INVESTIGATING GENETIC CIRCUIT FAILURES

by

Pedro Fontanarrosa

A dissertation submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Biomedical Engineering

The University of Utah

December 2022

The University of Utah Graduate School

STATEMENT OF DISSERTATION APPROVAL

The dissertation of **Pedro Fontanarrosa**
has been approved by the following supervisory committee members:

Chris J. Myers ,	Chair(s)	**9/9/2022** Date Approved
Tara Lynn Deans ,	Member	**8/29/2022** Date Approved
Orly Alter ,	Member	**8/29/2022** Date Approved
Tamara Carla Bidone ,	Member	**8/31/2022** Date Approved
Yuval Dorfan ,	Member	**9/17/2022** Date Approved

by **David W. Grainger** , Chair/Dean of
the Department/College/School of **Biomedical Engineering**
and by **David B. Kieda** , Dean of The Graduate School.

ABSTRACT

Synthetic biology is an engineering discipline in which biological components are assembled to form devices with user-defined functions. As a nascent discipline, genetic circuit design is reserved only for experienced researchers with an in-depth knowledge of biology. This work aims to alleviate some of these constraints by developing software to facilitate genetic circuit design and analysis so that more researchers can participate in this thriving discipline and help elucidate the causes of circuit failures.

Firstly, an automatic dynamic model generator can be implemented to predict a circuit's behavior between steady states and determine the amount of time needed to reach such steady states. Moreover, the analysis of the predicted dynamic behavior will help the designers understand the risks of applying specific input changes and decide whether the risk is critical for the designed systems' intended purposes. Extrinsic and intrinsic noise can contribute to the observed output variability of a clonal population. Therefore, to account for a genetic circuits' stochastic behavior, this work aims to develop stochastic modeling using extrinsic and intrinsic noise contributions that can help infer glitch *probabilities* and elucidate the causes of circuit failure.

All the methodologies developed in this work will serve the overacting aim of redesigning genetic circuits to avoid circuit failures. Dynamic ODE modeling will predict glitching behavior and the time to reach said states; stochastic modeling will be used to predict glitch propensities; hazard-preserving transformations will be used to avoid solvable hazards.

Facilitated dynamic modeling of genetic circuits would be an instrumental technique for synthetic biologists, especially if it can be accompanied by a circuit design automation tool, such as those proposed in this work. This would help automation in synthetic biology and provide a way to debug circuit designs before construction and compare predictions with experimental data once the synthesized circuit is implemented, saving time, effort, and money. This project aims to expand such capabilities to help researchers through

the design process with the development of automated modeling, logic synthesis, hazard identification, and genetic circuit redesign.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

GRN Genetic Regulatory Network

ODE Ordinary Differential Equation

CCK Classical Chemical Kinetics

SSA Stochastic Simulation Algorithm

FBA Flux Balance Analysis

RPU Relative Promoter Units

a.u. arbitrary units

DNA Deoxyribonucleic Acid

RNA Ribonucleic Acid

RNAP RNA Polymerase

mRNA messenger RNA

RBS Ribosome Binding Site

CDS Coding Sequence

TF Transcription Factor

YFP Yellow Fluorescent Protein

GFP Green Fluorescent Protein

Ara Arabinose

aTc anhydrotetracycline

IPTG $Isopropyl\beta - D - 1 - thiogalactopyranoside$

HSL oxohexanoyl-homoserine lactone

BE β-estradiol

Dox doxycycline hyclate

EL Early-Lag

LL Late-Lag

EE Early-Exponential

ME Mid-Exponential

LE Late-Exponential

S Stationary

DBT Design-Build-Test

DBTL Design-Build-Test-Learn

DBTS Design-Build-Test-Scale

OLC Optimal Laboratory Conditions

OTLC Outside-the-Laboratory Conditions

GDA Genetic Design Automation

CAD Computer-Aided Design

SBML Systems Biology Markup Language

SED-ML Simulation Experiment Description Markup Language

SBOL Synthetic Biology Open Language

OMEX Open Modeling EXchange format

VPR Virtual Parts Repository

SVP Standard Virtual Parts

SBO Systems Biology Ontology

SO Sequence Ontology

UCF User Constraint File

DSGRN Dynamic Signatures Generated by Regulatory Networks

PSA parametric sensitivity analysis

CHAPTER 1

INTRODUCTION

Synthetic biology is a multi-disciplinary area of research that attracts a variety of researchers [22]. The *de novo* approach to the construction and design of artificial biological systems [143] is what separates synthetic biology from classical genetic engineering. To do so, researchers strive to imprint engineering principles into classical genetic engineering. Some of those principles, like standardization and part characterization, are well on their way to development. Others, like decoupling, where a complicated problem is separated into simpler independent problems, are harder to instill in this area of research. However, given the complex and unpredictable nature of biological systems [22], synthetic biology engineering conveys a whole new range of challenges that need to be overcome in order to have reliable and reproducible systems.

As a nascent discipline, genetic circuit design is reserved only for experienced researchers with a deep knowledge of biology. To a large degree, this is due to the inherent complexity of biological systems [97]. However, as the field advances, so does the development of software that allows for more researchers to participate in synthetic biology. A key aspect of any engineering discipline is the ability to model and simulate designs. Modeling and simulation allows researchers to predict the intended behavior of a designed system's outcome, confirming correct behavior before building the circuit in the laboratory, which takes time and money. Furthermore, comparing predicted outcomes with empirical data can reveal unknown part interactions or other biological phenomena not previously studied [95]. As synthetic biology advances and genetic circuits become more complex, so does modeling. This complexity, in turn, makes the modeling process an even more difficult challenge, and, therefore, incentivizes the development of tools for easy and automated modeling of genetic circuits. However, a critical advantage of modeling is the ability to predict problems, malfunctions or undesired transient behavior in biological

systems. Predicting a system's potential for malfunctions can provide researchers an opportunity to redesign a system to avoid such problems or implement it with knowledge of the system's flaws. Debugging is especially valuable in synthetic biology, since it can take a long time and be expensive to genetically engineer an organism with a synthetic genetic circuit; or if a genetic circuit's output is a toxic pharmaceutical or produces an irreversible effect in another system. As we move from proof-of-concept designs to more real-life applications of designed biological systems [18], this process becomes ever more critical for the safe operation of genetic circuits.

1.1 Synthetic Biology's Promise

The goal to engineer DNA as others would engineer electric circuits is what has attracted interest amongst a diverse group of researchers, from molecular biologists to computer scientists, to develop engineering methods for biology. Though most of the development is done by synthetic biologists with substantial expertise, there has been a general effort to produce a more automated method to design genetic circuits for researchers without extensive knowledge of genetic design. Scientists have tried to implement foundational technologies that would make synthetic biology a genuine engineering discipline, where three of the most relevant methods to implement are *standardization*, *abstraction*, and *decoupling* [48]. Standardization is not only the use of data standard representations of genetic circuits and models for reproducibility of results, but also standards for the definition, description, and characterization of modular and reusable genetic parts [48, 82]. Abstraction is used for simplifying mathematical models, which reduces the effort of describing different genetic parts and reactions mathematically and, consequently, increases the number of components that can be effectively modeled and simulated efficiently [143]. And finally, decoupling is the effort to separate a complicated problem, like engineering a synthetic organism with a specific function, into different, modular problems that can be worked independently. For synthetic biology, this would be decoupling designing and modeling constraints from the building technicalities of a synthetic genetic circuit. In other words, people not intimately familiar with the construction process can still design systems that work. [120, 143]. Standardization, abstraction, and decoupling are essential for model-based design of genetic circuits [120] and computer-aided design.

Applying engineering principles to biology is not something new, but many agree that a distinct biological engineering discipline, synthetic biology, started with the modeling, construction, and testing of two unique genetic circuits, a genetic toggle switch and a synthetic oscillatory network [46, 60]. There was something very particular about these two circuits: their function, rather than their output, was what interested the researchers. These circuits stemmed from mathematical models and concluded in design, which established a precedent where the design and construction of engineered *Genetic Regulatory Networks* (GRNs) were facilitated by theory with predictive capacity. Nonetheless, there were some discrepancies between the theoretical model and the experimental results, as expected. Our understanding of how genetic circuits behave is reflected in the precision of our models, and thus any differences between predicted and observed results can be used as a tool to study further the dynamics of genetic networks [64, 76, 114]. This unique way of designing circuits has led to increased developments in modeling for GRNs (see Section 1.3) and of a library of orthogonal genetic parts with characterized behavior. This library is a key component for an automated procedure for the design of genetic circuits for synthetic biology, as well as its subsequent analysis, which would provide a more detailed design/build/test/scale pipeline, as described in section 1.5. Such increased variety of models and library of orthogonal genetic parts has pushed model-based design of genetic circuits [31, 70], as well as tools for *Computer-Aided Design* (CAD) of genetic circuits, like Cello [129]. As the need to design ever more complex genetic circuits increases, the demand for CAD tools that can design, model, and simulate GRNs becomes more critical. Therefore, model-driven design that clearly identifies design, build, and test phases are becoming common in the life sciences as they have been in other engineering fields [134].

1.2 Real Life Applications and Circuit Failures

Synthetic biology's potential for "out-of-the-lab" applications, such as bioproduction, biosensing, therapeutic, and probiotic delivery, has not progressed as it was expected despite the discipline's development in research [18]. Furthermore, if genetic circuits are going to be designed for industrial, health or environmental applications, these will require special consideration, particularly if the outcome production of these circuits would have irreversible effects. First, more detailed models that can simulate dynamic

behavior is needed, and, second, further analysis of how/why these circuits can fail are critical for the safe operation of genetic circuits in real-life applications. There is little to no work done on genetic circuit failures [16, 19], since most research publications are centered on the correct and expected behavior of genetic circuits. However, as we intend to move towards more real-life applications, there is a greater need to study and be able to predict genetic circuit failures and deviation from expected behavior [18]. The purpose of this work is to further these objectives in order to design more reliable genetic circuits, or at least be aware to its shortcomings.

Designing genetic circuits with an expected behavior can be challenging, especially if one considers all the potential sources of failure. A genetic circuit can fail to perform the originally designed function for a number of reasons, but being able to predict, understand and work with these shortcomings is a huge benefit. For combinational circuits, including combinational GRNs, input changes can cause unwanted switching variations in the circuit's output. Unwanted signal transitions occur when the system has not reached a steady state, and the output signal varies from the expected behavior. In some cases, this variance is harmless or well-tolerated. For example, these unwanted signal transitions should not be a major concern if the output of the circuit is only sampled when the circuit has reached a steady state. Nevertheless, this glitching behavior can have drastic consequences if it causes an irreversible change. For example, causing a cascade of responses, inducing apoptosis, or inappropriately releasing a toxic pharmaceutical. Therefore, for the safe operation of a genetic circuit, avoiding such unwanted variations in a circuit's output can be crucial. When moving from *proof-of-concept* genetic circuit designs to real-life applications intended for industry or health, not only steady-state failures must be analyzed and accounted for, but also transient circuit failures. For this purpose, modeling is critical.

1.3 Modeling

Genetic circuits have reached such a degree of complexity that even experienced researchers have a difficult time considering all the interactions between parts of a system. Therefore, mathematical descriptions of genetic networks become a necessity and that is why genetic design is usually model-driven [134]. It is from these mathematical descriptions that one can model the system as a whole and obtain predictions of its behavior.

The simulations and subsequent analyses can become instrumental not only to study the nature of genetics, but also to expose errors in design, parametrization, or the model itself.

Modeling of GRNs with appropriate use of parameters is expected to yield indispensable contributions and aid in complex genetic design, furthering the advancement of synthetic biology. It is common to find abstract mathematical models describing GRNs that use nonspecific (or generic) parameters obtained from the literature. The inaccuracy of many models stem from the use of these nonspecific parameters because even if the predictive accuracy of a model fits observation for a specific organism, using these parameters for other organisms/systems would produce inaccurate results [39], and thus it is essential to have a model generator that is accompanied by the correct set of parameters. Therefore, a model generator that works with a popular set of parameterized genetic gate repository is needed to help facilitate design/build/test of genetic circuits and reproducibility in the synthetic biology area of research. However, to model and analyze circuit failures like function or logic hazards, set-up, and/or hold-state failures, one needs to use dynamic modeling (see Section 3). There are many model generators, but very few that make use of and predict genetic circuit temporal dynamics. Modeling and analysis of circuit failures can provide the designer with different options to consider: either going back to the drawing board if a design does not work as expected for critical computations; or use the design *as is* but using it cautiously while knowing of the design's shortcomings.

Biological systems are highly complex due to the uncountable number of interactions and interconnectivity, which makes the prediction of behavior almost impossible without the use of models and simulations. Even though mathematical models and correct parametrization can help simplify the design process, experimentation and comparison of the results are imperative to correct the models, fine-tune the parameters, and unearth unknown interactions. Computer simulations using mathematical models can help scientists understand the biological mechanisms and unknown phenomena [95], as well as help to bridge the gap between predictions and experimental results, denoting previously missing experimental data.

As the synthetic biology community deepens its knowledge of genetic interactions, more sophisticated modeling tools can be created to predict their behavior with higher accuracy and fidelity. However, there is a drawback in these types of models if the number

of equations and parameters used to describe the system becomes overwhelming. To reduce complexity, there are different assumptions and simplifications that can be made to abstract the model and simplify it without losing predictive capabilities or accuracy. Abstraction can help scientists produce more complex and novel genetic designs for various applications in the industry [143]. Some studies have done this not only by simplifying the equations, but also by combining and redesigning genetic components into bigger, composite parts to reduce variability and increase accuracy [48, 129].

Many different approaches have been developed to model and simulate genetic regulatory systems [40, 77, 79, 97, 119, 148] with different focuses and aims in mind and each having different advantages/disadvantages. It has been shown that kinetic modeling is an appropriate way to model genetic regulatory networks [102, 119]. Starting with a kinetic model, a common way to describe GRNs is by utilizing *Ordinary Differential Equations* (ODEs), which represent the rate of change of species and concentrations of the system. For a system of ODEs, a steady-state model assumes that all the rate equations are in equilibrium, effectively removing time from the model and focusing on the stable states of the system. Instead, when using quasi-steady-state assumptions, a model can produce different time points, and, as such, effectively predict information on the concentration of molecular species over time. Dynamic modeling, as it is called, has a significant advantage for modeling the dynamics of the circuits and their transition states. Dynamic modeling provides a more detailed description of GRN dynamics that can be used to determine circuit failures, to optimize for speed of transitions or cell source-allocation, and to better understand how the different genetic components interact. Since dynamic modeling can predict transition states, it can also be used to determine unwanted transition states between different stable-states (also called *hazards*) due to gate propagation delay in that circuit. Therefore, an automatic model generator that produces dynamic models that use standardized parameters from a well characterized genetic gate library is essential to facilitate genetic circuit design and implementation and promote reproducibility.

1.4 Standards

A critical piece of the puzzle for model-based design is the use of globally accepted standards. Standards not only allow for the sharing and contrasting of knowledge, but

also tackle one of the most significant obstacles facing any engineering discipline today: the problem of reproducibility [81, 166]. For synthetic biology, reproducibility not only requires the use of standards for modeling and simulation but also for experimental setups, DNA design, and parametrization. Furthermore, the use of data standards is necessary to allow for data exchange through different online data repositories and a higher sharing of knowledge amongst different research groups. For these reasons, this work uses data standards for the genetic designs, the mathematical models that describe their dynamics, and their simulation environments so that anyone can reproduce the results obtained here. Data standards used in this dissertation are the *Synthetic Biology Open Language* (SBOL) [56] for the representation of genetic designs and their function; the *Systems Biology Markup Language* (SBML) [26, 72] for the mathematical model representation of the different genetic circuits and their interactions; and the *Simulation Experiment Description Markup Language* (SED-ML) [165] for the simulation description of the mathematical models.

Standardization is important not only to represent data, but also for different methods like gate parametrization, circuit failure analysis, noise modeling and many other facets of the *Design-Build-Test-Learn* (DBTL) cycle in synthetic biology. Efforts placed upon this goal would help reproducibility, re-usability of experimental results and help designers design and debug their systems. This dissertation proposes a standard of representation for genetic parts and their parametrization by following the architecture of the circuits designed in Cello [129], to better model and predict genetic circuit failures.

1.5 Contributions

This work builds upon previous work, specifically [54], to further expand the modeling, simulation and prediction capacities, as well as analysis of circuit failures, for software tools developed to help researchers design, build, and debug reliable genetic circuits. In particular, this dissertation provides the following contributions:

- Implemented an automated dynamic modeler for genetic networks using a more abstracted model that allows for more accessible characterization of genetic parts and helps detect genetic circuit failures between steady-states.
- Developed a hazard analysis method for genetic circuits and redesigned a circuit layout to avoid logic hazards.

- Analyzed the effect of noise as a source of a circuit's output variability and its effect on genetic circuit failures.
- Collaborated with the circuit failure analysis and redesigning genetic circuits for bacteria and yeast.

An automatic dynamic modeler has been developed that allows a researcher to predict a circuit's outcome in between states, and also simulate production over time. This not only helps with the detection of circuit failures in-between steady-states, but also predict the time needed to reach a particular steady-state (which might be part of the design as shown in Chapter 3). Furthermore, with the dynamic model generator of this work, designers will also be able to model and predict unwanted switching transitions or failures of a genetic circuit. The ability to simulate and predict potential circuit failures will help designers debug genetic circuits for incorrect circuit outputs or unwanted switching behavior and either go back to the drawing board or implement the circuit with knowledge and understanding of its shortcomings.

Model-driven design is of paramount importance when designing highly complex genetic circuits [25, 77, 134]. Modeling is instrumental in showing faults in the genetic design, understanding of underlying biological processes, and the dynamical transition stages of a genetic circuit and potential glitches in the system. However, devising a model for genetic circuits can be a tedious and complicated endeavor. Additionally, parametrization is usually lacking for different models, thus making a model inaccurate. Limited availability of reaction rate constants and other parameters is a major hindrance for the development of accurate models, and therefore, for model-driven design [77]. In this respect, lower model resolution or abstraction of different parameters into more generalized parameters is an advantage as it requires less characterization in the laboratory and less detailed understanding of the regulatory mechanisms that underlie a GRN [48, 77, 129, 152]. The proposed model, parametrization, and gate composition used in this work provide an abstracted design to reduce part characterization and facilitate modeling. Dynamic modeling is not something new in synthetic biology; however, the dynamical model used in this work is adapted to use parameters that reflect a popular parametrization and gate composition repository, used by Cello [129], a popular CAD tool in synthetic biology. If adopted by other research groups, this modular design would

help create a more prosperous and more diverse library of genetic parts to use with this work's automated model generator. This can help expand the variety of genetic circuits designed and also open a new standard for other CAD tools to implement when designing genetic circuits. The proposed gate composition also provides a more straightforward, more abstract way to parameterize gates and simplify experiments to do so, as it requires fewer parameters than other characterizations. Therefore, if it would be accepted as a standard for genetic gates, part re-usability across different laboratories would grow, and cooperation would then be facilitated.

Modeling and simulation give the capacity to predict a system's outcome and detect problems or malfunctions during the design process. This debugging capability is especially valuable in synthetic biology since it can be time-consuming and expensive to genetically engineer an organism with a synthetic genetic circuit. Empirical data that differs from predictions can also shed light on part interactions or other biological phenomena not previously studied [95]. The simulations and subsequent analyses can become instrumental not only to study the nature of genetics, but also to expose errors in design, parametrization, or the model itself. Therefore, this work also presents a method for analyzing and predicting various circuit failures for combinational genetic circuits. Though glitches have been observed for genetic circuits [112, 129], this phenomenon's causes are not so well understood as in the asynchronous electronic community [118]. This work can help introduce awareness of the subject to the synthetic biology community and automate much of the analysis needed to understand genetic circuits' glitching behavior. It is important to understand how and why genetic circuits fail, and more importantly, what is the probability of that failure. This work also explores ways to circumvent or avoid certain circuit failures. Even if a circuit failure is unavoidable, knowledge of the presence of that probability is of critical importance for a genetic circuit designer, especially as more circuits are designed for *out-of-the-lab* purposes [18]. The analysis and prediction of glitches is of paramount importance for the safe operation of a genetic circuit for medical purposes. If the output of a genetic circuit is a toxic pharmaceutical, or causes irreversible effects on the cell, then understanding, predicting and avoiding circuit failures is critical for the biomedical engineering community.

In addition, this work expands the accountability of variability in genetic circuits due

to different sources of noise. This is especially important for genetic circuits intended for real-life applications [18], since outside of a controlled laboratory environment, sources of variability (as noise) increase substantially. This noise could potentially increase the probability of the genetic circuit failures analyzed in this work, and thus it is important to account for the different sources of noise, their level of incidence in a circuit's output. Furthermore, analyzing a genetic circuit's output with different levels of noise can help designers determine the general "robustness" of a genetic circuit.

Facilitated dynamic modeling of genetic circuits would be an instrumental technique for synthetic biologists, especially if it can be accompanied by a circuit design automation tools, as the one proposed in this work. This would help automation in synthetic biology and provide a way to debug circuit designs before construction and compare predictions with experimental data once the synthesized circuit is implemented. This project aims to expand such capabilities to help researchers through the design process with the development of automated modeling, logic synthesis, hazard identification, and genetic circuit redesign.

An implicit, iterative *Design-Build-Test* (DBT) process is often used to develop these ingenious genetic circuits [1]. However, bias is introduced into the DBT process in almost all of its steps and the variability of environmental factors that affect a circuits' behavior is often not taken into account. This might hinder a circuit's expected performance when applied in *Outside-the-Laboratory Conditions* (OTLC). Models used by *Genetic Design Automation* (GDA) tools are mostly based on experiments carried out under *Optimal Laboratory Conditions* (OLC) [2, 75, 129]. Furthermore, most rely only on the expression of a fluorescent protein as an output reporter under OLC. This setup leads to an inaccurate *Scale* step with regard to the actual circuits' performance when applied in OTLC that can produce erroneous or faulty behavior with unpredictable outcomes. Furthermore, with a narrow *Test* step, the *learning* usually is limited to a post-hoc description of circuit dynamics. This would be especially perilous for engineered systems that are aimed to operate in dynamic environments, such as living therapeutics and whole cell biosensors.

This work applies a broader *Test* step to a designed delay-signal circuit to include more environmental dynamic factors and reporters (as shown in **Figure 1.1**). The circuit's output, as well as the time for output detection, were observed to be highly variable

for different temperatures, mediums, inducer concentration, bacterial growth-phases, and output reporters. If the performance of the delay circuit is compromised by the tested experimental factors presented here, it will inevitably alter its behavior in other contexts, which would not have been predicted by GDA tools.

We propose to introduce a *Scale* step as part of a new and improved *Design-Build-Test-Scale* (DBTS) process. Scaling refers to the process of considering the variability of factors that can affect genetic circuit performance in real-life applications. Most studies either have a non-existent *Scale* step, or it consists only of a post-hoc description of the designed circuit's performance at OLC. This work not only provides a re-parametrization effort for different experimental conditions, but also produces a new model to determine the necessary predictions for untested conditions. As a case study, we focused on the effect of growth phase on the circuit's output, in which we observed a trend in delay and total output production. This, in turn, allowed for a deeper *Scale* step, which ultimately resulted in a new model that estimates these trends, thus enabling the capacity to predict untested delays and output production of the circuit, which can be further applied for scaling.

Thus, we propose that a greater emphasis in the *Test* and *Scale* steps of a DBTS cycle are needed to build more predictive models and to reduce bias across the entire DBTS cycle. This, in turn, will enable the possibility of finding design alternatives to any unexpected behavior and performance when the circuits are used in applications, improving a genetic circuit's robustness [170]. As we move from proof-of-concept designs to more real-life applications, a thorough *Test* step provides the necessary data that allows for a significant *Learn* step and, therefore, an appropriate *Scale* step.

This work should not only greatly facilitate the design and construction of more complex genetic circuits, but also serve as a tool to spot failures in design as well as unexpected interactions with the host organism. Additionally, comparing the model results with experimental data can help to understand the underlying biological phenomena, and can the researcher's intention. This bridges the gap between experimentalists and designers as it helps both sides with the results obtained. Designers can use data to better fit the model to produce more accurate predictions, and experimentalists can use these predictions to debug genetic circuits and predict their behavior before constructing them, saving time,

effort, and money.

Ultimately, we hope that methods and standards developed in this work will propagate through the synthetic biology community, standardizing hazard analysis, noise simulation, characterization experiments, and parametrization methods to bridge the current gap in the DBTS pipeline.

1.6 Dissertation Overview

Chapter 2 goes further describing the necessary background to set the context of this work. The chapter will start by expanding upon the discipline of synthetic biology as an interdisciplinary research area and focus mainly on the topics of which are more relevant for this work. First, it introduces genetic combinational circuits (Section 2.1.1), their uses, and some examples of their applications in synthetic biology. This chapter continues to explain how these genetic circuits are constructed from well-defined and characterized genetic parts, and how synthetic biologists are exploring new parts and creating a library of orthogonal components so that others can use them in their designs (Section 2.5). These parts repositories are built to be shared amongst researchers, and thus this chapter also explains the importance of standards (Section 2.4), and why it is so imperative to develop globally accepted synthetic biology standards, as this fosters not only reproducibility but also sharing of knowledge between different labs for the greater advancement of science. This chapter also introduces the reader to the mathematical modeling of genetic regulatory networks (Section 2.3), specifically the Hill-equation based models that have been used extensively for genetic circuit design (Section 2.6), and in particular for the Cello project (Section 2.6.1); uses of data standards and online repositories in synthetic biology; and, finally, an introduction to GDA tools.

Chapter 3 presents a new dynamic model for genetic circuits and an automated generation procedure to automatically generate dynamic models. This chapter goes into detail of how our procedure handles the different parameters and gene dynamics to construct a mathematical model and, ultimately, simulate it to get meaningful results. This work expands on previous automatic model generation procedures [54] by adding the ability to simulate between steady-states and roadblocking for tandem promoters, both explained in detail in the chapter. This will allow to more accurately predict a genetic circuit's outcome

through time (not only states), and, therefore, allow a designer to predict possible genetic circuit failures (or unwanted outcome production) *in between* steady-states.

Chapter 4 describes glitches due to hazards and other circuit failures for genetic regulatory networks, how these can be detected and simulated, and how some of these failures can be avoided. This chapter also expands on other circuit failures like set-up and hold-state failures (see Section 1.2). The chapter finishes with some circuit failure analysis on genetic circuits with different layouts for the same function.

Chapter 5 adds a deeper layer to circuit failure analysis, by adding uncertainty and stochasticity to the simulations as "noise". It expands on the different sources of noise that can have an effect on the expected outcome of a genetic circuit, and how these noise sources could affect the probability of a genetic circuit failure occurring.

Chapter 6 uses the sources, software and analysis tools, and workflows developed in Chapters 3, 4, and 5 to simulate, predict, and redesign genetic circuits with certain applications in mind. This chapter not only will serve as a proof-of-concept for the utility of the work developed in this dissertation, but also to show the sparked collaborations and usages in real-life cases of genetic circuit design.

Finally, Chapter 7 presents a summary of our results, workflow and conclusions, for the use and development of the work done in this dissertation. It also expands on the future venues this research can develop and how these will contribute to the design of genetic circuits and their safe (or at least informed) application for medical, industrial, or environmental purposes.

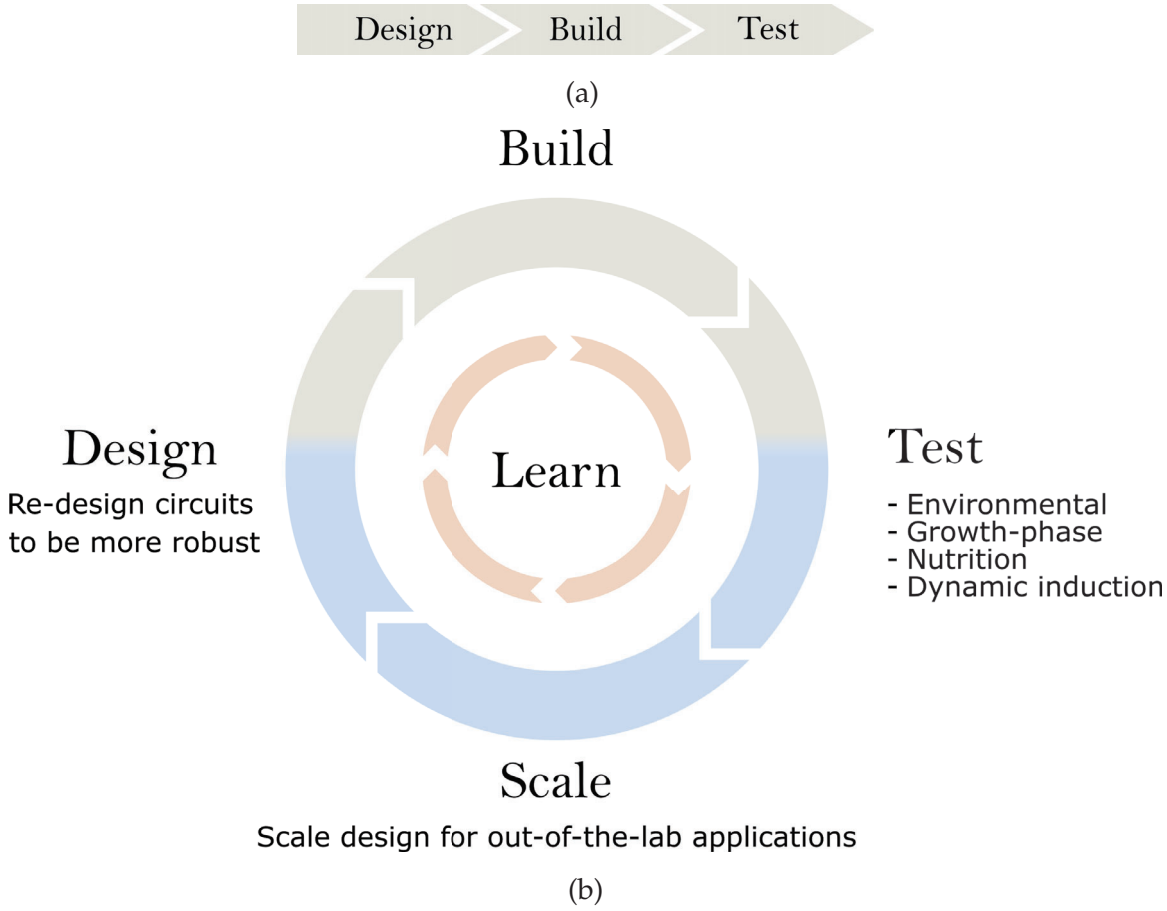


Figure 1.1. DBTS workflows in synthetic biology. **(a)** Most common workflow, where testing is done only in optimal lab conditions and there is no feedback to alter design. **(b)** Proposed workflow in which the *Test* step includes different conditions that may affect the outcome of a circuit. This can be utilized in the *Scale* step to obtain new model predictions, which will allow one to make better design choices. Ultimately, this will produce robust designs.

CHAPTER 2

BACKGROUND

This chapter outlines the background research for the work presented in this dissertation. This chapter introduces the field of synthetic biology, introducing what genetic circuits are (Section 2.1.1), and how synthetic biologists set out to design them using standardized genetic parts (Section 2.1.2). However, given that biological systems are inherently complex and variable, many of these designs cannot perform as intended. Section 2.2 introduces the reader to some of the potential failures or deviations from expected behavior that *Genetic Regulatory Networks* (GRNs) can have. After describing how circuits can fail, this chapter, this chapter continues to explain the different ways to mathematically describe genetic circuit predicted behavior and how designers can use models to understand genetic circuit failures (Section 2.3), and how these mathematical models and circuit specifications are encoded in data standard files shared by the synthetic biology community (Section 2.4). This chapter continues describing the online repositories (Section 2.5) where parts, genetic circuits, and models are uploaded and shared between different research laboratories, and how this can be used to spur reproducibility in the synthetic biology community. Finally, this chapter describes different capabilities of software tools used to design, model, and simulate GRNs (Section 2.6).

2.1 Synthetic Biology

Synthetic biology aims to design genetic circuits with different applications in mind, some of them being safety-critical. In this area of research, some strive to do so by imprinting engineering principles into classical genetic engineering as not to only tinker with naturally occurring biological systems but to consciously design complex genetic circuits with specific design aims and functionality in mind. One of the basics of any engineering discipline is to have different characterized modular parts and synthetic biology is not an exception. This work focuses on the design, modeling, and analysis of

genetic combinational logic circuits that are built using modular genetic parts, as described in the next sections.

2.1.1 Genetic Regulatory Networks

Genetic circuits are designed GRNs, composed of synthetic genetic parts, which perform a specific function specified by a researcher. In recent years, there has been a plethora of new engineered genetic circuits with specific functions, an expansion of reusable modular genetic parts and sensors, and an exploration of synthesizing genetic circuits in organisms other than bacteria [92, 94, 114, 123, 156]. Classically, genetic engineers would modify/add/knockout one, two, or many more genes of a living organism in order to study their role in the organism or to try to introduce a new feature or function to it. In the area of synthetic biology, characterized modular genetic parts or gates are combined to compose a genetic circuit with a user-defined function or implementation. It is this bottom-up design approach of genetic manipulation that distinguishes synthetic biology from classical genetic engineering.

Synthetic biologists draw inspiration from other engineering disciplines in order to have a more modular, predictable methodology for the design of genetic circuits. The first such genetic circuits, which were designed with a model-driven approach, are the genetic toggle switch [60] and a synthetic oscillatory network [46]. These genetic circuits were designed with a specific function in mind, and later built to test the design using genetic parts available. Nowadays, the design of genetic circuits enables researchers to engineer cells to process input signals, make logical decisions, implement memory, and to communicate with each other [163]. These circuits can also be designed to produce an output with a range of different purposes like inciting a biological response within or with other cells, producing a chemical for the environment or for industrial purposes, and many others.

Transcription and translation regulators that influence the flux of *Ribonucleic Acid* (RNA) and protein production are commonly used to carry out Boolean logic and, therefore, are called *logic gates*. These logic gates can be built on the basis of different regulator types using DNA-binding proteins, recombinases, CRISPRi regulation, RNA regulation, or protein-protein interactions [19]. Each gate can be designed to perform a specific logic

function, like for example an AND, OR, NOT, NOR, NAND gates, and many others. With this, a researcher can link these logic gates to various cellular or environmental sensors and actuators, to generate circuits with precise desired behaviors in response to specific inter and intracellular signaling inputs [167]. However, to design these genetic circuits, there is a need for libraries of well-characterized, modular and standardized genetic parts and computational tools for easier design and to tune them [19], which is the topic of the next sections.

