Cellular Computation and Communications using Engineered Genetic Regulatory Networks

by

Ron Weiss
B.A., Brandeis University (1992)
S.M., Massachusetts Institute of Technology (1994)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
September 2001

Author .....
Department of Electrical Engineering and Computer Science
August 24, 2001

Certified by .....
Thomas F. Knight, Jr.
Senior Research Scientist
Thesis Supervisor

Certified by .....
Gerald Jay Sussman
Matsushita Professor of Electrical Engineering, MIT
Thesis Supervisor

Certified by .....
Harold Abelson
Class of 1992 Professor of Computer Science and Engineering,MIT
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Cellular Computation and Communications using Engineered Genetic Regulatory Networks

by

Ron Weiss

Submitted to the Department of Electrical Engineering and Computer Science
on August 24, 2001, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

In this thesis, I present an engineering discipline for obtaining complex, predictable, and reliable cell behaviors by embedding biochemical logic circuits and programmed intercellular communications into cells. To accomplish this goal, I provide a well-characterized component library, a biocircuit design methodology, and software design tools. I have built and characterized an initial cellular gate library with biochemical gates that implement the NOT, IMPLIES, and AND logic functions in *E. coli* cells. The logic gates perform computation using DNA-binding proteins, small molecules that interact with these proteins, and segments of DNA that regulate the expression of the proteins. I introduce *genetic process engineering*, a methodology for modifying the DNA encoding of existing genetic elements to achieve the desired input/output behavior for constructing reliable circuits of significant complexity. I demonstrate the feasibility of digital computation in cells by building several operational *in-vivo* digital logic circuits, each composed of three gates that have been optimized by genetic process engineering. I also demonstrate engineered intercellular communications with programmed enzymatic activity and chemical diffusions to carry messages, using DNA from the *Vibrio fischeri lux* operon. The programmed communications is essential for obtaining coordinated behavior from cell aggregates.

In addition to the above experimental contributions, I have developed *BioSPICE*, a prototype software tool for biocircuit design. It supports both static and dynamic simulations and analysis of single cell environments and small cell aggregates. Finally, I present the Microbial Colony Language (MCL), a model for programming cell aggregates. The language is expressive enough for interesting applications, yet relies on simple primitives that can be mapped to the engineered biological processes described above.

Thesis Supervisor: Thomas F. Knight, Jr.
Title: Senior Research Scientist

Thesis Supervisor: Gerald Jay Sussman
Title: Matsushita Professor of Electrical Engineering, MIT

Thesis Supervisor: Harold Abelson
Title: Class of 1992 Professor of Computer Science and Engineering,MIT

Acknowledgments

I am indebted to my advisor, Tom Knight, for the original inspiration to do this work and for the frequent interactions in the lab. I will always admire Tom's vision and imagination, as well as his love for getting his hands wet and performing experiments. I am also very grateful to my advisors Gerry Sussman and Hal Abelson for the unique and wonderful environment they have created in the Switzerland Group, fostering camaraderie as well as non-traditional and imaginative thinking.

Special thanks to Nick Papadakis for providing support in the lab for just about everything imaginable. Thanks to Dylan Hirsch-Shell for constructing several plasmids and performing various experiments. Other past or current members of the Knight Lab have also been helpful: Matt Frank, Jered Floyd, Geo Homsy, and Rebecca Schulman. The 7.02 teaching staff, especially Debbie Kruzel, and Bridey Maxwell from the Salk Institute, helped get me started performing experiments in the lab.

Rahdika Nagpal, my office mate, has provided valuable feedback throughout my entire thesis. Stephen Adams and Daniel Coore implemented aspects of the software simulation infrastructure that I used for building the microbial colony simulator. Chris Hanson helped both in intellectual discussions and in maintaining the group's software and hardware infrastructure. I have also benefitted from interactions with other members of the Amorphous Computing Group: Eric Rauch, Jacob Katzanelson, Corrina Sherman, Piotr Mitros, Jake Beal, Bill Butera, Don Allen, Rebecca Frankel, Bob Hearn, Attila Kondacs, and Keith Winstein.

Anthony Zolnik, Becky Bisbee, Jeanne Darling, and Jessica Strong have always been very helpful to me and have certainly made my life easier and more fun at MIT.

Glenn Paradis, head of MIT's Flow Cytometry Core Facility, has provided valuable expertise to ensure that I was able to measure the behavior of my genetic circuits. Other people that have helped me study the behavior of the engineered cells include Michael Jennings from flow cytometry, Libby Shaw from the Center for Materials Science and Engineering, and Nicki Watson from the Whitehead Institute. Erik Winfree, Michael Elowitz, and Drew Endy have engaged me in many interesting discussions about biochemical and genetic computation.

Thanks to Jon Babb, a close friend, a fellow windsurfer, and an always interesting person to talk to about many different subjects.

Special thanks to my parents Dr. Zvi and Irit Weiss for their love, support, and the desire they have instilled in me to learn and excel. My daughter Abigail and son Ethan give me joy every day and have had patience and understanding for their Aba who had to stay late at work many nights and weekends to finish this thesis.

Most of all, I thank my wonderful wife Kim who has helped me throughout this entire process and deserves much credit for this thesis. She has been strong and supportive, and I truly appreciate all that she has done in raising our children while working full time, and providing a warm and loving environment for us as a family.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Genetic engineering with recombinant DNA technology is a powerful and widespread laboratory technique that enables biologists to redesign life forms by modifying or extending their DNA. It is currently used in a diverse set of applications that includes synthesis of pharmaceutical products, biochemical manufacturing, biomedical research, genetically modified crops and animals, and human therapeutics. In these applications, biologists typically modify organisms to express one or a few additional proteins. For example, a newly inserted protein will interact with the modified cells and improve a specific property of the organism, such as an enzyme that synthesizes growth hormones to increase crop yields. Alternatively, the modified cells serve as factories for manufacturing additional biochemicals that are completely independent of the cell (for example, human insulin). While already providing great benefits, existing genetic engineering applications only hint at the possibilities for harnessing cells to our benefit.

The goal of this thesis is to lay the foundations of an engineering discipline for building novel living systems with well-defined purposes and behaviors using standardized, well-characterized components. Cells are miniature, energy efficient, self-reproduce, and can manufacture biochemical products. These unique characteristics make cells attractive for many novel applications that require precise programmed control over the behavior of the cells. The applications include nanoscale fabrication, embedded intelligence in materials, sensor/effector arrays, patterned biomaterial man-

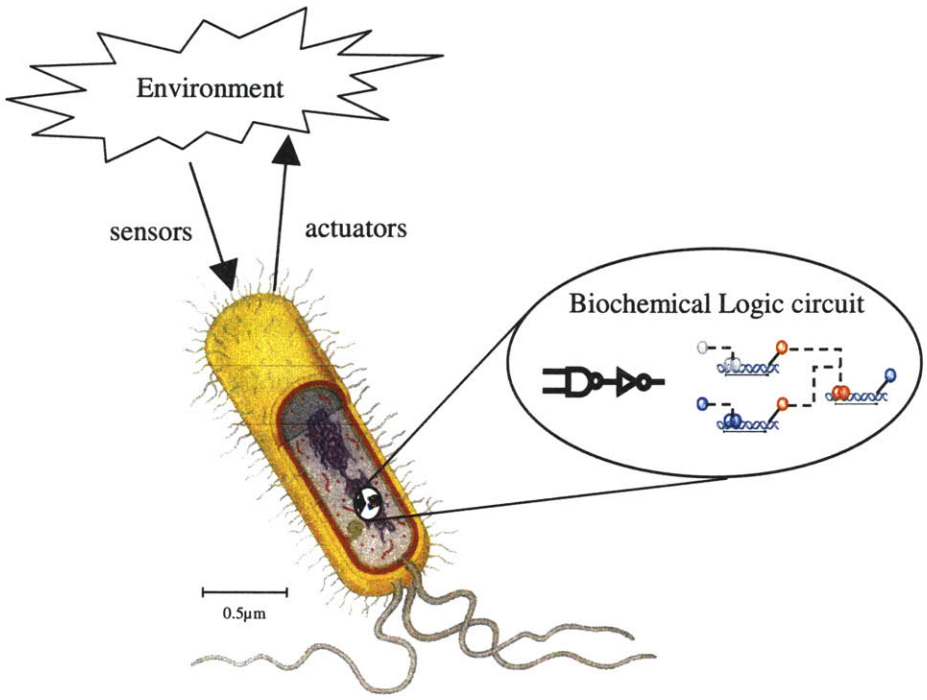


Figure 1-1: Embedding biochemical logic circuits in cells for internal computation and programmed intercellular communications, extending and modifying the behavior of cells and cell aggregates.

ufacturing, improved pharmaceutical synthesis, programmed therapeutics, and as a sophisticated tool for *in-vivo* studies of genetic regulatory networks. These applications require synthesis of sophisticated and reliable cell behaviors that instruct cells to make logic decisions based on factors such as existing environmental conditions and current cell state. For example, a cell may be programmed to secrete particular timed sequences of biochemicals depending on the type of messages sent by its neighbors. The approach proposed in this thesis for engineering the requisite precision control is to embed internal computation and programmed intercellular communications into the cells (Figure 1-1). The challenge here is to provide robust computation and communications using a substrate where reliability and reproducible results are difficult to achieve.

Biological organisms as an engineering substrate are currently difficult to modify and control because of the poor understanding of their complexity. Genetic modifi-

cations to cells often result in unpredictable and unreliable behavior. A single *Escherichia coli* bacterial cell contains approximately 10^{10} active molecules, about 10^7 of which are protein molecules. The cumulative interaction of these molecules with each other and the environment determines the behavior of the single cell. Although complex, these interactions are not arbitrary. Rather, cells are highly optimized information processing units that monitor their environment and continuously make decisions on how to react to the given conditions. Moreover, from prokaryotes such as bacteria to eukaryotes such as human cells, cells act both as individual units and as a contributing part of larger and complex multicellular systems or organisms. Given these attributes and inherent complexity, how can we successfully modify and harness biological organisms for our purposes?

1.1 Thesis Statement

This thesis focuses on programming *de-novo* behavior in individual cells and cell aggregates, and makes the following claim:

> Controlled gene expression using engineered *in-vivo* digital-logic circuits and intercellular communications enables programmed cell behavior that is complex, predictable, and reliable.

The approach integrates several layers that include a library of well-characterized simple components synthesized to have the appropriate behavior, a methodology for combining these components into complex intracellular circuitry and multicellular systems with predictable and reliable behavior, and software tools for design and analysis. The following section elaborates this approach.

1.2 Approach

The first step in making programmed cell behavior a practical and useful engineering discipline is to assemble a component library. For this purpose, I engineered cellular gates that implement the NOT, IMPLIES, and AND logic functions. These gates are

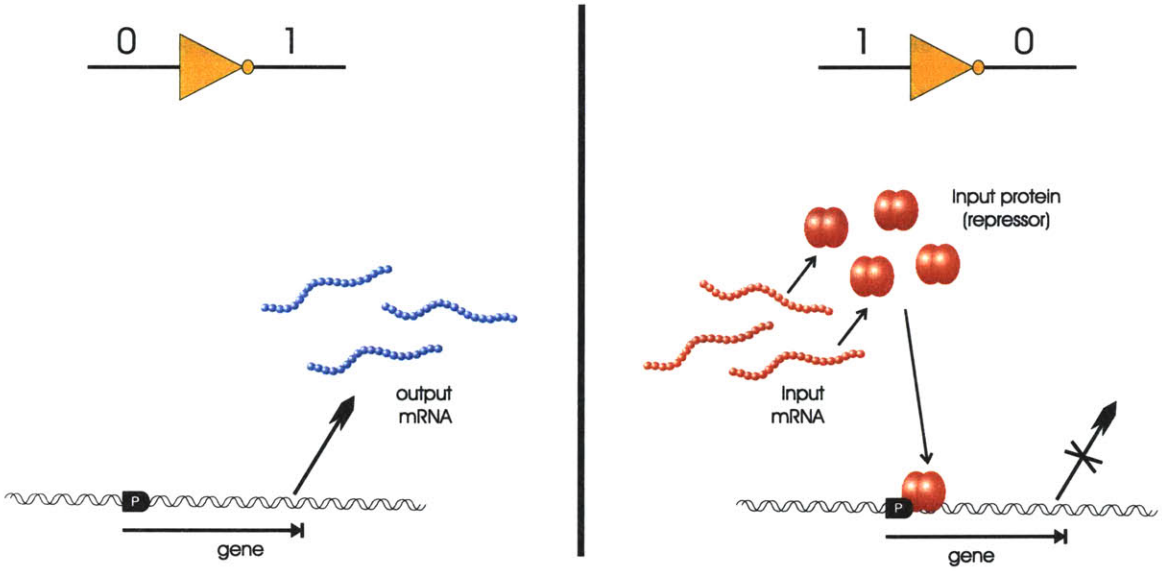


Figure 1-2: A simplified view of the two cases for a biochemical inverter. Here, the concentration of a particular messenger RNA (mRNA) molecule represents a logic signal. In the first case, the input mRNA is absent and the cell transcribes the gene for the output mRNA. In the second case, the input mRNA is present and the cell translates the input mRNA into the input protein. The input protein then binds specifically to the gene at the promoter site (labeled "P") and prevents the cell from synthesizing the output mRNA.

then combined into biochemical logic circuits for both intracellular computation and intercellular communications. In these biocircuits, chemical concentrations of specific messenger RNA (mRNA) and inducer molecules represent the logic signals. The logic gates perform computation and communications using mRNA, DNA-binding proteins, small inducer molecules that interact with these proteins, and segments of DNA that regulate the expression of the proteins. For example, Figure 1-2 describes how a cellular inverter achieves the two states in digital inversion using these genetic regulatory elements.

Given a library of components, biocircuit design is the process of assembling pre-existing components into logic circuits that implement specific behaviors. The most important element of biocircuit design is matching logic gates such that the couplings produce the correct behavior. Typically, naturally occurring components have widely varying kinetic characteristics and arbitrarily composing them into circuits is

not likely to work. In this thesis, I demonstrate *genetic process engineering* – modifying the DNA encoding of existing genetic elements until they achieve the desired behavior for constructing reliable circuits of significant complexity. The genetic modifications produce components that implement digital computation with good noise margins, signal restoration, and appropriate standard interfaces for complex system composition.

An important aspect of this thesis is engineering biological systems to exhibit digital behavior, because the digital abstraction is both convenient to use and feasible. The digital abstraction is a very useful programming paradigm because it offers a reliable and conceptually simple methodology for constructing complex behavior from a small number of simple components[88]. Digital computation provides reliability by reducing the noise in the system through signal restoration. For each component in the computation, the analog output signal represents the digital value better than the analog input signal. Engineers carefully combine these reliable components into complex systems that perform reliable computation. Experimental results in Chapter 4 describe *in-vivo* digital-logic circuits with good noise margins and signal restoration to demonstrate the feasibility of programming cells using digital computation. In the study of existing biological organisms, recent work[54, 52, 53, 64] suggests that cells routinely employ digital computation to make certain decisions that result in binary on/off behavior. Because the digital abstraction is both convenient to use and feasible, it offers a useful paradigm for programming cells and cell aggregates. However, much like desktop computers employ both digital and analog components, in the future I will also incorporate analog logic elements into engineered biological systems as the analog components become necessary for particular tasks.

In order to create novel biological systems, an engineer must be equipped with design and modeling software that prescribes how primitive components may be combined into complex systems with predictable and reproducible behavior. I present BioSPICE, a prototype tool that aids biocircuit designers manage the complexity of the substrate and achieve reliable systems. The inspiration to BioSPICE comes from the utility of tools such as SPICE[57] in the design of electrical circuits. BioSPICE

supports both static and dynamic analysis of single cell environments and small cell aggregates.

For obtaining coordinated aggregate cell behavior, I demonstrate programmed cell-to-cell communications using chemical diffusion to carry messages. Multicellular organisms create complex patterned structures from identical, unreliable components. This process of differentiation relies on communications between the cells that compose the system. Learning how to engineer such robust behavior from aggregates is important for having an improved understanding of distributed computing, a better understanding of the natural developmental process, and for engineering novel multicellular organisms with well-defined behaviors. Chemical diffusion is one of several communication mechanisms that can help achieve coordinated behavior in cell aggregates.

Finally, to achieve complex and reliable coordinated behavior, one must also consider the characteristics of the execution environment and its constraints. For example, cells frequently fail, their interconnect topology is constantly in flux, they have limited computational resources, and messages between cells are frequently lost. The Microbial Colony Language (MCL) is a programming and execution paradigm that takes into account the constraints of the substrate. It offers a model for programming cell aggregates that is simple enough for implementation in cells, yet expressive enough for interesting applications. The Microbial Colony Simulator supports simulations of the programmed behavior of aggregates that consist of up to ten thousand cells.

1.3 Summary of Contributions

In this dissertation, I make the following contributions:

- **The Cellular Gate Library:** I describe the construction and characterization of a small library of *in-vivo* logic gates in an effort to assemble the biological equivalent of a TTL data book[60]. The cellular gates use four different dna-binding proteins (*lacI*, *cI*, *tetR*, *luxR*), with sixteen variations of the gate based

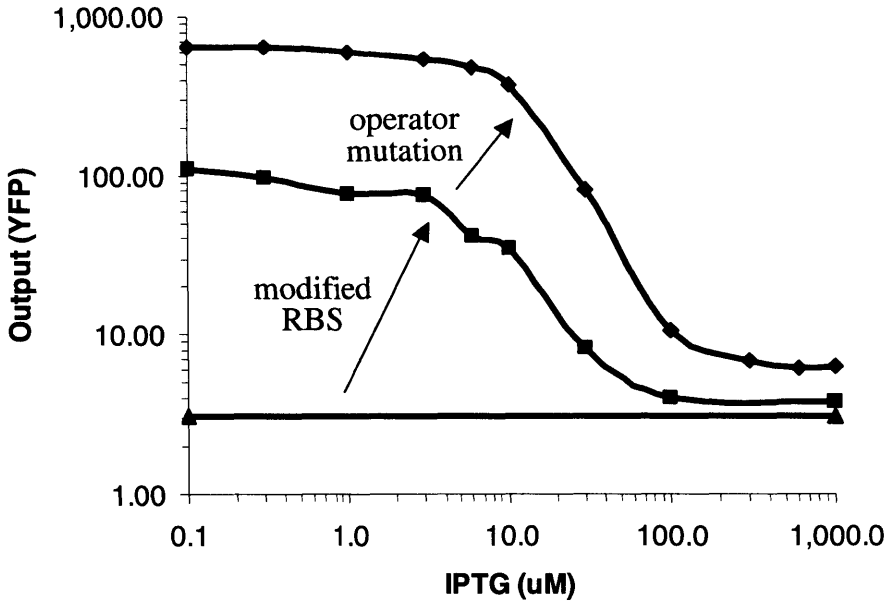


Figure 1-3: Genetic process engineering of the $cI/\lambda_{P(R-O12)}$ inverter. Experimental results with *E. coli* cells demonstrate how a series of genetic modifications converts a non-functional circuit into one that achieves the desired input/output behavior for digital computation. The evident inverse sigmoidal relationship of the improved inverter has good signal restoration and noise margins, and is useful for producing reliable digital circuits of significant complexity.

on the *cI* protein. I establish and experiment with criteria for characterizing and evaluating these gates, using transfer functions. The analysis emphasizes what can be conveniently measured *in-vivo*, and special substrate properties, such as signal fluctuations. This effort represents the first attempt to assemble a library of simple standardized biological components that can be combined in predictable ways to engineer novel cell behavior.

- **Genetic process engineering:** I demonstrate the fundamentals of *genetic process engineering* – taking existing genetic regulatory elements and modifying their DNA encoding so that they can be used in constructing complex *in vivo* digital-logic circuits. In particular, I mutated ribosome binding sites for the *cI* repressor protein and its associated operator for the Bacteriophage λ P(R) promoter so that the resulting cellular logic gates had good noise margins and signal-restoration characteristics (Figure 1-3). This work is important because

it shows how to synthesize biological components with *specific properties* useful for constructing reliable circuits of significant complexity.

- ***In-vivo* Logic Circuits:** I demonstrate several operational *in-vivo* logic circuits, with instances of three gates combined based on transfer functions. I show that the genetic circuits have favorable noise margins and significant gain. This work demonstrates the feasibility of using the digital abstraction for embedding computation and control in cells.

- **Intercellular Communications:** I describe the design, testing, and characterization of engineered intercellular communications using programmed enzymatic activity and chemical diffusions to carry messages, using DNA from the *Vibrio fischeri lux* operon. Specifically, I demonstrate the construction and testing of engineered genetic circuits that exhibit the ability to send a controlled signal from one cell, diffuse that signal through the intercellular medium, receive that signal within a second cell, and activate a remote transcriptional response. This work shows how to appropriate a natural intercellular signaling mechanism for the component library, and couple the new component to other logic elements for programmed cell behavior.

- **BioSPICE:** I introduce BioSPICE[1], a prototype tool for biocircuit design and analysis. It features analysis of the steady state, dynamics, and intercellular behavior of genetic logic circuits. BioSPICE also enables the designer to analyze and simulate modifications to the behavioral characteristics of cellular gates. It can thus provide valuable insights toward reliable system design, such as matching input/output threshold levels.

- **Microbial Colony Language:** I present the Microbial Colony Language (MCL), a convenient computing paradigm for programming cell aggregates. MCL is simple enough for implementing in cells, yet expressive enough for interesting applications. I also describe the simulator for the language and several

[1]Distinct from Arkin's Bio/SPICE[4]

simulation examples. Many aspects of the language can be directly mapped to the engineered biological processes described above. In the future, I expect that a bio-compiler will translate programs written in MCL into genetic circuits implementing the programs in cells.

1.4 Related Work

Related work includes various proposals for biochemical computation, study of the genetic elements used in my cellular gate library, models of genetic regulatory networks, and forward-engineered genetic regulatory networks.

Proposals for Biochemical Computation

At least as early as 1974, Roessler and others [66, 67, 68, 72] noted the possibility of building universal automata by coupling bistable chemical reactions, and that chemical reaction kinetics share a formal relationship with electronic circuit action. Okamoto *et al.* studied a cyclic enzyme system and showed that it had some properties of a McCulloch-Pitts neuron. In 1991, Hjelmfelt *et al.* [42] showed in principle how to construct neural networks from coupled chemical reactions, and determined specific connections for the construction of chemical logic gates. Later, Arkin and Ross [5] refined this method to allow use of enzymes with lower binding cooperativity, and applied their model to an in-depth analysis of a portion of the glycolytic cycle.

Study of Individual Genetic Elements

The genetic regulatory elements used in this thesis have been studied extensively. Review material covering research for these elements includes treatments of the lambda repressor(*cI*)[64], the lac operon(*lacI*)[56], the tetracycline repressor(*tetR*)[40, 41], and the lux operon(*luxR*)[9, 18]. I chose to incorporate these elements into my circuits partly because of the extensive literature that is available on them. However, in my thesis, I integrate these well-known genetic elements in novel ways that do not exist in the wild-type. The experimental results from the integrations show that the original

components are not likely to couple correctly and produce the desired results, which is the motivation for performing genetic process engineering.

Models of genetic regulatory networks

Monod and Jacob [55, 43], Sugita [79], Kauffman [48], and Thomas [84] have all made various and partially successful attempts at describing the global qualitative dynamics of genetic regulatory systems, by simplifying those systems to binary signal levels and pursuing a treatment in terms of Boolean networks. Neidhardt and Savageau [59] have noted the need for useful high-level logical abstractions to improve our understanding of the integrative molecular biology of the cell. Several other works have also modeled genetic regulatory networks using logical or discrete states [33, 34, 30, 31, 49, 71, 78, 85, 86]. Recently, McAdams and others [54, 52] have constructed mathematical models of various genetic regulatory networks *in vivo* that incorporate both digital and analog logic components.

Other modeling efforts are based on piece-wise linear[82], non-linear [5, 8, 11, 22, 32, 47, 92], statistical-mechanical[75, 76], rate equations with noise[12], and stochastic methods [6, 24, 25, 52, 53]. The stochastic models are computationally intensive, but offer promising detailed simulations suitable for modeling the effects of small numbers of molecules and noise.

Engineered genetic regulatory networks

Besides this thesis, three other recent projects have experimentally demonstrated forward-engineered genetic regulatory networks that perform specific tasks. Becskei's autorepressive construct[10] is a single gene that negatively regulates itself to achieve a more stable output. Gardner's toggle switch[29] is a genetic system where two proteins negatively regulate the synthesis of each other. This system is bistable, and sufficiently large perturbations can switch the state of the system. Elowitz's repressilator[23] is a genetic system where three proteins in a ring negatively repress each other. The system oscillates between LOW and HIGH values. These works analyze the requirements for correct operation for their particular genetic circuits based

on mathematical models of rate equations. My previous work[90, 89] has provided BioSPICE simulations of a ring oscillator and an RS-Latch which are genetically similar to the repressilator and toggle switch respectively. Some of the published toggle-switch attempts are monostable or appear to increase the protein concentration of their HIGH value over time, while only 40% of the cells in the repressilator experiments oscillate and the cell population exhibits significant variations in oscillation phase and amplitude. Nevertheless, these works complement my thesis and help demonstrate the feasibility of forward-engineering genetic regulatory networks to achieve specific functions.

In this thesis, I take a different approach to building genetic regulatory networks. Namely, the focus of my work here and in previous publications[90, 89, 91] is to establish a methodology for building *any* logic circuit by extensively characterizing and modifying the behavior of simple components to build a cellular gate library of well-characterized components suitable for use in reliable logic circuits. Given the cellular gate library, an engineer can predict the behavior of a complex system through the functional composition of the input/output behavior of the simple components.

1.5 Thesis Outline

In the remainder of this thesis, Chapter 2 introduces the logic gates implemented for this work. The NOT gate is the fundamental building block for constructing intracellular circuits, while the IMPLIES and AND gates are used for intercellular communications. Chapter 3 introduces the BioSPICE tool for biocircuit design and analysis. The chapter describes the model used for simulating a biochemical inverter, simulations of simple logic circuits in single cells, analysis of genetic modifications to achieve the desired gate behavior, and simulations of intercellular communications using chemical diffusions. Chapter 4 describes experimental measurements of the device physics of *in-vivo* logic gates, as well as genetic process engineering to modify gates until they have the desired behavior. Chapter 5 presents experimental results of programmed intercellular communications, including time response measurements and sensitivity

to variations in message concentrations. Finally, Chapter 6 describes the Microbial Colony Language, a useful paradigm for programming cell aggregates. The chapter describes the language syntax and semantics, as well as simulations of programs for pattern formation in cell aggregates.

Chapter 2

Cellular Gates

A fundamental chemical process in the cell is the production of proteins from genes encoded in the DNA. The cell performs important regulatory activities through DNA-binding proteins that repress or activate the production of specific proteins. Knight and Sussman [45] propose using repression to implement digital-logic inverters. This chapter presents a formal model of this inversion mechanism, including several modifications, and explains how to construct any finite digital-logic circuit using these inverters.

This chapter also introduces two additional logic gates that implement the IMPLIES and AND logic functions. The inputs to these gates are messenger RNA (mRNA) molecules that code for DNA-binding proteins, and small inducer molecules that affect the activity of these proteins. Because the inducer molecules freely diffuse through cell membranes, these two gates are useful for intercellular communications and other external interactions with the *in-vivo* circuits.

In this chapter, Section 2.1 describes the biochemical process of inversion. Section 2.2 explains how to build any finite intracellular logic circuits from inverters. Section 2.3 describes the two intercellular communications gates implementing the IMPLIES and AND logic functions.

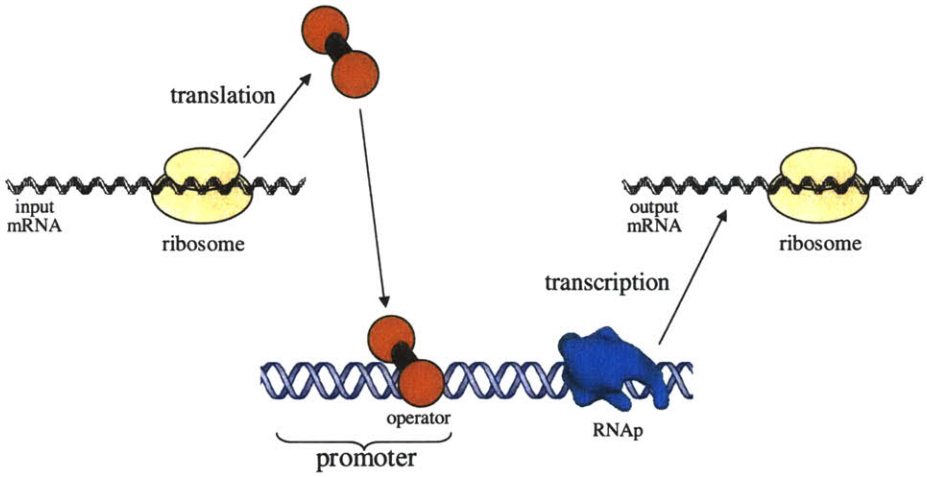


Figure 2-1: Biochemical inversion uses the transcription and translation cellular processes. Ribosomal RNA translates the input mRNA into an amino acid chain, which then folds into a three-dimensional protein structure. When the protein binds to an operator of the gene's promoter, it prevents transcription of the gene by RNA polymerase (RNAp). In the absence of the repressor protein, RNAp transcribes the gene into the output mRNA.

2.1 A Biochemical Inverter

Natural gene regulation systems exhibit characteristics useful for implementing *in vivo* logic circuits. Figure 1-2 presents a simplified view of the two states in the biochemical process of inversion in either the presence or absence of the input mRNA signal. Figure 2-1 shows a more detailed description of the inversion process, including the role of transcription and translation. In the original proposal by Knight and Sussman, dna-binding proteins serve as the signals. I chose to represent signals using mRNA rather than proteins because the mRNA representation is more convenient for measuring and modifying the device physics of cellular gates, as discussed in Section 4.1.

Figure 2-2 illustrates the functional construction of an inverter from its biochemical reaction stages. Let ψ_A denote the concentration level of the input mRNA, representing the input signal to the inverter. In the first stage, ribosomal RNA translates the mRNA product ψ_A into the input repressor protein ϕ_A. Let $\mathcal{L}$ (translation

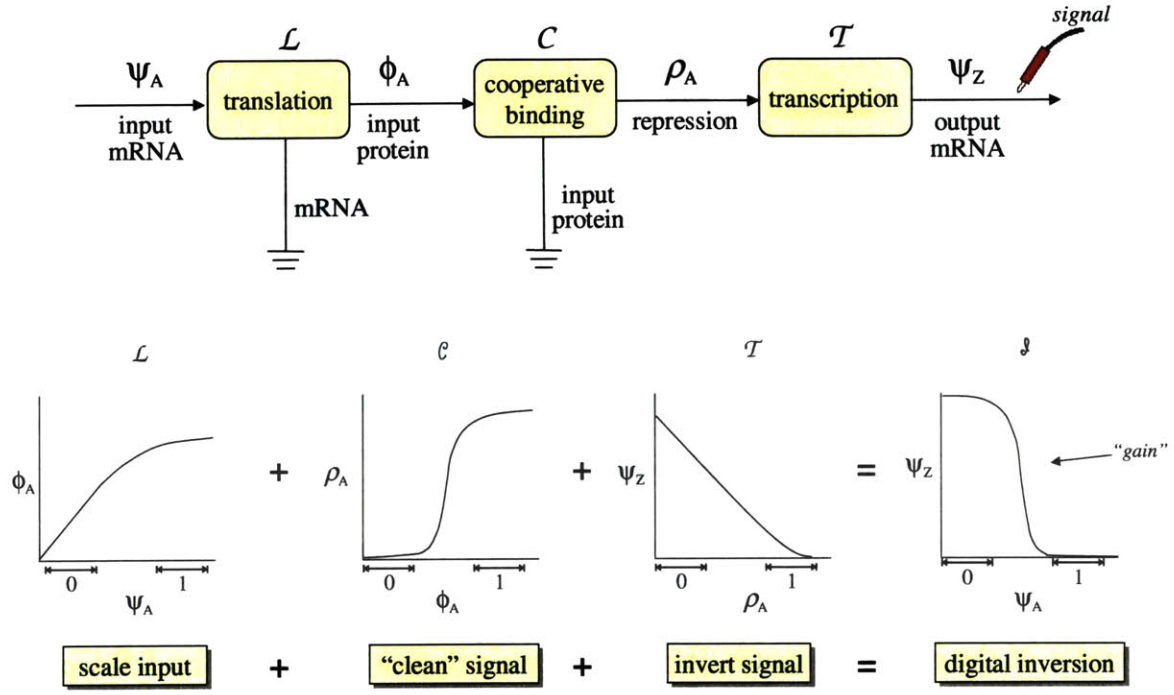


Figure 2-2: Functional composition of the inversion stages: the translation stage maps input mRNA levels (ψ_A) to input protein levels (ϕ_A), the cooperative binding stage maps input protein levels to bound operator levels (ρ_A), and the transcription stage maps bound operator levels to output mRNA levels (ψ_Z). The degradation of the mRNA and protein molecules is represented with the electrical ground symbol. The degradation of mRNA is part of the translation stage, while the degradation of proteins is part of the cooperative binding stage. The graphs illustrate the steady-state relationships for each of these stages and the overall inversion function that results from combining these stages.

stage) denote the steady state mapping between ψ_A and ϕ_A. In general, increases in ψ_A yield linear increases in ϕ_A until an asymptotic boundary is reached. Factors that determine this boundary include the amino acid synthesis capabilities of the cell, the efficiency of the ribosome binding site, and mRNA stability. Because the cell degrades both mRNA and input protein molecules, a continuous synthesis of the input mRNA is required for a steady level of the input protein.

The second stage in the inverter employs cooperative binding to reduce the digital noise. Here, the input protein monomers join to form polymers (often dimers, occasionally tetramers), which then bind to the operator and repress the gene. Because

the cells degrades the proteins, a continuous synthesis of the input protein is required for maintaining the repression activity. Let ρ_A denote the strength of this repression, defined as the concentration of operator that is bound by repressor. In steady state, the relation $\mathcal{C}$ (cooperative binding stage) between ϕ_A and ρ_A will generally be sigmoidal. For low values of ϕ_A, the amount of repression increases only slightly as the input protein concentrations increase because these concentrations are too low for significant dimerization. Without dimerization, the monomeric repressor cannot bind to the DNA. Then, at higher levels of ϕ_A (when the input proteins dimerize easily), cooperative binding and dimerization result in non-linear increases in the repression activity. Finally, at saturating levels of the input protein when the operator is mostly bound, the curve reaches an asymptotic boundary. Because the repressor activity is maximal, additional repressor molecules have no noticeable effect. The purpose for this stage is to provide signal restoration: as a result of this stage, the analog input signal better approximates its digital meaning.

In the last stage, RNA polymerase transcribes the structural gene and inverts the signal. Let ψ_Z denote the mRNA concentration for the output signal Z. Then, in the steady state relation $\mathcal{T}$ (transcription stage) between ψ_Z and ρ_A, ψ_Z decreases monotonically as ρ_A increases. With no repression, transcription proceeds at maximum pace (i.e. maximum level of ψ_Z). Any increase in repression activity results in a decrease in transcription activity, and hence the inversion of the signal.

As illustrated in Figure 2-2, the functional combination $\mathcal{I}$ of the above stages achieves digital inversion:

$$\psi_Z = \mathcal{I}(\psi_A) = \mathcal{T} \circ \mathcal{C} \circ \mathcal{L}(\psi_A)$$

$\mathcal{I}$ is the transfer function of the inverter.

2.2 Intracellular Logic Circuits

Biochemical inverters are the building blocks for constructing intracellular logic circuits. First, most protein coding sequences can be successfully fused to any given

promoter. Second, two inverters form a NAND gate by "wiring-OR" their outputs. These two features combine to provide a modular approach to logic circuit design of any finite complexity, as described below.

2.2.1 Modularity in Circuit Construction

The modularity in biocircuit design stems from the ability to designate almost any gene as the output of any logic gate. Consider a logic element A consisting of an input mRNA, M_A, that is translated into an input protein repressor R_A, acting on an operator, O_A, associated with a promoter P_A. Let P_A be fused to a structural gene G_Z coding for the output mRNA M_Z. Figure 2-1 illustrates these genetic elements. The DNA base pair sequence G_Z (or the corresponding output mRNA sequence M_Z) that codes for an output protein R_Z determines the gate connectivity because the output protein may bind to other operators in the system. The specific binding of R_Z to another downstream operator O_Z connects gates together because the level of R_Z affects the operation of the downstream gate.

To a first approximation, the choice of the sequence G_Z does not affect the transfer function $\mathcal{I}$ of the inverter, $M_Z = \mathcal{T} \circ \mathcal{C} \circ \mathcal{L}(M_A)$. An exception to this rule occurs when G_Z codes for a protein that interacts with operator O_A or with input protein repressor R_A. Thus, the designer of *in-vivo* logic circuits must ensure that the signal proteins do not interact with other circuit elements besides their corresponding operators. The circuit designer should experimentally verify this required protein non-interference prior to circuit design.[1] Any set of non-interacting proteins can then serve as a library of potential signals for constructing an integrated circuit.

