Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science

Proposal for Thesis Research in Partial Fulfillment
of the Requirements for the Degree of
Doctor of Philosophy

Title: Programming Colonies of Biological Cells

Submitted by: Ron Weiss
59 Athelstane Road
Newton, MA 02459

____
(Signature of author)

Date of submission: January 1999

Expected Date of Completion: January 2000

Laboratory where thesis will be done: Artificial Intelligence Laboratory

Brief Statement of the Problem:

Biological cells possess important characteristics, such as energy efficiency, self-reproduction, and miniature scale, that make them attractive for many programmed applications. Examples include embedded intelligence in materials, sensor/effector arrays, drug and biomaterial manufacturing, smart medicine, and nanoscale fabrication. The goal of this thesis is to demonstrate the feasibility of programming colonies of cells using *in vivo* digital circuits. The research effort will also offer tools for the design and verification of such circuits.

The approach for engineering *in vivo* digital circuits harnesses existing genetic regulatory mechanisms of repression and transcription. Logic signals are represented by the synthesis rates of cytoplasmic DNA binding proteins. Gates are constructed from promoter/operator regions that are regulated by input proteins, fused with structural genes coding for output proteins. I will use a mechanism introduced in this proposal to characterize the behavior of gates built in our laboratory. Once the behavior of several individual gates is quantified, they will be combined in a modular fashion to form more complex circuits. This thesis will provide BioSpice, a prototype genetic circuit simulation and verification tool, to aid in the microbial circuit design process. Finally, because cell colonies present a unique execution environment (e.g. unreliable computing elements), this thesis will also include a prototype high-level colony simulator that aids in designing programs that successfully accomplish tasks using this substrate.

1 Introduction

Advances in biology such as the proliferation of recombinant DNA technology, better understanding of gene function and associated protein interactions, and detection of *in vivo* gene activity using fluorescent proteins have greatly enhanced the field of bioengineering. From a computer science perspective, these enabling technologies also bring promise of programming biological cells to perform complex tasks that involve computation. Biological cells possess important characteristics, such as energy efficiency, self-reproduction, and miniature scale, that make them attractive for many programmed applications. Examples include embedded intelligence in materials, sensor/effector arrays, drug and biomaterial manufacturing, smart medicine, and nanoscale fabrication.

This thesis will examine the feasibility of implementing *in vivo* digital circuits for programming colonies of biological cells. The digital abstraction has proven itself as the dominant form for most programming tasks. Its advantages include reliable computation through noise reduction in successive stages of computation, as well as conceptual simplicity. The digital abstraction is therefore likely to be a useful metaphor for programming biological cells. However, since this substrate possesses different characteristics than silicon based technology, different engineering principles will be applicable. Another underlying goal of this thesis is to explore and use digital and non-digital programming abstractions that cells can perform reliably.

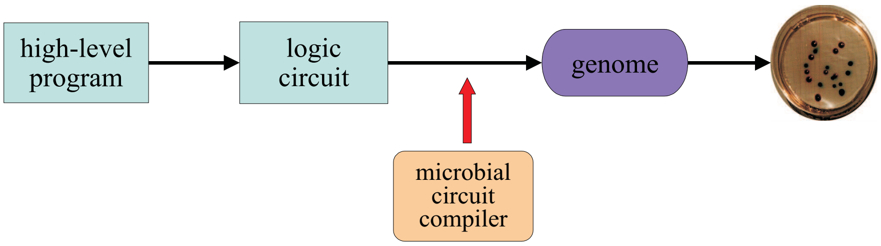


Figure 1: High Level Flow of Microbial Programming, from a high level task description or a logic circuit, to a genetic network whose chemical activity *in vivo* implements that task.

In programming cell colonies, one can devise a task with a logic circuit or use a special purpose *microbial program* that is then compiled to a logic circuit. The *microbial circuit design* process transforms the logic circuit into a genetic network whose chemical activity *in vivo* implements the computation specified by the logic circuit. The design process consults a database of protein reaction kinetics to determine how to combine simple gates into a complex circuit. This thesis will offer prototype tools for the design and verification of genetic circuits, as well as a higher-level simulator for a cell colony. Figure 1 illustrates the high level flow of the microbial programming process.

The approach for engineering *in vivo* digital circuits harnesses existing genetic regulatory mechanisms of repression and transcription. Logic signals are represented by the synthesis rates of cytoplasmic DNA binding proteins. Gates are constructed from promoter/operator regions that are regulated by input proteins, fused with structural genes coding for output proteins.

The most basic element of computation is the inverter, using the design proposed by Knight and Sussman [10]. Each inverter is encoded as a sequence of DNA bases, consisting of an operator (binding site for the input repressor protein), a promoter, and a structural gene coding for the

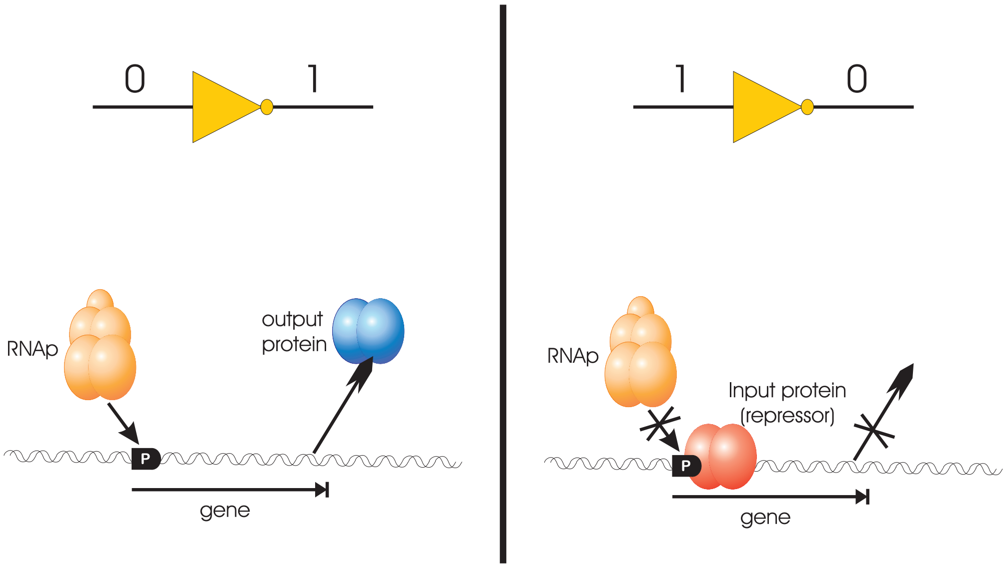


Figure 2: The two idealized cases for a biological inverter. If input repressor is absent, RNAP (RNA polymerase) transcribes the gene for the output protein and enables its synthesis. If input repressor is present, no output protein is synthesized.

output protein. In steady state, as the synthesis rate of the input protein increases, the synthesis rate of the output protein decreases monotonically. Figure 2 illustrates the two idealized cases in the truth table of the biological inverter. By using repressor proteins that cooperatively bind to the operator, the transfer functions of such inverters can be engineered to be sufficiently sigmoidal. To form a NAND gate, and therefore any complex digital logic circuit, the outputs of inverter gates are combined by assigning them same output protein.

Goal and Evaluation

The goal of this thesis is to demonstrate the feasibility of programming cell colonies using *in vivo* digital circuits. In demonstrating this feasibility, it must be first shown that cells have the ability to perform basic digital computation. Assuming the success of the first step, this thesis will then evaluate the proposed approach based on the following:

- **Ability to build experimentally consistent, reliable, and robust gates**: Seemingly identical biological experiments tend to have considerable variations, especially when recording quantitative rather than qualitative results. Therefore, for implementing digital circuits, it is imperative to devise procedures that repeatedly yield similar quantitative results for the same set of experiments. The gates must be robust and able to perform computation reliably in operational cells. Because the substrate is inherently unreliable, it is impossible to guarantee that all cells will carry out the computation as specified. Therefore, the reliability must be quantified. Also, the gates must be efficient at reducing digital noise and handling fluctuations in cellular metabolic behavior.

- **Capability to construct complex circuits and limitations**: In this approach, each gate uses different proteins, and therefore, in contrast to silicon-based technology, the gates will have varying behavioral characteristics. This thesis will include a discipline for complex circuit design by correctly combining gates with different characteristics. Limitations to the complexity of circuits include the amount of extracellular DNA that can be inserted into cells, reductions in a programmed cell's viability, and the selective pressures against programmed cells due to the extra load in performing the computations.
- **Useful programming tools**: Programming tools play an important roleIn any digital circuit design. The tools should help automate circuit design, and must be able to faithfully simulate and verify circuits before they are built. Also, the programming tools must clearly define the execution model of the substrate and provide simulation capabilities that mimic the substrate's characteristics.

Proposed Research

The thesis research will focus on demonstrating the feasibility of *in vivo* circuits by accomplishing the following tasks:

Define the components and abstraction of *in vivo* inversion: An appropriate abstraction of inversion can help reason about the process and help devise genetic modifications that may be required. Also, the abstraction in this proposal partitions the inversion mechanism so that gates are built in a modular fashion.

Simulate gates and small circuits: This proposal presents simulation results for a realistic model of a genetic regulatory mechanism using published data on reaction kinetics. The results suggest that *in vivo* digital circuits can be achieved. However, the simulated transfer function of inversion reveals that modifications to certain components (e.g. promoter and operator) may be required.

Design, build, and measure gates: The overall goal of this phase is to produce consistent, reliable, and robust gates. Gate design includes assembling DNA sequences with promoters, operators that bind repressor proteins, and structural genes coding for output proteins. Once I build gates in the laboratory, I will measure their behavior. This proposal describes a mechanism for measuring the steady state characteristics of gates, and outlines the requirements for measuring the dynamic behavior. Because some genetic modifications may be necessary, I will also construct and quantify the behavior of modifications to the original constructs.