2.1.2 Genetic Parts

Genetic circuits represent how information is going to be processed and which decisions are going to be made, and these are composed of *genetic parts or gates* such as sensors, actuators, and logic gates [163]. These genetic parts are used to specify when, where, and how the different parts interact and under what conditions are genes expressed [163]. However, a difficult challenge in the area of genetic engineering is the unpredictability of the behavior of assembled genetic parts in different genetic contexts [19, 24]. There are efforts to design genetic parts or gates with predictable and modular functions [10, 41, 44] with predictable gene expression and extensive characterization like in [23, 28, 84, 115–117], to cite a few. Endy [48] suggests that the biological engineering community would benefit not only from building a library of characterized modular parts, like the "iGEM Registry of Standard Biological Parts"[1], but also with the promulgation of standards that support the definition, description, and characterization of these biological parts.

Standardization of genetic parts is essential for reproducible experiments, reusability of genetic gates, and for model-driven design of circuits. Standardization can range from building techniques and gate conformation [82, 83, 115, 135, 151] to gate parametrization [28, 116]. A common method to characterize genetic gates in synthetic biology is using a standard for promoter activity, the *Relative Promoter Units* (RPU) [80]. RPU are used in many projects like iGEM and Cello (described in Section 2.6.1), and can be measured using a standardized kit for experiments, which makes it easy to adopt by different laboratories. This method is an effort to begin to address the challenge of characterizing promoters (and other types of standard biological parts) across the interdisciplinary community of

[1] http://parts.igem.org/

synthetic biology [80].

The construction of genetic circuits requires a library of basic components with shared inputs and outputs, which permits the composition into more complex devices and circuits [101]. As well, characterized genetic parts and modularity are an integral concept in synthetic biology, and it is essential for model-driven design of genetic circuits [23, 25]. One such library of well characterized modular parts with shared inputs and outputs is the Cello genetic gate library [129], and, therefore, it is a good choice to develop models using them.

2.2 Genetic Circuit Failures

For electronic circuits, unwanted output variations (or *circuit failures*) are typically filtered using a global clock signal to indicate when the circuit has stabilized to its final value. These types of circuits are known as *synchronous circuits*. It is not easy, however, to add such a clock signal to a genetic circuit, so they are typically *asynchronous circuits* [118, 127]. In electronic asynchronous circuits, a *hazard* is the *possibility* of an unwanted or unexpected output variation of a circuit in response to an input change [118]. The actual occurrence of a variation is called a *glitch*. Glitches are transient behavior that self-correct as the system reaches a steady state. The two main kinds of hazards are *function* hazards, which cannot be avoided since they are a property of the circuits' function and are inherently unavoidable, and *logic* hazards, which can be avoided by redesigning the logic of the circuit using hazard-preserving optimizations [118]. Though these terms are mostly used for electronic circuits, glitches have been observed in GRNs as well [112] (see Section 2.2.1 and Chapter 4).

Other circuit failures that are not usually modeled or analyzed are *set-up* and *hold-state* failures (see **Figure 2.1**). Set-up failures are glitches produced when a circuit is initialized and its components have not been stabilized yet, since they all initialize with no production at all. This means that a genetic circuit may produce the incorrect output once it has been initialized, before reaching the expected outcome at steady-state. Hold-state failures, on the other hand, is the incapacity of a genetic circuit to hold the correct/expected state for a constant input concentration due to random and sporadic changes in the inner circuit components' concentration. So, if the intended application of the circuit cannot fail at

set-up or needs to be particularly robust to hold-states failures, then the proper prediction of these circuit failures and their analysis is of critical importance.

Most designed genetic systems are tested in optimal lab conditions, with no regards to the variable and noisy environmental contexts in which they reside. However, as we move from proof-of-concept designs to more real-life applications of genetic circuits, the degree of environmental and noise effects on a circuit's output (or the *robustness* of such a circuit to these effects) is critical for a correct and safe behavior [146]. It is therefore, necessary to study and predict the likelihood of faulty behavior (*glitches*), or erroneous states, or other inherent frailties (contrasted with electronic circuits) that need to be studied and predicted accurately to avoid irreversible harmful effects. However, there are no tools or methods developed yet to systematically study circuit failures, and the amount of effort, time, and money needed to characterize parts for different environments can be prohibitive. Therefore, inspecting the extent one has to re-characterize parts and systems to have reliable predictions of circuits' robustness is paramount for saving time and money, and making better design choices.

2.2.1 Combinational Circuit Hazards

Genetic Regulatory Networks (GRNs), particularly combinational genetic circuits, can have unwanted switching variations (or *glitches*) in the circuit's output due to changes in the input concentrations. These unwanted signal transitions are only temporary and correct themselves as the system reaches steady-state. This may not be of major concern if the output of the circuit is only measured at steady-state or if it is a reporter protein. However, if the intention of the designer is for the output to have an irreversible effect on the cell or environment, like inducing cell apoptosis or causing a cascade of responses, then these circuit glitches become critical.

Previous work done by Fontanarrosa et al. [55] and Buecherl et al. [20] investigate the glitching behavior of a combinational genetic circuit first published by Nielsen et al. [129]. Fontanarrosa et al. [55] identified the glitching behaviors for the input transitions that contain function hazards and designed different circuit layouts to analyze the impact on glitch probabilities. Buecherl et al. [20] built on those results using stochastic simulation and stochastic model checking to determine the probability of the glitches discovered by

Fontanarrosa et al.

Multiple input changes can cause unwanted switching variations, or *glitches*, in the output of genetic combinational circuits. These glitches can have drastic effects if the output of the circuit causes irreversible changes within or with other cells such as a cascade of responses, apoptosis, or the release of a pharmaceutical in an off-target tissue. Therefore, avoiding unwanted variation of a circuit's output can be crucial for the safe operation of a genetic circuit. This work investigates what causes unwanted switching variations in combinational genetic circuits using hazard analysis and a new dynamic model generator (Chapter 4). The analysis is done in previously built and modeled genetic circuits with known glitching behavior. The dynamic models generated not only predict the same steady states as previous models but can also predict the unwanted switching variations that have been observed experimentally. Multiple input changes may cause glitches due to propagation delays within the circuit. Modifying the circuit's layout to alter these delays may change the likelihood of certain glitches, but it cannot eliminate the possibility that the glitch may occur. In other words, function hazards cannot be eliminated. Instead, they must be avoided by restricting the allowed input changes to the system. Logic hazards, on the other hand, can be avoided using hazard-free logic synthesis. This work demonstrates this by showing how a circuit designed using a popular genetic design automation tool can be redesigned to eliminate logic hazards.

2.2.2 Stochasticity and Noise

The deterministic framework of *Ordinary Differential Equation* (ODE) analysis is appropriate to describe the *average* (population) response simulation of a system to input changes. However, even single-strain cell populations can exhibit a high degree of gene expression variation for the same environment [6]. In other words, with ODEs modeling, there is no randomness or stochasticity associated with the model, and the same results are obtained given the same initial conditions [3]. However, the stochastic nature of biochemical reactions, even at the single-gene level [47], generates significant variability or *noise* to a system [137], which is why clonal populations of cells can exhibit substantial phenotypic variation [47]. So even if ODEs simulation predicts that there is little or no glitching behavior for certain input changes, it might be the case that a significant

percentage of a homogeneous population does manifest the unwanted switching behavior. All interesting genetic circuits with more than a single input have function hazards, and many of these hazards can turn into glitches. Therefore, stochastic analysis is required to fully understand glitches and their probabilities, especially for circuits where the output is deemed critical.

There are different sources of noise that would generate variability in a circuit's output. The inherent stochasticity of biochemical processes, such as transcription and translation, generates *intrinsic* noise [158]. This is especially significant in systems with low copy numbers of *messenger RNAs* (mRNAs) or proteins in living systems [158, 160]. Therefore, stochastic effects are thought to be particularly important for gene expression and have been invoked to explain cell–cell variations of output production in clonal populations [47, 158]. The "stochastic chemical kinetics" that arise due to random births and deaths of individual molecules give rise to jump Markov processes, which can be analyzed by means of master equations and simulated with stochastic simulation algorithms [63, 89].

Further research is needed to determine if these sources of noise have any effect on circuit failure probabilities, and which (intrinsic or extrinsic) has a higher incidence in a circuit's output variance, and, therefore, in aforementioned circuit failure probabilities. These results are also critical when moving from proof-of-concept circuit designs to real life applications into health, industry, or environment [18].

2.3 Modeling Genetic Regulatory Networks

A mathematical model can provide mechanistic understanding of a GRN. Models that accurately predict behavior of a system allow engineers to design genetic circuits *in silico* before going to the laboratory, avoiding large numbers of trial-and-error experiments [25]. There are many advantages in modeling: predicting, even in a limited manner, how a system will behave under novel conditions, understanding how highly nonlinear systems work, revealing deeply hidden properties of a system, understanding where the design fails when predicted behavior is not what it is intended, and many other reasons [11]. However, models only have a limited capacity of prediction and newer models usually replace older approaches when their predictive capabilities increase.

Dynamical modeling can be described as the "classical" way to mathematically model

GRNs [148]. The objective of these models is to describe the dynamic behavior of a set of genes with interconnected expression levels, and predicting the behavioral response to various environmental changes and stimuli.

Quantitative PCR, microarrays, Northern blotting, and other techniques are getting cheaper and can typically measure average concentration of mRNA in a population of cells [9]. Western blotting can do the same for proteins [62]. Therefore, a mathematical model that deals with concentration averages over time, and the proportional amount of time in which a promoter is being occupied is needed. A very common method to do so is to describe a GRN using the law of mass action, classical chemical kinetics, and Hill functions; all of which are explained in the next three sections.

2.3.1 Law of Mass Action

There is an associated *rate constant* for each chemical interaction (i.e., a parameter that is proportional to the frequency a reaction occurs). The *law of mass action* states that the rate of a chemical reaction is directly proportional to the product of the reactant concentrations, to the power of their stoichiometry. This means that the change in concentration of the reactants per unit of time (velocity of a reaction), or in other words, time derivative to reactant concentrations, is proportional to the product of these reactant concentrations. This quantity accounts for the probability of collisions amongst reactants under the assumption of a well-stirred system [102]. This can be used to convert a chemical reaction network into a set of ODEs that can be analyzed using *Classical Chemical Kinetics* (CCK) model, which is the subject of the next section.

2.3.2 Kinetic-Based Models

Once all the reaction and species that comprise a GRN are identified, a mathematical model can be constructed by determining how they interact [25, 102]. Using the law of mass action, a set of ODEs can be derived that describe the change of species over time. This set of ODEs that tracks the concentrations of each chemical species is known as a *kinetic based model*, and the differential equations that compose it are known as *reaction rate equations*. This model assumes that reactions occur continuously and deterministically [119]. This deterministic framework is appropriate to describe the mean behavior of biochemical systems [97]. While mass-action kinetics are strictly only valid for elemen-

tary reactions, they are widely and successfully applied in many fields of mathematical modeling in biology [147].

Reactions in biological systems are not only regulated by reactants and products, but also of other compounds that regulate the activity of these reactions like enzymes, often without being consumed during the reaction. In the next section, a description of a method that has proven to be appropriate to model enzymatic reactions [102] is explained.

2.3.3 Hill Equations

The Hill equation is a standard for characterization of regulated promoters because it demands only two parameters: the *Hill coefficient* (n) and the *Hill constant* (K_H or κ). These two parameters can be faithfully determined with experiments and represent an appropriate characterization of promoter/transcription-factor dynamics.

The Hill equation stems from the kinetic based model and assumes that the promoter of a transcription factor is momentarily occupied by transcription factors in a reversible reaction. Under the assumption that the concentration of the transcription factors and promoters is constant and under a steady-state condition, we can obtain two different forms of the Hill equation, depending on whether the the transcription factor activates (2.1) or represses (2.2) the promoter:

$$P^* = \frac{(\frac{A}{\kappa})^n}{1 + (\frac{A}{\kappa})^n} \cdot P_T \ , \tag{2.1}$$

$$P^* = \frac{1}{1 + (\frac{R}{\kappa})^n} \cdot P_T \ . \tag{2.2}$$

in which P^* is the concentration of promoter bound with a transcription factor, A and R are the concentration of transcription factors that activate or repress transcription, respectively, P_T is the total amount of promoters in the system, κ is the *Hill constant*, and n the *Hill coefficient*. In this model, κ gives the necessary concentration to activate or repress half of the promoters of the system, and n quantifies the *cooperativity* amongst activators (or repressors) when binding to a promoter [102, 119].

Kinetic-based models can take additional assumptions to further simplify the model without significantly affecting the ability to reproduce expected behavior [25]. One such common assumption is the *steady-state assumption*, which is described in the next section.

2.3.4 Steady-State Modeling

When the production and degradation rate of a chemical species are equal, its concentration will not change and is at *equilibrium*. The steady-state assumption assumes that all the chemical reactions of a system are at equilibrium or *steady-state*, meaning that the concentration of the chemical species do not change over time. For a system to be at steady-state, each variable in that system must be at steady-state. Such steady-states are found by setting all the first derivatives in a kinetic-based model equal to zero and solving the resulting set of algebraic equations [11]. Steady-state modeling has been shown to be appropriate for genetic regulatory network modeling [102].

When steady-state models fail to reproduce observed behaviors or predict dynamical behavior before reaching steady-state, some assumptions must be revisited [25] and allow for more relaxed assumptions to take place. The following section describes a different set of assumptions that allow for dynamic modeling.

2.3.5 Dynamic Modeling

As the complexity of a GRN increases, so does the behavior it presents, and, therefore, there is a need for a more accurate modeling technique [7, 161]. Instead of assuming that *all* species reach equilibrium as in steady-state modeling, some models only assume that some species (those that are involved in the fastest reactions) reach equilibrium before others. This would remove equations from the ODEs system, which describe the evolution of the variables at steady-state [15]. This assumption is usually applied to enzyme interactions [11, 136], since in many GRNs, the protein-protein dynamics are much faster than the transcription or translation process, meaning that the protein interactions reach equilibrium much faster than other interactions. However, depending on the system, this *quasi-steady-state assumption* can be made for any species in the system in which the modeler thinks there is a faster dynamic. This allows one to study the dynamics of species that are not at steady state with more precision. However, the quasi-steady-state assumption can only be safely made when the difference in timescales for the dynamics of species that reach equilibrium fast and those that do not are considerable [136].

2.3.6 Stochastic Models

The deterministic framework of ODE analysis is appropriate to describe the mean behavior of a system, with underlying assumptions such that the variables vary in a deterministic and continuous fashion. In other words, there is no randomness or stochasticity associated with the model, and the same results are obtained given the same initial conditions [3]. However, the stochastic nature of biochemical reactions, even at the single-gene level [47], generates significant intrinsic genetic noise to a system [137]. Furthermore, the underlying assumption of continuous-deterministic models that the number of molecules is high (or that the volume of the system is infinite) can be invalid for biochemical systems where there are very few transcription factors or only one copy of a *Deoxyribonucleic Acid* (DNA) segment [79]. Since transcription factors, enzymes, and DNA copies can exist in systems at a low concentration such as a single molecule per cell, any realistic analysis of these systems must include stochastic effects, and, therefore, stochastic modeling and analysis [31].

In order to capture the stochastic behavior, a *stochastic chemical kinetics* approach must be taken. With this approach, reactions are assigned propensities of occurring, rather than a rate of reaction, and molecule numbers can be estimated [119, 120]. Simulation requires a Monte Carlo approach, such as Gillespie's *Stochastic Simulation Algorithm* (SSA), which is already implemented in iBioSim [169]. In a SSA simulation, each simulation step selects a random time for the next reaction and a reaction to perform, and it repeats this process until a preselected time limit is reached [120], and, therefore, each simulation is unique. This generates different simulations for the same initial conditions every time the simulation is performed, and propensities and probabilities of certain states or dynamic behaviors occurring can be calculated.

2.3.7 Modeling Intrinsic and Extrinsic Noise

There are different sources of noise that would generate variability in a circuit's output. The inherent stochasticity of biochemical processes, such as transcription and translation, generates *intrinsic* noise [158]. This is especially significant in systems with low copy numbers of mRNAs or proteins in living systems [158, 160]. Therefore, stochastic effects are thought to be particularly important for gene expression and have been invoked

to explain cell–cell variations of output production in clonal populations [47, 158]. The "stochastic chemical kinetics" that arise due to random births and deaths of individual molecules give rise to jump Markov processes, which can be analyzed by means of master equations and simulated with stochastic simulation algorithms [63, 89]. However, Beal [6] argues that stochastic chemical kinetics cannot explain the observed variation, and thus the explanation of such variation falls back to extrinsic noise. *Extrinsic* noise is generally defined as fluctuations and variability in a system's reaction rates due to disturbances originated from its environment [132, 149]. This can be modeled as fluctuations in model parameters (such as transcription and degradation rates) [153].

2.4 Standards

Reproducibility is a critical issue for synthetic biology [81, 122, 166]. This rapidly advancing field has allowed for novel genetic circuit designs, modeling software, and assembly techniques. However, all of these developments are very labor-intensive with inputs from researchers with a multitude of different backgrounds, making the reusability of this information complicated. More mature engineering disciplines have tackled this issue with *standardization*, *abstraction*, and *decoupling* [41, 48, 120]. Some of these strategies, like abstraction and decoupling, are well under way of development. However, there is a growing awareness that the need for standardization is essential for the field to grow into a more predictable engineering discipline [41].

Standardization in synthetic biology ranges from standardized genetic parts and characterization [80], to standards for designing and visualizing genetic circuits, assembly methods, screening methods, reporting (modeling and simulation), and sharing (data repositories). Institutes like the National Institute of Standards and Technology (NIST) have convened meetings to discuss standardization efforts in synthetic biology, from DNA building blocks to documentation of experiments [69].

A key element for model-based design in synthetic biology is to develop *data representation formats* to allow for sharing of designs, models, and simulations in order to foster the interdisciplinary approach that is characteristic of this discipline [120]. A major initiative to encode biological information and to coordinate the development of data standards

happens under the "COmputational Modeling in BIology NEtwork" (COMBINE [2]) initiative [86, 120, 122].

In this section, a description of the three data standards curated under COMBINE and used by the model generator of this work are described. First, the data standard for specification of genetic circuits and their function (Section 2.4.1), followed by the standard used to describe mathematically the biological behavior of these genetic circuits (Section 2.4.2), and finally the data standard to report simulation results of the mathematical models developed (Section 2.4.3).

2.4.1 Synthetic Biology Open Language (SBOL)

The *Synthetic Biology Open Language* (SBOL) is an open standard for the representation of *in silico* biological designs[3] [56]. SBOL is a free and community-driven data standard used to encode structure and function of a genetic circuit or parts, with a focus on abstraction and composition [122]. SBOL is used to represent not only the sequences of genetic designs, but also functional interactions, proteins, metabolites, and biological chassis. This is a big advantage over other standards that encode DNA sequences like the FASTA format [133], GenBank's flat file format[4], and the Generic Feature Format[5], since researchers can describe a biological design or circuit without knowing necessarily the DNA sequences that will compose it. In this way, synthetic biologists can share designs of genetic circuits, their expected function, and interactions without even having to go to the laboratory and sequence DNA. Furthermore, SBOL allows for hierarchical designs that organize genetic parts into a more complex structure to describe a desired function, annotate environmental or experimental context information, computational models of behavior, and measurements of performance characteristics [56]. To guarantee interoperability and sharing between tools, SBOL permits assignments of roles and types to the functional components that compose the genetic design from ontologies, such as the

---

[2] http://co.mbine.org/

[3] http://sbolstandard.org/

[4] http://www.insdc.org/documents/feature-table

[5] https://github.com/The-Sequence-Ontology/Specifications/blob/master/gff3.md

Sequence Ontology (SO)[6] and the *Systems Biology Ontology* (SBO)[7] [140].

SBOL also allows one to create a *Model* class to document and link to external models written in standards other than SBOL, like for example the *Systems Biology Markup Language* (SBML) [26], which is discussed in the next subsection.

Most importantly, SBOL allows for the storage of any information not supported by the format in the form of custom and complex user annotations, so that no information is lost when encoding a design in the SBOL format.

2.4.2 Systems Biology Markup Language (SBML)

SBML is another standard under the COMBINE initiative [72], which is a free and open format for computer models of biological processes. SBML is useful for models of metabolism, cell signaling, genetic devices, and more. It is supported by an international community with many packages being developed from the users that expand its capabilities. It is supported and used by more than two hundred software tools,[8] which makes it an excellent format for the exchange and reuse of mathematical models between different areas of synthetic biology. This enables researchers to create, simulate, and annotate biological models that can be shared through different databases, like the BioModels database [27]. Furthermore, some parser libraries offer model checking, validation, and verification, as well as support for the SBO [14].

SBOL can be used to describe structural and qualitative behavior, whereas SBML is used to specify a mathematical model that describes the quantitative behavior of the system. Both are amply used standards, so naturally there exists converters from SBML to SBOL and vice-versa [128, 142]. Part of this work is essentially a new type of SBOL to SBML converter that creates dynamic models from genetic designs encoded in SBOL using the parametrization from the Cello project. While SBML is a widely accepted and used format for describing model structure, it does not cover the description of analysis or simulation performed to obtain predictions from the mathematical model. Therefore, this

---

[6] http://www.sequenceontology.org/

[7] http://www.ebi.ac.uk/sbo/main/

[8] http://sbml.org/SBML_Software_Guide

work uses another standard format to do so, which is described in the next subsection.

2.4.3 Simulation Experiment Description Language (SED-ML)

The *Simulation Experiment Description Markup Language* (SED-ML) is an XML-based format developed for the encoding of simulation and analysis experiments performed on a mathematical model [165]. It is a core standard of COMBINE, and it is used primarily to specify which models to use in an experiment, modifications to apply on the models before using them, which simulation procedures to run on each model, what analysis results to output, and how the results should be presented [165].

To enable shareable and reproducible analysis, authors should provide SED-ML files along with the SBML files with their publications, so other researchers can reproduce the presented simulation results obtained. This would allow other users to analyze and study under which conditions the simulation was carried out and test the results themselves.

2.5 Online Repositories

Data standards allow for standardization of ideas and information from a diversity of sources in synthetic biology. However, designs, models, and simulation results specified in these standards need to be stored somewhere in order to enable this data to be exchanged between different laboratories or researchers. *Online repositories* have been created for this purpose of storing and sharing data and are used by many researchers and tools. Some of them, like the iGEM Registry of Standard Parts[9] or JBEI-ICE [67], have been developed specifically for the storing and sharing of engineered biological deigns. However, many of these online repositories can only store sequences but not other information such as proteins, interactions, metabolites, biological chassis, or models that describe the function of a circuit, which is very important for genetic design. With this in mind, two online repositories have been developed to fulfill this gap which are discussed in the following sections.

[9] http://parts.igem.org/Main_Page

2.5.1 SynBioHub

SynBioHub [107] is a repository for genetic designs encoded in SBOL. SynBioHub is designed to store parts and designs in a linked format so that data can be findable, accessible, interoperable, and reusable (FAIR) [107]. This linked data and the use of sequence ontologies permits for a powerful querying capacity that facilitates searchability from tools and users alike. Designs can be shared with users or other applications, like, for example, Benchling.[10] Sequences can be shared with other applications that specialize in sequence editing and/or copying, and then transferred back into SynBioHub. Moreover, any custom annotations made by a researcher in the SBOL file are queryable, which makes the retrieval of information more straightforward. The Synthetic Biology Knowledge System (SBKS) [99] is one such instance of the SynBioHub repository that included text and data information that has been mined from papers published in different academic journal articles.

These functionalities make SynBioHub an excellent choice to upload genetic designs, models, and simulations, to search for existing designs, and to share or export for publication or collaboration.

2.5.2 BioModels

The *BioModels Database* is a public online resource that allows storing and sharing of published, peer-reviewed quantitative, dynamic models of biological processes [27]. Models uploaded to the BioModels Database are manually curated to ensure reliability and correspondence with the original publication's results. All models are annotated with controlled vocabulary terms and linked to external data, which facilitates model reuse and interoperability. Models are stored in the SBML format and are available to download in several other formats.

The submission of models to the BioModels repository has increased rapidly [27]. This allows modelers not only to share their models, but also to reuse models uploaded by other researchers to modify them and implement their own analysis and publish articles. This is a great resource to address the reproducibility crisis in synthetic biology [81, 166].

---

[10] https://www.benchling.com/

2.6 Genetic Design Automation Tools

Despite all this potential, genetic circuit design remains one of the most challenging aspects of genetic engineering [19, 137]. Due to the inherent complexity of biological systems, engineering complex genetic circuits is a bigger challenge than was anticipated [85, 96]. As the requirements of developing novel synthetic biological systems have become more complex, the need for models and software design tools has become more acute [97]. Several approaches have been implemented for the development of computational tools for synthetic biology [29, 96, 97, 101, 103]. However, there has been an increased focus on tackling this complexity of genetic circuit design and frame these recent computational tools by developing *Genetic Design Automation* (GDA) tools [29].

GDA tools rely on well characterized, modular genetic parts, in particular the development of orthogonal transcription factors [29]. This presents a challenge for the synthetic biology community since many genetic parts or gates have unbalanced regulator expressions, they behave differently when combined in a genetic system, and they have complicated states depending on the inputs [129]. However, one such project that has overcome some of these difficulties and developed a library of characterized modular parts to use to automatically design genetic circuits is described in the following subsection.

2.6.1 Cello

The design environment, referred to as *Cello* [129], is a GDA tool created to automatically design genetic circuits with user-defined behavioral response over a set of inputs changes. It was developed to accelerate circuit design, to enable non-experts to incorporate synthetic genetic circuits into their genetic engineering projects, and to enable one to specify a user-defined computational operation behavior of such a circuit. This design environment implements algorithms that derive a physical design (sequence of parts) from a textual specification, written in Verilog, in which the user specifies inputs, outputs, and an expected computational Boolean logic in the form of a truth table that the user wants the circuit to perform.

Cello needs three inputs in order to work. First, there is the DNA sequences of the sensor gates for the circuit, and their ON/OFF RPU output. The second is a *User Constraint File* (UCF) that contains information such as the functional information (transfer

functions in RPU) of the library of gates, the layout of the genetic system, organism, strain, operating conditions, toxicities, promoter road-blocking, and other constraints to be taken into account by the algorithm. And lastly, there is a Verilog code that captures the desired behavior (as a Boolean computational operation) of the genetic circuit to be designed [129].

Cello utilizes this information to automatically design a genetic circuit that connects to cell-based sensors and cellular actuators. It does so in three steps — first, the textual command is converted to a circuit diagram; second, Cello assigns specific regulators to each gate or node in the circuit diagram; the third and final step creates a linear DNA sequence based on the circuit diagram and gate assignment [129]. The output circuit is described using SBOL, and it contains the DNA sequences of all the parts of the circuit. The actuator of such circuit can then be connected to any cellular process by directing the output of the circuit as a stimulator or repressor of a metabolic pathway or other genes. Similarly, the sensor gates can be engineered to sense different cellular inputs or experimentally controlled variables such as temperature, pH, etc. The circuit performs Boolean logic computation based on the presence/absence of the sensors and produces the corresponding output from the implemented behavior.

The work that Nielsen *et al.* did in 2016 [129] used a library of gates based on prokaryotic repressors. Nonetheless, the Cello design environment can work with any gate that is repressible in different levels other than *RNA Polymerase* (RNAP) flux regulation, such as RNA-based regulation, protein-protein interactions, CRISPR/Cas-based regulations, or recombinases, as well as in different organisms other than bacteria.

This GDA tool requires genetic logic gates that are sufficiently modular and reliable, such that their interconnected behavior can be predicted, in order to work. For this, the Cello project has developed a set of insulated NOT and NOR gates based on prokaryotic repressors [129, 157]. The following subsection describes the gates used and their parametrization.

2.6.1.1 Cello Gates and Parameters

As mentioned before, each gate in Cello behaves as a NOR or NOT gate, which is composed of an *engineered region* or *expression cassette* preceded by two (NOR gate) repressible promoters or one (NOT gate) repressible promoter, as shown in **Figure 2.2**. When the sim-

ulation environment selects different gates for each node in the circuit, it chooses from a library of these engineered regions or expression cassettes, instead of choosing the *Ribosome Binding Site* (RBS), *Coding Sequence* (CDS), and terminators individually. Composing RBS + CDS + terminator into a functional component this way reduces variability, but at the same time, it is simpler to model and to combine during the simulated annealing process of Cello. Additionally, composed engineered regions or expression cassettes are easier to characterize experimentally, requiring far less experiments. It is a form of abstraction that reduces complexity and saves time [48]. Characterization of this library of composed parts was obtained experimentally [129] as depicted in **Figure 2.3** and **Figure 2.4**. **Figure 2.3** shows how the **sensor promoters** are parameterized. First, a constitutive promoter is added before a sensor gate, which produces a sensor protein continuously. This sensor protein can repress the sensor promoter that is being characterized, unless an experimenter adds an input molecule that represses this repressor. In the same plasmid, the sensor promoter is placed before a "*Yellow Fluorescent Protein* (YFP) RPU cassette", which is a functional component that produces YFP. YFP production is measured, using RPU, under two different conditions: adding an excessive amount of input molecule, in which case the sensor promoter is not repressed and the YFP production is maximum; and without any input molecule, in which the sensor promoter is maximally repressed and only basal production of YFP occurs. With these experiments, two parameters are obtained for sensor promoters: y_{max} and y_{min}. The parameter y_{max} depicts the maximum promoter activity for the sensor promoter, and y_{min} the minimum, or basal promoter activity in RPU. Likewise, **Figure 2.4** depicts the parametrization for all other gates that are not sensor gates or promoters. It is similar in fashion to the sensor promoter parametrization, but there is an extra step. In this case, the "YFP RPU cassette" is preceded by the gate promoter. On the same plasmid, the gate that produces the *Transcription Factor* (TF) that represses this gate promoter is preceded by a sensor promoter. Finally, on a second plasmid, a sensor gate constitutively produces a sensor protein. In this way, without an input molecule that represses the sensor protein, the gate production is minimum, and the gate promoter is not being repressed (producing YFP). Conversely, when the input molecule is present in large amounts, the sensor protein is repressed, the gate being characterized produces maximum amounts of TF and the YFP production is reduced to a minimum. To characterize these

gates, an experimenter introduces different concentrations of input molecule, and the YFP production is measured in RPU. A response function for each gate is formulated, and after fitting it to a Hill equation, the parameters y_{max}, y_{min}, n, and κ are obtained. As with the sensor promoter characterization, y_{max} and y_{min} depict the maximum and minimum promoter activity in RPU, respectively. For the other two parameters, n is equivalent to the Hill coefficient, and κ is equivalent to the dissociation constant [129].

The parameters y_{max}, y_{min}, n, and κ were measured for each gate in isolation of other gates, as shown in **Figure 2.4**, and are stored in a UCF, which is then fed to Cello when designing a circuit. Once Cello designs the circuit and assigns gates to each individual node of the circuit, it stores all that information in a SBOL file. The next section describes this output.

2.6.1.2 SBOL Specification

Cello was used to design a large set of circuits (52) based on the insulated gates described earlier [129]. The output of Cello can be encoded in an SBOL file, as well as a netlist (JSON file), cytometry plot (PNG file), transcription values in RPU (CSV file), truth table (CSN file), or others. Each circuit is composed of multiple NOR and NOT gates, sensor gates, and the output gate. A NOR gate is composed of two repressible promoters and an expression cassette; an example is shown in **Figure 2.5**.

A collection of the Cello insulated gates (expression cassettes and promoters) encoded in SBOL used for this work is uploaded in a SynBioHub repository.[11] With this, one can design a circuit using other design environments other than Cello and use these parts. These parts not only contain parts and sequence information, but they also store the parameters y_{max}, y_{min}, n, and κ. These parameters are stored as SBOL annotations in each expression cassette for the case of Cello gates, and in the repression interaction of sensor proteins to sensor promoters for the case of sensor promoters. The automated model generator of this work searches for these parameters to generate the model that describes the dynamic behavior of a circuit.

---

[11] https://synbiohub.programmingbiology.org/public/Eco1C1G1T1/Eco1C1G1T1_collection/1

2.6.1.3 Cello's Circuit Performance Prediction

Qualitative predictions of circuit performance (output distributions) are obtained computing the combination for each individual gate's output distribution [129]. This is the last step performed by Cello after gate assignment and produces a prediction of the circuit's output as a distribution. To perform this prediction, there has to be experimental data to fit a response function for each gate. Thus, each gate in the circuit must have experimental cytometry distributions added to the UCF, with the fluorescence values reported in RPU [129].

Once all the gate distribution response functions are calculated, the qualitative predictions for the output product can be computed. For a particular input combination, the sensor values (concentrations) are fed to the first layer of the circuit (sensor gates). Each sensor gate has a distribution response function, so with the concentration of input molecule a vertical "slice" is obtained from the distribution response function to create an output histogram for the gate. Next, these gate output histograms become the input histograms for the second layer of gates. This is done for all the different layers, composing output histograms for each layer and feeding it as an input for the next layer, until the final circuit's output histogram is calculated. Then, finally, for each input signal combination, a histogram of output for each individual gate is estimated, and the signal is propagated throughout the entire circuit until the last one (circuit's output) is calculated to produce the truth table predictions of the work [129].

The composition of response functions to obtain a predicted output histogram is based on steady-state experimental results and is a steady-state outcome prediction. This means that any dynamic behavior the circuit undergoes before reaching steady-state is missed by this analysis. However, in the original science paper [129], researchers did a time-course experiment to obtain output production in RPU every hour for a particular circuit, until the circuit reached steady-state, shown in **Figure 2.6**. In this experiment they observed that the circuit, for some of the input combinations, behaved in an unpredicted manner: the output would vary in unexpected ways before reaching the correct predicted steady-state output of the circuit. This type of behavior cannot be predicted with steady-state modeling and simulation, and it is why it is so important to have a dynamic model that can do so. The ability to dynamically model genetic circuits using Cello gates and parametrization

to be able to predict this dynamic behavior before and after reaching steady-states is what inspired the work of this dissertation. Dynamic modeling would not only predict this kind of behavior, but also allow for a finer analysis of circuit dynamics to detect failures.