Once protein non-interference is established, modularity of the network design affords a free choice of signals. Any suitable repressor protein and its corresponding mRNA is a potential candidate for any signal, where the issue of "suitability" is

[1]For example, the following simple *in-vivo* experiment checks whether a protein affects a particular promoter. First, fuse a fluorescent protein to a promoter of interest and quantify the *in-vivo* fluorescence intensity. Next, add a genetic construct that overexpresses the protein of interest. Finally, check the new fluorescence intensity to determine whether the protein affects transcription.

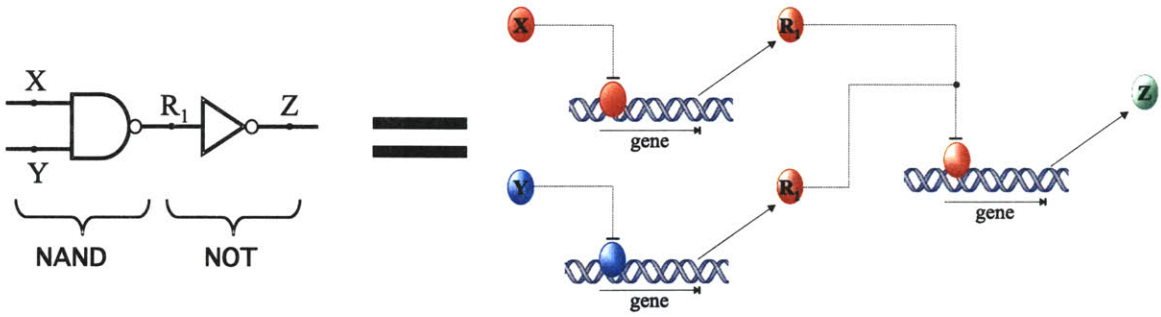


Figure 2-3: A logic circuit and its DNA implementation: The "wire-OR" of the outputs of two genes implements a NAND gate, and the choice of the output gene determines the gate connectivity.

discussed in Section 3.3.3. This modularity is necessary for implementing a "bio-compiler": a program that consults a library of repressor proteins and their associated operators and generates genetic logic circuits directly from gate-level descriptions. Contrast this modularity with the method of Hjelmfelt *et al.*[42], that requires proteins that modify other proteins, and where all signals are protein concentrations. In that case, the resulting physico-chemical interdependence of successive logic stages makes simple modularity almost impossible.

2.2.2 Implementation of Combinatorial Logic

The approach to combinatorial logic is to "wire-OR" the outputs of multiple inverters by assigning them the same output gene. The output mRNA is expressed in the absence of either input mRNA's, and is not be expressed only when both inputs are present. This configuration implements a NAND gate. Since the performance of a NAND gate relies solely on that of its constituent inverters, well-engineered inverters will yield well-engineered combinatorial gates.

Figure 2-3 illustrates a small circuit where a NAND gate connects to an inverter. Here, mRNA and their corresponding proteins serve as the logic circuit wires, while the promoter and protein/mRNA decay implement the gates. Since a NAND gate is a universal logic element and can be wired to other gates, any finite digital circuit can be built within practical limitations such as the number of distinct signal proteins

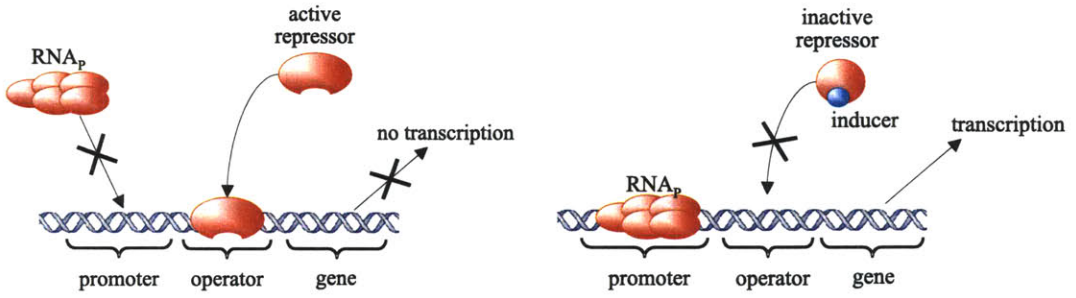
available.

2.2.3 Choice of Signals

The library of naturally available signal proteins includes approximately a few thousand candidates[51]. Any repressor protein with sufficiently cooperative DNA binding and that does not interfere with normal cell operation is a potential candidate. The experiments described in Chapter 4 use four different naturally occurring DNA binding proteins and their operators, as well as several novel mutations to one of the operators. The number of naturally occurring proteins that could potentially serve as signals may never limit biocircuit construction because other factors, such as the metabolic capabilities of cells, are likely to place lower limits on biocircuit complexity within a single cell. However, it may still be more efficient to synthesize artificial dna-binding proteins for use as signals rather than finding natural sources. In the future, combinatorial chemistry techniques, along with a method such as phage display, will yield large libraries of novel DNA binding proteins and corresponding operators. One potential source of a very large set of non-interacting signals is engineered Zinc Finger DNA binding proteins[37].

2.3 Intercellular Gates

While the biochemical inversion mechanism suffices for building intracellular circuits, external interaction with the cells requires additional logic gates. Small molecules known as inducers freely diffuse through cellular membranes and interact with DNA binding proteins. This section describes how the inducer-protein interactions implement two different intercellular gates. Chapters 4 and 5 report on experimental results where the IMPLIES gate enables human-to-cell communications while the AND gate facilitates cell-to-cell communications.



Repressor	Inducer	Output
0	0	1
0	1	1
1	0	0
1	1	1

Figure 2-4: A genetic gate for the IMPLIES logic function using repressors and inducers. Shown here are the two states when the repressor protein is present, the logic symbol for the gate, and the logic truth table.

2.3.1 The IMPLIES Gate

The IMPLIES gate allows cells to receive control messages sent by humans or detect certain environmental conditions. Figure 2-4 illustrates the biochemical reactions, the logic symbol, and the logic truth table for an intercellular gate that implements the IMPLIES logic function. In the absence of the input mRNA and its corresponding repressor, RNAp binds to the promoter and transcribes the output gene, yielding a high output. As with the inverter, if only the input repressor is present, it binds to the promoter and prevents transcription, yielding a low output. Finally, if both the repressor and the inducer are present, the inducer binds to the repressor and changes the conformation of the repressor. The conformation change prevents the repressor from binding to the operator, and allows RNAp to transcribe the gene, yielding a high output.

The IMPLIES gate has the same three biochemical stages as the inverter: translation, cooperative binding, and transcription. The inducer concentration levels affect the cooperative binding stage C', which now has two inputs. Let v_A denote the inducer concentration level, let ϕ_A denote the concentration level of the input mRNA,

and let ϕ_Z denote the concentration level of the output mRNA. Here, the repression activity ρ_A resulting from the cooperative binding stage, $\rho_A = \mathcal{C}'(\phi_A, \upsilon_A)$, depends non-linearly on the level of the repressor and on the level of the inducer. The binding affinities of the active repressor to the operator, and of the inducer molecule to the repressor, determine the shape of $\mathcal{C}'$. The transfer function $\mathcal{I}$ of the IMPLIES logic gate is the mapping:

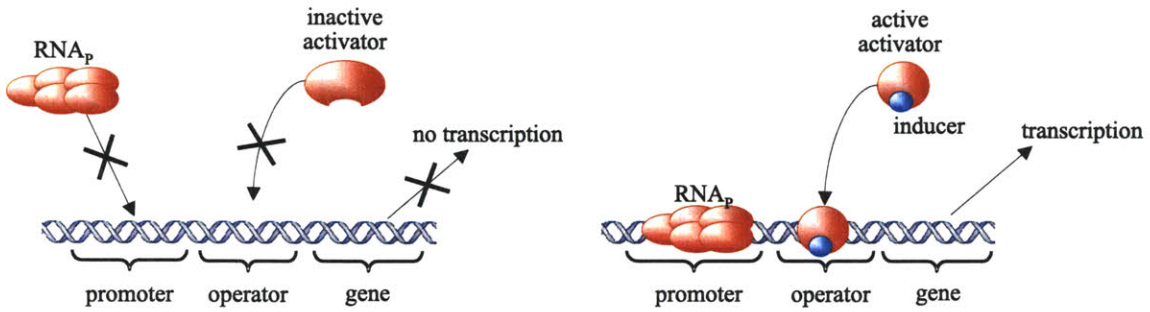
$$\psi_Z = \mathcal{I}(\psi_A) = \mathcal{T} \circ \mathcal{C}'(\upsilon_A, \mathcal{L}(\psi_A))$$

The two gate inputs are not interchangeable. The input and output repressors can be connected to any other circuit component, but the inducer input is an intercellular signal, and is specifically coupled to the input repressor. As with the different inverter signals, before building a circuit the designer should experimentally check for unintended interactions between a specific inducer, other repressors, and other inducers. Any set of non-interfering repressor/inducer pairs can then serve as a library of potential signals for constructing intercellular circuits.

Chapter 4 describes experimental results demonstrating quantitative and predictable external interactions with cells by introducing known quantities of synthetic inducer molecules to the growth medium.

2.3.2 The AND Gate

The AND gate allows cells to detect incoming messages from other cells. Figure 2-5 illustrates the biochemical reactions, the logic symbol, and the truth table for an intercellular gate that implements the AND logic function. Here, RNAp normally has a low affinity for the promoter and basal transcription is correspondingly small. Therefore, in the absence of the activator and inducer inputs, the output is low. If the activator is present but the inducer is not present, the activator has a low affinity for the promoter and does not bind to it. In this case, the output is still low. Finally, if the inducer and the activator are both present, the inducer binds to the activator. The newly formed bond changes the conformation of the activator and allows the



Activator	Inducer	Output
0	0	0
0	1	0
1	0	0
1	1	1

Figure 2-5: A genetic gate for the AND logic function using activators and inducers: two of the states when the activator is present, the logic symbol for the gate, and the logic truth table.

activator/inducer complex to bind to the operator. In turn, the activator/inducer complex helps recruit RNAp to the promoter and initiate transcription, yielding a high output. Often, a dimeric form of the activator is necessary for complex formation.

Similar to the biochemical stages of the NOT and IMPLIES gates, the AND gate stages include translation, cooperative binding, and transcription. The first stage, translation, is similar in all three cases. For the AND gate, the following cooperative binding stage $\mathcal{C}''$ maps the activator protein ϕ_A and inducer υ_A inputs to an *activation* level, ω_A, rather than repression. Similar to the two-input cooperative binding stage for the IMPLIES gate, the activation level $\omega_A = \mathcal{C}''(\phi_A, \upsilon_A)$ depends non-linearly on the concentration of activator and inducer. The binding affinities of the inducer molecule to the activator and of the activator/inducer complex to the operator determine the shape of $\mathcal{C}''$. Lastly, the transcription stage maps the activation level, ω_A, to output mRNA, ψ_Z, in a positive relation. The transfer function $\mathcal{I}$ of the AND logic gate is the mapping:

$$\psi_Z = \mathcal{I}(\psi_A) = \mathcal{T} \circ \mathcal{C}''(\upsilon_A, \mathcal{L}(\psi_A))$$

As with the IMPLIES gate, the two inputs are not interchangeable, and the inducer must always be coupled with the corresponding activator. Again, the interference between inducers and other circuit elements should be checked experimentally prior to circuit design. Beyond the challenge of finding non-interfering activator/inducer pairs, any multicellular system design must also address the spatial issues of intercellular communications.

2.3.3 The SEND Gate

To initiate communications to receiver cells that have the AND gate, sender cells synthesize inducer molecules using a SEND gate. The input to the SEND gate is mRNA coding for a special catalytic enzyme, with a concentration of ϕ_A. The enzymatic reaction of the gate, $\mathcal{E}$, produces inducer with an intracellular concentration υ_A, where the inducer also diffuses into the medium. The steady state mapping, $\upsilon_A = \mathcal{E}(\phi_A)$, is a positive relation with an asymptotic boundary defined by the availability of the appropriate metabolic precursors in the cell. The intracellular level of the inducer, υ_A, is also affected by the surrounding extra-cellular concentration of the inducer.

Chapter 5 describes experimental results that demonstrate engineered cell-to-cell communications using the SEND and AND gates. First, sender cells produce the catalytic enzyme *luxI*, an input to the SEND gate. *luxI* molecules then catalyze the formation of VAI (3-N-oxohexanoyl-L-Homoserine-lactone), the output of the SEND gate, using metabolic precursors found in the cell. VAI diffuses into the medium and enters neighboring receiver cells that have the AND gate. The receiver cells constitutively express *luxR*, the activator input to the AND gate. When VAI enters the receiver's cytoplasm, VAI induces *luxR* to form a dimeric or multimeric complex, which then binds to the lux P(R) promoter and activates transcription of the green fluorescent protein (GFP). Using various fluorescence detection tools, a researcher can then observe and quantify when cells receive messages from their neighbors.

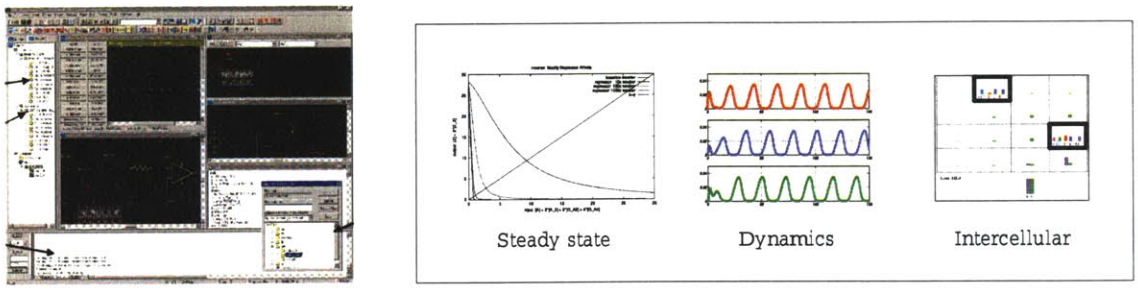
The next chapter describes biocircuit design and analysis with BioSPICE, a tool for simulating and analyzing *in-vivo* digital-logic circuits and intercellular communications.

Chapter 3

Biocircuit Design and Analysis

Biocircuit design poses several important challanges, some that are in common with electrical circuit design, and some that are unique. As with electrical circuits, the behavior of biocircuits depends on the characteristics of the component gates. It is therefore important to first characterize the behavior of the gates, by measuring their device physics, before attempting to design circuits of significant complexity. Based on the characteristics of individual gates, one can predict the behavior of complex circuits built from these components. In this chapter, I use BioSPICE to develop and simulate a model of biochemical inversion, and then demonstrate how to predict the behavior of biocircuits based on this model. I also show that the correct behavior of these circuits is highly dependent on using gates with matching characteristics.

Initial attempts at constructing gates often yield devices with unacceptable behavior. For example, a particular device may have insufficient signal restoration capabilities or inadequate noise margins. In designing electrical circuits, one uses process engineering to modify the characteristics of the devices (for example, gain or trigger levels) until they obtain the desired behavior. In this chapter, I introduce *genetic process engineering*, the analogous mechanism for biocircuit design. I demonstrate how BioSPICE facilitates genetic process engineering by predicting the new behavior of devices that results from genetic modification to specific stages in biochemical inversion. The analysis and insights gained from these BioSPICE predictions guide the engineer in genetically modifying existing components to achieve the appropriate



Spice **BioSPICE**

Figure 3-1: Tools for circuit design: Spice and BioSPICE.

behavior for building reliable circuits of significant complexity.

In contrast to electrical circuit design where identical components are separated spatially, each component in a biocircuit shares the same physical space but relies on different biochemical reactions. The complexity of biocircuit design is exacerbated by the fact that the components typically have widely varying kinetic characteristics. These gates are built from dna-binding proteins, ribosome binding sites, and protein binding sites with inherently different kinetic characteristics. Therefore, a critical element in biocircuit design is analyzing and optimizing the behavior of each new gate included in the cellular gate library. In this chapter, I describe the conditions necessary for matching gates to achieve correct digital behavior. This analysis motivates specific genetic modifications to achieve gate device physics that match with other gates, for correct design and construction of complex circuitry.

In discussing biocircuit design, this chapter often highlights the role of BioSPICE. With electrical circuits, tools such as SPICE[57] help designers manage the complexity of their substrate and achieve reliable systems. The BioSPICE prototype tool aims to help biocircuit designers in a similar manner. It is an integrated collection of software tools that features analysis of the steady state, dynamics, and intercellular behavior of genetic logic circuits (Figure 3-1). BioSPICE enables the designer to analyze and simulate modifications to the behavioral characteristics of cellular gates. It can thus provide valuable insights toward reliable system design, such as matching gate input/output threshold levels.

In the rest of this chapter, Section 3.1 presents a biochemical model of a reaction system implementing an inverter. Section 3.2 describes simulation results of two simple but interesting logic circuits built from inverters. Section 3.3 analyzes the steady state behavior of the inverter model, the requirements for gate matching, the effects of genetic modifications on the transfer curves of inverters, and other issues in biocircuit design. Finally, Section 3.4 describes BioSPICE simulations of intercellular communications using genetic logic circuits.

3.1 A Biochemical Model of an Inverter

To implement digital-logic computation, an inverter combines several natural gene regulatory mechanisms. These mechanisms include transcriptional control, translation of mRNA, repression through cooperative binding, and degradation of proteins and mRNA transcripts.

Table 3.1 presents one possible chemical model of the reactions involved in biochemical inversion. In particular, this model incorporates characteristics from the Bacteriophage λ *cI* repressor operating on the P(R) promoter and the O_R1 and O_R2 operators. The $mRNA_A$ molecule represents the input signal, and the $mRNA_Z$ molecule represents the output signal. Ribosomal RNA (rRNA) translates $mRNA_A$ into the input protein repressor A, and A_2 denotes the dimeric form of A. P_Z denotes the concentration of the *active* form of the promoter for Z. A promoter is active only when its associated operator is *unbound* by a repressor. $P_Z A_2$ and $P_Z A_4$ represent the repressed (i.e. inactive) forms of the promoter, where either one dimer or two dimers are bound to the promoter respectively. RNA polymerase (RNAp)[1] initiates transcription from the active form of the promoter, P_Z, into $mRNA_Z$, the gene transcript. This gene transcript typically codes for other signals (e.g. protein repressors or activators), or for structural and enzymatic proteins that perform certain cellular

[1]The simulations in this section assume that the concentrations of RNA_p and $rRNA$ are fixed. Chapter 4 discusses how to measure the effect of fluctuations in these concentrations, as well as other factors, on the inverter's behavior. Once these effects have been quantified, robust gates can be designed.

$$mRNA_A + rRNA \xrightarrow[\text{translate}]{k_{xlate}} mRNA_A + rRNA + A \quad (3.1)$$

$$mRNA_A \xrightarrow[\text{decay}]{k_{dec(mrna)}} \quad (3.2)$$

$$A + A \xrightarrow[\text{dimerization}]{k_{dim(a)}} A_2 \quad (3.3)$$

$$A_2 \xrightarrow[\text{single}]{k_{sngl(a)}} A + A \quad (3.4)$$

$$A \xrightarrow[\text{decay}]{k_{dec(a)}} \quad (3.5)$$

$$A_2 \xrightarrow[\text{decay}]{k_{dec(a2)}} \quad (3.6)$$

$$P_Z + A_2 \xrightarrow[\text{repress 1}]{k_{rprs(a2)}} P_Z A_2 \quad (3.7)$$

$$P_Z A_2 \xrightarrow[\text{dissociation}]{k_{dis(a2)}} P_Z + A_2 \quad (3.8)$$

$$P_Z A_2 \xrightarrow[\text{decay}]{k_{dec(ga2)}} P_Z \quad (3.9)$$

$$P_Z A_4 \xrightarrow[\text{decay}]{k_{dec(ga4)}} P_Z A_2 \quad (3.10)$$

$$P_Z A_2 + A_2 \xrightarrow[\text{repress 2}]{k_{rprs(a4)}} P_Z A_4 \quad (3.11)$$

$$P_Z A_4 \xrightarrow[\text{dissociation}]{k_{dis(a4)}} P_Z A_2 + A_2 \quad (3.12)$$

$$P_Z + RNA_p \xrightarrow[\text{transcribe}]{k_{xscribe}} P_Z + RNA_p + mRNA_Z \quad (3.13)$$

$$mRNA_Z \xrightarrow[\text{decay}]{k_{dec(mrna)}} \quad (3.14)$$

Table 3.1: Biochemical reactions that model an inverter. $mRNA_A$ is the input and $mRNA_Z$ the output.

tasks.

The model includes the components described in Section 2.1 in the following reactions: translation of the input protein from the input mRNA (reactions 3.1–3.2), input protein dimerization and decay (reactions 3.3–3.6), cooperative binding of the input protein (reactions 3.7–3.12), transcription (reaction 3.13), and degradation of the output mRNA (reaction 3.14).

To simulate an inverter, BioSPICE automatically converts the chemical equations of this model to the system of ordinary differential equations in Table 3.2. The system of differential equations includes an entry for each of the molecular species

$$d(mRNA_A) = \text{drive}_{mRNA_A}(t) - k_{dec(mrna)} \cdot mRNA_A \quad (3.15)$$

$$d(A) = 2 \cdot k_{sngl(a)} \cdot A_2 - k_{dec(a)} \cdot A + k_{xlate} \cdot rRNA \cdot mRNA_A - 2 \cdot k_{dim(a)} \cdot A^2 \quad (3.16)$$

$$d(A_2) = k_{dim(a)} \cdot A^2 - k_{sngl(a)} \cdot A_2 - k_{dec(a2)} \cdot A_2 - k_{rprs(a2)} \cdot P_Z \cdot A_2 + k_{dis(a2)} \cdot P_Z A_2 - k_{rprs(a4)} \cdot P_Z A_2 \cdot A_2 + k_{dis(a4)} \cdot P_Z A_4 \quad (3.17)$$

$$d(P_Z) = k_{dis(a2)} \cdot P_Z A_2 - k_{rprs(a2)} \cdot P_Z \cdot A_2 + k_{dec(ga2)} \cdot P_Z A_2 \quad (3.18)$$

$$d(P_Z A_2) = k_{rprs(a2)} \cdot P_Z \cdot A_2 - k_{dis(a2)} \cdot P_Z A_2 - k_{rprs(a4)} \cdot P_Z A_2 \cdot A_2 + k_{dec(ga4)} \cdot P_Z A_4 - k_{dec(ga2)} \cdot P_Z A_2 + k_{dis(a4)} \cdot P_Z A_4 \quad (3.19)$$

$$d(P_Z A_4) = k_{rprs(a4)} \cdot P_Z A_2 \cdot A_2 - k_{dis(a4)} \cdot P_Z A_4 - k_{dec(ga4)} \cdot P_Z A_4 \quad (3.20)$$

$$d(mRNA_Z) = k_{xscribe} \cdot P_Z \cdot RNA_p - k_{dec(mrna)} \cdot mRNA_Z \quad (3.21)$$

Table 3.2: Ordinary differential equations used to simulate a single inverter. These equations are derived from the biochemical inversion model in Table 3.1.

involved in inversion: the input mRNA $mRNA_A$, the input protein monomer A, the dimeric form of the input repressor protein A_2, the unbound promoter P_Z, the promoter bound with one repressor protein dimer $P_Z A_2$, the promoter bound with two repressor protein dimers $P_Z A_4$, and the output mRNA $mRNA_Z$. Each differential equation describes the time-domain behavior of a particular molecular species based on all the equations in the biochemical model that include that particular molecule. For example, the "$k_{dim(a)} \cdot A^2$" term of differential equation 3.17 is derived from model equation 3.3, while the "$-k_{sngl(a)} \cdot A_2$" term is derived from model equation 3.4. Note that the "$\text{drive}_{mRNA_A}(t)$" term in differential equation 3.15 allows the user to specify an external drive to vary the input levels of the inverter over time.

To simulate a circuit with BioSPICE, the user first defines the configuration of the circuit. For each inverter in the circuit, the user specifies the input protein, the output protein, and the promoter/operator region. BioSPICE then consults a biochemical kinetics database and converts the circuit to a Mathworks MATLAB M-File[38] that contains differential equations such as the ones in Table 3.2. These equations include the kinetic rates that are associated with the specific choices for the genetic elements. For example, the following circuit definition comprises an inverter with an external drive for the input:

```
(let ((circuit '((invert A Pr_Op_A 1 Z)
                 (drive mRNA_A "drive_mRNA_A"))))
  (circuit->matlab-derivs circuit "protein-db.scm" "mrna_inverter_circuit")))
```

The inverter has an input protein A, a promoter/operator region Pr_Op_A, a gene cistron count of 1, and an output protein Z. The file `drive_mRNA_A` defines the time-domain behavior of an external drive that adds input $mRNA\_A$ to the system. BioSPICE converts this circuit based on the kinetic rates in the database `protein-db.scm` to a system of differential equations, and stores the result in the MATLAB file `mrna_inverter_circuit.m`. The user then defines the simulation parameters, such as the duration of the simulation. Finally, the user can observe the dynamic behavior of the circuit by directing BioSPICE to solve the system of differential equations using MATLAB's stiff differential equations solver `ode15s`[73]. This solver is a variable order solver based on the numerical differentiation formulas that optionally uses backward differentiation formulas (Gear's method).

Figure 3-2 shows a BioSPICE simulation of the dynamic behavior of the inverter circuit with the above chemical reactions in response to an external stimulus. The kinetic constants (Table 3.3) used in this simulation were obtained from the literature describing the phage λ promoter P_R and repressor (cI) mechanism [39, 64]. The graphs show the concentrations of the molecules involved in the inversion, with blue curves representing mRNA, red curves representing proteins, and green curves representing genes. The top graph is for the input mRNA_A, followed by graphs for

$k_{dim(a)}$	8.333	$k_{rprs(a2)}$	66.67	$k_{dec(a)}$	.5775	$k_{dec(ga2)}$	.2887
$k_{sngl(a)}$	.1667	$k_{dis(a2)}$	.2	$k_{dec(a2)}$	.5775	$k_{dec(ga4)}$	.2887
$k_{dim(z)}$	8.333	$k_{rprs(a4)}$	333.3	$k_{dec(z)}$	.5775	$k_{dec(mrna)}$	2.0
$k_{sngl(z)}$	.1667	$k_{dis(a4)}$	.25	$k_{dec(z2)}$	.5775	$k_{xscribe}$	.0001
						k_{xlate}	.03

Table 3.3: Kinetic constants used in BioSPICE simulations. The units for the first order reactions are 100 sec^{-1} and the units for the second order reactions are $\mu M^{-1} \cdot$ 100 sec^{-1}.

the input protein repressor and its dimeric form, followed by graphs for the active and inactive forms of the gene, and finally the graph for the output mRNA_Z.

The reactions proceed as follows: At first, no input mRNA or input protein repressor are present. As a result, P_Z is active and RNAp transcribes the gene into the output mRNA_Z. The level of mRNA_Z increases until it stabilizes when the gene expression and decay reactions reach a balance. At this stage, the input signal is LOW while the output signal is HIGH.

Then, an externally-imposed drive increases the input mRNA_A, which rRNA translates into the input repressor protein A. The protein begins to form dimers, and these dimers bind to the promoter's free operators. The system quickly reaches a state where each promoter is essentially completely bound by two dimers. The almost total inactivation of the promoters occurs at a fairly low concentration of the dimer A_2, and indicates the strong repression efficiency of the *cI* repressor that is used for this simulation. As a result of the promoter inactivation, transcription stops and the output mRNA_Z decays to zero. At the end of this stage, the input signal is HIGH while the output signal is LOW.

Finally, the external drive of mRNA_A stops, resulting in the decay of mRNA_A, A, and A_2. Slowly, the repressor dimers dissociate from the P_Z operators, and the level of the active promoter P_Z rises back to the original level. This allows RNAp to resume transcription of P_Z, and the level of the output mRNA_Z rises again. At this stage, the input signal reverts to LOW, while the output signal reverts to HIGH.

The simulation shows that the gate switching time (measured in minutes for this

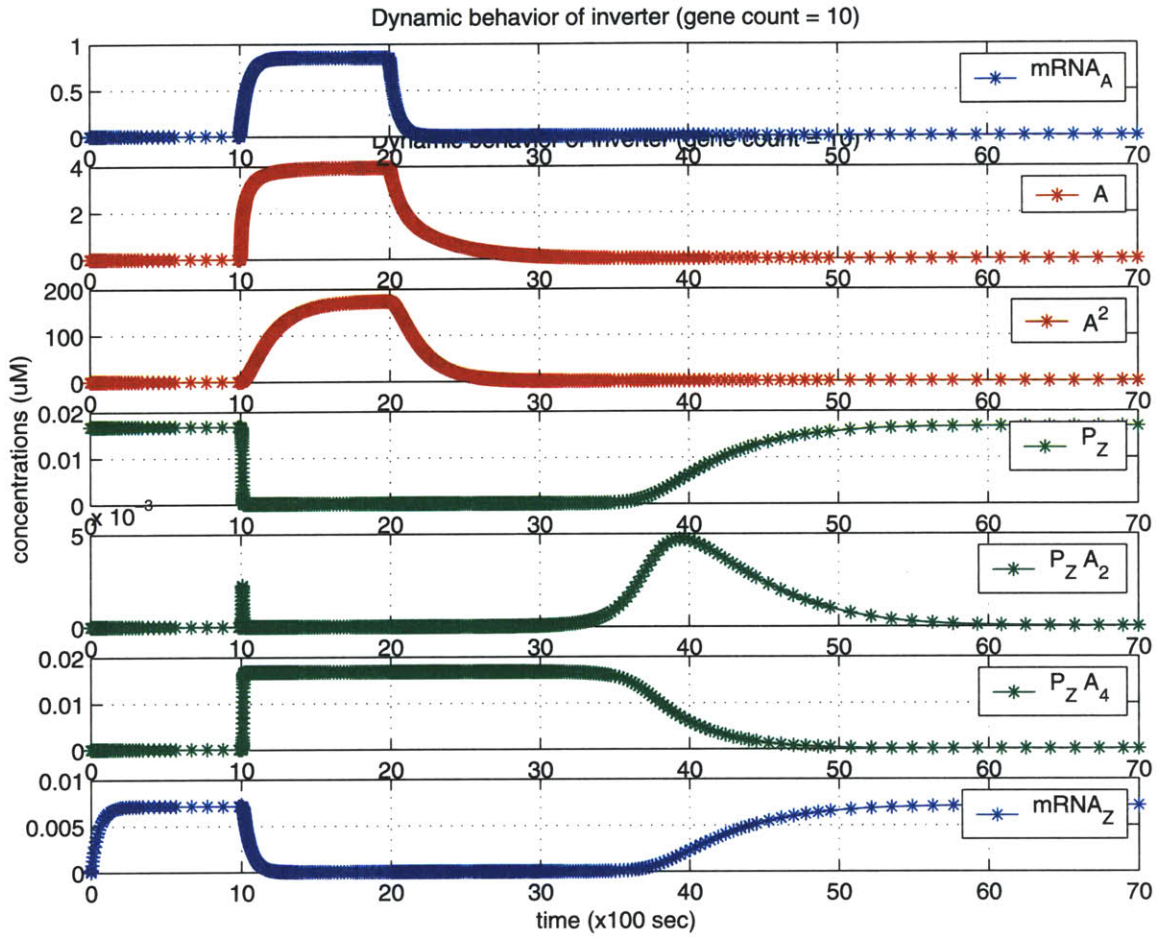


Figure 3-2: The dynamic behavior of the inverter. The graphs show a time-series of the molecular concentrations involved in inversion, in response to a stimulus of input mRNA.

mechanism) is governed by the rate of recovery of P_Z. The limiting rate is therefore the dissociation of bound repressor dimer A_2 from P_Z. Figure 3-3 shows simulation results of how an engineered reduction in the repressor binding coefficient improves the gate delay. Through simulations such as this, BioSPICE offers insights into biocircuit design and ultimately motivates laboratory experiments. For example, Section 4.4.2 describes experimental results of reducing the binding efficiency of cI to the λ promoter by mutating base pairs in the O_R1 region.

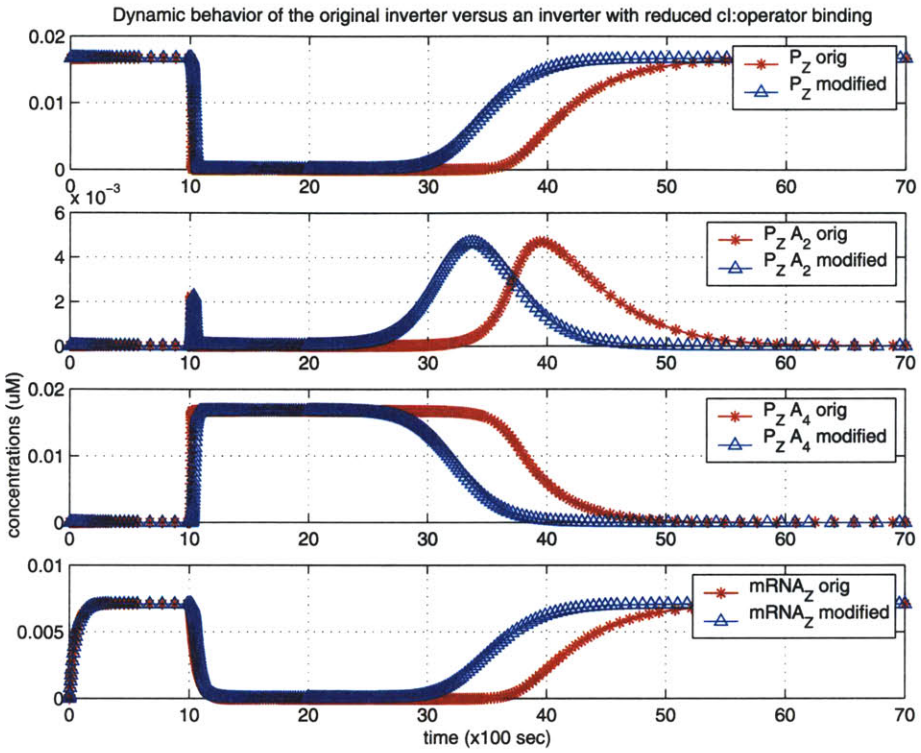
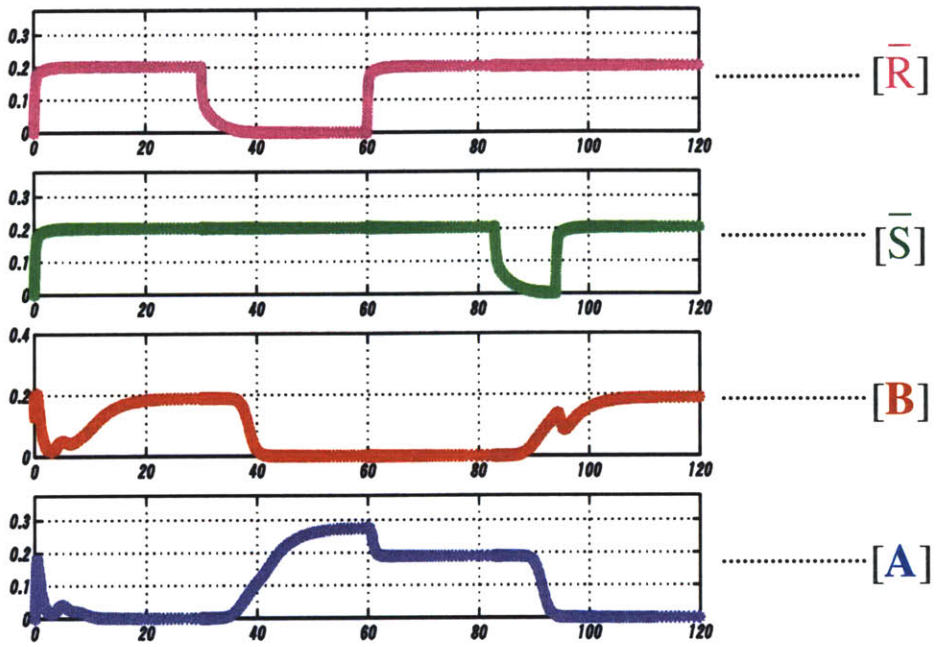


Figure 3-3: Improving gate delay: the effect of reducing the repressor binding efficiency by a factor of one hundred. For the promoter and output mRNA, the graphs compare the molecular concentrations of the original gate and of the modified gate. Overall, the modification reduces the gate delay in switching from LOW to HIGH output.

3.2 Simulations of "Proof of Concept" Circuits

This section describes BioSPICE simulation results of an RS Latch and a ring oscillator, two simple but interesting logic circuits built from the inverter model discussed above. The simulations provide a first indication of the feasibility of using the proposed chemical reactions to implement logic circuits. Furthermore, the analysis and simulations of modifications to the kinetic coefficients provide valuable insights toward reliable system design.



