful one in terms of modeling the dynamics of DNA devices, as has been demonstrated by the Visual DSD project from Microsoft Research,

which consists of a semantic language [54] for describing DNA strand displacement reactions and simulation software that closely mimics the kinetics of real strand displacement reactions. Visual DSD has been used to successfully evaluate experimental models and accurately predict real-world results, despite also abstracting away the sequence level design of these devices [55].

2.3 Educating Players about DNA Strand Displacement

Nanocrafter featured a sequence of introductory levels that are meant to teach players about game mechanics and build their knowledge of DNA nanotechnology. The introductory levels have specific goals that must be met to complete the level and move on to the next one.

In the introductory levels, particular goal domains are marked with stars. These domains must end up unbonded when the simulation competes (this is accomplished by displacing them). If the simulation completes and any goal domains are still bonded, the level is lost and the player can try again, editing their invention and re-simulating. On levels that can oscillate indefinitely (meaning the simulation will never complete) the level is won when all goal domains are released, and the simulation oscillates between a limited number of states for a fixed number of reactions. Goal domains are inspired by “molecular beacons”, fluorophore tagged DNA segments hybridized with segments that have been tagged with a quenching dye, resulting in easily-measured fluorescence when an experiment completes successfully, and limited or absent fluorescence when the fluorophore remains bound [14].

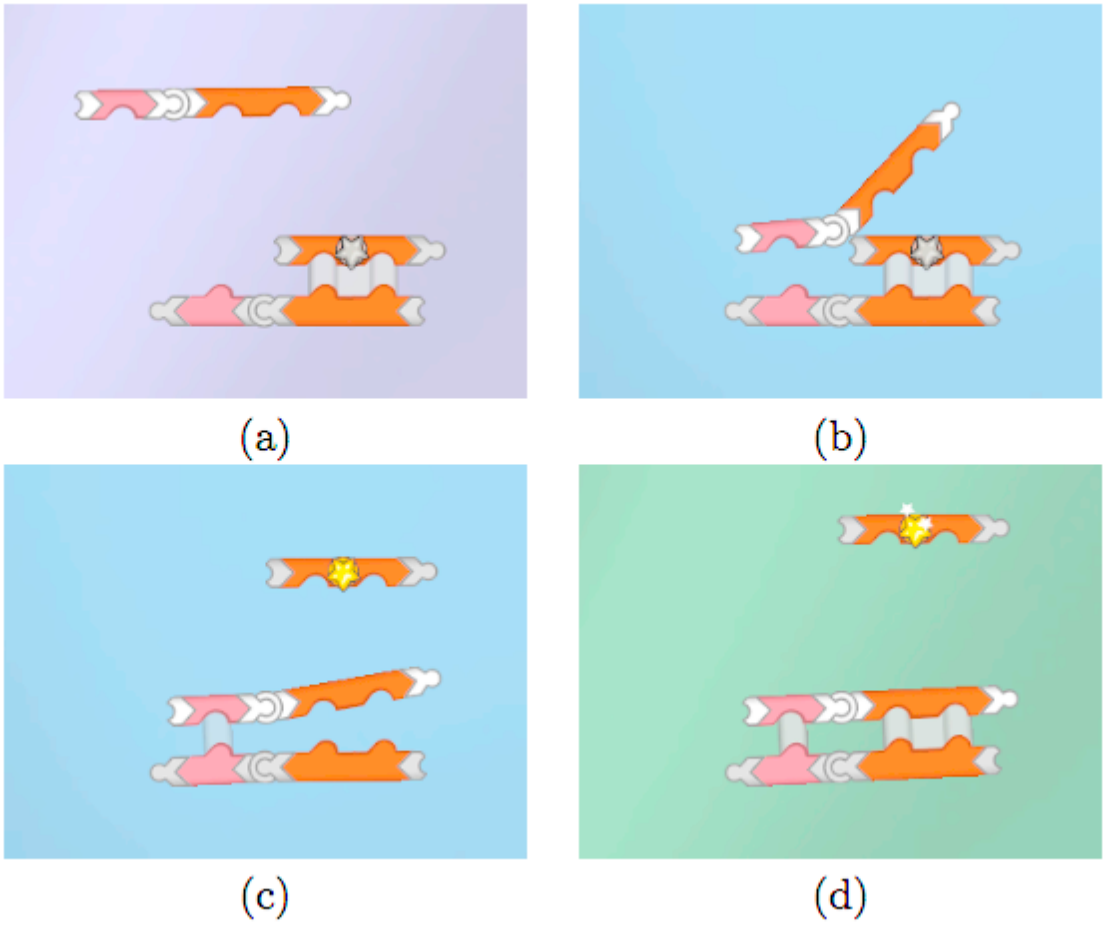

Figure 2.2. Nanocrafter DNA Strand Displacement Reaction.
a) A player has arranged the active (white-ended) strand to complement the longer, inactive (grey-ended) strand. b) The player starts the simulation, and the un-hybridized, complementary pink domains react and form a bond. (c) Strand displacement occurs, with the orange target domain (marked with a fluorophore, represented in-game as a star) unhybridizing as the longer strand displaces it. (d) The simulator reaches a metastable state, and triggers the win animation since the target domain has been released.

So that goal domains cannot be unbonded directly, some domains in the initial setup of the levels are grayed out and cannot be edited by the player, but rather must be manipulated by building an invention. To add challenge, some levels also contain cracked domains, which must remain bonded throughout the simulation of the invention. If a cracked domain ever becomes unbonded, the level is lost immediately. In Figure 2.3, the top-right yellow domain is a goal

domain, and the top orange domain is a cracked domain. Cracked domains have no scientific analog, rather they were added to have an easily understood failure state in the early introductory levels. This allows more esoteric failure states reached by incorrect or inefficient inventions to be explained later, after the player has better grounding in the game's mechanics.

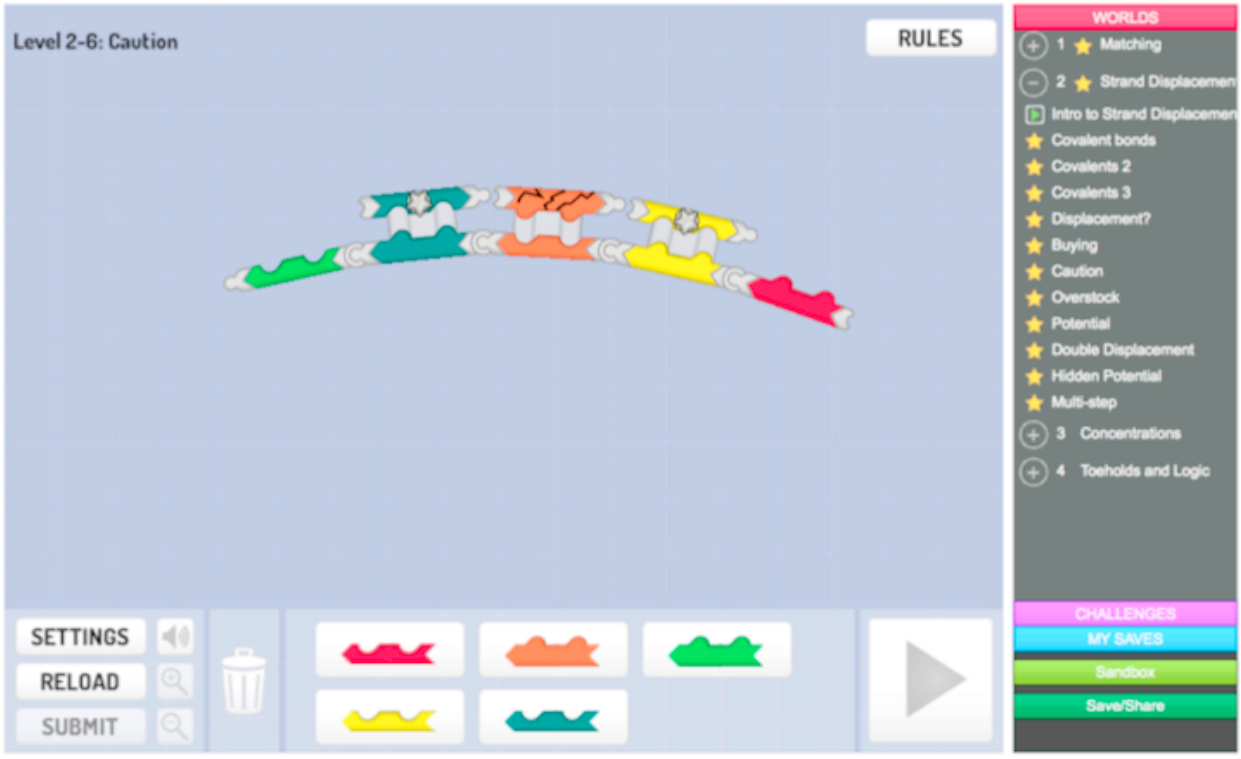

Figure 2.3. A Screenshot of Nanocrafter, showing the introductory level *Caution.*
The player must build an invention, using the domains from the menu on the bottom of the screen, that displaces the two starred domains while leaving the cracked domain hydrogen bonded (hybridized). The play button in the lower right starts and pauses the simulation. The sidebar on the right allows convenient navigation between different parts of the game.

2.4 DNA Nanotechnology Sandbox

While Nanocrafter was actively maintained, the game hosted challenge puzzles that generally lasted for two weeks and invited players to create inventions that addressed open-ended text

prompts, such as “Create three totally independent reactions using as few distinct colors as you can” or “Build two double-stranded molecules that exchange a strand when play is pressed”. Players could submit their inventions to an online gallery where other players could view and rate them.

To develop and refine the game, the team working on the game used an iterative design strategy involving the development team, play testers, and domain experts in synthetic biology, incorporating feedback to improve the game and address design challenges. Nanocrafter was released online in April 2014. The game was available to play for free on its website [30]. As of April 2015, the game had over 12,000 players, who submitted over 540 inventions to 55 challenges.

Nanocrafter also includes, in the challenge puzzles, drawing challenges. These are the same as regular challenge puzzles, except that they do not allow the simulator to be run (thus players can only create static layouts), and the challenge prompt relates to using domains to draw a picture. These drawing challenges were meant to be a fun and inviting introduction to the game’s challenge puzzles, bridging the gap between the introductory levels and the challenge puzzles, and allowing players to create and submit inventions and receive points for them without requiring them to address a complex scientific challenge. Some example inventions submitted by players to a drawing challenge are shown in Figure 2.4. These types of challenges also serve as a bridge to some of the structural DNA origami designs, designs for which could also be crowd sourced using a game strategy.

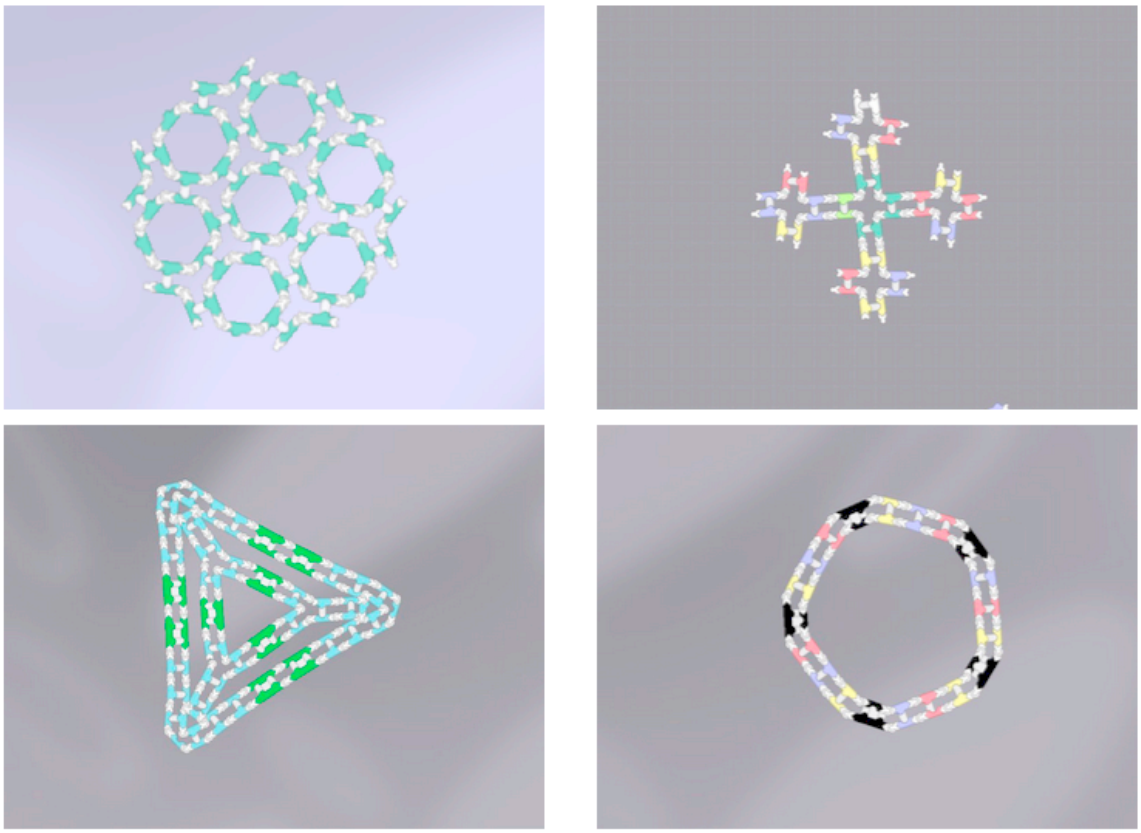

Figure 2.4. A Collection of Player Inventions Submitted to the Challenge *Draw a Fractal!*
Players were given the text prompt "Using the buttons below to add domains, draw something that looks similar at different zooms".

Challenges are interleaved with the introductory levels in such a way that players are exposed to the former at appropriate levels of understanding, e.g. they are presented with the drawing challenge after the introductory level that teaches how to add pieces, and with a concentration-related challenge after solving several concentration introductory levels.

The primary modes of interaction for players in the game are constructing an invention and simulating it. In order to construct an invention, players can create domains from a menu on the bottom of the screen (seen in Figure 2.3) within which they can customize the available domains. Once domains have been created, players can create a bond or a connection by dragging

domains close to each other and break existing connections or bonds by clicking on them. The layout of plexes is handled by a 2D rigid body simulation. This layout system allows players to build a variety of spatially complex systems. Players can bring up text bubbles showing the concentrations for plexes and also use that to edit concentrations.

Players control the simulation through the use of a play/pause button on the menu. They can also rewind and reset the simulation. Thus, the game has a single, unified construction and simulation space. Early game designs had two spaces: a construction space in which players would assemble plexes, and a separate simulation space where players would move plexes. Plexes would begin simulating as soon as they were added to the simulation space. However, the separate spaces seemed to be confusing for players, especially when players wanted to remove plexes from the simulation space after simulation had taken place. The single, unified space appeared to be more intuitive for players.

Scoring inventions in Nanocrafter presented a particular challenge due to the extremely open-ended nature of the game; there is no standard scoring function to use. We wanted to reward players for addressing the challenge puzzle prompt, but also to be as flexible as possible so that we would not presuppose specific types of solutions. Therefore, we chose not to implement any automated or objective scoring for challenge puzzles, but rather allow other players to rate inventions and for players to accumulate score through these ratings. This means that players do not receive immediate feedback on their inventions; however, they can start receiving ratings as soon as they submit an invention.

When rating inventions, players can choose to rate for a specific open challenge puzzle or browse the gallery. Inventions can be rated as any combination of valid (meaning they satisfy the

challenge prompt), interesting, or surprising. The player who created an invention will receive one point for each rating their invention receives in any category, shown on a leaderboard.

2.5 Validating DNA Inventions

Informal assessment of a subset of submissions (those which received the most favorable peer reviews, along with some hand-screened by the authors) suggests that the game achieved some success in cultivating mastery of DNA nanotechnology concepts in its players.

Figure 2.5 demonstrates how solutions submitted by Nanocrafter players both satisfy the constraints of the presented challenge and align closely with the results generated by Visual DSD. Visual DSD uses a domain representation similar to that of Nanocrafter, so the final state between the two programs should look very similar if they are given the same start state. Because of differing constraints between the two simulations, the start and end states are not absolutely identical; Visual DSD does not allow unbound long domains in the initial state, and it cannot visualize junctions between three or more strands.

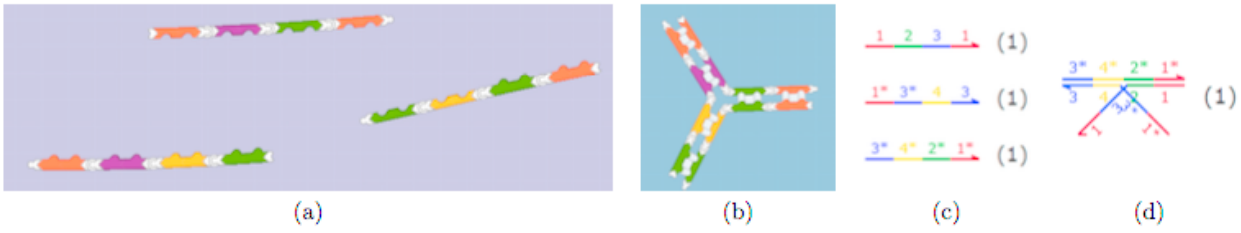

Figure 2.5. A Player Created Self-Assembling Three Way Junction.
Here, we show the (a) start and (b) end states from Nanocrafter and c) start and d) end states from Visual DSD. Note that in Visual DSD, the long domains were changed to short domains and one branch of the junction remains unbound as Visual DSD could not render junctions between more than two strands at the time of comparison.

Branched DNA is a topic of interest in DNA nanotechnology [15]. The player solution to the challenge “Create three strands that, when played, each bind to the other two strands” (Figure 2.5) demonstrates a working understanding of how multi-way junctions in DNA are formed. This entry was the highest-rated invention as judged by the other players, showing that players were able to correctly identify it as a strong solution. We also demonstrated that Visual DSD’s output is identical for the same input.

Self-assembly of arbitrarily large DNA structures is being investigated by several labs [56]. We put forth the expert-level challenge “Create some number of molecules that combine into a single repeating figure when played” to see if players could create a self-assembling system. The player solution (Figure 2.6) creates a system very similar to the one detailed in [16], elegantly creating a polymer using many copies of two distinct strands. This entry was the highest-rated invention as judged by the other players, and we were able to reproduce the output in Visual DSD, albeit with some non-functional changes to the input to satisfy Visual DSD’s constraints.

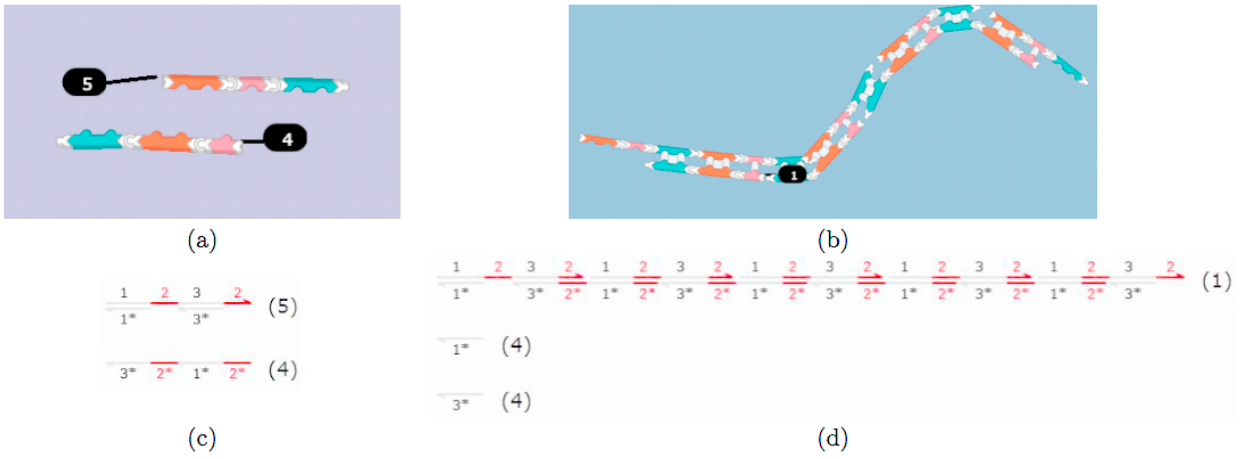

Figure 2.6. A Player Created Self-Assembling Polymer.
Here, we show the (a) start and (b) end states from Nanocrafter, and the (c) start and (d) end states from Visual DSD. Note that in Visual DSD the higher-concentration strand had complements added to its long domains, to satisfy Visual DSD's constraint that there are no reactive long domains in the initial state, and a second toehold was added to each strand to compensate for the bound long domains.

Reaction networks driven by DNA-based logic circuits are a key feature of DNA nanotechnology research [8]. Nanocrafter's tutorial levels feature the construction and use of several types of logic gates, and the early expert-level challenge "Make a system that releases the star only if one or both of the inputs on the left are deleted" tested to see if players could construct a working NAND gate. The highest-rated player solution Figure 2.7 comes extremely close to building a functional NAND gate, with a flaw in the directionality of one input and receptor preventing correct execution in some cases. Early demonstration of how a relatively complex logic gate is meant to function was encouraging. We were able to demonstrate that a very similar system in Visual DSD, fixing the directionality error and taking similar configuration liberties as in the previous example, produced the expected NAND gate logic table over the range of inputs.

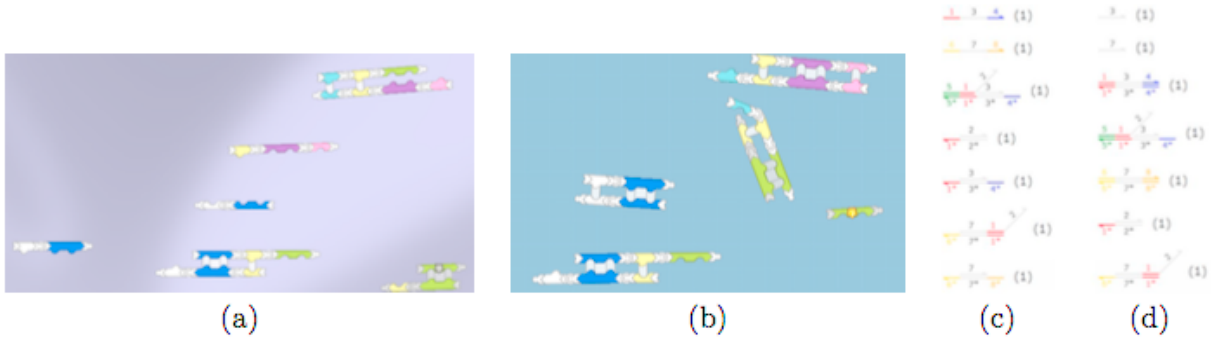

Figure 2.7. A Player Created, Partially Functional NAND Gate.
Here, we show one possible start and end state (one of the two possible inputs created) of the (a) start and (b) end state from Nanocrafter and (c) start and (d) end state from Visual DSD. Note that in Visual DSD one of the two logic gates and both receptors had complements added to their long domains, to satisfy Visual DSD's constraint that there are no reactive long domains in the initial state.

2.6 Challenges and Future Directions

2.6.1 *Validating Designs Experimentally*

Designing these DSD systems becomes more interesting if they can be experimentally built and verified. One method to programmatically verify the designs that are built in the game would be to use a Smart Wet Lab system. The SOS lab at the University of Washington has implemented a smart wet lab system where experimental protocols are written to be displayed on a screen and followed precisely by a lab technician. The idea is to have technicians bulk perform wet lab protocols that are repeated frequently instead of having scientists each do them individually.

The smart wet lab could interface with a scientific discovery game to rapidly build and verify DNA Nanotechnology designs. This would work by automatically translating the designs from the Nanocrafter game into a sequence-level design using DNA sequence design tools, like NUPACK [57] or Coral [58]. The sequences could then be inputted into the smart wet lab

database, ordered automatically, and then passed through a series of experiments to test the validity of the designs.

As a proof of concept, I wrote smart lab protocols that would bulk produce plasmids with a particular DNA sequence (maxipreps) and then digest them with enzymes to create nicked, double-stranded DNA gates. Lab technicians were able to successfully create the plasmids of interest, some of which were used in experiments done in Chapter 3. Unfortunately, this idea was not pursued further due to the cost of using a smart wet lab system like this (it's much cheaper to just use grad student labor than to pay for technician labor) and the difficulty of posing interesting challenges in the Nanocrafter game.

2.6.2 Fidelity of DNA Simulations

While working on the game, we continually faced the challenge of how much to abstract away the scientific reality to make the game understandable and fun for players. One challenge is that the DNA reactions are stochastic in nature when you zoom down to the single molecule level (which is typically what the game visualizes).

One way we tried to address this was to add concentrations (stacks) to increase the numbers from single molecules interacting at a time. One other challenge was to address the idea that toeholds could interact with multiple complexes. This proved to be a simulation and visualization challenge. Many of the game developers felt that having a stochastic simulation makes gameplay frustrating for players that may expect a design they build to act the same way every time they simulate it. The ideal way to deal with this would be to make use of a stochastic simulation engine, perhaps the same core algorithm as Visual DSD, that considers the many

possible reaction trajectories in the game, then visualizes it to players in an intuitive but clean way on the screen.

2.6.3 Social Elements

Nanocrafter included a chat box where players could talk to each other, however, adding groups or teams would allow players to easily collaborate and work together on their solutions, rather than working individually. The idea is to give players the tools to create modular components that could then be shareable or collectively edited. For example, once a player has built a NAND gate, they could save it as a component and reuse it to create more complex logic circuits or share it with other players who could then make their own variations. This approach could enable the more rapid construction of advanced inventions and allow expertise and discoveries to diffuse more easily through the player community.

2.7 Game Conclusions

Our team of artists, developers, and scientists were able to successfully create a visual programming language for DNA strand displacement that was intuitive to understand for game players. Challenges that occurred were faithfully rendering the stochastic nature of molecular interactions and posing challenges that would result in novel inventions from game players. Successfully, players were able to understand the basic mechanics of DNA strand displacement and build several inventions in response to challenges posed by the game administrators. Additionally, the game was used in several classes taught at the University of Washington to teach students the basic rules of DNA strand displacement. This work was presented at several conferences, including MPP 2013 and FDG 2015.

Chapter 3. CONTEXT-INDEPENDENT NICKED-DOUBLE STRANDED DNA GATES

3.1 MOTIVATION

Two domain nicked-double stranded DNA gates have been shown to be able to implement arbitrary chemical reaction networks, serving as a primitive for a molecular programming language. We would like to be able to build arbitrary chemical reaction networks with predictive kinetics that will allow us to compose larger circuits without the necessity of characterizing individual toehold kinetics. In light of this, we designed context-independent gates to fit specific design criteria that should standardize the kinetic reaction rate and allow a single toehold reaction rate to accurately describe each of the strand displacement kinetics in the circuit. The criteria we use to design the context-independent gates was determined by studying the performance of previous implementations of chemical reaction networks using this two-domain architecture.

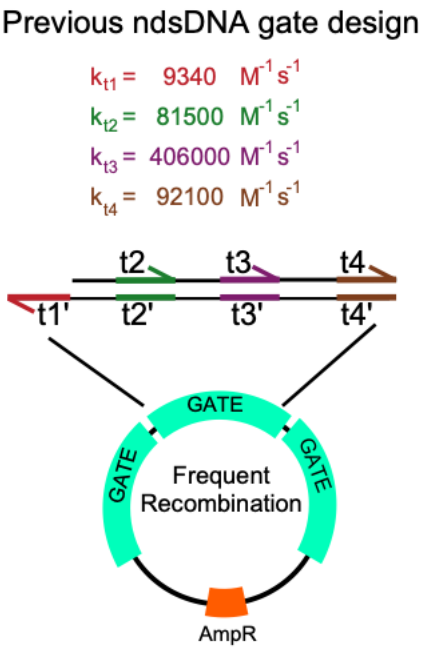

Figure 3.1. Previous ndsDNA Gate Design.
Above, the join gate design for building any arbitrary CRN with DNA components is pictured. Every toehold has a different sequence and has a different strand displacement reaction rate. These gates are derived via enzymatic nicking from a plasmid cloned with Ampicilin resistance and three copies of the full double stranded version of the gate.

3.2 Chemical Reaction Network Mechanism

Arbitrary chemical reaction networks can be implemented with nicked double stranded DNA gates [6]. Each formal chemical reaction is implemented as a sequence of individual strand displacement reactions that emulate the correct behavior in combination. Species in the chemical reaction are represented by single-stranded DNA molecules that are comprised of a short toehold followed by a longer domain that encodes the signal's identity. These species don't react directly together, but instead two nicked double stranded DNA (ndsDNA) gates are needed to emulate the chemical reaction. The first gate, termed a "Join" gate, binds the reactant and releases a translator strand through a series of reversible strand displacement reactions facilitated by helper strands. An additional helper strand is bound to irreversibly finish the Join gate reactions.

The translator strand expelled by the Join gate can then bind to the open toehold on the "Fork" gate. This results in the release of the reaction product after another cascade of strand displacement reactions facilitated by helper strands. A final helper strand binds irreversibly with the Fork gate to end the reaction. The output of the reaction can be read via a fluorometric output when the product triggers a reporter complex. Alternatively, the product, which has the same structure as the reactants, can participate in a downstream chemical reaction.

Any arbitrary chemical reaction can be emulated with this modular design as the join and fork gates can easily be modified to accommodate the binding and release of any combination of reactants and products. Additionally, signal strands all have the same structure such that multiple reactions can be coupled together into arbitrary networks. However, despite the many advantages of this design, a major drawback is that the kinetics of the overall chemical reaction networks are not predictable without extensive characterization of the individual strand displacement reactions.