The Cello simulation environment is used mainly for the design and implementation of genetic circuits. However, there are other GDA tools that not only allow for the design, but also the modeling and simulation of genetic circuits. One such tool, iBioSim, is discussed in the next section.

2.6.2 iBioSim

iBioSim is a GDA tool for the design, modeling, and analysis of genetic circuits that is being actively developed at the University of Utah and the University of Colorado Boulde [98, 121, 169]. This tool has been developed to promote model-based design of genetic circuits using community-developed data standards such as SBOL, SBML, and SED-ML. While Cello is a GDA tool to automatically design genetic circuits, iBioSim is not restricted to genetic logic circuits. iBioSim allows for a wider range of genetic parts and metabolic species, modeling, and simulation using data standards, automatic uploading to online repositories among other things. The following is a high-level description of the key features of iBioSim:

- **Genetic Circuit Design**
    1. Incorporated sequence editor tool SBOLDesigner [174]. Genetic designs can be viewed, edited, and create hierarchical levels of design.
    2. Front-end connection to SynBioHub for the uploading, downloading, and sharing of genetic circuits.
- **Model Generation**
    1. The *Virtual Parts Repository* (VPR) model generator is used to obtain and enrich SBOL files with interaction data, small molecules, and more for designed circuits.
    2. Integrated SBOL to SBML converter [142] that can be used to translate structural and functional information to create a quantitative model expressed in SBML using generic or user-defined parameters.
    3. User interface to edit and refine the model using the model editor GUI.

- **Analysis**
  1. Variety of simulation methods to analyze SBML models such as ODEs and stochastic simulation, and many others using SED-ML.
  2. Perform *Flux Balance Analysis* (FBA) on SBML models.
  3. Perform stochastic model checking, and simulation of grid-based, hierarchical models of dynamic cellular populations.
  4. View simulation results plotted in a graph.
- **Synthesis**
  1. Automated methods for part selection using a process known as technology mapping [141].
  2. Technology mapping for asynchronous sequential genetic circuits [127].

iBioSim provides automatic SBOL to SBML converter, though not one that can use Cello's parts and parametrization to generate a dynamic model. This dissertation implements an automatic dynamic model generator for genetic parts that uses Cello parametrization in iBioSim. In the next subsections, a more detailed description of the VPR, the SBOL to SBML converter, and dynamic modeler of iBioSim is provided.

2.6.2.1 Virtual Parts Repository (VPR)

Modular genetic parts for synthetic biology not only can be reused for different projects, but also provides modular and reusable models and information to be shared. Modular models facilitate the process of model-centered design and the availability of databases of modular models is essential to support automated model generation tools like iBioSim. The VPR has been developed with this emphasis on mind. VPR is a repository of *Standard Virtual Parts* (SVP), which are reusable, modular, composable, and shareable models of physical biological parts for synthetic biology [109]. The computational models and interactions are available as SBML documents and in SBOL format for standardization purposes. The repository was populated with data mined from an ontology representation of the BacillOndex dataset [108, 111], which includes around 3000 virtual parts and 700 models of interactions between them.

An *application programming interface* (API) is also available to enable programs to access

VPR via a Web service. This can be used to retrieve SVPs, a list of interactions for a part or SBML models of parts and interactions to construct models of biological systems [109].

These features are used by iBioSim to obtain interaction data and add functional information to the SBOL description of a genetic design. It can add proteins as well as coding sequences in the same SBOL document in which the design is specified [169]. The use of VPR for automated processes like the automated generation of models for GRNs is particularly suitable for these reasons [110].

2.6.2.2 SBOL to SBML Converter

iBioSim also comes with an integrated SBOL to SBML converter [142]. This utility is used to convert qualitative models and structure encoded in SBOL to quantitative models expressed in SBML [110, 169]. During the construction, the species and reactions generated for the mathematical model encoded in SBML are also annotated with elements from the SBOL document in order to preserve provenance of the model and the molecular identities of the species [142]. This converter adds default parameter values to the interactions [110], but these parameter values can be later modified within the iBioSim model editor. The derivation of these rate laws is based on the law of mass action and some model abstraction techniques like the operator site reduction or quasi-steady-state approximation [142]. For a more detailed review of these abstractions can be found in the literature [119]. SBML models constructed this way can then be simulated in a variety of methods.

Part of the work presented in this dissertation is a new SBOL to SBML converter that uses functional and structural information of a genetic circuit encoded in SBOL to produce a mathematical model described in SBML, using parameters and characterization as in the Cello project to create a dynamic model of GRNs.

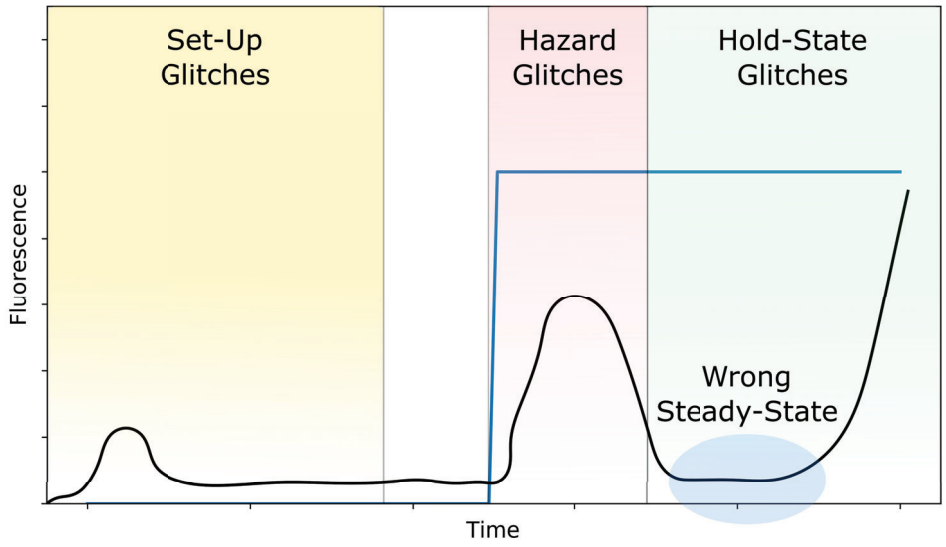


Figure 2.1. Example diagram showing the different circuit failures. Set-up glitches can be produced when the circuit has not been initialized properly. This means, the initial internal states of the circuit (until they reach steady-state) produce an unwanted or unexpected circuit output. Hazard glitches can be produced when multiple input changes occur and are a transient behavior that self-corrects as the system reaches a steady state. Hold-state glitches occur when the circuit's output is altered due to *noise* in the system, which can randomly switch or alter the circuit's steady-state output.

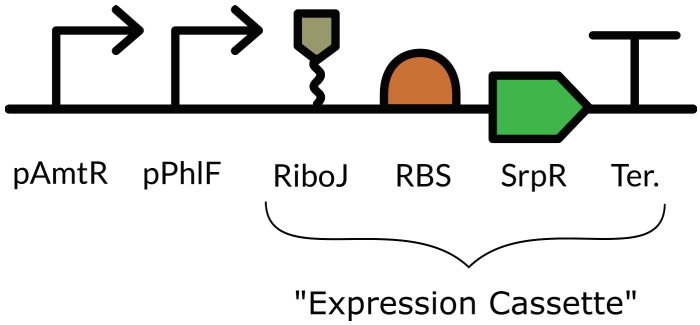


Figure 2.2. A Cello NOR gate [129]. Each gate in Cello consists of a genetic "expression cassette" (in this case the gate is "S4_SrpR") or engineered region that interacts with a downstream promoter. In this example, it is preceded by two repressible promoters, ("pAmtR" and "pPhlF"), which cause the gate to behave as a NOR gate. In this figure: **pAmtR** (promoter repressed by AmtR), **pPhlF** (promoter repressed by PhlF), **RiboJ** (insulator), **RBS** (Ribosome Binding Site), **SrpR** (SrpR coding sequence), and **Ter.** (terminator).

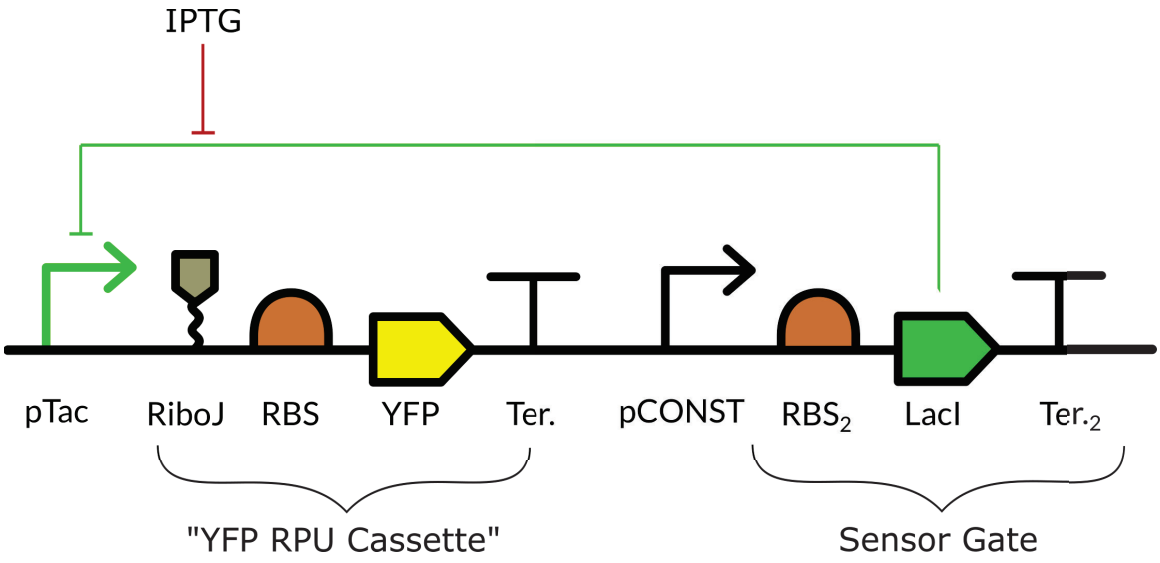


Figure 2.3. Sensor gate parametrization in Cello [129]. Each gate in Cello consists of a genetic "expression cassette" (in this case, a sensor gate) or engineered region that interacts with a promoter (in this case "pTac"). To characterize the RPU activity of a sensor promoter, the sensor promoter ("pTac", green) is positioned in front of an YFP "expression cassette" (or the "YFP RPU cassette") on a plasmid. On the same plasmid, a constitutive promoter ("pCONST") is placed in front of a sensor gate producing the TF, which represses the sensor promoter. In the case of sensor gate characterization, the YFP production is measured in RPU units at different concentrations of inducer (in this case "IPTG"). These data are then fit to obtain the values of y_{max} and y_{min} for the promoter pTac. In this figure: **pTac** (promoter repressed by LacI), **RiboJ** (insulator), **RBS** (Ribosome Binding Site), **YFP** (Yellow Fluorescent Protein coding sequence), **Ter.** (terminator), **pConst** (Constitutive promoter), **LacI** (LacI coding sequence), and **IPTG** ($Isopropyl\beta - D - 1 - thiogalactopyranoside$).

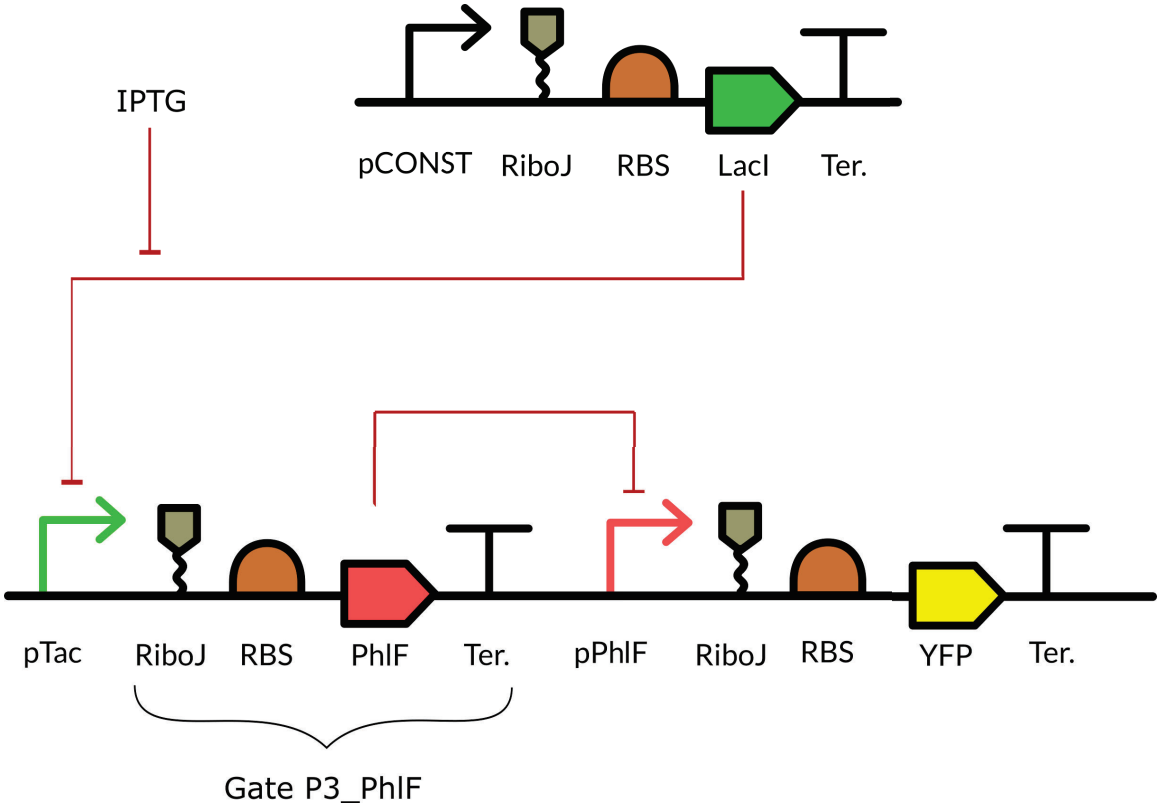


Figure 2.4. Genetic gate parametrization in Cello [129]. Each gate consists of a genetic "expression cassette" (in this case "Gate P3_PhlF") or engineered region that interacts with a promoter (in this case "pPhlF", red). An inducer (in this case IPTG) is added at different concentrations, and the YFP production is measured in RPU units to create a response function (not shown). A Hill equation is fit to the response curve to obtain the values of y_{max}, y_{min}, n, and κ. In this figure: **pTac** (promoter repressed by LacI), **RiboJ** (insulator), **RBS** (Ribosome Binding Site), **YFP** (Yellow Fluorescent Protein coding sequence), **Ter.** (terminator), **pConst** (Constitutive promoter), **LacI** (LacI coding sequence), and **IPTG** ($Isopropyl\beta - D - 1 - thiogalactopyranoside$).

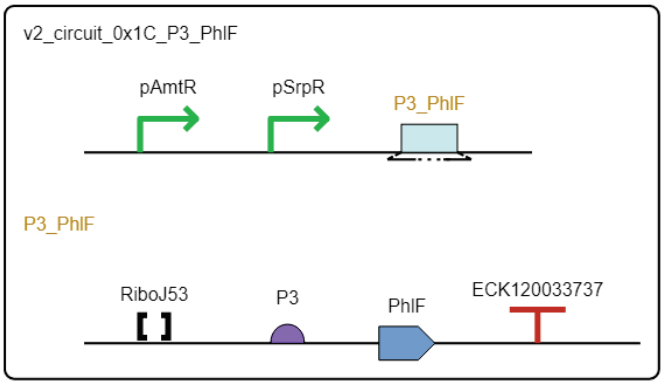


Figure 2.5. SBOL Visual [30] representation of a genetic gate. This gate corresponds to circuit 0x1C in [129]. This gate consists of two repressible promoters (pAmtR and pSrpR) followed by an engineered region or expression cassette (P3_PhlF). This expression cassette is composed of a ribozyme-based insulator (RiboJ), a ribosome binding site (P3), a coding sequence (PhlF), and a terminator (ECK120033737).

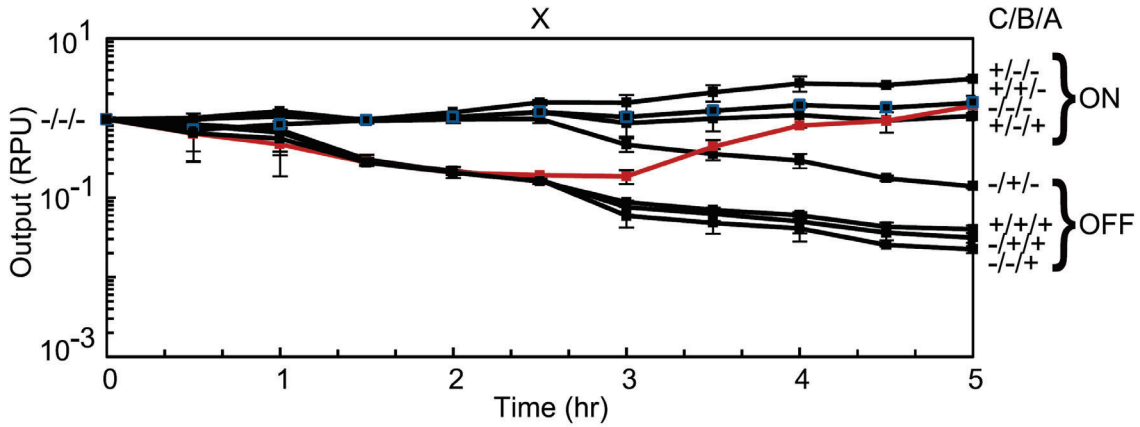


Figure 2.6. Time-course data for circuit 0x8E (courtesy of [129]). Each line represents the output YFP production (in RPU) over time (in hours) for the circuit 0x8E for each combination of input molecules. This circuit senses three input molecules: *Arabinose* (Ara), *anhydrotetracycline* (aTc), and $Isopropyl\beta - D - 1 - thiogalactopyranoside$ (IPTG). In the image, +/+/+ (Ara/aTc/IPTG) represents all input molecules are present and -/-/- represents no input molecules present. Also, the *ON* and *OFF* states represent the predicted outcome at steady-state. All outputs behave as expected, except for the +/-/+ state, which experiences an undesirable decay before rising to the *ON* state (red line).

CHAPTER 3

EXPANDING AUTOMATED MODEL GENERATION AND SIMULATION IN IBIOSIM

As described in prior sections, modeling is an essential piece of the *Design-Build-Test-Scale* (DBTS) cycle in synthetic biology, as it not only allows for verification of proper genetic circuit behavior, but also for the detection of possible genetic circuit failures and hazards. Consequently, the more accurate a model is, the more predictive and/or analytic power it can provide for synthetic biologists. This chapter starts by briefly explaining the model generator developed in Fontanarrosa [54] (Section 3.1), followed by the expansions contributed by this dissertation. Explicitly, Section 3.2 will explain how dynamic (temporal) modeling and simulation was added, Section 3.3 adds the effect of roadblocking to tandem promoter genetic circuit designs, and Section 3.4 explains how all this fits in the automation scheme of synthetic biology.

The model used by the automatic model generator of this work is composed of a steady-state model and a dynamic model that describe the circuits behavior. The steady-state model was proposed by Shin et al. [152] and the dynamic model by Moser et al. [113]. The procedure used here is not only suitable to model genetic circuits generated using the Cello tool, but also any circuit as long as the appropriate parameters are available.

3.1 Review of Previous Model Generation Automations

The automatic model generator of iBioSim [169] produces steady-state modeling of *Genetic Regulatory Networks* (GRNs) by simulating all the interactions known to occur for transcription and translation processes. However, many of these interactions are hard to evaluate experimentally and obtain meaningful parameter values to produce accurate models. Therefore, in Fontanarrosa [54], an automated model generator using bundled pa-

rameters following the model proposed in Cello [129] was implemented. The steady-state model stems from the Michaelis-Menten scheme and basic equilibrium kinetics that are rearranged to create a response function [23] for each genetic gate with variables that can accommodate specific parameters [152]. This response function describes the steady-state *RNA Polymerase* (RNAP) flux [51] output in *Relative Promoter Units* (RPU) [80] of a gate over the *output promoter* (promoter that the gate has an effect on), as a function of the RNAP fluxes of the input promoters. The *input promoters* can be a single or tandem promoters (upstream and downstream promoter) which can have roadblocking interactions (see Section 3.3). Equations 3.1 and 3.2 describe the steady-state output and input RNAP fluxes as:

$$y_{i_{SS}} = y_{i_{min}} + (y_{i_{max}} - y_{i_{min}}) \frac{\kappa_i^{n_i}}{\kappa_i^{n_i} + f(u,d)^{n_i}} , \tag{3.1}$$

where $y_{i_{SS}}$ is the steady-state output RNAP flux of gate i; $y_{i_{min}}$ and $y_{i_{max}}$ are the minimal and maximal output RNAP fluxes, respectively, for gate i; κ and n are obtained from the affinity and cooperativity of transcription factor binding; and, finally, $f(u,d)$ is the non-additive input RNAP fluxes from the input promoters of each gate. However, if there are no roadblocking parameter values, we can assume this input RNAP flux is an *additive* function of both input promoters. The steady-state calculation of input (sensor) promoter activities of input gates has the following form:

$$x_{i_{SS}} = \delta(1-q)\,(x_{i_{max}} - x_{i_{min}}) + x_{i_{min}} , \tag{3.2}$$

where $x_{i_{SS}}$ is the steady-state output RNAP flux of sensor gate i; $x_{i_{min}}$ and $x_{i_{max}}$ are the minimal and maximal output RNAP fluxes, respectively, for sensor gate i; q is the presence ($q = 1$) or absence ($q = 0$) of inducer molecules; and $\delta(1-q)$ is 1 when there are inducer molecules present and 0 when there are not.

Steady-state modeling is appropriate if the designer is only concerned about the correct behavior of the circuit at certain time-points, or if the intended purpose of the circuit is just to report the current state of input concentrations. However, for genetic circuits intended to compute specific logic that produces an irreversible effect on a cell (or other cells), or in situations where the output of a circuit needs to be determined at all times, then dynamic (temporal) genetic circuit modeling and simulation is needed. Furthermore, most combinational genetic circuits designs use Boolean functions to compute desired

functions. Given that NOR and AND gates are functionally complete (meaning that any computational operation can be implemented using layers of either of these gates alone), they are often used in genetic circuit designs [58, 159]. However, it is a well-known phenomenon that tandem promoters (used in NOR gates), produce a road-blocking effect [12, 21, 49, 106, 131, 150, 155, 173], due to RNAP transcription initiation blocking (*transcriptional interference*). Therefore, in this work we set up to expand the automatic model generator of Fontanarrosa [54] to include these features to provide it with more accurate, dynamic models of GRNs as explained in the coming sections.

3.2 Dynamic Modeling

The dynamic model is composed of a set of *Ordinary Differential Equations* (ODEs) for each genetic gate that describes the timescale by which a gate turns *ON* or *OFF*, using a simplified model that uses only two parameters (τ_y^{ON} and τ_y^{OFF}) as shown in the following equation:

$$\frac{dy}{dt} = \begin{cases} \tau_y^{ON}\,(y_{i_{SS}} - y_i) & \text{if } y_i < y_{i_{SS}} \\ \tau_y^{OFF}\,(y_{i_{SS}} - y_i) & \text{otherwise} \end{cases}, \tag{3.3}$$

where $y_{i_{SS}}$ is the RNAP flux of gate i at steady state (Equation 3.1), y_i is the current RNAP flux of gate i, and τ_y^{ON} and τ_y^{OFF} which are the bundled kinetic parameters that capture the response time to go to a steady state that is higher than the current output (τ_y^{ON}) or lower (τ_y^{OFF}) [113, 152]. Finally, to calculate the RPU output of the promoter controlling the output fluorescence protein expression, like *Yellow Fluorescent Protein* (YFP), we use Equation 3.4:

$$\frac{dYFP}{dt} = \tau_{YFP}^{ON} \cdot f(u,d) - \tau_{YFP}^{OFF} \cdot YFP\,, \tag{3.4}$$

where τ_{YFP}^{ON} and τ_{YFP}^{OFF} which are bundled kinetic parameters that capture the response time to go to a steady state that is higher than the current output (τ_{YFP}^{ON}) or lower (τ_{YFP}^{OFF}) [113, 152], $f(u,d)$ is the non-additive input RNAP fluxes from the input promoters of the reporter gate, and YFP is the output RPU of the promoter controlling YFP expression.

3.3 Roadblocking

Interference between two tandem promoters with the same orientation in gene expression is a known phenomenon [159]. Mathematical modeling methods have been described to represent this roadblocking effect of tandem promoters [155] and could be accounted

for in the automatic model generator in iBioSim. Using the model described in Shin et al. [152], a roadblocking effect can be included into the dynamic model generator using the following non-additive model:

$$f(u,d) = y_u \alpha \left(\frac{y_d - y_{d_{min}} + \beta \left(y_{d_{max}} - y_d \right)}{y_{d_{max}} - y_{d_{min}}} \right) + y_d \ , \tag{3.5}$$

where y_u and y_d are the RNAP fluxes of the upstream and downstream input promoters respectively, $y_{d_{min}}$ and $y_{d_{max}}$ are the minimal and maximal RNAP flux from the downstream promoter, α represents the non-specific suppression of the upstream promoter, and β which captures competing effects between the kinetics of the repressor off rate and RNAP dissociation rate. These last two parameters, α and β, are dimensionless and represent the degree of the roadblocking effect on the input RNAP flux of the upstream promoter from the downstream promoter.

However, if we want to make the input RNAP flux an additive model (meaning there is no roadblocking effect), we can simply set the values of α and β equal to 1 (meaning the interference effects go to zero). It would prove to be very interesting for future work to learn how these values will change the model predictions.

3.4 Automation of DBTS

The automatic dynamic model generator described in [110] has been modified in this work to use the model described in this section and to be able to predict the dynamical behavior of circuits within and in between steady states, which resulted in a publication [55]. This model generator is implemented in the iBioSim [169] software tool that is available in: https://github.com/MyersResearchGroup/iBioSim. As a demonstration, this section shows the output of the model generator for a simple genetic gate (shown in **Figure 3.1**) in iBioSim.

The automated model generator of this work will produce the dynamic model for the gate shown in **Figure 3.1** using, in particular, the following steps:

- Use the Virtual Parts API [110] to enrich the *Synthetic Biology Open Language* (SBOL) representation with interactions and their participants from the chosen repository and create a top module that encompasses all the different sub-modules.
- Extract the parameter information from parts stored in SynBioHub.

- Create a *Systems Biology Markup Language* (SBML) [26] document and an empty model.
- Create all the species, reactions, and mathematical assignment rules in the model.
- Generate the mathematical equations that compose the dynamic model for the chosen parts using equations (3.1), (3.5), (3.2), and (3.3).

Once this process is done, the user can look for the mathematical formulas of each reaction and assignment rule. The equations generated for this example (**Figure 3.1**) are:

$$PhlF_{SS} = y_{PhlF_{min}} + \left(y_{PhlF_{max}} - y_{PhlF_{min}}\right) \frac{\kappa_{PhlF}^{n_{PhlF}}}{\kappa_{PhlF}^{n_{PhlF}} + f(y_{BetI}, y_{SrpR})^{n_{PhlF}}} - y_{PhlF} \ ,$$

which is the assignment rule that calculates $PhlF_{SS}$ as the difference between the steady-state output of the gate (Equation 3.1) and the current RNAP flux output of the gate (y_{PhlF}), and

$$f(y_{BetI}, y_{SrpR}) = y_{BetI} \cdot \alpha_{SrpR} \left(\frac{y_{SrpR} - y_{SrpR_{min}} + \beta_{SrpR} \left(y_{SrpR_{max}} - y_{SrpR}\right)}{y_{SrpR_{max}} - y_{SrpR_{min}}} \right) + y_{SrpR} \ ,$$

which is the input RNAP flux of the tandem input promoters (Equation 3.5). This is then used to calculate the dynamic output of the gate using Equation 3.3 as:

$$\frac{dy_{PhlF}}{dt} = \begin{cases} \tau_{PhlF}^{ON}\left(PhlF_{SS}\right) & \text{if } PhlF_{SS} > 0 \\ \tau_{PhlF}^{OFF}\left(PhlF_{SS}\right) & \text{otherwise} \end{cases} \ ,$$

which is the reaction that describes the dynamic behavior of the output RNAP flux for the gate.

The resulting mathematical model describes how the RNAP fluxes for each promoter change over time depending on other RNAP fluxes and inputs, but it does not describe in which context this circuit is implemented. To specify the environment and sequence of inputs this circuit is subjected to, a simulation environment is created where the user can designate input value changes over time. Simulation of the dynamic mathematical model of a genetic circuit in an environment yields the expected dynamic behavior of the genetic gate or circuit for the specific changes described in the environment. The resulting complete model is then analyzed using the Runge-Kutta-Fehlberg (4,5) method [53] implemented in iBioSim [169].

A high-level depiction of a workflow using the automatic model generator implemented in this work is shown in **Figure 3.2**. The process starts with the stored information on the genetic parts to be used in a genetic design in an online repository, like

SynBioHub [107], and finishes with the complete circuit design, genetic parts, model, and simulations being stored, again in SynBioHub, for sharing or publication purposes. A genetic design tool, like Cello [129], extracts part sequences and other information, encoded in SBOL [56], from SynBioHub and produces a genetic circuit design, also encoded in SBOL. This design can be further enriched using the *Virtual Parts Repository* (VPR) [108, 111], which obtains interaction data from SynBioHub to add functional information to the SBOL description of the design [110]. This enriched design can then be imported into iBioSim [169], which converts the design (specified in SBOL) into a mathematical model (specified in SBML [26]) using the automatic model generator described in this work. To proceed with the simulation, iBioSim uses the mathematical model encoded in SBML to produce a *Simulation Experiment Description Markup Language* (SED-ML) [165] document that stores all the simulation conditions and results. All of these documents can then be uploaded to SynBioHub as an *Open Modeling EXchange format* (OMEX) [8] file, a single file that supports the exchange of all the information necessary for reproducing the model and simulation results for sharing or publication purposes. It should be noted that iBioSim is capable of retrieving information from SynBioHub, design genetic circuits using SBOLDesigner [174], and enriching the design using VPR, all from within the iBioSim environment since it integrates all of these tools [169].

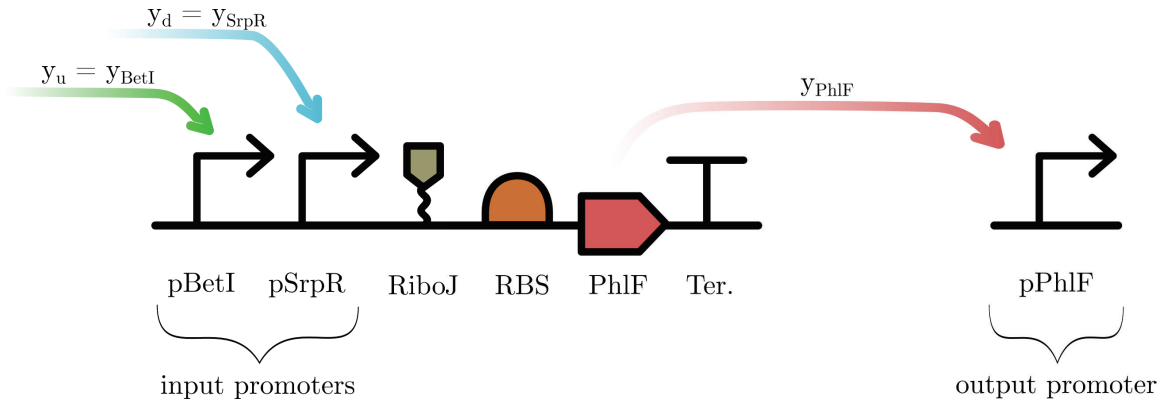


Figure 3.1. Diagram of a genetic gate with repressible tandem promoters. This diagram depicts the output RNAP flux of the gate (y_{PhlF}) to the output promoter ($pPhlF$) and the input RNAP fluxes of the gate (y_{BetI} for the upstream promoter and y_{SrpR} for the downstream promoter) that affect the input tandem promoters ($pBetI$ and $pSrpR$).

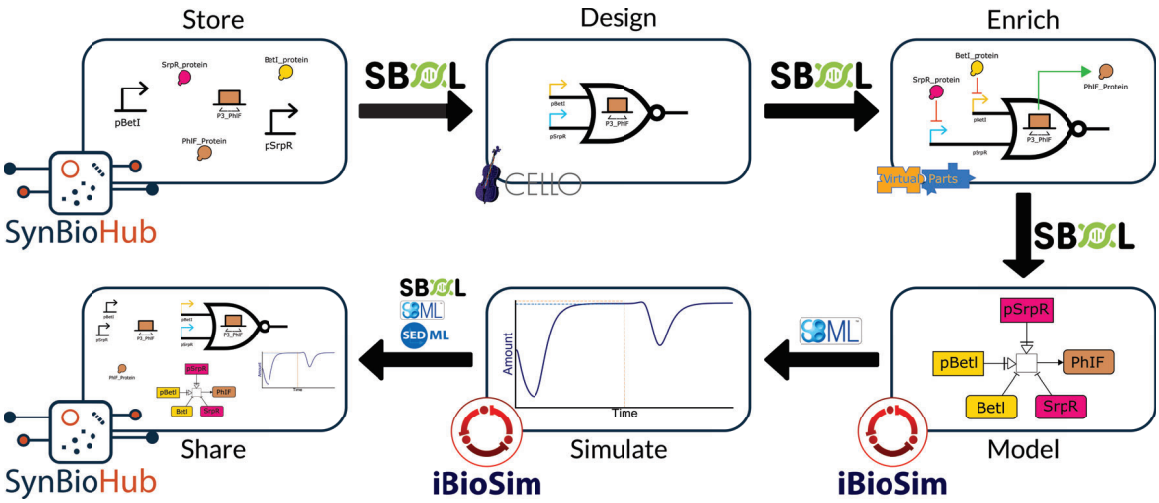


Figure 3.2. Automated model generator workflow. This figure represents the back-end of the automatic model generator implemented in this work, in which boxes represent tools and arrows represent data being shared between these tools. Genetic parts sequences and parametrization are stored in an online repository like SynBioHub ( ) where a genetic design tool, like Cello ( ), can extract information from. The design tool generates a SBOL-encoded design which can be further enriched using the Virtual Parts API ( ), which looks for interactions and molecules/proteins that the designed circuit would interact with. This enriched design, encoded in SBOL, can then be imported into iBioSim ( ). Then, the automatic model generator produces the dynamical model as described in this work and creates an SBML document with the mathematical model that describes the behavior of the circuit. This method then analyzes the model using ODEs simulation methods implemented in iBioSim and generates a SED-ML record of the analysis performed along with corresponding time course data. The user can then upload the project (with the design encoded in SBOL, the model encoded in SBML and the simulation in SED-ML) to an online repository, such as SynBioHub, for sharing or publication purposes.

