Biomolecular and Computational Frameworks for Genetic Circuit Design

by

Alec A.K. Nielsen

B.S. in Bioengineering, Electrical Engineering
University of Washington (2009)

Submitted to the Department of Biological Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Biological Engineering

at the

MASSCHUSETTS INSTITUTE OF TECHNOLOGY

February 2017

Signature redacted

Author.....
Department of Biological Engineering
September 8, 2016

Signature redacted

Certified by.....
Christopher A. Voigt
Professor of Biological Engineering
Thesis Supervisor

Signature redacted

Accepted by.....
Forest M. White
Chair, Department Committee on Graduate Theses

For mom and dad

Preface

The work described in this thesis was the result of collaborations with many talented researchers at MIT, Boston University, the National Institute of Standards and Technology. I am privileged to have worked alongside some of the sharpest and most creative biological engineers, and I am especially grateful to Prof. Christopher Voigt and the extremely talented members of his group.

Parts of Chapter 1 were published as a review in *Current Opinion in Chemical Biology* in December of 2013. I co-authored this review with fellow graduate student Thomas Segall-Shapiro, and it provides a useful overview of recent advances in genetic parts and circuit design.

Parts of Chapter 2 were published in *Nature Chemical Biology* in February of 2014. Brynne Stanton initiated this project as a postdoc and took me on as a lowly rotation student. She planned and developed many aspects of this work, including the repressor-binding operator discovery microarray assay central to the project – the details of which are not provided in this thesis, but are available in the original publication. I would also like to acknowledge Alvin Tamsir for his insights into the practicalities of genetic circuit design, and for laying the foundation for much of the work present in this thesis.

Chapter 3 was published in *Molecular Systems Biology* in November of 2014. I started this project on the heels of Wendell Lim's group's seminal publication, "Repurposing CRISPR as an RNA-guided platform for sequence-specific control of gene expression". I am grateful to Samira Kiani, Fahim Farzadfard, Sam Perli, and Lior Nissim in Prof. Timothy Lu's and Prof. Ron Weiss's groups for their early contributions to CRISPR-based genetic regulation and for fruitful discussions.

Chapter 4 was published in *Science* in April of 2016. It represented the culmination of several years of effort, and would not have been possible without the contributions of

Bryan Der, Jonghyeon Shin, Prashant Vaidyanathan, and Doug Densmore. I would also like to thank David Ross, Vanya Paralanov, and Elizabeth Strychalski at the National Institute for Standards and Technology for working tirelessly to perform the single molecule FISH experiments under the pressure of a tight deadline and remote interactions with the rest of the paper's authors.

The work done in this thesis was thesis supported in part by the Defense Advanced Research Project Agency (DARPA CLIO N66001-12-C-4018), the Office of Naval Research (ONR) Multidisciplinary University Research Initiative (MURI Grant N00014-13-1-0074; Boston University MURI award 4500000552), Life Technologies, the US National Science Foundation Synthetic Biology Engineering Research Center (SynBERC), the National Defense Science and Engineering Graduate (NDSEG) Fellowship, 32 CFR 168a, and the Siebel Scholars Foundation.

Acknowledgements

First and foremost, I would like to thank my advisor Chris Voigt. He not only provided an incredible research environment, but gave me the freedom to pursue new ideas and opportunities to collaborate, travel, give talks, and meet interesting people. His remarkable intuition and creativity for great science, when combined with his inimitable sense of humor, made for inspiring, entertaining, and unforgettable mentorship. It is a rare privilege to work alongside your heroes, and it's been one of the most rewarding experiences of my life.

I would also like to thank my thesis committee members. Ron Weiss, whose PhD thesis is the closest thing to a spiritual predecessor and guiding light for my own, was the first synthetic biologist whose research I was exposed to as an undergrad. His work continues to inspire, and I'm honored to have interacted with him over the course of my PhD. My thesis chair and head of the Biological Engineering department, Doug Lauffenburger, is a scientific heavyweight who has been an excellent resource for research insight and career advice. He has also managed to create one of the most collaborative and respected departments on the planet. BE grad students seem happier than most, and that is in large part due to Doug's tireless efforts to engage students and improve the system in which they inhabit.

The other grad students and postdocs of the Voigt lab are unreasonably talented, and I had the good fortune of being trained by some of the best synthetic biologists in the world. I would like to extend a special thanks to those who I consider technical mentors—Brynne Stanton, Mike Smanski, and Brian Caliando. Working with others in the area of genetic circuits was a daily pleasure, including Chunbo Lou, Bryan Der, Jonghyeon Shin, Thomas Gorochowski, YongJin Park, Thomas Segall-Shapiro, Adam Meyer, Jenn Brophy, Felix Moser, Shuyi Zhang, and Lauren Woodruff. Thanks to my skilled collaborators at Boston University (Doug Densmore and Prashant

Vaidyanathan), NIST (David Ross, Vanya Paralanov, and Elizabeth Strychalski), and MIT Media Lab (Will Patrick, Steven Keating, Neri Oxman, and David Kong).

Grad school wouldn't have been anywhere near as fun without the great group of friends I was able to surround myself with. In particular, thanks to the band members of The Craigslist Killers and friends in the social groups Serenity of the Lewd and The OB, including Brian Bonk, Raja Srinivas, James Weis, Rob Kimmerling, Vyas Ramanan, Andrew Warren, Nathan Stebbins, and Baris Sevinc. I laugh nonstop whenever I'm with you guys. I'm also glad that despite being spread out across the country, my college friends have all managed to stay close: Ben Guthrie, Chris Hoyt, Kevin Grimes, Brad Zimmerman, Alex Martin, Earl Bryan Thompson III, Jeff Blume, Derek Dolan, Dave Mantell, Tom Petry, and Kevin Merritt. I hope our antics never cease. My roommates were another huge source of entertainment and distraction from the stresses of grad school, including those from The Burrow at 18 Kinnaird St. (Seb Eastham, Jenny Kay, Thomas Segall-Shapiro, Kendal Spires, Josh, and the rotating sixth roommate. Please forgive me for Kent.), and The Manor at 23 Tufts St. (Daniel Rothenberg, Conor McClune, Marcus Parrish, and Gus. I haven't seen the yodeling pickle in four months and it's making me increasingly uneasy.) Also thanks to Kathy Lee and Sarah Luppino, who gave love and support when I needed it most.

Lastly, thank you to my family. I love you so much. That I was born to parents so encouraging and loving is the luckiest aspect of my existence. Thank you for helping me to pursue my dreams. To my mother, the kindest and most caring person I know, I owe everything to you having raised me. To my father, who imbued me with a fascination for the natural world, you are deeply missed. Thanks to Sandy, Kim, and Kraig for being a wonderful addition to my family, and continuing to play a huge role in my life. And thanks to my little brother, Nate—I couldn't ask for a better bro. You inspire me, I look up to you, and I love you.

Biomolecular and Computational Frameworks for Genetic Circuit Design

by

Alec A.K. Nielsen

Submitted to the Department of Biological Engineering
on September 8, 2016 in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Biological Engineering

Abstract

Living cells naturally use gene regulatory networks termed "genetic circuits" to exhibit complex behaviors such as signal processing, decision-making, and spatial organization. The ability to rationally engineer genetic circuits has applications in several biotechnology areas including therapeutics, agriculture, and materials. However, genetic circuit construction has traditionally been time- and labor-intensive; tuning regulator expression often requires manual trial-and-error, and the results frequently function incorrectly. To improve the reliability and pace of genetic circuit engineering, we have developed biomolecular and computational frameworks for designing genetic circuits.

A scalable biomolecular platform is a prerequisite for genetic circuits design. In this thesis, we explore TetR-family repressors and the CRISPRi system as candidates. First, we applied 'part mining' to build a library of TetR-family repressors gleaned from prokaryotic genomes. A subset were used to build synthetic 'NOT gates' for use in genetic circuits. Second, we tested catalytically-inactive dCas9, which employs small guide RNAs (sgRNAs) to repress genetic loci via the programmability of RNA:DNA base pairing. To this end, we use dCas9 and synthetic sgRNAs to build transcriptional logic gates with high on-target repression and negligible cross-talk, and connected them to perform computation in living cells. We further demonstrate that a synthetic circuit can directly interface a native E. coli regulatory network.

To accelerate the design of circuits that employ these biomolecular platforms, we created a software design tool called Cello, in which a user writes a high-level functional specification that is automatically compiled to a DNA sequence. Algorithms first construct a circuit diagram, then assign and connect genetic "gates", and simulate performance. Reliable circuit design requires the insulation of gates from genetic

context, so that they function identically when used in different circuits. We used Cello to design the largest library of genetic circuits to date, where each DNA sequence was built as predicted by the software with no additional tuning. Across all circuits 92% of the output states functioned as predicted. Design automation simplifies the incorporation of genetic circuits into biotechnology projects that require decision-making, control, sensing, or spatial organization.

Thesis supervisor: Christopher A. Voigt
Title: Professor
Department of Biological Engineering

Contents

Chapter 1

1 Introduction

Cells naturally control gene expression using a variety of RNA, protein, and DNA-modifying regulators (Ptashne, 1986b; Ideker *et al*, 2001; Alon, 2007). It was recognized early that interactions between these regulators could lead to computational operations that are analogous to electronic circuits (Monod & Jacob, 1961; Ptashne, 1986a; McAdams & Shapiro, 1995). Genetic engineers have attempted to build synthetic circuits that would implement artificial programs of gene expression. This could have a revolutionary impact on biotechnology, such as programming bacteria to individually respond to transient conditions in a bioreactor (Moser *et al*, 2012a), designing therapeutic cells to sense and respond to diseased states within the human body (Ruder *et al*, 2011; Chen & Smolke, 2011; Ghosh *et al*, 2012; Huh *et al*, 2013; Hasty, 2012), or smart plants that can respond to changing conditions in the environment (Bowen *et al*, 2008). However, building synthetic circuits remains one of the greatest challenges in the field, where even simple circuitry is labor intensive to build and lacks the performance of its natural counterparts. As a result, synthetic genetic circuits have been slow to appear in practical applications (Purnick & Weiss, 2009a).

There are several reasons why genetic circuit design has been challenging compared to other areas in genetic engineering. First, functional genetic circuits require precise tuning in the expression levels of their component regulators (Ang *et al*, 2013). This is less essential when engineering cells to make small molecules or individual proteins, where genes tend to be maximally expressed. Second, regulators are prone to being toxic and, even when slight, can inhibit growth and lead to evolutionary instability and a reduction in performance. Third, the regulatory interactions comprising a circuit all occur within the cell and crosstalk between them or with the host can impact circuit behavior (Andrianantoandro *et al*, 2006). Fourth, there are few design rules for the systematic improvement of circuit performance (speed, dynamic range, robustness, and cell-to-cell variability). Finally, the physical construction of circuits requires the assembly of many parts, which until recently, has been technically challenging (Czar *et al*, 2009a; Gibson *et al*, 2009; Engler *et al*, 2009). Often, these parts appear in genetic contexts that are different than that for which they were characterized and this can lead to interference (Moser *et al*, 2012a).

In this review, we focus on recent advances in synthetic circuit design for bacteria. There have been other reviews looking at circuit design for eukaryotes and higher organisms (Keasling, 2008; Purnick & Weiss, 2009a; Ellis *et al*, 2009; Greber & Fussenegger, 2007; Wang *et al*, 2013b; Mukherji & van Oudenaarden, 2009). In section 1.1, we describe new approaches to identifying how to assemble and tune regulators to produce a desired circuit function. In section 1.2, we describe how the toolbox of regulators has expanded, both in increasing the number of characterized regulators from different families, as well as the discovery of new biochemistries that can be harnessed for circuit design. Finally in section 1.3, we review new approaches to obtain precision expression control and its potential impact on building sophisticated circuitry.

1.1 Advanced circuit designs

To date, most of the genetic circuits that have been constructed are so small that there has been little need to utilize advanced concepts or algorithms in their design. As they get more sophisticated however, it will become more difficult to identify a pattern of regulatory interactions that can produce a desired function. To this end, approaches for digital and analog circuit design from electrical engineering have begun to be applied, and are described below. Realizing these designs requires that regulators be functionally connected. This will require better control over their response functions (how each regulator converts different levels of input signal to output signal) as well as handling of other circuit characteristics such as retroactivity and instability. New approaches for these concerns also reviewed in this section.

1.1.1 Layered Digital Circuits

Digital circuits produce signals at discrete levels (most commonly, high and low or 1 and 0), as opposed to operating in a continuous range. Their advantage is in their designability; there are many design tools that can abstract a desired circuit function into a large assembly of logic gates (Clancy & Voigt, 2010). This comes at a cost of size and power requirements. Many more digital gates may be needed to produce a computational function than what would be required if continuous variables were allowed. In terms of genetic circuits, this manifests as more DNA, regulators, and energetic resources (Lu *et al*, 2009; Qian & Winfree, 2011; Moon *et al*, 2012a).

Many genetic circuits have been built that produce Boolean logic functions or 'logic gates' (Tamsir *et al*, 2011a; Anderson *et al*, 2007; Guet *et al*, 2002). Note that while these are often described as being digital, all of these circuits exhibit analogue

features (so-called fuzzy logic), where there is a continuous change in output. This can be used as the basis for the construction of analog circuitry (next section).

If genetic logic gates are designed to have inputs and outputs that have the same signal, they can be layered to produce more complex computational operations. In practice, this has been on the level of transcription, where the inputs and outputs are promoters. This approach is modular, but it is also slow, with each layer requiring a step of transcription and translation with a timescale of 20 minutes (Hooshangi *et al*, 2005). Further, if one of the signals skips a layer, this can produce a fault where the output is transiently incorrect. Such faults have been exploited in the construction of pulse-generating genetic circuits, in the form of incoherent feed-forward loops (Mangan & Alon, 2003; Entus *et al*, 2007; Basu *et al*, 2004).

There have been several studies to layer logic gates to produce more complex functions. This is closely related to work to build cascades through the connection of gates in a linear series (Hooshangi *et al*, 2005; Pedraza & Oudenaarden, 2005; Rosenfeld *et al*, 2005). As a proof-of-principle, a 4-input AND gate was built by layering three 2-input AND gates along with additional layers that contain the 4 sensors and an output (Moon *et al*, 2012a) (Figure 1a). It has also been shown that a set of orthogonal NOR gates can be layered to form different logic operations by permuting the input and output promoters to reproduce different wiring diagrams (Stanton *et al*, 2013). Both of these examples perform relatively simple computations that could be designed by hand and required more gates than the minimal set that could be imagined to generate each of these functions. It would require significantly larger circuits to realize the benefits of digital gates and computational design automation (Beal *et al*, 2012; Bilitchenko *et al*, 2011; Clancy & Voigt, 2010).

1.1.2 Analog Circuits

Analog circuits operate with continuous signals. In electronic circuits, they are used when there are limitations in the number of components or power that can be used (*e.g.*, in medical devices) (Sarpeshkar, 2010). This comes at a cost of designability, where each circuit has to be individually designed and simulated, which limits the size and flexibility of the circuits. In practice, every genetic circuit – natural or synthetic – is analog to some degree and this needs to be accounted for in their design. The question is to what extent the design of genetic circuits can benefit from the principles used for building analogue electronic circuits.

The value in considering analogue circuit design was recently demonstrated in work by Lu and co-workers (Daniel *et al*, 2013a). In this work, analog circuits were implemented to solve mathematical functions that would otherwise require many digital gates, including logarithm and power-law functions, and continuous addition and division (Figure 1b). A circuit was built that generates a wide dynamic range response function using a positive feedback loop and a means to titrate away the activator. This design computes a logarithm and the introduction of a second positive feedback loop produces a circuit that computes log-domain addition of two inducers. A log-domain division circuit was further engineered by having the two feedback loops compute the ratio between the two inducers. Remarkably, all of these arithmetic functions could be computed using only two transcription factors.

1.1.3 Recombinase-based Memory and Logic

Logic gates based on transcription factors often exhibit analogue features, with high off-states and graded switch transitions. In contrast, more digital switches can be built using recombinases that catalyze a sequence-specific change the orientation of a unit of DNA, where each orientation corresponds to a different signal level. Recombinases

have been used as the basis for a number of synthetic circuits (Moon *et al*, 2011; Ham *et al*, 2008, 2006) and have been layered to form a cascade (Friedland *et al*, 2009). Previously, the recombinases used were either irreversible (where the inversion is unidirectional) or reversible (where the same recombinase catalyzes both directions). Recently, a rewriteable switch has been built based on a system where an integrase catalyzes the switch in one direction and an integrase/excisionase pair catalyzes the reverse reaction (Bonnet *et al*, 2012). This is a significant improvement in that it allows the signal to both hold permanently and be able to switch back to the initial state.

Multiple recombinases have been built into circuits that function as "memory logic" devices, where the rearrangement of DNA is conditional to two input inducers (Siuti *et al*, 2013; Bonnet *et al*, 2013). In one paper, two recombinases (Bxb1 and phiC31) irreversibly invert promoters, unidirectional terminators, and GFP. Only when transcription initiation, termination, and GFP are in the correct orientation is fluorescent output seen, and memory circuits for all irreversible 2-input logic functions were successfully constructed (Siuti *et al*, 2013) (Figure 1c). In a second paper, two recombinases (Bxb1 and TP901-1) irreversibly invert or excise terminators and promoters to implement six irreversible 2-input logic functions (Bonnet *et al*, 2013).

There are several advantages to this approach in building logic gates. The gate response has a larger dynamic range that is easier to connect to downstream gates and the signal levels are more easily distinguishable (two different DNA orientations versus the presence or absence of a regulator). The hold state is also permanent, surviving over many generations and even after cell death. These gates could be layered as with those based on transcription factors and the size of the DNA per gate is about the same. However, there are two disadvantages with using recombinases. First, they are not true logic in that a transient but temporally separate induction of the two signals

leads to a permanent change in the output. Also, they tend to be slow, with each layer requiring between up to eight hours (Moon *et al*, 2011) to complete.

1.1.4 Control of the Response Function

Building genetic circuits by connecting logic gates requires that their response functions match, that is, the output range of upstream gates matches the input range of downstream gates. In Section 1.3, we look at methods based on changing the expression levels of the regulators. Here, we look at new approaches to change the shape of their response functions, including the basal level of the off state, dynamic range, and cooperativity (Ang *et al*, 2013). These approaches could be applied to digital or analogue circuits and those based on different regulator families.

One of the biggest practical problems in constructing complex circuits is that the basal level of activity from the off-state of regulators ("leak") can often be sufficient to trigger the next layer of a circuit. This has been difficult to control, but there are several promising new approaches. First, leakage can be minimized using riboregulation to suppress translation. This has been used effectively to minimize the uninduced activity of toxic proteins (Callura *et al*, 2010). A similar approach is to use small RNAs (sRNAs) to bind an RNA chaperone Hfq and the target mRNA to both inhibit translation and also target the mRNA for destruction by RNase E (Aiba, 2007). Second, in natural prokaryotic genetics, leakiness is controlled using 5'-terminators that come directly after a promoter to attenuate transcription (Naville & Gautheret, 2010). A similar strategy could be used in synthetic systems to reduce leaky transcription of mRNA.

Increasing the nonlinearity of a response curve can also be important in circuit design. Nonlinearity occurs naturally via cooperativity, but this can be challenging to engineer *de novo*. An easier approach is to incorporate interactions that sequester the

regulator at the DNA (Lee & Maheshri, 2012b), RNA, or protein level (Buchler & Louis, 2008). The level of effector must exceed the binding capacity of the competitively sequestering partner to pass the threshold, bind the cognate partner, and take action. Examples of competitive binding partners include anti-sigmas sequestering sigma-factor (Chen & Arkin, 2012a), sRNA sequestering mRNA (Levine *et al*, 2007), and decoy operators sequestering DNA-binding proteins (Lee & Maheshri, 2012b).

1.1.5 Buffering retroactivity

Genetic gates within a cell inevitably share resources; for example, they utilize the host RNA polymerase and ribosomes. Shared resources can cause coupling between gates that are otherwise unconnected and can cause a downstream gate to affect the behavior of an upstream one (Del Vecchio *et al*, 2008; Jayanthi *et al*, 2013). It has been experimentally shown that increasing the number of downstream operators can impact the response time and threshold of a gate (Jayanthi *et al*, 2013). Since the size of genetic circuits has been small, retroactivity has not yet emerged as a significant problem, but this is expected to worsen ascircuit size increases and when the output of a gate is connected to many downstream circuits or actuators ("fan-out").

An interesting approach for insulating modules from retroactivity has been mathematically investigated (Del Vecchio *et al*, 2008). While the general solution is well known from control theory (high input amplification and high negative feedback), the authors of this study looked at specific biological mechanisms that could fill the role. Surprisingly, simple circuit modifications like a non-leaky input promoter (for high input amplification) and rapid degradation of the transcription factor (for negative feedback) effective insulate retroactivity in mathematical models.

1.1.6 Engineering evolutionary stability

The selection of a particular circuit topology and genetic implementation also impacts its evolutionary stability. Recent work has begun to illuminate the design choices that lead to instability, which can in turn be used to guide future designs. It has been observed that if the resting state of a gate requires the expression of an active regulator, the gate is more evolutionarily unstable (Canton *et al*, 2008a). Thus, selecting an architecture that minimizes the number of regulators that have to be expressed at a given time could increase stability (Sleight & Sauro, 2013). The modes by which cells reject circuits are also becoming better understood. Unsurprisingly, the use of plasmids and the repetition of DNA sequences in a design leads to instability (Sleight & Sauro, 2013; Sleight *et al*, 2010; Moser *et al*, 2012a). The re-use of strong double terminators has been found to be particularly bad and their diversification dramatically increases the number of generations before a circuit is lost due to homologous recombination (Chen *et al*, 2013; Sleight *et al*, 2010).

Additionally, a recent large-scale effort to ascertain which heterologous genes are toxic in E. coli has begun to shed light on how synthetic constructs affect host fitness (Kimelman *et al*, 2012). By examining cloning gaps in over 9.3 million sequencing clones from 393 microbial genomes, more than 15,000 were found to have toxic expression products. Through subsequent validation and analysis, new restriction enzymes, toxin-antitoxic systems, and toxic small RNAs were discovered. Toxic DNA-binding motifs were also observed that likely titrate away DnaA and inhibit normal replication. This wealth of information about toxic DNA elements is a fantastic resource for predicting how genes may affect host systems, and could guide which regulators can be effectively used in circuits.

1.2 Classes of regulators

The last few years has seen an explosion in the number of well-characterized regulators that are available for building genetic circuits. Before this, there were relatively few that were available (e.g., LacI, TetR, AraC, and CI) and these were re-used in many designs. A goal has been to expand the number of variants within each family that are orthogonal, that is, do not cross react with each other such that they can be used together in a circuit (Lucks *et al*, 2008; Rao, 2012). This has been achieved via two approaches. First, bioinformatics and whole gene DNA synthesis has been used to access regulators from the sequence databases ("part mining") (Bayer *et al*, 2009). Second, families of regulators have been characterized that are conducive to the rational design of orthogonal sets (zinc finger proteins, TALEs, and CRISPR-Cas9). Computational methods have played a role both to predict the orthogonality of regulators identified in databases as well as in structure-guided design. Collectively, this has resulted in 100s of regulators that could theoretically be used together in a single large circuit in a bacterium.

Despite efforts to standardize and collect data surrounding biological parts (Canton *et al*, 2008b; http://parts.igem.org/), the majority of the information is buried in individual papers, making direct comparisons difficult. In Table 1, we show data comparing 15 common regulator families along with the properties of each family. For the families, the number of characterized orthogonal regulators are shown (with a metric of crosstalk), along with the size in basepairs of the potential gate, and dynamic range that has been achieved with that regulator. Note that the table focuses on bacteria and there is more data for some families in eukaryotic cells.

1.2.1 Protein Regulators of Transcription

Proteins that directly bind DNA to regulate transcription make up the majority of the regulatory parts available for use in bacteria (Figure 2a). One way in which proteins can regulate transcription is by initiating transcription at promoters. The native *E. coli* RNAP can be directed to new promoters by expressing sigma factors from other organisms (Chen & Arkin, 2012a; Rhodius *et al*). A large set of orthogonal sigma factors has been generated through part mining, in which sigma factors from many organisms were synthesized and their activities characterized (Rhodius *et al*). Alternatively, the phage RNAP from T7 is often used and the promoter specificity of this polymerase has been changed through rational design and part mining (Temme *et al*, 2012; Raskin *et al*, 1993; Shis & Bennett, 2013) as well as random mutagenesis (Chelliserrykattil *et al*, 2001; Esvelt *et al*, 2011).

Activators upregulate transcription by binding to a promoter to recruit RNAP. Classically, there are a number of natural activator proteins that have been used in genetic engineering, such as λ *cI* and *luxR*. A small library of *crp* activators was built by using bioinformatics to direct mutations at residues responsible for operator specificity (Desai *et al*, 2009). Part mining has been applied to identify activators that require a second chaperone protein for activity and this has been used as the basis for building AND gates (Moon *et al*, 2012a).

Repressors block transcription by blocking the binding or progression of RNAP. Recently, there have been efforts to increase the number of orthogonal repressors available for circuit design. To expand the LacI family, mutations were made to specific DNA residues in the binding site and DNA binding residues in the protein and a set of orthogonal repressors was selected (Zhan *et al*, 2010). Part mining has been applied to expand the number of available TetR homologues and this led to the identification of an orthogonal set of 16 repressors (Stanton *et al*, 2013).

There are several modular classes of transcription factors that have modular protein structures that facilitate their engineering to target particular DNA sequences. Zinc finger proteins (ZFPs) and transcription activator like effectors (TALEs) have such a structure and have been particularly successful in being used in eukaryotic cells (Desjarlais & Berg, 1992; Moscou & Bogdanove, 2009; Boch *et al*, 2009a; Beerli & Barbas, 2002a; Morbitzer *et al*, 2010a; Garg *et al*, 2012b). It has been surprisingly difficult to get these regulators to work in bacteria, but there examples of ZFPs being used as activators and (Lee *et al*, 2008) a TALE as a repressor in *E. coli* (Politz *et al*, 2013a).

1.2.2 RNA Regulators of Translation

RNA parts that regulate translation take advantage of fact that RNA base pairing follows a simple code that is computationally predictable (Markham & Zuker, 2008) (Figure 2b). Two parts families of this type are variants of riboregulators, which alter the accessibility of the ribosome binding site (RBS) controlling translation initiation. RNA-IN/OUT parts consist of a modified natural system (Kittle *et al*, 1989) in which an RNA molecule base pairs to the 5' end of an mRNA (including the RBS) such that the ribosome cannot initiate translation (Mutalik *et al*, 2012a). The orthogonal set of these regulators was increased through computational design and experimentally confirmed. A second part family uses trans-activating RNAs that work by disrupting a secondary structure that blocks the RBS by default, leading to translational activation (Isaacs *et al*, 2004; Callura *et al*, 2010, 2012). Finally, it has been shown that the expression of modified 16S RNA that has been engineered to bind a non-canonical Shine Delgarno sequence can recruit ribosomes and this has been used as the basis to build gates (Chubiz & Rao, 2008; Rackham & Chin, 2005; An & Chin, 2009).

When designing gates, the challenge with using RNA that acts on the level of translation is that it is difficult to convert an RNA input to an RNA output of the same form. Therefore, the resulting gates are not layerable. An approach to this problem is to use a *cis* element that converts translation into transcription (Liu *et al*, 2012). This component utilizes a modified sequence from the *tnaC* operon (Gong *et al*, 2001) and makes transcription of a downstream region dependent on translation of a short peptide, effectively linking the two. It has been successfully applied to both classes of riboregulators.

1.2.3 RNA Regulators of Transcription

An exciting development over the last year has been in the development of RNA-based systems that can directly regulate transcription by behaving as a repressor (Figure 2b). This is based on Cas9, which is a protein that uses a small guide RNA to target a DNA sequence as part of the CRISPR/Cas bacterial immunity system (Horvath & Barrangou, 2010). Normally, Cas9 functions as a nuclease and cleaves DNA, but it was shown that if the nuclease activity is mutated, then the complex blocks RNAP (Figure 1d). This either can be used as a repressor or as an activator by fusing an activation domain to Cas9 (Qi *et al*, 2013; Bikard *et al*, 2013). An advantage of this system is the potential to design vast numbers of orthogonal regulators by building guide RNAs that target different "operator" sequences. The cross reactions that may result from this approach are just beginning to be characterized and understood (Fu *et al*, 2013a; Hsu *et al*, 2013a). Once it is shown that the Cas9-based systems can be layered, this will become a powerful toolbox for circuit engineering. However, a practical challenge with using this system is the acute toxicity of Cas9 when expressed in many organisms.

A more developed approach for RNA to control transcription is based on the PT181 attenuation system (Takahashi & Lucks, 2013; Lucks *et al*, 2011; Brantl & Wagner, 2000). When an antisense RNA is present and binds a target sequence on a transcript RNA, the nascent transcript folds into a transcriptional terminator and attenuates the message. This part family was expanded both through both part mining and random mutagenesis, including utilizing some of the orthogonal RNA pairs from the RNA-IN/OUT system. These attenuators have been shown to be fully compostable into cascades and logic gates (Lucks *et al*, 2011).

1.2.4 Proteins that Modify DNA

Thus far, DNA modification has been built in bacteria by using recombinases to invert segments of DNA (Figure 2d). This mechanism is commonly found in bacteria, phages, and mobile genetic elements, providing a diversity of natural parts to exploit in synthetic systems (Hirano *et al*, 2011).

The current parts for engineering DNA flipping are all natural recombinases, which vary in a number of ways. The most commonly used recombinases in genetic engineering are the simple tyrosine recombinases Cre and Flp (Nagy, 2000). Tyrosine recombinases are bidirectional – they can flip the region between their recognition sites in both directions, leading to an even distribution of orientations. Additionally, if their recognition sites are oriented in the same direction, they catalyze DNA excision. These two recombinases have been used in bacteria, and shown to function properly and orthogonally (Friedland *et al*, 2009). The invertases FimB and Hin have also been used together in a bacterial system (Ham *et al*, 2008). These invertases are similarly bidirectional, but they lack the capability for excision. Finally, phage integrases are a class of recombinases that catalyze unidirectional flipping – leading to the accumulation of a specific DNA orientation (Groth & Calos, 2004). Three integrases

(BxBI, φC3, and TP901-1) have been characterized and used to build bacterial systems (Siuti *et al*, 2013; Bonnet *et al*, 2013). Additionally, a number of these integrases have matching excisionases, which allow for the reversal of DNA flipping (Groth & Calos, 2004; Bonnet *et al*, 2012).

In addition to part mining natural recombinases, there has been progress in generating new families of recombinase parts through modifications. One such approach was to iteratively make mutations to the DNA binding region in the recombinase and select for proteins with new specificities (Santoro & Schultz, 2002; Gaj *et al*, 2011). Another promising method is to generate new parts by creating recombinase fusions with zinc-finger and TALE DNA binding domains (Mercer *et al*, 2012).

Finally, while current efforts have focused on recombinases, other mechanisms of DNA modification may also hold promise for building circuitry. Recently, there have been successful efforts to selectively methylate DNA in bacteria (Chaikind *et al*, 2012), and mammals (Konermann *et al*, 2013) using modular zinc-finger and TALE designs. An *in vivo* means of reading methylation state would open up these parts for use in gene circuits. This seems feasible, as bacteria are known to contain many regulatory systems that respond to DNA methylation (Løbner-Olesen *et al*, 2005; Casadesús & Low, 2006).

1.3 Precision gene expression

Building, tuning, and connecting genetic circuits require the ability to engineer precise changes in gene expression. Further, when gates are combined to build a complex circuit, the genetic context changes, which can impact their function (Lou *et al*, 2012a). There have been many recent advances in the development of "tuning

knobs" that allow for the fine-tuned control of transcription and translation (Figure 3). These can take the form of part libraries or computational tools. Further, insulator parts have been developed that decouple the contribution of various parts to expression. This has led to a redefinition of the classical expression cassette.

1.3.1 Tuning knobs for expression

Promoters. Libraries of constitutive promoters for different species have been built by mutating the -10 and -35 RNAP binding regions of the promoter or the region affecting DNA melting (-10 to +2) (Kosuri *et al*, 2013; Mutalik *et al*, 2013b; Mey *et al*, 2007; Jensen & Hammer, 1998; Rud *et al*, 2006; Seghezzi *et al*, 2011). Advances in oligonucleotide synthesis have enabled these libraries to become very large. For example, >10,000 combinations of promoters and 5-UTR's were built from a pooled oligonucleotide library and screened by combining cell sorting and deep sequencing (flow-seq) (Kosuri *et al*, 2013). Computational models of promoters have also been developed that are based on the free energy of RNAP binding to the -10/-35 sites and promoter melting (Rhodius *et al*, 2012; Brewster *et al*, 2012). These show promise in predicting promoter strength, however, a complete model that balances all contributions has yet to be built.

Ribosome Binding Sites. The RBS is a part that is relatively simple to tune to achieve different expression levels. As a result, it has been broadly applied to tuning response functions for building genetic circuits (Stanton *et al*, 2013; Moon *et al*, 2012a; Chen *et al*, 2012; Gardner *et al*, 2000a; Basu *et al*, 2004; Egbert & Klavins, 2012). The ribosome makes contacts with the Shine-Delgarno sequence and start codon and binding is influenced by RNA base-pairing, the spacing between these regions and the mRNA secondary structure. The RBS Calculator is a computational tool based on a biophysical model that balances these contributions (Salis *et al*, 2009; Voigt, 2011).

There are additional terms that influence the strength of the RBS and several of these, including the role of the standby site, have been characterized and incorporated into new versions of the software (de Smit & van Duin, 2003; Voigt, 2011). There is much to be learned from non-canonical RBSs (Boni *et al*, 2001) including leaderless RNAs (Laursen *et al*, 2005) and a better understanding of these processes could improve the model. Libraries of 5'-UTRs that include the RBS have been measured (Kosuri *et al*, 2013; Mutalik *et al*, 2013b). Additionally, a recent technique to tune RBS strength using hypermutable sequence repeats between the Shine-Dalgarno region and start codon was used to explore expression parameters for a bistable switch (Egbert & Klavins, 2012).

Terminators. During transcription, mRNA is released when the RNAP reaches a terminator. Terminators are important in circuit design for three reasons. First, they offer a means to tune expression by modulating read-through and could potentially decrease leaky expression. Second, many terminators are required when gates are combined to build circuits. These circuits can have many transcription units, each of which needs strong termination to avoid interference with other circuit elements. The terminators have to be sequence diverse to avoid recombination. Finally, the recombinase-based memory circuits utilize unidirectional terminators as a core part of their design. To address these needs, there have been several major efforts to use part mining to build large libraries of terminators gleaned from the genomes of bacteria (Cambray *et al*, 2013; Chen *et al*, 2013).

Origins. To finely control plasmid replication beyond the standard plasmid systems, tunable-copy-number plasmids can be used. Increasing the expression of trans-acting replication factors (repA and pir) increases the plasmid copy number (for ColE2 and R6K origins, respectively). Using this system, a library of "DIAL" strains that constitutively express replication factors from the genome has been constructed that

can yield between 1 and 250 plasmid copies by transforming the corresponding plasmids into the different strains (Kittleson *et al*, 2011).

1.3.2 Insulators to buffer the impact of genetic context

Part function is impacted by their genetic context, in other words the sequences of the neighboring parts (Lou *et al*, 2012a; Qi *et al*, 2012; Davis *et al*, 2011; Mutalik *et al*, 2013a; Kosuri *et al*, 2013). In turn, this can impact the response function of the entire genetic circuit. Context effects can have two forms. First, there is a direct interference of one part type on another. For example, the strength of a RBS is influenced by the promoter and the first codons of the expressed gene. Second, when two parts are combined a new function can appear at their interface. For example, promoters have been inadvertently constructed by the assembly of two parts containing an LVA-degradation tag, a DNA barcode, and a BioBrick scar (Yao *et al*, 2013). To overcome this issue, insulator parts have been developed to diminish the effect of genetic context. Figure 3 shows a conceptual re-visiting of the expression cassette, where insulators are strategically placed between key parts (Mutalik *et al*, 2013a).

The first insulators are based on bidirectional terminators, which flank the expression cassette to reduce transcriptional read through in or out of the expression cassette (Chen *et al*, 2013). Typically, the promoters that are used in synthetic biology are too small and only capture the -35 and -10 regions. This can cause the promoter to have different strengths depending on the up and downstream sequence. Longer promoters should be used that at least encompass the UP element (-35 to -64) that binds the ⊠-subunit of RNAP (Rhodius *et al*, 2012; Estrem *et al*, 1998). Taking this further, it has been shown that the addition of upstream sequences (up to -105) and

downstream sequences (down to +55) will further insulate promoter activity (Davis *et al*, 2011).

The transcription start site of promoters is not always known and a "promoter part" is rarely annotated to end at the +1 position. This can be further complicated by the observation that there is sometimes a distribution of mRNA produced from the same promoter and single mutations to the promoter can change the start site. The RBS is particularly sensitive to changes in the 5'-UTR, even by a single nucleotide. Both self-cleaving ribozymes (Lou *et al*, 2012a) and CRISPR RNA-processing (Qi *et al*, 2012) have been used to physically cut and detach the variable 5'-UTR from the mRNA, thereby shortening and making constant the 5'-mRNA context. These tools are particularly important when combining gates to build a circuit, where the promoter inputs and output of each gate occur in new contexts.

A bicistronic RBS sequence has been shown to reduce the impact of the secondary structure of the 5'-UTR on the RBS (Mutalik *et al*, 2013a). A small 'leader' peptide-coding sequence with its own RBS is positioned upstream from the gene of interest such that the peptide sequence overlaps the RBS for the downstream gene and the peptide's stop codon ends at the gene of interest's start codon. The natural helicase activity of ribosomes loaded at the first RBS unfolds the mRNA secondary structure near the second RBS that controls gene expression, decoupling the translation initiation rate of the second RBS from the downstream coding sequence.

Lastly, the variability in gene expression when coupled with various terminators can be reduced by encoding an RNase III site in the 3'-UTR of the mRNA (Schmeissner *et al*, 1984). Post-transcriptional processing of mRNAs by RNase III standardizes the 3'-end of the mRNA, such that the sequence and secondary structure of the cleaved RNA can no longer contribute to mRNA stability and degradation.

1.4 Discussion

Genetic circuit design is at an inflection point regarding the size and sophistication of computational operations that can be implemented within living cells. The first phase of the field involved the construction of individual circuit functions (*e.g.*, an oscillator or a gate) by piecing together the necessary biochemistries. For an extended period, the complexity of these circuits remained relatively flat as the rules for composing a circuit were explored (Purnick & Weiss, 2009a). After this, there have been tedious efforts to build out the number of available regulators. During this time, there has been highly technical work to better understand how to control and insulate expression by revisiting old paradigms, like the expression cassette. These efforts have yielded large sets of regulators that have been thoroughly characterized and for which the rules of assembly are better understood. This is expected to lead to a period, over the next few years, where there is sudden scale-up in the complexity of circuits. Ultimately, this will lead to circuit design as a regular component of genetic engineering projects, along with protein, pathway, and strain engineering. In this thesis, we explore TetR-family repressors and CRISPRi-based repression as platforms for scalable genetic circuit design. Further, we create a genetic circuit design software tool called Cello that automates the construction of genetic circuits using libraries of characterized parts. The goal of Cello is to accelerate and improve the reliability of genetic circuit design.

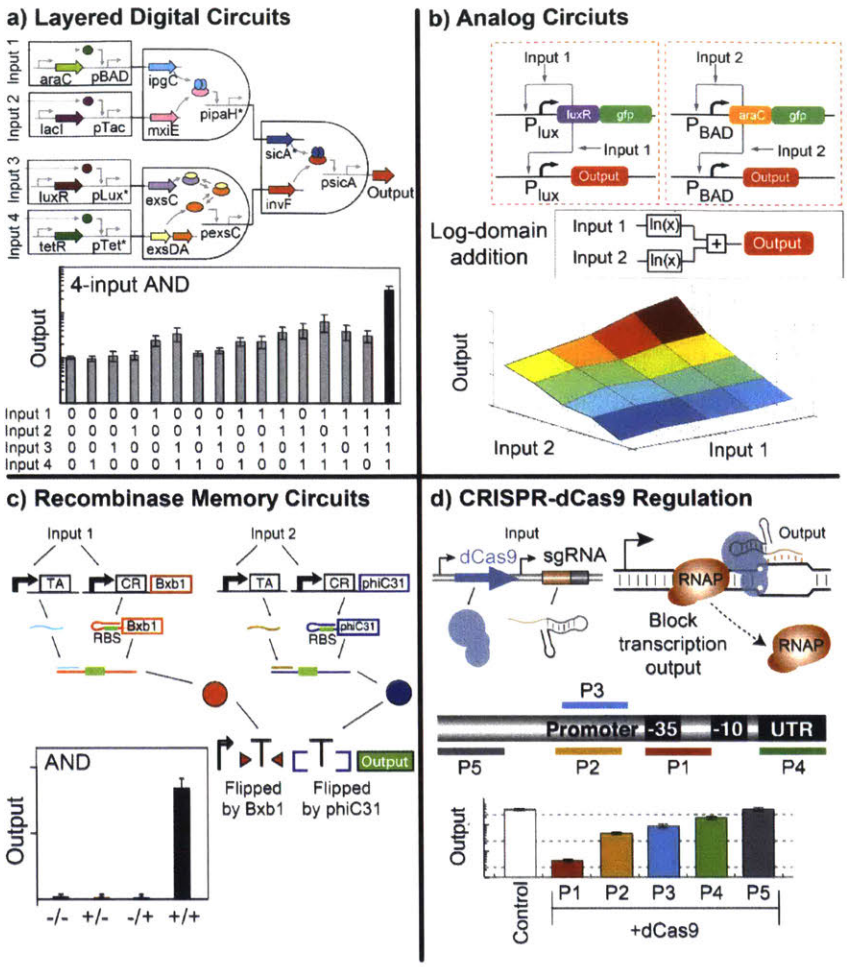

Figure 1-1: Advanced genetic circuit designs. Recent progress in building more complex genetic circuits has been enabled through development of new part families, and more sophisticated circuit architectures. (A) An intracellular 4-input AND gate built by layering three 2-input logic gates. Each 2-input logic gate works by expressing an activator from one input and a chaperon from the other output that complex and activate the output. The entire AND gate's output is high only when all four inputs (arabinose, IPTG, AHL, and aTc) are present. Adapted from (Moon *et al*, 2012). (B) An analog circuit that computes log-domain addition of arabinose and AHL concentrations was built using two wide dynamic input range feedback loops (AraC and LuxR) that express a common output. The dynamic input ranges for each feedback loop were extended by providing a "shunt" plasmid with a promoter that titrates away the transcription factor from the feedback loop. Adapted from (Daniel *et al*, 2013). (C) Recombinase memory circuits effect stable inversions of genetic regulatory elements.

An irreversible 2-input AND memory circuit was built by expressing recombinases Bxb1 and phiC31 with AHL and aTc, such that two unidirectional terminators are flipped into a non-terminating orientation when both recombinases are expressed. This circuit stably maintains its output state for several days. Adapted from (Siuti *et al*, 2013). (D) CRISPR-based gene regulation uses a catalytically-inactive Cas9 protein (dCas9) and a targeting short guide RNA (sgRNA) to guide the sgRNA-dCas9 complex specific DNA loci. By targeting various regions of a bacterial promoter, dCas9 was shown to repress transcription initiation using a highly-programmable targeting mechanism. Adapted from (Qi *et al*, 2013).

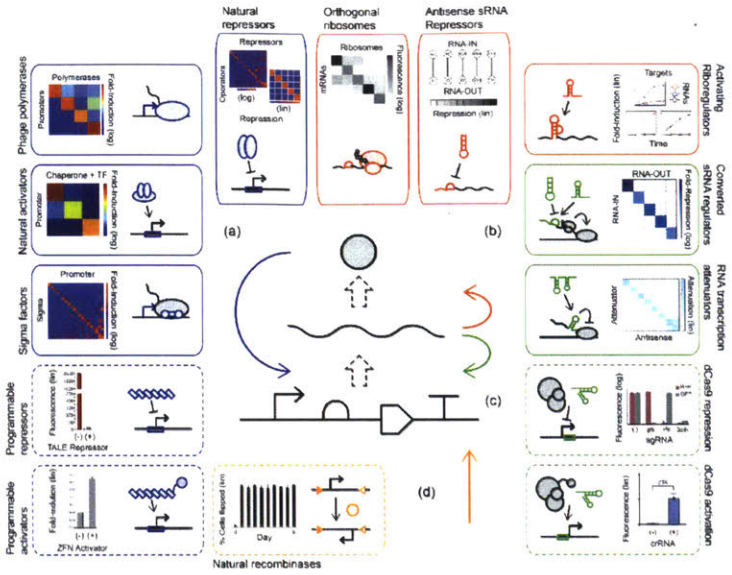

Figure 1-2: The diversity of genetic regulatory parts available for building bacterial genetic circuits. Schematics and representative data for a selection of different circuit-building mechanisms are shown. The colored regions in the schematic indicate the variable regions that make up each part. A solid line surrounding a part type indicates that there is a well-characterized, orthogonal set of parts of this type available for circuit building. A dashed line indicates that there is a proof-of-concept part. The data shown either demonstrates the orthogonality and size of a parts family, or if that is not available, it shows proof-of-concept activity. (A) Protein parts that act on transcription include natural repressors and activators, phage polymerases, sigma factors, and repressors and activators based on programmable DNA binding proteins. Data shown is from: Natural repressors (Zhan *et al*, 2010), Phage polymerases (Temme *et al*, 2012), Natural activators (Moon *et al*, 2012), Sigma factors (Rhodius *et al*), Programmable repressors (Politz *et al*, 2013b), Programmable activators (Lee *et al*, 2008). (B) RNA parts that act on translation include orthogonal ribosomes, the RNAIN/OUT system of repressing riboregulators, and activating riboregulators. Data shown is from: Orthogonal ribosomes (Chubiz & Rao, 2008), RNA-IN/OUT (Mutalik *et al*, 2012), Activating riboregulators (Callura *et al*, 2012). (C) RNA parts that act on transcription include riboregulators converted to affect transcription, transcription attenuation using a PT181-like hairpin, dCas9 repression, and dCas9-ω activation. Data shown is from: Converted regulators (Liu *et al*, 2012), PT181 attenuation (Takahashi & Lucks, 2013), dCas9 repression (Qi *et al*, 2013), dCas9 activation (Bikard *et al*, 2013). (D) Natural recombinases have been used to modify DNA and implement logic. Data shown is from (Siuti *et al*, 2013).

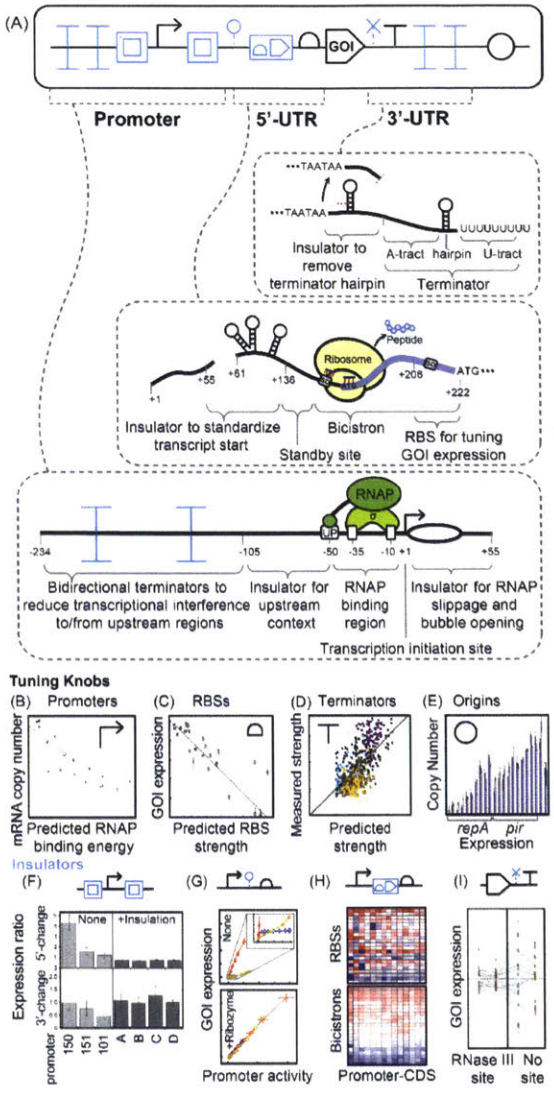

Figure 1-3: A modern expression cassette comprising genetic tuning knobs and insulators. (A) A cassette is shown for precise expression of a gene of interest (GOI). Insulating parts at the junctions are highlighted in blue. The cassette is shown using symbols from the synthetic biology open language visual (SBOLv). (B) Biophysical models of transcription based on the RNAP binding energy have been constructed to compliment empirically-characterized promoter libraries. Data is from (Brewster *et al*, 2012). (C) The RBS Calculator provides a computational framework for designing RBS sequences of a given strength based on a thermodynamic model of translation initiation. Data is from (Salis *et al*, 2009). (D) Terminator strength is partially informed from a biophysical model relating various aspects of the terminator sequence. Data is from (Chen *et al*, 2013). (E) Tunable copy-number plasmids allow for a wide range of gene expression based on the repA/ColE2 and pir/R6K plasmid systems. Data is from (Kittleson *et al*, 2011).(F) Promoter-insulating sequences have been shown to reduce the change in promoter activity when upstream and downstream sequences are introduced to the flanking promoter context. Data is from (Davis *et al*, 2011). (G) Ribozymes have been used to improve the predictability of gene expression by reducing promoter-5'UTR coupling effects. Data is from (Lou *et al*, 2012). (H) Bicistronic RBS designs cause the rank order of expression constructs to be more predictable compared to single RBSs. Data is from (Mutalik *et al*, 2013). (I) RNase III sites in the 3'UTR reduce the variability in gene expression for reporters coupled with libraries of promoters. Data is from (Cambray *et al*, 2013).

Table 1-1: Characteristics of part families currently available for constructing genetic circuits in bacteria.

		Family Size	Genetic Size	Maximum dynamic range	Largest tested parts set with minimum orthogonal range				References	
					2	5	10	50		
	DNA Modification									
	Recombinases	2[d]	L	92	2	2	2		(Siuti *et al*, 2013)[b]	Nonlinear; Implements passive memory
	Transcription Regulation									
Protein Parts	T7 polymerases	4	L	270	4	4	3		(Temme *et al*, 2012)[b]	
	TetR repressors	20	M	210	16	15	12	7	(Stanton *et al*, 2013)[a]	Nonlinear
	LacI repressors	5	M	45	5	5	4		(Zhan *et al*, 2010)[c]	Nonlinear
	Chaperone activators	3	M	69	3	2	2		(Moon *et al*, 2012)[a]	Each chaperone-activator pair can be used as an AND gate.
	Sigma factors	27	M	470	17	15	13	9	(Rhodius *et al*)[a]	A set of anti-sigma factors enable sequestration.
	TAL repressors[4]	1+[e]	L	110					(Politz *et al*, 2013a)[a]	
	Zinc finger activators[5]	1+[f]	M	3					(Morbitzer *et al*, 2010)[b]	
	Zing finger repressors	1+[f]	M	8					(Morbitzer *et al*, 2010)[b]	
RNA Parts	Converted RNA-IN/OUT	5	S	880	5	5	5	5	(Liu *et al*, 2012)[b]	
	PT181 attenuators	14	S	8	8	2			(Takahashi & Lucks, 2013; Lucks *et al*, 2011)[a]	Attenuator can be repeated to increase signal.
	dCas9 repression	1+[g]	S	340					(Qi *et al*, 2013; Bikard *et al*, 2013)[b]	Requires dCas9 expression.
	dCas9-ω activation	1+[g]	S	23					(Bikard *et al*, 2013)[a]	Requires dCas9-ω expression.

Translation Regulation								
RNA-IN/OUT	23	S	10	13	6		(Mutalik *et al*, 2012)[a]	Additional large sets of RNAs characterized & computed
Activating riboregulators	4	S	45	3	2	2	(Callura *et al*, 2012; Isaacs *et al*, 2004)[b]	
O-ribosomes	5	L	78	5	5	3	(Chubiz & Rao, 2008; Rackham & Chin, 2005)[c]	

This table presents a brief comparison of a number of regulatory part families that have been characterized and are available for use in building genetic circuits in bacteria. Each part is defined as a trans-acting element and the target of this element: for example a transcriptional repressor and its binding site. Part families were chosen to either have at least three characterized members or be based on a technology proven to be extendable (TALE, zinc finger, and CRISPR-based parts).

Characterized family size indicates the number of parts of this type that have been characterized for crosstalk. A '+' indicates that this parts family is based off of a technology proven to enable orthogonal, programmable DNA binding and so the parts set may be predictably extendable.

Maximum dynamic range is the largest reported fold change between the on and off states (i.e. with the trans-acting element present and absent) of a single member of the part family.

The largest tested parts sets show the largest number of parts in each family that have been shown to function above specific thresholds of orthogonal range. Orthogonal range is a conservative measurement of the orthogonality of a parts set; it represents the fold change between the on-target effect of a part and the worst off-target effect on that part. For example, the 'T7 polymerases' family has a 3 part set that functions above 10x orthogonal range, meaning that there is a group of three polymerases where each activates its target promoter to a level more than 10x the level that either of the other two polymerases activate it.

a Numerical data was used from this reference for the part family in this row.

b Data computed from bar or line plots in this reference was used in this row.

c Data was read from colored orthogonality grids in this reference. Note that the numbers in this row may be less accurate because of uncertainties in this method.

d While only two recombinases have been tested together at a time, at least seven have been used in genetic circuits in bacterial systems (Ham *et al*, 2008, 2006; Friedland *et al*, 2009; Bonnet *et al*, 2012; Siuti *et al*, 2013; Bonnet *et al*, 2013).

e TAL repressors and activators are much more widely used in eukaryotes. A set of 8 orthogonal TAL activators has been tested in mammalian cells, and it is predicted that many more could be built (Garg *et al*, 2012).

f Zinc fingers have also been widely used in eukaryotes (Beerli & Barbas, 2002).

g CRISPR repression and activation is being widely adopted in eukaryotes. Crosstalk data from these organisms suggests that many orthogonal variants could also be made in bacteria (Farzadfard *et al*, 2013; Fu *et al*, 2013; Hsu *et al*, 2013).

Chapter 2

2 Genomic mining of prokaryotic repressors for orthogonal logic gates

2.1 Background

Living cells can be programmed by incorporating integrated genetic gates into their DNA (Weiss & Jr, 2000). These gates rely on biochemical interactions to perform computational operations, including switches, logic and memory (Khalil & Collins, 2010; Weber & Fussenegger, 2011). Gates can be connected to each other when they are designed to be extensible, meaning that the form of their input and output signals are the same. For example, if both the inputs and outputs are promoters, then this signal is defined as the flux of RNA polymerase on DNA (Endy, 2005). To date, the complexity of circuits has been low, consisting of the few available gates based on the transcription factors reused across labs and projects (Purnick & Weiss, 2009b). Increasing the number of available gates will enable the construction of larger circuits to encode more sophisticated algorithms (Moser *et al*, 2012b). The challenge has been that all of the gates within a circuit need to be orthogonal; in other words, the biochemical interactions on which they are based cannot cross-react (Thompson *et al*,

2012). It becomes increasingly difficult to add gates because the number of potential cross-reactions grows quickly as N^2–N.

NOT and NOR gates are simple and broadly useful functions. A transcriptional NOT gate (the output is OFF when the input is ON) can be implemented by using an input promoter to drive expression of a repressor, which turns off expression of an output promoter (Fig. 2-1a) (Yokobayashi *et al*, 2002a). Even these simple gates can perform signal-processing functions, for example, converting a dark sensor into a light sensor (Tabor *et al*, 2011) and a male sensor into a female sensor (Fu *et al*, 2010). A NOR gate is a logic function where the output is ON only when both inputs are OFF. NOR gates are Boolean complete, meaning that they can be combined to generate any computational operation. A genetic NOR gate can be built by adding a second input promoter in series to the NOT gate so that both input promoters drive the expression of the repressor (Tamsir *et al*, 2011b). Two gates are orthogonal if their repressors do not bind each other's promoters. Obtaining more gates that can be used as part of the same circuit requires having a set of repressors that bind different operator sequences.

There are a number of biochemical mechanisms that could be used to produce the repressing function required by a NOT gate. The most common is to use a protein-based repressor, which binds an operator DNA sequence within its target promoter. NOT gates have been built using different classes of natural repressors including phage repressors (e.g., cI), LacI-family and TetR-family repressors (Yokobayashi *et al*, 2002a; Gardner *et al*, 2000b; Elowitz & Leibler, 2000a). Several modular scaffolds, such as zinc finger proteins (ZFPs) (Hurt *et al*, 2003) and transcription activator–like effectors (TALEs) (Boch *et al*, 2009b), have a domain architecture that allows proteins to be designed to bind target sequences. ZFPs and TALEs have been used to control expression in eukaryotic cells (Khalil *et al*, 2012; Garg *et al*, 2012a; Zhang *et al*, 2011) and to a lesser degree in prokaryotes (Politz *et al*, 2013b; Durai *et al*, 2006).

Recently, it has been shown that transcription can be repressed with a CRISPR-Cas system ('CRISPRi') that uses a nuclease-null Cas9 protein and an RNA guide sequence to block transcription at a specific site (Qi *et al*, 2013). Because of the programmability of the RNA-DNA interaction, this system holds promise for building orthogonal repressors; the use of CRISPRi to build layerable gates has the potential to be a powerful tool in the construction of circuits.

In this paper, we decided to target TetR homologs for several reasons. First, TetR is one of the earliest and most pervasive transcription factors used in biotechnology and has appeared in numerous applications (Ramos *et al*, 2005). As an inducible system, it is part of a classic multi-plasmid system (Lutz & Bujard, 1997) and has been used in a broad range of host organisms, including bacteria and archea (Guss *et al*, 2008), fungi (Dingermann *et al*, 1992), insects (Lycett *et al*, 2004), plants (Gatz & Quail, 1988, 10), mammalian cells (Gossen & Bujard, 1992), and live animals (Saez *et al*, 1997). Second, it has been used in many genetic circuits in synthetic biology, including a toggle switch (Gardner *et al*, 2000b) and oscillator (Elowitz & Leibler, 2000a) in *Escherichia coli*. It has also been used to build a time-delay circuit in mice (Weber *et al*, 2007) and a NOT gate in mosquitoes (Fu *et al*, 2010). Third, TetR and most homologs have a simple mode of repression where dimers bind a promoter and physically block RNA polymerase (Orth *et al*, 2000). Fourth, they are able to achieve specificity with relatively short operator sequences (Ramos *et al*, 2005). Finally, tens of thousands of homologs are available from many host organisms, and there is evidence that they exhibit sequence specific binding to disparate operator sequences (Lutz & Bujard, 1997). Small differences in the amino acid sequence and operator nucleotides have been shown to yield high-affinity, orthogonal interactions (Helbl *et al*, 1998; Krueger *et al*, 2007). The potential for orthogonality is also large; coding

theory predicts that there is an upper limit of 130 helix-turn-helix repressors that could function in one cell without exhibiting cross-talk (Itzkovitz *et al*, 2006).

To increase the number of available gates, we used DNA synthesis to access repressors selected from the sequence database and screened them to identify an orthogonal subset. Using an *in vitro* microarray assay, the DNA binding preferences for individual repressors were comprehensively examined, from which well-defined motifs were obtained. This information, together with previously identified operator sequences, was used to construct synthetic promoter libraries to identify those that were highly repressed. The resulting repressor-promoter pairs were systematically converted into NOT gates, their cross-reactions were measured in all combinations and then they were used to construct composite circuits *in vivo*. Overall, this work represents a large set of compatible, orthogonal components from which user-defined circuits can be constructed by simply changing the pattern of input and output promoters between a set of conserved gates.

2.2 Characterization of a TetR homolog library

We developed a pipeline to expand the number of available TetR family repressors, to exhaustively measure their activity and orthogonality and to characterize them in the context of genetic gates (Fig. 2-1b). TetR homologs encompass one of the largest families of transcription factors, with 82,017 members currently annotated in EMBL-EBI (Hunter *et al*, 2012). To build a library of homologs, we started with 73 repressors obtained from a collated list of TetR homologs with known regulatory functions from diverse organisms (Ramos *et al*, 2005) (Fig. 2-1c). Redundant sequences and incomplete entries were excluded from the list. This set contains homologs from 45 distinct prokaryotic species and has an average amino acid identity of 21%. Genes were

codon optimized for expression in a set of target organisms and built using DNA synthesis.

For the majority of repressors in the library, operators were determined using an assay based on cognate site identifier array analysis (Stanton *et al*, 2013). The operators for McbR, PsrA, QacR and ScbR had been previously identified, and the array data closely matched sequences from the literature. Substantial diversity exists among the operator sequences bound by different repressors within the library.

2.3 Synthetic promoter design and orthogonality characterization

Synthetic promoters were designed to contain operator sequences that were either identified using the array or obtained from the literature (Methods). A strong constitutive *E. coli* promoter (BBa_J23119) was used as a backbone into which an operator was placed (Kelly *et al*, 2009a). Promoter libraries were constructed to determine the optimal placement and sequence of the operators. The data from the array were used to determine an 'operator motif' that captures the functional diversity of the operator sequence (Fig. 2-2a). Sequences consistent with the motif were constructed using degenerate oligonucleotides and inserted into various positions in the promoter around and between the −35 and −10 sequences. The promoter libraries were then screened in the presence and absence of their cognate repressor by eye or using flow cytometry (Fig. 2-2b). From each library, the promoter that generated the highest dynamic range was identified, sequenced and then confirmed. At the end of this process, we identified promoters that were responsive to 20 repressors (Fig. 2-2c). This set consists of ten promoters whose operators were obtained from the CSI array and ten that were obtained from the literature (Table 2-6).

To measure all of the possible cross-reactions, we assayed the activity of each repressor against the set of 20 promoters. Repressor expression was controlled by the HSL-inducible P_{Lux} promoter in a colE1 plasmid (Fig. 2-3). The promoters were fused to YFP in a p15A plasmid (Fig. 2-14). The repressor and promoter plasmids were cotransformed in all combinations. The resulting 400 strains were grown in the presence of inducer, the promoter activity was measured using cytometry and the fold repression was reported as the ratio between the non-repressor-containing control plasmid and the induced repressor. These data were used to construct an orthogonality matrix that shows the specificity of each promoter and repressor (Fig. 2-2d). The repressors are remarkably orthogonal, and a core set of 16 have minimal cross-reactions (TetR, IcaRA, AmtR, BetI, SrpR, Orf2, BM3R1, ButR, PhlF, AmeR, QacR, LmrA, PsrA, HlyIIR, McbR, ScbR, TarA, LitR, HapR, and SmcR). Among this orthogonal set, the sequence diversity of the DNA-binding region is noteworthy (Fig. 2-8) (Orth *et al*, 2000). Previous work shows that within the recognition region of the DNA-binding domain (residues 25–44 in TetR), residues 28 and 37 are particularly important for binding specificity (Orth *et al*, 2000; Krueger *et al*, 2007). Out of the set of 16 orthogonal repressors, 11 and 9 different amino acids are represented at these positions, respectively.

Several groups of repressors are not orthogonal, and this corresponds to amino acid similarity in their DNA-binding domain (Fig. 2-9). The HapR, LitR, and SmcR repressors (all from *Vibrio* species) interact with each other's promoters and have similar patterns of cross-talk. Similarly, the HlyIIR and PsrA repressors share amino acid identity and bind similar operators. Unexpectedly, the converse is not true, where similarity between promoters is not predictive of cross-talk. This is largely a result of the variability observed in the acceptable spacing distance between the 6-mer repeat

of the operator. For example, the LitR and HapR promoters have spacers of 11 bp and 3 bp, but the repressors cross-react equally well with both.

2.4 Design and measurement of genetic NOT gates

The repressors and their synthetic promoters were used to build a library of NOT gates. To measure the response as a function of the activity of an input promoter, the IPTG-inducible P_{Tac} promoter was connected to each gate. The output was measured by having the repressible promoter drive the expression of YFP. Each NOT gate consists of a 5′ UTR, repressor gene, terminator and synthetic promoter. These parts, along with the P_{Tac}-inducible system and YFP, were assembled into a p15a plasmid (Fig. 2-14).

Ensuring repressor expression is within the appropriate range to generate a large output response represents a challenge in converting repressors into gates. Expression levels cannot be changed by varying the dynamic range of the input promoter because, for circuit construction, inputs must be swapped without further modification. Thus, we define the beginning of each gate to be the transcription start site and vary the repressor level by changing the strength of the ribosome binding site (RBS). Each repressor has unique properties that influence their absolute protein levels (e.g., codon usage, mRNA and protein stability and binding affinity). Hence, the RBS of the repressors had to be individually tuned to maximize the dynamic range. To accomplish this, we used the RBS calculator (Methods) to design a set of sequences that systematically vary the predicted expression from medium-high to low (because the optimal desired expression is not known a priori). These were used to design a degenerate oligonucleotide, from which an RBS library was constructed for 19 out of

the 20 gates (Table 2-4). These libraries were screened, and the RBS that produced the highest dynamic range was selected (Table 2-5).

The response function of a gate captures how the activity of the output promoter changes as a function of the input promoter. This information is critical in determining how gates will operate when connected in a circuit. It is also important that the inputs and outputs are reported in the same units (Endy, 2005). As such, we reported these values as relative expression units (REUs) (Fig. 2-15) (Kelly *et al*, 2009a). REUs are calculated by normalizing the YFP output values by that measured from a reference standard and by separately measuring the activity of the P_{Tac} input promoter as a function of IPTG (Fig. 2-16), using the same reference standard. With these data, it is theoretically possible to know whether the range of the output of one gate is sufficient to serve as an input to the next gate in series.

Each gate produces a unique response function (Fig. 2-3). The dynamic ranges of the gates vary from 207-fold (SrpR) to 5-fold (SmcR and ButR), with an average of 51.3-fold (Table 2-1). Cytometry distributions are shown for the ON and OFF states, which are narrow and have good separation, even for those that have a smaller dynamic range (Fig. 2-5). The response functions can be fit to a Hill equation:

$$y = f(x) = y_{min} + (y_{max} - y_{min})\frac{K^n}{K^n + x^n}$$

where y is the activity of the output promoter, y_{min} is the minimum output, y_{max} is the maximum output, n is the Hill coefficient and K is the threshold level of input where the output is half-maximal. The Hill equation was used to fit the data for each gate, and the parameters are shown in Table 2-1.

The thresholds for the gates are similar with an average of $K = 0.4$ REU and a range of 0.1 REU (TarA) to 1.3 REU (ButR). Considering this, all of the NOT gates have sufficiently high ON states (between 3 REU and 70 REU) to achieve full repression by crossing the threshold required by a downstream circuit. However, the

OFF states range between 0.1 REU and 2.1 REU. Because the OFF states are similar in magnitude to the thresholds, this can be problematic when connecting gates and can lead to a degradation in the signal as the number of layers in the circuit increases (Yokobayashi *et al*, 2002a).

Gates that exhibit ultrasensitivity generate a large output response with little change in the input signal. This also comes at a cost: it becomes increasingly difficult to balance the input to span the range required to achieve the maximum response. The cooperativity for the majority of gates is $n \approx 2$, which is consistent with that measured for TetR, and a mechanism of dimers binding to a single operator (Elowitz & Leibler, 2000a). Five of the repressors yield gates with $n > 3$, with the largest being 6.1 for Orf2. This has been observed before with TetR homologs, which can bind with higher cooperativities by assembling as multimers or multiple dimers within a single operator (Grkovic *et al*, 2001).

Transcription factors can be toxic and exhibit slow growth when expressed above a critical threshold (Kittleson *et al*, 2012). We measured the impact on cell growth by recording the OD_{600} 6 hours after induction for each NOT gate at various levels of induction (Fig. 2-6). The majority of repressors are nontoxic, even when maximally expressed. Six repressors showed toxicity at high input levels: TarA, ScbR, ButR, SmcR, Orf2 and HapR (with toxicity defined as >25% reduction of growth) (Fig. 2-7). In each case, the toxicity occurs after the output promoter has been repressed. The quantification of regions of toxicity enables a designer to build circuits that avoid expression above these levels. Further, it enables a comparison between different biochemistries that can be used for the construction of integrated circuits.

2.5 Connecting gates to create layered genetic circuits

The NOT gates can be converted into multi-input NOR gates by connecting multiple promoters in series to drive repressor expression (Tamsir *et al*, 2011b). Logic minimization algorithms, such as ESPRESSO (Rudell, 1986), can convert any arbitrary user-defined truth table into a wiring diagram composed of layered NOR gates (Brown & Vranesic, 2013). The wiring diagram can then be replicated as a genetic circuit through assembling a particular pattern of input and output promoters connected to the gates (Fig. 2-4). By changing this assembly pattern, the same set of underlying orthogonal gates can be used to build any desired circuit.

To demonstrate the assembly of gates, we constructed two simple circuits that perform the AND (the output is ON exclusively in the presence of both inputs) and NAND (the output is OFF exclusively in the presence of both inputs) logic functions through different permutations of the NOT and NOR gates. The inputs to the circuits consist of different combinations of inducible promoters: P_{Tac} (IPTG), P_{Lux} (HSL) and P_{Tet} (aTc) (Fig. 2-12). The NAND gate consists of two NOT gates (based on PhlF and LmrA), which invert the two input signals (Fig. 2-4a). The output of the NOT gates are assembled in series to form an OR gate, which then serves as the output of the circuit. The circuit produces the correct NAND function, with a sixfold difference between the OFF state (+/+) and the lowest ON state. The OFF state is high, which is consistent with the leakiness of the LmrA promoter.

The AND circuit was constructed by combining three gates (Fig. 2-4b). The PhlF NOT gate (the same as that used for the NAND circuit) serves to invert one of the input promoters. The other input promoter is inverted by the QacR NOT gate. The output promoters of these gates are connected to BetI to form a NOR gate, the output of which drives the expression of YFP. This circuit produces a 4.4-fold

response when the ON state (+/+) and the highest OFF state are compared. Flow cytometry histograms for each circuit and the terminal gates are illustrated in Figure 2-10.

To determine whether individually measured response functions of gates (Fig. 2-3) can be used to predict their combined response as a circuit, we developed a simple model of the NAND and AND circuits. This model simply adds the response functions of the inducible inputs and the gates to obtain the response of the circuit as a whole, with no additional fit parameters. The OFF and ON states of inducible promoters that serve as inputs (P_{Tac}, P_{Lux} and P_{Tet}) were measured independently and converted into REU. The (OFF–ON) states of the inducible promoters are: P_{Tac}(0.06–6.2), P_{Lux} (0.7–8.2) and P_{Tet} (0.07–9.8). To determine the predicted function of a circuit, we tracked the combinations of signals from the input promoters through the gates using their response functions. This process is visualized in Figure 2-13.

To model the NAND circuit, the range of P_{Tac} is inserted into the PhlF response function, yielding outputs of 16 REU and 0.1 REU (Table 2-3). Similarly, the range of the P_{Lux} input is converted to 61 REU and 1.4 REU by the LmrA response function. The output of the OR gate is treated as the simple sum of the outputs of the tandem promoters. The predicted values for the four combinations of input states closely match the experimental data (Fig. 5a).

To model the AND circuit, the output of P_{Tac} connected to the PhlF gate is the same as reported above (16 REU and 0.1 REU), and the output of P_{Tet} connected to the QacR gate is 20 REU and 0.4 REU (Table 2-2). To model the NOR gate, the outputs of these promoters are summed as $x = x_1 + x_2$ and serve as the input to the BetI response function. As with the NAND circuit, the predicted response closely matches the experimental measurements (Fig. 2-4b). Both circuits have some quantitative differences between the predictions and experimental data. This is most

likely due to the simplicity of the model, which does not account for changes in genetic context, promoter interference between tandem promoters, plasmid copy number variation (Moser *et al*, 2012b; Lou *et al*, 2012b) or the growth phase under which the outputs were measured (Fig. 2-11).

2.6 Discussion

The ability to manipulate gene regulation is one of the last frontiers in genetic engineering. The implementation of computing in cells has the potential to affect many applications in biotechnology. However, the field has been limited in the size and sophistication of circuits that could be constructed from a small number of characterized transcription factors. Here, we substantially expand the number of repressors that are available for circuit c

onstruction. Further, we have rigorously measured the cross-reactions to identify a core orthogonal set. Each member of this set is converted into a gate and fully characterized. Finally, we introduced a generalized method by which circuits can be assembled by changing the pattern of input and output promoters to reproduce a wiring diagram composed of NOT and NOR gates. For simplicity, we demonstrate this by building two circuits that perform digital Boolean logic operations. Notably, the same approach could be applied to build analog (Daniel *et al*, 2013b) and dynamic (Elowitz & Leibler, 2000a) circuits.

The mining effort described here started with 73 homologous repressors and ended with a set of 16 orthogonal gates. By considering all of the possible ways that these gates can be combined, one can imagine a 'circuit space' that consists of all of the possible wiring diagrams. The size of this space can be estimated by

$$N = \sum_{k=0}^{n} \frac{n!(3k+1)^{2k}}{k!(n-k)!}$$

where n is the size of the orthogonal set, and k is the number of repressors in the circuit. This takes into account that (i) there are up to $2k$ sensor inputs to the circuit as a whole; (ii) only NOT and 2-input NOR gates are considered; (iii) for a gate, each input can comprise one of the circuit inputs or an output from another gate or can be unconnected, yielding $(3k + 1)$ possibilities; and (iv) functionally redundant or isomorphic circuits are not removed from the set. This estimates that $n = 16$ orthogonal gates can be used to build $N > 10^{54}$ possible circuits. This set includes feedback loops and is not limited to digital logic. Each of these circuits can be accessed by permuting the input and output promoters into a particular pattern.

The challenge now becomes achieving a degree of reliability where the gates can be assembled into any of these circuits with a reasonable chance of functioning properly. Although we demonstrate this mapping with a few circuits, accessing the potential of the space remains a challenge. The first generation of gates presented here was designed to be simple and consist of a single operator in a constitutive promoter. This simplicity leads to gates that exhibit low cooperativity, a high OFF state and sensitivity to genetic context (Lou *et al*, 2012). Further, each gate uses the same pair of terminators, which can lead to evolutionary instability for large circuits (Chen *et al*, 2013; Sleight & Sauro, 2013). Analyzing the gates in different contexts and identifying the failure modes could lead to second-generation designs that are engineered to be faster, tunable and robust by implementing design rules that have emerged from control theory and systems biology (Lou *et al*, 2012b; Lee & Maheshri, 2012a; Buchler & Cross, 2009; Bintu *et al*, 2005).

We selected TetR homologs because of their high specificity, stability and proven capability to operate in synthetic circuits. Other biochemistries could be used to expand the number of orthogonal gates in our library. To be compatible, the only constraint is that the inputs and outputs of each gate must be promoters, thus allowing

the gates to be layered. The repressors could be other classes of proteins that bind DNA, such as TALEs, ZFPs or the guide RNA that directs Cas9 as part of CRISPRi; a single large circuit could contain mixtures of these biochemistries. Indeed, this may be a mechanism to expand the gate library beyond the informatic limit of any one family. For example, our TetR library already covers 15% of the predicted upper limit on helix-turn-helix repressors (Itzkovitz *et al*, 2006).

The set of orthogonal gates we present is sufficiently large to implement nontrivial circuits of direct relevance to applications in biotechnology, which includes multi-input logic control for environmental or metabolite sensing, timers to control when different genes are expressed, multiple toggle switches for memory and simple algorithms from control theory. However, now that parts are no longer limiting, it remains a challenge to build large circuits. To this end, computational tools will most likely play a more central role in design. Changing the inputs and outputs to gates by rearranging the pattern of input and output promoters is a sufficiently simple operation to be performed by a computer. The co-development of simple schemes for genetic programming, as well as gates designed specifically to be compatible with these schemes, will enable the broader application of genetically encoded algorithms to program cells.

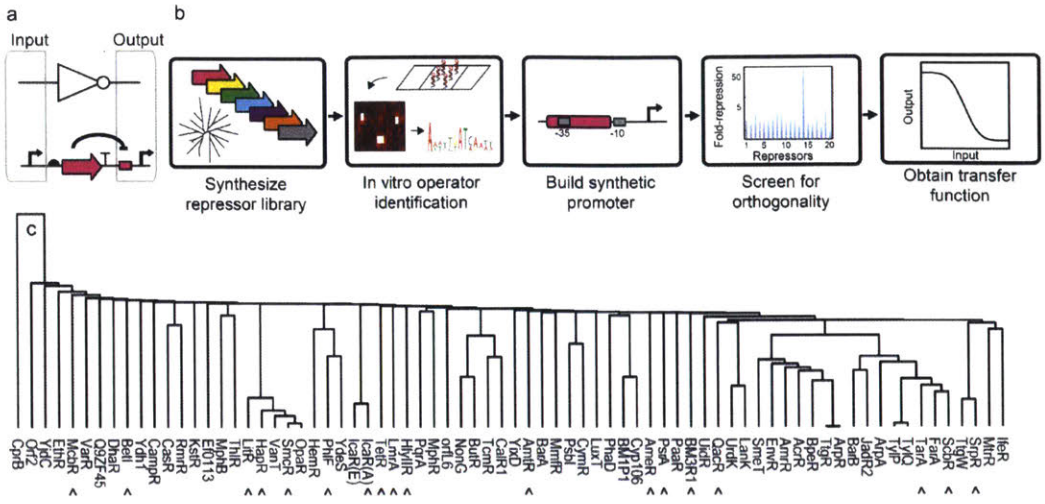

Figure 2-1: A large repressor library is compiled using genome mining. (a) A genetic NOT gate (symbol shown) can be built using a repressor (pink arrow) that binds an operator (pink box) in an output promoter. (b) The pipeline for the discovery and characterization of orthogonal repressors is shown. The second panel depicts a portion of the CSI microarray used to determine the operator sequence. (c) The complete library of 73 synthesized repressors (plus TetR) are organized into a phylogenetic tree diagram, where carets indicate repressors that appear in the final orthogonality matrix illustrated in Figure 2-3d. The tree was aligned on the basis of respective repressor protein sequences, and branch lengths correspond to relative divergence in amino acid sequence. The two IcaR orthologs originate from two distinct host organisms where A indicates Staphylococcus aureus and E indicates Staphylococcus epidermidis.

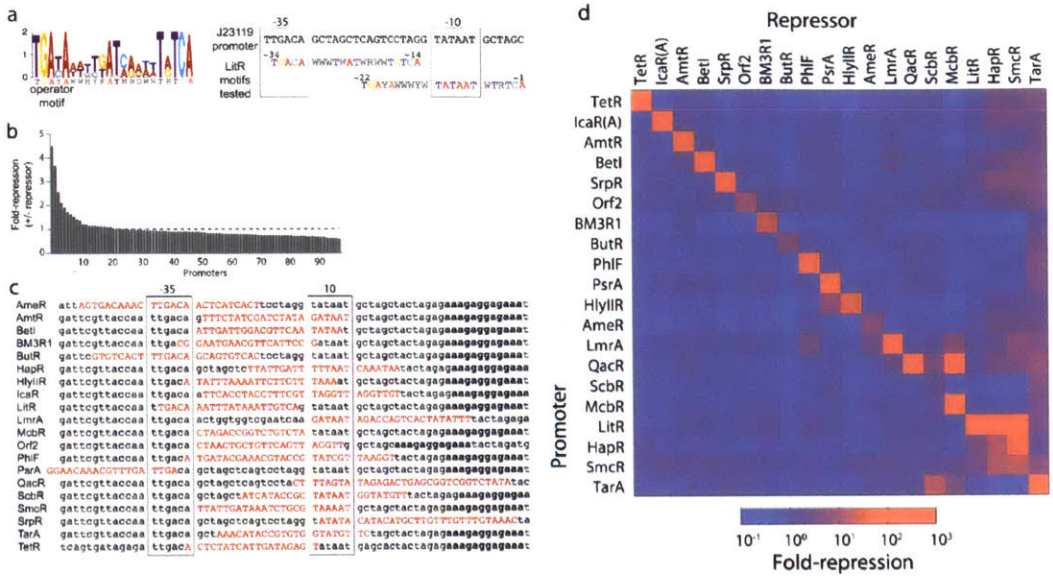

Figure 2-2: Design and screening of orthogonal promoters. (a) Degeneracy in operator sequences (Stanton *et al*, 2014) is converted into a single motif. The LitR motif is shown (W is A/T, H is A/T/C, Y is T/C, K is G/T, M is C/A, R is A/G and D is A/T/G). The degenerate operator is placed in the BBa_J23119 constitutive promoter spanning either the −35 or −10 element (right panel). (b) The results of screening the LitR promoter library are shown. The fold repression is calculated as the ratio of fluorescence from the promoter alone and that obtained when the repressor is present and uninduced for a single replicate. (c) The best promoters identified in the screens are shown for each repressor that are part of the final set of 20 repressors. The operator sequence is shown in capital red letters, and the Shine-Dalgarno sequence is in bold letters. Those promoters lacking the Shine-Dalgarno sequence contain this sequence adjacent to the 3′ end of the sequence listed; when not shown, the sequence up to the ATG start is identical. (d) The promoters driving YFP expression are carried on a p15a plasmid, and the repressors are under 3OC6-N-(β-ketocaproyl)-L-homoserine lactone–inducible control on a ColE1 plasmid (Figs. 2-8 and 2-9). The matrix has been sorted by eye such that the most orthogonal promoters appear at the top and the least at the bottom, and similar patterns of cross-reactivity are clustered together. Repressor expression is induced by 20 μM HSL (except in the case where such concentrations of HSL are toxic, including HapR, Orf2, ScbR and SmcR, which were induced with 2 μM, 0.02 μM, 0.2 μM and 0.2 μM HSL, respectively). The data represent the average of three replicates collected on different days.

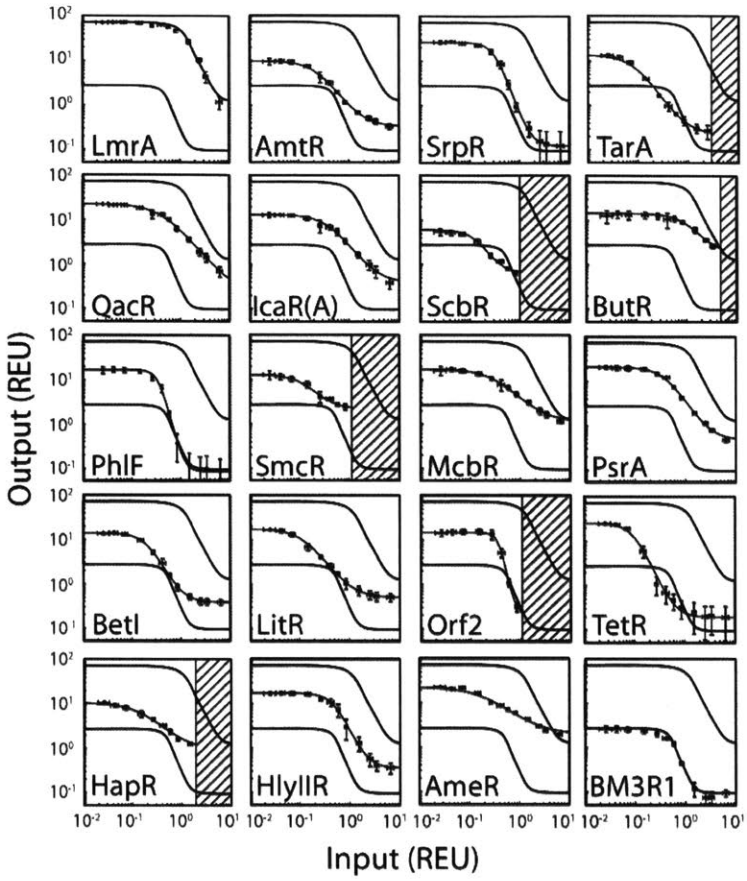

Figure 2-3: Response function measurement. The response functions are measured using the IPTG-inducible P_{Tac} promoter as an input and measuring the response of the output promoter. The activity of the input promoter is measured separately using YFP. The activities of the input and output promoters are converted to REU. The response functions of the NOT gates are shown. From left to right, the concentration of IPTG is: 0 μM, 5 μM, 10 μM, 20 μM, 30 μM, 40 μM, 50 μM, 70 μM, 100 μM, 150 μM, 200 μM, 500 μM and 1,000 μM. As a guide to the eye, the highest (LmrA) and lowest (BM3R1) response functions are shown on each plot, with the region between them in gray. The dashed regions indicate the levels of expression beyond which toxicity is observed (Figs. 2-15 and 2-16). The data represent the average of three replicates collected on different days, and error bars correspond to the s.d. between these measurements.

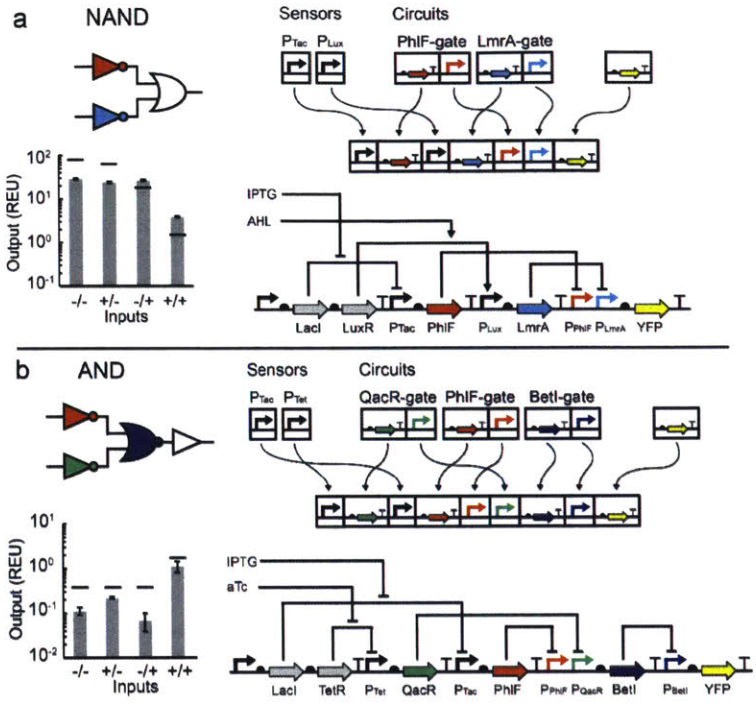

Figure 2-4: Construction and characterization of integrated circuits. (a) The process of promoter mapping for the assembly of gates into a desired circuit is shown for the NAND circuit. The measured data are grown under conditions of no inducer $(-/-)$, 1 mM IPTG $(+/-)$, 20 μM HSL $(-/+)$ and 1 mM IPTG and 20 μM HSL $(+/+)$. The bar graph details the measured output levels under all of the input combinations. Small black bars indicate the predicted output value for the indicated input. The data represent the average of three replicates collected on different days, and error bars correspond to the s.d. between these measurements. (b) The design, construction and characterization of the AND circuit is illustrated. Note that when multiple promoters are placed upstream of a repressor, the gate is converted from the NOT to NOR function. The measured data are grown under conditions of no inducer $(-/-)$, 1 mM IPTG $(+/-)$, 100 ng/mL aTc $(-/+)$, and 1 mM and 100 ng/mL aTc $(+/+)$. The bar graph details the measured output levels under all input combinations. Small black bars indicate the predicted output value for the indicated input. The data represent the average of three replicates collected on different days, and error bars correspond to the s.d. between these measurements.