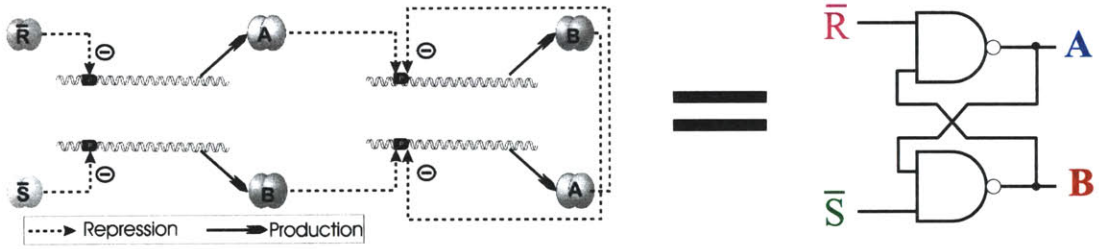


Figure 3-4: Dynamic behavior and circuit diagrams of the RS latch. The inputs $\overline{S}$ and $\overline{R}$ are normally HIGH, and are set to LOW to toggle the state of the outputs A and B. The simulations shows that the gate operates correctly in response to relatively long and short input pulses.

3.2.1 Storage: Analysis of an RS Latch

The RS Latch is a good initial test circuit for the biochemical inversion model and should operate correctly even if its constituent parts are not perfectly matched. It persistently maintains a data bit which can be toggled ON and OFF. The RS Latch consists of two cross coupled NAND gates, with inputs $\overline{S}$ and $\overline{R}$ for setting and resetting the complementary output values A and B (Figure 3-4). The inverters with

inputs $\overline{R}$ and B and common output A constitute one of the NAND gates, while the inverters with inputs $\overline{S}$ and A and common output B constitute the other NAND gate. The inputs $\overline{S}$ and $\overline{R}$ are normally HIGH, and are set to LOW to toggle the latch.

The following is the BioSPICE definition used to simulate this circuit, where each of the inverters has the kinetic characteristics from Table 3.3:

```
(let ((circuit '((invert A Pr_Op_A 1 B)
                 (invert B Pr_Op_B 1 A)
                 (invert R Pr_Op_R 1 A)
                 (invert S Pr_Op_S 1 B)
                 (drive mRNA_R "driveRi")
                 (drive mRNA_S "driveSi"))))
  (circuit->matlab-derivs circuit "protein-db.scm" "rs_latch_circuit"))
```

Figure 3-4 shows the correct simulated dynamic behavior of this RS latch in response to relatively short and long input pulses. Initially, both inputs $\overline{S}$ and $\overline{R}$ are high, and the outputs A and B compete for dominance. In this simulation, B becomes HIGH while A becomes LOW. In a physical implementation of this circuit, factors such as the relative repression efficiency, original concentration level, and stochastic noise determine which signal initially becomes HIGH.

After the initial output values settle into a steady state, an external stimulus reduces the level of the input $\overline{R}$ in order to toggle the value stored by the latch. This relatively long pulse results in expression of the output A and a subsequent decay of the output B. When $\overline{R}$ regains its original HIGH level, the circuit still maintains a HIGH level for A and a LOW level for B. Notice that the expression of A from two genes during the toggle phase results in a level of A that is higher than than the level of A during the steady state following the toggle. However, the circuit functions correctly because the higher analog value of A does not exceed the range defined to be a digital "1". Because A is a repressor, once A reaches a saturating repression level, any additional increases in concentration do not affect the output of the gate.

Finally, a short external stimulus reduces the level of $\overline{S}$ in order to toggle the RS latch back to the original state. In this case, $\overline{S}$ regains its HIGH level before B builds

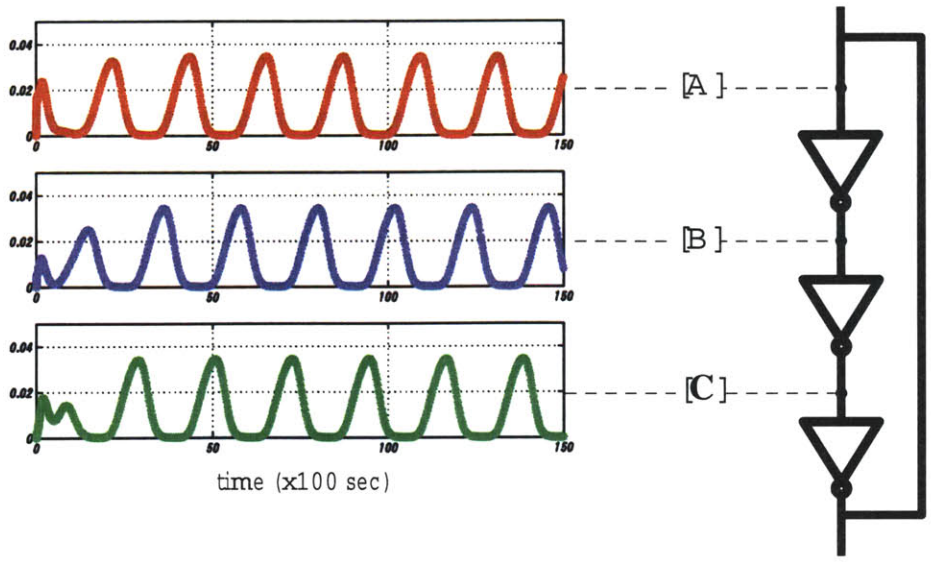


Figure 3-5: Dynamic behavior of a ring oscillator. The three curves are the outputs of the three inverters. Note the 120° phase shift between successive stages.

up to its own steady state HIGH level. The level of B drops for a short period, but then B rises back up to the appropriate HIGH level. Therefore, as expected, both long and short pulses effectively set and reset the latch.

3.2.2 Connections: Analysis of a Ring Oscillator

A ring oscillator is another useful test circuit, especially because the correct behavior of the circuit is highly sensitive to the "device physics" of its components. The oscillator consists of three inverters connected in a series loop without any external drive:

```
(let ((circuit '((invert A Pr_Op_A 1 B)
                 (invert B Pr_Op_B 1 C)
                 (invert C Pr_Op_C 1 A))))
   (circuit->matlab-derivs circuit "protein-db.scm" "mrna_ring_osc")))
```

The simulation results in Figure 3-5 depict the expected oscillation in signal concentrations, as well as a phase shift between the values (values shown are protein concentrations, not mRNA). Note, however, that oscillation occurs close to the LOW

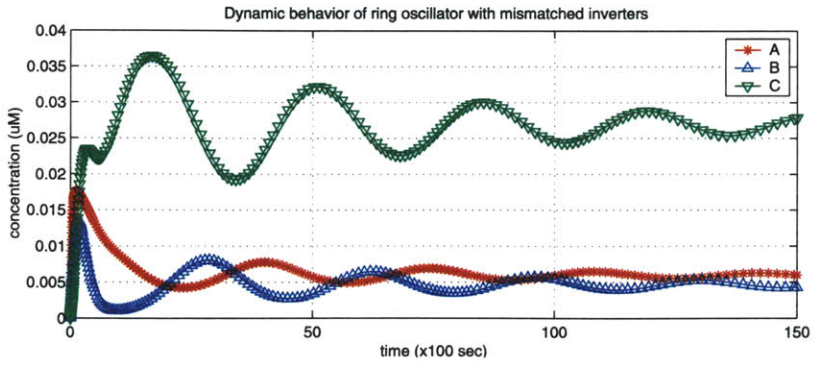


Figure 3-6: A time-series simulation illustrating the incorrect behavior of a ring oscillator with mismatched inverters. The second inverter's repressor binding coefficient is three times lower than the original, while the third inverter's transcription rate is twice as strong as the original.

end of the signal values. This results from the skewed transfer curve that describes the steady state characteristics of the inverters. Basically, for each inverter in the circuit a low level of the input repressor is sufficient to inactivate the corresponding promoter. Once the input of an inverter reaches that threshold, the inverter's output will begin to decay. Sections 3.3.3 and 4.1.2 discuss methods for correcting the skew in the transfer curve.

While the circuit oscillates correctly when the gates are perfectly matched, incorrect behavior may result from coupling mismatched components. Figure 3-6 shows the effect of mismatched inverters on the dynamic behavior of the ring oscillator. The inverters have different binding coefficients and transcription rates. Specifically, the values of the kinetic constants $k_{rprs(2)}$ and $k_{rprs(4)}$ for protein repressor B are now a third of the original cI values, and the value of $k_{xscribe}$ for C's promoter is now twice the original. As a result, the output of the inverter with the strongest transcription rate settles into HIGH, while the other two outputs settle into LOW. Clearly, correct behavior of the circuit is highly sensitive to the "device physics" of its components. Therefore, an integrated approach using BioSPICE and laboratory experiments is crucial for the success of biocircuit design.

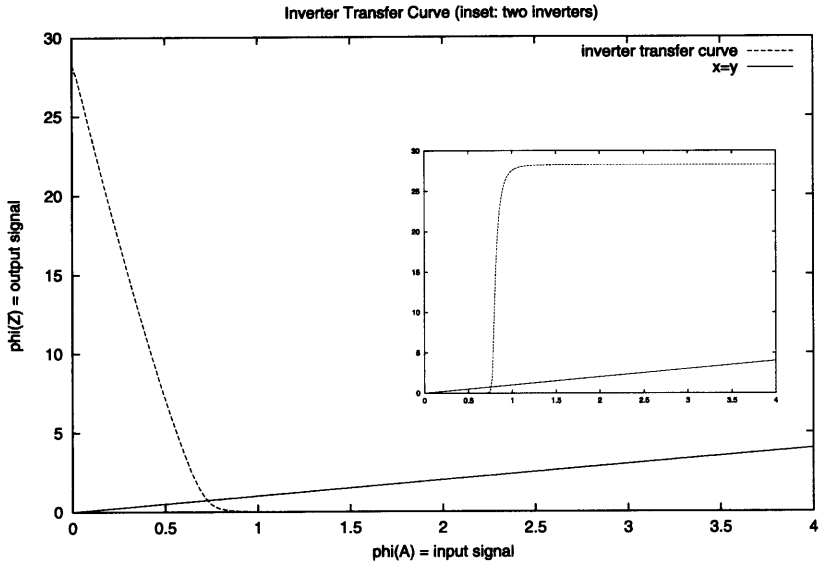


Figure 3-7: The simulated transfer curve of the λ cI/P(R) inverter, with the transfer curve of two such inverters connected in series shown in the inset. Both graphs plot ϕ_A versus ϕ_Z.

3.3 Steady State: Design and Analysis

As shown above, the correct dynamic behavior of bio-circuits is highly dependent on the steady state characteristics of the component gates. The complexity of bio-circuit design is exacerbated by the fact that each of the components uses different protein and dna-binding sites. These typically have widely varying kinetic characteristics. Therefore, a critical element in bio-circuit design is analyzing the steady state behavior of the cellular gates. The analysis motivates specific genetic modifications to achieve gate device physics that match with other gates, for correct design and construction of complex circuitry. This section describes BioSPICE simulations to compute the transfer curve of an inverter, the steady state conditions necessary for matching gates, BioSPICE analysis of genetic mutations to optimize the inverter's steady state characteristics, and prediction of the behavior of circuits using transfer functions of individual inverters.

3.3.1 Steady State Behavior

The transfer curve of an inverter, useful for analyzing the steady state behavior, maps an input level ϕ_A to an output level ϕ_Z. Figure 3-7 shows the transfer curve of an inverter with the kinetic rates from Table 3.3, as computed by BioSPICE. To compute the transfer curve of a given circuit, BioSPICE performs a set of simulations, where each simulation has a different steady rate of input signal synthesis. For each synthesis rate, BioSPICE records the level of the corresponding output signal if the system settles into a steady state. In this case, the definition of a steady state is some number of simulation time steps where all the state variables do not fluctuate by more than a small threshold. Each simulation that settles into a steady state contributes a point to the approximation of the transfer curve. The inset illustrates the transfer curve of two such inverters connected in series.

The skewed curve shows that an inverter based on the λ *cI* repressor and P(R) promoter is very sensitive to low concentrations of the input protein. Thus, for a HIGH output signal, even small increases in the synthesis rate of the input may alter the output signal to LOW. Section 3.3.3 discusses how to modify the behavior of these gates in order to make them more robust to such fluctuations. Chapter 4 describes in detail how to measure the transfer curve, systematic fluctuations, and noise of *in-vivo* biochemical gates. The discussion also reports on experimental results in measuring and modifying the transfer curves of these gates that achieve more balanced transfer curves.

3.3.2 Matching Thresholds

An inverter's transfer curve describes the behavior of a single gate, but the information is also useful for connecting multiple gates into operational logic circuits. Transfer functions suitable for implementing digital-logic circuits gates must have LOW and HIGH ranges such that signals in the LOW range map strictly into the HIGH range, and vice versa. The shape of the curve, represented by its *gain* (i.e. slope), determines how well the gate reduces noise from input to output. For electronic digital circuits,

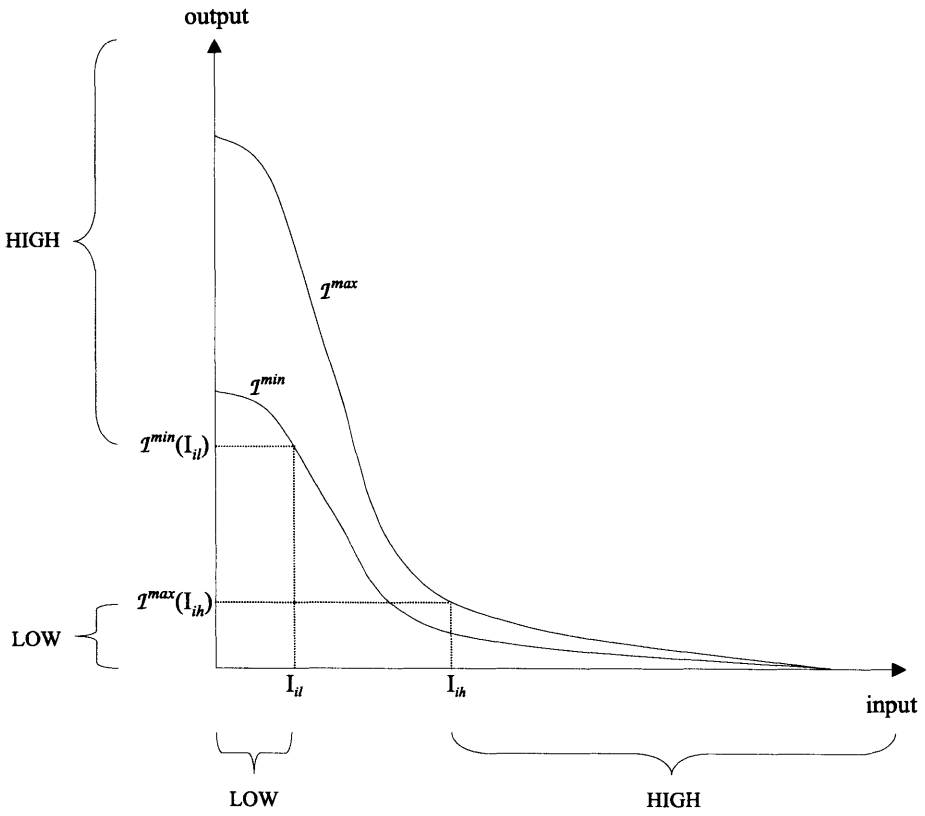


Figure 3-8: Transfer band thresholds: HIGH and LOW input ranges for a hypothetical inverter. The transfer band, capturing systematic fluctuations in signals, is defined by the two curves.

the LOW and HIGH signal ranges are typically the same for all gates because the circuit is composed of transistors with identical threshold voltages, spatially arranged. However, in biochemical digital circuits, the gate components (e.g. proteins and promoters) have different characteristics depending on their reaction kinetics. Therefore, the designer of biological digital circuits must take explicit steps to ensure that the signal ranges for coupled gates are matched appropriately, as described below.

Before discussing the rules for biochemical gate coupling, I introduce a variation of the transfer function, the *transfer band*, that captures systematic fluctuations in signal levels. It is especially important to consider fluctuations in biological settings. Experiments in Chapter 4 report signals fluctuatation by a factor of ten for cells with the same genetic circuit grown under the same conditions. The transfer band

captures these fluctuatations with a region enclosed by a pair of transfer functions, as shown in Figure 3-8. $\mathcal{I}^{min}$ is the function that maps an input to the minimum corresponding observed output, and $\mathcal{I}^{max}$ is the function that maps an input to the maximum corresponding observed output.

Let I_{il} and I_{ih} be the input thresholds. Then, the LOW and HIGH gate matching requirement from above is:

$$\begin{array}{lcl} \text{[in LOW]} & \langle 0, I_{il} \rangle \xrightarrow{\text{into}} \langle \mathcal{I}^{min}(I_{il}), \mathcal{I}^{max}(0) \rangle & \text{[out HIGH]} \\ \text{[in HIGH]} & \langle I_{ih}, \infty \rangle \xrightarrow{\text{into}} \langle 0, \mathcal{I}^{max}(I_{ih}) \rangle & \text{[out LOW]} \end{array}$$

Consider the case of two inverters, I and J, with J's output coupled to I's input. Then, the coupling is correct *iff*:

$$\begin{aligned} \langle \mathcal{J}^{min}(J_{il}), \mathcal{J}^{max}(0) \rangle &\subset \langle I_{ih}, \infty \rangle \\ \langle 0, \mathcal{J}^{max}(J_{ih}) \rangle &\subset \langle 0, I_{il} \rangle \end{aligned}$$

Then, assuming monotonic functions, the following conditions are necessary and sufficient for correct coupling:

$$\begin{aligned} \mathcal{J}^{min}(J_{il}) &> I_{ih} \\ \mathcal{J}^{max}(J_{ih}) &< I_{il} \end{aligned}$$

3.3.3 Genetic Process Engineering

The first step in developing biocircuits is designing, building, and characterizing several inverters. It is likely that these inverters will not match correctly according to the definitions above. Fortunately, there are biochemical techniques to adjust inverters to obtain the correct behavior for use in complex circuits (Figure 3-9). These include:

- **Modifying the repressor/operator binding affinity:** Certain base pair mutations at an operator site alter the affinity of the repressor to the operator.

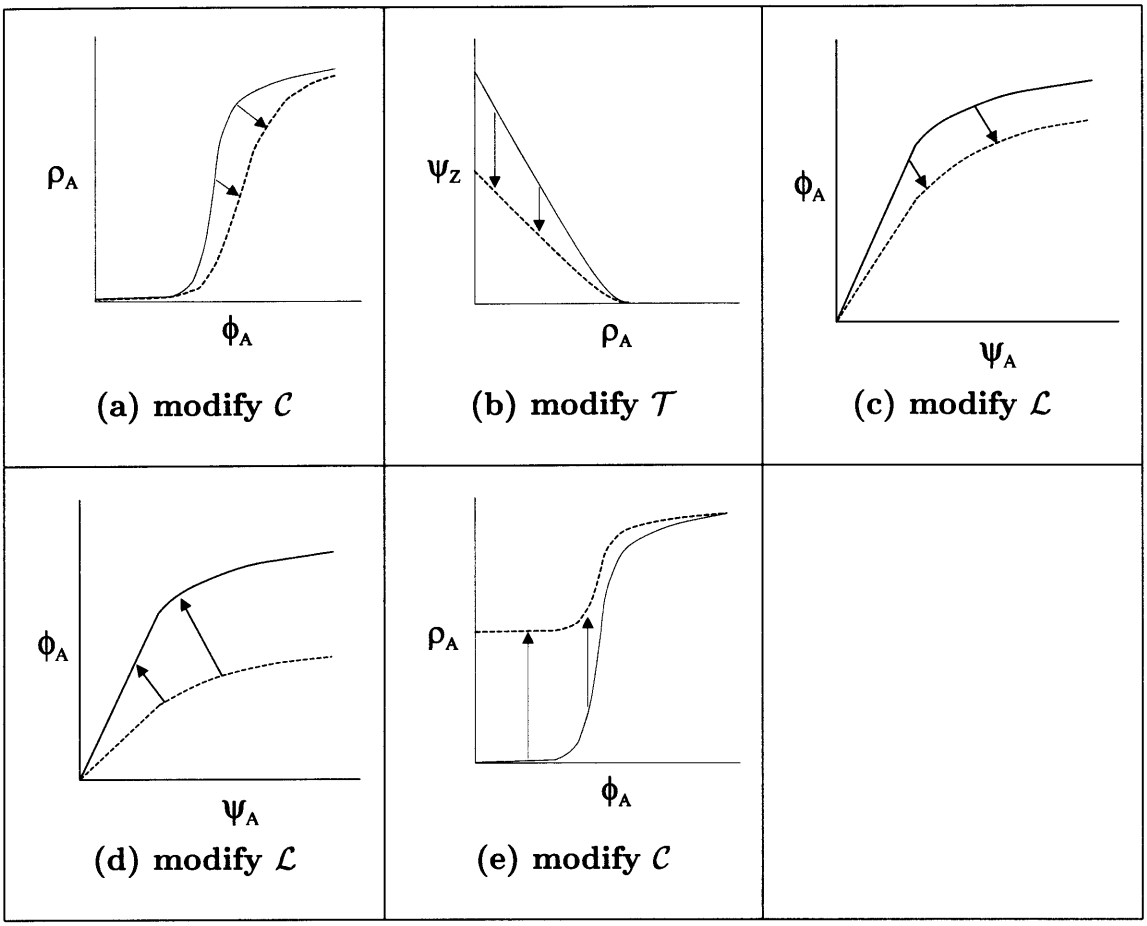


Figure 3-9: Modifications to specific stages in the inversion process: **(a)** Reducing the repressor/operator binding affinity **(b)** Reducing the strength of the promoter **(c)** Reducing the strength of the RBS **(d)** Increasing the cistron count **(e)** Adding autorepression

As illustrated in Figure 3-9(a), weakening the repressor affinity shifts $\mathcal{C}$, the cooperative binding stage mapping the input signal ϕ_A to the repression activity ρ_A, outward to the right. For a given level of the input repressor protein, there is now less repression activity than before the mutation. The mutations result in different behaviors for each repressor/operator pair. Figure 3-10 shows BioSPICE simulations where hypothetical reductions in the repression affinity of cI to O_R1 shift the overall transfer function of the cI/P(R) inverter outward. In these simulations, the original values of $k_{rprs(a2)}$ and $k_{rprs(a4)}$ were scaled down by factors of ten, one hundred, and one thousand. Section 4.4.2 reports experimental transfer functions results where the affinity of cI to the λ

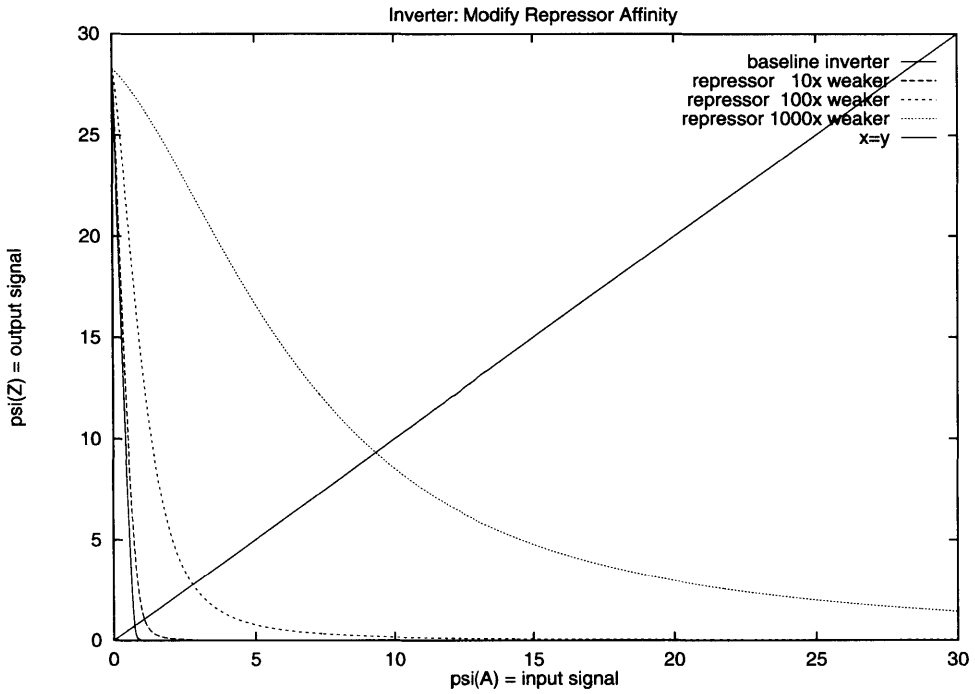


Figure 3-10: Reducing the binding affinity of the repressor to the operator shifts the transfer function of an inverter outward.

promoter is modified by mutating base pairs on OR_1. The experimental results demonstrate the desired shift outward.

- **Modifying the strength of the promoter:** Certain base pair mutations at the promoter site alter the affinity of the RNA polymerase to the promoter. DNA sequence determinants of promoter strengths have been studied extensively [17, 87, 44]. As illustrated by Figure 3-9(b), weakening a promoter shifts $\mathcal{T}$, the transcription stage mapping between ρ_A and ψ_Z, inward. Any given level of the active promoter (the complement of ρ_A) now results in less transcription and therefore less mRNA output (ψ_Z). BioSPICE simulations (Figure 3-11) show the overall inward shifts of the transfer curve of the cI/P(R) inverter that result from weakening the P(R) promoter. In these simulations, the original value of $k_{xscribe}$ was scaled down by factors of two, five, and ten.

- **Modifying the strength of the ribosomal binding site (RBS):** Modi-

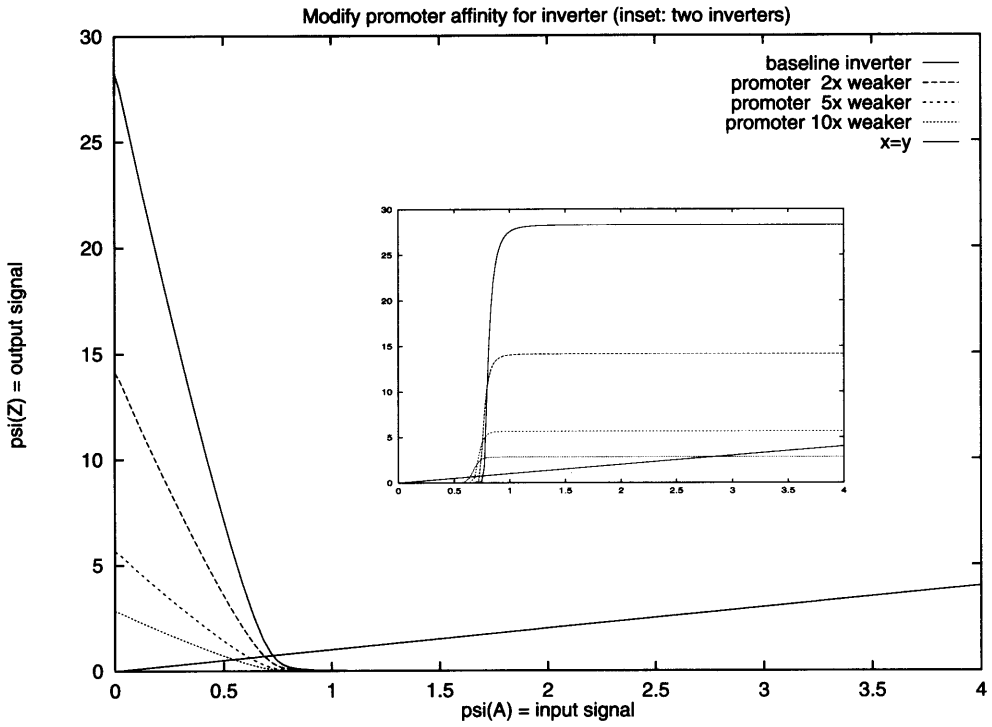


Figure 3-11: Reducing the binding affinity of the RNA polymerase shifts the transfer functions of the cI/P(R) inverter inward. Inset shows the effects on the transfer functions of two inverters in series. The diagonal lines correspond to input equals output.

fications to the RBS alter the affinity of rRNA to the mRNA. Figure 3-9(c) illustrates the shift in $\mathcal{L}$, the translation stage mapping between ψ_A and ϕ_A, resulting from a reduction in rRNA affinity. For any given level of the input mRNA ψ_A there is now less translational activity and therefore less input protein ϕ_A. Section 4.4.1 reports on experimental transfer curve results with different ribosome binding sites for cI. Weaker RBS's caused an outward shift to the overall transfer curve of the original cI/P(R) inverter.

- **Increasing the cistron count:** The cistron count is increased by adding copies of the coding sequence for the output protein downstream of the promoter. As illustrated in Figure 3-9(d), the increase in cistron count on shifts $\mathcal{L}$, the translation stage mapping between ψ_A and ϕ_A, upward. For any given level of the mRNA input ψ_A there is now additional input protein repressor ϕ_A. The curve

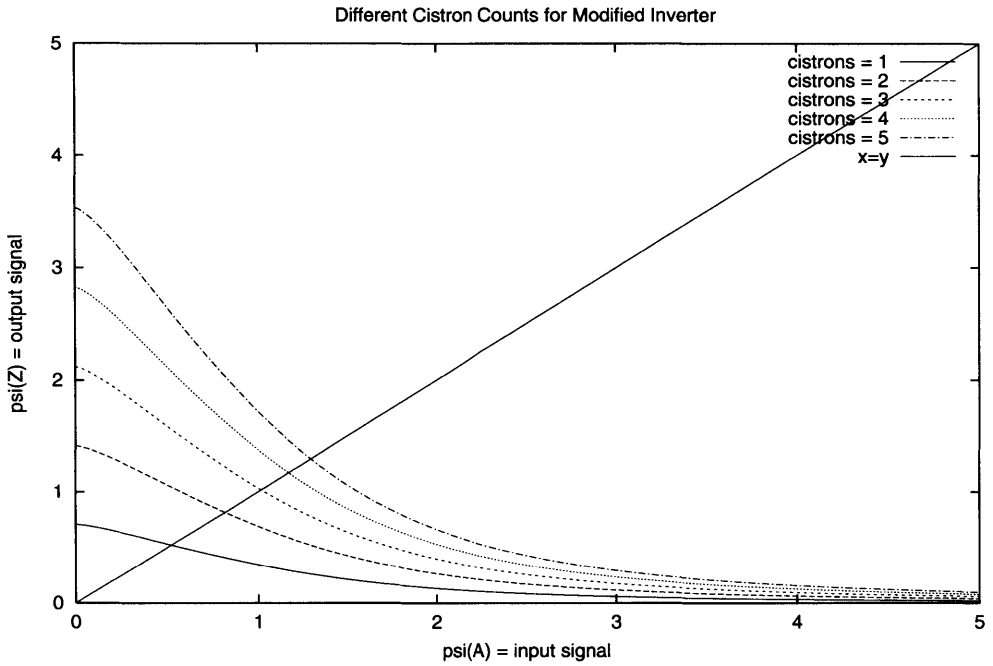


Figure 3-12: The effects of increasing the cistron count per gene (i.e. the number of structural genes per operator region). In this case, the original inverter mechanism has been modified to 100x less binding affinity of the repressor and 40x less binding affinity of the promoter.

values are scaled by a linear factor in the initial range of the translation before reaching metabolic constraints of the cell's translational machinary. BioSPICEs simulations (Figure 3-12) demonstrate the overall outward shift of the transfer curve of the cI/P(R) inverter as the cistron count increases from one to five.

- **Altering the degradation rate of a protein:** The degradation rate of proteins can be normally achieved by changing a few amino acid residues on the C terminus [13, 62, 63]. For example, decreasing the half life of the input protein A causes $\mathcal{L}$, the translation stage mapping between ψ_A and ϕ_A, to shift downward. Because of the increase in the protein degradation rate, for any given level of the input mRNA ψ_A, there is now less repressor protein ϕ_A present. Another possibility for increasing the degradation rate is to choose bacterial strains that have high concentrations of effective proteases [80].

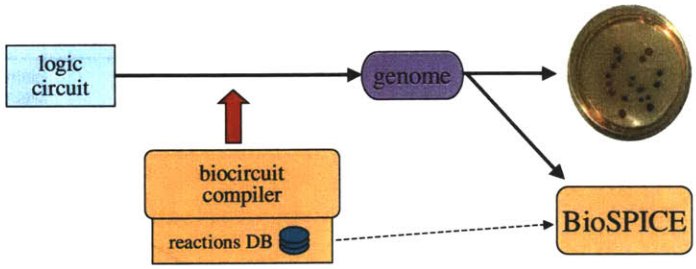


Figure 3-13: Biocircuit design components, showing compilation of a logic circuit into a genome that encodes that circuit, and BioSPICE, a tool for simulating and analyzing the resulting genome.

- **Adding autorepression:** An operator that binds the output protein may be added to the promoter/operator region to accomplish autorepression. The simulated transfer functions shown above are not balanced because they are more sensitive to fluctuations in LOW input signals versus fluctuations in HIGH input signals. Autorepression reduces the maximum output protein synthesis rate and therefore reduces some of the asymmetry in the signal sensitivities. Figure 3-9(e) illustrates the effect of adding autorepression on $\mathcal{C}$, the cooperative binding stage of inversion. For low levels of the input protein repressor, the promoter is active and transcribes the output protein. This output protein binds to the operator of its own promoter, and therefore there is always some concentration of inactive/bound promoter ρ_A. High levels of input repressor increase ρ_A until saturation.

3.3.4 Functional Composition of Transfer Functions

In *biocircuit design*, the engineer creates a genetic logic circuit using a small set of basic gates and a database of biochemical reaction kinetic rates (Figure 3-13). The kinetics database contains transfer curve measurements obtained using the mechanism described in Chapter 4. Given transfer curve measurements, the engineer predicts the behavior of complex circuits through the functional composition of behavioral data describing only the basic components. For example, Figures 3-7 and 3-11 show the

predicted steady state behavior of circuits with two inverters based on the transfer curves of the constituent inverters.

Without autorepression, the transfer function of an inverter is determined by the input mRNA ϕ_A, RBS, input protein A, operator, promoter, and output mRNA ϕ_Z, but not by the output protein Z. This means that the relation $\phi_Z = \mathcal{I}_1(\phi_A)$ does not depend on Z. Therefore, gate couplings are independent of the output protein. To predict the transfer function of two gates in a series, $\phi_Y = \mathcal{I}_2(\phi_Z)$ connected to $\phi_Z = \mathcal{I}_1(\phi_A)$, the database only needs to contain $* = \mathcal{I}_2(\phi_Z)$ and $* = \mathcal{I}_1(\phi_A)$, where $*$ denotes any protein. In this thesis, I use the cyan and yellow derivatives of the green fluorescent protein[35] to measure transfer curves. The measurements I obtain with these reporters allow prediction of the composition of gates.

If the inversion uses autorepression, then the relation $\phi_Z = \mathcal{I}_3(\phi_A)$ also depends on the characteristics of Z. To compute the transfer function of a gates coupling $\phi_Y = \mathcal{I}_4(\phi_Z)$ connected to $\phi_Z = \mathcal{I}_3(\phi_A)$, the database must specifically include the transfer functions of the input/output protein pairs Y/Z and Z/A and their associated mRNA.

Using the transfer curve measurements, the efficacy of a particular transfer function in implementing the digital abstraction is evaluated in terms of factors such as gain and noise margins. The transfer function of inverters in a series is the functional composition of their respective transfer functions. A series of inverters is then also evaluated in terms of gain and noise margins. Because the transfer functions are different for each inverter, the gates must be matched as described in Section 3.3.2. In addition, the matching process must also evaluate the gain and noise margins resulting from gate couplings, and only select proteins that achieve satisfactory levels of these factors.

3.3.5 Implementation Choices

The objective of *biocircuit design* is to take a desired logic circuit and a database of kinetic rates as input, and produce a genetic network that implements the circuit. The design process searches the database and assigns suitable proteins to each gate, where

the overall behavior of the circuit depends on these choices. To prevent interference between the gates, a different protein is used for each unique signal. Therefore, the number of proteins needed to implement a circuit is proportional to the complexity of the circuit. The gates must be robust enough to use a wide variety of proteins with different reaction kinetics. Some of the key choices in biocircuit and gate design are:

- *Global gene copy number:* The circuits are typically implemented on one or several plasmids. High copy number plasmids place a metabolic burden on the cell, while low copy number plasmids may result in large stochastic noise. In this thesis, I use medium copy number plasmids (e.g. 20-50 copies per cell) based on *pBR322* and *p15A* origins of replication.

- *Output proteins:* An output protein must be soluble, bind some known operator site(s), and not interfere with normal cell function. To ensure sufficient gain and noise margins, DNA binding should be highly cooperative (*e.g.* λ *cI* represses using two dimers).

- *Promoter/operator regions:* Operators should bind repressors cooperatively, and promoters should not be too strong as they may saturate subsequent gate inputs and interfere with normal cell metabolism.

- *Signal threshold levels:* The gate input thresholds must be chosen to provide high gain near the switching threshold, adequate noise margins at the HIGH and LOW signal levels, and balanced transition times.

- *Per-gate cistron count:* The cistron count can be adjusted for each output protein to match threshold levels.

- *Plasmids versus chromosomal encodings of circuits:* A genome encodes the entire logic circuit with a set of genes, each with its own promoter, operator, and structural genes. Normally this genome construct is encoded on one or more extracellular DNA strands (e.g. plasmids). In the future, chromosomal insertions of genetic circuits may significantly improve the stability of DNA coding

for the biocircuit in the organism. The chromosomal encodings may also allow more complex circuitry as these circuits will place a smaller metabolic burden on the cell.