CHAPTER 4

HAZARD ANALYSIS AND CIRCUIT FAILURES

The automatic model generator of this work would not only help with model predictions of genetic circuits before the building stage, but also in recognizing circuit failures of a circuit to either go back to the drawing board or building the circuit with known restrictions on the circuit implementation. Some of these circuit failures can be *glitches* (unwanted switching variation in the circuits' output) produced by hazards like *function hazards* (non-solvable hazards that may produce glitches when more than one input variable changes at the same time). Though glitches have been observed for genetic circuits [112, 129], this phenomenon's causes are not so well understood as in the asynchronous electronic community. This work can help introduce awareness of the subject to the synthetic biology community and automate much of the analysis needed to understand genetic circuits' glitching behavior.

This chapter analyzes a genetic circuit with known glitching behavior that has been previously designed, built, and modeled using steady-state modeling with Cello [129]. Cello is a *Genetic Design Automation* (GDA) tool developed to automatically design genetic combinational circuits that respond to a set of sensor inputs with a user-defined behavioral output response. This GDA tool implements algorithms that derive a physical design (sequence of genetic parts) from a textual specification using the Verilog design language in which the user designates inputs, outputs, and an expected combinational Boolean logic behavior.

To better understand what is causing the unwanted behavior, this chapter utilizes hazard analysis and a new version of our dynamic model generator [110] (described in Section 3.2) that automatically generates a dynamic mathematical model composed of a set of *Ordinary Differential Equations* (ODEs). This model then uses parametrization data

found in a Cello genetic gate library [129] [1] to simulate and predict glitches that cannot be observed with steady-state analysis.

The analysis and prediction of glitches is of paramount importance for the safe operation of a genetic circuit for medical purposes. If the output of a genetic circuit is a toxic pharmaceutical, or causes irreversible effects on the cell, then understanding, predicting and avoiding circuit failures is critical for the biomedical engineering community. This work will help elucidate the problem of circuit failures and provide insight as to how to avoid them.

4.1 Hazards

For combinational circuits, including combinational *Genetic Regulatory Networks* (GRNs), input changes can cause unwanted switching variations in the circuit's output. Unwanted signal transitions occur when the system has not reached a steady state, and the output signal varies from the expected behavior. In some cases, this variance is harmless or well-tolerated. For example, these unwanted signal transitions should not be a major concern if the output of the circuit is only sampled when the circuit has reached a steady state. Nevertheless, this glitching behavior can have drastic consequences if it causes an irreversible change. For example, causing a cascade of responses, inducing apoptosis, or inappropriately releasing a toxic pharmaceutical. Therefore, for the safe operation of a genetic circuit avoiding such unwanted variations in a circuit's output can be crucial.

For electronic circuits, these unwanted output variations are typically filtered using a global clock signal to indicate when the circuit has stabilized to its final value. These types of circuits are known as *synchronous circuits*. It is not easy, however, to add such a clock signal to a genetic circuit, so they are typically *asynchronous circuits* [118, 127].

In electronic asynchronous circuits, a *hazard* is the *possibility* of an unwanted or unexpected output variation of a circuit in response to an input change [118]. The actual occurrence of a variation is called a *glitch*. Glitches are transient behavior that self-correct as the system reaches a steady state. The two main kinds of hazards are *function* hazards, which cannot be avoided since they are a property of the circuits' function and are inher-

[1] https://synbiohub.programmingbiology.org/public/Eco1C1G1T1/Eco1C1G1T1_collection/1

ently unavoidable, and *logic* hazards, which can be avoided by redesigning the logic of the circuit using hazard-preserving optimizations [118]. Whereas a hazard is the possibility of an unwanted behavior, a glitch is an actual occurrence of it. Though these terms are mostly used for electronic circuits, glitches have been observed in GRNs as well [112].

The Cello project [129] designed and built 52 different circuits that were initially analyzed using steady-state modeling. A high-level logic specification, written in Verilog, is used to specify the desired behavior for each genetic circuit to be designed, which is then parsed to generate a truth table. Cello then utilizes a technique known as logic synthesis to produce a circuit diagram, and a Monte Carlo simulated annealing search to assign genetic parts to each gate of the circuit. With this process, Cello automatically designs genetic circuits that perform Boolean logic computations based on the presence/absence of inducer molecules being detected by genetic sensors and produces the specified output response via a reporter, such as *Yellow Fluorescent Protein* (YFP). For circuit 0x8E, for example, the expected behavior can be described using the truth table shown in **Figure 4.1**(a). This is then used by Cello to yield a genetic circuit with the logical structure shown in **Figure 4.1**(b). In the original Cello project [129], experiments were done to measure output using *Relative Promoter Units* (RPU) for different inducer concentrations for each circuit designed. They discovered that some of these circuits presented a glitching behavior [129]. For example, experiments done with circuit 0x8E showed that the output starts high, goes low, then goes high again for a particular change in inducer concentrations (see **Figure 4.1**(c), red line). In the original publication, the cause of this behavior was not investigated or discussed. Furthermore, since only steady-state modeling was available for these circuits, the unwanted fluctuations in the circuits' outputs were not predicted before the design was constructed.

This work generates dynamic models for the genetic circuits designed with Cello [129]. A full workflow, from importing the genetic circuits encoded in *Synthetic Biology Open Language* (SBOL) [140] to uploading the finished models and their simulation to an online repository is performed during the process. This workflow is used to analyze circuit 0x8E [129], predict the circuit's glitching behavior, and understand the phenomena underlying these undesired or unexpected behaviors. Even though the notion of steady state in bacteria is an approximation, since the inducer concentration changes with the

growth cycle in vivo, glitching behavior has been observed in experiments as shown in **Figure 4.1**(c). ODE models represent the average behavior of the group of cells, so even if each cell individually may not appear to be in steady state, the population as a whole would appear to be. If there is a hazard that produces a glitch, we observe this as a population average in our model predictions.

The simulation results for circuit 0x8E are shown in **Figure 4.2**(c). This figure shows the *RNA Polymerase* (RNAP) flux from the final output promoter controlling the YFP expression (dark-blue line) in RPU [80] over time for different inducer values of *Arabinose* (Ara), *anhydrotetracycline* (aTc), and $Isopropyl\beta - D - 1 - thiogalactopyranoside$ (IPTG). The inducer molecules Ara, aTc and IPTG increase, respectively, the activity of the circuit's input promoters *pBAD*, *pTet* and *pTac*. The presence/absence of these inducer molecules is represented with the bar graph at the bottom of **Figure 4.2**(c). After each different combination of inducers is simulated, the system is restored to the default inducer values, which is the absence of inducer molecules. This sequence is selected to better compare with the original experimental results [129]. The simulations show step functions for the inducer molecules Ara, aTc, and IPTG, as is common practice. We recognize that adding/removing inducer molecules does not necessarily behave in this manner; these computational inputs are meant to resemble instantaneous addition of a chemical inducer and sudden dilution into inducer-absent media. The steady-state values reached in our dynamic simulation after each combination of inducer values as shown in **Figure 4.2**(c) match the steady state predictions made by Cello paper [129]. This means that the output of the circuit (activity of the output promoter) is considered to be *ON* and *OFF* for the same states as in the Cello paper [129]. The same has been observed for the other circuits simulated, showing that the model generator implemented in this work successfully predicts the same steady states as the steady-state modeling in the Cello project. The model generator of this work also predicts the dynamic behavior between these steady states, which is not available in the Cello paper [129]. Our dynamic simulation results are also consistent with the time-course experimental results for the 0x8E circuit, reported previously [129] (redrawn here in **Figure 4.1**(c)).

In the dynamic prediction shown in **Figure 4.2**(c), unwanted switching variations (glitches) of the activity of the output promoter can be observed before the system reaches

steady state (highlighted with the violet and red-shaded areas). For example, when transitioning from a state with *no inducers* present 0/0/0 (Ara = 0 / IPTG = 0 / aTc = 0), to the state with Ara and IPTG present (1/1/0), there is an unexpected decrease in the activity of the output promoter before reaching the *ON* steady state (**Figure 4.2**(c), violet-shaded area). This was observed experimentally [129] as shown in **Figure 4.1**((c), red line). This simulation shows that the dynamic model generator is able to predict the experimentally verified decay of YFP production, when moving through the two states. Let's consider why this happens in more detail in **Figure 4.2**(b). This graph shows the RNAP flux output RPU prediction for each gate in the circuit throughout this transition. There are three distinct regions (marked *1*, *2*, and *3*), which correspond to the initial state when Ara and IPTG are present, the intermediate state when Ara and IPTG are removed but the system hasn't reached steady state, and the end state when the system has reached a steady state. State 2 is where the glitching behavior occurs: there is a decrease in the activity of the output promoter from state 1 to 2, as shown by the shallow dip in the yellow curve (**Figure 4.2**(b), dotted line). **Figure 4.2**(a) describes, using the circuit's schematic, why this glitch is happening. The inner workings of the system are simplified to show *activated* paths as bold lines, *deactivated* paths as gray lines, and dotted lines for paths that are fluctuating activation states. Since the circuit is composed mostly of NOR gates (except for the YFP producing gate, which is an OR gate), when any path connected to the gate is activated (bold line), the gate turns OFF. When adding IPTG and Ara to the system, the circuit's output should remain high during this transition. However, the circuit experiences a decrease in the activity of the output promoter. The glitch occurs because when moving from state 1 to state 2 (see **Figure 4.2**(a)), the PhlF gate (orange gate) turns OFF, deactivating the path connected to the OR gate (yellow, YFP producing gate), before the BetI gate (violet gate) turns ON, activating the path. This delay between one gate turning OFF and the other turning ON causes a temporary decrease in the circuit output, which causes the glitch.

The automatic model generator of this work also predicts other decreases in circuit output (**Figure 4.2**(c), red-shaded areas). For example, when transitioning from a state with Ara and IPTG present (1/1/0) to a state with no inducer molecules present (0/0/0). This was neither predicted nor experimentally observed previously [129]. Another decrease in

the output promoter flux can be observed when moving from a state with Ara and aTc present (1/0/1) to a state with no inducer molecules (0/0/0).

4.1.1 Function Hazards

The simulation results and the logic of the circuit can be analyzed to better understand the cause of these glitches. The logic of this circuit is described using a truth table, shown in **Figure 4.1**(a). This truth table summarizes the steady-state behavior of the circuit for different combinations of input promoter activities. The values in the truth table indicate the activity of the output promoter (1: *High*, 0: *Low*) for different combinations of input promoter activities (1: *High*, 0: *Low*). The table also associates the input promoters *pBAD*, *pTac* and *pTet* with their relevant inducer molecules Ara, IPTG, and aTc. **Figure 4.1**(a) can be represented in a more compact form, known as a *Karnaugh map* [78] as shown in **Figure 4.3**. The first column and row of this map show all the possible combinations of input promoter activities (1: *High*, 0: *Low*), and the values indicate the steady-state RNAP flux of the output promoter (1: *High*, 0: *Low*) for the different combinations of input promoter activations. For example, when (pBAD/pTac/pTet) = (0/0/0), then the activity of the output promoter = 1; and when (pBAD/pTac/pTet) = (1/1/1), the activity of the output promoter = 0.

The Karnaugh map (**Figure 4.3**) and the dynamic simulation (**Figure 4.2**(c)) can be analyzed to determine the cause of these glitches. Let us first consider when the environment changes from a state with no inducer molecules present to a state with IPTG and aTc present, hence the activity of input promoters pTac and pTet are *high*. In the Karnaugh map, this moves the circuit from the states where (pBAD/pTac/pTet) = (0/0/0) to (0/1/1). Though the circuit is experiencing two changes in inducer concentration *simultaneously*, it may *"sense"* one concentration change before the other. When there are two changes in inducer molecule concentration to a system that occurs *simultaneously*, there are two different paths from the initial state to the end state (**Figure 4.4**(a)), depending on which inducer change the circuit *senses* first. If the circuit senses the aTc change first, then the activity of input promoter pTet increases and the circuit momentarily passes through state (0/0/1) (**Figure 4.4**(a), green line), where it momentarily evaluates to a *low* output, before reaching the final state (0/1/1), where it also evaluates to *low*. Likewise, if the circuit senses the IPTG change first, then the activity of input promoter pTac increases and the circuit

momentarily passes through state (0/1/0) (**Figure 4.4**(a), blue line), which evaluates to a *low* output as does the end state. In both of these transient states and the final state, the circuit evaluates to *low*, so the circuit makes a monotonic change from *high* to *low*, no matter which inducer change the circuit senses first. There is no possibility of a glitch, and as the simulation shows (**Figure 4.2**(c)), there is no predicted glitch behavior.

Now, let us consider a transition from a state with no inducer molecules (pBAD/pTac/pTet = 0/0/0) to a state with Ara and IPTG present (pBAD/pTac/pTet = 1/1/0). If the circuit first senses the Ara inducer molecule, then the activity of the promoter pBAD increaes and the system passes through state (1/0/0) before reaching state (1/1/0), and evaluates to *high* in all these states (**Figure 4.4**(b), green line). However, if the circuit senses IPTG before it senses Ara, then the activity of pTet increases before that of pBAD and the system momentarily passes through state (0/1/0), which evaluates to a *low* output, before reaching the end state where the output is *high* (**Figure 4.4**(b), blue line). This would produce the glitch that is observed both in the simulation (**Figure 4.2**(c)) and experimental results (**Figure 4.1**(c)). Since the order in which the inducer molecule changes are sensed affects the output behavior, this circuit has what is known in the asynchronous logic community as a *function hazard* [45, 118]. The existence of a function hazard means that regardless of how the circuit is implemented, the possibility of a glitch remains, because a function hazard is a property of the *function* and not of the circuit implementation.

However, the existence of a function hazard does not necessarily mean a glitch occurs. The transition from no inducer molecules present (pBAD/pTac/pTet = 0/0/0) to a state with Ara and aTc present (pBAD/pTac/pTet = 1/0/1) also requires two changes in inducer concentration, so there are two different paths from the initial state to the end state (**Figure 4.4**(c)). These paths lead to two different transient outputs; therefore, a function hazard exists. Yet, the glitching behavior is not observed in the simulation shown in **Figure 4.2**(c). Thus, glitching does not necessarily occur even if there is a function hazard.

Once the role of function hazards and glitches was recognized, we proceeded to identify all two- and three-input change function hazards for circuit 0x8E. **Figure 4.5** shows the dynamic simulation for all the two and three input changes that have function hazards. The figure shows many occurrences of glitching behavior (highlighted within the red-shaded areas), especially for the two-input change hazard simulation. This glitch-

ing behavior could not be predicted with the steady-state modeling and was not tested experimentally in the Cello project [129].

Glitches manifest due to variations in the propagation delay that each separate input experiences along different paths through the logic. The circuit diagram for circuit 0x8E, shown in **Figure 4.1**(b), illustrates the difference in the path lengths from sensing an inducer molecule, to the production of YFP. A glitch is observed, since there are two inducers that are changed with one going through a shorter logic path and the other going through a longer logic path. This delay can cause slower response to some inducer changes, which produces unwanted switching variations in the output. Therefore, a possible solution to remove the glitch is adding more delay to the shorter path via redundant logic to the circuit (like two successive NOT gates) as shown **Figure 4.6**. However, since these function hazards are a property of the *function* and not of the circuit implementation, the possibility of a glitch remains. **Figure 4.7** shows a comparison of the simulation results for the modified circuit (red-dashed line) with the original circuit (solid dark-blue line). The results indicate that adding the redundant logic avoids or diminishes some, but not all, glitches (blue-shaded areas), while it exacerbates others (orange-shaded areas) and, the response is generally slower (**Figure 4.7**(a) and (b), red-dashed line). This result occurs because glitches are a product of delays in the propagation of signal through the circuit, and adding more delay like in this example can cause some glitches to happen with a higher probability. Other delay mechanisms would have similar effects (i.e., it changes some glitch behavior probabilities but does not eliminate or avoid function hazards completely). Furthermore, a larger genetic circuit means an increased metabolic burden on the host, as it diverts away resources from vital processes of the cell [13, 24, 130]. However, in the future with the aid of stochastic modeling, we could quantify the effects of these delays on the probabilities of glitches, and thus use it to make design choices.

The only way to avoid function hazards is to restrict the allowed input changes to the system. Restricting input changes may be necessary if glitches lead to irreversible effects that should be avoided. As an example of this, limiting the input changes to only *single* input changes produces a very smooth and glitch-free function for circuit 0x8E, as shown by the simulation in **Figure 4.8**. In the simulation, the inducer concentrations are modified following the gray code [42], which successive values differ in only a single bit (simulating

only 8 of the 24 possible single-bit transitions). This does not necessarily mean that only single-input changes are needed to avoid function hazards. As shown previously, there are multiple input changes that do not contain function hazards. Therefore, understanding which combination of input changes do not have function hazards is extremely valuable.

4.1.2 Logic Hazards

Care should be taken though in interpreting these results. While restricting to single-input change does by definition eliminate the possibility of function hazards, a circuit implementation may still have *logic hazards*. These hazards are a property of the logic implementation rather than the function being implemented. Given that the Karnaugh map for circuit 0x8E, shown in **Figure 4.3**, shows that all the *cubes* (the ovals in the Karnaugh map) are connected, then there are no single-input change logic hazards for this circuit when it is implemented as a 2-level *sum-of-products cover* (one AND gate per cube combined with an OR gate). Unfortunately, when the logic was transformed from the user behavior specification to use only two-input NOR gates to produce the circuit shown in **Figure 4.1**(b), logic hazards were introduced. Fortunately, these logic hazards did not manifest as glitches in the simulation shown in **Figure 4.8**.

Unfortunately, a glitch does show up when simulating function hazard-free multiple-input changes. **Figure 4.9** shows the expected activity of the output promoter controlling YFP expression for all two-input changes for circuit 0x8E that *do not* contain function hazards. However, the simulation shows that the circuit still has glitching behavior (**Figure 4.9**, red-shaded area) when moving from a state with IPTG and aTc (pBAD/pTac/pTet = 0/1/1) to a state with no inducers present (pBAD/pTac/pTet = 0/0/0). This simulation demonstrates that the 0x8E circuit designed in [129] does contain logic hazards.

Logic hazards can cause undesired switching variations when there are certain input changes due to delay caused by the logic elements of the circuit (AND, OR, NOT gates, etc.), which causes the logic not to perform correctly. Fortunately, logic hazards, unlike function hazards, *can* be avoided using a logic hazard-free design procedure. As described earlier, the original 2-level sum-of-products cover is logic hazard free. In order to avoid logic hazards when transforming the circuit to use only NOT and NOR gates, one must

only apply hazard-preserving logic transformations. In particular, the logic should be manipulated using the associative law (i.e., $A + (B + C) \Leftrightarrow A + B + C$ or $A(BC) \Leftrightarrow ABC$), DeMorgan's Theorem (i.e., $\overline{(A + B)} \Leftrightarrow \overline{A}\,\overline{B}$ or $\overline{AB} \Leftrightarrow \overline{A} + \overline{B}$), the distributive law (i.e., $AB + AC \Rightarrow A(B + C)$), or the absorptive law (i.e., $A + AB \Rightarrow A$ or $A + \overline{A}B \Rightarrow A + B$). Applying other Boolean logic transformations or simplifications can lead to the introduction of logic hazards, which is what happened in the original design of the 0x8E circuit. Redesigning the 0x8E circuit using only hazard-preserving logic transformations would proceed as follows:

$$
\begin{aligned}
\text{YFP} = \;& \overline{\text{pTac}}\,\overline{\text{pTet}} + \overline{\text{pTac}}\,\text{pBad} + \overline{\text{pTet}}\,\text{pBad} && \\
\text{YFP} = \;& \overline{\text{pTac}}(\overline{\text{pTet}} + \text{pBad}) + \overline{\text{pTac}}\,\text{pBad} && \text{(Distributive Law)} \\
\text{YFP} = \;& \overline{\text{pTac} + \overline{(\overline{\text{pTet}} + \text{pBad})}} + \overline{\text{pTac}}\,\text{pBad} && \text{(DeMorgan's Theorem)} \\
\text{YFP} = \;& \overline{\text{pTac} + \overline{(\overline{\text{pTet}} + \text{pBad})}} + \overline{(\text{pTet} + \overline{\text{pBad}})} && \text{(DeMorgan's Theorem)} \qquad (4.1)
\end{aligned}
$$

The result is the logic hazard-free circuit implementation shown in **Figure 4.10**. Using the same input changes shown in **Figure 4.9**, we can now predict the activity of the output promoter for the new circuit. **Figure 4.11** shows the activity of the output promoter of the new circuit (red-dashed line) compared with that of circuit 0x8E (solid-blue line). This figure shows that the glitching behavior has been removed (**Figure 4.11**, blue-shaded area). Therefore, GDA tools should consider the use of hazard-free logic synthesis to avoid logic hazards.

Changing a circuits' implementation to avoid logic hazards does not change the function. Therefore, removing the logic hazards does not remove the function hazards, since these are a property of the circuits' *function* and are inherently unavoidable. Solving for logic hazards does not change the *funciton* of the circuit, thus it contains the same function hazards as the original circuit, since function hazards are a property of the circuits' function and are inherently unavoidable. This means that the redesigned circuit (**Figure 4.10**) has the same function hazards as the original circuit (**Figure 4.1**). It can, however, change the probability of the glitching behavior. **Figure 4.12** shows the simulation for all the two- and three-input changes that contain function hazards for the logic-hazard free circuit (**Figure 4.12**, red-dashed line) in comparison with the original circuit (**Figure 4.12**, solid-blue line). The simulation shows how changing the circuit's implementation to avoid

logic hazards solves or diminishes some of the glitches due to function hazards (blue-shaded areas) but exacerbates others (orange-shaded areas), as was the case when adding redundant logic to the circuit. This simulation shows that changing the implementation of a circuit to avoid logic hazards does not eliminate the function hazards of the circuit [118].

One caveat though is that removing logic hazards can result in a larger genetic circuit with increased metabolic burden on the host, as it diverts away resources from vital processes of the cell [13, 24, 130]. Since some glitches can be tolerated, it could prove useful to add gate toxicity as a measure of the burden on cell economics for each gate, and thus be able to calculate the total burden on a cell for each circuit to help designers decide on the best possible circuit layout for their intended purposes.

4.2 Hold-State Failures and Set-Up Glitches

The expanded circuit failures analyzed in this work (Chapter 6) are *set-up* and *hold-states* failures. Set-up failures are glitches produced when a circuit is initialized and its components have not been stabilized yet, since they all initialize with no production at all. This means that a genetic circuit may produce the incorrect output once it has been initialized, before reaching the expected outcome at steady-state. Hold-state failures, on the other hand, is the incapacity of a genetic circuit to hold the correct/expected state for a constant input concentration due to random and sporadic changes in the inner circuit components' concentration.

4.3 Proposed Hazard Analysis

This work utilizes a model generator that automatically creates a dynamic model that produces predictions that can be used to analyze the behavior of a circuit over time. The dynamic behavior of a circuit can be used to identify input transition sequences that result in glitching behavior. Additionally, comparing the model results with experimental data can help to understand the underlying biological phenomena. This work also shows how dynamic modeling can expose unwanted transition states or the time it takes to reach said stable state, which is not provided by steady-state analysis.

A system with a function hazard always has the potential to glitch for the specified input change. Modifying the logic implementation can only change the likelihood of

glitching, but not eliminate the possibility of it occurring [17, 66, 118]. For example, this work demonstrates how adding redundant logic to increase the delay of the circuit's shorter path may reduce the likelihood of some glitches while increasing it for others. Function hazards are inherently unavoidable [118] unless one restricts the allowed input changes to the system to include only single-input changes and a restricted set of multiple input changes. Therefore, when designing genetic circuits, it would be useful to specify the desired sequences of input and output changes rather than just the state for each input combination. In other words, designs where glitching behavior must be avoided would follow an asynchronous state machine design style, which would avoid the possibility of function hazards. Furthermore, states that are found to either not be reachable or are not final states of transitions may introduce *don't cares* (i.e. states in which the design is allowed to take either output value) into the design process that can be specified to avoid the introduction of function hazards. The analysis of function and logic hazards can help both the design process of a genetic circuit, as well as the tools used to automatically generate these circuits. In particular, the analysis can help understand which input transitions with function hazards are likely to produce glitches so designers can decide which input transitions must be avoided.

Logic hazards, on the other hand, can be avoided using logic synthesis with hazard-preserving optimizations [118]. This work shows how a circuit designed using a GDA tool like Cello [129] can have logic hazards and in future work, we plan to develop a genetic circuit logic synthesis method that avoids these hazards. In particular, tools can use design restrictions and use hazard-preserving transformations in order to prevent certain hazards that are deemed critical for the user.

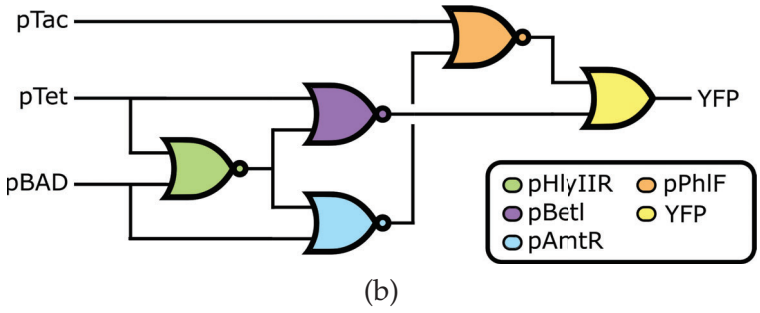
The dynamic model generator presented in this work produces ODEs-based models, which is well suited for *average* (population) response simulation to input changes. However, even single-strain cell populations can exhibit a high degree of variation of gene expression for the same environment [6]. So even if ODEs simulation predicts that there is little or no glitching behavior for certain input changes, it might be the case that a significant percentage of a homogeneous population does manifest the unwanted switching behavior. All Cello circuits with more than a single input have function hazards, and many of these hazards can turn into glitches. However, to fully understand glitches and their

probabilities, a stochastic analysis is required. Beal [6] showed that this cell-cell variation may be accounted for by the emergent properties of complex reaction networks, which drive a log-normal distribution of genetic expression levels across a population. Drawing from the measured parameter distributions of the Cello part library, we can implement the log-normal model proposed by Beal [6] to calculate the incidence of glitching behavior in a population. This in turn can help designers understand what the risks of applying certain input changes are and decide whether the risk is critical or not for the intended purposes of the designed systems. Furthermore, the percentage population that express glitching behavior could potentially serve as a standard metric for genetic circuit design purposes.

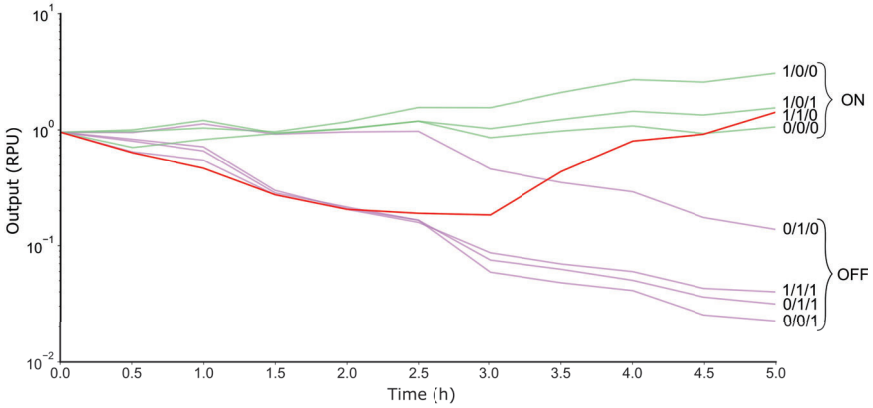
This new model generation method bridges the gap between experimentalists and designers as it helps both sides with the results obtained. Designers can use data to better fit the model to produce more accurate predictions, and experimentalists can use these predictions to debug genetic circuits and predict their behavior before constructing them, saving time, effort, and money using an automated workflow as shown in **Figure 4.13**.

Inputs			Output
Ara	IPTG	aTc	YFP
pBAD	pTac	pTet	yYFP
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

(a)



(b)



(c)

Figure 4.1. Time-course data of Cello Circuit. **(a)** Truth table for circuit 0x8E. The steady-state RNAP flux output from the final output promoter controlling YFP expression (yYFP, 4th column), is matched to each combination of input promoters (pBAD, pTac, pTet) or inducer concentrations (Ara, IPTG, aTc). A *low* input/output is represented with a 0 and *high* input/output is represented with a 1. **(b)** Circuit diagram for circuit 0x8E adapted from Nielsen et al. [129]. In this image ⊃D○ represents a NOR gate and ⊃D represents an OR gate. pBAD, pTac, and pTet are the input promoters for this circuit, and *YFP* is the output reporter. **(c)** Time-course data for circuit 0x8E that was originally reported in the Cello paper [129]. Each line represents the output YFP production (in RPU) over time (in hours) for the circuit 0x8E for each combination of input promoters. This circuit senses three inducer molecules: Ara, aTc, and IPTG. In the image, 1/1/1 (Ara/IPTG/aTc) represents all inducer molecules are present and 0/0/0 represents no inducer molecules are present. The *ON* and *OFF* states represent the predicted outcome at steady state, in where ON means an expected *HIGH* output (green lines), and OFF means an expected *LOW* output (purple lines). All outputs behave as expected, except for the 1/1/0 state, which experiences an undesirable decay before rising to the *ON* state (red line).

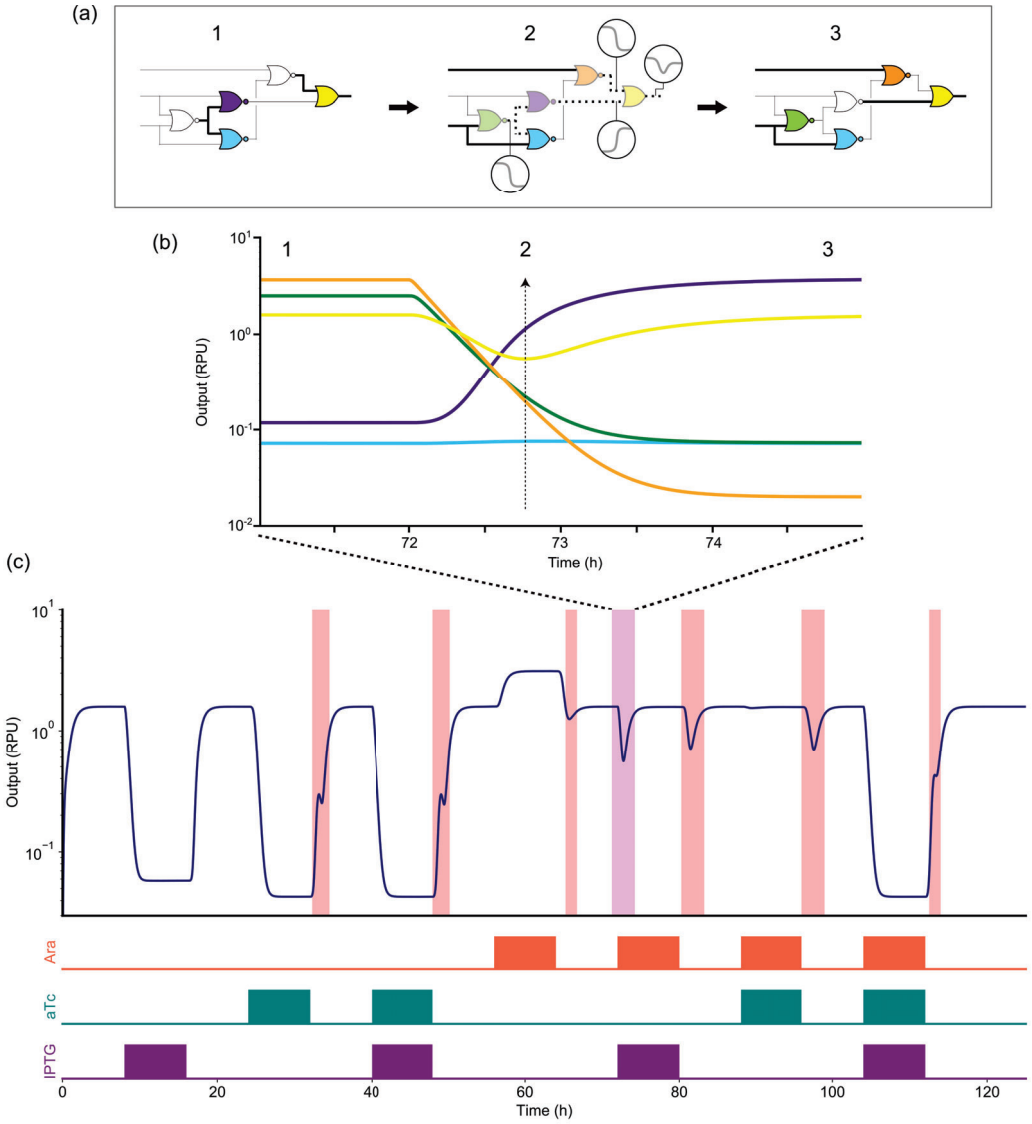


Figure 4.2. Circuit 0x8E [129] simulation. **(a)** Schematic of the circuit for each state (*1*, *2*, and *3*) of the middle graph. Bold lines represent that the path is *activated*, gray lines show that the path is *deactivated*, and dotted lines mean there is a fluctuation in the activation state (change in the RNAP flux). Given that the orange gate (PhlF) deactivates before the purple gate (BetI) activates, a small decay in the production of YFP is produced before it reaches steady state. **(b)** A detailed overview of an unwanted transient behavior (glitch) is shown with the RNAP flux output (y_i) in RPU at log scale over time (in hours) of each gate of the circuit. The color of each curve matches the gate color in (a), that is: YFP (yellow line), y_{AmtR} (light-blue line), y_{BetI} (violet line), y_{PhlF} (orange line), and y_{HlyIIR} (green line). A small depression can be observed for the YFP output (state 2) before reaching steady state (state 3). **(c)** Simulation of circuit 0x8E for each combination of inducer molecule values (IPTG, aTc, Ara). RNAP flux from the final output promoter controlling the YFP expression (in RPU using a logarithmic scale, dark-blue line) over time (in hours). Red- and violet-shaded areas show predicted glitching behavior.

pBAD \ pTac pTet	0 0	0 1	1 1	1 0
0	1	0	0	0
1	1	1	0	1

Figure 4.3. Karnaugh map for circuit 0x8E [129]. A *low* input/output is represented with a 0 and *high* input/output is represented with a 1. The header row and first column represent the different combinations of input promoter activations, and the values of the table represent the steady-state activity of the output promoter of the circuit. The ovals represent the minimal *Sum-of-Products* (SOP) cover of this function.