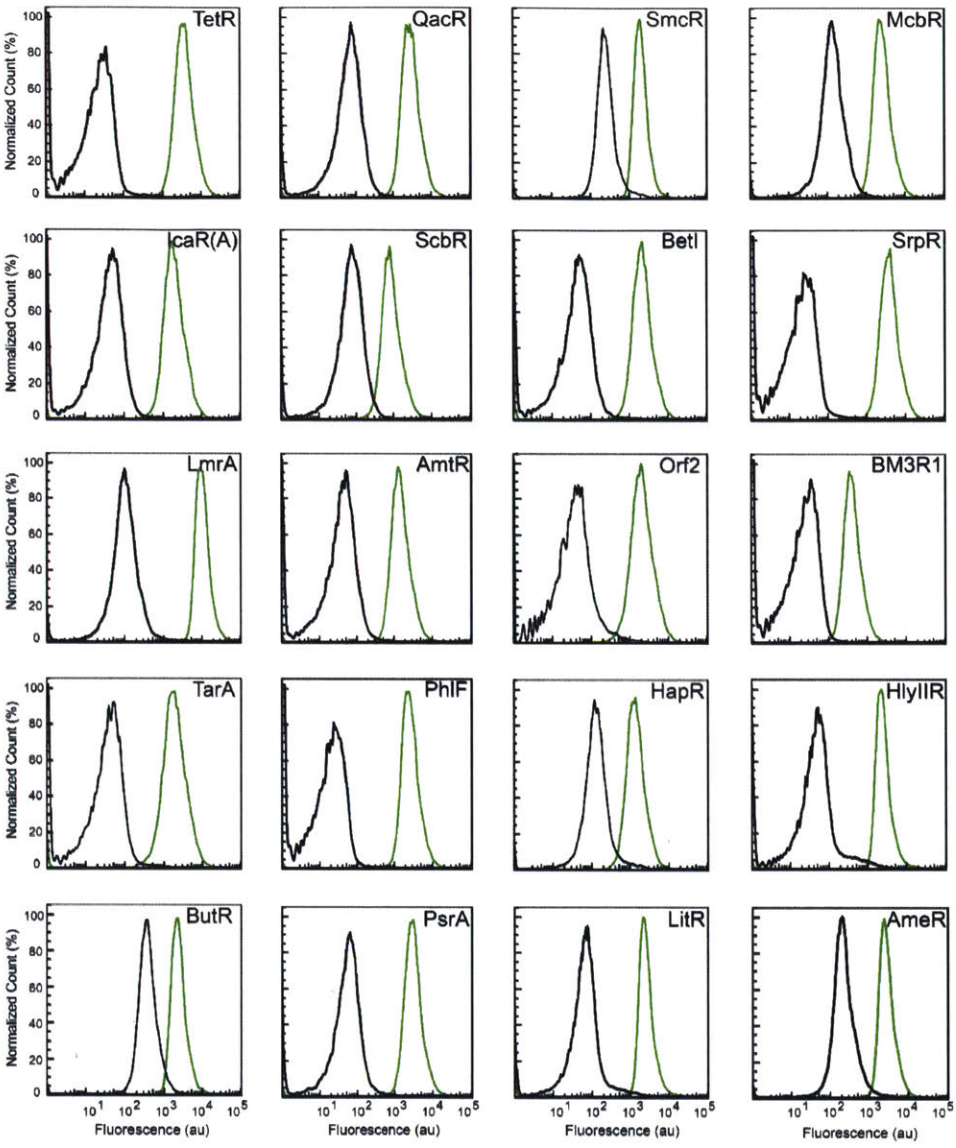

Figure 2-5: Flow cytometry data for each NOT gate. Fluorescence histograms correspond to representative single cytometry replicates for induced (black) and uninduced (green) states. The induced state corresponds to the highest IPTG concentration before toxicity was observed (200 μM for ButR, 150 μM for TarA, 100 μM for HapR, 70 μM for ScbR, 70 μM for SmcR, 70 μM for Orf2, and 1 mM IPTG for all other repressors). Each histogram comprises >10000 cells.

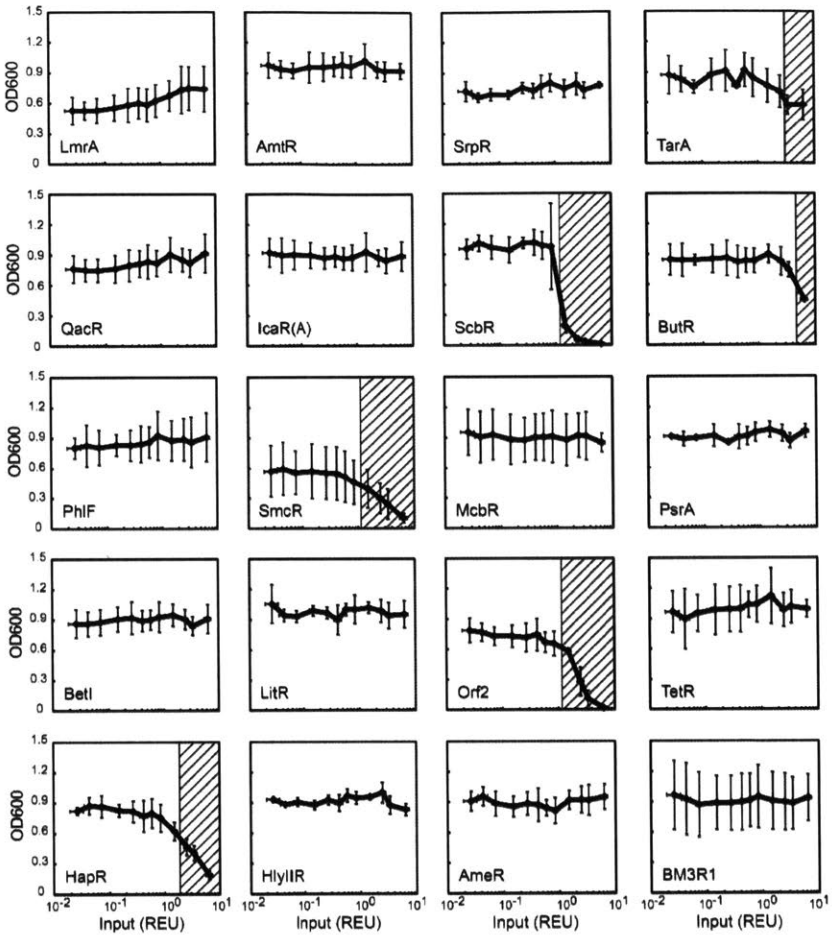

Figure 2-6: Growth measurements for NOT gate response functions. The optical density at 600 nanometers was measured for all NOT gates at each of the twelve inducer concentrations: 0, 5, 10, 20, 30, 40, 50, 70, 100, 150, 200, 1000 μM IPTG in an analogous manner to the response functions (Figure 2-3). The x-axis values are converted to the REU values measured for the response function assay. Toxicity is indicated by the hash-marked region, and begins when the cell growth falls below 75 percent of the uninduced cell growth. Each data point was measured in triplicate on three separate days, and the data represent mean values ± 1 standard deviation.

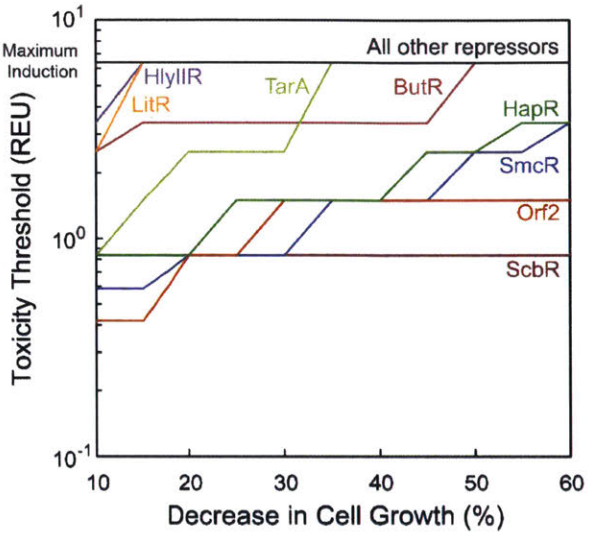

Figure 2-7: Toxic induction threshold versus decrease in cell growth. The highest input level before toxicity is observed is plotted versus the percent decrease in cell growth. For most repressors, toxicity is not observed, and is indicated by the horizontal black line at the top of the graph. HlyIIR and LitR exhibit a 10 percent decrease in growth at high induction levels. The cross-section of the toxicity trajectories at 25% decrease in cell growth for TarA, ButR, HapR, SmcR, Orf2, and ScbR is reflected in the toxic regions of Figure 2-3 and Figure 2-6. Threshold data (y-axis) represents mean maximum induction levels before the growth decreases beyond a mean percentage (x-axis) from three separate experiments.

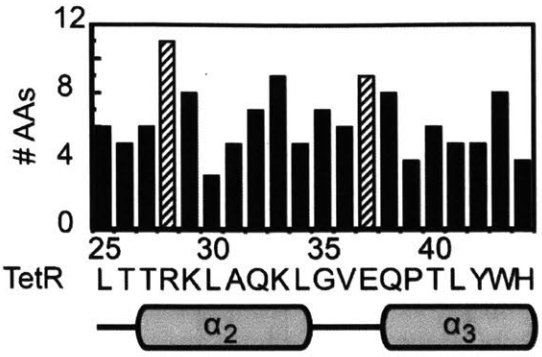

Figure 2-6: DNA-binding domain recognition region diversity. The recognition regions of the DNA-binding domains for all 20 repressors were aligned, and the number of different residues at each position across the set was counted. The wild-type sequence of TetR is shown below the plot for reference, along with the secondary structure of the protein.

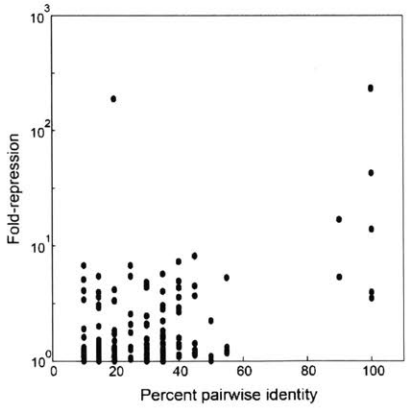

Figure 2-9: Fold-repression versus percent pairwise identity of the recognition region. The fold-repression values of all repressor-promoter pairings are the mean repression values from triplicate orthogonality measurements (Figure 2-2d). These data are plotted versus the corresponding percent pairwise sequence identity of the recognition regions of the repressors' DNA-binding domains.

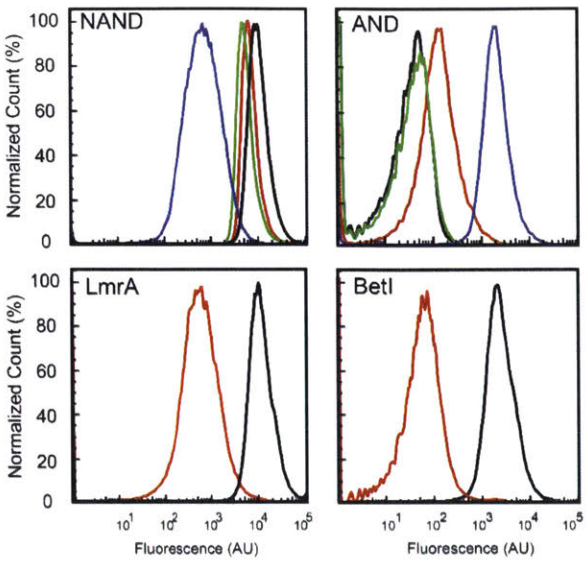

Figure 2-10: Flow cytometry data for logic circuits and terminal gates. Upper panel: Representative fluorescence histograms that correspond to the average fluorescence values in Figure 2-4a, b. For the NAND circuit, the black line corresponds to no inducer, green to 1 mM IPTG, red to 20 μM 3OC6HSL, and blue to the presence of both IPTG and 3OC6HSL. For the AND circuit, the black line corresponds to no inducer, red to 1 mM IPTG, green to 100 ng/mL aTc, and blue to the presence of both IPTG and aTc. Lower panel: Representative fluorescence histograms for repressors connected to circuit outputs. The output distributions for the terminal repressors were taken from response function characterization data, and input levels were chosen such that they approximate the predicted levels seen within the circuits. Each histogram comprises >10000 cells.

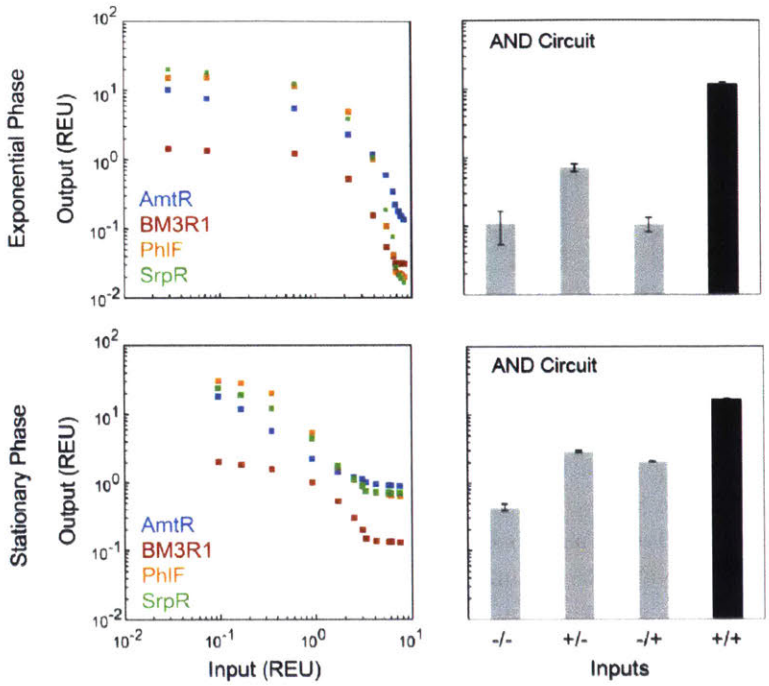

Figure 2-11: Growth phase robustness of repressors and AND gate. Left panel: Response functions for AmtR (blue squares), BM3R1 (red squares), PhlF (orange squares), and SrpR (green squares) measured in exponential phase (top) and stationary phase (bottom) and grown in LB media. Data points represent the geometric mean of a fluorescence histogram at each data point. Right panel: Output values for AND gate measured in exponential phase and stationary phase in LB media. The measured data are grown under conditions of no inducer (-/-), 1 mM IPTG (+/-), 100 ng/mL aTc (-/+), and 1 mM IPTG and 100 ng/mL aTc (+/+). Bars corresponding to the ON and OFF states are colored black and gray, respectively. Data was collected in triplicate on three different days and points represent mean values ± 1 standard deviation.

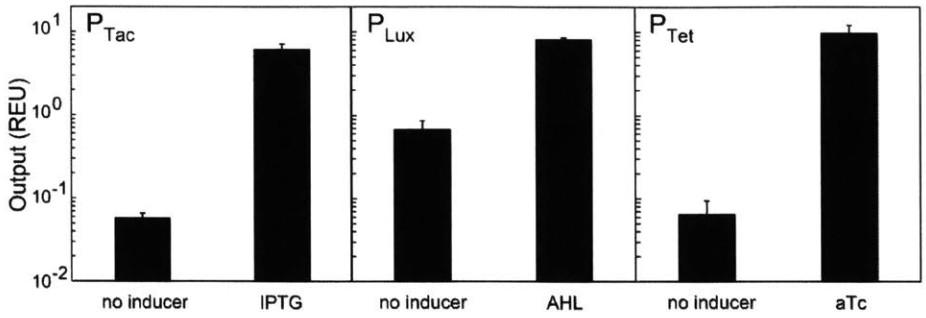

Figure 2-12: Characterization of inducible promoters. Promoters P_{Tac}, P_{Lux}, and P_{Tet} drive yellow fluorescent protein expression and were induced with 1 mM IPTG, 20 μM 3OC6-HSL, and 100 ng/mL aTc, respectively. Cells grown under maximum inducing and non-inducing conditions were measured via cytometry; fluorescence values were normalized by an in vivo reference standard to obtain the promoters' outputs in REU (Figure 2-16). Data was collected in triplicate on three different days and points represent mean values $\pm$ 1 standard deviation.

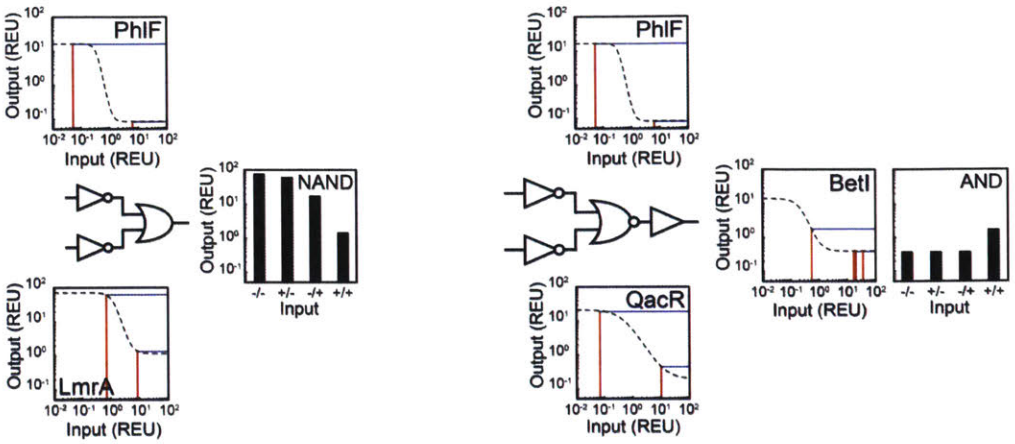

Figure 2-13: Modeling of genetic circuits. For the first layer of gates, experimentally characterized input promoter values (red lines) are mapped onto Hill-equation fits of NOT gate response functions (dashed lines), resulting in predicted output values (blue lines) that feed into the next logic layer. For the NAND gate, the individual NOT gate output values from the first layer are summed to yield the final circuit output. For the AND gate, the individual NOT gate outputs from the first layer are summed to yield the BetI inputs (red lines) that drive the final NOR gate output.

Figure 2-14: NOT gate plasmid maps. These plasmids are used to calculate the response functions shown in Figure 2-3. The Response Function vectors (pRF-) contain an individual repressor, whose expression is controlled by the P_{Tac} inducible promoter (which corresponds to a version of P_{tac1} that has been modified to contain a perfect inverted repeat sequence for the Lac operator). Each NOT gate also contains the cognate promoter for the repressor, which controls expression of the YFP output. The terminator present after the repressor coding sequence corresponds to BBa_B0015, a double terminator consisting of both BBa_B0010 and BBa_B0012 (partsregistry.org). The wild type promoter of the Lac Repressor (labeled P_{Const}) constitutively expresses both LacI and LuxR. These components are maintained on a lower copy number plasmid that was derived from the expression plasmid pEXT20. Activation of repressor expression by IPTG results in repression of the promoter driving YFP (Figure 2-3).

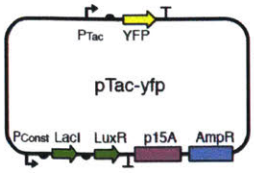

Figure 2-15: Response function input measurement plasmid. To report the response function input as REU, the activity of the input promoter is measured separately.

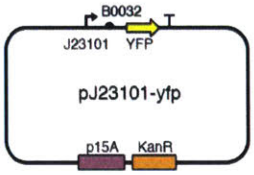

Figure 2-16: The reference plasmid is shown for converting fluorescence units to REU. The fluorescent measurements are normalized by the fluorescence produced from a constitutive promoter (BBa_J23101). The corresponding output, defined as a single REU, serves as the unit to which all other fluorescence values are normalized (Methods).

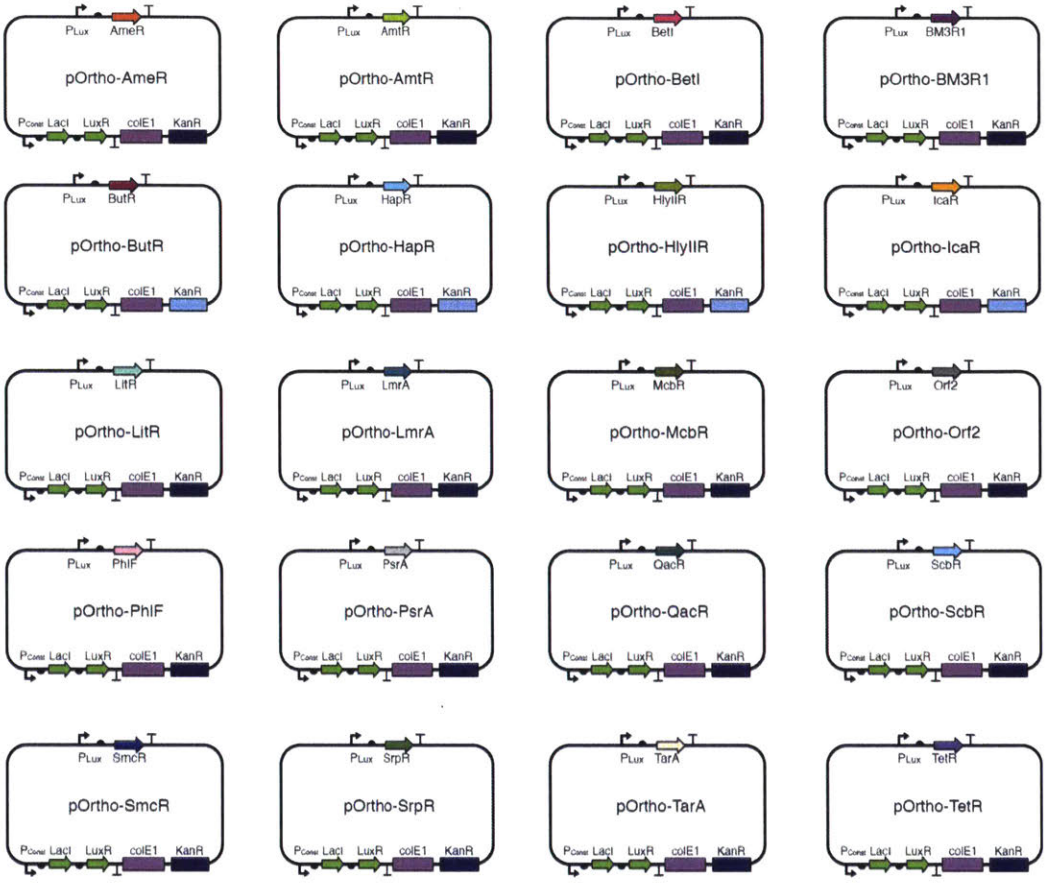

Figure 2-17: Orthogonality measurement plasmids maps. Orthogonality measurements were obtained using two plasmids: one expresses the repressor and the second contains the promoter reporters. In this way, the two sets of plasmids can be co-transformed to build all of the strains required for the orthogonality screen. For the repressor library, each repressor is placed under the control of a 3OC6HSL inducible system (the pOrtho set of plasmids). For the reporters, the same plasmids are used as were built to measure the response functions (Figure 2-14), but the repressors encoded by these plasmids are not induced.

Table 2-1: NOT gate response function parameters

Name	K	n	y_{max} (REU)	y_{min} (REU)	Fold-change[a]
TetR	0.1	2.7	24	0.2	120
QacR	0.5	1.4	21	0.2	105
IcaR(A)	0.4	1.8	13	0.4	33
AmeR	0.5	1.4	17	1.7	11
ScbR	0.2	2.6	5	0.6	8
LmrA	1.2	3.1	70	1.1	64
AmtR	0.2	1.8	9	0.3	30
SmcR	0.1	2.0	13	2.1	6
McbR	0.4	1.6	16	1.1	15
BetI	0.2	2.4	13	0.4	33
SrpR	0.3	3.2	25	0.1	250
Orf2	0.4	6.1	14	0.2	70
BM3R1	0.6	4.5	3	0.1	30
TarA	0.1	1.8	13	0.2	65
PhlF	0.4	4.5	16	0.1	160
ButR	1.3	2.4	12	1.8	7
PsrA	0.4	2.0	20	0.5	40
HapR	0.2	1.4	10	0.9	11
HlyIIR	0.5	2.7	17	0.3	57
LitR	0.1	1.9	16	0.5	32

a. Fold-change was calculated as the ratio of the maximum and minimum output values from the Hill-equation.

Table 2-2: AND circuit modeling

Input[a]		Internal[a]		Output[a]
P_{Tac}	P_{Tet}	P_{PhlF}	P_{QacR}	P_{BetI}
0.06	0.07	16	20	0.4
6.2	0.07	0.1	20	0.4
0.06	9.8	16	0.4	0.4
6.2	9.8	0.1	0.4	1.7

a. All values are in REU.

Table 2-3: NAND circuit modeling

Input[a]		Internal[a]		Output[a]
P_{Tac}	P_{Lux}	P_{PhlF}	P_{LmrA}	P_{PhlF}-P_{LmrA}
0.06	0.7	16	61	78
6.2	0.7	0.1	61	62
0.06	8.2	16	1.4	17
6.2	8.2	0.1	1.4	1.4

a. All values are in REU.

Table 2-4: Degenerate NOT gate repressor RBS sequences

Repressor	RBS Library Sequence[a]
AmeR	CTATGGACTATGTTTTCACANANGANGNGGATTAG**ATG**
AmtR	CTATGGACTATGTTTGANAGANANAATACTAG**ATG**
BetI	GCTACGACTTGCTCATTTGANAGAGGANAANTACTA**GTG**
BM3R1	CTATGGACTATGTTTNAANTACTAG**ATG**
ButR	CTATGGACTATGTTTTCASASRGGARRTACTASG**ATG**
HapR	CTATGGACTATGTTTAAAGAGGANANNTACTAG**ATG**
HylIIR	CTATGGACTATGTTTGAAAGAGGGANAAANACTAN**ATG**
IcaR	CTATGGACTATGTTTTCACACAGGGSCYSG**ATG**
LitR	CTATGGACTATGTTTTCACACAGGTTTTCACACAGRARARRCCTCGATG
LmrA	CTATGGACTATGTTTTCACACAGGAAAGGNCTCG**ATG**
McbR	CTATGGACTATGNAGGANAANTACTAG**ATG**
Orf2	CTATGGACTATGTTTTGAAAGAGGAGAAANNCTAG**ATG**
PhlF	CTATGGACTATGTTTGANANGGANAANTACTAG**ATG**
PsrA	CTATGGACTATGTTTSAMASAGGATACRAMMTACTAG**ATG**
QacR	GCCATGCCATTGGCTTTTCACACAGGACACCGGTTAGTACTAG**ATG**
ScbR	CTATGGACTATGTTTAMASAGGARAMSTACTAG**ATG**
SmcR	CTATGGACTATGTTTSAMASAGGARRRRWWYTMG**ATG**
SrpR	CTATGGACTATGTTTTSAMASAGGAAMTACMAGS**ATG**
TarA	CTATGGACTATGTTTTTSAMASAGGARAMMTACTAG**ATG**
TetR	CTATGGACTATGTTTTCACACAGGAAAGGCCTCG**ATG**

a. Codes are defined as N = A,T,G, or C, S = G or C, R = A or G, Y = T or C, M = A or C, K = G or T, and W = A or T.

Table 2-5: NOT gate repressor RBS sequences

Repressor	RBS sequence
AmeR	CTATGGACTATGTTTTCACATACGAGGGGGATTAG**ATG**
AmtR	CTATGGACTATGTTTGAAAGAGAGAATACTAG**ATG**
BetI	GCTACGACTTGCTCATTTGACAGAGGATAACTACTA**GTG**
BM3R1	CTATGGACTATGTTTTAACTACTAG**ATG**
ButR	CTATGGACTATGTTTTCACACAGGAAATACTACG**ATG**
HapR	CTATGGACTATGTTTAAAGAGGACACATACTAG**ATG**
HylIIR	CTATGGACTATGTTTGAAAGAGGGACAAACACTAA**ATG**
IcaR (A)	CTATGGACTATGTTTTCACACAGGGGCCGG**ATG**
LitR	CTATGGACTATGTTTTCACACAGGGTTTTCACACAGGAGAAACCTCG**ATG**
LmrA	CTATGGACTATGTTTTCACACAGGAAAGGCCTCG**ATG**
McbR	CTATGGACTATGTAGGAGAAATACTAG**ATG**
Orf2	CTATGGACTATGTTTTGAAAGAGGAGAAACACTAG**ATG**
PhlF	CTATGGACTATGTTTGAAAGGGAGAAATACTAG**ATG**
PsrA	CTATGGACTATGTTTGAAAGAGGATACGAACTACTAG**ATG**
QacR	GCCATGCCATTGGCTTTTCACACAGGACACCGGTTAG**ATG**
ScbR	CTATGGACTATGTTTAAAGAGGAAAAGTACTAG**ATG**
SmcR	CTATGGACTATGTTTGAAAGAGGAGAAATACTAG**ATG**
SrpR	CTATGGACTATGTTTTCACACAGGAAATACCAGG**ATG**
TarA	CTATGGACTATGTTTTCAAAGAGGAGAAATACTAGATG
TetR	CTATGGACTATGTTTTCACACAGGAAAGGCCTCG**ATG**

Table 2-6: Native operator sequences

Repressor	Operator Sequence
AmtR	TTTCTATCGATCTATAGATAAT
BetI	ATTGATTGGACGTTCAATATAA
BM3R1	CGGAATGAACGTTCATTCCG
HapR	TTATTGATTTTTAATCAAATAA
HlyIIR	ATATTTAAAATTCTTGTTTAAA
IcaR (A)	TTCACCTACCTTTCGTTAGGTTA
LmrA	GATAATAGACCAGTCACTATATTT
PhlF	ATGATACGAAACGTACCGTATCGTTAAGGT
SmcR	TTATTGATAAATCTGCGTAAAAT
TetR	TCCCTATCAGTGATAGA

Chapter 3

3 3 Multi-input CRISPR/Cas genetic circuits that interface host regulatory networks

3.1 Background

Genome editing has been revolutionized by the RNA-guided endonuclease Cas9 from *Streptococcus pyogenes* due to its ability to target DNA sequences adjacent to 'NGG' motifs using a guide RNA (Cong *et al*, 2013; Jiang *et al*, 2013; Wang *et al*, 2013a; Esvelt *et al*, 2013; Zhou *et al*, 2014; Shalem *et al*, 2013). This programmability has been harnessed for gene regulation using a Cas9 double mutant that eliminates nuclease activity (dCas9) so that guide RNAs cause it to bind tightly to the corresponding DNA sequence without cleaving it (Jinek *et al*, 2012). This complex can serve as a repressor by blocking RNAP binding to a promoter or by terminating transcription (Qi *et al*, 2013; Bikard *et al*, 2013; Esvelt *et al*, 2013). A chimeric small guide RNA (sgRNA) is sufficient to drive Cas9 to a target (Jinek *et al*, 2012), and it comprises a complementary domain that binds to the DNA followed by a "handle" that is bound by Cas9. Considering the programmability of DNA:RNA interactions and the existence of a "seed" region at the 3'-end of the sgRNA's complementary region, this system could yield ~10^7 orthogonal sgRNA:DNA pairs. This is a potentially versatile

platform for building genetic circuits, which have been limited in size and sophistication by the number of available orthogonal transcription factors.

Extensible circuits, whose inputs and outputs are of an identical form, can be connected in different ways in order to perform user-defined computational operations (Nielsen *et al*). For genetic circuits, the simplest way to achieve this is to design gates whose inputs and outputs are both promoters (Moon *et al*, 2012b; Tamsir *et al*, 2011a; Stanton *et al*, 2014). In this formalism, the common signal carrier is RNAP flux and gates are connected by having the output of one serve as the input to the next. The majority of transcriptional gates have been built using DNA-binding proteins. The challenge has been to obtain large sets of orthogonal proteins that do not cross-react with each other's binding sites. These sets can be obtained either by part mining, where bioinformatics is applied to search databases for classes of regulators that are synthesized and screened (Moon *et al*, 2012b; Stanton *et al*, 2014; Rhodius *et al*, 2013), or by building variants of modular DNA-binding proteins whose domains can be engineered to target different operators (e.g., ZFPs (Beerli & Barbas, 2002b; Miller *et al*, 2007) and TALEs (Morbitzer *et al*, 2010b; Miller *et al*, 2011)). For both approaches, cross-reactions are prevalent and many variations have to be screened to obtain an orthogonal core set. Another challenge is that within a regulator class, some can be non-toxic whereas others exhibit extreme toxicity (Kimelman *et al*, 2012; Stanton *et al*, 2014). Collectively, restrictions on function, orthogonality, and toxicity reduce the size of the libraries dramatically; for example, an initial set of 73 TetR homologues was reduced to 16 repressors (Stanton *et al*, 2014).

Here, we build a set of transcriptional gates based on sgRNA-guided repression of a synthetic *E. coli* σ_{70} promoter (Figure 3-1a). The input to the sgRNA NOT gate is a promoter that contains a precise transcription start site (+1) so that additional nucleotides are not added to the 5'-end of the sgRNA, which has been shown to reduce

activity (Larson *et al*, 2013). The sgRNA includes a guide region that targets dCas9 to the cognate bacterial promoter. A strong terminator (Qi *et al*, 2013; Chen *et al*, 2013) is placed after the sgRNA to stop transcription. The output is an *E. coli* constitutive promoter (BBa_J23101) that has been modified to include both forward and reverse 'NGG' PAMs (for targeting either the template or non-template strands of the promoter), and a unique 13bp "operator" region between the -35 and -10 $\boxtimes_{70}$ binding sites (Figure 3-2c). The entire transcription unit (promoter, sgRNA, and terminator) can be constructed from a pair of ≤200nt single-stranded DNA oligonucleotides that are annealed and extended at the dCas9 handle region. These ssDNA oligos also encode Type IIs restriction enzyme recognition sites that flank the transcription unit. The resulting dsDNA modules can then be combined into a final circuit plasmid using a one-pot Golden Gate assembly reaction (Engler *et al*, 2009) (Figure 3-1b).

Multi-input NOR and NAND gates are "Boolean complete" and are each sufficient to build any user-defined computational operation (Katz & Boriello, 2004). Transcription factor-based NOR gates have previously been built by placing two input promoters in series upstream of a repressor gene (Tamsir *et al*, 2011a; Stanton *et al*, 2014). Without additional RNA processing, this design does not work for sgRNA-circuits because of the detrimental influence of 5'-mismatches (Larson *et al*, 2013) and the 'roadblocking' effect of CRISPRi (small for template-targeting sgRNAs, substantial for non-template-targeting sgRNAs) (Qi *et al*, 2013). Hammerhead ribozymes and endoRNase cleavage of 5'-mismatches have both been shown to effectively remove extraneous 5'-RNA from sgRNAs (Gao & Zhao, 2014; Nissim *et al*, 2014) and could be employed in multi-input dCas9 circuits. Instead, our design is based on two transcription units, each of which contains a different input promoter. When either promoter is active, the sgRNA is transcribed and represses the output promoter. This

design allows larger circuits to be constructed simply by changing the pattern of input and output promoters around the sgRNAs. This approach requires that the sgRNAs be able to be layered into a cascade, which has been shown to work in mammalian cells (Kiani *et al*, 2014; Nissim *et al*, 2014).

Linking the output(s) of a genetic circuit to regulate host genes provides control over cellular responses. For example, cells could be programmed to sense the cell density in a fermenter and respond by expressing enzymes to redirect flux through global metabolism (Nielsen *et al*, 2014). Similarly, the cell phenotype could be controlled, like the ability to swim or associate into biofilms. Various approaches have been taken to link synthetic circuits to endogenous genes. Church and co-workers used MAGE to insert T7 RNAP promoters upstream of genes participating in lycopene biosynthesis and upregulated production by expressing the polymerase as a circuit output (Wang *et al*, 2009). Natural and synthetic sRNAs have been used to knockdown endogenous genes involved in motility (Sharma *et al*, 2013), iron metabolism (Kang *et al*, 2012), acetone-formation (Tummala *et al*, 2003), β-glucuronidase (Man *et al*, 2011), membrane porin and flagellin genes (Sharma *et al*, 2012), and to increase tyrosine and cadverine production (Na *et al*, 2013). Finally, strains have been constructed that express a protein that can be targeted to the genome (ZFP (Beerli & Barbas, 2002b), TALE (Zhang *et al*, 2011; Morbitzer *et al*, 2010b), or dCas9 (Gilbert *et al*, 2013; Farzadfard *et al*, 2013; Qi *et al*, 2013)) to upregulate or knockdown endogenous genes. Here, we link the synthetic dCas9-based circuits to the native *E. coli* regulatory network by designing the final sgRNA in a circuit to target a transcription factor on the host genome. This provides a generalizable mechanism by which the same biochemistry is used to both perform computation and also actuate host phenotype in response to conditions defined by the circuitry (Figure 3-1c).

3.2 Orthogonal NOT gates based on dCas9 and sgRNAs

A three-plasmid system was built to measure sgRNA orthogonality and characterize their performance in the context of a gate (Figure 3-2a). The first plasmid controls the expression of *S. pyogenes* dCas9 from an aTc-inducible P_{Tet} promoter. The sgRNA is carried on a high-copy plasmid and transcribed using a variant of the arabinose-inducible P_{BAD} promoter that is truncated to end at the transcription start site (+1). Finally, the output promoter repressed by the dCas9-sgRNA is transcriptionally fused to red fluorescent protein (RFP) and carried on a low-copy plasmid.

dCas9 can exhibit toxicity when overexpressed. To reduce background expression, we selected an aTc-inducible P_{Tet} variant that exhibits low leakiness and added the strong L3S3P21 terminator (Chen *et al*, 2013) upstream to block read-through transcription. As the expression of dCas9 is increased, higher fold-repression is observed but this comes at the cost of reduced cell growth (Figure 3-2b). These effects are balanced at 0.625 ng/ml aTc, which elicits near-full repression with a growth impact of less than 15 percent (after 6 hours, an OD_{600} of 0.44 versus 0.51). This induction level is used for all subsequent experiments.

A set of five synthetic promoters (P_{A1}-P_{A5}) were designed to be targeted by corresponding sgRNAs. An *E. coli* constitutive promoter (BBa_J23101) was chosen as a scaffold and the operator that is recognized by the sgRNA was inserted between the -35 and -10 consensus sites where the housekeeping σ_{70} binds (Figure 3-2c). The region between these sites is 17bp, the center of which contains a unique 13 bp sequence that is bound by the "seed" of the sgRNA complementary region, which is less tolerant of RNA-DNA mismatches (Jinek *et al*, 2012). This is flanked by forward and reverse 'NGG' protospacer adjacent motifs (PAMs), which are required for dCas9 binding (Marraffini & Sontheimer, 2010). When dCas9 is directed to this region by a

corresponding sgRNA, the promoter is repressed by sterically blocking the binding of *E. coli* RNAP. The orthogonal sgRNAs were designed by selecting a set of five distinct 13bp seed sequences that have no matches to PAM-proximal sequences in the *E. coli* genome. Two variants of each sgRNA were built that target the non-template (—NT) and template (—T) strands of each promoter. Each of the sgRNAs strongly represses its target promoter (56- to 440-fold), with no preference for the non-template or template strand, as observed previously (Bikard *et al*, 2013). The orthogonality of the promoters and sgRNAs are near-perfect, with essentially no off-target interactions (Figure 3-2d). In addition, we observe only a small amount of toxicity when the sgRNAs are highly expressed, and no growth differences between the sgRNA variants.

The response function of a gate captures how the output changes as a function of input. This is critical in predicting how gates can be connected to form larger circuits. To characterize the gates, the P_{BAD} promoter serves as the input, which we characterized separately as a function of arabinose concentration. This is used to rescale the data to report it as a function of promoter activity, as opposed to inducer concentration (Figure 3-2e). The log-linear shape of this response curve is approximated well by a power law, and is very different from those observed from similar gates based on transcription factors, which saturate as a Langmuir isotherm. This log-linearity is also evident when observing the relationship between the intermediate and output promoters of an sgRNA cascade (Figure 3-3b, right).

The dynamics of repression were also measured (Figure 3-2f). After induction, there is an initial delay of 1.5 hours corresponding to the activation of P_{Tet}/P_{BAD} and the accumulation of dCas9/sgRNA. After this delay, there is a consistent exponential decline in RFP ($t_{1/2}$ = 33 min) over seven hours, which is consistent with the dilution rate of the reporter expected from cell division.

3.2.1 Measurement of NOT gate response functions

The response function of a NOT gate captures how the output promoter changes as a function of the input promoter. Because the gate is measured using an inducible promoter (in our case arabinose-inducible P_{BAD}), the concentration of inducer has to be exchanged for the activity of the inducible promoter(Anderson *et al*, 2007). To do this, the activity of P_{BAD} is measured as a function of [arabinose] and this is used to rescale the input (x-axis of the function). For example, to generate the response function for sgRNA-1T (Figure 3-2e), we induced cells harboring pAN-P_{BAD}-sgRNA-A1T, pAN-P_{A1}-RFP, and pAN-P_{Tet}-dCas9 in 0.625 ng/mL aTc and various arabinose concentrations, and then performed flow cytometry (Figure 3-5, bottom panel). Additionally, in order to determine what the underlying activity of P_{BAD} was in these experiments, we induced cells harboring pAN-P_{BAD}-YFP in an identical manner (Figure 3-5, top panel).

A plot of P_{BAD}-YFP as a function of arabinose shows the plateaus of the promoter at low and high arabinose concentrations (Figure 3-6a). Similarly, a plot of P_{A1}-RFP as a function of arabinose shows a similar plateauing at high and low concentrations due to the underlying P_{BAD} saturation (Figure 3-6b). In order to visualize the relationship between P_{BAD} activity and P_{A1} activity, we convert the x-axis of Figure 3-6b to units of P_{BAD}-YFP (Figure 3-6c). The response functions for all sgRNAs with their cognate promoters are shown in Figure 3-7.

3.2.2 Cytometry data for sgRNA orthogonality

The programmability of RNA-DNA interactions potentially allows for a large number of orthogonal sgRNAs and cognate promoters to be designed. Figure 3-8 shows the raw data for the full orthogonality grid shown in Figure 3-2d. Although each template sgRNA shares its six 5'-nucleotides with every other template sgRNA in order

to bind the -35 σ_{70}-binding site of the promoter (similarly for the non-template sgRNAs and the -10 σ_{70}-biding site), the subsequent twelve 3'-nucleotides are unique and comprise a "seed" region that does not tolerate mismatches.

3.2.3 Design of sgRNA sequences

Each sgRNA was designed so that the first eight nucleotides of the guide region bind the -35 or -10 sites (for template and non-template targeting sgRNAs, respectively) followed by a 'CC' for the opposite strand's PAM. The subsequent twelve nucleotides of the guide region bind the promoter-specific sgRNA operator for each promoter P_{A1} through P_{A5}. Tables 3-1 and 3-2 list the sequences and fold-repression values for sgRNA NOT gates. Fold-repression values were calculated from the orthogonality grid experiment, and represent the RFP output for the uninduced state (no aTc, no arabinose) divided by the RFP output of the fully induced state (0.625 ng/mL aTc and 2 mM arabinose).

3.2.4 Comparison of response functions from gates based on sgRNA and TetR-family repressors

Figure 3-9 shows a comparison of response functions. Previously, we measured the response functions for a library of NOT gates based on TetR-family repressors. The average of 14 response functions is shown in Figure 3-9 (green line) along with the highest (LmrA) and lowest (BM3R1) individual response functions (Stanton et al, 2014). The average line was generated by calculating the average of a set of parameters (half-max threshold, Hill coefficient, maximum and minimum) and then generating a line corresponding to these parameters. The purple line is the power law fit to the sgRNA response function from Figure 3-3b. In Figure 3-9a, the average line, LmrA,

and BM3R1 y-axis values are scaled by the maximum output value of the average line. Simiarly, the sgRNA output values are scaled by its maximum output value. The x-axis values are scaled by maximum values measured for the input promoter. In Figure 3-9b, both the y-axis and x-axis are scaled so that input and output range for both lines spans from 10^{-3} to 10^{0}. This is done to both show the difference in dynamic range and overall shape of the response functions.

3.2.5 Toxicity of sgRNA expression

High expression of dCas9 can be very toxic to the host cell (Figure 3-2b). To determine the toxicity of sgRNA expression, we induced the expression of sgRNAs at various levels and measured the optical density after six hours. Both dCas9 and RFP were expressed in the cells as well. Two sgRNAs were tested: 1) an sgRNA that targets an operator in an otherwise functionless region of the high-copy sgRNA plasmid (blue squares), and 2) a scrambled sgRNA that does not target any DNA sequence in the cell (red squares). Only a slight decrease in the growth is observed and both the functional and scrambled sequence have identical behaviour (Figure 3-10).

3.3 Circuits based on layered sgRNA gates

The advantage of transcriptional gates is that they can be easily interconnected in order to build more complex circuit functions. Gates where repression is based on a non-coding RNA (ncRNA) can be challenging to connect in series for three reasons. First, they require more precision in the promoter start site or additional RNA processing due to sensitivities in the addition or removal of nucleotides at the 5'-end. Second, changing the ribosome binding site (RBS) has been an important lever for functionally connecting protein-based gates. The RBS is not relevant for an ncRNA-

based gate and matching gate responses by promoter tuning is more challenging. This is exacerbated by the shape of the response functions for the sgRNA-based gates, which do not plateau at high or low input promoter levels (Figure 3-2e); therefore the input to any gate needs to have a very wide dynamic range in order to avoid signal degradation at each layer. However, despite these challenges, sgRNA-mediated repression has desirable properties that other ncRNA technologies do not possess, such as high dynamic range, specificity, and the ability to be composed into cascades (Qi & Arkin, 2014).

The layering of two NOT gates based on sgRNAs has been previously demonstrated in mammalian cells (Kiani *et al*, 2014; Nissim *et al*, 2014). We started by building a similar circuit architecture by connecting two of our sgRNA-based gates in series in *E. coli* (Figure 3-3a). These were connected simply by combining the parts from the sgRNA-A2NT and sgRNA-A4NT gates in the appropriate order with no additional tuning. dCas9 is induced from a low-leakage variant of P_{Tet}, as was done for the characterization of individual gates. In the absence of dCas9, the background activity of the output promoter (P_{A4}) is 1040 au (arbitrary units, Figure 3-3b, leftmost bar). When dCas9 and the input to the circuit (P_{PhlF}) are both induced, this lead to a 98-fold repression of the circuit output (P_{A4}) compared to no sgRNA production (Figure 3-3b, left). When the input promoter is induced with DAPG, the output state recovers completely to the level of the dCas9 (—) control. By observing the middle promoter (P_{A2}) in the cascade in a separate experiment, the trade-off between P_{A2} and P_{A4} expression can be seen at intermediate sgRNA induction levels (Figure 3-3b, right). The log-linearity of the curve spans almost three orders of magnitude.

In addition to layering, the construction of more complex circuits requires that gates be able to receive multiple inputs. So-called "Boolean complete" logic gates – NOR and NAND functions – are particularly useful because they can be connected to

build any computational operation. Genetic NOR gates have proven to be particularly easy to build using transcriptional regulation where two input promoters drive the expression of a repressor that turns off an output promoter. The capacity for the orthogonality of sgRNA:promoter interactions has the potential to enable a very large number of NOR gates, which could be used to realize large integrated circuits. However, to date, it has not been shown that dCas9-based gates can be designed to respond to more than one input promoter.

To build a simple NOR gate, we connected two input promoters to the transcription of independent copies of sgRNA-2NT (Figure 3-3c), either of which will repress a single output promoter (P_{A2}). These two input promoters are responsive to small molecule inducers: DAPG (P_{PhlF}) and arabinose (P_{BAD}). In the presence of dCas9, but neither arabinose nor DAPG, the NOR gate output from promoter P_{A2} remains high at only 2.3-fold reduction compared to the dCas9 (—) control due to leaky sgRNA production. When both inducers are added, there is 100-fold repression of the output promoter (Figure 3-3d, left), which is on par with the best gates that use protein-based repressors. The OFF state is ~3-fold higher when only arabinose is added, which is likely due to the lower maximum activity from the P_{BAD} promoter as compared to P_{PhlF}. While this does not significantly degrade the function of the NOR gate alone, it is representative of the sensitivity of sgRNA-based gates to the dynamic range of the inputs and is potentially problematic when building longer cascades.

Next, we connected multiple NOR and NOT gates to build larger layered circuits. First, we built a simple circuit that inverts the output of the NOR gate to make an OR gate (Figure 3-3e). The P_{A2} output of the NOR gate is used to drive the transcription of sgRNA-A4NT, which in turn represses the P_{A4} output promoter. A challenge that emerged from building these circuits is transcriptional readthrough, which occurs because the output promoters are strong and the sgRNAs short. To

mitigate this, strong unique terminators (Chen *et al*, 2013) are placed after each sgRNA, immediately downstream from the dCas9 handle and *S. pyogenes* terminator regions of the sgRNA (Qi *et al*, 2013). For the OR gate, the TrrnB and L3S2P55 terminators (terminator strengths, Ts = 84 for TrrnB and Ts = 260 for L3S2P55, respectively (Chen *et al*, 2013)) are placed after the two sgRNA-A2NT sequences and L3S2P21 (Ts = 380) is placed after sgRNA-A4NT. The output of the OR gate is strongly repressed >100-fold in the absence of both inducers (Figure 3-3f).

We then built a larger circuit by connecting three gates based on four sgRNAs. A cascade with two branches is formed by the A2NT and A4NT sgRNAs, which invert the output of the arabinose- and DAPG- inducible systems, respectively (Figure 3-3g). The output promoters from these NOT gates then connect to a NOR gate by using each to drive a different copy of sgRNA-A1NT. The computing portion of the circuit requires 1234nt to encode. This circuit should produce an AND logic operation and, indeed, there is a 107-fold difference between the OFF and ON states when both inducers are absent and present (Figure 3-3h). There is some leakiness when either input is induced alone and these states show 2.6- to 5.0-fold activity above the OFF state observed in the absence of both inducers. Four versions of this circuit were designed with varied sgRNA positions and orientations. Other versions were slightly less functional, with higher OFF states and lower ON state; the best version is presented here. This circuit can be compared to a similar AND gate design built from TetR homologues. That circuit generated a ~5-fold response and required 2577nt to encode (Stanton *et al*, 2014).

3.3.1 Cytometry data for genetic circuits

Representative fluorescence histograms corresponding to the five input states for the genetic circuits of Figure 3-3b (NOT-NOT), 3d (NOR), 3f (OR), 3h (AND) and

Figure 3-4b (NOR from OR-MalT-3NT) are shown (Figure 3-11). Black histograms correspond to no induction of dCas9 and reflect the "maximum output" achievable. Colored histograms each have dCas9 induced and correspond to the four digital induction conditions for expressing input promoters.

3.4 Interfacing a circuit with a native *E. coli* regulatory network

Guide RNAs can be designed to knock down genes encoded in the host genome (Qi *et al*, 2013). In this way, native cellular processes can be easily actuated as an output of an sgRNA-based circuit using the same biochemistry. To demonstrate this, we started with the OR circuit (Figure 3-3e) and substituted the sgRNA used for the NOT gate with one designed to target the *malT* gene in the *E. coli* genome (Figure 3-4a). MalT is a positive regulator of the maltose utilization operons. A knockdown would alter sugar utilization and has additional impacts on the cellular phenotype (Boos & Böhm, 2000; Tchetina & Newman, 1995). Notably, it decreases the production of LamB — the lambdaphage receptor — resulting in decreased susceptibility of *E. coli* to lambdaphage infection (Thirion & Hofnung, 1972). To target *malT*, we designed sgRNA-MalT-3NT to target the non-template strand of the protein coding sequence from the 110^{th} to the 117^{th} codon. By targeting the non-template strand, the roadblock formed by dCas9 would disrupt any transcription from upstream promoters (Qi *et al*, 2013; Bikard *et al*, 2013).

Cells harboring this circuit exhibit a 240-fold reduction in lambda plaque formation in the absence of both inducers (Figure 3-4c). When either or both inducers are present, the cells show wild-type phage infectivity. In addition, we can separately report the activity of an internal state of the circuit by using P_{A2}, which is the output of the

NOR gate alone, to drive the transcription of a fluorescent reporter (RFP). This results in a NOR gate that is repressed 120-fold when either inducer is present (Figure 3-4b). These experiments demonstrate that a heterologous output (knockdown of RFP) and an endogenous response (knockdown of MalT) can be simultaneously co-regulated according to different logic operations using the same underlying circuit.

3.5 Discussion

Extensible NOR and NOT gates are fundamental logic operations from which more complex circuitry can be built. Previously, these gates have been based transcription factors that bind to DNA, such as phage repressors, LacI, and TetR homologues. Gates based on dCas9 and guide RNAs offer several advantages. The most significant is the ease by which new sgRNA:promoter pairs can be designed and the orthogonality that they exhibit with each other. While there has been much discussion regarding off-target Cas9 interactions and several efforts seeking to reduce it (Fu *et al*, 2013b; Cradick *et al*, 2013; Pattanayak *et al*, 2013; Hsu *et al*, 2013b; Mali *et al*, 2013; Ran *et al*, 2013; Guilinger *et al*, 2014; Tsai *et al*, 2014; Fu *et al*, 2014; Kuscu *et al*, 2014; Wu *et al*, 2014), this is not as relevant for synthetic circuits because sgRNAs can be designed to be maximally different from each other and the host genome. Indeed, no designed sgRNAs had to be discarded from the orthogonal set that we built, either for activity, orthogonality, or growth defects. Further, one transcriptomic analysis of CRISPR interference revealed no off-target signatures (Qi *et al*, 2013). This is a major improvement over the protein-based gates, which have problems in all of these areas. The "operator" that corresponds to the sgRNA is also relatively small (13bp) and can be easily inserted between the -10 and -35 region of a promoter (TetR homologue operators range from 20-50bp). In addition, the gates are small and can be easily

synthesized as oligos, including in pooled libraries (Kosuri *et al*, 2013). The gates also reliably produce >50-fold dynamic ranges. This is akin to the best protein-based gates, but those exhibit far more diversity with such gates in the leakiness, dynamic range, and shape of the response function.

Toxicity is observed from dCas9, where high levels reduce cell growth in *Escherichia coli*. While the mechanism of toxicity is still unclear, it has been reported to be more severe in other species. This may reduce the long-term evolutionary stability of dCas9 in engineered cells, as has been observed for other toxic genetic circuits (Sleight *et al*, 2010; Sleight & Sauro, 2013; Chen *et al*, 2013). However, we find that the toxicity can be managed by controlling the level of expression while still eliciting a substantial circuit response. Also, once dealt with, we do not observe substantial toxicity as more sgRNAs are transcribed. This is in contrast to protein-based gates, which may have less toxicity individually, but can be problematic if multiple repressors are used in a design because their growth defects often stack and can become severe.

There are also some challenges in working with dCas9 that are unique compared to protein-based gates. The shape of the response function, where no saturation is observed at high or low levels, poses a problem when layering gates. Without non-linearity, the signal is degraded at each layer. Indeed, we attempted to add another layer to the AND gate and this yielded a non-responsive circuit likely for this reason. Because there is no RBS to tune, it is difficult to fix this problem through the rational modification of the gate. No cooperativity also impedes the use of these gates for dynamic and multistable circuits, such as bistable toggle switches, pulse generators, or oscillators. Adding cooperativity could potentially be accomplished through dCas9-dimerization to effect promoter looping, sgRNA feedback latching motifs, or sequestration-based techniques such as "decoy operators" to titrate sgRNA away from cognate promoters. While the graded response could be of value for analog circuit

construction, an inability to change its shape could remain problematic. It may be possible to change the position of the response function by engineering specific mismatches to reduce the effectiveness of repression (Farzadfard *et al*, 2013). In addition, it is more difficult to connect input promoters upstream in series before an sgRNA, which has been a valuable design strategy for protein-based gates. Doing this would both require processing to remove the 5'-mismatch from the sgRNA, and also minimization of transcriptional roadblocking, which could occur at the downstream promoter. Finally, because all of the gates require the same dCas9, this could impose retroactivity in the system where the activity state of upstream gates impacts the performance of downstream gates. An approach to circumvent this for larger circuits may be to use multiple orthogonal Cas9 homologues in a design (Esvelt *et al*, 2013).

It has been challenging to build genetic circuits that are as robust or capable as their natural counterparts. The potential for dCas9 to address this problem is vast. Synthetic sgRNAs can be designed to target a large number of sequences—synthetic and natural—and the sgRNA circuit architecture can be encoded in compact genetic constructs. This could allow the paradigm of analog and digital computing to be applied *in vivo* without requiring large and cumbersome constructs. dCas9 circuits also offer a mechanism whereby the same biochemistry can be used to both build circuitry that is orthogonal to the host and to directly interface host processes by design.

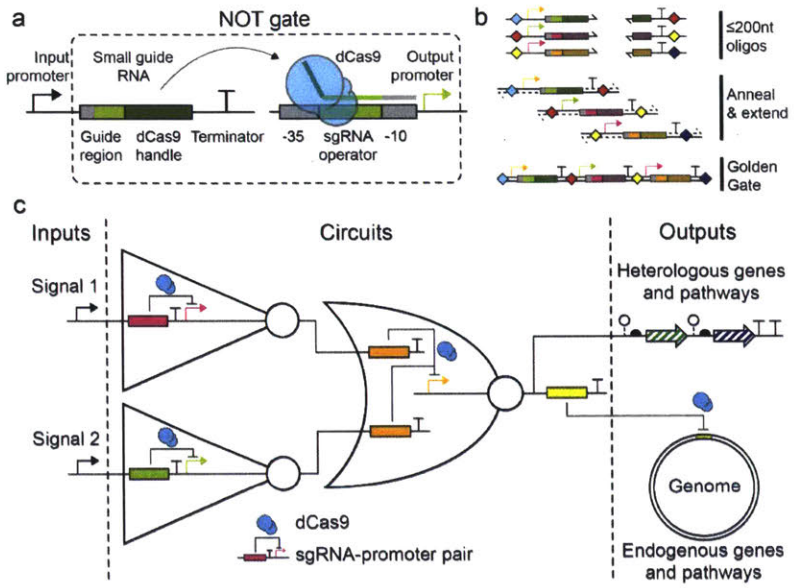

Figure 3-1: Schematic of dCas9 logic circuit design and construction. (A) CRISPR/Cas-based NOT gates comprise a catalytically-dead dCas9 protein, an input promoter that transcribes a small guide RNA (sgRNA), and a synthetic output promoter with an sgRNA operator between the -35 and -10 sigma factor binding sites. When the dCas9 handle of the sgRNA (dark green) complexes with dCas9 (blue), the sgRNA binds the operator (light green) and a sigma factor binding site (gray), causing steric repression of transcription initiation at the output promoter. (B) CRISPR/Cas genetic circuits are easily constructed from pairs of ssDNA oligonucleotides $\leq$ 200nt long that encode the necessary genetic parts (promoter, sgRNA, terminator, assembly scars, and restriction enzyme recognition sites). These oligos are annealed to each other at the dCas9 handle and extended. The resulting dsDNA modules are assembled in a one-pot Golden Gate assembly reaction (colored diamonds are assembly scars). (C) Complex genetic circuits that respond to chemical input signals can be constructed from simple NOT- and NOR-gate motifs. In these circuits, dCas9 (blue) mediates repression of synthetic promoters by programmable sgRNAs (visualized as solid colored rectangles from here on). Both heterologous and endogenous genes can be regulated at circuit outputs by expressing sgRNAs tailored to target transcription initiation or elongation.

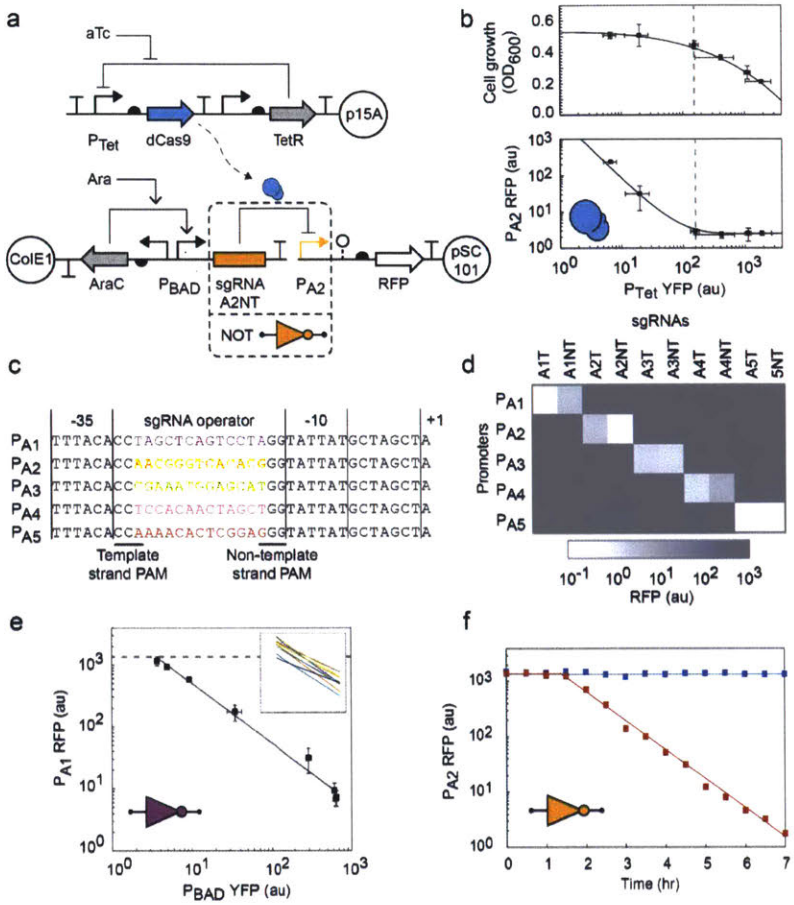

Figure 3-2: Characterization of dCas9 and orthogonal sgRNA NOT gates. (A) The inducible dCas9 and sgRNA system comprises a medium-copy plasmid with P_{Tet}-inducible dCas9, a high-copy plasmid with P_{BAD}-inducible sgRNAs, and a low-copy plasmid encoding a synthetic sgRNA-repressible promoter driving RFP. (B) When sgRNA-A2NT is induced, increasing dCas9 expression causes greater repression of P_{A2} (lower panel), at the cost of decreased cell growth (upper panel). All samples were grown in the presence of 2 mM arabinose. Concentrations of aTc used from left to right (ng/mL): 0.0391, 0.313, 0.625, 1.25, 5, and 10. A single intermediate expression value for dCas9 was used for the remaining experiments (0.625 ng/mL aTc, dashed lines). (C) Synthetic repressible promoters designed by modifying the sequence of promoter BBa_J23101. The -35 and -10 σ_{70} binding sites flank forward and reverse 'NGG' protospacer adjacent motifs (PAMs) and a promoter-specific 13bp sgRNA operator. An sgRNA bound to dCas9 will base-pair with either the template or non-template strand of a promoter's sgRNA operator and one of the σ_{70} binding sites,

causing steric repression of transcription initiation. In the absence of repression, transcription of the downstream RNA begins at the +1 site. (D) The cross-talk map for all combinations of sgRNAs and synthetic promoters is shown. The heat map indicates the amount of RFP observed for that sgRNA-promoter pair. Only cognate pairs of sgRNAs and promoters exhibit significant repression, whereas non-cognate pairs interact negligibly. Samples were grown in the presence of 0.625 ng/mL aTc and 2 mM arabinose. (E) The response function for sgRNA-A1T measured by expressing intermediate levels of sgRNA-A1T reveals a non-cooperative, log-linear relationship between the input and output promoters. The solid line visualizes a power law fit to the data points. Error bars represent the standard deviation of fluorescence geometric mean for three independent experiments on different days. The reporter expression when dCas9 is not induced is shown (dashed line), and all other samples were grown in the presence of 0.625 ng/mL aTc. Concentrations of arabinose used from left to right (mM): 0, 0.0313, 0.0625, 0.125, 0.25, 0.5, 1, and 2. Inset: the power law fits for each of the 10 sgRNAs and their cognate promoters (data presented in Figure S3); axes values are the same as the encompassing figure. (F) The temporal dynamics of dCas9 and sgRNA induction are shown. Red squares indicate induction of both dCas9 (0.625 ng/mL aTc) and sgRNA-A2NT (2 mM arabinose) commencing at t = 0 hrs. Blue squares indicate uninduced cultures. After a ~90 minute delay, fluorescence decreases concomitantly with cell dilution—occurring at a rate of 33 minutes per doubling.

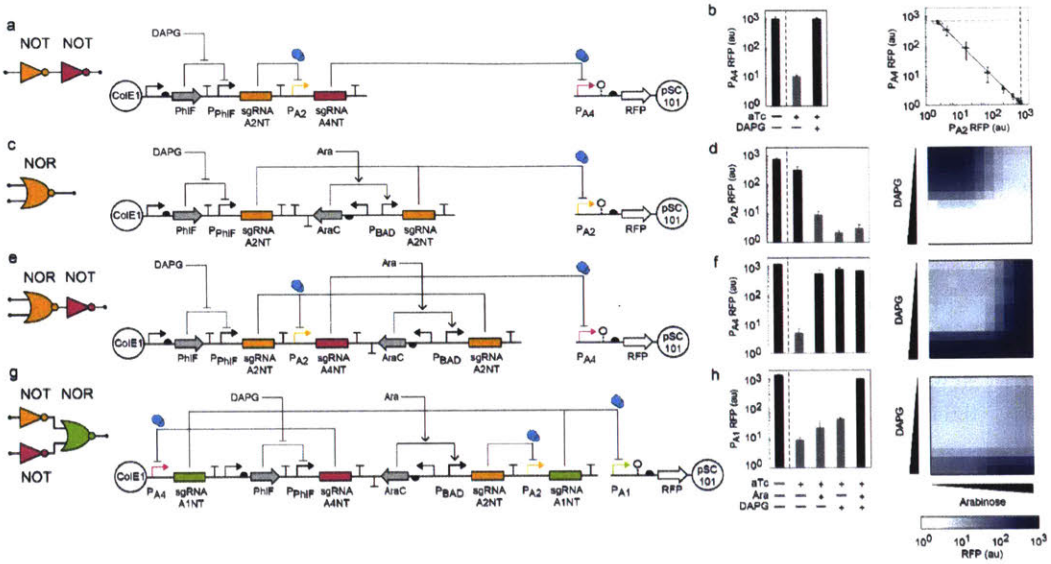

Figure 3-3: Design and characterization of synthetic circuits. (A) The wiring diagram and genetic schematic for a double inverter circuit is shown. The sgRNA-A2NT/P_{A2} pair is shown in orange, the sgRNA-A4NT/P_{A4} pair is shown in magenta, dCas9 is shown in blue, positive regulation is indicated by arrows, and negative regulation is indicated by flat-headed arrows. (B) The digital RFP response of the NOT-NOT gate is shown for the two input inducer states (dCas9 induced with 0.625 ng/mL aTc): no DAPG and 25 μM DAPG. Also shown is the RFP output without dCas9 induction (leftmost column), which represents the maximum achievable output. Gray columns are expected to be OFF and black columns are expected to be ON (left). The trade-off in expression between the middle and output promoters (P_{A2} and P_{A4}, respectively) is shown for intermediate sgRNA induction levels (right). DAPG concentrations from left to right (μM) are: 0, 2.42, 3.39, 4.74, 6.64, 9.30, 13.0, 18.2, 25.5, 35.7, and 50. Dashed lines are uninduced dCas9 control experiments and represent the maximum output for each promoter. Error bars represent the standard deviation of three independent experiments on different days. (C) The wiring diagram and genetic schematic for a NOR(A,B) gate is shown. The sgRNA-A2NT/P_{A2} pair is shown in orange, and dCas9 is shown in blue. (D) The NOR gate digital RFP response is shown (left) for the four input inducer states (with dCas9 induced by 0.625 ng/mL aTc): no arabinose or DAPG, arabinose (2 mM), DAPG (25 μM), and arabinose and DAPG (2 mM and 25 μM). Also shown is the output without dCas9 induction (leftmost column). In addition, the circuit response to intermediate inducer values is shown to the right. (E) The wiring diagram and genetic schematic for a layered NOT(NOR(A,B)) gate (i.e., an OR gate) is shown. The sgRNA-A2NT/P_{A2} pair is

shown in orange, the sgRNA-A4NT/P_{A4} pair is shown in magenta, and dCas9 is shown in blue. (F) The OR digital RFP response is shown (left) for five input inducer states (as in D). Intermediate values are also shown (right). (G) The wiring diagram and genetic schematic for a four sgRNA circuit with NOR(NOT(A),NOT(B)) functionality (i.e., an AND gate) is shown. The sgRNA-A2NT/P_{A2} pair is shown in orange, the sgRNA-A4NT/P_{A4} pair is shown in magenta, the sgRNA-A1NT/P_{A1} pair is shown in green, and dCas9 is shown in blue. (H) The AND gate digital RFP response is shown (left) for five input inducer states (as in D). Intermediate values are also shown (right). For graded induction of circuits in (D), (F), and (H), aTc was added to 0.625 ng/mL; arabinose was added to the following final concentrations (mM): 0, .00391, .00781, .0156, .0313, .0625, 0.125, 0.25, 0.5, 1, and 2; 2,4-diacetylphloroglucinol was added to the following final concentrations (µM): 0, 0.0244, 0.0488, 0.0977, 0.391, 0.781, 1.56, 3.13, 6.25, 12.5, and 25.

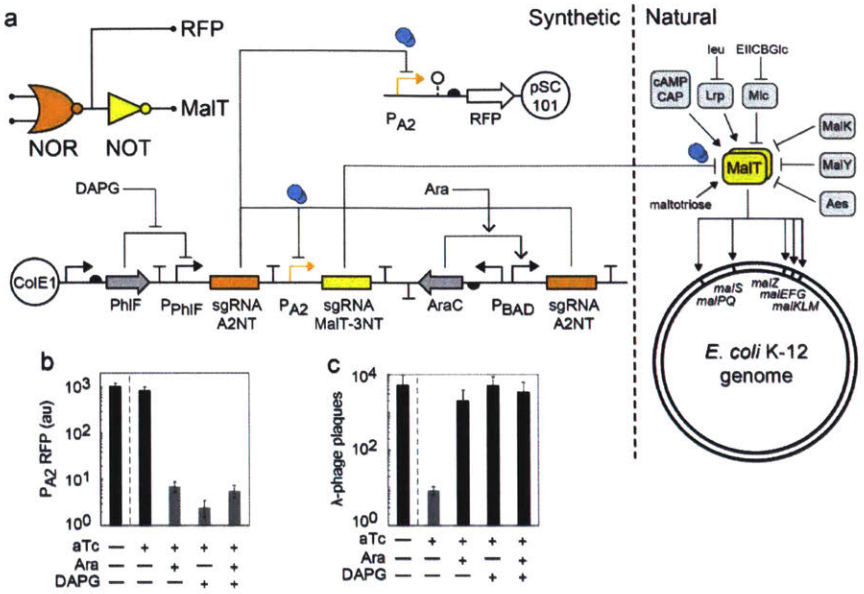

Figure 3-4: Interfacing logic circuits with host physiology. (A) The wiring diagram and genetic schematic for a NOT(NOR(A,B)) gate is shown (i.e., an OR gate). The sgRNA-A2NT/P_{A2} pair is shown in orange, the sgRNA-A4NT/P_{A4} pair is shown in magenta, dCas9 is shown in blue, and both sgRNA-MalT-3NT and the MalT gene are shown in yellow. (B) The NOR gate digital RFP response is shown for the four input inducer states (with dCas9 induced by 0.625 ng/mL aTc): no input inducer, arabinose (2 mM), DAPG (25 μM), and arabinose and DAPG (2 mM and 25 μM). Also shown is the output without dCas9 induction (leftmost column). Gray columns are expected to be OFF and black columns are expected to be ON. Error bars represent the standard deviation of three independent experiments on different days. (C) The OR gate digital lambdaphage infectivity response is shown for five input inducer states (as in B), where infectivity is measured by the number of lambdaphage plaques formed on a bacterial lawn on an agar plate. Error bars represent the standard deviation of three independent experiments on different days.

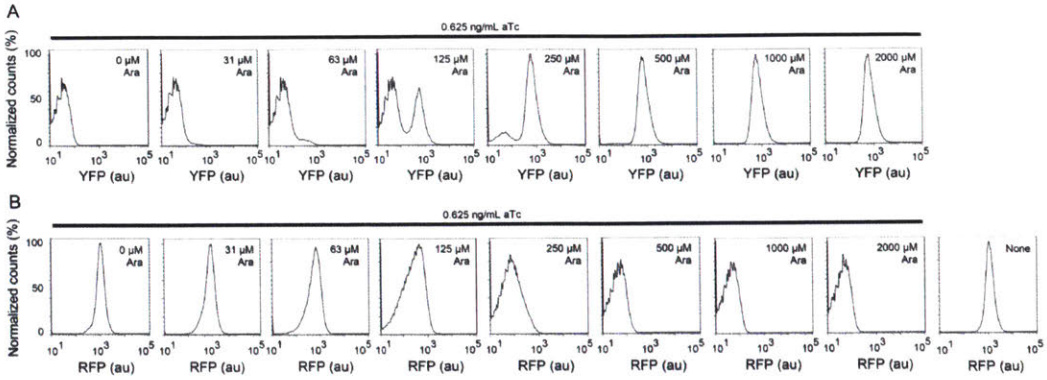

Figure 3-5: Cytometry data used to rescale the response function of the NOT gate based on sgRNA-A1T. (A) YFP histograms for the inducible promoter control, P_{BAD} driving YFP (plasmids pAN-P_{Tet}-dCas9 and pAN-P_{A1}-RFP). All samples were grown in the presence of 0.625 ng/mL aTc to induce dCas9 and the stated amount of arabinose. (B) The raw data for the response function of the NOT gate based on sgRNA-A1T is shown. All samples were grown in the presence of 0.625 ng/mL aTc to induce dCas9 and the stated amount of arabinose, except for the right-most histogram which was grown in the absence of both inducers and provides a maximum achievable reporter output for P_{A1}-RFP. Plasmid maps are shown in Figure S8.

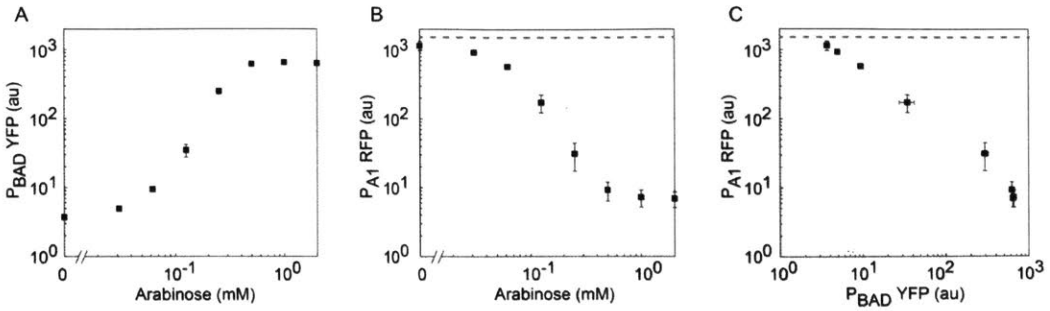

Figure 3-6: Creation of the response function for the NOT gate (sgRNA-A1T). (A) The data shown is for the induction of P_{BAD} and is calculated using the geometric mean of the cytometry data in Figure S1 (top). (B) The activity of P_{A1}-RFP output as a function of arabinose. Dashed lines indicate the maximum achievable RFP output, determine from an experimental treatment where dCas9 was not induced (Figure S1 bottom, rightmost panel). (C) The x-axis of the P_{A1}-RFP plot is transformed to P_{BAD}-YFP units to visualize the relationship between the input promoter that drives sgRNA-A1T and the cognate repressible promoter that drives RFP. Data points represent the average and standard deviation of three experiments.

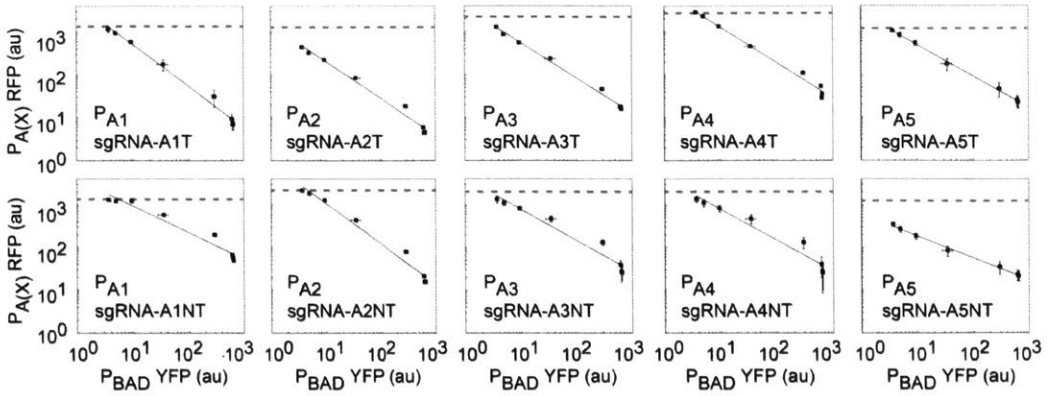

Figure 3-7: Response functions for all of the NOT gates based on orthogonal sgRNAs repressing their cognate promoters. Dashed lines indicate the maximum achievable RFP output, determine from an experimental treatment where dCas9 was not induced. Solid lines are power law fits to the data and correspond to the lines shown in the Figure 2e inset. Data points represent the average of the geometric means of three experiments on different days.

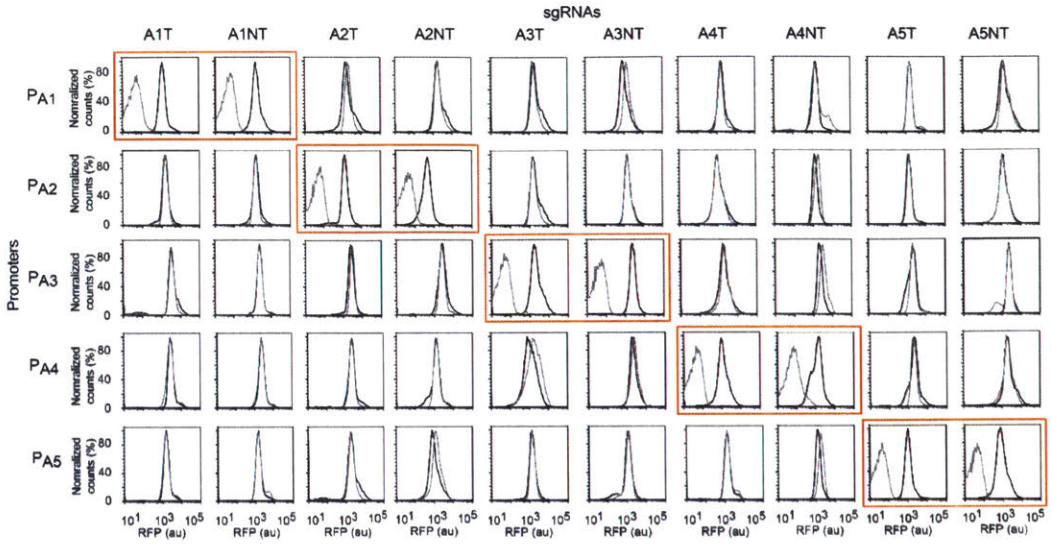

Figure 3-8: Representative cytometry data corresponding to the values used for the cross-talk map in Figure 2d. Histograms are for RFP produced from pAN-$P_{A(X)}$-RFP while being repressed by sgRNAs produced from pAN-P_{BAD}-sgRNA-A(X)(T/NT). Cells also harbor pAN-P_{Tet}-dCas9. Black histograms correspond to dCas9 induction with aTc, but no sgRNA induction. Gray histograms correspond to dCas9 and sgRNA induction with 0.625 ng/mL aTc and 2mM arabinose, respectively. Red boxes indicate cognate sgRNA-promoter pairs.

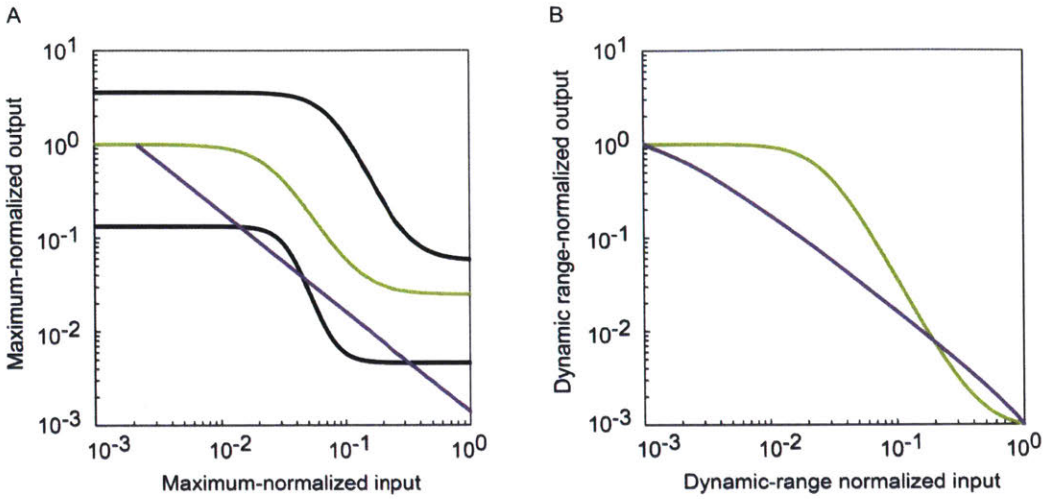

Figure 3-9: Comparison of NOT gate response functions generated by sgRNAs versus TetR-family repressors. Purple lines: Power law fit to the sgRNA cascade relationship from Figure 3b. Green lines: Hill-function generated from the average Hill-equation parameters of 14 TetR homologues (K, n, max, min)[2]. Black lines: Hill-function fits to the highest and lowest response curves, LmrA and BM3R, respectively. (A) For dynamic range comparison, all input and output values are re-scaled so that their maxima equal 1 (except for the black line outputs, which are scaled using the green line maximum). (B) Same as in A, except the y-axis is also normalized by the minimum value to compare the shape of the curves.

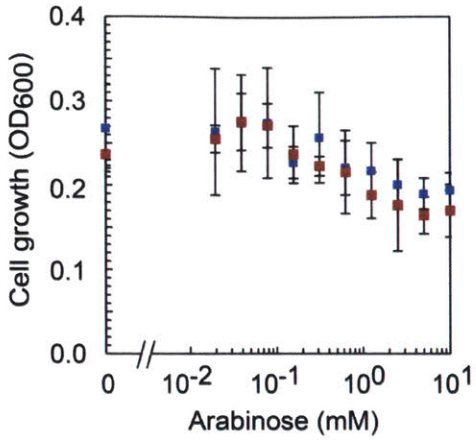

Figure 3-10: Toxicity of sgRNA expression from an arabinose-inducible promoter. Blue squares: expression from pAN-P_{BAD}-sgRNA-VR, which binds an operator on its high-copy plasmid backbone. Red squares: expression from pAN-P_{BAD}-sgRNA-scramble, a "scrambled" sgRNA that does not target any genetic locus in the cell. All samples had dCas9 induced with 0.625 ng/mL aTc, and RFP constitutively expressed.

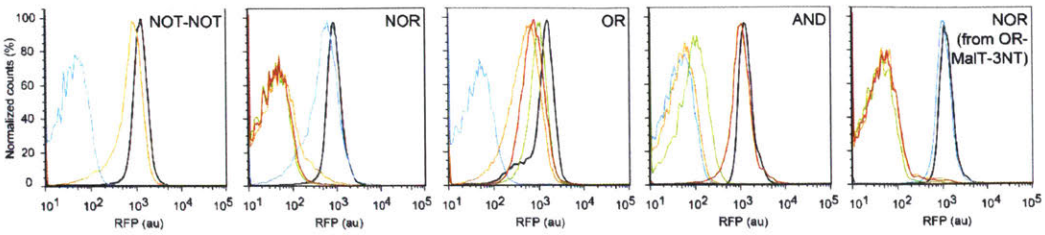

Figure 3-11: Histograms for genetic circuits encoded on pAN-NOT-NOT, pAN-NOR-pAN-OR, pAN-AND, and pAN-OR-MalT-3NT. The black histograms indicate cultures without inducer and correspond to the maximum value achievable for the output promoters, blue is with 0.625 ng/mL aTc, orange is with 0.625 ng/mL aTc and 2mM arabinose, green is with 0.625 ng/mL aTc and 25 µM DAPG, and red is with 0.625 ng/mL aTc, 2mM arabinose, and 25 µM DAPG. The plasmid maps are shown in Figure S8 and S9.