Implement a prototype of the microbial circuit design process: This process combines simple gates to form complex circuits. First, gates must be correctly coupled by matching their digital signal thresholds. In forming circuits, the design process will consult a database of reaction kinetics derived from gate measurements. To validate the microbial circuit design mechanisms, I will build and measure a small circuit using this process.

Offer programming simulation and verification tools: This thesis will offer two simulation and verification tools: BioSpice and the Microbial Colony Simulator (MCS). BioSpice is a prototype system for simulating and verifying genetic digital circuits. It simulates the time-domain behavior

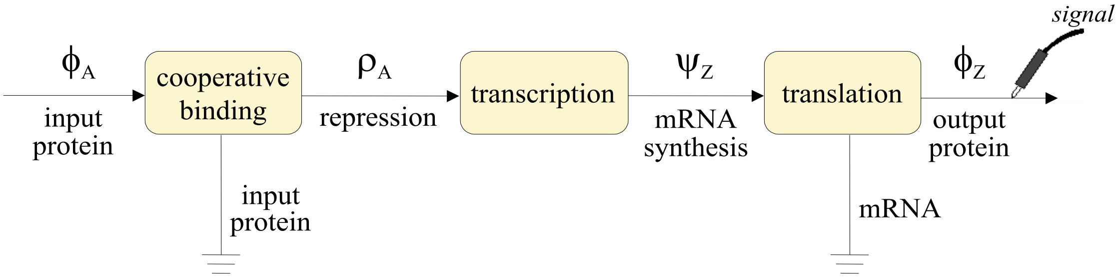


Figure 3: The components that implement a biological inverter: cooperative binding, transcription, translation, and degradation of proteins and mRNA.

of concentration of intracellular proteins and intercellular message passing chemicals for individual cells or very small colonies. MCS targets large scale colony simulations and addresses issues of the program execution environment (e.g. unreliable elements).

2 The Basics of *in vivo* Digital 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 the production of specific proteins. Knight and Sussman [10] propose using this regulatory mechanism to implement digital logic inverters. This section begins to examine the feasibility of using this mechanism to implement *in vivo* digital circuits. The focus is on the basic elements of circuit design: computation (an inverter), connecting gates (ring oscillator), and storage (rs latch).

2.1 Computation: Analysis of an Inverter

2.1.1 Components of Inversion

Natural gene regulation systems exhibit characteristics useful for implementing *in vivo* logic circuits. Figure 3 illustrates how these characteristics can be combined to construct an inverter. Let ϕ_A denote a signal, defined as the synthesis rate of a cytoplasmic DNA binding protein A. The synthesis rate of the input protein A is therefore the incoming signal to the inverter.

The first 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. The proteins are also degraded by the cell, and therefore 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 more input protein is synthesized because the protein concentrations are too low for significant dimerization. Then, at higher levels of ϕ_A (when the input proteins dimerize easily), cooperative binding results 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. The purpose for this stage is to provide gain: as a result of this stage, the analog input signal better approximates its digital meaning.

The next stage involves transcription of the structural gene by the RNA polymerase, and results in the inversion of the signal. Let ψ_Z denote the rate of mRNA synthesis for the output protein 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, the transcription process proceeds at maximum pace (i.e. maximum level of ϕ_Z). Any increase in repression activity results in a decrease in transcription activity, and hence the inversion of the signal.

In the last stage, ribosomal RNA translates the mRNA product into the output protein. The rate of this process is defined as the output signal ϕ_Z of the inverter. Let $\mathcal{L}$ (translation stage) denote the mapping between ψ_Z and ϕ_Z. In general, increases in ψ_Z yield linear increases in ϕ_Z until an asymptotic boundary is reached. This boundary is defined by the protein synthesis capabilities of the cell, as well as other factors such as mRNA stability.

Inversion is accomplished by the combination of the above stages:

$$\phi_Z = \mathcal{L} \circ \mathcal{T} \circ \mathcal{C}(\phi_A)$$

2.1.2 Chemical Reactions that Implement an Inverter

Table 1 presents one possible chemical model of the reactions involved in an inverter (modified from a model by [8]). In particular, this model reflects the characteristics of the lambda CI repressor operating on the lambda $\mathrm{O_R}1$ and $\mathrm{O_R}2$ operators.

The repressor protein A represents the input signal, and protein Z represents the output signal. G_Z denotes the concentration of the *active* form of the structural gene for Z. A structural gene is active only when its associated operator is *unbound* by a repressor. A_2 and Z_2 denote the dimeric forms of A and Z respectively, and $G_Z A_2$ and $G_Z A_4$ represent the repressed (i.e. inactive) forms of the gene. $mRNA_Z$ is the gene transcript coding for Z. RNA_p is RNA polymerase, and $rRNA$ is ribosomal RNA that participates in translation of the $mRNA$. [1]

The model includes the components described in Section 3 in the following reactions: protein dimerization and cooperative binding of the input protein (reactions 1 - 2, 5 - 8), input protein decay (reactions 9 - 10, 13 - 14), transcription (reaction 16), translation (reaction 17), and degradation of the mRNA (reaction 15). The kinetic constants used in this simulation were obtained from the literature describing the phage λ promoter P_R and repressor (cI) mechanism [14, 7].

[1]The simulations in this section assume that the concentrations of RNA_p and $rRNA$ are fixed. Section 3 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.

$$A + A \xrightarrow[\text{dimerization}]{k_{dim(a)}} A_2 \quad (1)$$

$$A_2 \xrightarrow[\text{single}]{k_{sngl(a)}} A + A \quad (2)$$

$$Z + Z \xrightarrow[\text{dimerization}]{k_{dim(Z)}} Z_2 \quad (3)$$

$$Z_2 \xrightarrow[\text{single}]{k_{sngl(Z)}} Z + Z \quad (4)$$

$$G_Z + A_2 \xrightarrow[\text{repress 1}]{k_{rprs(a2)}} G_Z A_2 \quad (5)$$

$$G_Z A_2 \xrightarrow[\text{dissociation}]{k_{dis(a2)}} G_Z + A_2 \quad (6)$$

$$G_Z A_2 + A_2 \xrightarrow[\text{repress 2}]{k_{rprs(a4)}} G_Z A_4 \quad (7)$$

$$G_Z A_4 \xrightarrow[\text{dissociation}]{k_{dis(a4)}} G_Z A_2 + A_2 \quad (8)$$

$$A \xrightarrow[\text{decay}]{k_{dec(a)}} \quad (9)$$

$$A_2 \xrightarrow[\text{decay}]{k_{dec(a2)}} \quad (10)$$

$$Z \xrightarrow[\text{decay}]{k_{dec(Z)}} \quad (11)$$

$$Z_2 \xrightarrow[\text{decay}]{k_{dec(Z2)}} \quad (12)$$

$$G_Z A_2 \xrightarrow[\text{decay}]{k_{dec(ga2)}} G_Z \quad (13)$$

$$G_Z A_4 \xrightarrow[\text{decay}]{k_{dec(ga4)}} G_Z A_2 \quad (14)$$

$$mRNA_Z \xrightarrow[\text{decay}]{k_{dec(rna)}} \quad (15)$$

$$G_Z + RNA_p \xrightarrow[\text{transcribe}]{k_{xscribe}} G_Z + RNA_p + mRNA_Z \quad (16)$$

$$mRNA_Z + rRNA \xrightarrow[\text{translate}]{k_{xlate}} mRNA_Z + rRNA + Z \quad (17)$$

Table 1: Chemical reactions that implement an inverter. A is the input protein and Z the output.

2.1.3 Equilibrium Behavior

Figure 4 shows the equilibrium behavior of the inverter. The graph, which shows the transfer curve mapping ϕ_A to ϕ_Z, was computed by performing a set of simulations. Each simulation included a different steady rate of input protein synthesis, and recorded the level of the output protein if the system settled 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 settled into a steady state contributed a point to the approximation of the transfer curve. The inset illustrates the transfer curve resulting from two such inverters connected in series.

The skewed curve shows that the inverter is very sensitive to low concentrations of the input

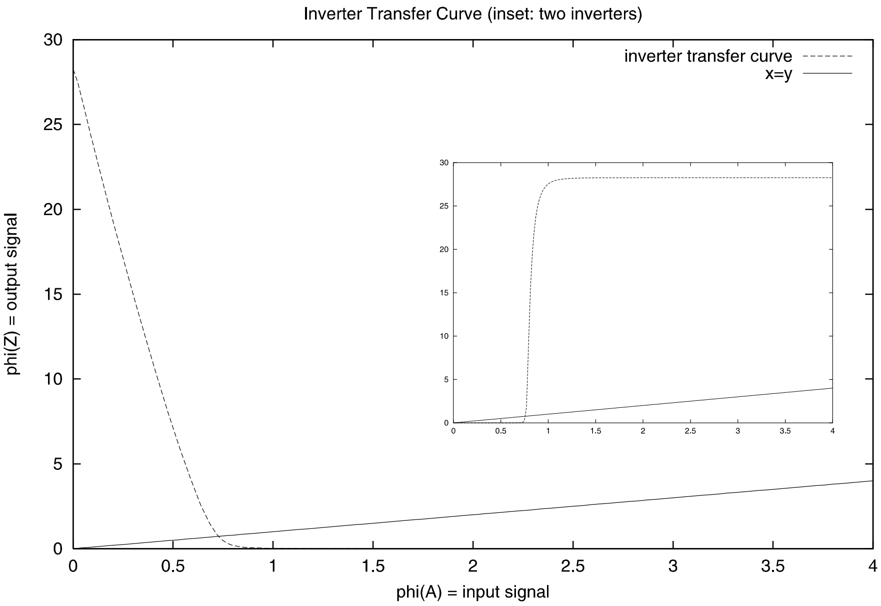


Figure 4: The transfer curve of the inverter, with the transfer curve of two inverters connected in series shown in the inset. Both plots graph ϕ_A versus ϕ_Z.