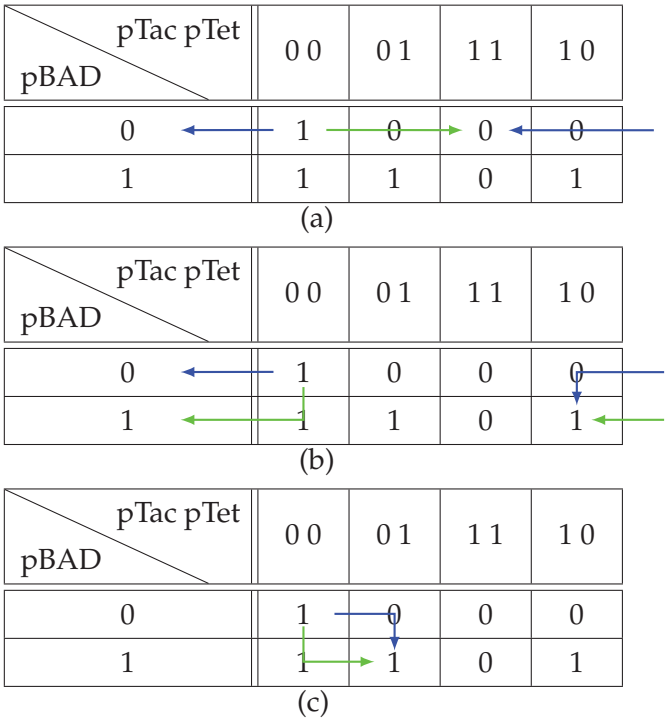
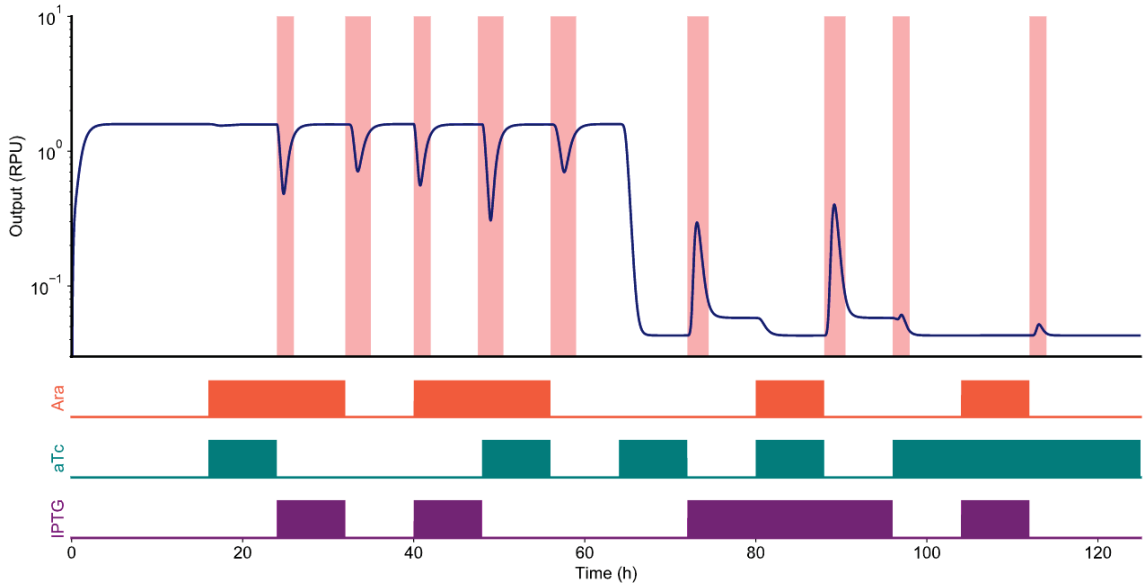
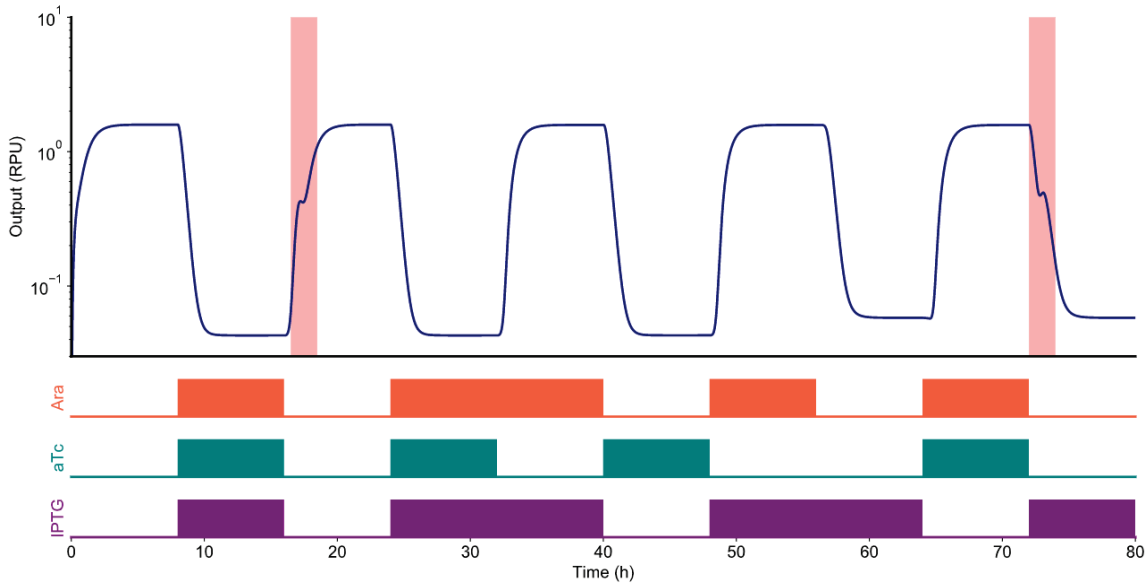


Figure 4.4. Using a Karnaugh map for circuit 0x8E to analyze function hazards. Green and blue arrows represent the different paths a system can transition to when moving from the initial state to the end state. **(a)** Moving from state (0/0/0) to state (0/1/1), no function hazard and no glitch observed. **(b)** Moving from state (0/0/0) to state (1/1/0), function hazard present and glitch observed. **(c)** Moving from state (0/0/0) to state (1/0/1), function hazard present but no glitch observed.



(a)



(b)

Figure 4.5. Simulation of all two- and three-input changes that have function hazards. Simulations show the activity of the output promoter controlling YFP production (in RPU at log scale) over time (in hours) for circuit 0x8E. Red-shaded areas show predicted glitch behaviors. **(a)** Two-input change function hazard simulation for circuit 0x8E. **(b)** Three-input change function hazard simulation for circuit 0x8E.

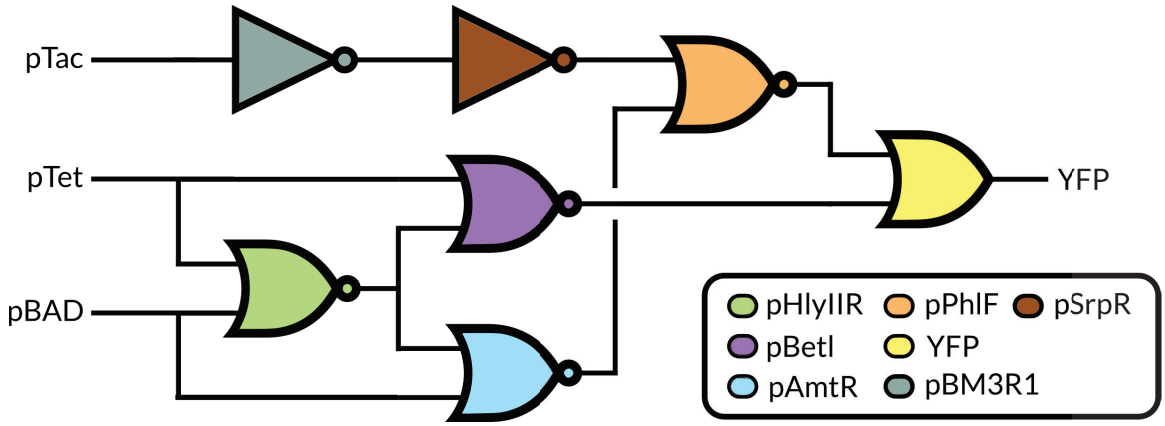
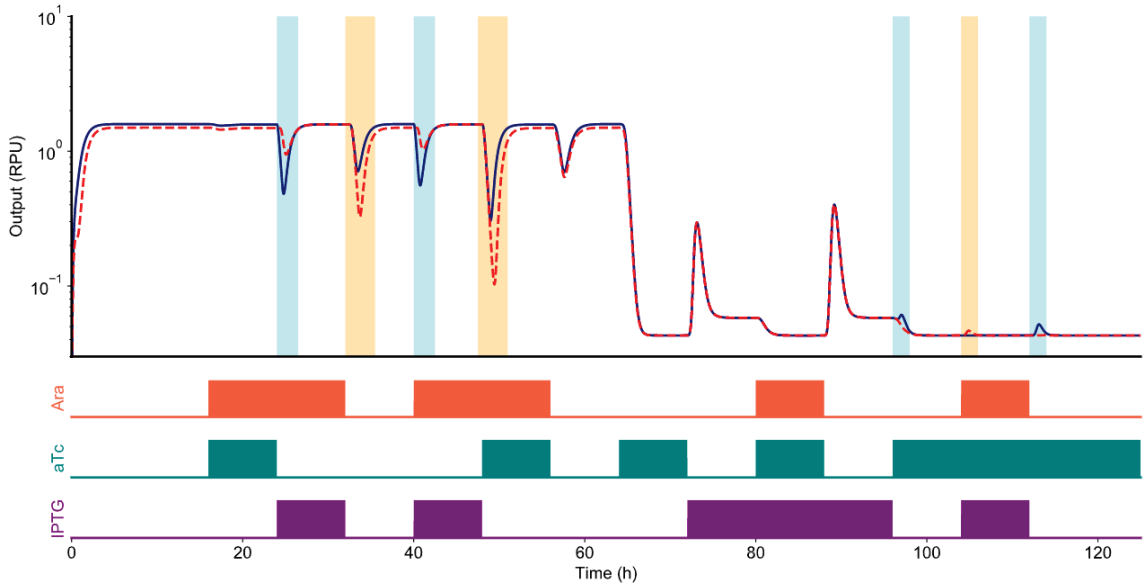
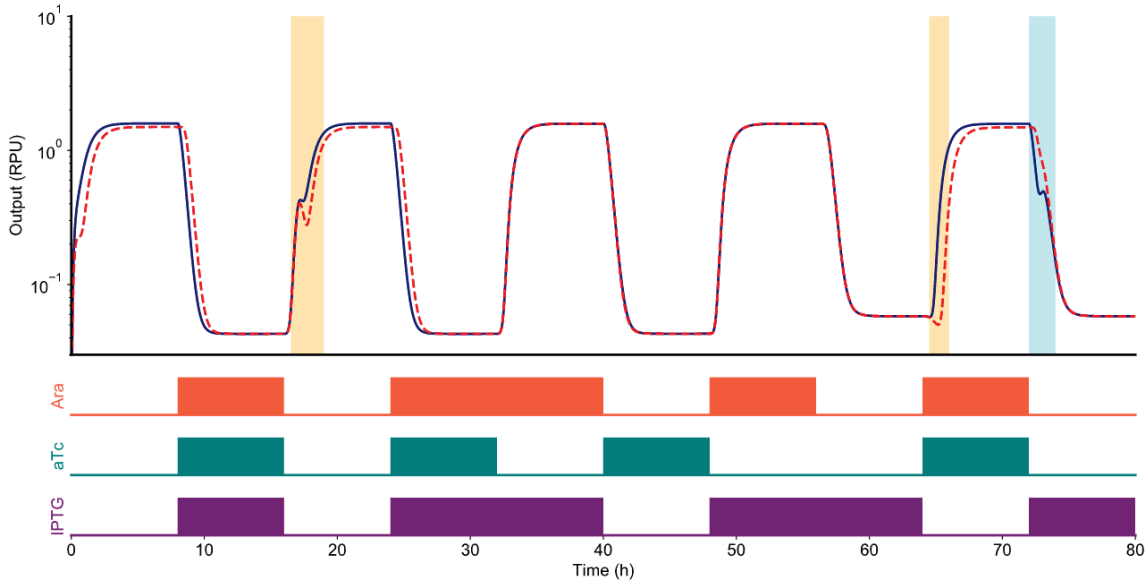


Figure 4.6. Circuit diagram for circuit 0x8E with redundant logic. In this image ⊐D○ represents a NOR gate, ⊐▷○ represents a NOT gate, and ⊐D⊐ represents an OR gate. pBAD, pTac, and pTet are the input promoters for this circuit, and *YFP* is the output reporter.



(a)



(b)

Figure 4.7. Comparison of two- and three-input change function hazard simulation for circuit 0x8E with and without redundant delay logic added. Simulation comparison of the activities of the output promoter controlling YFP production (in RPU at log scale) over time (in hours) between circuit 0x8E (dark-blue line) and circuit 0x8E with redundant logic (red-dashed line). Blue-shaded areas show "solved" or diminished glitches, orange-shaded areas show new or more profound glitching behavior. **(a)** Simulation for all two-input change function hazards of the circuits. **(b)** Simulation for all three-input change function hazards of the circuits.

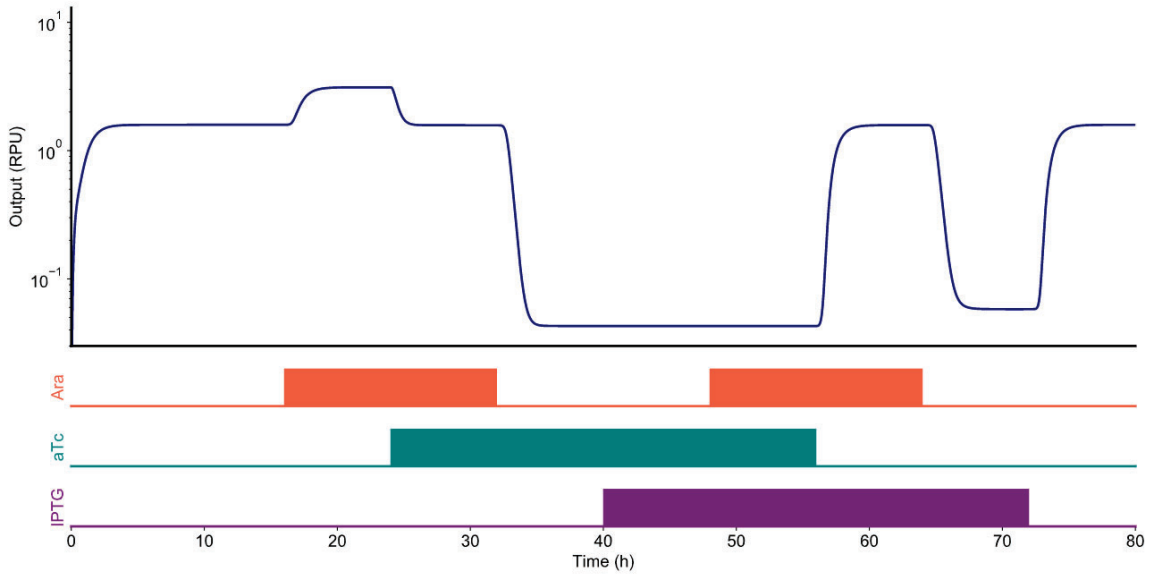


Figure 4.8. Single input change simulation for circuit 0x8E. Activity of the output promoter controlling YFP production (in RPU at log scale) over time (in hours) for circuit 0x8E during a gray code sequence of single-input changes. For this simulation, the circuit is subjected to different states while only changing the concentration of one inducer at a time.

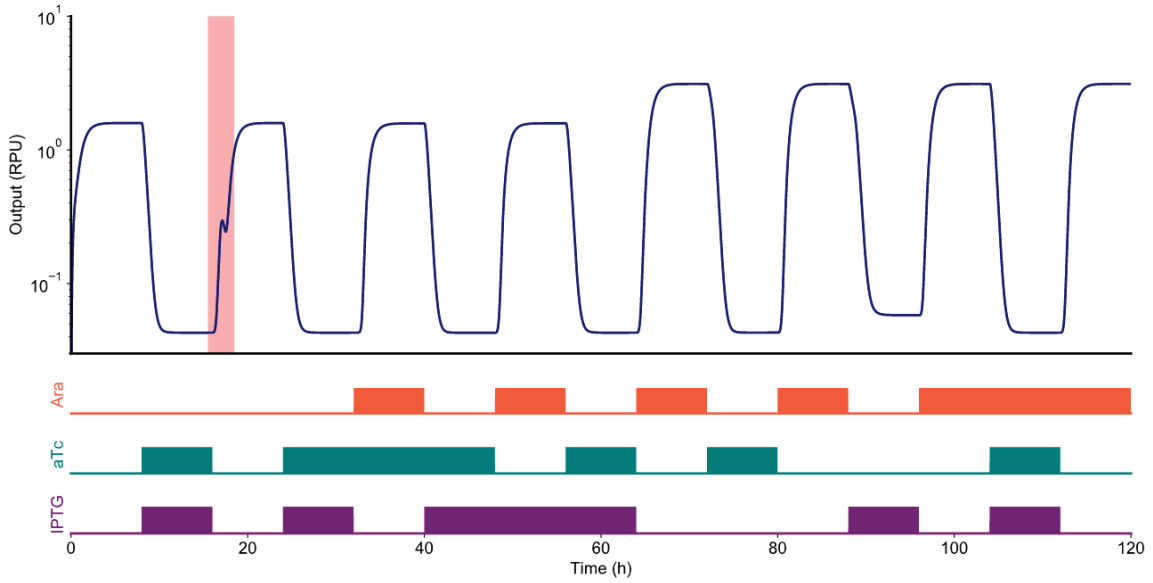


Figure 4.9. Simulation of all two-input changes that do not have function hazards for circuit 0x8E. Simulations show the activity of the output promoter controlling YFP production (in RPU at log scale) over time (in hours) for circuit 0x8E. Red-shaded area shows a predicted glitch behavior.

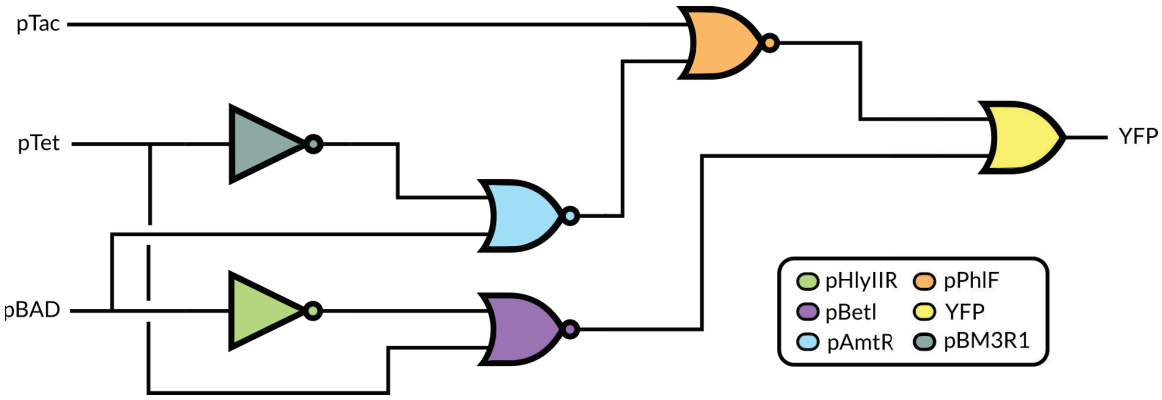


Figure 4.10. A logic hazard free adaptation of 0x8E circuit. In this image ⊐D- represents a NOR gate, -▷- represents a NOT gate, and ⊐D- represents an OR gate. pBAD, pTac, and pTet are the input promoters for this circuit, and *YFP* is the output reporter.

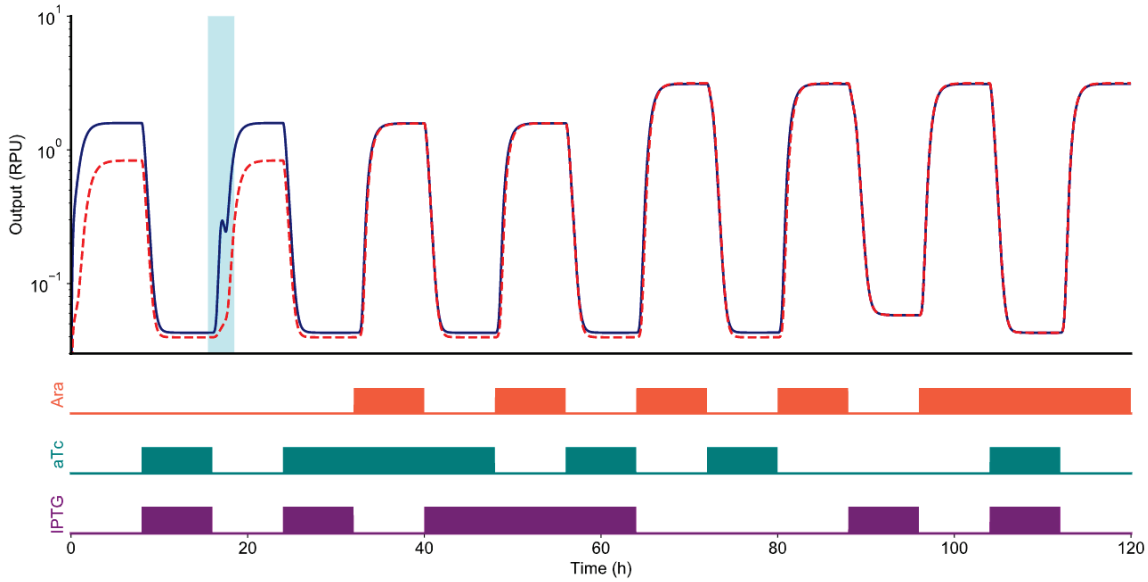
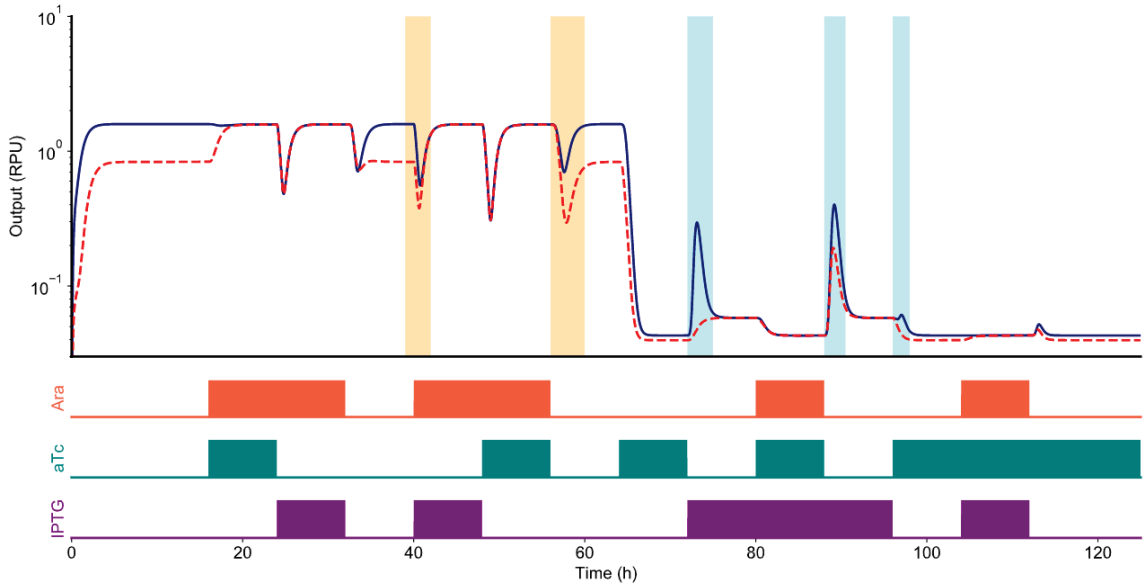
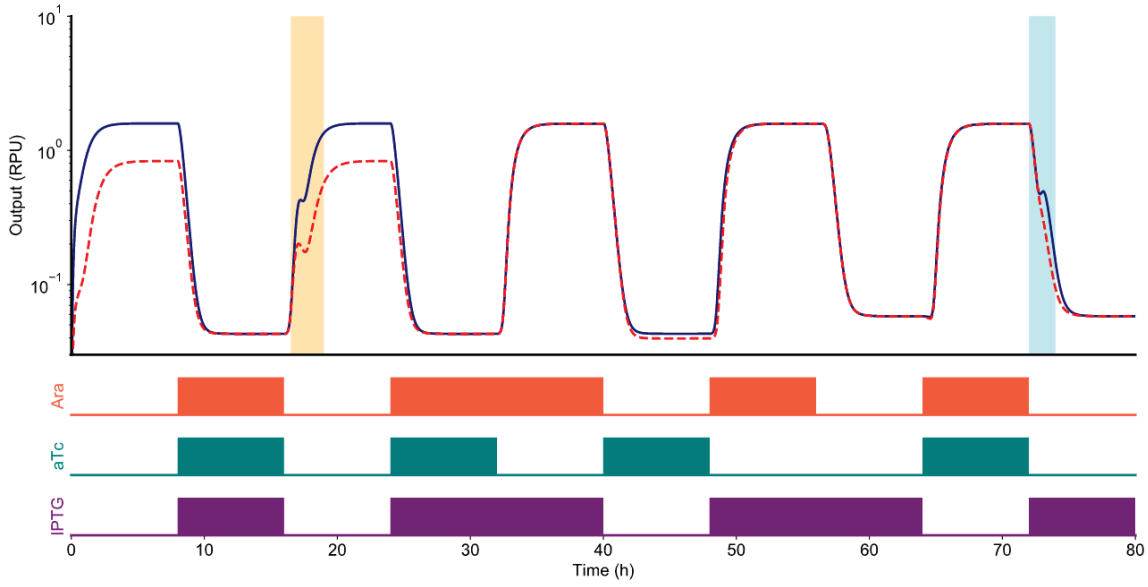


Figure 4.11. Simulation all two-input changes that do not have function hazards for the logic hazard free adaptation of circuit 0x8E. Simulations show the activity of the output promoter controlling YFP production (in RPU at log scale) over time (in hours) for the circuit. The blue line indicates the results for the original 0x8E circuit, while the red line shows the results for the logic hazard-free circuit. The blue shaded area shows that the glitch has been removed using logic hazard-free design.



(a)



(b)

Figure 4.12. Comparison of two- and three-input change function hazard simulation for circuit 0x8E with and without solved logic hazards. Simulation comparison of the activities of the output promoter controlling YFP production (in RPU at log scale) over time (in hours) between circuit 0x8E (dark-blue line) and circuit 0x8E redesigned without logic hazards (red-dashed line). Blue-shaded areas show "solved" or diminished glitches, orange-shaded areas show new or more profound glitching behavior. **(a)** Simulation for all two-input change function hazards of the circuits. **(b)** Simulation for all three-input change function hazards of the circuits.

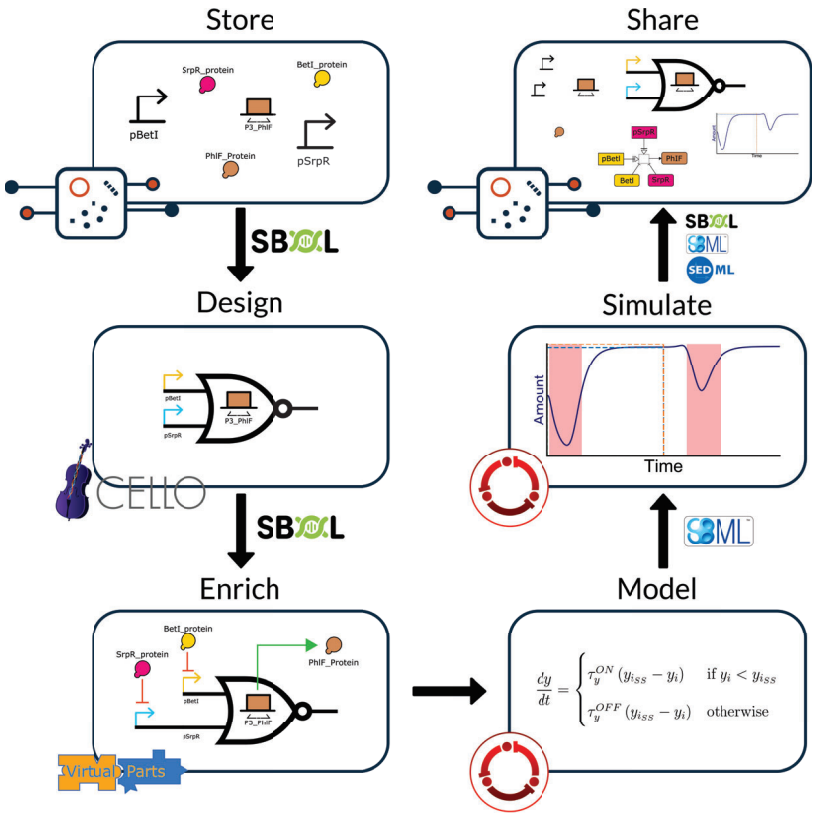


Figure 4.13. Workflow showing the automatic dynamic model generator instantiated in iBioSim.

CHAPTER 5

SIMULATING NOISE FOR GENETIC REGULATORY NETWORKS IN IBIOSIM

As we move from proof-of-concept circuit designs to more real-life applications, a circuits' robustness to a changing environment and other noise factors that may induce circuit errors is of critical importance for the safe and controlled application of designed systems. Both stochasticity inherent in the biochemical process of gene expression (*intrinsic noise*) and fluctuations in other cellular components (*extrinsic noise*) can contribute substantially to overall observed phenotypic variation [47], and depending on which is predominant, different stochastic analysis may ensue. The fundamentally stochastic nature of the interactions of these molecular participants requires stochastic modeling and analysis [31]. Furthermore, for systems where transcription factors, enzymes, and DNA copies can exist at a low concentration, such as a single molecule per cell, any realistic analysis of these systems must include the cell-to-cell variability in transcriptional outputs captured by the chemical master equation [63]. However, for systems with strong genetic expression, the stochastic variation in gene expression, like transcription factor producing genes, may come from fluctuations in the quantity or states of other cellular components [6, 158], which is best modeled as probability density functions for different reaction rates [132].

The current state of modeling gene network dynamics is characterized by a trade-off between the model's ability to quantitatively match the experimental data, and the need for a large number of kinetic parameters to parameterize the model (Karlebach and Shamir, 2008; Heatha and Kavria, 2009; Machado et al., 2011; Goncalves et al., 2013). Properly parameterized ordinary differential equation models can provide a good quantitative match and are easily generalized (Chen et al., 2004; Tyson and Novak, 2013). However, numerical simulation of these models require knowledge of kinetic parameters that are usually not known. The indirect estimate of these parameters by comparing the output of

the model to the experimental data suffers from at least three fundamental problems: (i) the correspondence between dynamics and the structure of the network is not one-to-one; (ii) the need to match data corrupted by significant intrinsic and experimental noise to an individual solution of the *Ordinary Differential Equation* (ODE) model; and (iii) the lack of methods to search high dimensional parameter spaces for dynamic signatures observed in the data.

Using accurate modeling to predict a *Genetic Regulatory Network*'s (GRN) robustness is of great advantage for designed circuits that operate with sensitive outputs or safety-critical products. However, how much more information do different models and parametrization efforts provide to design choices based on robustness? Developing complex models to produce more accurate predictions, and characterization experiments to obtain meaningful parameter values is time-consuming and costly. Moreover, determining the correct model to represent a specific biological system and developing fitting algorithms to re-parameterize each component requires expertise in the subject, baring many designers in pursuing this endeavor. Furthermore, there is a simulation-time cost associated with these models: the more complex a model is, the longer it takes for simulations to run or the harder it is to model-checking on them [20, 125, 139]. Do different models, and/or parameter values obtained from characterization experiments lead to different design choices with regard to robustness? Are the costs related to having more predictive power overcome by more informed design choices?

This chapter investigates robustness and predictability for genetic circuits *in silico*, using different models and levels of characterization for three different circuit layouts with identical expected functions (**Figure 5.1**), to observe if there are differences in predicted robustness. This work predicts robustness by evaluating the likelihood of unwanted switching variations in a genetic circuit's output both when changing states, its ability to hold a steady-state, the probability of an incorrect steady-state, and its likelihood of failure. The circuit's predictability is analyzed by simulating its response to extrinsic noise for different models, and using parameter values obtained from literature or from characterization experiments.

Designers can use the method in this work to determine circuit failure probabilities, and, ultimately, go back to the drawing board if the failure propensities calculated are

deemed critical for the expected safe application of the designed circuits. Designers can also redesign circuits (without changing function) and compare results to observe which one fairs better for the states and input changes the researcher is interested in. This, in turn, will help designers save time and effort exploring different design choices before building them in the lab for testing.