In the previous implementation of the two-domain CRN gate architecture, strand displacement reaction rates had to be parameterized separately, and fitted rates were found to vary over more than an order of magnitude, despite being designed to have similar toehold sequences (see Figure 3.1). Therefore, it is clear that it is not sufficient to assign strand displacement rates based purely on toehold sequences; the sequence context surrounding the toehold, that of the recognition domain, affects the strand displacement rate as well. This presents a problem for designing larger circuits, as the time necessary for characterizing individual reactions makes it prohibitive to build larger circuits.

3.3 GATE DESIGN

We would like to improve upon the design of the nicked-double stranded DNA gates, such that the kinetics of the chemical reactions implemented by these gates can be accurately modeled by using a single toehold reaction rate to describe every strand displacement reaction that occurs in the system. Context-independent DNA gates were designed, by myself and Yuan Jyue-Chen, with specific criteria in mind, determined by studying the performance of the previous design. The first criteria is that all toeholds in the gate should have the same "universal" sequence. This is to ensure there are no effects from differing binding energies and variations in sequence composition. Additionally, the toeholds are designed to be palindromic meaning that the sequence is the same when read starting from either the 5' or 3' end. This is because reactions on the join gate happen in one direction, whereas they occur in the other direction in the fork gate.

The second criteria used to design these gates is that all toeholds are designed to be "internal", meaning that they must be flanked by a double stranded region of DNA on both the 5' and 3' end. This is done because internal toeholds appear to result in much faster strand displacement reactions as opposed to "external" toeholds, which are only flanked on one side. To accomplish this design criteria, we add dummy domains on either side of the gate.

The third design criteria used is to standardize the regions of DNA bases on either side of the toehold. This is done to ensure that all the toeholds have similar stacking energies. This is done by making the toehold neighbor sequences (5-9 bases) the same on either side of the toehold. Because of the nature of our gates being enzymatically digested, the sequences of these regions is largely determined by the sequence of bases necessary to achieve nicks in the correct places. Ideally, we'd hope that these design criteria should ensure that all strand displacement steps have an identical strand displacement reaction rate. We can test this by using a model with

a single, universal reaction rate to fit all the strand displacement steps and see if it can correctly describe the kinetic behavior.

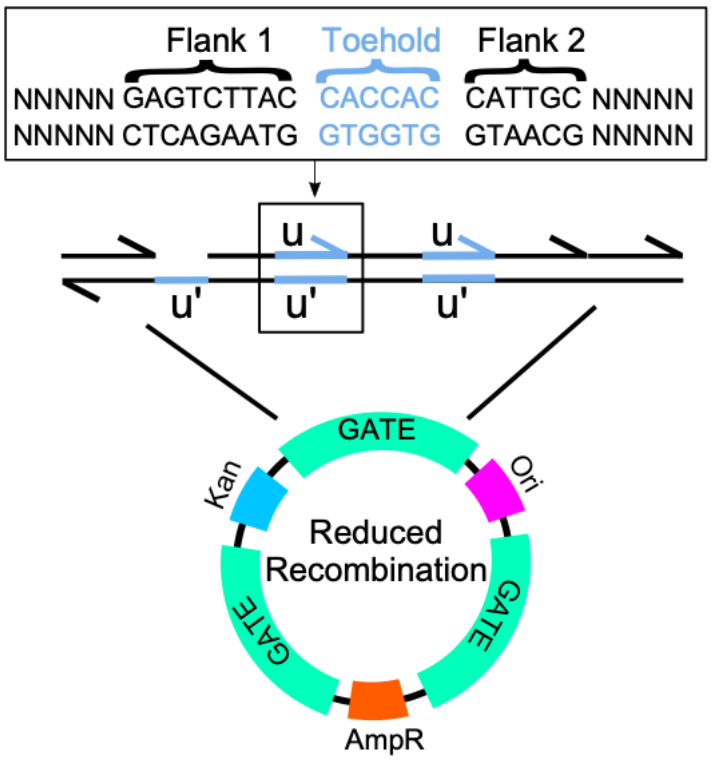

Figure 3.2. The Context-Independent Gate Design.
This design results in a consistent strand displacement reaction rate. Toeholds are designed such that they have the same, palindromic sequence and are flanked by the same sequence on either side. The ILN plasmid design has copies of these gates cloned in between important markers for plasmid viability so that recombination is reduced. The gates are then processed into a functional form with a short series of enzymatic digestions.

3.1 Palindromic Gates

A question we might initially ask when trying to design gates is what is the effect of just standardizing the toehold region? Additionally, since we know that our gates require strand displacement reactions to happen in both directions (displacement reactions on the join gate

happen in a different direction than displacement reactions on the fork gate), we might be curious to know what the differences, if any, are between the different strand displacement reaction rates.

We designed “palindromic” gates, which consist solely of palindromic domains to see if the reaction rates were different from either side. This starts to approach several of our criteria for designing context-independent toeholds; having the same toehold and having one of the same flanking region. However, there is not a flanking region on the other side, so these toeholds are still considered to be “external”.

Interestingly, the two different gates do appear to behave slightly differently. These gates were tested using a spectrophotometer and were modeled using the inference module of VisualDSD [59]. The reverse palindromic gate seems to have a reaction rate that is slower than that of the forward palindromic gate. The forward palindromic gate is modeled well by a rate of 0.00266 /nM/s and the reverse gate is modeled well by a rate of 0.00143 /nM/s. This suggests that that direction of strand displacement may be slower than the forward direction. However, these gates are already within an order of magnitude of one another when modeled with a universal toehold reaction rate. This is already a vast improvement over the previous design which had toehold rates that varied over several orders of magnitude.

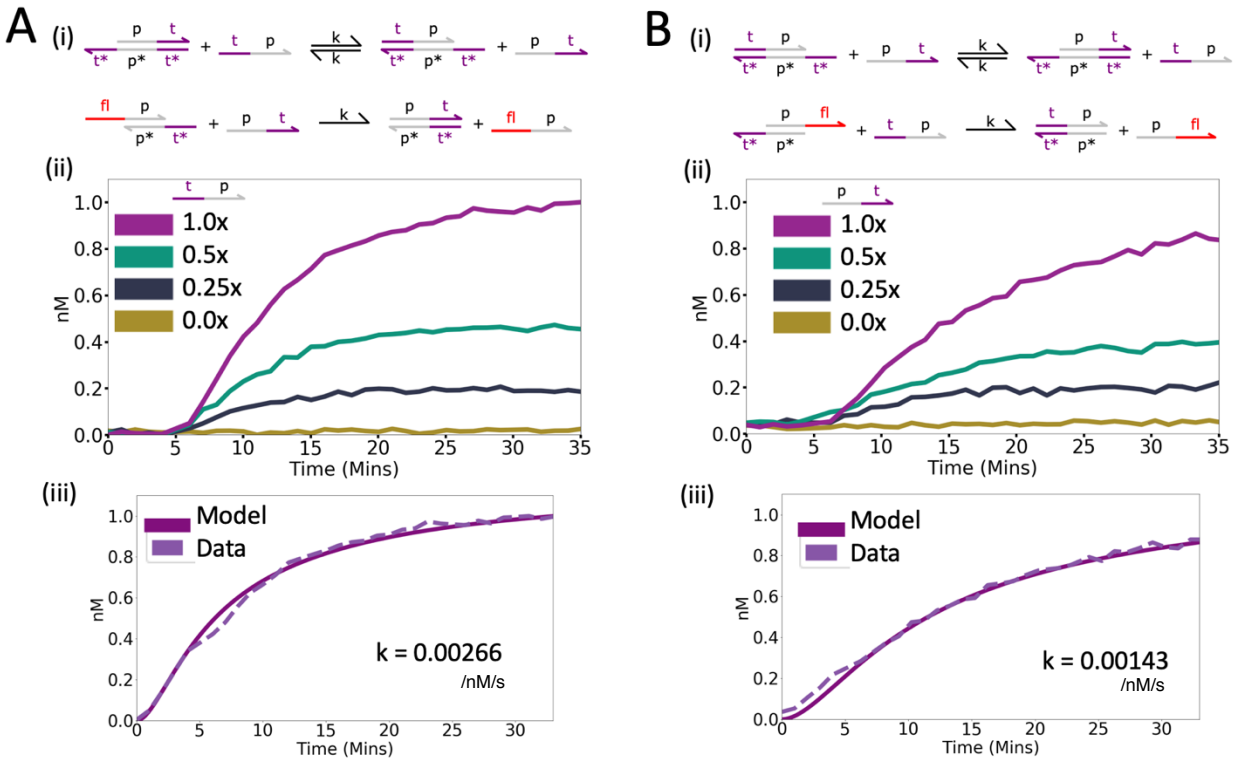

Figure 3.3. Palindromic DNA Gates.
Panels in A) show the forward palindromic gate while panels in B) show the reverse palindromic gate. Panels i) show the VisualDSD DNA implementation of the palindromic gates. Panels ii) show the kinetics of the palindromic gates when gates are at 1.5x, reporter complexes at 3x and a varied amount of the input is added (1x = 1 nM). Panels iii) show how a VisualDSD model of the DNA gate describes the kinetic behavior of these gates and gives a kinetic rate for the strand displacement reactions. All experiments are run with 0.15% SDS and 1 uM polyT in 12.5 mM TAE/mg2+ buffer.

3.2 Elementary Gates

Since our plasmid-derived join and fork gates have strand displacement reactions that happen in different directions, our gates each require different enzymatic processing to take on their functional form. We wondered what the effect of using different nicking enzymes for each gate and the different direction of displacement would be. We designed elementary gates that only took one input and could be digested to be either fork or join gates, depending on the enzymes

that were used. These elementary gates now have toeholds that are all internal, so we can see if this standardizes the rate further.

We then tested the kinetics of these gates at low concentration and analyzed the kinetic rate using the inference module of VisualDSD. The model follows the general framework as outlined in the mechanistic strand displacement level model section of this thesis, including parameter terms for the kinetic rate, a leak and bad rate, a time of trigger addition, and a kinetic rate for a blunt end.

After testing the gates at low concentrations, we discovered that the X4e_join (X4 elementary join gate) and X4e_fork gate (X4 elementary fork gate) had remarkably similar reaction rates. This suggests that standardizing the toeholds to be internal does indeed standardize the reaction rates of the gates. Notably, the X4e_fork gate was still somewhat slower than the X4e join gate, suggesting that the fork kinetics may have strand displacement reactions that occur at a slightly slower rate. This is consistent with other experimentation in the field, but these differences due to strand displacement directionality may matter little when trying to model and predict the overall dynamics of a larger DNA circuit or even an individual chemical reaction, which always require strand displacement reactions in both directions to complete a full reaction.

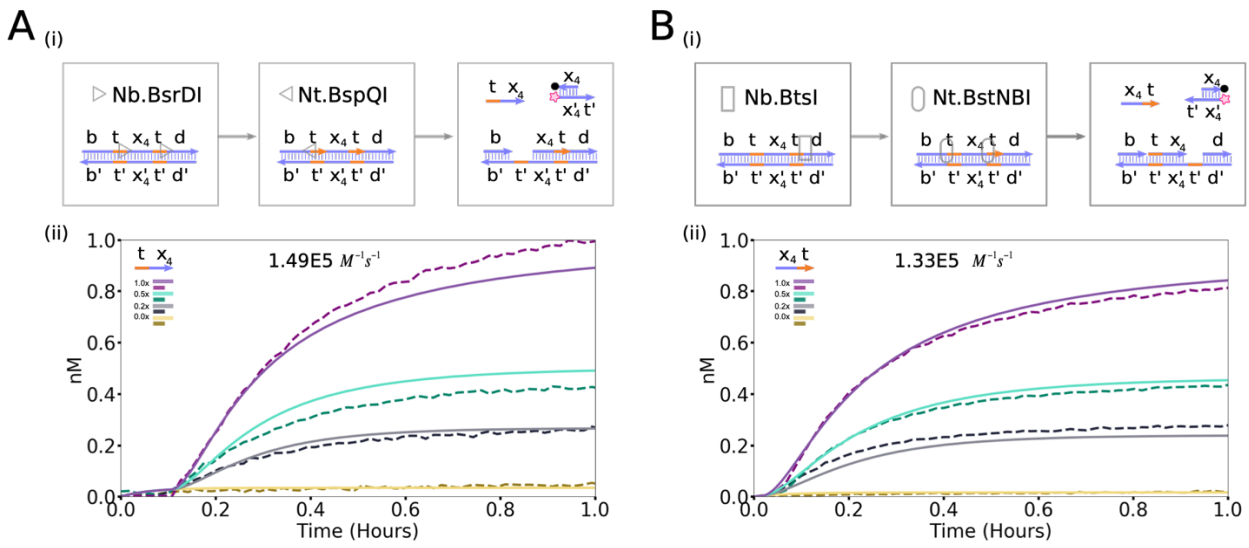

Figure 3.4. Elementary Gate Dynamics.
Panel A) shows the X4 elementary join gate and panel B) depicts the X4 elementary fork gate. Panels (i) show the digestion scheme of these gates. Notably, these gates start out with the same double stranded DNA structure and then are digested with different methods to produce either the join or fork version of the gate. Panels (ii) show the kinetics of the reaction. Gates are at 1.5 nM, Reporters at 3 uM and the input is varied as shown in the legen (1x = 1 nM). The dotted lines show the experimental kinetics and the solid is the model.

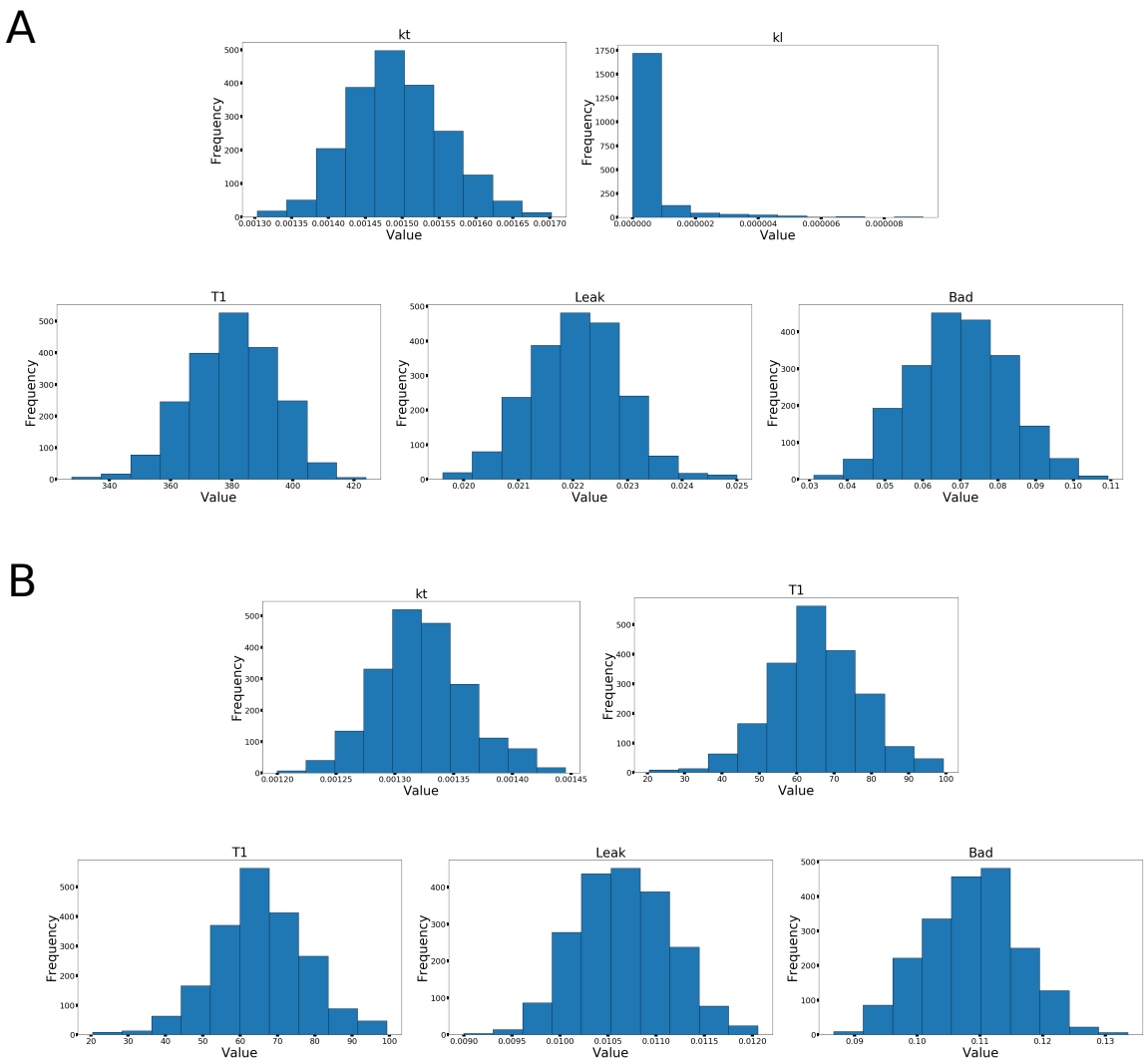

Figure 3.5. Elementary Gate Parameters.
Panels A and B show the histograms of the parameters inferred with the VisualDSD model for the X4 elementary join gate and the X4 elementary fork gate respectively.

3.3 CONTEXT INDEPENDENT DNA CRNS

The impetus behind designing context-independent gates is that engineers can come up with the reaction that they would like to implement, create a strand displacement implementation of the reaction that includes the universal toehold design, and now the reaction should be modelable by

a single toehold reaction rate, making the reactions quantitatively predictable and greatly simplifying the characterization necessary to implement larger circuits.

Context-independent use the original two-domain architecture, which is that reactants in the CRN are represented by single-stranded DNA that include a toehold followed by a longer domain-identifying region. In discussions of these reactants, we will now often simply refer to the name of the identifying domain, since all reactants have the same toehold now. Instead of these strands of DNA reacting directly together, they interact with a join gate that consumes the reactants and then causes strand displacements to release products in the same two-domain format.

The constraints for designing context independent gates results in the join and fork gates shown in Figure 3.6. Each gate now is now flanked on either end with a double stranded region of DNA to ensure that every strand displacement reaction occurs with an internal toehold. All the toeholds used are the same, palindromic sequence. Additionally, the join and fork gate now both have reactions that can occur to irreversibly react them away into waste. This is slightly different from the original design but does not affect the stoichiometric way in which these gates react. Figure 3.6 C) depicts the strand displacement level reactions with the new context-independent gates that occur to implement the overall chemical reaction $X \rightarrow X + Y$.

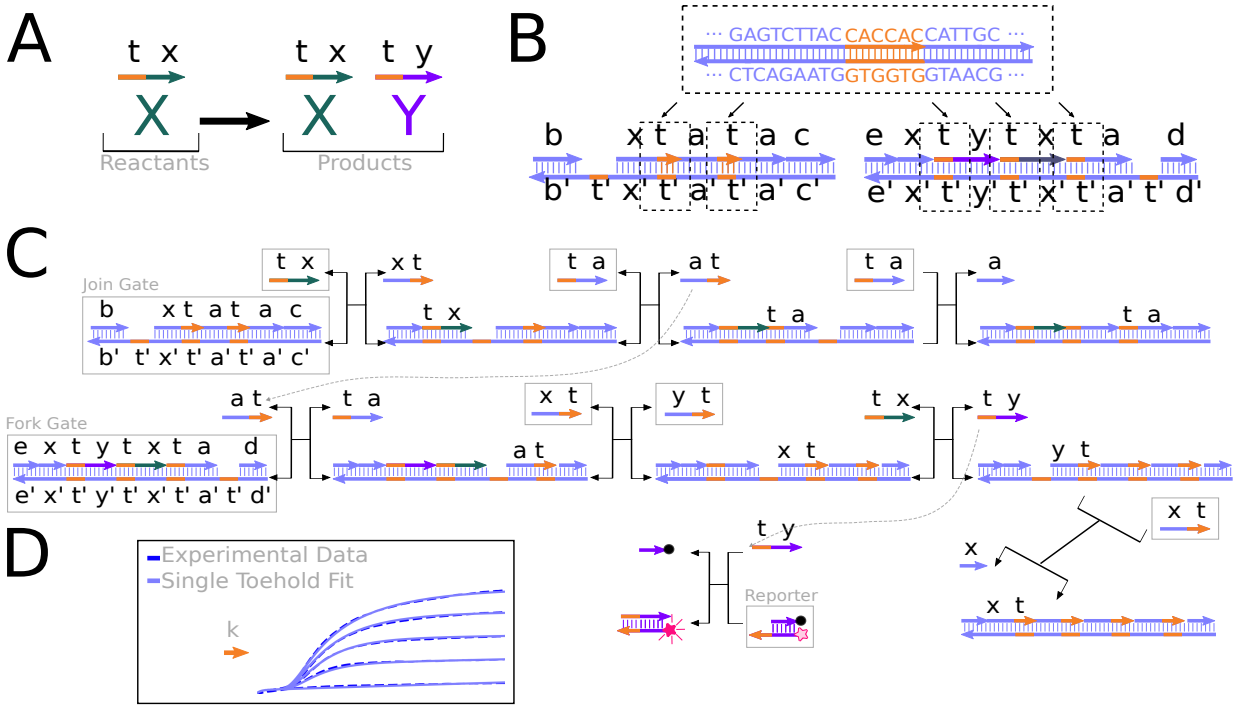

Figure 3.6. DNA Realization of Formal CRNs with Context-Independent Gates.
A) Target behaviors can be achieved by designing DNA architectures specified by the programming language, which is the CRN formalism. B) Schematic for next generation plasmid-derived gate design for the reaction X → X + Y. Join gate that first binds yellow single stranded input X, and then an auxiliary strand in a cascade of DNA strand displacement reactions, releasing the blue translator strand. The cascade of join gate reactions is terminated by an irreversible last reaction, where an auxiliary strand binds. C) The fork gate binds the translator strand and then several auxiliary strands that cause the release of the products X and Y and the termination of the strand displacement cascade by an irreversible reaction. D) The reporter strategy for the reaction takes the strand Y and releases a fluorophore from a quencher strand via a strand displacement reaction. E) Previous ndsDNA gate designs required individual parameter fits for each strand displacement reaction, which varied widely per toehold. F) Next-generation gate design architecture is designed such that a single parameter for a “universal” toehold kinetic rate is sufficient to model the CRN well. Universal toeholds each have the same, palindromic sequence and are flanked on both ends, to ensure toeholds have similar stacking energies.

3.3.1 *Methods*

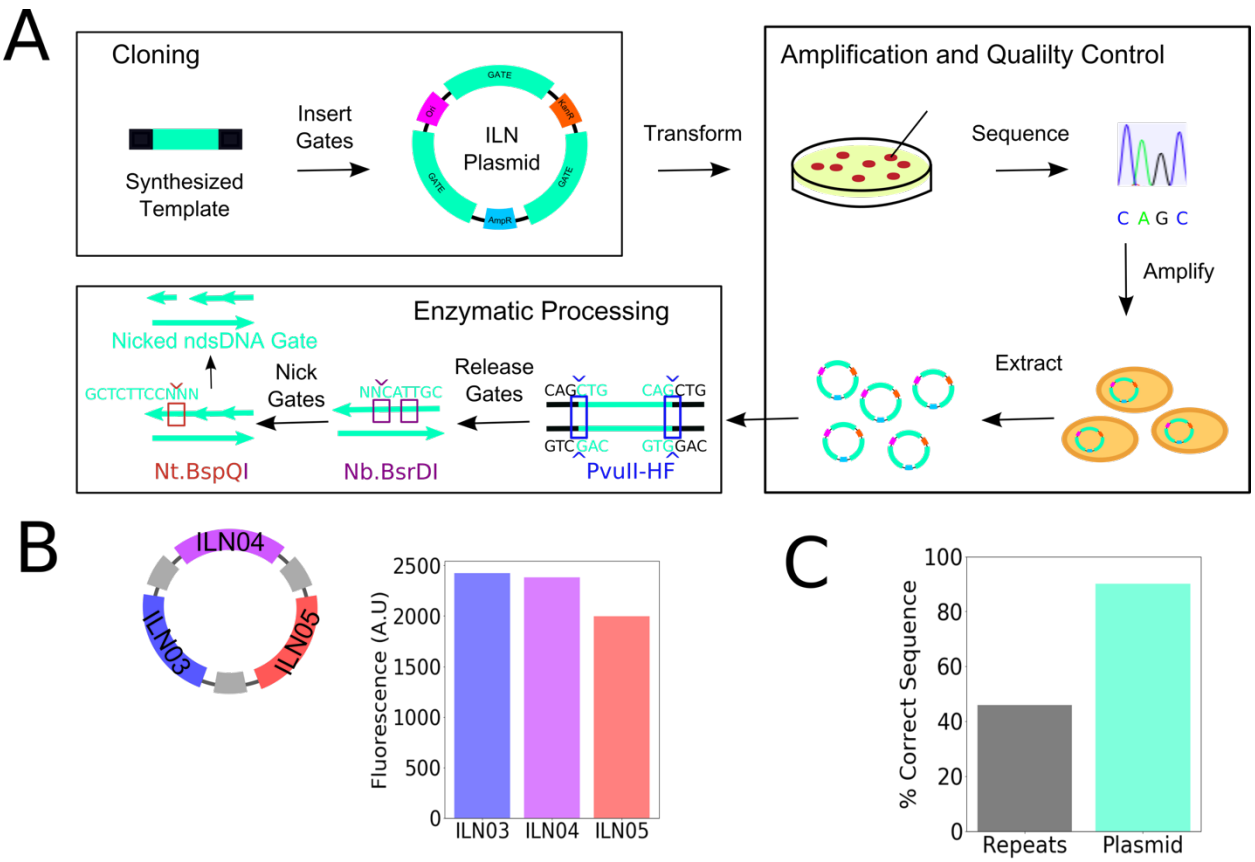

Figure 3.7. DNA Gate Production.
High fidelity ndsDNA gates can be sourced from plasmids. A) Multiple copies of a double-stranded template are cloned into a plasmid using the ILN design. Plasmids are then transformed and sequence verified. Sequence correct clones are amplified and extracted. Finally, a series of restriction enzyme digests release and process the gates into a functional form (Join gate enzymes shown in figure). B) Three different join gates are put on a plasmid. When individual gates are triggered to their max and reveal similar fluorometric outputs, showing that this method can give us stoichiometric outputs. C) When sequenced from plasmid stocks or glycerol stocks, ILN plasmids are correct 90% of the time as opposed to the method where the gates are cloned into a plasmid as direct repeats (which only gives correct results 46% of the time).

High fidelity ndsDNA gates for implementing chemical reaction networks with the two-domain architecture can be sourced from plasmids [60]. When encoding multiple copies of the same sequence into a single plasmid, deleterious recombination events become problematic. These deletions may occur in a few ways, most commonly strand slippage or homologous recombination. The most likely case of deletions in plasmids with multiple copies of cloned

gates is due to intra-molecular recombination between direct repeats of DNA [61]. In this case, direct repeats of DNA may result in a recombination event that loops out the intervening DNA. The length of the repeats increases the frequency of recombination. However, even short repeats may result in deletions.

This problem is addressed using inter-marker located ndsDNA (ILN) plasmids. In the ILN plasmid schematic, each fully double-stranded gate is separated by a genetic part necessary for plasmid viability. This ILN design uses the Ampicillin resistance gene, the Kanmycin resistance gene and the pMB1 origin for pUC as the genetic markers that separate copies of the gates. This way, if a loop-out recombination occurs, the plasmid is no longer viable. In principle, more copies of the gates can be added to the plasmids by adding more markers necessary for viability.

The previous cloning strategy was to encode three or four copies of the gate in a row on a plasmid with Ampicillin resistance. From 160 sequencing results (106 of the ILN design, 54 of the old design), ILN plasmids transformed from plasmid stocks or plated from glycerol stocks resulted in correct sequences 90% of the time, an improvement from the previous designs 46%. Virtually none of the ILN plasmids showed simple loop-out recombination.

3.3.1.1 ILN Plasmid Cloning

Multiple copies of a double-stranded DNA template are inserted into a high copy number plasmid via Gibson cloning. The plasmids are then chemically transformed into *Escherichia coli* strain JM109, which is recA- to minimize recombination and endA- to improve plasmid DNA quality. A single colony is then picked from the Ampicillin and Kanmycin selective plate and grown in a 3 mL of TB with Ampicillin (100 mg/L) and Kanmycin (50 mg/L). Plasmids are sequence verified via Sanger sequencing and then amplified using a Qiagen HiSpeed Maxi-prep

kit on an 800 mL overnight culture. Maxi-prepped plasmids are sequence verified again to ensure deleterious recombinations did not occur.

3.3.1.2 Enzymatic Digestions

Gates are released from the plasmid backbone via digestion at 37° C for 1 hour with 5 units of the restriction enzyme PvuII-HF per 1 ug of plasmid. This first enzymatic digest has been shown to be optional. Gates will still trigger if not released as fragments from the plasmid. However, if cleanup of excess DNA is desired, this step should be performed. For the second digest, Join gates were then digested with 2.5 units Nb.BsrDI per ug plasmid at 65° C. Fork gates are digested with 4 units Nb.BtsI per ug of plasmid at 37° C for 1 hour. Following the second digest is a brief concentrating and buffer exchange step, using Amicon Ultra-0.5 Centrifugal Filter Units, to optimize conditions for the third digest. Alternatively, an ethanol precipitation can be performed and then gates can be resuspended in 100 uL of molecular grade water. The third digest uses 1 unit Nt.BspQI per ug DNA for the join gates to be digested at 50° C and 1.15 units of Nt.BspNBI per ug for the fork gates to be digested at 55° C for 1 hour. The third digest opens the initiating toehold, by nicking the other side of the toehold, and allowing the DNA bound to the toehold to dissociate from the gate.