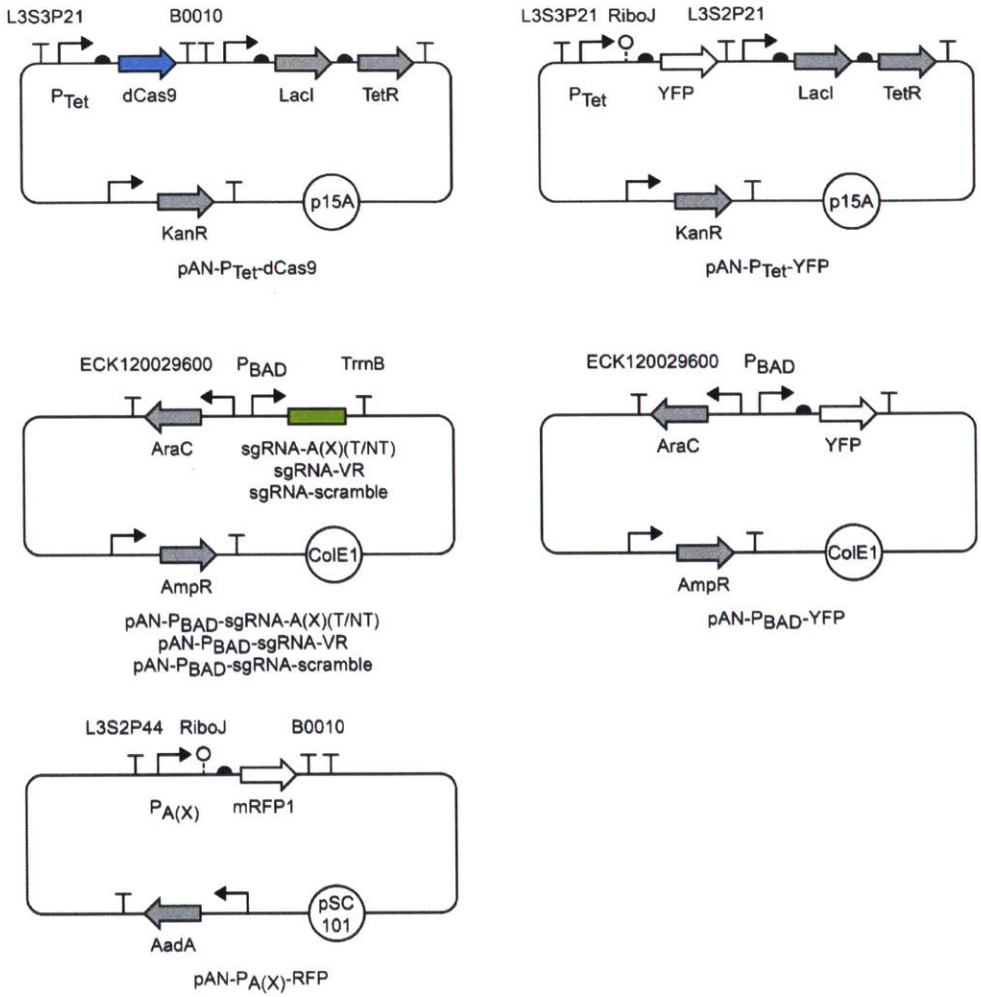

Figure 3-12: Plasmids encoding basic circuit components. The pAN-P_{Tet}-dCas9 plasmid (p15A, KanR) encodes the insulated, tight-off aTc-inducible dCas9 used for all experiments. The pAN-$P_{A(X)}$-RFP series of reporter plasmids (pSC101, AadA) encode one of five synthetic sgRNA-repressible promoters that express mRFP1. The pAN-P_{BAD}-sgRNA-A(X)(T/NT) and pAN-P_{PhlF}-sgRNA-A(X)(T/NT) series of plasmids (ColE1, AmpR) drive one of ten sgRNAs from either the arabinose- or DAPG-inducible promoters, respectively. The pAN-P_{Tet}-YFP and pAN-P_{BAD}-YFP plasmids were used to characterize the promoter activities of P_{Tet} and P_{BAD}, respectively, for dCas9 and sgRNA response functions.

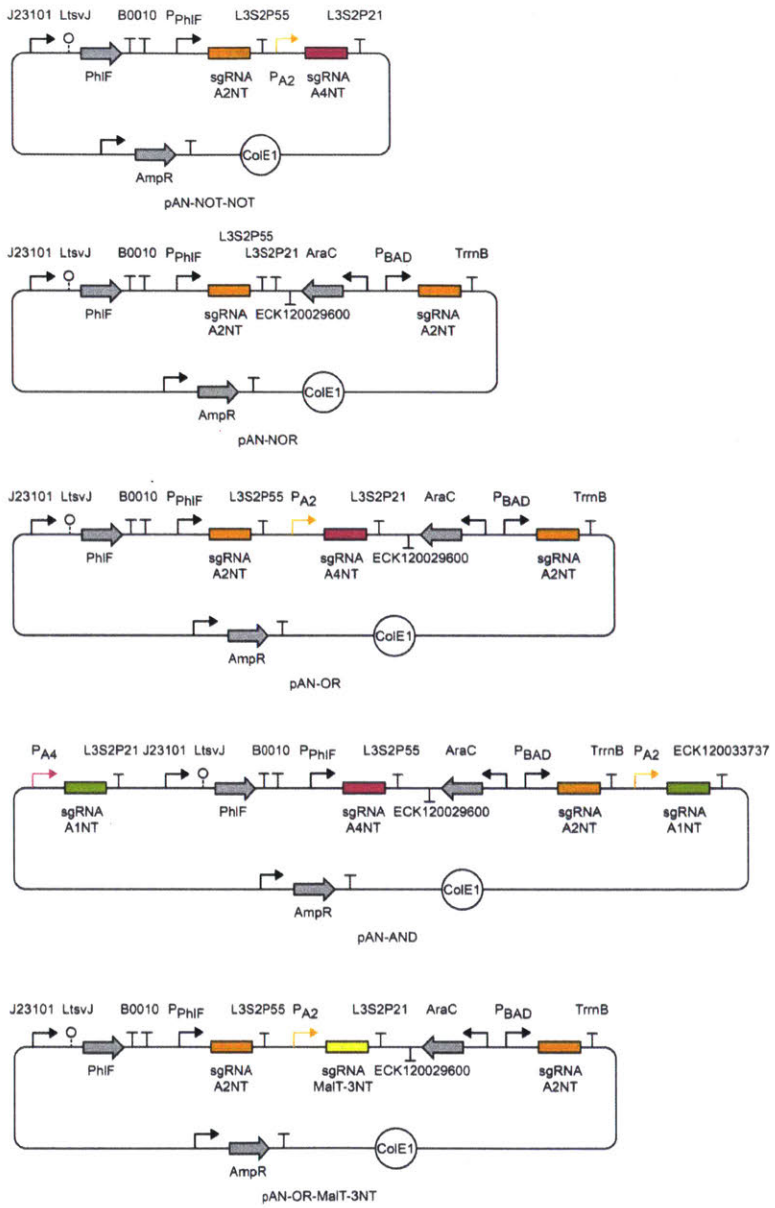

Figure 3-13: Plasmids encoding genetic circuits. The pAN-NOT-NOT, pAN-NOR, pAN-OR, and pAN-AND plasmids encode the sgRNA circuits from Figure 3. The pAN-OR-MalT-3NT encodes the MalT knockdown logic circuit from Figure 4. These plasmids were co-transformed with pAN-P_{Tet}-dCas9 and the appropriate pAN-$P_{A(X)}$-RFP plasmids to implement complete systems.

Table 3-1. Sequences and fold-repression values for template-targeting sgRNAs

Name	Region that binds -35 and reverse PAM	Region that binds the promoter-specific operator	Fold-repression
sgRNA-A1T	UUUACA**CC**	UAGCUCAGUCCU	280
sgRNA-A2T	UUUACA**CC**	AACGGGUCACAC	100
sgRNA-A3T	UUUACA**CC**	CGAAAUGGAGCA	220
sgRNA-A4T	UUUACA**CC**	UCCACAACUAGC	190
sgRNA-A5T	UUUACA**CC**	AAAACACUCGGA	440

Table 3-2. Sequences and fold-repression values for non-template-targeting sgRNAs

Name	Region that binds -10 and forward PAM	Region that binds the promoter-specific operator	Fold-repression
sgRNA-A1NT	AUAAUA**CC**	UAGGACUGAGCU	94
sgRNA-A2NT	AUAAUA**CC**	CGUGUGACCCGU	250
sgRNA-A3NT	AUAAUA**CC**	AUGCUCCAUUUC	340
sgRNA-A4NT	AUAAUA**CC**	AGCUAGUUGUGG	56
sgRNA-A5NT	AUAAUA**CC**	CUCCGAGUGUUU	270

Table 3-3. Sequences of genetic parts used in this work

Part name	Type	DNA sequence
BBa_J23101	promoter	tttacagctagctcagtcctaggtattatgctagc
P_{Const}	promoter	gcggcgcgccatcgaatggcgcaaaacctttcgcggtatggcatgatagcgcccggaaga gagtcaattcagggtggtgaat
P_{BAD}	promoter[3]	actttttcatactcccgccattcagagaagaaaccaattgtccatattgcatcagacattg ccgtcactgcgtcttttactggctcttctcgctaaccaaaccggtaaccccgcttattaa aagcattctgtaacaaagcgggaccaaagccatgacaaaaacgcgtaacaaaagtgtcta taatcacggcagaaaagtccacattgattatttgcacggcgtcacactttgctatgccat agcatttttatccataagattagcggatcctacctgacgctttttatcgcaactctctac tgtttctccata
P_{PhlF}	promoter[2]	cgacgtacggtggaatctgattcgttaccaattgacatgatacgaaacgtaccgtatcgt taaggt
P_{Tet}	promoter[2]	tactccaccgttggctttttccctatcagtgatagagattgacatccctatcagtgata gagataatgagcac
P_{A1}	promoter	tttacacctagctcagtcctaggtattatgctagc
P_{A2}	promoter	tttacaccaacgggtcacacgggtattatgctagc
P_{A3}	promoter	tttacacccgaaatggagcatggtattatgctagc
P_{A4}	promoter	tttacacctccacaactagctggtattatgctagc
P_{A5}	promoter	tttacaccaaaacactcggagggtattatgctagc
P_{AN} spacer	spacer	tccgaatgacatgcgtctccgttttagagctagaaatagcaagttaaaataaggctagtc cgattgcaacttgaaaaagtggcaccgagtcggtgctttttt
sgRNA-A1T	sgRNA	tttacacctagctcagtcctgttttagagctagaaatagcaagttaaaataaggctagtc cgattgcaacttgaaaaagtggcaccgagtcggtgctttttt
sgRNA-A1NT	sgRNA	ataatacctaggactgagctgttttagagctagaaatagcaagttaaaataaggctagtc cgattgcaacttgaaaaagtggcaccgagtcggtgctttttt
sgRNA-A2T	sgRNA	tttacaccaacgggtcacacgttttagagctagaaatagcaagttaaaataaggctagtc cgattgcaacttgaaaaagtggcaccgagtcggtgctttttt
sgRNA-A2NT	sgRNA	ataatacccgtgtgacccgtgttttagagctagaaatagcaagttaaaataaggctagtc cgttatcaacttgaaaaagtggcaccgagtcggtgctttttt
sgRNA-A3T	sgRNA	Tttacacccgaaatggagcagttttagagctagaaatagcaagttaaaataaggctagtc cgattgcaacttgaaaaagtggcaccgagtcggtgctttttt
sgRNA-A3NT	sgRNA	ataataccatgctccatttcgttttagagctagaaatagcaagttaaaataaggctagtc cgattgcaacttgaaaaagtggcaccgagtcggtgctttttt
sgRNA-A4T	sgRNA	tttacacctccacaactagcgttttagagctagaaatagcaagttaaaataaggctagtc cgattgcaacttgaaaaagtggcaccgagtcggtgctttttt
sgRNA-A4NT	sgRNA	ataataccagctagttgtgggttttagagctagaaatagcaagttaaaataaggctagtc cgttatcaacttgaaaaagtggcaccgagtcggtgctttttt
sgRNA-A5T	sgRNA	tttacaccaaaacactcggagttttagagctagaaatagcaagttaaaataaggctagtc cgattgcaacttgaaaaagtggcaccgagtcggtgctttttt
sgRNA-A5NT	sgRNA	ataataccctccgagtgtttgttttagagctagaaatagcaagttaaaataaggctagtc cgattgcaacttgaaaaagtggcaccgagtcggtgctttttt
sgRNA-VR	sgRNA	tgcgctcggtcgttcggctggttttagagctagaaatagcaagttaaaataaggctagtc cgttatcaacttgaaaaagtggcaccgagtcggtgctttttt
sgRNA-scramble	sgRNA	aacccctgattgtatccgcagttttagagctagaaatagcaagttaaaataaggctagtc cgttatcaacttgaaaaagtggcaccgagtcggtgctttttt
RiboJ	insulator[4]	agctgtcaccggatgtgctttccggtctgatgagtccgtgaggacgaaacagcctctaca aataattttgtttaa
LtsvJ	insulator[4]	agtacgtctgagcgtgataccgctcactgaagatggcccggtagggccgaaacgtacct ctacaaataattttgtttaa
dCas9	gene[5,6]	atggataagaaatactcaataggcttagctatcggcacaaatagcgtcggatgggcggtg atcactgatgaatataaggttccgtctaaaaagttcaaggttctgggaaatacagaccgc cacagtatcaaaaaaaatcttataggggctcttttatttgacagtggagagacagcggaa gcgactcgtctcaaacggacagctcgtagaaggtatacacgtcggaagaatcgtatttgt tatctacaggagatttttttcaaatgagatggcgaaagtagatgatagtttctttcatcga cttgaagagtctttttttggtggaagaagacaagaagcatgaacgtcatcctatttttgga aatatagtagatgaagttgcttatcatgagaaatatccaactatctatcatctgcgaaaa aaattggtagattctactgataaagcggatttgcgcttaatctatttggccttagcgcat atgattaagtttcgtggtcatttttttgattgagggagatttaaatcctgataatagtgat gtggacaaactatttatccagttggtacaaacctacaatcaattatttgaagaaaaccct attaacgcaagtggagtagatgctaaagcgattctttctgcacgattgagtaaatcaaga cgattagaaaatctcattgctcagctccccggtgagaagaaaaatggcttatttgggaat ctcattgctttgtcattgggtttgacccctaattttaaatcaaattttgatttggcagaa gatgctaaattacagctttcaaaagatacttacgatgatgatttagataatttattggcg

		caaattggagatcaatatgctgatttgtttttggcagctaagaatttatcagatgctatt
		ttactttcagatatcctaagagtaaatactgaaataactaaggctcccctatcagcttca
		atgattaaacgctacgatgaacatcatcaagacttgactcttttaaaagctttagttcga
		caacaacttccagaaaagtataaagaaatctttttgatcaatcaaaaaacggatatgca
		ggttatattgatgggggagctagccaagaagaattttataaatttatcaaaccaatttta
		gaaaaaatggatggtactgaggaattattggtgaaactaaatcgtgaagatttgctgcgc
		aagcaacggacctttgacaacggctctattccccatcaaattcacttgggtgagctgcat
		gctattttgagaagacaagaagacttttatccatttttaaaagacaatcgtgagaagatt
		gaaaaaatcttgacttttcgaattccttattatgttggtccattggcgcgtggcaatagt
		cgttttgcatggatgactcggaagtctgaagaaacaattaccccatggaattttgaagaa
		gttgtcgataaaggtgcttcagctcaatcatttattgaacgcatgacaaactttgataaa
		aatcttccaaatgaaaaagtactaccaaaacatagtttgctttatgagtattttacggtt
		tataacgaattgacaaaggtcaaatatgttactgaaggaatgcgaaaaccagcatttctt
		tcaggtgaacagaagaaagccattgttgatttactcttcaaaacaaatcgaaaagtaacc
		gttaagcaattaaaagaagattatttcaaaaaaatagaatgttttgatagtgttgaaatt
		tcaggagttgaagatagatttaatgcttcattaggtacctaccatgatttgctaaaaatt
		attaaagataaagatttttggataatgaagaaaatgaagatatcttagaggatattgtt
		ttaacattgaccttatttgaagatagggagatgattgaggaaagacttaaaacatatgct
		cacctctttgatgataaggtgatgaaacagcttaaacgtcgccgttatactggttgggga
		cgtttgtctcgaaaattgattaatggtattagggataagcaatctggcaaaacaatatta
		gatttttgaaatcagatggttttgccaatcgcaattttatgcagctgatccatgatgat
		agtttgacatttaaagaagacattcaaaaagcacaagtgtctggacaaggcgatagttta
		catgaacatattgcaaatttagctggtagccctgctattaaaaaaggtattttacagact
		gtaaaagttgttgatgaattggtcaaagtaatggggcggcataagccagaaaatatcgtt
		attgaaatggcacgtgaaaatcagacaactcaaaagggccagaaaaattcgcgagagcgt
		atgaaacgaatcgaagaaggtatcaaagaattaggaagtcagattcttaaagagcatcct
		gttgaaaatactcaattgcaaaatgaaaagctctatctctattatctccaaaatggaaga
		gacatgtatgtggaccaagaattagatattaatcgtttaagtgattatgatgtcgatgcc
		attgttccacaaagtttccttaaagacgattcaatagacaataaggtcttaacgcgttct
		gataaaaatcgtggtaaatcggataacgttccaagtgaagaagtagtcaaaaagatgaaa
		aactattggagacaacttctaaacgccaagttaatcactcaacgtaagtttgataattta
		acgaaagctgaacgtggaggtttgagtgaacttgataaagctggtttatcaaacgccaa
		ttggttgaaactcgccaaatcactaagcatgtggcacaaattttggatagtcgcatgaat
		actaaatacgatgaaaatgataaacttattcgagaggttaaagtgattaccttaaaatct
		aaattagtttctgacttccgaaaagatttccaattctataaagtacgtgagattaacaat
		taccatcatgcccatgatgcgtatctaaatgccgtcgttggaactgctttgattaagaaa
		tatccaaaacttgaatcggagtttgtctatggtgattataaagttttatgatgttcgtaaa
		atgattgctaagtctgagcaagaaataggcaaagcaaccgcaaaatatttctttttactct
		aatatcatgaacttcttcaaaacagaaattacacttgcaaatggagagattcgcaaacgc
		cctctaatcgaaactaatggggaaactggagaaattgtctgggataaagggcgagatttt
		gccacagtgcgcaaagtattgtccatgccccaagtcaatattgtcaagaaaacagaagta
		cagacaggcggattctccaaggagtcaattttaccaaaaagaaattcggacaagcttatt
		gctcgtaaaaaagactgggatccaaaaaaatatggtggttttgatagtccaacggtagct
		tattcagtcctagtggttgctaaggtggaaaaagggaaatcgaagaagttaaaatccgtt
		aaagagttactagggatcacaattatggaaagaagttcctttgaaaaaaatccgattgac
		tttttagaagctaaaggatataaggaagttaaaaaagacttaatcattaaactacctaaa
		tatagtctttttgagttagaaaacggtcgtaaacggatgctggctagtgccggagaatta
		caaaaaggaaatgagctggctctgccaagcaaatatgtgaatttttatatttagctagt
		cattatgaaaagttgaagggtagtccagaagataacgaacaaaaacaattgtttgtggag
		cagcataagcattatttagatgagattattgagcaaatcagtgaattttctaagcgtgtt
		attttagcagatgccaatttagataaagttcttagtgcatataacaaacatagagacaaa
		ccaatacgtgaacaagcagaaaatattattcatttatttacgttgacgaatcttggagct
		cccgctgcttttaaatattttgatacaacaattgatcgtaaacgatatacgtctacaaaa
		gaagttttagatgccactcttatccatcaatccatcactggtctttatgaaacacgcatt
		gatttgagtcagctaggaggtgactaa
		atggcttcctccgaagacgttatcaaagagttcatgcgtttcaaagttcgtatggaaggt
		tccgttaacggtcacgagttcgaaatcgaaggtgaaggtgaaggtcgtccgtacgaaggt
		acccagaccgctaaactgaaagttaccaaaggtggtccgctgccgttcgcttgggacatc
		ctgtccccgcagttccagtacggttccaaagcttacgttaaacacccggctgacatcccg
		gactacctgaaactgtccttcccggaaggtttcaaatgggaacgtgttatgaacttcgaa
mrfp1	gene[7]	gacggtggtgttgttaccgttacccaggactcctccctgcaagacggtgagttcatctac
		aaagttaaactgcgtggtaccaacttcccgtccgacggtccggttatgcagaaaaaaacc
		atgggttgggaagcttccaccgaacgtatgtacccggaagacggtgctctgaaaggtgaa
		atcaaaatgcgtctgaaactgaaagacggtggtcactacgacgctgaagttaaaaccacc
		tacatggctaaaaaaccggttcagctgccgggtgcttacaaaaccgacatcaaactggac
		atcacctcccacaacgaagactacaccatcgttgaacagtacgaacgtgctgaaggtcgt
		cactccaccggtgcttaataa
		atggtgagcaagggcgaggagctgttcaccggggtggtgcccatcctggtcgagctggac
		ggcgacgtaaacggccacaagttcagcgtgtccggcgagggcgagggcgatgccacctac
yfp	gene[8]	ggcaagctgaccctgaagttcatctgcaccaccggcaagctgcccgtgccctggcccacc
		ctcgtgaccaccttcggctacggcctgcaatgcttcgcccgctaccccgaccacatgaag
		ctgcacgacttcttcaagtccgccatgcccgaaggctacgtccaggagcgcaccatcttc

araC	gene[3]	ttcaaggacgacggcaactacaagacccgcgccgaggtgaagttcgagggcgacaccctg gtgaaccgcatcgagctgaagggcatcgacttcaaggaggacggcaacatcctggggcac aagctggagtacaactacaacagccacaacgtctatatcatggccgacaagcagaagaac ggcatcaaggtgaacttcaagatccgccacaacatcgaggacggcagcgtgcagctcgcc gaccactaccagcagaacaccccatcggcgacggccccgtgctgctgcccgacaaccac tacctgagctaccagtccgccctgagcaaagaccccaacgagaagcgcgatcacatggtc ctgctggagttcgtgaccgccgccgggatcactctcggcatggacgagctgtacaagtaa taa
lacI	gene[2]	atggctgaagcgcaaaatgatcccctgctgccgggatactcgtttaatgcccatctggtg gcgggtttaacgccgattgaggccaacggttatctcgatttttttatcgaccgaccgctg ggaatgaaaggttatattctcaatctcaccattcgcggtcaggggtggtgaaaaatcag ggacgagaatttgtttgccgaccgggtgatattttgctgttcccgccaggagagattcat cactacggtcgtcatccggaggctcgcgaatggtatcaccagtgggtttactttcgtccg cgcgcctactggcatgaatggcttaactggccgtcaatatttgccaatacggggttcttt cgcccggatgaagcgcaccagccgcatttcagcgacctgtttgggcaaatcattaacgcc gggcaaggggaagggcgctattcggagctgctggcgataaatctgcttgagcaattgtta ctgcggcgcatggaagcgattaacgagtcgctccatccaccgatggataatcgggtacgc gaggcttgtcagtacatcagcgatcacctggcagacagcaattttgatatcgccagcgtc gcacagcatgtttgcttgtcgccgtcgcgtctgtcacatcttttccgccagcagttaggg attagcgtcttaagctggcgcgaggaccaacgtatcagccaggcgaagctgcttttgagc accacccggatgcctatcgccaccgtcggtcgcaatgttggttttgacgatcaactctat ttctcgcgggtatttaaaaaatgcaccggggccagcccgagcgagttccgtgccggttgt gaagaaaaagtgaatgatgtagccgtcaagttgtcataa
tetR	gene[2]	atgaaaccagtaacgttatacgatgtcgcagagtatgccggtgtctcttatcagaccgtt tcccgcgtggtgaaccaggccagccacgtttctgcgaaaacgcgggaaaaagtggaagcg gcgatggcggagctgaattacattcccaaccgcgtggcacaacaactggcgggcaaacag tcgttgctgattggcgttgccacctccagtctggccctgcacgcgccgtcgcaaattgtc gcggcgattaaatctcgcgccgatcaactgggtgccagcgtggtggtgtcgatggtagaa cgaagcggcgtcgaagcctgtaaagcggcggtgcacaatcttctcgcgcaacgcgtcagt gggctgatcattaactatccgctggatgaccaggatgccattgctgtggaagctgcctgc actaatgttccggcgttatttcttgatgtctctgaccagacacccatcaacagtattatt ttctcccatgaggacggtacgcgactgggcgtggagcatctggtcgcattgggtcaccag caaatcgcgctgttagcgggcccattaagttctgtctcggcgcgtctgcgtctggctggc tggcataaatatctcactcgcaatcaaattcagccgatagcggaacgggaaggcgactgg agtgccatgtccggttttcaacaaaccatgcaaatgctgaatgagggcatcgttcccact gcgatgctggttgccaacgatcagatggcgctgggcgcaatgcgcgccattaccgagtcc gggctgcgcgttggtgcggatatctcggtagtgggatacgacgataccgaagatagctca tgttatatcccgccgttaaccaccatcaaacaggattttcgcctgctggggcaaaccagc gtggaccgcttgctgcaactctctcagggccaggcggtgaagggcaatcagctgttgcca gtctcactggtgaaaagaaaaaccaccctggcgcccaatacgcaaaccgcctctccccgc gcgttggccgattcattaatgcagctggcacgacaggtttcccgactggaaagcgggcag tgataa
phlF	gene[2]	atgtccagattagataaaagtaaagtgattaacagcgcattagagctgcttaatgaggtc ggaatcgaaggtttaacaacccgtaaactcgcccagaagctaggtgtagagcagcctaca ttgtattggcatgtaaaaaatgcgggctttgctcgacgccttagccattgagatgtta gataggcaccatactcacttttgccctttagaaggggaaagctggcaagatttttacgt aataacgctaaaagttttagatgtgctttactaagtcatcgcgatggagcaaaagtacat ttaggtacacggcctacagaaaaacagtatgaaactctcgaaaatcaattagccttttta tgccaacaaggttttcactagagaatgcattatatgcactcagcgctgtgggcatttt actttaggttgcgtattggaagatcaagagcatcaagtcgctaaagaagaaagggaaaca cctactactgatagtatgccgccattattacgacaagctatcgaattatttgatcaccaa ggtgcagagccagcctcttcttattcggccttgaattgatcatatgcggattagaaaaacaa cttaaatgtgaaagtgggtcctaa
TrrnB	terminator[5]	atggcacgtaccccgagccgtagcagcattggtagcctgcgtagtccgcatacccataaa gcaattctgaccagcaccattgaaatcctgaaagaatgtggttatagcggtctgagcatt gaaagcgttgcacgtcgtgccggtgcaagcaaaccgaccatttatcgttggtggaccaat aaagcagcactgattgccgaagtgtatgaaaatgaaagcgaacaggtgcgtaaatttccg gatctgggtagctttaaagccgatctggatttttctgctgcgtaatctgtggaaagtttgg cgtgaaaccatttgtggtgaagcatttcgttgtgttattgcagaagcacagctggaccct gcaaccctgacccagctgaaagatcagtttatggaacgtcgtcgtgagatgccgaaaaaa ctggttgaaaatgccattagcaatggtgaactgccgaaagataccaatcgtgaactgctg ctggatatgatttttggttttttgttggtatcgcctgctgaccgaacagctgaccgttgaa caggatattgaagaatttaccttcctgctgattaatggtgtttgtccgggtacacagcgt taa
		gaagcttgggcccgaacaaaaactcatctcagaagaggatctgaatagcgccgtcgacca tcatcatcatcatcattgagtttaaacggtctccagcttggctgttttggcggatgagag aagattttcagcctgatacagattaaatcagaacgcagaagcggtctgataaaacagaat ttgcctggcggcagtagcgcggtggtcccacctgaccccatgccgaactcagaagtgaaa cgccgtagcgccgatggtagtgtggggtctccccatgcgagagtagggaactgccaggca tcaaataaaacgaaaggctcagtcgaaagactgggcctttcgttttatctgttgtttgtc ggtgaact

BBa_B0015	terminator	ccaggcatcaaataaaacgaaaggctcagtcgaaagactgggcctttcgttttatctgtt gtttgtcggtgaacgctctctactagagtcacactggctcaccttcgggtgggcctttct gcgtttata
ECK120029600	terminator 9	ttgagaagagaaaagaaaaccgccgatcctgtccaccgcattactgcaaggtagtggaca agaccggcggtcttaagttttttggctgaa
ECK120033737	terminator 9	ggaaacacagaaaaaagcccgcacctgacagtgcgggctttttttttcgaccaaagg
L3S3P21	terminator 9	ccaattattgaaggcctccctaacgggggcctttttttgtttctggtctccc
L3S2P55	terminator 9	ctcggtaccaaagacgaacaataagacgctgaaaagcgtctttttttcgttttggtcc
L3S2P21	terminator 9	ctcggtaccaaattccagaaaagaggcctcccgaaaggggggcctttttttcgttttggtc c
L3S2P11	terminator 9	ctcggtaccaaattccagaaaagagacgctttcgagcgtctttttttcgttttggtcc

Chapter 4

4 Genetic circuit design automation

4.1 Background

Electronic design automation (EDA) software tools aid engineers in the design and analysis of semiconductor-based electronics (Hasty *et al*, 2002). Prior to EDA, integrated circuit design was a manual process performed by hand. This was accelerated by the development of hardware description languages (*e.g.*, Verilog) that enabled a user to design an electronic system through textual commands that are transformed to a circuit patterned on silicon. We applied this approach to genetic circuits, so that a Verilog design is transformed to a linear DNA sequence that can be constructed and run in living cells. The design environment, referred to as Cello (Cellular Logic), implements algorithms that derive the detailed physical design from the textual specification (Fig. 4-1). Cello requires genetic logic gates that are sufficiently modular and reliable such that their interconnection can be automated.

Moving computing into cells enables programmable control over biological functions (Hasty *et al*, 2002; Sprinzak & Elowitz, 2005; Drubin *et al*, 2007; Endy, 2011; Gibson *et al*, 2010). This is crucial for fully realizing the potential of engineering biology, where applications require that different sets of genes be active under different conditions (Weber & Fussenegger, 2012; Ruder *et al*, 2011; Boyle & Silver, 2012; Holtz

& Keasling, 2010). Cells are naturally able to respond to their environment, make decisions, construct intricate structures, and coordinate to distribute tasks. These functions are controlled by a regulatory network of interacting proteins, RNA, and DNA. Patterns of such interactions generate computational operations analogous to those used in electronic circuits (McAdams & Arkin, 1998; McAdams & Shapiro, 1995; Ptashne, 1986a; Alon, 2007; Buchler *et al*, 2003), and regulators can be combined to build synthetic genetic circuits (Elowitz & Leibler, 2000b; Stricker *et al*, 2008; Basu *et al*, 2005; Lim, 2010). This approach has led to digital logic gates (Moon *et al*, 2012b; Tamsir *et al*, 2011a; Stanton *et al*, 2014; Anderson *et al*, 2007; Ausländer *et al*, 2012; Teo & Chang, 2014), memory devices (Siuti *et al*, 2013; Bonnet *et al*, 2013, 2012; Yang *et al*, 2014; Kramer & Fussenegger, 2005), analog computation (Daniel *et al*, 2013c) and dynamic circuits (*e.g.*, timers and oscillators) (Elowitz & Leibler, 2000b; Stricker *et al*, 2008; Danino *et al*, 2010; Basu *et al*, 2004; Tigges *et al*, 2009). These have begun to be integrated into biotechnological applications (Lo *et al*, 2013; Ellis *et al*, 2009), for example, to implement feedback control in a metabolic pathway (Zhang *et al*, 2012). However, the construction of simple circuits consisting of only a few regulators remains a time consuming task and this has limited their widespread implementation.

Genetic circuit design is challenging for several reasons (Kwok, 2010; Purnick & Weiss, 2009a). First, circuits require precise balancing of regulator expression (Yokobayashi *et al*, 2002b; Anderson *et al*, 2007). Second, many parts are combined to build a circuit and their function can vary depending on genetic context, strain, and growth conditions (Kosuri *et al*, 2013; Goodman *et al*, 2013; Lou *et al*, 2012a; Mutalik *et al*, 2013a; Moser *et al*, 2012a; Yordanov *et al*, 2014; Cardinale *et al*, 2013). Third, circuits are defined by many states (their response to different inputs or how they change over time) and this can be cumbersome to characterize (Rosenfeld *et al*, 2005; Elowitz & Leibler, 2000b; Gardner *et al*, 2000a; Balagaddé *et al*, 2005). Finally, many

regulators are toxic when overexpressed and even mild effects can combine to drive negative selection against the circuit (Arkin & Fletcher, 2006). Balancing these issues is difficult to do by hand. Thus, computational tools have been developed for the study of natural networks and to aid circuit design by predicting how parts or devices will perform when connected (Chandran *et al*, 2009; Myers *et al*, 2009; Beal *et al*, 2011; Czar *et al*, 2009b; Marchisio & Stelling, 2011; Sauro *et al*, 2003; Rudge *et al*, 2012).

We developed Cello to accelerate circuit design, allow increased complexity of circuits, and enable non-experts to incorporate synthetic gene regulation into genetic engineering projects (Fig. 4-1). The focus is on the design of a circuit that performs a desired computational operation, which connects to cell-based sensors and cellular functions (actuators). A user provides three specifications to Cello. The first are the DNA sequences for the sensors: the sequences of their output promoters and data for their ON/OFF signal strengths in standardized units (see below) (Kelly *et al*, 2009b). The second is the "user constraints file" (UCF), which contains the functional details of the gate library, the layout of the genetic system, the organism and strain, and the operating conditions for which the circuit design is valid. The third is Verilog code that captures the desired computational operation. Cello uses this information to automatically design a DNA sequence encoding the desired genetic circuit by connecting a set of simpler gates that implement Boolean logic to the sensors and each other. The output of the circuit can be connected to cellular processes by directing the output promoter to control a cellular function (e.g., a metabolic pathway), either directly or through an intermediate (*e.g.*, a phage RNA polymerase) (Segall-Shapiro *et al*, 2014; Wang *et al*, 2012). The sensors, circuit, and actuator are inserted into specific genetic locations and transformed into a strain, both of which are defined in the UCF (Fig. 4-1b).

Cello builds circuits by connecting transcriptional gates, whose common signal carrier is RNA polymerase (RNAP) flux on DNA (Canton *et al*, 2008a). This conversion allows gates to be layered by having the output promoter from one gate serve as the input to the next. This modularizes the design, so that a circuit is defined by a pattern of promoters in front of regulators on a linear DNA strand (Fig. 4-2a). Within this paradigm, the regulators performing the gate biochemistry could be transcription factors (Weiss, 2001; Stanton *et al*, 2014), RNA based regulation (Green *et al*, 2014; Mutalik *et al*, 2012b; Chappell *et al*, 2015), protein-protein interactions (Moon *et al*, 2012b; Chen & Arkin, 2012b), CRISPR-Cas based regulation (Qi *et al*, 2013; Bikard *et al*, 2013; Nielsen & Voigt, 2014; Esvelt *et al*, 2013), or recombinases (Siuti *et al*, 2013; Bonnet *et al*, 2013; Ham *et al*, 2008). In this manuscript, we develop a set of insulated NOT and NOR gates based on prokaryotic repressors (Stanton *et al*, 2014). These repressor-based gates were characterized in isolation as NOT gates. To facilitate the connection of gates and sensors, we adopt the BBa_J23101 constitutive promoter as a standard (Kelly *et al*, 2009b). The output of an insulated version of this promoter (The standard differs from the Kelly standard and contains: an insulating upstream terminator, a different spacer upstream of the promoter (as opposed to a BioBricks prefix), RiboJ, RBS B0064, a different terminator, and three silent mutations to yfp.) is defined as 1 RPU and, working with National Institute of Standards and Technology (NIST) collaborators, this was measured to correspond to 24.7 $\pm$ 5.7 mRNAs per cell, which is approximately 0.02 RNAP/s-promoter (Fig. 4-38). These data were used by Cello to automatically generate a large set of circuits. The sequences were built as specified by the software output with no additional tuning, which facilitates the iterative improvement of the quality of the gates and design rules.

4.2 Cello design environment

Verilog is a commonly used hardware description language for electronic system design (Thomas & Moorby, 2002). It is hardware-independent, meaning that a circuit can be described by abstract textual commands and then transformed to different physical implementations (*i.e.,* chip types). Verilog is often accompanied by a simulation package that aids the evaluation of a design *in silico* before building the system. Verilog code has a hierarchical organization centered on modules that communicate through wires to propagate signals. In our implementation, circuit function can be defined by *case*, *assign*, or *structural* statements within modules (Fig. 4-28). Initially, our focus with Cello is on the creation of asynchronous combinational logic without feedback. This is useful in the design of genetic circuits that can process multiple environmental sensors in order to choose amongst different cellular functions. However, Verilog provides the framework to extend the designs to include more complex circuits, including those with specified timing and signal strengths as well as analog (Verilog-AMS) functions.

The philosophy behind Cello is to generate circuits for highly specified physical systems and operating conditions. This is defined by the User Constraint File (UCF), which specifies:

(1) The gate technology, including DNA sequences and functional data,

(2) Defined physical locations for the circuit (*e.g.*, plasmid or genomic locus),

(3) The organism, strain, and genotype,

(4) Operating conditions where the circuit design is valid,

(5) Architectural rules to constrain the part arrangement,

(6) Preferred logic motifs to be incorporated during logic synthesis.

The UCF follows the JSON (JavaScript Object Notation) standard (<douglas@crockford.com>), which is both human- and machine-readable and is

convertible with SBOL (Synthetic Biology Open Language) (Galdzicki *et al*, 2014). We developed the Eco1C1G1T1 UCF for *E. coli* (NEB 10-beta) and gate technology based on a set of 12 prokaryotic repressors (Stanton *et al*, 2014). The development of additional UCFs would enable a circuit design to be transferred to other organisms, conditions, or gate technologies.

When a user selects a UCF and synthesizes a circuit from Verilog code, the corresponding DNA sequence is designed in three steps (Fig. 4-1). First, the textual commands are converted to a circuit diagram. Algorithms parse the Verilog code and derive a truth table (Fig. 4-28), which is converted to an initial circuit diagram by the logic synthesis program ABC (Brayton & Mishchenko, 2010) and subsequently modified to only contain logic operations for gates available in the UCF (Fig. 4-30). The second step is to assign specific regulators to each gate in the diagram. Functionally connecting gates requires that the outputs from the first gate span the input threshold of the second gate (Fig. 4-2b). Because gates based on different regulators have different response functions, not all pairs can be functionally connected (Fig. 4-2c). Identifying the optimal assignment is an NP-complete problem (Fig. 4-2d) (Roehner & Myers, 2014; Yaman *et al*, 2012; Rodrigo & Jaramillo, 2013; Huynh & Tagkopoulos, 2014). We implemented a Monte Carlo simulated annealing algorithm to rapidly identify an assignment that produces the desired response (Fig. 4-2e and 4-33). The third step is to create the linear DNA sequence based on the circuit diagram and gate assignment. The assignment is converted to a set of parts and constraints between the parts (written with the Eugene language (Oberortner *et al*, 2014)). The UCF can also include additional constraints on the genetic architecture, for example, to forbid a particular combination of parts. A combinatorial design algorithm (Smanski *et al*, 2014), is used to build a genetic construct that conforms to the constraints (Fig. 4-34). This allows a user to design multiple constructs containing the same circuit

function and genetic constraints, while varying unconstrained design elements to build a library that can be screened.

Cello then simulates the performance of the genetic circuit. When flow cytometry data is provided in the UCF for the gates, this provides the cell-to-cell variation in the response for a population of cells. We developed a computational approach to quantify how population variability propagates from the sensors, through the gates, to the output promoters (Fig. 4-35). Cello applies a simple algorithm to determine how signals propagate from the sensors through the gates to the output promoters. This generates predicted cytometry distribution for all combinations of input states, which can be directly compared to experiments. Finally, for each gate, the load on the cell for carrying the gates is estimated based on their impact on growth (% reduction of OD_{600}) as a function of the activity of the input promoter (Methods). For any combination of inputs, if the predicted growth reduction exceeds a threshold, this information can guide multi-objective circuit optimization or be provided as a warning to the user (Fig. 4-32).

4.2.1 Specification: Verilog hardware description language

The Cello software provides a design automation environment whose input is a high-level specification from a hardware description language (Verilog). The first step is to parse the Verilog code to compute the truth table. The truth table is the starting point for logic synthesis, which generates the circuit diagram.

A subset of Verilog is synthesizable, meaning the program can be directly mapped to a physical implementation in hardware. Synthesizable Verilog is transformed to a netlist (a list of connected primitive gates that can be mapped to a hardware technology), which is functionally equivalent to the Verilog code. The subset of Verilog used in Cello is described in this section.

Verilog module. Verilog is written using modules, where each module has a name, and the line defining the module name also requires input definitions and output definitions. The following box defines a module named "example" with output "x", and inputs "a" and "b". Keywords are shown in blue.

```
module example(output x, input a, b);
endmodule
```

Assign statement. Within a Verilog module, Cello accepts and parses assign statements, case statements, and structural statements. An assignment provides a concise way to specify a combinational logic function. Assign statements use an = operator to set the value of a wire on the left-hand side based on the wire values and logic operators on the right-hand side.

```
module example(output x, input a, b);
  assign x = a & b;
endmodule
endmodule
```

Additional Verilog operators that can be used in assign statements are:

a & b	a AND b
a | b	a OR b
~a	NOT a

The following statement uses multiple operators.

```
module example(output x, input a, b, c);
  assign x = a & b | ~c;
endmodule
endmodule
```

The order of operations proceeds from left to right. Parenthesis can be used to specify a different order of operations to implement a different function.

```
module example(output x, input a, b, c);
  assign x = a & (b | ~c);
endmodule
endmodule
```

More complex assign statements can use internal wires to carry values within the module. To use internal wires, the names must be defined, and they must be assigned (appearing on the left-hand side of the equation) before they can be used as an operand on the right-hand side. The function above can also be implemented using internal wires.

```
module example(output x, input a, b, c);
  wire w1, w2;
  assign w1 = ~c;
  assign w2 = b | w1;
  assign x = w1 & w2;
endmodule
```

Case statement. A case statement provides a way to specify a truth table in Verilog. Since all combinational logic functions can be represented as a truth table, the case statement can be used to specify any combinational logic function as input to Cello. A case statement is placed within an "always" block. An always block contains a

"sensitive list", meaning the always block executes the code within the begin/end keywords whenever a value changes for a member of the sensitive list. The sensitive list below contains in1 and in2.

```
module example(output out, input in1, in2);
always@(in1, in2)
    begin
    end
endmodule
```

The case statement is placed within the begin/end lines within the always block. The line case({in1,in2}) indicates that the argument of the case statement is {in1,in2}. In Verilog, the brackets indicate concatenation, meaning the argument for the case statement is one value that is the concatenation of in1 and in2. If in1 is 0 and in2 is 1, the argument would be 01.

```
module example(output out, input in1, in2);
always@(in1, in2)
    begin
      case({in1,in2})
      endcase
    end
endmodule
```

The actual cases within the case statement are specified using a bit-wise numbering system: 2'b01: {out} = 1'b0. This individual case executes when the argument is a 2-bit number in binary notation (2'b) equal to 01. When this case executes, the value 0 for a 1-bit number in binary notation (1'b) is assigned to the wire named "out". By

specifying all combinations of input values as individual cases, a complete truth table can be specified. The following example specifies the truth table for a 2-input AND operation.

```
module example(output out, input in1, in2);
always@(in1, in2)
    begin
      case({in1,in2})
        2'b00: {out} = 1'b0;
        2'b01: {out} = 1'b0;
        2'b10: {out} = 1'b0;
        2'b11: {out} = 1'b1;

      endcase
    end
endmodule
```

More complex case statements can be used to specify n-input m-output truth tables, such as the multiple output truth table for the priority circuit (Figure 4-4a).

```
module example (output x, y, z, input a, b, c);
always@(a, b, c)
    begin
      case({a,b,c})
        3'b000: {x,y,z} = 3'b000;
        3'b001: {x,y,z} = 3'b001;
        3'b010: {x,y,z} = 3'b010;
        3'b011: {x,y,z} = 3'b010;
        3'b100: {x,y,z} = 3'b100;
        3'b101: {x,y,z} = 3'b100;
        3'b110: {x,y,z} = 3'b100;
        3'b111: {x,y,z} = 3'b100;

      endcase
    end
endmodule
```

The names of multiple output wires are concatenated within brackets, so the concatenated value of xyz equals 000 in the first case, equals 001 in the second case, and so on. Due to concatenation within brackets, the order of names matters in {a,b,c} and {x,y,z}. However, the order of names in the sensitive list does not matter.

always@(a, b, c) is the same as always@(c, b, a)

{x, y, z} is different than {z, y, x}

case({a, b, c}) is different than case({c, b, a})

Structural statement. Assign statements and case statements are forms of "behavioral" Verilog, meaning that the function is specified without considering a gate-level schematic. Structural Verilog can be used to directly specify the desired circuit topology using the same form as a netlist, which specifies a gate type, the output wire name, followed by the input wire names.

```
nor (x, a, b);
```

The above example specifies the function x = a nor b. Note that gate types are specified in lowercase in Verilog. Each gate can only have a single output, but can have multiple inputs. Allowed gate types include: not, or, nor, and, nand, xor, xnor, buf. For example, the following specifies the function x = a and b and c.

```
and (x, a, b, c);
```

To use structural elements within a Verilog module, the above lines just need to be written within a Verilog module:

```
module example (output x, input a, b, c);
   and (x, a, b, c);
endmodule
```

Internal wires can be used to build up more complex structural statements. The next example also implements 3-input AND logic, but uses a combination of four NOT gates and two NOR gates:

```
module example (output x, input a, b, c);
  wire w1, w2, w3, w4, w5;
  not (w1, c);
  not (w5, b);
  not (w4, a);
  nor (w3, w4, w5);
  not (w2, w3);
  nor (x, w1, w2);

endmodule
```

Even though structural Verilog can be used to specify a wiring diagram, logic synthesis is used to convert certain primitive gate types might not be available in the genetic gates library and to minimize number of gates in the circuit, if possible.

Combining Verilog statements. Explanations and examples of Verilog case statements, assign statements, and structural statements provided above were limited to one type of statement per module. However, these forms can also be combined in a module to build more complex programs. An example is provided below that combines the following commands:

Define a module name, input wire names, and output wire names:

module example(output out, input a, b, c);

Initialize the internal wire names that will be required to carry values within the module:

wire w1, w2, w3, w4;

Assign: Let w1 carry the value of the logical operation a AND c:

assign w1 = a & c;

Assign: Let w2 carry the value of the logical operation (NOT a) AND (NOT c):

assign w2 = ~a & ~c;

Structural: Define a NOR gate with output wire w3 and input wires w1 and w2:

nor (w3, w1, w2);

Structural: Define a NOT gate with output wire w4 and input wire w3:

not (w4, w3);

Case: Use a case statement to define a truth table for a 2-input AND function with inputs w4 and b, and output out. Members of the sensitive list are w4 and b, so the begin/end block will execute when w4 or b changes value. The argument for the case statement is the concatenated value of w4 and b. Only set the value of wire out to 1 when the concatenated value of w4 and b equals 11.

End the Verilog module:

endmodule

```
module example(output out, input a, b, c);
wire w1, w2, w3, w4;
assign w1 = a & c;
assign w2 = ~a & ~c;
nor (w3, w1, w2);
not (w4, w3);
always@(w4, b)
    begin
      case({w4,b})
        2'b00: {out} = 1'b0;
        2'b01: {out} = 1'b0;
        2'b10: {out} = 1'b0;
        2'b11: {out} = 1'b1;
      endcase
    end

endmodule
```

4.2.2 Parsing Verilog to generate a truth table

Section V.A explained the syntax for writing Verilog code. All combinational logic functions can be expressed in the form of a truth table, which is the entry point to logic synthesis. In this section, we describe how the Verilog program is parsed to a naïve netlist (list of connected gates), and how the naïve netlist is used to generate a truth table.

The first Verilog line is the module definition, which is parsed to obtain the input names and output name(s). From there, individual assign, structural, and case statements are parsed from the Verilog file. Each individual statement is converted to a logic node that can contain a single gate, multiple gates, or a truth table.

Assign. A line starting with the assign keyword indicates an assign statement, which is parsed to a tree data structure in which input wire names are the leaf nodes, the output wire name is the root node, logic operators (~, |, &) are the internal nodes, and parentheses inform the branching. This tree is functionally equivalent to a circuit diagram, which is used as a logic node with one or more logic gates.

Structural. A line starting with the lowercase name of a gate type (not, nor, or, and, nand, xor, xnor) indicates a structural statement, which is parsed to a single-gate logic node of that type, where the first argument indicates the node's output wire name, and all subsequent arguments indicate input wire names.

Case. An always block containing a case keyword is parsed to a truth table, where the wire name within curly brackets (for example, {out}) is the output wire name of the node, and the case argument (for example, {w4,B}) indicates the input wire names. Gate types are not used in the logic node parsed from a case statement; instead, the truth table is used to relate output values to the input values of the node.

Connecting all nodes according to the input/output wire names results in a graph that can be used to propagate logic through each node to calculate the truth table specified by the input Verilog (Figure 4-28).

Nested Verilog modules. The above example used different types of Verilog statements in the same module. However, Verilog modules can also be nested to form more complex programs. In the module hierarchy, the referencing module is called the parent module, and the referenced modules are called child modules (Figure 4-29). This nesting implements the reuse of previously written modules, which is helpful when scaling up to larger logic programs.

4.2.3 Logic synthesis

The previous section describes how the Verilog code is parsed to create a truth table. This section focuses on the next step, which is to convert the truth table to a circuit diagram. This is a process known as logic synthesis and our approach relies on algorithms that are typically applied to electronic circuits, with additional steps to incorporate constraints that arise from working with a limited set of genetic gates.

Logic synthesis is performed in several steps (Figure 4-30). First, the truth table is converted to a NOR-Inverter Graph (NIG). Second, logic motifs can be swapped for equivalent subcircuits to reduce circuit size. Logic motifs can be stored and retrieved from the UCF to biasing the circuit toward particular motifs that are desirable given the biochemistries of the gates in the library.

To convert a truth table to a NOR-Inverter Graph (NIG), an intermediate step uses the logic synthesis tool ABC (Brayton & Mishchenko, 2010) to generate an AND-Inverter Graph (AIG) built exclusively from 2-input AND and NOT gates. ABC minimizes the number of gates (nodes) and layers (longest path) in the AIG. The AIG is converted to an NIG containing 2-input NOR and NOT gates. This conversion is

done by replacing (A AND B) with the equivalent (NOT A) NOR (NOT B) according to DeMorgan's rule.

As an alternative to ABC, we also developed a path to the circuit diagram using Espresso (Brayton *et al*, 1984), another commonly used tool for logic synthesis. This approach differs in that it first converts a truth table to a minimized Product of Sums (POS), which we then convert to an NIG. Both the ABC and Espresso routes are implemented in Cello and the one that produces the circuit diagram with the minimum number of gates is selected. In approximately 95% of the cases, the number of gates after the ABC route is less than or equal to the number of gates after the Espresso route.

The user may have preferred logic motifs that they would like to have in the circuit diagram, if possible. These could represent optimized combinations of gates (both ABC and Espresso are not guaranteed to find the global minimum) or motifs that are particularly robust for a given biochemistry. For example, the UCF we developed has a list of small 3-input 1-output motifs generated from brute-force enumeration (Methods). Additionally, this is a simple mechanism to introduce non-NOR logic functions for which genetic gates may be available. The Eco1C1G1T1 UCF motif library contains: (1) a 2-input OUTPUT_OR motif to replace a NOR-NOT subcircuit at an output, (2) a 2-input 1-output optimal XNOR motif, and (3) small 3-input 1-output NOR/NOT motifs.

An attempt to incorporate the user-defined circuit architecture motifs into circuit diagrams occurs during the final step of logic synthesis. Starting with an initial NOR/NOT circuit diagram, subcircuits are replaced with a set of user-defined motifs, if possible. This is performed by the following steps. First, all possible subcircuits in the initial circuit diagram with ≤ 4 input wires and 1 output wire are enumerated. This is done by visiting each gate's output wire, then performing a breadth-first search

on the incoming wires and gates, proceeding until the circuit inputs are reached. During this search, unique subcircuits are added to a list. Second, the truth table for each subcircuit and each user-defined motif is evaluated. If a subcircuit and a motif have Boolean equivalence (also checking permuted input wire order), then the motif is substituted in place of the subcircuit. If multiple subcircuit/motif matches are found, the motif that reduces the number of circuit gates the most is used. Finally, each time a motif replacement is made, the replacement algorithm is performed again until no more replacements can be made.

Motifs in the library can use gate types other than NOR/NOT, such as AND, NAND, OR, XOR, or XNOR. To constrain the logic gates according to the number and types of gates available in the genetic gates library, a cost function is used during subcircuit substitution. The cost is the total number of gates in the circuit that exceed the gates available in the library. For example, if there are 6 NOR/NOT gates and 1 AND gate in the library, and the circuit has 7 NOR/NOT gates and 2 AND gates, the cost would evaluate to (7-6) + (2-1) = 2. If there are enough available gates in the library to cover the gates in the circuit, the cost is 0. A substitution is rejected if the cost increases, and is accepted if the cost decreases or does not change. This cost evaluation guides logic synthesis to produce a circuit that can be covered by the gates library. However, after subcircuit substitution converges and no more substitutions are possible, if the cost is still greater than 0, the circuit is reported as "not synthesizable".

4.2.4 Repressor assignment

The previous section describes how the circuit diagram is generated. The next step is to assign genetic regulators to the gates in the diagram. Each gate is based on a unique biochemistry and thus generates a different response function. The assignment

problem is to identify the optimal way to select and connect these gates to generate the maximum overall dynamic range for the circuit. In this section, we first describe how we score a particular repressor assignment. Next, the search algorithm is described that optimizes the assignment.

One approach to the assignment problem would be to permute all possible combinations of gates and identify the one that generates the best circuit. This would guarantee the identification of the global optimum. However this method becomes intractable as circuit size and library size grow. The number of unique assignments (with a single RBS variant per gate) is given by $r!/(r\text{-}g)!$, where g is the number of gates in the circuit and r is the number of repressors in the library. With our library of repressors (including RBS variants), a 9-gate circuit has $\sim 10^{11}$ permutations. A search algorithm needs to be implemented to scale to larger circuits and libraries, but often comes with the tradeoff of introducing stochasticity into the search and can converge on local optima.

Calculating the circuit score. The circuit score S captures how closely the logic function generated by a repressor assignment matches the desired truth table for the circuit. Because the output of genetic circuits is not digital, the ON and OFF states have numerical values and a larger difference between these values (the dynamic range) is desirable. Calculating S requires two steps. First, the output is calculated for all combinations of input states. An example is shown in Figure 4-31, where there are two sensors and four input states. The activity of the sensors feeds into the gates and their response functions are used to calculate how the signal propagates through the circuit. Then, S is calculated by comparing the lowest output for a state that should be ON and the highest output for a state that should be OFF:

$$S = \frac{\min(ON)}{\max(OFF)}$$

Calculation of predicted circuit toxicity. For each gate, normalized cell growth is measured as a function of input promoter activity (Figure 4-15). For a circuit, certain input states can lead to the expression of multiple repressors and this can lead to toxicity. For each gate in a circuit, the input RPU is calculated, and the cell growth value is interpolated from the two nearest experimentally-measured normalized cell growth values from the UCF. The toxicity of the whole circuit for a particular input combination is calculated as the product of normalized cell growth for each of the individual gates. There is no theoretical basis for this; rather, it was chosen to strongly bias against circuits where any repressors are expressed beyond their empirical toxicity threshold. After the toxicities of all the input states are calculated, the toxicity of the circuit as a whole ("growth score") is taken as the worst input state.

As shown in Figure 4-32 for the Majority circuit, there is a trade-off between the circuits with the highest circuit score (S) and those that are at risk of reducing growth, creating a Pareto-optimal curve. The current algorithm applies a cutoff (0.75) with respect to the growth score and only allows circuit assignments that fall above the cutoff.

Simulated Annealing Assignment Algorithm. The goal of repressor assignment is to find the combination of gates that maximizes the circuit score, S. The repressor assignment problem has a large discrete search space for which we implemented a Monte Carlo simulated annealing algorithm (Aarts et al, 2005; Metropolis & Ulam, 1949) to identify an optimum assignment. The search initializes with gates from the library being randomly chosen and assigned to a gate in the circuit. Any gate can be assigned to any position in the circuit. Each iteration of the Monte Carlo algorithm swaps the assignments of two gates. This is done by randomly selecting one gate in the circuit, randomly selecting a second gate either in the circuit or in the gate library,

and then performing the swap. After the swap, the circuit score for the new assignment S' is calculated and the move is accepted with a probability based on the score change and the temperature factor T:

$$P = e^{-\left(\frac{S-S'}{T}\right)}$$

After calculating the probability, a random number R between 0 and 1 is generated: if R $<$ P, the swap is accepted, and if R $>$ P, the swap is rejected. If the swap improved S, then P $>$ 1, and the move is always accepted. After the first assignment is initialized, the probability of accepting a move decreases as the temperature anneals with exponential decay:

$$T_i = T_{max} \cdot e^{-Ci}$$

where i is the current iteration, T_{max} is the starting temperature, and C is a constant that determines the rate of cooling. After reaching the end of annealing, the run continues at $T = 0$ until 10,000 steps progress with no additional improvement. The simulated annealing results in Figure 4-33 show convergent solutions for the circuits ranging from 5 to 9 gates, where $T_{max} = 100$, and C is 5 x 10^{-5}.

Several modifications were made to the basic algorithm described above to allow for additional constraints that do not appear in the S calculation. Some gates have multiple RBS options, but a repressor cannot be used more than once in a circuit. To prevent illegal swaps that reuse a repressor, a list of gates that can be legally swapped is generated. Gates with the same repressor as the selected gate are allowed in the list, because this swap simply replaces the RBS for the gate. Gates with a different repressor are only allowed in the list if another gate from the circuit does not use the same repressor. To avoid repressor reuse, gate group names are also specified in the UCF (*gates* collection). The group name will typically be the repressor name, but different repressors can also be grouped if they exhibit cross-talk.

Additional constraints can be applied to reject assignments that would otherwise be accepted based on S. For example, we have implemented the rejection of assignments whose growth score is below a threshold (previous sub-section) or when two "roadblocking" promoters have to be connected as inputs to a gate.

4.2.5 Combinatorial design of circuit layouts

After a gate assignment has been found for a circuit diagram, a linear DNA sequence that contains the complete circuit is generated. This is done using combinatorial design (Smanski et al, 2014; Bhatia et al), which has been applied to build DNA sequences using a set of parts, constraints between parts, and organizational rules. The assignment algorithm leads to a list of parts in the circuit as well as constraints between parts (*e.g.*, due to roadblocking). The UCF can also contain additional organization rules, such that the repressors have to appear in a specified order and orientation. After the assignment algorithm, the parts and rules for a circuit are automatically used to build a Eugene file. From this Eugene file, combinatorial algorithms described previously(Smanski *et al*, 2014; Bhatia *et al*) are used to build the DNA sequence, which is the output of Cello. The Eugene file itself is also an output of Cello so that it can be run at a later time to generate additional constructs (https://cidar.bu.edu/EugeneLab/).

One of the advantages of using combinatorial design is that many constructs can be built that preserve the same underlying rules–and therefore produce the same circuit function–but unconstrained aspects of the design are allowed to vary. Before the user runs Cello, an option is available to specify the number of desired constructs. Building and testing a library of designs instead of a single design can help identify a functional variant. Additionally, identifying failed and successful designs provides a

data set for learning new organizational rules (Smanski et al, 2014). An example of this are the Majority circuits in Figure 4-5e, where multiple constructs are shown.

This section describes how gates and their component parts are organized in Eugene as well as the impact of adding organizational constraints to the UCF. A hierarchical design is used to describe circuits in Eugene (Level 1: Parts, Level 2: Gates, and Level 3: Circuit).

In Level 1, part types are defined, and individual parts with those types are defined. Parts in Eugene require a type and a name, while other attributes such as DNA sequence can be added optionally. Below are examples of part definitions for a promoter, ribozyme, RBS, CDS and terminator that make up a gate.

Level 1: Part type definitions

```
PartType Promoter;
PartType Ribozyme;
PartType RBS;
PartType CDS;
PartType Terminator;
```

Level 1: Part definitions

```
Promoter pTac;
Ribozyme RiboJ53;
RBS P3;
CDS PhlF;
Terminator ECK120033737;
```

In Level 2, we assemble parts into gate devices (a device is defined as a collection of parts). The gate device contains the ribozyme insulator, RBS (sometimes multiple variants are allowed), repressor, and terminator. For a NOR gate, there can be two additional undefined promoters. For example, the device for the PhlF gate (with the P3 RBS) is as follows.

Level 2: Gate device

```
Device PhlF_device(
    Promoter, Promoter, RiboJ53, P3, PhlF, ECK120033737
);
```

We then define a set of rules that act on the P3_PhlF device. These rules define the promoters that drive PhlF according to the circuit diagram, and an enforced order of those promoters (e.g., to avoid roadblocking). The ALL_FORWARD rule just orients all parts in the forward direction (this gate will be allowed in the reverse direction in a later step). Note that rules on different lines must be joined with the AND keyword.

Level 2: Gate rules

```
Rule PhlF_rules
(ON PhlF_device:
    CONTAINS pBM3R1 AND
    CONTAINS pHlyIIR AND
    pBM3R1 BEFORE pHlyIIR AND
    ALL_FORWARD
);
```

Given the devices and rules for each gate, an enumeration of variants for each device is performed using the 'product' function. The PhlF device shown above only allows a single device variant.

Level 2: Design of gate variants

```
PhlF_devices    = product(PhlF_device);
SrpR_devices    = product(SrpR_device);
BM3R1_devices   = product(BM3R1_device);
HlyIIR_devices = product(HlyIIR_device);
BetI_devices    = product(BetI_device);
AmtR_devices    = product(AmtR_device);
```

In Level 3, gate device variants will be combined into the circuit device. First, we must initialize the circuit device and each gate device, where each gate is named by the repressor to allow rules from the UCF to be applied according to that name.

Level 3: Initializing the circuit device, and its component devices.

```
Device circuit();

Device gate_PhlF();
Device gate_SrpR();
Device gate_BM3R1();
Device gate_HlyIIR();
Device gate_BetI();
Device gate_AmtR();
```

Rules are applied to the circuit device before enumerating circuit variants. The EXACTLY 1 counting rule ensures that each gate appears once and only once in the circuit. Additional rules can also be specified, for example requiring the PhlF gate to be in the first position and in the forward orientation, and requiring each gate to alternate orientation, as follows.

Level 3: Circuit device rules

```
Rule circuit_rules
(ON circuit:
    gate_PhlF    EXACTLY 1 AND
    gate_SrpR    EXACTLY 1 AND
    gate_BM3R1   EXACTLY 1 AND
    gate_HlyIIR  EXACTLY 1 AND
    gate_BetI    EXACTLY 1 AND
    gate_AmtR    EXACTLY 1 AND
    STARTSWITH gate_PhlF AND
    FORWARD gate_PhlF AND
    ALTERNATE_ORIENTATION
);
```

Now that we have specified the circuit rules, the combinatorial design step can be performed. Nested for-loops iterate through all gate device variants from Level 2 (there might only be a single variant for each gate, or there might be variants with different promoter orders). In the innermost loop, the current set of gate device

variants is used to build the circuit device in Level 3 in the 'permute' function. Each set of designs in the inner loop is appended to an array called 'allResults'.

Level 3: Design of circuit variants

```
Array allResults;

for(num i0=0; i0<sizeof(PhlF_devices);   i0=i0+1) {
for(num i1=0; i1<sizeof(BetI_devices);   i1=i1+1) {
for(num i2=0; i2<sizeof(SrpR_devices);   i2=i2+1) {
for(num i3=0; i3<sizeof(HlyIIR_devices); i3=i3+1) {
for(num i4=0; i4<sizeof(AmtR_devices);   i4=i4+1) {
for(num i5=0; i5<sizeof(QacR_devices);   i5=i5+1) {

gate_PhlF   = PhlF_devices[i0];
gate_BetI   = BetI_devices[i1];
gate_SrpR   = SrpR_devices[i2];
gate_HlyIIR = HlyIIR_devices[i3];
gate_AmtR   = AmtR_devices[i4];
gate_QacR   = QacR_devices[i5];

Device circuit(
   gate_PhlF,
   gate_BetI,
   gate_SrpR,
   gate_HlyIIR,
   gate_AmtR,
   gate_QacR
);

result = permute(circuit);

allResults = allResults + result;

}}}}}}
```

Next, we explain how Eugene rules specify the design space of allowed circuit layouts (Figure 4-34). At the gate device level (Level 2), the only degree of freedom is promoter order within each gate. NOT gates can only have 1 variant. NOR gates can have two variants, where either promoter order is allowed. This degree of freedom allows 2^N variants, where N is the number of 2-input gates. Enforcing roadblocking rules (STARTSWITH) constrains promoter order, but for tandem non-roadblocking promoters, either promoter order is still allowed (Figure 4-34, Gate devices).

In addition to promoter order, gate order/orientation is the other degree of freedom. At the circuit device level (Level 3), if the repressor order is constrained, and all gates

are in the forward orientation (as specified in Eco1C1G1T1), then the only degree of freedom is promoter order from Level 2. In the example circuit assignment, repressor order and all forward rules result in 4 solutions (Figure 4-34, Panel 1).

Additional variants can be generated by removing order/orientation rules. For example, removing the FORWARD gate_PhlF rule allows the reverse orientation of the PhlF gate and results in 8 solutions (Panel 2). Removing a second rule, gate_PhlF BEFORE gate_BetI, now allows the PhlF gate in any position and results in 40 solutions (Panel 3). Removing all remaining gate order rules (a BEFORE b) allows unconstrained shuffling of gate order and results in 960 solutions (Panel 4). Removing all remaining FORWARD rules allows all gate orders and orientations and results in 15,360 solutions (Panel 5).

The Eugene rules specified in the UCF (*eugene_rules* collection) include roadblocking rules in Level 2 gate devices, and repressor order and all forward rules in the Level 3 circuit device. As described above, these rules can be removed to unconstrain the layout design space. Furthermore, any additional rules from the Eugene language (Oberortner et al, 2014) can be added to the UCF for user-specified constraints to the design space. The number of desired variants can be specified in the Options tab of the Cello web application. After layout design using Eugene, the Cello output has three forms. The first output is a file containing an ordered list of part names and part orientations (+, -) for each variant. The second output is a file containing an ordered list of gate names and gate orientations for each variant. The third output is a separate plasmid file for each variant in which the circuit module is inserted into the specified genetic location (Section VII, *genetic_locations* collection).

4.2.6 Predictions of circuit performance

Qualitative predictions of the circuit output distributions can be computed for each input combination. This is performed as a final step in Cello, after the gate assignment search has converged. In order to perform this step, each gate in the circuit must have experimental cytometry distributions added to the UCF, with fluorescence values converted to RPU.

As a first step, the experimentally-measured gate output RPU histograms are normalized to have a total of 10,000 events in evenly log-spaced bins (one bin every $10^{0.024x}$ RPU). At least 8 experimentally-measured output RPU histograms at various input RPU levels are required (the gates in this work use 12 input levels). Histograms *Y(x)* are generated at intermediate input levels x by positioning their medians, $<Y(x)>$, on the gate's response function. Once the medians are in place, the counts f for each bin are interpolated from the counts (f_L and f_R) of the experimentally-measured histograms that lie to either side of x. The experimentally measured histograms have input values, x_L and x_R. The parameter m is the bin relative to the median, $y/<Y>$. These relationships are captured by the equations:

$$\langle Y(x)\rangle = y_{min} + \frac{(y_{max} - y_{min})K^n}{x^n + K^n}$$

$$f(m) = f_L(m) + (f_R(m) - f_L(m))\frac{x - x_L}{x_R - x_L}$$

In this way, histograms at intermediate inputs are generated with medians on the response function, and shapes interpolated from the nearest experimental histograms.

Once all the gate distribution response functions are computed, the qualitative predictions for output distributions can be computed. For a particular input combination, sensor values feed into the first layer of gate distribution response functions (dashed vertical lines, Figure 4-35). This input value takes a vertical "slice"

of the distribution response function to create the output histogram. Next, those gate output histograms become input histograms for the second layer of gates.

Input histograms can be viewed as 10,000 individual input events, each of which produces its own output histogram—all of which are averaged to produce a histogram containing 10,000 events. At NOR and OR gates, input histograms are combined by first summing the histogram medians (or summing the histogram median and the sensor input value if a sensor promoter is an input), then shifting both histograms to be centered at that new median, and then averaging the counts in each bin to create a histogram with 10,000 events.

Sensor input signals are propagated through gate distribution response functions in the circuit until the output histograms are produced for each input row in the truth table (Figure 4-35c).

Input threshold analysis. Genetic gates output a continuous range of values as opposed to digital 0s and 1s. This is similar to electronic systems, where output voltages also take continuous values. For digital abstraction in electronic design, a maximum value for low-inputs and a minimum value for high-inputs specify the input ranges that produce outputs considered to be ON and OFF (Hauser, 1993). If an input signal falls between a gate's low/high thresholds, this can lead to an intermediate output or an incorrect ON/OFF output if the input is perturbed slightly. This section describes how we define and use input thresholds in Cello.

For two NOT gates connected in series, the low and high output levels of the first gate (*OL* and *OH*) must map onto either side of the low and high input thresholds of the second gate (*IL* and *IH*). The difference between *IL* and *OL* is the low margin (*ML*), and the difference between *OH* and *IH* is the high margin (*MH*). Both margins must be positive to satisfy the input threshold criteria. A negative margin indicates

an input that falls in the sensitive intermediate zone, the region between *IL* and *IH* (Figure 4-36a, diagonal hatching).

In electronic design, the low and high input thresholds are identical for all gates in a circuit. However, each genetic gate has unique thresholds due to their different response functions. Each gate's *IL* and *IH* thresholds are calculated from its response function using the input levels that cause the gate output to be 0.5x the maximum and 2x the minimum output, respectively (Figure 4-36a, black dots). *ML* and *MH* must be positive values for a gate connection to be valid, and all gate connections must be valid for the circuit as a whole to be valid. As an example, in the Majority circuit (Figure 4-5) the assignment had a good predicted circuit score, but the assignment did not satisfy all input margins. Specifically, the PhlF gate has a predicted input RPU that falls in the intermediate region (Figure 4-36b). Experimentally, the initial design for this circuit (Figure 4-20) had one high OFF state caused by the PhlF gate input falling in the sensitive intermediate region.

4.3 Characterization of sensors and gates for use with Cello

Different sensors and actuators can be connected to the circuits built by Cello. To make use of this, a user has to characterize the output promoter of their sensor(s) in the genetic context defined for the circuit (in the UCF). The sensor is characterized in two states, ON and OFF, under the conditions defined by the user. For example, it could be two different concentrations of a small molecule or the presence and absence of an environmental stimulus. This measurement has to be provided to Cello in standard units (RPUs). The DNA containing any regulators necessary for sensor function (referred to as the "sensor block") could either be uploaded to appear in the

circuit plasmid or separately inserted into a different context (*e.g.*, the genome). Sensor blocks are not necessarily required; for example, when the promoter depends only on native regulators. The connection of the output to a new actuator is more straightforward, where it simply requires knowing whether its dynamic range is sufficient to trigger a phenotypic response. A step-by-step guide for the characterization of new sensors such that they can be used with Cello is provided in this section.

Similarly, building new UCFs requires the characterization of new gates and/or gate libraries in different organisms or operating conditions. The procedure is similar to characterizing sensors and is also provided in this section. The details of the data organization for the gate library in the UCF are provided in the Appendix.

4.3.1 Measurement of RPU plasmid

Sensor and gate fluorescence characterization data must be converted into relative promoter units (RPUs) for incorporation into Cello. To convert characterization data to RPUs, an RPU standard plasmid must first be measured along with an autofluorescence control.

Transform the RPU plasmid to create an RPU standard strain and a non-YFP plasmid to create an autofluorescence control strain. Following work by Kelly et al. (Kelly *et al*, 2009), the promoter activities must be reported to Cello in standard units. The standard plasmid used for the Eco1C1G1T1 UCF is shown in Figure 4-37. New UCFs may define different standards. This is the same backbone as the circuit plasmid (Figure 4-1b) and the gate measurement plasmid (Figure 4-41), and contains the same YFP expression cassette. Note that while the standard constitutive promoter (BBa_J23101 (Part:BBa J23101 - parts.igem.org)) is the same, this plasmid is different from the Kelly standard (Kelly *et al*, 2009; Stanton *et al*, 2014), including an upstream

insulating terminator, an upstream promoter spacer, and a different RBS. Our RPU plasmid produces 4.2x the YFP signal as the Kelly standard. The plasmid should be transformed into the strain defined by the UCF, which for the Eco1C1G1T1 UCF is *E. coli* NEB 10-beta (New England Biolabs, MA, C3019).

Measure the fluorescence of the RPU standard strain and the autofluorescence control strain. For each set of inducer conditions to be applied to a circuit, gate, or sensor, YFP fluorescence measurements are collected for the RPU strain and the autofluorescence strain. Fluorescence can be measured using flow cytometry, a plate reader, or any instrument capable of measuring YFP fluorescence. The measurements should be made under media and growth conditions that are as close as possible to that defined in the UCF.

We performed additional characterization of the RPU standard to estimate the RNAP flux on the promoter J23101. The RPU standard plasmid was measured using smFISH (Raj *et al*, 2008) to obtain the rate of mRNA transcription from the P_{Tac} promoter. A background control and a measurement plasmid with an inducible promoter were also measured to generate a standard curve to determine the steady state number of *yfp* mRNAs per cell. We quantified the steady state number of *yfp* mRNA copies per cell at mid-exponential growth using single-molecule fluorescence *in situ* hybridization (smFISH) (Raj *et al*, 2008) following the method given in (Skinner *et al*, 2013), which we adapted for counting *yfp* mRNA in *E. coli* at single-transcript resolution. Briefly, we designed a set of 25 oligonucleotide probes, fluorescently labeled with TAMRA, each 20 bases in length, against the *yfp* transcript (Table 4-10) using Stellaris Probe Designer version 4.1. The mean *yfp* mRNA copy number for the RPU standard plasmid (pAN1717) strain was found to be 24.7 $\pm$ 5.7 molecules per cell (Figure 4-38). The half-life for *gfp* mRNA in *E. coli* has been determined to be approximately 2 minutes (Smolke *et al*, 2000), and the average plasmid copy number

for the p15A origin of replication has been determined to be approximately 15 per cell (Hiszczyńska-Sawicka & Kur, 1997). Therefore, the estimated mRNA production rate is 0.30 ± 0.07 mRNAs per second per cell or 0.02 ± 0.005 mRNAs per second per plasmid for the pAN1717 J23101 promoter. Stated errors are the standard deviation from replicate measurements and do not include any estimate of systematic bias of the measurement. For comparison, the mRNA production rate for the J23101 promoter was estimated to be approximately 0.03 mRNA per second per DNA copy in (Kelly *et al*, 2009).

4.3.2 Sensor characterization

Sensors convert signals (small molecules, light, etc.) into a transcriptional output. This section provides the steps required to characterize a sensor and report its output in standard units (RPUs). First, the output promoter needs to be cloned into the circuit plasmid defined by the UCF. Next, the sensor responses are characterized in the strain of interest and parallel measurements of the RPU standard's fluorescence and the strain's autofluorescence are made (previous section). These data are used to calculate sensor output RPU values, which are input into Cello along with the sequence of the output promoter and sequence of the sensor block.

Construct the plasmid to measure the sensor output promoter. The sensor is characterized in the same backbone as the circuit in Figure 4-1b (Figure 4-39, bottom). A version is provided that contains a P_{Lac}-lacZα module used for blue/white screening (Figure 4-39, top). The sensor output promoter is used to drive a YFP expression cassette, which matches the RPU standard. This cassette includes a ribozyme insulator (RiboJ) and the same RBS-YFP-terminator set as the RPU standard plasmid. The transcriptional fusion between the promoter and expression cassette should be scarless (note that a 4 bp scar is defined at the 5'-end of each ribozyme

insulator that can be used for cloning). To clone the entire cassette into the plasmid, BbsI Golden Gate sequences flanking the P_{Lac}-lacZα module simplify cloning, but do not have to be used.

*Transform the sensor plasmid from #1 into the strain defined by the UCF.*_The Eco1C1G1T1 UCF defines the strain as *E. coli* NEB 10-beta (New England Biolabs, MA, C3019).

Characterize the ON/OFF state of the sensors. For each set of conditions, fluorescence measurements are made for three strains containing different plasmids: the sensor, the RPU standard, and empty plasmid for autofluorescence. The measurements should be made under media and growth conditions that are as close as possible to that defined in the UCF. The following equation converts the median YFP fluorescence to RPU:

$$RPU = \frac{\langle YFP \rangle - \langle YFP \rangle_0}{\langle YFP \rangle_{RPU} - \langle YFP \rangle_0}$$

where $<YFP>$ is the median fluorescence of the cells containing the sensor, $<YFP>_{RPU}$ is the median fluorescence of the cells containing the standard plasmid, and $<YFP>_0$ is the median autofluorescence.

Example: Characterization of sensors (IPTG, aTc, arabinose). We demonstrate the characterization of the three sensors used in this manuscript. Three plasmids were constructed (Figure 4-40) to individually test the output promoters that respond to IPTG (P_{Tac}), aTc (P_{Tet}), and arabinose (P_{BAD}). The same sensor block was used containing the necessary regulators (LacI, TetR, and AraC*). We measured each of these sensors in response to "low" and "high" input signals: the absence or presence of 1 mM IPTG, 2 ng/mL aTc, and 5 mM L-arabinose respectively. The extraction of median fluorescence from the cytometry plots and conversion into RPUs are shown in Table 4-6.

4.3.3 Characterization of gates to be included in the UCF

Characterizing gates is similar to sensors, with three differences. First, both the input and output of the gate are promoter activities. This requires a separate characterization of the input promoter that is used to characterize the gate. Second, a full response function is required for the gate, as opposed to simpler ON/OFF values. Finally, in order to qualitatively predict population behavior, Cello requires RPU distributions and these are more complicated to normalize to RPUs. The protocol for gate measurement is below.

Clone the gate into the measurement plasmid. For the Eco1C1G1T1 UCF, the input promoter to the gate is P_{Tac} and a LacI expression cassette is provided in the backbone of the same plasmid used to characterize sensors and the circuits (Figure 4-1b). The output is the same YFP expression cassette used for the RPU standard plasmid. The gate—which consists of a ribozyme insulator, RBS, gene, terminator, and the output promoter—is cloned between P_{Tac} and the standard YFP expression cassette on the circuit backbone that encodes a constitutively expressed LacI (Figure 4-41). The resulting construct allows the gate to be induced with IPTG, and the output to be measured by quantifying YFP with cytometry.

Transform the gate plasmid from #1, the RPU standard plasmid, the autofluorescence measurement plasmid, and input promoter-YFP plasmid. Transform the gate plasmid (Figure 4-41), the same RPU standard plasmid and autofluorescence measurement plasmid for sensor measurement (Figure 4-37), and the P_{Tac}-YFP plasmid (Figure 4-42). The IPTG-inducible P_{Tac} promoter is used as the input to the gate. This allows the gate's response to different input promoter activities to be measured. The Eco1C1G1T1 UCF defines the strain as E. coli NEB 10-beta.

Characterize the fluorescence of cells carrying the gate plasmid, the input promoter-YFP plasmid, the RPU standard plasmid, and autofluorescence under a series of inducer concentrations. Grow the cells according to the UCF growth specifications. Induce the input promoter with various concentrations of inducer; at least six inducer concentrations should be used so that the gate output evenly spans the entire output range. For each inducer treatment, calculate the median fluorescence for the sensor, RPU plasmid, and empty cells. The RPU equation in the previous section is used to convert the median YFP fluorescence to RPU.

Fit the input-output gate data to an equation capturing the response function. Step #3 results in a series of data generated by different concentrations of inducer that can be used to generate the response function of the gate. This is done by plotting the activity of the input promoter (using the plasmid in Figure 4-42) versus the activity of the output promoter of the gate (using the plasmid in Figure 4-41) at each concentration of inducer. Both of these measurements are first normalized to RPUs. Then, an equation describing the gate response is fit to the data to generate the response function used by Cello. This response function has the form of a Hill equation when the regulator is a repressor,

$$y = y_{min} + (y_{max} - y_{min})\frac{K^n}{x^n + K^n}$$

where n is the Hill coefficient, K is the threshold, y_{max} and y_{min} is the maximum and minimum activity of the output promoter, and x is the activity of the input promoter.

(Optional) Convert the response function cytometry distributions to RPU. Fluorescence histograms can also be converted from arbitrary fluorescence units to RPU. This can be accomplished in a single step by taking the gate output histogram and multiplying all the fluorescence-axis values by the constant c:

$$c = \frac{\langle YFP \rangle - \langle YFP \rangle_0}{\langle YFP \rangle (\langle YFP \rangle_{RPU} - \langle YFP \rangle_0)}$$

Effectively, this rescaling performs two transformations of the data: (1) multiplying the x-axis by $(<YFP>—<YFP>_0)/<YFP>$ shifts the median of the gate's fluorescence distribution down in log-space by the autofluorescence median; and (2) division by $<YFP>_{RPU}—<YFP>_0$ normalizes the x-axis values to the RPU standard. These operations are visualized in Figure 4-43.

This section provides an example for the measurement of the PhlF gate with RBS-P2. We measured the YFP expression from the PhlF(RBS-P2) gate and the P_{Tac}-YFP plasmid in the following IPTG concentrations: 0, 5, 10, 20, 30, 40, 50, 70, 100, 150, 200, and 1000 µM. We also measured cellular autofluorescence and YFP expression using the RPU standard (pAN1717). The median fluorescence value for the RPU standard $<YFP>_{RPU} = 1540$ and the median autofluorescence value $<YFP>_0 = 17.4$. The median fluorescence values for the PhlF gate and P_{Tac}-YFP were converted to RPU for each concentration of IPTG, and then the PhlF output RPU values were plotted against the P_{Tac}-YFP RPU values to generate the gate's response curve (Figure 4-44). Using values $y_{max} = 4.1$ RPU and $y_{min} = 0.017$ RPU, a Hill equation was fit to the data points and resulted in the fitted parameters $K = 0.13$ RPU and $n = 0.92$.

4.4 Initial gate assembly and failure modes

We constructed a gate library based on a set of 16 Tet repressor (TetR) homologues that are orthogonal; that is, they do not bind to each other's promoters (Stanton *et al*, 2014). These can be converted into simple NOT/NOR gates by having the input promoter(s) drive the expression of the repressor, after which there is a terminator and output promoter. Due to a lack of strong terminators when these gates were built, the same terminator (BBa_B0015) was reused for each one. Each repressor had a different ribosome binding site (RBS), chosen to maximize the dynamic range.

These gates can be connected to form simple functional circuits; however, in each case additional tuning was required and the dynamic range of the output was low (Stanton *et al*, 2014). We tested the ability of the response functions of the gates to predict circuit behavior as a whole with no additional tuning. We designed a set of 8 simple circuits from these gates that required between one and four repressors (Fig. 4-3a and (Methods)). Nearly all of the circuits generated an incorrect response. Only the (A NIMPLY B) gate functioned properly and 6 out of 8 circuits had their output states either all OFF or all ON for every input condition. Across all the circuits, 13 out of 32 output states were correct, which is comparable to what would be expected from a process that generates random outputs.

We used this test set to determine common causes for circuit failure (Figs. 4-11 through 4-15). When paired with different promoters, gates often generated an unpredictable response and this was apparent even for circuits based on a single repressor (Lou *et al*, 2012a). The promoters generated transcripts with different untranslated regions (5'-UTRs), which can strongly influence gene expression (Kosuri *et al*, 2013; Salis *et al*, 2009). A second problem was that some promoters in the downstream "position 2" of a NOR gate (Fig. 4-2c) can reduce transcription from the upstream promoter, a phenomenon we refer to as "roadblocking". Third, some circuits had growth defects, which were caused by repressors that become toxic when expressed past a threshold (Fig. 4-3d). Fourth, several circuits were genetically unstable because of homologous recombination of parts re-used in the same circuit (Sleight & Sauro, 2013; Chen *et al*, 2013).

4.4.1 Non-insulated gates: predicted and measured outputs

Originally, the simple circuits (Figure 4-3a, left data column) were build based on non-insulated gates taken directly from a subset of the repressors previously

characterized (Stanton *et al*, 2014). We allowed a library of 16 members, each of which used the same terminator (BBa_B0015). Response functions for these gates were determined as the activity of the output promoter versus the activity of the input promoter (in RPU). The response function of each gate was fit to a Hill equation, the parameters of which are in Table 4-5.

The non-insulated gates were assembled to form the wiring diagrams shown in Figure 4-5. The gate assignments differ from those built with the insulated gates (indicated by color). The detailed parts are also different and shown in this figure. The genetic circuits were inserted into the same plasmid backbone as the insulated gates (Figure 4-1c) and included YFP on the same plasmid as the circuits (no output plasmid).

The assembly strategy used for the non-insulated circuits differed slightly from the insulated circuits. Golden Gate assembly was used to assemble the final circuits, but we used different 4 bp scars than for the insulated circuits. We also used a two-tier assembly where intermediate constructs with 1-4 transcription units were assembled first and then assembled to build the final circuit. The sensor block was also assembled with gate modules into intermediate constructs. This is in contrast to the insulated circuits where the sensor block was cloned into the plasmid before circuit assembly, and then all circuit modules were cloned into the backbone in one step.

4.4.2 Characterization of error modes

The circuits built from non-insulated gates were almost entirely non-functional. We identified several failure modes in these circuits, which when corrected fixed the circuit function. We describe the design solutions for five primary error modes in this section: mismatched response functions, promoter/5'-UTR contextual effects, promoter interference, homologous recombination, and toxicity.

Mismatched response functions. In the construction of the non-insulated gates, several of response functions were mismatched. The outputs from one gate frequently did not map onto either side of the threshold of the downstream gate (Figure 4-11). For example, in the initial construction of the XNOR circuit from non-insulated gates (Figure 4-11a), the outputs from the LmrA NOR gate (red gate) is very high, even in its repressed OFF state. These high OFF-state outputs map onto to the downstream response functions to the right of the threshold point. This causes the signal to deteriorate after the first layer. For subsequent circuit designs, the selection of repressor assignments based on the circuit's output dynamic range ensured that the response functions of gates connected to each other in a functional manner. For example, the XNOR circuit built from insulated gates (Figure 4-11b) has good predicted separation between ON and OFF promoter levels after each gate in the circuit.