5.1 Simulating Extrinsic Noise

Extrinsic sources of transcriptional variability refer to cell-to-cell differences in the transcriptional inputs as well as the transcriptional machinery itself [144]. Beal [6] showed that this cell-cell variation might be accounted for by the emergent properties of complex reaction networks, which drive a probability distribution of genetic expression levels across a population. For this work, we can draw from the Cello part library's measured parameter distributions, and implement a normal-distributed parameter value model to calculate the incidence of glitching behavior in a population.

The extrinsic noise model used in this work applies a simple case of *static* external perturbations, modeled as a random draw from a folded *normal* distribution for each parameter value used in the model at the beginning of each simulation run. The mean of each distribution is the default parameter value in iBioSim (obtained from literature), with a standard deviation equal to forty percent of the mean's absolute value (which emulates the "extrinsic noise"). This value of noise was obtained when calibrating different noise values, but is an arbitrary value that should be replaced with a better estimate obtained from experimentation (see Section 5.5). Beal [6] argues that these parameters follow a geometric distribution instead of a normal one, which is also part of our planned future work.

5.2 Model Selection and Parameter Values

The two models used for the extrinsic noise model comparisons are the model developed in [129, 152], explained in detail in Chapter 4, and the default model generated in iBioSim [169]. This default model generates reactions for the transcription, translation, protein-biding, protein degradation, transcriptor-function protein interactions and binding, repression, and activation reactions. It is a much more detailed model than that

used in [129, 152]. However, a more detailed model is generally harder to characterize (to obtain empirical parameters values), as many of these reaction constants are hard to measure experimentally. In addition, a comparison between default (obtained from literature) model parameter values and characterized gate parameter values were used to determine the effect on predicted circuit failure percentages. The default parameter values were used for both the *default* model in iBioSim [169], and the *Cello* model published in [129, 152]; and part-characterized model parameter values for each component obtained from experimentation [129] [1]. These comparisons are going to be used for the three different circuit layouts shown in **Figure 5.1**.

5.3 Considerations/Assumptions

This work compares which circuit layout does each noise model, given different modeling alternatives and characterizations, predict to be the most robust for certain input transitions or state-holding capacities. However, it is out of this works' scope to determine what noise-source is a more accurate representation of the true GRN behavior, or what is the magnitude of each noise source's influence on the predicted output. The main objective is to determine if there *are* any differences in robustness predictions if we use abstracted models or not, and/or if we use characterized parameter values or literature-obtained values. Therefore, certain assumptions were made which are listed as follows:

1. The magnitude of noise level for the normal distribution was chosen to be = 0.4, which came from initial testing. However, the absolute value (magnitude) of intrinsic or extrinsic noise could be different for GRNs, and if results show that there are differences in robustness predictions, should be measured for more accurate results.
2. We assume the probability distribution for *extrinsic* noise follows a normal distribution. However, Beal [6] argues that this noise is better described using a log-normal distribution. For the purpose of this work, we chose the truncated-normal distribution for parameter values to simulate extrinsic noise for simplicity and celerity of simulations, though further work could include log-normal distributions for these.
3. The level of extrinsic noise could be different for each reaction (therefore have different effect on a parameter's value distribution). However, for this work, we

[1] https://synbiohub.programmingbiology.org/public/Eco1C1G1T1/Eco1C1G1T1_collection/1

are assuming that the magnitude of noise is the same for each reaction parameter.

4. For all simulations runs, we assume that the change in input molecule concentrations is instantaneous, instead of being a gradual process.
5. The Cello model described in Nielsen et al, and Shin et al. [129, 152] can use τ_{ON} and τ_{OFF} parameters to describe how quickly a gate turns ON or OFF. However, for this work we are assuming all the different parts have the same values of τ_{ON} and τ_{OFF} given that there are no experimental parameter values for them.

However, if these results do determine that there are differences in robustness' predictions, then it calls for further investigation to determine the validity of these assumptions, given that it might affect a designer's choices.

5.4 Results

Following are the simulation results of each model, with or without experimental parameter values, for the original design circuit layout (**Table 5.1**), the two-inverted circuit design (**Table 5.2**), and the logic-hazard free circuit layout (**Table 5.3**). The results show that for each different layout, there is a significant difference in circuit failure percentage predictions for the different models used, whether it is parameterized or not. This could largely be due to the *ON* or *OFF* constraint values or simulation times chosen to determine if a circuit failed for each model, since the output fluorescence *arbitrary units* (a.u.) predicted and simulation time points changes for the different models. This means that *if* noise has a meaningful effect on a GRN's output, then characterization of the magnitude, source of noise (whether intrinsic or extrinsic), and the probability of distribution of parameter values due to noise is critical for the accurate prediction of circuit failures. Furthermore, the different models (default vs. Cello) also have significant predicted circuit failure differences, meaning that the model a designer chooses to encode behavior does change the predicted outcomes of a GRN. However, to further understand if the choice of model, or the use of characterized parameter values or not, changes the choice of the most "robust" circuit layout, a comparison of each models' circuit failure predictions for the different circuit layouts should be studied.

Table 5.4 summarizes, using a color scheme, which circuit layout is predicted to be more robust for each model variation and input transition or steady-state. In this table, a

cell is colored ‘red‘ if it performed worse than the median for that transition (meaning if the predicted percentage failure is > 1.1 median value for that transition), and is colored ‘green‘ if it performed better (predicted percentage failures is < 0.9 median value for that transition). So, for example, if we consider the default model (with default parameters) to be the “correct” model for our GRNs, then the logic-hazard free circuit layout has a lower predicted probability of failing for the studied input transitions or hold-states, and the two-inverter circuit layout has the highest predicted failure percentages. Therefore, if we would choose this model to guide our design choices, we would go for the logic-hazard free circuit layout. However, if the Cello model (with characterized parameters) is assumed to be the most “accurate” or “correct” model for our circuits, then our choice of most robust circuit layout changes. In this case, the original design circuit layout would be chosen to be the more robust one.

5.5 Discussion

The *Design-Build-Test-Scale* (DBTS) workflow is critical in synthetic biology, with recent efforts to automate most of these steps to open the field to a broader, diversely skilled community. Efforts to develop better models and parameter values can produce more accurate circuit output predictions. However, the learn step (re-parameterizing characterized gates and model exploration) is still a very manual process that requires much expertise and skill. Therefore, attempts to obtain more detailed model predictions is usually dampened by this challenge.

This work focuses on determining if there are differences in predicted circuit failure percentages for three different circuit layouts with identical expected functions, using different model choices and parameter values. The results show that each circuit layouts’ predicted percentage failures changes for each model, using characterized parameters or default values. This means that the choice of model, or the use of characterized gates parameters or not, determines whether a circuit is predicted to fail, for example, 84% or 36% of the times for a given input transition. However, since the exact values of these percentages has to be determined by further experimentation (to determine the magnitude and source of noise, or the model appropriate for the system), this is a more qualitative than quantitative result. Nonetheless, **Table 5.4** demonstrates that the choice of model, or

the use of characterized parameters or not, has an effect on possible design *choices*. In this work, for example, a designer using a default model with default parameters may decide that the logic-hazard-free circuit behaves more robustly for the input transitions studied, and a designer using Cello modeling with characterized gate parameters, may decide that the original design layout would be the most robust one.

The results shown in this work emphasize that correct modeling and choice of parameters can influence our design choices; therefore, it follows that further studies to determine the most accurate models and parameter values should be used to guide design choices. However, this endeavor is time-consuming and costly. The more complex a model is, the harder it will be to characterize experimentally and the longer it would take to simulate, especially if the designer is considering different experimental conditions for a broader Test phase. Hence, finding the proper balance between effort and predictability varies greatly with the intended use of the GRN being designed, funding and time availability, or even resources available for the experimentalists. If the intended purpose of the designed GRN is just to monitor some other process, without producing an irreversible effect on other systems, then maybe a simpler model with literature-obtained parameters would suffice. However, for safety-critical GRNs' outputs, then determining these values would be essential. This is why, for this dissertation's work, we decided to model most of the circuits analyzed using the Cello modeling [129, 152]: it is a simpler model that bundles many reactions into one, simplifying the reaction-rate parameters needed to measure, as well as providing easier characterization experiments to determine their values. Furthermore, as there are fewer reactions, it is quicker to simulate (in this work, the default model simulations took around three hours to simulate, whereas Cello models take minutes). Moreover, since Cello models are used as a benchmark for many other experimental published works [4, 41, 61, 71, 113, 146, 152, 170], there is a ready availability of gate parametrization values for different laboratory conditions.

In this work, an extrinsic noise model was used to simulate variability across different populations. However, the different assumptions used for this experiment are multiple, and a further evaluation of their validity must ensue in order to provide accurate predictions of variability, and hence, circuit failure probabilities. First, further studies to determine the *source* and *magnitude* of noise for specific GRNs [6, 89, 132, 138, 149, 153, 158,

160], since the magnitude of each noise source depends on the biological network being studied; second, how to correctly model each source of noise has to be ascertained. For example, Beal [6] argues that extrinsic noise should be modeled as a log-normal probability distribution on parameter values, instead of a normal distribution as used in this work. Noise models can also serve as a design tool, in where a designer could increase the magnitude of simulated noise to evaluate which circuit topologies is best suited for the intended purposes of a design, and which ones are more "robust" to external sources of noise and variation.

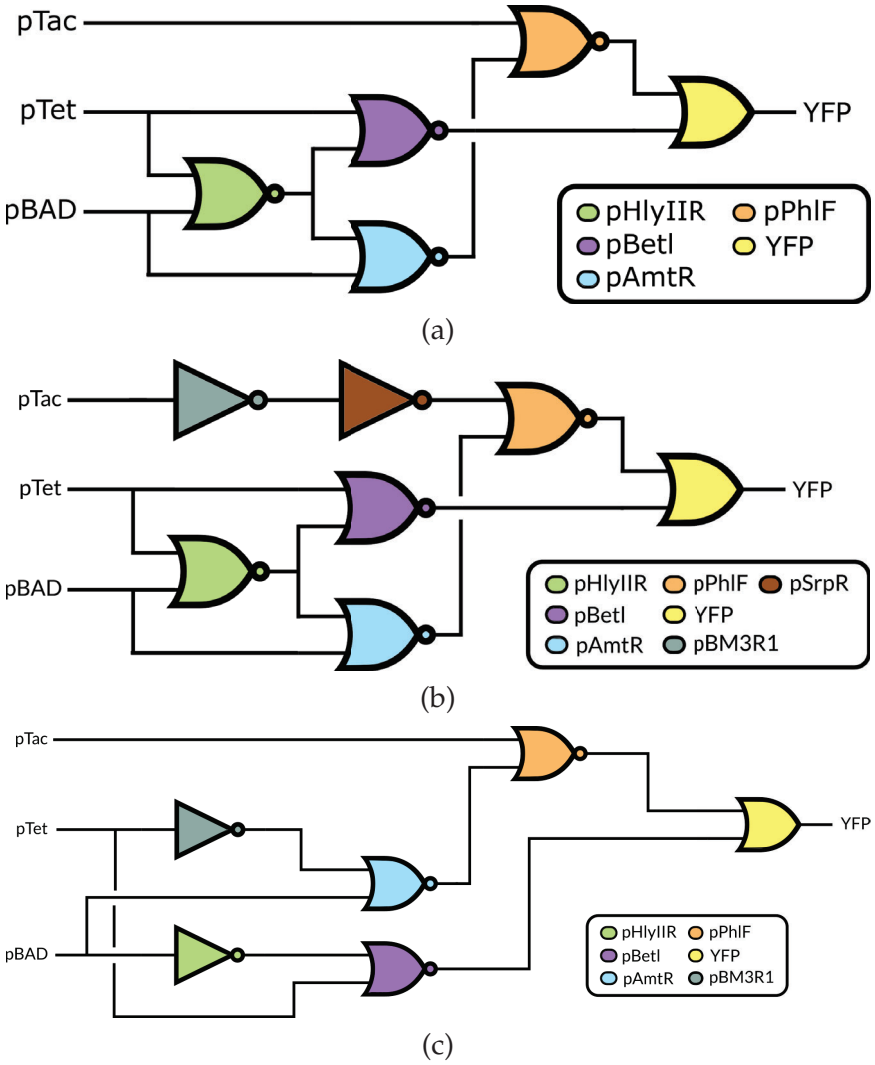


Figure 5.1. Three different logic layouts for the circuit 0x8E [129]. The three inducer molecules are *IPTG*, *aTc* and *Ara* and the output is *YFP*. The OR gate is represented by and the NOR gate by . (a) Original circuit layout as published in [129]. (b) Circuit implementation with added redundant logic as two NOT gates, which add an extra delay to the IPTG pathway. The NOT gate is represented by . (c) Circuit implementation with logic-hazard-free optimizations. More details on the implementations (b) and (c) can be found in [55].

Table 5.1. Percentage circuit failure results for the function hazard and hold-state failure analysis of the original design of circuit 0x8E. **D/D**: default model with default parameters; **C/D**: Cello model with default parameters; **C/C**: Cello model with characterized parameters.

		Original Design		
Circuit Failure	**Input**	D/D	C/D	C/C
0-Function Hazards	$(0,1,0) \rightarrow (1,1,1)$	0.09	0.21	0.23
	$(0,1,0) \rightarrow (1,0,0)$	0.36	0.49	0.84
	$(1,1,1) \rightarrow (1,0,0)$	0.48	0.59	0.76
	$(1,1,1) \rightarrow (0,1,0)$	0.36	0.43	0.41
	$(1,0,0) \rightarrow (0,1,0)$	0.33	0.45	0.53
	$(1,0,0) \rightarrow (1,1,1)$	0.13	0.16	0.30
1-Function Hazards	$(0,1,1) \rightarrow (1,0,1)$	0.85	0.57	0.12
	$(0,0,0) \rightarrow (0,1,1)$	0.37	0.15	0.12
	$(0,0,0) \rightarrow (1,0,1)$	0.82	0.43	0.11
	$(1,0,1) \rightarrow (0,1,1)$	0.83	0.54	0.25
	$(0,1,1) \rightarrow (0,0,0)$	0.52	0.24	0.13
	$(1,0,1) \rightarrow (0,0,0)$	0.59	0.32	0.15
Wrong Steady State	$(0,0,0)$	0.31	0.13	0.09
	$(0,0,1)$	0.15	0.00	0.01
	$(0,1,0)$	0.09	0.27	0.27
	$(0,1,1)$	0.30	0.10	0.11
	$(1,0,0)$	0.13	0.23	0.35
	$(1,0,1)$	0.37	0.14	0.05
	$(1,1,0)$	0.05	0.13	0.16
	$(1,1,1)$	0.05	0.15	0.18

Table 5.2. Percentage circuit failure results for the function hazard and hold-state failure analysis of the two-inverter design of circuit 0x8E. **D/D**: default model with default parameters; **C/D**: Cello model with default parameters; **C/C**: Cello model with characterized parameters.

		Two-Inverter Design		
Circuit Failure	**Input**	D/D	C/D	C/C
0-Function Hazards	$(0,1,0) \rightarrow (1,1,1)$	0.27	0.50	0.47
	$(0,1,0) \rightarrow (1,0,0)$	0.34	0.58	0.89
	$(1,1,1) \rightarrow (1,0,0)$	0.38	0.58	0.89
	$(1,1,1) \rightarrow (0,1,0)$	0.19	0.34	0.40
	$(1,0,0) \rightarrow (0,1,0)$	0.20	0.37	0.45
	$(1,0,0) \rightarrow (1,1,1)$	0.15	0.20	0.38
1-Function Hazards	$(0,1,1) \rightarrow (1,0,1)$	0.64	0.32	0.09
	$(0,0,0) \rightarrow (0,1,1)$	0.55	0.24	0.24
	$(0,0,0) \rightarrow (1,0,1)$	0.65	0.31	0.07
	$(1,0,1) \rightarrow (0,1,1)$	0.88	0.83	0.69
	$(0,1,1) \rightarrow (0,0,0)$	0.64	0.38	0.28
	$(1,0,1) \rightarrow (0,0,0)$	0.73	0.59	0.43
Wrong Steady State	$(0,0,0)$	0.41	0.24	0.23
	$(0,0,1)$	0.24	0.02	0.03
	$(0,1,0)$	0.10	0.25	0.34
	$(0,1,1)$	0.47	0.18	0.24
	$(1,0,0)$	0.14	0.21	0.42
	$(1,0,1)$	0.44	0.16	0.04
	$(1,1,0)$	0.06	0.11	0.26
	$(1,1,1)$	0.10	0.19	0.33

Table 5.3. Percentage circuit failure results for the function hazard and hold-state failure analysis of the logic-hazard free design version of circuit 0x8E. **D/D**: default model with default parameters; **C/D**: Cello model with default parameters; **C/C**: Cello model with characterized parameters.

		Logic-Hazard Free Design		
Circuit Failure	**Input**	D/D	C/D	C/C
0-Function Hazards	$(0,1,0) \rightarrow (1,1,1)$	0.09	0.14	0.21
	$(0,1,0) \rightarrow (1,0,0)$	0.08	0.18	0.40
	$(1,1,1) \rightarrow (1,0,0)$	0.31	0.35	0.78
	$(1,1,1) \rightarrow (0,1,0)$	0.37	0.53	0.41
	$(1,0,0) \rightarrow (0,1,0)$	0.29	0.53	0.28
	$(1,0,0) \rightarrow (1,1,1)$	0.05	0.15	0.30
1-Function Hazards	$(0,1,1) \rightarrow (1,0,1)$	0.88	0.46	0.08
	$(0,0,0) \rightarrow (0,1,1)$	0.43	0.17	0.24
	$(0,0,0) \rightarrow (1,0,1)$	0.90	0.43	0.14
	$(1,0,1) \rightarrow (0,1,1)$	0.86	0.44	0.29
	$(0,1,1) \rightarrow (0,0,0)$	0.54	0.27	0.37
	$(1,0,1) \rightarrow (0,0,0)$	0.60	0.26	0.34
Wrong Steady State	$(0,0,0)$	0.36	0.17	0.29
	$(0,0,1)$	0.13	0.03	0.00
	$(0,1,0)$	0.07	0.21	0.17
	$(0,1,1)$	0.34	0.15	0.14
	$(1,0,0)$	0.04	0.16	0.40
	$(1,0,1)$	0.36	0.13	0.02
	$(1,1,0)$	0.01	0.10	0.15
	$(1,1,1)$	0.04	0.13	0.21

Table 5.4. Comparison of all model predictions for of genetic circuit failures, and the most "robust" circuit choice for each model simulation. **D**: default parameter values; **C**: characterized parameter values; **O**: original design; **T**: two-inverter design; **N**: no-logic-hazard design (or logic-hazard free design).

		Default Model			**Cello Model**					
		D			D			C		
Circuit Failures		O	T	N	O	T	N	O	T	N
0-Function Hazards	$(0,1,0) \rightarrow (1,1,1)$									
	$(0,1,0) \rightarrow (1,0,0)$									
	$(1,1,1) \rightarrow (1,0,0)$									
	$(1,1,1) \rightarrow (0,1,0)$									
	$(1,0,0) \rightarrow (0,1,0)$									
	$(1,0,0) \rightarrow (1,1,1)$									
1-Function Hazards	$(0,1,1) \rightarrow (1,0,1)$									
	$(0,0,0) \rightarrow (0,1,1)$									
	$(0,0,0) \rightarrow (1,0,1)$									
	$(1,0,1) \rightarrow (0,1,1)$									
	$(0,1,1) \rightarrow (0,0,0)$									
	$(1,0,1) \rightarrow (0,0,0)$									
Wrong Steady-State	$(0,0,0)$									
	$(0,0,1)$									
	$(0,1,0)$									
	$(0,1,1)$									
	$(1,0,0)$									
	$(1,0,1)$									
	$(1,1,0)$									
	$(1,1,1)$									

This table compares each models' predictions for best and worst performance for each transition. A red color means that the circuit performed worse than the median, and green color means that the circuit performed better than the median, for that input molecule concentration or transition.

CHAPTER 6

DESIGNING AND REDESIGNING GENETIC CIRCUITS TO AVOID FAILURE

This chapter uses all the tools and methods developed in this thesis work and described in previous chapters to help in the design process of different *Genetic Regulatory Networks* (GRNs), in collaboration with other laboratories dedicated to synthetic biology research. The methods developed in this dissertation can help both in the design, as well as in the redesign processes of GRNs to produce dynamical model predictions, genetic circuit failure analyses, and gate characterizations for more accurate robustnesses predictions. These collaborations entailed laboratory experiments coupled with our modeling, noise analysis, and failure predictions to produce information that can be used to create more robust designs [36, 175].

6.1 DBTS Loop

An implicit, iterative *Design-Build-Test* (DBT) process is often used to develop genetic circuits [18, 68, 154]. However, bias is introduced into the DBT process in almost all of its steps and the variability of environmental factors that affect a circuits' behavior is often not taken into account. This might hinder a circuit's expected performance when applied to *Outside-the-Laboratory Conditions* (OTLC). Models used by *Genetic Design Automation* (GDA) tools are mostly based on experiments carried out under *Optimal Laboratory Conditions* (OLC) [2, 75, 129]. Furthermore, most rely only on the expression of a fluorescent protein as an output reporter under OLC. This setup leads to an inaccurate *Scale* step with regard to the actual circuits' performance when applied in non-OLC that can produce erroneous or faulty behavior with unpredictable outcomes. Furthermore, with a narrow *Test* step, the *learning* usually is limited to a post-hoc description of circuit dynamics. This would be especially perilous for engineered systems that are aimed to operate in dynamic environments, such as living therapeutics and whole cell biosensors.

This study applies a broader *Test* step to a designed delay-signal circuit to include more environmental dynamic factors and reporters (as shown in **Figure 1.1**). The circuit's output, as well as the time for output detection, were observed to be highly variable for different temperatures, mediums, inducer concentration, bacterial growth-phases, and output reporters. If the performance of the delay circuit is compromised by the tested experimental factors presented here, it will inevitably alter its behavior in other contexts, which would not have been predicted by GDA tools.

We propose to introduce a *Scale* step as part of a new and improved *Design-Build-Test-Scale* (DBTS) process. Scaling refers to the process of considering the variability of factors that can affect genetic circuit performance in real-life applications. Most studies either have a non-existent *Scale* step, or it consists only of a post-hoc description of the designed circuit's performance at OLC. This work not only provides a re-parametrization effort for different experimental conditions, but also produces a new model to determine the necessary predictions for untested conditions. As a case study, we focused on the effect of growth phase on the circuit's output, in which we observed a trend in delay and total output production. This, in turn, allowed for a deeper *Scale* step which ultimately resulted in a new model that estimates these trends, thus enabling the capacity to predict untested delays and output production of the circuit which can be further applied for scaling.

Thus, we propose that a greater emphasis in the *Test* and *Scale* steps of a DBTS cycle are needed to build more predictive models and to reduce bias across the entire DBTS cycle. This, in turn, will enable the possibility of finding design alternatives to any unexpected behavior and performance when the circuits are used in applications, improving a genetic circuit's robustness [170]. As we move from proof-of-concept designs to more real-life applications, a thorough *Test* step provides the necessary data that allows for a significant *Learn* step and, therefore, an appropriate *Scale* step.

6.1.1 Design of a Test Case GRN: The Delay Circuit

A naive implementation of delay in a genetic circuit is shown in **Figure 6.1** (a), where successive pairs of NOT gates can be used to add delay to a circuit (from a change of inputs to a change in outputs) without changing the circuits' function or behavior. However, this circuit can produce unwanted output production (*set-up* glitches) when it is initialized,

even in the absence of input molecules, since its components have not been stabilized yet [55]. This means that when the circuit is initialized, since the circuit is not stabilized, some internal gates will start randomly producing output before others. This can cause an erroneous or faulty initial state for the circuit, and therefore an unexpected or unwanted output protein production. So, for example, the gate that produces the output protein of a circuit (blue gate, **Figure 6.1** (a)) could start producing output before the circuit reaches steady-state without any inducer (input) present, at which point it will be repressed by the green gate.

However, it is possible to redesign the circuit in a way to avoid these initialization problems and properly locking the initial-state down, so that there is no unwanted switching behavior, or set-up glitches, when this circuit is initialized. **Figure 6.1** (b) shows such a design that would avoid set-up failures due to the initialization problem. When such a circuit is transformed into a bacteria, and there is random production from internal gates, since the circuit is not in steady-state, there will be no unwanted output production. This is because the output-producing gate is an AND gate, which needs the presence of two signals before it can produce the output signal. The second inducer is necessary so that even if there is some initial leakage production of the green gate, the output is not going to be produced.

A schematic of the actual circuit designed, built, and tested in this work is shown in **Figure 6.2** (a). This implementation uses Cello parts [129], and the specific transcriptional units were chosen to have good input/output correlated value matches. The intended purpose of this design is to provide some delay between an input concentration change and output production, whilst avoiding unwanted switching behavior due to initial propagation of an erroneous state. The circuit will not produce *Yellow Fluorescent Protein* (YFP), unless both *Arabinose* (Ara) and *oxohexanoyl-homoserine lactone* (HSL) are present. This design avoids unwanted production of an output YFP when the circuit is initialized in a cell without Ara: even if there is initial production of LuxR protein. Meaning, if there is no HSL, the circuit will not produce YFP and both Ara and HSL are needed to produce the circuit's output.

The initial model predictions of the circuit, shown in **Figure 6.2** (b), were done in iBioSim [169] using an automatic model generator to produce an *Ordinary Differential*

Equation (ODE) model of the circuit. The resulting complete model was then analyzed using the Runge-Kutta-Fehlberg method [53], also implemented in iBioSim [169].

The simulation results show that there is no YFP production when only HSL is present and, furthermore, there is a delay in the YFP production when Ara is added as expected. However, given that these simulations are using default parameters that were characterized under OLC (obtained from [129]), these only provide qualitative information on how the actual circuit is going to behave only when tested in OTLC.

6.1.2 Parametrization of Gates

After the initial design, model, and simulation of the circuit, we proceeded to build the circuit as described in [175]. The circuit was tested under different laboratory conditions, which produced different delays for the same circuit layout. However, from these results, introducing the input molecules at different times (meaning for different growth phases of the mono-cultures), produced the most substantial changes. Therefore, we proceeded to repeat those experiments for each individual gate of the circuit, in the hopes that the subsequent characterization and remodeling of the whole circuit with the new parameter values can replicate what we observed experimentally. The following subsection provides such enterprise to determine the steady-state parameters (Subsection 6.1.2.1), and the dynamic parameters (Subsection 6.1.2.2) of the genetic parts.

6.1.2.1 Hill Function Parameters

Equation 3.1 is derived from the Hill function, which is used to calculate the steady-state output of a gate [129, 152]. To acquire the Hill function parameter values used in this equation, gate induction experiments were performed (as described in [175]), and then fitting the fluorescence results to our model. This fitting can be done by minimizing the error of Equation 6.1 (for *activation*) and 6.2 (for *repression*) predictions by using parameter fit values and the experimental fluorescence measurements for the different growth phases.

$$y = y_{min} + \left(\frac{y_{max} - y_{min}) \cdot (K^n)}{(x^n) + (K^n)} \right) , \tag{6.1}$$

$$y = y_{min} + \left(\frac{y_{max} - y_{min}) \cdot (x^n)}{(x^n) + (K^n)} \right) . \tag{6.2}$$

For the Hill-function parametrization, a normalized least-squares method using the non-linear *Least-Squares Minimization and Curve-Fitting* (lmfit) Python package [126] was used,

and random initial parameter estimations following the GAMES workflow [43].

The Hill-function fitted parameter values obtained are shown in **Table 6.1**. These parameters can be used to calculate the *steady-state* output of the circuit using Equations 6.3 and 6.4. However, to be able to predict the *dynamical* behavior of these circuits, the τ_{ON} and τ_{OFF} parameters also need to be calculated.

6.1.2.2 Tau (τ) Parameters

Equation 3.3 describes the dynamical response of each gate, using the τ_{ON} and τ_{OFF} parameters. To obtain these dynamical parameter values, different *ON-to-OFF* and *OFF-to-ON* characterization experiments were performed using the same gate plasmids and methods as shown in Shin et al. [152] and in [175]. These experiments can be used to fit the data using Equations 6.3 and 6.4, obtained from the supplementary material in Shin et al. [152], to achieve parameter values for these conditions.

$$\frac{dx}{dt} = \tau_x^{ON} \cdot (x_{ss} - x) \ , \tag{6.3}$$

$$\frac{d[YFP]}{dt} = \tau_{YFP}^{ON} \cdot x - \tau_{YFP}^{OFF} \cdot [YFP] \ . \tag{6.4}$$

The lmfit Python package [126], which is based on the Levenberg-Marquardt minimization algorithm, was used to perform the fits and analyze the resulting parameter sets [126]. The fits were performed by minimizing the sum of the square of the relative error between each measured data point and the same point in a corresponding model simulation. As with the Hill-function characterization algorithm, a random initial parameter value search was implemented following the GAMES workflow [43], while simultaneously looking for the smallest chi-squared values for each fitting iteration.

Using the estimated values of τ_{YFP}^{OFF}, shown in **Table 6.2**, and using both Equations 6.3 and 6.4, the first fitting iteration was used to obtain τ_{YFP}^{ON}, τ_x^{ON}, and x_{ss} parameter estimate values using the *ON-to-OFF* characterization experiment results. Using the parameter estimation method proposed in [43], and the fixed values of τ_{YFP}^{OFF} obtained previously, the model was fitted to the experimental results using a minimizing function. The parameter values estimated with this method are shown in **Table 6.2**.

Using the *ON-to-OFF* characterization experiments, and assuming that the influence of input sensor promoter flux is zero, then fitting Equation 6.4 to the gradient of the

fluorescence loss over time produces estimates of τ_{YFP}^{OFF} parameter values.

To capture the effect of different growth phases on the production delays, as well as signal intensities of the designed circuit in this work, characterization experiments were done for the different growth phases. These experiments were performed following the methods described in Shin et al. [152] (see Methods). From these experiments, new parameter values for each growth phase were achieved as described below.

Using the parameters shown in **Table 6.2**, derived from the fitting algorithm, new simulations were produced for each growth phase (**Figure 6.3**). The new model simulations predict both lower production of YFP protein (signal intensity), as well as the decrease in time for reaching steady-state, for each successive growth phase, as observed experimentally.

However, the parametrization results (shown in **Table 6.2**) were produced without fixing any of the parameter values when using the fitting algorithm, meaning they were all treated as free variables. When there are no fixed parameters, then the fitting algorithm will find the best fit by manipulating the parameter values until the expected outcome best matches the experimental results. This will result in widely different parameter values to compensate for other parameters' minimization of error while fitting (see for example, x_{SS} values for different growth phases in **Table 6.2**). This means it will be hard to derive any parameter value trends indicative of what might be happening to their magnitude as the circuit is induced at different growth phases.

To discern if there are any parameter value trends that can be attained from fitting the experimental results, we proceeded to fix parameter values to reduce the number of free variables in the fitting algorithm. The first parameter value fixed was τ_y^{OFF}. This was done using the fitting algorithm with the ON-to-OFF gate characterization experiments (see *methods* section). Re-fitting the model to the experimental results, with τ_y^{OFF} as a fixed parameter value and the rest as free-variables, new parameter values were acquired. These re-fitting iterations were done then by fixing subsequent parameters values by calculating averages from previous iterations. First, x_{SS} average values were calculated, then fixed for the next iteration, then followed by τ_x^{ON}. With the last iteration of this re-fitting process, τ_y^{ON} was left as a free-variable, while the rest of the parameters were fixed. This process was carried out to understand the effect of the different experimental conditions on the

value of τ_y^{ON}. First, since τ_y^{ON} is the parameter closest to the measured parameters in the experiments (YFP fluorescence); and secondly, because if the other parameter values are not fixed, then if there is any parameter value trends, it is lost in the minimization process, when the fitting algorithm tries to find the solutions by increasing a parameter value and decreasing another one. **Figure 6.4** (a) shows τ_y^{ON} parameter values obtained following this procedure.

As hypothesized, the values of τ_y^{ON} shown in **Figure 6.4** (a) vary for the different growth phases of the clonal bacteria. When bacteria are induced at a later growth-phase, the values of τ_y^{ON} decrease. This coincides with previously observed experimental results where induction at later growth phases decreases the time it took to reach steady-state, as well as the maximum signal intensity at the steady-state. A predictive linear model of signal intensity over time can be created as shown in Equation 6.5:

$$\tau_y^{ON} = m * (induction\,time) + b\ , \tag{6.5}$$

where τ_y^{ON} is the predicted value of gate dynamics when the circuit evolves to an *ON* state [152]. Using linear regression, m was estimated to be $-1.869e^{-04}$ and b to be 0.13527606. With this equation, a researcher could estimate the value of τ_y^{ON} for different times of induction and, therefore, estimate the decrease in output production and delay of a delay circuit for untested conditions as shown in **Figure 6.4** (b).

6.1.3 DBTS Workflow

The DBTS cycle can be a powerful methodology to successfully implement GRN applications in the synthetic biology field. Yet, it is not flawless and is constantly needing to be re-examined to reduce the turn-around for synthetic biology applications. As an example, when engineering bacteria to act as a sensor for a specific molecule, the two features of time for fluorescence detection and signal intensity are very important. This work shows that if bacteria was sensing a molecule at the *Early-Exponential* (EE) stage, and was observed prior to its ability to produce a fluorescence signal, then it can lead to a false negative result. In addition, bacteria that sensed the molecule at the *Stationary* (S) phase, and therefore produced a significantly low fluorescence signal, can also lead to a false negative result. Furthermore, our results support the notion that when designing a genetic circuit, the range of inducer concentrations that can lead to a satisfying performance can vary across

different cultivation temperatures which is a major factor when transitioning outside the lab. Additionally, in this work we hypothesized the values of τ_y^{ON} shown in **Figure 6.4** (a) vary for the different growth phases of the clonal bacteria. After learning from the appropriate test results, we could acquire new models in the *Scale* step that could predict the behavior of these circuits on untested experimental conditions. Thus, by introducing the *Scale* step and broadening the *Test* step to include more environmental factors will subsequently advance the overall learning and will provide deeper understanding of the obstacles that genetic circuit's performance face when taken outside of the lab.