3.3.1.1 Calibrating Gates

After digesting the gates, we typically don't do any clean up for any of the excess DNA. Thus, it is not possible to measure the concentration of DNA gates via traditional methods, like the nanodrop. Instead, we create a calibration curve of the reporter complex specific to the last output of the gate in the case of the Join gate, specific for the translator strand), and trigger the gate with known concentrations of inputs. The concentration of the input strand is able to be measured and is thus very precise. In the nanomolar regime, this relationship is known to be

linear so we can simply do a linear regression to create a calibration curve that directly correlates the fluorescent output of the gate directly to the concentration of input. We can then trigger a known volume of gate with an excess of input strands, measure the fluorescent output, and then calculate the concentration of gate using the calibration curve.

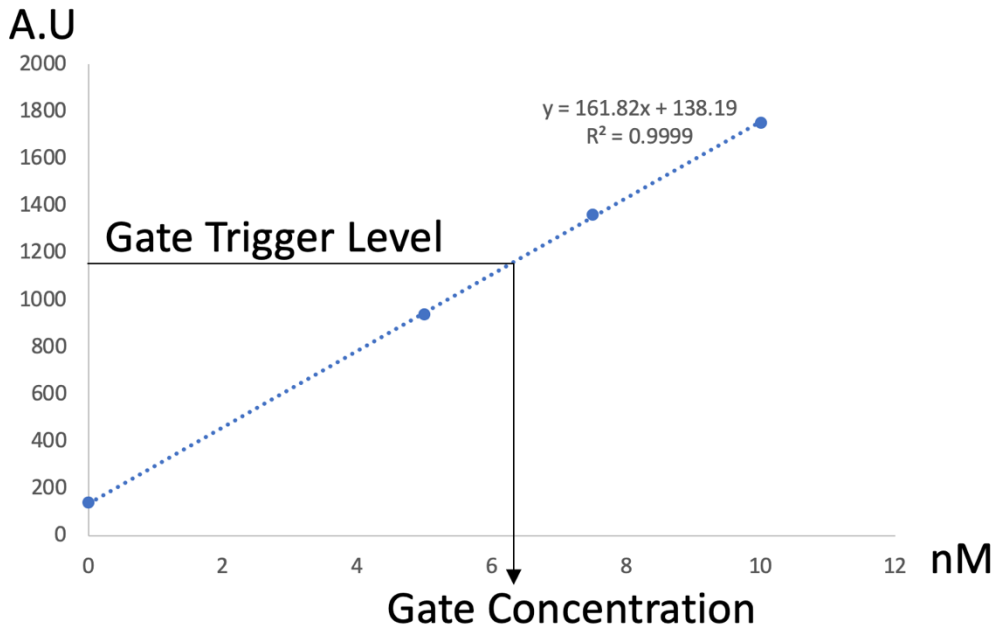

Figure 3.8. Calculating Gate Concentration.
The calibration curve is created using known concentrations of inputs to a reporter complex. Then the fluorescence level when a known volume of gate is triggered is used to determine the gate concentration.

3.3.1.2 Control of Stoichiometry

Control of stoichiometry can be achieved by putting multiple types of gates on one plasmid. In the case of the rock-paper-scissors oscillator (Section 3.5.2), we put all the join gates on one plasmid and all the fork gates on another plasmid. Individual gates are triggered to their max with an excess of helper strands and then normalized to reveal similar fluorometric outputs, showing that this method can give us stoichiometric outputs. We do a similar experiment with

the consensus network gates. The join gates are all reported by reacting the last output from the gate with a reporter complex containing the fluorophore TAMRA. The fluorescence values can be directly compared (as seen in Figure 3.7) and normalized levels of triggered gates are shown in Figure 3.9 A. Fork gates are all reported by reacting the last output from the fork gate with reporter complexes carrying various fluorophores so raw fluorescent values are not shown. Normalized fluorescent values are shown in Figure 3.9 B.

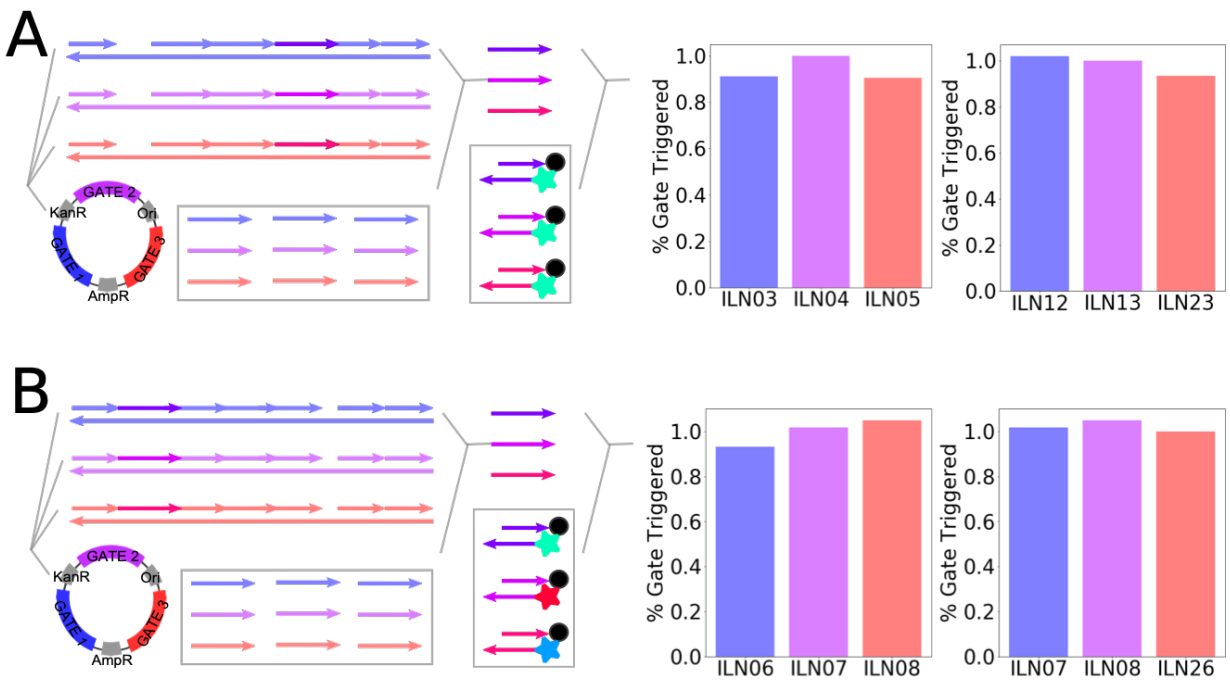

Figure 3.9. Stoichiometric Gates.
Panel A) show the stoichiometric design for inter-marker located nicked double stranded DNA join gates. Three different join gates can be placed on a single plasmid. In an experiment, we trigger each gate on the plasmid with an excess of helper strands separately and compare the final output level (normalized) to see how stoichiometric the gates actually are. Two different join plasmids were built and tested for stoichiometry. The trigger levels of each gate are consistent with one another. Panel B) show the stoichiometric design for inter-marker located nicked double stranded DNA fork gates. Three different fork gates can be placed on a single plasmid. In an experiment, we trigger each gate on the plasmid with an excess of helper strands separately and compare the final output level (normalized) to see how stoichiometric the gates actually are. Two different fork plasmids were built and stoichiometry experiment results are shown.

3.3.2 *Behavior of context-independent gates*

We use the principles outlined in the design and methods sections to build gates that will realize arbitrary chemical reactions when composed together. The flank regions are dictated by the enzymes that are necessary to digest the gates from the plasmid form. The rest of the domains are designed in NUPACK to ensure that the helper and input strands will not have much secondary structure or have secondary interactions with other helper strands. We built many gates using the ILN plasmid schema and tested them in either a spectrophotometer or a plate reader.

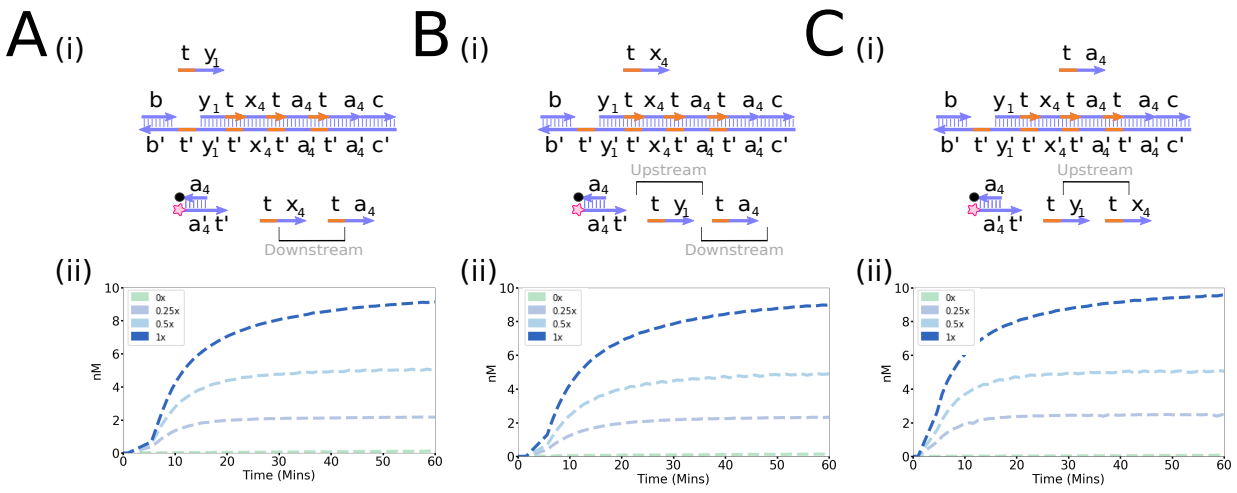

Figure 3.10. Kinetic Behavior of Join Gates.
Shown are the experimental observations for datasets that characterize strand displacement reactions on the JoinY1X4 gate. The experiments directly observe the forward reactions (from left to right on the gate A) <t y1> displacing <y1 t> B) <t x4> displacing <x4 t> C) <t a4> displacing <a4 t> In Panels A-C, 1x = 10 nM, and we start with 3x gates and reporters. All gates are pre-mixed with 10x of strands upstream and 3x of strands downstream of the target interaction.

We tested several join gates to see that their kinetic behavior matches what we would expect from the join gate. In Figure 3.10, we see a toehold experiment done on gate ILN14 which joins

input strands <t y1> and <tx4>. From now on, we'll refer to this gate as JoinY1X4 or ILN14_JoinY1X4 to denote the plasmid it was derived for. We'll continue this terminology throughout the document.

In the toehold experiment, we take the same gate and do several tests where we add each successive input or helper strands in varied amounts while adding an excess of any upstream helper strands and a limited amount of downstream strands. This helps us analyze the individual toehold reaction rates. We see in testing the later toeholds, the reaction rate is faster as we would expect in the experimental set-up. In Figure 3.10 panels (ii), we can look at the time the half max of the 1x input is reached in each case, or the time at which 5 nM is reached. In the third toehold, the half max is reached around 5 mins, in the second toehold, the half max is reached around 10 mins and the first toehold experiments sees the trace reach half max around 12 minutes. This is precisely what we would expect to see, as the reaction should proceed faster when several of the gate reactions have been triggered already.

Several of the join gates we built can be composed with fork gates to make autocatalytic reactions. In particular, ILN03_JoinX2X4, ILN04_JoinX3X4, and ILN05_JoinX4X3 are used in a set of autocatalytic reactions that could theoretically make an oscillatory network when composed (see Section 3.5.2). We look at the kinetics of these gates (see Figure 3.11) in particular to make sure that the gates have similar kinetics, necessary for creating a larger network where reaction rates must be similar. Despite differences in the final signal reached (which may be due to errors in gate calibration or pipetting technique), these gates achieve the half max in the 1x input scheme in around the same time scale (5-10 minutes) for the first toehold reaction. This is a good indication that our schema for standardizing the toehold reaction rates has been successful.

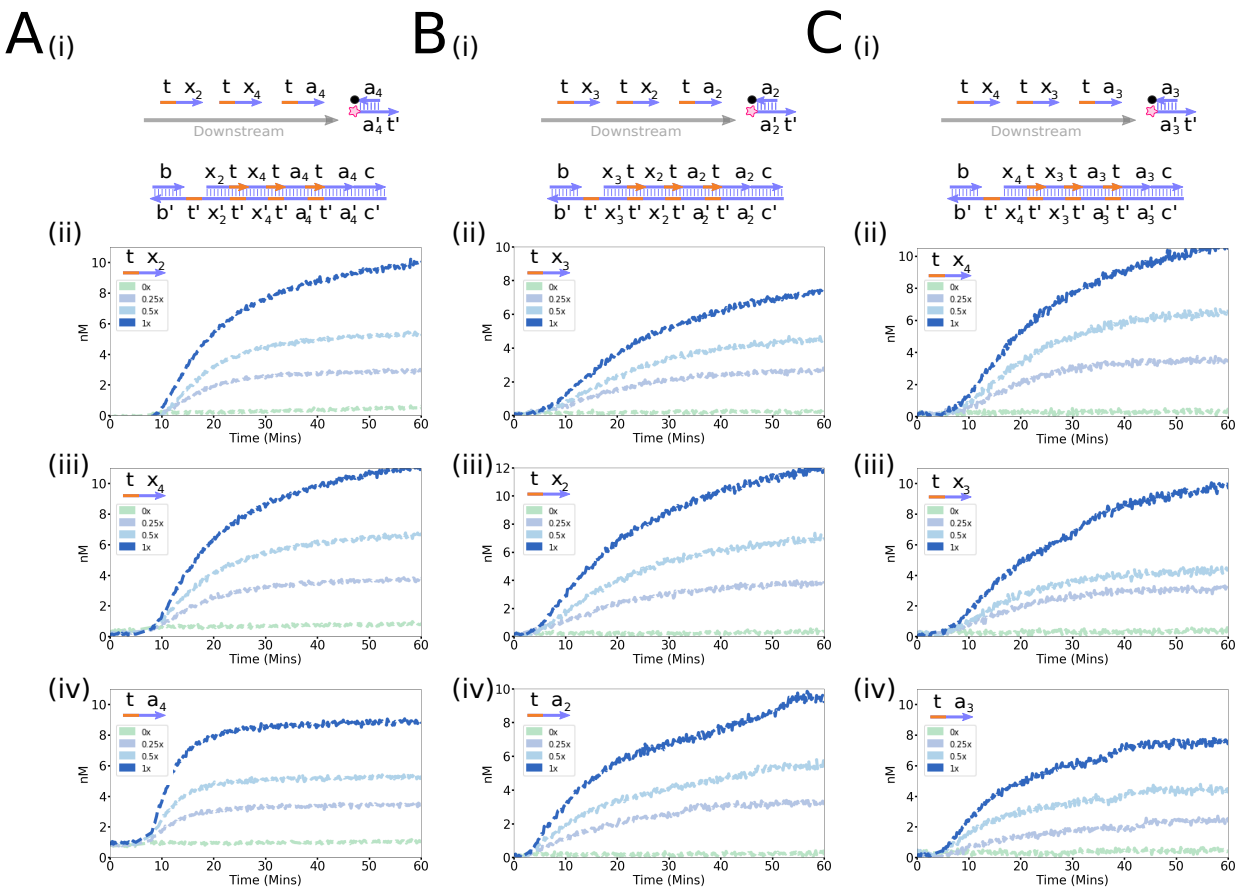

Figure 3.11. Join Gate Kinetics for Oscillator Gates.
Panel A) shows ILN03_JoinX2X4, Panel B) shows ILN04_JoinX3X2 and Panel C) shows ILN05_JoinX4X2 toehold kinetics. Panels (i), (ii), and (iii) show the gate kinetics varying successive input strands (boxed in red), with the upstream input strands and helper strands at 10x and the downstream strands at 3x. These experiments are done with the gate at 3x and the reporter strands at 3x.

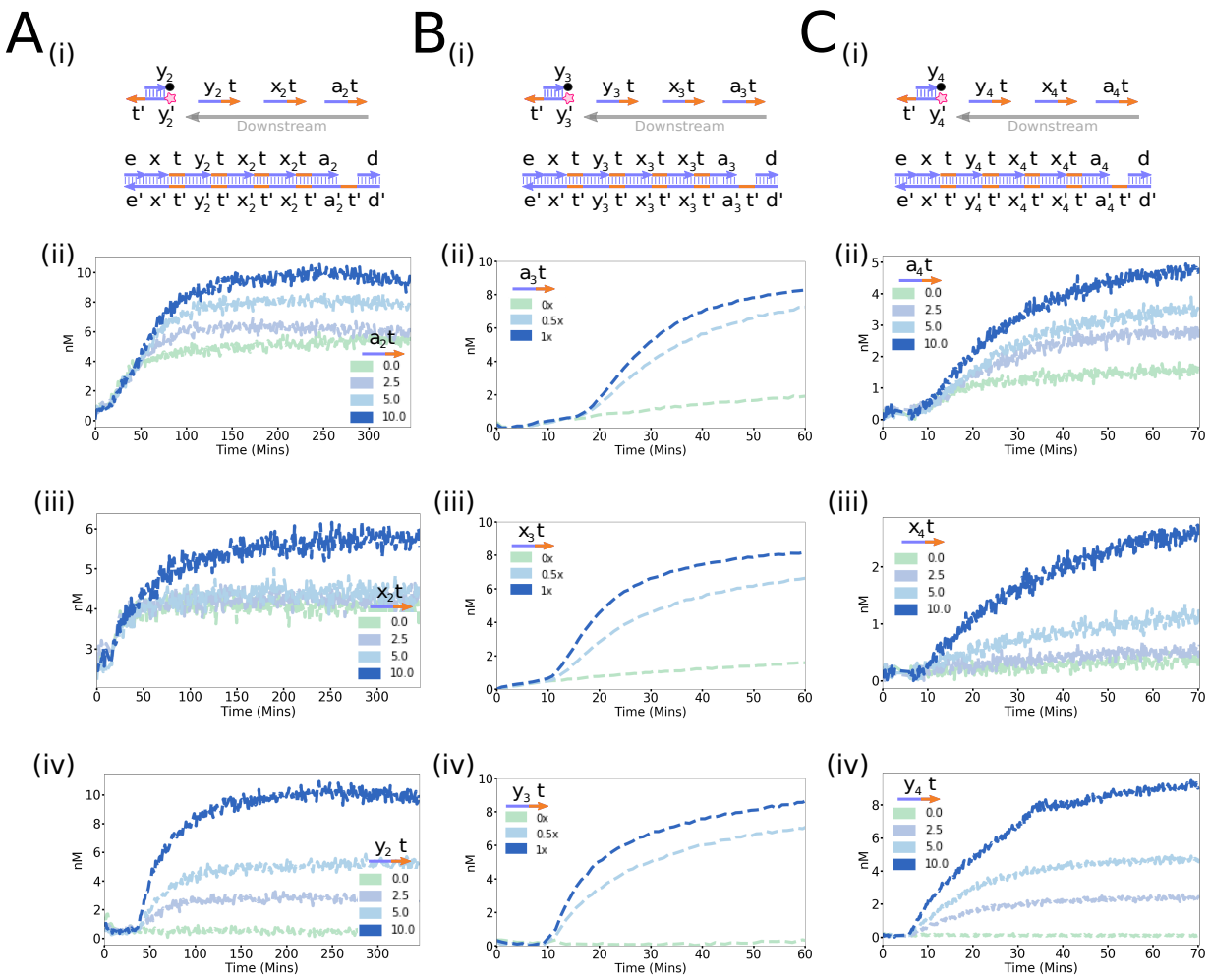

Figure 3.12. Fork Gate Kinetics for Oscillator Gates.
Panel A) shows ILN06, Panel B) shows ILN07 and Panel C) shows ILN08 toehold kinetics. Panels (i), (ii), and (iii) show the gate kinetics varying successive input strands, with the upstream input strands and helper strands at 10x and the downstream strands at 3x. These experiments are done with the gate at 3x and the reporter strands at 3x. Panel D) shows the gate architectures of the fork gates.

We do similar toehold tests with the fork gates that comprise the oscillatory reaction: ILN06_ForkX2X2Y2, ILN07_ForkX3X3Y3, and ILN08_ForkX4X4Y4 (see Section 3.5.2). The fork gates are much leakier than the join gates, although we do not show this in Figure 3.12 to give a better comparison to Figure 3.11. In Figure 3.12 Panels A and C, the gates and helper strands are added and mixed for a while before adding the trigger strands. The leak, which is likely largely caused by the helper strand which kicks off the input to the reporter complex,

makes it so panels (i) and (ii) for A and B do not act completely stoichiometrically, as the leak has taken over. In Figure 3.12 Panels B, the gates and helpers and triggers are added simultaneously. This also gives the illusion of the gates not being stoichiometric, because of the leak. For kinetic rate comparison, we can look at Figure 3.12 panels (iii), with the 10 nM input, where the leak affects the dynamics the least. It is seen that the half max is again reached in 5-10 minutes, which is again indicative of the gates having similar kinetics.

These join and fork are easily composed into larger reactions. In Figure 3.13, we see how join gate ILN09 and fork gate ILN10 can be combined together to form a catalytic reaction, X2 + X1 → X1 + Y1. In Figure 3.14, we see how join gate ILN03 and ILN08 can be combined together to form one of the reactions in the oscillatory network, X2 + X4 → X4 + X4 + Y4. Panel (i) of Figure 3.14 actually shows the average and the error bars of two different experiments, showing that the kinetics of these gates and reactions are extremely reproducible.

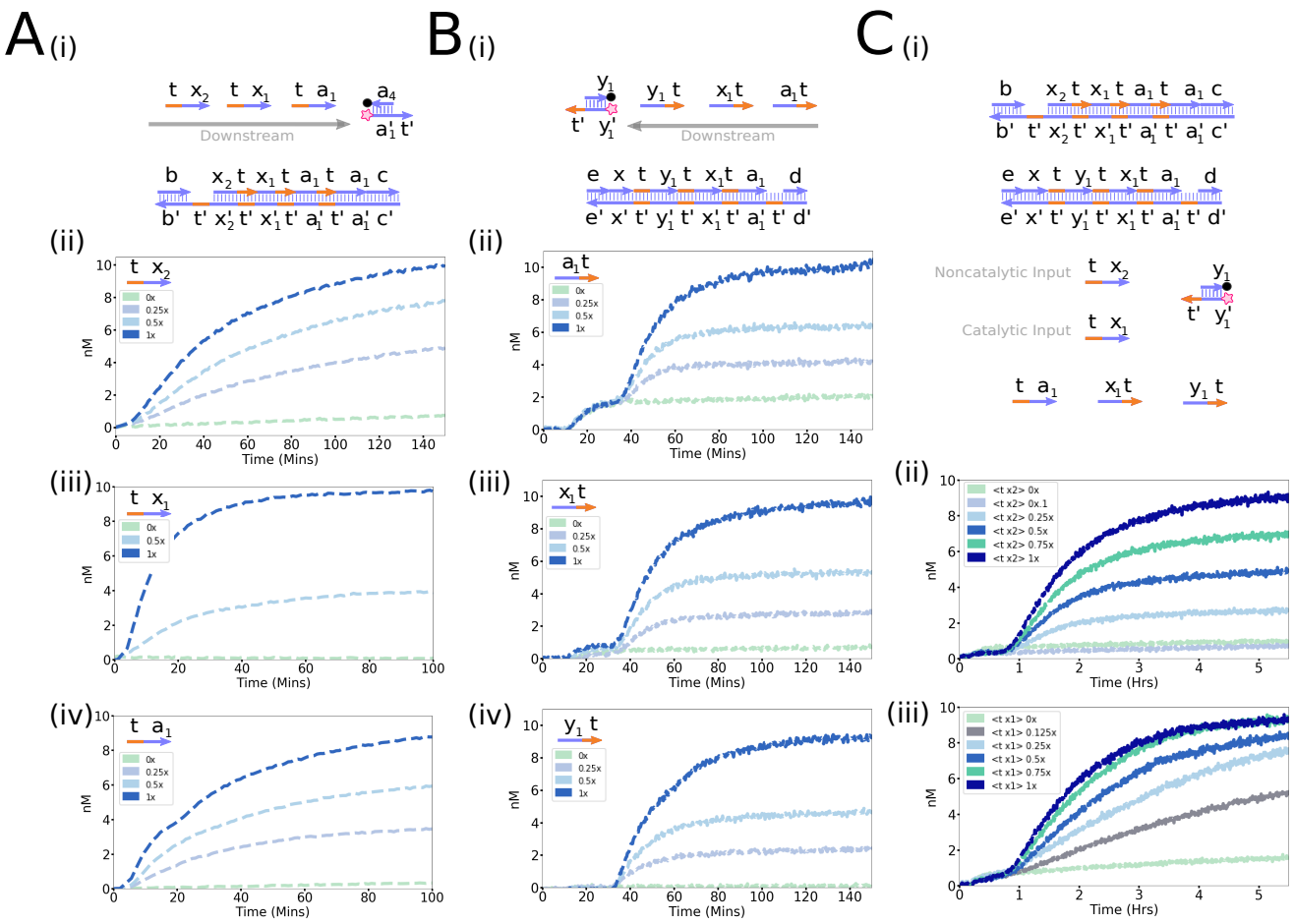

Figure 3.13. Example of a Full Catalytic Reaction X2 + X1 → X1 + Y1.
Panel A) shows the join gate ILN09. Panel B) shows the fork gate ILN10. Panels (i), (ii), and (iii) show the gate kinetics varying successive input strands, with the upstream input strands and helper strands at 10x and the downstream strands at 3x. These experiments are done with the gate at 3x and the reporter strands at 3x. The gates, helper strands, and reporter complex for the full reaction are shown in panel C). Panel C ii) and iii) shows the full reaction with gates at 1.5x and reporter gates and helper strands at 3x. In panel ii), the non-catalytic input is varied to show a stoichiometric output. In panel iii), the catalyst strand <t x1> is varied in this case to result in a catalytic output, meaning that all the non-catalytic input is eventually turned to product due to catalytic turnover.

The individual fork gate reactions (seen in Figure 3.13 and Figure 3.14 panels (ii)) reveal that the fork gates are much leakier than the join gates. In Figure 3.13 and Figure 3.14 panels (ii), the helper strands and reporters are added first, then, the gates are added, followed finally by the "trigger" input. It seems as though there is a blunt-end migration reaction occurring with some of the helper strands that creates a strong initial leak with the fork gates. This leak is likely

due to the enzymatic digestion process. This will certainly be an issue when composing larger networks later on, but it is useful information to know when attempting to model the reactions in VisualDSD.

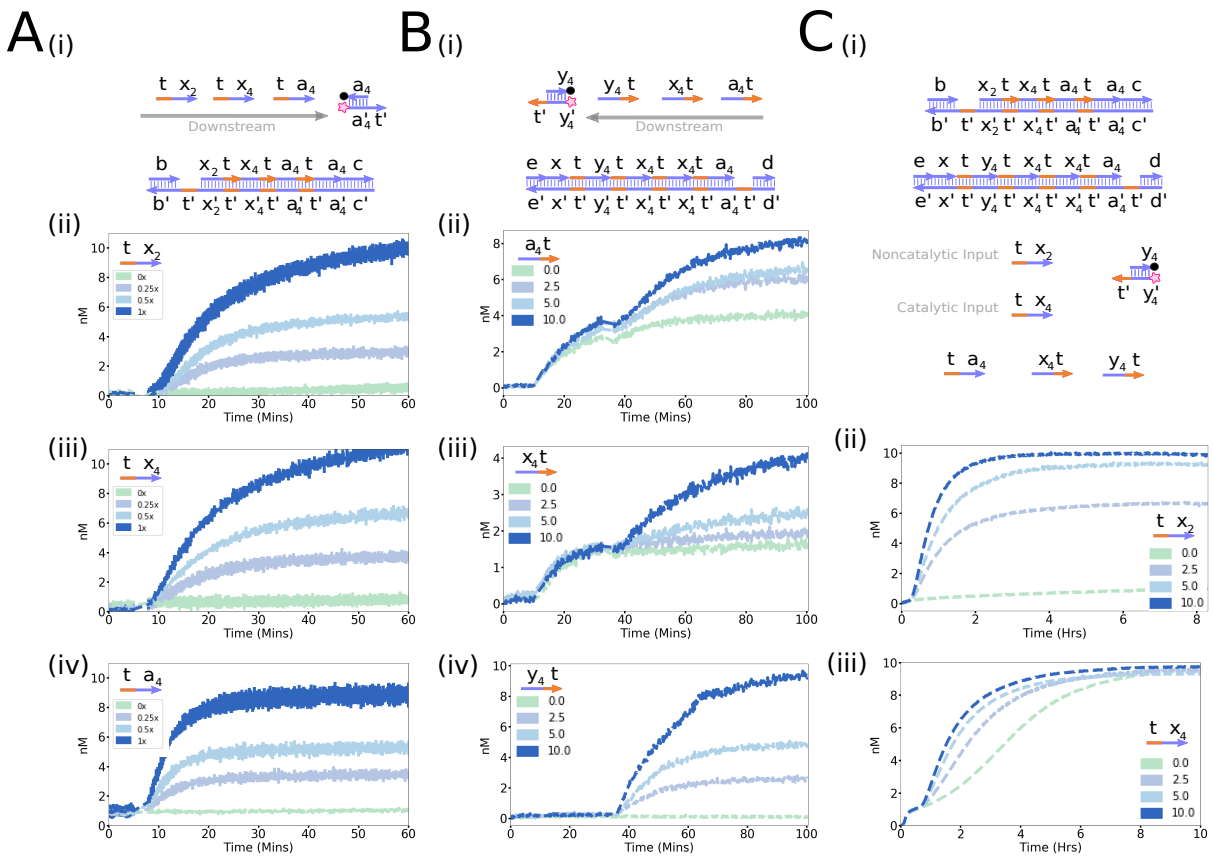

Figure 3.14. Example of a Full Autocatalytic Reaction.
Panel A) shows the average kinetics of two experiments done with join gate ILN03 toeholds. Panel B) shows the fork gate ILN08. Experiments are done by first adding only the helper strands and reporter gates, then adding the gates, then adding the varying input "trigger" strand last. Panels (i), (ii), and (iii) show the gate kinetics varying successive input strands, with the upstream input strands and helper strands at 10x and the downstream strands at 3x. These experiments are done with the gate at 3x and the reporter strands at 3x. The gates are shown in panel C). Panel D) shows the full reaction with gates at 1.5x and reporter gates and helper strands at 3x. The input strand <t x2> is varied in this case to result in a stoichiometric output. Panel E) shows the full reaction with gates at 1.5x and reporter gates and helper strands at 3x. The catalyst strand <t x4> is varied in this case to result in an autocatalytic output. The panel shows two different experiments, showing that there is some variation in the leak most likely due to gate processing.