Caveats

Published data on kinetic constants is scarce and often imprecise. In several cases, the constants used in the simulations here were guessed from published equilibrium constants. This situation is rapidly getting better, and it is expected that more accurate and complete data will be available in the near future. The experiments in Chapter 4 provide some initial data for this purpose.

Another issue that must be address is stochastic noise in gene expression. In cells, typical promoter copy counts correspond to very low concentrations. Therefore, the stochastic noise in concentrations resulting from the discreteness of the transcription reactions can be significant (see *e.g.* Arkin and Ross [5]). To decrease this stochastic variance, I use medium plasmid copy numbers. There are also several known plasmids, such as pSC101, with a tight copy number control mechanism and therefore significantly reduce the plasmid copy number fluctuations. Finally, recent results[83] suggest that decreasing the translational efficiency while increasing the transcriptional efficiency reduces fluctuations in gene expression.

As is apparent from the discussion above, there are many biochemical mechanisms that can be explored in order to design reliable intracellular gates and circuits. Next, we turn our attention to designing intercellular communications.

3.4 Intercelllar Communications

BioSPICE also provides a platform for simulating and analyzing intercellular communications coupled to genetic logic circuits. The input to an intercellular simulation is a network of gene expression systems (including the relevant protein products) and a small layout of cells on some medium. BioSPICE consults a database of reaction kinetics and diffusion rates to simulate the dynamic characteristics of the target system.

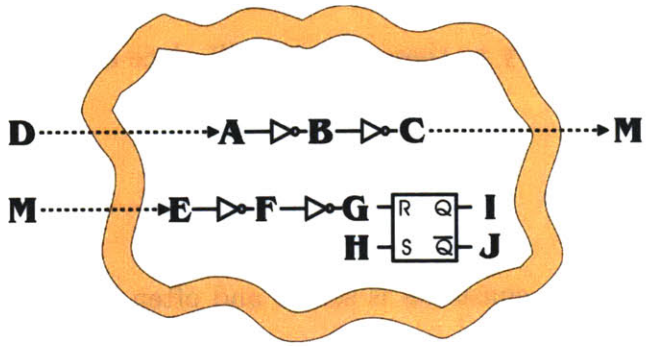
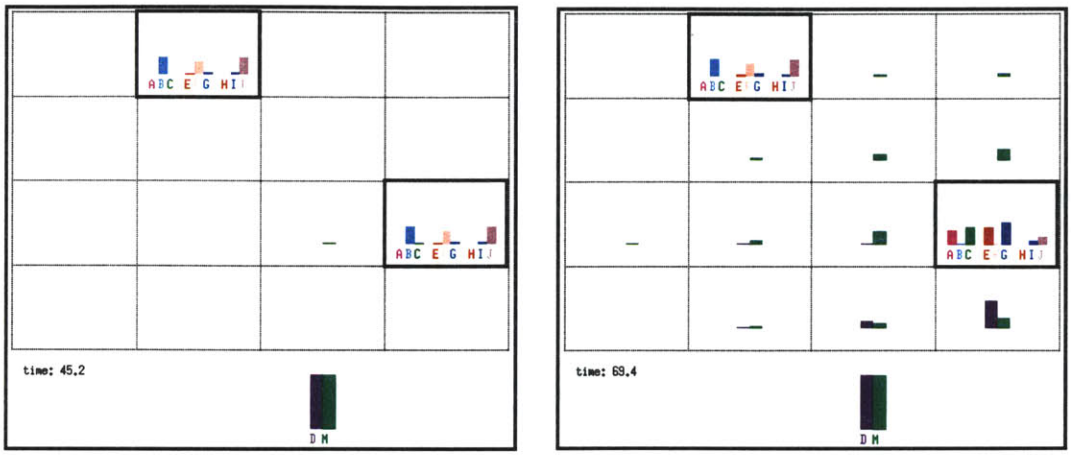


Figure 3-14: Communications Circuit: Gate level representation of a genetic circuit to accomplish a simple bacterial task.

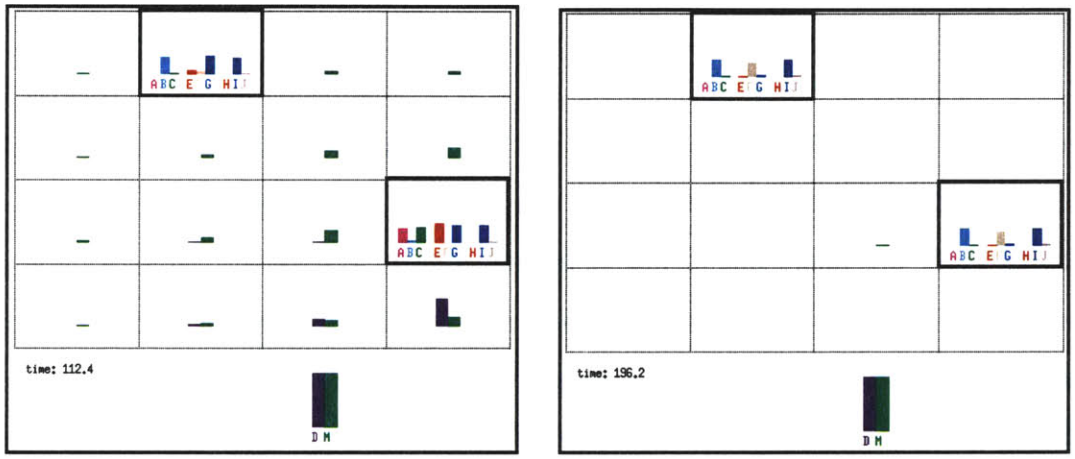
The simulation computes the time-domain behavior of concentrations of intracellular genetic elements and intercellular message passing chemicals.

Consider a simple bacterial task, where upon receipt of a message (represented by inward diffusion of a message passing chemical), a cell communicates to its neighbors and instructs them to set a state bit. Figure 3-14 presents an intercellular genetic logic circuit designed to perform this task. The initiating signal D is a chemical that traverses the cell membrane and results in the presence of protein A in the cytoplasm. Chapter 4 demonstrates how inducer molecules such as IPTG (Isopropylthio-β-galactoside) and aTc (Anhydrotetracycline) perform this function. The presence of A results in controlled synthesis of C. To amplify the signal, the inverter with input A can by adjusted to be sensitive to even small quantities of A. Once a sufficient concentration of A accumulates, the cell synthesizes C and secretes it into the surrounding environment as chemical message M. M diffuses through the medium and functions as a message to neighboring cells. In response to M, the neighbors each set their RS latch with output I to HIGH.

Figure 3.4 shows snapshots from a BioSPICE simulation of the above system on a 4×4 grid (representing the medium) with two bacterial cells (heavily shaded squares). Initially (image one), the output of the RS latch (represented by I) is LOW. Then, an external drive D is introduced into the environment next to the cell at the bottom-right corner (image two). The external drive instructs the bottom-right cell



(1) original, $I = 0$ (2) introduce D, send msg M



(3) receive msg, set I (4) msg decays, I latched

Figure 3-15: Biospice simulation snapshots showing intracellular protein and intercellular message chemical concentrations for the simple task described in this section. The shaded rectangles represent cells, and the rest of the grid represents the medium. The vertical bars denote the levels of protein concentrations.

to transmit a message M to its neighbors. Once the neighboring cell receives M, denoted when E is present, the cell uses G to set the RS latch. Finally, when the external drive dissipates and the message M decays, the value of I remains latched at HIGH.

The next chapter describes measurements and genetic modifications of *in-vivo* logic gates to obtain components with the desired behavior for constructing complex and reliable circuits.

Chapter 4

Measuring and Modifying Device Physics

Potentially the most important element of biocircuit design is matching gate characteristics. Experimental results in this chapter demonstrate that circuits with mismatched gates are likely not to function correctly. In generating Biology's complex genetic regulatory networks, natural forces of selection have resulted in finely tuned interconnects between the different regulatory components. Nature has optimized and matched the kinetic characteristics of these elements so that they cooperatively achieve the desired regulatory behavior. In building *de-novo* biocircuits, we frequently combine regulatory elements that do not interact in their wild-type settings. Therefore, naive coupling of these elements will likely produce systems that do not have the desired behavior.

In *genetic process engineering*, the biocircuit designer first determines the behavioral characteristics of the regulatory components, and then modifies the elements until the desired behavior is attained. In this chapter, I show experimental results of using this process to convert a non-functional circuit with mismatched gates into a circuit that achieves the correct response. The experiments focus on examining and modifying the steady state behavior of the genetic circuits, and represent the first example known to the author of designing robust genetic regulatory components for use in building reliable biocircuits of significant complexity. Future work will also

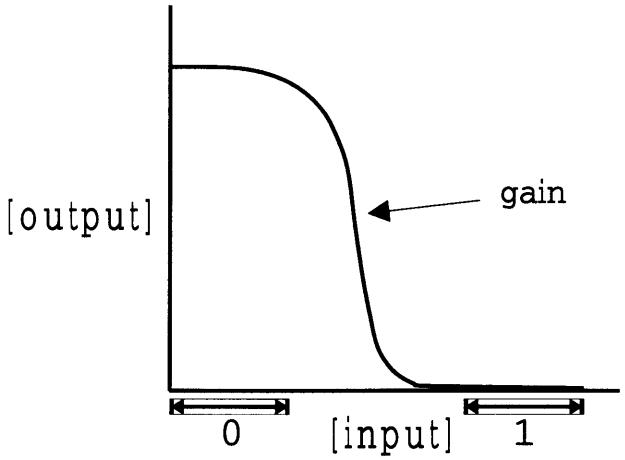


Figure 4-1: An idealized transfer curve for an inverter: The gain (or slope) is flat, then steep, then flat again. Analog ranges represent digital "zeros" and "ones." Because of the gain, the output of the inverter is a better representation of the digital value then the input (i.e. signal restoration).

consider the dynamic behavior of the circuits. The ultimate goal of the research reported in this chapter is to assemble a library of components with known and useful "device physics", akin to the TTL Data Book for electrical circuit design. The knowledge of device physics plays a fundamental role in achieving predictable and reliable biocircuit design.

In this chapter, Section 4.1 describes the formal framework for measuring signals and transfer curves. Section 4.2 introduces circuits used to experimentally set *in-vivo* signal levels. Section 4.3 reports on the transfer curve of the lacI/p(lac) inverter. Section 4.4 describes measurements and modifications to the transfer curves of inverters based on $cI/\lambda_{P(R-O12)}$. Section 4.5 reports on the transfer curve of an IMPLIES gate based on lacI/p(lac). Finally, Section 4.6 discusses several physical constraints that limit the size and complexity of circuits that can be embedded in single cells.

4.1 Signals and Transfer Curves

A transfer function is the relation between the input signal and the output signal of a gate or a circuit in steady state. Figure 4-1 illustrates an idealized transfer curve for an inverter that is required for achieving the digital abstraction. Section 4.1.1 describes how to measure an individual signal in a genetic circuit, by constructing a probe that measures expression activity *in vivo*. Section 4.1.2 introduces a mechanism for estimating the transfer function by measuring many different points of the transfer curve. The techniques outlined in this section are used to measure and analyze the characteristics of *in vivo* logic gates in subsequent sections in this chapter.

4.1.1 Measuring a Signal

Recall that ϕ_X, the mRNA level $mRNA_X$ of signal protein X from all genes coding for it, represents a logic signal. This section derives a direct correlation between measurements of fluorescence intensities for reporter proteins and actual signal intensities.

Assume only gene G_X codes for $mRNA_X$. Then, the relevant chemical reactions for a signal is the rate of transcription of the gene G_X and the decay rate of $mRNA_X$:

$$G_X + RNA_p \xrightarrow[\text{transcribe}]{k^X_{xscribe}} G_X + RNA_p + mRNA_X \tag{4.1}$$

$$mRNA_X \xrightarrow[\text{decay}]{k^X_{dec(mrna)}} \tag{4.2}$$

G_X is the active (unrepressed) form of the gene, and $k^X_{xscribe}$ represents the rate of transcription from G_X into the $mRNA$ product. Then, assuming this is the only production of X in the system, the signal level ϕ_X in steady state is:

$$\phi_X \equiv [mRNA_X] = \frac{k^X_{xscribe} \cdot [G_X] \cdot [RNA_p]}{k^X_{dec(mrna)}} \tag{4.3}$$

To measure the signal $mRNA_X$, insert a reporter protein RP_1 as an additional

structural gene into $mRNA_X$ with its own ribosome binding site. Let n represent the cistron count of signal protein X. Also, assume that for the concentrations of interest, RP_1 remains mostly in monomeric form:

$$mRNA_X + rRNA \xrightarrow[\text{translation}]{k_{xlate}^{RP_1}} mRNA_X + rRNA + n \cdot X + RP_1 \tag{4.4}$$

$$RP_1 \xrightarrow[\text{decay}]{k_{dec(rp_1)}} \tag{4.5}$$

Then, the time derivative for the reporter concentration is:

$$\frac{d[RP_1]}{dt} = k_{xlate}^{RP_1} \cdot [mRNA_X] \cdot [rRNA] - k_{dec(rp_1)} \cdot [RP_1] \tag{4.6}$$

At steady state:

$$0 = k_{xlate}^{RP_1} \cdot [mRNA_x] \cdot [rRNA] - k_{dec(rp)} \cdot [RP_1] \tag{4.7}$$

Then, the signal level is:

$$\phi_X \equiv [mRNA_X] = \frac{k_{dec(rp_1)}}{k_{xlate}^{RP_1} \cdot [rRNA]} \cdot [RP_1] \tag{4.8}$$

For a useful operating range, the reporter's fluorescence intensity F_1 approximates its concentration (by a multiplicative factor m_1):

$$[\mathrm{RP_1}] = F_1 \cdot m_1 \tag{4.9}$$

Then, a fluorescence measurement correlates to the actual signal by:

$$\phi_X \equiv [mRNA_X] = \frac{k_{dec(rp_1)} \cdot m_1}{k_{xlate}^{RP_1} \cdot [rRNA]} \cdot F_1 \tag{4.10}$$

Assuming that ribosomal RNA (rRNA) levels average to be fairly constant, then the actual *mRNA* signal can be approximated by the fluorescence intensity multiplied by a constant (unknown) factor. This signal approximation is useful for biocircuit design because knowing the relative signal values is sufficient for matching gates.

Measuring multiple signals

There are at least two alternatives for measuring multiple signals in the same circuit. One approach is to build multiple DNA constructs, where for each construct, the same reporter protein is inserted into one of the desired *mRNA* coding sequences. Then, each signal is measured separately in a different experiment where the conditions are duplicated as much as possible.

The second approach to measuring multiple signals is to use several different fluorescent proteins in the same construct. This approach allows simultaneous measurements of multiple signals in the same cell during the same experiment. For a signal ϕ_N that is associated with a reporter protein RP_n:

$$\phi_N \equiv [mRNA_N] = \frac{k_{dec(rp_n)} \cdot m_n}{k_{xlate}^{RP_n} \cdot [rRNA]} \cdot F_n \tag{4.11}$$

To normalize the simultaneous readings of different reporter proteins, first use the same RBS upstream of each reporter coding sequence. This sets the translation rate, $k_{xlate}^{RP_n}$, to be equivalent. Also, use reporter proteins with the same decay rate $k_{dec(rp_n)}$. Finally, the multiplicative factor m_n is correlated between the different reporter proteins by the method described in Section 4.2.

Signal representation

The choice to represent signals using mRNA rather than dna-binding proteins as originally proposed by Knight and Sussman[45] is more convenient for several reasons. There are two possibilities for using fluorescent proteins for signal measurements. In the first approach, the reporter fluorescent protein is fused to the dna-binding protein in either the N or C terminus. Here, a transcription event synthesizes a single protein-fusion molecular complex that contains both the dna-binding protein and the fluorescent protein.

While the fluorescence intensity in this approach is a direct measurement of the concentration of the dna-binding protein, the approach suffers from two drawbacks. First, the protein-fusion may alter the behavior of the original dna-binding protein under study. Second, certain dna-binding proteins such as the Bacteriophage λ *cI* are very efficient repressors, and even a small concentration of these repressors shuts down transcription from their corresponding promoters. Due to the auto-fluorescence of cells, such low protein concentrations cannot be detected with fluorescent proteins *in-vivo*. Therefore, with this method one cannot effectively measure the characteristics of gates based on highly efficient repressor proteins.

The second approach to measuring dna-binding protein signals using fluorescent reporters is to encode a polycistronic reporter[1], as in equation 4.4. However, this approach requires that both the signal protein and the fluorescent reporter have the same synthesis and decay rates. Otherwise, their concentrations will diverge, and the fluorescence intensity will not accurately reflect the concentrations of the signal protein. The efficiency of the ribosome binding site (RBS) largely controls the synthesis rate, and both proteins can have the same RBS to achieve approximately the same synthesis rates. However, in certain situations, it can be quite difficult to engineer both proteins to exhibit the same decay rates. Furthermore, this approach suffers from the same drawback as above when trying to measure the behavior of highly efficient dna-binding proteins.

[1] A polycistronic gene organization is one where a single mRNA is transcribed, but it codes for multiple gene products, where each product is translated independently from the mRNA

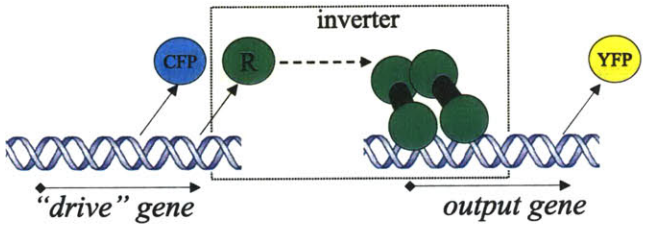


Figure 4-2: Genetic setup used for measuring transfer curves. The "drive" genetic construct enables the researcher to control and observe the input signal, using a fluorescent protein (e.g. CFP). Simultaneously, the output signal is observed using a different fluorescent protein (e.g. YFP).

As shown in equations 4.10 and 4.11, using mRNA as the signal enables measurements that are both directly correlated with the actual signal and where low repressor protein concentrations can still be observed. Specifically, the RBS for the reporter protein is typically engineered to be highly efficient, while the corresponding RBS for the repressor protein can be quite weak. In this manner, a gate with a highly efficient repressor can be characterized, because low concentrations of the repressor protein will be accompanied with higher concentrations of the reporter protein.

4.1.2 Measuring the Transfer Curve of an Inverter

Once individual signals can be measured, the transfer function of a gate is estimated by measuring many points on the curve. A point on the transfer curve is a steady state relation between the mRNA level ϕ_A of the input protein and the mRNA level ϕ_Z of the output protein. A point is measured by constructing a system with an unknown but fixed ϕ_A, and measuring ϕ_A and ϕ_Z.

To obtain many points, construct a system that can yield various fixed values of ϕ_A, and observe the corresponding values of ϕ_Z. Figure 4-2 shows a genetic setup that enables a researcher to control and observe input protein levels, and simultaneously observe resulting output levels. Let ${P_D}^j$ represent a promoter under a certain condition (i.e. "drive") that results in a fixed value of ϕ_A, say ${\phi_A}^j$. Let $\mathcal{I}$ denote the transfer function of inverter I. Then, for each drive ${P_D}^j$, the value pair $\left(\frac{{\phi_A}^j}{k_{dec(rp)}}, \frac{\mathcal{I}({\phi_A}^j)}{k_{dec(rp)}}\right)$ is

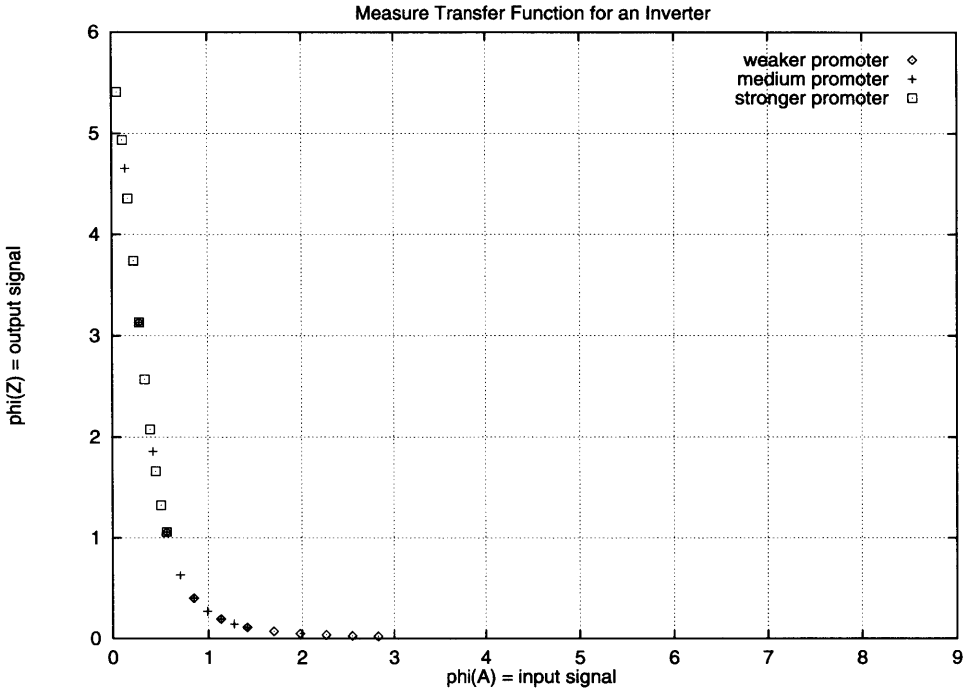


Figure 4-3: Simulation for measuring points on the transfer function of an inverter using three promoters, each with ten different cistron counts.

measured with the reporter *RP* as described above. One measures this value pair in a single experiment using two different fluorescent proteins. With a set of these points, one obtains the transfer curve of a gate, where all points are normalized to the same (albeit unknown) units.

Figure 4-3 illustrates points on an inverter's transfer curve, obtained by simulating thirty different drive intensities. To measure a complete transfer curve, the range of inputs must cover both the LOW and HIGH input ranges. This may require different genetic constructs with both strong and weak promoters. One does not need to know *a priori* about the characteristics of $P_D{}^j$ to use it for measuring points on the transfer curve. In addition, drives with similar characteristics simply add redundancy to the measurements.

Next, we turn to laboratory experiments for measuring and modifying the device physics of several biocircuits and gates. Unless otherwise noted, in the remainder of this thesis all graphs represent data with actual living *E. coli* cells.

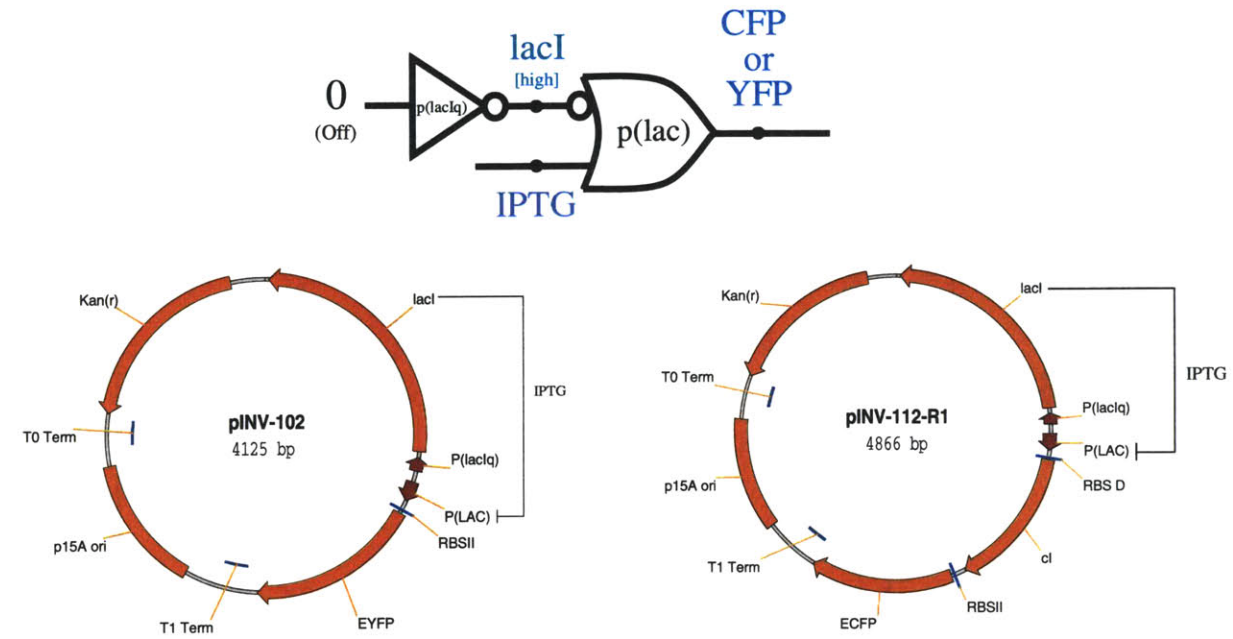


Figure 4-4: Plasmids with circuits to set protein expression levels. The concentration of IPTG (Isopropylthio-β-galactoside) controls the expression level of the output proteins, in these cases ECFP or EYFP. The inclusion of the *cI* coding sequence prior to the ECFP coding sequence should not have a noticeable effect on ECFP expression.

4.2 Genetic Constructs to Set a Signal Level

The first step in measuring the device physics of an inverter is to construct genetic circuits that allow the researcher to externally set the *in-vivo* level of a signal. This is performed using simple circuits with an inverter connected to an IMPLIES gate (Figure 4-4). I constructed two such circuits, one with the enhanced yellow fluorescent protein (EYFP) and one with the enhanced cyan fluorescent protein (ECFP), both from Clontech[35]. The inverter that comprises the constitutive promoter p(lacIq) has an input that is always set to LOW, because the cell does not contain a repressor for the p(lacIq) promoter. Therefore, the output of the inverter, *lacI*, is constantly HIGH. Then, since the *lacI* repressor input to the p(lac) IMPLIES gate is constantly HIGH, the level of the inducer molecule input, IPTG (Isopropylthio-β-galactoside), is positively correlated with the level of the output. The researcher controls the level of the output signal with this circuit by externally setting the level of IPTG, which

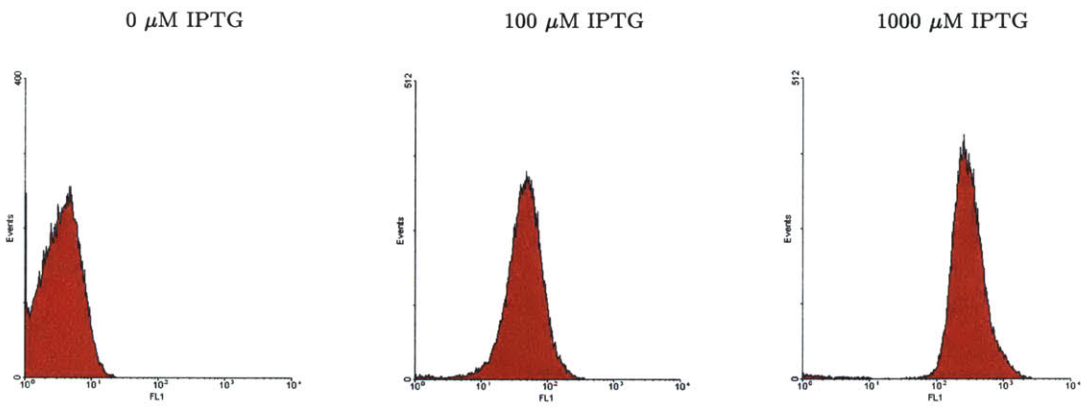


Figure 4-5: FACS data for IPTG inductions of pINV-102. The values for the 0μM culture reflect auto-fluorescence of the cells, while the values for the other two concentrations reflect high levels of EYFP expression.

freely diffuses into the cell.

Figure 4-5 shows Fluorescence-Activated Cell Sorting (FACS)[74] data of E. coli cells with the pINV-102 plasmid grown to steady state in three different concentrations of IPTG.[2] Each graph is a histogram correlating fluorescence intensities with the number of cells that have been observed with those particular fluorescence intensities. The cells that were grown in a medium containing 0μM IPTG emit approximately the same distribution of fluorescence values as cells that do not contain DNA that codes for any fluorescent protein (data not shown). Therefore, these fluorescence intensities are attributed to the auto-fluorescence of the cells. Cells grown in increasing levels of IPTG emit higher levels of fluorescence, representing higher levels of output mRNA and protein expression.

Figure 4-6 shows median fluorescence values of pINV-102 and pINV-112-R1 cell populations induced with a range of IPTG concentrations. The graph shows how to control an *in-vivo* signal using external induction with IPTG. Notice the sigmoidal shape of the fluorescence values. Initially, the IPTG concentrations have very little effect. Once the IPTG concentration reaches a critical threshold, the inducer

[2]Appendix A describes the materials and methods used for the experiments in this chapter.

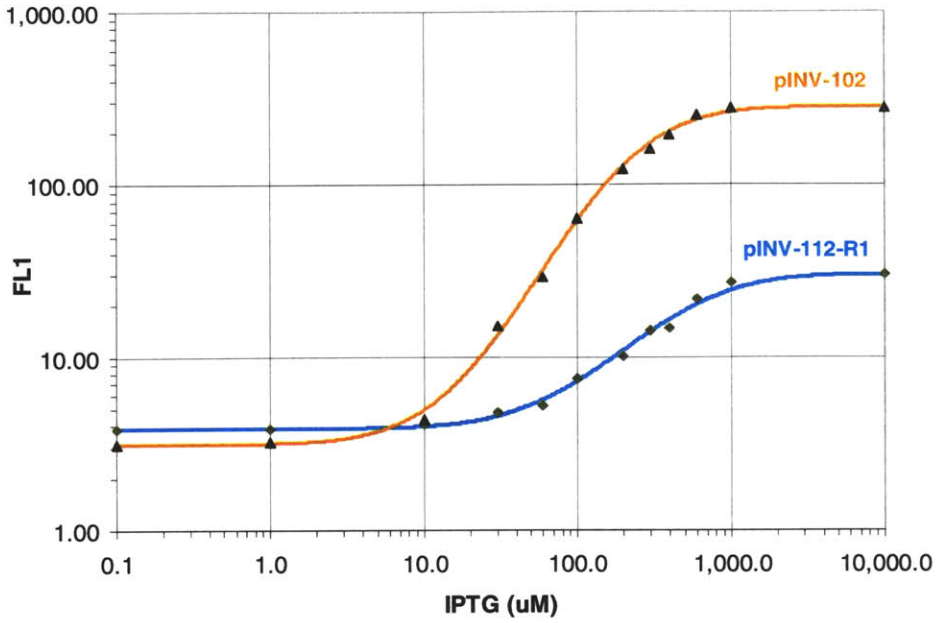


Figure 4-6: Controlling signal levels: The graph shows how to control an *in-vivo* signal using external induction with IPTG. The different curves show the fluorescence levels of ECFP and EYFP that correspond to the same signal level. The relationship between the two curves helps compute the normalization factor between ECFP and EYFP reporter levels.

molecules begin to de-activate enough *lacI* repressor molecules such that transcription from p(lac) proceeds occasionally. From 10μM to $1,000\mu$M the level of expression rises as the level of IPTG increases. Finally, at approximately $1,000\mu$M, when the IPTG molecules de-activate essentially all the *lacI* molecules, RNAp transcribes p(lac) at a maximum rate. Past the saturation point, additional IPTG has no effect on the output level.

The relationship between the ECFP and EYFP fluorescence intensities in Figure 4-6 is used to normalize between simultaneous ECFP/EYFP readings in subsequent experiments in this section. The graphs show sigmoidal curve fittings of the data points using the error function *erf*:

$$y = \gamma \cdot erf(\alpha \cdot x + \beta) + y_0 \tag{4.12}$$

where

$$erf(x) = \frac{2}{\sqrt{\pi}} \cdot \int_{0,x} e^{-t^2}\, dt \tag{4.13}$$

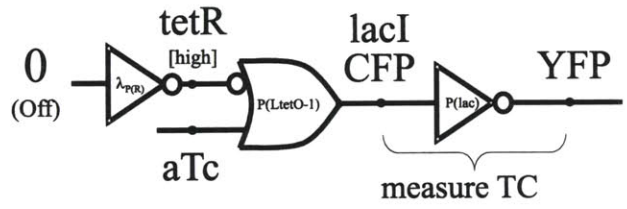
For a given IPTG level, the level of ECFP and EYFP expression is approximately the same. Therefore, a given ECFP fluorescence intensity f_c correlates to the corresponding EYFP fluorescence intensity by finding the IPTG level with an ECFP value of f_c and then determining the EYFP fluorescence for that level of IPTG.

These graphs demonstrate how to control the level of expression of a particular protein using an external stimulus. Such a genetic setup is used in the following sections to set the levels of input mRNA to the inverters under study.

4.3 Transfer Curve of a lacI/p(lac) Inverter

Figure 4-7 shows the genetic circuit used to measure the device physics of an inverter based on the *lacI* repressor and the p(lac) promoter. The first two logic gates set the level of the input signal to the inverter in a mechanism similar to one used in the circuit from Section 4.2. Here, the $\lambda_{P(R-O12)}$ inverter functions as a constitutive promoter (no *cI* in the system) to set a constant high level of the Tet repressor (*tetR*). Then, through the *tetR*/P(LtetO-1) IMPLIES gate, the concentration of the aTc (Anhydrotetracycline) inducer molecule controls the level of the lac repressor (*lacI*). *lacI* is the input protein to the inverter gate under study. The ECFP transcribed along with *lacI* reports the level of the input signal, as described in Section 4.1.1. Finally, EYFP reports the output signal expressed from the lacI/p(lac) inverter.

The output of this circuit is the logic NOT of the *aTc* input signal. Figure 4-8 shows FACS data of the EYFP output signal in different three experiments that results from varying the *aTc* input signal levels. For a LOW input signal of $1\frac{ng}{ml}$ aTc, the output of the circuit is appropriately HIGH. For a HIGH input signal of $100\frac{ng}{ml}$ aTc, the output of the circuit is correctly LOW. In addition, for an intermediate



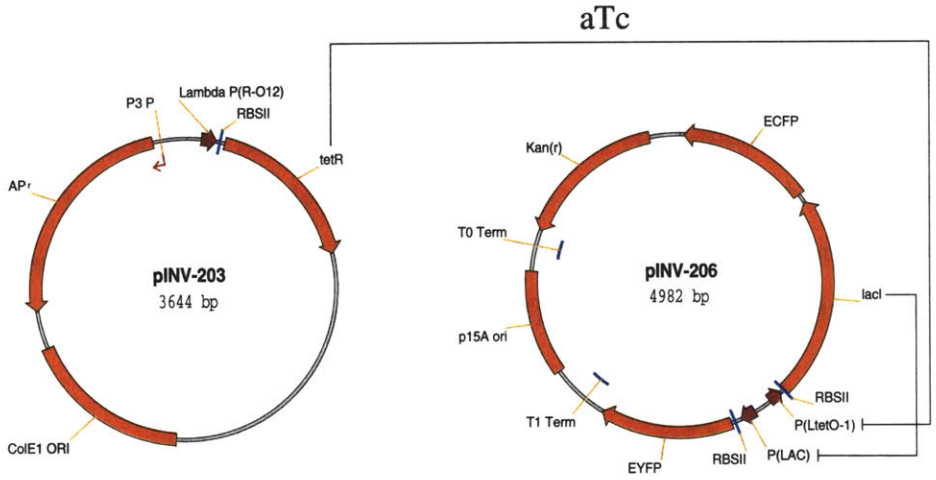


Figure 4-7: Genetic circuit to measure the transfer curve of the lacI/p(lac) inverter.