(repressor) proteins. Thus, for a HIGH output signal, even small fluctuations in the synthesis rate of the input may alter the output signal to LOW. Section 3 describes in detail how to measure the systematic fluctuations and noise in these biological gates. Section 4 then discusses how to modify the behavior of these gates in order to make them more robust.

2.1.4 Dynamic Behavior

Figure 5 shows the dynamic behavior of the inverter as modeled with the above chemical reactions. The three graphs show the concentrations of the input protein A (π_A), the active gene G_Z, and the output protein Z (π_Z). The concentrations include both the monomeric and dimeric forms.

The reactions proceed as follows: First, π_Z increases until it stabilizes when the expression and decay reactions reach a balance. Then, an externally-imposed drive increases π_A. As a result, the concentration of G_Z decreases as A binds free operator. Then, π_Z decreases as no additional Z is synthesized and existing Z is degraded. Finally, the drive π_A decreases, A degrades, G_Z recovers, and π_Z once again reaches the HIGH signal range. Note that the gate switching time (measured in minutes for this mechanism) is governed by the rate of recovery of G_Z.

2.2 Connections: Analysis of a Ring Oscillator

A ring oscillator is a simple circuit that can help determine the utility of our inverters for building complex logic circuits. The oscillator consists of three inverters connected in a series loop. The simulation results in Figure 6 depict the expected oscillation in protein concentrations, as well as

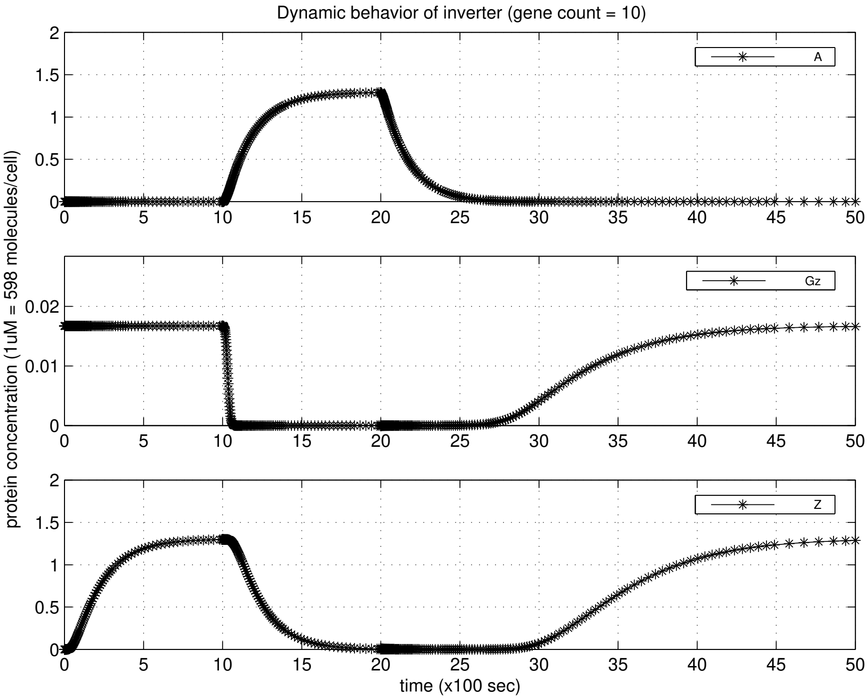


Figure 5: The dynamic behavior of the inverter. The top is the input protein, the middle is the active (unrepressed) form of the output gene, and the bottom is the output protein.

a phase shift between the values. Note, however, that oscillation occurs close to the LOW end of the signal values. This results from the skewed transfer curve that describes the steady state characteristics of the inverter. Sections 3 and 4 discuss this issue in depth.

2.3 Storage: Analysis of an RS-Latch

Another good test circuit is the RS latch, a component for persistently maintaining a data bit. It 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 7). 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. Figure 8 shows the dynamic behavior of this RS latch. As expected, both long and short pulses effectively set and reset the latch.

3 Measuring Gate Characteristics

This section introduces techniques for measuring the behavior of *in vivo* digital gates. Section 3.1 describes how to measure an individual signal in a genetic circuit, by constructing a probe that measures expression activity *in vivo*. A transfer function is the relation between the input signal and the output signal of a gate or circuit in steady state. Section 3.2 introduces a mechanism for es-

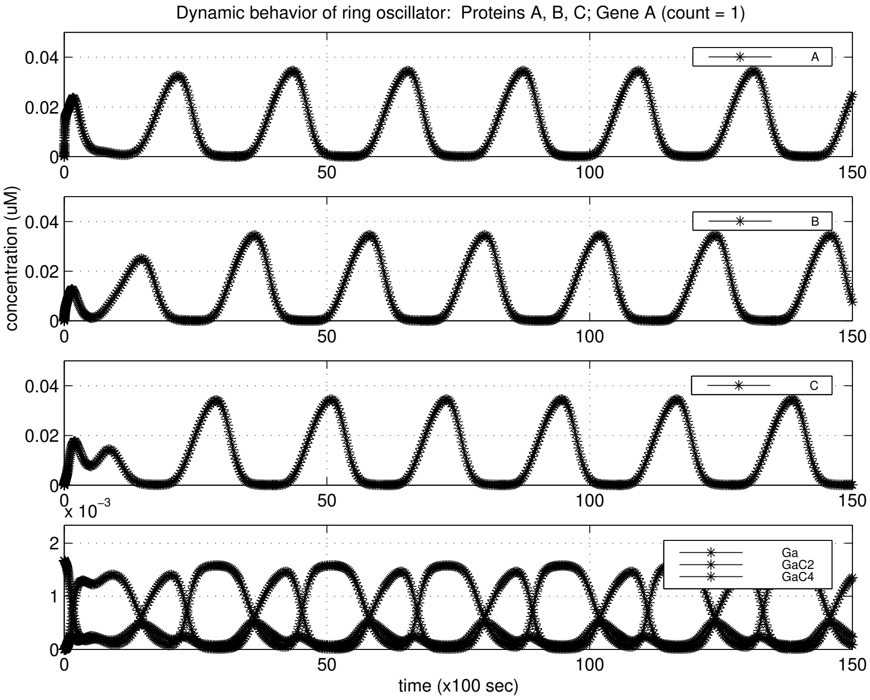


Figure 6: Dynamic behavior of a ring oscillator. The top three curves are the outputs of the three inverters. Note the 120° phase shift between successive stages. The bottom shows various repression states of the first inverter's output gene.

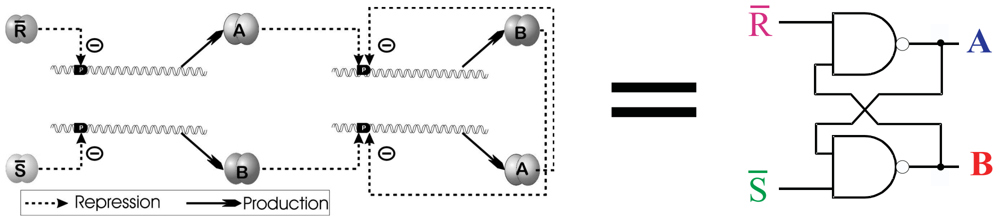


Figure 7: Gene logic and schematic representations of an RS latch, used for storage.

timating the transfer function by measuring many different points of the transfer curve. Section 3.3 discusses how to account for systematic fluctuations and noise inherent in biological systems, by generalizing the transfer function to a *transfer band*. Finally, Section 3.4 briefly describes measuring the dynamic behavior of the gates. I will use the techniques outlined in this section to measure and verify the characteristics of biological logic circuits *in vivo*.

3.1 Measuring a Signal

Recall that ϕ_Z, the synthesis rate of protein Z from all genes coding for it, represents a logic signal. Thus, the relevant chemical reaction for a signal is the rate of translation of the $mRNA$ product of

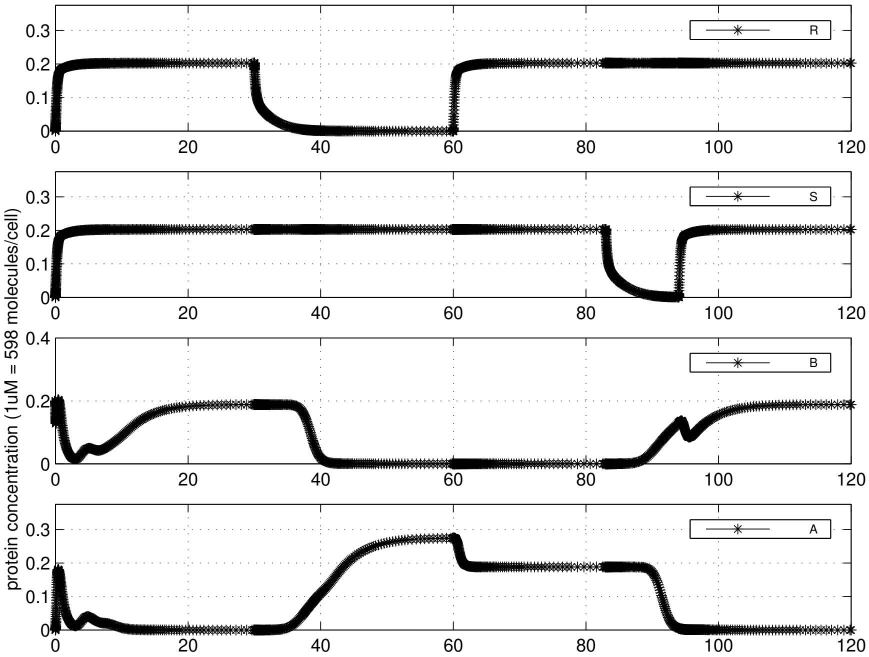


Figure 8: Dynamic behavior of the RS latch. The top two curves are the $\overline{\text{reset}}$ and $\overline{\text{set}}$ inputs, respectively. The bottom two curves are the complementary outputs. The initial behavior shows the system settling into a steady state.