3.3.3 *Reaction Types*

In fact, the context-independent two-domain architecture gates can easily implement any reaction type. As a proof of concept, we show that we can implement two-input chemical reactions of each major chemical reaction type: noncatalytic, catalytic, and autocatalytic. Additionally, we show a trimolecular reaction that produces three moles of autocatalyst per two moles autocatalyst input.

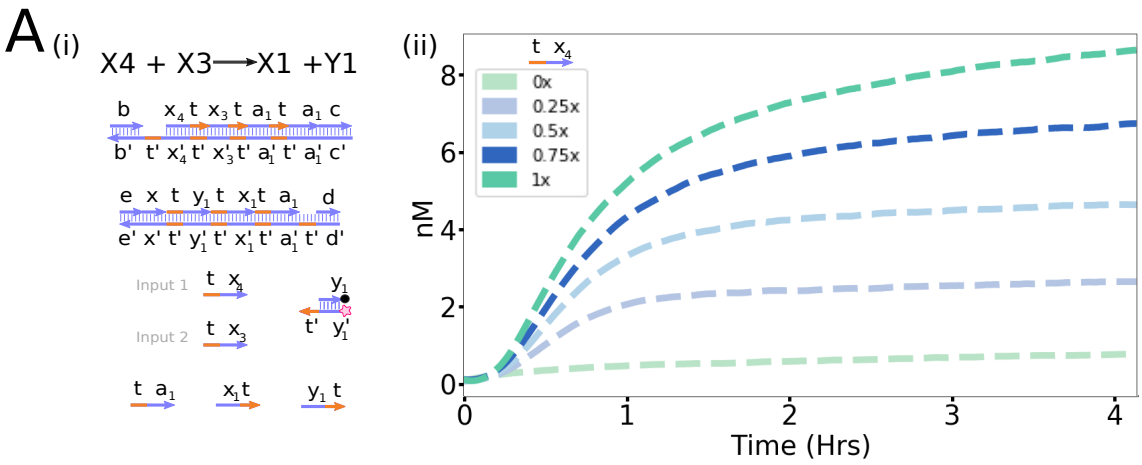

Figure 3.15. Noncatalytic Reaction.
Panel A) shows an example of a noncatalytic reaction using the context-independent gates. Panels (i) show the chemical reaction and the DNA implementation. Panels (ii) show the dynamics of noncatalytic reaction after adding varying amounts of the first input. The reaction is implemented with gates at 1.5x, reporters and helpers at 3x, input 2 at 1x, and input 1 at varying concentrations. Reactions are done in 1x TAE buffer with 12.5 mM Mg2+, 0.15% SDS, and 1 uM PolyT (1x = 10 nM).

Figure 3.15 shows an example of a noncatalytic reaction with two inputs and two outputs. Figure 3.15 B shows the dynamics of the noncatalytic reaction result in stoichiometric outputs, meaning that the output of the reaction is 1:1 with the concentration of the input 1 that is added. The signal is slightly lower, but that may be due to some strands getting sequestrated over the course of the reaction.

Examples of catalytic reactions can be made with the context-independent gates as well. Catalytic reactions are ubiquitous in natural and engineered control devices. Figure 3.16 shows two different catalytic reactions. The first is a two-input, two-output catalytic reaction. We also built a two-input, three-output catalytic reaction that has the same design as the other catalytic reaction, but has an extra output which is reported out as the signal. Both reactions show that all the input is turned over into product due to the regeneration of the catalyst. The three-output catalytic reaction is slightly slower than the two-output because it necessitates an extra strand displacement reaction to occur. Both reactions have a slight leak reaction rate, which can be seen in the trace that depicts the addition of 0x catalyst.

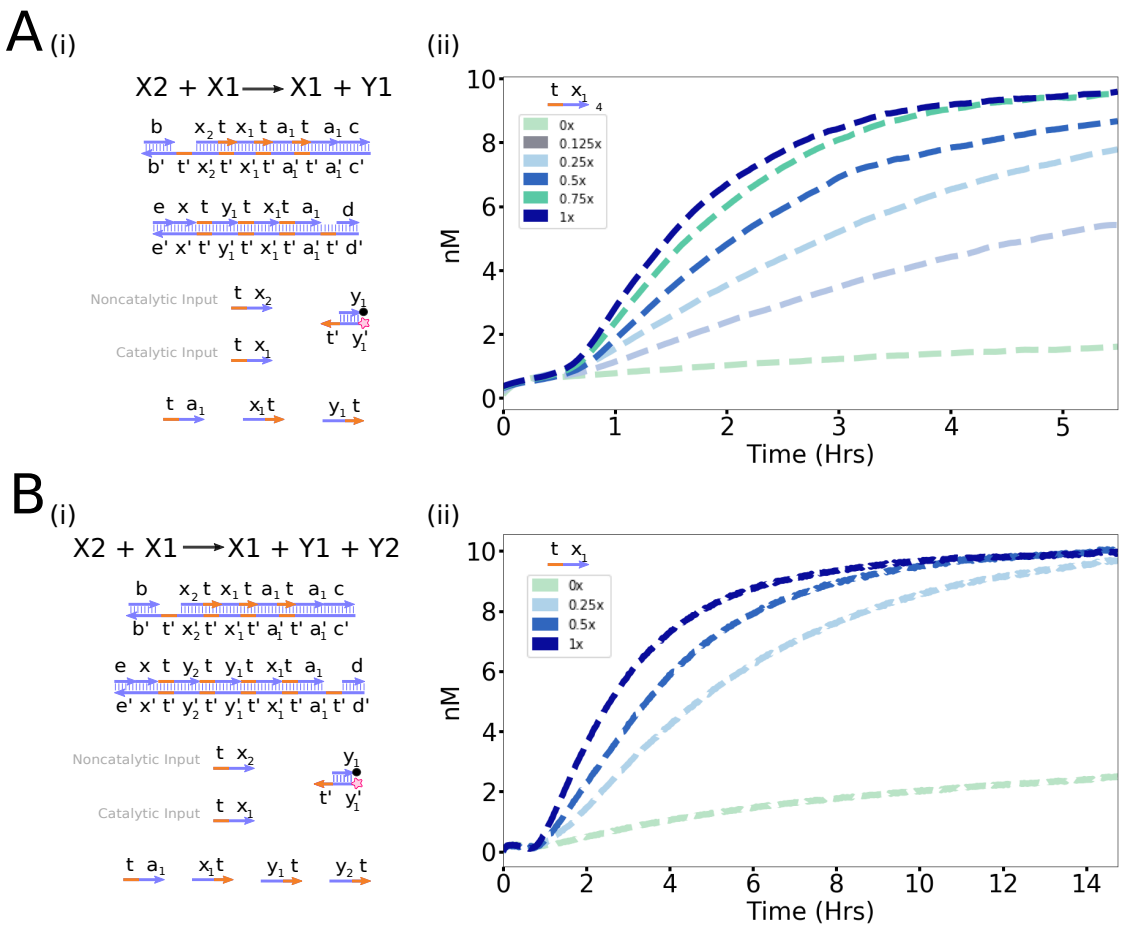

Figure 3.16. Catalytic Reactions.
Panels A) and B) show two different examples of catalytic reactions using the context-independent gates. Panels (i) show the chemical reaction and the DNA implementation. Panels (ii) show the dynamics of catalytic reactions after adding varying amounts of the autocatalyst. The reactions are implemented with gates at 1.5x, reporters and helpers at 3x, noncatalytic input at 1x, and the catalytic input at varying concentrations. Reactions are done in 1x TAE buffer with 12.5 mM Mg2+, 0.15% SDS, and 1 uM PolyT (1x = 10 nM) .

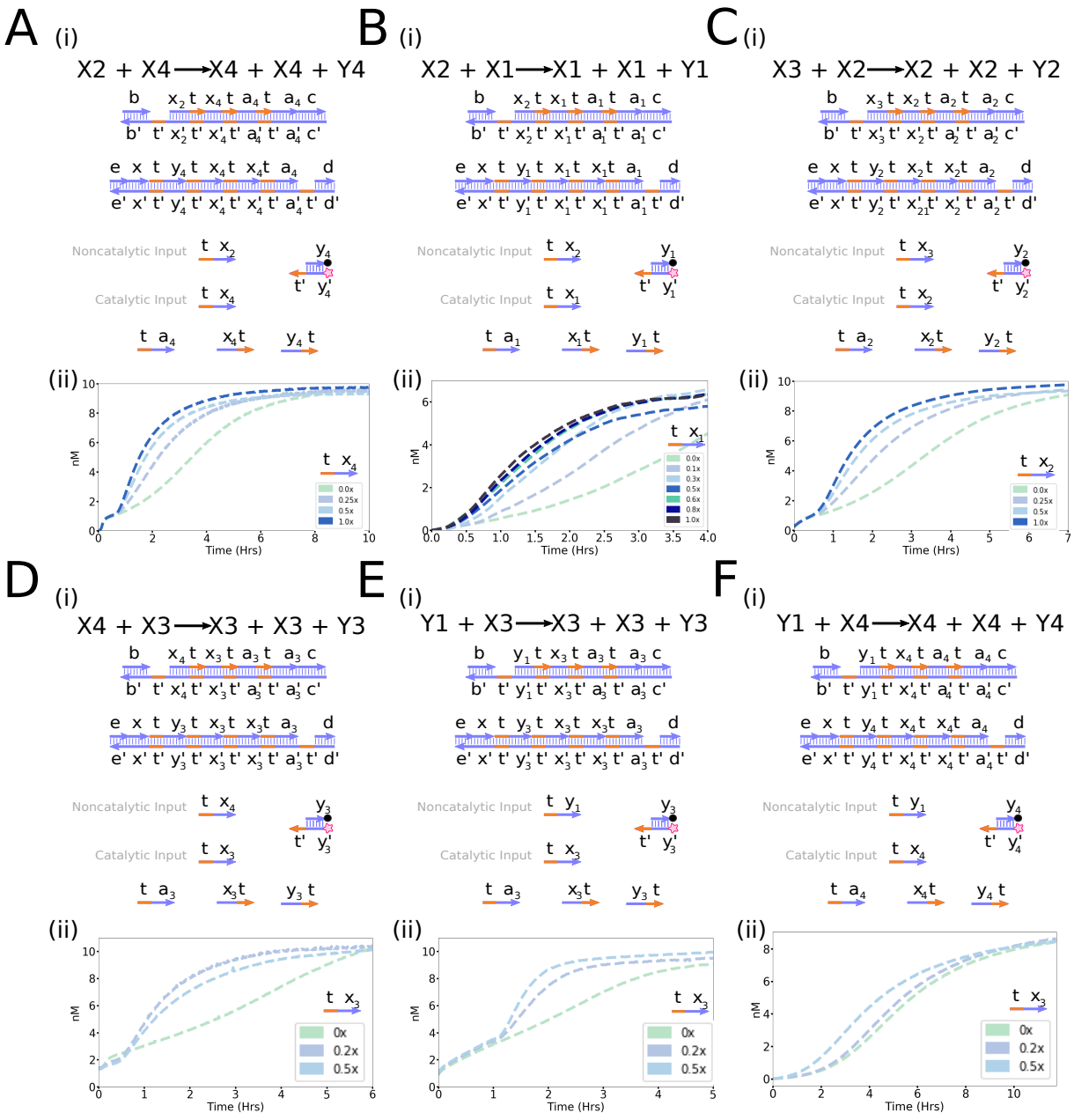

Figure 3.17. Autocatalytic Reactions.
Panels A-F) show six different examples of autocatalytic reactions using the context-independent gates. Panels (i) show the chemical reaction and the DNA implementation. Panels (ii) show the dynamics of autocatalytic reactions after adding varying amounts of the autocatalyst. All reactions occur with The reactions are implemented with gates at 1.5x, reporters and helpers at 3x, noncatalytic input at 1x, and the catalytic input at varying concentrations. Reactions are done in 1x TAE buffer with 12.5 mM Mg2+, 0.15% SDS, and 1 uM PolyT (1x = 10 nM) .

We also create several autocatalytic reactions. In autocatalytic reactions, a small amount of catalyst results in exponential release of the output, resulting in a sigmoidal kinetic curve characteristic of autocatalytic reactions. Interestingly, all of these reactions have similar half lives. It takes around an hour to achieve the half max with 10 nM of catalyst. This means that it should be possible to implement larger networks with these context-independent gates in a one-pot style reaction, without the necessity for additional characterization.

These autocatalytic reactions are each individual components of either a consensus or oscillatory reaction. Both of these networks should work as long as the reactions have the same reaction rates. Additionally, autocatalytic reactions have been shown to be essential for applications such as generating complex spatial patterns, where nonlinear dynamics are necessary.

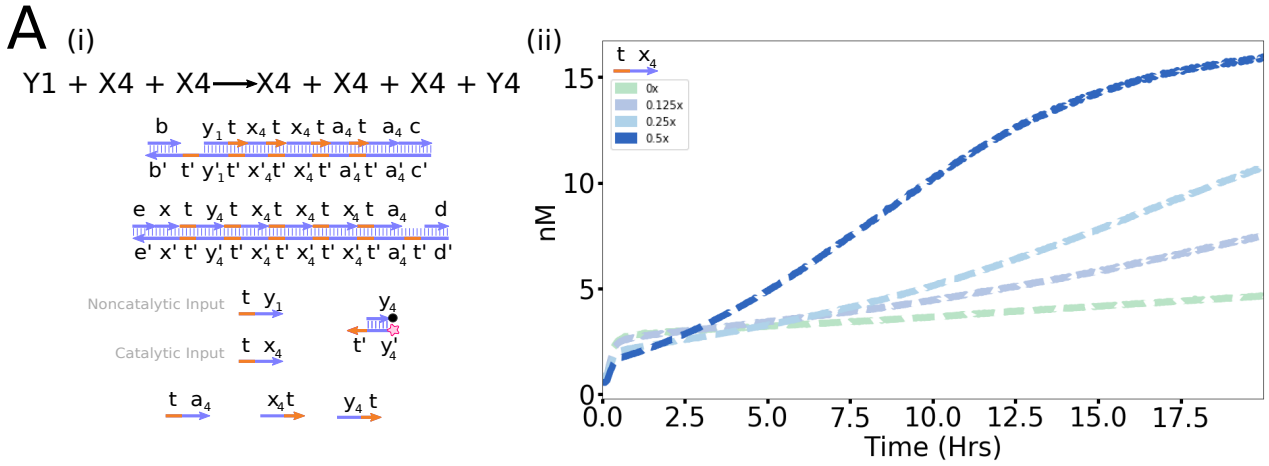

Figure 3.18. Example of a Trimolecular Autocatalytic Reaction.
Panel A) shows an example of a autocatalytic reaction with three inputs using the context-independent gates. Panels (i) show the chemical reaction and the DNA implementation. Panels (ii) show the dynamics of trimolecular autocatalytic reaction after adding varying amounts of the first input. The reaction is implemented with gates at 1.5x, reporters and helpers at 3x, the noncatalytic input is added at 1x, and the catalytic input is added at varying concentrations. Reactions are done in 1x TAE buffer with 12.5 mM Mg2+, 0.15% SDS, and 1 uM PolyT (1x = 10 nM).

Interestingly, the nicked-double stranded DNA gate structure makes it possible for us to make any chemical reaction that we're interested in. We implemented a trimolecular reaction, which would be difficult, if not impossible, to find in nature. Since the three-input, four-output reaction necessitates two to three more reactions than the reactions we've implemented previous to this point, the trimolecular reaction is actually much slower than the other reaction types.

3.4 Models for Context-Independent Gates

3.4.1 *Modeling Individual Gate Kinetics*

Several two-input join gates were tested using experimentally and modeled using the schematic outlined in section 3.4.3. In experiments, the first input to the gates was added in varying concentrations, while adding the other inputs and the auxiliary strands in excess. We inferred several parameters, most importantly the kinetic rate (*kt*), which is the universal toehold reaction rate which can accurately describe the dynamics of the gate. Fork gates were also tested in a similar manner and inferred reaction parameters resulted in high-quality model fits for all the gates that were tested.

Figure 3.19 shows the model-data dynamics of individual gates and also the universal toehold kinetic rates. These show that indeed a single, universal reaction rate was sufficient to model the reaction kinetics of each gate. Noticeably, the kinetic rates (*kt*) were very similar between different gates, regardless of whether they were Join or Fork gates, suggesting that indeed the careful design of toehold sequence and context can result in predictable kinetics. Other parameters used to model the individual gates include the time trigger was added (T1), a leak reaction rate, and a leak and bad term. The posterior distribution of these parameters can be seen in Figure 3.20 and Figure 3.21.

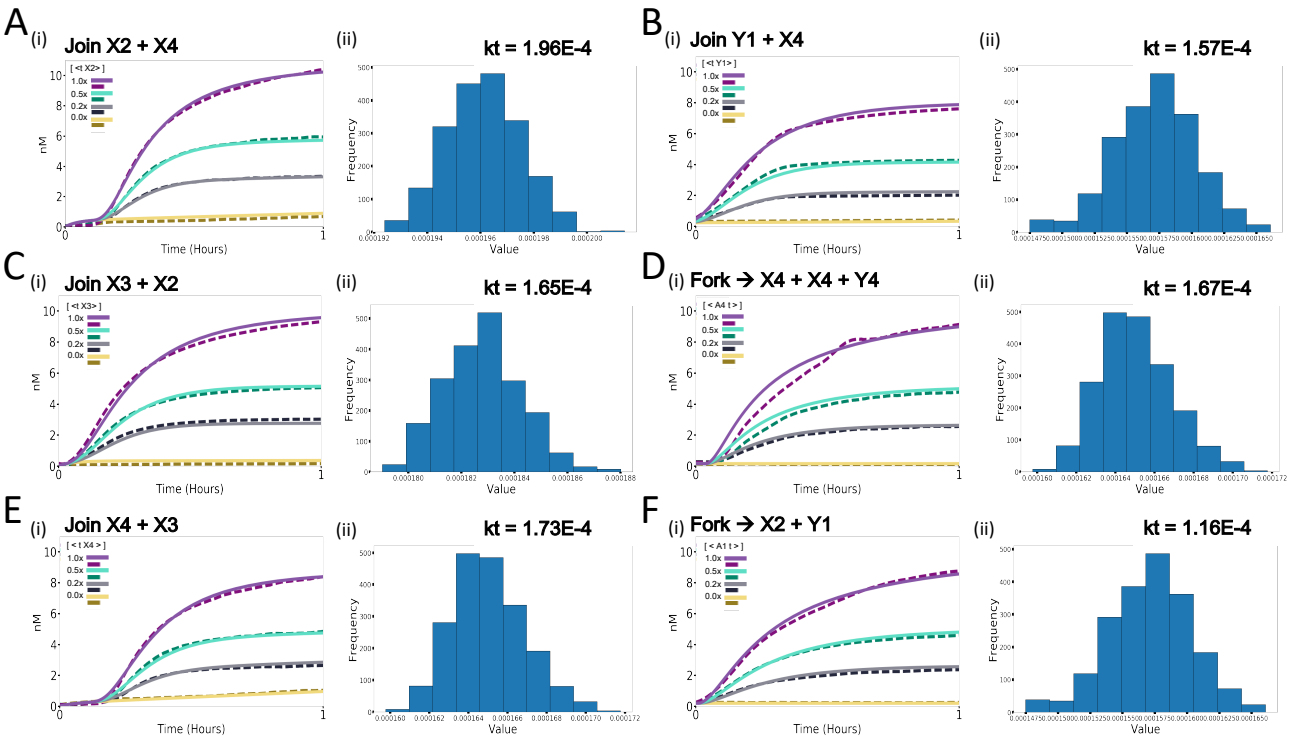

Figure 3.19. Individual Gates-Model Dynamics.
Individual gates were tested by adding a specific input to the gate in varying amounts while adding the other strands in excess. Join gates were tested by varying the first input to the gate (0x, 0.25x, 0.5x, 1x) while the other input strands were added at 3x. Fork gates were tested by adding the last input strand in varying amounts (0x, 0.25x, 0.5x, 1x) while the other strands were added at 10x. Gates were at 1.5x and the reactions were run in TAE buffer with 12.5 mM Mg2+. Panels (i) show the data as a dashed line while the mechanistic model fit is shown as a solid line. The posterior distribution of the fitted parameter kt (the strand displacement rate) is shown in panels (ii).

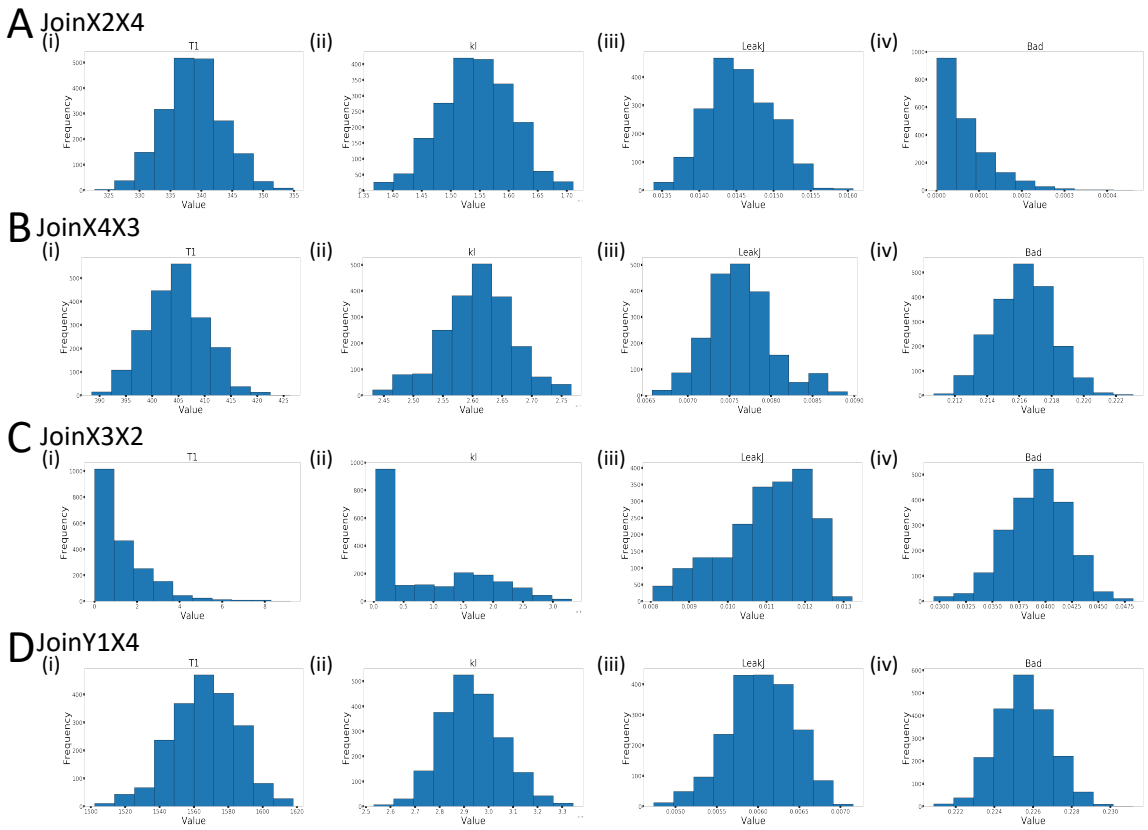

Figure 3.20. Posterior Distributions of the Inferred Join Gate Parameters.
Panels (i) shows the time of the input addition. Panels (ii) show the rate of a blunt end leak reaction. Panels (iii) show the distribution of the leak. Panels (iv) show the distribution of the bad parameter.

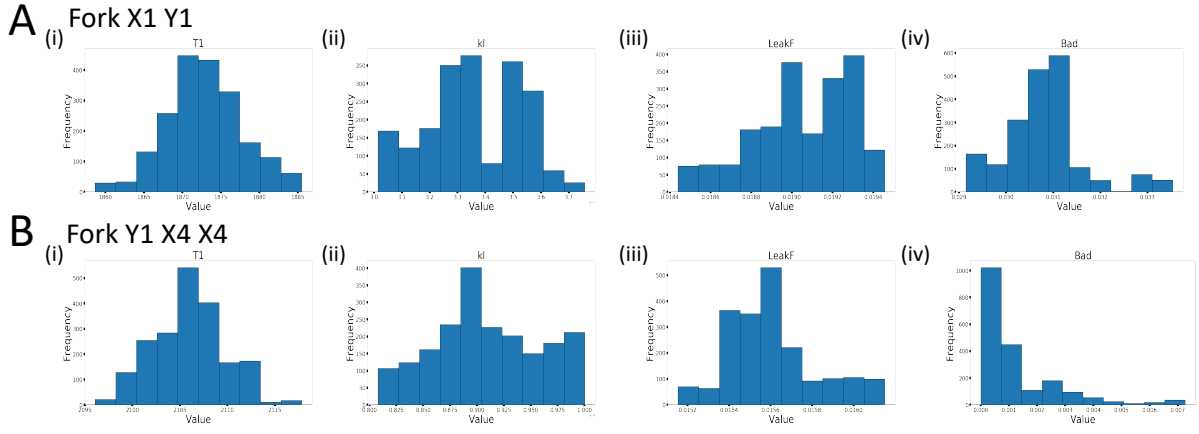

Figure 3.21. Posterior Distributions of the Inferred Fork Gate Parameters.
Panels (i) shows the time of the input addition. Panels (ii) show the rate of a blunt end leak reaction. Panels (iii) show the distribution of the leak. Panels (iv) show the distribution of the bad parameter.

3.4.2 *Modeling Fundamental Reaction Types*

As we've seen, the *context-independent* gate design allows for easy composition of different reaction with multiple input and multiple outputs. We tested and modeled reactions of the three major reaction classes; non-catalytic, catalytic, and autocatalytic. These all the reaction types necessary for composition of complex chemical reaction networks. We analyzed these reactions to see if we indeed get consistent kinetics and can use a single toehold reaction rate to accurately model the reactions.

The non-catalytic reaction X4 + X3 → X1 + Y1 generates products in a stoichiometric fashion (e.g the correct amount of product is produced for the amount of input that is consumed). The catalytic reaction X2 + X1 → X1 + Y1 was implemented as well. In this reaction type, even a small amount of catalytic input results in the full conversion of the reactants to products. Lastly, the autocatalytic reaction X2 + X4 → X4 + X4 + Y4 was tested with various amounts of the catalyst as input, showing sigmoidal dynamics.

These fundamental reaction types were parameterized with our inference model. The parameter inference strategy is similar to that which is done with the individual gates but a *leak* parameter is inferred for each gate separately (see parameters in Figure 3.23). Parameter inference for the full reactions showed that high quality fits could be achieved with a single, universal reaction rate *k* modeling the forward reaction rate for each bimolecular reaction within the strand displacement reaction cascade (see Figure 3.22). The kinetic rates of the full reactions, are similar to one another, and are also highly similar to the reaction rates fitted for the individual fates, suggesting that these gates are indeed highly composable.

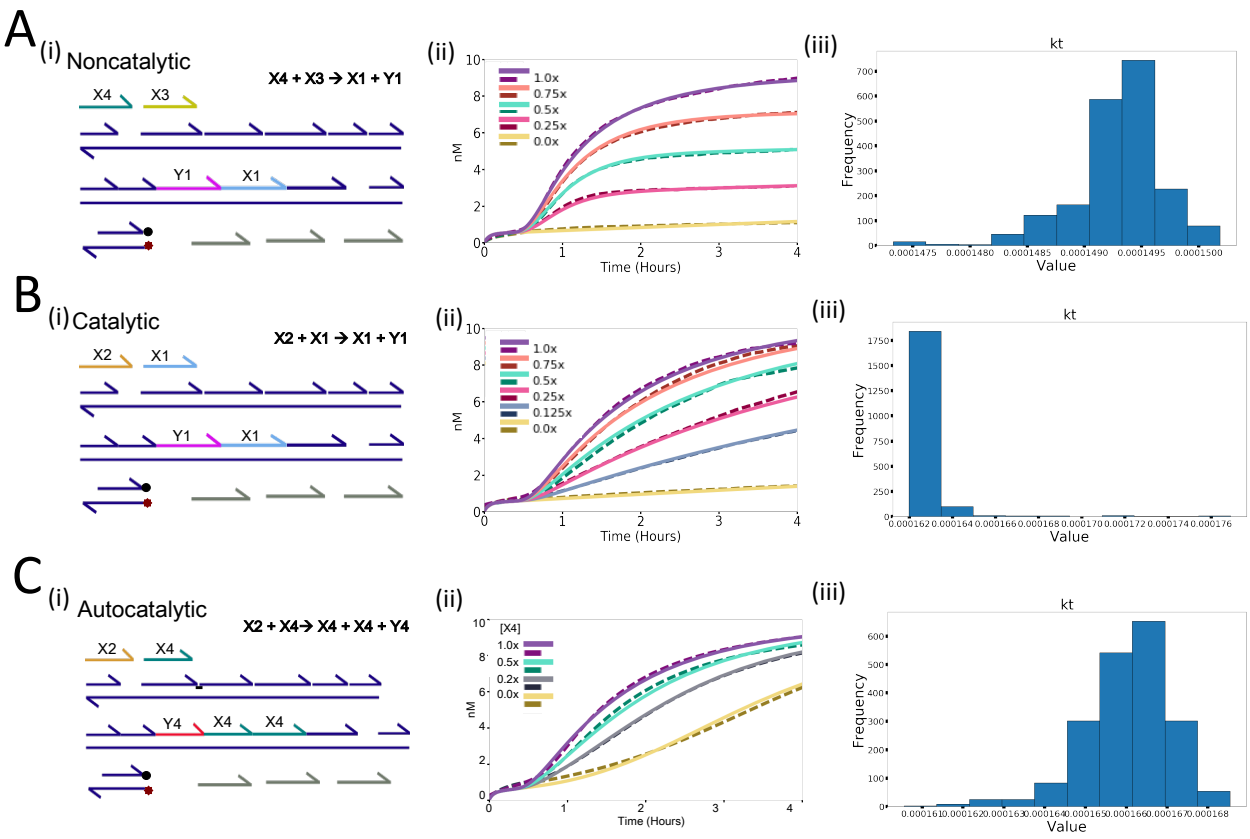

Figure 3.22. Model Data Dynamics of the Three Major Reaction Classes.
These major reaction classes can be realized with next-generation gate design and modeled using a universal toehold reaction rate. Panels i) indicate the chemical reaction being emulated and give the DNA strand level design of the reaction, with a simplified representation of the ndsDNA gates, input strands, and auxiliary strands used for the corresponding experiments. Panels iii) show the reaction kinetics with the model fits. Dashed lines indicate the experimental data and the solid lines are the model fit with the inferred parameters. All join and fork gates were at 1.5x and auxiliary strands were at 3x, 1x = 10 nM. A) Non-catalytic bimolecular reaction X4 + X3 → X1 + Y1. Signal X3 was at 2x, while varying stoichiometrically limiting concentrations of input X4 were added. B) Bimolecular catalytic reaction X2 + X1 → X1 + Y1. Signal X2 was at 1x, while varying amounts of the catalytic signal X1 were added to the system. C) Autocatalytic reaction X2 + X4 → X4 + X4 + Y4. Signal X2 was at 1x, while varying amounts of the autocatalyst X4 were added to the reaction.