Promoter/5'-UTR context effects. The response function of a NOT gate can change when connected to different input promoters (Lou *et al*, 2012). In addition, for NOR gates the connection of two promoters in series can lead to contextual effects as they create transcripts of different length (Figure 4-12), which changes the length of the 5'-UTR—a sensitive region for controlling expression (Goodman *et al*, 2013; Kosuri *et al*, 2013; Mutalik *et al*, 2013; Salis *et al*, 2009). This manifested as an error mode, where gates that functioned properly as NOT gates fail when converted to NOR gates. The promoter context and 5'-UTR effects can be mitigated through the inclusion of a ribozyme after the promoter, which cleaves the mRNA at a defined nucleotide. This makes the transcripts identical, whether they are produced by the upstream or downstream promoter.

Terminator recombination. The evolutionary stability of a genetic circuit is dependent on several factors. Repeated genetic sequences undergo homologous

recombination at a frequency that increases with the repeat length and the number of repetitions (Lovett *et al*, 2002; Shen & Huang, 1986). Initially, we designed genetic circuits that each contained the 129 bp double terminator BBa_B0015. The largest such circuit, XNOR, underwent rapid homologous recombination that resulted in a non-functional circuit (Figure 4-13). We sequenced the plasmid and found that the AmtR transcription unit had looped out of the plasmid by homologous recombination between two instances of the BBa_B0015 double terminator. This caused constitutive expression of SrpR and BM3R1 which lead to an always ON output from the circuit. To mitigate homologous recombination, we used a library of sequence-diverse strong terminators (Table 4-2) to terminate gene expression.

Roadblocking. Genetic NOR gates contain tandem promoters that drive expression of repressors. Our initial assumption was these promoters would function independently, where the activity of a downstream promoter would not impact the activity of an upstream promoter, and vice versa. In practice, we found that some promoters in the downstream position (position 2 of Figure 4-2c) could interfere with the upstream promoter when in the repressed state. We refer to this effect as "roadblocking" (the name is not intended to imply mechanism). We developed a simple system to measure the propensity of each promoter to roadblock when in the downstream position (Figure 4-14a). YFP is measured from a single promoter (P_{Tac} or P_{Tet}) by cytometry. Next, we insert a second promoter downstream from this promoter (position 2) and the impact on the upstream promoter was quantified (Figure 4-14b). We incorporated this roadblocking data into Cello by forbidding P_{BAD}, P_{Tac}, P_{PhlF}, P_{BM3R1}, P_{SrpR}, and P_{QacR} from occupying the second position in a tandem promoter. This appears as Eugene rules in the UCF and impacts the repressor assignment by disallowing multiple gates with output promoters that exhibit roadblocking to both serve as inputs to a downstream gate.

Toxicity. Certain repressors can be toxic when overexpressed, causing slow cell growth. For example, we constructed an AND gate from the PhlF, BM3R1, and QacR gates, and did not expect to see an impact on growth. However there was a growth defect when the cells were induced with 2 ng/mL aTc (which expresses QacR from P_{Tet}). When the repressors were initially characterized (Stanton *et al*, 2014), their toxicity was measured by inducing their expression from P_{Tac} using various concentrations of IPTG. However, we later found that repressors that were initially not measured to be toxic impacted cellular growth when expressed from promoters stronger than P_{Tac}, as in the case of QacR being expressed from P_{Tet}. To determine whether genes expressed at higher levels could exhibit toxicity, we cloned the repressors downstream from a tandem inducible promoter (P_{Tac}-P_{Tet}) and measured the impact on growth at various IPTG and aTc concentrations (Figure 4-15).

4.5 Insulated gates

A second generation of gates was constructed to address the observed failure modes (Fig. 4-3b). Changes took two forms: (i) new parts were added to gates to insulate them from genetic context, and (ii) rules were included in the UCF that disallow certain parts, positions, and part combinations that lead to unpredictable behavior. Transcriptional insulation was achieved for gates by adding a different strong terminator with sufficiently diverse sequences to avoid homologous recombination (Table 4-2) (Chen *et al*, 2013). The output promoters were also insulated on both sides from changes to their up- and down- stream context. Insulators consisting of a hammerhead ribozyme and downstream hairpin (RiboJ) ensure that a promoter generates the same response function irrespective of the downstream gene (Lou *et al*, 2012a). As with the terminators, to avoid recombination we had to create a library of RiboJ variants that are functionally identical but sequence-diverse so that each gate

had a unique insulator sequence (Fig. 4-8 and Table 4-1). To insulate the promoter from the upstream sequence, we added 15 nt of randomly generated DNA to extend the promoters to -50 to include regions that impact strength (Table 4-8) (Rhodius *et al*, 2012). Finally, the propensity for repressible promoters to roadblock was measured (Fig. 4-14) and these data were used to create Eugene rules in the UCF that disallow these promoters from position 2 in NOR gates (Fig. 4-2c).

The response functions were then experimentally measured for all the gates (Fig. 4-3c, 4-9, and Table 4-4). Several of the first version gates had response functions that were difficult to connect functionally to sensors or other gates (Fig. 4-2b). To increase the likelihood of finding a connection, we made versions of the gates with different RBSs that shift the response threshold (Table 4-3). The growth impact of each gate was then measured as a function of the input promoter activity to determine whether there is a toxicity threshold that should be avoided (Fig. 4-3d and 4-15). To eliminate toxic or cross-reacting repressors, the original set of 16 was reduced to 12. Only four of these caused a growth defect at high inputs and this could be avoided by the assignment algorithm (Fig. 4-32).

The 8 simple circuits were redesigned with the new gate library (Methods). The sequences were constructed as designed with no post-design tuning. All of the circuits functioned correctly, corresponding to a total of 32/32 correct output states (Fig. 4-3a).

4.5.1 Insulators of promoter context: design of ribozymes and spacers

The function of a genetic part can depend on its local genetic context; that is, the identity of up- and downstream parts (Bennett, 2010). Previously, we found that the inclusion of the RiboJ insulator ensured that the response function of a gate would not

be impacted by the identity of the input promoter (Lou *et al*, 2012). RiboJ is composed of two elements: (i) a hammerhead ribozyme derived from the satellite RNA of tobacco ringspot virus (sTRSV) that cleaves the 5'-UTR at a defined point and thereby removes upstream sequences that derive from the promoter, and (ii) an additional hairpin at the 3'-end of the ribozyme that helps expose the Shine-Dalgarno sequence of the RBS (Figure 4-5b). The entire RiboJ DNA sequence is 75 base pairs (bp), which is large enough to undergo homologous recombination if used more than once in a genetic circuit (Lovett *et al*, 2002; Sleight *et al*, 2010; Sleight & Sauro, 2013; Chen *et al*, 2013). Thus, each gate needs its own insulator with the same functionality of RiboJ but with a sequence that is different enough to prevent homologous recombination. To address this, we built and tested natural and engineered RiboJ variants and characterized both their cleavage activity and insulator functionality.

Two approaches were taken to identify ribozyme sequences that have diverse sequences but still function as insulators. First, "part mining" was performed to identify other hammerhead ribozymes derived from plant viroids and plant virus satellite RNAs. We built and tested sixteen hammerhead ribozymes (Khvorova *et al*, 2003; Dufour *et al*, 2009) (including RiboJ) and others that had been previously tested as insulators (Lou *et al*, 2012). This approach ultimately led to the characterization of nine functional natural ribozyme-based insulators (Fig. 4-5a, Table 4-1).

A second approach to library expansion was taken by diversifying the sTSRV scaffold. This was aided by a number of structural studies detailing ribozyme function (Khvorova *et al*, 2003; Dufour *et al*, 2009; Ruffner *et al*, 1990; Haseloff & Gerlach, 1988; Pley *et al*, 1994). Three design rules were implemented (Figure 4-5b). First, the sequences of the catalytic core residues (CTGATGA and GAAA) and two loops (GTGC and GTGA) were conserved (Dufour *et al*, 2009; Khvorova *et al*, 2003). Second, the total number of nucleotides and the hammerhead secondary structure were

kept intact. This was achieved by only mutating three stem regions: 5 bp of stem 1, 4 bp of stem 2, and 3 bp of stem 3. These mutated sequences were generated using the Random DNA Generator (http://www.faculty.ucr.edu/~mmaduro/random.htm; 50% GC-content). RNA secondary structures were predicted using mFold (Zuker, 2003) and were found to maintain their hammerhead structure when simulated in isolation from flanking sequences (conditions: 37°C, 1M NaCl). We built and tested 45 engineered ribozymes, of which seven were functional and used to insulate gates (Table 4-1). For both the natural and engineered RiboJ variants, the downstream hairpin sequence was held constant due to its short size (23 nucleotides), which is short enough that it should not lead to homologous recombination.

The natural and synthetic ribozymes were examined in two assays to measure cleavage activity and functional insulation. To measure cleavage, Rapid Amplification of Complementary DNA End (5'-RACE) was used to generate cDNA from mRNA by reverse-transcription for PCR amplification (Methods). Acrylamide gel analysis shows two bands: one from full-length, uncleaved mRNA and another from cleaved mRNA. The ratio between cleaved and total cDNA is used to calculate the efficiency (Figure 4-7). Several ribozymes, both engineered and natural, failed to achieve >75% cleavage efficiency. A set of 16 catalytically-active ribozymes is shown in Figure 4-7c.

A second assay was performed to determine the insulation functionality of each RiboJ variant. Following an assay developed by Lou *et al.* (Lou *et al*, 2012), we compared the expression of two genes (*gfp* and *cI-gfp*) from two different inducible promoters, P_{Tac} and $P_{LlacO-1}$ (Figure 4-8a). The *cI-gfp* fusion gene saturates when induced by pLlacO-1, whereas this saturation is not observed from the pTac promoter (Lou *et al*, 2012) (Figure 4-8b). The RiboJ insulator was originally selected because its inclusion between the pLlacO-1 promoter and the RBS ameliorated this saturation and caused the outputs from both promoters to converge onto the same line. Further,

the slopes of these lines are approximately constant, indicating that the two genes are expressed proportionally at different promoter activities. Thus, this assay is a direct measurement of insulation; in other words, the context effects that occur for particular promoter-gene combinations are reduced. All 16 RiboJ variants (including the original RiboJ) were tested via this assay and insulation was demonstrated for each (Figure 4-8c).

4.5.2 Terminator selection for transcriptional insulation

Strong terminators are needed for genetic circuits to prevent transcriptional read-through between gates. In addition, these terminators must be sequence-diverse to prevent homologous recombination (Sleight *et al*, 2010; Sleight & Sauro, 2013). For the later circuits discussed in this work, we used strong terminators that were measured previously (Chen *et al*, 2013) (Table 4-2). These replaced double terminator BBa_B0015 used in earlier circuits.

4.5.3 RBS selection to tune the response threshold

The strength of the RBS controlling repressor expression is one determinant of the threshold of a gate. When the ribozyme insulators were added to each gate, this impacted RBS strength and the thresholds shifted (or the response was completely eliminated). To alter the threshold of the insulated gates, we built and screened RBS libraries. For some gates, multiple RBSs were found that generated different thresholds. These were kept and included in the library so that there would be more ways in which the gate could be connected to others in the circuit.

The RBS libraries were built using PCR to amplify the gate plasmid with primers containing degenerate nucleotides in the region in and around the RBS (Methods).

The resulting PCR products were ligated and transformed in *E. coli* NEB 10-beta. Individual clones from the gate RBS library were screened by growing them in the presence and absence of inducer. Clones with the largest dynamic range were chosen for further characterization. The full response functions of these gates were measured. Representative cytometry histograms and Hill equation fits to the data are given in Figure 4-9. The final RBS sequences are given in Table 4-3, and the response function parameters and toxicity threshold are listed in Table 4-4.

4.5.4 Response functions and cytometry data for insulated gates

Production of YFP from the insulated gates' outputs were measured at various inducer concentrations by cytometry and converted to output relative expression units (RPU). For each of these gates, IPTG was used to induce gate expression from the P_{Tac} promoter. Additionally, inducer concentrations were converted to input promoter activity by measuring expression of YFP from P_{Tac} at those inducer concentrations (Figure 4-9a). The median input and output RPU values were plotted for each inducer concentration to create the experimental response function (Figure 4-9b).

4.5.5 Insulated gates: predicted and measured outputs

The circuits constructed from non-insulated gates were rebuilt using insulated gates and design rules extracted from the previous section. These repressor assignments were found using a MATLAB script that was developed prior to the complete Cello software (Methods). For each circuit, we started with the circuit diagram (Figure 4-3a) and a subset of repressors from Figure 4-3b. We enumerated all possible repressor assignments for every gate, with the exception of assignments that would result in a

promoter roadblock. For each gate assignment, we propagated sensor input signals through the gate response functions to the circuit output, summing promoter activities at NOR and OR gate inputs. Each circuit assignment was scored as the ratio between the lowest ON state and the highest OFF state. The highest scoring assignment was selected for construction and testing (Methods). For these circuits, the only promoters forbidden from being in position 2 of the NOR gates were P_{SrpR} and P_{Tac}. Figure 4-16 shows the experimental data from Figure 4-3a alongside the simulated outputs for the circuits.

4.6 Circuit design automation using Cello

Cello was used to design a large set of 52 additional circuits based on the insulated gates (Fig. 4-4). These circuits include a Priority Detector (that prioritizes the inputs and selects which output is ON based on the highest priority input that is ON), well-known functions (e.g., a multiplexor), as well logic underlying cellular automaton pattern formation (e.g., "Rule 30") (Wolfram, 2002). Additional 3-input 1-output logic circuits are built that demonstrate the ability to integrate inputs in different ways. This could be applied to turn a cellular function on or off in response to an environment defined by multiple signals. Each of the 52 circuits was specified either by using behavioral Verilog (and Cello performs the logic minimization step) or by performing a separate enumeration to identify the global minimum number of gates and specifying the circuit diagram using structural Verilog (This was done for 0x0E, 0x19, 0x1C, 0x38, 0x3D, 0x6E, 0x81, 0xB9, 0xC6, 0xC7, 0xBD, and 0xC8.). Subsequently, the global minimum 3-input logic gates were included in the UCF so that they could be incorporated as motifs in larger circuits in future designs (Fig. 4-30). For each circuit, the sensor promoters and ON/OFF values were specified, the Eco1C1G1T1 UCF

selected, and a DNA sequence was automatically generated by Cello. DNA synthesis (Kosuri & Church, 2014) and assembly was used to build each sequence, which contained up to 10 regulators and 55 parts. The output states of each circuit were measured by flow cytometry and compared with the Cello predictions. No additional tuning was done to diverge from the Cello-predicted sequence.

37 of the 52 circuits functioned as predicted, such that all of the output states matched desired ON and OFF levels (Fig. 4-4a). Further, the predicted cytometry distributions closely matched those measured experimentally. Out of 412 output states across all circuits we built, 92% were correct. The Consensus circuit (output is ON only when all three inputs agree) is the largest, containing 10 regulatory proteins (7 repressors from NOT/NOR gates and 3 from the inducible systems) and 55 genetic parts. Two of the circuits with 4 layers (0x3D and 0x8E) were selected to characterize the switching dynamics between states (Fig. 4-4b). Interestingly, 0x8E shows a transient incorrect state, known as a "fault" in electronics, when the inputs are changed from -/-/- to +/-/+. This is consistent with the last NOR gate transiently receiving ON/OFF inputs until one of the signals transits two layers.

Of the 52 circuits, 7 were incorrect in one state, 2 were incorrect in two states, and 5 had ≥3 failures (Figs. 4-17 through 4-19 and 4-25). As more gates were included in a design, there was a higher probability of failure (Fig. 4-5a). Two circuits were found to cause a growth defect (Fig. 4-5b). The circuits that failed in a few states tended to match the remaining states closely, so the initial design can be used as the basis for further rounds of optimization. Debugging experiments were performed to determine which gates fail as a means to focus optimization. This was done by creating a set of plasmids that contain each gate's output promoter fused to YFP. These plasmids were transformed with the circuit in lieu of the output plasmid and the response of the internal gate was measured for all combinations of inputs. An example of this is shown

in Fig. 4-5c and several other examples are shown in Fig. 4-26. From this analysis, most of the circuit failures point to unexpected behavior from the aTc sensor (7 circuits) or AmtR gate (2 circuits).

Screening design variants has the potential to increase the probability of success, particularly for larger circuits. To do this, Cello outputs a Eugene file that contains architectural rules from the UCF as well as constraints to enforce the circuit diagram and repressor assignments (Fig. 4-34). The user can specify the size of the library and a combinatorial design algorithm (Smanski *et al*, 2014) generates the target number of constructs. Although all of the systems should be functionally equivalent, subtle changes in their composition may impact circuit function through hidden effects (*e.g.*, transcriptional read-through or promoter interference). We tested this approach by designing a Majority circuit (Fig. 4-5d), whose output is ON when a majority of its inputs are ON. We built a small library of six constructs that maintained the same circuit diagram and repressor assignments, but in which the order and orientation of genes was allowed to vary (Fig. 4-5e). Several of these circuits functioned correctly and the response of the best is shown in Fig. 4-5f.

4.6.1 Complete circuit data

We constructed a library of 3-input, 1-output genetic circuits using Cello. All the fully functional circuits are shown with data and predictions in Figure 4. The remaining circuits are shown in Figures 4-17 through 4-19. Experimental/predicted distributions and replicates for the Majority circuit variants are shown in Figures 4-20 and 4-21. Experimental/predicted distributions and replicates for three alternate circuit assignments are shown in Figure 4-22 and 4-23. Replicates for the circuit library are shown in Figures 4-24 and 4-25.

4.6.2 Majority circuit variants

The original design for the Majority circuit produced an output that was higher than expected for input state 2 (+IPTG, —aTc, and —arabinose). We constructed an additional five layouts that retained the same repressor assignments to test whether we could fix that output state (Figure 4-5e). We hypothesized that subtle contextual effects might arise in different layouts (terminator read-through, part interference, cryptic promoters, etc.), and that these effects could improve the circuit's performance.

For the original circuit (Design #1), we used the default Cello layout where all transcription units point in the forward orientation and the repressors have a defined order (PhlF, SrpR, BM3R1, BetI, HlyIIR, AmtR). Design #2 reverses the order of all transcription units, keeping them pointed in the forward orientation. Design #3 clusters the three NOT gates together in the first half of the DNA sequence, and the three NOR gates together in the second half. Design #4 clusters one three gate sub-circuit (AmtR, BetI, SrpR) in the first half of the DNA sequence, and a second three gate sub-circuit (BM3R1, PhlF, HlyIIR) in the second half. Design #5 scrambles the order of the transcription units, keeping them pointed in the forward orientation. Design #6 uses the default transcription unit positions, but alternates their orientation so that the first transcription unit points backward, the second points forward, the third points backward, and so on. Each of these genetic layouts was physically constructed by simply changing the 4 bp Golden Gate scars that occur between transcription units.

The precise rules that govern the layout of these circuits were converted to Eugene code (Data File S3). Each variant used a different Eugene file, and the only differences are a small set of rules in the "circuit device". The different rule sets only use three different Eugene keywords, but in different combinations: ALL_FORWARD, ALTERNATE_ORIENTATION, and BEFORE (e.g. gate_PhlF BEFORE

gate_SrpR). The promoter order was also fixed due to roadblocking constraints in "gate devices".

In principle, if all the gates were perfectly modular and exhibited no contextual behavior differences, then all six layouts should function identically. Instead, we observed slight shifts in the output distributions for each circuit. The alternating orientation layout (Design #6) produced the greatest fold-change between the lowest ON state and the highest OFF state; furthermore, the high OFF state from Design #1 was decreased to more closely match the predictions. Representative experimentally-measured histograms and the predicted outputs are shown in Figure 4-20.

4.6.3 Alternate repressor assignments

In addition to testing different layouts for a single gate assignment, we constructed three of the circuits (Multiplexer, Consensus, and Majority) using alternate repressor assignments predicted by Cello (Figure 4-22). The alternate Multiplexer circuit replaces only the terminal PhlF NOR gate from the original assignment (Figure 4-4a) with a BM3R1 NOR gate. The outputs are correct for all of the assignments. The alternate Consensus circuit swaps two gate assignments (HlyIIR and PhlF) from the original circuit. This swap results in two failed output states, whereas the original version had all states correct (Figure 4-4a). The alternate Majority circuit changes the assignment for every gate, except for the HlyIIR NOT gate connected to the P_{Tac} input. While most of the repressors are present in the same circuit layer in both circuits, BM3R1 is absent in the alternate assignment and AmeR is present. The alternate Majority circuit's output behaves correctly for every input state.

4.6.4 Debugging genetic circuits

We developed a strategy for "debugging" a malfunctioning circuit to determine which gate is causing the failure. This was done by creating a series of plasmids that transcriptionally fuse the output promoter of each gate to *yfp*. These constructs are carried on a plasmid with a pSC101 origin of replication and a spectinomycin resistance marker. This is co-transformed with the circuit plasmid (the plasmid containing the output promoter of the circuit is not included) and the cells are grown and assayed in the same way that the circuit is characterized (Methods). This is done in two steps. First, a single screen is performed on all gates and all combinations of input conditions. From this, it can be seen which gates are failing to respond as expected. Then, the screen is followed up with more detailed measurements including replicates that focus on the failed gate. A summary of these experiments is shown in Fig. 4-26, which shows the subset of data highlighting the failure discovered. From these data, the gate where the problem originates can be deduced and the impact as the error propagates through the circuit observed. The most prevalent failure mode appears linked to the P_{Tet} promoter, an effect we observed in seven cases (0xF9, 0x06, 0x9F, 0xB9, 0x19, 0x36, and 0xC1). We also saw a repeated failure associated with the use of the AmtR gate (Figure 4-26: 0x98 and Figure 4-5c: 0xC9).

4.7 Discussion

The design of synthetic regulatory networks has been dominated by manual trial-and-error tinkering at the nucleotide level. Cello automates the selection and concatenation of parts and balancing the associated constraints. By doing this, it enables more rapid design of larger multi-part systems; the circuits that we present here are larger and more complex than most that have been built by hand. Out of 60

circuits designed automatically, 45 functioned as designed (Figs. 4-2a, 4-4, 4-5 and 4-22). Our largest circuit has 12 regulated promoters, doubling a plateau first noted in 2009 (Purnick & Weiss, 2009a). The DNA sequence output represents a testable prediction that either validates the underlying theory or reveals failure modes that can be addressed in the gate design. Experiments in which repressors were used to build synthetic logic gates showed that this often led to nonsensical functions that could not be predicted from the known interactions (Cox *et al*, 2007). Quantifying why predictions fail, where systems break, and how the host evolves can be addressed through engineering. Iterative co-development of robust gates and software converged on genetic systems that are highly repetitive and modular, in stark contrast to the encoding of natural networks.

The future of engineering biology will require integrated design across many subcellular systems, including the creation of sensors that can process many stimuli, management of resources and metabolites, and control over multiple cellular functions (communication, stress response, chemotaxis, etc.). Within this greater framework, our approach is to separate the design and construction of synthetic circuits from engineering considerations for other cellular processes. Working with transcriptional circuits establishes a discrete boundary that other methods can engage to create a desired circuit to specification. For, example could build circuits for which the sensors had been designed using all-atom biophysical models (Dahiyat & Mayo, 1997; Looger *et al*, 2003; Tinberg *et al*, 2013) and the outputs used to control enzyme expression levels, as determined via metabolic flux models (Henry *et al*, 2007). Integration with amorphous computing would enable spatial and community design (Rudge *et al*, 2012; Jang *et al*, 2012; Blanchard *et al*, 2014). Integrating across these computer aided design (CAD) tools in a way that automates design choices and balances constraints will be critical to advance the complexity of genetic engineering projects.

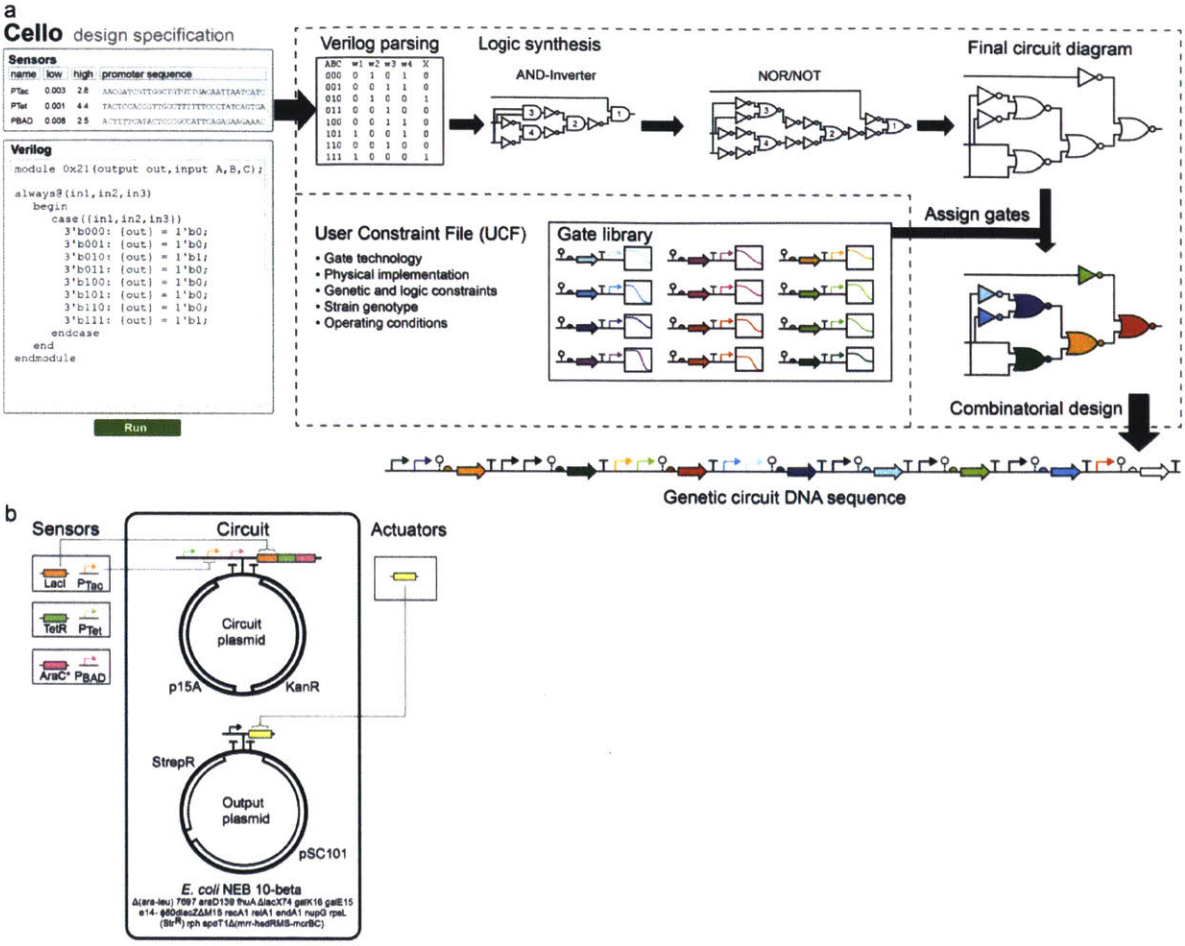

Figure 4-1: Overview of Cello. (A) Cello users write Verilog code and select or upload sensors and a UCF. Based on the Verilog design, a truth table is constructed, from which a circuit diagram is synthesized. Regulators are assigned from a library to each gate (each color is a different repressor). Combinatorial design is then used to concatenate parts into a linear DNA sequence. SBOL Visual (Quinn *et al*, 2015) is used for the part symbols. Raised arrows are promoters, circles with dashed stems are ribozyme insulators, hemispheres are RBSs, large arrows are protein coding sequences, and "T"s are terminators. Part colors correspond to physical gates. (B) The physical specification for the Eco1C1G1T1 UCF is shown. The circuit and sensors are inserted into one plasmid, whereas the other contains the circuit output promoter, which can be used to drive the expression of a fluorescent protein or other actuator. Both plasmids must be present in the specified strain for the design to be valid.

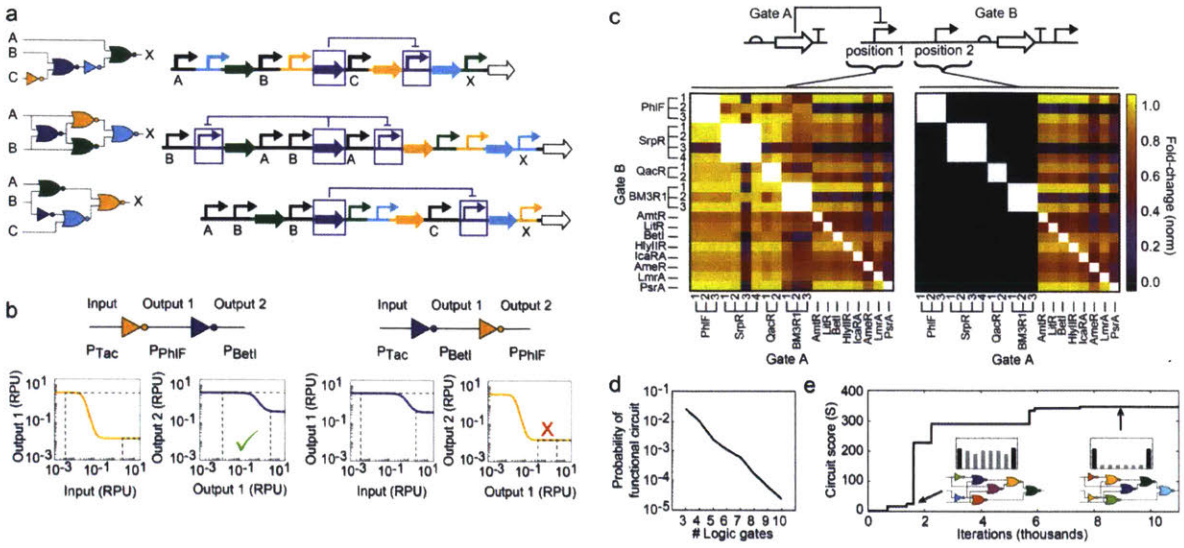

Figure 4-2: Assignment of genetic gates to the circuit diagram. (A) A set of four gates based on different repressors (colors) connected in various permutations to build different circuit functions. The inputs (A, B, and C) are sensor input promoters and the circuit output promoter (X) controls the actuating gene. (B) The shapes of the gate response functions determine whether they can be functionally connected. The orange gate (PhlF) has a large dynamic range (dashed lines) that spans the threshold of the purple gate (BetI). However, in the reverse order, the gates do not functionally connect. (C) Combinatorial relations of repressors from the insulated gate library are shown in the up- (Gate A) and down- (Gate B) stream position. Colors indicate whether the gates can be connected (yellow) or not (black). Fold-change (normalized) is calculated as the maximum output range that can be achieved by connecting Gate A to Gate B. Numbers indicate different RBSs. The left graph shows when Gate A regulates position 1 and the right graph when it regulates position 2. Gates that are excluded from position 2 due to roadblocking are shown in black (Fig. 4-14). (D) The probability of finding a functional circuit versus the number of logic gates. The probability of a functional circuit is defined as the likelihood that a random assignment passes input threshold analysis (Fig. 4-35) and has no roadblocking combinations (Fig. 4-2c). (E) The convergence of the simulated annealing gate assignment algorithm (Fig. 4-33). Inset: black bars should be ON, gray bars should be OFF; the y-axis is the output in RPU on a log scale and the x-axis is the input states (from left to right: 000, 001, 010, 011, 110, 101, 110, 111). The circuit score (S) is defined as the ratio of the lowest predicted ON state to the highest predicted OFF state (Fig. 4-31 and Equation S2). An example search is shown for the circuit diagram in the inset; colors correspond to repressors assigned to each gate (Fig. 4-3b).

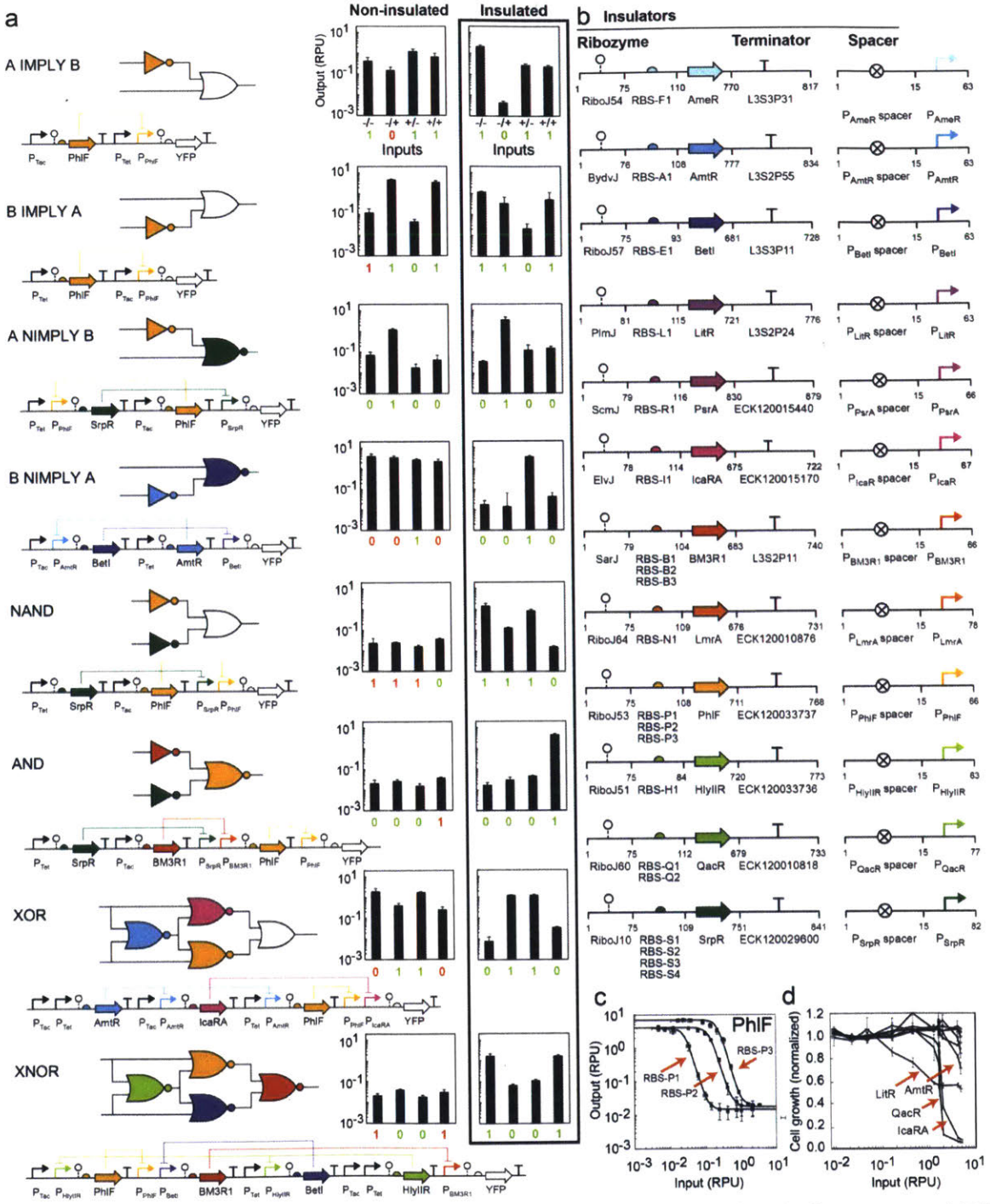

Figure 4-3: Impact of gate insulation. (A) The logic function, circuit diagram, and DNA construct are shown for each genetic circuit. Only the insulated circuit schematics are shown; the equivalent information for the non-insulated circuits is shown in Fig. 4-10. The expected output for each circuit is shown at the bottom of each bar graph as a 1 for

ON and 0 for OFF. The numbers are colored whether the state is predicted correctly (green) or incorrectly (red). For non-insulated circuits, inputs correspond to the absence or presence of 1 mM IPTG (right -/+) and 20 μM 3OC6HSL (left -/+). For insulated circuits, inputs correspond to the absence or presence of 1 mM IPTG (right -/+) and 2 ng/mL aTc (left -/+) Methods). (B) The architectures of the insulated gates. Some gates have multiple versions with different RBS sequences. The gate DNA sequences are provided in Table 4-8. (C) An example of a response function is shown for a NOT gate based on the PhlF repressor. The change in the threshold for the three RBSs is shown. Data for all insulated gates are shown in Fig. 4-9. (D) The impact of each gate on cell growth as a function of its input promoter activity. Cell growth was measured as OD_{600} and normalized by the growth of the no-inducer control six hours after induction (Methods). The four gates that reduced growth >20% are indicated by red arrows. Error bars are one standard deviation of normalized cell growth (y-axis: part d) and the median (y-axis: parts a and c; x-axis: parts c and d) for three independent experiments performed on the same day.

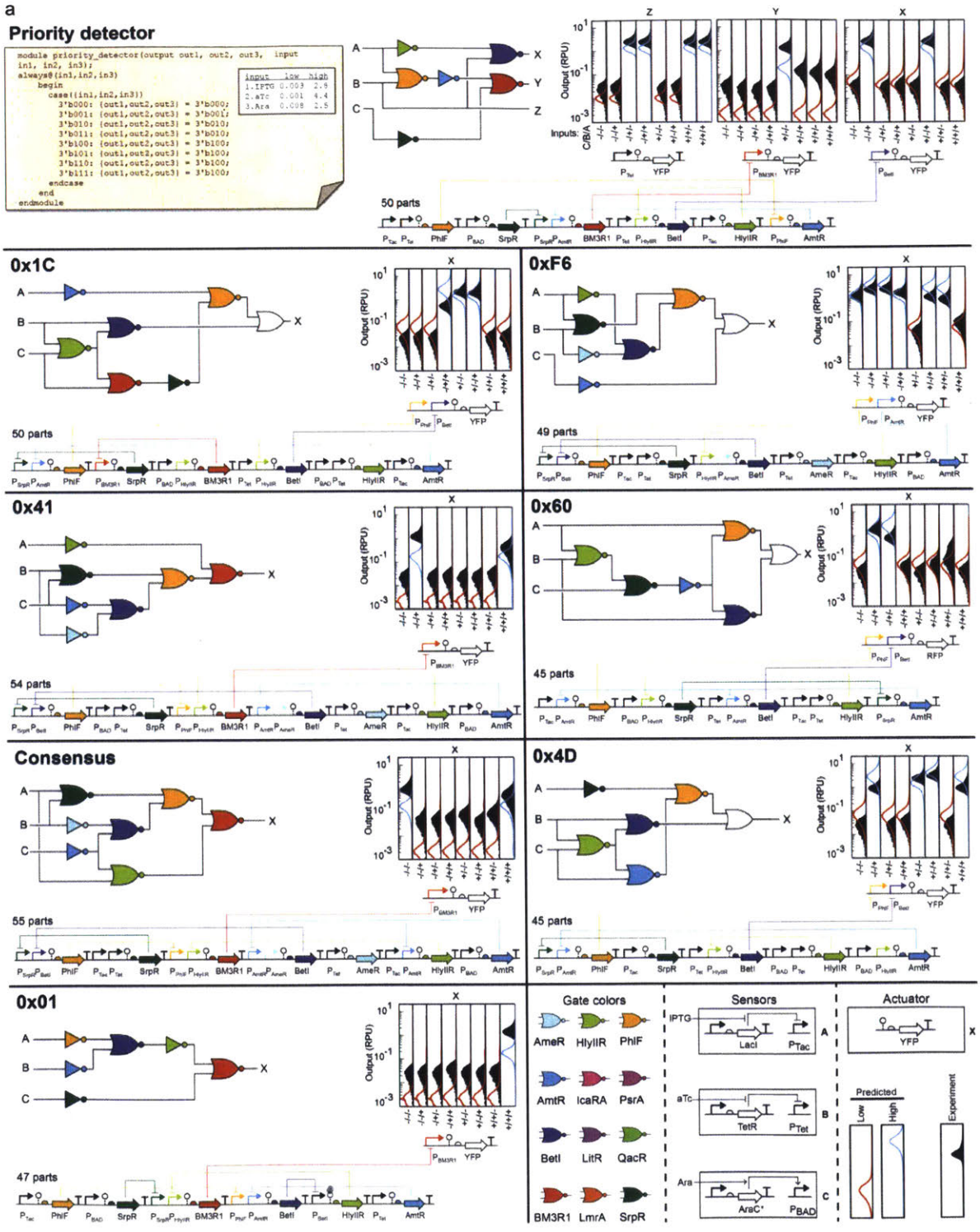

Figure 4-4 (part 1)

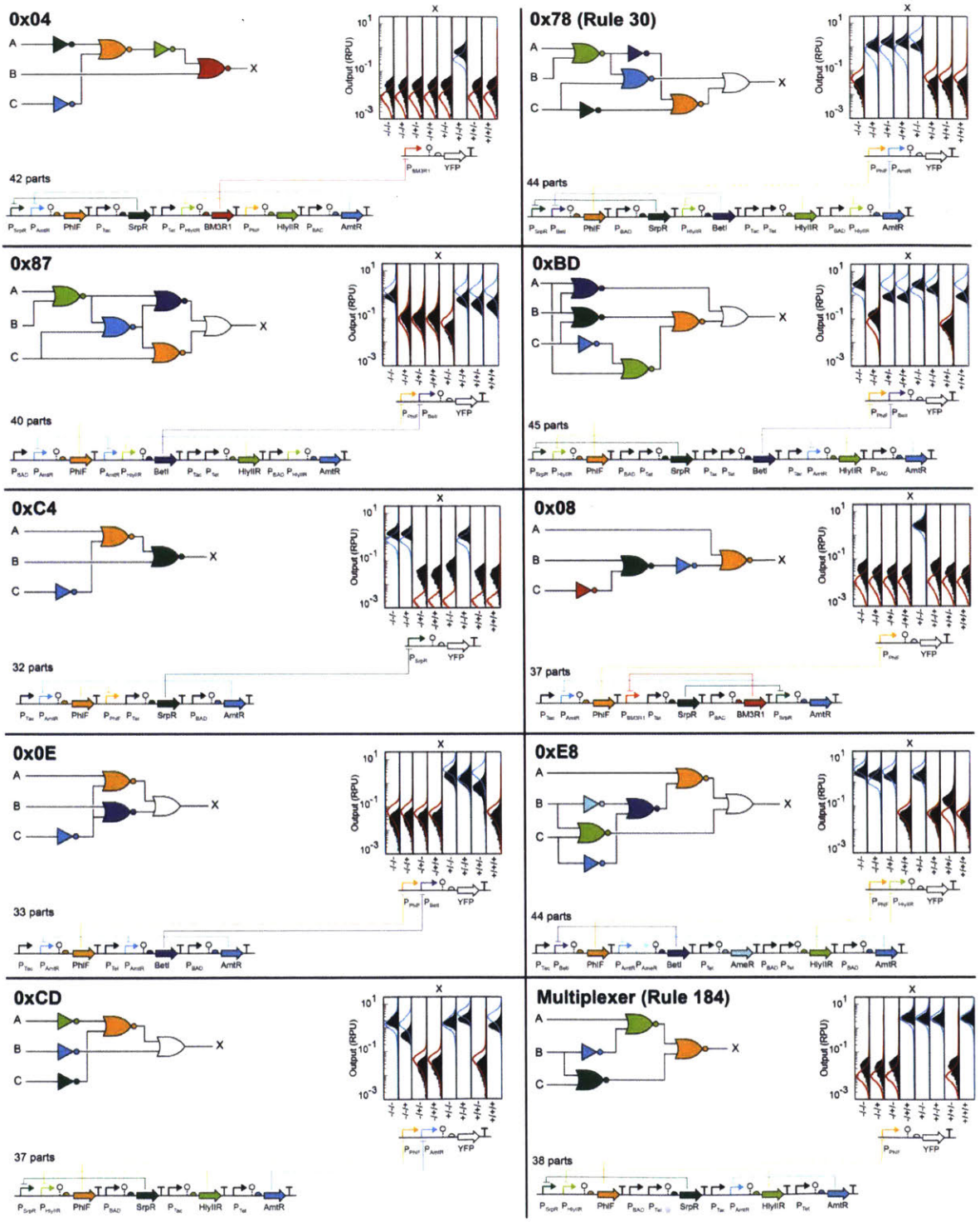

Figure 4-4 (part 2)

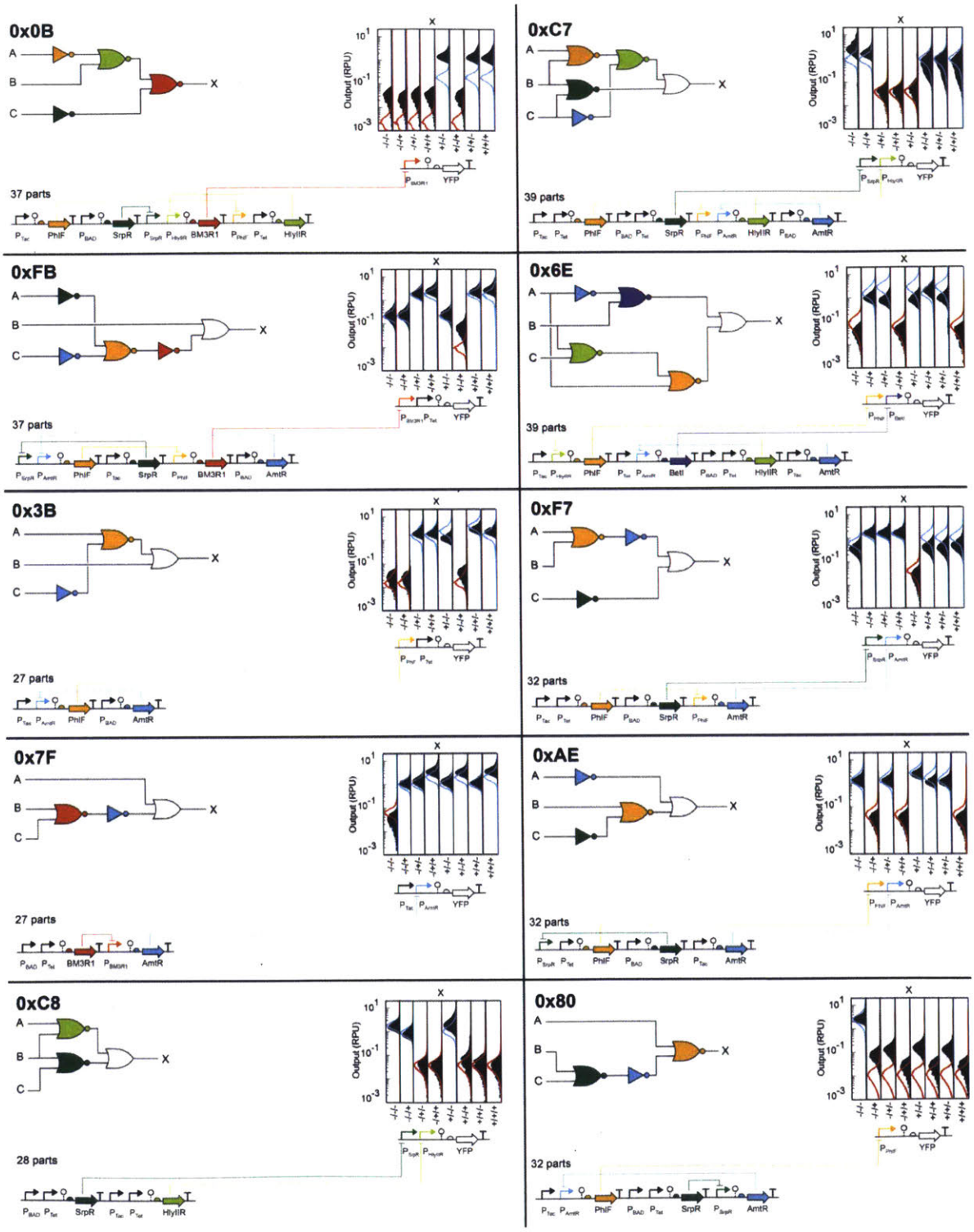

Figure 4-4 (part 3)

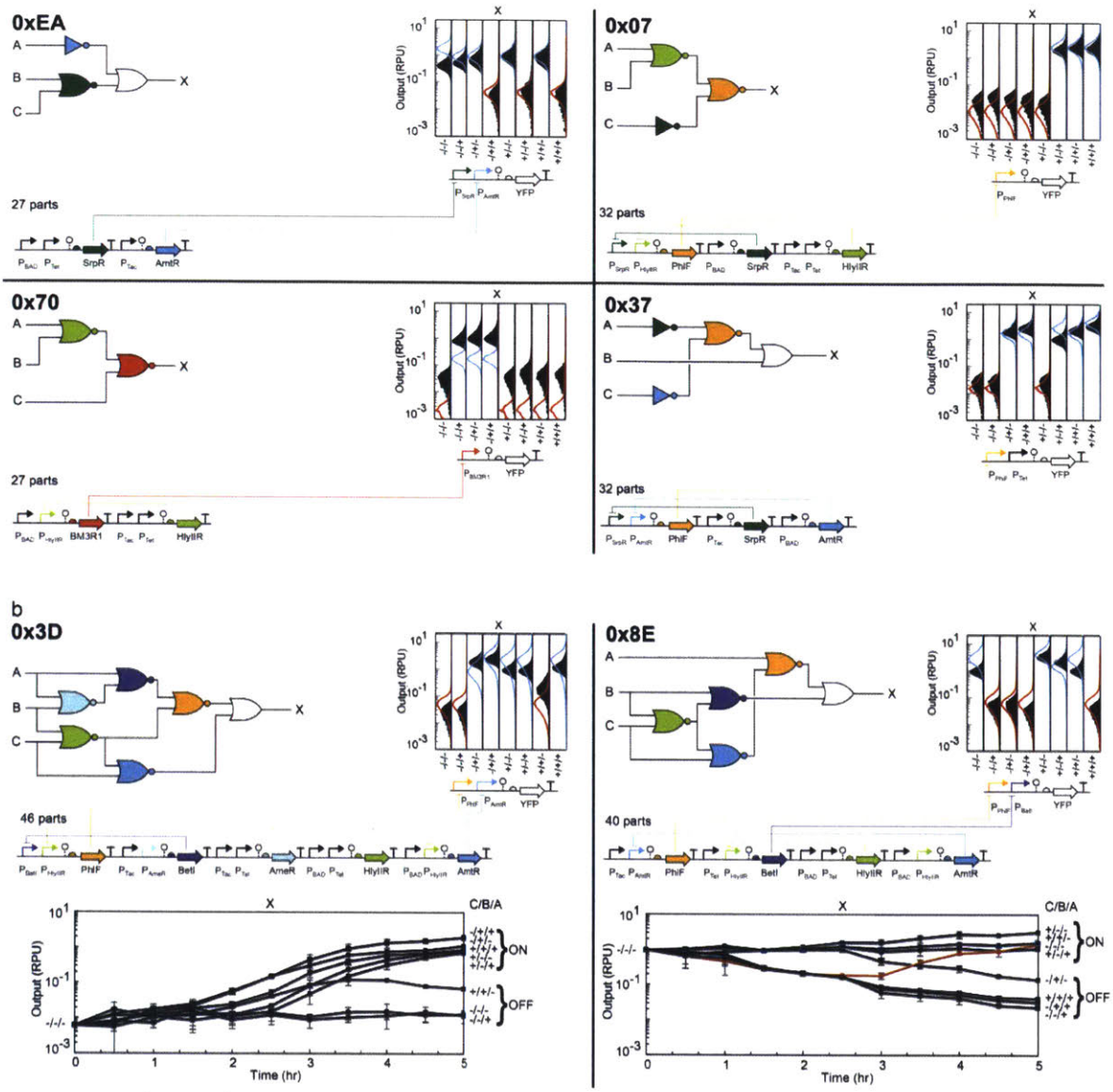

Figure 4-4 (part 4): Automated design of circuits by Cello. (A) An example of the code along with the input states (in RPU) is shown for the Priority Detector circuit. All circuits designed by Cello that had correct output states are shown along with their genetic schematics, output predictions, and experimental measurements. The inputs A, B, and C correspond to the P_{Tac}, P_{Tet}, and P_{BAD} sensor promoter activities; their corresponding regulators (LacI, TetR, and AraC*) are not shown in the schematics. The outputs (X, Y, and Z) correspond to YFP driven from output promoters in separate experiments. Solid black distributions are experimental data, and blue/red line distributions are computational predictions from Cello (Fig. 4-35). The number of parts for each circuit includes all functional DNA parts in the circuit (promoters, ribozymes, RBSs, protein

coding sequences, and terminators), plus 8 parts for the sensor block and 2 parts for the plasmid backbones. Inputs correspond to the absence or presence of 1 mM IPTG (top -/+), 2 ng/mL aTc (middle -/+), and 5 mM L-arabinose (bottom -/+). Replicates are provided in Fig. 4-24. When the circuit does not have a common name (*e.g.*, Priority Detector), a hexadecimal naming system is used (e.g., 0x41). The names starting with "Rule" refer to Wolfram's cellular automaton convention (Wolfram, 2002). (B) Time-course data are included for two circuits. The circuits are maintained in the -/-/- state for three hours prior to induction and then switched to the other eight possible states at time = 0 hr. Error bars are one standard deviation of RPU median performed on three separate days.

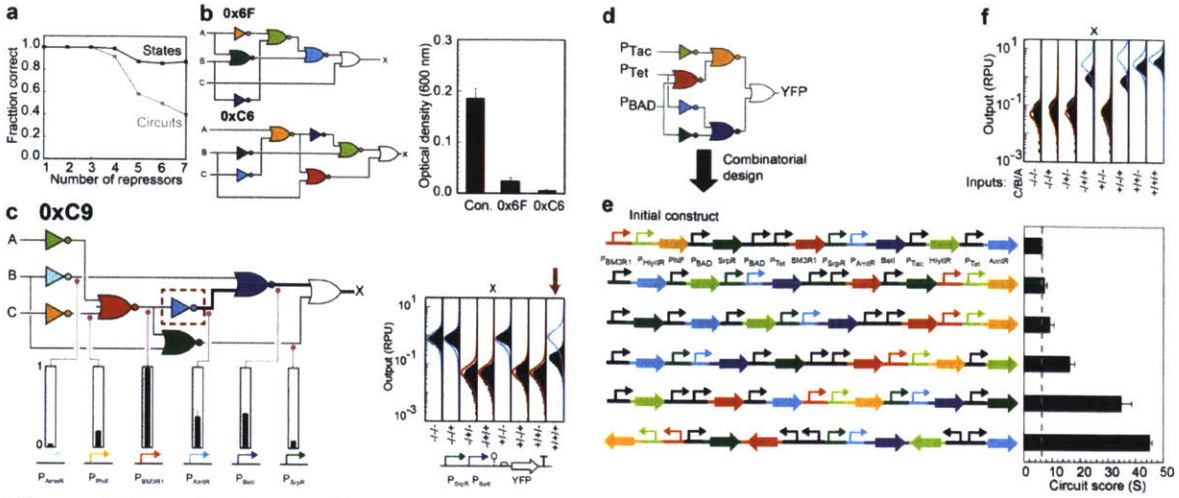

Figure 4-5: Analysis of circuit failures and the design of multiple constructs by combinatorial design. (A) For the library of 60 circuits (Figs. 4-3a, 4-4, 4-17 through 4-19, and 4-22), the fraction of correct states (black) and the fraction of fully correct circuits (gray) are shown versus the number of repressors in the circuit. (B) The impact on cell growth is shown for the two circuits that fail due to toxicity. The control bar is for cells containing the RPU standard plasmid only. The average of three experiments performed on different days and the error bars represent the standard deviation. (C) An example of circuit debugging is shown. All combinations of inputs for all wires were tested; for clarity, only a subset of debugging for the failed state (+/+/+) is shown. The data is normalized to [0,1] to correct for the dynamic range across gates. In this case, the failure originates when the AmtR gate produces an intermediate response that then propagates through the circuit. (D) The circuit diagram for a "Majority" circuit is shown with colors corresponding to repressors. (E) Six layouts were designed for this circuit that maintain the same repressor assignments, but allow the order and orientation of the gates to vary. The circuit score (S) is defined as the ratio of the lowest ON state's median to the highest OFF state's median. Error bars are one standard deviation for two experiments performed on different days. Cytometry distributions for each design are shown in Fig. 4-20. The dashed line marks the lowest circuit score in the library. (F) The predictions and cytometry distributions for the final design are shown. The format and inducer concentrations are as described in Fig. 4-4.

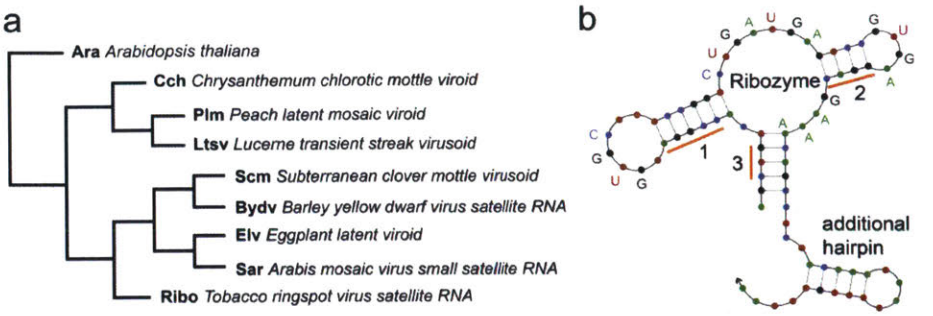

Figure 4-6: Expanding a library of hammerhead ribozymes. (A) Phylogenetic tree of functional hammerhead ribozyme-based insulators. "Ribo" is the sequence used to build RiboJ. (B) Secondary structure of a hammerhead ribozyme-based insulator including the downstream hairpin. Conserved sequence regions are shown as defined nucleotides and mutable regions are shown adjacent to orange lines (1, 2, and 3).

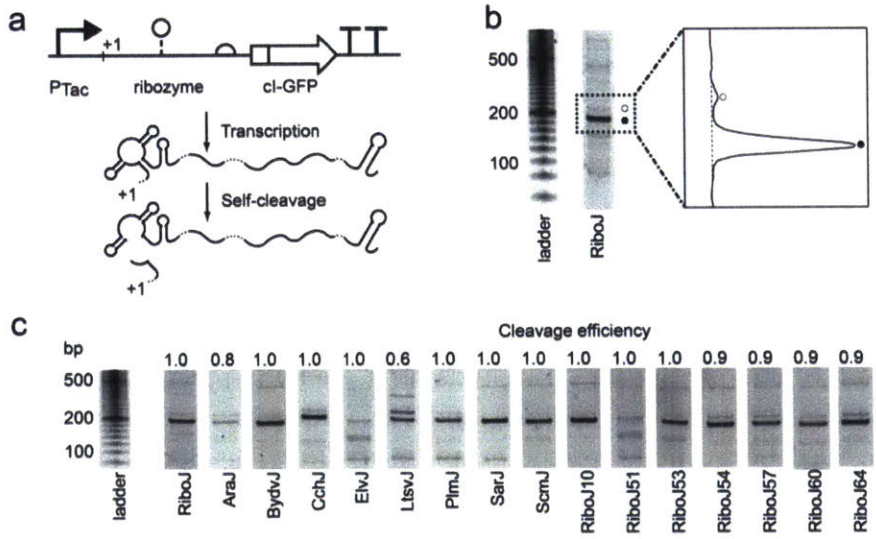

Figure 4-7: Cleavage activity of 16 ribozyme insulators. (A) Schematic of ribozyme activity and measurement using 5'-RACE (Methods). Post-transcription, the hammerhead ribozyme folds and cleaves itself at its 5'-end. The measurement plasmids are pJS1-pJS68. (B) Quantifying ribozyme cleavage activity using acrylamide gel electrophoresis and image processing. Full-length and cleaved cDNA products are separated and visualized, and then the band intensities (area under the curve, inset) are quantified using ImageJ. The intensity ratio of cleaved product (shorter band, filled circle) to the full-length product (longer band, empty circle) plus cleaved product yields the cleavage efficiency. (C) Acrylamide gel electrophoresis images and cleavage efficiencies of 16 ribozyme insulators.

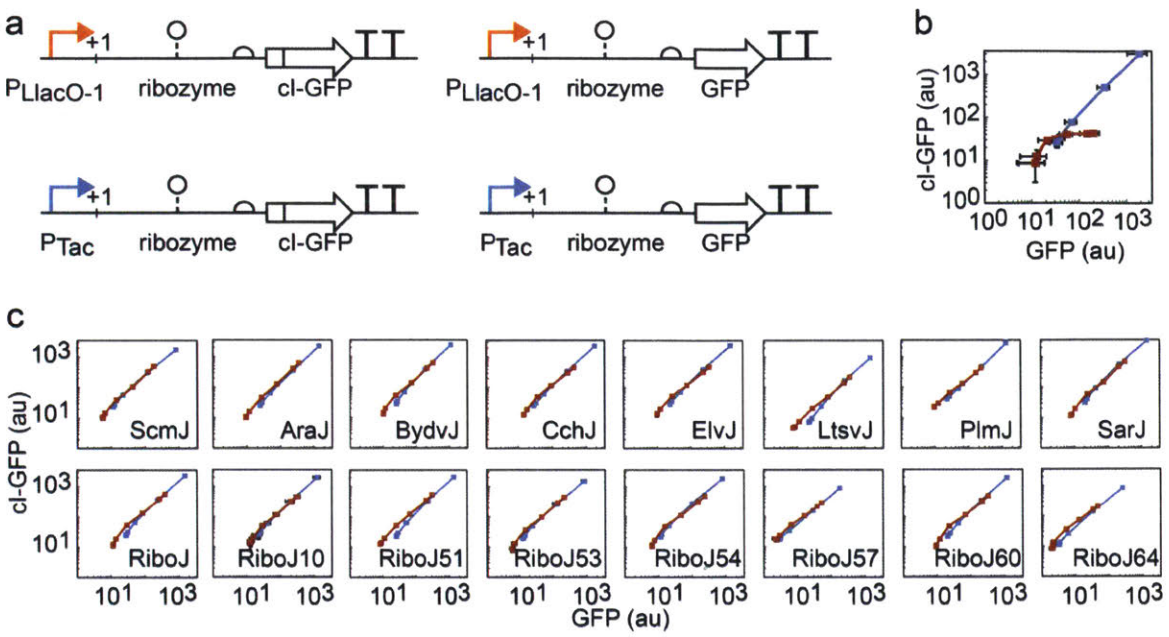

Figure 4-8: Insulating functionality of 16 ribozyme-based insulators. (A) Schematics of genetic constructs used to determine insulating functionality of ribozymes. Two genes (cI-GFP and GFP) are each induced from one of two promoters ($P_{LlacO-1}$ and P_{Tac}) with various IPTG concentrations. The plasmids used for this study are pJS1-pJS68. (B) Expression of cI-GFP versus GFP for $P_{LlacO-1}$ (red line) and P_{Tac} (blue line) when no ribozyme insulators are present. This experiment was performed to recapitulate the experiment in ref (Lou *et al*, 2012). Plasmids used are pJS1-pJS4. (C) Expression of cI-GFP versus GFP for $P_{LlacO-1}$ (red line) and P_{Tac} (blue line) when various ribozyme insulators are used between the promoter and 5'-UTR. The slopes of the $P_{LlacO-1}$ and P_{Tac} lines for each ribozyme are as follows: ScmJ ($P_{LlacO-1}$ = 2.8, P_{Tac} = 2.2); AraJ (1.8, 1.6); BydvJ (2.0, 2.2); CchJ (1.1, 1.3); ElvJ (1.4, 1.6); LtsvJ (0.60, 0.65); PlmJ (1.9, 2.2); SarJ (2.2, 2.3); RiboJ (1.5, 1.5); RiboJ10 (1.4, 1.6); RiboJ51 (1.8, 1.5); RiboJ53 (2.1, 1.8); RiboJ54 (2.0, 2.3); RiboJ57 (5.7, 5.0); RiboJ60 (1.4, 1.6); and RiboJ64 (4.9, 3.5). For panels (B) and (C), error bars are one standard deviation of the median for three experiments performed on different days.

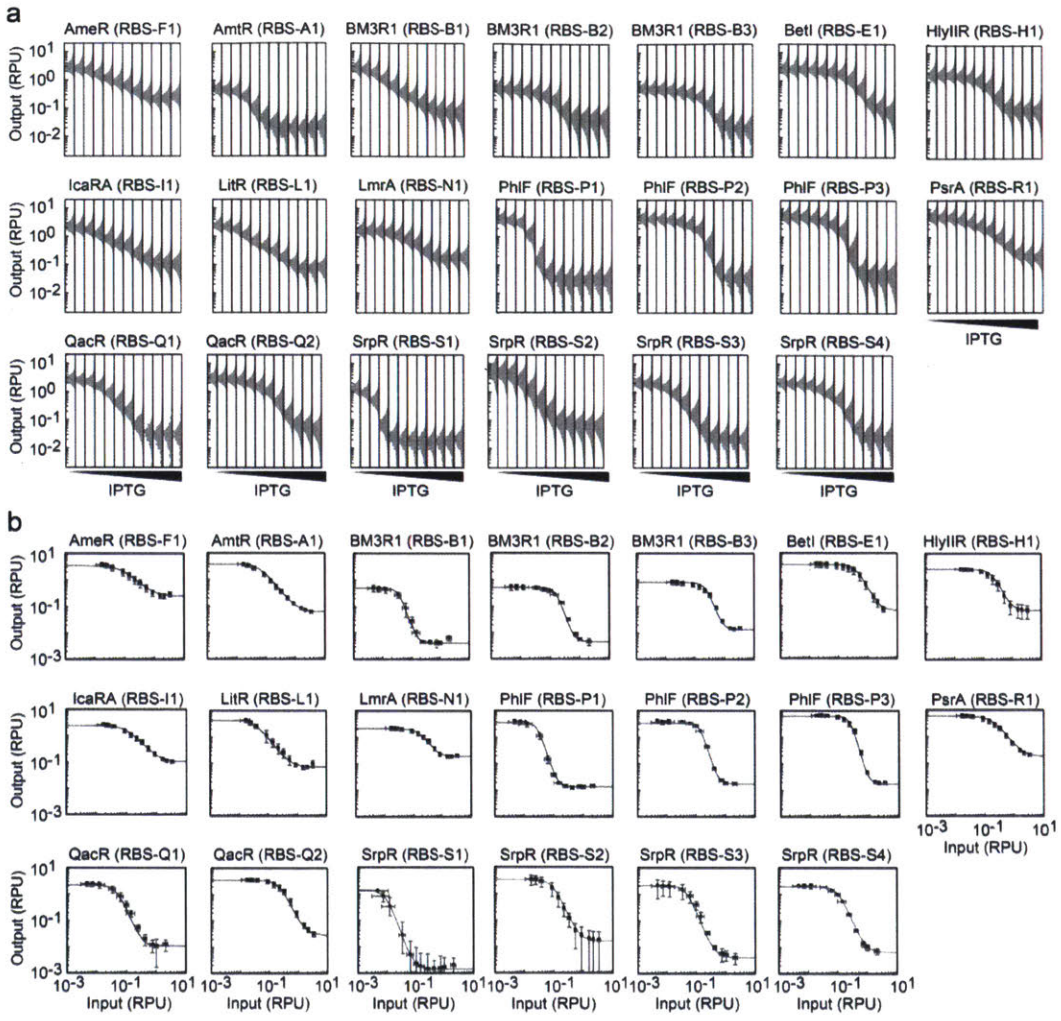

Figure 4-9: Distributions and response functions for insulated gates. (A) Representative YFP fluorescence histograms for each gate are each normalized to RPU. IPTG concentrations used were: 0, 5, 10, 20, 30, 40, 50, 70, 100, 150, 200, and 1000 μM. (B) The response functions are fit to Equation S1 (black lines). Error bars are one standard deviation of the median for three experiments performed on different days. Hill equation parameters are given in Table 4-4.

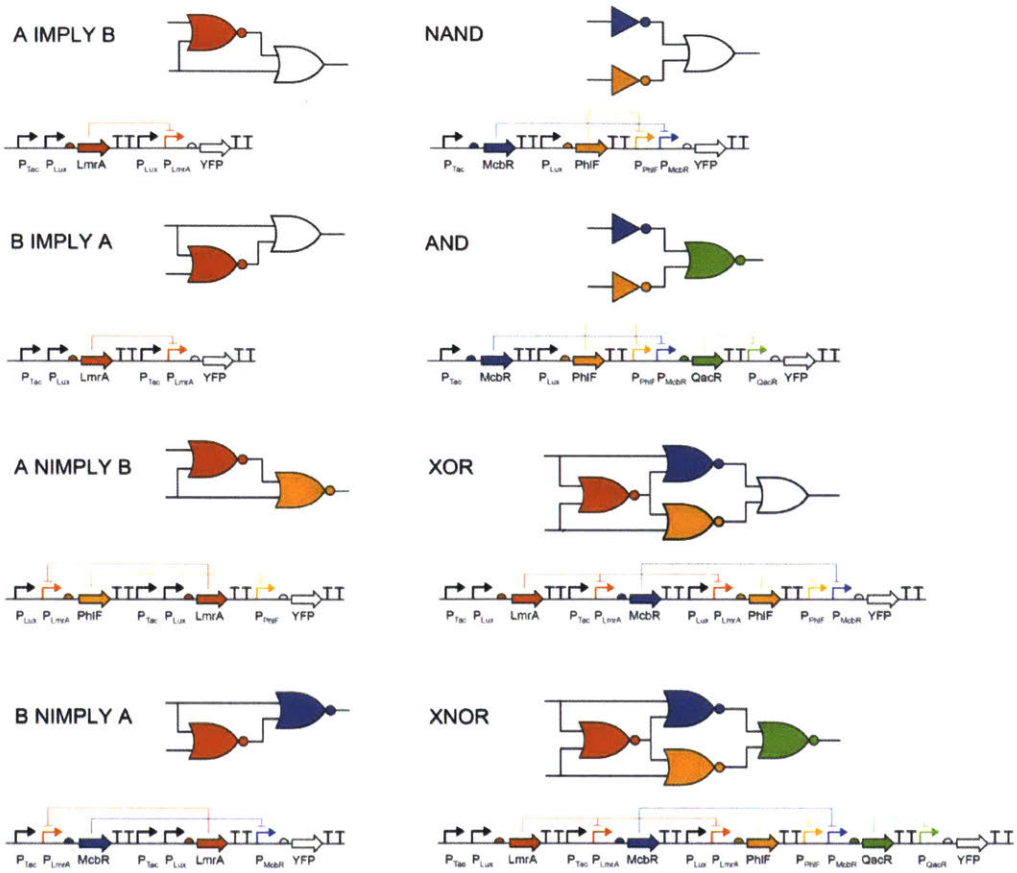

Figure 4-10: Circuit diagrams and genetic schematics for simple circuits built from non-insulated gates. This corresponds to the "non-insulated" data shown in Figure 4-3a. Gate colors correspond to the repressors in the genetic construct. All the terminators are the same (BBa_B0015) and are shown as a black "TT". Plasmids used were pAN901-908.

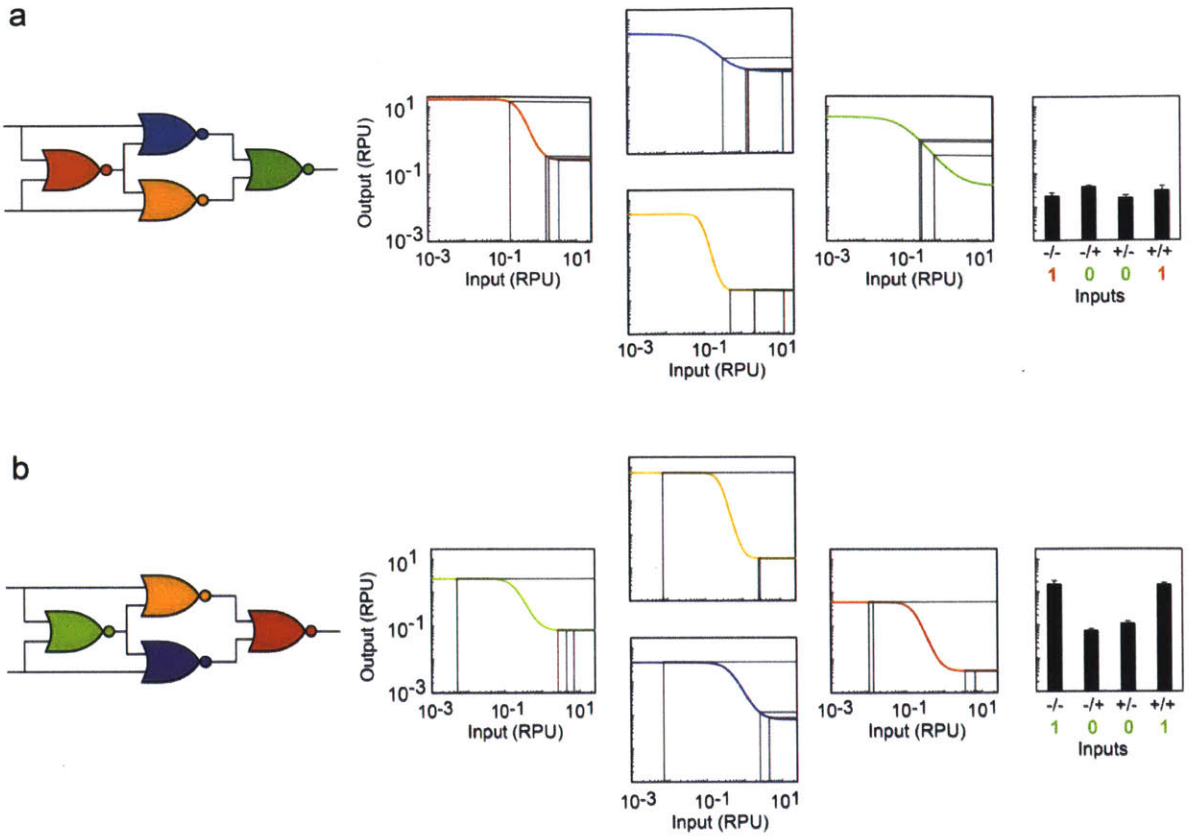

Figure 4-11: Response function matching in circuits. (A) XNOR circuit built from non-insulated gates. Assuming the response functions behave the same in the context of a circuit, the circuit is still predicted to be non-functional because all the output states from the first gate map onto the next gate's response function to the right of the threshold. Experimental data from Figure 4-3a shown at right. Inputs correspond to the absence or presence of 1 mM IPTG (right -/+) and 20 μM 3OC6HSL (left -/+). (B) XNOR circuit built from insulated gates (Figure 4-3a). The repressor assignment algorithms cause the outputs from the first gate to span the threshold of each gate. Experimental data from Figure 4-3a shown at right. Inputs correspond to the absence or presence of 1 mM IPTG (right -/+) and 2 ng/mL aTc (left -/+). For both panels, error bars are one standard deviation of the median for three experiments performed on different days.

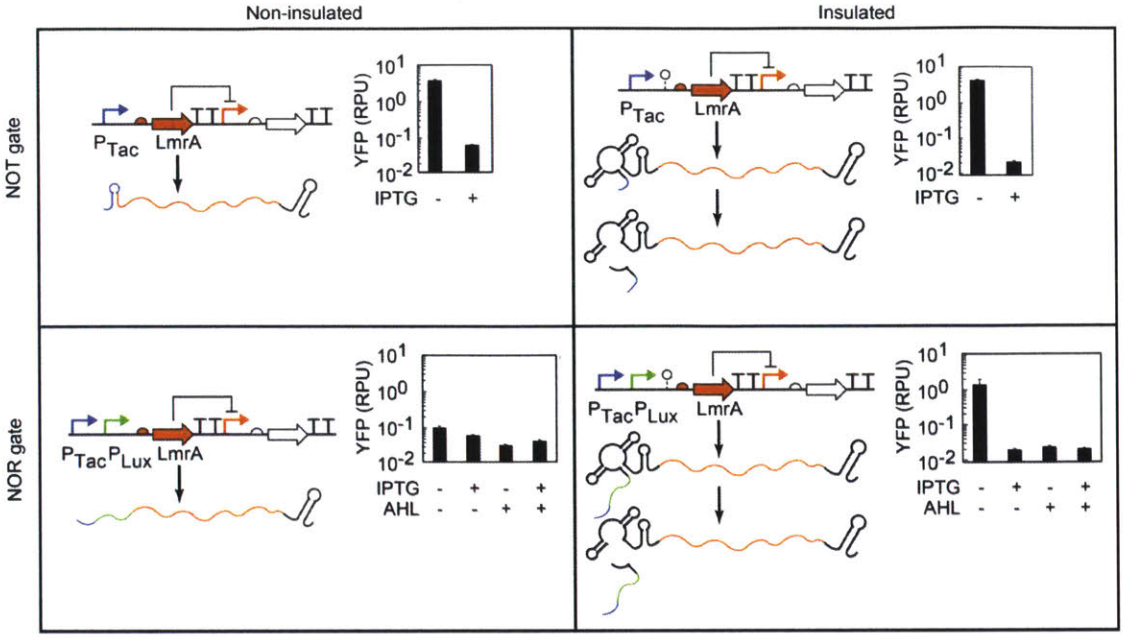

Figure 4-12: Comparison of non-insulated and insulated NOT/NOR gates. NOT and NOR gates without ribozymes contain promoter sequence in the mRNA transcript that can affect translation (left panel). Black bars are expected to be high and gray bars expected to be low. Cells were grown measured with the presence and absence of inducers: 1 mM IPTG and 20 μM 3OC6HSL (Methods). Error bars are one standard deviation of the median for three experiments performed on the same day. The plasmids used are: pAN215 = non-insulated NOT gate, pAN216 = non-insulated NOR gate, pAN412 = insulated NOT gate, and pAN413 = insulated NOR gate.

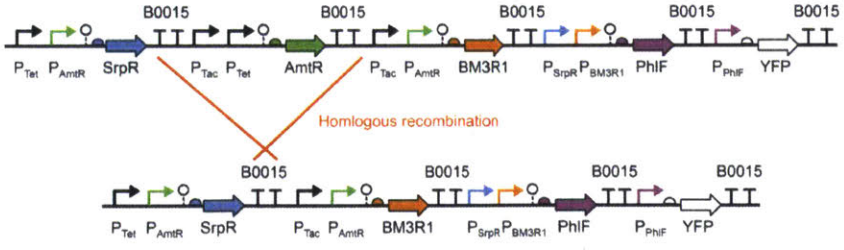

Figure 4-13: Repeated terminators cause high rates of homologous recombination. The top construct is the original design. The bottom construct was identified by sequencing, where the AmtR gate was deleted by recombination.

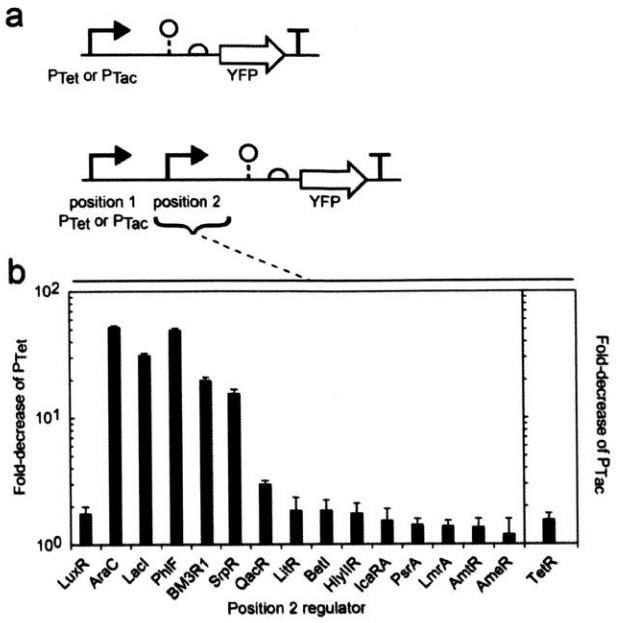

Figure 4-14: Measuring the roadblocking ability of various repressors. The ability of a repressed promoter in position 2 to reduce YFP expression from a promoter in position 1 was tested. (A) Promoter 1 alone (either P_{Tac} or P_{Tet}) was induced to express YFP and was measured by cytometry. Next, a second promoter was inserted downstream from P_{Tac} or P_{Tet}. The upstream promoter was induced, and the downstream promoter was repressed (inactivated in the case of LuxR and AraC). The decrease in YFP expression compared to the P_{Tac}- or P_{Tet}-only case was used to calculate roadblocking. Plasmids used are pAN1250 and pAN1681-pAN1697. (B) The fold-decrease caused by each repressed (or unactivated) promoter when in the downstream position. The upstream promoter is P_{Tet} in all cases, except for when the ability of P_{Tet} to roadblock is being measured, in which case the upstream promoter is P_{Tac}. Error bars are one standard deviation of the median for three experiments performed on different days (Methods).

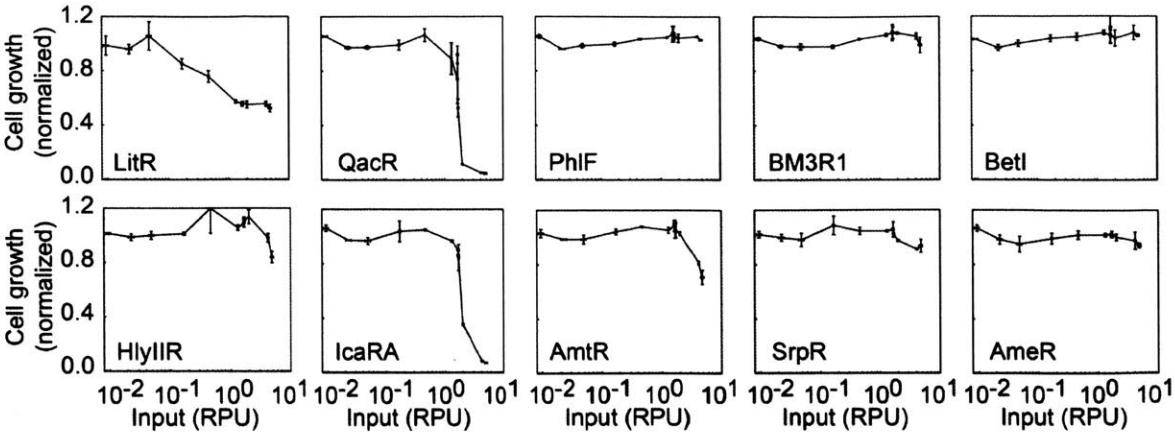

Figure 4-15: Toxic circuits and gate measurements. We induced expression of each repressor from the tandem promoter with seven IPTG concentrations: 0, 9.5, 19, 47.5, 95, 285, and 950 μM; for an additional five samples, we induced with 950 μM IPTG along with aTc at concentrations: 0.0095, 0.095, 0.285, 0.95, and 1.9 ng/mL (Methods). After a period of growth, we measured the cultures' absorbances at 600 nm and normalized the values to the uninduced sample. For x-axis values, YFP was measured from the same tandem promoter at the same inducer concentration and fluorescence was converted to RPU. Error bars are one standard deviation of absorbance (y-axis) and the median (x-axis) for three experiments performed on the same day. Plasmids used were pJS0101-pJS0109.

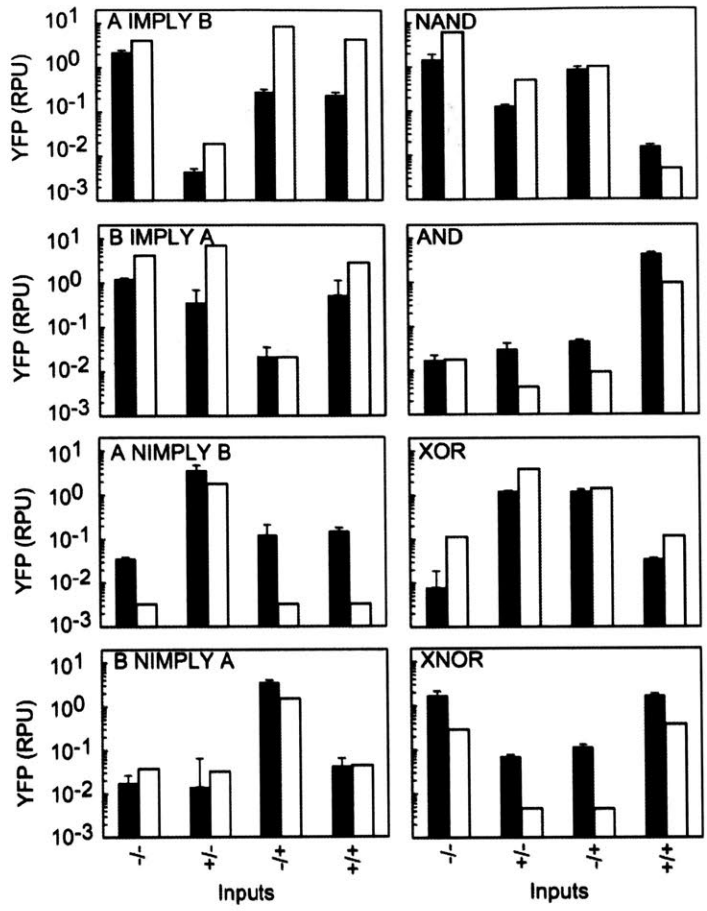

Figure 4-16: Predicted and measured outputs for simple circuits built from insulated gates. Experimentally measured outputs (black bars) for the circuits in Figure 4-3a, alongside predicted outputs (white bars) generated from sensor input levels and gate response functions. Inputs are the absence or presence of 1 mM IPTG (bottom -/+) and 2 ng/mL aTc (top -/+). Error bars are one standard deviation of the median for three experiments performed on different days. Plasmids used were pAN901-pAN908.