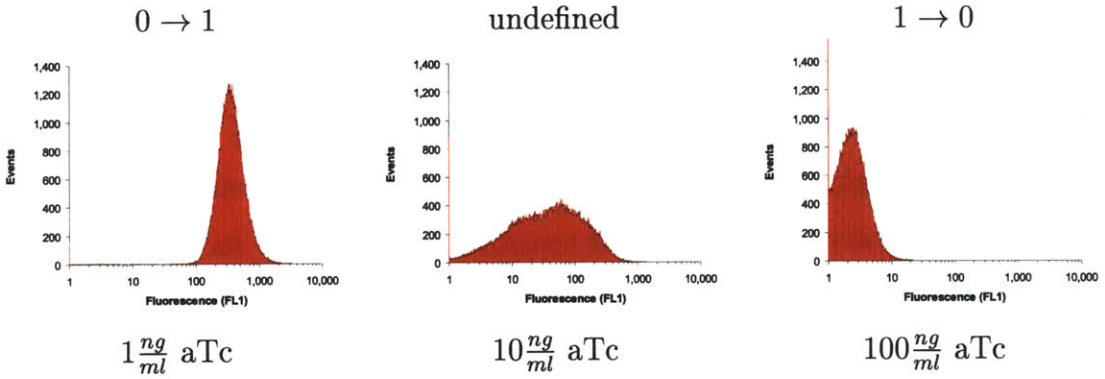
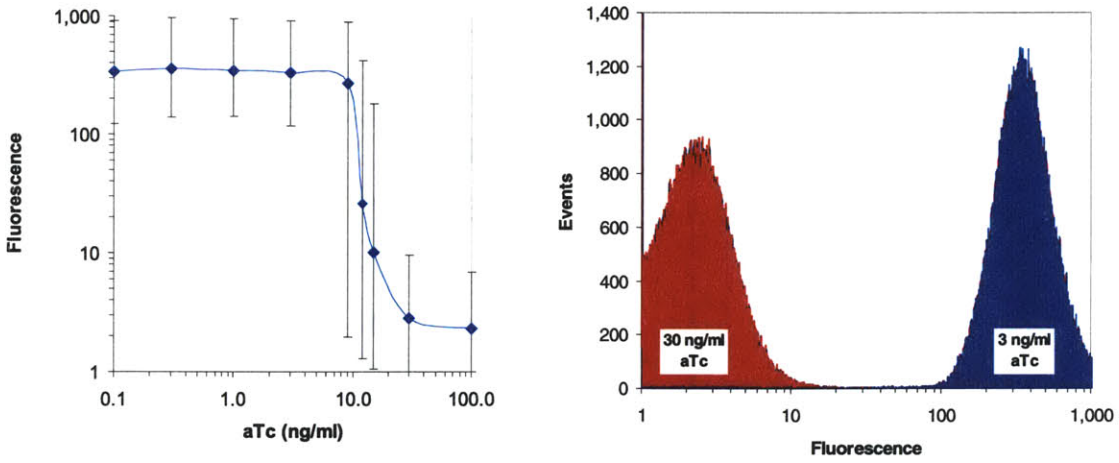


Figure 4-8: Transfer curve FACS data points for the lacI/p(lac) circuit, demonstrating inversion of the input signal.

input signal value of $10\frac{ng}{ml}$ aTc, the output values vary considerably among the cells. Therefore, in the same way that electrical logic circuits have forbidden zones, the output behavior is undefined for a region of intermediate input values.



(a) Gain / signal restoration (b) Noise margins

Figure 4-9: Transfer curve gain and noise margins for the lacI/p(lac) circuit.

Figure 4-9(a) illustrates the transfer function of the circuit with respect to the level of the inducer. The figure relates aTc input levels to EYFP output fluorescence levels, with error bars depicting the range that includes 95% of the flow cytometry fluorescence intensities recorded for that particular aTc level. The figure shows the range of input values with undefined output behavior, as well as the non-linear gain of the circuit. Overall, this *in-vivo* circuit exhibits significant gain and signal restoration.

Figure 4-9(b) illustrates the good noise margins between LOW and HIGH signal values. Input signal levels of $3\frac{ng}{ml}$ aTc are immediately before the sharp transition from HIGH to LOW output, while input signal levels of $30\frac{ng}{ml}$ aTc are correspondingly before the transition from LOW to HIGH output. As the figure shows, the output levels resulting from these two close input signals do not overlap. The favorable noise margins and significant gain of this circuit clearly demonstrate that digital-logic computation is feasible with genetic circuits.

By correlating ECFP and EYFP readings for the same experiment, Figure 4-10 shows the transfer curve of the lacI/p(lac) inverter. The ECFP fluorescence intensities are normalized to the EYFP levels according to the methodology described in Section 4.1.1 and based on the experimental results from Section 4.2. After nor-

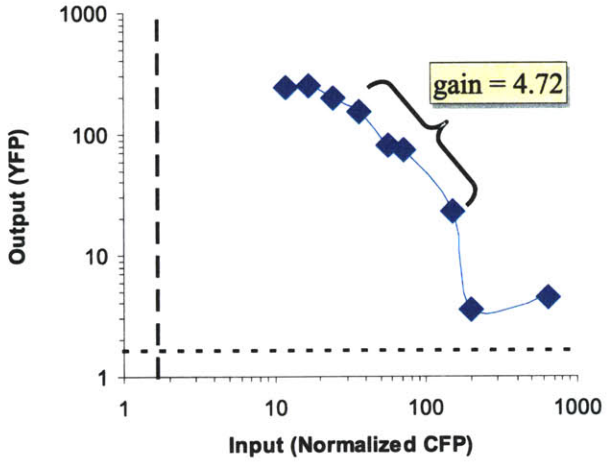


Figure 4-10: The transfer curve of the lacI/p(lac) inverter.

malization, each point represents the median fluorescence intensities for a particular experimental condition of the input signal (ECFP) versus the output signal (EYFP). The gain of the inverter of 4.72 is sufficient for digital-logic computation, and is likely to be related to the pentameric nature of *lacI* repression[56].

The lacI/p(lac) gate characterized in this section is the first component of the cellular gate library. Next, we describe the addition of the second component to the gate library, the $cI/\lambda_{P(R-O12)}$ inverter. In particular, the following section demonstrates genetic process engineering to modify the original behavior of this gate and obtain the desired behavior for digital computation.

4.4 Measuring and Modifying Transfer Curves of $cI/\lambda_{P(R-O12)}$ Inverters

While the gain exhibited by the lacI/p(lac) is sufficient, other repressor/promoter combinations can yield better signal restoration. One such example is the $cI/\lambda_{P(R-O12)}$ inverters. The *cI* repressor binds cooperatively to the λ O_R1 and O_R2 operators (Figure 4-11), which results in high gain according to the simulations in Section 3.3.1.

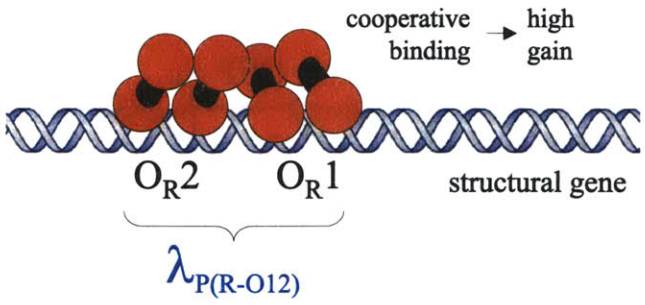


Figure 4-11: Cooperative binding of cI to $\lambda_{P(R-O12)}$ yields high gain.

The $\lambda_{P(R-O12)}$ is a synthetic promoter that I designed and that excludes O_R3 of the wild type λ promoter. The affinity of cI to O_R3 is weak and including this operator would not significantly enhance the repression efficiency of cI.

The cI monomer, also known as λ repressor, has an amino domain comprised of amino acids 1-92, a carboxyl domain of residues 132-236, and 40 remaining amino acids that connect the two domains[64]. The monomers associate to form dimers, which can then bind to the 17bp operator regions. cI's intrinsic affinity to O_R1 is about 10 times higher than to O_R2, and therefore typically binds O_R1 first. However, the binding of cI to O_R1 immediately increases the affinity of a second dimer to O_R2 because of the interaction with the previously bound dimer. As a result, repressor dimers bind to O_R1 and O_R2 almost simultaneously. From a circuit engineering perspective, this cooperative binding leads to a much desired high gain since the transition from low repression activity to high repression activity occurs over a small range of repressor concentrations.

The genetic circuit to measure the device physics of the $cI/\lambda_{P(R-O12)}$ is logically similar to the one used to measure the $lacI$/p(lac) inverter (Figure 4-12). Here, the $lacI$/p(lac) IMPLIES gate controls the level of the cI input. For a particular experiment, the researcher sets the repressor level by controlling the IPTG concentration.

The logic interconnect of this circuit should result in EYFP fluorescence intensities that are inversely correlated with the IPTG input levels. However, as shown in Figure 4-13, the circuit is completely unresponsive to variations in the IPTG levels.

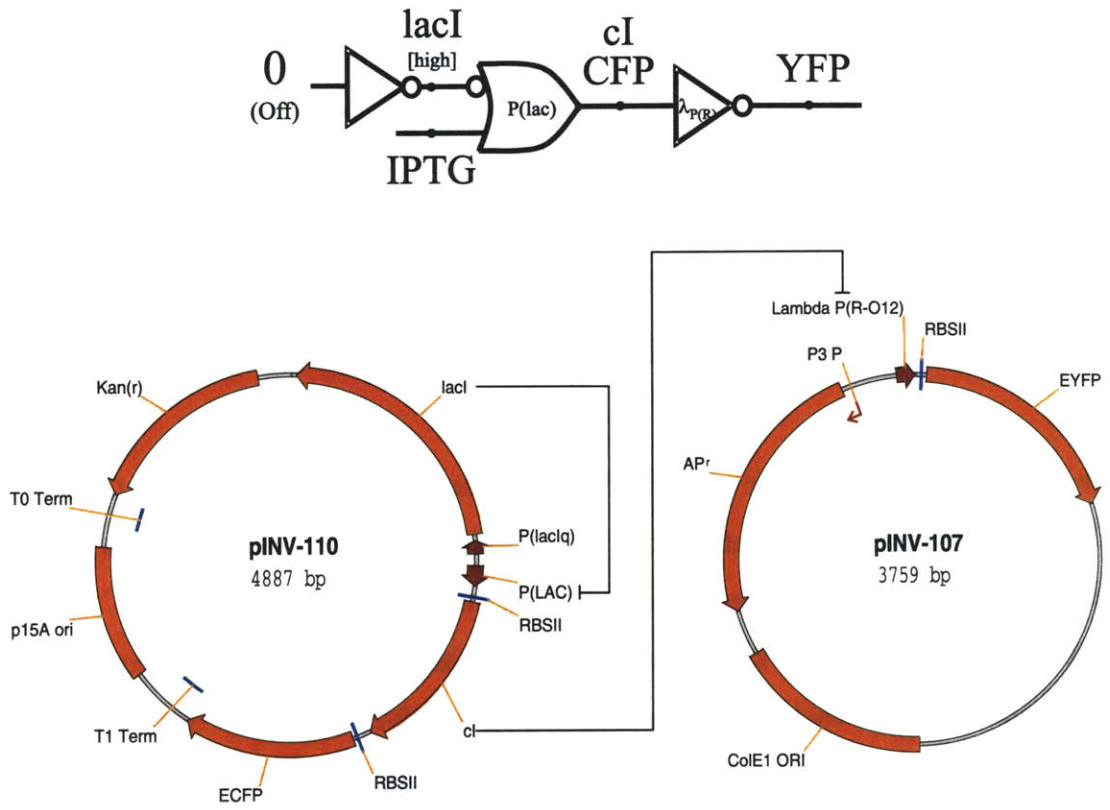


Figure 4-12: Genetic circuit to measure the transfer curve of $cI/\lambda_{P(R-O12)}$, using *lacI* as driver.

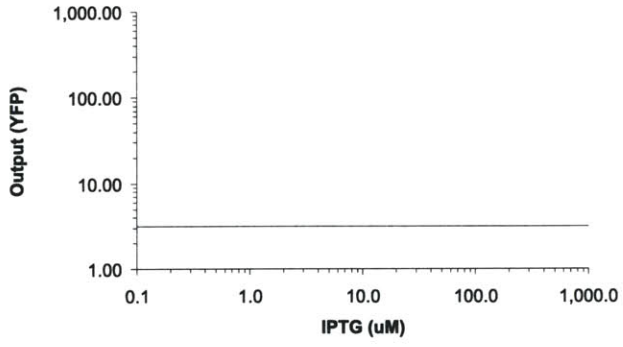
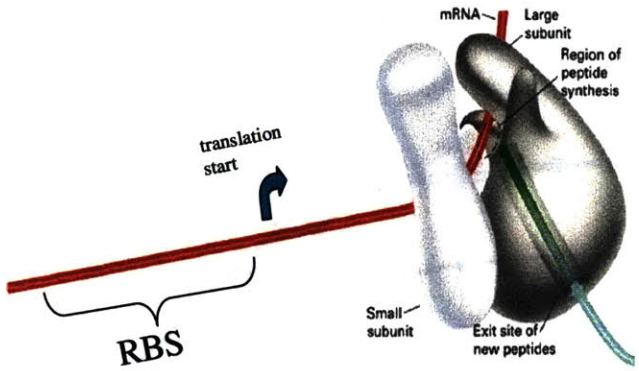


Figure 4-13: The unresponsive transfer curve of the original $cI/\lambda_{P(R-O12)}$ inverter.

The lack of response stems from the mismatch between the kinetic characteristics of the *lacI*/p(lac) gate versus the $cI/\lambda_{P(R-O12)}$ inverter. Specifically, with no IPTG, the




```
Orig:  ATTAAAGAGGAGAAATTAAGCATG   strong
RBS-1:  TCACACAGGAAACCGGTTCGATG     |
RBS-2:   TCACACAGGAAAGGCCTCGATG     |
RBS-3:    TCACACAGGACGGCCGGATG     weak
```

Figure 4-14: The Ribosome, mRNA, and different Ribosome Binding Sites.

fully repressed expression level from p(lac) still results in a low level of *cI* mRNA. Because the ribosome binding site is very efficient, the low mRNA level results in some translation of the *cI* protein. And because *cI* is a highly efficient repressor, even a low concentration represses the $\lambda_{P(R-O12)}$ promoter to the point where no fluorescence can be detected. This gate mismatch highlights the importance of understanding the device physics of the cellular gates. The following two sections describe genetic process engineering to modify genetic elements in the $cI/\lambda_{P(R-O12)}$ inverter such that the gate obtains the desired behavioral characteristics.

4.4.1 Modifying Ribosome Binding Sites

Ribosome Binding Site (RBS) sequences significantly control the rate of translation from the input mRNA signal to the input protein. These sequences align the ribosome onto the mRNA in the proper reading frame so that polypeptide synthesis can start correctly at the AUG initiation codon. The affinity of the ribosome's 30S subunit to the RBS that it binds determines the rate of translation. This translation rate, k_{xlate}, is part of the biochemical reaction 3.1 for modeling and simulating the inverter. For a given input mRNA level, a reduction in k_{xlate} yields a lower input protein level, which

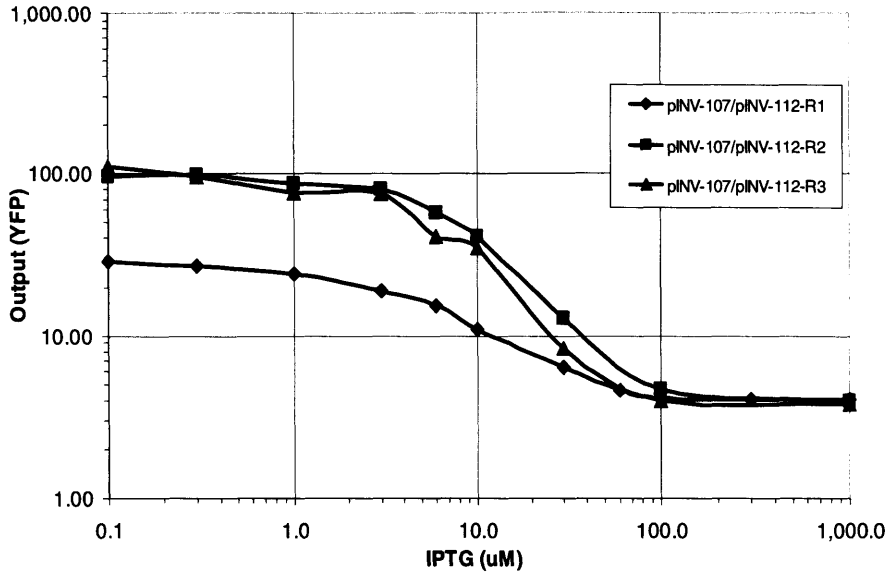


Figure 4-15: The effect of weaker RBS's on the behavior of the $cI/\lambda_{P(R-O12)}$ circuit. Here the output YFP exhibits the desired inverse sigmoidal relationship to the IPTG input.

in turn pushes the entire transfer curve upward and outward (Figure 3-9(c)). The motivation to perform this particular genetic modification stems from circuit design and analysis using BioSPICE simulations.

Figure 4-14 shows the original highly efficient *cI* RBS used in the circuit above, as well as three other less efficient RBS's from the literature[29]. Starting from pINV-110, I constructed three new plasmids (pINV-112-R1, pINV-112-R2, pINV-112-R3) where the three weaker RBS's replace the original RBS of the *cI*. pINV-112-R1 contains the strongest RBS, while pINV-112-R3 contains the weakest RBS.

Figure 4-15 shows the dramatic effect of the RBS change on the behavior of the circuit, where now the output YFP exhibits the desired inverse sigmoidal relationship to the IPTG input. The circuit with the strongest RBS (pINV-107/pINV-112-R1) shows a moderate sensitivity to IPTG, while the circuits with the other two RBS's display a more pronounced response to variations in IPTG.

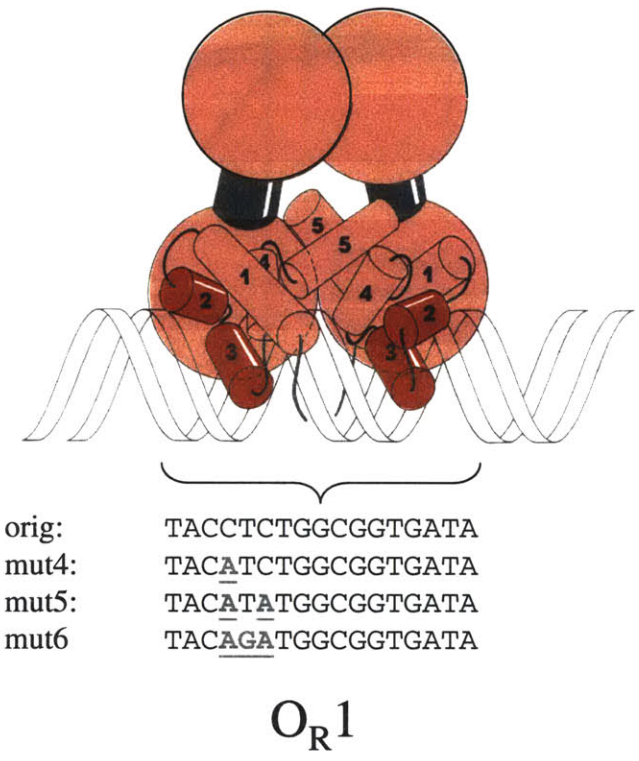


Figure 4-16: The *cI* repressor dimer, its α-helix motifs, and mutations to OR1.

4.4.2 Modifying Repressor/Operator Affinity

Replacing the strong ribosome binding site with weaker sites converted a non-functional circuit into a functional one, and demonstrated the utility of genetic process engineering. This section describes further modifications to the repressor/operator affinity that yield additional improvements in the performance of the circuit. These modifications are motivated by the BioSPICE simulations of Figures 3-9(a) and 3-10. The simulations show how reductions in k_{rprs}, the binding affinity of the repressor to the operator, reshape the transfer curve of the inverter upward and outward.

To reduce the repressor/operator affinity, I constructed three new plasmids with modified O_R1 sequences using site-directed mutagenesis (Figure 4-16). *cI*'s amino domain is folded into five successive stretches of α-helix, where α-helix 3 lies exposed along the surface of the molecule[64]. This α-helix recognizes the λ operators and binds the repressor to those particular DNA sequences. The two α-helix 3 motifs of the

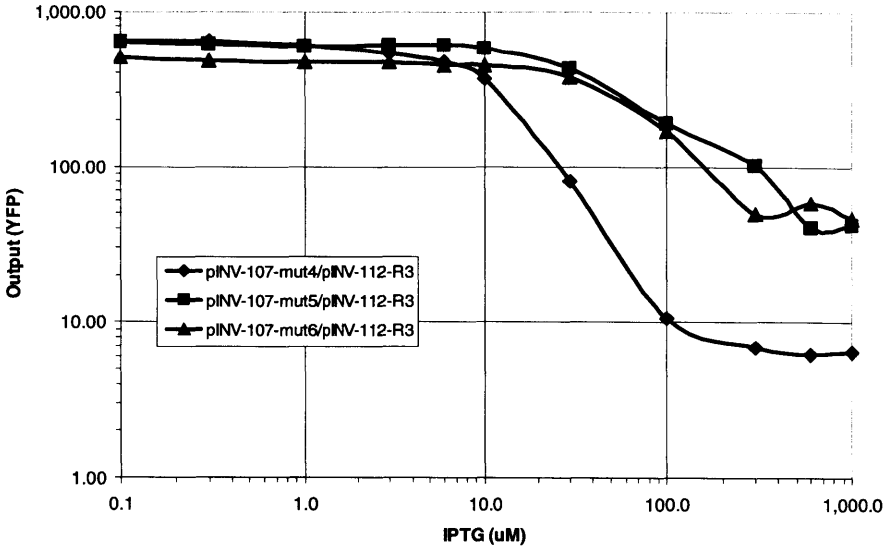


Figure 4-17: The effect of $\lambda_{P(R-O12)}$ O_R1 operator mutations on the behavior of the $cI/\lambda_{P(R-O12)}$ circuit

repressor's dimer complex are separated by the same distance as the one separating successive segments of the major groove along one face of the DNA. These motifs efficiently bind the repressor dimer to the mostly symmetric λ operator regions, where each operator consists of two half-sites. The following is the consensus sequence for the twelve operator half-sites in the wild-type Bacteriophage λ(subscripts correspond to the frequency of the base pair in the given position):

T_9	A_{12}	T_6	C_{12}	A_9	C_{11}	C_7	G_9	C_5
C_2		C_3		T_2	T_1	T_4	T_2	T_1
A_1		A_1		C_1		G_1	C_1	

In determining which mutations to perform, I conjectured that bases with high frequency in the consensus sequence would be significant to strong repressor/operator binding. mut4 is a one base pair mutation $C \rightarrow A$ of the fourth O_R1 position. mut5 is a two base pair mutation that also modifies the sixth O_R1 position $C \rightarrow A$, and mut6 is a three base mutation that also modifies the fifth O_R1 position $T \rightarrow G$.

The experimental results in Figure 4-17 demonstrate the effect of coupling the three $\lambda_{P(R-O12)}$ O_R1 operator mutations with the weakest ribosome binding site from

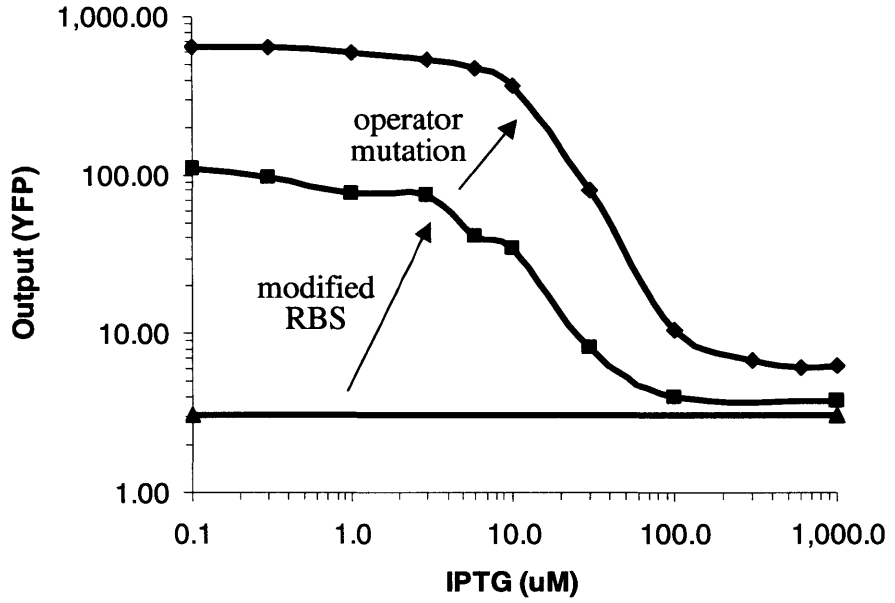


Figure 4-18: Genetic process engineering of the $cI/\lambda_{P(R-O12)}$ inverter. A series of genetic modifications converts a non-functional circuit into one that achieves the desired input/output behavior.

above. The two and three base pair O_R1 mutations, coupled with the weak RBS, produce a circuit where the highest levels of cI cannot repress the output of the $cI/\lambda_{P(R-O12)}$ gate. However, a one base pair mutation to O_R1 in plamids pINV-107-mut4/pINV-112-R3 yields a circuit with a well-behaved response to the IPTG signal, and is a good gate candidate for other biocircuits.

In summary, using genetic process engineering I first examined the behavioral characteristics of the $cI/\lambda_{P(R-O12)}$ inverter, and then genetically modified the gate until I produced a version with the desired inverse sigmoidal behavior. The design and experimental results illustrate how to convert a non-functional circuit with mismatched gates into a circuit that achieves the correct response (Figure 4-18). Note that in choosing genetic variations, the specific ribosome binding site modifications in Section 4.4.1 can be applied to any inverter and are independent from the specific genetic candidate. However, the operator mutations in Section 4.4.2 are specific to the $\lambda_{P(R)}$ and cI and cannot be generally applied to other operators. Accordingly, there are several strategies to optimizing a new component. First, one can test

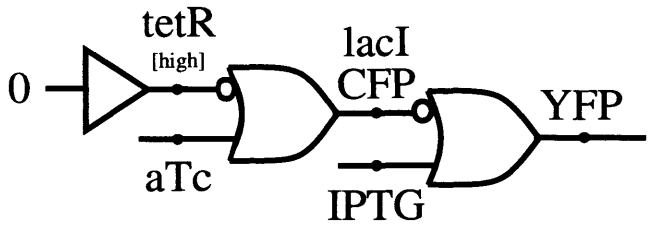


Figure 4-19: Logic circuit for measuring the lacI/p(lac) IMPLIES gate. aTc controls the *lacI* input, while the IPTG input is controlled directly. Yellow fluorescence reports the output signal.

modifications that are applicable to any component. Second, one can study the particular component by reading the literature and performing laboratory experiments, and choose mutations based on the understanding of the specific biochemistry of the element. Third, one can perform large-scale random mutations on the element, and screen for mutants that have the desired behavior.

4.5 Characterizing the lacI/p(lac) IMPLIES Gate

The *lacI*/p(lac) inverter gate, whose transfer curve is measured in Section 4.3, also implements the IMPLIES operation (Figure 4-19). A HIGH IPTG input always yields a HIGH output, while a HIGH aTc input in the absence of IPTG yields a LOW output. Figure 4-20 illustrates the response of the circuit to five different levels of IPTG simultaneously coupled with five different levels of aTc. The circuit behaves correctly with a HIGH output unless aTc is HIGH and IPTG is LOW.

The device physics measurements in this chapter facilitate biocircuit design because they enable prediction of the behavior of complex circuits using the characteristics of simple components, as described in Section 3.3.4. With the appropriate tools, the engineer of biocircuits can begin to design and produce large-scale circuits. However, there are limitations to the complexity of a circuit that one can implement using these biochemical mechanisms. Next, we turn our attention to this issue.

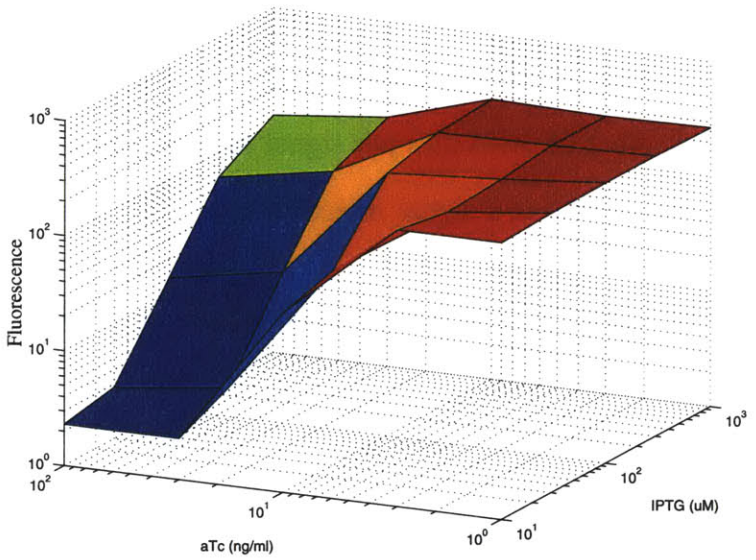


Figure 4-20: The circuit response of the lacI/p(lac) IMPLIES gate, mapping the IPTG and aTc inputs to the resulting fluorescence. The graph shows fluorescence intensities under twenty-five separate conditions with five different IPTG levels and five different aTc levels. The circuit behaves correctly with a LOW output only when aTc is HIGH and IPTG LOW.

4.6 Physical Constraints in Building Circuits

Certain physical constraints, such as volume and metabolic requirements, limit circuit complexity in a single cell. There are approximately 10^7 protein molecules per cell. If one assumes that to represent an individual signal in a cell requires roughly one hundred to five hundred molecules, then one thousand signals require an additional $1\%-5\%$ of the original number of protein molecules in a single cell. Given that the size of the signal protein molecules is average with respect to the other protein molecules in the cell, the additional volume required for a biocircuit with one thousand gates should not significantly affect the health of the engineered cells.

The metabolic requirements of operating a circuit depend on the size of the circuit, the number of molecules needed to represent a signal, and the rate at which the cell

needs to synthesize the signal molecules. Let s, m, and d be defined as follows:

$$\begin{aligned} s &= \text{number of signals in the circuit} \\ m &= \text{number of molecules per signal to represent a "1"} \\ d &= \text{decay rate of a signal molecule (i.e. } \frac{1}{2} \text{ life in minutes)} \end{aligned}$$

Then, the protein synthesis rate to sustain all signals at "1" is

$$\frac{s \cdot m}{2d} \frac{\text{mol}}{\text{min}}$$

Since *E. coli* cells multiply roughly every 30 minutes, each cell must synthesize at least 10^7 protein molecules in 30 minutes. The following table shows the effect of various parameter values on the synthesis rate required to operate the circuit in the cell:

			protein synthesis (per min)	
s	m	d	# of molecules	% of total
100	100	100	250	0.08%
500	250	10	6,250	1.88%
1000	500	2	125,000	37.50%

The table shows that a circuit with one thousand gates, where each gate requires five hundred protein molecules, and the half life of each protein is two minutes, requires the cell to synthesize proteins at an additional forty percent rate in order to sustain the engineered computation. This places a rather high metabolic strain on the cell, and a circuit with these characteristics is unlikely to be operational in a single cell. To engineer behavior with higher complexity in these cells, one must enlist the coordinated effort of multiple cells. This analysis provides motivation for the next chapter, where we discuss engineered cell-to-cell communications to enable coordinated behavior in cell aggregates. The work derives inspiration from the biological developmental process.

Chapter 5

Intercellular Communications

The biological development process requires coordinated, robust action among a very large number of essentially identical, unreliable components. In stark contrast to current computer science engineering practice, these developmental programs are highly fault tolerant. Imagine what would happen if any biological mechanism exhibited the same fragility as a modern microprocessor, operating system, or satellite.

Previous work in my group [1, 14, 15, 58, 65, 89, 90] has examined some of these robustness and pattern formation issues in simulation, with intriguing results. We found that the topic we call *amorphous computing* requires a different set of algorithms and a different approach to thinking about structures than conventional computer science.

However, we also must better understand the developmental process in a biological context. Although we are making significant progress, we simply do not fully understand the pattern formation of even the simplest of biological structures. But surely concepts from computer science, such as subroutines, divide-and-conquer, recursion and iteration will play a major role in understanding the genetic control of developmental diversity. Both biology and computer science have lessons to learn from a cooperative investigation of this field.

Even simple biological systems can exhibit complex developmental processes. The motile, gram-negative bacterium *Myxococcus xanthus*, for example, exhibits social behavior and cellular differentiation during cooperative feeding. The controlled, density

dependent, release of antibiotics and cell wall degrading enzymes to kill competitors allows moving swarms (so-called "wolf-packs") to act more effectively than individual cells [19]. Similarly, *M. xanthus* exhibits selection, during starvation, of a small number of cells out of a swarm of 100,000 to change form from rod-like bacteria to environmentally protected spherical myxospores. Spore formation requires high cell density, nutrient limitation, and a solid surface [20, 46].

In this chapter, I demonstrate a *biological implementation* of a key component in building such developmental pattern engineering techniques – cell-to-cell communications. Communication between cells is obviously essential to any kind of coordinated expression. However, in development and in the amorphous computing simulations, one kind of communication emerges as especially important – the ability to detect and act on chemical signal concentration gradients. Such gradient dependent expression is the building block of locally unique behavior, as well as the organizing principle that allows the construction of local coordinate systems through the creation and detection of chemical gradients. Such trophic behavior provides one basic organizing principle for complex patterned development.

I have isolated a specific chemical cell-to-cell signaling mechanism from a natural biological system, the quorum sensing system of *Vibrio fischeri.* This system encodes genes and promoter sequences that allow the controlled expression of the chemical *Vibrio fischeri* autoinducer (VAI) within one sender cell, and the detection and controlled expression of specific genes in another, receiving cell. The free diffusion of the VAI chemical within the medium and across cell membranes allows the establishment of chemical gradients and the controlled expression of genetic circuits as a result.

Specifically, I demonstrate the construction and testing of engineered genetic circuits which exhibit the ability to send a controlled signal from one cell, diffuse that signal through the intercellular medium, receive that signal within an a second cell, and activate a remote transcriptional response (Figure 5-1). The work reported in this Chapter implements and characterizes cell-to-cell communication components for the cellular gate library.

In the remainder of this chapter, I describe the mechanism of quorum sensing

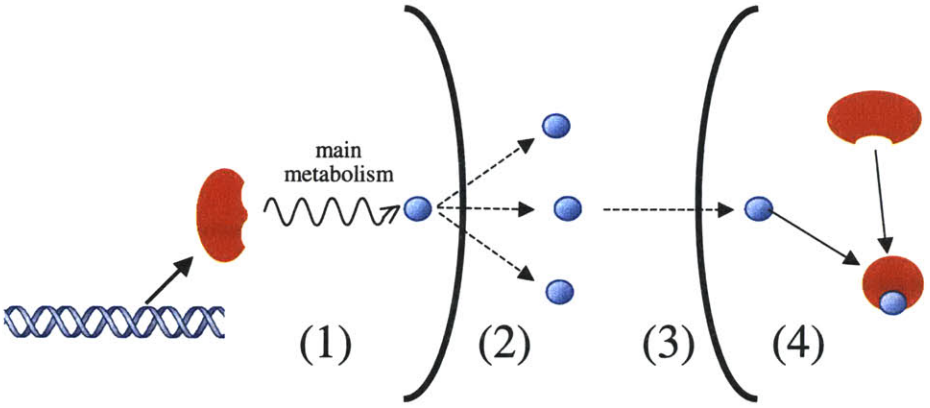


Figure 5-1: Cell-to-cell communication schematics: (1) The sender cell produces small signal molecules using certain metabolic pathways. (2) The small molecules diffuse outside the membrane and into the environment. (3) The signals then diffuse into neighboring cells (4) and interact with proteins in the receiver cells, and thereby change signal values.

in bacteria (Sections 5.1-5.1.1), present the plasmids engineered for communications, and report on experimental results (Section 5.2).

5.1 Quorum Sensing in bacteria

Vibrio fischeri is a gram-negative bioluminescent marine prokaryote that naturally occurs in two distinct environments. In seawater, it swims freely at concentrations of approximately ten cells per liter. It also grows naturally in a symbiotic relationship with a variety of invertebrate and vertebrate sea organisms, especially the Hawaiian sepiolid squid, *Euprymna scolopes* and the Japanese pinecone fish, *Monocentris japonica* [69]. In these symbiotic relationships, the bacteria grows to densities of approximately 10^{10} cells per liter.

In the free-living state, *Vibrio fischeri* emits essentially no light ($<$ 0.8 photons/second/cell). In the light organ of the Hawaiian Sepiolid Squid, however, the same bacteria emit more than 800 photons/second/cell, producing very visible bioluminescence. In culture, *Vibrio fischeri* demonstrates a similar density dependent bioluminescence, with induction occurring at about 10^{10} cells/liter.

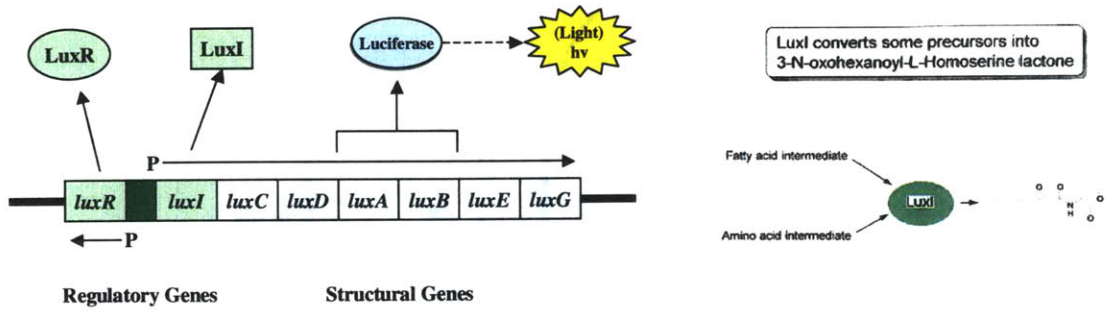


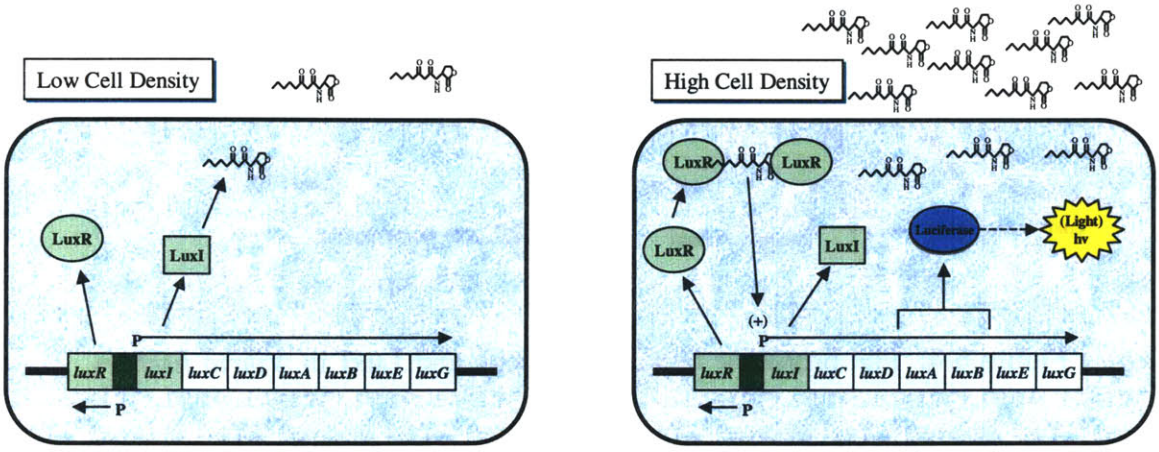
Figure 5-2: The lux Operon (on the left), and luxI metabolism that catalyzes the formation of VAI (on the right).