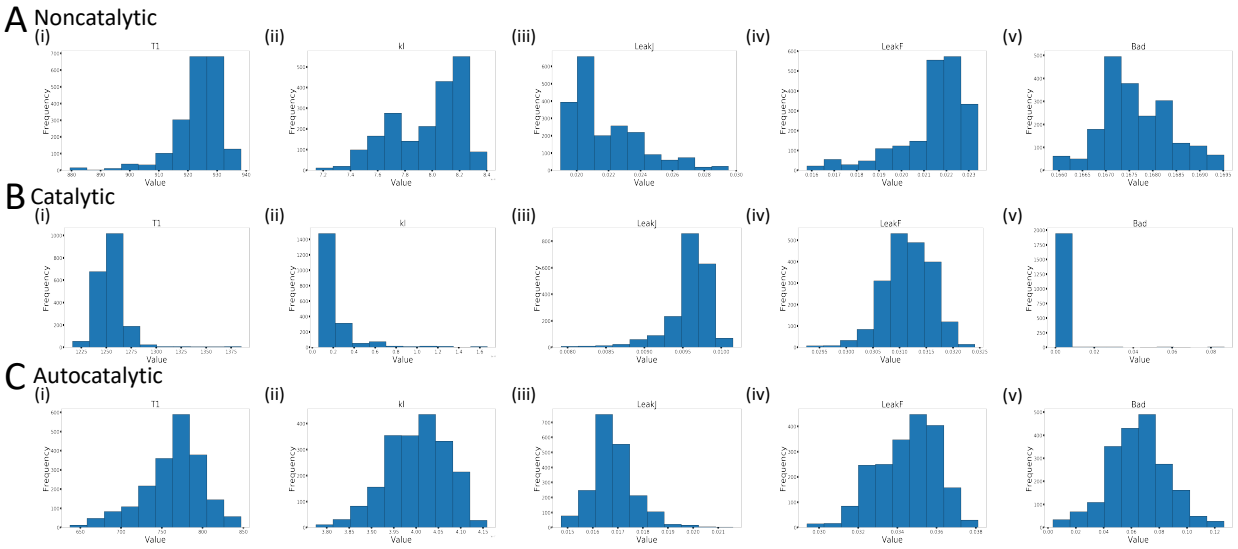

Figure 3.23. Posterior Distributions of the Inferred Parameters for the Various Reaction Types. Panels (i) shows the time of the input addition. Panels (ii) show the rate of a blunt end leak reaction. Panels (iii) show the distribution of the join gates leak parameter. Panels (iv) show the distribution of the fork gates leak parameter Panels (v) show the distribution of the bad parameter.

3.4.3 *Mechanistic Strand Displacement Level Model*

The mechanistic strand displacement-level model treats each individual strand displacement step as a bimolecular reaction between a signal or helper strand and the gate complex with the complementary matching open toehold. We use the VisualDSD [59] inference module to infer the dynamics of our system on this mechanistic level. Several parameters for model fits were inferred using VisualDSD, including the kinetic rates of the individual strand displacement steps. Previous models determined strand displacement rate constants with values ranging from 1x10^4 /M/s to 1.44 x 10^6 /M/s [6]. With *context-independent gates* we found that the mechanistic strand displacement-level model fit the data exceptionally well with only a single reaction rate constant for each strand displacement steps. The average kinetic rates of the context-independent plasmid-derived gates measured on the plate reader is 1.61 x 10^5 /M/s with a relative standard deviation of 13%, suggesting that careful design of toehold sequence and context do indeed

result in consistent kinetics amongst gates with different input and output signals. This is an important step in ensuring the scalability of this method of building arbitrary CRNs, as individual gates and reactions don't need to be fit individually and can be composed with predictable reaction rates.

Our model fits the data exceptionally well, even capturing leak reactions. We inferred several additional parameters to represent non-ideal gate behavior. *Leak* parameters were inferred to correspond to a fraction of gates that have spontaneously produced an output even in the absence of a trigger. *Bad* parameters were inferred which model the fraction of non-functional input strands that resulted in experimental measurements indicating a net loss of input strands. A third parameter was used to explain a slow leak that sometimes occurs. This type of leakage is most likely due to blunt-end displacement which may occur when bases adjacent to a nicked site spontaneously breathe and allow for the untimely invasion of a helper strand. This leak was modeled as a kinetic reaction in which blunt-end migration results in triggering of a translator or output strand which can then react with a reporter complex. The kinetic rate for this reaction (*kl*) was then inferred.

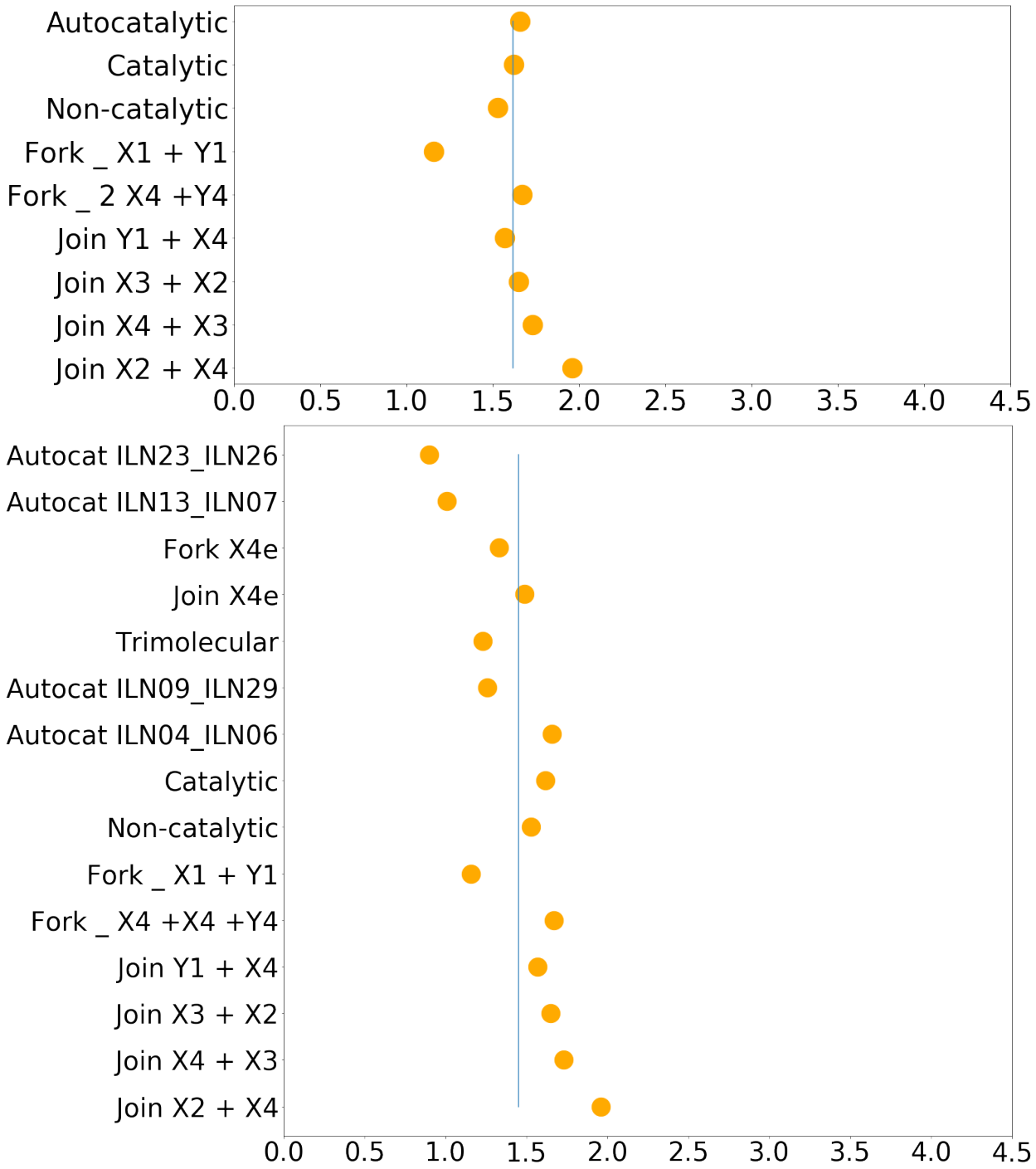

Figure 3.24. Strand Displacement Rate Constant Comparisons
Independently inferred for different rates and reactions are very similar (units are E5 /M/s), with an average kinetic rate of 1.61E5 /M/s (shown as the blue line) and a relative standard deviation of 13%, when all gates are measured on the plate reader (shown in the above panel). The bottom panel shows a mix of reactions, some of which were also measured on the spectrofluorometer. Even with the noisier data, the average kinetic rate 1.45E5 /M/s and a relative standard deviation of 28%, still much improved over previous designs.

3.4.4 *Modeling Multiple Gates and Reactions Simultaneously*

In theory, if the toehold kinetic rates are the same for all the universal toeholds, then we should be able to model multiple gates and full reactions simultaneously. In collaboration with Neil Dalchau and Andrew Phillips at Microsoft Research, we have been able to verify that multiple gates and multiple reactions can be modeled simultaneously, showing that indeed these context-independent gates standardize these chemical reaction network modules. Indeed, each toehold can be described by the same rate and additionally that rate is comparable to the fitted rate we got from fitting reactions individually. Figure 3.25 shows the results of modeling several experiments with the same gate simultaneously.

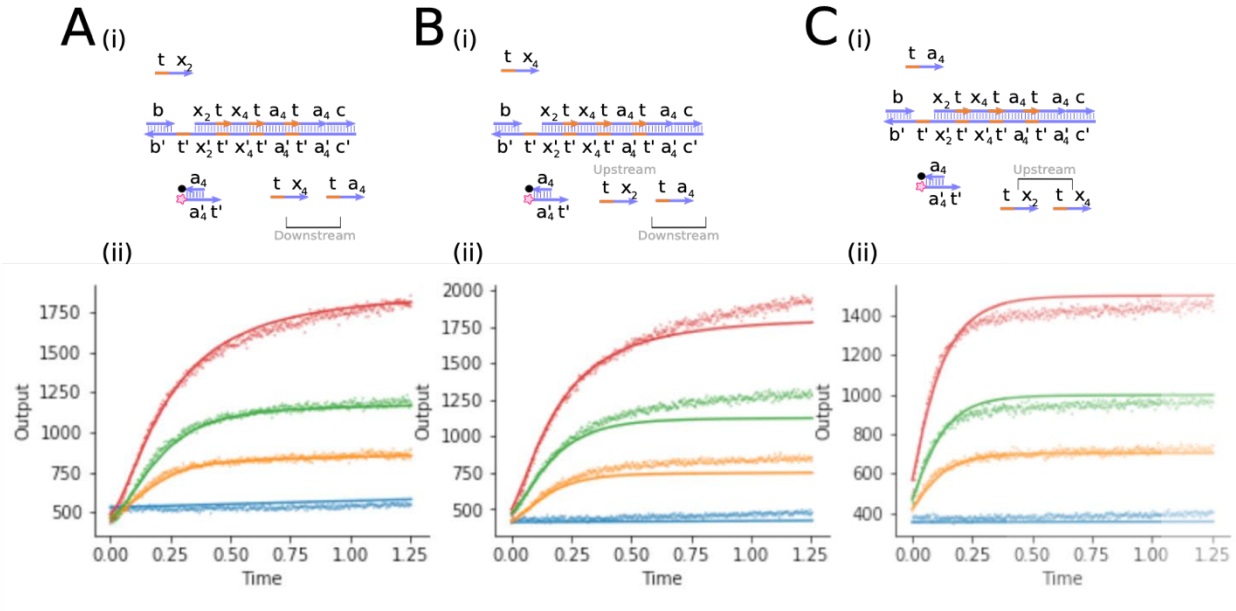

Figure 3.25. Modeling Multiple Gate Toehold Experiments Simultaneously.
Shown are the experimental observations for datasets that characterize strand displacement reactions on the ILN03 gate. The experiments directly observe the forward reactions (from left to right on the gate A) <t X2> displacing <x2 t> B) <t x4> displacing <x4 t> C) <t a4> displacing <a4 t> In Panels A-C, 1x = 10 nM, and we start with 3x gates and reporters. All gates are pre-mixed with 10x of strands upstream and 3x of strands downstream of the target interaction. Panels ii) show the experimental data (dashed) with the fits on top (solid). These fits were all done simultaneously, showing that the single toehold parameter describes all the toeholds well.

Figures 3.26 – 3.28 were all modeled simultaneously, using a single parameter for inferring the toehold kinetic rate. We can see that these fits fit the reactions well, except for one of the autocatalytic reactions. We hypothesize that the reason for the single bad fit is that the fork gates were particularly leaky in that experiment due to over-digestion of the gates. We may need a parameter to describe this leak in future, similar to the way we fit individual reactions. This result, in combination with our ability to build larger circuits (see below) is strong evidence that we can indeed use a universal toehold rate to describe our gates.

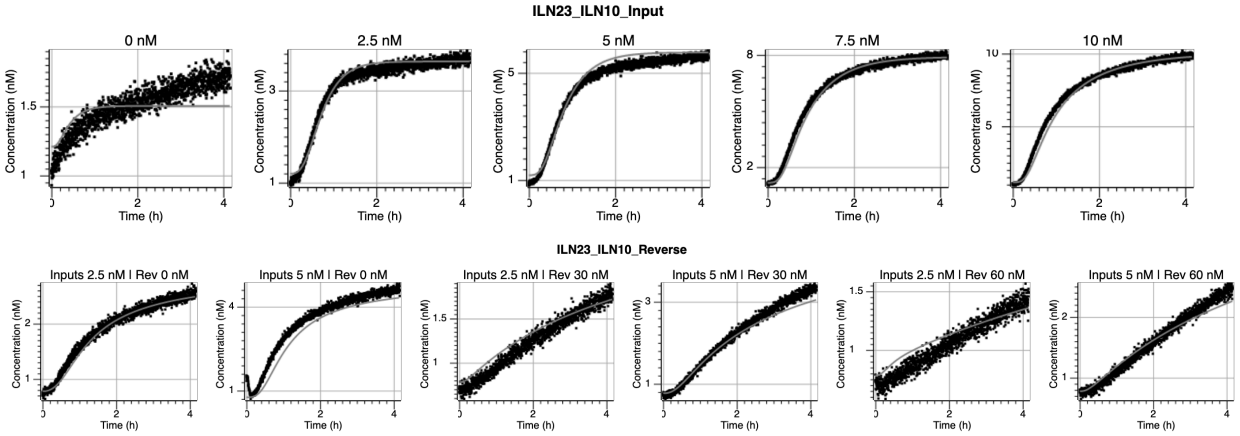

Figure 3.26. Noncatalytic Reaction Modeled Simultaneously with Other Full Reactions

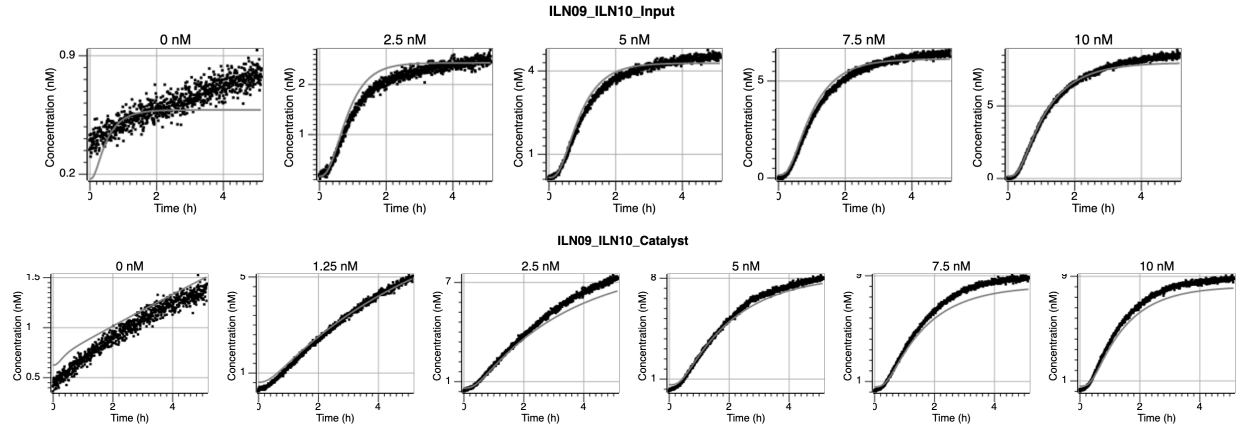

Figure 3.27. Catalytic Reaction Modeled Simultaneously with Other Full Reactions

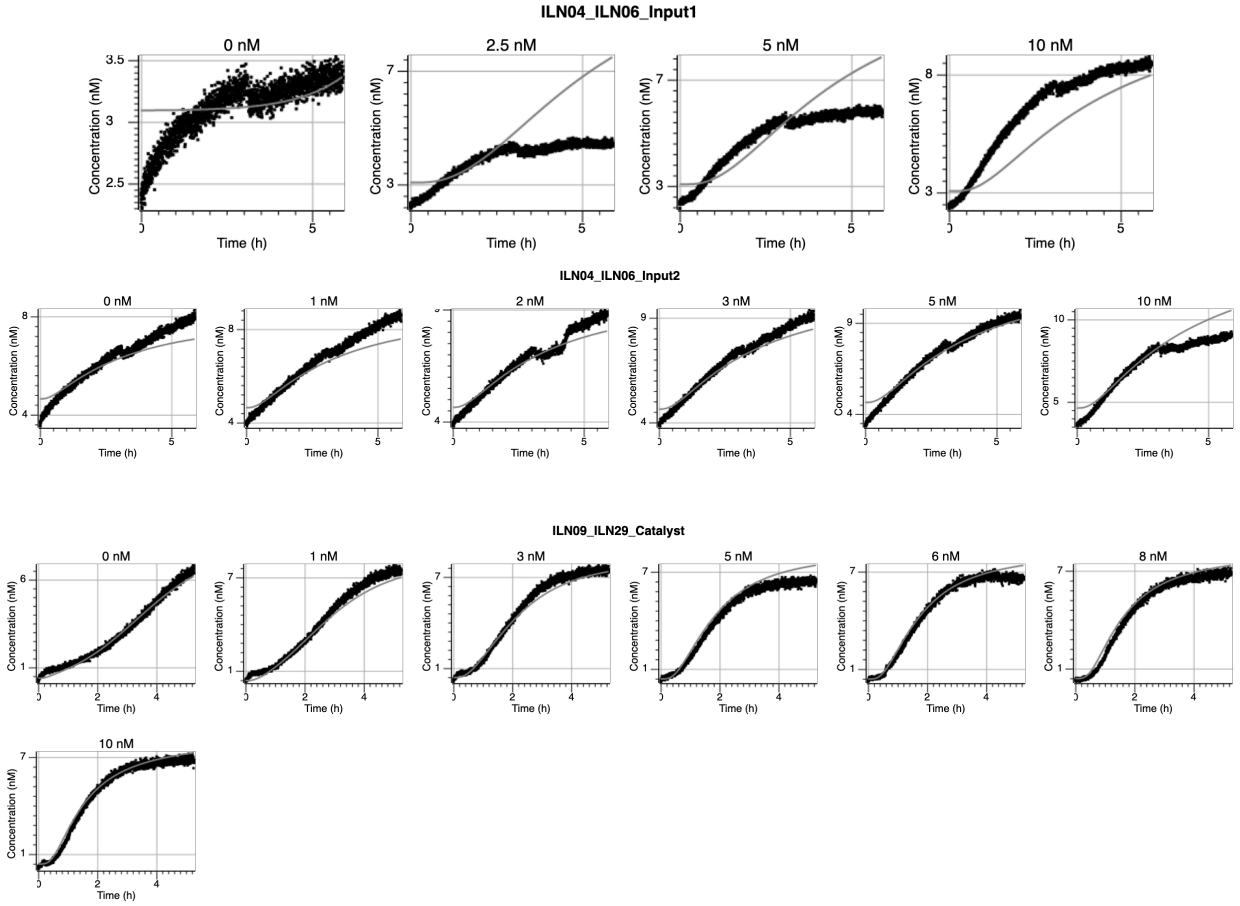

Figure 3.28. Autocatalytic Reactions Modeled Simultaneously with Other Full Reactions. Dotted lines are experimental data and solid lines are experimental fits.

3.4.5 *Changing Toehold Sequence Controls Kinetics*

When designing circuits, we might be interested in changing the toehold reaction rate, not just controlling it to be one speed. To that purpose, we designed several toeholds with varying ΔG to test whether we can control the speed of the toehold kinetics. We kept the design of these toeholds palindromic, so that these toeholds fit into the criteria for context-independent gates that we have outlined. We ordered synthetic DNA to make several one input "join" gates with the various toeholds, keeping all sequences the same, except for the varying toeholds.

We then tested the gates, by triggering them with their corresponding output, at low concentrations (1x = 1 nM). We analyzed the time it took for the signal to reach the 0.3 nM mark

and plotted the log time (seconds) versus the ΔG. Interestingly, the relationship follows a linear trajectory until it levels off at around a ΔG of -10. This is similar to other values that have been reported in literature [12]. Additionally, we took the average toehold kinetic rate, as reported above and modeled the time it would take to reach the 0.3 nM level. We then plotted this as well (see Figure 3.29 C) and see that it matches the trajectory of the other toehold rates.

Ultimately, we see that the design of the toehold that we've chosen is one of the fastest that can be achieved. However, it is possible to engineer other kinetic rates by slowing the toehold rates down.

Table 3.1. Toehold Sequences

Toehold Name	**Toehold Sequence**	**ΔG**
Universal	CACCAC	-10.29
T2	ACCCAC	-10.24
T3	ACTTCA	-8.81
T5	ACACA	-7.88
T8	ACCA	-6.53

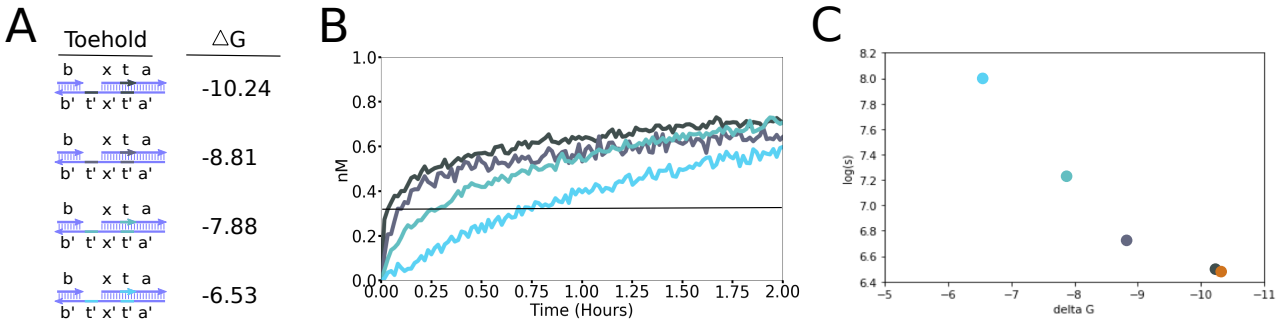

Figure 3.29. Varying Toeholds in Context-Independent Gates.
A) Four different designs of toeholds with different ΔG. B) Kinetics of the different gates at a low concentration (1x = 1 nM). C) Plot of log time it takes for each gate to turnover 0.3 nM of gate into signal versus ΔG. Orange dot is the simulated time it takes to reach 0.3 nM from the CACCAC toehold, used universally by us in other designs.

3.4.6 *Consensus Network*

One molecular computation that can be done using CRN architecture is the consensus network, or approximate majority network. Here the network is able to "sense" the majority signal and convert all of the minority signal to that majority. The network operates on two signal species (X and Y) and works via a system of two autocatalytic reactions and one non-catalytic reaction. The minority and majority species cancel each other, producing a buffer species B. Meanwhile, if the buffer species interacts with either signal, an autocatalytic reaction produces more of that particular signal.

An analogy to explain this network is voting. Imagine there are some people that support either party X or party Y. If a person supporting X and a person supporting party Y bump into each other, they neutralize each other's opinions and become neutral (species B). If a neutral person B bumps into an opinionated person, they become part of that party as well. In this manner, everybody eventually gets converted to the party that was in the majority initially.

While previous implementations of this network required clever determinations of concentrations and gates to use to make the reactions act as desired, our context-independent system requires no such finagling. For this network to be able to sense, small differences between the initial concentrations of minority and majority species, the autocatalytic reactions must happen at the same rate, something that our context independent gates can do.

In an experiment, all reactions and helper strands are added in equal ratios. We see that the network performs as desired, successfully converting the minority species to the majority. Additionally, the predictive model based on the previously inferred universal strand displacement reaction rate, correctly predicts the behavior of the network.

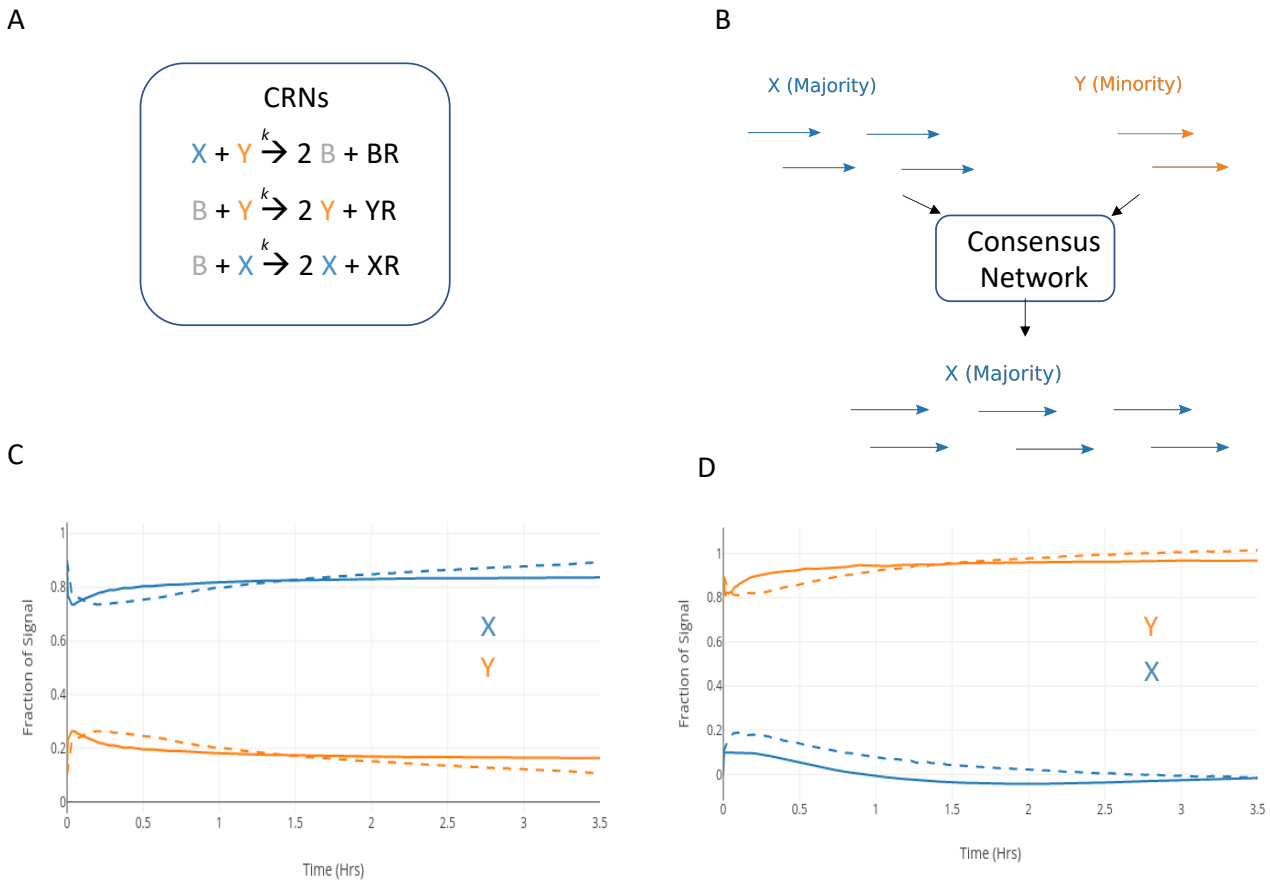

Figure 3.24. Consensus Network.
Panel A) The CRN of the consensus network is shown here. Panel B) depicts the idea is that the network should convert all the species into the majority species. Panel C) shows the result of a consensus network with 9:1 ratio of X in the majority, being converted to all X. Panel D) shows the result of a consensus network with 9:1 ratio of Y in the majority, being converted to all Y (1x = 10 nM). In both panels C and D, the dynamics of the behavior (dashed lines) match the predictive model (solid lines) fairly well.