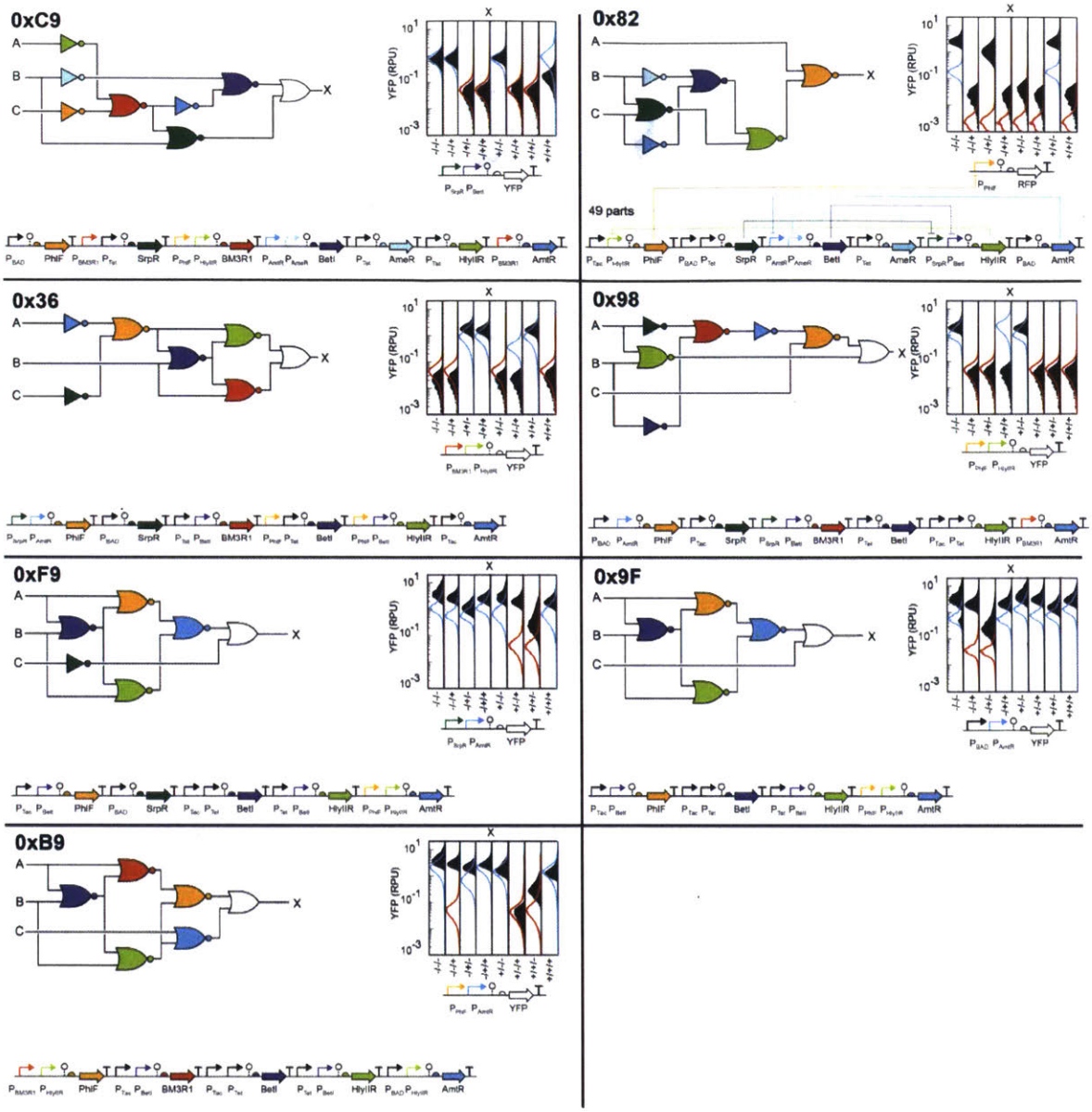

Figure 4-17: Circuits with 1 failed output state. Representative experimentally measured fluorescence histograms (black) and predicted distributions (blue and red lines) are shown for circuits with a single failed output state. Inputs correspond to the absence or presence of 1 mM IPTG (top -/+), 2 ng/mL aTc (middle -/+), and 5 mM L-arabinose (bottom -/+).

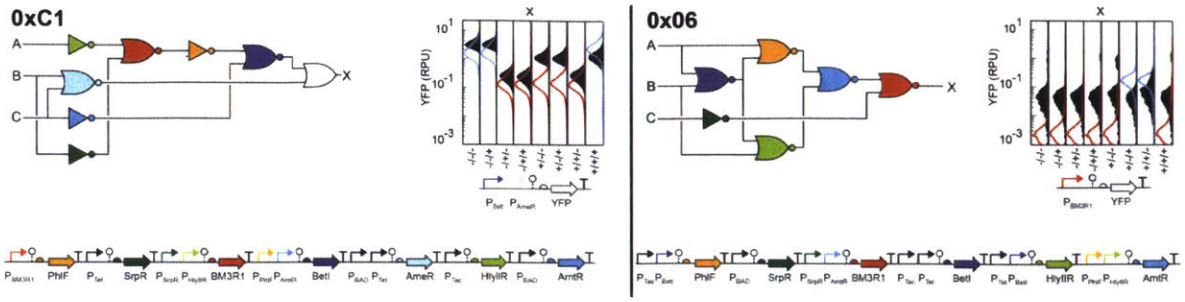

Figure 4-18: Circuits with 2 failed output states. Representative experimentally measured fluorescence histograms (black) and predicted distributions (blue and red lines) are shown for circuits with two failed output states. Inputs correspond to the absence or presence of 1 mM IPTG (top -/+), 2 ng/mL aTc (middle -/+), and 5 mM L-arabinose (bottom -/+).

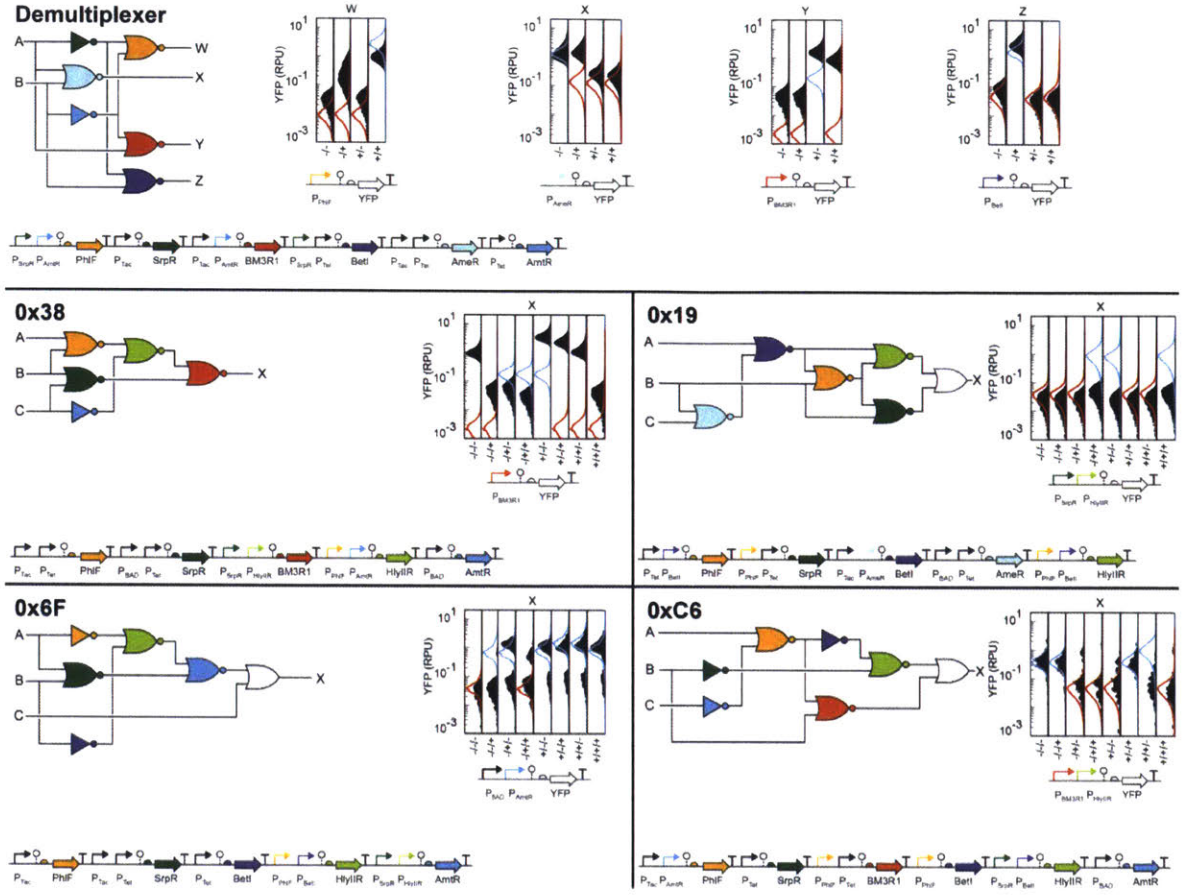

Figure 4-19: Circuits with 3 or more failed output states. Representative experimentally measured fluorescence histograms (black) and predicted distributions (blue and red lines) are shown for circuits with three or more failed output states. For the demultiplexer circuit, inputs correspond to the absence or presence of 1 mM IPTG (top -/+) and 2 ng/mL aTc (bottom -/+). For all other circuits, inputs correspond to the absence or presence of 1 mM IPTG (top -/+), 2 ng/mL aTc (middle -/+), and 5 mM L-arabinose (bottom -/+).

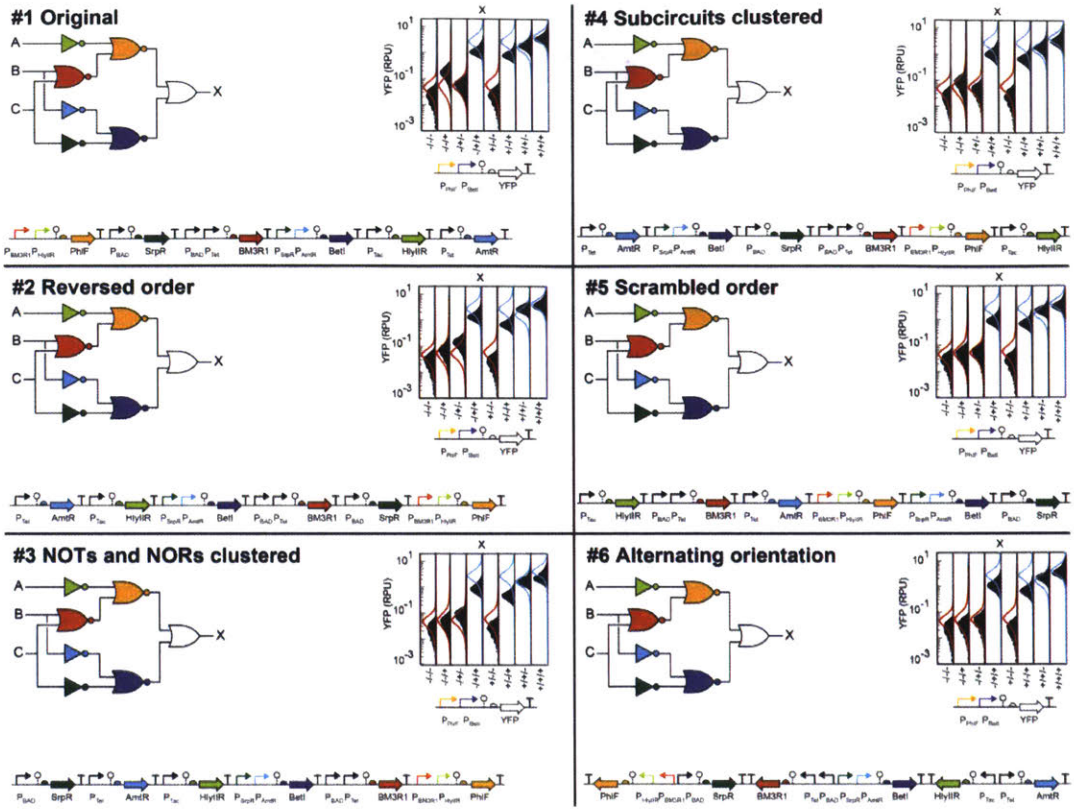

Figure 4-20: Majority circuit variants. Representative experimentally measured fluorescence histograms (RPU, black) and predicted distributions (blue and red lines) are shown for the Majority circuit variants in Figure 4-5.The simplified genetic schematics from Figure 4-5e are shown above the full, labeled schematics. Inputs correspond to the absence or presence of 1 mM IPTG (top -/+), 2 ng/mL aTc (middle -/+), and 5 mM L-arabinose (bottom -/+).

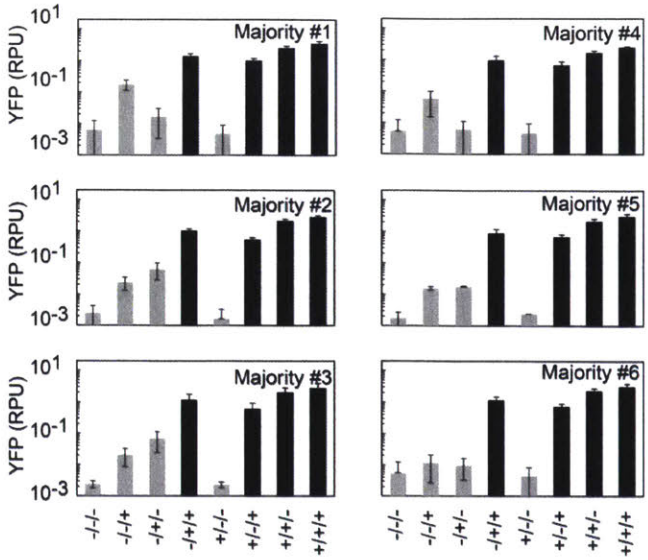

Figure 4-21: Replicates of majority circuit variants. Average output (RPU) for alternate repressor assignments circuits (Figure 4-20). Outputs are predicted to be high (black bars) or low (gray bars). Error bars are one standard deviation of the median for two experiments performed on different days. Inputs correspond to the absence or presence of 1 mM IPTG (top -/+), 2 ng/mL aTc (middle -/+), and 5 mM L-arabinose (bottom -/+).

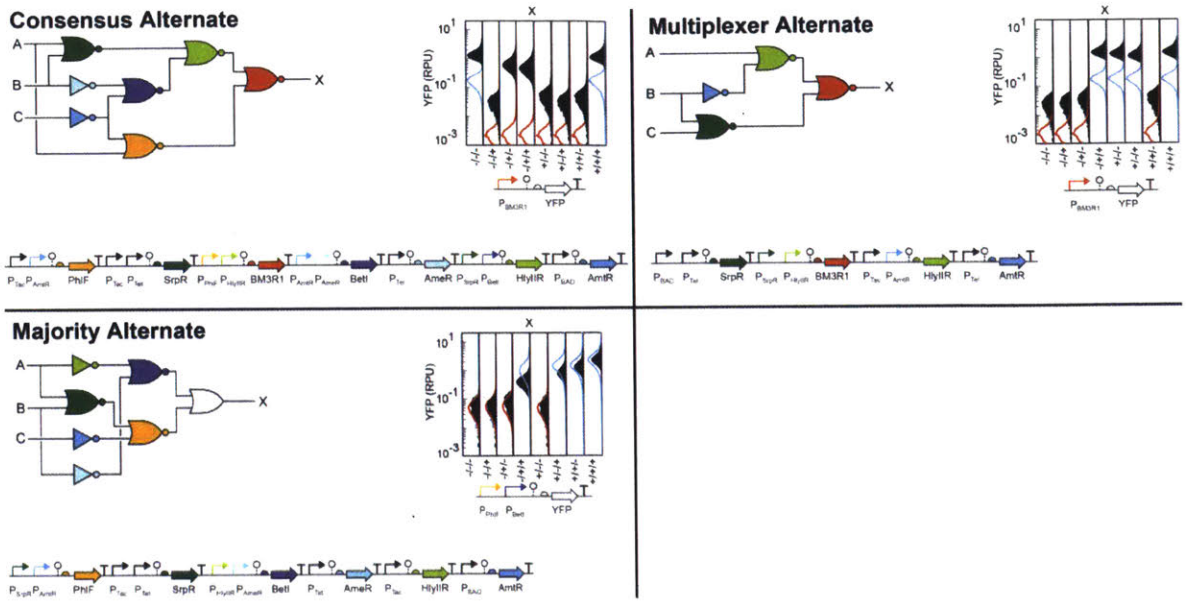

Figure 4-22: Alternate repressor assignments. Representative experimentally measured fluorescence histograms (black) and predicted distributions (blue and red lines) are shown. Inputs correspond to the absence or presence of 1 mM IPTG (top -/+), 2 ng/mL aTc (middle -/+), and 5 mM L-arabinose (bottom -/+).

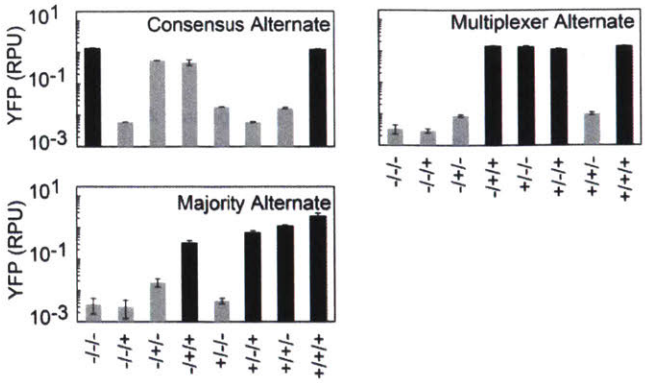

Figure 4-23: Replicates of alternate assignment circuits. Average output (RPU) for alternate repressor assignments circuits (Figure 4-22). Outputs are predicted to be high (black bars) or low (gray bars). Error bars are one standard deviation of the median for two experiments performed on different days. Inputs correspond to the absence or presence of 1mM IPTG (to -/+), 2ng/mL aTc (middle-/+), and 5mM L-arabinose (bottom -/+).

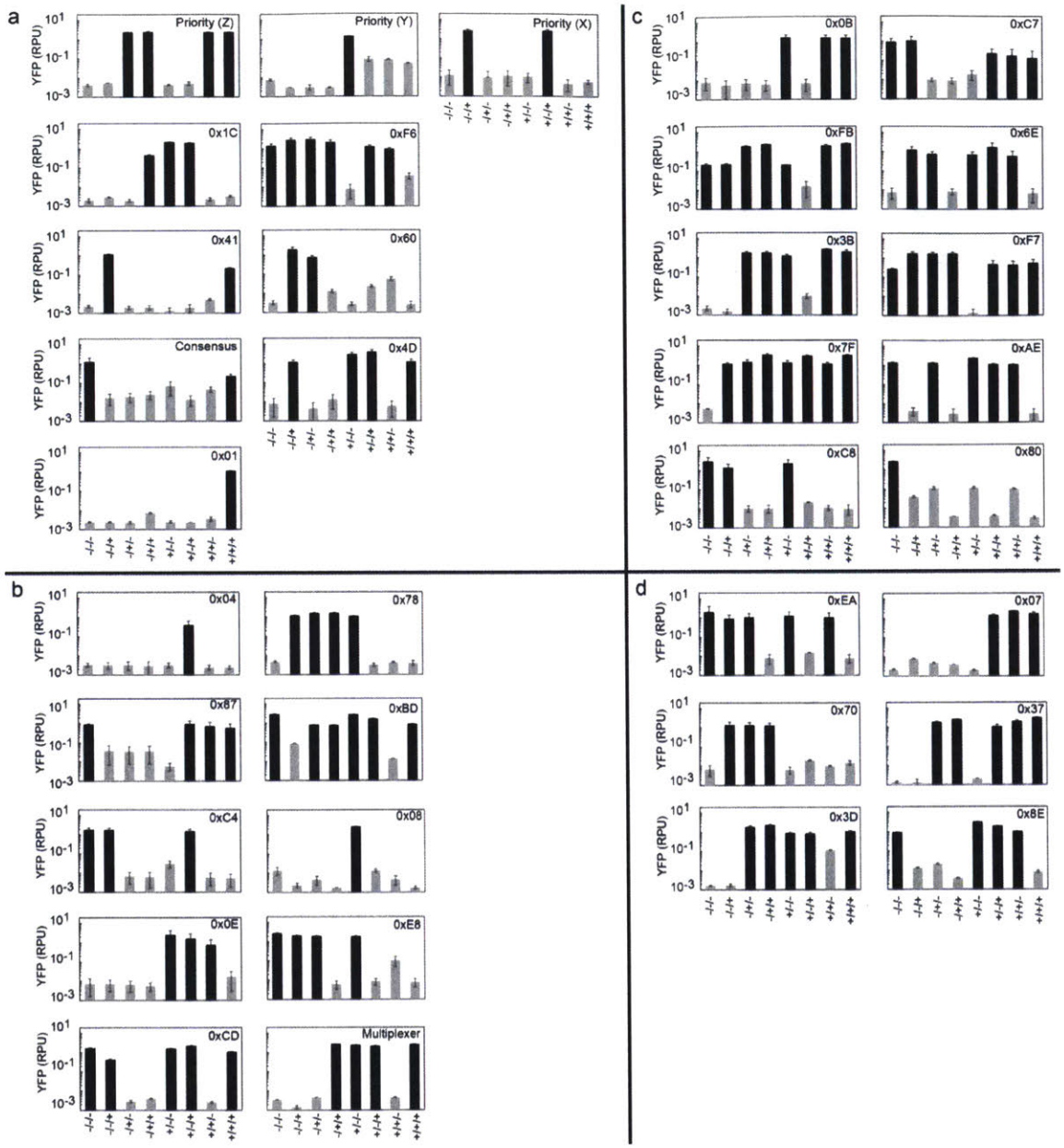

Figure 4-24: Replicates of functional circuits. Average output (RPU) for all functional 3-input circuits from the circuit library (Figure 4-4). The ordering of the circuits in panels (A)-(D) matches the four pages of circuits in Figure 4-4. Outputs (X, Y, and Z) correspond to YFP driven from output promoters in separate experiments, and are predicted to be high (black bars) or low (gray bars). Error bars represent one standard deviation of the median for two experiments performed on different days. Inputs correspond to the absence or presence of 1 mM IPTG (top -/+), 2 ng/mL aTc (middle -/+), and 5 mM L-arabinose (bottom -/+).

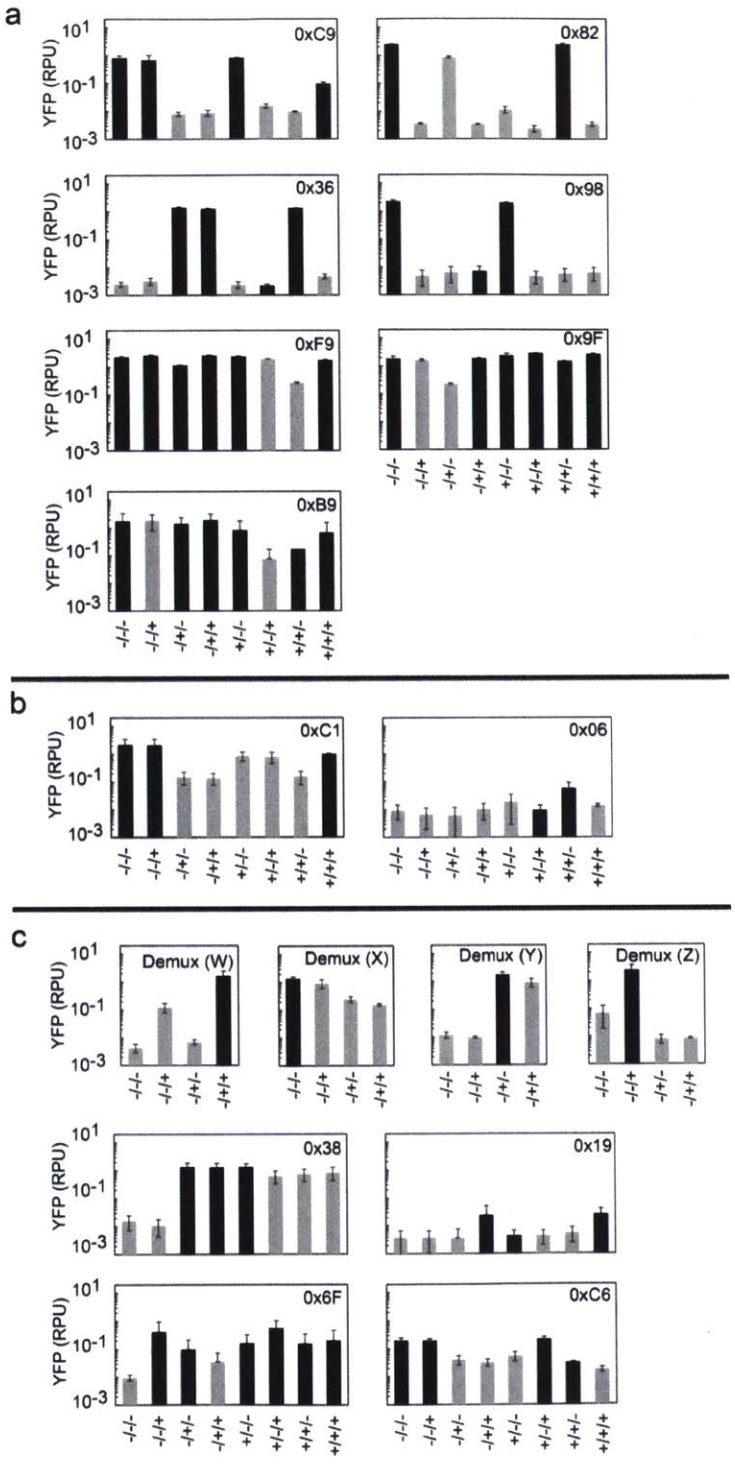

Figure 4-25: Replicates of circuits with failed output states. Average output for circuits with (A) 1 failed output state (Figure 4-17), (B) 2 failed output states (Figure 4-18), or (C) 3 or more failed output states (Figure 4-19). The ordering of the circuits matches the corresponding figures. Outputs (W, X, Y, and Z) correspond to YFP driven from output promoters in separate experiments, and are predicted to be high (black bars) or low (gray bars). Error bars are one standard deviation of the median for two experiments performed on different days. For 3-input circuits, inputs correspond to the absence or presence of 1 mM IPTG (top -/+), 2 ng/mL aTc (middle -/+), and 5 mM L-arabinose (bottom -/+). For the Demultiplexer circuit (Demux), inputs correspond to the absence or presence of 1 mM IPTG (top -/+) and 2 ng/mL aTc (bottom -/+).

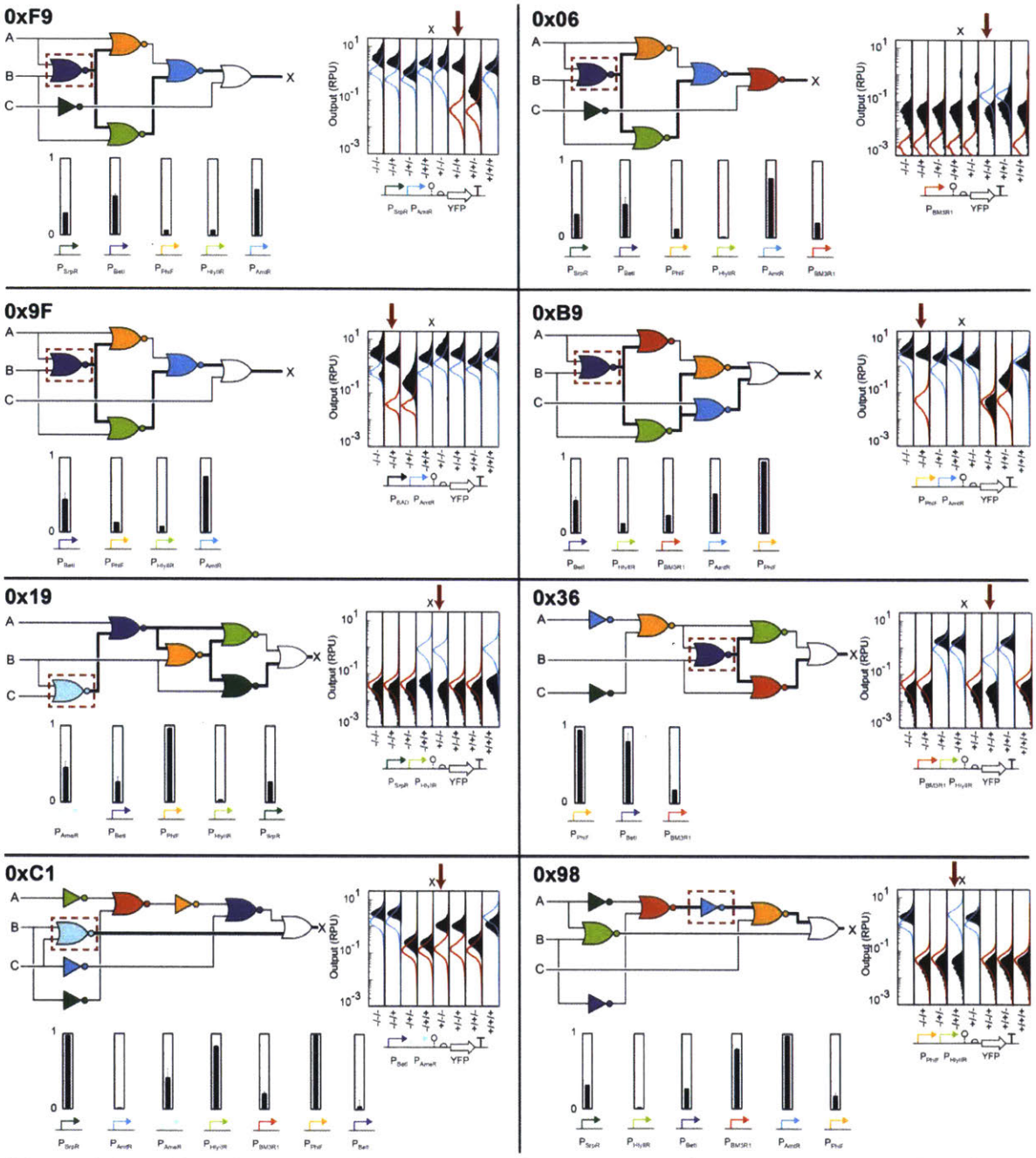

Figure 4-26: Experiments to determine gate failures internal to a circuit. Data are shown for 8 of the non-toxic circuits that show at least one failed state. In each case, an initial screen was performed where the debugging plasmids are substituted for the output plasmid (shown under the cytometry plot) and screened under all eight combinations of inputs. The bar graphs correspond to the activity of each promoter for the failed state under investigation (e.g., +/-/- for 0xC1, red arrow). To account for the dynamic range

differences of the gates, the fluorescence measured is normalized by the minimum and maximum fluorescence observed for the debugging plasmid across all circuits and states. This allows the reporting of the gate activity in the range [0,1]. The dashed red box shows where the error initiates and the thick black line shows how it propagates to the circuit output.

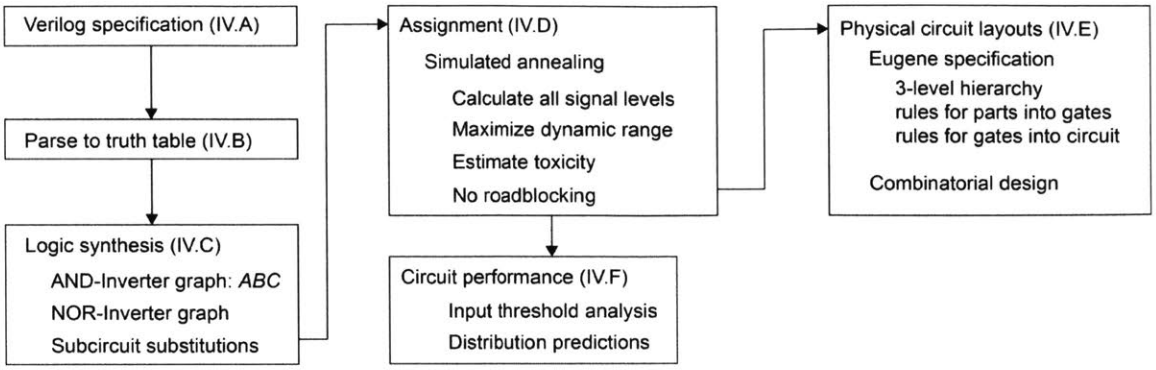

Figure 4-27: Overview of the Cello software. The Cello input is a high-level logic specification written in Verilog, a hardware description language. The code is parsed to generate a truth table, and logic synthesis produces a circuit diagram with the genetically available gate types to implement the truth table. The gates in the circuit are assigned using experimentally characterized genetic gates. In assignment, a predicted circuit score guides a Monte Carlo simulated annealing search. The assignment with the highest score is chosen, and this assignment can be physically implemented in a combinatorial number of different layouts. The Eugene language is used for rule-based constrained combinatorial design of one or more final DNA sequence(s) for the designed circuit.

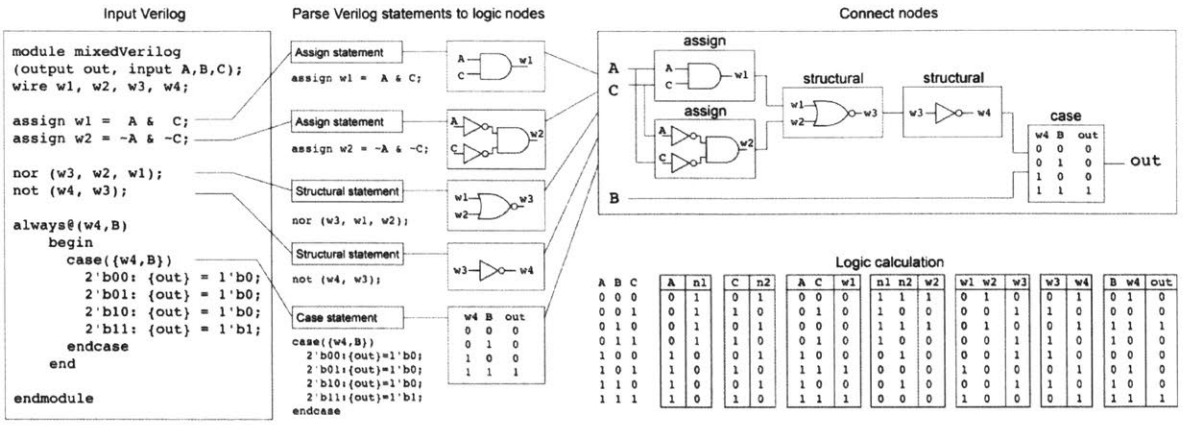

Figure 4-28: Flow of Verilog parsed to a truth table. A Verilog file is parsed into individual assign, structural, and case statements. Each statement is converted into a logic node, which can contain one or more gates, or a truth table. Logic nodes are connected by matching input/output wire names, and Boolean logic is propagated through each node to compute the truth table of the circuit output.

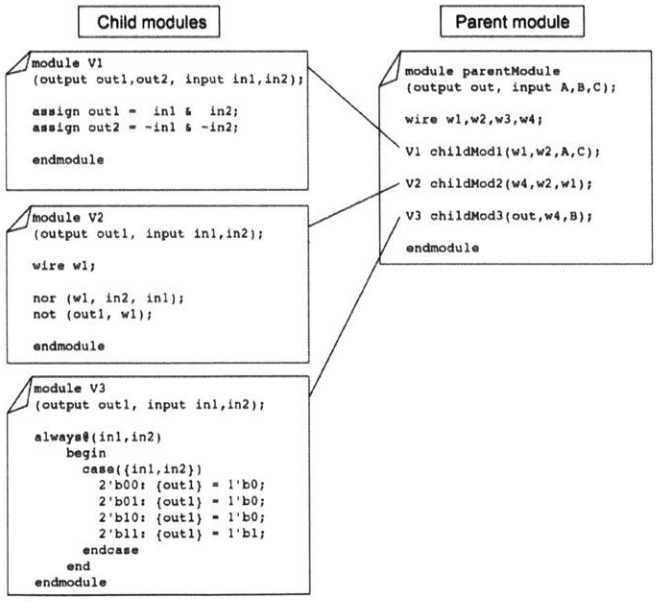

Figure 4-29: Nested Verilog modules for module reuse. The same logic function shown in Figure 4-28 is rewritten using a parent module that references child modules. In the same flow, each statement is converted to logic nodes, which are used to generate the truth table specified by the full program. In Cello, the parent and child modules would appear as a single long file.

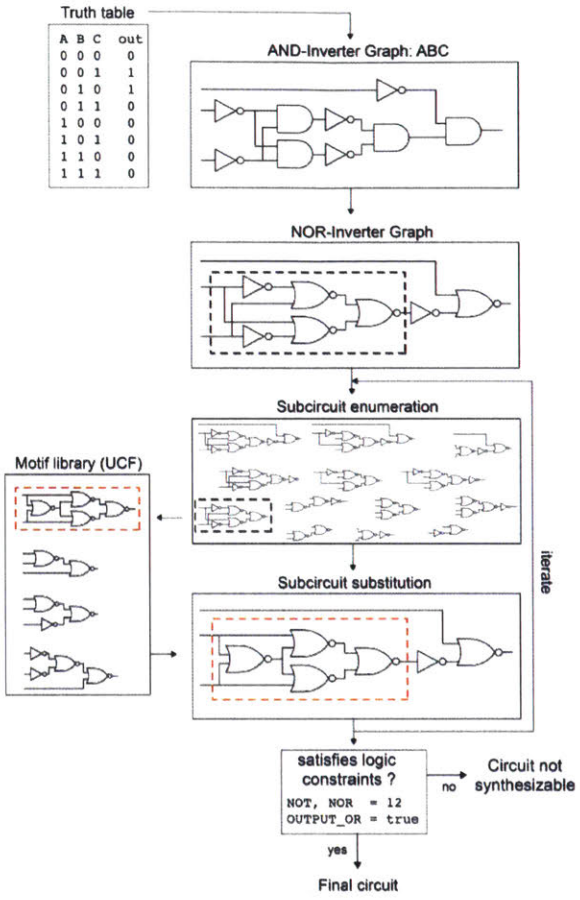

Figure 4-30: Logic synthesis workflow. The starting point is a truth table. The AIG is converted to an NIG using DeMorgan's rule: (A AND B) equals (NOT A) NOR (NOT B), and removing double NOT gates. Subcircuits in the initial circuit diagram can be substituted for user-defined logic motifs specified in the UCF. The black dashed box highlights one subcircuit from the initial circuit, and the red dashed box indicates a functionally equivalent motif from the library, which is substituted into the circuit. This process is done iteratively until no more substitutions are identified. The logic constraints are determined by the gate types available in the UCF, and the number of instances of each gate type in the UCF. In this example, there are a maximum of 12 NOR/NOT gates, and any number of OUTPUT_OR gates.

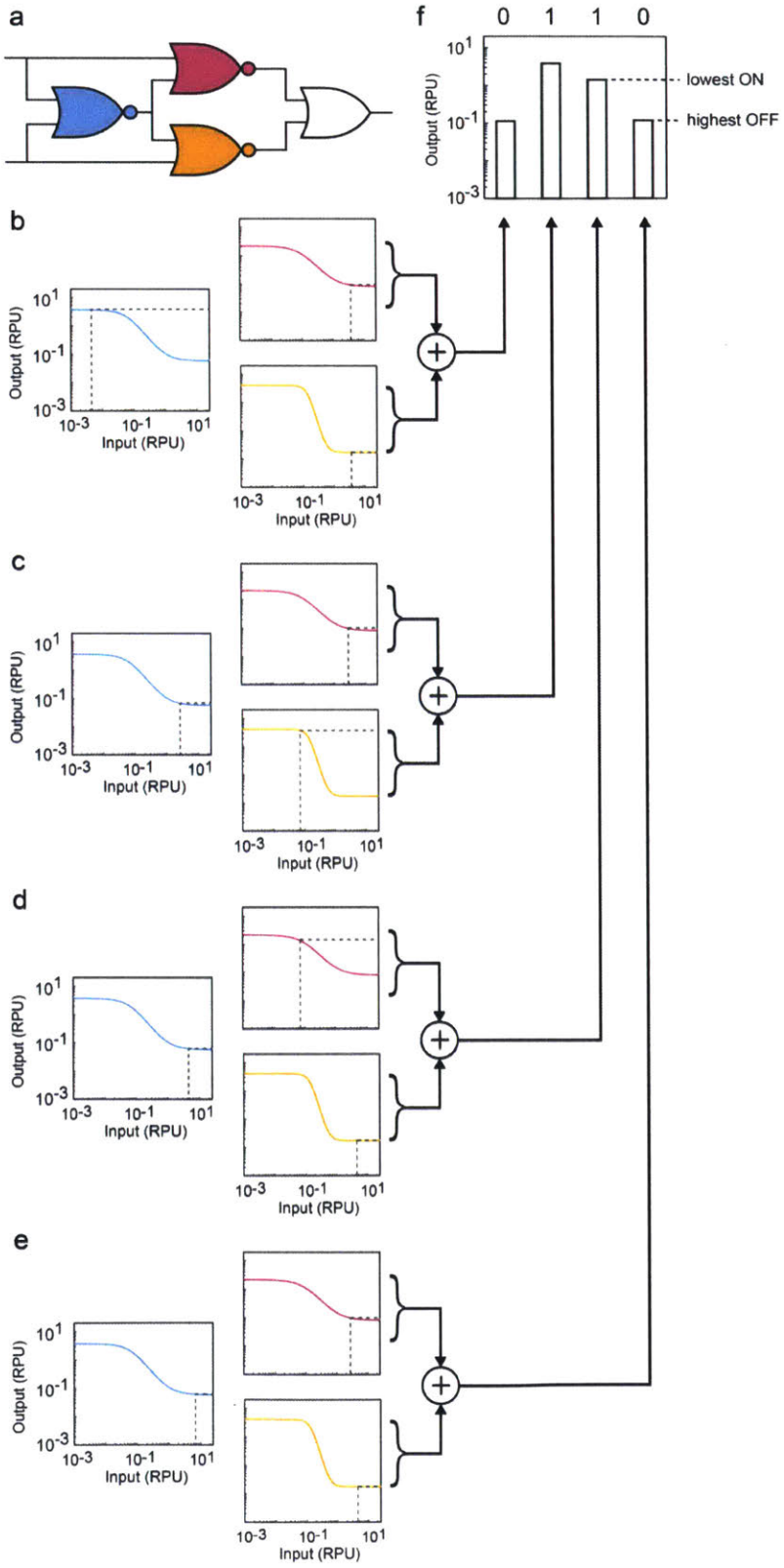

Figure 4-31: Circuit score calculation. (A) Circuit diagram for an XOR circuit with gate assignments AmtR (blue), IcaRA (magenta), and PhlF (orange). (B) Visualization of signal propagation for each of four input states. Colored curves are gate response functions (Equation S1) with the same coloring scheme from (A). Dashed vertical lines represent promoter input levels for the gate. Dashed horizontal lines represent promoter output levels. The "+" symbol indicates promoter outputs from the IcaRA and PhlF gate are summed at the terminal OR gate. (C) Predicted output levels for each of the four input combinations. The 0s and 1s at the top of the graph indicate the desired truth table behavior for each output. The lowest ON state and highest OFF state are marked, and the ratio of these values is the circuit score, S.

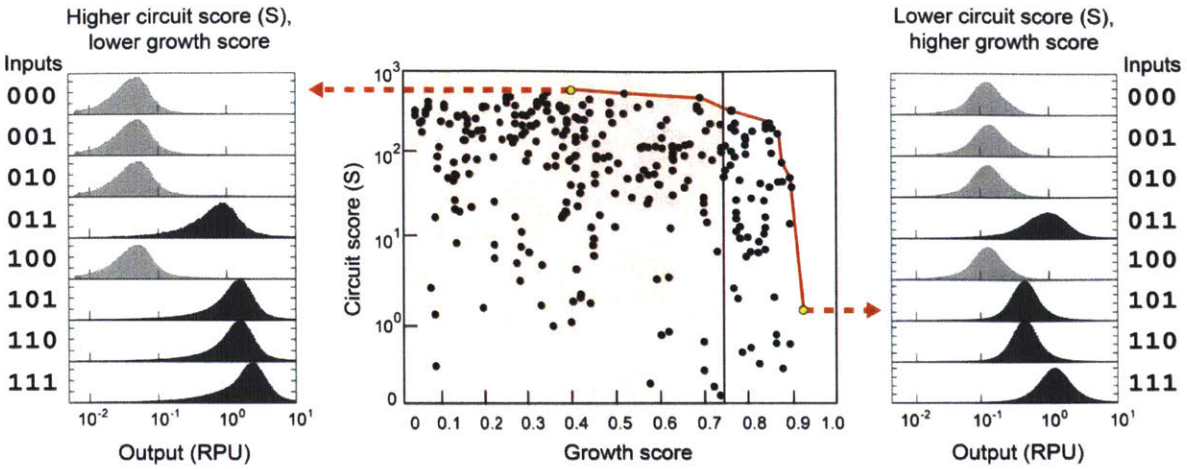

Figure 4-32: Tradeoff between circuit score and predicted cell growth. For the Majority circuit (Figure 4-5), each assignment has a circuit score (S) and a growth score (represented as a point on the scatter plot). The Pareto frontier is shown as a red line. A threshold is defined to eliminate toxic assignments from consideration (shaded region in center plot). Left (assignment highlighted yellow in center plot): Prediction of assignment with high S but toxic expression of IcaRA. Assigned gates: P2-PhlF, H2-HlyIIR, A2-AmtR, B3-BM3R1, I1-IcaRA, S4-SrpR. Right (assignment highlighted yellow in center plot): Prediction of assignment with normal cell growth but low S. Assigned gates: B3-BM3R1, F1-AmeR, S4-SrpR, A2-AmtR, P2-PhlF, H2-HlyIIR.

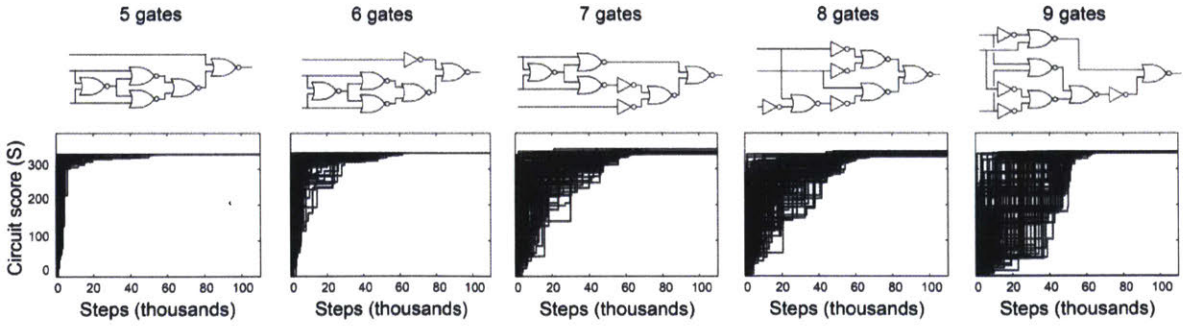

Figure 4-33: Simulated annealing search algorithm for repressor assignment. Each plot shows 100 trajectories, and as the number of steps increases for each trajectory, the highest S up to that point is plotted (black lines). The temperature factor annealed according to the schedule listed above.

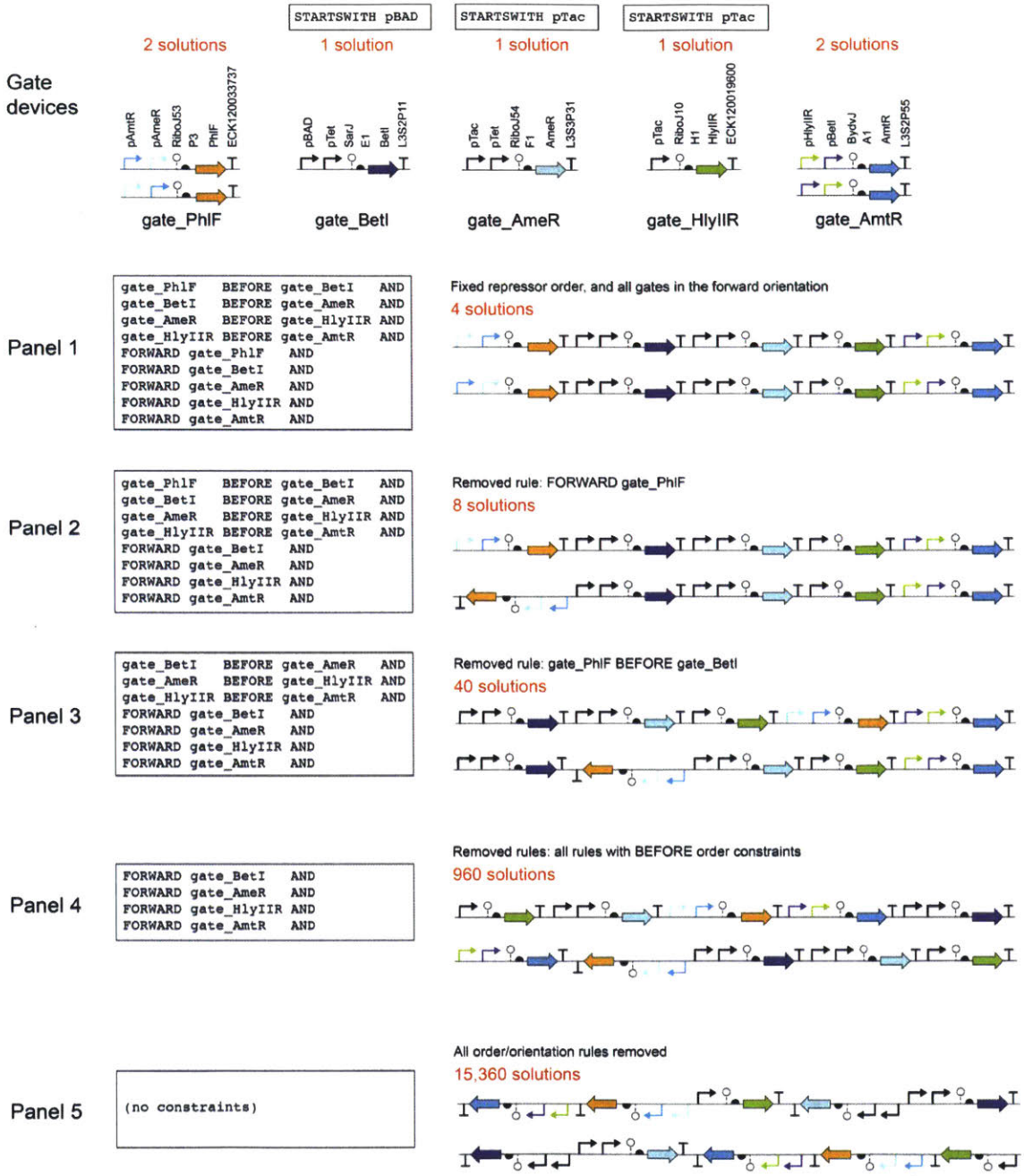

Figure 4-34: Example sets of Eugene rules for the Majority circuit. Parts (Level 1) are used to build gate devices. Promoter order rules are used to disallow roadblocking promoters in the downstream position. The resulting gate devices (Level 2) can be composed in to a circuit device (Level 3). The rules in this figure defined for a circuit device would be specified in "circuit_rules" block in Eugene file, in addition to the EXACTLY 1 gate assignment rules. Depending on the set of rules, the design space for this 5-gate circuit ranges from 4 to 15,360 layouts.

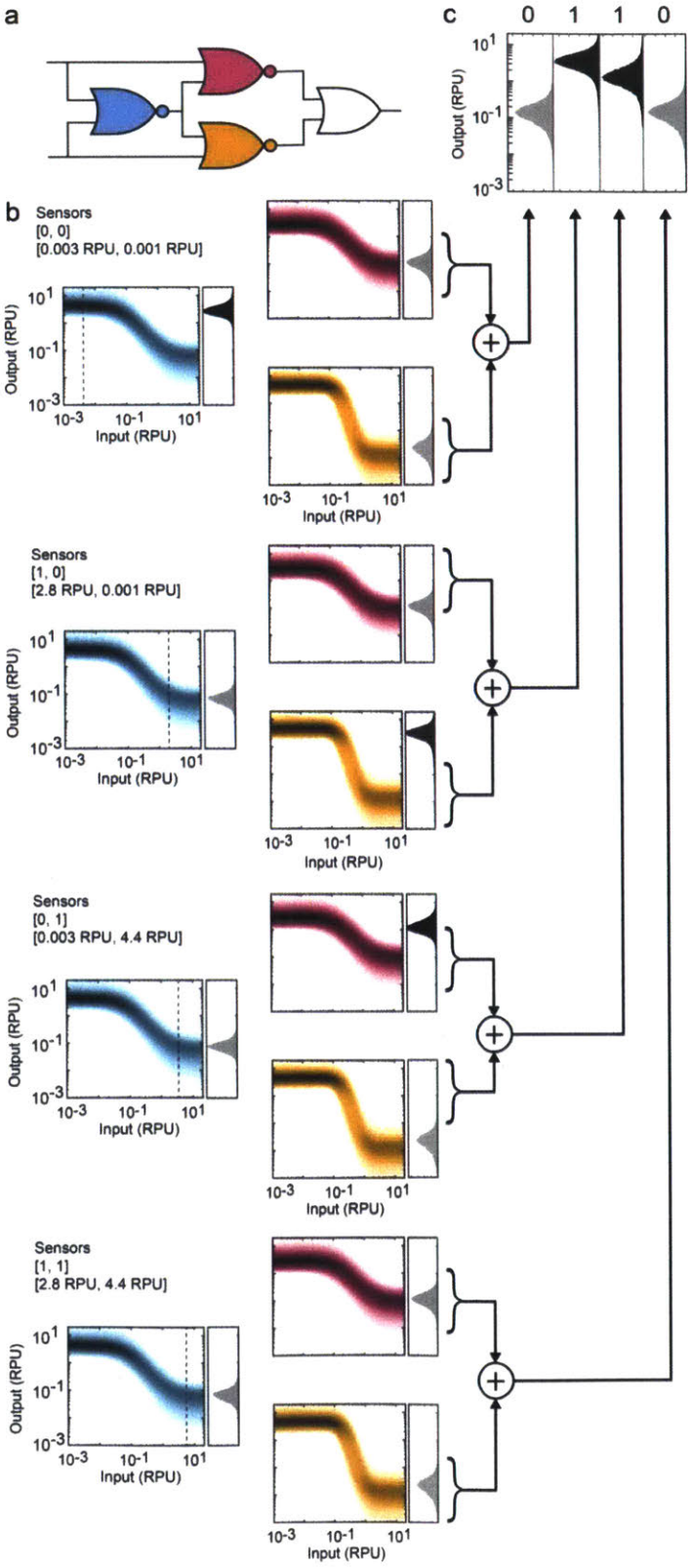

Figure 4-35: Circuit distribution calculation. (A) Circuit diagram for an XOR circuit with gate assignments AmtR (blue), IcaRA (magenta), and PhlF (orange). (B) Visualization of distribution propagation for each of four input combinations. Colored curves are gate distribution response functions with the same coloring scheme from (A). Dashed vertical lines represent sensor input levels for the gate. Vertical histograms to the right of each response function are the output histograms for the gate. The "+" symbol indicates the output histograms from the IcaRA and PhlF gate are summed at the terminal OR gate. (C) Predicted output histograms for each of the four input combinations. The 0s and 1s at the top of the graph indicate the desired truth table behavior for each output. Black and gray histograms are expected to be high and low, respectively.

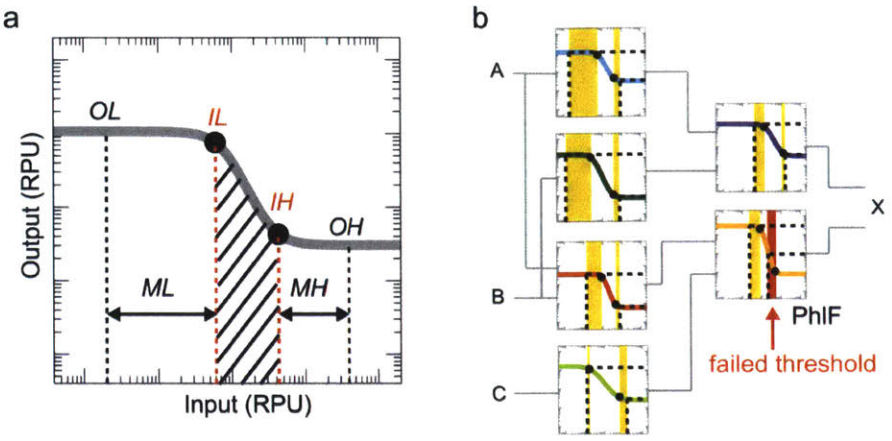

Figure 4-36: Input threshold analysis. (A) A low threshold and high threshold for a gate (IL, IH) are used to determine valid levels for inputs to that gate. In this example, the outputs from the previous gate (OL, OH) result in positive margins (ML, MH, horizontal arrows). A negative margin would indicate an input level in the forbidden zone (diagonal hatching). (B) Input threshold analysis for the Majority circuit (output of Cello). Yellow regions indicate positive margins that pass input threshold criteria. The red margin for PhlF (P3 RBS) indicates a negative input margin that fails the IH threshold criterion.

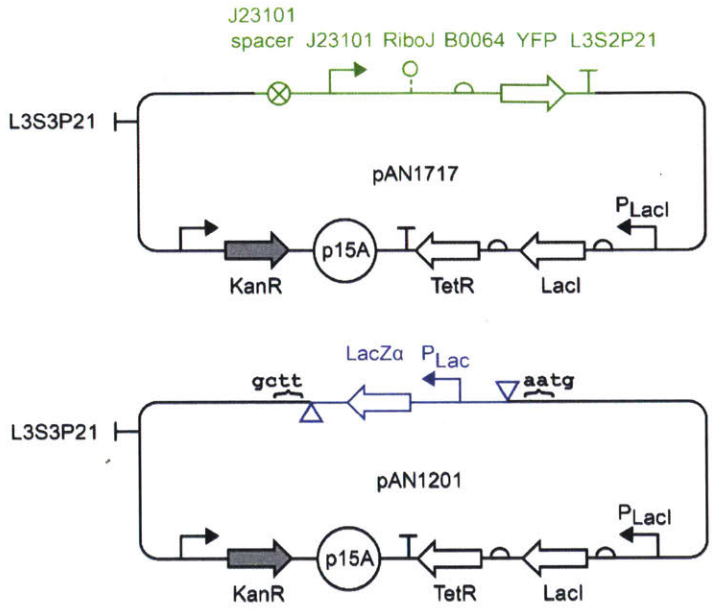

Figure 4-37: RPU standard plasmid and autofluorescence control. Part sequences are provided in Table 4-9. The RPU plasmid (top) promoter is J23101 (Part:BBa J23101 - parts.igem.org, 23101). The RiboJ sequence is the same as published previously (Lou *et al*, 2012), with an additional upstream cloning scar. The RBS is B0064 (Part:BBa B0064 - parts.igem.org). The YFP is as published previously (Cormack *et al*, 1996), with three synonymous mutations: C153A, C564A, and G606T. The YFP terminator is L3S2P21 (Chen *et al*, 2013). There is a 15 bp spacer upstream from J23101, and the upstream terminator L3S3P21 (Chen *et al*, 2013) insulates the YFP cassette from transcriptional readthrough from the plasmid backbone. Cells transformed with pAN1201 (bottom) are used to measure autofluorescence.

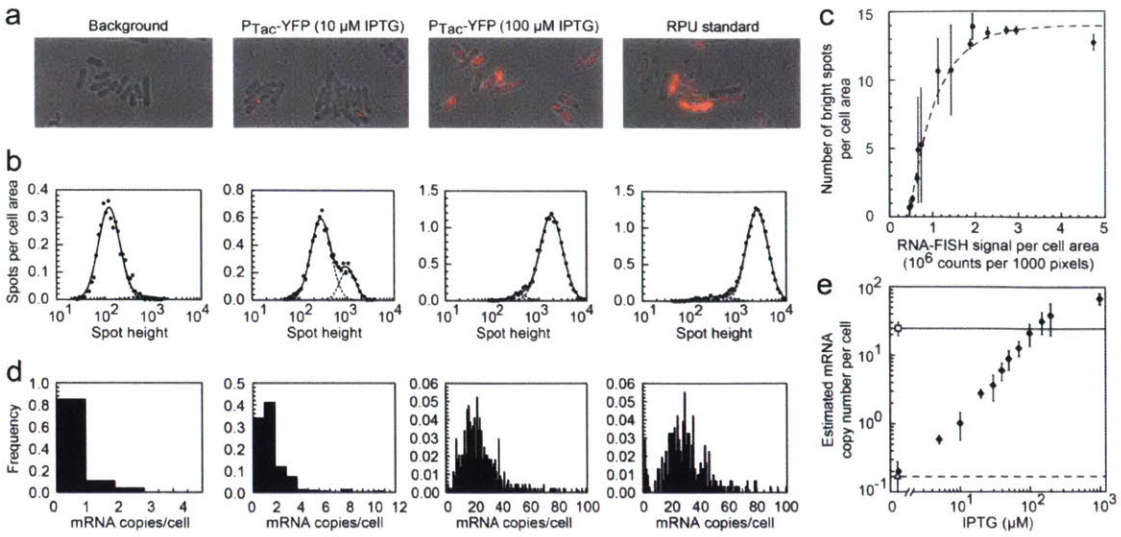

Figure 4-38: smRNA-FISH measurement of the RPU standard. (A) Merged fluorescent and brightfield micrographs of background (pAN1201), P_{Tac}-YFP (10 μM IPTG, pAN1818), P_{Tac}-YFP (100 μM IPTG, pAN1818), and the RPU standard (pAN1717). (B) Histograms of spot heights for the strains shown in (A). Y-axis units are number of spots per 1000 pixels. Histograms were fit to a sum (solid line) of two log-normal distributions (dashed lines). (C) Number of bright spots per cell area vs. FISH signal per cell area for one example replicate experiment. X-axis and y-axis units are both per 1000 pixels. Number of bright spots was estimated from the integrated area of the 2nd (brighter) log-normal distribution fit from (B). The error bars are the standard errors from that estimate. The dashed line is the fit result using a Poisson filling process model (Methods). (D) Histograms of estimated mRNA copy number per cell for the strains shown in (A). Estimates were obtained from the RNA-FISH signal for each cell using the linear extrapolation of the fit curve shown in (C). (E) Estimated mean mRNA copy number per cell vs. IPTG concentration for P_{Tac}-YFP (filled diamonds), background (open triangle), and RPU standard (open square). Plotted values and error bars are the averages and standard deviations of three replicate measurements of the mean mRNA copy number. Dotted and dashed lines represent copy number estimates for background and the RPU standard strains, respectively. The estimated mRNA copy number per cell for background is consistent with zero and the estimated mRNA copy number per cell for the RPU standard is 24.7 ± 5.7.

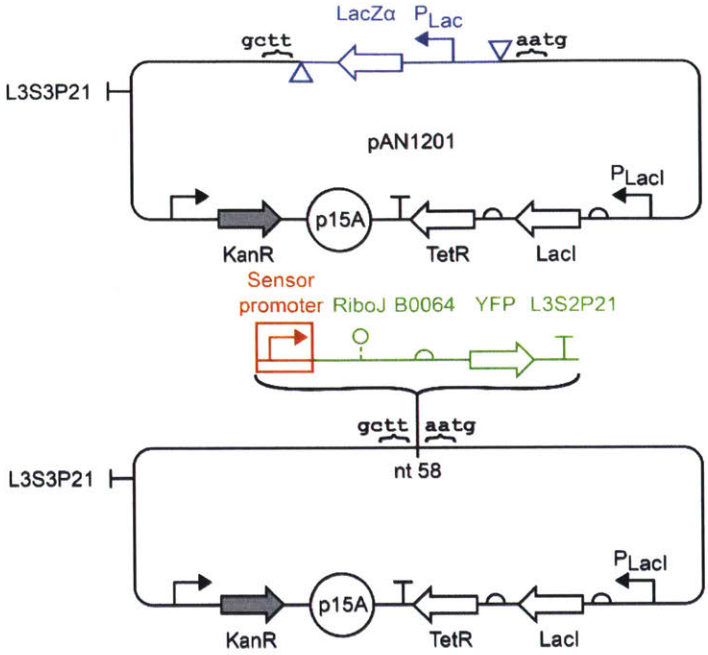

Figure 4-39: Constructing a sensor measurement plasmid. A sensor promoter (red) is positioned in front of the YFP RPU cassette (RiboJ-B0064-YFP-L3S2P21, green) to create a sensor measurement plasmid (bottom). This module is inserted into pAN1201, optionally using BbsI restriction enzyme recognition sites (triangles) and ligation into the insertion scars "gctt" and "aatg". The entire LacZα module (blue) is replaced upon successful insertion.

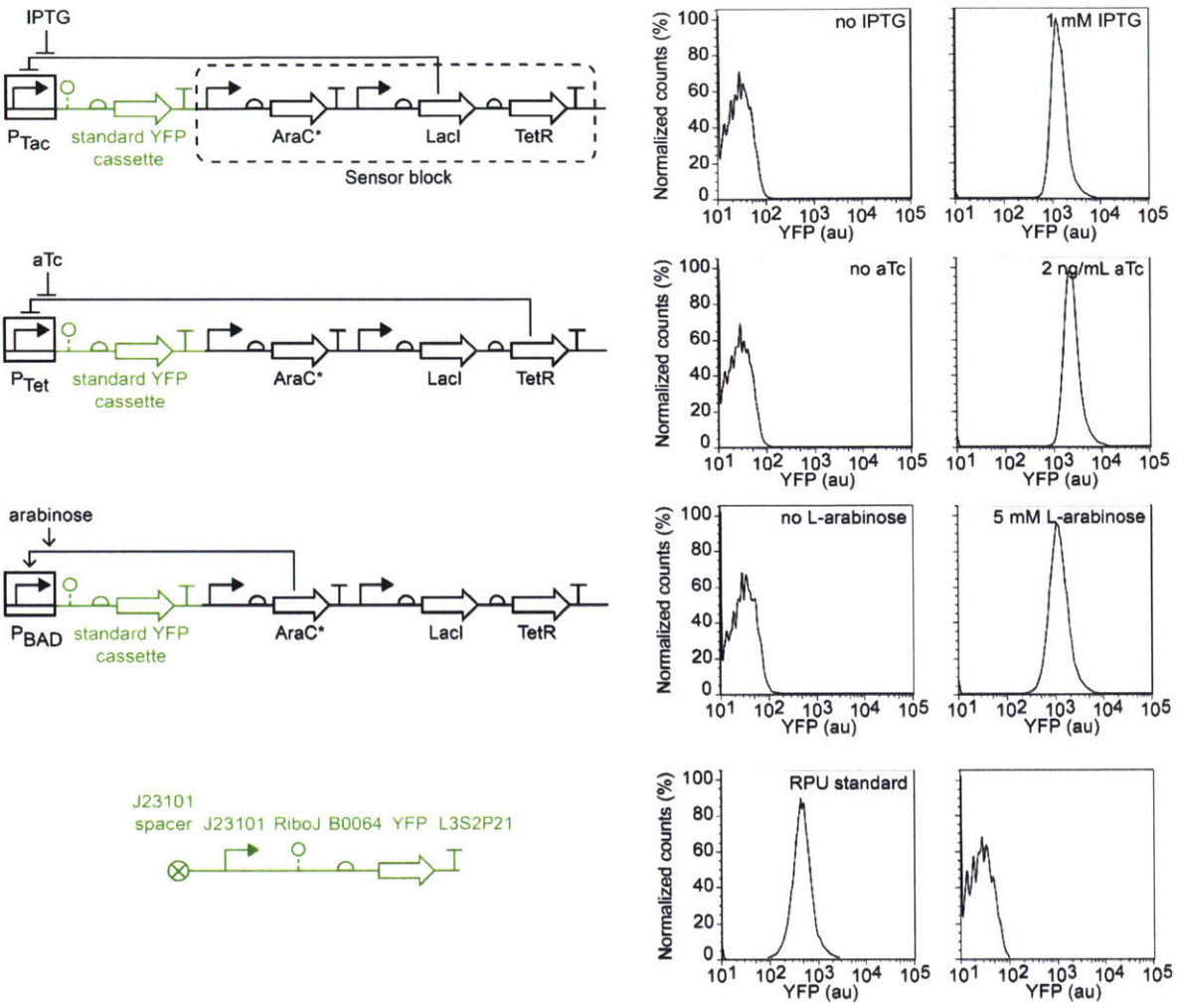

Figure 4-40: Fluorescence data for IPTG-, aTc-, and arabinose-sensors, the RPU standard, and autofluorescence. DNA sequences for these sensor measurement plasmids and RPU standard can be found in Table 4-9 (P_{Tac}: pAN1718, P_{Tet}: pAN1719, P_{BAD}: pAN1720, RPU standard: pAN1717).

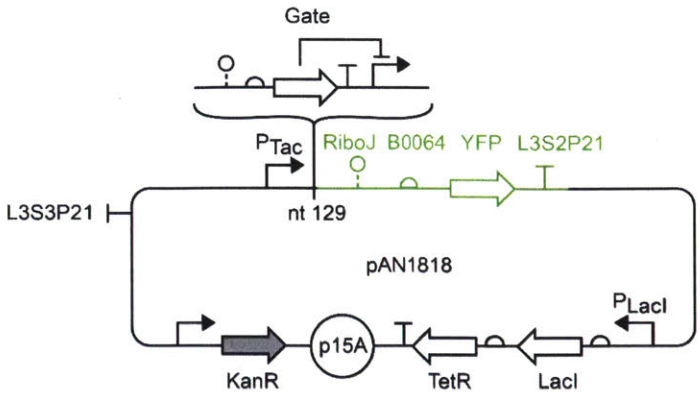

Figure 4-41: Constructing a gate measurement plasmid. A gate comprising a ribozyme insulator, RBS, protein coding sequence, terminator, and output promoter is inserted between P_{Tac} and the YFP expression cassette with constitutively expressed LacI. The gate is inserted at nucleotide position 129 on pAN1818.

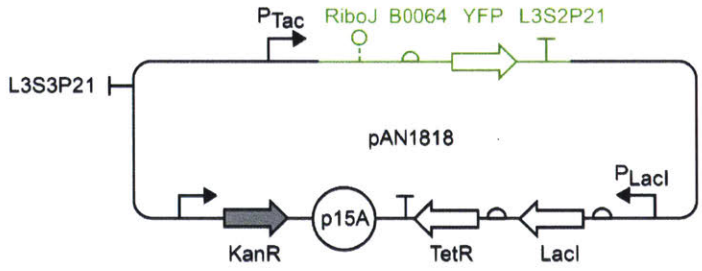

Figure 4-42: Plasmid to characterize the IPTG-inducible P_{Tac} input promoter. The P_{Tac} promoter with symmetric LacO (71 bp, sequence as previous published (Lou *et al*, 2012), with an upstream spacer from -51 to -37) is positioned in front the standard YFP cassette with constitutively expressed LacI (pAN1818). The LacI promoter, RBS, and CDS are the same as in the E. coli genome (Durfee *et al*, 2008), except the "GTG" start codon is replaced with "ATG". The LacI terminator is the genomic AraC terminator (TaraC) from the E. coli genome (Durfee *et al*, 2008).

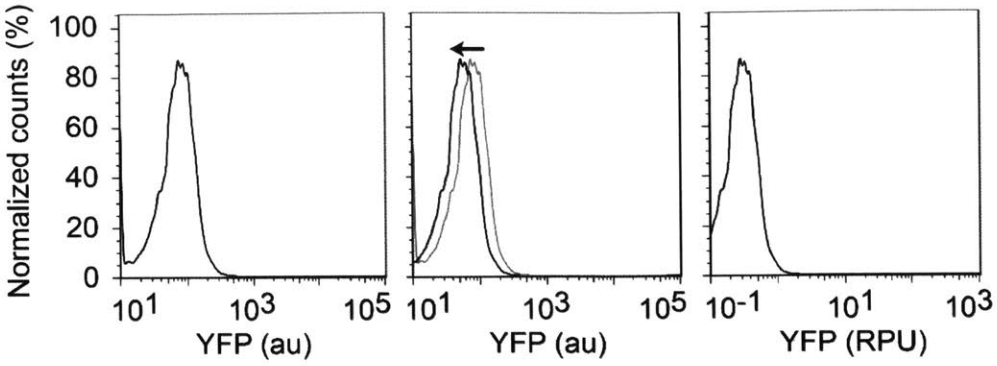

Figure 4-43: Converting a fluorescence histogram to RPU. Beginning with a raw fluorescence histogram (black histogram, left panel; gray histogram, middle panel), shift it left (black arrow) by the autofluorescence median value $<YFP>_0 = 15.0$ au to generate the autofluorescence-corrected histogram (black histogram, middle panel). Next, divide the x-axis units by the corrected median RPU standard fluorescence $<YFP>_{RPU}$— $<YFP>_0 = 460$ au to convert the x-axis to RPU (right panel). The resulting RPU histogram can then be incorporated into a UCF.

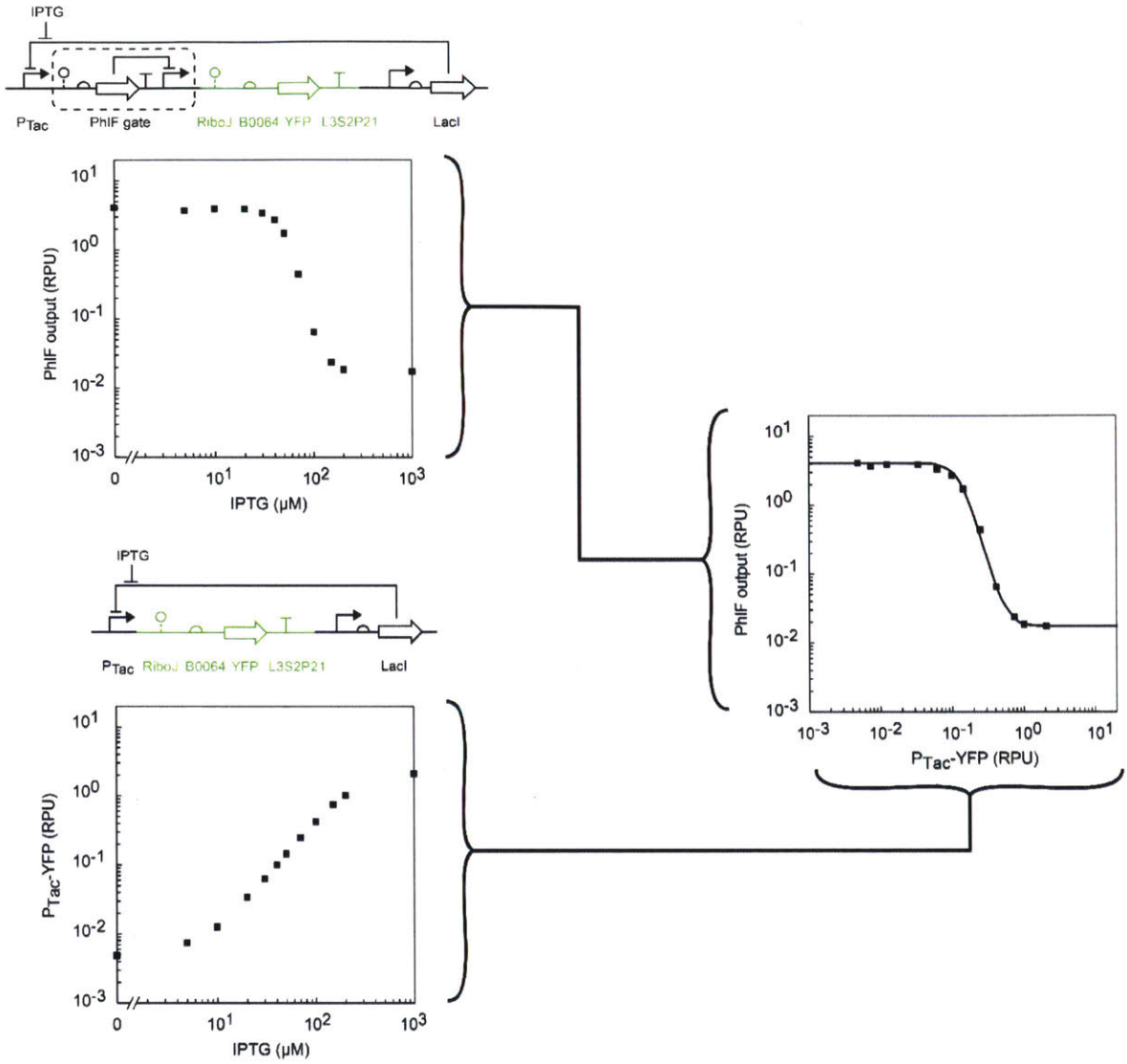

Figure 4-44: Measuring the response function for a gate. RPU measurements at different IPTG concentrations for a gate PhlF(RBS-P2) (upper left) and the input promoter P_{Tac}-YFP (lower left) are plotted against each other to create the response function (right). A Hill equation is fit to the response curve (Equation S1, solid line). IPTG concentrations used were: 0, 5, 10, 20, 30, 40, 50, 70, 100, 150, 200, and 1000 μM.

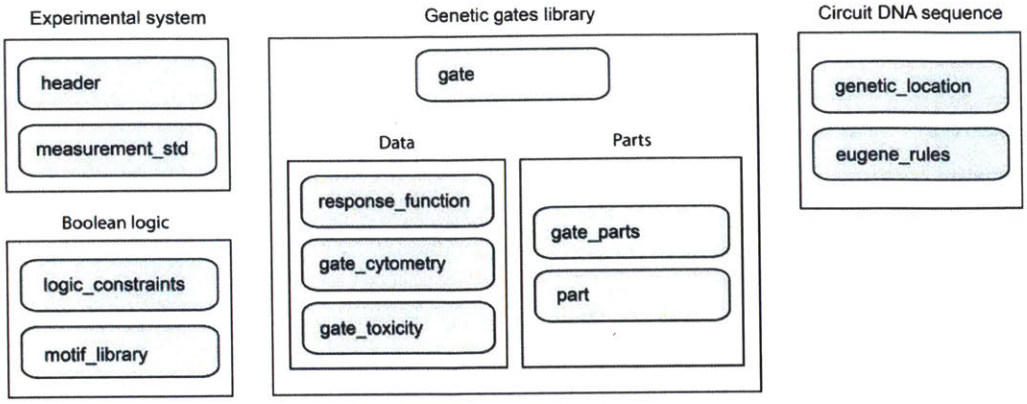

Figure 4-45: Overview of the Eco1C1G1T1 User Constraint File. All data objects in the UCF are organized by collection name (shaded gray).

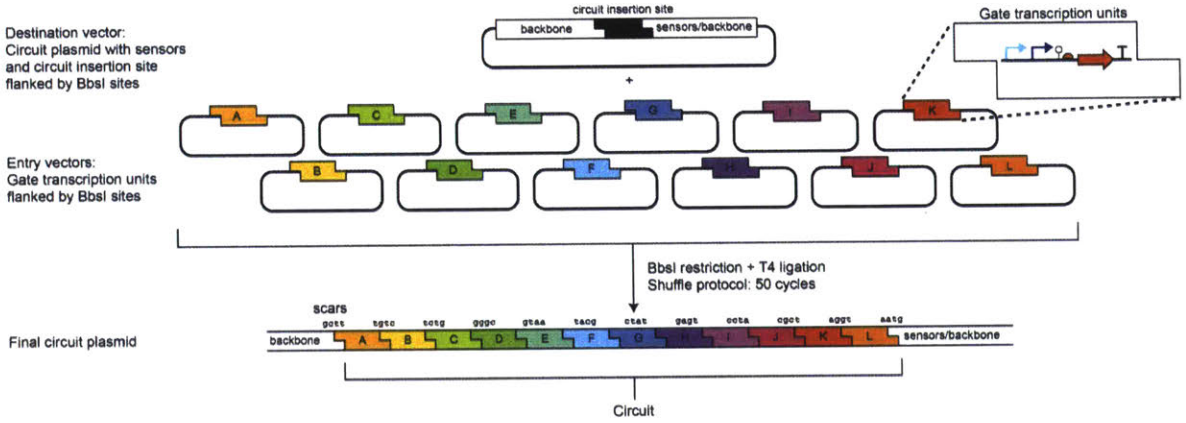

Figure 4-46: Genetic circuit assembly. Final circuit plasmids are constructed from one or more submodule plasmids inserted into the circuit backbone containing the sensor effectors. Submodule plasmids comprise transcriptional units containing different genes (colored shapes correspond to genes). Each overhang is a 4bp sticky end scar generated in the Golden Gate assembly that connects the submodules together. The assembled circuit replaces a P_{Lac}-lacZα cassette (black shape) between the insertion scars "gctt" and "aatg".

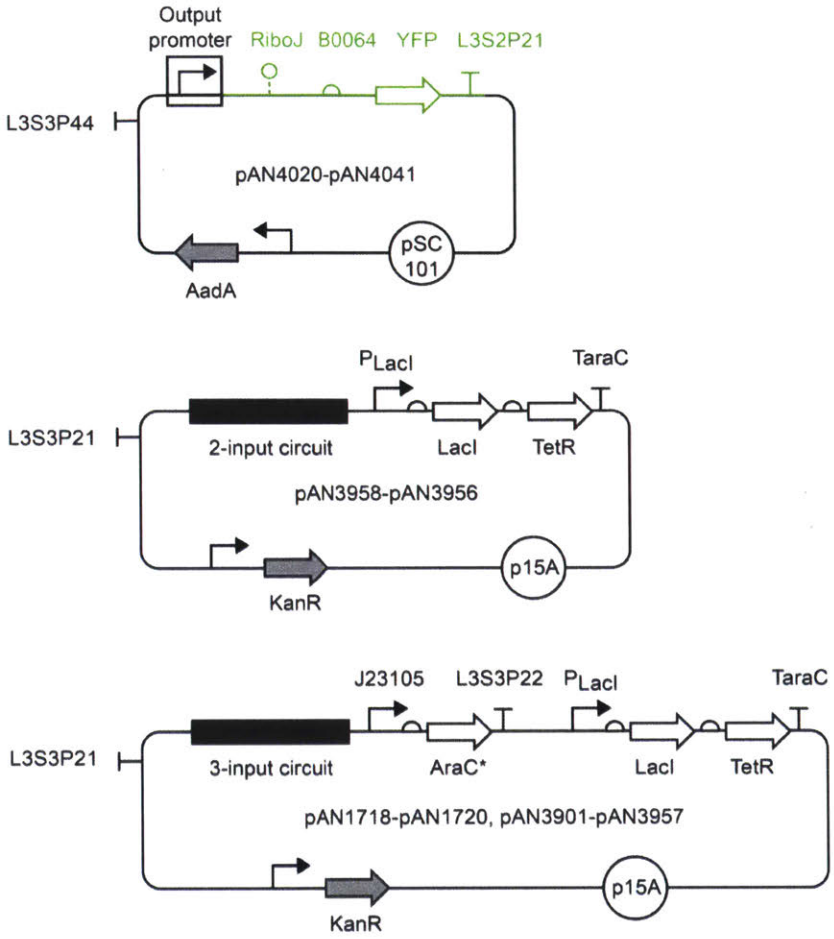

Figure 4-47: Plasmid backbones for measuring circuits. Top: output plasmid. Middle: 2-input circuit backbone. Bottom 3-input circuit backbone.

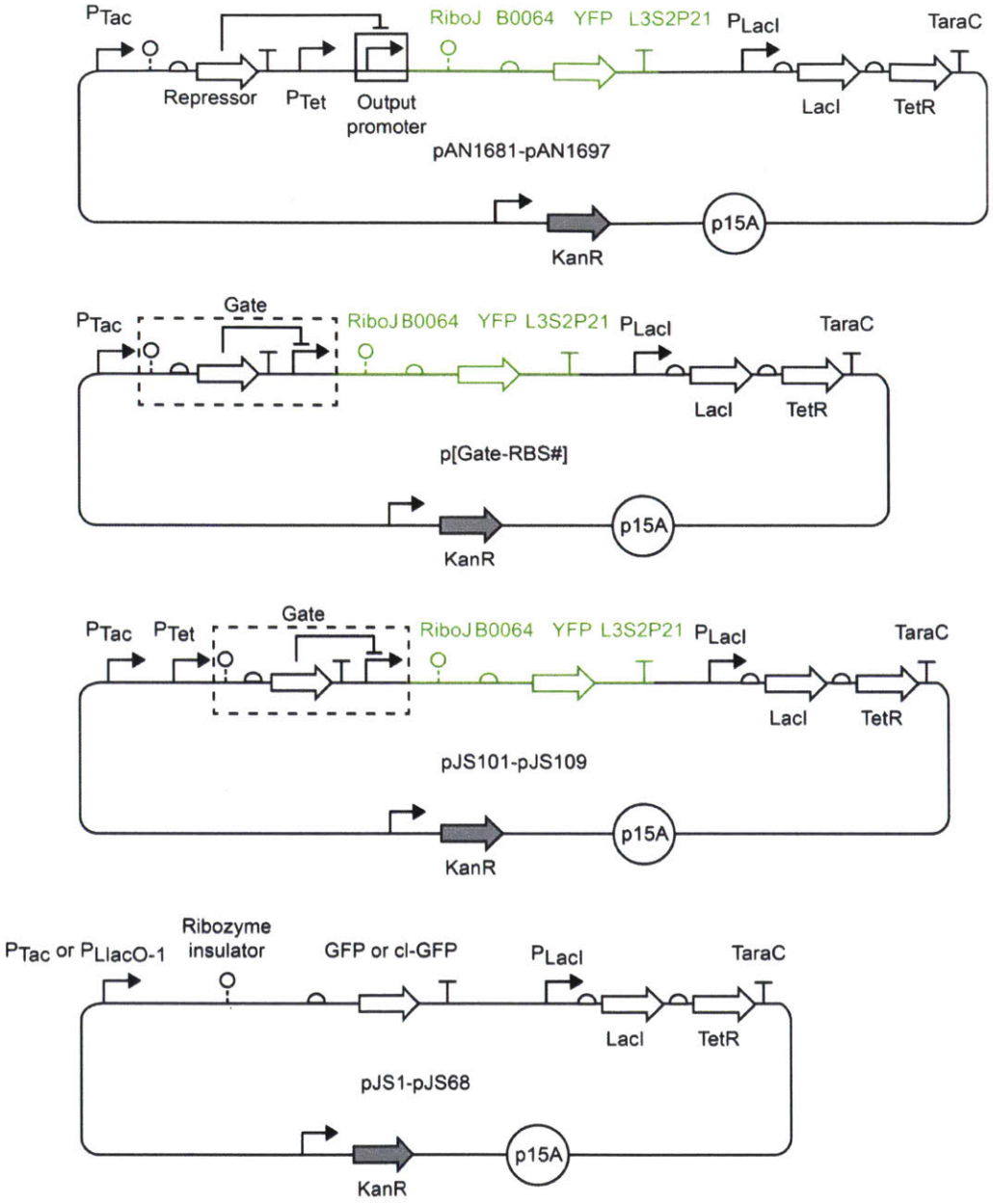

Figure 4-48: Gate characterization plasmids. Top: roadblocking test plasmids. Second from top: Insulated gate measurement plasmids. Third from top: toxicity measurement plasmids. Bottom: Ribozyme test plasmids.