Work over many years has established that this behavioral change is due to a natural mechanism of detecting cell densities, which has been termed quorum sensing [28]. The quorum sensing mechanism relies on the synthesis and detection of a very specific, species unique chemical, an *autoinducer*, which mediates intercellular communications. In *Vibrio fischeri*, this autoinducer chemical (VAI) has been identified as N-(3-oxohexanoyl)-3-amino-dihydro-2-(3H)-furanone [21]. The gene, LuxI, catalytic protein, and synthetic pathway for this chemical have also been identified [27].

Briefly, the LuxI gene encodes an acyl-homoserine lactone synthesase that uses highly available metabolic precursors found within most gram-negative prokaryotic bacteria – acyl-ACP from the fatty acid metabolic cycle, and S-adenosylmethionine (SAM) from the methionine pathway – to synthesize VAI.

The *Vibrio fischeri* autoinducer (VAI) freely diffuses across the bacterial cell membrane. Thus, at low cell densities, low VAI concentrations are available. Within a light organ, or at high culture densities, VAI builds up within the environment, resulting in a density dependent induction of bioluminescence.

The response mechanism to VAI concentration has also been extensively analyzed [77]. Briefly, the LuxR gene codes for a two domain DNA binding protein that interacts with VAI and the Lux box of the LuxICDABEG operon promoter to exercise transcriptional control (Figure 5-2). At nanomolar concentrations, VAI binds to the N terminal domain of the LuxR protein, which in turn activates the C-terminal helix-



free living, 10 cells/liter
< 0.8 photons/second/cell

symbiotic, 10^{10} cells/liter
800 photons/second/cell

Figure 5-3: The lux Operon and quorum sensing: density dependent bioluminescence.

turn-helix DNA binding domain. The LuxR protein acts as a transcriptional activator for the RNA polymerase holoenzyme complex. The activated protein likely binds in dimeric or multimeric forms, because of the evident dyadic symmetry of the Lux box binding domain. Figure 5-3 shows the genetics of bioluminescence in two distinct environments in which *Vibrio fischeri* naturally occurs.

Successful cloning and expression of the Lux genes into *E. coli* have established the genetic structure of the *Vibrio fischeri* Lux operon[26]. It is somewhat surprising (although common) for the transfer of regulatory genes and entire metabolic pathways to function straightforwardly across gram-negative species boundaries in this way.

5.1.1 Genetic Features of the LuxR/LuxI operons

The nucleotide structure of the sequenced regulatory region is shown in Figure 5-4. This region encodes two divergently transcribed promoters. The left operon constitutively expresses the LuxR transcript as coded by the left ORF. This operon has a standard σ^{70} binding region, consisting of a -10 and -35 sequence, and a CRP/CAMP binding site. The CRP/CAMP binding site allows catabolic repression on the left LuxR operon.

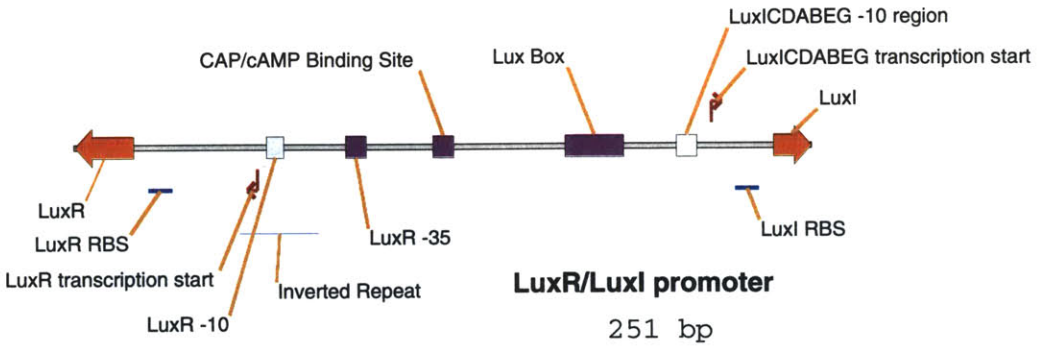


Figure 5-4: The promoter regions for LuxR and LuxI in *Vibrio fischeri.*

The right operon drives expression of the LuxICDABEG transcript, coding for autoinducer production (LuxI) and the bioluminescence cassette of LuxCDABEG. The operon consists of a standard -10 σ^{70} binding site, but is missing the -35 site. Instead, the *lux box*, a 20 base inverted palindromic repeat, allows dimeric binding of the active form of LuxR binding protein, activating the RNA polymerase holoenzyme complex, under control of the LuxR protein – and hence indirectly, the VAI concentration.

The lux box is a common motif in regulatory proteins of the LuxR family, and occurs upstream of many LuxR homologous genes. The sequence of the Lux box in this construct is 5'(ACCTGTAGGATCGTACAGGT); the consensus sequence for similar lux boxes in other constructs is [36] 5'(RNSTGYAXGATNXTRCASRT)3' (N = A, T, G, C; X = N or gap; S = G, C; R = A, G; Y = C, T).

Note that the dimeric binding of the LuxR product produces the kind of nonlinear concentration/response behavior discussed in [45, 89] and widely seen in DNA binding protein transcriptional control. This nonlinear response is an essential element of signal restoration and digital control of expression.

The transcription of the right operon also enhances the production of LuxI, and thus the VAI synthesase, and VAI. We see here the key component of a Schmidt-trigger positive feedback gate – once transcription is turned on, the enhancement is self-reinforcing, leading to hysteresis in the transfer curve.

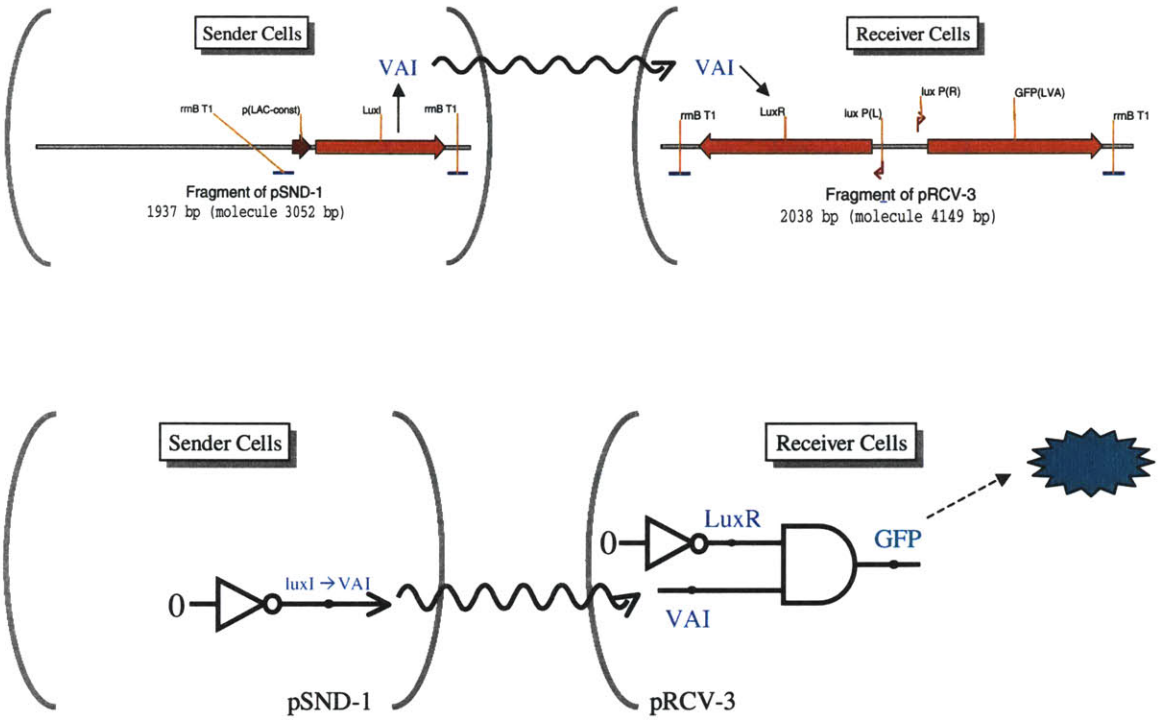


Figure 5-5: Genetic and logic circuits for pSND-1 sender and pRCV-3 receiver. The senders cell constitutively express *luxI*, which catalyzes the formation of VAI. VAI diffused into the environment and neighboring cells, which detect VAI through the transcriptional activation of lux P(R).

5.2 Intercellular Signaling Experiments

In order to experiment with engineered cell-to-cell communications, I constructed a series of plasmids, as described in Appendix B. The plasmids encode genetic logic circuits that enable cells to send messages, and logic circuits that enable cells to detect and respond to incoming messages. The following sections describe experiments to characterize the cell-to-cell communication capabilities engineered into *E. coli* cells using these genetic circuits.

5.2.1 Sending a constant cell to cell signal

The first intercellular communications experiment involved sending a constant signal from one cell type to another. The pSND-1 plasmid encodes a circuit that directs the

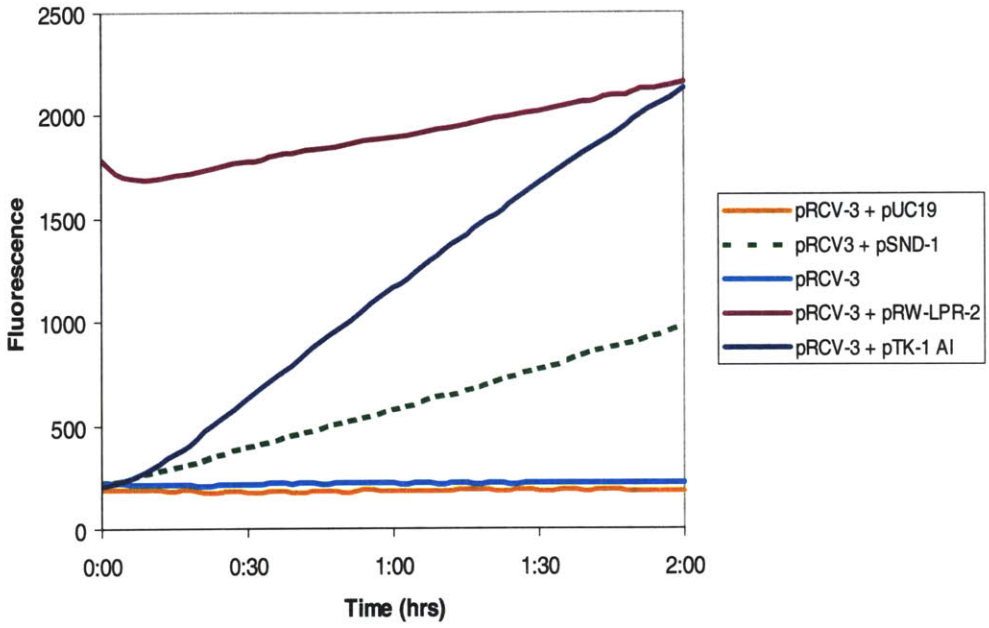


Figure 5-6: Time-series response of receivers to signal. The negative controls (pRCV-3 and pRCV-3) show no increase in fluorescence over time. The positive control with constitutive expression of GFP(LVA) (pRW-LPR-2) shows a high level of fluorescence. The mixed culture of senders and receivers (pRCV-3 + pSND-1) shows the increase in fluorescence over time due to the cell-to-cell communications. Finally, a highly concentrated extract of VAI mixed with the receivers (pRCV-3 + pTK-1 AI) shows a rapid increase in fluorescence.

cell to continuously send the VAI message. The pRCV-3 plasmid encodes a circuit that directs the cell to express GFP(LVA), a variant of the green fluorescent protein from Clontech, when VAI enters the cell (Figure 5-5). Cultures of *E. coli DH5α* transformed with the pRCV-3 plasmid and cultures of *E. coli DH5α* transformed with the pSND-1 plasmid were grown separately overnight @37°C in LB AMP. A 96-well clear bottom plate was loaded with 200μl of LB AMP in each well. 10μl of pSND-1 cells were loaded horizontally to each well, along with controls consisting of cells expressing GFP(LVA) constitutively with the pRW-LPR-2 plasmid, *E. coli DH5α* containing pUC19 to serve as a negative control, and a series of wells containing extracted VAI (see below).

Vertically, 10μl of cells containing the pRCV-3 construct were also loaded into each well. Thus, each well contained a variety of senders, and a uniform set of

receivers. The plate was grown in a Biotek FL-600 fluorescent plate reader for two hours, and fluorescence at the GFP(LVA) peak (excitation filter 485/20 nm, emission filter 516/20 nm) was measured every two minutes. Figure 5-6 shows the time-series response of the different cultures. Wells containing only the pRCV-3 cells, or with added pUC19 cells, showed no increase in fluorescence. The well containing pRCV-3 cells and pRW-LPR-2 cells (which express GFP(LVA)) served as a positive control for high levels of fluorescence. Wells containing the pRCV-3 cells plus extracted pTK1 autoinducer showed high, and increasing levels of fluorescence. Cells with pRCV-3 and pSND-1 showed the expected increase in fluorescence demonstrating successful cell-to-cell signaling.

5.2.2 Characterization of the receiver module

The genetic circuit to receive messages (plasmid pRCV-3) was further characterized by inducing the promoter with different levels of VAI extracted from cell culture. Cultures of *Vibrio fischeri* and of *E. coli* containing the pTK1 plasmid were grown overnight to stationary phase in GVM broth or LB AMP respectively @30°C which allows evaluation of their bioluminescence. After verification of light production, 100 ml of the cultures were centrifuged at 3300 g, and the supernatant collected. The supernatant was extracted with 10 ml of ethyl acetate by vigorous shaking in a separatory funnel for 10 minutes. The ethyl acetate extract (upper fraction) was separated and dried under vacuum. The resulting crude extract was redissolved in 1ml of DI water to provide 100x VAI extract.

I analyzed the effectiveness of serial dilutions of the VAI extracts from pTK1 and *Vibrio fischeri* in inducing GFP expression of the pRCV-3 cells. Both the *Vibrio fischeri* and pTK1 extracts were about equally effective at inducing expression of the pRCV-3 promoter, as measured by GFP production. Cells with different levels of VAI were incubated @37°C for four hours, and the maximum fluorescence achieved for each culture was recorded. Figure 5-7 shows that increasing levels of autoinducer yielded increasing GFP expression by the receiver. High levels of the extract, however, were toxic to the cells, and resulted in relatively low fluorescence levels.

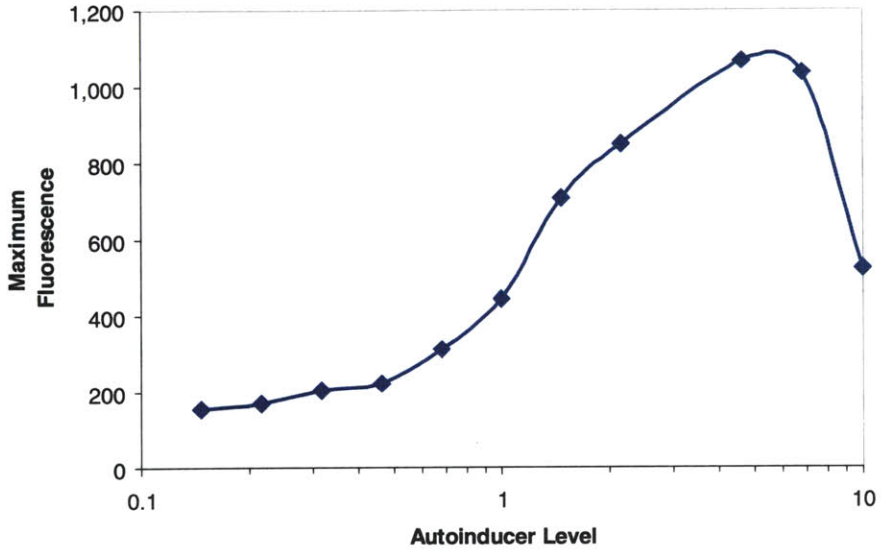


Figure 5-7: The effect of different autoinducer levels on the maximum fluorescence attained by the receivers.

5.2.3 Sending controlled cell-to-cell signals

The third experiment characterized the response of the receivers to variations in the strength of the message transmitted by the senders. For this, the LuxI gene was placed under the control of the Tet promoter from the Clontech pPROTet system. Figure 5-8 provides a schematic representation of the experiment. In one cell, the pLuxI-Tet-8 plasmid exerts controlled expression of the LuxI autoinducer synthesase using the Tet operon. The synthesase catalyzes the conversion of normal cellular metabolic products into VAI; thus, controlling the LuxI expression level controls the VAI production in the cells. The VAI produced within the cells migrates though the cell membrane of the sender, into the culture medium, and through the membrane of the receiver – a cell containing the pRCV-3 plasmid. There, it interacts with the N-terminal domain of the LuxR DNA binding protein product, disabling it from binding to the lux box binding site. The expression of the GFP reporter gene is enhanced, resulting in high levels of fluorescence.

The experiment involved the incubation of similar mixed cell cultures on 96-well clear bottom plates. One important difference was the culture medium – the pPRO-

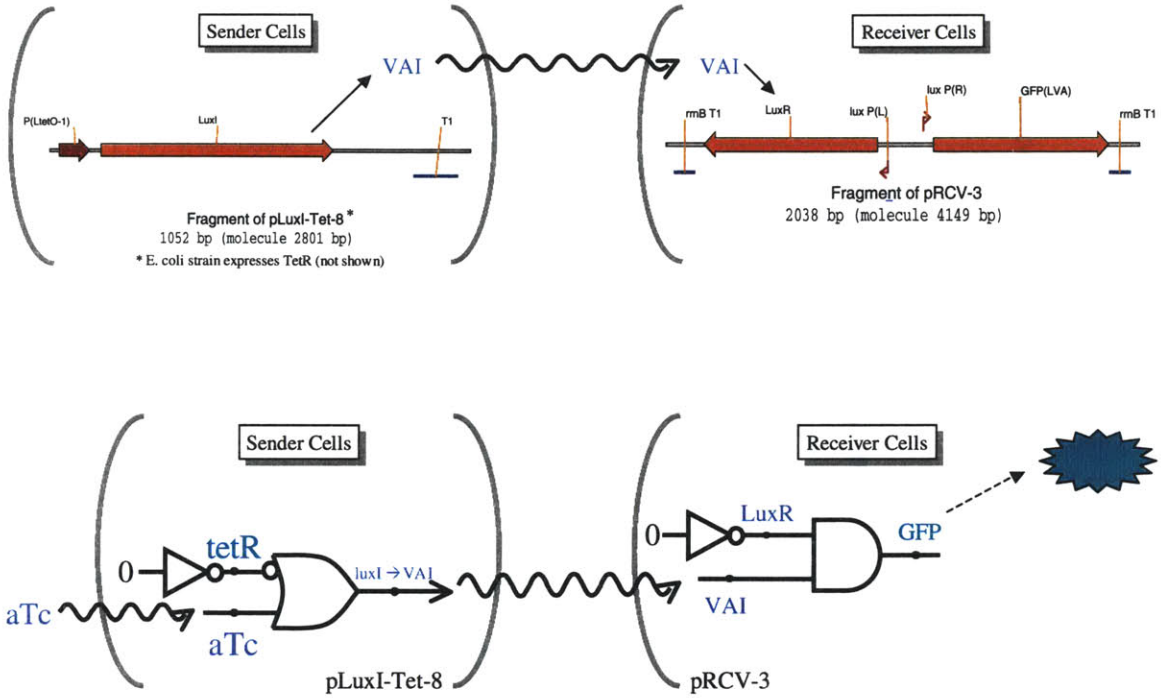


Figure 5-8: Genetic and logic circuits for pLuxI-Tet-8 sender and pRCV-3 receiver.

TET cells carry Spectinomycin and Chloramphenicol resistance, while the pRCV-3 cells carry Ampicillin resistance. The experiments were carried out by growing overnight cultures of both types of cells in the appropriate antibiotic containing medium, followed by centrifugation at 4000g to remove the medium, and resuspension to similar cell density in LB containing no antibiotics, so that both cell types could grow.

Three rows of pRCV-3 cells were loaded on a microplate, and two of these rows were also loaded with pLuxI-Tet-8 cells. The sender cells in the various columns were induced with different levels of aTc. Also, one control column included receivers that were induced directly with the VAI extract. Figure 5-9 shows the results of this experiment after culturing the plate for four hours @37°C. As expected, the null wells containing no aTc or VAI showed no enhancement of fluorescence, while the positive control wells with the 10x VAI extract exhibited fluorescence. The experiments labeled BL21-LuxITet include senders where the pLuxI-Tet-8 plasmid was transformed into BL21-PRO cells, while the experiments labeled DH5a-LuxITet include senders

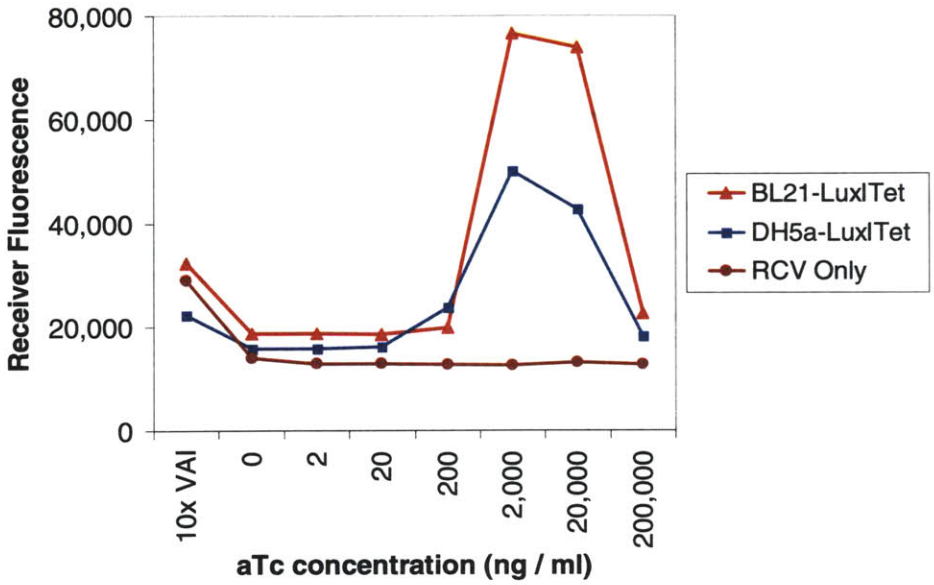


Figure 5-9: Controlling the sender's signal strength: maximum response of receivers to different aTc induction of senders. The 10X VAI positive control column shows the results of introducing the VAI extract into the wells. The rest of the columns illustrate the response of the receivers from induction of the senders with various levels of aTc.

where the pLuxI-Tet-8 plasmid was transformed into *E. coli DH5α* cells.[1] In wells containing sender cells induced with aTc at levels below about $20ng/ml$, the receiver cells exhibit only only a small fluorescent response. In wells induced with aTc levels above $200ng/ml$, the receiver cells exhibit a significant response. Sufficiently high levels of aTc inhibited cell growth.

5.2.4 Visual Observation of Communications

Finally, this section describes three visual observation experiments of cell-to-cell communications in order to understand the diffusion and reaction characteristics. First, pINV-112-R3 and pSND-1 were transformed into *E. coli* JM2.300 cells to create sender cells. With IPTG induction, these sender cells emit cyan fluorescence that serves as an easily identifiable marker due to the ECFP encoded downstream of p(lac) on pINV-112-R3. In addition, the pSND-1 plasmid directs the cells to constitutively express

[1] In BL21-PRO cells, TetR (needed for controlled induction of the Tet promoter) exists on a plasmid, while in *DH5α* TetR is part of the chromosomal DNA.

luxI, resulting in constant transmission of the message.

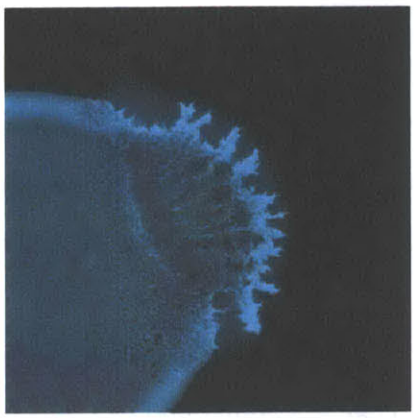
Second, pRCV-3 and pPROLar.A122 were transformed into *E. coli* JM2.300 cells to create receiver cells. The pRCV-3 plasmid instructs the cell to express GFP(LVA) when VAI enters the cytoplasm. The pPROLar.A122 plasmid confers Kanamycin resistance to the cell. Therefore, these sender and receiver cells have both Kanamycin and Ampicillin resistance.

The experiments reported here used a Nikon Eclipse E800 fluorescence microscope equipped with a Hamamatsu C4742 ORCA I CCD camera controlled by QED Imaging software. The cyan fluorescence filter is Chroma Cyan GFP (excitation: 436/20, emission: 480/40), the green fluorescence filter is a Chroma FITC/EGFP (excitation: 480/40, emission: 535/50), and the yellow fluorescence filter is a Chroma Yellow GFP (excitation: 500/20, emission: 535/30).

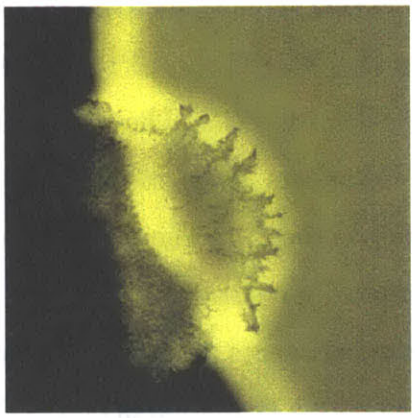
In the first visual observation experiment, sender and receiver cells were grown separately overnight @37°C, shaking at 250 RPM, each in 2ml LB Amp/Kan 1mM IPTG inside 14ml Falcon Polystyrene tubes (352051). Then, the JM2.300[pSND-1/pINV-112-R3] cells were pelleted at 6800 RPM, and resuspended in fresh 50μl LB Amp/Kan 1mM IPTG. The JM2.300[pRCV-3/pPROLar.A122] cells were pelleted at 6800 RPM, and resuspended in fresh 400μl LB Amp/Kan 1mM IPTG. 10μl droplets of receiver cells were spotted on an LB Amp/Kan 1mM IPTG agar plate, and dried for ten minutes. Then, 0.2μl droplets of sender cells were spotted next to the receivers such that the senders partially overlapped the receiver cells. A quick check under the microscope showed that the sender cells were emitting cyan fluorescence, while the receivers exhibited no fluorescence. The plate was then incubated for one hour @37°C.

Figure 5-10 shows the fluorescence pattern of a sender droplet partially overlapping a receiver droplet after the incubation period. The images were captured with a Plan Fluor 4X objective, using the cyan and yellow filters[2]. The physical width of each image is approximately 1.7mm. The senders were still emitting cyan fluorescence,

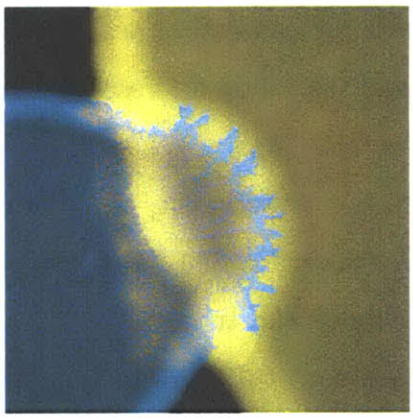
[2]The yellow filter is better than the green filter for distinguishing between cyan fluorescence and green fluorescence



Senders (cyan filter)



Receivers (yellow filter)



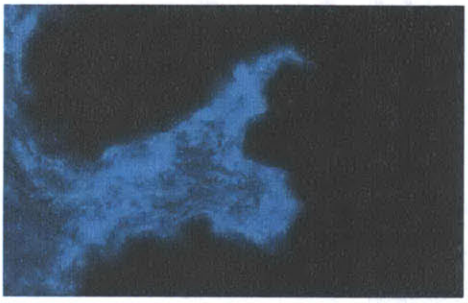
Overlay of images

Figure 5-10: Microscope fluorescence images of overlapping communicating cells (4X objective).

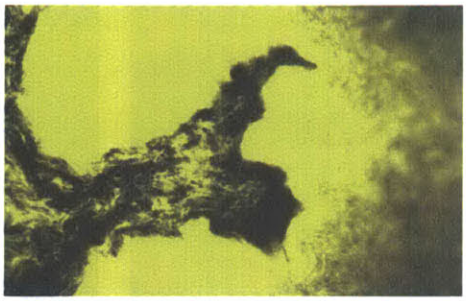
while the receiver cells were emitting green fluorescence due to the VAI that diffused from the senders. The interesting patterns resulted from liquid diffusion and mixing.

Figure 5-11 shows fluorescence images of the same communication experiment under a higher magnification (40X objective). Each picture represents a montage of three overlapping image captures, with a total physical width of approximately 0.27mm. At this degree of magnification, some individual bacterial cells can be distinguished.

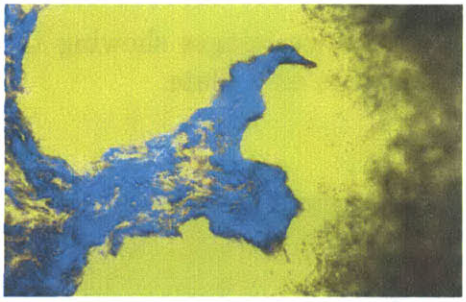
The second visual observation experiment captured a time-lapse series of images that illustrate the dynamic behavior of the communications (Figure 5-12). Sender and receiver cells were grown separately overnight @37°C, shaking at 250 RPM, in 2ml LB Amp/Kan cultures inside 14ml Falcon Polystyrene tubes, pelleted, and resuspended



Senders (cyan filter)



Receivers (yellow filter)



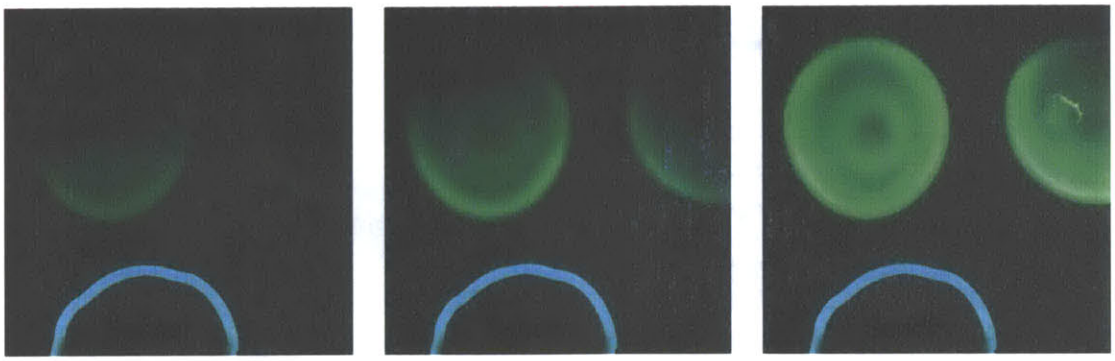
Overlay of images

Figure 5-11: Zoomed microscope fluorescence image of overlapping communicating cells (40X objective).

in 100μl fresh LB Amp/Kan. Two 7.0μl droplets of receiver cells were spotted on an LB Amp/Kan 1mM IPTG agar plate, and a 7.0μl droplet of sender cells was spotted near the two receiver droplet. For this experiment, the plate was incubated at room temperature under the microscope using a Plan Flour 1X objective. The physical width of each image is approximately 7mm.

First, a brightfield image was captured to record the location of the droplets. Then, a series of fluorescence images were captured at one minute intervals using the green filter. The cyan semi-circle in each fluorescence image is an artificial marker of the location of the sender droplet, superimposed based on the brightfield image.

The first fluorescence image was captured ten minutes after spotting the droplets. It shows that the receiver cells closest to the senders have already started responding to the VAI message. The two subsequent images display an increase in fluorescence intensity due to the diffusion and accumulation of VAI. Based on the fluorescence



10 minutes 45 minutes 90 minutes

Figure 5-12: Time-series fluorescence images showing the response of receivers to communication from nearby senders on a plate.

40 minutes 5:00 hours 7:30 hours

Figure 5-13: Time-series fluorescence images illustrating the response of smaller colonies of receivers to communication from nearby senders on a plate.

response, VAI appears to diffuse at approximately 1cm per hour through the agar.

The third visual observation experiment captured a time-lapse series of images with smaller colonies of receivers and a smaller droplet of servers (Figure 5-13). Sender cells were grown overnight @37°C, shaking at 250 RPM, in 2ml LB Amp/Kan inside 14ml Falcon Polystyrene tubes. Receiver cells were picked from −80°C cell stock into 1000μl LB Amp/Kan medium, then 20μl were plated on LB Amp/Kan, and incubated @37°C for 7 hours. Then, 0.3μl of sender cell were spotted, and bright-field and fluorescence images were captured as above. The three images show the communication gradient over time across the different receiver colonies.

Species	Relation to host	Regulate	I Gene	R Gene
Vibrio fischeri	marine symbiont	Bioluminescence	*luxI*	*luxR*
Vibrio harveyi	marine symbiont	Bioluminescence	*luxI*, M	*luxN,P,Q*
Pseudomonas aeruginosa	human pathogen	virulence factors	*lasI*	*lasR*
		Rhamnolipids	*rhlI*	*rhlR*
Yersinia enterocolitica	human pathogen	?	*yenI*	*yenR*
Chromobacterium violaceum	human pathogen	Violaceum production Hemolysin Exoprotease	*cviI*	*cviR*
Enterobacter agglomerans	human pathogen	?	*eagI*	?
Agrobacterium tumefaciens	plant pathogen	Ti plasmid conjugation	*traI*	*traR*
Erwinia caratovora	plant pathogen	virulence factors Carbapenem	*expI*	*expR*
Erwinia stewartii	plant pathogen	Extracellular capsule	*esaI*	*esaR*
Rhizobium leguminosarum	plant symbiont	Rhizome interactions	*rhiI*	*rhiR*
Pseudomonas aureofaciens	plant beneficial	Phenazine production	*phzI*	*phzR*

Table 5.1: Signaling Systems Similar to VAI: N-acyl-L-Homoserine Lactone Autoinducers in Bacteria.

5.3 Similar Signaling Systems

This chapter describes work that successfully isolates an important intercellular communication mechanism from a naturally occurring bacterial system, analyzes its components, and engineers its interfaces with standard genetic control and reporter mechanisms. While the work reported in this chapter captures one such communication mechanism, realistic genetically controlled developmental systems will require perhaps dozens of such signals. The LasI/LasR system from *Pseudomonas aeruginosa* [16], for example, appears to encode a similar regulatory system, but one that uses a different, and non-cross reacting autoinducer, and a different structure homologous to the lux box. Table 5.1 lists additional signaling systems similar to VAI that could serve as potential communication signals[81]. Isolation and characterization of such additional communication mechanisms will allow the construction of more complex multicellular systems.

Chapter 6

The Microbial Colony Language

The discussion so far has focused on realizing *in-vivo* digital-logic circuits and implementing rudimentary cell-to-cell communications. The digital abstraction reduces noise in computation and simplifies programming constructs for individual cells, while the diffusion-based communications serves as a foundation for coordinated behavior. However, in attempting to achieve complex and reliable coordinated behavior, one must consider the characteristics of the execution environment and its constraints. Most importantly, individual cells have limited computational capacity and frequently fail. In addition, cells continuously migrate, their interconnect topology is constantly in flux, messages between cells are frequently lost, and their program execution rates may differ due to fluctuations in kinetic rates.

Still, by programming large aggregates of cells that execute in parallel and coordinating their actions through intercellular communication, complex tasks can be accomplished. The goal of the Amorphous Computing project [2] is to develop novel paradigms for programming such substrates.