3.5 OSCILLATORS

The powerful language of chemical reaction networks should in principle be able to realize complex and autonomous dynamical behaviors. One such behavior, that of oscillations, can be implemented with chemical reaction networks. Here, we explore whether it is possible to build these oscillators with context-independent DNA strand displacement components alone.

3.5.1 *Lotka-Volterra Oscillator*

A classical network of chemical reactions that should produce oscillatory behavior is the Lotka-Volterra Oscillator, also known as the predator-prey reaction. The idea is that there is a pray species, say a rabbit (species Y) that can multiply as long as there is food (A) present in the environment. The fox (species X) multiplies when they can eat rabbits. Foxes eventually die out. This series of reactions produces an oscillatory behavior, with the rabbit population initially increasing, followed by an increase of the fox population as they experience an abundance of food. The increased population of foxes eats too many rabbits, so the rabbit population declines, which results in a fox population decline when they don't have enough to eat. Once the fox population declines, the rabbit population is free to increase again and the oscillatory behavior continues.

Lotka-Volterra Reaction Network:

$A + Y \rightarrow 2Y$

$X + Y \rightarrow 2Y$

$X \rightarrow 0$

Here, we show a Mathematica model of the Lotka-Volterra system implemented with our ILN gate design. Each reaction has a dummy output which can be reported when doing an actual experiment. The model is a mass action kinetics model that uses ODEs to model each strand displacement step.

The *in silica* model is shown in Figure 3.30. Oscillations are observed when we can significantly slow down the reaction where rabbits are produced (modeled as a slowdown in the first toehold or strand displacement step). One way to implement this slowdown experimentally might be to change the reaction rate for the toehold where the reaction initiates. It may be

possible to do this by covering a base in the toehold of the join gate of the reaction you'd like to slow down. However, in practice it is difficult to slow down a single strand displacement reaction with these context-independent gates the way that they're currently being derived from plasmids.

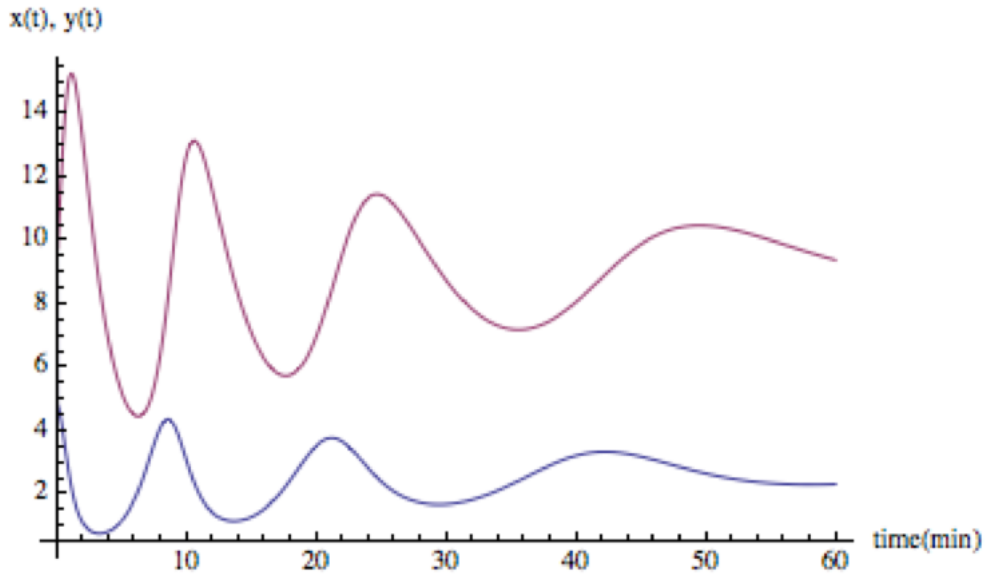

Figure 3.30. Simulated Lotka-Volterra Oscillations.

Oscillations are observed in a Lotka-Volterra oscillator when one of the reactions can be significantly slowed down. X (foxes) are depicted in blue and Y (rabbits) are depicted in red. Simulation done in Mathematica.

3.5.2 *Rock-Paper-Scissors Oscillator*

Here we implement a series of three autocatalytic reactions, which should in principle exhibit oscillatory behavior [62], contingent upon the kinetic rates of the autocatalytic reactions being the same and the gate and helper components being added to the test tube stoichiometrically. Our context-independent gates should ensure the kinetic rates being the same.

Reaction network:

$X2 + X4 \rightarrow X4 + X4 + Y4$

$X4 + X3 \rightarrow X3 + X3 + Y3$

$X3 + X2 \rightarrow X2 + X2 + Y2$

In order to control the stoichiometry of the reactions, we put all the join gates on a single plasmid and all the fork gates on their own plasmid. The gates are nicked on the plasmid but are not cut out and are instead used directly on the plasmid.

The reaction is kick-started with an initial amount of one of the oscillatory species. A fluorometric output is read of the third outputs from the reactions that do not participate in the dynamic behavior of the network (Y2, Y3, and Y4). Each of these reactions is read by a different fluorophore which overlaps in excitation and emission as little as possible with the other two fluorophores. Y2 is read out by Cy5 or Rox, Y3 is read out by FAM, and Y4 is read out by TAMRA or Cy5. The concentration of the species is then inferred from the increase of fluorescent reading of the inert species.

When testing the oscillator network, we see the start of a dynamic behavior begin to emerge. However, the limited energy in the reaction (some reactions are limited to only 70 nM of gates in the initial mixture to proceed with the reaction) limits the oscillatory behavior to that of a pulse generator; we see each oscillatory species take turns as the dominant species in the reaction chamber once. Figure 3.31 shows the instantiation of the oscillator network with various initial conditions. The concentration of oscillatory species is plotted by inferring the concentrations from the fluorescent readout of the inert species and then normalizing based on the max concentration of each species.

Panel D shows a large bias to starting the network with the X2 species. First the reaction X3 + X2 → X2 + X2 + Y2. As more X2, is generated, the reaction X2 + X4 → X4 + X4 + Y4, starts to take over the network. Lastly, as more X4 is generated, the reaction X4 + X3 → X3 + X3 + Y3 starts to take over and we see the generation of X3 starting to occur. However, the dynamic behavior of the reactions quickly loses steam because the system runs out of gates, as the initial condition has the oscillatory species at half the amount of gates in the system.

Panels E and F also show the dynamics of the network when the initial conditions are biased toward X2. While panel E is very noisy, we still observe that the same type of dynamics are occurring with X2 pulsing first, then X4. Panel F, which is run for a longer time period, and is less noisy, shows the same dynamic behavior of X2 pulsing first, then X4, followed by X3. The helper strands <x2 t>, <x3 t>, and <x4 t> are added at 10 nM and the rest are added at 80 nM to optimize the speed of the reactions by minimizing reverse reactions. These two panels have similar dynamics and suggest that the behavior of these gates is reproducible. Qualitatively, we can see that the period where each species is in the majority gets progressively longer and longer, suggesting that this network, with these initial conditions, is indeed a damped oscillator.

Panel G shows initial conditions that bias the oscillator towards pulsing X3 first. Here we see X3 pulsing first, followed by the rise in X2. This reaction is not run long enough to see the full dynamics of the pulse generation, but suggests that the dynamics of the system can be changed by altering the initial conditions.

In all experiments run with these gates, we see that the successive pulses take longer and longer to occur, mimicking the behavior of a damped oscillator due to the limited initial energy. With more careful selection of initial conditions, less leak in the autocatalytic reactions, or the

ability to add more energy initially (or to add gates to the reaction chamber continuously), we should see longer-term oscillatory behavior from this network.

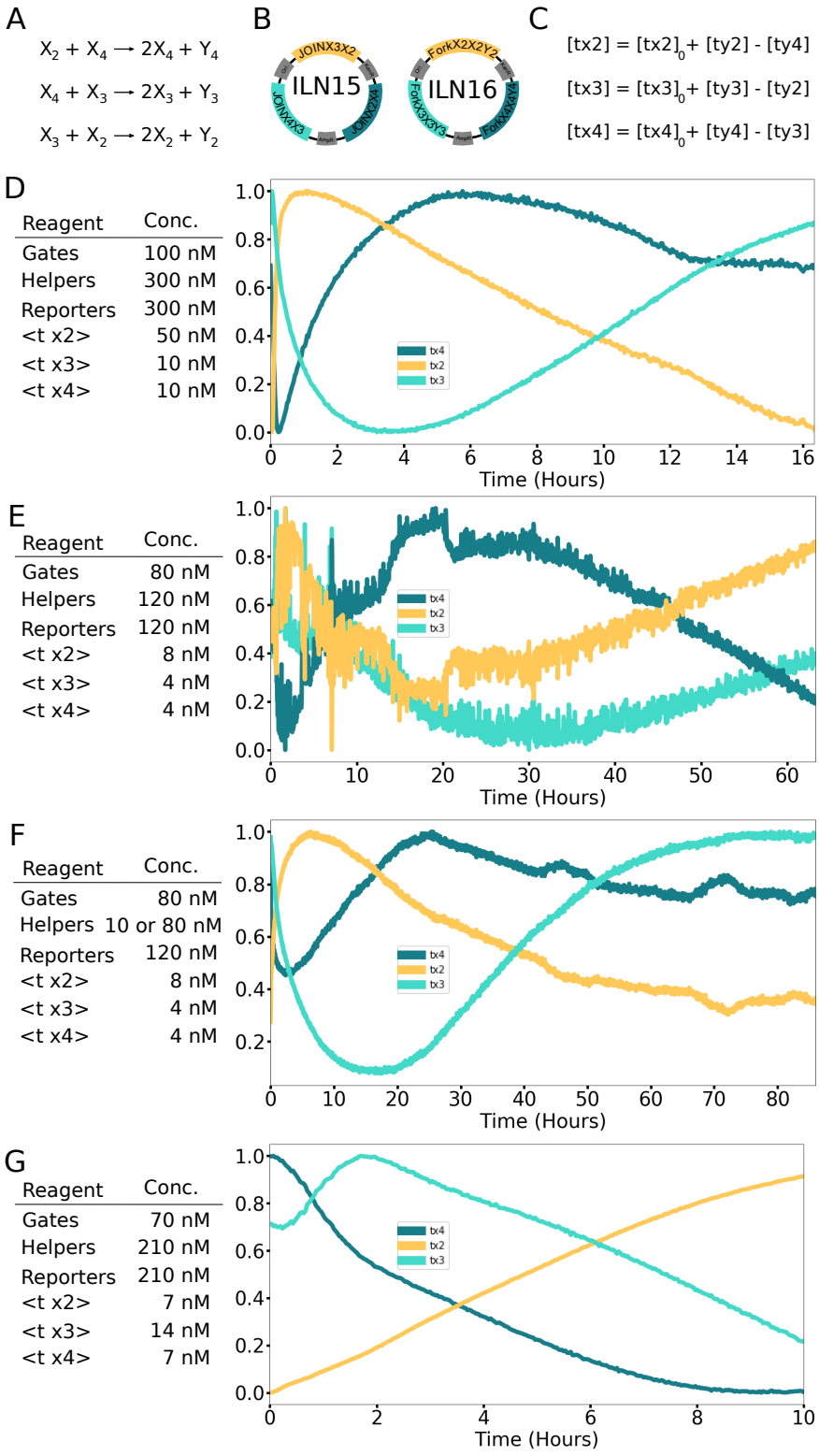

Figure 3.31. Rock-Paper-Scissors Circuit

A) Chemical Reaction Nework B) Stoichiometric plasmids designed for experiment C) Mathematical equations to deduce concentrations of oscillatory species. D-G) Different spectrophotometer experiments with varying initial conditions. The Y axis shows a normalized axis where each oscillatory species is normalized to the max of its concentration.

3.6 Temporal Conclusions

Our approach to building context-independent gates standardizes the toehold kinetic rates such that larger chemical reaction networks can easily be comprised from modular reactions. This is strong evidence that the flanking region around the toehold matters greatly when controlling toehold-mediated kinetic rates. Additionally, our ILN plasmid design allows for higher yield of DNA gates and a built-in mechanism for controlling stoichiometry of chemical reaction networks.

We built several chemical reactions, of various types, and composed some of them to build networks with interesting dynamical behaviors, including an oscillatory network and a consensus network. In the future, by controlling leak in our systems and increasing the yield of gates further, we should be able to see temporal dynamics, such as oscillations, generated for longer periods of time.

Chapter 4. MACROSCOPIC PATTERN FORMATION WITH DNA COMPONENTS

4.1 INTRODUCTION

Amongst the most spectacular and beautiful phenomena displayed by nature is its ability to generate complex spatial patterns, from the appearance of neatly arranged scalloped patterns on the bank of a sand dune to the arrangement of pigmentation patterns on living animals[63]. Bacteria have been shown to form patterns under oxidative stress [64], and chemical amalgams have been demonstrated to have oscillatory behavior [65]. The beauty of these systems is the apparent ability for order to arise from seeming disorder, for complexity to arise from apparent uniformity.

Pattern formation can be described as the creation of an inhomogeneous pattern from otherwise uniform starting conditions, a process that involves symmetry breaking and nonlinear dynamics. From a biological perspective, an interesting appearance of spatial patterns occurs in development during morphogenesis [66]. At its core, the problem of morphogenesis is one of self-organization, where the dynamics of cell-to-cell interactions directly dictate cellular activities and properties. Understanding the basic principles of pattern formation could be key to understanding these more fundamental cell-cell interactions. While the mechanics of many cellular interactions are very well understood, an open question that remains is which interactions are important in the process of cellular self-organization and how do these interactions affect the process?

This idea of pattern formation has been well studied in physics and chemistry for many years, resulting in much beautiful mathematical and theoretical explanations for the phenomena. One of the most well studied mechanisms for generating spatial pattern formation systems is

dictated by the idea of reaction-diffusion equations. Most famously, Alan Turing, in his seminal paper on pattern formation [67], suggested that the addition of diffusion to a heterogeneous reaction mixture could result in the formation of complex patterns.

DNA nanotechnologists dream of building a system in a programmable platform that allows us to discover bottom up construction principles for pattern formation [68]. The ideal system is one in which a pattern arises out of a homogenous mixture of reagents and has components that are highly engineerable. From this perspective, DNA Strand displacement (DSD) seems to be a good material for constructing reaction-diffusion systems from the bottom up because the inherent programmability of the mechanism gives it advantages over other synthetic biology methods [69]. The diffusion constants of DNA strand displacement systems can be controlled by altering the length of the DNA strands, adding modifications to the strands, altering the medium through which they diffuse or a combination of these methods.

Several groups have implemented pattern transformation systems, in which an input is transformed into a different, more complex patterns. One group constructed incoherent feedforward loops as a method for edge detection using DNA strand displacement [70]. The systems takes a binary input of light and dark and transforms it so that only the boundaries between light and dark are highlighted. The edge detection system system works by producing one functional part of the activated circuit in light conditions and another functional component of the activated circuit in dark conditions. Both components are necessary to get a desired output and are only present on the boundary between light and dark, where they have diffused to meet each other. This work demonstrates the power of the programmability of DNA nanotechnology.

Shulman et al have developed several DNA based frameworks for pattern transformation as well [71-73]. One system is a framework by which a specific, modular reaction pathway could

be initiated by a pre-patterned design to generate any arbitrary shape or pattern The reaction starts from a set of DNA sources and then cascades such that they can create designs, like stick figures. Alternatively, some systems use DNA controllers to change shapes within a DNA-hygrogel substrate. While these systems are easily programmable, the limitations of some of these systems are that some can only take in a programmed input and transform it into a similar pattern and many do not start with a homogenous mixture of molecules.

There have been many examples of synthetic reaction-diffusion systems that are capable of pattern formation as well. One of these systems is the Belousov-Zabotinskii system, in which a mixture of chemicals has been shown to exhibit oscillatory behavior [74]. A system designed by Rondelez [35] and company demonstrated patterns using enzymes and DNA. Basu et al designed a circuit in bacteria that demonstrated ring formation [75] and even traveling waves have been observed [76]. The limitation of these systems is that they may not be easily programmable. We would like to build upon these types of pattern formation systems to create a pattern-formation system implemented solely from DNA. The programmability of DNA will allow us to fully control all aspects of the pattern formation system and truly understand what is happening in the system. In our work, we take the approach of adding the previously developed temporal reactions to an environment that can accommodate diffusion to see the emergence of patterns on a spatial level.

4.2 Reaction-Diffusion Equations

Reaction- diffusion systems can be described by the following equation:

$$\frac{\partial \mathrm{u}}{\partial \mathrm{t}} = D\nabla^2 + f(u,p) \tag{4.3}$$

The first term describes “diffusion”, while the second term describes the “reaction” part of the system. In this equation, u = u (x, t) is a vector that typically represents the chemical concentration of a substance, or even a two-chemical system. ∇^2 denotes the Laplace operator and D is the diffusion coefficient, or a matrix of diffusion coefficients.

The second term $f(u, p)$ represents chemical reactions where p denotes the parameters of these reactions, for example, birth and death terms and rate constants. These equations also require boundary conditions to be provided for the system. Typically, these are zero flux (Neumann) boundary conditions for the chemicals in question. Meaning, at the boundaries of the reaction chamber,

$$\frac{\partial \mathrm{u}}{\partial \mathrm{t}} = 0 \ .$$

4.2.1 *Instability in Reaction-Diffusion Systems*

Turing postulated that there are some reaction-diffusion equations and boundary conditions that result in the appearance of heterogenous spatial patterns in steady state. Turing believed that that the presence of diffusion in an otherwise linearly stable system could drive the system into instability [67]. It turns out that there are several requirements for this to be the case. First, there must be more than one chemical in the system and the diffusion rates of the chemicals must be significantly different. Second, there must be a positive feedback mechanism and an inhibitory process. Lastly, it is favorable for the inhibitor to diffuse faster than the activator.

We can look at a system with two chemicals u_1 and u_2 to explore further:

$$\frac{\partial \mathrm{u_1}}{\partial \mathrm{t}} = D\nabla^2 + f_1(u_1, p) \tag{4.2}$$

$$\frac{\partial \mathrm{u}_2}{\partial \mathrm{t}} = D\nabla^2 + f_2(u_2, p) \tag{4.3}$$

The Jacobian Matrix J of the reaction-terms at a positive steady state $(\bar{u}_1, \bar{u}_2)$ looks as follows:

$$J = \begin{bmatrix} \frac{\partial \mathrm{f}_1}{\partial \mathrm{u}_1}(\bar{u}_1, \bar{u}_2) & \frac{\partial \mathrm{f}_1}{\partial \mathrm{u}_2}(\bar{u}_1, \bar{u}_2) \\ \frac{\partial \mathrm{f}_2}{\partial \mathrm{u}_1}(\bar{u}_1, \bar{u}_2) & \frac{\partial \mathrm{f}_2}{\partial \mathrm{u}_2}(\bar{u}_1, \bar{u}_2) \end{bmatrix} \tag{4.4}$$

The sign combinations of the Jacobian classify the reaction-diffusion systems into four equivalence classes [77].

$$\begin{bmatrix} + & - \\ + & - \end{bmatrix}, \begin{bmatrix} - & + \\ - & + \end{bmatrix}, \begin{bmatrix} + & + \\ - & - \end{bmatrix}, \begin{bmatrix} - & - \\ + & + \end{bmatrix}$$

These types of systems are known as activator-inhibitor systems. In this case, there is a fast-diffusing inhibitor that acts on production of a slowly diffusing activator. The activator acts on the inhibitor and also has a positive feedback loop on itself. An examples of reactions that give rise to these kinetics is the Gierer-Meinhardt system [78]:

$$f_1(u_1, u_2) = k_1 - k_2 u_1 + \frac{k_3 u_1^2}{u_2}$$

$$f_2(u_1, u_2) = k_4 u_1^2 - k_5 u_2 + k_6$$

In this case, u_1 is the activator and u_2 is the inhibitor. Additionally, $\frac{k_3 u_1^2}{u_2}$ is a nonlinear term for generation of the activator.

Alternatively, some of these systems are known as activator-substrate systems. In these instances, there is a slowly diffusing activator constantly supplied to the system, that has a positive feedback look on itself and consumes a fast-diffusing substrate. A specific instance of this is the Schnakenberg model [79] . The reactions that give rise to these kinetics look like:

$$f_1(u_1, u_2) = k_1 - k_2 u_1 + k_3 u_1 u_2$$

$$f_2(u_1, u_2) = k_4 - k_3 u_1 u_2$$

In these equations, all the rate constants are positive. Additionally, the term $k_3 u_1 u_2$ describes the autocatalytic production of the species u_1.

These types of reaction-diffusion systems provide a framework for producing pattern formation systems with chemical reactions in a diffusive setting. The nonlinear dynamics, in combination with symmetry breaking can result in interesting patterns. This is a pathway to creating synthetic pattern formation systems from DNA components.

4.3 Traveling Wave Propagation by an Autocatalytic Reaction

One of the simplest instantiations of a reaction diffusion system is a traveling wave. An autocatalytic system (X + Y → 2Y) could produce such a "pattern" under the correct conditions [80]. The one-dimensional reaction diffusion equations for the autocatalytic reaction are:

$$\frac{\partial[\mathrm{X}]}{\partial \mathrm{t}} = D\frac{\partial^2[\mathrm{X}]}{\partial x^2} - k[X][Y]$$

$$\frac{\partial[\mathrm{Y}]}{\partial \mathrm{t}} = D\frac{\partial^2[\mathrm{Y}]}{\partial x^2} + k[X][Y] \qquad (4.4)$$

Assuming that there is an excess of reactant X, such that diffusion of the reactant X is negligible,

$$[\mathrm{X}](\mathrm{t}\,,\,\mathrm{x}) \sim X_0 - [Y](t\,,x) \qquad (4.5)$$

Therefore, the evolutions of the travelling wave can be described by a single reaction-diffusion equation:

$$\frac{\partial[\mathrm{Y}]}{\partial \mathrm{t}} = D\frac{\partial^2[\mathrm{Y}]}{\partial x^2} + k(X_0 - [Y])[Y] \tag{4.6}$$

A rescaling of this equation, where [Y] =X0.θ gives us the Fisher equation [81]:

$$\frac{\partial \theta}{\partial \mathrm{t}} = D\frac{\partial^2 \theta}{\partial x^2} + f\theta(1-\theta) \tag{4.7}$$

with f = kX_0. This gives us travelling waves with the speed v = $2\sqrt{fD}$.

Previous theoretical work has indeed shown that it is possible to use the two-domain CRN architecture to see travelling wave propagation with an autocatalytic reaction in space [82].

4.4 Spatial Dynamics of DNA CRNS

Chapter 3 gave us a framework for engineering chemical reaction networks with predictable kinetics, allowing us to control the temporal dynamics of DNA strand displacement systems. We'd like to put these reactions in a diffusive setting in order to create systems where we can study the spatial dynamics of these DNA strand displacement systems.

The idea is to put the DNA CRN gates in a low percentage agarose gel which has the DNA gates and helper strands uniformly distributed throughout. We then spread this DNA-gel mixture to set in a reaction chamber, before punching out a hole in the center. We can then add the trigger strand for the reaction to the center of the chamber to initiate the reaction and see the emergence of a pattern. Given that our DNA CRNs take many strand displacement reactions to

correctly emulate the correct chemical reaction, we'll start with some simple reactions and make sure they work correctly in a spatial setting.

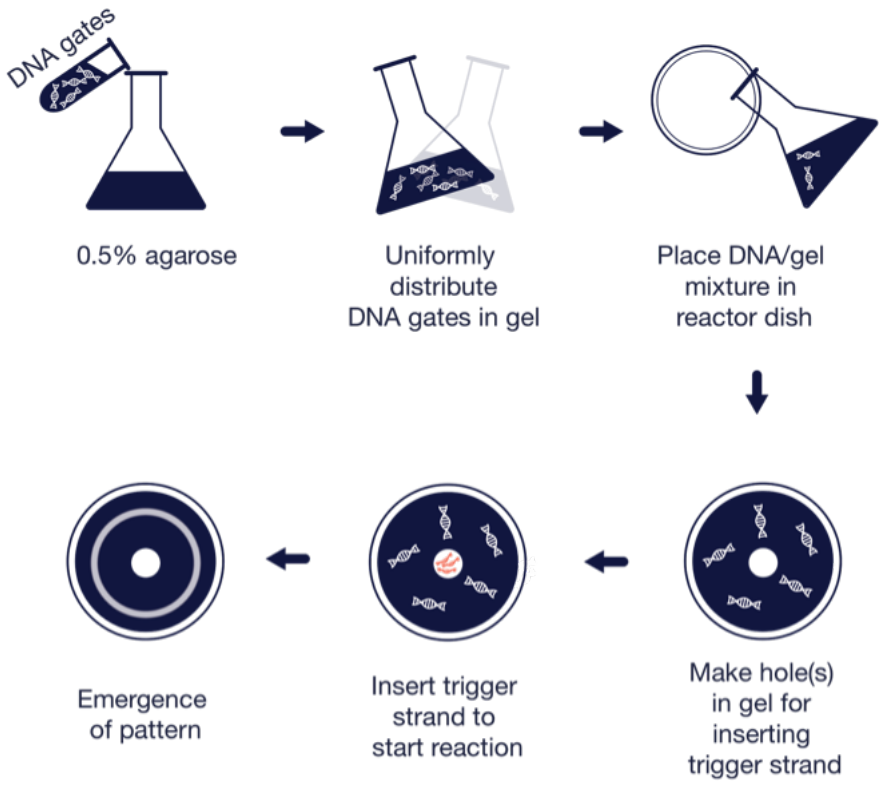

Figure 4.1. Schematic for Achieving Spatial Patterns with DNA Components.
Reproduced with permission from [83].

4.4.1 *Individual Gate Reactions*

Individual gate reactions were tested in a 0.5% agarose gel. In these tests, the control is to have the reporter gate uniformly distributed throughout the gel and then the reporter trigger is added to the center. In the gate experiments, the gates and all the helper strands and reporter complex are added to be uniformly distributed throughout the gel. Then the first input strand to the gate is added to the center of the gel to trigger the gate. Individual gates trigger successfully and diffuse at the same rate as the reporter gates by themselves. The intensity of the gates being triggered is

much lower. There are several reasons we hypothesize for this to be the case. First, the concentration of the gates is much lower than the concentration of the reporter complexes. This may allow the reporter gate alone to more quickly react. Additionally, the gate reactions require many more reactions to occur for signal to be seen. Strands may get sequestered as the reactions proceed, or the local concentrations of the helper strands or reacted gates may be low enough that the reactions do not proceed to the full strength.

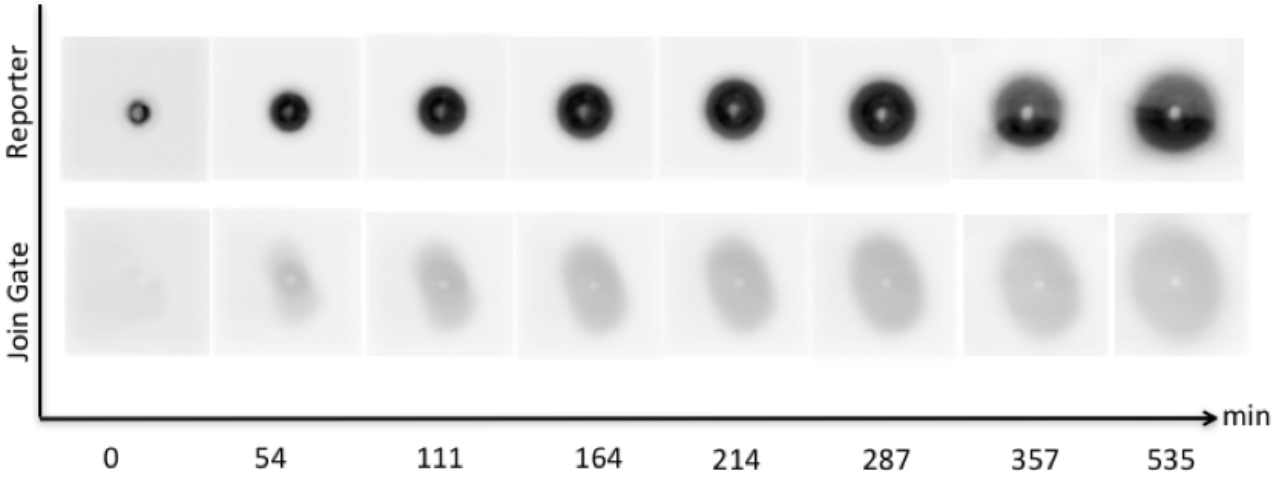

Figure 4.2. Join Gate in Spatial Setting.
This figure shows the Join gate in 0.5% Agarose Gel. The top row depicts a reporter gate which is triggered by 20 nM. The bottom row depicts the join gate and helper strands distributed homogenously in the gel. The reaction is triggered by 1x of the input strand. (1x = 20 nM).