G_Z into the protein product Z:

$$mRNA_Z + rRNA \xrightarrow[\text{translation}]{k_{xlate}} mRNA_Z + rRNA + n \cdot Z \tag{18}$$

G_z is the active (unrepressed) form of the gene, k_{xlate} represents the rate of translation from $mRNA$ into the protein product, and n denotes the cistron count. Then, assuming this is the only production of Z in the system,

$$\phi_Z \equiv n \cdot k_{xlate} \cdot [mRNA_Z] \cdot [rRNA] \tag{19}$$

To measure the signal, insert a reporter protein RP as an additional structural gene, and assume that for the concentrations of interest, RP remains mostly in monomeric form:

$$mRNA_Z + rRNA \xrightarrow[\text{translation}]{k_{xlate}} mRNA_Z + rRNA + n \cdot Z + RP \tag{20}$$

$$RP \xrightarrow[\text{decay}]{k_{dec(rp)}} \tag{21}$$

Then, the time derivative for the reporter concentration is:

$$\frac{d[RP]}{dt} = k_{xlate}[mRNA_Z][rRNA] - k_{dec(rp)}[RP] \tag{22}$$

At equilibrium:

$$0 = k_{xlate}[mRNA_Z][rRNA] - k_{dec(rp)}[RP] \tag{23}$$

Since $k_{xlate}[mRNA_Z][rRNA] = k_{dec(rp)}[RP]$, by substitution into 19,

$$\phi_Z = n \cdot k_{dec(rp)} \cdot [RP] \tag{24}$$

We know n, and can measure $[RP]$ (up to an unknown multiplicative factor) by picking for example a fluorescent protein for RP and measuring its fluorescence. By using the same reporter for each measured signal in the circuit, we obtain approximations of signals all scaled by the same factor.

The assumption that translation is linear (at least for low cistron counts) helps in the measurement process in two ways. First, as described above, one structural gene copy of the reporter can be used to estimate the synthesis rate of n copies of Z from the same gene. Second, in the case where the reporter signal is too weak to be detected reliably, we can use m copies of the reporter to yield a stronger probe signal. Then, the measured concentration of the reporter is divided by m to compute ϕ_Z:

$$\phi_Z = n \cdot k_{dec(rp)} \cdot \frac{[RP]}{m} \tag{25}$$

3.2 Measuring the Transfer Curve of an Inverter

Once an individual signal 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 synthesis rate ϕ_A of the input protein and the synthesis rate 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 multiple systems yielding various fixed values of ϕ_A, and observe the corresponding values of ϕ_Z. Let ${P_D}^j$ represent a constitutive promoter (i.e. "drive") resulting 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 $\left(\frac{{\phi_A}^j}{k_{dec(rp)}}, \frac{\mathcal{I}({\phi_A}^j)}{k_{dec(rp)}}\right)$ can be measured with the reporter RP as described above. This requires two separate experiments, one to measure the drive and one to measure the output. With a set of these points, we obtain the transfer curve of a gate, where all points are expressed in the same (albeit unknown) units.

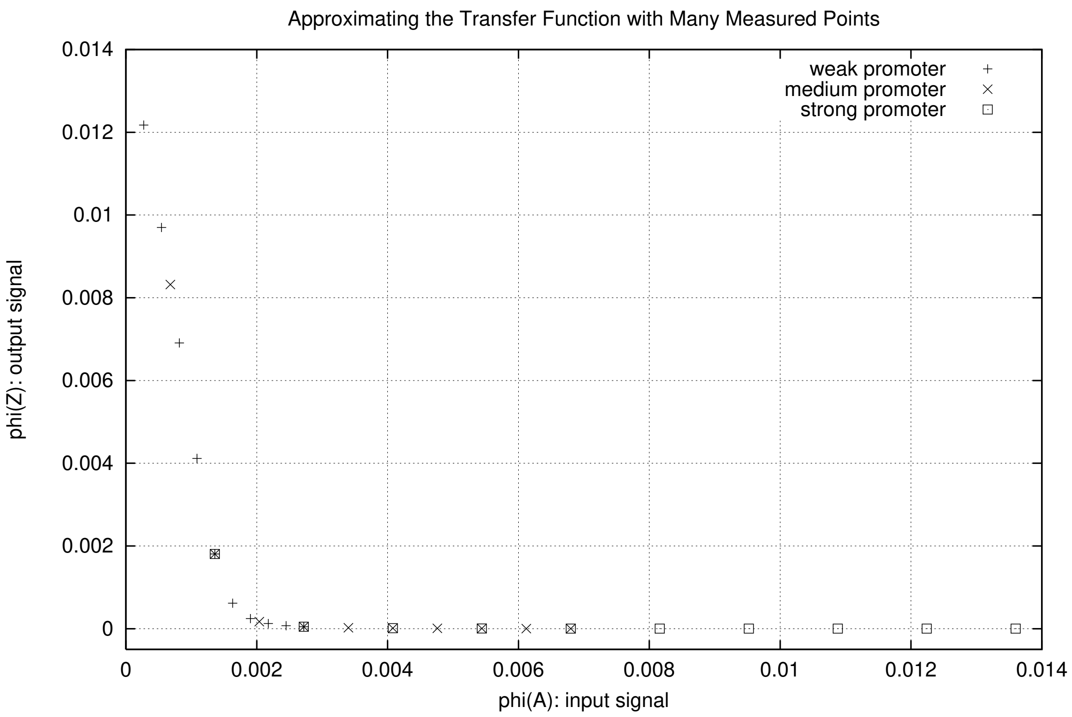
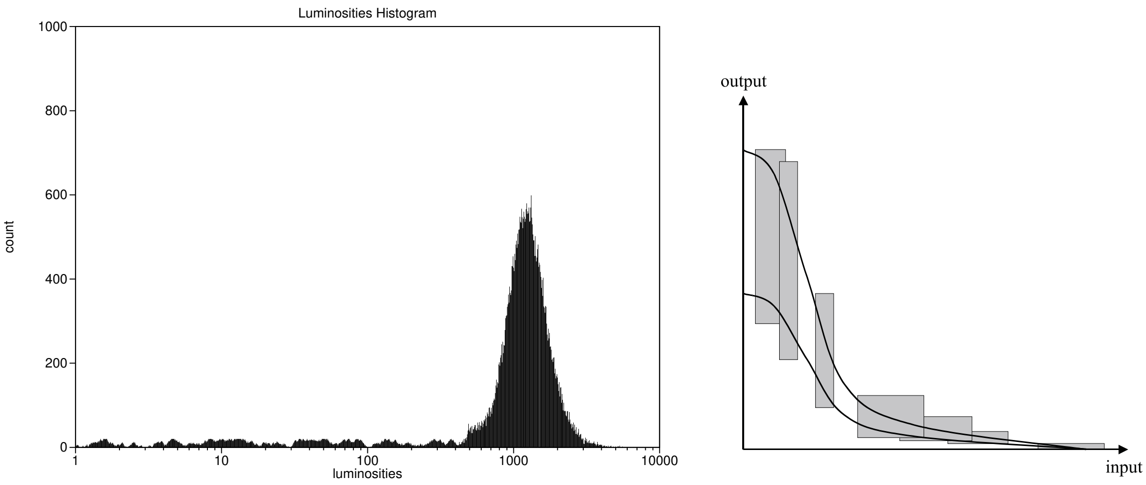


Figure 9: Measuring points on the transfer function of an inverter using three promoters, each with ten different cistron counts.

There are at least three mechanisms for obtaining a variety of drives. First, one can choose different promoters. Second, one can modify the strength of a given promoter through base-pair substitutions, resulting in different transcription initiation rates. Third, for a given promoter, increasing the cistron count of the gene for the drive yields a multiplicative increase in the drive. Figure 3.1 illustrates points on an inverter's transfer curve, obtained by simulating thirty different drives. The simulation computes ten points for each of three different promoters with different $\mathrm{RNA_p}$ affinities. For each promoter, the different points indicate the effect of including between one and ten cistrons.

To measure a complete transfer curve, the range of inputs must cover both the LOW and HIGH input ranges. This will require drives 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. Also, drives with similar characteristics simply add redundancy to the measurements.

To measure more complex circuits, measure the values of the relevant signals by inserting the structural gene for the reporter at the appropriate promoters.



(a) (b)

Figure 10: **(a)** A typical flow cytometry histogram of scaled luminosities, showing one dominant peak. **(b)** Approximation of a transfer band from several distinct drives. The measurement of each drive yields a shaded rectangle. The band lies between the two bold curves.

3.3 The Transfer Band: Models *vs.* Reality

While this simple model yields idealized discrete points on the transfer function, actual biological systems exhibit both systematic fluctuations and noise that are not modeled in Section 2.1. [2] The *transfer band* is a concept capturing these fluctuations. Specifically, the transfer band is the mapping from individual input values to the output value ranges exhibited by viable cells.