This section defines the *microbial colony language* (MCL), a simple computing paradigm for programming cell aggregates. MCL is simple enough for implementation in cells, yet expressive enough for interesting applications. The main features of this paradigm include unreliable computing elements, asynchronous execution, unreliable broadcast-based messaging with a small communications radius, and a simple event-driven rule-based programming language. This section also describes the Microbial

Colony Simulator (MCS), a software tool for this verifying the behavior of programs written in this language.

The language exposes programming mechanisms that cells can perform reliably, although the user cannot rely on the execution of any individual cell. The program for a single cell comprises event-triggered rules, Boolean state, Boolean operations, and limited range chemical diffusion for communication. The simplicity of language will likely enable programs written in MCL to be executed in cells, by using asynchronous *in vivo* digital circuits and cell-to-cell communications. I have implemented MCS, a language-level simulator that models cell aggregates executing microbial programs. Simulations show that these programs can produce large-scale pattern generation and coordinated group behavior.

Section 6.1 defines the syntax and semantics of the language. Section 6.2 describes the simulator and the pattern-forming behavior of example programs. Finally, Section 6.3 discusses the feasibility of implementing MCL programs in cell aggregates.

6.1 The Language

The execution model of MCL fits the substrate of biological cells. First, the model only allows the use of asynchronous logic due to continuous variations in kinetic rates of gates. Second, because of the dynamically changing interconnect, only broadcasts can be used for intercellular communication. The core of the language consists of messages between elements, Boolean markers stored in elements, and rules for taking action by the elements.

Rules

An MCL program comprises a set of event-driven rules. The general pattern for a rule is:

```
(event
 boolean-expression-of-markers
```

`(action_1 ... action_m))`

The meaning of the above pattern is:

if the Boolean expression of markers holds when the event takes place, perform the actions in any order

Events

An event can be an incoming message or marker decay. Rules are triggered by events and satisfaction of Boolean expressions of markers. If a single action enables multiple rules, then the activation order of the rules is unspecified. Importantly, the activation of rule r_1 resulting from event x could inhibit the activation of another rule r_2 from event x if r_1 unsets a marker required by r_2.

Messages

Messages between cells propagate by diffusion and decay after some time. The message tag and the arguments uniquely identify a message. The diffusion of a message is controlled by the hop-count (counting down). An element will react to an incoming message if either the element has not yet received the message, or the message has a higher hop-count than the same previous incoming messages to this element. A message may also decay after a given lifetime, and thus the element may react to the same message again in the future. If not specified, hop counts default to one, while lifetimes default to infinity. The general pattern for a *send* message action is:

(`send` *msg-tag* [(`diffuse` *hop-count*)] [(`decay` *lifetime*)])

Messages are matched in a rule as follows:

foo	:	message indicated by *foo* arrived
(*foo* . *args*)	:	message that arrived contains arguments
(*msg* (`diffuse` = 0))	:	message arrived with hopcount = 0
(*msg* (`diffuse` > 0))	:	message arrived with hopcount > 0
*	:	no event needed to trigger this rule

Markers

The general pattern for setting and unsetting a marker:

(**set** *marker* [(**decay** *lifetime*)])	:	set the marker, optionally to decay after some time
(**unset** *marker*)	:	unset the marker, also causing a marker decay event (if marker previously set)

A marker may decay with a particular lifetime. The decay event is matched in a rule as follows:

(*bar* (**decay** = 0))	:	*bar*'s lifetime has fallen to 0, or has been unset (at which point bar is no longer present)

6.2 MCS: Language-Level Simulator

For desiging, simulating, and experimenting with coordinated cell behavior, one needs to examine the predicted behavior of programs in large cell aggregates. BioSPICE (Chapter 3) cannot simulate large aggregates because the computation resources required are too great. I therefore implemented the Microbial Colony Simulator (MCS) on top of *hlsim*[3] and Coore's event-driven layer[14]. An MCL program, consisting of a set of asynchronous rules, is translated into an *hlsim* amorphous computing event-driven simulation. Because MCS simulates at a higher level than BioSPICE, it is appropriate for simulating cell aggregates with up to approximately ten thousand cells.

Figure 6-1 shows a simple MCL program that creates alternating segments of differentiated cells. The program consists of a set of independent asynchronously executing rules. Initially, a contiguous band of cells is distinguished as **Tube** cells, and a few cells at one of the ends of the band are classified as **Crest** cells. The **start** message directs **Crest** cells to send a message to neighboring cells. This triggers a chain reaction of message transmissions and setting values of Boolean variables, ultimately resulting in the pattern of cells such as the one shown in Figure 6-2. The left-most image depicts the initial conditions, where orange circles represent **Tube**

```
(start                                  (created                    }message
 Crest                                   (or C D)                   }condition
 ((send (make-seg C 1) 3)))              ((set Waiting 10))         }actions

((make-seg seg-type seg-index)          (*
 (and Tube (not C) (not D))              (and Bottom C 1 (Waiting (= 0)))
 ((set seg-type)                         ((send (make-seg D 1) 3)))
  (set seg-index)
  (send created 3)))                    (*
                                         (and Bottom D 1 (Waiting (= 0)))
(((make-seg) (= 0))                      ((send (make-seg C 2) 3)))
 Tube
 ((set Bottom)))                        (*
                                         (and Bottom C 2 (Waiting (= 0)))
(((make-seg) (> 0))                      ((send (make-seg D 2) 3)))
 Tube
 ((unset Bottom)))                      (*
                                         (and Bottom D 2 (Waiting (= 0)))
                                         ((send (make-seg C 3) 3)))
```

Figure 6-1: An MCL program for creating segments: The program consists of a set of independently executing rules.

cells and green circles represent **Crest** cells. At the end of the simulation, red circles represent **C** segment cells and blue circles represent **D** segment cells.

Programs written in MCL can achieve more complex behavior and differentiation patterns. Figure 6-3 shows a series of simulation snapshots for another MCL program that generates large-scale patterns. As above, each circle represents a cell, and the shading of the circle represents the particular state of the element. The pattern shown here is a caricature of cell differentiation during embryogenesis to form somites along the neural tube. Somites are paired wedge-shaped segmentations in the paraxial mesoderm on either side of the notochord. The somites contribute to the differentiation of musculature regions, limbs, and the axial skeleton.

6.3 Implementing MCL Programs in Cells

The programming elements of MCL conveniently map to biochemical reactions discussed in previous chapters. DNA binding proteins and RS latches. can implement

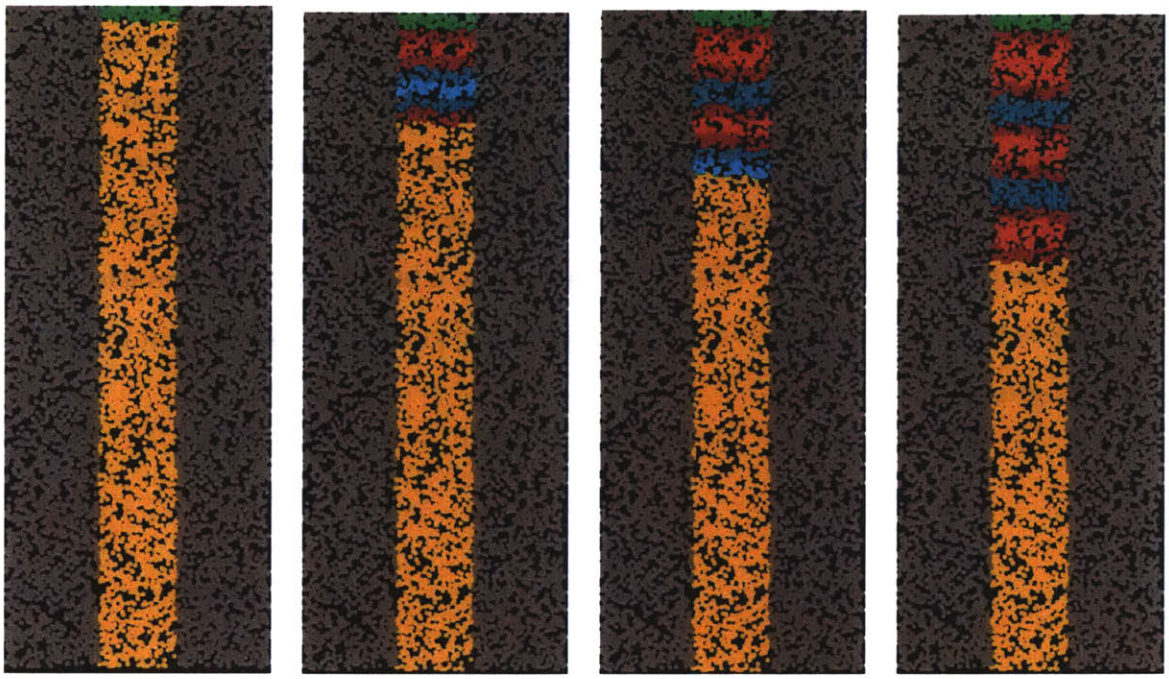


Figure 6-2: Segment formation: Time-series simulation snapshots of an MCL program execution resulting in cells that differentiate into an alternating pattern of C and D type segments.

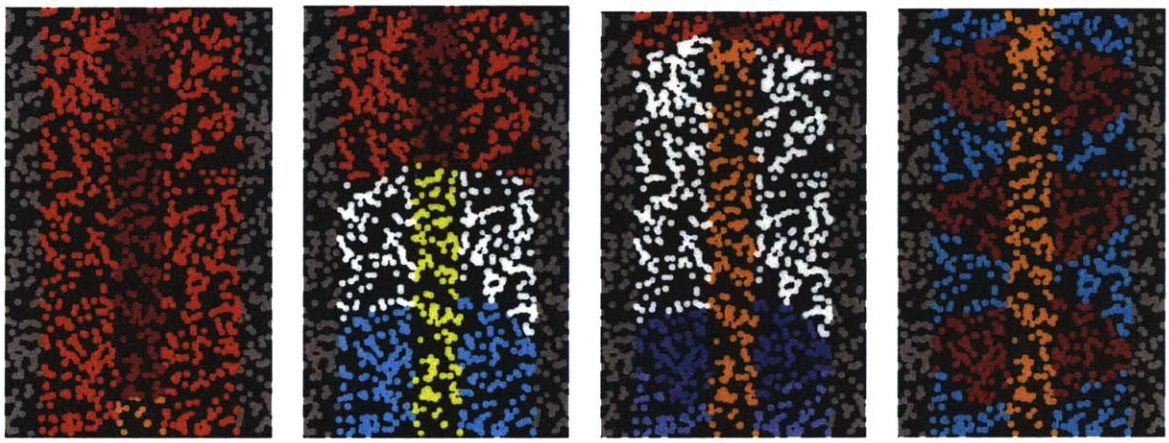


Figure 6-3: MCL pattern generation: Time-series simulation snapshots of pattern generation using another simple program written in MCL. The pattern formation is a caricature of somitegenesis, starting from the notochord and mesoderm (left-most image), and ending with three somites (right-most image).

Boolean state variables. Engineered genetic regulatory networks, such as the ones describes in Chapter 4, can implement logic statements. The diffusion-based communications can be implemented using enzymes such as luxI, intercellular signaling chemicals such as VAI, and DNA binding proteins such as luxR, as described in Chapter 5. Finally, controlled decay of markers can be achieved with amino acid tails that determine protein half-lives. With these mappings, a bio-compiler will translate small programs written in MCL into engineered genetic regulatory networks that implement the programs.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

This thesis lays the foundation of an engineering discipline for obtaining complex, predictable, and reliable cell behaviors by embedding biochemical logic circuits and programmed intercellular communications into cells. To accomplish this goal, this thesis provides a well-characterized cellular gate library, a biocircuit design methodology, intercellular communications for programming cell aggregates, and software design tools. The cellular gate library includes biochemical gates that implement the NOT, IMPLIES, and AND logic functions in *E. coli* cells.

This thesis introduces a biocircuit design methodology that comprises a mechanism for measuring the device physics of gates and criteria for evaluating, modifying, and matching gates based on their steady state behavior. By using the abstraction of logic circuits, complex and reliable behavior is synthesized from reliable, well-characterized components with matching input/output characteristics. An important element in biocircuit design is *genetic process engineering*, a methodology for mutating the DNA encoding of existing genetic elements to achieve the desired input/output behavior for constructing reliable circuits of significant complexity. The optimized components I synthesized with this process exhibit the desired signal restoration and noise margins for reliable digital computation. I demonstrated the feasibility of digital computation in cells by building several operational *in-vivo* digital logic circuits, each

composed of three gates that have been optimized by genetic process engineering.

Because of the limits to single cell circuit complexity, and to demonstrate coordinated behavior in cell aggregates, I also engineered intercellular communications. The mechanism relies on programmed enzymatic activity and chemical diffusions to carry messages, using DNA from the *Vibrio fischeri lux* operon. I built and characterized several circuits that integrate intracellular logic with *lux* operon-based intercellular communications.

In addition to the above experimental contributions, I developed BioSPICE, a prototype software tool for biocircuit design. BioSPICE simulates *in-vivo* logic circuits using ordinary differential equations that model biochemical rate equations. The kinetics for the rate constants were derived from the literature and yield simulation results that predict the behavior of engineered biocircuits. BioSPICE simulations of modified rate constants illustrate the effects on the static and dynamic behavior of the circuits, and serve as motivation for genetically modifying components in laboratory experiments. Finally, this thesis presents the Microbial Colony Language (MCL), a simple model for programming cell aggregates. This model is powerful enough for programming cell aggregates to form interesting patterns, but is simple enough so that the language primitives can be mapped to the genetic reactions described above.

7.2 Future Work

The section discusses future work relevant to the above components of my thesis and other research directions.

The cellular gate library

The cellular gate library described in this thesis uses genetic elements of the Bacteriophage λ *cI* repressor, lactose operon, tetracycline repressor, and lux operon. I chose these elements because they have been studied extensively in the literature and are readily available for cloning. In the future, the construction of more elaborate biocircuits will require the assembly of a larger component library.

There are two main sources for candidate genetic elements for the component library. The first is naturally occurring dna-binding proteins, which number in the thousands[51]. Only a handful of these elements have been studied extensively and are readily available from either commercial or academic sources. Therefore, a major effort will be isolating these genetic cadidates from their various organisms. In addition to finding natural sources, a second, potentially more efficient source for genetic candidates, is to synthesize artificial dna-binding proteins for use as signals. For example, combinatorial chemistry techniques, along with methods such as phage display, can yield large libraries of novel DNA binding proteins and corresponding operators. One potential source of a very large set of non-interacting signals is engineered Zinc Finger DNA binding proteins[37].

Future component libraries will also include elements that perform other digital logic functions (for example, OR, NAND, XOR), and gates implementing analog behavior. Electrical circuits can provide inspiration for implementing many types of logic components. However, it is also important to examine existing genetic regulatory systems to find biologically inspired computational elements that can be implemented reliably and efficiently in cells. For example, in cases where multiple transcription factors determine the activation of genes, it may be efficient to implement discrete logic components that have more than two logic states.

Genetic process engineering

As we move towards component libraries with genetic elements from a large variety of organisms and with widely different kinetic characteristics, genetic process engineering will become essential to synthesizing elements with standard interfaces and matching characteristics. It will be important to streamline and automate genetic process engineering in order to make it high-throughput. A major emphasis will focus on how to efficiently choose, synthesize, and test the genetic modifications. The biocircuit engineer may choose mutations based on existing knowledge, such as ribosome binding sites with known efficiencies or operator/protein mutations that affect DNA binding. Alternatively, the engineer may generate large scale libraries of mutant circuits and

screen for ones that have the desired behavior.

Constructing *in-vivo* logic circuits

While in theory the computing abstraction and implementation presented in this thesis allows the construction of biocircuits of any complexity, metabolic constraints are likely to limit the size of circuits in single bacterial cells to less than one thousand gates. We are still far from constructing circuits of such magnitude, although advances in cloning techniques, oligonucleotide synthesis, custom gene synthesis, and improved chromosomal integration techniques will make it possible to efficiently construct such large circuits in the foreseeable future.

Software design tools

For designing circuits with high complexity, an important requirement will be improved modeling and behavioral prediction techniques. There are two important aspects that will improve the utility of BioSPICE. First, the experimental data should be incorporated back into the database of kinetic constants. To achieve this requires normalizing fluorescence intensity data with actual protein concentrations in the given organisms. It also requires a mechanism for converting the transfer curve experimental results into kinetic constants that constitute the logical computation process. The second effort that will improve BioSPICE is to model the stochastic behavior of these synthetic biochemical systems and experimentally verify the predictions by observing the fluctuations exhibited by cell populations.

Intercellular communications

There are signalling mechanisms similar to the *Vibrio fischeri lux* operon that could serve as potential communication signals[81]. Isolation and characterization of additional diffusion-based communication mechanisms will facilitate the construction of more complex multicellular systems. Nature also provides other types of communication mechanisms, for example ones that involve specific cell-surface receptors. Such

mechanisms enable targetted intercellular communication where only pre-determined cell types are able to receive certain messages.

Other research directions

As discussed above, important research directions include large scale biocircuit modeling, design, synthesis, and testing. However, to make this technology useful for a broad range of applications will also require other research efforts. For example, an important direction is to implement these biocircuits in other organisms, such as the more complex eukaryotic cells (yeast, plants, mammalian cells), and the simpler mycoplasmas. Another exciting opportunity is to integrate synthetic biocircuits with existing cellular functions, both for observational purposes and for precise control of the production of certain proteins. Finally, there are many interesting challenges and opportunities in programming cell aggregates, where biocircuits can be used for a variety of applications such as programmed tissue engineering or directing cellular robots to assemble molecular scale structures. At this stage, we have only begun to explore the potential of engineering precise, reliable, and complex predetermined behavior into cells and cell aggregates, which can ultimately provide significant benefits to fields such as biomedicine, agriculture, biological research, and biomaterials manufacturing.

Appendix A

Transfer Curve Experiments

A.1 Plasmid Construction

plasmid	parent #1	parent #2	description of construct
pINV-4	pPROLar.A122	pGFP(LVA)	Insert p(lac):GFP(LVA) into p15A ori plasmid
pINV-5	pINV-4	pTrcHisA	Insert p(lacIq):lacI
pINV-101	pINV-5	pPW121	Insert RBS-II:cI downstream of p(lac)
pINV-102	pINV-5	pEYFP	Replace GFP(LVA) with EYFP
pINV-106	pINV-102	pBR322	Insert p(lac):EYFP into pBR322 backbone
pINV-107	pINV-106	pPW121	Replace p(lac) with $\lambda_{P(R-O12)}$
pINV-107-mut4	pINV-107	n/a	1bp mutation of $\lambda_{P(R-O12)}$
pINV-107-mut5	pINV-107	n/a	2bp mutation of $\lambda_{P(R-O12)}$
pINV-107-mut6	pINV-107	n/a	3bp mutation of $\lambda_{P(R-O12)}$
pINV-110	pINV-101	pECFP	Replace GFP(LVA) with ECFP
pINV-112-R1	pINV-110	n/a	Replace RBSII with RBS-D
pINV-112-R2	pINV-110	n/a	Replace RBSII with RBS-G
pINV-112-R3	pINV-110	n/a	Replace RBSII with RBS-H
pINV-202	pINV-102	pPROTet.E132	Replace p(lacIq) with P(LtetO-1)
pINV-203	pINV-107	pcDNA6/TR	Replace EYFP with *tetR*
pINV-206	pINV-202	pECFP	Add ECFP after *lacI*

Table A.1: Overview of the construction of the transfer curve plasmids.

Table A.1 provides an overview of the plasmids constructed for the transfer curve experiments. Figures A-1, A-2, and A-3 illustrate the lineage of the plasmids and their major features. These plasmids were constructed using basic molecular cloning techniques described in standard molecular biology laboratory manuals[7, 70]. Table A.2 lists the oligonucleotides and restriction enzymes used to construct these

plasmids. Entries with PCR primers indicate the use of GibcoBRL PCR Supermix High Fidelity to PCR amplify a parent plasmid with overhanging ends that contain the appropriate restriction sites. Otherwise, the plasmids were purified from transformed *E. coli DH5α* cells and digested with the appropriate enzymes. Restriction enzymes were purchased from New England Biolabs. Synthetic oligonucleotides were synthesized on an Applied Biosystems 394 DNA/RNA Synthesizer, purchased from MIT's biopolymer Laboratory, and purchased from Oligos Etc. All cloning work was performed with heat shock transformation using Max Efficiency *DH5α* Competent Cell from GibcoBRL.

Source plasmids were obtained as follows: pECFP, pEYFP, pGFP(LVA), pPROLar.A122, and pPROTet.E132 from Clontech, pBR322 and pPW121 from ATCC, and pcDNA6/TR and pTrcHisA from Invitrogen. DNA sequencing to verify the plasmids was done with the Applied Biosystems Prism 310 Genetic Analyzer.

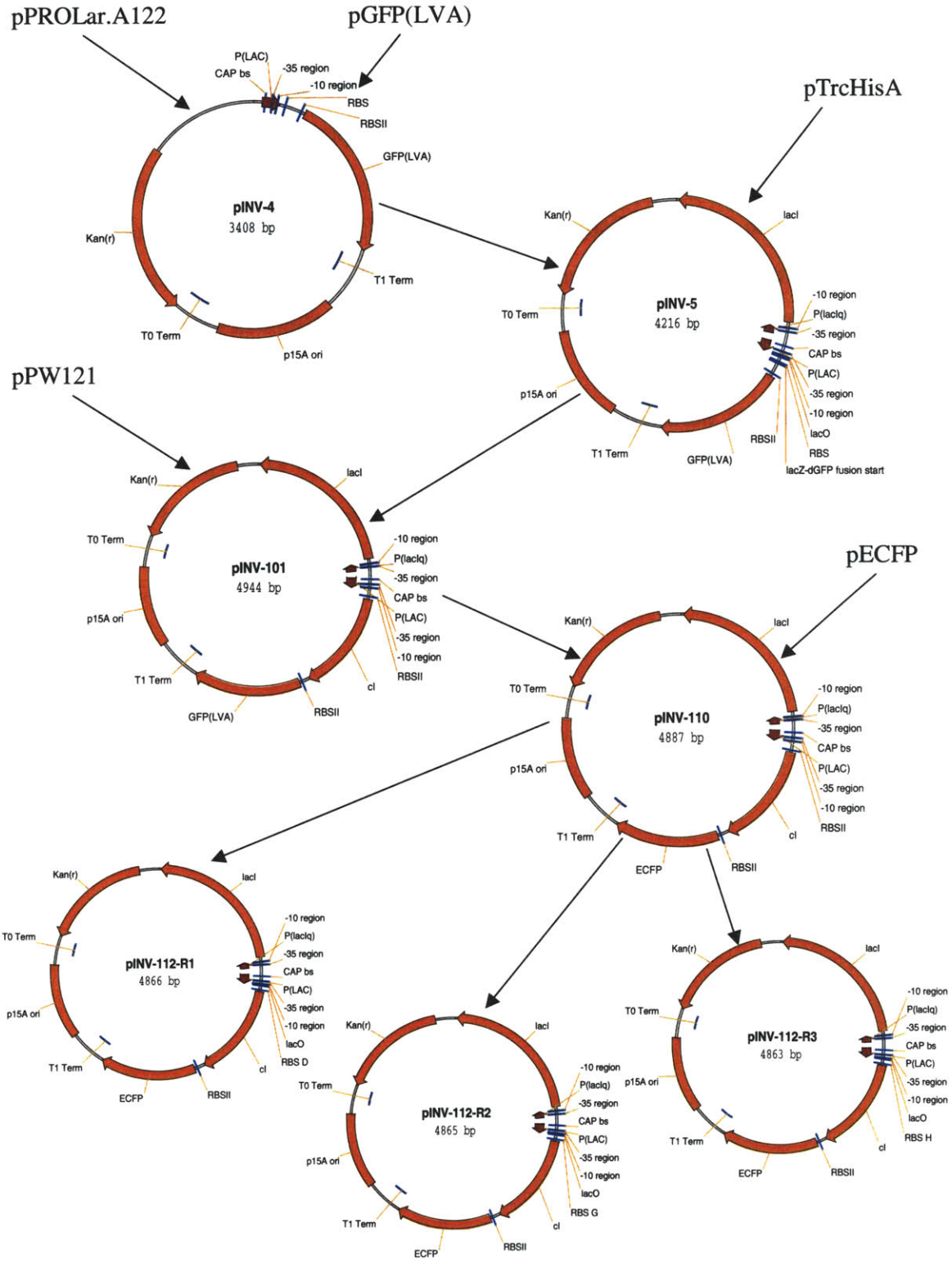


Figure A-1: Features and lineage of the transfer curve plasmids I.

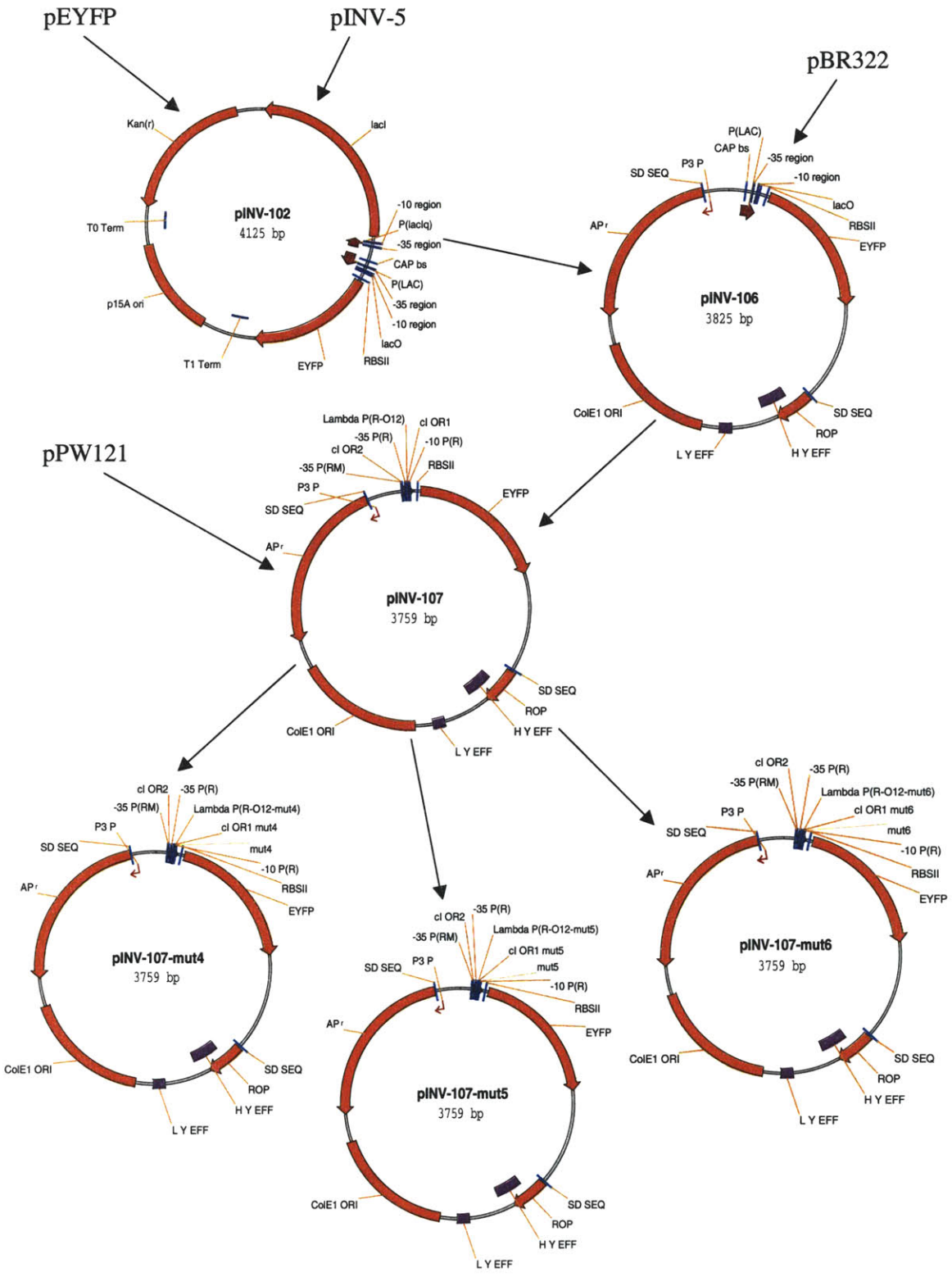


Figure A-2: Features and lineage of the transfer curve plasmids II.

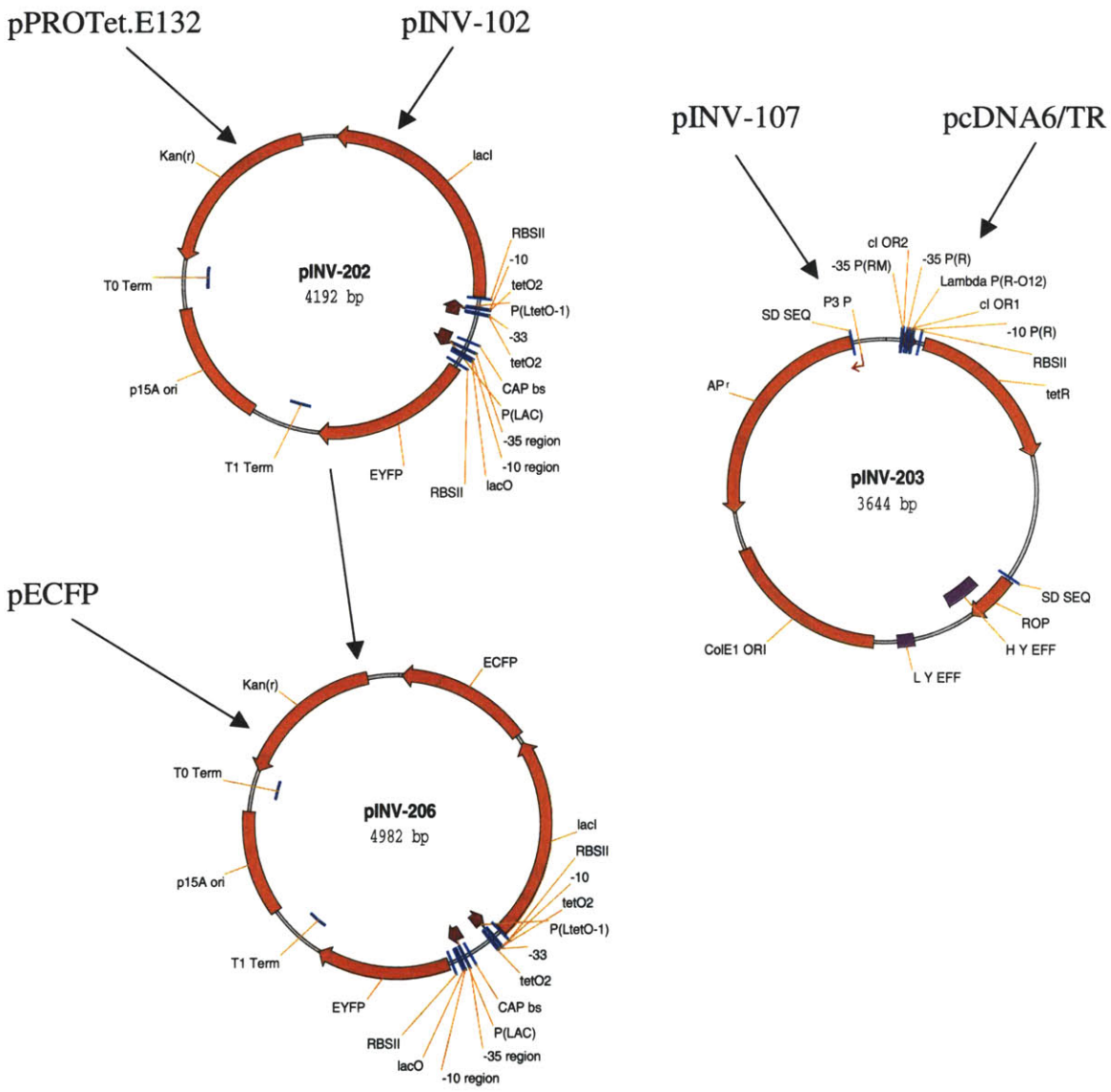


Figure A-3: Features and lineage of the transfer curve plasmids III.

plasmid	parent	digest	pcr primers
pINV-4	pPROLar.A122	NheI/XhoI	AACGCTCGAGTAGAGGCATCAAATAAAACG
			ATCGGCTAGCTTCATAGGTGATTGCTCAGG
	pGFP(LVA)	NheI/XhoI	ACTGGCTAGCACGACAGGTTTCCCGACTGG
			ACTGCTCGAGACGGCCGACTAGTAGGTCAGC
pINV-5	pTrcHisA	AatII/NheI	ATGACTGCTAGCTCTAGATTACGTTGACACCATCGAATGG
			ATGCGACGTCATTGCGTTGCGCTCACTGCC
	pINV-4	AatII/NheI	*none*
pINV-101	pINV-5	Acc65I	ATCATCGGTACCTAGAATTAAAGAGG
			TCAATCGGTACCCCTGTGTGAAATTGTTATCC
	pPW121	Acc65I	ATGATGGGTACCAATTGTTATCAGCTATGCGC
			AGAAGAGGTACCAATTAAAGAGGAGAAATTAAGCATGAGCACAAAAAAGAAACC
pINV-102	pEFYP	EcoRI	AGTAGGGAATTCAATTAAAGAGGAGAAATTAAGCATGGTGAGCAAGGGCGAGGAG
			ATCTCATCAGTTGGAATTCTAGAGTCG
	pINV-5	EcoRI	ATACCTGAATTCTTAGCTGACCTACTAGTCGG
			TCAATCGGTACCCCTGTGTGAAATTGTTATCC
pINV-106	pINV-102	AatII/BsaI	TACCTAGACGTCACCATTCGATGGTGTCAACG
			AGGTTAGGTCTCACCGAATCAGCTAAGAATTCTAGAGTCG
	pBR322	AatII/AvaI	*none*
pINV-107	pINV-106	AvrII	AGTCTACCTAGGAATTCAATTAAAGAGGAG
			ATCAGACCTAGGAACCTGTCGTGCTAGCTCTAG
	pPW121	BsaI	ATGGATGGTCTCACTAGTAAATATCTAACACCGTGCG
			AGTTACGGTCTCACTAGGACATGCAACCATTATCACCG
pINV-107-mut4	pINV-107	BspMI	TCATCGACCTGCATCGTTACATCTGGCGGTGATAATGGTTGCATG
			ACGTAGACCTGCAGAGGTAAAATAGTCAACACGCAC
pINV-107-mut5	pINV-107	BspMI	TCATCGACCTGCATCGTTACATATGGCGGTGATAATGGTTGCATG
			ACGTAGACCTGCAGAGGTAAAATAGTCAACACGCAC
pINV-107-mut6	pINV-107	BspMI	TCATCGACCTGCATCGTTACAGATGGCGGTGATAATGGTTGCATG
			ACGTAGACCTGCAGAGGTAAAATAGTCAACACGCAC
pINV-110	pINV-101	BsaI	AGGTATGGTCTCATCGGCCGTCTCGAGTAGAGGC
			ATCAGTGGTCTCAGCAATTCTAGGTACCAATTGTTATC
	pECFP	BsaI	ATGATCGGTCTCATTGCTTAAAGAGGAGAAATTAAGCATGGTGAGCAAGGGCGAGGAGCTG
			ATCACTGGTCTCACCGATTACTTGTACAGCTCGTCC
pINV-112-R1	pINV-110	SalI	ACTGATGTCGACTCACACAGGAAACCGGTTCGATGAGCACAAAAAAGAAACC
			ATCTACGTCGACAATTGTTATCCGCTCACAATTCC
pINV-112-R2	pINV-110	SalI	ACTGATGTCGACTCACACAGGAAAGGCCTCGATGAGCACAAAAAAGAAACC
			ATCTACGTCGACAATTGTTATCCGCTCACAATTCC
pINV-112-R3	pINV-110	SalI	ACTGATGTCGACTCACACAGGACGGCCGGATGAGCACAAAAAAGAAACC
			ATCTACGTCGACAATTGTTATCCGCTCACAATTCC
pINV-202	pPROTet.E132	BsaI	ACTTCAGGTCTCTACCATAGGCGTATCACGAGGCCCTTTC
			TCTATCGGTCTCATTGCTGAATTCGGTCAGTGCGTCCTG
	pINV-102	BsaI	AGGTTAGGTCTCATGGTGTCAACGTAATCTAGAG
			ATCAGTGGTCTCAGCAAAGAGGAGAAATTAAGCGTGAAACCAGTAACGTTATACG
pINV-203	pcDNA6/TR	BspMI	ACTGATACCTGCTACGATGTCTAGATTAGATAAAAGTAAAG
			ATCTGAACCTGCACGATTAAGACCCACTTTCACATTTAAGTTG
	pINV-107	BspMI	ACAGTAACCTGCTGACTTAATTCTTAGCTGATTCGGGCAG
			TAGTGAACCTGCTAGTACATGCTTAATTTCTCCTCTTTAATTG
pINV-206	pINV-202	BsaI	TCAGTAGGTCTCATGCTTGGCAATTCCGACGTCATTGC
			ATCTGAGGTCTCTACCTTCTGGTAAGGTTGGGAAGCC
	pECFP	BsaI	TGTTCAGGTCTCGAGCAAAGAGGAGAAATTAAGCATGGTGAGCAAGGGCGAGGAG
			TGAGCAGGTCTCTAGGTACAAGTTGGTAATGGTAGCGACC

Table A.2: Oligonucleotides and restriction enzymes used to construct the transfer curve plasmids

A.2 Strains, Growth, Conditions, Chemicals

The host cell for all transfer curve experiment was *E. coli* JM2.300 [$\lambda-$, *lacI22, rpsL135 (StrR), thi-1*] (CGSC strain 5002). This cell does not contain *cI* (λ repressor) or *tetR* and carries a non-functional Lac repressor (lacI22), and therefore the cell's existing genetic elements do not interfere with the operation of the transfer curve circuits. The JM2.300 strain is also fast growing and capable of overexpressing plasmid bound genes.