Expansion of the *Test* step will inevitably promote new prediction models and tools developed in the *Scale* step, able to predict the behavior of genetic circuits under different conditions (even untested ones), and therefore will enable better design choices than those provided by GDA tools. Furthermore, these studies can identify, which experimental conditions have a greater effect on a genetic circuit's performance. Currently, there is a lack of standardized methodologies and/or software tools to help researchers perform a meaningful *Scale* step and benefit from its results. Even popular GDA tools (like Cello [129]), which provide extensive and automated *Design*, *Build*, and even *Test* steps, lack of a proper *Scale* step, which will be beneficial to help researchers with better design choices for OTLC genetic circuit applications. However, as of now, there is no consensus on what these methodologies should look like, nor tools to help with this process.

6.2 DSGRN

The *Dynamic Signatures Generated by Regulatory Networks* (DSGRN) [33] is a computational tool that explores network dynamics using a parameter graph space for a given GRN and input interactions. This tool computes the range of dynamic behaviors supported by a given network [32]. In collaboration with Rutgers and Montana State University, we were tasked on determining the noise model GRN failure predictions, re-characterization of genetic gates, and modeling simulations of the different circuit layouts calculated using DSGRN-design GRNs.

In Bree et al. [36], a design-build-test-learn loop called *Design Assemble Round Trip* (DART) is presented for the rational design of synthetic biology genetic logic circuits. In principle, the technology is generalizable to dynamically-complex circuit functions

beyond logic [34] that are of interest to the synthetic biology community [91, 171, 172]. The design/prediction tools are DSGRN [34, 57][1] and *Combinatorial Design Model* (CDM) [50][2]. CDM is a neural-network based model that makes in-silico predictions of experiments by using context and data from a subset of conditions and predicts the outcome in all combinations of conditions. The application of CDM in this work optimized a combination of genetic parts for a given circuit using training data from single parts.

Circuit robustness was examined by comparing the performance of simple and complex network topologies of OR and NOR synthetic circuits in the yeast *Saccharomyces cerevisiae*, while also testing the predictions for two different sets of parts for each topology using CDM. The performance of a circuit is evaluated as a circuit's ability to express fluorescence (ON) or not (OFF) as intended by the circuit's logic given the presence or absence of chemical inputs. Logic circuits designed to exhibit OR and NOR logic were chosen based on preliminary data analysis in which previously built OR and NOR circuits performed poorly [35, 52]. **Figure 6.5** shows the schematics of the designs discussed in this work, where each quadrant shows one topology with two CDM designs. The top row shows the simplest topologies that are capable of producing the desired circuit behavior. The alternative topologies discovered using the DSGRN Design Interface are called DSGRN topologies and shown in the second row.

The parts labeled with r# are associated to constitutively expressed CRISPR guide RNA (gRNA) gene products introduced in [59] that repress transcription when bound. Inducible versions of these parts were built in this study to use in tandem with the previously built constitutively expressed parts. Specifically, binding sites for *β-estradiol* (BE) and *doxycycline hyclate* (Dox) were added to the gRNA promoters. In the presence of an inducer, the associated gRNA is expressed and represses the production of its downstream target, either another gRNA or *Green Fluorescent Protein* (GFP) [36].

6.2.1 Noise Models for DSGRN

For analysis of the circuits, models for each circuit design were generated using iBioSim's [169] graphical editor, which uses the *Systems Biology Markup Language* (SBML) [26]

[1] https://github.com/marciogameiro/DSGRN

[2] https://github.com/SD2E/CDM

to encode the different species and reactions. The parameters used by the models were obtained from literature, and all reactions are automatically populated with them. All input transitions, for each circuit, were also modeled using Petri-nets (a collection of directed arcs connecting places and transitions) encoded in SBML models to simulate changes in input concentration changes. These models were then simulated using Runge–Kutta–Fehlberg (4, 5) method (rkf45) for the ODE models, and Gillespie's *Stochastic Simulation Algorithm* (SSA) method for the stochastic (Monte Carlo) models. All simulations were run for 3000 time points, with input changes occurring at time point 1500 (for transitions that contain function hazards). Then, the circuit's predicted output was measured at different time points for different circuit failures analyzed: at 1600 time points for transitions that contain function hazards, at 250 time points for set-up glitches, and at 2000 time points for hold-state failures. Simulations were terminated if a glitch was observed (only static glitches were analyzed). A constraint value of eleven (11) was considered for all circuit failures. This means, if the output of a circuit is expected to be low (or 0) throughout the simulation, it is considered a glitch if the output protein production of the circuit surpassed the constraint value. On the other hand, if the expected output production of the circuit is high (or around 60 molecules) throughout the simulation, then it is considered a glitch if the output production is less than the constraint value. Set-up glitches were analyzed only if the output of the system is expected to remain low, thus it is considered a glitch if production increases above the constraint value, as for static-0 hazards. For hold-state failures if the output was higher than the constraint value for expected low output states, or lower than the constraint value for expected high output states, then the circuit was considered to have failed the hold-state simulation and the simulation would be terminated. The simulation terminations are counted for the percent failures shown in **Tables 6.3**, **6.4**, and **6.5**.

The extrinsic noise model used in this work applies a simple case of *static* external perturbations, modeled as a random draw from a folded *normal* distribution for each parameter value used in the model at the beginning of each simulation run (as shown in Chapter 5). The mean of each distribution is the default parameter value in iBioSim (obtained from literature), with a standard deviation equal to forty percent of the mean's absolute value (which emulates the "extrinsic noise"). This value of noise was obtained

when calibrating different noise values but is an arbitrary value that should be replaced with a better estimate obtained from experimentation. For example, Beal [6] argues that these parameters follow a geometric distribution instead of a normal one, which is also part of our planned future work. To simulate the extrinsic noise models, eight hundred (800) ODE simulation runs were simulated for each transition, for each circuit using the Runge–Kutta–Fehlberg (4, 5) method (rkf45). Extrinsic noise model ODE simulation runs take a long time to simulate, thus 800 simulation runs took around 4.5 hours to finish. More simulation runs had minor, and non-significant, changes on the simulation results, however adding hours to the total simulation time. Therefore, 800 simulation runs are generated for each input transition of all the circuits analyzed using the extrinsic noise model.

For each input transition of all the circuits analyzed using the intrinsic noise model, a thousand (1000) stochastic simulation runs were simulated using a Monte Carlo approach, such as Gillespie's SSA. More simulation runs for the intrinsic noise models didn't change the results obtained, thus this number of simulation runs per transition was chosen.

6.2.2 Hazard Analysis for Circuit Failure Predictions

Table 6.3 shows the intrinsic noise model circuit failure predictions for the different circuit implementations. The table shows that:

1. The redesigned OR circuit fairs worse (has higher predicted circuit failure percentages) than the original design for all circuit failures analyzed except for the set-up glitches and hold-state failures at state (0,0).
2. A comparison of the NOR circuit original design and redesign shows that the redesign has less circuit failure percentages than the original design, except for set-up glitches at state (1,1) and hold-state failures at state (0,0).

These results show that the intrinsic noise model predicts that for the OR circuit layout, the original design is less prone to glitches than the redesign, for the transitions and states analyzed in this work. Conversely, for the NOR circuit layouts, the intrinsic noise model predicts that the redesign is less prone to show circuit failures for the transitions and states analyzed. Therefore, if the main source of noise for these circuits is intrinsic noise, then these models predict that the original OR design and the NOR redesign are the better

design choices.

Table 6.4 shows the extrinsic noise model circuit failure predictions for the different circuit implementations. The table shows that:

1. For the OR circuit layouts, the redesigned circuit has equal or higher predicted circuit failure percentages than the original design for all transitions and states analyzed, except for set-up glitches and hold-state failures at state (0,0).
2. For the NOR circuit layouts, the redesign also has equal or higher circuit failure percentages than the original design for all transitions and states analyzed.

Therefore, if the noise source with the highest incidence on the circuits' output is extrinsic noise sources, these models predict that the redesigned circuits for both the OR and NOR circuit layouts are more prone to express circuit failures than the original designs.

Table 6.5 shows both the combined intrinsic and extrinsic noise model circuit failure predictions for the different circuit implementations. The table shows that:

1. For the OR circuit layouts, the redesigned circuit has equal or higher circuit failure percentages than the original design for all transitions and states analyzed, except for set-up glitches and hold-state failures at state (0,0).
2. When looking at both model predictions for the NOR circuits, the redesigned circuit fairs better than the original design only for set-up glitches (0,1) and (1,0).

The combined intrinsic and extrinsic noise model results indicate that the original design is a better design choice for both the OR and NOR circuit layouts. However, these results reflect mostly what the extrinsic noise model predictions suggested (see **Table 6.4**), indicating that the combined noise model has a higher influence from the extrinsic noise model than the intrinsic noise model. A possible explanation to these results can be that the simpler the circuits' layout, the less noise propagation between the different layers, which would lead to a smaller variation in a circuits' output. However, depending on the intended use purposes of the designs, the DSGRN-designed circuit may be more suitable for detecting signals and producing substantially different *ON* and *OFF* output distributions.

These results emphasize the necessity to study the relative influence of either noise source on these kinds of genetic circuits, to accurately determine which circuit layout is less prone to glitch. Since intrinsic and extrinsic noise model predictions differ when de-

termining which circuit has lower circuit failure percentages, then studying which one has a higher influence on a circuit's output (if not both) would be critical for a researcher that is interested in determining which circuit layout is a better design choice for the intended application of the circuit. This, ultimately, is dependent on the circuits components' output production rate. At higher rates of output production for each component, the stochastic intrinsic noise model predicts less variation, thus approximating ODE simulation results (and decreasing intrinsic noise as a source of variation). Therefore, an investigation of the different genetic parts' output production numbers, and relative influence of intrinsic and extrinsic noise sources of variation on the circuit's output is extremely relevant to obtain more accurate circuit failure predictions, and ultimately have designers make choices with more information.

6.2.3 Parameterization of DSGRN Gates

Experimental results have shown that the simple NOR design is an anomalously poor performer compared to all other designs [35, 52]. This is interesting because all the other builds contain similar NOR gates, although they use different gRNA parts (see **Figure 6.5**). Either the simple NOR builds are unexpectedly fragile, or there is perhaps synergistic activity when multiple NOR gates act in concert.

To further explore this performance failure, a Hill function ordinary differential equation model of the designs was created using parameter fits from the same dosage response experiments used to train CDM. A data-fitting algorithm using a Nelder–Mead minimization method [124] was implemented to determine the Hill function [145] parameter values for the induction and repression dynamics of the different genetic parts used in the OR/NOR circuit designs. The experimental data used for the fitting algorithm was obtained from the geometric mean of flow cytometry data distributions from the dosage response experiments as described in [36]. The experimental data were fitted to Equation 6.6 (for activations), and Equation 6.7 (for repressions) derived from Cello [129]:

$$y_{i+1_{SS}} = y_{i_{min}} + (y_{i_{max}} - y_{i_{min}}) \frac{1}{\left(\frac{\kappa_i^{n_i}}{y_{i-1_{SS}}}\right)^{n_i} + 1} \tag{6.6}$$

$$y_{i_{SS}} = y_{i_{min}} + (y_{i_{max}} - y_{i_{min}}) \frac{1}{\left(\frac{y_{i-1_{SS}}}{\kappa_i^{n_i}}\right)^{n_i} + 1}, \tag{6.7}$$

where $y_{i_{SS}}$ is the steady-state output promoter activity of part i; $y_{i_{min}}$ and $y_{i_{max}}$ are the minimal and maximal output promoter activities (obtained from experimental results), respectively, for part i; κ_i and n_i are the affinity and cooperativity of transcription factor binding (obtained with the fitting algorithm); and, finally, $y_{i-1_{SS}}$ is the steady-state input promoter activity from the previous part's output (calculated also using Equations 6.6 or 6.7). Using the Hill function parameter value estimations, a resulting ODE model is then analyzed using the Runge-Kutta-Fehlberg (4,5) method [53] implemented in iBioSim [169] to obtain steady-state output predictions for each design under different input concentrations (shown in **Figure 6.6** and **Figure 6.7**).

In general, the Hill model predicted that circuits should respond more strongly to Dox (**Figure 6.8** (a)), but that the dosage response to BE was acceptable (**Figure 6.8** (b)) except for the two simple NOR designs, in which the presence of BE alone is predicted to fail to turn the circuit OFF (**Figure 6.8** (c)). **Figure 6.8** (a) shows the DSGRN OR/CDM high design and illustrates the difference in BE and Dox performance. A success is a high GFP signal in all five bars, where the left-most bar is BE alone at its highest titration concentration and the remaining four bars are combinations of BE and Dox, with Dox at various titration levels. High GFP signal is achieved even for the BE-alone condition, since the low GFP steady state corresponded to about 1500-2000 *arbitrary units* (a.u.), substantially lower than the left-most bar. However, the BE-induced GFP signal is markedly lower than that for the BE+Dox combinations. Dox in isolation produced GFP signal in comparable amounts to BE+Dox (**Figure 6.6**, **Figure 6.7**).

6.3 Concluding Remarks

This chapter shows how the models and methods developed in this dissertation can be used to aid in the design processes of GRNs, without omitting the limitations and future work needed to improve future contributions. Even though most of the results have a more *qualitative* than *quantitative* nature, they do highlight the need to improve the characterization methods to provide more accurate prediction results. The characterizations needed are not only for genetic parts' dynamics, but also relative influence of external and internal noise sources of variability. Furthermore, the tremendous manual work that is required to obtain meaningful parameter values for most of the models used in synthetic

biology requires sufficient expertise in the subject, barring many synthetic biologists from providing reliable genetic circuit failure predictions to their designs. As we move from proof-of-concept to application-oriented genetic circuit designs using a more streamlined DBTS process, the avoidance of genetic circuit failures to produce more robust and reliable genetic circuit behavior is of crucial importance. The work done in this dissertation is one of the first steps in that direction.

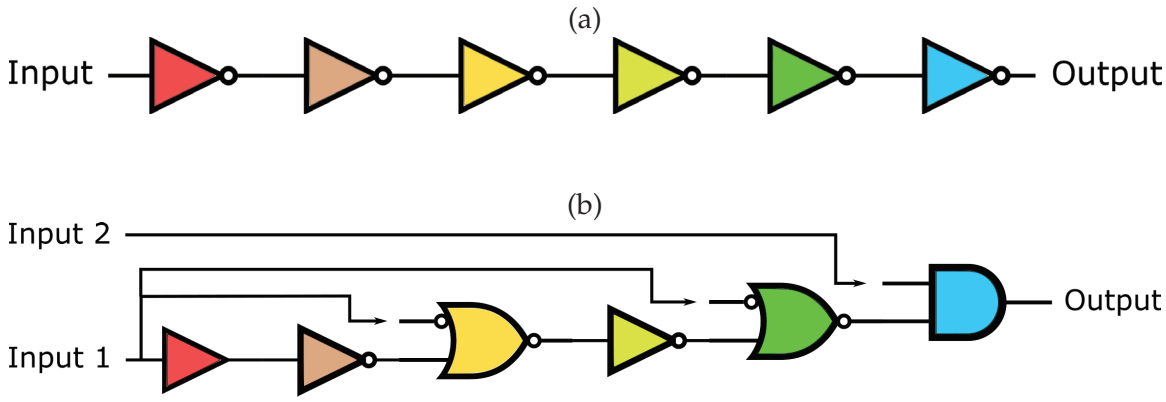
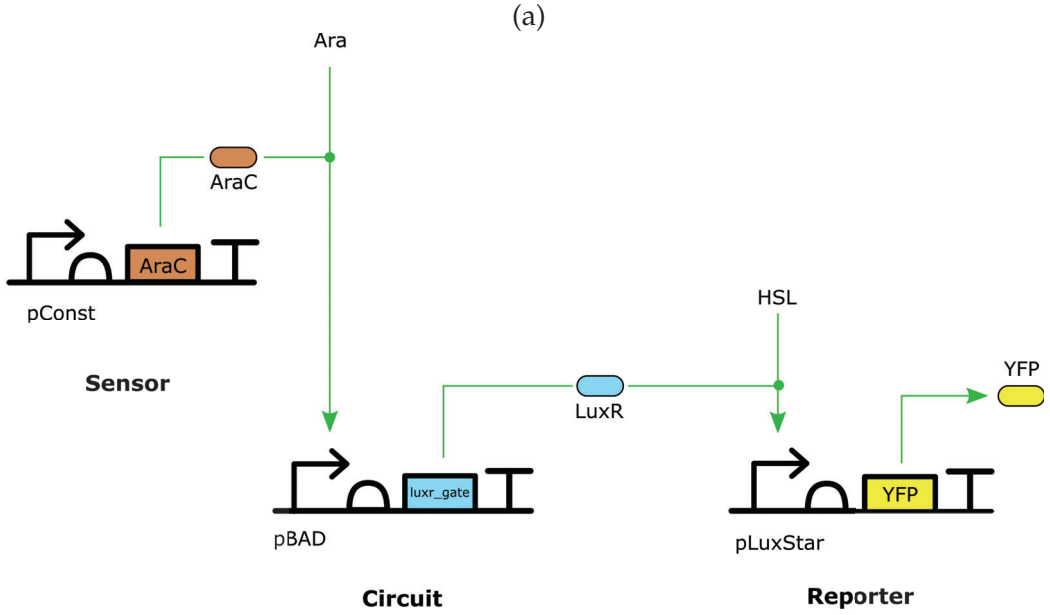


Figure 6.1. Naive and set-up failure-free delay circuit designs. (a) Simple delay circuit. Two successive NOT gates (represented as ⊣▷⊸) add delay to a circuit without changing the circuit's behavior. In this image, each logic gate is represented with a different color to represent different gate assignments. (b) Set-up failure-free delay circuit. In this figure ⊣▷⊸ represents a NOT gate, ⊣↛⊸ an NIMPLY gate, ⊣D⊢ and AND gate, and ⊣▷⊢ a *buffer* gate. Each logic gate is represented with a different color to represent different gate assignments.

(a)



(b)

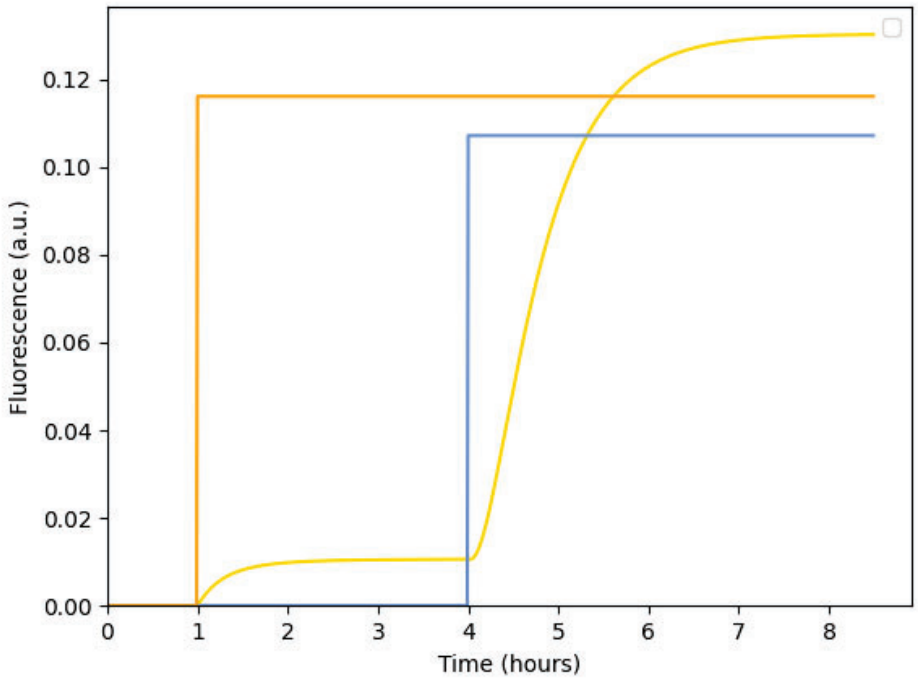


Figure 6.2. Design and simulation results of delay circuit designed. (a) Designed delay circuit using Cello gates [129]. Sequences obtained from SynBioHub [107]. This circuit produces YFP after a delay when both Ara and HSL are present. (b) Delay circuit simulation results using default parameters. YFP production (in a.u.) over increasing simulated time-points.

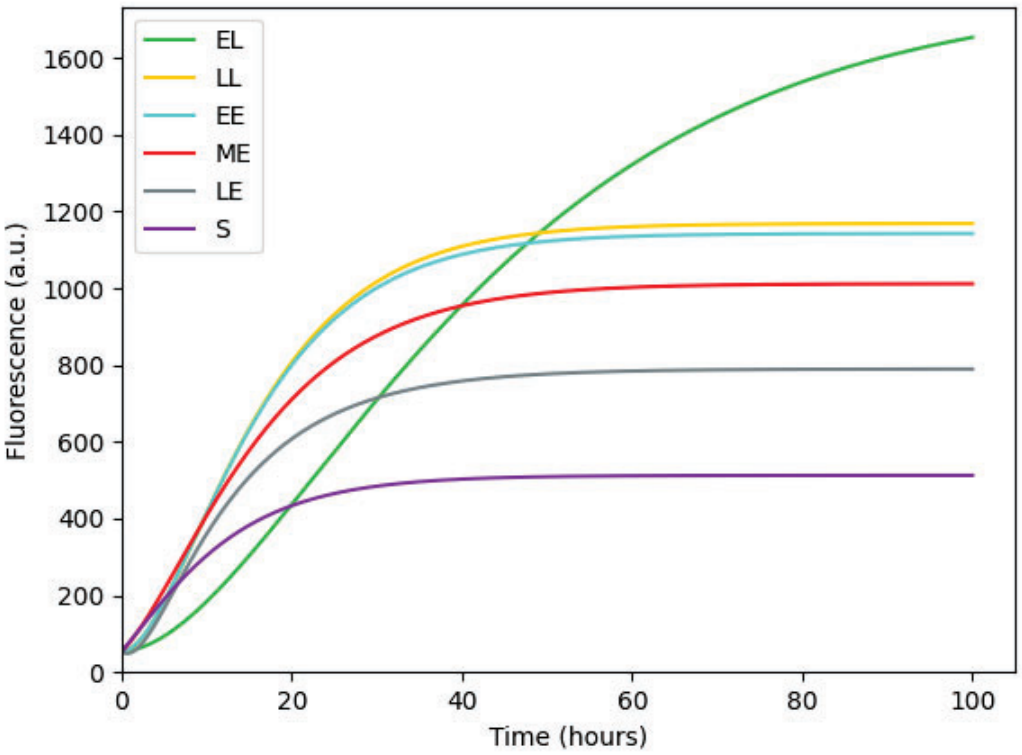
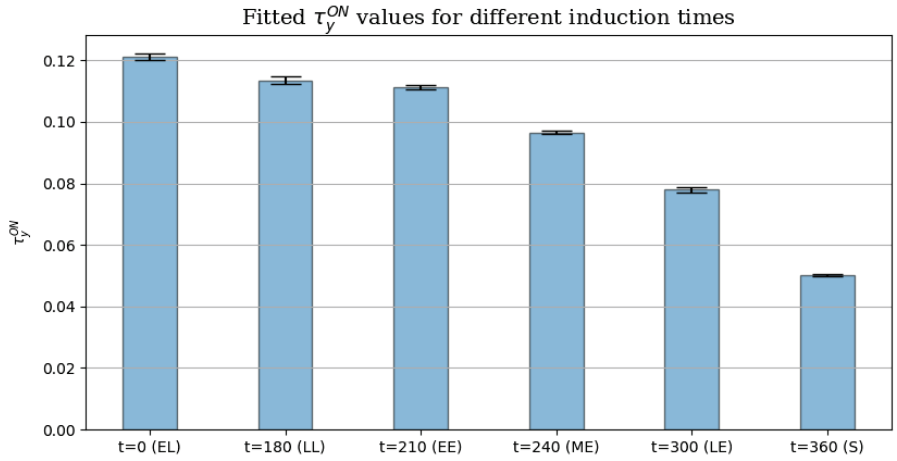


Figure 6.3. Simulation results for the different growth phases of the AraC gate, using fitted parameters shown in **Table 6.2**, obtained using the lmfit Python package [126]. The growth phases are *Early-Lag* (EL), *Late-Lag* (LL), EE, *Mid-Exponential* (ME), *Late-Exponential* (LE), and S.

(a)



(b)

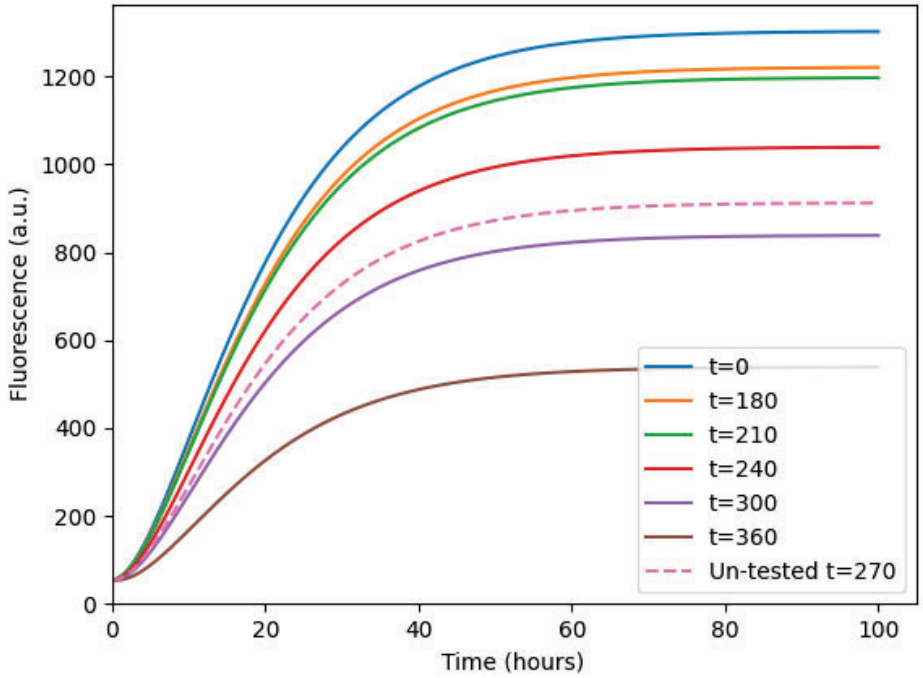
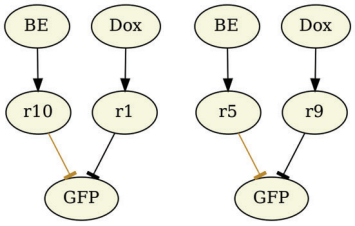
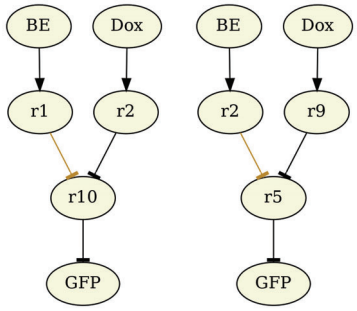
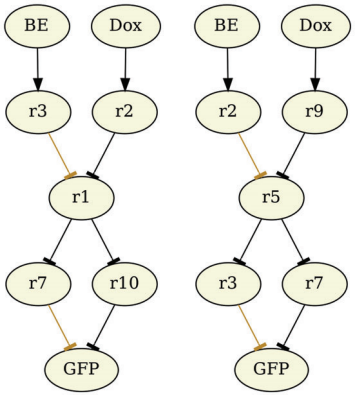
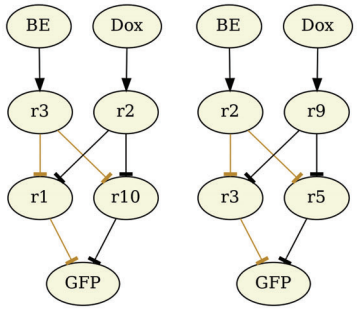


Figure 6.4. Fitted parameter values trend and prediction of un-tested YFP output production. (a) Fitted τ_y^{ON} values obtained when fitting fluorescence values for different induction times (t=0, t=180, t=210, etc.) using fitted parameters obtained using the lmfit Python package [126]. (b) Un-tested YFP output prediction for induction time of 270 minutes obtained using the linear regression model to estimate τ_y^{ON}'s value for the untested condition.

Figure 6.5. The designs for the built circuits, with one topology and two CDM designs in each quadrant. Upper left: The simple NOR topology published in [58], re-engineered with two different collections of gRNA inducible parts. The one on the left is predicted by CDM to perform worse than the one on the right. Lower left: The DSGRN NOR circuit predicted by DSGRN to perform more robustly than the simple NOR circuit, with the two CDM predicted gRNA arrangements. Upper right: The simple OR topology published in [58], with CDM-selected parts assignments. Lower right: The DSGRN OR designs predicted to perform more robustly than the simple OR designs. Scoring: The difference in CDM scores between the low and high designs is shown in each row above the corresponding topology. Topology robustness scores predicted by DSGRN for the NOR and OR circuit topologies are shown in each row below the corresponding topology. These numerical scores should be interpreted ordinally rather than as absolute values with specific interpretation.

	NOR circuit		OR circuit	
	CDM low design	CDM high design	CDM low design	CDM high design
CDM score difference	1.49		2.54	
Simple topologies				
Topology Robustness Score	0.08		0.07	

CDM score difference	2.66		2.65	
DSGRN topologies				
Topology Robustness Score	0.22		0.28	

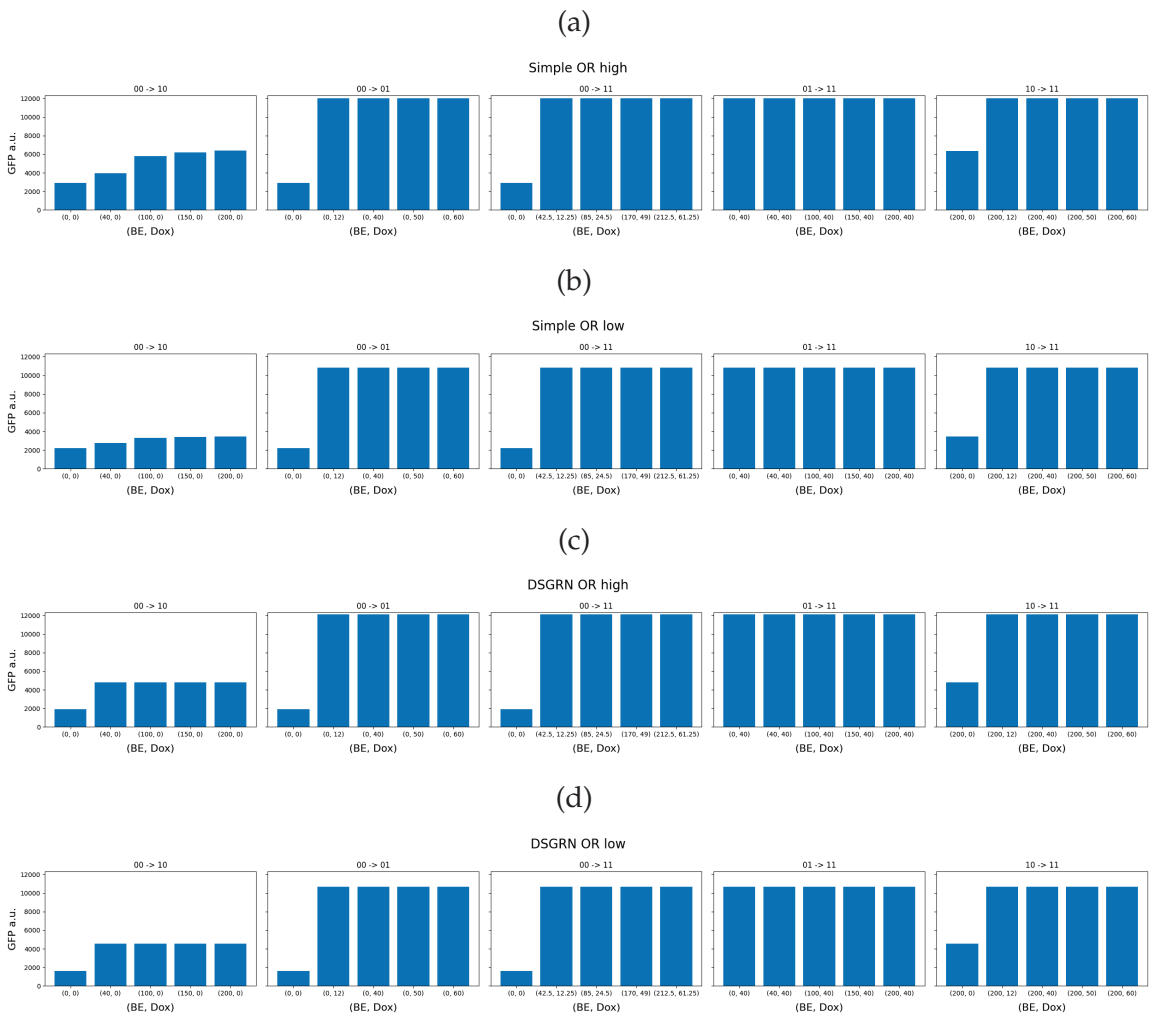


Figure 6.6. Predicted steady-state values of the geometric mean of the flow cytometry distribution of GFP a.u. using estimated Hill function parameter values for: (a) Simple OR/CDM high design, (b) Simple OR/CDM low design, (c) DSGRN OR/CDM high design, and (d) DSGRN OR/CDM low design. An OFF circuit state corresponds to approximately 1500-2000 a.u.; robust ON circuit states occur at about 8000 a.u. and up, and weak ON states occur at about 4000 a.u. Columns 1-3 (transitions 00 $\rightarrow$ 10, 00 $\rightarrow$ 01, 00 $\rightarrow$ 11): The left-most bar in each bar graph should be OFF. All other bars should be ON. Columns 4-5 (transitions 01 $\rightarrow$ 11, 10 $\rightarrow$ 11): All bars should be ON.