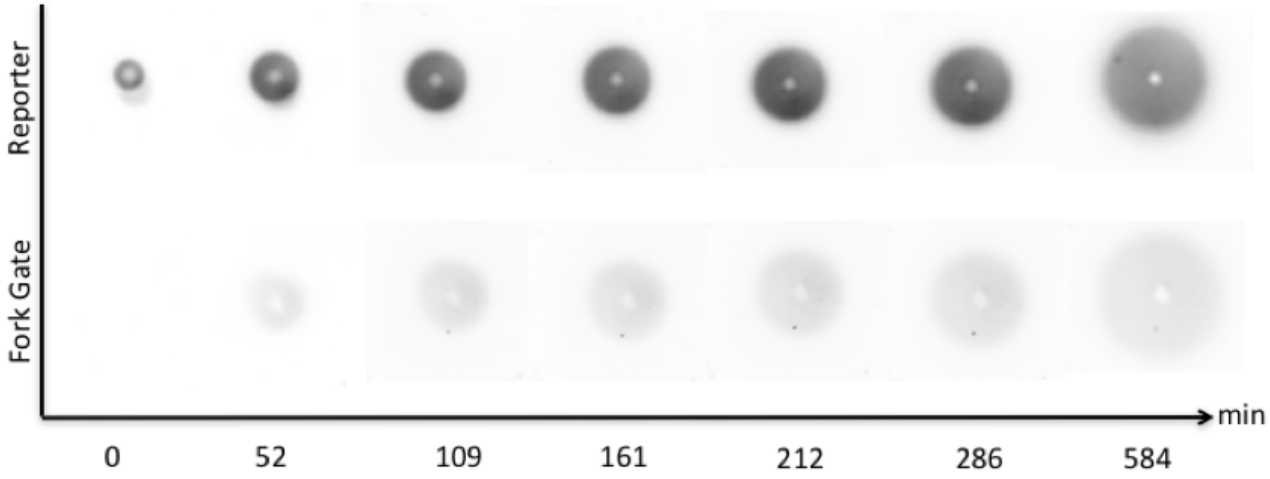

Figure 4.3. Fork Gate in Spatial Setting.
Fork gate in 0.5% agarose gel. The top row depicts a reporter gate which is triggered by 20 nM. The bottom row depicts the fork gate and helper strands distributed homogeneously in the gel. The reaction is triggered by 1x of the input strand. (1x = 20 nM)

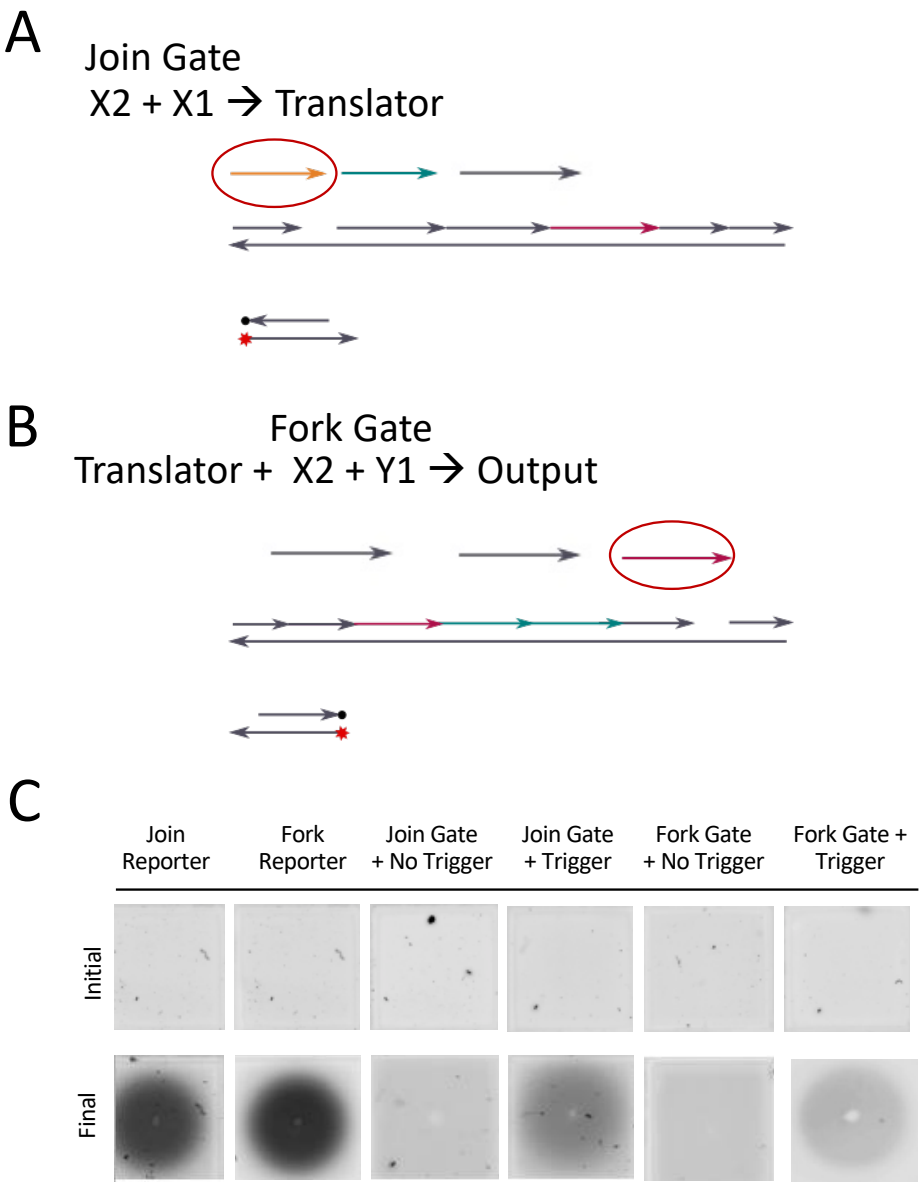

Figure 4.4: Individual Gates Exclusively Trigger with Correct Input in a Spatial Setting. Panel A) shows the schematic for the join gate. The input circled (the first input) is added to the center of the gel as the trigger, while the rest of the components pictured are uniformly distributed throughout the 0.5% agarose gel. Panel B) shows the schematic for the fork gate. The input circled (the first input) is added to the center of the gel as the trigger, while the rest of the components pictured are uniformly distributed throughout the 0.5% agarose gel. Panel C) shows that gates exclusively trigger in the presence of the trigger strand. The control gels are the triggered reporter gates shown in Panels A and B. (1x = 20 nM). Gel reactions are done in TAE buffer with 12.5 mM mg2+ and 0.15% SDS.

Additionally, we see that the gates do not trigger in the absence of the input of a trigger strand. Indicating that we are not just seeing some leak reactions, but indeed are witnessing the triggering of the gates themselves.

4.4.2 *Full Reactions in a Spatial Setting*

We have also seen that full reactions in a diffusive setting work as well. We can see that a non-catalytic reaction and a catalytic reaction work in a diffusive setting as well.

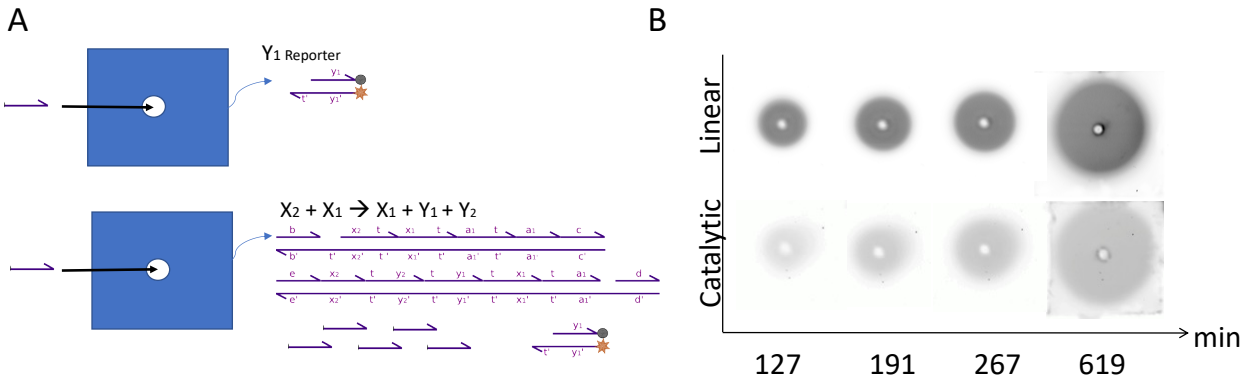

Figure 4.5: Catalytic Reaction in a Spatial Setting.
Panel A) shows the strands that go into the 0.5% agarose gel. The top row depicts a reporter gate which is triggered by 20 nM input. The bottom row depicts the gates required for the catalytic reaction and helper strands distributed homogeneously in the gel. The reaction is triggered by 1x of the catalyst (1x = 20 nM). Panel B) shows the expansion of the triggered reactions in the spatial setting.

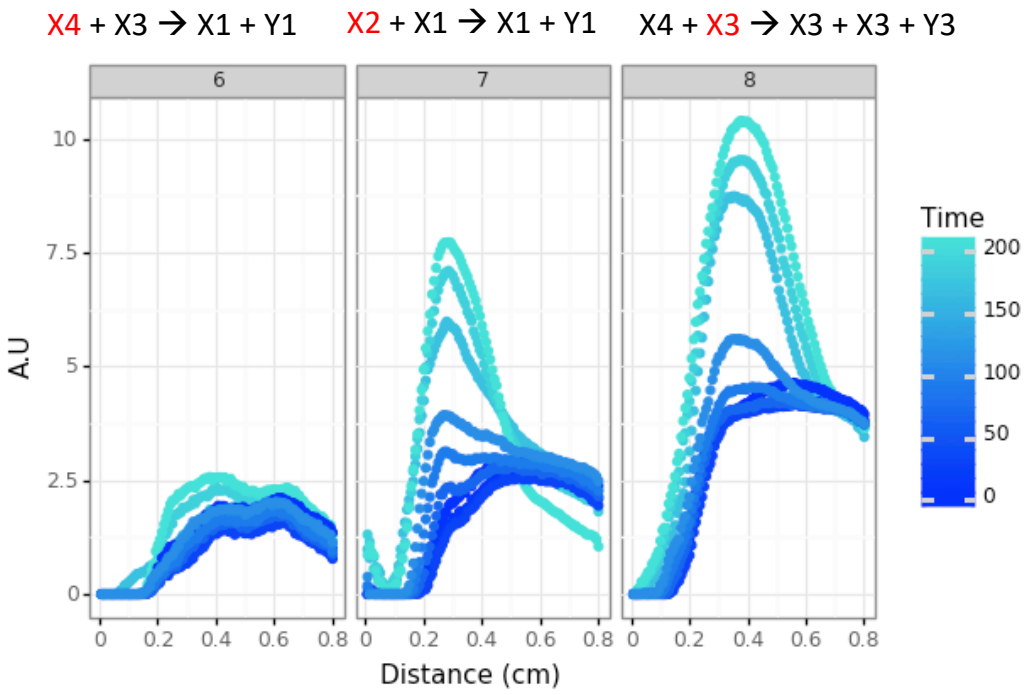

Figure 4.6. Various Reaction Types in Spatial Setting.
Here, we see the radially symmetric plots of each of the fundamental reaction types reacting in a spatial setting. All the components of the reactions are homogenously distributed in the gel except for the strand corresponding to the input to the reaction highlighted in red. These reactions show that the autocatalytic reaction increases in fluorescence the fastest, followed by the catalytic reaction and then the noncatalytic reaction.

4.4.3 *Experimental Travelling Wave Propagation*

Of particular interest is the autocatalytic reaction in a spatial setting. Nonlinear dynamics are typically required to be able to produce interesting spatial patterns. An autocatalytic reaction on its own should produce a propagating wave front. We see that there is a marked difference between the expansion of the autocatalytic reaction and the reporter complex on its own.

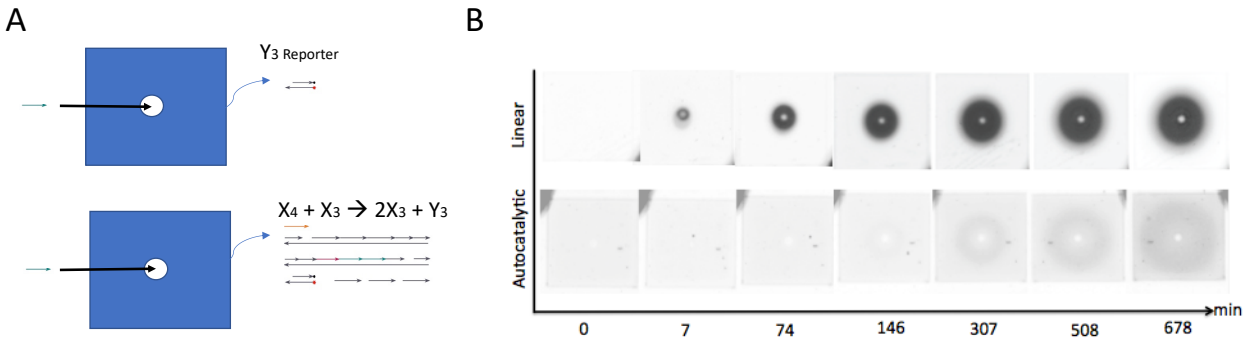

Figure 4.7: Autocatalytic Reaction in a Spatial Setting.
Panel A) shows the strands that go into the 0.5% agarose gel. The top row depicts a reporter gate which is triggered by 20 nM input. The bottom row depicts the gates required for the autocatalytic reaction and helper strands distributed homogeneously in the gel. The reaction is triggered by 1x of the catalyst (1x = 20 nM). Panel B) shows the expansion of the triggered reactions in the spatial setting.

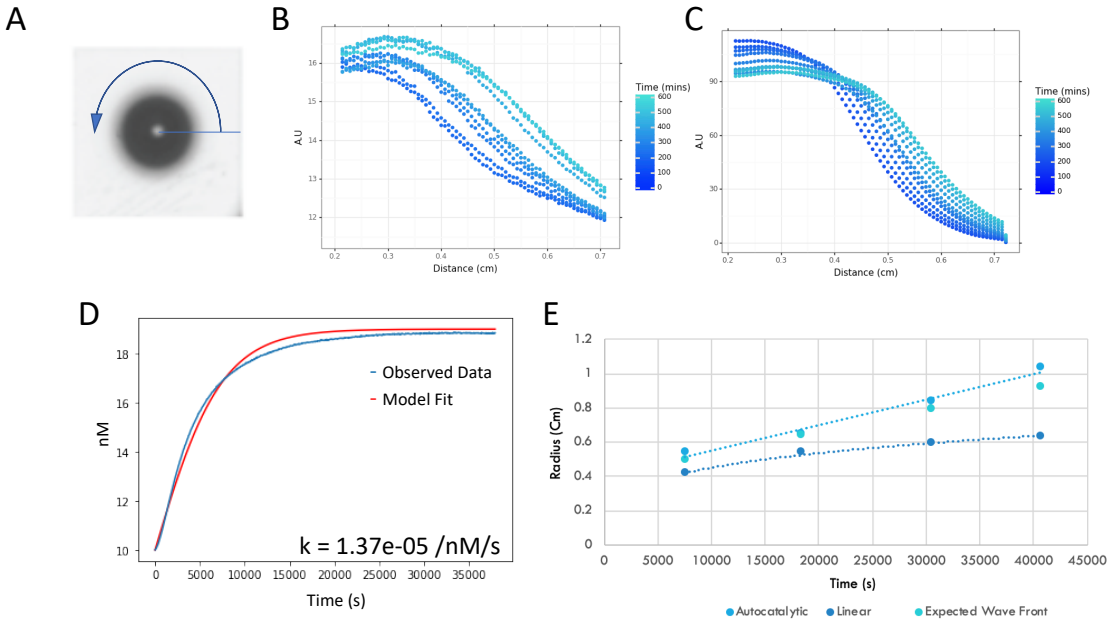

Figure 4.8: Radially Averaged View of Autocatalytic Reaction.
A) Depiction of how the radially symmetric plots are calculated B) Radially averaged plot of autocatalytic reaction being triggered in the gel (same reaction as Figure 4.7) C) Radially averaged plots of reporter triggering in gel. 1x = 20 nM D) Shows a model fit for the formal autocatalytic reaction. E) shows the expected wave front calculated by using the model fit from panel D.

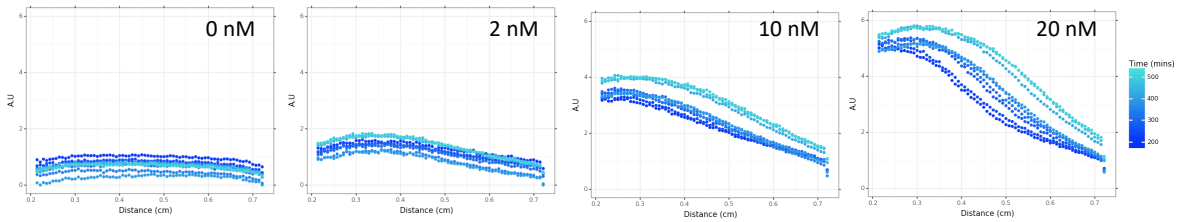

Figure 4.9: Varying Catalytic Input to Autocatalytic Reaction Changes Wave Intensity.
Here, we can see that varying the catalytic input to the autocatalytic reaction in a spatial setting changes the intensity of the wave as it propagates in space. Each panel depicts varying amounts of the catalyst being added to the center, while the rest of the components are homogenously distributed in the gel (1x = 20 nM).

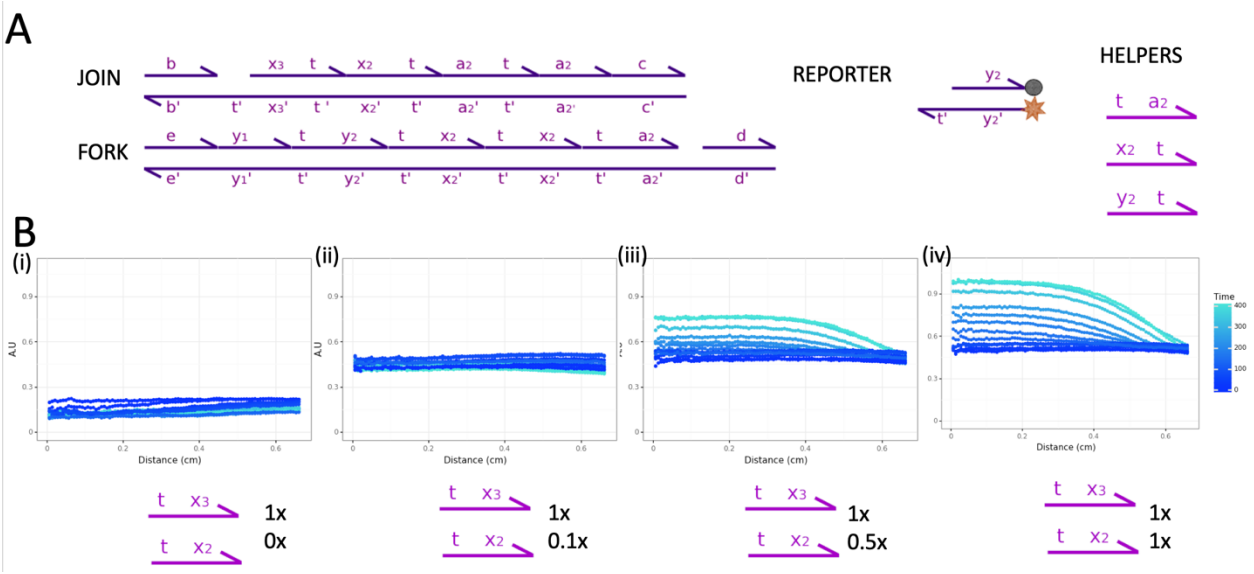

Figure 4.10: Varying Non-catalytic Input to Autocatalytic Reaction in a Spatial Setting.
Panel A) depicts the DNA strands that are necessary for the autocatalytic reaction ILN04_ILN06 in a spatial setting. Panel B) shows the radially averaged trace of the expansion of the fluorescent output of a reporter being triggered after the autocatalytic reaction occurs in the gel. (1x = 20 nM)., as various amounts of the noncatalytic input are added. The reaction increases in fluorescence in time.

When we put an autocatalytic reaction in a spatial setting we can start to see a traveling wave propagating. Figure 4.11 shows that there is a qualitative and quantitative difference between the autocatalytic reaction and the linear reaction in space. The autocatalytic reaction looks as though

there is a source for the reaction, and keeps the same fluorescent maximum, whereas the linear trace looks as though it is dominated by diffusion. This is confirmed in panel E of Figure 4.12, where the half-max radius of the autocatalytic reaction follows a linear trajectory, whereas the reporter radius follows a square root trajectory. Additionally, when we calculate the formal reaction rate k of the overall autocatalytic reaction (not the mechanistic strand displacement level reaction rate), we see that using this rate can give us a good prediction of where the front of the propagating wave should be. Additionally, we see that changing the amount of catalyst added to the reaction changes the intensity of the wave, though the dynamics remain largely the same. Changing the non-catalyst may change the speed of the wave propagation, but further experimentation is needed to see if this is true (to allow the reaction to reach saturation).

4.5 Spatial Conclusions

In this section, we saw that these DNA chemical reaction networks do indeed work in a spatial setting. We attempted the simplest reaction-diffusion system, which is a travelling wave generated by an autocatalytic reaction in a spatial setting. We see that the autocatalytic reaction does indeed seem to be moving faster outwards than a simple linear system (just a reporter) and follows the trajectory that we expect. However, the leak in the autocatalytic system quickly takes over the system and makes the existence of traveling waves difficult to follow for longer periods of time.

This work the first to use plasmid-derived gates in a spatial setting to see reactions on a centimeter scale. This is the first step towards an autonomous pattern formation using DNA components on the centimeter scale. When issues of leak are addressed, perhaps using some of

the strategies proposed by others in the field [84], it is feasible that stable pattern formation systems can be achieved using only DNA based components.

BIBLIOGRAPHY

[1] C. Jung and A. D. Ellington, "Diagnostic applications of nucleic acid circuits," (in eng), *Acc Chem Res,* vol. 47, no. 6, pp. 1825-35, Jun 2014, doi: 10.1021/ar500059c.

[2] V. Linko, A. Ora, and M. A. Kostiainen, "DNA Nanostructures as Smart Drug-Delivery Vehicles and Molecular Devices," (in eng), *Trends Biotechnol,* vol. 33, no. 10, pp. 586-594, Oct 2015, doi: 10.1016/j.tibtech.2015.08.001.

[3] R. Lopez, R. Wang, and G. Seelig, "A molecular multi-gene classifier for disease diagnostics," (in eng), *Nat Chem,* vol. 10, no. 7, pp. 746-754, 07 2018, doi: 10.1038/s41557-018-0056-1.

[4] B. M. Mognetti, "Programmable interactions with biomimetic DNA linkers at fluid membranes and interfaces," vol. 82, P. Cicuta, Ed., ed. Reports on Progress in Physics: IOP Publishing, 2019, p. 116601.

[5] T. R. Damase, A. Spencer, B. Samuel, and P. B. Allen, "Biomimetic Molecular Signaling using DNA Walkers on Microparticles," (in eng), *Sci Rep,* vol. 7, no. 1, p. 4081, 06 2017, doi: 10.1038/s41598-017-04316-1.

[6] Y. J. Chen *et al.*, "Programmable chemical controllers made from DNA," (in eng), *Nat Nanotechnol,* vol. 8, no. 10, pp. 755-62, Oct 2013, doi: 10.1038/nnano.2013.189.

[7] Fages F., Le Guludec G., and P. A. Bournez O., "Strong Turing Completeness of Continuous Chemical Reaction Networks and Compilation of Mixed Analog-Digital Programs.," vol. 10545, ed. Computational Methods in Systems Biology. CMSB 2017. Lecture Notes in Computer Science: Springer, Cham, 2017.

[8] D. Y. Zhang and G. Seelig, "Dynamic DNA nanotechnology using strand-displacement reactions," (in eng), *Nat Chem,* vol. 3, no. 2, pp. 103-13, Feb 2011, doi: 10.1038/nchem.957.

[9] L. M. Adleman, "Molecular computation of solutions to combinatorial problems," (in eng), *Science,* vol. 266, no. 5187, pp. 1021-4, Nov 1994, doi: 10.1126/science.7973651.

[10] D. Scalise and R. Schulman, "Controlling Matter at the Molecular Scale with DNA Circuits," (in eng), *Annu Rev Biomed Eng,* vol. 21, pp. 469-493, Jun 2019, doi: 10.1146/annurev-bioeng-060418-052357.

[11] R. R. Machinek, T. E. Ouldridge, N. E. Haley, J. Bath, and A. J. Turberfield, "Programmable energy landscapes for kinetic control of DNA strand displacement," (in eng), *Nat Commun,* vol. 5, p. 5324, Nov 2014, doi: 10.1038/ncomms6324.

[12] D. Y. Zhang and E. Winfree, "Control of DNA strand displacement kinetics using toehold exchange," (in eng), *J Am Chem Soc,* vol. 131, no. 47, pp. 17303-14, Dec 2009, doi: 10.1021/ja906987s.

[13] S. X. Chen and G. Seelig, "An Engineered Kinetic Amplification Mechanism for Single Nucleotide Variant Discrimination by DNA Hybridization Probes," (in eng), *J Am Chem Soc,* vol. 138, no. 15, pp. 5076-86, Apr 2016, doi: 10.1021/jacs.6b00277.

[14] S. Tyagi and F. R. Kramer, "Molecular beacons: probes that fluoresce upon hybridization," (in eng), *Nat Biotechnol,* vol. 14, no. 3, pp. 303-8, Mar 1996, doi: 10.1038/nbt0396-303.

[15] N. C. Seeman, "Nucleic acid junctions and lattices," (in eng), *J Theor Biol,* vol. 99, no. 2, pp. 237-47, Nov 1982, doi: 10.1016/0022-5193(82)90002-9.

[16] R. M. Dirks and N. A. Pierce, "Triggered amplification by hybridization chain reaction," (in eng), *Proc Natl Acad Sci U S A,* vol. 101, no. 43, pp. 15275-8, Oct 2004, doi: 10.1073/pnas.0407024101.

[17] G. Tikhomirov, P. Petersen, and L. Qian, "Triangular DNA Origami Tilings," (in eng), *J Am Chem Soc,* vol. 140, no. 50, pp. 17361-17364, 12 2018, doi: 10.1021/jacs.8b10609.

[18] P. Petersen, G. Tikhomirov, and L. Qian, "Information-based autonomous reconfiguration in systems of interacting DNA nanostructures," (in eng), *Nat Commun,* vol. 9, no. 1, p. 5362, 12 2018, doi: 10.1038/s41467-018-07805-7.

[19] D. Woods *et al.*, "Diverse and robust molecular algorithms using reprogrammable DNA self-assembly," (in eng), *Nature,* vol. 567, no. 7748, pp. 366-372, 03 2019, doi: 10.1038/s41586-019-1014-9.

[20] G. Seelig, D. Soloveichik, D. Y. Zhang, and E. Winfree, "Enzyme-free nucleic acid logic circuits," (in eng), *Science,* vol. 314, no. 5805, pp. 1585-8, Dec 2006, doi: 10.1126/science.1132493.

[21] G. Chatterjee, N. Dalchau, R. A. Muscat, A. Phillips, and G. Seelig, "A spatially localized architecture for fast and modular DNA computing," (in eng), *Nat Nanotechnol,* vol. 12, no. 9, pp. 920-927, 09 2017, doi: 10.1038/nnano.2017.127.

[22] P. Yin, H. Yan, X. G. Daniell, A. J. Turberfield, and J. H. Reif, "A unidirectional DNA walker that moves autonomously along a track," (in eng), *Angew Chem Int Ed Engl,* vol. 43, no. 37, pp. 4906-11, Sep 2004, doi: 10.1002/anie.200460522.

[23] A. J. Thubagere *et al.*, "A cargo-sorting DNA robot," (in eng), *Science,* vol. 357, no. 6356, 09 2017, doi: 10.1126/science.aan6558.

[24] E. S. Andersen *et al.*, "Self-assembly of a nanoscale DNA box with a controllable lid," (in eng), *Nature,* vol. 459, no. 7243, pp. 73-6, May 2009, doi: 10.1038/nature07971.

[25] L. Organick *et al.*, "Random access in large-scale DNA data storage," (in eng), *Nat Biotechnol,* vol. 36, no. 3, pp. 242-248, 03 2018, doi: 10.1038/nbt.4079.

[26] R. Lopez *et al.*, "DNA assembly for nanopore data storage readout," (in eng), *Nat Commun,* vol. 10, no. 1, p. 2933, 07 2019, doi: 10.1038/s41467-019-10978-4.

[27] L. Qian and E. Winfree, "Scaling up digital circuit computation with DNA strand displacement cascades," (in eng), *Science,* vol. 332, no. 6034, pp. 1196-201, Jun 2011, doi: 10.1126/science.1200520.

[28] L. Qian, E. Winfree, and J. Bruck, "Neural network computation with DNA strand displacement cascades," (in eng), *Nature,* vol. 475, no. 7356, pp. 368-72, Jul 2011, doi: 10.1038/nature10262.

[29] P. Wang *et al.*, "Practical aspects of structural and dynamic DNA nanotechnology," ed. *MRS Bulletin*, 2017, pp. 889-896.

[30] Y. J. Chen, B. Groves, R. A. Muscat, and G. Seelig, "DNA nanotechnology from the test tube to the cell," (in eng), *Nat Nanotechnol,* vol. 10, no. 9, pp. 748-60, Sep 2015, doi: 10.1038/nnano.2015.195.

[31] G. Chatterjee, Y. J. Chen, and G. Seelig, "Nucleic Acid Strand Displacement with Synthetic mRNA Inputs in Living Mammalian Cells," (in eng), *ACS Synth Biol,* vol. 7, no. 12, pp. 2737-2741, 12 2018, doi: 10.1021/acssynbio.8b00288.