Flow cytometry [6] is a technology useful for quantifying gene expression activities of individual cells. First, a gene coding for a fluorescent reporter (*e.g.* GFP) is fused to the same promoter as the gene of interest. Then luminosity values are measured for individual cells as they flow through a cytometer's capillary, yielding a histogram of luminosities. Figure 10(a) shows such a histogram for a large number of "identical" cells (*i.e.* with the same promoter/reporter construct). The variance results from systematic fluctuations in protein expression rates (*e.g.* due to cell growth cycle and environmental factors), and idiosyncratic fluctuations due to stochastic noise in gene expression, measurement error, and non-viable or damaged cells.

The experiments are likely to yield histograms with one clearly dominant peak. For a drive ${P_D}^j$, ${\phi_A}^j$ is now a distribution. Then, let ${\underline{\phi}_A}^j$ be the minimum value of the drive distribution's peak, and ${\overline{\phi}_A}^j$ be the maximum value of that peak. Cells with values in this range are said to be *operational*. In the same manner, let ${\phi_Z}^j$ represent the corresponding distribution of output values, and let ${\underline{\phi}_Z}^j$ and ${\overline{\phi}_Z}^j$ be the minimum and maximum value of the output distribution's peak.

Then the measurement of the input and output distributions for each drive yields a rectangular region with the lower left corner at $\left({\underline{\phi}_A}^j, {\underline{\phi}_Z}^j\right)$ and the upper right corner at $\left({\overline{\phi}_A}^j, {\overline{\phi}_Z}^j\right)$. If the sam-

[2] For example, fluctuations in the RNA_p concentration results in transcriptional fluctuations, rRNA in translational fluctuations, and proteases in decay fluctuations.

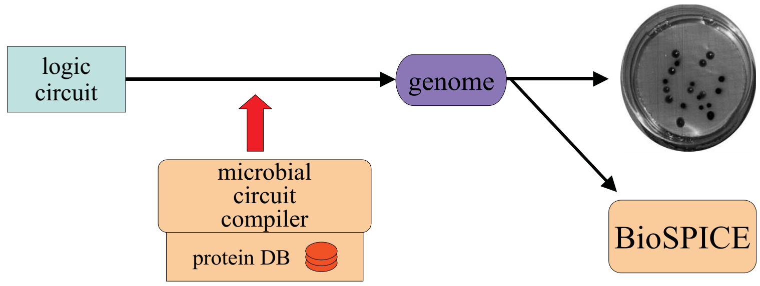


Figure 11: Microbial circuit design components, showing compilation of a logic circuit into a genome that encodes that circuit, and BioSpice, a tool for verifying the resulting genome.

ple density is high, then the transfer band lies within the union of all such rectangles. Figure 10(b) illustrates the approximation of a transfer band from several flow cytometry measurements.

3.4 Measuring Dynamic Behavior

The switching time between LOW and HIGH output signal in response to a change in the input signal defines the dynamic behavior of an inverter. To measure the switching delay of digital gates *in vivo*, I plan to use either laser scanning cytometry [11] or image microscopy. These technologies record luminosity values for individual cells over time. Note that for accurate measurements of dynamic behavior, it is important to use reporters with short half lives, such as destabilized GFP.

As with the steady state behavior, the dynamic behavior of gates is likely to fluctuate both for a given individual cell over time and across different cells. The experiments will therefore quantify and record these variations. One possible experiment is a ring oscillator. This experiment's advantage is that it does not require any external stimulus to produce interesting dynamic behavior. Another possible experiment is introducing a transmembrane chemical, such as IPTG, into the medium, and then measuring the lag in switching the output signals of downstream gates connected to the output of the lac promoter (which is induced by IPTG).

4 Microbial Circuit Design

Microbial circuit design creates a biological digital circuit using a small set of basic gates and a database of protein reaction kinetic rates (Figure 11). 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 design process requires searching the database and assigning suitable proteins to each gate. Because the gates use different proteins, their static and dynamic characteristics will vary. Section 4.1 describes the necessary conditions for matching the threshold levels of connected inverters. Section 4.2 then discusses how to modify the characteristics of the inverters to help satisfy the threshold constraints. Section 4.3 briefly describes some of the issues in building circuits from simple gates whose behavior has been measured.

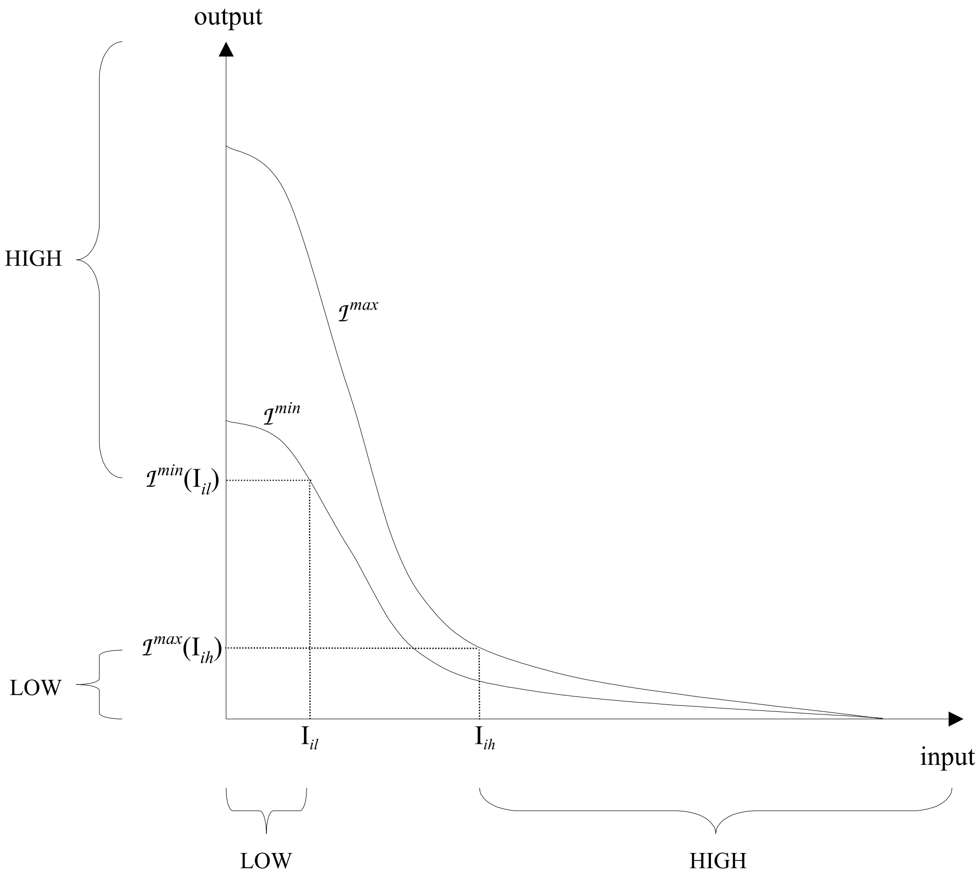


Figure 12: HIGH and LOW input ranges for a hypothetical inverter. The transfer band is defined by the two curves.

A genome encodes the entire digital circuit with a set of genes, each with its own operator, promoter, and structural genes. Normally this genome construct will be implemented on one or more extracellular DNA strands (e.g. plasmids). Before inserting the plasmid into biological cells, the genome construct should be checked by a tool such as BioSpice (Section 4.4), which is a prototype simulator for verifying genetic digital circuits.

4.1 Matching Gate Thresholds

Transfer functions suitable for implementing digital gates must have LOW and HIGH ranges such that signals in the LOW range map strictly into the HIGH range, and *vice versa*. The strictness of the inclusion reduces noise from input to output. For electronic digital circuits, the LOW and HIGH signal ranges are the same for all gates because the circuit is composed of transistors with identical threshold voltages, spatially arranged. However, in biological digital circuits, the gate components (proteins) 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 appropriately matched.

A given transfer band can be defined by a pair of functions. As shown in Figure 12, let $\mathcal{I}^{min}$ be the function that maps an input to the minimum corresponding operational output, and let $\mathcal{I}^{max}$ be the function that maps an input to the maximum corresponding operational output.

If I_{il} and I_{ih} are the input thresholds, then the suitability condition given above can be written as:

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

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

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

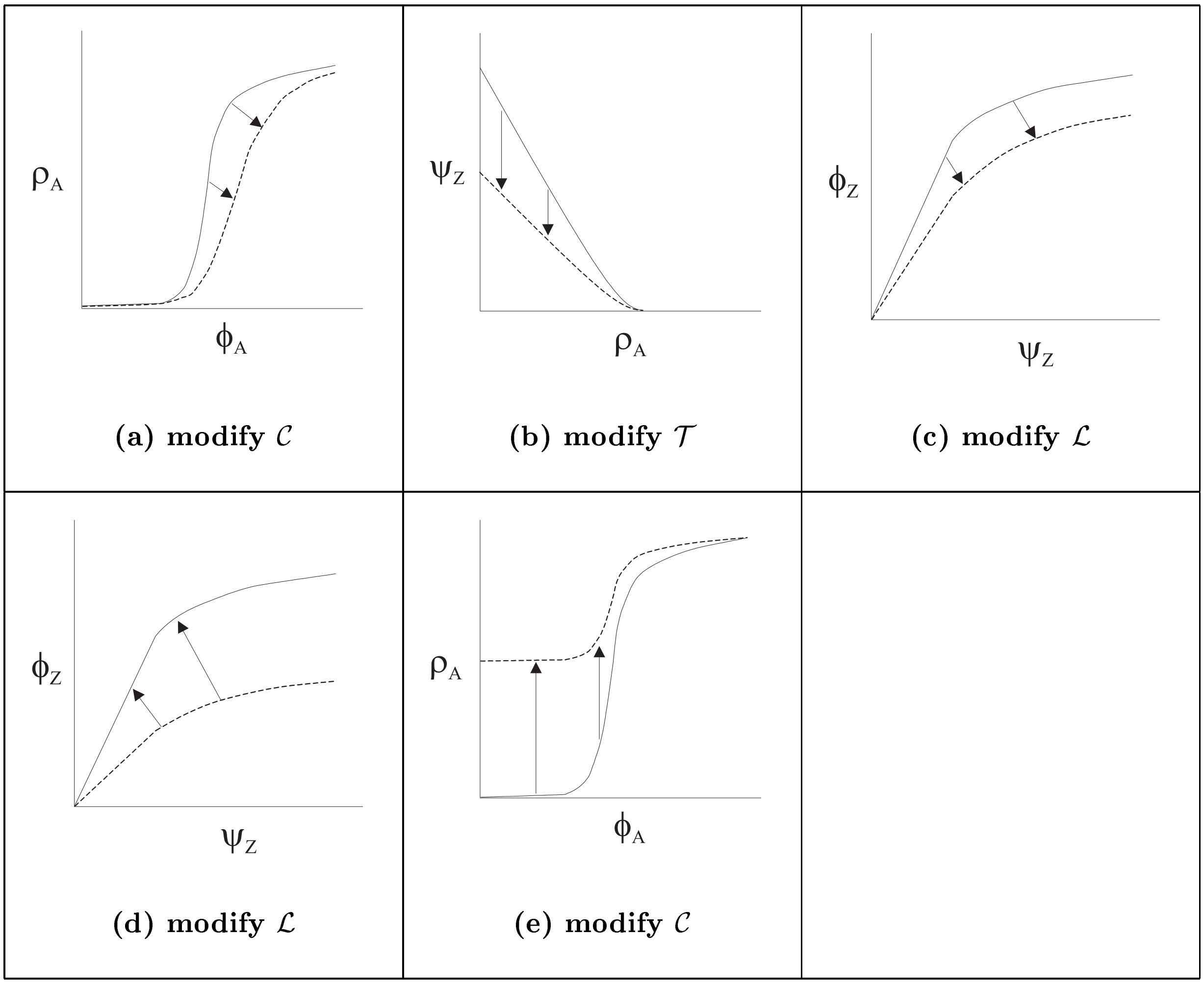
Then the following conditions are necessary and sufficient for correct coupling:

$$\begin{aligned} \mathcal{J}^{min}(\mathrm{J}_{il}) &> \mathrm{I}_{ih} \\ \mathcal{J}^{max}(\mathrm{J}_{ih}) &< \mathrm{I}_{il} \end{aligned}$$

4.2 Modifying the Inverter Characteristics

The first step in developing the microbial circuit design process is to design, build, and characterize several inverters. It is likely that these inverters will not match correctly according to the definitions above. Fortunately, there are techniques to adjust them so they are matched for use in complex circuits. These include:

- **Modifying the repressor/operator binding affinity:** This can be accomplished via base pair substitutions at the operator site. A reduction in the strength of repression alters $\mathcal{C}$, the cooperative binding mapping from the input signal ϕ_A to ρ_A. Figure 13(a) illustrates this effect on the cooperative binding stage of inversion. The substitutions will result in different behavior with each repressor/operator pair. Figure 4.2 shows the effect of hypothetical reductions in the repression affinity on the transfer function of an inverter.
- **Modifying the strength of the promoter:** This can be accomplished via base pair substitutions at the promoter site, which alters the affinity of the RNA polymerase to the promoter. DNA sequence determinants of promoter strengths have been studied extensively [5, 16, 9]. Figure 13(b) illustrates an example of a reduction in promoter strength on $\mathcal{T}$, the transcription stage mapping between ρ_A and ψ_Z. Figure 4.2 shows the effect of several such reductions on the transfer functions of an inverter and two inverters in series.



(a) modify $\mathcal{C}$ **(b) modify $\mathcal{T}$** **(c) modify $\mathcal{L}$**

(d) modify $\mathcal{L}$ **(e) modify $\mathcal{C}$**

Figure 13: 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

- **Modifying the strength of the ribosomal binding site (RBS):** This can be accomplished via base pair substitutions at the RBS, which alters the affinity of rRNA to the mRNA. Figure 13(c) shows the effect of a reduction in rRNA affinity on $\mathcal{L}$, the translation stage mapping between ψ_Z and ϕ_Z.

- **Increasing the cistron count:** This is accomplished by adding copies of the coding sequence for the output protein to the structural gene. Figure 13(d) shows the effect of doubling the cistron count on $\mathcal{L}$, the translation stage mapping between ψ_Z and ϕ_Z. The graph values double in the linear range of the translation.

- **Altering the degradation rate of a protein:** This can be done on a per-protein basis by changing the few amino acid residues on the C terminus [3, 12, 13]. The modification of input protein A will change $\mathcal{C}$, the cooperative binding stage mapping between ϕ_A and ρ_A. Another possibility for increasing the degradation rate is to choose bacterial strains that have

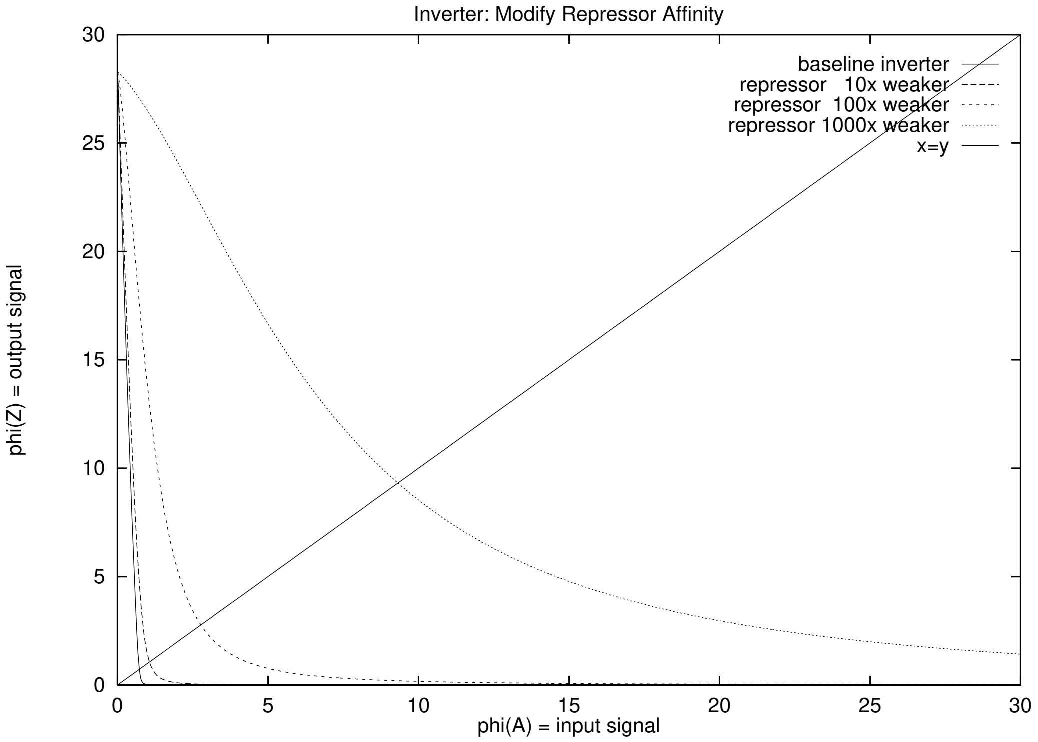


Figure 14: The effects of reducing the binding affinity of the repressor to the operator on the transfer function of an inverter.

high concentrations of effective proteases [15].

- **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 ideal because they are much more sensitive to fluctuations in LOW input signals versus fluctuations in HIGH input signals. Autorepression will reduce the maximum output protein synthesis rate and therefore will reduce some of the asymmetry in the signal sensitivities. Figure 13(e) shows the effect of adding autorepression on $\mathcal{C}$, the cooperative binding stage of inversion.

4.3 Building a Circuit

A kinetics database will contain the measurements obtained using the mechanism described in the Section 3. If the inversion mechanism does not use autorepression, then the transfer function is defined by the input protein, operator, promoter, and RBS, but not by the output protein (i.e., a relation $Z = \mathcal{I}_1(A)$ does not depend on Z). This implies that gate couplings are independent of the output protein. To compute the transfer function of two gates in a series, $Y = \mathcal{I}_2(Z)$ to