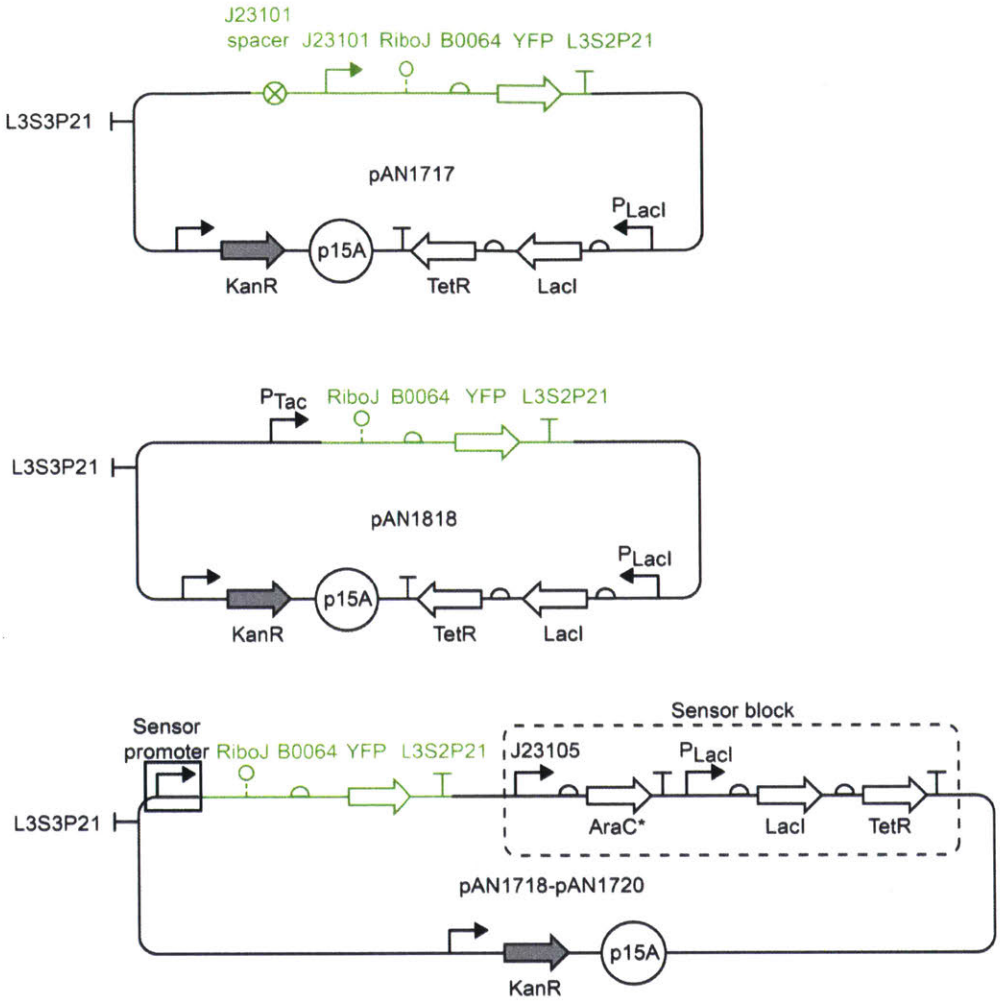

Figure 4-49: Sensor measurement and RPU plasmids. Top: RPU standard plasmid. Middle: P_{Tac} activity plasmid. Bottom: Sensor measurement plasmids.

Table 4-1: Ribozyme sequences

Name	Sequence[a]
RiboJ (Lou *et al*, 2012; Khvorova *et al*, 2003)	AGCTGTC|ACCGGATGTGCTTTCCGGTCTGATGAGTCCGTGAGGACGAAACAGCCTCTACAAATAATTTTGTTTAA
Natural	
AraJ (Dufour *et al*, 2009)	AGTGGTC|GTGATCTGAAACTCGATCACCTGATGAGCTCAAGGCAGAGCGAAACCACCTCTACAAATAATTTTGTTTAA
BydvJ (Khvorova *et al*, 2003)	AGGGTGTC|TCAAGGTGCGTACCTTGACTGATGAGTCCGAAAGGACGAAACACCCCTCTACAAATAATTTTGTTTAA
CchJ (Khvorova *et al*, 2003)	AGTTCCAGTC|GAGACCTGAAGTGGGTTTCCTGATGAGGCTGTGGAGAGAGCGAAAGCTTTACTCCCGCACAAGCCGAAACT GGAACCTCTACAAATAATTTTGTTTAA
ElvJ (Dufour *et al*, 2009)	AGCCCCATA|GGGTGGTGTGTACCACCCCTGATGAGTCCAAAAGGACGAAATGGGGCCTCTACAAATAATTTTGTTTAA
LtsvJ (Lou *et al*, 2012; Khvorova *et al*, 2003)	AGTACGTC|TGAGCGTGATACCCGCTCACTGAAGATGGCCCGGTAGGGCCGAAACGTACCTCTACAAATAATTTTGTTTAA
PlmJ (Lou *et al*, 2012; Khvorova *et al*, 2003)	AGTCATAAGTC|TGGGCTAAGCCCACTGATGAGTCGCTGAAATGCGACGAAACTTATGACCTCTACAAATAATTTTGTTTAA
SarJ (Lou *et al*, 2012; Khvorova *et al*, 2003)	AGACTGTC|GCCGGATGTGTATCCGACCTGACGATGGCCCAAAAGGGCCGAAACAGTCCTCTACAAATAATTTTGTTTAA
ScmJ (Khvorova *et al*, 2003)	AGCGCTGTC|TGTACTTGTATCAGTACACTGACGAGTCCCTAAAGGACGAAACACCGCCTCTACAAATAATTTTGTTTAA
Engineered	
RiboJ10	AGCGCTC|AACGGGTGTGCTTCCCGTTCTGATGAGTCCGTGAGGACGAAAGCGCCTCTACAAATAATTTTGTTTAA
RiboJ51	AGTAGTC|ACCGGCTGTGCTTGCCGGTCTGATGAGCCTGTGAAGGCGAAACTACCTCTACAAATAATTTTGTTTAA
RiboJ53	AGCGGTC|AACGCATGTGCTTTGCGTTCTGATGAGACAGTGATGTCGAAACCGCCTCTACAAATAATTTTGTTTAA
RiboJ54	AGGGGTC|AGTTGATGTGCTTTCAACTCTGATGAGTCAGTGATGACGAAACCCCCTCTACAAATAATTTTGTTTAA
RiboJ57	AGAAGTC|AATTAATGTGCTTTTAATTCTGATGAGTCGGTGACGACGAAACTTCCTCTACAAATAATTTTGTTTAA
RiboJ60	AGTCGTC|AAGTGCTGTGCTTGCACTTCTGATGAGGCAGTGATGCCGAAACGACCTCTACAAATAATTTTGTTTAA
RiboJ64	AGGAGTC|AATTAATGTGCTTTTAATTCTGATGAGACGGTGACGTCGAAACTCCCTCTACAAATAATTTTGTTTAA

a. *Each insulator is annotated for functional sequences: | cleavage site, red catalytic cores, blue loops, underlined 3'-hairpin.*

Table 4-2: Terminator sequence alignment

Name	Strength[a]	Sequence
Natural		
ECK120033737	310	------GGAA----------ACACAGAAAAAAGCCCGCACCTGACAGTGCGGGCT-TTTTTTTTCGACCAAAGG
ECK120029600	380	TTCAGCCAAAAAACTTAAGACCGCCGGTCTTGTCCACTACCTTGCAGTAATGCGGTGGACAGGATCGGCGGTTTTCTTTTCTCTTCTCAA--
ECK120015440	120	------TCCGGC----------AATTAAAAAAGCGGCTAACCACGCCGCTTTTTTTACGTCTGCA-----
ECK120010876	97	-TAAGGTTGAAA----------AATAAAAACGGCGCTAAAAAGCGCCGTTTTTTTTGACGGTGGTA----
ECK120033736	170	----------AACGCATGA--GAAAGCCCCCGGAAG-ATCACCTTCCGGGGGCTTTTTTATTGCGC---------
ECK120010818	150	----------GTCAGTTTCA--CCTGTTTTACGTAAAAACCCGCTTCGGCGGGTTTTTACTTTTGG----------
ECK120015170	86	----------ACAATTTTCGAAAAAACCCGCTTCGGCGGGTTTTTTTATAGCTAAAA------
Engineered		
L3S3P31[b]	110	----------CCAATTATTGAACACCCTAACGGGTGTTTTTTTTTTTTGGTCTACC--
L3S3P11[b]	170	----------CCAATTATTGAACACCCTTCGGGGTGTTTTTTTGTTTCTGGTCTACC--
L3S2P24	150	----------CTCGGTACCA---AATTCCAGAAAAGACACC--CGAAAGGGTGTTTTTTCGTTTTGGTCC------
L3S2P11	260	----------CTCGGTACCA---AATTCCAGAAAAGAGACGCTTTCGAGCGTCTTTTTTCGTTTTGGTCC------
L3S2P55	260	----------CTCGGTACCA---AAGACGAACAATAAGACGCTGAAAAGCGTCTTTTTTCGTTTTGGTCC------

a. Strength values reproduced from ref (Chen *et al*, 2013).

b. The "C" at nucleotide 45 from was mutated to "A" to eliminate a BsaI recognition site.

Table 4-3: Insulated gate RBS sequences

Repressor	RBS	DNA sequence
AmeR	F1	CTATGGACTATGTTTTCACATACGAGGGGGATTAG
AmtR	A1	AATGTTCCCTAATAATCAGCAAAGAGGTTACTAG
BetI	E1	CCCCCCGAGGAGTAGCAC
BM3R1	B1	CTATGGACTATGTTTTAACTACTAG
	B2	CTATGGACTATGTTTTTCAAAGACGAAAAACTACTAG
	B3	CCAAACGAGGCCGGGAGG
HlyIIR	H1	ACCCCCGAG
IcaRA	I1	ATTGCTATGGACTATGTTTCAAAGTGAGAATACTAG
LitR	L1	GTCCTATGGACTTTTTCATACAGGAGAACCCTCG
LmrA	N1	TACGCTATGGACTATGTTTTCTGCTATGGACTATGTTTTCACACACGAGATGCCTCG
PhlF	P1	CTATGGACTATGTTTGAAAGGGAGAAATACTAG
	P2	GGAGCTATGGACTATGTTTGAAAGGCTGAAATACTAG
	P3	CTTTACGAGGGCGATCCT
PsrA	R1	TTTAATTCGCGGAAGCGCAGAGATAAGGGGTATC
QacR	Q1	GTAAGCCATGCCATTGGCTTTTGATAGAGGATAACTACTAG
	Q2	GCCATGCCATTGGCTTTTGATAGAGGACAACTACTAG
SrpR	S1	GAGTCTATGGACTATGTTTTCACAGAGGAGGTACCAGG
	S2	GAGTCTATGGACTATGTTTTCACATATGAGATACCAGG
	S3	GAGTCTATGGACTATGTTTTCACAAAGGAAGTACCAGG
	S4	CTATGGACTATGTTTTCACACAGGAAATACCAGG

Table 4-4: Insulated gate response function parameters

Repressor	RBS	y_{min}[a]	y_{max}[a]	K[a]	n	Toxicity (RPU)[b]
AmeR	F1	0.2	3.8	0.09	1.4	-
AmtR	A1	0.06	3.8	0.07	1.6	4.1
BetI	E1	0.07	3.8	0.41	2.4	-
BM3R1	B1	0.004	0.5	0.04	3.4	-
	B2	0.005	0.5	0.15	2.9	-
	B3	0.01	0.8	0.26	3.4	-
HlyIIR	H1	0.07	2.5	0.19	2.6	-
IcaRA	I1	0.08	2.2	0.10	1.4	1.7
LitR	L1	0.07	4.3	0.05	1.7	0.2
LmrA	N1	0.2	2.2	0.18	2.1	-
PhlF	P1	0.01	3.9	0.03	4.0	-
	P2	0.02	4.1	0.13	3.9	-
	P3	0.02	6.8	0.23	4.2	-
PsrA	R1	0.2	5.9	0.19	1.8	-
QacR	Q1	0.01	2.4	0.05	2.7	N.D.
	Q2	0.03	2.8	0.21	2.4	1.7
SrpR	S1	0.003	1.3	0.01	2.9	-
	S2	0.003	2.1	0.04	2.6	-
	S3	0.004	2.1	0.06	2.8	-
	S4	0.007	2.1	0.10	2.8	-

a. In units of RPU.
b. Highest input RPU achieved before cell growth was reduced >20% compared to a control (Methods). Dashes indicate no toxicity observed at the highest inducer levels. N.D. means no data collected.

Table 4-5: Non-insulated gate parameters[a]

Name	K[b]	n	y_{max}[b]	y_{min}[b]
AmeR	0.11	1.4	3.9	0.40
AmtR	0.06	1.8	2.2	0.08
BetI	0.05	2.4	3.1	0.09
BM3R1	0.13	4.5	0.61	0.02
ButR	0.30	2.4	2.9	0.44
HlyIIR	0.12	2.7	3.9	0.08
IcaR(A)	0.10	1.8	3.0	0.09
LitR	0.03	1.9	3.9	0.12
LmrA	0.29	3.1	17	0.27
McbR	0.10	1.6	3.8	0.27
PhlF	0.09	4.5	3.8	0.02
PsrA	0.10	2.0	4.7	0.11
QacR	0.11	1.4	5.0	0.05
SrpR	0.07	3.2	6.0	0.03
TarA	0.02	1.8	3.0	0.05

a. The RPU standard in ref (Stanton *et al*, 2014) differs from this manuscript (Figure 4-37). These values were recalculated based on the new standard.
b. In units of RPU.

Table 4-6: Calculation of Sensor OFF/ON activities

		<YFP>	$\langle YFP \rangle_{RPU}$	$\langle YFP \rangle_0$	RPUs
P_{Tac}	OFF	16.6	475	15	0.0034
	ON	1300	475	15	2.8
P_{Tet}	OFF	15.6	475	15	0.0013
	ON	2020	475	15	4.4
P_{BAD}	OFF	18.8	475	15	0.0082
	ON	1170	475	15	2.5

Table 4-7: List of plasmids used in this work

Plasmid name	Description
pAN215	Non-insulated LmrA NOT gate
pAN216	Non-insulated LmrA NOR gate
pAN412	Insulated LmrA NOT gate
pAN413	Insulated LmrA NOR gate
pAN901	Circuit plasmid (no insulating terminator): A IMPLY B v1
pAN902	Circuit plasmid (no insulating terminator): B IMPLY A v1
pAN903	Circuit plasmid (no insulating terminator): A NIMPLY B v1
pAN904	Circuit plasmid (no insulating terminator): B NIMPLY A v1
pAN905	Circuit plasmid (no insulating terminator): NAND v1
pAN906	Circuit plasmid (no insulating terminator): AND v1
pAN907	Circuit plasmid (no insulating terminator): XOR v1
pAN908	Circuit plasmid (no insulating terminator): XNOR v1
pAN1201	Circuit backbone plasmid (LacZa insert)
pAN1250	Circuit plasmid (no insulating terminator): PTac-YFP
pAmeR-F1	Insulated AmeR NOT gate, RBS F1
pAmtR-A1	Insulated AmtR NOT gate, RBS A1
pBetI-E1	Insulated BetI NOT gate, RBS E1
pBM3R1-B1	Insulated BM3R1 NOT gate, RBS B1
pBM3R1-B2	Insulated BM3R1 NOT gate, RBS B2
pBM3R1-B3	Insulated BM3R1 NOT gate, RBS B3
pHlyIIR-H1	Insulated HlyIIR NOT gate, RBS H1
pIcaRA-I1	Insulated IcaRA NOT gate, RBS I1
pLitR-L1	Insulated LitR NOT gate, RBS L1
pLmrA-N1	Insulated LmrA NOT gate, RBS N1
pPhlF-P1	Insulated PhlF NOT gate, RBS P1
pPhlF-P2	Insulated PhlF NOT gate, RBS P2
pPhlF-P3	Insulated PhlF NOT gate, RBS P3
pPsrA-R1	Insulated PsrA NOT gate, RBS R1
pQacR-Q1	Insulated QacR NOT gate, RBS Q1
pQacR-Q2	Insulated QacR NOT gate, RBS Q2
pSrpR-S1	Insulated SrpR NOT gate, RBS S1
pSrpR-S2	Insulated SrpR NOT gate, RBS S2
pSrpR-S3	Insulated SrpR NOT gate, RBS S3
pSrpR-S4	Insulated SrpR NOT gate, RBS S4
pAN1681	Roadblocking test: PTet-YFP
pAN1682	Roadblocking test: PTet-PTac-YFP
pAN1683	Roadblocking test: PTet-PLux*-YFP
pAN1684	Roadblocking test: PTet-PBAD-YFP
pAN1685	Roadblocking test: PTet-PPhlF-YFP
pAN1686	Roadblocking test: PTet-PSrpR-YFP
pAN1687	Roadblocking test: PTet-PBM3R1-YFP
pAN1688	Roadblocking test: PTet-PBetI-YFP
pAN1689	Roadblocking test: PTet-PQacR-YFP
pAN1690	Roadblocking test: PTet-PAmtR-YFP
pAN1691	Roadblocking test: PTet-PHlyIIR-YFP
pAN1692	Roadblocking test: PTet-PIcaRA-YFP
pAN1693	Roadblocking test: PTet-PAmeR-YFP
pAN1694	Roadblocking test: PTet-PLitR-YFP
pAN1695	Roadblocking test: PTet-PPsrA-YFP
pAN1696	Roadblocking test: PTet-PLmrA-YFP
pAN1697	Roadblocking test: PTac-PTet-YFP
pAN1717	RPU standard plasmid
pAN1718	Circuit plasmid: PTac-YFP (constitutive AraC*, LacI, TetR)
pAN1719	Circuit plasmid: PTet-YFP (constitutive AraC*, LacI, TetR)
pAN1720	Circuit plasmid: PBAD-YFP (constitutive AraC*, LacI, TetR)
pAN1818	Circuit plasmid: PTac-YFP (constitutive LacI)
pAN3901	Circuit plasmid: 0xEA
pAN3902	Circuit plasmid: 0x70
pAN3903	Circuit plasmid: 0x3B
pAN3904	Circuit plasmid: 0x7F
pAN3905	Circuit plasmid: 0xC8
pAN3906	Circuit plasmid: 0x07
pAN3907	Circuit plasmid: 0x37
pAN3908	Circuit plasmid: 0xF7
pAN3909	Circuit plasmid: 0xAE
pAN3910	Circuit plasmid: 0x80
pAN3911	Circuit plasmid: 0xC4
pAN3912	Circuit plasmid: 0x0E
pAN3913	Circuit plasmid: 0xCD
pAN3914	Circuit plasmid: 0x0B
pAN3915	Circuit plasmid: 0xFB
pAN3916	Circuit plasmid: 0x08
pAN3917	Circuit plasmid: Multiplexer alternative assignment
pAN3918	Circuit plasmid: Multiplexer
pAN3919	Circuit plasmid: 0xC7
pAN3920	Circuit plasmid: 0x6E
pAN3921	Circuit plasmid: 0x8E
pAN3922	Circuit plasmid: 0x9F
pAN3923	Circuit plasmid: 0x87
pAN3924	Circuit plasmid: 0x04
pAN3925	Circuit plasmid: 0x38
pAN3926	Circuit plasmid: 0x6F
pAN3927	Circuit plasmid: 0xE8
pAN3928	Circuit plasmid: 0x78
pAN3929	Circuit plasmid: 0x4D

pAN3930	Circuit plasmid: 0xBD
pAN3931	Circuit plasmid: 0xF9
pAN3932	Circuit plasmid: 0x60
pAN3933	Circuit plasmid: 0x3D
pAN3934	Circuit plasmid: 0xB9
pAN3935	Circuit plasmid: 0x19
pAN3936	Circuit plasmid: 0x01
pAN3937	Circuit plasmid: Majority alternative assignment
pAN3938	Circuit plasmid: 0xF6
pAN3939	Circuit plasmid: 0x98
pAN3940	Circuit plasmid: 0xC6
pAN3941	Circuit plasmid: 0x82
pAN3942	Circuit plasmid: 0x06
pAN3943	Circuit plasmid: 0x36
pAN3944	Circuit plasmid: 0x1C
pAN3945	Circuit plasmid: 0x41
pAN3946	Circuit plasmid: 0xC9
pAN3947	Circuit plasmid: 0xC1
pAN3948	Circuit plasmid: Consensus alternative assignment
pAN3949	Circuit plasmid: Consensus
pAN3950	Circuit plasmid: Priority detector
pAN3951	Circuit plasmid: Demultiplexer
pAN3952	Circuit plasmid: Majority #1 - Original
pAN3953	Circuit plasmid: Majority #2 - Reversed order
pAN3954	Circuit plasmid: Majority #3 - NOTs and NORs clustered
pAN3955	Circuit plasmid: Majority #4 - Subcircuits clustered
pAN3956	Circuit plasmid: Majority #5 - Scrambled order
pAN3957	Circuit plasmid: Majority #6 - Alternating orientation
pAN3958	Circuit plasmid: A IMPLY B v2
pAN3959	Circuit plasmid: B IMPLY A v2
pAN3960	Circuit plasmid: A NIMPLY B v2
pAN3961	Circuit plasmid: B NIMPLY A v2
pAN3962	Circuit plasmid: NAND v2
pAN3963	Circuit plasmid: AND v2
pAN3964	Circuit plasmid: XOR v2
pAN3965	Circuit plasmid: XNOR v2
pAN4020	Output backbone plasmid (LacZalpha insert)
pAN4021	Output plasmid: PTac-YFP
pAN4022	Output plasmid: PTet-YFP
pAN4023	Output plasmid: PBM3R1-YFP
pAN4024	Output plasmid: PBetI-YFP
pAN4025	Output plasmid: PAmeR-YFP
pAN4026	Output plasmid: PPhlF-YFP
pAN4027	Output plasmid: PSrpR-YFP
pAN4028	Output plasmid: PTac-PAmeR-YFP
pAN4029	Output plasmid: PTac-PAmtR-YFP
pAN4030	Output plasmid: PTet-PAmtR-YFP
pAN4031	Output plasmid: PBAD-PAmtR-YFP
pAN4032	Output plasmid: PBM3R1-PHlyIIR-YFP
pAN4033	Output plasmid: PBM3R1-PTet-YFP
pAN4034	Output plasmid: PBetI-PAmeR-YFP
pAN4035	Output plasmid: PPhlF-PTet-YFP
pAN4036	Output plasmid: PPhlF-PAmtR-YFP
pAN4037	Output plasmid: PPhlF-PBetI-YFP
pAN4038	Output plasmid: PPhlF-PHlyIIR-YFP
pAN4039	Output plasmid: PSrpR-PAmtR-YFP
pAN4040	Output plasmid: PSrpR-PBetI-YFP
pAN4041	Output plasmid: PSrpR-PHlyIIR-YFP
pJS1	Ribozyme test plasmid: pTac-GFP
pJS2	Ribozyme test plasmid: pTac-CI-GFP
pJS3	Ribozyme test plasmid: pLlacO-1-GFP
pJS4	Ribozyme test plasmid: pLlacO-1-CI-GFP
pJS5	Ribozyme test plasmid: pTac-RiboJ00-GFP
pJS6	Ribozyme test plasmid: pTac-RiboJ00-CI-GFP
pJS7	Ribozyme test plasmid: pLlacO-1-RiboJ00-GFP
pJS8	Ribozyme test plasmid: pLlacO-1-RiboJ00-CI-GFP
pJS9	Ribozyme test plasmid: pTac-RiboJ10-GFP
pJS10	Ribozyme test plasmid: pTac-RiboJ10-CI-GFP
pJS11	Ribozyme test plasmid: pLlacO-1-RiboJ10-GFP
pJS12	Ribozyme test plasmid: pLlacO-1-RiboJ10-CI-GFP
pJS13	Ribozyme test plasmid: pTac-RiboJ51-GFP
pJS14	Ribozyme test plasmid: pTac-RiboJ51-CI-GFP
pJS15	Ribozyme test plasmid: pLlacO-1-RiboJ51-GFP
pJS16	Ribozyme test plasmid: pLlacO-1-RiboJ51-CI-GFP
pJS17	Ribozyme test plasmid: pTac-RiboJ53-GFP
pJS18	Ribozyme test plasmid: pTac-RiboJ53-CI-GFP
pJS19	Ribozyme test plasmid: pLlacO-1-RiboJ53-GFP
pJS20	Ribozyme test plasmid: pLlacO-1-RiboJ53-CI-GFP
pJS21	Ribozyme test plasmid: pTac-RiboJ54-GFP
pJS22	Ribozyme test plasmid: pTac-RiboJ54-CI-GFP
pJS23	Ribozyme test plasmid: pLlacO-1-RiboJ54-GFP
pJS24	Ribozyme test plasmid: pLlacO-1-RiboJ54-CI-GFP
pJS25	Ribozyme test plasmid: pTac-RiboJ57-GFP
pJS26	Ribozyme test plasmid: pTac-RiboJ57-CI-GFP
pJS27	Ribozyme test plasmid: pLlacO-1-RiboJ57-GFP
pJS28	Ribozyme test plasmid: pLlacO-1-RiboJ57-CI-GFP
pJS29	Ribozyme test plasmid: pTac-RiboJ60-GFP
pJS30	Ribozyme test plasmid: pTac-RiboJ60-CI-GFP
pJS31	Ribozyme test plasmid: pLlacO-1-RiboJ60-GFP
pJS32	Ribozyme test plasmid: pLlacO-1-RiboJ60-CI-GFP

pJS33	Ribozyme test plasmid: pTac-RiboJ64-GFP
pJS34	Ribozyme test plasmid: pTac-RiboJ64-CI-GFP
pJS35	Ribozyme test plasmid: pLlacO-1-RiboJ64-GFP
pJS36	Ribozyme test plasmid: pLlacO-1-RiboJ64-CI-GFP
pJS37	Ribozyme test plasmid: pTac-AraJ-GFP
pJS38	Ribozyme test plasmid: pTac-AraJ-CI-GFP
pJS39	Ribozyme test plasmid: pLlacO-1-AraJ-GFP
pJS40	Ribozyme test plasmid: pLlacO-1-AraJ-CI-GFP
pJS41	Ribozyme test plasmid: pTac-BydvJ-GFP
pJS42	Ribozyme test plasmid: pTac-BydvJ-CI-GFP
pJS43	Ribozyme test plasmid: pLlacO-1-BydvJ-GFP
pJS44	Ribozyme test plasmid: pLlacO-1-BydvJ-CI-GFP
pJS45	Ribozyme test plasmid: pTac-CchJ-GFP
pJS46	Ribozyme test plasmid: pTac-CchJ-CI-GFP
pJS47	Ribozyme test plasmid: pLlacO-1-CchJ-GFP
pJS48	Ribozyme test plasmid: pLlacO-1-CchJ-CI-GFP
pJS49	Ribozyme test plasmid: pTac-ElvJ-GFP
pJS50	Ribozyme test plasmid: pTac-ElvJ-CI-GFP
pJS51	Ribozyme test plasmid: pLlacO-1-ElvJ-GFP
pJS52	Ribozyme test plasmid: pLlacO-1-ElvJ-CI-GFP
pJS53	Ribozyme test plasmid: pTac-LtsvJ-GFP
pJS54	Ribozyme test plasmid: pTac-LtsvJ-CI-GFP
pJS55	Ribozyme test plasmid: pLlacO-1-LtsvJ-GFP
pJS56	Ribozyme test plasmid: pLlacO-1-LtsvJ-CI-GFP
pJS57	Ribozyme test plasmid: pTac-PlmJ-GFP
pJS58	Ribozyme test plasmid: pTac-PlmJ-CI-GFP
pJS59	Ribozyme test plasmid: pLlacO-1-PlmJ-GFP
pJS60	Ribozyme test plasmid: pLlacO-1-PlmJ-CI-GFP
pJS61	Ribozyme test plasmid: pTac-SarJ-GFP
pJS62	Ribozyme test plasmid: pTac-SarJ-CI-GFP
pJS63	Ribozyme test plasmid: pLlacO-1-SarJ-GFP
pJS64	Ribozyme test plasmid: pLlacO-1-SarJ-CI-GFP
pJS65	Ribozyme test plasmid: pTac-ScmJ-GFP
pJS66	Ribozyme test plasmid: pTac-ScmJ-CI-GFP
pJS67	Ribozyme test plasmid: pLlacO-1-ScmJ-GFP
pJS68	Ribozyme test plasmid: pLlacO-1-ScmJ-CI-GFP
pJS101	Toxicity test plasmid: pTac-pTet-SrpR
pJS102	Toxicity test plasmid: pTac-pTet-PhlF
pJS103	Toxicity test plasmid: pTac-pTet-BM3R1
pJS104	Toxicity test plasmid: pTac-pTet-AmtR
pJS105	Toxicity test plasmid: pTac-pTet-HlyIIR
pJS106	Toxicity test plasmid: pTac-pTet-LitR
pJS107	Toxicity test plasmid: pTac-pTet-QacR
pJS108	Toxicity test plasmid: pTac-pTet-IcaRA
pJS109	Toxicity test plasmid: pTac-pTet-BetI

Table 4-8: Sequences of insulated gates and sensor modules used in this work

Part name	Type	DNA sequence[a]
AmeR (RBS-F1) gate	gate	CTGAAGGGGTCAGTTGATGTGCTTTCAACTCTGATGAGTCAGTGATGACGAAACCCCCTCTACAAATAATTTTGTTT AACTATGGACTATGTTTTCACATACGAGGGGGATTAGATGAACAAAACCATTGATCAGGTGCGTAAAGGTGATCGTA AAAGCGATCTGCCGGTTCGTCGTCGTCCGCGTCGTAGTGCCGAAGAAACCCGTCGTGATATTCTGGCAAAAGCCGAA GAACTGTTTCGTGAACGTGGTTTTAATGCAGTTGCCATTGCAGATATTGCAAGCGCACTGAATATGAGTCCGGCAAA TGTGTTTAAACATTTTAGCAGCAAAAACGCACTGGTTGATGCAATTGGTTTTGGTCAGATTGGTGTTTTTGAACGTC AGATTTGTCCGCTGGATAAAAGCCATGCACCGCTGGATCGTCTGCGTCATCTGGCACGTAATCTGATGGAACAGCAT CATCAGGATCATTTCAAACACATACGGGTTTTTATTCAGATCCTGATGACCGCCAAACAGGATATGAAATGTGGCGA TTATTACAAAAGCGTGATTGCAAAACTGCTGGCCGAAATTATTCGTGATGGTGTTGAAGCAGGTCTGTATATTGCAA CCGATATTCCGGTTCTGGCAGAAACCGTTCTGCATGCACTGACCAGCGTTATTCATCCGGTTCTGATTGCACAAGAA GATATTGGTAATCTGGCAACCCGTTGTGATCAGCTGGTTGATCTGATTGATGCAGGTCTGCGTAATCCGCTGGCAAA ATAACCAATTATTGAACACCCTAACGGGTGTTTTTTTTTTTTTGGTCTACCTCGTCACTAGAGGGCGATAGTGACAA ACTTGACAACTGATCACTTCCTAGGTATAATGCTAGC
AmtR (RBS-A1) gate	gate	CTGAAGGGTGTCTCAAGGTGCGTACCTTGACTGATGAGTCCGAAAGGACGAAACACCCCTCTACAAATAATTTTGTT TAAAATGTTCCCTAATAATCAGCAAAGAGGTTACTAGATGGCAGGCGCAGTTGGTCGTCCGCGTCGTAGTGCACCGC GTCGTGCAGGTAAAAATCCGCGTGAAGAAATTCTGGATGCAAGCGCAGAACTGTTTACCCGTCAGGGTTTTGCAACC ACCAGTACCCATCAGATTGCAGATGCAGTTGGTATTCGTCAGGCAAGCCTGTATTATCATTTTCCGAGCAAAACCGA AATCTTTCTGACCCTGCTGAAAAGCACCGTTGAACCGAGCACCGTTCTGGCAGAAGATCTGAGCACCCTGGATGCAG GTCCGGAAATGCGTCTGTGGGCAATTGTTGCAAGCGAAGTTCGTCTGCTGCTGAGCACCAAATGGAATGTTGGTCGT CTGTATCAGCTGCCGATTGTTGGTAGCGAAGAATTTGCAGAATATCATAGCCAGCGTGAAGCACTGACCAATGTTTT TCGTGATCTGGCAACCGAAATTGTTGGTGATGATCCGCGTGCAGAACTGCCGTTTCATATTACCATGAGCGTTATTG AAATGCGTCGCAATGATGGTAAAATTCCGAGTCCGCTGAGCGCAGATAGCCTGCCGGAAACCGCAATTATGCTGGCA GATGCAAGCCTGGCAGTTCTGGGTGCACCGCTGCCTGCAGATCGTGTTGAAAAAACCCTGGAACTGATTAAACAGGC AGATGCAAAATAACTCGGTACCAAAGACGAACAATAAGACGCTGAAAAGCGTCTTTTTTCGTTTTGGTCCCTTGTCC AACCAAATGATTCGTTACCAATTGACAGTTTCTATCGATCTATAGATAATGCTAGC
BM3R1 (RBS-B1) gate	gate	CTGAAGACTGTCGCCGGATGTGTATCCGACCTGACGATGGCCCAAAAGGGCCGAAACAGTCCTCTACAAATAATTTT GTTTAACTATGGACTATGTTTTAACTACTAGATGGAAAGCACCCCGACCAAACAGAAAGCAATTTTTAGCGCAAGCC TGCTGCTGTTTGCAGAACGTGGTTTTGATGCAACCACCATGCCGATGATTGCAGAAAATGCAAAAGTTGGTGCAGGC ACCATTTATCGCTATTTCAAAAACAAAGAAAGCCTGGTGAACGAACTGTTTCAGCAGCATGTTAATGAATTTCTGCA GTGTATTGAAAGCGGTCTGGCAAATGAACGTGATGGTTATCGTGATGGCTTTCATCACATTTTTGAAGGTATGGTGA CCTTTACCAAAAATCATCCGCGTGCACTGGGTTTTATCAAAACCCATAGCCAGGGCACCTTTCTGACCGAAGAAAGC CGTCTGGCATATCAGAAACTGGTTGAATTTGTGTGCACCTTTTTTCGTGAAGGTCAGAAACAGGGTGTGATTCGTAA TCTGCCGGAAAATGCACTGATTGCAATTCTGTTTGGCAGCTTTATGGAAGTGTATGAAATGATCGAGAACGATTATC TGAGCCTGACCGATGAACTGCTGACCGGTGTTGAAGAAAGCCTGTGGGCAGCACTGAGCCGTCAGAGCTAACTCGGT ACCAAATTCCAGAAAAGAGACGCTTTCGAGCGTCTTTTTTCGTTTTGGTCCAATCCGCGTGATAGGTCTGATTCGTT ACCAATTGACGGAATGAACGTTCATTCCGATAATGCTAGC
BM3R1 (RBS-B2) gate	gate	CTGAAGACTGTCGCCGGATGTGTATCCGACCTGACGATGGCCCAAAAGGGCCGAAACAGTCCTCTACAAATAATTTT GTTTAACTATGGACTATGTTTTTCAAAGACGAAAAACTACTAGATGGAAAGCACCCCGACCAAACAGAAAGCAATTT TTAGCGCAAGCCTGCTGCTGTTTGCAGAACGTGGTTTTGATGCAACCACCATGCCGATGATTGCAGAAAATGCAAAA GTTGGTGCAGGCACCATTTATCGCTATTTCAAAAACAAAGAAAGCCTGGTGAACGAACTGTTTCAGCAGCATGTTAA TGAATTTCTGCAGTGTATTGAAAGCGGTCTGGCAAATGAACGTGATGGTTATCGTGATGGCTTTCATCACATTTTTG AAGGTATGGTGACCTTTACCAAAAATCATCCGCGTGCACTGGGTTTTATCAAAACCCATAGCCAGGGCACCTTTCTG ACCGAAGAAAGCCGTCTGGCATATCAGAAACTGGTTGAATTTGTGTGCACCTTTTTTCGTGAAGGTCAGAAACAGGG TGTGATTCGTAATCTGCCGGAAAATGCACTGATTGCAATTCTGTTTGGCAGCTTTATGGAAGTGTATGAAATGATCG AGAACGATTATCTGAGCCTGACCGATGAACTGCTGACCGGTGTTGAAGAAAGCCTGTGGGCAGCACTGAGCCGTCAG AGCTAACTCGGTACCAAATTCCAGAAAAGAGACGCTTTCGAGCGTCTTTTTTCGTTTTGGTCCAATCCGCGTGATAG GTCTGATTCGTTACCAATTGACGGAATGAACGTTCATTCCGATAATGCTAGC
BM3R1 (RBS-B3) gate	gate	CTGAAGACTGTCGCCGGATGTGTATCCGACCTGACGATGGCCCAAAAGGGCCGAAACAGTCCTCTACAAATAATTTT GTTTAACCAAACGAGGCCGGGAGGATGGAAAGCACCCCGACCAAACAGAAAGCAATTTTTAGCGCAAGCCTGCTGCT GTTTGCAGAACGTGGTTTTGATGCAACCACCATGCCGATGATTGCAGAAAATGCAAAAGTTGGTGCAGGCACCATTT ATCGCTATTTCAAAAACAAAGAAAGCCTGGTGAACGAACTGTTTCAGCAGCATGTTAATGAATTTCTGCAGTGTATT GAAAGCGGTCTGGCAAATGAACGTGATGGTTATCGTGATGGCTTTCATCACATTTTTGAAGGTATGGTGACCTTTAC CAAAAATCATCCGCGTGCACTGGGTTTTATCAAAACCCATAGCCAGGGCACCTTTCTGACCGAAGAAAGCCGTCTGG CATATCAGAAACTGGTTGAATTTGTGTGCACCTTTTTTCGTGAAGGTCAGAAACAGGGTGTGATTCGTAATCTGCCG GAAAATGCACTGATTGCAATTCTGTTTGGCAGCTTTATGGAAGTGTATGAAATGATCGAGAACGATTATCTGAGCCT GACCGATGAACTGCTGACCGGTGTTGAAGAAAGCCTGTGGGCAGCACTGAGCCGTCAGAGCTAACTCGGTACCAAAT TCCAGAAAAGAGACGCTTTCGAGCGTCTTTTTTCGTTTTGGTCCAATCCGCGTGATAGGTCTGATTCGTTACCAATT GACGGAATGAACGTTCATTCCGATAATGCTAGC
BetI (RBS-E1) gate	gate	CTGAAGAAGTCAATTAATGTGCTTTTAATTCTGATGAGTCGGTGACGACGAAACTTCCTCTACAAATAATTTTGTTT AACCCCCCGAGGAGTAGGACATGCCGAAACTGGGTATGCAGAGCATTCGTCGTCGTCAGCTGATTGATGCAACCCTG GAAGCAATTAATGAAGTTGGTATGCATGATGCAACCATTGCACAGATTGCACGTCGTGCCGGTGTTAGCACCGGTAT TATTAGCCATTATTTCCGCGATAAAAACGGTCTGCTGGAAGCAACCATGCGTGATATTACCAGCCAGCTGCGTGATG CAGTTCTGAATCGTCTGCATGCACTGCCGCAGGGTAGCGCAGAACAGCGTCTGCAGGCAATTGTTGGTGGTAATTTT GATGAAACCCAGGTTAGCAGCGCAGCAATGAAAGCATGGCTGGCATTTTGGGCAAGCAGCATGCATCAGCCGATGCT GTATCGTCTGCAGCAGGTTAGCAGTCGTCGTCTGCTGAGCAATCTGGTTAGCGAATTTCGTCGTGAACTGCCTCGTG AACAGGCACAAGAGGCAGGTTATGGTCTGGCAGCACTGATTGATGGTCTGTGGCTGCGTGCAGCACTGAGCGGTAAA CCGCTGGATAAAACCCGTGCAAATAGCCTGACCCGTCATTTTATCACCCAGCATCTGCCGACCGATTAACCAATTAT TGAACACCCTTCGGGGTGTTTTTTTGTTTCTGGTCTACCAGCGCGGGTGAGAGGGATTCGTTACCAATTGACAATTG ATTGGACGTTCAATATAATGCTAGC
HlyIIR (RBS-H1) gate	gate	CTGAAGTAGTCACCGGCTGTGCTTGCCGGTCTGATGAGCCTGTGAAGGCGAAACTACCTCTACAAATAATTTTGTTT AAACCCCCGAGATGAAATACATCCTGTTTGAGGTGTGCGAAATGGGTAAAAGCCGTGAACAGACCATGGAAAATATT CTGAAAGCAGCCAAAAAGAAATTCGGCGAACGTGGTTATGAAGGCACCAGCATTCAAGAAATTACCAAAGAAGCCAA AGTTAACGTTGCAATGGCCAGCTATTACTTTAATGGCAAAGAGAACCTGTACTACGAGGTGTTCAAAAAATACGGTC TGGCAAATGAACTGCCGAACTTTCTGGAAAAAAACCAGTTTAATCCGATTAATGCCCTGCGTGAATATCTGACCGTT TTTACCACCCACATTAAAGAAAATCCGGAAATTGGCACCCTGGCCTATGAAGAAATTATCAAAGAAAGCGCACGCCT GGAAAAAATCAAACCGTATTTTATCGGCAGCTTCGAACAGCTGAAAGAAATTCTGCAAGAGGGTGAAAAACAGGGTG TGTTTCACTTTTTTAGCATCAACCATACCATCCATTGGATTACCAGCATTGTTCTGTTTCCGAAATTCAAAAAATTC ATCGATAGCCTGGGTCCGAATGAAACCAATGATACCAATCATGAATGGATGCCGGAAGATCTGGTTAGCCGTATTAT TAGCGCACTGACCGATAAACCGAACATTTAAAACGCATGAGAAAGCCCCCGGAAGATCACCTTCCGGGGGCTTTTTT

IcaRA (RBS-I1) gate	gate	ATTGCGCACCAGGAATCTGAACGATTCGTTACCAATTGACATATTTAAAATTCTTGTTTAAAATGCTAGC CTGAAGCCCCATAGGGTGGTGTGTACCACCCCTGATGAGTCCAAAAGGACGAAATGGGGCCTCTACAAATAATTTTG TTTAAATTGCTATGGACTATGTTTCAAAGTGAGAATACTAGGTGAAAGACAAAATTATCGATAACGCCATCACCCTG TTTAGCGAAAAAGGTTATGACGGCACCACCCTGGATGATATTGCAAAAAGCGTGAACATCAAAAAAGCCAGCCTGTA TTATCACTTTGATAGCAAAAAAAGCATCTACGAGCAGAGCGTTAAATGCTGTTTCGATTATCTGAACAACATCATCA TGATGAACCAGAACAAAAGCAACTATAGCATCGATGCCCTGTATCAGTTTCTGTTTGAGTTCATCTTCGATATCGAG GAACGCTATATTCGTATGTATGTTCAGCTGAGCAACACACCGGAAGAATTTTCAGGTAACATTTATGGCCAGATCCA GGATCTGAATCAGAGCCTGAGCAAAGAAATCGCCAAATTCTATGACGAAAGCAAAATCAAAATGACCAAAGAGGACT TCCAGAATCTGATTCTGCTGTTTCTGGAAAGCTGGTATCTGAAAGCCAGCTTTAGCCAGAAATTTGGTGCAGTTGAA GAAAGCAAAAGCCAGTTTAAAGATGAGGTTTATAGCCTGCTGAACATCTTTCTGAAGAAATAAACAATTTTCGAAAA AACCCGCTTCGGCGGGTTTTTTTATAGCTAAAAGTCAACTCATAAGATTCTGATTCGTTACCAATTGACAATTCACC TACCTTTCGTTAGGTTAGGTTGT
LitR (RBS-L1) gate	gate	CTGAAGTCATAAGTCTGGGCTAAGCCCACTGATGAGTCGCTGAAATGCGACGAAACTTATGACCTCTACAAATAATT TTGTTTAAGTCCTATGGACTTTTTCATACAGGAGAACCCTCGATGGATACCATTCAGAAACGTCCGCGTACCCGTCT GAGTCCGGAAAAACGTAAAGAACAGCTGCTGGATATTGCCATTGAAGTTTTTAGCCAGCGTGGTATTGGTCGTGGTG GTCATGCAGATATTGCAGAAATTGCACAGGTTAGCGTTGCAACCGTGTTTAACTATTTTCCGACCCGTGAAGATCTG GTTGATGATGTTCTGAACAAAGTGGAAAACGAGTTTCACCAGTTCATCAATAACAGCATTAGCCTGGATCTGGATGT TCGTAGCAATCTGAATACCCTGCTGCTGAACATTATTGATAGCGTTCAGACCGGCAACAAATGGATTAAAGTTTGGT TTGAATGGTCAACCAGCACCCGTGATGAAGTTTGGCCTCTGTTTCTGAGCACCCATAGCAATACCAATCAGGTGATC AAAACCATGTTTGAAGAGGGTATTGAACGCAATGAAGTGTGCAATGATCATACACCGGAAAATCTGACCAAAATGCT GCATGGTATTTGCTATAGCGTGTTTATTCAGGCCAATCGTAATAGCAGCAGCGAAGAAATGGAAGAAACCGCAAATT GCTTTCTGAATATGCTGTGCATCTACAAATAACTCGGTACCAAATTCCAGAAAAGACACCCGAAAGGGTGTTTTTTC GTTTTGGTCCGGAGCGTAGAGCTTAGATTCGTTACCAATTGACAAATTTATAAATTGTCAGTATAATGCTAGC
LmrA (RBS-N1) gate	gate	CTGAAGGAGTCAATTAATGTGCTTTTAATTCTGATGAGACGGTGACGTCGAAACTCCCTCTACAAATAATTTTGTTT AATACGCTATGGACTATGTTTTCTGCTATGGACTATGTTTTCACACACGAGATGCCTCGATGAGCTATGGTGATAGC CGTGAAAAAATTCTGAGCGCAGCAACCCGTCTGTTTCAGCTGCAGGGTTATTATGGCACCGGTCTGAATCAGATTAT CAAAGAAAGCGGTGCACCGAAAGGTAGCCTGTATTATCATTTTCCGGGTGGTAAAGAACAGCTGGCAATTGAAGCAG TGAACGAAATGAAAGAATATATCCGCCAGAAAATCGCCGATTGTATGGAAGCATGTACCGATCCGGCAGAAGGTATT CAGGCATTTCTGAAAGAACTGAGCTGTCAGTTTAGCTGTACCGAAGATATTGAAGGTCTGCCGGTTGGTCTGCTGGC AGCAGAAACCAGCCTGAAAAGCGAACCGCTGCGTGAAGCATGTCATGAAGCATATAAAGAATGGGCCAGCGTGTATG AAGAAAAACTGCGTCAGACCGGTTGTAGCGAAAGCCGTGCAAAAGAAGCAAGCACCGTTGTTAATGCAATGATTGAA GGTGGTATTCTGCTGAGCCTGACCGCAAAAAATAGCACACCGCTGCTGCATATTAGCAGCTGTATTCCGGATCTGCT GAAACGTTAATAAGGTTGAAAAATAAAAACGGCGCTAAAAAGCGCCGTTTTTTTTGACGGTGGTACGCTCATTCACT AGGTCTGATTCGTTACCAATTGACAACTGGTGTTCGAATCAAGATAATAGACCAGTCACTATATTT
PhlF (RBS-P1) gate	gate	CTGAAGCGGTCAACGCATGTGCTTTGCGTTCTGATGAGACAGTGATGTCGAAACCGCCTCTACAAATAATTTTGTTT AACTATGGACTATGTTTGAAAGGGAGAAATACTAGATGGCACGTACCCCGAGCCGTAGCAGCATTGGTAGCCTGCGT AGTCCGCATACCCATAAAGCAATTCTGACCAGCACCATTGAAATCCTGAAAGAATGTGGTTATAGCGGTCTGAGCAT TGAAAGCGTTGCACGTCGTGCCGGTGCAAGCAAACCGACCATTTATCGTTGGTGGACCAATAAAGCAGCACTGATTG CCGAAGTGTATGAAAATGAAAGCGAACAGGTGCGTAAATTTCCGGATCTGGGTAGCTTTAAAGCCGATCTGGATTTT CTGCTGCGTAATCTGTGGAAAGTTTGGCGTGAAACCATTTGTGGTGAAGCATTTCGTTGTGTTATTGCAGAAGCACA GCTGGACCCTGCAACCCTGACCCAGCTGAAAGATCAGTTTATGGAACGTCGTCGTGAGATGCCGAAAAAACTGGTTG AAAATGCCATTAGCAATGGTGAACTGCCGAAAGATACCAATCGTGAACTGCTGCTGGATATGATTTTTGGTTTTGT TGGTATCGCCTGCTGACCGAACAGCTGACCGTTGAACAGGATATTGAAGAATTTACCTTCCTGCTGATTAATGGTGT TTGTCCGGGTACACAGCGTTAAGGAAACACAGAAAAAAGCCCGCACCTGACAGTGCGGGCTTTTTTTTTCGACCAAA GGCGACGTACGGTGGAATCTGATTCGTTACCAATTGACATGATACGAAACGTACCGTATCGTTAAGGT
PhlF (RBS-P2) gate	gate	CTGAAGCGGTCAACGCATGTGCTTTGCGTTCTGATGAGACAGTGATGTCGAAACCGCCTCTACAAATAATTTTGTTT AAGGAGCTATGGACTATGTTTGAAAGGCTGAAATACTAGATGGCACGTACCCCGAGCCGTAGCAGCATTGGTAGCCT GCGTAGTCCGCATACCCATAAAGCAATTCTGACCAGCACCATTGAAATCCTGAAAGAATGTGGTTATAGCGGTCTGA GCATTGAAAGCGTTGCACGTCGTGCCGGTGCAAGCAAACCGACCATTTATCGTTGGTGGACCAATAAAGCAGCACTG ATTGCCGAAGTGTATGAAAATGAAAGCGAACAGGTGCGTAAATTTCCGGATCTGGGTAGCTTTAAAGCCGATCTGGA TTTTCTGCTGCGTAATCTGTGGAAAGTTTGGCGTGAAACCATTTGTGGTGAAGCATTTCGTTGTGTTATTGCAGAAG CACAGCTGGACCCTGCAACCCTGACCCAGCTGAAAGATCAGTTTATGGAACGTCGTCGTGAGATGCCGAAAAAACTG GTTGAAAATGCCATTAGCAATGGTGAACTGCCGAAAGATACCAATCGTGAACTGCTGCTGGATATGATTTTTGGTTT TTGTTGGTATCGCCTGCTGACCGAACAGCTGACCGTTGAACAGGATATTGAAGAATTTACCTTCCTGCTGATTAATG GTGTTTGTCCGGGTACACAGCGTTAAGGAAACACAGAAAAAAGCCCGCACCTGACAGTGCGGGCTTTTTTTTTCGAC CAAAGGCGACGTACGGTGGAATCTGATTCGTTACCAATTGACATGATACGAAACGTACCGTATCGTTAAGGT
PhlF (RBS-P3) gate	gate	CTGAAGCGGTCAACGCATGTGCTTTGCGTTCTGATGAGACAGTGATGTCGAAACCGCCTCTACAAATAATTTTGTTT AACTTTACGAGGGCGATCCTATGGCACGTACCCCGAGCCGTAGCAGCATTGGTAGCCTGCGTAGTCCGCATACCCAT AAAGCAATTCTGACCAGCACCATTGAAATCCTGAAAGAATGTGGTTATAGCGGTCTGAGCATTGAAAGCGTTGCACG TCGTGCCGGTGCAAGCAAACCGACCATTTATCGTTGGTGGACCAATAAAGCAGCACTGATTGCCGAAGTGTATGAAA ATGAAAGCGAACAGGTGCGTAAATTTCCGGATCTGGGTAGCTTTAAAGCCGATCTGGATTTTCTGCTGCGTAATCTG TGGAAAGTTTGGCGTGAAACCATTTGTGGTGAAGCATTTCGTTGTGTTATTGCAGAAGCACAGCTGGACCCTGCAAC CCTGACCCAGCTGAAAGATCAGTTTATGGAACGTCGTCGTGAGATGCCGAAAAAACTGGTTGAAAATGCCATTAGCA ATGGTGAACTGCCGAAAGATACCAATCGTGAACTGCTGCTGGATATGATTTTTGGTTTTTGTTGGTATCGCCTGCTG ACCGAACAGCTGACCGTTGAACAGGATATTGAAGAATTTACCTTCCTGCTGATTAATGGTGTTTGTCCGGGTACACA GCGTTAAGGAAACACAGAAAAAAGCCCGCACCTGACAGTGCGGGCTTTTTTTTTCGACCAAAGGCGACGTACGGTGG AATCTGATTCGTTACCAATTGACATGATACGAAACGTACCGTATCGTTAAGGT
PsrA (RBS-R1) gate	gate	CTGAAGCGCTGTCTGTACTTGTATCAGTACACTGACGAGTCCCTAAAGGACGAAACACCGCCTCTACAAATAATTTT GTTTAATTTAATTCGCGGAAGCGCAGAGATAAGGGGTATCATGGCACAGAGCGAAACCGTTGAACGTATTCTGGATG CAGCAGAACAGCTGTTTGCAGAACGTGGTTTTGCAGAAACCAGCCTGCGTCTGATTACCAGCAAAGCCGGTGTTAAT CTGGCAGCAGTGAATTATCATTTTGGCAGCAAAAAAGCACTGATTCAGGCAGTTTTTAGCCGTTTTCTGGGTCCGTT TTGTGCAAGCCTGGAACGTGAACTGGAACGTCGTCAGGCACGTCCGGAACAGAAACCGAGCCTGGAAGAACTGCTGG AAATGCTGGTTGAACAGGCACTGGCAGTTCAGCCTCGTAGCAATAATGATCTGAGCATTTTTATGTGTCTGCTGGGT CTGGCATTTAGCCAGAGCCAGGGTCATCTGCGTCGTTATCTGGAAGATATGTATGGTAAAGTGTTCCGTCGTTATAT GCTGCTGGTTAATGAAGCAGCACCGCGTGTTCCGCCTCTGGAACTGTTTTGGCGTGTTCATTTTATGCTGGGTGCAG CAGCATTTAGCATGAGCGGTATTAAAGCACTGCGTGCAATTGCAGAAACCGATTTTGGTATTAACACCAGCATTGAA CAGGTTATGCGTCTGATGGTTCCGTTTCTGGCAGCAGGTATGCGTGCAGATAGCGGTGTTACCGATGAAGCAATGGC AGCAGCACAGCTGCGTCCGCGTAGCAAAACCAGCACCAGCGCAACCACCGCAAAAGCATAATCCGGCAATTAAAAAA GCGGCTAACCACGCCGCTTTTTTTACGTCTGCATGATCGAACGCTTCAAGGAACAAACGTTTGATTGACAGCTAGCT CAGTCCTAGGTAGAGTGCTAGC
QacR (RBS-Q1)	gate	CTGAAGTCGTCAAGTGCTGTGCTTGCACTTCTGATGAGGCAGTGATGCCGAAACGACCTCTACAAATAATTTTGTTT

gate		AAGTAAGCCATGCCATTGGCTTTTGATAGAGGATAACTACTAGATGAACCTGAAAGATAAAATTCTGGGCGTTGCCA AAGAACTGTTTATCAAAAATGGCTATAACGCAACCACCACCGGTGAAATTGTTAAACTGAGCGAAAGCAGCAAAGGC AATCTGTATTATCACTTTAAAACCAAAGAGAACCTGTTTCTGGAAATCCTGAACATCGAAGAAAGCAAATGGCAAGA GCAGTGGAAAAAAGAACAAATCAAATGCAAAACCAACCGCGAGAAATTCTATCTGTATAATGAACTGAGCCTGACCA CCGAATATTACTATCCGCTGCAGAATGCCATCATCGAGTTTTATACCGAGTACTATAAAACCAACAGCATCAACGAG AAAATGAACAAACTGGAAAACAAATACATCGATGCCTACCACGTGATCTTTAAAGAAGGTAATCTGAACGGCGAATG GTGCATTAATGATGTTAATGCCGTGAGCAAAATTGCAGCAAATGCCGTTAATGGCATTGTTACCTTTACCCATGAGC AGAATATCAACGAACGCATTAAACTGATGAACAAATTCAGCCAGATCTTTCTGAATGGCCTGAGCAAATAAGTCAGT TTCACCTGTTTTACGTAAAAACCCGCTTCGGCGGGTTTTTACTTTTGGGGTATGGAAGCTATACGTTACCAATTGAC AGCTAGCTCAGTCCTACTTTAGTATATAGACCGTGCGATCGGTCTATA
QacR (RBS-Q2) gate	gate	CTGAAGTCGTCAAGTGCTGTGCTTGCACTTCTGATGAGGCAGTGATGCCGAAACGACCTCTACAAATAATTTTGTTT AAGCCATGCCATTGGCTTTTGATAGAGGACAACTACTAGATGAACCTGAAAGATAAAATTCTGGGCGTTGCCAAAGA ACTGTTTATCAAAAATGGCTATAACGCAACCACCACCGGTGAAATTGTTAAACTGAGCGAAAGCAGCAAAGGCAATC TGTATTATCACTTTAAAACCAAAGAGAACCTGTTTCTGGAAATCCTGAACATCGAAGAAAGCAAATGGCAAGAGCAG TGGAAAAAAGAACAAATCAAATGCAAAACCAACCGCGAGAAATTCTATCTGTATAATGAACTGAGCCTGACCACCGA ATATTACTATCCGCTGCAGAATGCCATCATCGAGTTTTATACCGAGTACTATAAAACCAACAGCATCAACGAGAAAA TGAACAAACTGGAAAACAAATACATCGATGCCTACCACGTGATCTTTAAAGAAGGTAATCTGAACGGCGAATGGTGC ATTAATGATGTTAATGCCGTGAGCAAAATTGCAGCAAATGCCGTTAATGGCATTGTTACCTTTACCCATGAGCAGAA TATCAACGAACGCATTAAACTGATGAACAAATTCAGCCAGATCTTTCTGAATGGCCTGAGCAAATAAGTCAGTTTCA CCTGTTTTACGTAAAAACCCGCTTCGGCGGGTTTTTACTTTTGGGGTATGGAAGCTATACGTTACCAATTGACAGCT AGCTCAGTCCTACTTTAGTATATAGACCGTGCGATCGGTCTATA
SrpR (RBS-S1) gate	gate	CTGAAGCGCTCAACGGGTGTGCTTCCCGTTCTGATGAGTCCGTGAGGACGAAAGCGCCTCTACAAATAATTTTGTTT AAGAGTCTATGGACTATGTTTTCACAGAGGAGGTACCAGGATGGCACGTAAAACCGCAGCAGAAGCAGAAGAAACCC GTCAGCGTATTATTGATGCAGCACTGGAAGTTTTTGTTGCACAGGGTGTTAGTGATGCAACCCTGGATCAGATTGCA CGTAAAGCCGGTGTTACCCGTGGTGCAGTTTATTGGCATTTTAATGGTAAACTGGAAGTTCTGCAGGCAGTTCTGGC AAGCCGTCAGCATCCGCTGGAACTGGATTTTACACCGGATCTGGGTATTGAACGTAGCTGGGAAGCAGTTGTTGTTG CAATGCTGGATGCAGTTCATAGTCCGCAGAGCAAACAGTTTAGCGAAATTCTGATTTATCAGGGTCTGGATGAAAGC GGTCTGATTCATAATCGTATGGTTCAGGCAAGCGATCGTTTTCTGCAGTATATTCATCAGGTTCTGCGTCATGCAGT TACCCAGGGTGAACTGCCGATTAATCTGGATCTGCAGACCAGCATTGGTGTTTTTAAAGGTCTGATTACCGGTCTGC TGTATGAAGGTCTGCGTAGCAAAGATCAGCAGGCACAGATTATCAAAGTTGCACTGGGTAGCTTTTGGGCACTGCTG CGTGAACCGCCTCGTTTTCTGCTGTGTGAAGAAGCACAGATTAAACAGGTGAAATCCTTCGAATAATTCAGCCAAAA AACTTAAGACCGCCGGTCTTGTCCACTACCTTGCAGTAATGCGGTGGACAGGATCGGCGGTTTTCTTTTCTCTTCTC AATCTATGATTGGTCCAGATTCGTTACCAATTGACAGCTAGCTCAGTCCTAGGTATATACATACATGCTTGTTTGTT TGTAAAC
SrpR (RBS-S2) gate	gate	CTGAAGCGCTCAACGGGTGTGCTTCCCGTTCTGATGAGTCCGTGAGGACGAAAGCGCCTCTACAAATAATTTTGTTT AAGAGTCTATGGACTATGTTTTCACATATGAGATACCAGGATGGCACGTAAAACCGCAGCAGAAGCAGAAGAAACCC GTCAGCGTATTATTGATGCAGCACTGGAAGTTTTTGTTGCACAGGGTGTTAGTGATGCAACCCTGGATCAGATTGCA CGTAAAGCCGGTGTTACCCGTGGTGCAGTTTATTGGCATTTTAATGGTAAACTGGAAGTTCTGCAGGCAGTTCTGGC AAGCCGTCAGCATCCGCTGGAACTGGATTTTACACCGGATCTGGGTATTGAACGTAGCTGGGAAGCAGTTGTTGTTG CAATGCTGGATGCAGTTCATAGTCCGCAGAGCAAACAGTTTAGCGAAATTCTGATTTATCAGGGTCTGGATGAAAGC GGTCTGATTCATAATCGTATGGTTCAGGCAAGCGATCGTTTTCTGCAGTATATTCATCAGGTTCTGCGTCATGCAGT TACCCAGGGTGAACTGCCGATTAATCTGGATCTGCAGACCAGCATTGGTGTTTTTAAAGGTCTGATTACCGGTCTGC TGTATGAAGGTCTGCGTAGCAAAGATCAGCAGGCACAGATTATCAAAGTTGCACTGGGTAGCTTTTGGGCACTGCTG CGTGAACCGCCTCGTTTTCTGCTGTGTGAAGAAGCACAGATTAAACAGGTGAAATCCTTCGAATAATTCAGCCAAAA AACTTAAGACCGCCGGTCTTGTCCACTACCTTGCAGTAATGCGGTGGACAGGATCGGCGGTTTTCTTTTCTCTTCTC AATCTATGATTGGTCCAGATTCGTTACCAATTGACAGCTAGCTCAGTCCTAGGTATATACATACATGCTTGTTTGTT TGTAAAC
SrpR (RBS-S3) gate	gate	CTGAAGCGCTCAACGGGTGTGCTTCCCGTTCTGATGAGTCCGTGAGGACGAAAGCGCCTCTACAAATAATTTTGTTT AAGAGTCTATGGACTATGTTTTCACAAAGGAAGTACCAGGATGGCACGTAAAACCGCAGCAGAAGCAGAAGAAACCC GTCAGCGTATTATTGATGCAGCACTGGAAGTTTTTGTTGCACAGGGTGTTAGTGATGCAACCCTGGATCAGATTGCA CGTAAAGCCGGTGTTACCCGTGGTGCAGTTTATTGGCATTTTAATGGTAAACTGGAAGTTCTGCAGGCAGTTCTGGC AAGCCGTCAGCATCCGCTGGAACTGGATTTTACACCGGATCTGGGTATTGAACGTAGCTGGGAAGCAGTTGTTGTTG CAATGCTGGATGCAGTTCATAGTCCGCAGAGCAAACAGTTTAGCGAAATTCTGATTTATCAGGGTCTGGATGAAAGC GGTCTGATTCATAATCGTATGGTTCAGGCAAGCGATCGTTTTCTGCAGTATATTCATCAGGTTCTGCGTCATGCAGT TACCCAGGGTGAACTGCCGATTAATCTGGATCTGCAGACCAGCATTGGTGTTTTTAAAGGTCTGATTACCGGTCTGC TGTATGAAGGTCTGCGTAGCAAAGATCAGCAGGCACAGATTATCAAAGTTGCACTGGGTAGCTTTTGGGCACTGCTG CGTGAACCGCCTCGTTTTCTGCTGTGTGAAGAAGCACAGATTAAACAGGTGAAATCCTTCGAATAATTCAGCCAAAA AACTTAAGACCGCCGGTCTTGTCCACTACCTTGCAGTAATGCGGTGGACAGGATCGGCGGTTTTCTTTTCTCTTCTC AATCTATGATTGGTCCAGATTCGTTACCAATTGACAGCTAGCTCAGTCCTAGGTATATACATACATGCTTGTTTGTT TGTAAAC
SrpR (RBS-S4) gate	gate	CTGAAGCGCTCAACGGGTGTGCTTCCCGTTCTGATGAGTCCGTGAGGACGAAAGCGCCTCTACAAATAATTTTGTTT AACTATGGACTATGTTTTCACACAGGAAATACCAGGATGGCACGTAAAACCGCAGCAGAAGCAGAAGAAACCCGTCA GCGTATTATTGATGCAGCACTGGAAGTTTTTGTTGCACAGGGTGTTAGTGATGCAACCCTGGATCAGATTGCACGTA AAGCCGGTGTTACCCGTGGTGCAGTTTATTGGCATTTTAATGGTAAACTGGAAGTTCTGCAGGCAGTTCTGGCAAGC CGTCAGCATCCGCTGGAACTGGATTTTACACCGGATCTGGGTATTGAACGTAGCTGGGAAGCAGTTGTTGTTGCAAT GCTGGATGCAGTTCATAGTCCGCAGAGCAAACAGTTTAGCGAAATTCTGATTTATCAGGGTCTGGATGAAAGCGGTC TGATTCATAATCGTATGGTTCAGGCAAGCGATCGTTTTCTGCAGTATATTCATCAGGTTCTGCGTCATGCAGTTACC CAGGGTGAACTGCCGATTAATCTGGATCTGCAGACCAGCATTGGTGTTTTTAAAGGTCTGATTACCGGTCTGCTGTA TGAAGGTCTGCGTAGCAAAGATCAGCAGGCACAGATTATCAAAGTTGCACTGGGTAGCTTTTGGGCACTGCTGCGTG AACCGCCTCGTTTTCTGCTGTGTGAAGAAGCACAGATTAAACAGGTGAAATCCTTCGAATAATTCAGCCAAAAAACT TAAGACCGCCGGTCTTGTCCACTACCTTGCAGTAATGCGGTGGACAGGATCGGCGGTTTTCTTTTCTCTTCTCAA TATGATTGGTCCAGATTCGTTACCAATTGACAGCTAGCTCAGTCCTAGGTATATACATACATGCTTGTTTGTTTGTA AAC
AraC* transcription unit	sensor module	TCACCATATATCAAGTTTACGGCTAGCTCAGTCCTAGGTACTATGCTAGCTACTAGAGAAAGAGGAGAAATACTAGA TGGCTGAAGCGCAAAATGATCCCCTGCTGCCGGGATACTCGTTTAATGCCCATCTGGTGGCGGGTTTAACGCCGATT GAGGCCAACGGTTATCTCGATTTTTTTATCGACCGACCGCTGGGAATGAAAGGTTATATTCTCAATCTCACCATTCG CGGTCAGGGGGTGGTGAAAAATCAGGGACGAGAATTTGTTTGCCGACCGGGTGATATTTTGCTGTTCCCGCCAGGAG AGATTCATCACTACGGTCGTCATCCGGAGGCTCGCGAATGGTATCACCAGTGGGTTTACTTTCGTCCGCGCGCCTAC TGGCATGAATGGCTTAACTGGCCGTCAATATTTGCCAATACGGGGTTCTTTCGCCCGGATGAAGCGCACCAGCCGCA TTTCAGCGACCTGTTTGGGCAAATCATTAACGCCGGGCAAGGGGAAGGGCGCTATTCGGAGCTGCTGGCGATAAATC TGCTTGAGCAATTGTTACTGCGGCGCATGGAAGCGATTAACGAGTCGCTCCATCCACCGATGGATAATCGGGTACGC

LacI and TetR sensor transcription unit module	GAGGCTTGTCAGTACATCAGCGATCACCTGGCAGACAGCAATTTTGATATCGCCAGCGTCGCACAGCATGTTTGCTT GTCGCCGTCGCGTCTGTCACATCTTTTCCGCCAGCAGTTAGGGATTAGCGTCTTAAGCTGGCGCGAGGACCAACGTA TCAGCCAGGCGAAGCTGCTTTTGAGCACCACCCGGATGCCTATCGCCACCGTCGGTCGCAATGTTGGTTTTGACGAT CAACTCTATTTCTCGCGGGTATTTAAAAAATGCACCGGGGCCAGCCCGAGCGAGTTCCGTGCCGGTTAATAACCAAT TATTGAAGGCCGCTAACGCGGCCTTTTTTTGTTTCTGGTCTCCC GCGGCGCGCCATCGAATGGCGCAAAACCTTTCGCGGTATGGCATGATAGCGCCCGGAAGAGAGTCAATTCAGGGTGG TGAATATGAAACCAGTAACGTTATACGATGTCGCAGAGTATGCCGGTGTCTCTTATCAGACCGTTTCCCGCGTGGTG AACCAGGCCAGCCACGTTTCTGCGAAAACGCGGGAAAAAGTGGAAGCGGCGATGGCGGAGCTGAATTACATTCCCAA CCGCGTGGCACAACAACTGGCGGGCAAACAGTCGTTGCTGATTGGCGTTGCCACCTCCAGTCTGGCCCTGCACGCGC CGTCGCAAATTGTCGCGGCGATTAAATCTCGCGCCGATCAACTGGGTGCCAGCGTGGTGGTGTCGATGGTAGAACGA AGCGGCGTCGAAGCCTGTAAAGCGGCGGTGCACAATCTTCTCGCGCAACGCGTCAGTGGGCTGATCATTAACTATCC GCTGGATGACCAGGATGCCATTGCTGTGGAAGCTGCCTGCACTAATGTTCCGGCGTTATTTCTTGATGTCTCTGACC AGACACCCATCAACAGTATTATTTTCTCCCATGAGGACGGTACGCGACTGGGCGTGGAGCATCTGGTCGCATTGGGT CACCAGCAAATCGCGCTGTTAGCGGGCCCATTAAGTTCTGTCTCGGCGCGTCTGCGTCTGGCTGGCTGGCATAAATA TCTCACTCGCAATCAAATTCAGCCGATAGCGGAACGGGAAGGCGACTGGAGTGCCATGTCCGGTTTTCAACAAACCA TGCAAATGCTGAATGAGGGCATCGTTCCCACTGCGATGCTGGTTGCCAACGATCAGATGGCGCTGGGCGCAATGCGC GCCATTACCGAGTCCGGGCTGCGCGTTGGTGCGGATATCTCGGTAGTGGGATACGACGATACCGAAGATAGCTCATG TTATATCCCGCCGTTAACCACCATCAAACAGGATTTTCGCCTGCTGGGGCAAACCAGCGTGGACCGCTTGCTGCAAC TCTCTCAGGGCCAGGCGGTGAAGGGCAATCAGCTGTTGCCAGTCTCACTGGTGAAAAGAAAAACCACCCTGGCGCCC AATACGCAAACCGCCTCTCCCCGCGCGTTGGCCGATTCATTAATGCAGCTGGCACGACAGGTTTCCCGACTGGAAAG CGGGCAGTGATAATCCAGGAGGAAAAAAATGTCCAGATTAGATAAAAGTAAAGTGATTAACAGCGCATTAGAGCTGC TTAATGAGGTCGGAATCGAAGGTTTAACAACCCGTAAACTCGCCCAGAAGCTAGGTGTAGAGCAGCCTACATTGTAT TGGCATGTAAAAAATAAGCGGGCTTTGCTCGACGCCTTAGCCATTGAGATGTTAGATAGGCACCATACTCACTTTTG CCCTTTAGAAGGGGAAAGCTGGCAAGATTTTTTACGTAATAACGCTAAAAGTTTTAGATGTGCTTTACTAAGTCATC GCGATGGAGCAAAAGTACATTTAGGTACACGGCCTACAGAAAAACAGTATGAAACTCTCGAAAATCAATTAGCCTTT TTATGCCAACAAGGTTTTTCACTAGAGAATGCATTATATGCACTCAGCGCTGTGGGGCATTTTACTTTAGGTTGCGT ATTGGAAGATCAAGAGCATCAAGTCGCTAAAGAAGAAAGGGAAACACCTACTACTGATAGTATGCCGCCATTATTAC GACAAGCTATCGAATTATTTGATCACCAAGGTGCAGAGCCAGCCTTCTTATTCGGCCTTGAATTGATCATATGCGGA TTAGAAAAACAACTTAAATGTGAAAGTGGGTCCTAATAA

a. DNA sequence colors correspond to ribozyme insulators (blue), RBSs (green), protein coding sequences (red), terminators (black), output promoters (orange), and sensor transcription units (purple).

Table 4-9: Genetic part sequences

Part name	Type	DNA sequence[a]
BBa_J23101	promoter	tttacagctagctcagtcctaggtattatgctagc
BBa_J23105	promoter	tttacggctagctcagtcctaggtactatgctagc
P_{LacI}	promoter	gcggcgcgccatcgaatggcgcaaaacctttcgcggtatggcatgatagcgcccggaagagagtcaattcag ggtggtgaat
P_{Tac}	promoter (Boer *et al*, 1983)	aacgatcgttggctgtgttgacaattaatcatcggctcgtataatgtgtggaattgtgagcgctcacaatt
P_{Tet}	promoter (Stanton *et al*, 2014)	tactccaccgttggctttttttccctatcagtgatagagattgacatccctatcagtgatagagataatgagc ac
P_{BAD}	promoter (Moon *et al*, 2011)	acttttcatactcccgccattcagagaagaaaccaattgtccatattgcatcagacattgccgtcactgcgt cttttactggctcttctcgctaaccaaaccggtaaccccgcttattaaaagcattctgtaacaaagcgggac caaagccatgacaaaaacgcgtaacaaaagtgtctataatcacggcagaaaagtccacattgattatttgca cggcgtcacactttgctatgccatagcatttttatccataagattagcggatcctacctgacgcttttatc gcaactctctactgtttctccatacccgtttttttgggctagc
P_{AmeR}	promoter (Stanton *et al*, 2014)	tcgtcactagagggcgatagtgacaaacttgacaactcatcacttcctaggtataatgctagc
P_{AmtR}	promoter (Stanton *et al*, 2014)	cttgtccaaccaaatgattcgttaccaattgacagtttctatcgatctatagataatgctagc
P_{BetI}	promoter (Stanton *et al*, 2014)	agcgcgggtgagagggattcgttaccaattgacaattgattggacgttcaatataatgctagc
P_{BM3R1}	promoter (Stanton *et al*, 2014)	aatccgcgtgataggtctgattcgttaccaattgacggaatgaacgttcattccgataatgctagc
P_{HlyIIR}	promoter (Stanton *et al*, 2014)	accaggaatctgaacgattcgttaccaattgacatatttaaaattcttgtttaaaatgctagc
P_{IcaRA}	promoter (Stanton *et al*, 2014)	gtcaactcataagattctgattcgttaccaattgacaattcacctacctttcgttaggttaggttgt
P_{LitR}	promoter (Stanton *et al*, 2014)	cgagcgtagagcttagattcgttaccaattgacaaatttataaattgtcagtataatgctagc
P_{LmrA}	promoter (Stanton *et al*, 2014)	cgctcattcactaggtctgattcgttaccaattgacaactggtggtcgaatcaagataatagaccagtcact atattt
P_{PhlF}	promoter (Stanton *et al*, 2014)	cgacgtacggtggaatctgattcgttaccaattgacatgatacgaaacgtaccgtatcgttaaggt
P_{PsrA}	promoter (Stanton *et al*, 2014)	tgatcgaacgcttcaaggaacaaacgtttgattgacagctagctcagtcctaggtataatgctagc
P_{QacR}	promoter (Stanton *et al*, 2014)	ggtatggaagctatacgttaccaattgacagctagctcagtcctactttagtatatagaccgtgcgatcggt ctata
P_{SrpR}	promoter (Stanton *et al*, 2014)	tctatgattggtccagattcgttaccaattgacagctagctcagtcctaggtatatacatacatgcttgttt gtttgtaaac
RiboJ	insulator (Lou *et al*, 2012)	agctgtcaccggatgtgctttccggtctgatgagtccgtgaggacgaaacagcctctacaaataatttttgtt taa
RiboJ54	Insulator	aggggtcagttgatgtgctttcaactctgatgagtcagtgatgacgaaaccccctctacaaataatttttgtt taa
BydvJ	Insulator	agggtgtctcaaggtgcgtaccttgactgatgagtccgaaaggacgaaacacccctctacaaataatttttgt ttaa
RiboJ57	Insulator	agaagtcaattaatgtgcttttaattctgatgagtcggtgacgacgaaacttcctctacaaataatttttgtt taa
SarJ	insulator (Lou *et al*, 2012)	gactgtcgccggatgtgtatccgacctgacgatggcccaaaagggccgaaacagtcctctacaaataatttt gtttaa
RiboJ51	Insulator	agtagtcaccggctgtgcttgccggtctgatgagcctgtgaaggcgaaactacctctacaaataatttttgtt taa

ElvJ	Insulator	agccccatagggtggtgtgtaccacccctgatgagtccaaaaggacgaaatggggcctctacaaataatttttgtttaa
PlmJ	insulator (Lou *et al*, 2012)	agtcataagtctgggctaagcccactgatgagtcgctgaaatgcgacgaaacttatgacctctacaaataattttgtttaa
RiboJ64	insulator	aggagtcaattaatgtgcttttaattctgatgagacggtgacgtcgaaactccctctacaaataatttttgttaa
RiboJ53	insulator	agcggtcaacgcatgtgctttgcgttctgatgagacagtgatgtcgaaaccgcctctacaaataatttttgttaa
ScmJ	insulator	agcgctgtctgtacttgtatcagtacactgacgagtccctaaaggacgaaacaccgcctctacaaataatttgtttaa
RiboJ60	insulator	ctgaagtcgtcaagtgctgtgcttgcacttctgatgaggcagtgatgccgaaacgacctctacaaataatttgtttaa
RiboJ10	insulator	agcgctcaacgggtgtgcttcccgttctgatgagtccgtgaggacgaaagcgcctctacaaataatttttgttaa
yfp	gene (Cormack *et al*, 1996)	atggtgagcaagggcgaggagctgttcaccggggtggtgcccatcctggtcgagctggacggcgacgtaaacggcccacaagttcagcgtgtccggcgagggcgagggcgatgccacctacggcaagctgaccctgaagttcatctgcaccacaggcaagctgcccgtgccctggcccaccctcgtgaccaccttcggctacggcctgcaatgtttccgcccgctaccccgaccacatgaagctgcacgacttcttcaagtccgccatgcccgaaggctacgtccaggagcgcaccatcttcttcaaggacgacggcaactacaagacccgcgccgaggtgaagttcgagggcgacaccctggtgaaccgcatcgagctgaagggcatcgacttcaaggaggacggcaacatcctggggcacaagctggagtacaactacaacagccacaacgtctatatcatggccgacaagcagaagaacggcatcaaggtgaacttcaagatccgccacaacatcgaggacggcagcgtgcagctcgccgaccactaccagcagaacaccccaatcggcgacggccccgtgctgctgcccgacaaccactaccttagctaccagtccgccctgagcaaagaccccaacgagaagcgcgatcacatggtcctgctggagttcgtgaccgccgccgggatcactctcggcatggacgagctgtacaagtaaatgaaaccagtaacgttatacgatgtcgcagagtatgccggtgtctcttatcagaccgtttcccgcgtggtgaaccaggccagccacgtttctgcgaaaacgcgggaaaaagtggaagcggcgatggcggagctgaattacattcccaaccgcgtggcacaacaactggcgggcaaacagtcgttgctgattggcgttgccacctccagtctggccctgcacgcgccgtcgcaaattgtcgcggcgattaaatctcgcgccgatcaactgggtgccagcgtggtggtgtcgatggtagaacgaagcggcgtcgaagcctgtaaagcggcggtgcacaatcttctcgcgcaacgcgtcagtgggctgatcattaactatccgctggatgaccaggatgccattgctgtggaagctgcctgcactaatgttccggcgttatttcttgatgtctctgaccagacacccatcaacagtattattttctcccatgaggacggtacgcgactgggcgtggagcatctggtcgcattgggtcaccagcaaatcgcgctgttagcgggcccattaagttctgtctcggcgcgtctgcgtctggctggctggcataaatatctcactcgcaatcaaattcagccgatagcggaacgggaaggcgactggagtgccatgtccggttttcaacaaaccatgcaaatgctgaatgagggcatcgttcccactgcgatgctggttgccaacgatcagatggcgctgggcgcaatgcgcgccattaccgagtccgggctgcgcgttggtgcggatatctcggtagtgggatacgacgataccgaagatagctcatgttatatcccgccgttaacccacatcaaacaggattttcgcctgctggggcaaacccagcgtggaccgcttgctgcaactctctcagggccaggcggtgaagggcaatcagctgttgccagtctcactggtgaaaagaaaaaccaccctggcgcccaatacgcaaaccgcctctccccgcgcgttggccgattcattaatgcagctggcacgacaggtttcccgactggaaagcgggcagtgataa
lacI	gene (Stanton *et al*, 2014)	atgtccagattagataaaagtaaagtgattaacagcgcattagagctgcttaatgaggtcggaatcgaaggttaacaacccgtaaactcgcccagaagctaggtgtagagcagcctacattgtattggcatgtaaaaaataagcgggctttgctcgacgccttagccattgagatgttagataggcaccatactcacttttgccctttagaaggggaaagctggcaagatttttacgtaataacgctaaaagttttagatgtgctttactaagtcatcgcgatggagcaaaagtacatttaggtacacggcctacagaaaaacagtatgaaactctcgaaaatcaattagccttttttgccaacaaggttttcactagagaatgcattatatgcactcagcgctgtggggcattttactttaggttgcgtattggaagatcaagagcatcaagtcgctaaagaagaaagggaaacacctactactgatagtatgccgccattattacgacaagctatcgaattatttgatcaccaaggtgcagagccagccttcttattcggccttgaattgatcatatgcggattagaaaaacaacttaaatgtgaaagtgggtcctaa
*araC**	gene (Lee *et al*, 2007)	atggctgaagcgcaaaatgatcccctgctgccgggatactcgtttaatgcccatctggtggcgggtttaacgccgattgaggccaacggttatctcgatttttttatcgaccgaccgctgggaatgaaaggttatattctcaatctcaccattcgcggtcaggggtggtgaaaaatcagggacgagaatttgtttgccgaccgggtgatattttgctgttcccgccaggagagattcatcactacggtcgtcatccggaggctcgcgaatggtatcaccagtgggtttactttcgtccgcgcgcctactggcatgaatggcttaactggccgtcaatatttgccaatacggggttctttcgccccggatgaagcgcaccagccgcatttcagcgacctgtttgggcaaatcattaacgccgggcaaggggaagggcgctattcggagctgctggcgataaatctgcttgagcaattgttactgcggcgcatggaagcgattaacgagtcgctccatccaccgatggataatcgggtacgcgaggcttgtcagtacatcagcgatcacctggcagacagcaattttgatatcgccagcgtcgcacagcatgtttgcttgtcgccgtcgcgtctgtcacatcttttccgccagcagttagggattagcgtcttaagctggcgcgaggaccaacgtatcagccaggcgaagctgcttttgagcaccacccggatgcctatcgccaccgtcggtcgcaatgttggttttgacgatcaactctatttctcgcgggtatttaaaaaatgcaccggggccagcccgagcgagttccgtgccggttgtgaagaaaaagtgaatgatgtagccgtcaagttgtcataa
ameR	gene (Stanton *et al*, 2014)	atgaacaaaaccattgatcaggtgcgtaaaggtgatcgtaaaagcgatctgccggttcgtcgtcgtccgcgtcgtagtgccgaagaaacccgtcgtgatattctggcaaaagccgaagaactgtttcgtgaacgtggttttaatgcagttgccattgcagatattgcaagcgcactgaatatgagtccggcaaatgtgtttaaacattttagcagcaaaaacgcactggttgatgcaattggtttttggtcagattggtgtttttgaacgtcagatttgtccgctggataaaagccatgcaccgctggatcgtctgcgtcatctggcacgtaatctgatggaacagcatcatcaggatcattcaaacacatacgggtttttattcagatcctgatgaccgccaaacaggatatgaaatgtggcgattattacaaaagcgtgattgcaaaactgctggccgaaattattcgtgatggtgttgaagcaggtctgtatattgcaaccgatattccggttctggcagaaaccgttctgcatgcactgaccagcgttattcatccggttctgattgcacaagaagatattggtaatctggcaacccgttgtgatcagctggttgatctgattgatgcaggtctgcgtaatccgctggcaaaataa
amtR	gene (Stanton *et al*, 2014)	atggcaggcgcagttggtcgtccgcgtcgtagtgcaccgcgtcgtgcaggtaaaaatccgcgtgaagaaattctggatgcaagcgcagaactgtttacccgtcagggttttgcaaccaccagtacccatcagattgcagatgcagttggtattcgtcaggcaagcctgtattatcattttccgagcaaaaccgaaatctttctgaccctgctgaaaagcaccgttgaaccgagcaccgttctggcagaagatctgagcaccctggatgcaggtccggaaatgcgtctgtgggcaattgttgcaagcgaagttcgtctgctgctgagcaccaaatggaatgttggtcgtctgtatcagctgccgattgttggtagcgaagaatttgcagaatatcatagccagcgtgaagcactgaccaatgtttttcgtgatctggcaaccgaaattgttggtgatgatccgcgtgcagaactgccgtttcatattaccatgagcgttattgaaatgcgtcgcaatgatggtaaaattccgagtccgctgagcgcagatagcctgccggaaaccgcaattatgctg

betI	gene (Stanton *et al*, 2014)	gcagatgcaagcctggcagttctgggtgcaccgctgcctgcagatcgtgttgaaaaaaccctggaactgatt aaacaggcagatgcaaaataa atgccgaaactgggtatgcagagcattcgtcgtcgtcagctgattgatgcaaccctggaagcaattaatgaa gttggtatgcatgatgcaaccattgcacagattgcacgtcgtgccggtgttagcaccggtattattagccat tatttccgcgataaaaacggtctgctggaagcaaccatgcgtgatattaccagccagctgcgtgatgcagtt ctgaatcgtctgcatgcactgccgcagggtagcgcagaacagcgtctgcaggcaattgttggtggtaatttt gatgaaacccaggttagcagcgcagcaatgaaagcatggctggcattttgggcaagcagcatgcatcagccg atgctgtatcgtctgcagcaggttagcagtcgtcgtctgctgagcaatctggttagcgaatttcgtcgtgaa ctgcctcgtgaacaggcacaagaggcaggttatggtctggcagcactgattgatggtctgtggctgcgtgca gcactgagcggtaaaccgctggataaaacccgtgcaaatagcctgacccgtcattttatcacccagcatctg ccgaccgattaa
bm3R1	gene (Stanton *et al*, 2014)	atggaaagcaccccgaccaaacagaaagcaatttttagcgcaagcctgctgctgtttgcagaacgtggtttt gatgcaaccaccatgccgatgattgcagaaaatgcaaaagttggtgcaggcaccatttatcgctatttcaaa aacaaagaaagcctggtgaacgaactgtttcagcagcatgttaatgaatttctgcagtgtattgaaagcggt ctggcaaatgaacgtgatggttatcgtgatggctttcatcacatttttgaaggtatggtgacctttaccaaa aatcatccgcgtgcactgggttttatcaaaacccatagccagggcacctttctgaccgaagaaagccgtctg gcatatcagaaaactggttgaatttgtgtgcaccttttttcgtgaaggtcagaaacagggtgtgattcgtaat ctgccggaaaatgcactgattgcaattctgtttggcagctttatggaagtgtatgaaatgatcgagaacgat tatctgagcctgaccgatgaactgctgaccggtgttgaagaaagcctgtgggcagcactgagccgtcagagc taa
hlyIIR	gene (Stanton *et al*, 2014)	atgaaatacatcctgtttgaggtgtgcgaaatgggtaaaagccgtgaacagaccatggaaaatattctgaaa gcagccaaaaagaaattcggcgaacgtggttatgaaggcaccagcattcaagaaattaccaaagaagccaaa gttaacgttgcaatggccagctattactttaatggcaaagagaacctgtactacgaggtgttcaaaaaatac ggtctggcaaatgaactgccgaactttctggaaaaaaaccagtttaatccgattaatgccctgcgtgaatat ctgaccgtttttaccacccacattaaagaaaatccggaaattggcaccctggcctatgaagaaattatcaaa gaaagcgcacgcctggaaaaaatcaaaccgtatttttatcggcagcttcgaacagctgaaagaaattctgcaa gagggtgaaaaacagggtgtgtttcacttttttagcatcaaccataccatccattggattaccagcattgtt ctgtttccgaaattcaaaaaattcatcgatagcctgggtccgaatgaaaccaatgatacccaatcatgaatgg atgccggaagatctggttagccgtattattagcgcactgaccgataaaccgaacatttaa
icaRA	gene (Stanton *et al*, 2014)	gtgaaagacaaaattatcgataacgccatcaccctgtttagcgaaaaaggttatgacggcaccaccctggat gatattgcaaaaagcgtgaacatcaaaaaagccagcctgtattatcacttgatagcaaaaaaagcatctac gagcagagcgttaaatgctgtttcgattatctgaacaacatcatcatgatgaaccagaacaaaagcaactat agcatcgatgccctgtatcagtttctgtttgagttcatcttcgatatcgaggaacgctatattcgtatgtat gttcagctgagcaacacaccggaagaattttcaggtaacatttatggccagatccaggatctgaatcagagc ctgagcaaagaaatcgccaaattctatgacgaaagcaaaatcaaaatgaccaaagaggacttccagaatctg attctgctgtttctggaaagctggtatctgaaagccagctttagccagaaatttggtgcagttgaagaaagc aaaagccagtttaaagatgaggtttatagcctgctgaacatctttctgaagaaataa
litR	gene (Stanton *et al*, 2014)	atggataccattcagaaacgtccgcgtacccgtctgagtccggaaaaacgtaaagaacagctgctggatatt gccattgaagtttttagccagcgtggtattggtcgtggtggtcatgcagatattgcagaaattgcacaggtt agcgttgcaaccgtgtttaactattttccgacccgtgaagatctggttgatgatgttctgaacaaagtggaa aacgagtttcaccagttcatcaataacagcattagcctggatctggatgttcgtagcaatctgaataccctg ctgctgaacattattgatagcgttcagaccggcaacaaatggattaaagtttggtttgaatggtcaaccagc acccgtgatgaagtttggcctctgtttctgagcacccatagcaataccaatcaggtgatcaaaaccatgttt gaagagggtattgaacgcaatgaagtgtgcaatgatcatacaccggaaaatctgaccaaaatgctgcatggt atttgctatagcgtgtttattcaggccaatcgtaatagcagcagcgaagaaatggaagaaaccgcaaattgc tttctgaatatgctgtgcatctacaaataa
lmrA	gene (Stanton *et al*, 2014)	atgagctatggtgatagccgtgaaaaaattctgagcgcagcaacccgtctgtttcagctgcagggttattat ggcaccggtctgaatcagattatcaaagaaagcggtgcaccgaaaggtagcctgtattatcattttccgggt ggtaaagaacagctggcaattgaagcagtgaacgaaatgaaagaatatatccgccagaaaatcgccgattgt atggaagcatgtaccgatccggcagaaggtattcaggcatttctgaaagaactgagctgtcagtttagctgt accgaagatattgaaggtctgccggttggtctgctggcagcagaaaccagcctgaaaagcgaaccgctgcgt gaagcatgtcatgaagcatataaagaatgggccagcgtgtatgaagaaaaactgcgtcagaccggttgtagc gaaagccgtgcaaaagaagcaagcaccgttgttaatgcaatgattgaaggtggtattctgctgagcctgacc gcaaaaaatagcacaccgctgctgcatattagcagctgtattccggatctgctgaaacgttaa
phlF	gene (Stanton *et al*, 2014)	atggcacgtaccccgagccgtagcagcattggtagcctgcgtagtccgcatacccataaagcaattctgacc agcaccattgaaatcctgaaagaatgtggttatagcggtctgagcattgaaagcgttgcacgtcgtgccggt gcaagcaaaccgaccatttatcgttggtggaccaataaagcagcactgattgccgaagtgtatgaaaatgaa agcgaacaggtgcgtaaatttccggatctgggtagctttaaagccgatctggattttctgctgcgtaatctg tggaaagtttggcgtgaaaccatttgtggtgaagcatttcgttgtgttattgcagaagcacagctggaccct gcaaccctgacccagctgaaagatcagtttatggaacgtcgtcgtgagatgccgaaaaaactggttgaaaat gccattagcaatggtgaactgccgaaagataccaatcgtgaactgctgctggatatgatttttggttttgt tggtatcgcctgctgaccgaacagctgaccgttgaacaggatattgaagaatttaccttcctgctgattaat ggtgtttgtccgggtacacagcgttaa
psrA	gene (Stanton *et al*, 2014)	atggcacagagcgaaaccgttgaacgtattctggatgcagcagaacagctgtttgcagaacgtggttttgca gaaaccagcctgcgtctgattaccagcaaagccggtgttaatctggcagcagtgaattatcattttggcagc aaaaaagcactgattcaggcagtttttagccgttttctgggtccgttttgtgcaagcctggaacgtgaactg gaacgtcgtcaggcacgtccggaacagaaaccgagcctggaagaactgctggaaatgctggttgaacaggca ctggcagttcagcctcgtagcaataatgatctgagcatttttatgcgtctgctgggtctggcatttagccag agccagggtcatctgcgtcgttatctggaagatatgtatggtaaagtgttccgtcgttatatgctgctggtt aatgaagcagcaccgcgtgttccgcctctggaactgttttggcgtgttcattttatgctgggtgcagcagca tttagcatgagcggtattaaagcactgcgtgcaattgcagaaaccgattttggtattaacaccagcattgaa caggttatgcgtctgatggttccgtttctggcagcaggtatgcgtgcagatagcggtgttaccgatgaagca atggcagcagcacagctgcgtccgcgtagcaaaaccagcaccagcgcaaccaccgcaaaagcataa
qacR	gene (Stanton *et al*, 2014)	atgaacctgaaagataaaattctgggcgttgccaaagaactgtttatcaaaaatggctataacgcaaccacc accggtgaaattgttaaactgagcgaaagcagcaaaggcaatctgtattatcactttaaaaccaaagagaac ctgtttctggaaatcctgaacatcgaagaaagcaaatggcaagagcagtggaaaaaagaacaaatcaaatgc aaaaccaaccgcgagaaattctatctgtataatgaactgagcctgaccaccgaatattactatccgctgcag aatgccatcatcgagttttataccgagtactataaaaccaacagcatcaacgagaaaatgaacaaactggaa aacaaatacatcgatgcctaccacgtgatctttaaagaaggtaatctgaacggcgaatggtgcattaatgat gttaatgccgtgagcaaaattgcagcaaatgccgttaatggcattgttacctttacccatgagcagaatatc aacgaacgcattaaactgatgaacaaattcagccagatctttctgaatggcctgagcaaataa

srpR	gene (Stanton *et al*, 2014)	atggcacgtaaaaccgcagcagaagcagaagaaaccccgtcagcgtattattgatgcagcactggaagttttt gttgcacagggtgttagtgatgcaaccctggatcagattgcacgtaaagccggtgttacccgtggtgcagtt tattggcattttaatggtaaactggaagttctgcaggcagttctggcaagccgtcagcatccgctggaactg gattttacaccggatctgggtattgaacgtagctgggaagcagttgttgttgcaatgctggatgcagttcat agtccgcagagcaaacagtttagcgaaattctgatttatcagggtctggatgaaagcggtctgattcataat cgtatggttcaggcaagcgatcgttttctgcagtatattcatcaggttctgcgtcatgcagttacccagggt gaactgccgattaatctggatctgcagaccagcattggtgtttttaaaggtctgattaccggtctgctgtat gaaggtctgcgtagcaaagatcagcaggcacagattatcaaagttgcactgggtagcttttgggcactgctg cgtgaaccgcctcgttttctgctgtgtgaagaagcacagattaaacaggtgaaatccttcgaataa
L3S2P21	terminator (Chen *et al*, 2013)	ctcggtaccaaattccagaaaagaggcctcccgaaaggggggccttttttcgttttggtcc
L3S3P31[b]	terminator (Chen *et al*, 2013)	ccaattattgaacaccctaacgggtgtttttttttttttggtctacc
L3S2P55	terminator (Chen *et al*, 2013)	ctcggtaccaaagacgaacaataagacgctgaaaagcgtctttttcgttttggtcc
L3S3P11[b]	terminator (Chen *et al*, 2013)	ccaattattgaacacccttcggggtgtttttttgtttctggtctacc
L3S2P11	terminator (Chen *et al*, 2013)	ctcggtaccaaattccagaaaagagacgctttcgagcgtctttttcgttttggtcc
ECK120033736	terminator (Chen *et al*, 2013)	aacgcatgagaaagccccccggaagatcaccttccgggggctttttttattgcgc
ECK120015170	terminator (Chen *et al*, 2013)	acaattttcgaaaaaacccgcttcggcgggtttttttatagctaaaa
L3S2P24	terminator (Chen *et al*, 2013)	ctcggtaccaaattccagaaaagacacccgaaagggtgtttttttcgttttggtcc
ECK120010876	terminator (Chen *et al*, 2013)	taaggttgaaaaataaaaacggcgctaaaaagcgccgtttttttttgacggtggta
ECK120033737	terminator (Chen *et al*, 2013)	ggaaacacagaaaaaagcccgcacctgacagtgcgggctttttttttcgaccaaagg
ECK120015440	terminator (Chen *et al*, 2013)	tccggcaattaaaaaagcggctaaccacgccgctttttttacgtctgca
ECK120010818	terminator (Chen *et al*, 2013)	gtcagtttcacctgttttacgtaaaaacccgcttcggcgggtttttactttgg
ECK120029600	terminator (Chen *et al*, 2013)	ttgagaagagaaaagaaaaccgccgatcctgtccaccgcattactgcaaggtagtggacaagaccggcggtc ttaagttttttggctgaa

a. Underline indicates the upstream promoter spacer.
b. The "C" at nucleotide 45 from was mutated to "A" to eliminate a BsaI recognition site.