[32] D. Soloveichik, G. Seelig, and E. Winfree, "DNA as a universal substrate for chemical kinetics," (in eng), *Proc Natl Acad Sci U S A,* vol. 107, no. 12, pp. 5393-8, Mar 2010, doi: 10.1073/pnas.0909380107.

[33] L. Cardelli, "Two-domain DNA strand displacement," *Mathematical Structures in Computer Science,* vol. 23, no. 2, pp. 247-271, 2013.

[34] A. N. Zaikin and A. M. Zhabotinsky, "Concentration wave propagation in two-dimensional liquid-phase self-oscillating system," (in eng), *Nature,* vol. 225, no. 5232, pp. 535-7, Feb 1970, doi: 10.1038/225535b0.

[35] A. Padirac, T. Fujii, A. Estévez-Torres, and Y. Rondelez, "Spatial waves in synthetic biochemical networks," (in eng), *J Am Chem Soc,* vol. 135, no. 39, pp. 14586-92, Oct 2013, doi: 10.1021/ja403584p.

[36] P. W. Rothemund, "Folding DNA to create nanoscale shapes and patterns," (in eng), *Nature,* vol. 440, no. 7082, pp. 297-302, Mar 2006, doi: 10.1038/nature04586.

[37] S. Cooper *et al.*, "Predicting protein structures with a multiplayer online game," (in eng), *Nature,* vol. 466, no. 7307, pp. 756-60, Aug 2010, doi: 10.1038/nature09304.

[38] J. Anderson-Lee *et al.*, "Principles for Predicting RNA Secondary Structure Design Difficulty," (in eng), *J Mol Biol,* vol. 428, no. 5 Pt A, pp. 748-757, Feb 2016, doi: 10.1016/j.jmb.2015.11.013.

[39] A. Kawrykow *et al.*, "Phylo: a citizen science approach for improving multiple sequence alignment," (in eng), *PLoS One,* vol. 7, no. 3, p. e31362, 2012, doi: 10.1371/journal.pone.0031362.

[40] J. S. Kim *et al.*, "Space-time wiring specificity supports direction selectivity in the retina," (in eng), *Nature,* vol. 509, no. 7500, pp. 331-336, May 2014, doi: 10.1038/nature13240.

[41] Algoraph, "http://algoraph.hope.edu/?page=main," ed.

[42] W. Dietl *et al.*, "Verification games: Making verification fun.," presented at the Proceedings of the 14th Workshop on Formal Techniques for Java-like Programs, New York, NY, USA, 2012.

[43] J. J. Sørensen *et al.*, "Exploring the quantum speed limit with computer games," (in eng), *Nature,* vol. 532, no. 7598, pp. 210-3, Apr 2016, doi: 10.1038/nature17620.

[44] Minecraft, "https://www.minecraft.net/en-us/," ed.

[45] SpaceChem, "http://www.zachtronics.com/spacechem/," ed.

[46] Infinifactory, "https://www.gog.com/game/infinifactory," ed.

[47] Lightbot, "https://www.lightbot.com/flash.html," ed.

[48] RoboZZle, "http://www.robozzle.com/," ed.

[49] Scribblenauts, "https://www.scribblenauts.com/," ed.

[50] J. Barone *et al.*, "Nanocrafter: Design and Evaluation of a DNA Nanotechnology Game," in *FDG*, 2015.

[51] J. Maloney, M. Resnick, N. Rusk, S. B., and E. Eastmond, "The Scratch Programming Language and Environment," ed. *ACM Transactions on Computing Education*, 2010.

[52] F. C. Simmel, B. Yurke, and H. R. Singh, "Principles and Applications of Nucleic Acid Strand Displacement Reactions," (in eng), *Chem Rev,* vol. 119, no. 10, pp. 6326-6369, May 2019, doi: 10.1021/acs.chemrev.8b00580.

[53] Y. Krishnan and F. C. Simmel, "Nucleic acid based molecular devices," (in eng), *Angew Chem Int Ed Engl,* vol. 50, no. 14, pp. 3124-56, Mar 2011, doi: 10.1002/anie.200907223.

[54] M. R. Lakin, S. Youssef, L. Cardelli, and A. Phillips, "Abstractions for DNA circuit design," (in eng), *J R Soc Interface,* vol. 9, no. 68, pp. 470-86, Mar 2012, doi: 10.1098/rsif.2011.0343.

[55] A. Phillips and L. Cardelli, "A programming language for composable DNA circuits," (in eng), *J R Soc Interface,* vol. 6 Suppl 4, pp. S419-36, Aug 2009, doi: 10.1098/rsif.2009.0072.focus.

[56] N. C. Seeman, *Structural DNA nanotechnology*. Cambridge, United Kingdom: Cambridge University Press is part of the University of Cambridge, 2015, pp. x, 257 pages.

[57] J. N. Zadeh *et al.*, "NUPACK: Analysis and design of nucleic acid systems," (in eng), *J Comput Chem,* vol. 32, no. 1, pp. 170-3, Jan 2011, doi: 10.1002/jcc.21596.

[58] N. Bolten, "Coral," ed. https://pypi.org/project/coral/.

[59] M. R. Lakin, S. Youssef, F. Polo, S. Emmott, and A. Phillips, "Visual DSD: a design and analysis tool for DNA strand displacement systems," (in eng), *Bioinformatics,* vol. 27, no. 22, pp. 3211-3, Nov 2011, doi: 10.1093/bioinformatics/btr543.

[60] Y. J. Chen, S. D. Rao, and G. Seelig, "Plasmid-derived DNA Strand Displacement Gates for Implementing Chemical Reaction Networks," (in eng), *J Vis Exp,* no. 105, Nov 2015, doi: 10.3791/53087.

[61] A. M. Albertini, M. Hofer, M. P. Calos, and J. H. Miller, "On the formation of spontaneous deletions: the importance of short sequence homologies in the generation of large deletions," (in eng), *Cell,* vol. 29, no. 2, pp. 319-28, Jun 1982, doi: 10.1016/0092-8674(82)90148-9.

[62] N. Srinivas, J. Parkin, G. Seelig, E. Winfree, and D. Soloveichik, "Enzyme-free nucleic acid dynamical systems," (in eng), *Science,* vol. 358, no. 6369, 12 2017, doi: 10.1126/science.aal2052.

[63] C. Liu *et al.*, "Sequential establishment of stripe patterns in an expanding cell population," (in eng), *Science,* vol. 334, no. 6053, pp. 238-41, Oct 2011, doi: 10.1126/science.1209042.

[64] L. Tsimring *et al.*, "Aggregation Patterns in Stressed Bacteria," (in eng), *Phys Rev Lett,* vol. 75, no. 9, pp. 1859-1862, Aug 1995, doi: 10.1103/PhysRevLett.75.1859.

[65] P. De Kepper, V. Castets, E. Dulos, and J. Boissonade, "Turing-type chemical patterns in the chlorite-iodide-malonic acid reaction," vol. 49, ed. Physica D: Nonlinear Phenomena, 1991, pp. 161-169.

[66] M. Isalan, C. Lemerle, and L. Serrano, "Engineering gene networks to emulate Drosophila embryonic pattern formation," (in eng), *PLoS Biol,* vol. 3, no. 3, p. e64, Mar 2005, doi: 10.1371/journal.pbio.0030064.

[67] A. M. Turing, "The chemical basis of morphogenesis. 1953," (in eng), *Bull Math Biol,* vol. 52, no. 1-2, pp. 153-97; discussion 119-52, 1990, doi: 10.1007/bf02459572.

[68] S. S. Wang and A. D. Ellington, "Pattern Generation with Nucleic Acid Chemical Reaction Networks," (in eng), *Chem Rev,* vol. 119, no. 10, pp. 6370-6383, May 2019, doi: 10.1021/acs.chemrev.8b00625.

[69] L. Diambra, V. R. Senthivel, D. B. Menendez, and M. Isalan, "Cooperativity to increase Turing pattern space for synthetic biology," (in eng), *ACS Synth Biol,* vol. 4, no. 2, pp. 177-86, Feb 2015, doi: 10.1021/sb500233u.

[70] S. M. Chirieleison, P. B. Allen, Z. B. Simpson, A. D. Ellington, and X. Chen, "Pattern transformation with DNA circuits," (in eng), *Nat Chem,* vol. 5, no. 12, pp. 1000-5, Dec 2013, doi: 10.1038/nchem.1764.

[71] Scalise, D., and R. Schulman, "Designing modular reaction-diffusion programs for complex pattern formation," vol. 2, ed. Technology, 2014, pp. 55–66.

[72] J. Fern and R. Schulman, "Modular DNA strand-displacement controllers for directing material expansion," (in eng), *Nat Commun,* vol. 9, no. 1, p. 3766, 09 2018, doi: 10.1038/s41467-018-06218-w.

[73] A. Cangialosi *et al.*, "DNA sequence-directed shape change of photopatterned hydrogels via high-degree swelling," (in eng), *Science,* vol. 357, no. 6356, pp. 1126-1130, 09 2017, doi: 10.1126/science.aan3925.

[74] V. K. Vanag and I. R. Epstein, "Pattern formation in a tunable medium: the Belousov-Zhabotinsky reaction in an aerosol OT microemulsion," (in eng), *Phys Rev Lett,* vol. 87, no. 22, p. 228301, Nov 2001, doi: 10.1103/PhysRevLett.87.228301.

[75] S. Basu, Y. Gerchman, C. H. Collins, F. H. Arnold, and R. Weiss, "A synthetic multicellular system for programmed pattern formation," (in eng), *Nature,* vol. 434, no. 7037, pp. 1130-4, Apr 2005, doi: 10.1038/nature03461.

[76] A. S. Zadorin, Y. Rondelez, J. C. Galas, and A. Estevez-Torres, "Synthesis of programmable reaction-diffusion fronts using DNA catalyzers," (in eng), *Phys Rev Lett,* vol. 114, no. 6, p. 068301, Feb 2015, doi: 10.1103/PhysRevLett.114.068301.

[77] R. C. Hilborn, *Chaos and nonlinear dynamics : an introduction for scientists and engineers*, 2nd ed. Oxford ; New York: Oxford University Press, 2000, pp. xxi, 650 p.

[78] A. Gierer and H. Meinhardt, "A theory of biological pattern formation," (in eng), *Kybernetik,* vol. 12, no. 1, pp. 30-9, Dec 1972, doi: 10.1007/bf00289234.

[79] J. Schnakenberg, "Simple chemical reaction systems with limit cycle behaviour," (in eng), *J Theor Biol,* vol. 81, no. 3, pp. 389-400, Dec 1979, doi: 10.1016/0022-5193(79)90042-0.

[80] G. J. Bauer, J. S. McCaskill, and H. Otten, "Traveling waves of in vitro evolving RNA," (in eng), *Proc Natl Acad Sci U S A,* vol. 86, no. 20, pp. 7937-41, Oct 1989, doi: 10.1073/pnas.86.20.7937.

[81] R. A. Fisher, "152: The Wave of Advance of Advantageous Genes.," vol. 7, ed. Annals of Eugenics, 1937, pp. 355-369.

[82] N. Dalchau, G. Seelig, and P. A., "Computational Design of Reaction-Diffusion Patterns Using DNA-Based Chemical Reaction Networks," presented at the DNA Computing and Molecular Programming, 2014.

[83] S. Chen and G. Seelig, "Programmable patterns in a DNA-based reaction-diffusion system," ed. BioArXiv, 2019.

[84] B. Wang, C. Thachuk, A. D. Ellington, E. Winfree, and D. Soloveichik, "Effective design principles for leakless strand displacement systems," (in eng), *Proc Natl Acad Sci U S A,* vol. 115, no. 52, pp. E12182-E12191, 12 2018, doi: 10.1073/pnas.1806859115.

[85] N. Srinivas *et al.*, "On the biophysics and kinetics of toehold-mediated DNA strand displacement," *Nucleic acids research,* vol. 41, no. 22, pp. 10641-10658, 2013.

APPENDIX A

Sample Palindromic Gates Code:

```
directive simulation {
  final=4000;
  plots=[6];
}
directive simulator deterministic
directive parameters [
  k = 0.0014257768794900605, {distribution=Uniform(0.001,1)};
  u = 0.1;
]
directive inference {burnin=100; samples=100}
directive data [74]
directive compilation infinite
dom t = {bind=k;unbind=u;colour="purple"}
def Input1() = < p t^>
def Join() = [t^ p]{t^*}
def Reporter() = {t^*}[p]<fl^>
def Signal() = <p fl^  >
( 1.1 Input1()
| 1.5 Join()
| 3 Reporter()
| 0 Signal()
)
```

Sample Inference code for autocatalytic reaction:

```
directive duration 40000.0 points 2000
directive plot Signal()
directive scale 10.0
directive compilation infinite
directive simulation deterministicstiff
directive event <t^ x3> N*(1.0-Bad)*10.0 @ T1
directive parameters [
  N = 1.0;
  kt,(1.0e-5,2.0e-3),0.0002141779,log,random;
  kl,(1.0e-10,1.0e-3),3.751260979e-7,log,random;
  T1,(-3600.0,8000.0),640.96,real,random;
  LeakJ,(0.0,0.05),0.0499279,real,random;
  LeakF,(0.0,0.05),0.0499279,real,random;
  Bad,(0.0,1.5),0.45627101,real,random
]

(* Directives for parameter inference *)
(* Burn-in, runs, optimisation burn-in, optimisation of maximum likelihood,... *)
directive fit_run { burnin = 2000
                  ; samples = 20000
```

```
                  ; mle_burnin = 1000
                  ; mle_samples = 1000
                  ; thin = 10 }
directive fit { s1; autocatX4 ; <y3 fl^ >}
directive sweep s1 = {N = [79]}

def ut = 0.1
new t@kt,ut

def Input1() = <t^ x4>
def Input2() = <t^ x3>
def Input3() = <t^ a3>
def Input4() = <x3 t^>
def Input5() = <y3 t^>
def Output() = <t^ y3>
def Translator() = <a3 t^>
def Join() = :{t^*}[x4 t^]:[x3 t^]:[a3 t^]:[a3]:
def Fork() =[85]:[y1]:[t^ y3]:[t^ x3]:[t^ x3]:[t^ a3]:{t^*}:[d]
def ForkLeaked() =[85]:[y1]:{t^*}:[y3 t^]:[t^ x3]:<t^ >[x3]:[t^ a3]:{t^*}:[d]
def Reporter() = {t^*}[y3]<fl^>
def Signal() = <y3 fl^ >
( 10.0*Input1()
| 0.0*Input2()
| 30.0*Input3()
| 30.0*Input4()
| 30.0*Input5()
| 15.0*(1.0-LeakJ)*Join()
| 15.0*LeakJ*Translator()
| 15.0*(1.0-LeakF)*Fork()
| 15.0*LeakF*Output()
| 30.0*Reporter()
| 0.0*Signal()
| 0.0*ForkLeaked()
| rxn Fork() + Input4() ->{kl} <t^ x3> + ForkLeaked()
)
```

APPENDIX B

Table 4.2. Palindromic Gate Sequences

Name	**Sequence**	**Bases**	**Extinction Co.**
t p	CACCAC CATTGCTATTCACACTTATCGTTAC	31	284600
p t	CATTGCTATTCACACTTATCGTTAC CACCAC	31	284600
t*p*t*	GTGGTG GTAACGATAAGTGTGAATAGCAATG GTGGTG	37	380700
Reporters			
p-RQ	CATTGCTATTCACACTTATCGTTAC	25	275506
Cy5-p*-t*	GTAACGATAAGTGTGAATAGCAATG GTGGTG	31	332900
RQ-p	CATTGCTATTCACACTTATCGTTAC	25	281457
t*-p*-Cy5	GTGGTG GTAACGATAAGTGTGAATAGCAATG	31	332900

Table 4.3. Domain Sequences

Name	Sequence
t	CACCAC
X1	CATTGCACATTATATCGAGTCTTAC
X2	CATTGCCACATCCTCCGAGTCTTAC
X3	CATTGCATCTATAAACGAGTCTTAC
X4	CATTGCTTAACATACAGAGTCTTAC
X5	CATTGCTATCAACTTAGAGTCTTAC
Y1	CATTGCCCTACCACCTGAGTCTTAC
Y2	CATTGCCATTTCCCTAGAGTCTTAC
Y3	CATTGCTTCTATCTTTGAGTCTTAC
Y4	CATTGCTTCTTATATTGAGTCTTAC
A1	CATTGCTATTCACACAGAGTCTTAC
A2	CATTGCCTTATAAATTGAGTCTTAC
A3	CATTGCATATTTCCCAGAGTCTTAC
A4	CATTGCCAACCCATTCGAGTCTTAC
A5	CATTGCATCATATAGTCAGTCTTAC
b	CTGTCCTTTATTTCGCTCTTCC
c	CATTGCCACCTCCACCCTCCAG
d	CACTGCACCAACAACCAATCAG
e	CTGCCCATATTTCGAGTCTTAC

Table 4.4. Reporter Gate Strands

Name	Ext Co
RQ-a1	287357
t'a'-TAMRA	349200
RQ-a2	288957
t*-a2*-TAMRA	354000
RQ-a3	284057
t*-a3*-TAMRA	351900
RQ-a4	279857
t*-a4*-TAMRA	347300
x1-FQ	253344
FAM-x't'	336460
x3-RQ	287006
ROX-x3*-t*	334200
x4-RQ	268406
ROX-x4*-t*	333000
y1-RQ	268106
Cy5-y1't'	323900
ROX-y't'	336500
y2-RQ	271906
Cy5-y2*-t*	326300
y3-FQ	237244
FAM-y3*-t*	347560
y4-RQ	275206
Cy5-y4*-t*	335400
RQ-x1	290457
t'x1'-Cy5	325500
RQ -x2	275057
t'-x2'-Cy5	322900
RQ-X3	292957
t'x3'Cy5	321600
RQ-x4	292757
t'x4'Cy5	320400

Table 4.5. ILN Plasmid Gate Sequences and VDSD Representations

Name	VDSD Representation	Main Sequence
ILN01Pro_JOINY1X4_X4X4Y4		CAG CTGTCCTTTATTTCGCTCTTCC CACCAC CATTGCCCTACCACCTGAGTCTTAC CACCAC CATTGCTTAACATACAGAGTCTTAC CACCAC CATTGCCAACCCATTCGAGTCTTAC CACCAC CATTGCCAACCCATTCGAGTCTTAC CATTGCCACCTCCACCCTCCAG CTG
ILN02Pro_JOINY1X4_rev		CAG CTGTCCTTTATTTCGCTCTTCC CACCAC CATTGCCCTACCACCTGAGTCTTAC CACCAC CATTGCTTAACATACAGAGTCTTAC CACCAC CATTGCCAACCCATTCGAGTCTTAC CACCAC CACTGCACCAACAACCAATCAG CTG
ILN03Pro_JOINX2X4_X4X4Y4	[b]:{t^*}[x2 t^]:[x4 t^]:[a4 t^]:[a4]:[c]	CTGTCCTTTATTTCGCTCTTCC CACCAC CATTGCCACATCCTCCGAGTCTTAC CACCAC CATTGCTTAACATACAGAGTCTTAC CACCAC CATTGCCAACCCATTCGAGTCTTAC CACCAC CATTGCCAACCCATTCGAGTCTTAC CATTGCCACCTCCACCCTCCAG CTG
ILN04Pro_JOINX3X2_X2X2Y2	[b]:{t^*}[x3 t^]:[x2 t^]:[a2 t^]:[a2]:[c]	CAG CTGTCCTTTATTTCGCTCTTCC CACCAC CATTGCATCTATAAACGAGTCTTAC CACCAC CATTGCCACATCCTCCGAGTCTTAC CACCAC CATTGCCTTATAAATTGAGTCTTAC CACCAC CATTGCCTTATAAATTGAGTCTTAC CATTGCCACCTCCACCCTCCAG CTG
ILN05Pro_JOINX4X3_X3X3Y3	[b]:{t^*}[x4 t^]:[x3 t^]:[a3 t^]:[a3]:[c]	CAG CTGTCCTTTATTTCGCTCTTCC CACCAC CATTGCTTAACATACAGAGTCTTAC CACCAC CATTGCATCTATAAACGAGTCTTAC CACCAC CATTGCATATTTCCCAGAGTCTTAC CACCAC CATTGCATATTTCCCAGAGTCTTAC CATTGCCACCTCCACCCTCCAG CTG

ILN06Pro_FORKX3X2 _X2X2Y2	[e]:[y1]:[t^ y2]:[t^ x2]:[t^ x2]:[t^ a2]:{t^*}:[d]	CAG CTGCCCATATTTCGAGTCTTAC CATTGCCCTACCACCTGAGTCTTAC CACCAC CATTGCCATTTCCCTAGAGTCTTAC CACCAC CATTGCCACATCCTCCGAGTCTTAC CACCAC CATTGCCACATCCTCCGAGTCTTAC CACCAC CATTGCCTTATAAATTGAGTCTTAC CACCAC CACTGCACCAACAACCAATCAG CTG
ILN07Pro_FORKY1X3_X3X3Y3	[e]:[y1]:[t^ y3]:[t^ x3]:[t^ x3]:[t^ a3]:{t^*}:[d]	CAG CTGCCCATATTTCGAGTCTTAC CATTGCCCTACCACCTGAGTCTTAC CACCAC CATTGCTTCTATCTTTGAGTCTTAC CACCAC CATTGCATCTATAAACGAGTCTTAC CACCAC CATTGCATCTATAAACGAGTCTTAC CACCAC CATTGCATATTTCCCAGAGTCTTAC CACCAC CACTGCACCAACAACCAATCAG CTG
ILN08Pro_FORKY1X4_X4X4Y4	[e]:[y1]:[t^ y4]:[t^ x4]:[t^ x4]:[t^ a4]:{t^*}:[d]	CAG CTGCCCATATTTCGAGTCTTAC CATTGCCCTACCACCTGAGTCTTAC CACCAC CATTGCTTCTTATATTGAGTCTTAC CACCAC CATTGCTTAACATACAGAGTCTTAC CACCAC CATTGCTTAACATACAGAGTCTTAC CACCAC CATTGCCAACCCATTCGAGTCTTAC CACCAC CACTGCACCAACAACCAATCAG CTG
ILN09Pro_JOINX2X1_X1Y1	[b]:{t^*}[x2 t^]:[x1 t^]:[a1 t^]:[a1]:[c]	CAG CTGTCCTTTATTTCGCTCTTCC CACCAC CATTGCCACATCCTCCGAGTCTTAC CACCAC CATTGCACATTATATCGAGTCTTAC CACCAC CATTGCTATTCACACAGAGTCTTAC CACCAC CATTGCTATTCACACAGAGTCTTAC CATTGCCACCTCCACCCTCCAG CTG
ILN10Pro_FORKX2X1_X1Y1	[e]:[y1]:[t^ y1]:[t^ x1]:[t^ a1]:{t^*}:[d]	CAG CTGCCCATATTTCGAGTCTTAC CATTGCCACATCCTCCGAGTCTTAC CACCAC CATTGCCCTACCACCTGAGTCTTAC CACCAC CATTGCACATTATATCGAGTCTTAC CACCAC CATTGCTATTCACACAGAGTCTTAC CACCAC CACTGCACCAACAACCAATCAG CTG
ILN11Pro_FORKX2X1_X1Y1Y2		CAG CTGCCCATATTTCGAGTCTTAC CATTGCCACATCCTCCGAGTCTTAC CACCAC

		CATTGCCATTTCCCTAGAGTCTTAC CACCAC CATTGCCCTACCACCTGAGTCTTAC CACCAC CATTGCACATTATATCGAGTCTTAC CACCAC CATTGCTATTCACACAGAGTCTTAC CACCAC CACTGCACCAACAACCAATCAG CTG
ILN12Pro_JOINY1X3_X3X3Y3	[b]:{t^*}[y1 t^]:[x3 t^]:[a3 t^]:[a3]:[c]	CAG CTGTCCTTTATTTCGCTCTTCC CACCAC CATTGCCCTACCACCTGAGTCTTAC CACCAC CATTGCATCTATAAACGAGTCTTAC CACCAC CATTGCATATTTCCCAGAGTCTTAC CACCAC CATTGCATATTTCCCAGAGTCTTAC CATTGCCACCTCCACCCTCCAG CTG
ILN13Flip_JOINY1X4_X4X4Y4	[b]:{t^*}[y1 t^]:[x4 t^]:[a4 t^]:[a4]:[c]	CAG CTGGAGGGTGGAGGTGGCAATGGTAA GACTCGAATGGGTTGGCAATG GTGGTG GTAAGACTCGAATGGGTTGGCAATG GTGGTG GTAAGACTCTGTATGTTAAGCAATG GTGGTG GTAAGACTCAGGTGGTAGGGCAATG GTGGTG GGAAGAGCGAAATAAAGGACAG CTG
ILN14Flip_FORK_Y1X4_X4X4Y4		
ILN15Pro_AllRingJoins		ILN3_GBKpro, ILN4_GKO,ILN5_GOB
ILN16PRO_AllRing Forks		ILN06_GBKpro, ILN07_GKO,ILN08_GOB
ILN17Pro_X1e_true		CAG CTGTCCTTTATTTCGCTCTTCC CACCAC CATTGCACATTATATCGAGTCTTAC CACCAC CACTGCACCAACAACCAATCAG CTG
ILN18Pro_X2e_true		CAG CTGTCCTTTATTTCGCTCTTCC CACCAC CATTGCCACATCCTCCGAGTCTTAC CACCAC CACTGCACCAACAACCAATCAG CTG
ILN19Pro_X3e_true		CAG CTGTCCTTTATTTCGCTCTTCC CACCAC CATTGCATCTATAAACGAGTCTTAC CACCAC CACTGCACCAACAACCAATCAG CTG
ILN20Pro_X4e_true		CAG CTGTCCTTTATTTCGCTCTTCC CACCAC CATTGCTTAACATACAGAGTCTTAC CACCAC CACTGCACCAACAACCAATCAG CTG
ILN21Pro_JoinY1X4X4_X4X4X4Y4		CAG CTGTCCTTTATTTCGCTCTTCC CACCAC CATTGCCCTACCACCTGAGTCTTAC CACCAC

		CATTGCTTAACATACAGAGTCTTAC CACCAC CATTGCTTAACATACAGAGTCTTAC CACCAC CATTGCCAACCCATTCGAGTCTTAC CACCAC CATTGCCAACCCATTCGAGTCTTAC CATTGCCACCTCCACCCTCCAG CTG
ILN22Pro_ForkY1X4X4_X4X4X4Y4		CAG CTGCCCATATTTCGAGTCTTAC CATTGCCCTACCACCTGAGTCTTAC CACCAC CATTGCTTCTTATATTGAGTCTTAC CACCAC CATTGCTTAACATACAGAGTCTTAC CACCAC CATTGCTTAACATACAGAGTCTTAC CACCAC **CATTGCTTAACATACAGAGTCTTAC CACCAC** CATTGCCAACCCATTCGAGTCTTAC CACCAC CACTGCACCAACAACCAATCAG CTG
ILN23Pro_JoinX4X3_X1Y1	[b]:{t^*}[x4 t^]:[x3 t^]:[a1 t^]:[a1]:[c]	CAG CTGTCCTTTATTTCGCTCTTCC CACCAC CATTGCTTAACATACAGAGTCTTAC CACCAC CATTGCATCTATAAACGAGTCTTAC CACCAC CATTGCTATTCACACAGAGTCTTAC CACCAC CATTGCTATTCACACAGAGTCTTAC CATTGCCACCTCCACCCTCCAG CTG
ILN24Pro_JoinY3_X4_simp		CAG CTGTCCTTTATTTCGCTCTTCC CACCAC CATTGCTTCTATCTTTGAGTCTTAC CACCAC CATTGCATATTTCCCAGAGTCTTAC CACCAC CATTGCCACCTCCACCCTCCAG CTG
ILN25Pro_ForkY3_X4_simp		CAG CTGCCCATATTTCGAGTCTTAC CACCAC CATTGCTTAACATACAGAGTCTTAC CACCAC CATTGCATATTTCCCAGAGTCTTAC CACCAC CACTGCACCAACAACCAATCAG CTG
ILN26Pro_FORKX2X1_X1Y1Y1	[e]:[y1]:[t^ y1]:[t^ y1]:[t^ x1]:[t^ a1]:{t^*}:[d]	CAG CTGCCCATATTTCGAGTCTTAC CATTGCCACATCCTCCGAGTCTTAC CACCAC CATTGCCCTACCACCTGAGTCTTAC CACCAC CATTGCCCTACCACCTGAGTCTTAC CACCAC CATTGCACATTATATCGAGTCTTAC CACCAC CATTGCTATTCACACAGAGTCTTAC CACCAC CACTGCACCAACAACCAATCAG CTG
ILN27Pro_AllConsJoins		ILN12_GBKpro, ILN13_GKO, ILN23_GOB

ILN28Pro_AllConsForks		ILN07_GBKpro, ILN08_GKO, ILN26_GOB
		CAG CTGCCCATATTTCGAGTCTTAC
		CATTGCCACATCCTCCGAGTCTTAC CACCAC
		CATTGCCCTACCACCTGAGTCTTAC CACCAC
	[e]:[y1]:[t^	CATTGCACATTATATCGAGTCTTAC CACCAC
	y1]:[t^ x1]:[t^	CATTGCACATTATATCGAGTCTTAC CACCAC
	x1]:[t^	CATTGCTATTCACACAGAGTCTTAC CACCAC
ILN29Pro_JoinX2X1_X1X1Y1	a1]:{t^*}:[d]	CACTGCACCAACAACCAATCAG CTG

VITA

Sundipta Rao received her B.S in Biophysics from the University of California, Los Angeles in 2012 and her MSEE in Electrical Engineering from the University of Washington in 2016. Her research interests include dynamic nucleic acid circuits for computation and pattern formation, for which she was a recipient of the National Science Foundation Graduate Research Fellowship. Sundipta is a first-generation American, the daughter of two Indian American immigrants. Outside of research, she is passionate about STEM education and outreach.