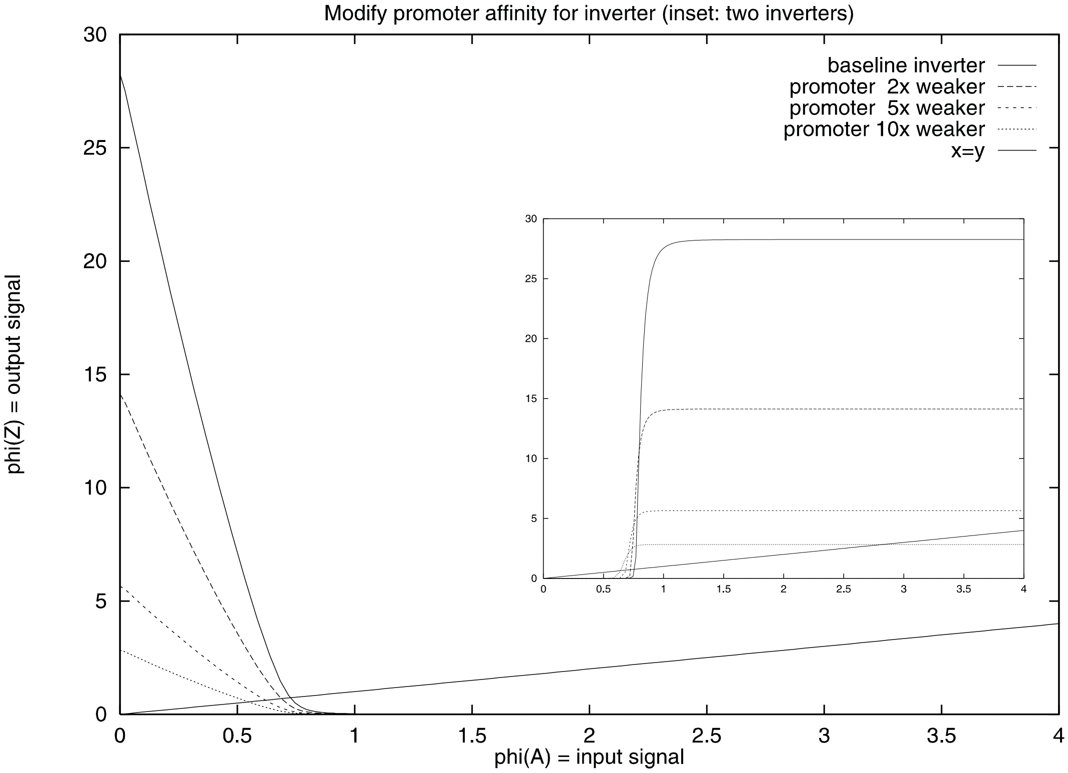


Figure 15: The effects of reducing the binding affinity of the RNA polymerase on the transfer functions of an inverter. Inset shows the effects on the transfer functions of two inverters in series. The diagonal lines correspond to input equals output.

$Z = \mathcal{I}_1(A)$, the database only needs to contain $* = \mathcal{I}_2(Z)$ and $* = \mathcal{I}_1(A)$, where $*$ denotes any protein. However, if the inversion uses autorepression, then a relation $Z = \mathcal{I}_3(A)$ also depends on the characteristics of Z. Then, to compute the transfer function of any coupled gates $Y = \mathcal{I}_4(Z)$ to $Z = \mathcal{I}_3(A)$, the database must specifically include the transfer functions of the input/output protein pairs Y/Z and Z/A.

Using the measurements, the suitability of transfer function to implement the digital abstraction can be 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. In the same manner as done for an individual inverter, a series of inverters can also be 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 4.1. 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. I plan to investigate mechanisms that will help automate the matching process using the measurements database. For example, an evaluation metric that considers gain and noise margins can be defined for a transfer function. Then, this metric will serve as the evaluation function in an automated search process for finding protein configurations that

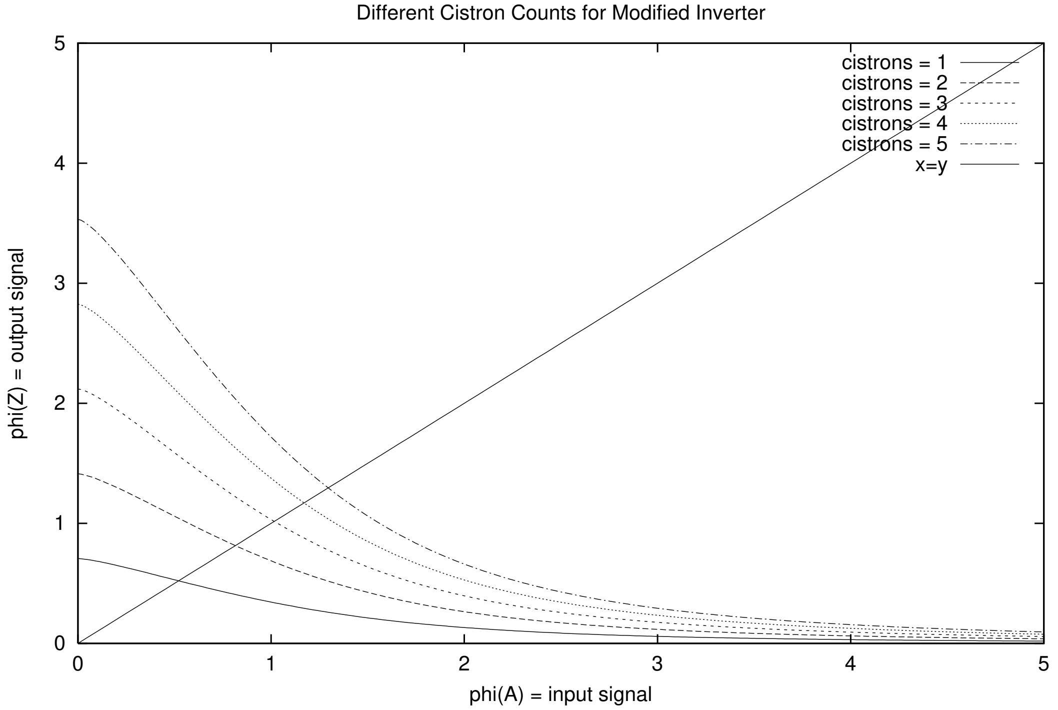


Figure 16: The effects of increasing the cistron count per gene (i.e. the number of structural genes per operator region). In this case, the original inverter mechanism has been modified to 100x less binding affinity of the repressor and 40x less binding affinity of the promoter.

achieve satisfactory levels of gain and noise margins.

4.4 BioSpice

BioSpice is a prototype system for simulating and verifying genetic digital circuits. It takes as inputs a network of gene expression systems (including the relevant protein products) and a small layout of cells on some medium. BioSpice then consults a database of reaction kinetics and diffusion rates in order to simulate the dynamic characteristics of the target system. The simulation computes the time-domain behavior of concentration of intracellular proteins 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 17 represents a genetic digital 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. This can be achieved with certain signal transduction pathways. The presence of A results in controlled synthesis of C. Notice that the gate with input A can be chosen or adjusted to be sensitive to even small quantities of A. Once a sufficient concentration of A accumulates, C

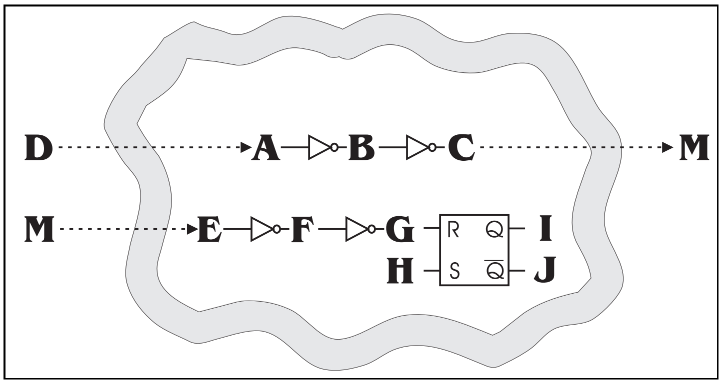


Figure 17: Gate level representation of a genetic circuit to accomplish a simple bacterial task.

is synthesized and secreted into the surrounding environment as protein M. M diffuses through the medium and serves as a message to neighboring cells. In response to M, the neighbors each set their RS latch, whose output is I.

Figure 4.4 shows a BioSpice simulation of the above system on a 4×4 grid (representing the medium) with two bacterial cells (heavily shaded squares). The initial condition, depicted in the top-left snapshot, shows that the output of the RS latch (represented by I) is LOW. Then, a drive D is introduced into the environment next to one of the cells, as illustrated in the top-right snapshot. This causes the cell to transmit a message M. Once the other cell receives M (recognizable by the presence of E) it uses G to set the RS latch. Finally, when the drive is removed and the message M decays, the value of I remains latched at HIGH.

The BioSpice prototype will also address issues such as cell division and fluctuations in rRNA concentrations, RNA polymerase concentrations, and plasmid copy numbers. The gates designed to operate *in vivo* must be robust enough so that they function correctly despite all these systematic fluctuations.

5 The Microbial Colony Computation Paradigm

The discussion so far has focused on achieving digital logic circuits within biological cells. While the digital abstraction reduces noise in computation and simplifies programming constructs, one must consider the execution environment when actually programming cell colonies. 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 colonies of cells that execute in parallel and coordinating their actions through intercellular communication, significant computational power can be achieved. The goal of the Amorphous Computing project [1] is to develop novel paradigms for programming such substrates.

This section defines the *microbial colony language* (MCL), an achievable computation paradigm