Table 4-10: FISH probe sequences

Probe name	Size (nt)	%GC	DNA sequence
eYFP 1	20	60	TCCTCGCCCTTGCTCACCAT
eYFP 2	20	50	GCTGAACTTGTGGCCGTTTA
eYFP 3	20	65	CAGGGTCAGCTTGCCGTAGG
eYFP 4	20	55	TGCCTGTGGTGCAGATGAAC
eYFP 5	20	60	GTAGCCGAAGGTGGTCACGA
eYFP 6	20	60	TAGCGGGCGAAGCATTGCAG
eYFP 7	20	60	GTGCAGCTTCATGTGGTCGG
eYFP 8	20	55	GCATGGCGGACTTGAAGAAG
eYFP 9	20	65	CGCTCCTGGACGTAGCCTTC
eYFP 10	20	50	GTCGTCCTTGAAGAAGATGG
eYFP 11	20	65	CGGCGCGGGTCTTGTAGTTG
eYFP 12	20	60	GTGTCGCCCTCGAACTTCAC
eYFP 13	20	55	TTCAGCTCGATGCGGTTCAC
eYFP 14	20	55	CGTCCTCCTTGAAGTCGATG
eYFP 15	20	55	AGCTTGTGCCCCAGGATGTT
eYFP 16	20	45	GTGGCTGTTGTAGTTGTACT
eYFP 17	20	50	TGTCGGCCATGATATAGACG
eYFP 18	20	50	ACCTTGATGCCGTTCTTCTG
eYFP 19	20	50	TGTTGTGGCGGATCTTGAAG
eYFP 20	20	55	TGTTCTGCTGGTAGTGGTCG
eYFP 21	20	55	CTAAGGTAGTGGTTGTCGGG
eYFP 22	20	65	CTTTGCTCAGGGCGGACTGG
eYFP 23	20	60	TGATCGCGCTTCTCGTTGGG
eYFP 24	20	60	CACGAACTCCAGCAGGACCA
eYFP 25	20	45	TACTTGTACAGCTCGTCCAT

5 Appendix: Methods

5.1 Methods for genomic mining of prokaryotic repressors for orthogonal logic gates

5.1.1 Strains and media

E. coli strain DH10B (F⁻ mcrA Δ(mrr-hsdRMS-mcrBC) Φ80lacZΔM15ΔlacX74 recA1 endA1 araΔ139 Δ(ara, leu)7697 galU galK λ-rpsL (StrR) nupG) was used for all experiments, except in logic gate measurement where DH5α (fhuA2 lac(del)U169 phoA glnV44 Φ80′ lacZ(del)M15 gyrA96 recA1 relA1 endA1 thi-1 hsdR17) was used and in protein expression and purification where BL21(DE3)pLysS (F⁻ ompT gal dcm lon hsdS$_B$ (r_B^- m_B^-) λ(DE3) pLysS(cm^R)) was used. Cells were grown in LB Miller Broth, M9 minimal medium ((6.8 g/L Na_2HPO_4, 3 g/L KH_2PO_4, 0.5 g/LNaCl, 1 g/L NH_4Cl; Sigma), 2 mM $MgSO_4$, 100 μM $CaCl_2$, 0.4% glucose, 0.2% casamino acids, 340 mg/L thiamine (vitamin B1)) or Super Optimal Broth (SOB). Ampicillin (50 μg/ml), kanamycin (25 μg/ml) and/or chloramphenicol (37 μg/ml) were used where appropriate. Isopropyl β-D-1-thiogalactopyranoside (IPTG) or 3OC6-*N*-(β-ketocaproyl)-L-homoserine lactone (HSL) inducers were used as inducers for the various repressor constructs. Each of the newly constructed plasmids was made by the one-step isothermal DNA assembly method or inverse PCR (see below). In all cases, YFP (Cormack *et al*, 1996) was used as the reporter.

5.1.2 Codon optimization and gene synthesis

Repressor coding sequences were optimized for production in *E. coli*, chloroplasts and *Bacillus subtilis* using multiparameter gene optimization methods (Fath *et al*, 2011). Optimized sequences were synthesized by GeneArt, are contained within a pET21a-derived plasmid (where each repressor contains an N-terminal His_6 tag) and were sequence verified.

5.1.3 Calculation of REU

REUs were calculated through use of a strain harboring pJ23101-YFP (Fig. 2-16), which contains a constitutive promoter (BBa_J23101) followed by a 5′ UTR (BBa_B0032) and YFP. A plasmid containing the reference standard was transformed into DH10B cells, resulting in the *in vivo* reference strain. The reference strain was grown under conditions identical to an experimental strain (in this work, strains harboring NOT gates or genetic circuits). The mean reference fluorescence of three replicates minus white cell fluorescence was set to 1 REU. The mean fluorescence from experimental strains was divided by the reference standard to obtain their output in REU.

5.1.4 Repressor expression and purification

Plasmids encoding the synthesized repressor were transformed into BL21(DE3)pLysS cells. Single colonies were selected for by growth on LB Miller medium containing ampicillin andchloramphenicol. Cells were inoculated in SOB containing ampicillin and chloramphenicol and grown overnight at 37 °C. The

following morning, cells were diluted back to an OD_{600} of 0.1 in 50 mL fresh SOB medium without antibiotics and were induced using 1 mM IPTG once cells reached an OD_{600} between 0.6–0.8. Cells were grown for 6 h at 37 °C at 250 r.p.m. in a shaking incubator and spun down at 4,000 r.p.m. at 4 °C, the supernatant was discarded and pellets were stored at −80 °C.

Cell pellets were resuspended on ice in 5 mL binding buffer (0.5 M NaCl, 20 mM HEPES (pH 8), 5 mM imidazole, 50 mM phenylalanine, 50 mM isoleucine, 10% glycerol and 0.1 μM DTT) containing protease inhibitors and 0.1% Igepal detergent. Resuspended cells were lysed by sonication at room temperature with a setting of 20% duty cycle and 0.1-s pulses, using two 20-s cycles, followed by a final 10-s cycle, with icing in between. Lysates were clarified by centrifugation at 4 °C at 10,000 r.p.m. for 30 min. Clarified extracts were then filtered and applied to 0.5 ml Nickel resin (that had been equilibrated with binding buffer for 30 min at room temperature using a Nutator), and the resin was collected using a gravity flow column. The repressor-bound resin was washed with 5 ml binding buffer and 10 ml wash buffer (0.5 M NaCl, 20 mM HEPES (pH 8), 25 mMimidazole, 50 mM phenylalanine, 50 mM isoleucine, 10% glycerol and 0.1 μM DTT) and was eluted in 0.5 ml elution buffer (0.5 M NaCl, 20 mM HEPES (pH 8), 0.5 M imidazole, 50 mM phenylalanine, 50 mM isoleucine, 10% glycerol and 0.1 μM DTT). Binding buffer (3.5 mL) was added to the eluate, and it was applied to a 15-ml microconcentrator and spun down at 4,000*g* for 20 min at 4 °C. The concentrated eluate was stored on ice, the concentration was determined using Bradford reagent, distributed into approximately 150-μg aliquots, flash frozen in liquid nitrogen and stored at −80 °C.

5.1.5 Library screening to identify repressible promoters

A single-letter, degenerate code was defined for each position within an array-identified motif on the basis of MEME-identified consensus sequences (Stanton *et al*,

2014) to generate an operator motif (Fig. 2-2a). Degenerate oligonucleotides representing the resulting operator motif were designed to insert the operator motif into a strong, constitutive synthetic BioBricks BBA_J23119 standard promoter (Kelly *et al*, 2009b). Operator motifs were inserted, in various positions, between or around the −35 and −10 elements of the BBA_J23119 promoter using inverse PCR. Specifically, vector sequences were PCR amplified using Phusion DNA polymerase (NEB) along with the degenerate, operator motif–containing oligonucleotides. The resulting product was run on an agarose gel, extracted and digested with DpnI. The blunt-ended, DpnI-digested product was phosphorylated (T4 Polynucleotide Kinase) and ligated (T4 DNA ligase) in a single reaction at room temperature, transformed into chemically competent DH10B cells and plated on selective LB medium. Libraries containing individual sequence variants of an operator motif were screened for fluorescence using a blue light transilluminator to ensure that the resulting promoters containing operator motifs retained activity. Those operator motif variants that promoted fluorescence were also screened for repression by transformation together with the cognate repressor (Fig. 2-2b). Briefly, DH10B cells containing a repressor plasmid expressing the cognate repressor were made competent using the Z-competent cell kit (Zymo Research). Plasmid DNA was prepared, in 96-well format, from individual fluorescent operator motif variants. The resulting plasmid DNA was transformed into Z-competent DH10B cells containing the cognate repressor. Overnights were made from cells containing the fluorescent operator motif reporters only and from cells containing the reporter co-transformed with the cognate repressor. Overnight culture (1 μl) was diluted into 200 μl 1× PBS, and flow cytometry was carried out to quantify fluorescence in the presence and absence of repressor for the LitR and McbR repressors (and then assessed by eye for all other repressor or reporter screens using a blue light transilluminator). The promoter variant associated

with the largest difference in fluorescence in the absence and presence of repressor was selected to be the cognate promoter for a given repressor. Promoters were also constructed using the previously identified operator sequences for the AmtR, BetI, BM3R1, HapR, HlyIIR, IcaR(A), LmrA, PhlF, SmcR, and TetR repressors listed in Table 2-6. Individual operator sequences were inserted into the BioBricks BBA_J23119 standard promoter in various positions surrounding either or both the −35 and −10 elements. Those promoters that retained constitutive activity were screened for repression by their cognate repressor using the methods outlined above.

5.1.6 Construction and tuning of repressor expression

The reverse engineering feature of the RBS calculator (Salis *et al*, 2009) (https://salis.psu.edu/software/reverse/) was used to identify a weak and a strong RBS sequence for each individual repressor, with the following settings: free energy model, v1.1; organism (16s rRNA), *Escherichia coli* str. K12 substr. DH10B ACCTCCTTA. Specifically, RBS sequences were reverse engineered using the following four RBS sequences to obtain their translation initiation rate for an individual repressor: B0034 GAAAGAGGAGAAATACTAGATG, rbs1 TCACACAGGAAACCGGTTCGATG, rbs2 TCACACAGGAAAGGCCTCGATG or rbs3 TCACACAGGACGGCCGGATG. Successive single-base substitutions were made until RBSs of the desired strength were obtained. This strategy was used to identify both a weak and strong RBS for a given repressor. The two respective strength RBS sequences were aligned and combined into a single, degenerate RBS (except in the case of TetR, where a single RBS was used; Table 2-4). The sequence content based on the alignment and relative translation initiation strength information for each sequence variant were taken into account when assigning degenerate codes to each position within an RBS. Oligonucleotides were designed to encode the degenerate RBS,

which was inserted upstream of the repressor coding sequence, to generate an RBS library. The repressor ORF, reporter fragment and vector backbone were PCR amplified using Phusion DNA polymerase (NEB) and fused into a single vector using Gibson assembly to generate a single response function vector (Fig. 2-14). The entire 20-μl Gibson reaction was transformed into chemically competent DH10B cells and plated onto LB-selective medium containing ampicillin. Single colonies were inoculated and grown for 6 h at 37 °C in SOB medium containing ampicillin, in 96-well format, in the presence and absence of 1 mM IPTG. Fluorescence was quantified using flow cytometry to deduce the fold change of the induced and uninduced clones as outlined above. Those clones demonstrating high fluorescence in the absence of inducer and low fluorescence in the presence of inducer were selected. The RBSs that give rise to the highest fold change are shown in Table 2-5 and were used for the orthogonality measurements and in the construction of NOT gates.

5.1.7 Measurement of orthogonality matrix

Competent *E. coli* DH10B cells were made using the Z-competent cell kit (Zymo Research) that contained individual NOT gates (pRF-; Fig. 2-14), which serve as the reporter. Cells were transformed with an additional vector containing the repressor (pOrtho-; Fig. 2-17), whose expression was controlled by the HSL-inducible P_{Lux} promoter (Urbanowski *et al*, 2004), in all possible combinations. Specifically, 10–50 ng of plasmid DNA were incubated with 10–20 μl Z-competent cells on ice for 10 min in a 96-well plate. SOC broth (150 μl) was added, and cells were outgrown at 37 °C for 1 h with shaking at 1,000 r.p.m. in an ELMI shaker (ELMI Ltd) and plated on LB agar. Plated cells were inoculated into LB containing ampicillin and kanamycin, and grown overnight at 37 °C with shaking at 1,000 r.p.m. The following morning, stationary-phase cultures were diluted 1:200 into LB-containing antibiotics and grown

in a 96-well shaking incubator for 4 h at 37 °C with shaking at 1,000 r.p.m. The cultures were diluted 1:100 into LB-containing antibiotics and 20 μMHSL, except in the cases of HapR, Orf2, ScbR and SmcR, where 2 μM, 20 nM, 200 nM and 200 nM HSL were used, respectively, owing to toxicity. The induced cells were grown at 37 °C for 6 h with shaking at 1,000 r.p.m., and then fluorescence was measured by diluting the induced culture 1:40 in PBS and carrying out flow cytometry as described below. Induction assays were run in triplicate for each repressor-reporter combination, and a control plasmid for the orthogonality assays (that corresponds to the pOrtho vector lacking a repressor coding sequence) was used as a normalization control to signify the unrepressed state for individual reporters. The data represent the average of three replicates collected on different days.

5.1.8 Measurement of NOT gate response functions

E. coli DH10B cultures containing NOT gate constructs were grown overnight for 16 h in liquid SOB medium containing ampicillin. The cells were grown in a 96-well shaking incubator at 37 °C and 1,000 r.p.m. The next day, stationary-phase cultures were diluted 1:200 into antibiotic-containing minimal M9 medium supplemented with glucose and grown in the 96-well shaking incubator for 3 h using the same shaking and temperature settings as the overnight growth. Subsequently, the cultures were diluted 1:700 into antibiotic-containing minimal M9 medium supplemented withglucose containing different concentrations of IPTG and then grown for 6 h in the shaking incubator to obtain sufficient exponential-phase cell density for cytometric analysis. The IPTG concentrations used were 0 μM, 5 μM, 10 μM, 20 μM, 30 μM, 40 μM, 50 μM, 70 μM, 100 μM, 150 μM, 200 μM and 1,000 μM. At the end of the final growth period, cultures were diluted 1:5 into PBS. Strains containing the plasmids for the measurement of input promoter activity (Fig. 2-15) and the conversion to REU

(Fig. 2-16) were grown and measured concurrently with these strains. Flow cytometry was performed as described below. The data represent the average of three replicates collected on different days, and error bars correspond to the s.d. between these measurements.

5.1.9 Measurement of genetic circuits

E. coli DH5α cultures containing the plasmids encoding the circuits were grown overnight in liquid SOB medium containing kanamycin and ampicillin (for the 2-plasmid NAND circuit) or kanamycin(for the 1-plasmid AND circuit) in a 96-well incubator at 37 °C shaking at 1,000 r.p.m. After 16 h of growth, cultures were diluted 1:200 into LB medium with antibiotics and grown in the 96-well shaking incubator for 3 h using the same shaking and temperature settings as the overnight growth. Subsequently, the cultures were diluted 1:700 into LB medium with inducers and then grown for 6 h in the shaking incubator. The inducer concentrations used are: 1 mM IPTG, 20 μMHSL and 100 ng/mL aTc. Cultures were diluted 1:20 into PBS, and fluorescence was measured by flow cytometry as described below.

5.1.10 Cytometry measurement experiments

At the end of growth, cultures were diluted into PBS with 2 mg/mL kanamycin to arrest cell growth. Cells were analyzed by flow cytometry, using a BD Biosciences LSRII flow cytometer with a blue (488 nm) laser. An injection volume of 10 μL and the flow rate of 0.5 μL/s were used.

5.1.11 Cytometry data analysis

Cells were analyzed using FlowJo (TreeStar Inc., Ashland, OR), and populations were gated on the forward scatter area from 100 to 50,000, and on the side scatter area from 50 to 50,000. The gated population consisted of thousands of cells. The fluorescence geometric mean of the gated population was calculated, and the mean autofluorescence of a 'white cell' control sample was subtracted from the experimental sample's mean. Fold change is calculated by dividing the mean fluorescence of the ON state by the mean fluorescence of the OFF state (Table 2-5). The data represent the average of three replicates collected on different days, and error bars correspond to the s.d. between these measurements.

5.1.12 Cellular growth and toxicity assay

Repressor toxicity was assessed by comparing the growth of induced, NOT gate–containing cells to the growth of uninduced cells (Fig. 2-6). Cells were grown identically to the response function assay. A 100-μL culture aliquot was placed into an optically clear-bottom 96-well plate, and absorbance was measured at 600 nm using a BioTek Synergy H1 Hybrid Microplate Reader. Repressors were considered toxic under conditions where cell growth is less than 75% of the uninduced culture growth. The final nontoxic induction point occurs at 200 μM, 150 μM, 100 μM, 70 μM, 70 μM and 70 μM IPTG for ButR, TarA, HapR, ScbR, SmcR and Orf2, respectively. If the threshold for toxicity is redefined to a different number, a plot of the maximum induction levels (REU) for a given toxicity threshold is provided (Fig. 2-7). The data represent the average of three replicates collected on different days, and error bars correspond to the s.d. between these measurements.

5.2 Methods for multi-input CRISPR/Cas genetic circuits that interface host regulatory networks

5.2.1 Strains and media

E. coli DH10b (F– *mcr*A Δ(*mrr-hsd*RMS-*mcr*BC) Φ80*lacZ*ΔM15 Δ*lac*X74 *rec*A1 *end*A1 *ara*D139 Δ(*ara leu*) 7697 *galU galK rpsL nupG* λ–)(Durfee *et al*, 2008) was used for cloning (New England Biolabs, MA, C3019). *E.* coli K-12 MG1655* (F- λ- ilvG- rfb-50 rph-1 Δ(araCBAD) Δ(LacI))(Blattner *et al*, 1997) was used for measurement experiments. Cells were grown in LB Miller broth (Difco, MI, 90003-350) for overnight growth and cloning, and MOPS EZ Rich Defined Medium (Teknova, CA, M2105) with 0.4% glycerol carbon source for measurement experiments. Ampicillin (100 μg/ml), kanamycin (50 μg/ml), and spectinomycin sulfate (50 μg/mL) were used to maintain plasmids. Arabinose (Sigma Aldrich, MO, A3256), 2,4-diacetylphloroglucinol (Santa Cruz Biotechnology, TX, CAS 2161-86-6) and anhydrotetracycline (aTc) (Sigma Aldrich, MO, 37919) were used as chemical inducers. The fluorescent protein reporters YFP(Cormack *et al*, 1996) and mRFP1(Campbell *et al*, 2002) were measured with cytometry to determine gene expression.

5.2.2 Flow cytometry analysis

Fluorescent protein production was measured using the LSRII Fortessa flow cytometer (BD Biosciences, San Jose, CA). Between 10^4 and 10^5 events were collected for subsequent analysis with the software tool FlowJo v10 (TreeStar, Inc., Ashland, OR). From the resulting fluorescence histograms for YFP and RFP, we calculated the geometric means of each sample, and then corrected for cellular autofluorescence by subtracting the geometric mean of a strain harboring only pAN-P_{Tet}-dCas9 that was grown in an identical manner.

5.2.3 Computational design of sgRNA-promoter pairs

DNA sequences of 13 nucleotides in length were generated using the Random DNA Sequence Generator (www.faculty.ucr.edu/~mmaduro/random.htm), with a GC content probability parameter of 0.5. The resulting sequences were flanked by forward and reverse PAMs and the -35 and -10 sigma factor binding sites to generate sgRNA repressible promoters. If the forward sequence for the promoter contained any stretches with more than three guanine nucleotides, the promoter design was discarded due to the difficulty in synthesizing oligos with G-quadruplexes (Burge *et al*, 2006). Next, the 12 nucleotides adjacent to either the forward or reverse PAM were searched for in the genome of *E. coli* strain K-12 substrain MG1655 (taxid: 511145) using Standard Nucleotide BLAST (blast.st-va.ncbi.nlm.nih.gov/Blast.cgi?PROGRAM=blastn) (Altschul *et al*, 1990) to search for somewhat similar sequences (blastn). The following parameters were used: short queries was enabled; expect threshold = 10; word size = 11; match/mismatch scores = 2,-3; gap costs = existence: 5, extension: 2; and low complexity regions unmasked. Of the ten sgRNAs designed, no 12nt seed regions had complete homology to a PAM-adjacent locus in the *E. coli* genome. If the resulting 20 nucleotide sgRNAs had GC content less than 35% or greater than 80%, the sequence was discarded and redesigned.

5.2.4 Induction endpoint assays

E. coli MG1655* cells were transformed with three plasmids encoding (1) inducible dCas9, (2) one or more sgRNAs, and (3) a fluorescent reporter. Cells were plated on LB agar plates with appropriate antibiotics. Transformed colonies were inoculated into MOPS EZ Rich Defined Medium with 0.4% glycerol and appropriate antibiotics, and

were then grown overnight in V-bottom 96-well plates (Nunc, Roskilde, Denmark, 249952) in an ELMI Digital Thermos Microplates shaker incubator (Elmi Ltd, Riga, Latvia) at 1000 RPM and 37°C. The next day, cultures were diluted 180-fold into EZ Rich Medium with antibiotics, and grown with the same shaking incubator parameters for three hours. At three hours, cells were diluted 700-fold into EZ Rich Medium with antibiotics and inducers. The cells were grown using the same shaking incubator parameters for six hours. For cytometry measurements, 40 µL of the cell culture was added to 160 µL of phosphate buffered saline with 0.5 mg/mL kanamycin to arrest cell growth. The cells were placed in a 4°C refrigerator for one hour to allow the fluorophores to mature prior to cytometry analysis.

5.2.5 Toxicity measurements

For dCas9 toxicity measurements, cells were grown identically to the induction endpoint assays until the second dilution after the three hour growth. From here, the cultures were diluted 360-fold into EZ Rich Defined Medium with 0.4% glycerol with antibiotics and inducers in 2 mL 96-deep well plates (USA Scientific, FL, 1896-2000) and were grown for six hours in a Multitron Pro shaker-incubator (In Vitro Technologies, VIC, Australia) at 37°C and 1000 RPM. At this point, cultures were transferred to 1 cm optical cuvettes and the cultures optical density at 600 nm was measured for the cell cultures, after a blank measurement with EZ Rich Medium. For sgRNA toxicity measurements, cells were grown identically to the induction endpoint assays.

5.2.6 Induction timecourse assays

Timecourse experiments were performed identically to endpoint assays, with the exception that cells were grown in 14 mL round-bottom polystyrene culture tubes (VWR, PA, 60819-524). After the second dilution into inducers, culture samples were taken every 30 minutes for seven hours and were added to phosphate buffered saline with 0.5 mg/mL kanamycin for subsequent cytometry analysis.

5.2.7 Inducer concentrations

For dCas9 toxicity measurements, arabinose was added to 2 mM, and aTc was added to the following final concentrations (ng/mL): 0, 0.3125, 0.625, 1.25, 5, and 10. For sgRNA response curve experiments, aTc was added to 0.625 ng/mL and arabinose was added to the following final concentrations (mM): 0, 0.03125, 0.0625, 0.125, 0.25, and 0.5. For timecourse and orthogonality experiments, aTc was added to 0.625 ng/mL and arabinose was added to 2 mM. For digital genetic circuit measurements and lambdaphage infection experiments, inducers were either absent or added to the following final concentrations: 0.625 ng/mL aTc, 2 mM arabinose, and 25 µM 2,4-diacetylphloroglucinol. For the intermediate genetic circuit measurements, aTc was added to 0.625 ng/mL; arabinose was added to the following final concentrations (mM): 0, .00391, .00781, .0156, .0313, .0625, 0.125, 0.25, 0.5, 1, and 2; 2,4-diacetylphloroglucinol was added to the following final concentrations (µM): 0, 0.0244, 0.0488, 0.0977, 0.391, 0.781, 1.56, 3.13, 6.25, 12.5, and 25.

5.2.8 Lambdaphage infection assay

E. coli MG1655* cells were grown from colonies overnight in EZ Rich Defined Media with antibiotics. The next day, cultures were diluted 180-fold into EZ Rich

Medium with 0.4% glycerol and antibiotics, and grown at 37°C shaking at 250 RPM in culture tubes for three hours. Next, cells were diluted 180-fold once again into five different tubes of 4 mL of EZ Rich Medium with antibiotics and containing the five different inducer conditions. These cells were grown for six hours using the same shaking incubator conditions in culture tubes. After six hours, each culture was pelleted at 4000 g and then resuspended in 100 µL of 10 mM MgSO4. Half of each resuspension (50 µL) was diluted into 950 µL of 10 mM MgSO4 and the optical density at 600 nanometers was measured. The remaining 50 µL of each cell resuspension were diluted to an OD600 of 3.0 in 10 mM MgSO4. Next, 1 µL of lambdaphage was added to 100 µL of each cell resuspension, vortexed lightly, and then allowed to incubate at 37°C for one hour. Finally, all 100 µL of cells were plated onto 1.5% agar LB Miller plate and allowed to grow overnight at 37°C. The next day, phage plaques were counted on each plate.

5.3 Methods for genetic circuit design automation

5.3.1 Circuit induction and measurement guide.

A step-by-step guide to transforming, inducing, and measuring circuits is provided below. The 0xF6 circuit is used as a specific example.

Co-transform the 0xF6 plasmid (pAN3938) and the P_{PhlF}-P_{AmtR}-YFP output plasmid (pAN4044) into chemically competent NEB 10-beta (New England Biolabs, MA, C3019). Add 1 µl of each purified plasmid to 50 µl of thawed chemically competent cells. Incubate mixture on ice for half an hour, and then heat shock at 42°C for 30 seconds. Incubate on ice for 2 more minutes, and then add 1 mL room temperature SOC media. Incubate at 37°C for one hour. Plate serial dilutions of recovered cells on LB agar plates with 50 µg/mL kanamycin (Gold Biotechnology,

MO, K-120-5) and 50 µg/mL spectinomycin (Gold Biotechnology, MO, S-140-5). Grow plates at 37°C overnight.

The day after transformation, pick single colonies and inoculate into 200 µl of M9 glucose with antibiotics in a V-bottom 96-well plate (Nunc, Roskilde, Denmark, 249952). M9 glucose media is composed of M9 media salts (6.78 g/L Na_2HPO_4, 3 g/L KH_2PO_4, 1 g/L NH_4Cl, 0.5 g/L NaCl; Sigma-Aldrich, MO, M6030), 0.34 g/L thiamine hydrochloride (Sigma-Aldrich, MO, T4625), 0.4% D-glucose (Sigma-Aldrich, MO, G8270), 0.2% Casamino acids (Acros, NJ, AC61204-5000), 2 mM $MgSO_4$ (Sigma-Aldrich, MO, 230391), and 0.1 mM $CaCl_2$ (Sigma-Aldrich, MO, 449709). Antibiotic concentrations in M9 glucose media are 50 µg/mL kanamycin and 50 µg/mL spectinomycin.

Grow single colonies in V-bottom 96-well plates overnight (16 hours) at 37°C and 1000 RPM in an ELMI Digital Thermos Microplates shaker incubator (Elmi Ltd, Riga, Latvia).

The next day, dilute the overnight cultures 178-fold by adding 15 µL of culture into 185 µL of M9 glucose media, and then 15 µL of that dilution into 185 µL of M9 glucose media with 50 µg/mL kanamycin and 50 µg/mL spectinomycin in a V-bottom 96-well plate.

Grow the diluted cultures in an ELMI shaker incubator at 37 °C and 1000 RPM for three hours.

Dilute the cultures by adding 15 µL of culture into 185 µL of M9 glucose media.

Take 3 µL aliquots of that dilution and distribute into eight wells with 145 µL of inducer-containing M9 glucose media with 50 µg/mL kanamycin and 50 µg/mL spectinomycin in a V-bottom 96-well plate. The eight wells correspond to the inducer conditions (—/—/—), (—/—/+), (—/+/—), (—/+/+), (+/—/—), (+/—/+), (+/+/—), and (+/+/+). Each — or + corresponds to the absence or presence of 5

mM L-arabinose (Sigma-Aldrich, MO, A3256), 2 ng/mL aTc (anhydrotetracycline hydrochloride; Sigma-Aldrich, MO, 37919), and 1 mM IPTG (isopropyl β-D-1-thiogalactopyranoside; Sigma-Aldrich, MO, I6758).

Grow the cultures containing inducer in an ELMI shaker incubator at 37°C and 1000 RPM for five hours. Note: at the end of five hours, the cultures should still be in exponential-growth phase, and not in stationary phase.

Aliquot 10 µL of cell culture into 190 µL of phosphate buffered saline (PBS) containing 2 mg/mL kanamycin to arrest protein production and cell growth. Incubate this mixture for one hour at room temperature.

Measure the fluorescence of >1000 cells per inducer condition using flow cytometry (see Flow cytometry analysis).

5.3.2 Circuits library measurement and time-courses.

For 2-input circuits, the protocol was as above except that the four inducer combinations were the presence or absence of 1 mM IPTG and 2 ng/mL aTc. For all 3-input circuits, the protocol was identical to above. For time-course measurements, we took an initial sample for cytometry before dilution into inducer-containing medium. Next, we performed 10 sets of parallel circuit inductions (all eight states) for a given circuit, and removed 50 µL from a consecutive set every 30 minutes for cytometry analysis.

5.3.3 Circuit analysis.

After fluorescence measurement by flow cytometry (see Flow cytometry analysis), the medians of the YFP histograms were calculated and converted to RPU. Individual states were deemed "successful" if the experimental distributions were near the

predicted distributions, as measured by eye. Because the output plasmid is lower copy than the gate measurement plasmid, the amount of YFP produced is lower. We measured P_{Tac} induction of the YFP RPU cassette on the circuit plasmid and output plasmid, and observed a 2.5-fold decrease in the amount of YFP produced from the output plasmid. We used the conversion factor to downscale all predicted output values; it is stored in the "genetic_locations" collection of the Eco1C1G1T1 UCF, as the "unit_conversion" attribute in "output_module_location".

5.3.4 Strain, media, and inducers.

E. coli NEB 10-beta *Δ(ara-leu) 7697 araD139 fhuA ΔlacX74 galK16 galE15 e14-ϕ80dlacZΔM15 recA1 relA1 endA1 nupG rpsL* (Str^R) *rph spoT1 Δ(mrr-hsdRMS-mcrBC), a DH10B derviative*(Durfee *et al*, 2008), was used for cloning and measurements (New England Biolabs, MA, C3019). Cells were grown in LB Miller broth (Difco, MI, 90003-350) for harvesting plasmid. Cells were grown M9 glucose media for measurements. M9 glucose media was composed of M9 media salts (6.78 g/L Na_2HPO_4, 3 g/L KH_2PO_4, 1 g/L NH_4Cl, 0.5 g/L NaCl; Sigma-Aldrich, MO, M6030), 0.34 g/L thiamine hydrochloride (Sigma-Aldrich, MO, T4625), 0.4% D-glucose (Sigma-Aldrich, MO, G8270), 0.2% Casamino acids (Acros, NJ, AC61204-5000), 2 mM $MgSO_4$ (Sigma-Aldrich, MO, 230391), and 0.1 mM $CaCl_2$ (Sigma-Aldrich, MO, 449709). Chemical inducers used as inputs for sensor promoters were isopropyl β-D-1-thiogalactopyranoside (IPTG; Sigma-Aldrich, MO, I6758), anhydrotetracycline hydrochloride (aTc; Sigma-Aldrich, MO, 37919), and L-arabinose (Sigma-Aldrich, MO, A3256). Antibiotics used to select for the presence of plasmids were 100 μg/ml ampicillin (Gold Biotechnology, MO, A-301-5), 50 μg/ml kanamycin (Gold Biotechnology, MO, K-120-5), and 50 μg/ml spectinomycin (Gold Biotechnology, MO, S-140-5).

5.3.5 Design and assembly of 2-input circuits.

The circuits in Figure 3 based on non-insulated gates were constructed by using the DNA sequences previously described(Stanton *et al*, 2014) and patterning the promoters in front of the repressors consistent with the desired circuit diagram (Figure S5). The insulated circuits in Figure 3 were constructed automatically, but using software developed in MATLAB that was the precursor to Cello. In this program, all possible gate assignments were exhaustively checked and their performance scored as min(ON)/max(OFF). Promoter activities (in RPU) were propagated through a circuit using the response functions of the insulated gates. The activity of tandem promoters was taken as the sum of the activities of the individual promoters. An early version of roadblocking rules was included to disallow certain promoters in downstream positions.

5.3.6 Ribozyme cleavage assay.

For *in vitro* quantification of cleavage, we performed the Rapid Amplification of cDNA End (RACE) assay(5´ RACE System for Rapid Amplification of cDNA Ends). For each sample, one colony was inoculated into 1 mL LB Miller broth with 20 µg/mL chloramphenicol and then grown for 16 hours at 37 °C shaking at 250 rpm. The next day, the liquid culture was diluted 1000-fold into M9 glucose media (1 µL into 1 mL) with chloramphenicol and 1 mM of IPTG, and then grown until an OD_{600} of 0.2. Cells were then harvested and total mRNA was extracted using the RiboPure bacteria kit (Ambion, CA, AM1924). To ligate a unique RNA adaptor to the 5'-end of the mRNA, three enzymatic steps were performed sequentially. First, 15 µg of purified mRNA was treated with 10 U of T4 polynucleotide kinase (NEB, MA, M0201S) in a total volume of 50 µl 1X T4 DNA ligase buffer and incubated for one hour at 37 °C to phosphorylate

the end of the cleaved mRNA. Second, the mRNA was purified by phenol/chloroform extraction (USB, CA, 75831) and ethanol precipitation (VWR, PA, V1016) and then treated with 10 U of Tobacco acid pyrophosphatase (TAP, Epicenter, T19250) in 50 µl of 1X TAP buffer for two hours at 37 °C to convert the triphosphate of uncleaved mRNA to monophosphate. The treated mRNA was phenol/chloroform extracted and ethanol precipitated once again. Next, 1 µL of 100 µM RNA adaptor (5'-GAGGACUCGAGCUCAAGC-3') was ligated to all extracted mRNA using 15 U of T4 RNA ligase (Ambion, CA, AM2140) in 30 µl of 1X RNA ligase buffer for 2 hours at 37 °C. The mRNA was phenol/chloroform extracted and ethanol precipitated one final time. Next, we reverse transcribed the mRNA using 200 U of SuperScript III (Invitrogen, CA, 18080-044) with a gene specific primer (GSP1, 5'-ATCCCCATCTTGTCTGCGACAG-3') in 20 µl of 1X SuperScript III buffer. In each previous enzymatic step, 20 U of RNasin (Promega, WI, N2611) or 40 U of RNaseOUT (Invitrogen, CA, 100000840) was added to inhibit RNase activity. After reverse transcription, 2 U of RNase H (Invitrogen, CA, 18021-014) was added directly to the 20 µl volume to remove RNA from any RNA/DNA duplex. The cDNA was used as a template for PCR amplification using two primers (the first being the DNA version of the RNA adapter: 5'-GAGGACTCGAGCTCAAGC-3', and the second being the gene-specific primer named GSP2: 5'-TCCTGGGATAAGCCAAGTTC-3'). We performed the PCR using 5 pmol each primer, 2 µl of cDNA template, and Phusion Hi-Fi DNA polymerase (NEB, MA, M0530L) with a 58 °C annealing temperature and 10 second elongation time for 29 cycles. The resulting PCR product comprises multiple sized bands that correspond to cleaved and uncleaved mRNA fragments. These were separated by gel electrophoresis on a 15% acrylamide gel (Bio-Rad, CA, 456-5053) for 1 hour at 100 V. The band corresponding to the cleaved mRNA was excised and placed into 50 µl of water, allowing the DNA to diffuse into the water over 24 hours. This

aqueous DNA solution was used as template for a second PCR (performed identically to above). This PCR product was submitted for Sanger sequencing using primer GSP2. To quantitate ribozyme cleavage efficiency, the 15% acrylamide gel with PCR products separated by gel electrophoresis was imaged using a ChemiDoc MP (Biorad, CA, 170-8280). The band intensity of each fragment was integrated using ImageJ 1.47v (National Institute of Health, MD, http://imagej.nih.gov/ij/). The "rectangular selection tool" was used to select the region surrounding both the cleaved and uncleaved bands between 150 bp and 250 bp. Using the "gel analysis tools", band intensity was plotted, and background was subtracted to obtain a single value corresponding to the intensity of the cleaved and uncleaved band. The intensity of cleaved band was divided by the total sum intensity of both bands to obtain the fraction of cleaved mRNA.

5.3.7 In vivo ribozyme insulation assay.

The four ribozyme-insulator constructs (Figure S3) were transformed into separate aliquots of *E. coli* NEB 10-beta (New England Biolabs, MA, C3019). One colony from each transformant was inoculated into 1 mL of LB with 20 µg/mL chloramphenicol in a culture tube and grown for 16 hours at 37 °C shaking at 250 rpm. The next day, the culture was diluted 178-fold (two serial dilutions of 15 µL into 185 µL) into M9 glucose media with chloramphenicol in a V-bottom 96-well plate (Nunc, Roskilde, Denmark, 249952) and grown for three hours at 37 °C shaking at 1000 rpm in an ELMI Digital Thermos Microplates shaker incubator (Elmi Ltd, Riga, Latvia). Next, the cells were diluted 658-fold (two serial dilutions of 15 µL into 185 µL, then 3 µL into 145 µL) into M9 glucose media with chloramphenicol and IPTG. IPTG concentrations used for the P_{Tac} constructs were: 0, 0.12, 0.48, 1.9, 7.6, 30.4, and 121.6 µM; IPTG concentrations used for the $P_{LlacO-1}$ were: 0, 1.9, 7.6, 30, 120, 490, and 1900 µM. Cells were grown in

the same shaking-incubator conditions for six hours, and then 40 µL of culture was added to 160 µL of phosphate buffered saline (PBS) with 2 mg/mL kanamycin to halt protein production. Cells were incubated in PBS for one hour to allow YFP to mature, and then flow cytometry was performed. The fluorescence values were white-cell subtracted, and then plots of CI-GFP production versus GFP production were created for both P_{Tac} and $P_{LlacO-1}$ (see Section I.A.).

5.3.8 Construction and screening of RBS libraries.

Mutations in the ribosomal binding site of several gates were introduced to shift the threshold of the gates' response curves. These RBS libraries were created using oligonucleotide primers containing multiple degenerate nucleotides in the 18 bases immediately upstream from the start codon. These primers were used to amplify the entire gate characterization plasmid using two primers diverging from each other at the gate's RBS. 100 ng of linear dsDNA PCR product was phosphorylated and ligated in a one-pot reaction using 0.5 µL of T4 DNA ligase (New England Biolabs, MA, M0202S) and 0.5 µL of T4 polynucleotide kinase (New England Biolabs, MA, M0201S) in 10 µL 1X T4 ligase buffer, and then transformed into *E. coli* NEB 10-beta (New England Biolabs, MA, C3019). Individual clones from the gate RBS library were screened by growing them in the presence and absence of 1 mM IPTG. Clones with the highest ON/OFF range were chosen for further characterization. The full response functions of these gates were measured (Methods), and a subset of the gates that had unique threshold values were kept (see Section I.C).

5.3.9 Gate construction and characterization.

To characterize gate response functions, the IPTG-inducible promoter P_{Tac} was positioned directly upstream from a gate expression cassette on the circuit backbone (without the 5'-insulating terminator L3S3P21). Following the P_{Tac}-driven gate expression cassette, the cognate promoter for the gate was positioned upstream from the standard RPU cassette. The plasmid backbone also encoded the sensors LacI and TetR in an operon driven from a constitutive promoter. These gate measurement plasmids were transformed into *E. coli* NEB 10-beta, and then a colony was inoculated into 200 µl of M9 glucose media with 50 µg/mL kanamycin in a V-bottom 96-well plate (Nunc, Roskilde, Denmark, 249952) and then grown for 16 hours at 37 °C shaking at 1000 rpm in an ELMI Digital Thermos Microplates shaker incubator (Elmi Ltd, Riga, Latvia). The next day, liquid culture was diluted 178-fold (two serial dilutions of 15 µL into 185 µL) into M9 glucose media with kanamycin and grown for three hours in the same shaking-incubator conditions. Subsequently, the culture was diluted 658-fold (two serial dilutions of 15 µL into 185 µL, then 3 µL into 145 µL) into M9 glucose media with kanamycin and IPTG to induce the gate. The IPTG concentrations used were used were: 0, 5, 10, 20, 30, 40, 50, 70, 100, 150, 200, and 1000 µM. Cells were grown for five hours in the same shaking-incubator conditions, and then 40 µL of culture was added to 160 µL of phosphate buffered saline (PBS) with 2 mg/mL kanamycin to halt protein production. These cells were incubated in PBS for one hour to allow YFP to mature, and then flow cytometry was performed (see Flow cytometry analysis), the data was converted to RPU (see Conversion of fluorescence to RPU, below). In addition to the gate characterization, we also measured a strain containing a similar plasmid that contained P_{Tac} driving the YFP RPU cassette directly; we converted its fluorescent output to RPU. Three independent replicates were performed on three separate days for each measurement, and the average RPU was calculated.

We next plotted the average gate output RPU versus the average P_{Tac}-YFP RPU output to visualize the response of the gate output promoter activity as a function of P_{Tac} input promoter activity. This relationship was fit to a Hill equation using the Solver add-in for Microsoft Excel.

5.3.10 Characterization of gate impact on cell growth.

To quantify how growth is impacted by expression of various repressors, we constructed tandem inducible gate measurement plasmids to achieve higher levels of gate expression (pJS101-109). These plasmids are identical to the gate measurement plasmids (p[Gate-RBS#]), with the exception that a tandem P_{Tac}-P_{Tet} promoter drives the gate expression cassette. We inoculated and grew these strains in an identical manner to the gate characterization experiments, except we used different inducer concentrations to span the wider inducible range of the tandem promoter. For each construct, we induced with seven IPTG concentrations: 0, 9.5, 19, 48, 95, 290, and 950 µM; for an additional five samples, we induced with 950 µM IPTG along with aTc at concentrations: 0.0095, 0.095, 0.29, 0.95, and 1.9 ng/mL. These induced cells were grown for six hours in the same shaking-incubator conditions. After the induction experiment, 200 µl of cells were added to an optically clear bottom 96 well plate. The optical density of the cultures was measured at 600 nm using a BioTek Synergy H1 Hybrid Microplate Reader. We also measured 200 µl of blank media to determine the background absorbance of the media. For each gate, the final absorbance measurements were normalized to the absorbance of the first sample that had no inducer added (Figure 3d).

5.3.11 Flow cytometry analysis.

Fluorescence was measured using an LSRII Fortessa flow cytometer (BD Biosciences, San Jose, CA) run by the BD FACSDiva software. An FSC voltage of 437 V, SSC voltage of 289 V, and a green laser (488 nm) voltage of 425 V were used. An SSC and FSC threshold of >200 was used to limit collection to cell-sized particles. Between 10^3 and 10^5 gated events were collected for analysis. To calculate YFP fluorescence values for bar graphs, we used the flow cytometry software FlowJo (TreeStar, Inc., Ashland, OR), and used the median statistical tool. For conversion of the cytometry data to RPU, we used MATLAB to perform the fluorescence-axis transformations and to normalize the distributions.

5.3.12 Conversion of fluorescence to RPU.

The raw fluorescence from measurement of a sensor, gate, or circuit using the measurement protocol must be converted to relative expression units (RPU). The RPU standard used in this study differs from the Kelly standard (Kelly *et al*, 2009) in that we use an upstream insulating terminator, a 5'-promoter spacer, and a ribozyme insulator to reduce contextual variations. We also maintain an identical RBS, YFP, terminator, and plasmid backbone to the circuit measurement constructs (pAN1717). *E. coli* NEB 10-beta (New England Biolabs, MA, C3019) were transformed with the RPU standard plasmid and plated on LB agar with 50 µM kanamycin. Transformed colonies were inoculated into 1 mL M9 glucose media with kanamycin and grown for 16 hours at 37 °C and shaking at 250 rpm. Next, the cells were diluted 178-fold (two serial dilutions of 15 µL into 185 µL) into M9 glucose media with kanamycin and grown for three hours in the same shaking-incubator conditions. Next, cells were diluted 658-fold (two serial dilutions of 15 µL into 185 µL and then 3 µL into 145 µL) M9 glucose media and grown for six hours in the same shaking-incubator conditions.

At this point 40 µL of cells were added to 160 µL of phosphate buffered saline (PBS) with 2 mg/mL kanamycin to halt protein production. Cells were incubated in PBS for an hour to allow YFP to mature, and then flow cytometry was performed. Additionally, un-transformed *E. coli* NEB 10-beta cells ("white cells") were grown in an identical manner alongside the RPU standard-harboring strain, but without any antibiotics. These cells' autofluorescence were measured using flow cytometry as well. After flow cytometry as performed, the median of YFP fluorescence was calculated for the both the RPU standard and the white cells.

The median fluorescence measurements of sensors, gates, and circuits were converted to RPU using the following procedure. The median autofluorescence value from the white cells was first subtracted from all fluorescence values to correct for this non-YFP derived signal. In our experiments with our cytometer settings, the white cell fluorescence was approximately 15 au. The median fluorescence of the RPU standard was also subtracted by the white cell fluorescence. Next, the experimental sample (sensor measurement, etc.) as divided by the median fluorescence of the RPU standard (after autofluorescence correction). In our experiments, our corrected RPU standard fluorescence is 460 au. To return values to corrected arbitrary units, multiply the RPU numbers by the RPU standard's median (460 au for our measurements).

5.3.13 Genetic circuit assembly.

The genetic circuits in this research comprise codon optimized repressors and their cognate promoters (from ref (Stanton *et al*, 2014)) with additional 5'-promoter spacers, hammerhead ribozyme-based insulators (from ref (Lou *et al*, 2012) and this work), ribosomal binding sites (from this work), and transcriptional terminators (from ref (Chen *et al*, 2013)). The sensors used include a truncated AraC (AraC-C280*, referred to as AraC* in this work) that has reduced cross-talk with IPTG(Lee *et al*, 2007) and

its output promoter P_{BAD} which is induced using L-arabinose; LacI and its output promoter with a symmetric lac-operator P_{Tac} which is IPTG-inducible(Dykxhoorn *et al*, 1996); and TetR with its output promoter P_{Tet}(Lutz & Bujard, 1997). LacI and TetR are transcribed from the native P_{LacI} promoter. AraC* is transcribed from BBa_J23105 and terminated by L3S3P22(Chen *et al*, 2013). The circuit measurement backbone harbors a medium-copy p15A origin of replication and kanamycin resistance gene (from ref (Lutz & Bujard, 1997)). The circuit insertion site is flanked by an upstream insulating terminator L3S3P21(Chen *et al*, 2013), and a downstream insulating terminator the native AraC terminator TaraC. The actuator used in this research is a variant of yellow fluorescent protein (YFP)(Cormack *et al*, 1996). The output plasmid harbors a pSC101 origin of replication(Lutz & Bujard, 1997) and encodes the spectinomycin resistance gene, aadA. The output insertion site is flanked by an upstream insulating terminator L3S2P44(Chen *et al*, 2013) and a downstream insulating terminator L3S2P21(Chen *et al*, 2013). Each transcription unit for a circuit was cloned into a submodule plasmid with the ampicillin-resistance gene, ampR, and flanked on either side by 4bp scars and BbsI restriction enzyme recognition sites. To assemble a final circuit plasmid, submodule plasmids and the circuit measurement plasmid (with sensors already inserted) were mini-prepped prior to assembly (Qiagen, Limburg, 27104) and their concentration was measured using a NanoDrop 1000 spectrophotometer (Thermo Fisher Scientific, MA). In one tube, 40 fmol of each DNA plasmid were combined. In addition, 2 µL of ligase buffer, 0.5 µL of T4 DNA ligase HC (Promega, WI, M1794), and 2 µL of BbsI (New England Biolabs, MA, R0539L) were added to the tube (Figure S41). Lastly, filtered, deionized water was added to the tube to a total volume of 20 µL. This mixture was heated and cooled in a thermocycler repeatedly: 37 °C for 2 min, then 16 °C for 5 min, repeated for 10-100 cycles, depending on the number of pieces of DNA being assembled (10 cycles per piece

of DNA). After the cycling, the reaction was heated to 50 °C for 10 minutes to inactivate the ligase, and then 80 °C to inactivate the restriction enzyme. Then, 10 µL of assembly mixture was then transformed into 50 µL of NEB 10-beta chemically competent *E. coli*, allowed to recover for an hour, and then plated on agar with antibiotics.

5.3.14 Hexadecimal and Wolfram Rule naming conventions.

The convention for naming 3-input circuits is to first order the input states so that P_{Tac} activity is the least significant input bit, P_{Tet} is the middle significance input bit, and P_{BAD} is the most significant input bit. The corresponding expected output states for all inputs from 000 to 111 are converted to hexadecimal—four binary bits are converted to a single hexadecimal digit. The resulting two-digit hexadecimal number is listed after the hexadecimal indicator "0x" to create a name of the form "0xNN". This is similar to the Wolfram Rule naming system, where the input rows are arranged 111, 110, ..., 001, 000 and then the binary output vector is converted to decimal (e.g., "Rule 110" has binary output vector *01101110*).

5.3.15 Software tools.

The following software, languages, and libraries were used in this work. The Cello source code is written in Java (version 1.8.0_31) with software project management by Apache Maven (version 3.2.1). Constrained combinatorial designs of genetic architectures are produced using Eugene (version 2.0) (Oberortner *et al*, 2014), and the synthetic biology open language library (libSBOLj version 1.1) is used to store circuit designs in a hierarchy of annotated DNA components (Galdzicki *et al*, 2014).

Logic minimization uses Espresso (Brayton *et al*, 1984) (version 2.3) and ABC(Brayton & Mishchenko, 2010) (UC Berkeley, version 1.01 March 2014).

Several figures for data visualization are generated during a Cello design run. Directed graphs are produced using Graphviz (version 2.34.0). Data plots for response function calculations and predicted output distributions are produced using Gnuplot (version 4.6). Part-based circuit representations are produced using Dnaplotlib, which uses the python matplotlib 2D plotting library for programmable rendering of highly customizable genetic diagrams. A static plasmid image is produced using EMBOSS cirdna (version 6.6.0.0).

The Cello web application is hosted using the Amazon Web Service (AWS) and is deployed using the Jetty web server (version 8.1.13). The browser (client-side) sends data to and retrieves data from the Amazon server (server-side) using AJAX (Asynchronous JavaScript and XML). The web interface uses JavaScript, jQuery, and jQuery UI (version 1.10.2) for user-interactive event handling and dynamic interface manipulation. CSS Bootstrap (version 2.3.1) is used to style the content, and CodeMirror (version 3.13) is used for Verilog syntax highlighting.

For parameterizing Hill equations to response functions, the fit was performed by minimizing the sum of relative error magnitudes between the trend line and the data points using the Excel Solver add-in with the GRG Nonlinear solving method. The initial version of Cello that performed gate assignments by simulating input signal propagation through response functions was implemented using MATLAB version R2012a (7.14.0.739). Design of the Cello circuits library used the distribution propagation to screen for circuits, and used the simulated annealing algorithm with a temperature of 0. Additionally, assignments using QacR, LitR, IcaRA, PsrA, and LmrA repressors were disallowed.

Ribozyme secondary structure was simulated using RNA mFold (Zuker, 2003) (http://mfold.rna.albany.edu/?q=mfold/RNA-Folding-Form) using the following parameters: 37°C, 1M NaCl, 5 percent suboptimality folds computed, 50 maximum computed foldings, maximum interior bulge/loop size = 30, maximum asymmetry of an interior bulge/loop = 30, no limit for maximum distance between paired bases. Sequence alignments were performed with Clustal Omega (1.2.1) multiple DNA sequence alignment using the default parameters (http://www.ebi.ac.uk/Tools/msa/clustalo/). The ribozyme phylogenetic tree was also generated using Clustal Omega using the default tree format, distance correction off, exclude gaps off, the UPGMA clustering method, and the "real" phylogram branch length setting.

5.3.16 Precomputing 3-input 1-output NOR circuit diagrams.

In the library of user-defined circuit motifs (Section V.C), we used a precomputed list of small 3-input 1-output NOR/NOT circuits. These circuits were found by computationally enumerating all NOR/NOT circuits with ≤ 6 layers, evaluating each circuit's truth table, and then selecting the circuit with the fewest number of gates for each truth table. We computationally enumerated all circuits by constructing them in levels. Level 1 comprises the circuit input wires: A, B, C, and 0, where 0 is a Boolean 'false'. Note that when 0 is one of the inputs to a NOR gate, this results in a NOT gate for the other input. To enumerate all circuits in Level 2, all pairwise combinations of Level 1 output wires (A, B, C, and 0) are input into a NOR gate. For example, (A NOR B) is a Level 2 circuit. To enumerate all circuits in Level 3, all pairwise combinations of wires containing an output from Level 2 and an output from any level are input into a NOR gate. For example, ((A NOR B) NOR A) is an example of a

Level 3 circuit. If the two circuits being joined have a duplicate logic motif (in the previous example, input A was specified twice), a fan-out wire is used and the redundant gates are removed. This process was continued until all Level 6 circuits were enumerated. After each individual circuit construction, the circuit's truth table output was evaluated. If the circuit used fewer gates than all previous circuits implementing that truth table, the circuit was stored until a smaller one was found. This algorithm resulted in small motifs for each 3-input 1-output circuit. We used these motifs for subcircuit replacement in the final step of logic synthesis.

5.3.17 RPU plasmid characterization using smRNA-FISH.

To determine the steady state number of *yfp* mRNA copies per cell at mid-exponential growth with the RPU standard plasmid (pAN1717), we used smRNA-FISH to label the *yfp* mRNA molecules. We designed a set of 25 oligonucleotide probes, fluorescently labeled with TAMRA, each 20 bases in length, against the *yfp* transcript (Table S10) using Stellaris Probe Designer version 4.1. Three independent replicates were performed on three separate days for each measurement, and the average number of *yfp* mRNA/cell was calculated.

Sample preparation. The RPU plasmid (pAN1717), the non-YFP plasmid (pAN1201) and the measurement plasmid (pAN1818) were transformed to create an RPU standard, a background control, and a standard curve for mRNA/cell estimates, respectively. The plasmids were transformed in separate reactions into *E. coli* NEB 10-beta (New England Biolabs, MA, C3019), grown on LB + 50 µg/mL kanamycin agar plates, and then a colony was inoculated into 200 µL of M9 minimal media with 50 µg/mL kanamycin in a V-bottom 96-well plate (Nunc, Roskilde, Denmark, 249952). The cultures were grown for 16 hours at 37 °C shaking at 1000 rpm in an ELMI Digital Thermos Microplates shaker incubator (Elmi Ltd, Riga, Latvia). The next day, the

liquid culture was diluted 178-fold into M9 minimal media with kanamycin and grown for three hours in the same shaking-incubator conditions. Subsequently, the culture was diluted 658-fold into M9 minimal media with kanamycin and IPTG to induce the YFP production from the pAN1818 plasmid. The IPTG concentrations used were: 0, 5, 10, 20, 30, 40, 50, 70, 100, 150, 200, and 1000 µmol/L. Cells were grown for five hours in the same shaking-incubator conditions, and then 6 mL of culture per sample was pooled together in 15 mL Corning centrifuge tube and the cells were pelleted by centrifugation (10 minutes, 4000×g, 4 °C). The supernatant was removed and the cells were resuspended in 1 mL 1× PBS (diluted from 10× PBS, Ambion, #AM9625) to wash the cells.

The cells were transferred to RNase free microfuge tubes and pelleted by centrifugation (5 minutes, 4500×g, 4 °C). The supernatant was removed and the cells were resuspended in 1 mL freshly prepared 3.7% formaldehyde (Fisher, #BP531) in 1× PBS (diluted from 10× PBS). The cells were then mixed on a nutator at room temperature for 30 minutes. The cells were pelleted by centrifugation (8 minutes, 400×g). The supernatant was removed and the cells were washed in 1 mL 1× PBS twice (i.e. resuspended in 1 mL 1× PBS, centrifuged at 600×g for 3.5 minutes, and supernatant removed). The cells were resuspended in 300 µL water and then 700 µL of 100% ethanol was added and mixed twice to get to a final concentration of 70% ethanol. The cells were left at room temperature with mixing on a nutator for 1 hour to permeabilize the cell membrane.

Hybridization procedure. After permeabilization, cells were centrifuged (7 minutes, 600×*g*) and the supernatant was removed. The cells were resuspended in 1 mL of 50% formamide wash buffer A (Biosearch Technologies Cat no SMF-WA1-60). Reagents containing formamide were prepared fresh, right before use. The stock formamide was stored at -20 °C in 1.5 mL aliquots and thawed right before use. Next, 50% formamide

hybridization buffer (Biosearch Technologies Cat no SMF-HB1-10) was prepared by adding 12.5 µmol/L mixed probe stock to make a final 62.5 µmol/L probes concentration. The cells were then centrifuged (7 minutes, 600×g) and the supernatant was removed. The cells were resuspended in 50 µL of the 50% formamide hybridization buffer with probes and left to incubate in the dark at 30 °C overnight. Next, 400 µL of 50% formamide Wash Buffer A was added to the tube and mixed well. Cells were pelleted by centrifugation (7 minutes, 600×g) and the supernatant was removed. The cells were washed 3 more times (i.e. resuspended in 200 µL of 50% formamide Wash Buffer A, incubated at 30°C for 30 minutes, centrifuged at 600×g for 3.5 minutes, and supernatant removed). 4',6-diamidino-2-phenylindole (DAPI, Fisher Scientific, #PI-46190) was added to the wash solution to a final concentration of 10 µg/mL in the last wash. The cells were resuspended in 500 µL of Wash Buffer B (Biosearch Technologies Cat no SMF-WB1-20), centrifuged at 600×g for 3.5 minutes, and supernatant removed. The cells were resuspended in 40 µL to 50 µL of 2×SSC and imaged under the microscope.

Microscopy. 2 µL of sample was pipetted onto a 45 mm × 50 mm #1 coverslip (Fisher Scientific, #12-544F). A 1 mm thick × 10 mm × 7 mm 1.5% agarose gel pad (in 1× PBS) was laid on the sample. A 22 mm × 22 mm #1 coverslip (Fisher Scientific, #12-545B) was placed on top of the agarose gel pad. The sample was imaged using an inverted epifluorescence microscope (Zeiss Axio Observer.Z1), a 100×, N.A. 1.46 oil immersion objective (Zeiss, alpha-Plan APO), and a cooled digital CMOS camera (Hamamatsu Orca Flash 4.0). The microscope and camera were controlled using the Zen Pro Software (Zeiss). The mRNA labeled by smFISH probes were imaged using a TAMRA filter set (Zeiss, 43 HE), a HXP-200 excitation light source set on 50% intensity, and an integration time of 1 s. DNA stained by DAPI was also imaged using a multi-band filter set (Zeiss, 81 HE), 353 nm excitation with an LED source (Zeiss,

Colibri) set to 100%, and an integration time of 50 ms. *Z*-stacks with 9 slices and 200 nm spacing were acquired for bright field and TAMRA images. Each sample was imaged at multiple locations to get a total of at least 300 cells per sample.

Image and data analysis. Image processing and data analysis were performed using MATLAB and Mathematica. Cell recognition and segmentation was performed on brightfield images of cells using the *Schnitzcells* MATLAB module (Young *et al*, 2012). The program applies edge detection and other morphological operations, using the MATLAB Image Processing Toolbox. The output was checked and corrected using the manual interface offered by *Schnitzcells*.

Spot recognition was performed on the segmented TAMRA fluorescence images using the *Spatzcells* MATLAB module (Skinner *et al*, 2013). The *Spatzcells* software detects each fluorescent spot within the segmented cell image stacks, finds its location (x, y, z-slice), and fits it to a 2D-Gaussian function to obtain the height and intensity of the spot.

Estimating mRNA copy numbers. For cells with low mRNA copy number, the smFISH spots are typically well separated within the cells, and so they can be visualized and counted as individual spots (see example images in Figure S33a). For cells with higher copy number, the spots overlap significantly. Quantitative estimation of the copy number for a full range of expression levels therefore requires a method for extrapolating from the low expression regime to the high expression regime. The mRNA target described in (Skinner *et al*, 2013) was relatively long (~ 3000 base pairs), so that 72 different probes (each ~20 nucleotides long) could be designed to span the length of the target sequence. The mRNA target for the measurement described here (*yfp*) is considerably shorter (720 base pairs) so that only 20 different 20 nucleotide probes could be designed to span the target sequence (see Table S10). Consequently, the spot intensities (the heights of the fitted 2D-Gaussians) for the bright spots

corresponding to intact *yfp* mRNAs were only partially resolved from the background, lower-intensity spots. Because of the partial overlap in the typical spot heights for the labeled mRNA and background spots, the thresholding method described in (Skinner *et al*, 2013) for distinguishing the two types of spots did not work reliably. Therefore, a new method for extrapolation to high expression levels was used, based on the assumption that the total FISH fluorescence signal measured for each cell is a linear function of the number of mRNAs in the cell.

Briefly, for each sample, a histogram was constructed of the spot heights for all detected spots (see Figure S33b), and the histogram was fitted to a sum of two log-normal distributions to obtain estimates of the total number of dim spots and bright spots for that sample. For low-expression samples, the two spot populations were partially resolved and the fit was unconstrained (see, for example, Figure S33b, panel 2, "P_{Tac}-YFP (10 µM IPTG)"). The fits to the low expression sample histograms were used to define a constraint value equal to the mid-way point between the locations of the two fitted log-normals.

For higher expression, the location parameters for the 1st log-normal distribution was constrained to be less than the constraint value and the location parameter for the 2nd log-normal was constrained to be greater than the constraint value. The total number of bright spots for each sample was estimated from the integrated area of the 2nd log-normal (the one corresponding to the brighter spots) from each fit. The number of bright spots divided by the total cell area was then plotted vs. the average FISH signal per cell area, and the result was fit to a form assuming a Poisson filling process of the available image area with spots:

$n_{spots} = n_{max}\left(1 - \exp\left(-\frac{x-\beta}{\alpha \cdot n_{max}}\right)\right)$, where n_{spots} is the number of bright spots per cell area, x is the FISH signal per cell area, and the fitting parameters: n_{max} is the maximum number of resolvable spots per cell area, α is the typical FISH signal for a

single mRNA, and β is and the background FISH signal per cell area (see Figure S33c). The linear portion of the fit curve was then used to extrapolate to higher expression levels, giving an estimate for the mRNA copy number for each cell: $N_{est.} = \frac{S-\beta \cdot A}{\alpha}$, where S is the total FISH signal for the cell and A is the image area of the cell (pixels). Example histograms of the estimated mRNA copy number are shown in Figure S33d. The mean mRNA copy number per cell was calculated for each sample.

The estimation procedure (including fitting to the spot height histograms, estimation of number of bright spots, fitting with the Poisson filling process model, and estimation of the mean mRNA copy number per cell) was done independently for each replicate experiment. The values obtained from the replicate measurements were then averaged to produce the final mRNA copy number per cell estimates as shown in Figure S33e.

6 Appendix: User Constraint File (UCF) for genetic circuit design automation

6.1 File overview

Verilog code is compiled to a circuit architecture that is defined by the user constraint file (Figure 4-45). This is a highly specified system that defines a particular library of gates as well as rules to be enforced for preferred logic motifs and genetic structure. In addition, it contains the definition of the particular strain and "landing pad" (*e.g.*, a defined set of plasmids or genomic locations) as well as the environmental conditions where the circuit models are valid. New UCFs can be developed for new gate libraries and/or strains and environments. While in practice a particular UCF may be valid for differing genotypes or changes in media/growth rate, out recommendation is that a new UCF file should be built for each end application.

This section describes the format of the UCF, as well as the specific Eco1C1G1T1 file used in this manuscript. Within the genetic gates library category, genetic parts and experimental data for the each gate are specified. The experimental measurements and associated standards for data in the UCF are described in Chapter 4. This section focuses on the data structure of the UCF, with the intention of guiding the composition of files for new gate libraries, organisms, and operating conditions.

6.2 File format

The UCF is specified using JavaScript Object Notation (JSON). JSON is a widely used and language-independent format based on attribute:value pairs, which is human readable, machine parseable, and can be converted to common data structures in other languages. In a JSON attribute:value pair, the values are restricted to these types: string, number, boolean, null, array (square brackets), and object (curly brackets).

SBOL XML was also considered as a file format for the UCF. SBOL Version 1.1 (Galdzicki et al, 2014) is tailored for specifying parts and composite parts in a genetic design hierarchy, where URIs (uniform resource identifiers) are used to uniquely and globally identify parts via the World Wide Web. While this format would be directly applicable for our *gate_parts* and *parts* UCF collections, the other collections required a more flexible representation. However, the proposed versions of SBOL will have more versatile data model (Roehner *et al*, 2015), have the ability to specify custom objects, and will be able to read/write data in JSON format.

Required collections:

header, measurement_std, logic_constraints, gates, response_functions, gate_parts, parts

These collections specify the experimental system, the available gate types for logic synthesis, response function data for assignment, and gate parts to build the final DNA sequence.

Optional collections:

motif_library: if omitted, subgraph substitution will not occur as an optimization step in logic synthesis.

gate_cytometry: if omitted, the output predictions will be median values, as opposed to cytometry distributions.

gate_toxicity: if omitted, the prediction of growth impact will not be calculated.

eugene_rules: if omitted, unconstrained circuit layout design will occur. This will result in variations in tandem promoter order, variations in gate order, and variations in gate orientation.

genetic_locations: if omitted, the output DNA sequence will contain the circuit components only, and the user will be responsible for deciding and implementing the genetic context of the circuit design.

In the sections below, each describes a different collection with a brief description and a box containing an example object. For all collections, attribute names are parsed in Cello.

6.2.1 Header

The header collection specifies the operating conditions, strain, and genetic location where the gate measurements were made and the circuit predictions would be valid. These data do not impact circuit design in Cello. However, it is required to describe the operating conditions for which the circuit designs are valid. Thus, it is required by Cello to accept the file as a valid UCF.

version: This demarcates the iteration of the UCF. Version updates could include larger gate libraries, changes in experimental conditions, more accurate data, etc. Our current numbering system is shown below, but this particular format is not required.

example:

"version": "Eco1C1G1T1"

<string> Organism identifier (Eco for *E. coli*)

Eco<number> Strain identifier (counting up from 1; Eco1 = NEB 10-beta)

C<number> Experimental conditions identifier

G<number> Genetic gates and insertion location identifier

T<number> Technology mapping and motifs identifier

required: yes

author: Intended to help document versions and modifications of UCF files.

required: no

```
{
  "collection": "header",

  "version": "Eco1C1G1T1",

  "date": "2015-04-08",

  "author": ["Bryan Der", ''Alec Nielsen'', ''Prashant Vaidyanathan''],

  "organism": "Escherichia coli NEB 10-beta",

  "genome": "NEB 10-beta Δ(ara-leu) 7697 araD139 fhuA ΔlacX74 galK16 galE15 e14-
ϕ80dlacZΔM15 recA1 relA1 endA1 nupG rpsL (Str^R) rph spoT1 Δ(mrr-hsdRMS-mcrBC) (New England
Biolabs)",

  ''media'': ''M9 minimal media composed of M9 media salts (6.78 g/L Na2HPO4, 3 g/L KH2PO4, 1
g/L NH4Cl, 0.5 g/L NaCl, 0.34 g/L thiamine hydrochloride, 0.4% D-glucose, 0.2% Casamino acids,
2 mM MgSO4, and 0.1 mM CaCl2; kanamycin (50 ug/ml), spectinomycin (50 ug/ml)'',

  "temperature": "37",

  "growth": "Inoculation: Individual colonies into M9 media, 16 hours overnight in plate
shaker. Dilution: Next day, cells dilute ~200-fold into M9 media with antibiotics, growth for
3 hours. Induction: Cells diluted ~650-fold into M9 media with antibiotics. Growth: shaking
incubator for 5 hours. Arrest protein production: PBS and 2mg/ml kanamycin. Measurement: flow
cytometry, data processing for REU normalization. ",
}
```

organism: Defines the organism, species, and strain for which the circuits are compiled.

example:

"organism": "Escherichia coli NEB 10-beta"

required: yes

allowed values: should be the full organism name

genome: Specifies the genotype of the organism.

example:

"genome": "NEB 10 Δ(ara-leu) 7697 araD139 fhuA ΔlacX74 galK16 galE15 e14- φ80dlacZΔM15 recA1 relA1 endA1 nupG rpsL (StrR) rph spoT1 Δ(mrr-hsdRMS-mcrBC)"

example:

"genome": "K-12 MG1655* [F-lambda-ilvG-rfb-50 rph-1 Δ(araCBAD) Δ(LacI)]

required: yes

allowed values: any string, including the entire genome sequence

temperature: Specifies the temperature at which the circuits are expected to perform (and gates measured).

required: yes

allowed values: units of Celsius

growth: Specifies the growth conditions at which the circuits are expected to perform (and gates measures).

required: yes

6.2.2 Measurement standard

This collection specifies the unit of measurement that is used for all signals (sensor ON/OFF levels, gate response functions, output levels). The standard unit in UCF Eco1C1G1T1 is RPU (relative expression unit). It also includes a description of the standard plasmid, which contains a constitutively expressed YFP cassette from a

standard promoter with ribozyme insulation, the complete plasmid DNA sequence for the plasmid is specified, and instructions for normalization.

required: yes

```
{
  "collection": "measurement_std",

  ''signal_carrier_units'': ''REU'',

  ''plasmid_description'': ''p15A plasmid backbone with kanamycin resistance and a YFP expression cassette. Upstream isulation by terminator L3S3P21 and a 5'-promoter spacer. Promoter BBa_J23101, ribozyme RiboJ, RBS BBa_B0064 drives constitutive YFP expression, with transcriptional termination by L3S3P21.'',

  ''plasmid_sequence'': [all lines of the Genbank file (not shown) for the measurement standard plasmid],

  ''normalization_instructions'': ''The following equation converts the median YFP fluorescence to REU.  REU = (YFP - YFP0)/(YFPREU - YFP0), where YFP is the median fluorescence of the cells of interest, YFP0 is the median autofluorescence, and YFPREU is the median fluorescence of the cells containing the measurement standard plasmid'',

}
```

signal_carrier_units: Different circuit design frameworks might use different units for quantifying high / low signals. If a different unit is used, the unit should be used for all signal-related fields in a UCF (*response_function, gate_cytometry, gate_toxicity*). The following data should all have the same signal carrier unit: sensor ON/OFF levels and gate response function input/output levels. For example, in Eco1C1G1T1, *response_function* collection is generically written, but the input/output response has units of RPU as specified by this attribute.

required: yes

example: RPU

plasmid_description: Instructions and experimental conditions for normalizing experimental measurements for the standardized units (RPU in Eco1C1G1T1). Instructions should include the full plasmid name, growth, measurement, and other conditions necessary to normalize the data.

required: yes

allowed values: plain text

plasmid_sequence: Plasmid DNA sequence containing the expression cassette that serves as the measurement standard for normalization of all signal levels in the UCF (RPU).

required: yes

allowed values: Genbank file format, with or without annotations before the sequence.

normalization_instructions: A brief set of instructions describing how data is normalized using the measurement standard.

required: yes

6.2.3 Logic constraints

In this collection, the allowed Boolean gate types are specified, and the maximum number of instances is specified for each gate type. These Boolean constraints cannot exceed the genetic gates available in the library, but they can be more restrictive. For example, if 12 NOR gates are in the library, the user could constrain logic synthesis to use a maximum of 9 NOR gates. Furthermore, if there are 20 NOT gates in a library but only 12 unique repressors due to RBS variants, the Boolean constraint would be 12 NOT/NOR gates.

```
{
  "collection": "logic_constraints",
  ''available_gates'': [
     {
       ''type'' : ''NOR'',
       ''max_instances'' : 12
     },
     {
       ''type'' : ''OUTPUT_OR'',
       ''max_instances'' : null
     }
  ]
}
```

available_gates: Specifies a gate type and maximum number of instances of each gate type for a circuit topology from logic synthesis.

required: yes

Note: A 1-input NOR gate is equivalent to a NOT gate, so a value of 12 for *max_instances* indicates a maximum of 12 NOR + NOT gates in the circuit.

Note: An OUTPUT_OR gate does not require a transcription factor, so a value of null indicates that there are no restrictions on the number of OUTPUT_OR gates in the circuit.

Note: Specifying fan-in constraints, such as allowing a 3-input NOR gate motif, is not done here. This NOR motif would be specified in the *motif_library* collection.

6.2.4 Motif library

In this collection, the user can define logic motifs that can be swapped for logically equivalent subcircuits in the last stage of logic synthesis. Subcircuits are specified using a "netlist" format (Sleight *et al*, 2010), which is standard for specifying the connectivity of a list of gates. Specifying subcircuits in this way is similar to specifying structural

Verilog, and the input names and output names must be defined before listing the gate connectivity.

```
{
  "collection": "motif_library",
  ''inputs'':[''a'',''b'',''c''],
  ''outputs''[''y'']
  "netlist": [
    "NOT(Wire0, b)",
    "NOR(Wire1, Wire0, c)",
    "NOR(Wire2, Wire1, a)",
    "NOR(Wire3, Wire2, a)",
    "NOR(Wire4, Wire2, Wire1)",
    "NOR(y, Wire3, Wire4)"
  ]
}
```

inputs: the names of input wires

example:

"inputs" : ["a","b","c"]

example:

"inputs" : ["in1","in2","in3"]

required: yes

allowed values: input names must match the input wire names in the netlist.

outputs: the names of output wires. Allows single- or multiple-output subcircuits to be defined.

example:

"outputs" : ["y"]

example:

"outputs" : ["out1","out2"]

allowed values: output names must match the output wire names in the netlist.

netlist: A gate type is listed, followed by the output wire name, followed by the list of input names. The examples show a multi-level (layered) NOR/NOT motif, an OUTPUT_OR motif, a 3-input NOR motif, and a primitive AND motif:

example of a precomputed NOR/NOT motif:

"netlist" : [

"NOT(Wire0, b)",

"NOR(Wire1, Wire0, c)",

"NOR(Wire2, Wire1, a)",

"NOR(Wire3, Wire2, a)",

"NOR(Wire4, Wire2, Wire1)",

"NOR(y, Wire3, Wire4)

]

example of an OUTPUT_OR gate motif:

"netlist" : [

"OUTPUT_OR(y, a, b)"

]

example of a 3-input NOR gate motif:

"netlist" : [

"NOR(y, a, b, c)"

]

example of an AND gate motif:

"netlist" : [

"AND(y, a, b)"

]

required: yes

allowed values: gate names are restricted to NOT, NOR, AND, OR, OUTPUT_OR, NAND, XOR, XNOR.