For the experiments, cells were grown in LB medium (DIFCO) with various inducers and antibiotics (Sigma) in log phase between 9 and 12 hours @37°C with a single 1000:1 dilution step into fresh medium, shaking at 250 RPM inside 14ml Falcon Polystyrene tubes in a New Brunswick Scientific Innvoa 4230 refrigerated incubator shaker. Depending on the plasmid's antibiotic resistance, a combination of $100\frac{\mu g}{ml}$ Ampicilin and $50\frac{\mu g}{ml}$ Kanamycin was used. Isopropylthio-β-galactoside (IPTG) and Anhydrotetracycline (aTc) were used in various concentrations.

A.3 Gene Expression Assay

Gene expression data of fluorescence intensities were collected with a Becton-Dickinson FACSVantage flow cytometer with an argon excitation laser. For detecting cyan fluorescence, the laser excitation was set at 458nm with an emissions filter of 485/22nm. For detecting yellow fluorescence, the laser excitation was set at 514nm with an emissions filter of 575/26. Filters were obtained from Omege Optical.

In preparing the samples, after reaching optical density of 0.5 measured at 600nm, cells were peletted at 6800 RPM for 1 minute, washed with $0.22\mu m$ filtered PBS, peletted again, and resuspended in PBS. For each culture, 50,000 events were collected. WinMDI software (J. Trotter, The Scripps Research Institute, available at `http://facs.scripps.edu/software.html`) was used to analyze the data. The flow cytometry files were also analyzed with Matlab after conversion into ASCII format using MFI (E. Martz, University of Massachusetts, Amherst, available at `http://marlin.bio.umass.edu/mcbfacs/flowcat.html#mfi`).

Appendix B

Communications Plasmids

In order to experiment with intercellular communications, I constructed a series of plasmids, and then transformed them into *E. coli* cells. The plasmids can be roughly categorized into three groups: preliminary plasmids (Section B.1), plasmids that enable cells to transmit the message by catalyzing the formation of autoinducer (Section B.3), and plasmids that enable cells to respond to the message through the use of the appropriate region of the lux operon (Section B.2). Table B.1 gives an overview of the plasmids used for these experiments.

B.1 Preliminary Plasmids

Initially, we constructed a series of plasmids (Figure B-1) that could serve as templates for cloning the final sender and receiver plasmids. The first plasmid, pRW7-1, combines the backbone of the general purpose high copy number plasmid pUC19 with GFP(LVA) from Clontech pGFP(LVA). Both pUC19 and pGFP(LVA) were digested with SpeI and XmaI, and the GFP(LVA) CDS and its associated synthetic ribosome binding site (RBSII) were cloned into pUC19. GFP(LVA) is a variant of the green fluorescent protein with a destabilizing tail (amino acids RPAANDENYLVA) that results in a protein half life of approximately 40 minutes.

Next, to produce pRW7-2, a transcription termination region (rrnB T1) based on a sequence from pKK232-8 [61] was cloned into pRW7-1 using two oligonucleotides. The oligos were annealed by incubating @97°C for 10 minutes, then incubating @65°C for 15 minutes, incubating @24°C for 15 minutes, and finally storing @4°C, to produce

plasmid	parent #1	parent #2	description of construct
pRW7-1	pUC19	pGFP(LVA)	Insert GFP(LVA) into pUC19 backbone
pRW7-2	pRW7-1		Insert rrnB T1 before GFP(LVA)
pRW7-3	pRW7-3		Insert rrnB T1 after GFP(LVA)
pRW7-4	pRW7-3		Insert p(LAC-const) before GFP(LVA)
pRW-LPR-2	pRW7-3	pPW121	Replace p(LAC-const) with $\lambda_{P(R)}$
pRCV-3	pRW7-3	pTK-1	Insert luxR:P(L):P(R) before GFP(LVA)
pRCV-4	pRW7-3	pTK-1	Insert luxR:P(L):P(R):luxI before GFP(LVA)
pTK-1	*Vibrio fischeri*	n/a	Insert lux operon into pUC19 backbone
pLux19-S1	pTK-1	pUC19	Insert luxI after p(lac)
pLuxI-Tet-8	pTK-1	pPROTet.E132	Insert luxI after P(Ltet-O1)
pSND-1	pTK-1	pRW7-4	Insert luxI after p(LAC-const)

Table B.1: Plasmids for cell-to-cell communications.

the following double stranded segment with overhangs that match an AatII and XmaI digest:

ACCCGGGAATTCCCAGGCATCAAATAAAACGAAAGGCTCAGTCGAAAGACTGGGCCTTTC GTTTTATCTGTTGTTTGTCGGTGAACGCTCTCACCGGT
TGCATGGGCCCTTAAGGGTCCGTAGTTTATTTTGCTTTCCGAGTCAGCTTTCTGACCCGGAAAG CAAAATAGACAACAAACAGCCACTTGCGAGAGTGGCCAGGCC

The annealed oligos were then ligated into pRW7-2 digested with AatII and XmaI. The plasmid pRW7-3, which includes the same transcription termination region but on the 5' end of GFP(LVA), was constructed in a similar fashion. The oligos used have HindIII and XbaI overhangs, and were cloned into a pRW7-2 HindIII/XbaI digest.

The final preliminary plasmid pRW7-4 includes p(LAC-const), a new constitutive synthetic promoter, in front of the GFP(LVA) CDS. We designed the constitutive promoter p(LAC-const) based on the LAC promoter, as shown in Figure B-2. In p(LAC-const), the lacO and CAP binding sites have been removed, and the -10 and -35 regions have been modified to resemble the consensus -10 and -35 regions respectively [50]. p(LAC-const) was introduced into pRW7-3 (digested with AgeI/Acc65I) using a pair of oligos with AgeI and Acc65I overhangs that were annealed using the same procedure as above. The plasmid pRW7-4 was transformed into *E. coli DH5α* chemically competent cells. The construct consisting of p(LAC-cons) followed by GFP(LVA) was verified by detecting the fluorescense of the cells (data not shown).

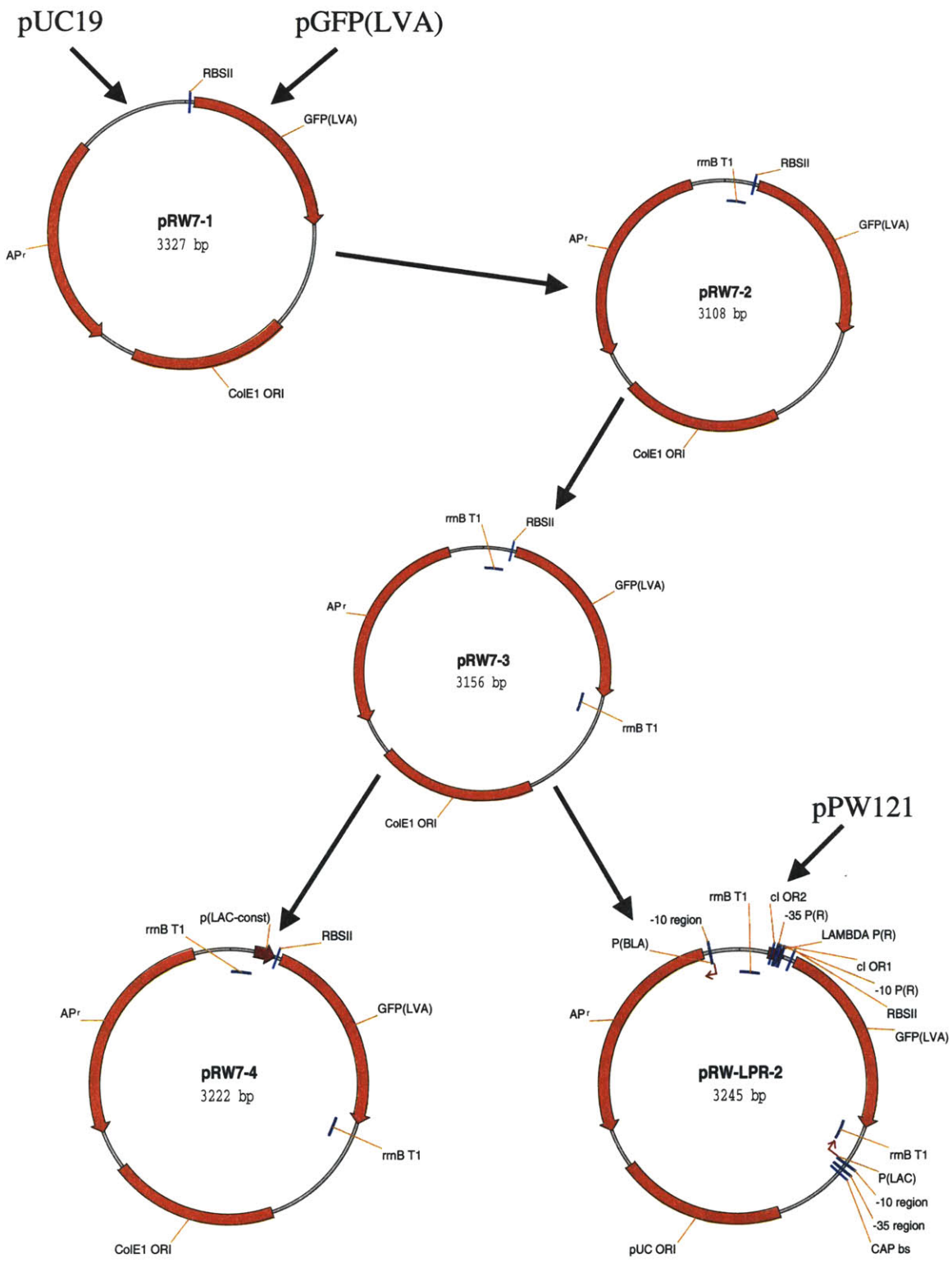


Figure B-1: Preliminary plasmids

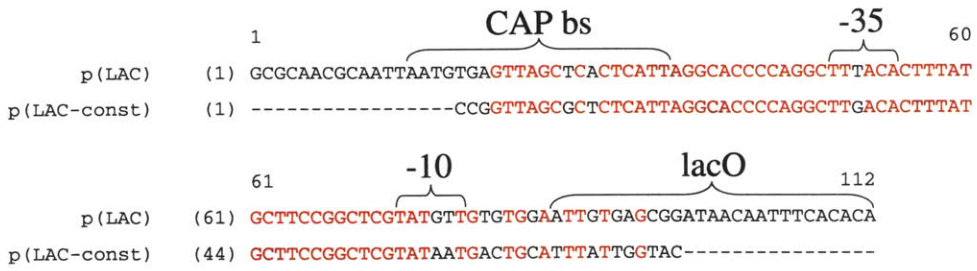


Figure B-2: Comparison of p(LAC) with p(LAC-const)

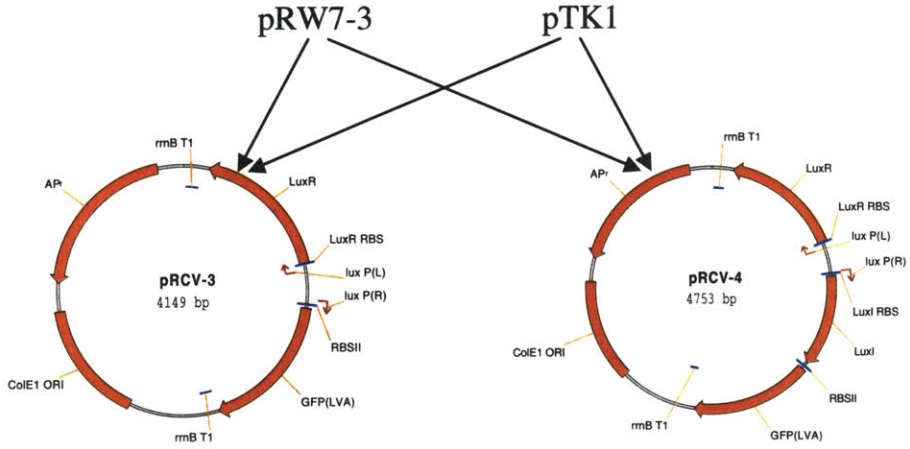


Figure B-3: Receiver plasmids

B.2 Receivers

The receiver plasmid pRCV-3 was constructed using pRW7-3 as the plasmid backbone and by inserting the luxR $P_L P_R$ region from pTK1 upstream of GFP(LVA). We performed a PCR reaction using forward primer 5'(CATGGGTACCTCCGGAATAAAGCTTTACTTACGTAC)3' and reverse primer 5'(CATGGGTACCGGCCGGTTTATTCGACTATAACAAACC)3', yielding the luxR $P_L P_R$ region with Acc65I cut sites at both tails. The PCR product was then ligated into a pRW7-3 Acc65I digest, and the resulting colonies were screened by restriction mapping and and partial plasmid sequencing to ensure that the insert was in the correct orientaion.

The receiver plasmid pRCV-4 served as a control plasmid to verify the ability of the lux operon to exert positive control on the synthesis of GFP(LVA). The luxR$P_L P_R$luxI region from pTK1 was extracted with a PCR reaction using forward primer 5'(CATGGGTACCTCCGGAATAAAGCTTTACTTACGTAC)3' and reverse primer

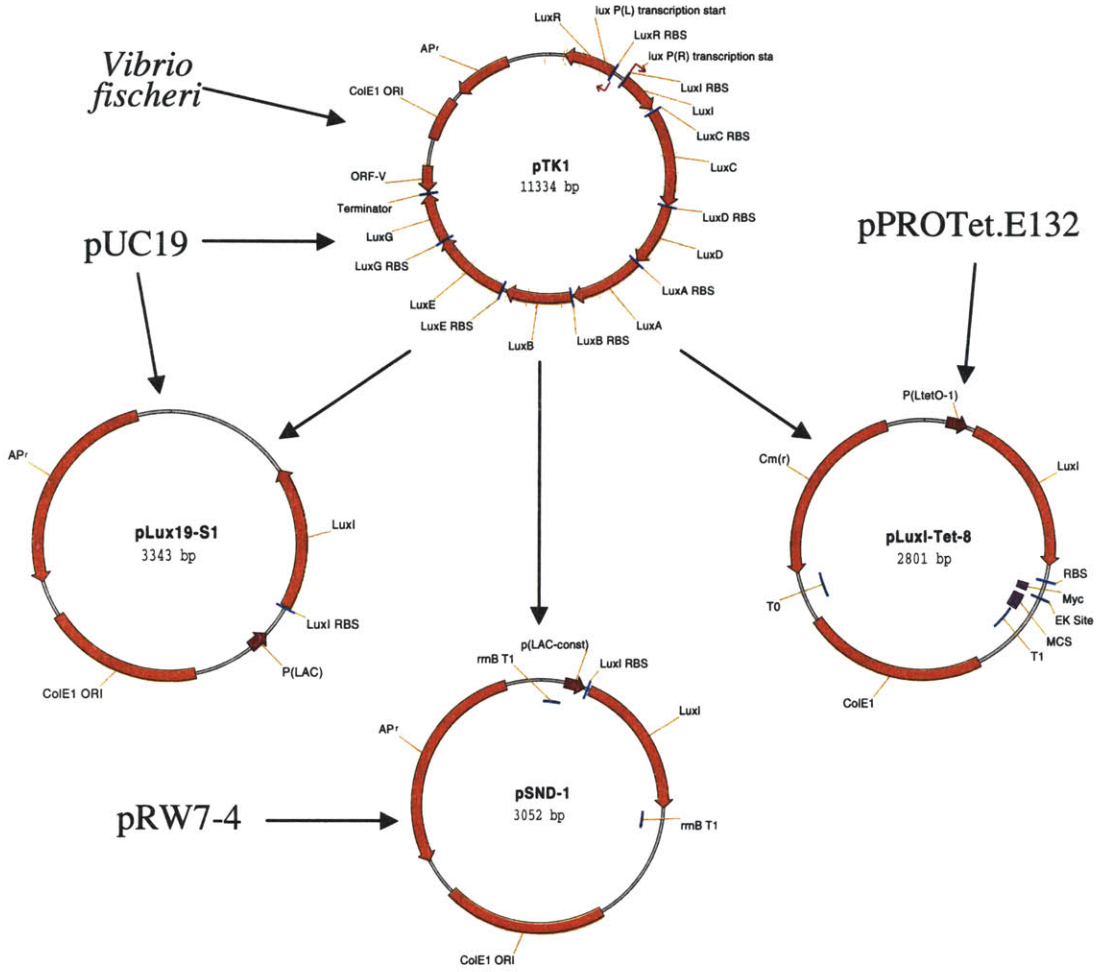


Figure B-4: Sender plasmids

5'(CCTTGGTACCGGCCGAACAACATTAATTTAAGACTGC)3'. As above, the PCR product was then ligated into a pRW7-3 Acc65I digest, and the resulting colonies were screened by restriction mapping and partial plasmid sequencing to ensure that the insert was in the correct orientaion.

B.3 Senders

We isolated individual components of the *Vibrio fischeri* system for further use. Plasmids described in this section are shown Figure B-4. The LuxI coding region was cloned and placed under control of the Lac promoter of the pUC19 plasmid. This was done by PCR of the pTK1 plasmid DNA using selected primers which included non-matching 5' EcoRI cut sites. Specifically, we performed a PCR reaction

using forward primer 5'(AGG↓AATTCGAATAAACGCAAGGGAG)3' and reverse primer 5'(CCG↓AATTCCCTATAATATACTTAG)3', yielding the full length LuxI coding sequence, including the ribosomal binding site, but with paired, distal EcoRI cut sites. PCR was performed using Life Technology High Fidelity PCR Supermix (25μl), 1μl of each primer, and 1μl of 300$ng/\mu l$ pTK1 plasmid DNA. The reaction was denatured 5 minutes @94°C, followed by 30 cycles of denaturing 30 seconds @94°C, annealing 30 seconds @50°C, and extension 1 minute @70°C. Reaction products were verified by gel electrophoresis, and separated from primers using the Bio101 Geneclean spin protocol. The purified PCR product was digested with EcoRI, and ligated with prepared pUC19 vector, which had been cut with EcoRI and dephosphorylated with Amersham shrimp alkaline phosphatase.

The resulting ligation was transformed into *E. coli DH5α* and plated on LB AMP. The transformed colonies exhibited two distinct morphologies, clear, small colonies and opaque, large colonies. Six of each colony morphology were streaked, grown, and minipreped. Restriction digests and gel eletrophoresis showed that the small colonies contained the LuxI gene in the correct, expressing, orientation. One such clone, pLuxI19-S1 was chosen for further study.

The same EcoRI digested LuxI PCR product was also similarly cloned into the Clonetech pPROTET.E332 plasmid. This plasmid contains a Col-E1 ori, chlomamphenicol resistance gene, and a TetO controlled promoter. The TetO promoter is inhibited by the TetR gene product, in the presence of the anitbiotic tetracycline. The TetR gene is chromosomally carried in a special version of *E. coli*, which also carries the spectinomycin resistance gene. As a first step, the ligation reaction was transformed into subcloning efficiency DH5α cells, grown up in LB chloramphenicol (50$\mu g/ml$). After verification of the correct insert, miniprep DNA was re-transformed into the TetR containing strain, which was then grown in LB spectinomycin chloramphenicol broth.

The PROTet system allows controlled expression of the inserted gene using varying amounts of a non-growth-inhibitory version of tetracycline, anhydro-tetracycline (aTc). In this way, we can control expression of the LuxI gene, and hence the level

of VAI, in these cells, through control over the aTc concentration.

The plasmid pSND-1 was constructed for constitutive expression of luxI, by removing the GFP(LVA) CDS from pRW7-4 and replacing it with luxI from pTK1. A PCR reaction using forward primer 5'(CATGGGTACCTCCGGAATAAAGCTTTACTTACGTAC)3' and reverse primer 5'(CATGAAGCTTAACAACATTAATTTAAGACTGC)3' yielded the luxI coding sequence, including the ribosomal binding site. The PCR product was then ligated with a pRW7-4 Acc65I/HindIII digest, and transformed into chemically competent *DH5α*.

Bibliography

[1] H. Abelson, D. Allen, D. Coore, C. Hanson, G. Homsy, T. F. Knight Jr., R. Nagpal, E. Rauch, G. J. Sussman, and R. Weiss. Amorphous computing. Technical Report AI Memo No. 1665, MIT Artificial Intelligence Laboratory, 1999.

[2] H. Abelson, T. F. Knight Jr., and G. J. Sussman. Amorphous computing. *White paper*, October 1995.

[3] S. Adams. A high level simulator for gunk. Internal Publication, Amorphous Computing Project, October 1997.

[4] A. Arkin. *Simulac* and *deduce*. Available via the World Wide Web at `http://gobi.lbl.gov/~aparkin/Stuff/Software.html`, 2001.

[5] A. Arkin and J. Ross. Computational functions in biochemical reaction networks. *Biophysical Journal*, 67:560–578, August 1994.

[6] A. Arkin, J. Ross, and H. H. McAdams. Stochastic kinetic analysis of developmental pathway bifurcation in phage λ-infected *escherichia coli* cells. *Genetics*, 149:1633–1648, 1998.

[7] F. M. Ausubel, R. Brent, R. E. Kingston, D. D. Moore, J. G. Seidman, J. A. Smith, and K. Struhl. *Short Protocols in Molecular Biology*. Wiley, 1999.

[8] A. Babloyantz and G. Nicolis. Chemical instabilities and multiple steady-state transitions in monod-jacob type models. *Journal of Theoretical Biology*, 34:185–192, 1972.

[9] B. L. Bassler. How bacteria talk to each other: regulation of gene expression by quorum sensing. *Current Opinion in Microbiology*, 2:582–587, 1999.

[10] A. Becskei and L. Serrano. Engineering stability in gene networks by autoregulation. *Nature*, 405:590–593, June 2000.

[11] U. S. Bhalla and R. Iyengar. Emergent properties of networks of biological signalling pathways. *Science*, 283:381–387, 1999.

[12] W. Bialek. Stability and noise in biochemical switches. *Advances in Neural Information Processing Systems 13*, pages 103–109, 2001.

[13] J. U. Bowie and R. T. Sauer. Identification of c-terminal extensions that protect proteins from intracellular proteolysis. *Journal of Biological Chemistry*, 264(13):7596–7602, 1989.

[14] D. Coore. *Botanical Computing: A Developmental Approach to Generating Interconnect Topologies on an Amorphous Computer.* PhD thesis, Massachusetts Institute of Technology, January 1999.

[15] D. Coore, R. Nagpal, and R. Weiss. Paradigms for structure in an amorphous computing. Technical Report AI Memo No. 1614, MIT Artificial Intelligence Laboratory, 1997.

[16] T. deKievit, P. C. Seed, J. Nezezon, L. Passador, and B. H. Iglewski. Rsal, a novel repressor of virulence gene expression in *pseudomonas aeruginosa. J. Bacteriol*, 181:2175–2184, 1999.

[17] D. E. Draper. Translational initiation. In Frederick C. Neidhardt, editor, *Escherichia Coli and Salmonella*, pages 902–908. ASM Press, Washington, D.C., 2 edition, 1992.

[18] G. M. Dunny and S. C. Winans, editors. *Cell-Cell Signaling in Bacteria.* ASM Press, Washington, DC, 1999.

[19] M. Dworkin. Cell-cell interactions in the myxobacteria. In *Symp. Soc. Gen. Microbiol.*, volume 25, pages 135–147, 1973.

[20] M. Dworkin and D. Kaiser. Cell interactions in myxobacterial growth and development. *Science*, 230:18–24, 1985.

[21] A. Eberhard, A. L. Burlingame, C. Eberhard, G. L. Kenyon, K. H. Nealson, and N. J. Oppenheimer. Structural identification of autoinducer of *photobacterium fischeri* luciferase. In *Biochemistry*, volume 20, pages 2444–2449, 1981.

[22] L. Edelstein-Keshet. *Mathematical Models in Biology.* McGraw-Hill, New York, 1988.

[23] M. Elowitz and S. Leibler. A synthetic oscillatory network of transcriptional regulators. *Nature*, 403:335–338, January 2000.

[24] D. Endy, D. Kong, and J. Yin. Intracellular kinetics of a growing virus: a genetically structured simulation for a bacteriophage t7. *Biotechnology and Bioengineering*, 55(2):375–389, 1997.

[25] D. Endy, L. You, J. Yin, and I. J. Molineux. Computation, prediction, and experimental tests of fitness for bacteriophage t7 mutants with permuted genomes. In *Proceedings of the National Academy of Science*, volume 97, pages 5375–5380, 2000.

[26] J. Engebrecht, K. H. Nealson, and M. Silverman. Bacterial bioluminescence: isolation and genetic analysis of the functions from *vibrio fischeri.* In *Cell*, volume 32, pages 773–781, 1983.

[27] W. C. Fuqua and A. Eberhard. *Signal generation in autoinduction systems: synthesis of acylated homoserine lactones by LuxI-type proteins*, pages 211–230. G. M. Dunny and S. C. Winans, Washington, DC, 1999.

[28] W. C. Fuqua, S. Winans, and E. P. Greenberg. Quorum sensing in bacteria: The luxr-luxi family of cell density-responsive transcriptional regulators. *J. Bacteriol*, 176:269–275, 1994.

[29] T. Gardner, R. Cantor, and J. Collins. Construction of a genetic toggle switch in *escherichia coli. Nature*, 403:339–342, January 2000.

[30] L. Glass. Classification of biological networks by their qualitative dynamics. *Journal of Theoretical Biology*, 54:85–107, 1975.

[31] L. Glass. Combinatorial and topological methods in nonlinear chemical kinetics. *Journal of Chemical Physics*, 63:1325–1335, 1975.

[32] L. Glass and S. A. Kauffman. Co-operative components, spatial localization and oscillatory cellular dynamics. *Journal of Theoretical Biology*, 34:219–237, 1972.

[33] L. Glass and S. A. Kauffman. The logical analysis of continuous, non-linear biochemical control networks. *Journal of Theoretical Biology*, 39:103–129, 1973.

[34] L. Glass and S. A. Kauffman. The logical analysis of continuous, non-linear biochemical control networks. *Journal of Theoretical Biology*, 44:167–190, 1974.

[35] G. Green, S. R. Kain, and R. Angres. Dual color detection of cyan and yellow derivatives of green fluorescent protein using conventional microscopy and 35-mm photography. In *Methods in Enzymology*, volume 327, pages 89–94, 2000.

[36] EP Greenberg. Quorum sensing in gram-negative bacteria. In *ASM News*, volume 63, pages 371–377, 1997.

[37] H. A. Greisman and C. O. Pabo. A general strategy for selecting high-affinity zinc finger proteins for diverse dna target sites. *Science*, 275:657–661, 1997.

[38] D. C. Hanselman and B. Littlefield. *Mastering MATLAB 6: A Comprehensive Tutorial and Reference.* Prentice Hall, Englewood Cliffs, NJ, 2001.

[39] R. W. Hendrix. *Lambda II.* Cold Spring Harbor Press, Cold Spring Harbor, New York, 1983.

[40] W. Hillen and C. Berens. Mechanisms underlying expression of tn10 encoded tetracycline resistance. *Annual Review of Microbiology*, 48:345–369, 1994.

[41] W. Hinrichs, C. Kisker, M. Düvel, A. Müller, K. Tovar, W. Hillen, and W. Saenger. Structure of the tet repressor-tetracycline complex and regulation of antibiotic resistance. *Science*, 264:418–420, 1994.

[42] A. Hjelmfelt, E. D. Weinberger, and J. Ross. Chemical implementation of neural networks and turing machines. *Proc. Natl. Acad. Sci.*, 88:10983–10987, December 1991.

[43] F. Jacob and J. Monod. *On the regulation of gene activity*, volume 26, pages 193–211. Cold Spring Harbor Symposia on Quantitative Biology, New York, 1961.

[44] M. T. Record Jr., W. S. Reznikoff, M. L. Craig, K. L. McQuade, and P. J. Schlax. Escherichia coli rna polymerase ($e\sigma^{70}$), promoters, and the kinetics of the steps of transcription initiation. In Frederick C. Neidhardt, editor, *Escherichia Coli and Salmonella*, pages 792–821. ASM Press, Washington, D.C., 2 edition, 1992.

[45] T. F. Knight Jr. and G. J. Sussman. Cellular gate technology. In *Proceedings of UCM98: First International Conference on Unconventional Models of Computation*, pages 257–272, Auckland, NZ, January 1998.

[46] D. Kaiser. Regulation of multicellular development in myxobacteria. *Microbial Development*, pages 197–218, 1984.

[47] D. Kaplan and L. Glass. *Understanding Nonlinear Dynamics.* Springer-Verlag, New York, 1997.

[48] S. A. Kauffman. Gene regulation networks: a theory for their global structures and behavior. *Current Topics in Developmental Biology*, pages 145–181, 1971.

[49] S. A. Kauffman. The large-scale structure and dynamics of gene control circuits: an ensemble approach. *Journal of Theoretical Biology*, 44:167–190, 1974.

[50] S. Lisser and H. Margalit. Compilation of *escherichia coli* mrna promoter sequences. *Nuc. Acids Res.*, 21(7):1507–1516, 1993.

[51] H. H. McAdams, April 2001. personal communication.

[52] H. H. McAdams and A. Arkin. Stochastic mechanisms in gene expression. *Proceedings of the National Academy of Science*, 94:814–819, February 1997.

[53] H. H. McAdams and A. Arkin. Simulation of prokaryotic genetic circuits. *Annu. Rev. Biophys. Biomol. Struc.*, 27:199–224, 1998.

[54] H. H. McAdams and L. Shapiro. Circuit simulation of genetic networks. *Science*, 269:650–656, August 1995.

[55] J. Monod and F. Jacob. *General Conclusions: Teleonomic Mechanisms in Cellular Metabolism, Growth and Differentiation*, volume 26, pages 389–401. Cold Spring Harbor Symposia on Quantitative Biology, New York, 1961.

[56] B. Muller-Hill. *The* lac *Operon: A Short History of a Genetic Paradigm.* Walter de Gruyter, 1996.

[57] L. W. Nagel. SPICE2: A computer program to simulate semiconductor circuits. Technical Report ERL Memo. No. UCB/ERL M75/520, University of California at Berkeley, May 1975.

[58] R. Nagpal. Organizing a global coordinate system from local information on an amorphous computer. Technical Report AI Memo No. 1666, MIT Artificial Intelligence Laboratory, 1999.

[59] Frederick C. Neidhardt and Michael A. Savageau. Regulation beyond the operon. In Frederick C. Neidhardt, editor, *Escherichia Coli and Salmonella*, pages 1310–1324. ASM Press, Washington, D.C., 2 edition, 1992.

[60] The Engineering Staff of Texas Instruments Incorporated Semiconductor Group. *The TTL Data Book for Design Engineers.* Texas Instruments Incorporated, second edition, 1981.

[61] A. Orosz, I. Boros, and P. Venetianer. Analysis of the complex transcription termination region of the *escherichia coli rrnb* gene. In *Eur. J. Biochem*, volume 201, pages 653–659, 1991.

[62] A. A. Pakula and R. T. Sauer. Genetic analysis of protein stability and function. *Annual Review of Genetics*, 23:289–310, 1989.

[63] D. A. Parsell, K. R. Silber, and R. T. Sauer. Carboxy-terminal determinants of intracellular protein degradation. *Genes and Development*, 4:277–286, 1990.

[64] M. Ptashne. *A Genetic Switch: Phage lambda and Higher Organisms.* Cell Press and Blackwell Scientific Publications, Cambridge, MA, 2 edition, 1986.

[65] E. Rauch. Discrete, amorphous physical models. Master's thesis, Massachusetts Institute of Technology, 1999.

[66] O. Roessler. *Journal of Theoretical Biology*, 36:413–417, 1972.

[67] O. Roessler. In M. Conrad, W. Guettinger, and M. Dal Cin, editors, *Lecture Notes in Biomathematics 4*, pages 399–418. Springer, Berlin, 1974.

[68] O. Roessler. In M. Conrad, W. Guettinger, and M. Dal Cin, editors, *Lecture Notes in Biomathematics 4*, pages 546–582. Springer, Berlin, 1974.

[69] E. G. Ruby and K. H. Nealson. Symbiotic association of *photobacterium fischeri* with the marine luminous fish *monocentris japonica:* a model of symboisis based on bacterial studies. *Biol. Bull*, 151:574–586, 1976.

[70] J. Sambrook, E. F. Fritsch, and T. Maniatis. *Molecular Cloning: A Laboratory Manual.* Cold Spring Harbor Laboratory Press, Plainview, NY, 1989.

[71] M. A. Savageau. Comparison of classical and autogenuous systems of regulation in inducible operons. *Nature*, pages 546–549, 1974.

[72] F. Seelig and O. Roessler. *Z. Naturforsch*, 27:1441–1444, 1972.

[73] L. F. Shampine and M. W. Reichelt. The MATLAB ODE suite. *SIAM Journal on Scientific Computing*, 18:1–22, 1997.

[74] H. M. Shapiro. *Practical Flow Cytometry.* Wiley-Liss, New York, NY, 1995. Third Edition.

[75] M. A. Shea and G. K. Ackers. The O_R control system of bacteriophage Lambda: a physical-chemical for gene regulation. *Journal of Molecular Biology*, 181:211–230, 1985.

[76] T. F. Smith, J. R. Sadler, and W. Goad. Statistical-mechanical modeling of a regulatory protein: the lactose operator. *Mathematical Biosciences*, 36:61–86, 1977.

[77] A. M. Stevens and E. P. Greenberg. *Cell-Cell Signaling in Bacteria*, chapter Transcriptional Activation by LuxR. G. M. Dunny and S. C. Winans, 1999.

[78] M. Sugita. Functional analysis of chemical systems *in-vivo* using a logical circuit equivalent. *Journal of Theoretical Biology*, 1:415–430, 1961.

[79] M. Sugita. Functional analysis of chemical systems *in-vivo* using a logical circuit equivalent. ii. the idea of a molecular automation. *Journal of Theoretical Biology*, 4:179–192, 1963.

[80] J. R. Swartz. *Escherichia coli and Salmonella: Cellular and Molecular Biology*, volume 2, chapter Escherichia coli Recombinant DNA Technology, pages 1693–1711. ASM Press, Washington D.C., 2 edition, 1996.

[81] S. Swift, P. Williams, and G. S.A.B. Stewart. *N-Acylhomoserine Lactones and Quorum Sensing in Proteobacteria*, pages 291–313. G. M. Dunny and S. C. Winans, Washington, DC, 1999.

[82] R. N. Tchuraev. A new method for the analysis of the dynamics of the molecular genetic control systems. *Journal of Theoretical Biology*, 151:71–87, 1991.

[83] M. Thattai and A. von Oudenaarden. Intrinsic noise in gene regulatory networks. *Proceedings of the National Academy of Science*, 98(8614), 2001.

[84] R. Thomas. Boolean formalization of genetic control circuits. *Journal of Theoretical Biology*, 42:563–585, 1973.

[85] R. Thomas. Logical analysis of systems comprising feedback loops. *Journal of Theoretical Biology*, 73:631–656, 1978.

[86] R. Thomas. Regulatory networks seen as asynchronous automata: a logical description. *Journal of Theoretical Biology*, 153:1–23, 1991.

[87] P. H. von Hippel, T. D. Yager, and S. C. Gill. *Quantitative Aspects of the Transcription Cycle in Escherichia coli*, pages 179–201. Cold Spring Harbor Laboratory Press, Cold Spring Harbor, New York, 1992.

[88] S. A. Ward and R. H. Halstead Jr. *Computation Structures.* The MIT Press, 1990.

[89] R. Weiss, G. Homsy, and T. F. Knight Jr. Toward in-vivo digital circuits. In *Dimacs Workshop on Evolution as Computation*, Princeton, NJ, January 1999.

[90] R. Weiss, G. Homsy, and R. Nagpal. Programming biological cells. In *Eighth International Conference on Architectural Support for Programming Languages and Operating Systems, Wild and Crazy Ideas Session*, San Jose, California, October 1998.

[91] R. Weiss and T. F. Knight Jr. Engineered communications for microbial robotics. In *DNA6: Sixth International Workshop on DNA-Based Computers, DNA2000*, pages 1–16, Leiden, The Netherlands, June 2000. Springer-Verlag.

[92] G. Yagil and E. Yagil. On the relation between effector concentration and the rate of induced enzyme synthesis. *Biophysical Journal*, 11:11–27, 1971.