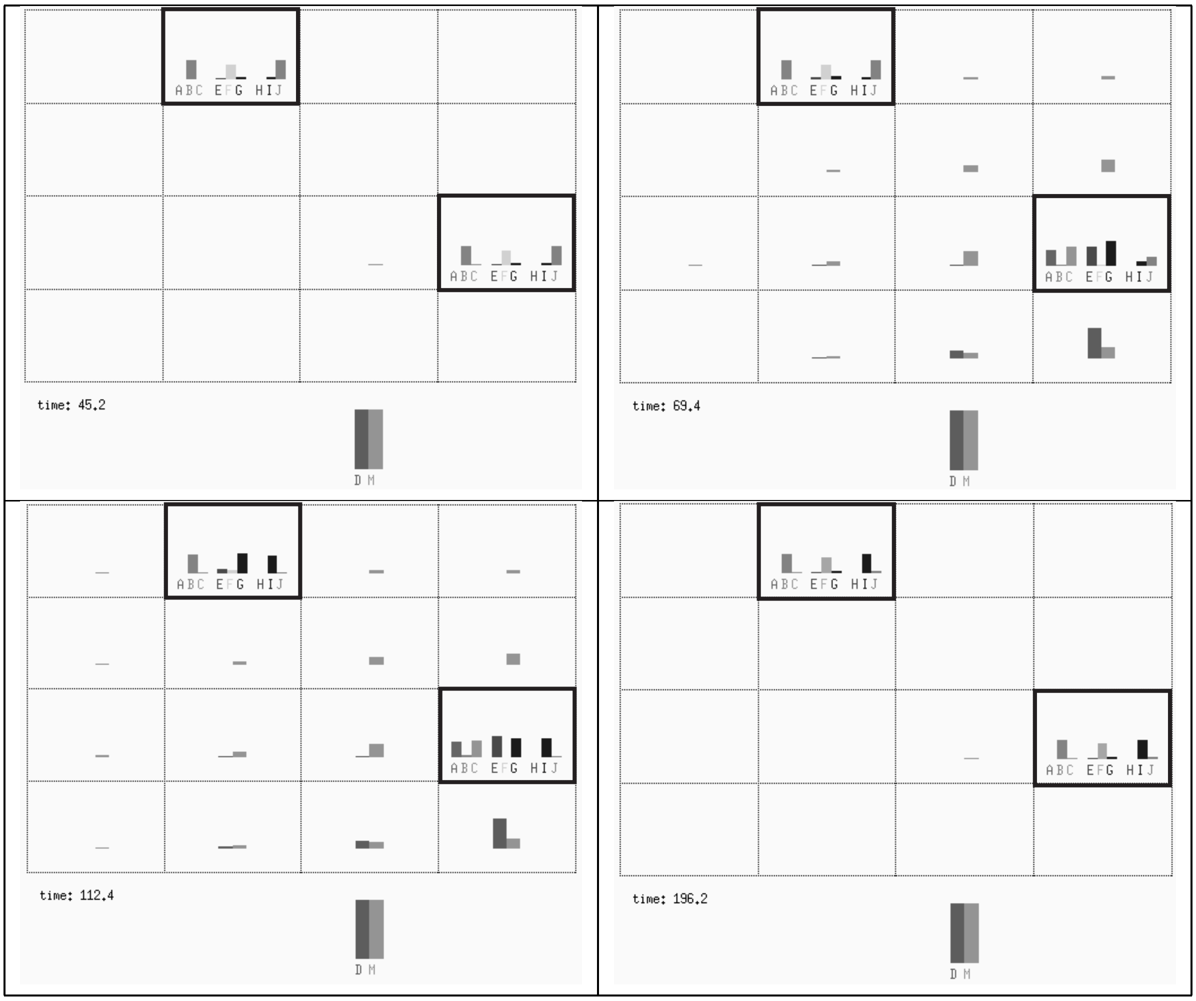


Figure 18: BioSpice simulation snapshots of intracellular protein and intercellular message chemical concentration for the simple task described in this section. The shaded rectangles represent cells, and the rest of the grid represents the medium. The vertical bars denote different levels of protein concentrations.

for programming colonies of biological cells. This section also describes MCS, a simulator for this language. MCL is simple enough for biological cells, yet expressive enough to implement 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.

The language exposes programming mechanisms that biological 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 simpliciy of language will likely enable programs written in MCL to be executed in cells (by using asynchronous *in vivo* digital circuits). I have implemented a language-level simulator that models cell colonies executing microbial programs. Simulations show that these programs can produce large scale pattern generation and coordinated group behavior.

5.1 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 broadcast 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 a marker decay. Rules are triggered by events and satisfaction of boolean expressions of markers. If multiple rules are enabled by a single action, then their activation order 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. A message is uniquely identified by the message tag and the arguments. 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, hopcounts 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)

5.2 Language-Level Simulator

BioSpice cannot simulate large cell colonies because the computation resources required are too great. I therefore implemented the Microbial Colony Simulator (MCS) on top of *hlsim* [2] and Coore's event-driven layer [4] that translates MCL program rules into an amorphous event-driven simulation. Because MCS simulates at a higher level than BioSpice, it is appropriate for simulating colonies with a large number of cells. Figure 5.2 shows a series of simulation snapshots for a simple MCL program that generates large scale patterns. Each circle represents a computation element, and the shading of the circle represents the particular state of the element.

6 Conclusion and Research Plan

This document outlines an approach for demonstrating the feasibility of programming colonies of biological cells using *in vivo* digital circuits. Simulation results suggest that *in vivo* digital circuits can be achieved, although the simulated transfer function of inversion reveals that modifications to certain components (e.g. promoter and operator) may be required. This proposal describes a mechanism for measuring the characteristics of gates and discusses the elements of microbial

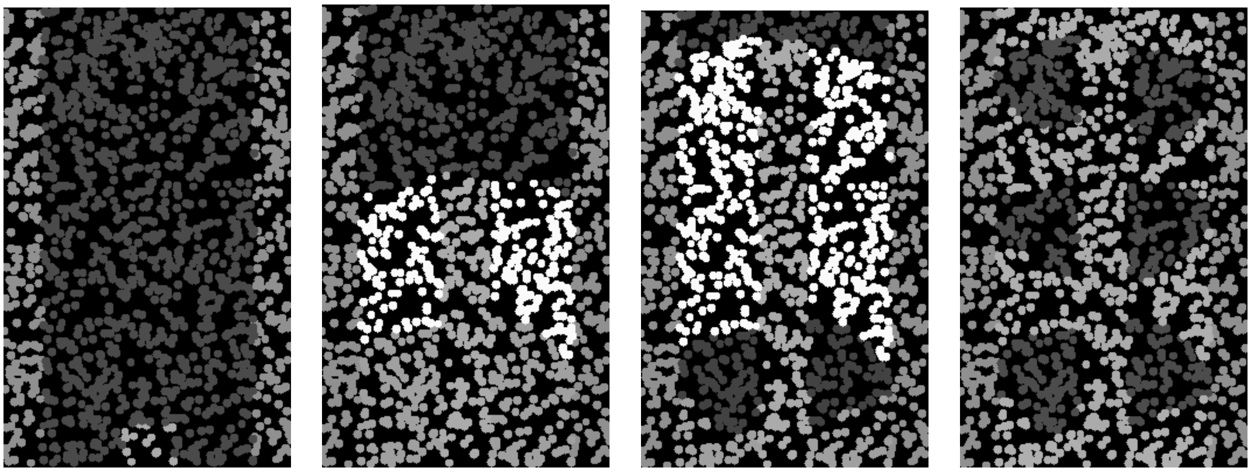


Figure 19: Simulation snapshots of pattern generation using a simple program written in MCL.

circuit design. Finally, it describes BioSpice and MCS, two tools that will aid in programming for this substrate. The research plan for completing the thesis and achieving its stated goal can be partitioned into three categories:

Theoretical work: To complete a sufficient theoretical underpinning for building complex circuits, I will formalize the evaluation of gate couplings and devise a mechanism for selecting and matching simple gates to form complex circuits. I will also formalize a mechanism for measuring the dynamic behavior of gates.

Biological experiments: I will build and measure the behavioral characteristics of gates and small circuits *in vivo*. The laboratory work will proceed as follows: measure the transfer function of a single gate, measure the transfer functions of several gates whose operator and promoter have been modified from the original gate, measure the transfer functions of gates built from other repressor/operator complexes, select several gates that satisfy the gate matching criteria to build and measure a small circuit, measure the dynamic characteristics of the above constructs.

Software infrastructure: I will concentrate on providing three main components of the software infrastructure necessary for programming cells. The first component is a microbial circuit compiler that constructs genetic networks implementing logic circuits. The second component is BioSpice, and the third component is the Microbial Colony Simulator (MCS).

6.1 Thesis Outline

The following is a proposed outline for the final version of the thesis:

1. Introduction
2. Related Work

3. Background and Approach
4. Building Biological Gates
5. The *in vivo* Behavior of Gates and Small Circuits
6. Microbial Circuit Design
7. BioSpice
8. The Microbial Colony Computation Paradigm
9. Conclusion and Future Work

6.2 Milestones

Here is an approximate schedule of the expected date of milestones in the completion of the dissertation:

1999	
January	Finish the definition of the microbial circuit design process.
February	Build an inverter, measure its static behavior.
March	Build modified inverters, measure their static behavior.
April	Build inverters using other proteins and measure.
May	Build a simple circuit (e.g. three inverters in series) and measure.
June	Measure dynamic behavior of above gate constructs.
July	Finish first implementation of BioSpice.
November	Write up first draft of thesis and submit for comments.
2000	
January	Defense and submission of completed dissertation.

References

[1] Hal Abelson, Tom Knight, and Gerry Sussman. Amorphous computing. *White paper*, October 1995.

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

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

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

[5] David E. Draper. Escherichia coli and salmonella. chapter Translational Initiation, pages 902–908. ASM Press, Washington, D.C., 2 edition, 1992.

[6] Alice Longobardi Givan. *Flow Cytometry: First Principles.* Wiley-Liss, New York, 1992.

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

[8] George E. Homsy, July 1998. personal communication.

[9] M. Thomas Record Jr., William S. Reznikoff, Maria L. Craig, Kristi L. McQuade, and Paula J. Schlax. Escherichia coli and salmonella. chapter Escherichia coli RNA Polymerase (Eσ^{70}), Promoters, and the kinetics of the steps of transcription initiation, pages 792–821. ASM Press, Washington, D.C., 2 edition, 1992.

[10] Thomas F. Knight Jr. and Gerald Jay Sussman. Cellular gate technology. In *Unconventional Models of Computation*, pages 257–272, 1997.

[11] L. A. Kamentsky, D. E. Burger, R. J. Gershman, L. D. Kamentsky, and E. Luther. Slide based laser scanning cytometry. *Acta Cytol*, 1(41):123–143, January 1997.

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

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

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

[15] James 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.

[16] Peter H. von Hippel, Thomas D. Yager, and Stanley C. Gill. In *Transcriptional Regulation*, chapter Quantitative Aspects of the Transcription Cycle in Escherichia coli, pages 179–201. Cold Spring Harbor Laboratory Press, Cold Spring Harbor, New York, 1992.