6.2.5 Gates

Gates in the library are specified in this collection. This is a concise collection that does not include other gate-related data collections (gate_parts, response_functions, gate_cytometry, gate_toxicity). For modularity, these data are stored in other collections, which are linked to the gate through the gate_name attribute. The attribute system allows other genetic logic systems to be specified in future versions, including zinc fingers, TALEs, CRISPRi, activator-chaperone pairs, etc.

```
{
  "collection": "gates",
  ''group_name'' : ''BM3R1'',
  "gate_name": ''B3_BM3R1'',
  "gate_type": ''NOR'',
  "system": ''TetR'',
}
```

group_name: this attribute is used to group variants of gates that cannot be used together in a circuit design. For example, all RBS variants of a certain repressor (B1_BM3R1, B2_BM3R1, B3_BM3R1) would belong to the same group (BM3R1), since a repressor is can only be used once per circuit assignment. Furthermore, if known cross-talking interactions between different repressors are known, these could also be put into the same group. Similarly, if homologous recombination is a concern and two gates have the same large part, then they can be placed in the same group.

required: yes

allowed values: Using the repressor name would be typical for RBS variants, but any name can be used.

gate_name: this attribute is used to link gate data in other collections to the gate object (*response_function, gate_parts*, etc).

required: yes

allowed values: Only alphanumeric and underscore characters are allowed, which complies with allowed names in the Eugene language. The gate name must be identical to the string used in the *gate_name* attribute in other collections

gate_type: used during the assignment algorithm; for example, a genetic NOR gate cannot be assigned to an AND gate in the circuit topology.

required: yes

allowed values: NOR, NOT, OR, AND, NAND, XOR, XNOR. Note: To allow for multiple inputs, NOT gates must be specified here as a NOR gate. If a repressing gate can only have a single input, the gate type can be NOT.

system: used to specify the type of biochemistry from which the gates are built. A single UCF could have gates based on different biochemistries.

required: yes

allowed values: any string, such as "TetR", "CRISPRi", or "Activator-chaperone"

6.2.6 Gate parts

This collection specifies the transcription units and output promoters for a genetic gate, which is mapped to a gate through the *gate_name* attribute. A NOR or NOT gate may have a different number of parts compared to, say, an AND gate. Thus, the

transcription_ units attribute is an array instead of a single object for flexibility for different genetic gate types. As with Boolean primitive gates, all genetic primitive gates are restricted to have a single output promoter. The restriction of a single output promoter name is not to be confused with fan-out, where multiple instances of the named promoter are used in the circuit.

```
{
  "collection": "gate_parts",
  "gate_name": "A2_AmtR",
  "transcription_units": [
     [
      "BydvJ", "A2", "AmtR", "L3S2P55"
     ]
  ],
  "promoter": "pAmtR"
}
```

gate_ name: The name of the gate.

required: yes

allowed values: the name must be identical to the *gate_ name* value in the gate collection.

transcription_ units: The part composition of the gate. The regulable promoter is listed separately from the transcription unit, because the promoter driving the transcription unit depends on the circuit diagram.

example:

Some gate types might require two transcription units, such as an activator-chaperone AND gate (Moon *et al*, 2012). For this reason, the value of this attribute is an array of arrays. The first element of the array is the transcription unit for InvF, and the second element of the array is the transcription unit for SicA:

```
"collection": "gate_parts",
 "gate_name": "SicA-InvF",
 "transcription_units": [ //note that this begins the outer array
   [ //element 1 of the inner array
    "RiboJ11", "RBS-InvF0", "InvF", "M13"
   ],
   [ //element 2 of the inner array
    "RiboJ10", "RBS-SicA0", "SicA", "BBa_B1006U10"
   ]
 ],
 "promoter": "pSicA"
```

required: yes

allowed value: array of arrays. The outer array contains the list of transcription units, excluding promoters, and the inner array contains the list of part names that make up each transcription unit. The part names can be any string, but the string must match a part name in the *part* collection.

promoter: The output promoter of a gate.

required: yes

allowed value: single promoter name (alphanumeric and underscore characters only).

6.2.7 Parts

This collection specifies basic genetic parts: promoters, ribozymes, ribosome binding sites, coding sequences, terminators, scars, spacers, etc. This collection specifies the part name, part type, and part DNA sequence. The part names listed in the *gate_parts* collection must match a part name from this collection. For example, all parts used in the A2_AmtR gate from the previous section are specified:

```
    {
      "collection": "parts",
      "type": "ribozyme",
      "name": "BydvJ",
      "dnasequence":
"CTGAAGGGTGTCTCAAGGTGCGTACCTTGACTGATGAGTCCGAAAGGACGAAACACCCCTCTACAAATAATTTTGTTTAA"
    }
```

```
    {
      "collection": "parts",
      "type": "rbs",
      "name": "A2",
      "dnasequence": "AATGTTCCCTAATAATCAGCAAAGAGGTTACTAG"
    }
```

```
    {
      "collection": "parts",
      "type": "cds",
      "name": "AmtR",
      "dnasequence":
"ATGGCAGGCGCAGTTGGTCGTCCGCGTCGTAGTGCACCGCGTCGTGCAGGTAAAAATCCGCGTGAAGAAATTCTGGATGCAAGCGCAGAACTG
TTTACCCGTCAGGGTTTTGCAACCACCAGTACCCATCAGATTGCAGATGCAGTTGGTATTCGTCAGGCAAGCCTGTATTATCATTTTCCGAGCA
AAACCGAAATCTTTCTGACCCTGCTGAAAAGCACCGTTGAACCGAGCACCGTTCTGGCAGAAGATCTGAGCACCCTGGATGCAGGTCCGGAAAT
GCGTCTGTGGGCAATTGTTGCAAGCGAAGTTCGTCTGCTGCTGAGCACCAAATGGAATGTTGGTCGTCTGTATCAGCTGCCGATTGTTGGTAGC
GAAGAATTTGCAGAATATCATAGCCAGCGTGAAGCACTGACCAATGTTTTTCGTGATCTGGCAACCGAAATTGTTGGTGATGATCCGCGTGCAG
AACTGCCGTTTCATATTACCATGAGCGTTATTGAAATGCGTCGCAATGATGGTAAAATTCCGAGTCCGCTGAGCGCAGATAGCCTGCCGGAAAC
CGCAATTATGCTGGCAGATGCAAGCCTGGCAGTTCTGGGTGCACCGCTGCCTGCAGATCGTGTTGAAAAAACCCTGGAACTGATTAAACAGGCA
GATGCAAAATAATAA"
    }
```

```
{
  "collection": "parts",
  "type": "terminator",
  "name": "L3S2P55",
  "dnasequence": "CTCGGTACCAAAGACGAACAATAAGACGCTGAAAAGCGTCTTTTTTCGTTTTGGTCC"
}
```


type: The part class.

required: yes

allowed values: any string, but the string might be used during enumeration of part-based Eugene rules, and will be used for annotation of the output DNA sequence.

name: The name of the part.

required: yes

allowed values: the part name must match the part name specified in the *gate_parts* collection.

dnasequence: The sequence of the part.

required: yes

allowed values: only ATCGatcg characters and the DNA sequence is in the forward orientation.

6.2.8 Response functions

This collection specifies the response function for a gate identified by the gate_name attribute. The response function includes a mathematical equation as well as a set of parameters that map to the variables in that equation. Different gate types may be captured by different mathematical forms and this can be specified in this

collection. For NOT/NOR gates, a Hill equation describes the cooperative and monotonic decrease in output with respect to input, and the parameters y_{max}, y_{min}, K, and n are fitted from experimental data for each gate. Note that y_{max}, y_{min}, and K are specified generically without units; units are described in the header collection, and these units should be used consistently throughout all data related to signal levels.

Cello uses a math evaluator that solves equations expressed as strings. User-defined equations with user-defined parameters can be accommodated, as long as parameter names match the variable names in the equation string. This is an example of a Hill equation for the A2_AmtR gate:


```
{
  "collection": "response_functions",
  "gate_name": "A2_AmtR",
  ''variables'': [''x'']
  ''parameters'': [
    {
      ''name'': ''ymax'',
      ''value'': 13.18696
    },
    {
      ''name'': ''ymin'',
      ''value'': 0.316394
    },
    {
      ''name'': ''K'',
      ''value'': 0.169953
    },
    {
      ''name'': ''n'',
      ''value'': 1.319126
    },
  ],
  "equation": "ymin+(ymax-ymin)/(1.0+(x/K)^n)"
}
```


gate_name: The name of the gate.

required: yes

allowed values: name must match the intended gate name from the *gate* collection.

variables: The input to each gate; for example, the activity of the input promoter.

example: if there are multiple inputs (x and y) to the response function:

"variables": ["x", "y"],

required: yes

allowed values: array of strings, where each string must match a variable name in the equation.

parameters: Definition and numerical value of each parameter in the response function.

required: yes

allowed values: array of objects, where each object is a parameter with a name attribute and value attribute. The name must match the equation string.

equation: The mathematical form of the response function.

example: a two input response function, this equation can be used for an AND gate:

"equation": -log(D + ((A-D)/(1+((x/C)^B))) + D + ((A-D)/(1+((y/C)^B)))),

required: yes

allowed values: right-hand side of an equation of interest; the calculated left-hand side value is returned by the evaluate function. It can take any form and is not restricted to a Hill equation. The math evaluator class supports common operators, constants, and functions such as:

^ power

```
*       multiply
(       implicit multiply
/       divide
+       add
-       subtract
LN2     natural log
log10   log base 10
        min     minimum of
        max     maximum of
        sqrt    square root
```

6.2.9 Gate cytometry

This collection specifies histograms that describe the response function of each gate. Note that cytometry data is not required for Cello to run, but including it allows Cello to predict distributions for the simulated circuit output. In its absence, the output of Cello will be predicted values, as opposed to predicted cytometry distributions.

When the response function is characterized, a set of distributions is measured for different activities of the input promoter. Thus, *cytometry_data* for a gate must be in the form of an array, where each element of the array represents a promoter activity. The data could represent one representative experiment or could be obtained by averaging the distributions from experimental replicates. In the *cytometry_data* data structure, an input signal level (input) is paired with a histogram representing the measured output level (*output_bins, output_counts*). Importantly, cytometry data must be consistently binned for all gates; the number of bins (NBINS), minimal value (MIN), and maximum value (MAX) must be used when generating the histogram for a cytometry sample. This consistency is required to propagate distributions through each layer in the circuit. Typical values would be:

```
NBINS = 250.
MAX   = 100.    (RPUs, linear space, not log space)
MIN   = 0.001.  (RPUs, linear space, not log space)
```

To generate the cytometry data for Cello, fluorescence values from flow cytometry must first be converted from arbitrary units to RPUs. This process is described in

Chapter 4. Thus, for a single response function, each titration point with a defined input level has a corresponding histogram. These discrete titration points are used to generate a continuous distribution response function by overlaying histograms on the median determined by the Hill equation. As a result, any input RPU value can produce a predicted output histogram. In Cello, a single histogram can still be used to generate histograms for the entire distribution response function (if the parameters for the average response function are provided), though histograms from 8 or more titration points are the expected use case.

input: This attribute specifies the input promoter activity of the distribution.

required: yes

allowed values: a single RPU value for the current titration.

output_bins: For the given input level of the current titration, the bins of output levels are listed.

required: yes

allowed values: an array of values (RPU) specifying the histogram bins. The array length (number of bins) must be the same for all histograms specified in this collection. When generating the histogram, binning must be done in log space (log10 RPU), but the bin values must be specified in linear space (RPU). For consistency in the UCF, all RPU values are specified in linear space.

output_counts: counts for each output bin.

required: yes

allowed values: an array of counts for the histogram. Counts can be integers, or fractional counts: Cello will normalize each histogram so the sum of all counts

equals 10,000. The array length must be the same as the *output_ bins* array, and must be the same for all histograms specified in this collection. Scientific notation (E) is allowed for very low fractional count values.

Thus, the JSON object for each cytometry titration point consists of the attributes *input*, *output_ bins*, and *output_ counts*. Each titration point is listed in an array representing the full response function titration, and this array of objects is the value of an attribute called *cytometry_ data*:

cytometry_ data: list of titrations for characterizing a response function using flow cytometry

required: yes

allowed values: a list of JSON objects for each titration point, described above (*input*, *output_ bins*, and *output_ counts*). Note that the number of titration points can range from 1 to N, where N is any number of titration points used in response function characterization. Unlike the histogram binning, which must be consistent across all gates, the number of titration points can differ across gates.

An example of a JSON object in the gate_cytometry collection is shown below. For readability, only two titrations with a small number of values are shown. Notice that the output_bins are consistent, but the output_counts vary between titrations. See the provided UCF for an example of this organization.

```
{
  "collection": "gate_cytometry",
  "gate_name": "B3_BM3R1",
  "cytometry_data": [
    {
      ''input'': 0.018,
      ''output_bins":   [ ... 0.129, 0.136, 0.144, 0.152, 0.161, ...],
      ''output_counts": [ ... 0.005, 0.002, 0.003, 0.001, 0.0005,...]
    },
    {
      ''input'': 0.028,
      ''output_bins":   [ ... 0.129, 0.136, 0.144, 0.152, 0.161, ...],
      ''output_counts": [ ... 0.002, 0.003, 0.004, 0.004, 0.003, ...],
    }
  (only two titration points shown)
}
```

6.2.10 Gate toxicity

This collection specifies the growth curve for each gate. The object has two data attributes: "input" is the input level (promoter activity in standard units) driving expression of the gate, and "growth" contains growth values normalized by a control cell population. A value of 1.0 indicates full growth, and a value of 0.0 indicates no growth. Growth measurements could take the form of OD_{600} endpoint measurements, cytometry events, growth rates, colony counts, etc. In this manuscript, growth values are OD_{600} endpoints when P_{Tac} is driving the NOT gate, divided by the OD_{600} of cells with P_{Tac} driving YFP. Because titrations are discrete and input levels during circuit simulation are continuous, an input promoter activity requires interpolation of the data to generate a growth value for that specific input level. Interpolation is used to compute a growth value that is a weighted average of the two nearest *growth* values, where the weight is determined by proximity of the input level to the two nearest *input* values. If an input level is less than the lowest input in the growth curve, the

first growth value is used. If an input level is greater than the highest input in the growth curve, the last growth value is used.


```
{
  "collection": "gate_toxicity",
  "gate_name": "A2_AmtR",
  "input": [
    0.004,
    0.007,
    0.012,
    0.034,
    0.062,
    0.099,
    0.144,
    0.247,
    0.418,
    0.739,
    1.012,
    2.078
],
  "growth": [
    1.03,
    0.99,
    0.99,
    1.04,
    1.08,
    1.05,
    1.08,
    1.1,
    1.05,
    1.03,
    0.81,
    0.71
  ]
}
```


gate_name: The name of the gate.

required: yes

allowed values: gate name must correspond to the correct name in the *gate* collection.

input: promoter activity driving expression of the regulator.

required: yes

allowed values: input RPU values used in the growth curve titration. Values must be ordered from low to high.

growth: normalized cell growth measurement.

required: yes

allowed values: growth values normalized by a control cell population. The length of the array must be equal to the length of the *input* array. After Cello reads the data, values < 0.0 will be set to 0.0, values > 1.0 will be set to 1.0.

6.2.11 Eugene rules

This collection specifies constraints on the physical layout of the circuit, written using Eugene (Oberortner *et al*, 2014). These rules are used in tandem with combinatorial design algorithms to build the DNA sequence of the circuit that is the output of Cello. In addition, these rules will be enforced if more than one construct is designed. Our use of the Eugene rules is organized into two attributes in this UCF collection: *eugene_part_rules* and *eugene_gate_rules*. Due to the hierarchical Eugene specification, rules are applied to the parts within a gate device separately from the rules applied to the gates within a circuit device (Section V.E). Note that "device" is a Eugene term for a collection of parts, or a collection of other devices. Rule is also a Eugene term, where a rule is applied to a device to constrain its design.

Part rules. The part rules in the Eco1C1G1T1 UCF enforce the following. Roadblocking requires that a promoter (*e.g.*, P_{SrpR}) be in the upstream position of a

NOR gate. In other words, when P_{SrpR} is in the forward orientation, P_{SrpR} must be the first part in the gate device (and this is inverted if the gate is in the reverse orientation). Roadblocking is not the only reason to enforce particular promoter orders and this is a generalized approach to constraining a particular part order.

example:

STARTSWITH pSrpR

In Eco1C1G1T1, we also use rules to constrain a preferred promoter order (Figure 4-13).

example:

pAmtR BEFORE pBetI

For a given gate device in the Eugene file (Section V.E), all part rules are parsed from the UCF, but a part rule will not be applied to the gate device if the gate does not contain a part with that name.

Gate rules. Gate rules can be used to specify the order in which regulators appear in the circuit construct. Because the UCF must accommodate any possible circuit assignment, a combinatorial list of rules is needed to constrain the gate order for any assignment. Given a circuit assignment, Cello scans all of the *eugene_gate_rules*, and if any gate name in a rule is absent from the current assignment, that rule is omitted. For example, the rule below would be omitted if PhlF is not assigned to the current circuit.

example:

gate_PhlF BEFORE gate_BM3R1

Note that some gate rules do not list a gate name, such as ALL_FORWARD. For the ALL_FORWARD rule, no gate names are parsed, so it is not possible for the ALL_FORWARD rule to be omitted based on the gates assigned to the circuit.

examples:

ALL_FORWARD

(all gates will be in the 5' to 3' forward orientation)

ALTERNATE_ORIENTATION

(example: gate 1 forward, gate 2 reverse, gate 3 forward, gate 4 reverse)

(example: gate 1 reverse, gate 2 forward, gate 3 reverse, gate 4 forward)

SOME_REVERSE

(one or more gates will be in the 3' to 5' reverse orientation)

eugene_part_rules: rules that constrain parts within a gate device

required: no

allowed values: Any Eugene rule can be used(Oberortner *et al*, 2014). Part names in the Eugene rules must be identical to the part names specified in the *parts* collection of the UCF.

eugene_gate_rules: rules that constrain gate order/orientation within a circuit device

required: no

allowed values: Any Eugene rule can be used(Oberortner *et al*, 2014). Gate names must follow a naming convention for correct automatic generation of the Eugene file. This convention uses the "gate_" prefix followed by the repressor name, for example gate_PhlF.

6.2.12 Genetic location

This collection defines the physical location of the sensor module, circuit module, and output module in the context of the plasmid and/or genomic landing pads. The sensor module encodes the transcription factors that are required by the sensors. This could be a transcription factor and its necessary transcription/translation parts (*e.g.*, promoter, RBS, AraC, terminator). If all of the machinery is endogenous to the host organism, then there may not be a sensor module. The circuit module encompasses the circuit designed by Cello. The output module contains the circuit regulable promoter(s) assigned to the output gates in the circuit, followed by the actuator gene(s) of interest. Note that the locations for each of the modules may differ from the context in which the gates were characterized. Attributes are provided for correction factors to be included (*e.g.*, to correct for copy number or the fluorescent protein used).

locations: This attribute lists genetic locations that will be referred to by name in the sensor, circuit, and output location attributes.

name: the name of the plasmid or genomic landing pad.

required: yes

allowed values: Any string, but the name needs to be internally consistent when referred to in the *sensor_module_location, circuit_module_location,* and *output_module_location* objects.

file: The NCBI sequence file for the plasmid.

required: yes

allowed values: all lines of the GenBank file. The GenBank file can have annotations prior to the first base pair of the DNA sequence, but the sequence should have the following format.

```
ORIGIN
        1 gcttcctcgc tcactgactc gctgcacgag gcagacctca gcgctagcgg agtgtatact
       61 ggcttactat gttggcactg atgagggtgt cagtgaagtg cttcatgtgg caggagaaaa
      121 aaggctgcac cggtgcgtca gcagaatatg tgatacagga tatattccgc ttcctcgctc
      181 actgactcgc tacgctcggt cgttcgactg cggcgagcgg aaatggctta cgaacggggc
      241 ggagatttcc tggaagatgc caggaagata cttaacaggg aagtgagagg gccgcggcaa
      301 agccgttttt ccataggctc cgccccctg acaagcatca cgaaatctga cgctcaaatc
      361 agtggtggcg aaacccgaca ggactataaa gataccaggc gtttccccctg gcggctccct
      421 cgtgcgctct cctgttcctg cctttcggtt taccggtgtc attccgctgt tatggccgcg
      481 tttgtctcat tccacgcctg acactcagtt ccgggtaggc agttcgctcc aagctggact
```

The JSON collection for *genetic_ locations* is given below, but DNA sequence files are not shown. Note that each of the locations is structured as an array of objects, where each object is a location. The example only includes one location per array, but there might be designs that require multiple sensor modules. It is also possible to insert sensor modules in series, rather than concatenating individual sensor modules ahead of time. Thus, the UCF is written to accommodate a list of locations for each type of module (sensor, circuit, output).

```
{
  ''collection'': ''genetic_locations'',

  ''locations'' : [
    {
      ''name'' : ''pAN1717'',
      ''file'' : FULL GENBANK FILE NOT SHOWN, SEE BELOW
    },
    {
      ''name'' : ''pAN1201'',
      ''file'' : FULL GENBANK FILE NOT SHOWN, SEE BELOW
    },
    {
      ''name'' : ''pAN4020'',
      ''file'' : FULL GENBANK FILE NOT SHOWN, SEE BELOW
    }
  ]

  ''sensor_module_location'' :
     [
       {''location_name'': ''pAN1201'',
        ''bp_range'' : [58, 556]
       }
     ]

  ''circuit_module_location'' :
     [
       {''location_name'': ''pAN1201'',
        ''bp_range'' : [58, 556]
       }
     ]

  ''output_module_location'' :
     [
       {''location_name'': ''pAN4020'',
        ''bp_range'' : [953, 953],
        ''unit_conversion'' : 0.40
       }
     ]
}
```

sensor_module_location: genetic location where the sensor module (if any) will be inserted. The sensor module encodes transcription factors that regulate the circuit input promoters, and contain all of the necessary parts for expression (constitutive promoter, RBS, CDS, terminator).

required: yes

circuit_module_location: genetic location where the circuit module will be inserted. The circuit module is designed by Cello, and it contains the user-defined input promoters, and parts from the Cello gates library.

required: yes

output_module_location: genetic location where the output module will be inserted. Expression of the output module/modules is/are driven by the promoters assigned by Cello.

required: no

location_name: the name of the GenBank file listed in the *locations* attribute where the module(s) will be incorporated.

bp_range: the starting and ending base pair numbers in the GenBank file where the module will be inserted.

allowed values: to insert without removing any bases, the start, end base pair numbers will be the same. If region of DNA is removed during cloning, for example, a region between two restriction sites, the start, and base pair numbers should span that range.

unit_conversion: If the output plasmid differs from the plasmid used to characterize the circuit, a conversion factor may be necessary. One example would be if it has a different copy number. In the case of the Eco1C1G1T1 UCF, we measured a conversion factor of 0.40 to convert promoter activities on p15A (RPU) to promoter activities on pSC101 (RPU) (Methods). All output levels (RPU) are multiplied by this conversion factor.

The sensor, circuit, and/or output modules could be inserted into the genome, rather than plasmids. Here we provide an example of a possible specification for choosing a genomic site for the circuit output module. This example specifies genomic

landing pad that is highly expressed and is amenable to large sequence insertions in any orientation.

```
    ''output_module_genomic_locations'' : [

        ''organism'': ''Escherichia coli str. K-12 substr. DH10B'',

        ''taxid'': 316385,

        ''location_name'': ''atpI-gidB intergenic region'',

        ''bp_range'': [4018174, 4018497],

        ''flanking_upstream_sequence'': ''ATACGGTGCGCCCCCGTGATTTCAAACAATAA'',
        ''flanking_downstream_sequence'': ''TTGTGATATTTTCACTAATGACTTATTTTCTGCT''
    ]
```

7 Appendix: References

<douglas@crockford.com> DC The application/json Media Type for JavaScript Object Notation (JSON). Available at: https://tools.ietf.org/html/rfc4627 [Accessed March 11, 2015]

Aiba H (2007) Mechanism of RNA silencing by Hfq-binding small RNAs. *Curr. Opin. Microbiol.* 10: 134–139

Alon U (2007) An Introduction to Systems Biology: Design Principles of Biological Circuits CRC Press

Altschul SF, Gish W, Miller W, Myers EW & Lipman DJ (1990) Basic local alignment search tool. *J. Mol. Biol.* 215: 403–410

An W & Chin JW (2009) Synthesis of orthogonal transcription-translation networks. *Proc. Natl. Acad. Sci.* 106: 8477–8482

Anderson JC, Voigt CA & Arkin AP (2007) Environmental signal integration by a modular AND gate. *Mol. Syst. Biol.* 3: Available at: http://www.nature.com/msb/journal/v3/n1/full/msb4100173.html [Accessed September 9, 2013]

Andrianantoandro E, Basu S, Karig DK & Weiss R (2006) Synthetic biology: new engineering rules for an emerging discipline. *Mol. Syst. Biol.* 2: Available at: http://www.nature.com/msb/journal/v2/n1/full/msb4100073.html [Accessed September 9, 2013]

Ang J, Harris E, Hussey BJ, Kil R & McMillen DR (2013) Tuning Response Curves for Synthetic Biology. *ACS Synth. Biol.* Available at: http://dx.doi.org/10.1021/sb4000564 [Accessed September 7, 2013]

Arkin AP & Fletcher DA (2006) Fast, cheap and somewhat in control. *Genome Biol.* 7: 1–6

Ausländer S, Ausländer D, Müller M, Wieland M & Fussenegger M (2012) Programmable single-cell mammalian biocomputers. *Nature* 487: 123–127

Bailey TL, Williams N, Misleh C & Li WW (2006) MEME: discovering and analyzing DNA and protein sequence motifs. *Nucleic Acids Res.* 34: W369–W373

Balagaddé FK, You L, Hansen CL, Arnold FH & Quake SR (2005) Long-Term Monitoring of Bacteria Undergoing Programmed Population Control in a Microchemostat. *Science* 309: 137–140

Basu S, Gerchman Y, Collins CH, Arnold FH & Weiss R (2005) A synthetic multicellular system for programmed pattern formation. *Nature* 434: 1130–1134

Basu S, Mehreja R, Thiberge S, Chen M-T & Weiss R (2004) Spatiotemporal control of gene expression with pulse-generating networks. *Proc. Natl. Acad. Sci. U. S. A.* 101: 6355–6360

Bayer TS, Widmaier DM, Temme K, Mirsky EA, Santi DV & Voigt CA (2009) Synthesis of Methyl Halides from Biomass Using Engineered Microbes. *J. Am. Chem. Soc.* 131: 6508–6515

Beal J, Lu T & Weiss R (2011) Automatic Compilation from High-Level Biologically-Oriented Programming Language to Genetic Regulatory Networks. *PLoS ONE* 6: e22490

Beal J, Weiss R, Densmore D, Adler A, Appleton E, Babb J, Bhatia S, Davidsohn N, Haddock T, Loyall J, Schantz R, Vasilev V & Yaman F (2012) An End-to-End Workflow for Engineering of Biological Networks from High-Level Specifications. *ACS Synth. Biol.* 1: 317–331

Beerli RR & Barbas CF (2002a) Engineering polydactyl zinc-finger transcription factors. *Nat. Biotechnol.* 20: 135–141

Beerli RR & Barbas CF (2002b) Engineering polydactyl zinc-finger transcription factors. *Nat. Biotechnol.* 20: 135–141

Bikard D, Jiang W, Samai P, Hochschild A, Zhang F & Marraffini LA (2013) Programmable repression and activation of bacterial gene expression using an engineered CRISPR-Cas system. *Nucleic Acids Res.*: gkt520

Bilitchenko L, Liu A, Cheung S, Weeding E, Xia B, Leguia M, Anderson JC & Densmore D (2011) Eugene – A Domain Specific Language for Specifying and

Constraining Synthetic Biological Parts, Devices, and Systems. *PLoS ONE* 6: e18882

Bintu L, Buchler NE, Garcia HG, Gerland U, Hwa T, Kondev J, Kuhlman T & Phillips R (2005) Transcriptional regulation by the numbers: applications. *Curr. Opin. Genet. Dev.* 15: 125–135

Blanchard AE, Celik V & Lu T (2014) Extinction, coexistence, and localized patterns of a bacterial population with contact-dependent inhibition. *BMC Syst. Biol.* 8: 23

Blattner FR, Plunkett G, Bloch CA, Perna NT, Burland V, Riley M, Collado-Vides J, Glasner JD, Rode CK, Mayhew GF, Gregor J, Davis NW, Kirkpatrick HA, Goeden MA, Rose DJ, Mau B & Shao Y (1997) The Complete Genome Sequence of Escherichia coli K-12. *Science* 277: 1453–1462

Boch J, Scholze H, Schornack S, Landgraf A, Hahn S, Kay S, Lahaye T, Nickstadt A & Bonas U (2009a) Breaking the Code of DNA Binding Specificity of TAL-Type III Effectors. *Science* 326: 1509–1512

Boch J, Scholze H, Schornack S, Landgraf A, Hahn S, Kay S, Lahaye T, Nickstadt A & Bonas U (2009b) Breaking the code of DNA binding specificity of TAL-type III effectors. *Science* 326: 1509–1512

Boni IV, Artamonova VS, Tzareva NV & Dreyfus M (2001) Non-canonical mechanism for translational control in bacteria: synthesis of ribosomal protein S1. *EMBO J.* 20: 4222–4232

Bonnet J, Subsoontorn P & Endy D (2012) Rewritable digital data storage in live cells via engineered control of recombination directionality. *Proc. Natl. Acad. Sci.* 109: 8884–8889

Bonnet J, Yin P, Ortiz ME, Subsoontorn P & Endy D (2013) Amplifying Genetic Logic Gates. *Science* 340: 599–603

Boos W & Böhm A (2000) Learning new tricks from an old dog: MalT of the Escherichia coli maltose system is part of a complex regulatory network. *Trends Genet.* 16: 404–409

Bowen TA, Zdunek JK & Medford JI (2008) Cultivating plant synthetic biology from systems biology. *New Phytol.* 179: 583–587

Boyle PM & Silver PA (2012) Parts plus pipes: Synthetic biology approaches to metabolic engineering. *Metab. Eng.* 14: 223–232

Brantl S & Wagner EGH (2000) Antisense RNA-mediated transcriptional attenuation: an in vitro study of plasmid pT181. *Mol. Microbiol.* 35: 1469–1482

Brayton R & Mishchenko A (2010) ABC: An Academic Industrial-Strength Verification Tool. In *Computer Aided Verification*, Touili T Cook B & Jackson P (eds) pp 24–40. Springer Berlin Heidelberg Available at: http://link.springer.com/chapter/10.1007/978-3-642-14295-6_5 [Accessed March 11, 2015]

Brewster RC, Jones DL & Phillips R (2012) Tuning Promoter Strength through RNA Polymerase Binding Site Design in Escherichia coli. *PLoS Comput Biol* 8: e1002811

Brown S & Vranesic Z (2013) Fundamentals of Digital Logic with Verilog Design 3 edition. New York: McGraw-Hill Education

Buchler NE & Cross FR (2009) Protein sequestration generates a flexible ultrasensitive response in a genetic network. *Mol. Syst. Biol.* 5: 272

Buchler NE, Gerland U & Hwa T (2003) On schemes of combinatorial transcription logic. *Proc. Natl. Acad. Sci.* 100: 5136–5141

Buchler NE & Louis M (2008) Molecular Titration and Ultrasensitivity in Regulatory Networks. *J. Mol. Biol.* 384: 1106–1119

Burge S, Parkinson GN, Hazel P, Todd AK & Neidle S (2006) Quadruplex DNA: sequence, topology and structure. *Nucleic Acids Res.* 34: 5402–5415

Callura JM, Cantor CR & Collins JJ (2012) Genetic switchboard for synthetic biology applications. *Proc. Natl. Acad. Sci.* 109: 5850–5855

Callura JM, Dwyer DJ, Isaacs FJ, Cantor CR & Collins JJ (2010) Tracking, tuning, and terminating microbial physiology using synthetic riboregulators. *Proc. Natl. Acad. Sci.* 107: 15898–15903

Cambray G, Guimaraes JC, Mutalik VK, Lam C, Mai Q-A, Thimmaiah T, Carothers JM, Arkin AP & Endy D (2013) Measurement and modeling of intrinsic transcription terminators. *Nucleic Acids Res.* 41: 5139–5148

Campbell RE, Tour O, Palmer AE, Steinbach PA, Baird GS, Zacharias DA & Tsien RY (2002) A monomeric red fluorescent protein. *Proc. Natl. Acad. Sci.* 99: 7877–7882

Canton B, Labno A & Endy D (2008a) Refinement and standardization of synthetic biological parts and devices. *Nat. Biotechnol.* 26: 787–793

Canton B, Labno A & Endy D (2008b) Refinement and standardization of synthetic biological parts and devices. *Nat. Biotechnol.* 26: 787–793

Cardinale S, Joachimiak MP & Arkin AP (2013) Effects of Genetic Variation on the E. coli Host-Circuit Interface. *Cell Rep.* 4: 231–237

Casadesús J & Low D (2006) Epigenetic Gene Regulation in the Bacterial World. *Microbiol. Mol. Biol. Rev.* 70: 830–856

Chaikind B, Kilambi KP, Gray JJ & Ostermeier M (2012) Targeted DNA Methylation Using an Artificially Bisected M.HhaI Fused to Zinc Fingers. *PLoS ONE* 7: e44852

Chandran D, Bergmann FT & Sauro HM (2009) TinkerCell: modular CAD tool for synthetic biology. *J. Biol. Eng.* 3: 19

Chappell J, Takahashi MK & Lucks JB (2015) Creating small transcription activating RNAs. *Nat. Chem. Biol.* 11: 214–220

Chelliserrykattil J, Cai G & Ellington AD (2001) A combined in vitro / in vivo selection for polymerases with novel promoter specificities. *BMC Biotechnol.* 1: 13

Chen D & Arkin AP (2012a) Sequestration-based bistability enables tuning of the switching boundaries and design of a latch. *Mol. Syst. Biol.* 8: Available at: http://www.nature.com/msb/journal/v8/n1/full/msb201252.html [Accessed August 16, 2013]

Chen D & Arkin AP (2012b) Sequestration-based bistability enables tuning of the switching boundaries and design of a latch. *Mol. Syst. Biol.* 8: Available at: http://www.nature.com/msb/journal/v8/n1/full/msb201252.html [Accessed September 10, 2013]

Chen S, Zhang H, Shi H, Ji W, Feng J, Gong Y, Yang Z & Ouyang Q (2012) Automated Design of Genetic Toggle Switches with Predetermined Bistability. *ACS Synth. Biol.* 1: 284–290

Chen Y-J, Liu P, Nielsen AAK, Brophy JAN, Clancy K, Peterson T & Voigt CA (2013) Characterization of 582 natural and synthetic terminators and quantification of their design constraints. *Nat. Methods* 10: 659–664

Chen YY & Smolke CD (2011) From DNA to Targeted Therapeutics: Bringing Synthetic Biology to the Clinic. *Sci. Transl. Med.* 3: 106ps42

Chubiz LM & Rao CV (2008) Computational design of orthogonal ribosomes. *Nucleic Acids Res.* 36: 4038–4046

Clancy K & Voigt CA (2010) Programming cells: towards an automated 'Genetic Compiler'. *Curr. Opin. Biotechnol.* 21: 572–581

Cong L, Ran FA, Cox D, Lin S, Barretto R, Habib N, Hsu PD, Wu X, Jiang W, Marraffini LA & Zhang F (2013) Multiplex Genome Engineering Using CRISPR/Cas Systems. *Science* 339: 819–823

Cormack BP, Valdivia RH & Falkow S (1996) FACS-optimized mutants of the green fluorescent protein (GFP). *Gene* 173: 33–38

Cox RS, Surette MG & Elowitz MB (2007) Programming gene expression with combinatorial promoters. *Mol. Syst. Biol.* 3: n/a-n/a

Cradick TJ, Fine EJ, Antico CJ & Bao G (2013) CRISPR/Cas9 systems targeting β-globin and CCR5 genes have substantial off-target activity. *Nucleic Acids Res.* 41: 9584–9592

Czar MJ, Anderson JC, Bader JS & Peccoud J (2009a) Gene synthesis demystified. *Trends Biotechnol.* 27: 63–72

Czar MJ, Cai Y & Peccoud J (2009b) Writing DNA with GenoCAD™. *Nucleic Acids Res.* 37: W40–W47

Dahiyat BI & Mayo SL (1997) De Novo Protein Design: Fully Automated Sequence Selection. *Science* 278: 82–87

Daniel R, Rubens JR, Sarpeshkar R & Lu TK (2013a) Synthetic analog computation in living cells. *Nature* advance online publication: Available at:

http://www.nature.com.libproxy.mit.edu/nature/journal/vaop/ncurrent/full/nature12148.html [Accessed August 26, 2013]

Daniel R, Rubens JR, Sarpeshkar R & Lu TK (2013b) Synthetic analog computation in living cells. *Nature* 497: 619–623

Daniel R, Rubens JR, Sarpeshkar R & Lu TK (2013c) Synthetic analog computation in living cells. *Nature* 497: 619–623

Danino T, Mondragón-Palomino O, Tsimring L & Hasty J (2010) A synchronized quorum of genetic clocks. *Nature* 463: 326–330

Davis JH, Rubin AJ & Sauer RT (2011) Design, construction and characterization of a set of insulated bacterial promoters. *Nucleic Acids Res.* 39: 1131–1141

de Smit MH & van Duin J (2003) Translational Standby Sites: How Ribosomes May Deal with the Rapid Folding Kinetics of mRNA. *J. Mol. Biol.* 331: 737–743

Del Vecchio D, Ninfa AJ & Sontag ED (2008) Modular cell biology: retroactivity and insulation. *Mol. Syst. Biol.* 4: Available at: http://europepmc.org/abstract/MED/18277378/reload=0;jsessionid=i6ZJiOYfjoroTQLK2osF.28 [Accessed September 6, 2013]

Desai TA, Rodionov DA, Gelfand MS, Alm EJ & Rao CV (2009) Engineering transcription factors with novel DNA-binding specificity using comparative genomics. *Nucleic Acids Res.* 37: 2493–2503

Desjarlais JR & Berg JM (1992) Toward rules relating zinc finger protein sequences and DNA binding site preferences. *Proc. Natl. Acad. Sci.* 89: 7345–7349

Dingermann T, Frank-Stoll U, Werner H, Wissmann A, Hillen W, Jacquet M & Marschalek R (1992) RNA polymerase III catalysed transcription can be regulated in Saccharomyces cerevisiae by the bacterial tetracycline repressor-operator system. *EMBO J.* 11: 1487–1492

Drubin DA, Way JC & Silver PA (2007) Designing biological systems. *Genes Dev.* 21: 242–254

Durai S, Bosley A, Abulencia AB, Chandrasegaran S & Ostermeier M (2006) A bacterial one-hybrid selection system for interrogating zinc finger-DNA interactions. *Comb. Chem. High Throughput Screen.* 9: 301–311

Durfee T, Nelson R, Baldwin S, Plunkett G, Burland V, Mau B, Petrosino JF, Qin X, Muzny DM, Ayele M, Gibbs RA, Csörgő B, Pósfai G, Weinstock GM & Blattner FR (2008) The Complete Genome Sequence of Escherichia coli DH10B: Insights into the Biology of a Laboratory Workhorse. *J. Bacteriol.* 190: 2597–2606

Egbert RG & Klavins E (2012) Fine-tuning gene networks using simple sequence repeats. *Proc. Natl. Acad. Sci.* 109: 16817–16822

Ellis T, Wang X & Collins JJ (2009) Diversity-based, model-guided construction of synthetic gene networks with predicted functions. *Nat. Biotechnol.* 27: 465–471

Elowitz MB & Leibler S (2000a) A synthetic oscillatory network of transcriptional regulators. *Nature* 403: 335–338

Elowitz MB & Leibler S (2000b) A synthetic oscillatory network of transcriptional regulators. *Nature* 403: 335–338

Endy D (2005) Foundations for engineering biology. *Nature* 438: 449–453

Endy D (2011) Drew Endy: Better Computing for the Things We Care About Most. *N. Y. Times* Available at: http://www.nytimes.com/2011/12/06/science/drew-endy-better-computing-for-the-things-we-care-about-most.html [Accessed March 6, 2015]

Engler C, Gruetzner R, Kandzia R & Marillonnet S (2009) Golden Gate Shuffling: A One-Pot DNA Shuffling Method Based on Type IIs Restriction Enzymes. *PLoS ONE* 4: e5553

Entus R, Aufderheide B & Sauro HM (2007) Design and implementation of three incoherent feed-forward motif based biological concentration sensors. *Syst. Synth. Biol.* 1: 119–128

Estrem ST, Gaal T, Ross W & Gourse RL (1998) Identification of an UP element consensus sequence for bacterial promoters. *Proc. Natl. Acad. Sci.* 95: 9761–9766

Esvelt KM, Carlson JC & Liu DR (2011) A system for the continuous directed evolution of biomolecules. *Nature* 472: 499–503

Esvelt KM, Mali P, Braff JL, Moosburner M, Yaung SJ & Church GM (2013) Orthogonal Cas9 proteins for RNA-guided gene regulation and editing. *Nat. Methods* 10: 1116–1121

Farzadfard F, Perli SD & Lu TK (2013) Tunable and Multifunctional Eukaryotic Transcription Factors Based on CRISPR/Cas. *ACS Synth. Biol.* 2: 604–613

Fath S, Bauer AP, Liss M, Spriestersbach A, Maertens B, Hahn P, Ludwig C, Schäfer F, Graf M & Wagner R (2011) Multiparameter RNA and codon optimization: a standardized tool to assess and enhance autologous mammalian gene expression. *PloS One* 6: e17596

Feist AM, Henry CS, Reed JL, Krummenacker M, Joyce AR, Karp PD, Broadbelt LJ, Hatzimanikatis V & Palsson BØ (2007) A genome-scale metabolic reconstruction for Escherichia coli K-12 MG1655 that accounts for 1260 ORFs and thermodynamic information. *Mol. Syst. Biol.* 3: n/a-n/a

Friedland AE, Lu TK, Wang X, Shi D, Church G & Collins JJ (2009) Synthetic Gene Networks that Count. *Science* 324: 1199–1202

Fu G, Lees RS, Nimmo D, Aw D, Jin L, Gray P, Berendonk TU, White-Cooper H, Scaife S, Kim Phuc H, Marinotti O, Jasinskiene N, James AA & Alphey L (2010) Female-specific flightless phenotype for mosquito control. *Proc. Natl. Acad. Sci. U. S. A.* 107: 4550–4554

Fu Y, Foden JA, Khayter C, Maeder ML, Reyon D, Joung JK & Sander JD (2013a) High-frequency off-target mutagenesis induced by CRISPR-Cas nucleases in human cells. *Nat. Biotechnol.* advance online publication: Available at: http://www.nature.com/nbt/journal/vaop/ncurrent/full/nbt.2623.html [Accessed September 6, 2013]

Fu Y, Foden JA, Khayter C, Maeder ML, Reyon D, Joung JK & Sander JD (2013b) High-frequency off-target mutagenesis induced by CRISPR-Cas nucleases in human cells. *Nat. Biotechnol.* 31: 822–826

Fu Y, Sander JD, Reyon D, Cascio VM & Joung JK (2014) Improving CRISPR-Cas nuclease specificity using truncated guide RNAs. *Nat. Biotechnol.* 32: 279–284

Gaj T, Mercer AC, Gersbach CA, Gordley RM & Barbas CF (2011) Structure-guided reprogramming of serine recombinase DNA sequence specificity. *Proc. Natl. Acad. Sci.* 108: 498–503

Galdzicki M, Clancy KP, Oberortner E, Pocock M, Quinn JY, Rodriguez CA, Roehner N, Wilson ML, Adam L, Anderson JC, Bartley BA, Beal J, Chandran D, Chen J, Densmore D, Endy D, Grünberg R, Hallinan J, Hillson NJ, Johnson JD, et al (2014) The Synthetic Biology Open Language (SBOL) provides a community

standard for communicating designs in synthetic biology. *Nat. Biotechnol.* 32: 545–550

Gao Y & Zhao Y (2014) Self-processing of ribozyme-flanked RNAs into guide RNAs in vitro and in vivo for CRISPR-mediated genome editing. *J. Integr. Plant Biol.* 56: 343–349

Gardner TS, Cantor CR & Collins JJ (2000a) Construction of a genetic toggle switch in Escherichia coli. *Nature* 403: 339–342

Gardner TS, Cantor CR & Collins JJ (2000b) Construction of a genetic toggle switch in Escherichia coli. *Nature* 403: 339–342

Garg A, Lohmueller JJ, Silver PA & Armel TZ (2012a) Engineering synthetic TAL effectors with orthogonal target sites. *Nucleic Acids Res.* 40: 7584–7595

Garg A, Lohmueller JJ, Silver PA & Armel TZ (2012b) Engineering synthetic TAL effectors with orthogonal target sites. *Nucleic Acids Res.* 40: 7584–7595

Gatz C & Quail PH (1988) Tn10-encoded tet repressor can regulate an operator-containing plant promoter. *Proc. Natl. Acad. Sci. U. S. A.* 85: 1394–1397

Ghosh D, Kohli AG, Moser F, Endy D & Belcher AM (2012) Refactored M13 Bacteriophage as a Platform for Tumor Cell Imaging and Drug Delivery. *ACS Synth. Biol.* 1: 576–582

Gibson DG, Glass JI, Lartigue C, Noskov VN, Chuang R-Y, Algire MA, Benders GA, Montague MG, Ma L, Moodie MM, Merryman C, Vashee S, Krishnakumar R, Assad-Garcia N, Andrews-Pfannkoch C, Denisova EA, Young L, Qi Z-Q, Segall-Shapiro TH, Calvey CH, et al (2010) Creation of a Bacterial Cell Controlled by a Chemically Synthesized Genome. *Science* 329: 52–56

Gibson DG, Young L, Chuang R-Y, Venter JC, Hutchison CA & Smith HO (2009) Enzymatic assembly of DNA molecules up to several hundred kilobases. *Nat. Methods* 6: 343–345

Gilbert LA, Larson MH, Morsut L, Liu Z, Brar GA, Torres SE, Stern-Ginossar N, Brandman O, Whitehead EH, Doudna JA, Lim WA, Weissman JS & Qi LS (2013) CRISPR-Mediated Modular RNA-Guided Regulation of Transcription in Eukaryotes. *Cell* 154: 442–451

Gong F, Ito K, Nakamura Y & Yanofsky C (2001) The mechanism of tryptophan induction of tryptophanase operon expression: Tryptophan inhibits release factor-mediated cleavage of TnaC-peptidyl-tRNAPro. *Proc. Natl. Acad. Sci.* 98: 8997–9001

Goodman DB, Church GM & Kosuri S (2013) Causes and Effects of N-Terminal Codon Bias in Bacterial Genes. *Science*: 1241934

Gossen M & Bujard H (1992) Tight control of gene expression in mammalian cells by tetracycline-responsive promoters. *Proc. Natl. Acad. Sci. U. S. A.* 89: 5547–5551

Greber D & Fussenegger M (2007) Mammalian synthetic biology: Engineering of sophisticated gene networks. *J. Biotechnol.* 130: 329–345

Green AA, Silver PA, Collins JJ & Yin P (2014) Toehold Switches: De-Novo-Designed Regulators of Gene Expression. *Cell* 159: 925–939

Grkovic S, Brown MH, Schumacher MA, Brennan RG & Skurray RA (2001) The staphylococcal QacR multidrug regulator binds a correctly spaced operator as a pair of dimers. *J. Bacteriol.* 183: 7102–7109

Groth AC & Calos MP (2004) Phage Integrases: Biology and Applications. *J. Mol. Biol.* 335: 667–678

Guet CC, Elowitz MB, Hsing W & Leibler S (2002) Combinatorial Synthesis of Genetic Networks. *Science* 296: 1466–1470

Guilinger JP, Thompson DB & Liu DR (2014) Fusion of catalytically inactive Cas9 to FokI nuclease improves the specificity of genome modification. *Nat. Biotechnol.* 32: 577–582

Guss AM, Rother M, Zhang JK, Kulkarni G & Metcalf WW (2008) New methods for tightly regulated gene expression and highly efficient chromosomal integration of cloned genes for Methanosarcina species. *Archaea Vanc. BC* 2: 193–203

Ham TS, Lee SK, Keasling JD & Arkin AP (2006) A tightly regulated inducible expression system utilizing the fim inversion recombination switch. *Biotechnol. Bioeng.* 94: 1–4

Ham TS, Lee SK, Keasling JD & Arkin AP (2008) Design and Construction of a Double Inversion Recombination Switch for Heritable Sequential Genetic Memory. *PLoS ONE* 3: e2815

Hasty J (2012) Engineered Microbes for Therapeutic Applications. *ACS Synth. Biol.* 1: 438–439

Hasty J, McMillen D & Collins JJ (2002) Engineered gene circuits. *Nature* 420: 224–230

Helbl V, Tiebel B & Hillen W (1998) Stepwise selection of TetR variants recognizing tet operator 6C with high affinity and specificity. *J. Mol. Biol.* 276: 319–324

Henry CS, Broadbelt LJ & Hatzimanikatis V (2007) Thermodynamics-Based Metabolic Flux Analysis. *Biophys. J.* 92: 1792–1805

Hirano N, Muroi T, Takahashi H & Haruki M (2011) Site-specific recombinases as tools for heterologous gene integration. *Appl. Microbiol. Biotechnol.* 92: 227–239

Holtz WJ & Keasling JD (2010) Engineering Static and Dynamic Control of Synthetic Pathways. *Cell* 140: 19–23

Hooshangi S, Thiberge S & Weiss R (2005) Ultrasensitivity and noise propagation in a synthetic transcriptional cascade. *Proc. Natl. Acad. Sci. U. S. A.* 102: 3581–3586

Horvath P & Barrangou R (2010) CRISPR/Cas, the Immune System of Bacteria and Archaea. *Science* 327: 167–170

Hsu PD, Scott DA, Weinstein JA, Ran FA, Konermann S, Agarwala V, Li Y, Fine EJ, Wu X, Shalem O, Cradick TJ, Marraffini LA, Bao G & Zhang F (2013a) DNA targeting specificity of RNA-guided Cas9 nucleases. *Nat. Biotechnol.* 31: 827–832

Hsu PD, Scott DA, Weinstein JA, Ran FA, Konermann S, Agarwala V, Li Y, Fine EJ, Wu X, Shalem O, Cradick TJ, Marraffini LA, Bao G & Zhang F (2013b) DNA targeting specificity of RNA-guided Cas9 nucleases. *Nat. Biotechnol.* 31: 827–832

http://parts.igem.org/

Huh JH, Kittleson JT, Arkin AP & Anderson JC (2013) Modular Design of a Synthetic Payload Delivery Device. *ACS Synth. Biol.* 2: 418–424

Hunter S, Jones P, Mitchell A, Apweiler R, Attwood TK, Bateman A, Bernard T, Binns D, Bork P, Burge S, de Castro E, Coggill P, Corbett M, Das U, Daugherty

L, Duquenne L, Finn RD, Fraser M, Gough J, Haft D, et al (2012) InterPro in 2011: new developments in the family and domain prediction database. *Nucleic Acids Res.* 40: D306-312

Hurt JA, Thibodeau SA, Hirsh AS, Pabo CO & Joung JK (2003) Highly specific zinc finger proteins obtained by directed domain shuffling and cell-based selection. *Proc. Natl. Acad. Sci. U. S. A.* 100: 12271–12276

Huynh L & Tagkopoulos I (2014) Optimal Part and Module Selection for Synthetic Gene Circuit Design Automation. *ACS Synth. Biol.* 3: 556–564

Ideker T, Galitski T & Hood L (2001) A NEW APPROACH TO DECODING LIFE: Systems Biology. *Annu. Rev. Genomics Hum. Genet.* 2: 343–372

Isaacs FJ, Dwyer DJ, Ding C, Pervouchine DD, Cantor CR & Collins JJ (2004) Engineered riboregulators enable post-transcriptional control of gene expression. *Nat. Biotechnol.* 22: 841–847

Itzkovitz S, Tlusty T & Alon U (2006) Coding limits on the number of transcription factors. *BMC Genomics* 7: 239

Jang SS, Oishi KT, Egbert RG & Klavins E (2012) Specification and Simulation of Synthetic Multicelled Behaviors. *ACS Synth. Biol.* 1: 365–374

Jayanthi S, Nilgiriwala KS & Del Vecchio D (2013) Retroactivity Controls the Temporal Dynamics of Gene Transcription. *ACS Synth. Biol.* 2: 431–441

Jensen PR & Hammer K (1998) The Sequence of Spacers between the Consensus Sequences Modulates the Strength of Prokaryotic Promoters. *Appl. Environ. Microbiol.* 64: 82–87

Jiang W, Bikard D, Cox D, Zhang F & Marraffini LA (2013) RNA-guided editing of bacterial genomes using CRISPR-Cas systems. *Nat. Biotechnol.* 31: 233–239

Jinek M, Chylinski K, Fonfara I, Hauer M, Doudna JA & Charpentier E (2012) A Programmable Dual-RNA–Guided DNA Endonuclease in Adaptive Bacterial Immunity. *Science* 337: 816–821

Kang Z, Wang X, Li Y, Wang Q & Qi Q (2012) Small RNA RyhB as a potential tool used for metabolic engineering in Escherichia coli. *Biotechnol. Lett.* 34: 527–531

Katz RH & Boriello G (2004) Contemporary Logic Design

Keasling JD (2008) Synthetic Biology for Synthetic Chemistry. *ACS Chem. Biol.* 3: 64–76

Kelly JR, Rubin AJ, Davis JH, Ajo-Franklin CM, Cumbers J, Czar MJ, de Mora K, Glieberman AL, Monie DD & Endy D (2009a) Measuring the activity of BioBrick promoters using an in vivo reference standard. *J. Biol. Eng.* 3: 4

Kelly JR, Rubin AJ, Davis JH, Ajo-Franklin CM, Cumbers J, Czar MJ, Mora K de, Glieberman AL, Monie DD & Endy D (2009b) Measuring the activity of BioBrick promoters using an in vivo reference standard. *J. Biol. Eng.* 3: 4

Khalil AS & Collins JJ (2010) Synthetic biology: applications come of age. *Nat. Rev. Genet.* 11: 367–379

Khalil AS, Lu TK, Bashor CJ, Ramirez CL, Pyenson NC, Joung JK & Collins JJ (2012) A Synthetic Biology Framework for Programming Eukaryotic Transcription Functions. *Cell* 150: 647–658

Kiani S, Beal J, Ebrahimkhani MR, Huh J, Hall RN, Xie Z, Li Y & Weiss R (2014) CRISPR transcriptional repression devices and layered circuits in mammalian cells. *Nat. Methods* 11: 723–726

Kimelman A, Levy A, Sberro H, Kidron S, Leavitt A, Amitai G, Yoder-Himes DR, Wurtzel O, Zhu Y, Rubin EM & Sorek R (2012) A vast collection of microbial genes that are toxic to bacteria. *Genome Res.* 22: 802–809

Kittle JD, Simons RW, Lee J & Kleckner N (1989) Insertion sequence IS10 anti-sense pairing initiates by an interaction between the 5′ end of the target RNA and a loop in the anti-sense RNA. *J. Mol. Biol.* 210: 561–572

Kittleson JT, Cheung S & Anderson JC (2011) Rapid optimization of gene dosage in E. coli using DIAL strains. *J. Biol. Eng.* 5: 10

Kittleson JT, Wu GC & Anderson JC (2012) Successes and failures in modular genetic engineering. *Curr. Opin. Chem. Biol.* 16: 329–336

Konermann S, Brigham MD, Trevino A, Hsu PD, Heidenreich M, Le Cong, Platt RJ, Scott DA, Church GM & Zhang F (2013) Optical control of mammalian endogenous transcription and epigenetic states. *Nature* advance online publication: Available at: http://www.nature.com/nature/journal/vnfv/ncurrent/full/nature12466.html [Accessed August 23, 2013]

Kosuri S & Church GM (2014) Large-scale de novo DNA synthesis: technologies and applications. *Nat. Methods* 11: 499–507

Kosuri S, Goodman DB, Cambray G, Mutalik VK, Gao Y, Arkin AP, Endy D & Church GM (2013) Composability of regulatory sequences controlling transcription and translation in Escherichia coli. *Proc. Natl. Acad. Sci.* 110: 14024–14029

Kramer BP & Fussenegger M (2005) Hysteresis in a synthetic mammalian gene network. *Proc. Natl. Acad. Sci. U. S. A.* 102: 9517–9522

Krueger M, Scholz O, Wisshak S & Hillen W (2007) Engineered Tet repressors with recognition specificity for the tetO-4C5G operator variant. *Gene* 404: 93–100

Kuscu C, Arslan S, Singh R, Thorpe J & Adli M (2014) Genome-wide analysis reveals characteristics of off-target sites bound by the Cas9 endonuclease. *Nat. Biotechnol.* 32: 677–683

Kwok R (2010) Five hard truths for synthetic biology. *Nat. News* 463: 288–290

Larson MH, Gilbert LA, Wang X, Lim WA, Weissman JS & Qi LS (2013) CRISPR interference (CRISPRi) for sequence-specific control of gene expression. *Nat. Protoc.* 8: 2180–2196

Laursen BS, Sørensen HP, Mortensen KK & Sperling-Petersen HU (2005) Initiation of Protein Synthesis in Bacteria. *Microbiol. Mol. Biol. Rev.* 69: 101–123

Lee JY, Sung BH, Yu BJ, Lee JH, Lee SH, Kim MS, Koob MD & Kim SC (2008) Phenotypic engineering by reprogramming gene transcription using novel artificial transcription factors in Escherichia coli. *Nucleic Acids Res.* 36: e102–e102

Lee T-H & Maheshri N (2012a) A regulatory role for repeated decoy transcription factor binding sites in target gene expression. *Mol. Syst. Biol.* 8: 576

Lee T-H & Maheshri N (2012b) A regulatory role for repeated decoy transcription factor binding sites in target gene expression. *Mol. Syst. Biol.* 8: Available at: http://www.nature.com/doifinder/10.1038/msb.2012.7 [Accessed September 2, 2013]

Levine E, Zhang Z, Kuhlman T & Hwa T (2007) Quantitative Characteristics of Gene Regulation by Small RNA. *PLoS Biol* 5: e229

Lim WA (2010) Designing customized cell signalling circuits. *Nat. Rev. Mol. Cell Biol.* 11: 393–403

Liu CC, Qi L, Lucks JB, Segall-Shapiro TH, Wang D, Mutalik VK & Arkin AP (2012) An adaptor from translational to transcriptional control enables predictable assembly of complex regulation. *Nat. Methods* 9: 1088–1094

Lo T-M, Tan MH, Hwang IY & Chang MW (2013) Designing a synthetic genetic circuit that enables cell density-dependent auto-regulatory lysis for macromolecule release. *Chem. Eng. Sci.* 103: 29–35

Løbner-Olesen A, Skovgaard O & Marinus MG (2005) Dam methylation: coordinating cellular processes. *Curr. Opin. Microbiol.* 8: 154–160

Looger LL, Dwyer MA, Smith JJ & Hellinga HW (2003) Computational design of receptor and sensor proteins with novel functions. *Nature* 423: 185–190

Lou C, Stanton B, Chen Y-J, Munsky B & Voigt CA (2012a) Ribozyme-based insulator parts buffer synthetic circuits from genetic context. *Nat. Biotechnol.* 30: 1137–1142

Lou C, Stanton B, Chen Y-J, Munsky B & Voigt CA (2012b) Ribozyme-based insulator parts buffer synthetic circuits from genetic context. *Nat. Biotechnol.* 30: 1137–1142

Lu TK, Khalil AS & Collins JJ (2009) Next-generation synthetic gene networks. *Nat. Biotechnol.* 27: 1139–1150

Lucks JB, Qi L, Mutalik VK, Wang D & Arkin AP (2011) Versatile RNA-sensing transcriptional regulators for engineering genetic networks. *Proc. Natl. Acad. Sci.* 108: 8617–8622

Lucks JB, Qi L, Whitaker WR & Arkin AP (2008) Toward scalable parts families for predictable design of biological circuits. *Curr. Opin. Microbiol.* 11: 567–573

Lutz R & Bujard H (1997) Independent and tight regulation of transcriptional units in Escherichia coli via the LacR/O, the TetR/O and AraC/I1-I2 regulatory elements. *Nucleic Acids Res.* 25: 1203–1210

Lycett GJ, Kafatos FC & Loukeris TG (2004) Conditional expression in the malaria mosquito Anopheles stephensi with Tet-On and Tet-Off systems. *Genetics* 167: 1781–1790

Mali P, Aach J, Stranges PB, Esvelt KM, Moosburner M, Kosuri S, Yang L & Church GM (2013) CAS9 transcriptional activators for target specificity screening and paired nickases for cooperative genome engineering. *Nat. Biotechnol.* 31: 833–838

Man S, Cheng R, Miao C, Gong Q, Gu Y, Lu X, Han F & Yu W (2011) Artificial trans-encoded small non-coding RNAs specifically silence the selected gene expression in bacteria. *Nucleic Acids Res.* 39: e50–e50

Mangan S & Alon U (2003) Structure and function of the feed-forward loop network motif. *Proc. Natl. Acad. Sci.* 100: 11980–11985

Marchisio MA & Stelling J (2011) Automatic Design of Digital Synthetic Gene Circuits. *PLoS Comput. Biol.* 7: Available at: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3048778/ [Accessed March 11, 2015]

Markham NR & Zuker M (2008) UNAFold. In *Bioinformatics*, Keith JM (ed) pp 3–31. Humana Press Available at: http://link.springer.com/protocol/10.1007/978-1-60327-429-6_1 [Accessed September 7, 2013]

Marraffini LA & Sontheimer EJ (2010) CRISPR interference: RNA-directed adaptive immunity in bacteria and archaea. *Nat. Rev. Genet.* 11: 181–190

Materials and methods are available as supplementary materials on Science Online.

McAdams HH & Arkin A (1998) Simulation of Prokaryotic Genetic Circuits. *Annu. Rev. Biophys. Biomol. Struct.* 27: 199–224

McAdams HH & Shapiro L (1995) Circuit simulation of genetic networks. *Science* 269: 650–656

Mercer AC, Gaj T, Fuller RP & Barbas CF (2012) Chimeric TALE recombinases with programmable DNA sequence specificity. *Nucleic Acids Res.* 40: 11163–11172

Mey MD, Maertens J, Lequeux GJ, Soetaert WK & Vandamme EJ (2007) Construction and model-based analysis of a promoter library for E. coli: an indispensable tool for metabolic engineering. *BMC Biotechnol.* 7: 34

Miller JC, Holmes MC, Wang J, Guschin DY, Lee Y-L, Rupniewski I, Beausejour CM, Waite AJ, Wang NS, Kim KA, Gregory PD, Pabo CO & Rebar EJ (2007) An

improved zinc-finger nuclease architecture for highly specific genome editing. *Nat. Biotechnol.* 25: 778–785

Miller JC, Tan S, Qiao G, Barlow KA, Wang J, Xia DF, Meng X, Paschon DE, Leung E, Hinkley SJ, Dulay GP, Hua KL, Ankoudinova I, Cost GJ, Urnov FD, Zhang HS, Holmes MC, Zhang L, Gregory PD & Rebar EJ (2011) A TALE nuclease architecture for efficient genome editing. *Nat. Biotechnol.* 29: 143–148

Monod J & Jacob F (1961) General Conclusions: Teleonomic Mechanisms in Cellular Metabolism, Growth, and Differentiation. *Cold Spring Harb. Symp. Quant. Biol.* 26: 389–401

Moon TS, Clarke EJ, Groban ES, Tamsir A, Clark RM, Eames M, Kortemme T & Voigt CA (2011) Construction of a Genetic Multiplexer to Toggle between Chemosensory Pathways in Escherichia coli. *J. Mol. Biol.* 406: 215–227

Moon TS, Lou C, Tamsir A, Stanton BC & Voigt CA (2012a) Genetic programs constructed from layered logic gates in single cells. *Nature* 491: 249–253

Moon TS, Lou C, Tamsir A, Stanton BC & Voigt CA (2012b) Genetic programs constructed from layered logic gates in single cells. *Nature* 491: 249–253

Morbitzer R, Römer P, Boch J & Lahaye T (2010a) Regulation of selected genome loci using de novo-engineered transcription activator-like effector (TALE)-type transcription factors. *Proc. Natl. Acad. Sci.* 107: 21617–21622

Morbitzer R, Römer P, Boch J & Lahaye T (2010b) Regulation of selected genome loci using de novo-engineered transcription activator-like effector (TALE)-type transcription factors. *Proc. Natl. Acad. Sci.* 107: 21617–21622

Moscou MJ & Bogdanove AJ (2009) A Simple Cipher Governs DNA Recognition by TAL Effectors. *Science* 326: 1501–1501

Moser F, Broers NJ, Hartmans S, Tamsir A, Kerkman R, Roubos JA, Bovenberg R & Voigt CA (2012a) Genetic Circuit Performance under Conditions Relevant for Industrial Bioreactors. *ACS Synth. Biol.* 1: 555–564

Moser F, Broers NJ, Hartmans S, Tamsir A, Kerkman R, Roubos JA, Bovenberg R & Voigt CA (2012b) Genetic circuit performance under conditions relevant for industrial bioreactors. *ACS Synth. Biol.* 1: 555–564

Mukherji S & van Oudenaarden A (2009) Synthetic biology: understanding biological design from synthetic circuits. *Nat. Rev. Genet.* 10: 859–871

Mutalik VK, Guimaraes JC, Cambray G, Lam C, Christoffersen MJ, Mai Q-A, Tran AB, Paull M, Keasling JD, Arkin AP & Endy D (2013a) Precise and reliable gene expression via standard transcription and translation initiation elements. *Nat. Methods* 10: 354–360

Mutalik VK, Guimaraes JC, Cambray G, Mai Q-A, Christoffersen MJ, Martin L, Yu A, Lam C, Rodriguez C, Bennett G, Keasling JD, Endy D & Arkin AP (2013b) Quantitative estimation of activity and quality for collections of functional genetic elements. *Nat. Methods* 10: 347–353

Mutalik VK, Qi L, Guimaraes JC, Lucks JB & Arkin AP (2012a) Rationally designed families of orthogonal RNA regulators of translation. *Nat. Chem. Biol.* 8: 447–454

Mutalik VK, Qi L, Guimaraes JC, Lucks JB & Arkin AP (2012b) Rationally designed families of orthogonal RNA regulators of translation. *Nat. Chem. Biol.* 8: 447–454

Myers CJ, Barker N, Jones K, Kuwahara H, Madsen C & Nguyen N-PD (2009) iBioSim: a tool for the analysis and design of genetic circuits. *Bioinformatics* 25: 2848–2849

Na D, Yoo SM, Chung H, Park H, Park JH & Lee SY (2013) Metabolic engineering of Escherichia coli using synthetic small regulatory RNAs. *Nat. Biotechnol.* 31: 170–174

Nagy A (2000) Cre recombinase: The universal reagent for genome tailoring. *genesis* 26: 99–109

Naville M & Gautheret D (2010) Premature terminator analysis sheds light on a hidden world of bacterial transcriptional attenuation. *Genome Biol.* 11: 1–17

Nielsen AA, Segall-Shapiro TH & Voigt CA Advances in genetic circuit design: novel biochemistries, deep part mining, and precision gene expression. *Curr. Opin. Chem. Biol.* Available at: http://www.sciencedirect.com/science/article/pii/S1367593113001713 [Accessed November 20, 2013]

Nielsen AA & Voigt CA (2014) Multi-input CRISPR/Cas genetic circuits that interface host regulatory networks. *Mol. Syst. Biol.* 10: 763

Nielsen J, Fussenegger M, Keasling J, Lee SY, Liao JC, Prather K & Palsson B (2014) Engineering synergy in biotechnology. *Nat. Chem. Biol.* 10: 319–322

Nissim L, Perli SD, Fridkin A, Perez-Pinera P & Lu TK (2014) Multiplexed and Programmable Regulation of Gene Networks with an Integrated RNA and CRISPR/Cas Toolkit in Human Cells. *Mol. Cell* 54: 698–710

Oberortner E, Bhatia S, Lindgren E & Densmore D (2014) A Rule-Based Design Specification Language for Synthetic Biology. *J Emerg Technol Comput Syst* 11: 25:1–25:19

Orth P, Schnappinger D, Hillen W, Saenger W & Hinrichs W (2000) Structural basis of gene regulation by the tetracycline inducible Tet repressor-operator system. *Nat. Struct. Biol.* 7: 215–219

Pattanayak V, Lin S, Guilinger JP, Ma E, Doudna JA & Liu DR (2013) High-throughput profiling of off-target DNA cleavage reveals RNA-programmed Cas9 nuclease specificity. *Nat. Biotechnol.* 31: 839–843

Pedraza JM & Oudenaarden A van (2005) Noise Propagation in Gene Networks. *Science* 307: 1965–1969

Politz MC, Copeland MF & Pfleger BF (2013a) Artificial repressors for controlling gene expression in bacteria. *Chem. Commun.* 49: 4325–4327

Politz MC, Copeland MF & Pfleger BF (2013b) Artificial repressors for controlling gene expression in bacteria. *Chem. Commun. Camb. Engl.* 49: 4325–4327

Ptashne M (1986a) A genetic switch: Gene control and phage. lambda. Available at: http://www.osti.gov/scitech/biblio/5413898 [Accessed September 8, 2013]

Ptashne M (1986b) Gene regulation by proteins acting nearby and at a distance. *Nature* 322: 697–701

Purnick PEM & Weiss R (2009a) The second wave of synthetic biology: from modules to systems. *Nat. Rev. Mol. Cell Biol.* 10: 410–422

Purnick PEM & Weiss R (2009b) The second wave of synthetic biology: from modules to systems. *Nat. Rev. Mol. Cell Biol.* 10: 410–422

Qi L, Haurwitz RE, Shao W, Doudna JA & Arkin AP (2012) RNA processing enables predictable programming of gene expression. *Nat. Biotechnol.* 30: 1002–1006

Qi LS & Arkin AP (2014) A versatile framework for microbial engineering using synthetic non-coding RNAs. *Nat. Rev. Microbiol.* 12: 341–354

Qi LS, Larson MH, Gilbert LA, Doudna JA, Weissman JS, Arkin AP & Lim WA (2013) Repurposing CRISPR as an RNA-Guided Platform for Sequence-Specific Control of Gene Expression. *Cell* 152: 1173–1183

Qian L & Winfree E (2011) Scaling Up Digital Circuit Computation with DNA Strand Displacement Cascades. *Science* 332: 1196–1201

Rackham O & Chin JW (2005) A network of orthogonal ribosome·mRNA pairs. *Nat. Chem. Biol.* 1: 159–166

Ramos JL, Martínez-Bueno M, Molina-Henares AJ, Terán W, Watanabe K, Zhang X, Gallegos MT, Brennan R & Tobes R (2005) The TetR family of transcriptional repressors. *Microbiol. Mol. Biol. Rev. MMBR* 69: 326–356

Ran FA, Hsu PD, Lin C-Y, Gootenberg JS, Konermann S, Trevino AE, Scott DA, Inoue A, Matoba S, Zhang Y & Zhang F (2013) Double Nicking by RNA-Guided CRISPR Cas9 for Enhanced Genome Editing Specificity. *Cell* 154: 1380–1389

Rao CV (2012) Expanding the synthetic biology toolbox: engineering orthogonal regulators of gene expression. *Curr. Opin. Biotechnol.* 23: 689–694

Raskin CA, Diaz GA & McAllister WT (1993) T7 RNA polymerase mutants with altered promoter specificities. *Proc. Natl. Acad. Sci.* 90: 3147–3151

Rhodius VA, Mutalik VK & Gross CA (2012) Predicting the strength of UP-elements and full-length E. coli σE promoters. *Nucleic Acids Res.* 40: 2907–2924

Rhodius VA, Segall-Shapiro TH, Ghodasara A, Orlova E, Tabakh H, Clancy K, Peterson TC, Gross CA & Voigt CA Design of orthogonal genetic switches based on a crosstalk map of ⊠s, anti-⊠s, and promoters. *Submiss.*

Rhodius VA, Segall-Shapiro TH, Sharon BD, Ghodasara A, Orlova E, Tabakh H, Burkhardt DH, Clancy K, Peterson TC, Gross CA & Voigt CA (2013) Design of orthogonal genetic switches based on a crosstalk map of σs, anti-σs, and promoters. *Mol. Syst. Biol.* 9: Available at:

http://www.nature.com/msb/journal/v9/n1/full/msb201358.html [Accessed November 8, 2013]

Rodrigo G & Jaramillo A (2013) AutoBioCAD: Full Biodesign Automation of Genetic Circuits. *ACS Synth. Biol.* 2: 230–236

Roehner N & Myers CJ (2014) Directed Acyclic Graph-Based Technology Mapping of Genetic Circuit Models. *ACS Synth. Biol.* 3: 543–555

Rosenfeld N, Young JW, Alon U, Swain PS & Elowitz MB (2005) Gene Regulation at the Single-Cell Level. *Science* 307: 1962–1965

Rud I, Jensen PR, Naterstad K & Axelsson L (2006) A synthetic promoter library for constitutive gene expression in Lactobacillus plantarum. *Microbiology* 152: 1011–1019

Rudell RL (1986) Multiple-Valued Logic Minimization for PLA Synthesis

Ruder WC, Lu T & Collins JJ (2011) Synthetic Biology Moving into the Clinic. *Science* 333: 1248–1252

Rudge TJ, Steiner PJ, Phillips A & Haseloff J (2012) Computational Modeling of Synthetic Microbial Biofilms. *ACS Synth. Biol.* 1: 345–352

Saez E, No D, West A & Evans RM (1997) Inducible gene expression in mammalian cells and transgenic mice. *Curr. Opin. Biotechnol.* 8: 608–616

Salis HM, Mirsky EA & Voigt CA (2009) Automated design of synthetic ribosome binding sites to control protein expression. *Nat. Biotechnol.* 27: 946–950

Santoro SW & Schultz PG (2002) Directed evolution of the site specificity of Cre recombinase. *Proc. Natl. Acad. Sci.* 99: 4185–4190

Sarpeshkar R (2010) Ultra Low Power Bioelectronics

Sauro HM, Hucka M, Finney A, Wellock C, Bolouri H, Doyle J & Kitano H (2003) Next Generation Simulation Tools: The Systems Biology Workbench and BioSPICE Integration. *OMICS J. Integr. Biol.* 7: 355–372

Schmeissner U, McKenney K, Rosenberg M & Court D (1984) Removal of a terminator structure by RNA processing regulates int gene expression. *J. Mol. Biol.* 176: 39–53

Segall-Shapiro TH, Meyer AJ, Ellington AD, Sontag ED & Voigt CA (2014) A 'resource allocator' for transcription based on a highly fragmented T7 RNA polymerase. *Mol. Syst. Biol.* 10: 742

Seghezzi N, Amar P, Koebmann B, Jensen PR & Virolle M-J (2011) The construction of a library of synthetic promoters revealed some specific features of strong Streptomyces promoters. *Appl. Microbiol. Biotechnol.* 90: 615–623

Shalem O, Sanjana NE, Hartenian E, Shi X, Scott DA, Mikkelson T, Heckl D, Ebert BL, Root DE, Doench JG & Zhang F (2013) Genome-Scale CRISPR-Cas9 Knockout Screening in Human Cells. *Science*: 1247005

Sharma V, Sakai Y, Smythe KA & Yokobayashi Y (2013) Knockdown of recA gene expression by artificial small RNAs in Escherichia coli. *Biochem. Biophys. Res. Commun.* 430: 256–259

Sharma V, Yamamura A & Yokobayashi Y (2012) Engineering Artificial Small RNAs for Conditional Gene Silencing in Escherichia coli. *ACS Synth. Biol.* 1: 6–13

Shis DL & Bennett MR (2013) Library of synthetic transcriptional AND gates built with split T7 RNA polymerase mutants. *Proc. Natl. Acad. Sci.* Available at: http://www.pnas.org/content/early/2013/03/08/1220157110 [Accessed May 13, 2013]

Siuti P, Yazbek J & Lu TK (2013) Synthetic circuits integrating logic and memory in living cells. *Nat. Biotechnol.* 31: 448–452

Sleight SC, Bartley BA, Lieviant JA & Sauro HM (2010) Designing and engineering evolutionary robust genetic circuits. *J. Biol. Eng.* 4: 1–20

Sleight SC & Sauro HM (2013) Visualization of Evolutionary Stability Dynamics and Competitive Fitness of Escherichia coli Engineered with Randomized Multigene Circuits. *ACS Synth. Biol.* Available at: http://dx.doi.org/10.1021/sb400055h [Accessed September 6, 2013]

Smanski MJ, Bhatia S, Zhao D, Park Y, Woodruff LBA, Giannoukos G, Ciulla D, Busby M, Calderon J, Nicol R, Gordon DB, Densmore D & Voigt CA (2014) Functional optimization of gene clusters by combinatorial design and assembly. *Nat. Biotechnol.* 32: 1241–1249

Sprinzak D & Elowitz MB (2005) Reconstruction of genetic circuits. *Nature* 438: 443–448

Stanton BC, Giles SS, Kruzel EK, Warren CL, Ansari AZ & Hull CM (2009) Cognate Site Identifier analysis reveals novel binding properties of the Sex Inducer homeodomain proteins of Cryptococcus neoformans. *Mol. Microbiol.* 72: 1334–1347

Stanton BC, Nielsen AAK, Tamsir A, Clancy K, Peterson T & Voigt CA (2013) Genomic Mining of Prokaryotic Repressors for Scalable Logic Gates.

Stanton BC, Nielsen AAK, Tamsir A, Clancy K, Peterson T & Voigt CA (2014) Genomic mining of prokaryotic repressors for orthogonal logic gates. *Nat. Chem. Biol.* 10: 99–105

Stricker J, Cookson S, Bennett MR, Mather WH, Tsimring LS & Hasty J (2008) A fast, robust and tunable synthetic gene oscillator. *Nature* 456: 516–519

Tabor JJ, Levskaya A & Voigt CA (2011) Multichromatic control of gene expression in Escherichia coli. *J. Mol. Biol.* 405: 315–324

Takahashi MK & Lucks JB (2013) A modular strategy for engineering orthogonal chimeric RNA transcription regulators. *Nucleic Acids Res.*: gkt452

Tamsir A, Tabor JJ & Voigt CA (2011a) Robust multicellular computing using genetically encoded NOR gates and chemical /`wires/'. *Nature* 469: 212–215

Tamsir A, Tabor JJ & Voigt CA (2011b) Robust multicellular computing using genetically encoded NOR gates and chemical 'wires'. *Nature* 469: 212–215

Tchetina E & Newman EB (1995) Identification of Lrp-regulated genes by inverse PCR and sequencing: regulation of two mal operons of Escherichia coli by leucine-responsive regulatory protein. *J. Bacteriol.* 177: 2679–2683

Temme K, Hill R, Segall-Shapiro TH, Moser F & Voigt CA (2012) Modular control of multiple pathways using engineered orthogonal T7 polymerases. *Nucleic Acids Res.* 40: 8773–8781

Teo WS & Chang MW (2014) Development and characterization of AND-gate dynamic controllers with a modular synthetic GAL1 core promoter in Saccharomyces cerevisiae. *Biotechnol. Bioeng.* 111: 144–151

The standard differs from the Kelly standard and contains: an insulating upstream terminator, a different spacer upstream of the promoter (as opposed to a

BioBricks prefix), RiboJ, RBS B0064, a different terminator, and three silent mutations to yfp.

Thirion JP & Hofnung M (1972) On Some Genetic Aspects of Phage Λ Resistance in E. Coli K12. *Genetics* 71: 207–216

This was done for 0x0E, 0x19, 0x1C, 0x38, 0x3D, 0x6E, 0x81, 0xB9, 0xC6, 0xC7, 0xBD, and 0xC8.

Thomas DE & Moorby PR (2002) The Verilog® Hardware Description Language Springer Science & Business Media

Thompson KE, Bashor CJ, Lim WA & Keating AE (2012) SYNZIP protein interaction toolbox: in vitro and in vivo specifications of heterospecific coiled-coil interaction domains. *ACS Synth. Biol.* 1: 118–129

Tigges M, Marquez-Lago TT, Stelling J & Fussenegger M (2009) A tunable synthetic mammalian oscillator. *Nature* 457: 309–312

Tinberg CE, Khare SD, Dou J, Doyle L, Nelson JW, Schena A, Jankowski W, Kalodimos CG, Johnsson K, Stoddard BL & Baker D (2013) Computational design of ligand-binding proteins with high affinity and selectivity. *Nature* 501: 212–216

Tsai SQ, Wyvekens N, Khayter C, Foden JA, Thapar V, Reyon D, Goodwin MJ, Aryee MJ & Joung JK (2014) Dimeric CRISPR RNA-guided FokI nucleases for highly specific genome editing. *Nat. Biotechnol.* 32: 569–576

Tummala SB, Welker NE & Papoutsakis ET (2003) Design of Antisense RNA Constructs for Downregulation of the Acetone Formation Pathway of Clostridium acetobutylicum. *J. Bacteriol.* 185: 1923–1934

Urbanowski ML, Lostroh CP & Greenberg EP (2004) Reversible acyl-homoserine lactone binding to purified Vibrio fischeri LuxR protein. *J. Bacteriol.* 186: 631–637

Voigt C (2011) Synthetic Biology: Computer Aided Design and DNA Assembly Academic Press

Wang HH, Isaacs FJ, Carr PA, Sun ZZ, Xu G, Forest CR & Church GM (2009) Programming cells by multiplex genome engineering and accelerated evolution. *Nature* 460: 894–898

Wang HH, Kim H, Cong L, Jeong J, Bang D & Church GM (2012) Genome-scale promoter engineering by coselection MAGE. *Nat. Methods* 9: 591–593

Wang T, Wei JJ, Sabatini DM & Lander ES (2013a) Genetic Screens in Human Cells Using the CRISPR/Cas9 System. *Science*: 1246981

Wang Y-H, Wei KY & Smolke CD (2013b) Synthetic Biology: Advancing the Design of Diverse Genetic Systems. *Annu. Rev. Chem. Biomol. Eng.* 4: 69–102

Warren CL, Kratochvil NCS, Hauschild KE, Foister S, Brezinski ML, Dervan PB, Phillips GN & Ansari AZ (2006) Defining the sequence-recognition profile of DNA-binding molecules. *Proc. Natl. Acad. Sci. U. S. A.* 103: 867–872

Weber W & Fussenegger M (2011) Molecular diversity--the toolbox for synthetic gene switches and networks. *Curr. Opin. Chem. Biol.* 15: 414–420

Weber W & Fussenegger M (2012) Emerging biomedical applications of synthetic biology. *Nat. Rev. Genet.* 13: 21–35

Weber W, Stelling J, Rimann M, Keller B, Daoud-El Baba M, Weber CC, Aubel D & Fussenegger M (2007) A synthetic time-delay circuit in mammalian cells and mice. *Proc. Natl. Acad. Sci. U. S. A.* 104: 2643–2648

Weiss R (2001) Cellular computation and communications using engineered genetic regulatory networks. Available at: http://dspace.mit.edu/handle/1721.1/8228 [Accessed March 6, 2015]

Weiss R & Jr TFK (2000) Engineered communications for microbial robotics. In *DNA Computing*, Condon A & Rozenberg G (eds) pp 1–16. Springer Berlin Heidelberg Available at: http://link.springer.com/chapter/10.1007/3-540-44992-2_1 [Accessed August 21, 2016]

Whitaker WR, Davis SA, Arkin AP & Dueber JE (2012) Engineering robust control of two-component system phosphotransfer using modular scaffolds. *Proc. Natl. Acad. Sci.* 109: 18090–18095

Wolfram S (2002) A New Kind of Science

Wu X, Scott DA, Kriz AJ, Chiu AC, Hsu PD, Dadon DB, Cheng AW, Trevino AE, Konermann S, Chen S, Jaenisch R, Zhang F & Sharp PA (2014) Genome-wide binding of the CRISPR endonuclease Cas9 in mammalian cells. *Nat. Biotechnol.* 32: 670–676

Yaman F, Bhatia S, Adler A, Densmore D & Beal J (2012) Automated Selection of Synthetic Biology Parts for Genetic Regulatory Networks. *ACS Synth. Biol.* 1: 332–344

Yang L, Nielsen AAK, Fernandez-Rodriguez J, McClune CJ, Laub MT, Lu TK & Voigt CA (2014) Permanent genetic memory with >1-byte capacity. *Nat. Methods* 11: 1261–1266

Yao AI, Fenton TA, Owsley K, Seitzer P, Larsen DJ, Sit H, Lau J, Nair A, Tantiongloc J, Tagkopoulos I & Facciotti MT (2013) Promoter Element Arising from the Fusion of Standard BioBrick Parts. *ACS Synth. Biol.* 2: 111–120

Yokobayashi Y, Weiss R & Arnold FH (2002a) Directed evolution of a genetic circuit. *Proc. Natl. Acad. Sci.* 99: 16587–16591

Yokobayashi Y, Weiss R & Arnold FH (2002b) Directed evolution of a genetic circuit. *Proc. Natl. Acad. Sci.* 99: 16587–16591

Yordanov B, Dalchau N, Grant PK, Pedersen M, Emmott S, Haseloff J & Phillips A (2014) A Computational Method for Automated Characterization of Genetic Components. *ACS Synth. Biol.* 3: 578–588

Zhan J, Ding B, Ma R, Ma X, Su X, Zhao Y, Liu Z, Wu J & Liu H (2010) Develop reusable and combinable designs for transcriptional logic gates. *Mol. Syst. Biol.* 6: Available at: http://www.nature.com/msb/journal/v6/n1/full/msb201042.html [Accessed August 25, 2013]

Zhang F, Carothers JM & Keasling JD (2012) Design of a dynamic sensor-regulator system for production of chemicals and fuels derived from fatty acids. *Nat. Biotechnol.* 30: 354–359

Zhang F, Cong L, Lodato S, Kosuri S, Church GM & Arlotta P (2011) Efficient construction of sequence-specific TAL effectors for modulating mammalian transcription. *Nat. Biotechnol.* 29: 149–153

Zhou Y, Zhu S, Cai C, Yuan P, Li C, Huang Y & Wei W (2014) High-throughput screening of a CRISPR/Cas9 library for functional genomics in human cells. *Nature* 509: 487–491