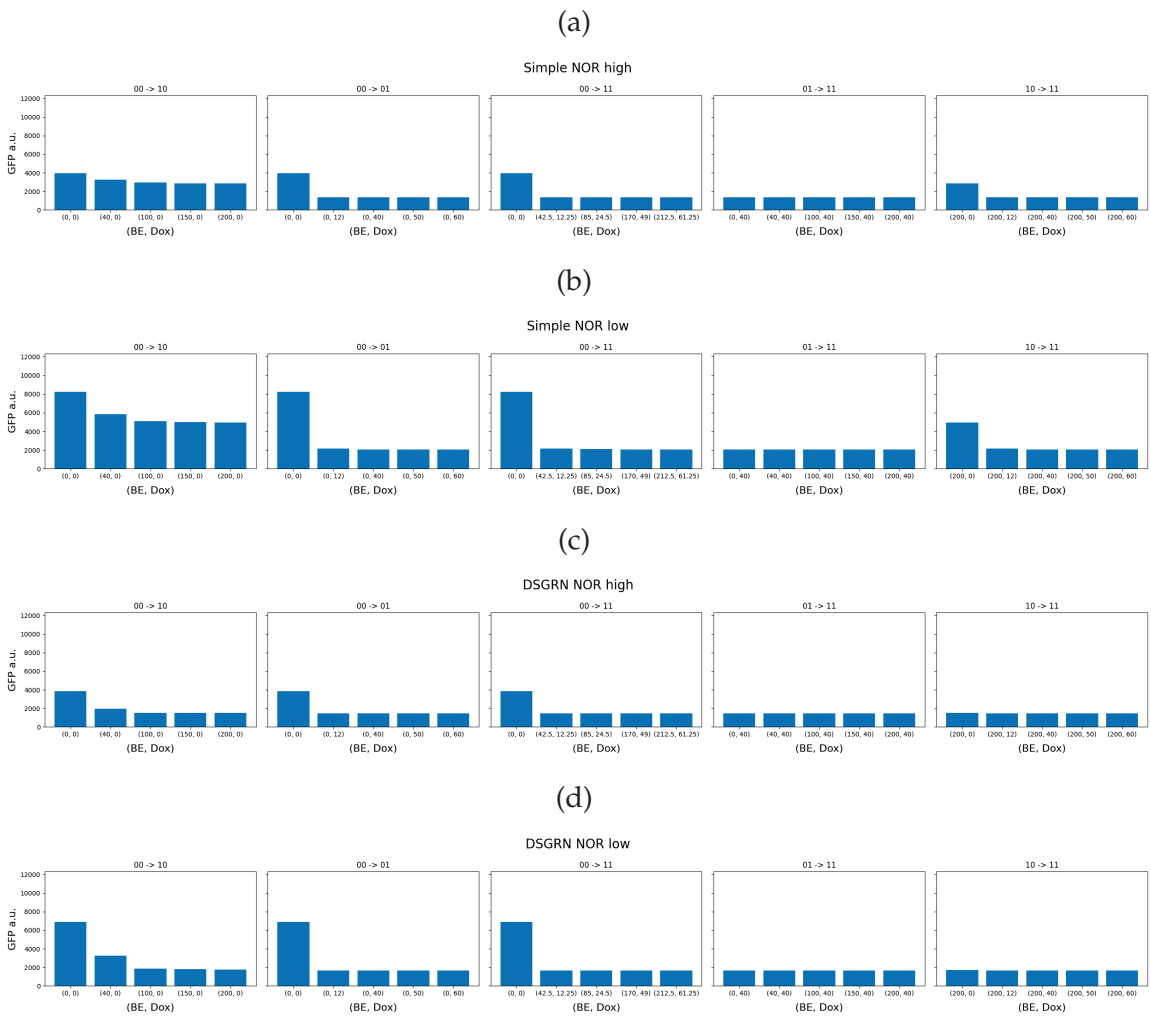
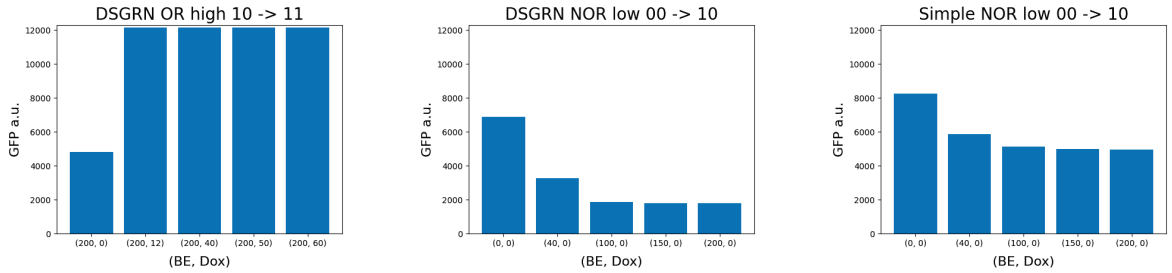


Figure 6.7. Predicted steady-state values of the geometric mean of the flow cytometry distribution of GFP a.u. using estimated Hill function parameter values for: (a) Simple NOR/CDM low design, (b) Simple NOR/CDM low design, (c) DSGRN NOR/CDM high design, and (d) DSGRN NOR/CDM low design. An OFF circuit state corresponds to approximately 1500-2000 a.u.; robust ON circuit states occur at about 8000 a.u. and up, and weak ON states occur at about 4000 a.u. Columns 1-3 (transitions 00 $\rightarrow$ 10, 00 $\rightarrow$ 01, 00 $\rightarrow$ 11): The left-most bar in each bar graph should be ON. All other bars should be OFF. Columns 4-5 (transitions 01 $\rightarrow$ 11, 10 $\rightarrow$ 11): All bars should be OFF.



(a) (b) (c)

Figure 6.8. Example predicted steady state values of the geometric mean of the flow cytometry distribution of GFP a.u. from Hill function models. (a) The DSGRN OR/CDM high design as it transitions from an initial state of BE only to BE+Dox for various titration levels of Dox. The logical circuit should show a high fluorescence signal at all conditions. The BE-only steady state has a strong GFP signal in comparison to the low GFP steady state (1500-2000 a.u.), but is significantly lower than inducer conditions where Dox is present. (b) The DSGRN NOR/CDM low design as it transitions from an initial state of no inducers to BE at various titration levels. The condition (0,0) should show high fluorescence and all subsequent bars should show low fluorescence. (c) The same as (b) for the simple NOR/CDM low design. It is seen that BE does not effectively repress GFP signal in panel (c).

Table 6.1. Hill-function parameter value estimations for different gates, obtained by fitting Equations 6.1 and 6.2 to part-characterization experiments at different growth-phases (EL: early-lag, LE: late-exponential). Values rounded to three significant digits.

	AraC			
Growth-phase	y_{max}	y_{min}	κ	n
EL	1730	342	4.16×10^{-2}	2.60
LE	1150	366	6.68×10^{-2}	2.89
	LuxR			
EL	4310	223	2.33	8.40×10^{-1}
LE	3230	308	5.52×10^{-1}	8.08×10^{-1}

Table 6.2. Dynamic parameter value estimations for different gates, obtained by fitting Equations 6.3 and 6.4 to *ON-to-OFF* and *OFF-to-ON* part-characterization experiments for different growth-phases (EL: early-lag, LL: late-lag, EE: early-exponential, ME: middle-exponential, LE: late-exponential, S: stationary). Values rounded to three significant digits.

	AraC			
Growth-phase	τ_x^{ON}	τ_{YFP}^{ON}	τ_{YFP}^{OFF}	x_{ss}
EL	0.0822	0.126	0.0893	976
LL	0.169	0.109	0.0933	1000
EE	0.117	0.13	0.114	1000
ME	0.111	0.112	0.111	999
LE	0.275	0.0749	0.096	998
S	0.208	0.0538	0.105	1000
	LuxR			
EL	0.143	0.335	0.166	995
LL	0.195	1.3	0.221	211
EE	0.11	0.274	0.11	938
ME	0.173	0.234	0.173	977
LE	0.0729	0.173	0.0726	1000

Table 6.3. Intrinsic noise model predictions of circuit failure percentages.

Circuit Failures		OR circuit		NOR circuit	
		Original Design	Redesign	Original Design	Redesign
Function Hazard	$(1,0) \rightarrow (0,1)$	12.1	23.1	7.4	2.8
	$(0,1) \rightarrow (1,0)$	10.4	24.9	4.1	2.4
Set-Up Glitches	$(0,0)$	34.8	11.1	-	-
	$(0,1)$	-	-	60.7	51.1
	$(1,0)$	-	-	60.7	51.0
	$(1,1)$	-	-	38.9	42.6
Hold-States	$(0,0)$	1.4	0.0	5.0	5.9
	$(0,1)$	44.3	51.9	52.4	7.4
	$(1,0)$	44.5	63.4	52.8	8.1
	$(1,1)$	10.9	18.5	8.0	0.2

Percent failure predictions for input transitions that contain a function hazard, set-up glitches, and hold-state failures for different circuit layouts, using the intrinsic noise model. In this table, input transitions and states are described as a tuple (x,x), where the first value is the concentration of the first inducer (1: *high* and 0: *low*), and the second value is the concentration of the second inducer (1: *high* and 0: *low*).

Table 6.4. Extrinsic noise model predictions of circuit failure percentages.

Circuit Failures		OR circuit		NOR circuit	
		Original Design	Redesign	Original Design	Redesign
Function Hazard	$(1,0) \rightarrow (0,1)$	38.8	41.2	23.6	36.0
	$(0,1) \rightarrow (1,0)$	38.0	58.4	22.6	36.0
Set-Up Glitches	$(0,0)$	8.4	4.6	-	-
	$(0,1)$	-	-	42.8	49.0
	$(1,0)$	-	-	42.8	49.0
	$(1,1)$	-	-	36.8	45.2
Hold-States	$(0,0)$	4.6	2.8	3.4	8.8
	$(0,1)$	39.8	44.2	20.2	38.0
	$(1,0)$	39.8	44.2	19.8	38.0
	$(1,1)$	35.2	37.2	15.2	32.4

Percent failure predictions for input transitions that contain a function hazard, set-up glitches, and hold-state failures for different circuit layouts, using the extrinsic noise model. In this table, input transitions and states are described as a tuple (x,x), where the first value is the concentration of the first inducer (1: *high* and 0: *low*), and the second value is the concentration of the second inducer (1: *high* and 0: *low*).

Table 6.5. Intrinsic and extrinsic noise model predictions of circuit failure percentages.

Circuit Failures		OR circuit		NOR circuit	
		Original Design	Redesign	Original Design	Redesign
Function Hazard	$(1,0) \rightarrow (0,1)$	51.4	53.6	35.6	43.6
	$(0,1) \rightarrow (1,0)$	50.8	52.6	35.0	41.6
Set-Up Glitches	$(0,0)$	38.6	22.4	-	-
	$(0,1)$	-	-	67.4	48.2
	$(1,0)$	-	-	67.4	48.2
	$(1,1)$	-	-	45.0	70.6
Hold-States	$(0,0)$	22.8	12.4	25.0	37.2
	$(0,1)$	66.0	69.8	51.6	59.8
	$(1,0)$	66.0	70.0	51.4	60.6
	$(1,1)$	58.6	62.2	38.2	49.6

Percent failure predictions for input transitions that contain a function hazard, set-up glitches, and hold-state failures for different circuit layouts, using the combined intrinsic and extrinsic noise models. In this table, input transitions and states are described as a tuple (x,x), where the first value is the concentration of the first inducer (1: *high* and 0: *low*), and the second value is the concentration of the second inducer (1: *high* and 0: *low*).

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

Synthetic biology witnessed a surge of development over the past two decades [1, 95]. Nonetheless, genetic circuit complexity has not developed on par with genetic engineering technologies [79, 154]. This is not only due to the inherent complexity of biological systems and highly interconnected genetic parts, but also due to a lack of software that helps to cope with this intricacy. Model-driven design is of paramount importance when designing ever more complex genetic circuits. Modeling is instrumental to show faults in the genetic design, improve our understanding of underlying biological processes, and predict the dynamical transition stages of a genetic circuit and potential glitches in the system. However, devising a model for genetic circuits can be a tedious and complex endeavor. Additionally, parameterization is usually lacking for different models, thus making a model inaccurate.

We expect that all the methodologies developed in this work will serve the overarching aim of redesigning genetic circuits to avoid circuit failures. Facilitated dynamic modeling of genetic circuits would be an instrumental technique for synthetic biologists, especially if it can be accompanied by a circuit design automation tool, such as those proposed in this work, and a standardized method to detect and predict circuit failures. Furthermore, modeling noise to determine a genetic circuits' robustness to variability could provide further testing of *Genetic Regulatory Networks* (GRNs) designed to operate in *Outside-the-Laboratory Conditions* (OTLC). This would help automation in synthetic biology and provide a way to debug circuit designs before construction and compare predictions with experimental data once the synthesized circuit is implemented, saving time, effort, and money. This project aims to expand such capabilities to help researchers through the design process with the development of automated modeling, hazard identification, and genetic circuit redesign. A new model generation method and circuit failure analyses bridge the gap between experimentalists and designers as it helps both sides with the results obtained.

Designers can use data to better fit the model to produce more accurate predictions, and experimentalists can use these predictions to debug genetic circuits and predict their behavior before constructing them, saving time, effort, and money. However, most of this work is still a very manual process and requires specific expertise in certain topics that could prevent synthetic biologists on fully exploiting its benefits. Therefore, new ways of automating these methodologies are a priority to extend these benefits for all synthetic biologists, regardless of whether they have expertise in the topics developed in this dissertation or not, which is the topic of the next section.

7.1 Future Work

This thesis work shows that we can predict average expected behavior, and even do genetic circuit failure analysis, using generic or literature-obtained parameter values. However, the true potential benefits of modeling comes when characterized parameters for the specific GRN is used. Usually, experimentally-obtained parameter values are difficult to come by, especially for novel genetic parts that haven't been used in many designs yet. Even more difficult is to know how to characterize the experimental results using fitting algorithms or packages, since the correct method to do so varies depending on the type of gene-expression regulation that is being used in the design. Furthermore, characterization experiments are certainly *not* standardized, and even worse, characterization fitting algorithms/workflows/software tools are *greatly* missing, making it more of a blind-guessing game than anything else.

Hence, a future direction for design automation would be to standardize gate characterization experiments and automate fitting algorithms/software tools to ease the *Design-Build-Test-Scale* (DBTS) pipeline, which is the subject of the following subsections.

7.1.1 Test-Scale-Design Gap

Synthetic biology projects typically rely on iterative workflows composed of different tasks [96]. As mentioned in Section 1.5, this work is anticipated to be part of a larger workflow in the DBTS pipeline, shown in greater detail in **Figure 1.1** (b). There have already been contributions with different software developers that would fit in this workflow, such as Puppeteer [164] and, of course, Cello [129, 152]. Other projects are

aiming at parameterizing more genetic gates using Cello parametrization, and debugging a genetic circuit using RNA-seq [64]. The automatic model generator of this work would not only help with model predictions of genetic circuits before the building stage, but also in recognizing circuit failures, function hazards, and glitches of a circuit to either go back to the drawing board, or building the circuit with known restrictions on the circuit implementation. In all, the automated model generation of this work would help filter Cello's copious output designs for function and circuit anomalies using data standards such as SBOL, SBML, and SED-ML for reproducibility and reuse in the community.

However, most published research regarding genetic circuit design looks more like **Figure 1.1** (a), in which a genetic circuit is designed, and then tested to prove that it behaves according to what it is expected or not. As we move from proof-of-concept designs to real-life, reliable, and industrial applications of genetic circuit designs, a more thorough testing phase, followed by characterization efforts and redesigns, is needed to provide more robust genetic circuits. Therefore, to close the gap between the testing phase and the design (or redesign) phase of the DBTS workflow, efforts in automating characterization experiments to produce more streamlined analyses of GRN's performances under a wide range of experimental conditions is needed, which is the topic of the next subsections.

7.1.1.1 Experimental Data

Hypothesis or model driven genetic design is dependent upon experimental data [25, 90, 101]. Experimental data provides parameter values and validation data, and both modeling and experiments can profit from each other in an iterative learning process. With rapid characterization methods and ever-increasing options for using different DNA parts in constructs, the final challenge for synthetic biology is to develop new computational tools that capture the essential dynamics necessary to predict robustness and host-construct interactions to enable them to be considered in designs [13]. Therefore, automation of experiments and data acquisition can be achieved with the development of standard experimental protocols for synthetic biology. There are already efforts underway to initiate this endeavor, for example, to develop software to automatically generate and share experimental protocols using standardized specification languages and cloud-based tools [5, 74, 87, 93], or hardware to automate the experiments themselves, minimizing

human error. The advent of affordable and open-source automated robots, like those developed by Opetrons,[1] is paving the way for low-cost, reproducible and automated experimental workflows [162].

7.1.1.2 Characterization Experiments

Systems biology has developed new *-omics* tools that offer the potential to take a "snapshot" of the inner workings of a circuit in a single experiment [64]. One very popular transcriptomic method, *RNA sequencing* (RNA-seq), enables one to quantify the mRNA levels of each gate of a circuit with nucleotide resolution [168]. This high-throughput experiment allows for a complete analysis of a genetic circuit and its impact on the organism, the transfer functions of each genetic gate or part when not in isolation, maximum and minimum levels of transcription, and many other transcriptome analyses [170]. High-throughput characterization using RNAseq data and the model predictions of the mathematical model can help with the debugging and comparison of genetic circuits performance and underlying biological phenomena (like gate-toxicity) [64]. Additionally, characterization of new genetic parts not only allows for modeling of genetic circuits in other organisms besides *Escherichia coli* (*E. coli*), but also for different genetic contexts (like genomic vs. plasmid genetic circuits, or other host interactions), and experimental contexts. However, there is no consensus on how the characterization experiments for GRNs should be approached for each different regulatory system, or how to report these findings in a standardized way to enhance reproducibility [80].

7.1.1.3 Parametrization

Finding the best way to obtain parameter values from the characterization experiments can prove to be a significant effort, especially for those not proficient in this subject. The choice of a correct model, minimization algorithm, data-cleaning and normalization methods, and parameter-space exploration can be daunting for novice researchers or engineers who wish to use context-dependent parameter values for their DBTS pipeline. Furthermore, the lack of any widely-used software tools or automation processes to do so can expand the gap in the test-scale-design loop as it can prove a laborious undertaking

[1] https://opentrons.com/

for most synthetic biologists. Therefore, automation efforts should also be directed to the development of methods and software to automatically determine parameter values for common characterization experiments within the synthetic biology community.

This dissertation performs the parameterization using the Cello model [129, 152] using the lmfit Python package [126], and the workflow recommended in Dray et al. [43]. We are not necessarily advocating that this method is the best for the parametrization of repressible *Transcription Factor* (TF) genetic gates, but since most of the models and characterization experiment protocols are readily available, it is the most straightforward.

The development of a parametrization software/package that allows for the estimation of biochemical parameters from RNA-seq, Ribo-seq and proteomics profiling data could be implemented that would allow synthetic biologists to use the automated model generator of this work with genetic gates characterized in different environments or hosts. This means that the user could feed the automated model generator RNA-seq and Ribo-seq data, and the model generator could estimate the Cello (or model of choice) parametrization of each gate, and therefore automatically generate a dynamic model for the circuit with the correct parameter values.

The ideal modular, orthogonal genetic part, or gate would have the same response function in different genetic and biochemical contexts. However, this is rarely true for most genetic parts. Most genetic gates, when composed into a genetic circuit, have divergent behavior from when they are characterized in isolation. Furthermore, each organism has a unique biochemical and genetic environment that greatly influences the dynamic behavior of genetic gates. The automated estimation of parameters using RNA-seq and Ribo-seq could be used to estimate the Cello parameters of each gate when combined as a genetic circuit for a particular environment. This can help to calibrate already-known parameters of genetic gates when in different genetic or biochemical environments and account for the context-dependent variability of them. Moreover, if we want to have a predicted behavior of a circuit for an organism we do not have characterization for, we can adjust the model using Ribo and RNA-seq data of the most similar organism available, and have a better estimate of how a circuit would work on the novel chassis.

7.1.1.4 Gate Dynamics

Individual gate dynamics can have a critical influence in genetic circuit failures, as shown in Chapters 4 and 6. Delays in output production can induce incorrect temporary unwanted steady-states in GRNs, and therefore it is important to characterize. However, there has been little effort to standardize characterization experiments and parametrization for delays in genetic parts [152]. However, to accurately predict genetic circuit failures using dynamic modeling, these parameter values must be obtained, using characterization and parametrization workflows as proposed in earlier sections.

7.1.2 Noise Simulations

Sources of variability that have an effect on a GRN's output can also have an effect on genetic circuit failures. Chapter 5 explores some of these interactions. However, a more in-depth analysis of the relative effect and magnitude of extrinsic and intrinsic sources of noise for GRNs is appropriate for when the robustness of a genetic circuit is being considered for OTLC applications. The following subsections will describe some advantages of noise modeling.

7.1.2.1 Parametric Sensitivity Analysis

A *parametric sensitivity analysis* (PSA) analyses how changes in the output of a model can be appropriated to different parameter values with a wide range of applications for systems biology. The reasons for performing a PSA may vary from the determination of which parameters require additional research at the stage of model calibration and identification, to analysis of the robustness of a circuit and the model results, to a reduction or abstraction of the model via the identification of the parameters that are not relevant for its dynamics [38]. This is useful not only during the design process of a circuit, but also on the iterative build/test/scale process workflow once a genetic circuit is designed. It can help researchers learn which genetic gates need a bigger difference in the ON/OFF promoter activity, or which gates need more isolation/stability from the environment, to identify model predictions that are inconsistent with experimental data, suggesting novel experimentation to either validate or falsify a model and many other uses [88].

Adding noise models to dynamic GRN models can increase the number of parameter values on which one can perform a PSA. For example, a stochastic model would also

allow one to implement a sensitivity analysis of the kinetic model [37, 38, 65, 73, 105], which would allow one to systematically study the dependence of the quantities of interest on the parameters that define the model and the initial conditions in which it is simulated. The automated model generator of this work could be adapted to have the option to perform a PSA on the automatically generated stochastic model.

7.1.2.2 Glitch Propensity

An automatically generated stochastic model can also allow one to calculate the *probability* of certain states or dynamic behavior, running the analysis multiple times and computing the probability from the results. In this dissertation, we analyzed genetic circuit failures using the deterministic-continuous approach of an *Ordinary Differential Equation* (ODE) model analysis, adding variability as extrinsic noise for each simulation run. However, since the parameter values of the noise models used are unknown, the glitch propensities obtained are *relative*, and do not necessarily provide an actual failure percentage. However, with analysis of a stochastic model of the same circuits, using both extrinsic and intrinsic noise modeling and characterized parameter values, we could compute the *probabilities* of each possible glitch behavior and have the automated model generator report these probabilities. In this way, anyone using this model generator can have an idea of the risks of this unwanted switching behavior happening and determine if they need to restrict the allowed input changes to the circuit to make certain that the glitching behavior does not occur.

7.1.2.3 Circuit Performance or Robustness

Circuit performance can be thought of as the capacity of a circuit to reproduce faithfully or successfully the truth table. There are three essential factors that determine the performance of a circuit, which are the extent that the *high* and *low* output signal of each genetic gate can be practically distinguished, and the transient dynamics after changes in the inputs that may produce incorrect results [100, 104], and the effect of noise on a GRN's output.

This dissertation simulates an extrinsic noise model by generating normal-distributed parameter values on which the model parameters were drawn from on each run. However, the *magnitude* of the extrinsic noise model was fixed for all simulations (see Chapter 5).

Without needing the verified noise sources and magnitudes for each gate, a researcher could vary, for each simulation, the simulated intrinsic and/or extrinsic noise for each reaction, in an effort to determine the sensitivity of a GRN's output to noise. This could serve as a measurement of a circuit's robustness or performance.

The automated model generator of this work could be expanded to calculate a predicted circuit performance. This metric could be based solely on a statistical analysis of the difference between the predicted steady-state output and experimental results. Marchisio et al. [100, 104] propose the difference of the minimal and maximal output at steady-state for each gate as a main parameter to quantify gate and circuit performance. Whichever method is preferred, a report could be provided as soon as the automated model generator produces the dynamic model, so as to help the user in the design process of a genetic circuit.

REFERENCES

[1] E. Andrianantoandro, S. Basu, D. K. Karig, and R. Weiss. Synthetic biology: New engineering rules for an emerging discipline. *Mol. Syst. Biol.*, 2(1):205–211, 2006.

[2] E. Appleton, C. Madsen, N. Roehner, and D. Densmore. Design automation in synthetic biology. *Cold Spring Harbor Perspect. Biol.*, 9(4):A023978, 2017.

[3] G. Baldwin, T. Bayer, R. Dickinson, T. Ellis, P. S. Freemont, R. I. Kitney, K. M. Polizzi, and G.-B. Stan, editors. *Synthetic biology: A primer*. Imperial College Press; World Scientific Publishing Co. Pte. Ltd, Hackensack, NJ, 2016.

[4] V. Bartoli, G. A. Meaker, M. di Bernardo, and T. E. Gorochowski. Tunable genetic devices through simultaneous control of transcription and translation. *Nat. Commun.*, 11(1):2095, 2020.

[5] M. Bates, A. J. Berliner, J. Lachoff, P. R. Jaschke, and E. S. Groban. Wet Lab Accelerator: A web-based application democratizing laboratory automation for synthetic biology. *ACS Synth. Biol.*, 6(1):167–171, 2017.

[6] J. Beal. Biochemical complexity drives log-normal variation in genetic expression. *Eng. Biol.*, 1(1):55–60, 2017.

[7] M. R. Bennett, D. Volfson, L. Tsimring, and J. Hasty. Transient dynamics of genetic regulatory networks. *Biophys. J.*, 92(10):3501–3512, 2007.

[8] F. T. Bergmann, R. Adams, S. Moodie, J. Cooper, M. Glont, M. Golebiewski, M. Hucka, C. Laibe, A. K. Miller, D. P. Nickerson, B. G. Olivier, N. Rodriguez, H. M. Sauro, M. Scharm, S. Soiland-Reyes, D. Waltemath, F. Yvon, and N. Le Novère. COMBINE archive and OMEX format: One file to share all information to reproduce a modeling project. *BMC Bioinform.*, 15(1):369, 2014.

[9] J. A. Bernstein, A. B. Khodursky, P.-H. Lin, S. Lin-Chao, and S. N. Cohen. Global analysis of mRNA decay and abundance in *Escherichia coli* at single-gene resolution using two-color fluorescent DNA microarrays. *PNAS*, 99(15):9697–9702, 2002.

[10] C. R. Boehm and R. Bock. Recent advances and current challenges in synthetic biology of the plastid genetic system and metabolism. *Plant Physiol.*, 179(3):794–802, 2019.

[11] H. Bolouri. *Computational modelling of gene regulatory networks – A primer*. Imperial College Press, London, 2008.

[12] A. E. Bordoy, N. J. O'Connor, and A. Chatterjee. Construction of two-input logic gates using transcriptional interference. *ACS Synth. Biol.*, 8(10):2428–2441, 2019.

[13] O. Borkowski, F. Ceroni, G.-B. Stan, and T. Ellis. Overloaded and stressed: Whole-cell considerations for bacterial synthetic biology. *Curr. Opin. Microbiol.*, 33:123–130, 2016.

[14] B. J. Bornstein, S. M. Keating, A. Jouraku, and M. Hucka. LibSBML: An API library for SBML. *Bioinformatics*, 24(6):880–881, 2008.

[15] F. Boulier, M. Lefranc, F. Lemaire, and P.-E. Morant. Applying a rigorous quasi-steady state approximation method for proving the absence of oscillations in models of genetic circuits. In K. Horimoto, G. Regensburger, M. Rosenkranz, and H. Yoshida, editors, *Algebraic biology*, pages 56–64. Springer Berlin Heidelberg, Berlin, 2008.

[16] R. W. Bradley, M. Buck, and B. Wang. Tools and principles for microbial gene circuit engineering. *J. Mol. Biol.*, 428(5, Part B):862–888, 2016.

[17] J. G. Bredeson and P. T. Hulina. Elimination of static and dynamic hazards for multiple input changes in combinational switching circuits. *Inform. Contr.*, 20(2):114–124, 1972.

[18] S. M. Brooks and H. S. Alper. Applications, challenges, and needs for employing synthetic biology beyond the lab. *Nat. Commun.*, 12(1):1390, 2021.

[19] J. A. N. Brophy and C. A. Voigt. Principles of genetic circuit design. *Nat. Methods*, 11(5):508–520, 2014.

[20] L. Buecherl, R. Roberts, P. Fontanarrosa, P. J. Thomas, J. Mante, Z. Zhang, and C. J. Myers. Stochastic hazard analysis of genetic circuits in iBioSim and STAMINA. *ACS Synth. Biol.*, 10(100:2532–2540, 2021.

[21] B. P. Callen, K. E. Shearwin, and J. B. Egan. Transcriptional interference between convergent promoters caused by elongation over the promoter. *Mol. Cell*, 14(5):647–656, 2004.

[22] D. E. Cameron, C. J. Bashor, and J. J. Collins. A brief history of synthetic biology. *Nat. Rev. Microbiol.*, 12(5):381–390, 2014.

[23] B. Canton, A. Labno, and D. Endy. Refinement and standardization of synthetic biological parts and devices. *Nat. Biotechnol.*, 26(7):787–793, 2008.

[24] S. Cardinale and A. P. Arkin. Contextualizing context for synthetic biology – Identifying causes of failure of synthetic biological systems. *Biotechnol. J.*, 7(7):856–866, 2012.

[25] D. Chandran, W. Copeland, S. Sleight, and H. Sauro. Mathematical modeling and synthetic biology. *Drug Dis. Today Dis. Models*, 5(4):299–309, 2008.

[26] C. Chaouiya, D. Bérenguier, S. M. Keating, A. Naldi, M. P. van Iersel, N. Rodriguez, A. Dräger, F. Büchel, T. Cokelaer, B. Kowal, B. Wicks, E. Gonçalves, J. Dorier, M. Page, P. T. Monteiro, A. von Kamp, I. Xenarios, H. de Jong, M. Hucka, S. Klamt, D. Thieffry, N. Le Novère, J. Saez-Rodriguez, and T. Helikar. SBML qualitative models: A model representation format and infrastructure to foster interactions between qualitative modelling formalisms and tools. *BMC Syst. Biol.*, 7(1):135, 2013.

[27] V. Chelliah, C. Laibe, and N. L. Novère. BioModels database: A repository of mathematical models of biological processes. In W. Dubitzky, O. Wolkenhauer, K.-H. Cho, and H. Yokota, editors, *Encyclopedia of systems biology*, pages 134–138. Springer New York, New York, NY, 2013.

[28] Y.-J. Chen, P. Liu, A. A. K. Nielsen, J. A. N. Brophy, K. Clancy, T. Peterson, and C. A. Voigt. Characterization of 582 natural and synthetic terminators and quantification of their design constraints. *Nat. Methods*, 10(7):659–664, 2013.

[29] K. Clancy and C. A. Voigt. Programming cells: Towards an automated 'genetic compiler'. *Curr. Opin. Biotech.*, 21(4):572–581, 2010.

[30] R. S. Cox, C. Madsen, J. McLaughlin, T. Nguyen, N. Roehner, B. Bartley, S. Bhatia, M. Bissell, K. Clancy, T. Gorochowski, R. Grünberg, A. Luna, N. N. Le, M. Pocock, H. Sauro, J. T. Sexton, G.-B. Stan, J. J. Tabor, C. A. Voigt, Z. Zundel, C. Myers, J. Beal, and A. Wipat. Synthetic Biology Open Language Visual (SBOL Visual) Version 2.0. *J. Integr. Bioinform.*, 15(1):20170074, 2018.

[31] N. Crook and H. S. Alper. Model-based design of synthetic, biological systems. *Chem. Eng. Sci.*, 103:2–11, 2013.

[32] B. Cummins, T. Gedeon, S. Harker, and K. Mischaikow. Database of Dynamic Signatures Generated by Regulatory Networks (DSGRN). In J. Feret and H. Koeppl, editors, *Computational methods in systems biology*, pages 300–308, Springer International Publishing, Cham, Switzerland, 2017.

[33] B. Cummins, T. Gedeon, S. Harker, and K. Mischaikow. DSGRN: Examining the Dynamics of families of logical models. *Front. Physiol.*, 9:549, 2018.

[34] B. Cummins, T. Gedeon, S. Harker, K. Mischaikow, and K. Mok. Combinatorial representation of parameter space for switching networks. *SIAM J. Appl. Dyn. Syst.*, 15(4):2176–2212, 2016.

[35] B. Cummins, R. C. Moseley, A. Deckard, M. Weston, G. Zheng, D. Bryce, S. Gopaulakrishnan, T. Johnson, J. Nowak, M. Gameiro, T. Gedeon, K. Mischaikow, M. Vaughn, N. I. Gaffney, J. Urrutia, R. P. Goldman, J. Beal, B. Bartley, T. T. Nguyen, N. Roehner, T. Mitchell, J. D. Vrana, K. J. Clowers, N. Maheshri, D. Becker, E. Mikhalev, V. Biggers, T. R. Higa, L. A. Mosqueda, and S. B. Haase. Computational prediction of synthetic circuit function across growth conditions, June 2022. bioRxiv: 2022.06.13.495701.

[36] B. Cummins, J. Vrana, R. C. Moseley, H. Eramian, A. Deckard, P. Fontanarrosa, D. Bryce, M. Weston, G. Zheng, J. Nowak, F. C. Motta, M. Eslami, K. L. Johnson, R. P. Goldman, C. J. Myers, T. Johnson, M. W. Vaughn, N. Gaffney, J. Urrutia, S. Gopaulakrishnan, V. Biggers, T. R. Higa, L. A. Mosqueda, M. Gameiro, T. Gedeon, K. Mischaikow, J. Beal, B. Bartley, T. Mitchell, T. T. Nguyen, N. Roehner, and S. B. Haase. Robustness and reproducibility of simple and complex synthetic logic circuit designs using a DBTL loop, June 2022. bioRxiv: 2022.06.10.495560.

[37] D. K. Dacol and H. Rabitz. Sensitivity analysis of stochastic kinetic models. *J. Math. Phys.*, 25(9):2716–2727, 1984.

[38] C. Damiani, A. Filisetti, A. Graudenzi, and P. Lecca. Parameter sensitivity analysis of stochastic models: Application to catalytic reaction networks. *Comput. Biol. Chem.*, 42:5–17, 2013.

[39] N. Davidsohn, J. Beal, S. Kiani, A. Adler, F. Yaman, Y. Li, Z. Xie, and R. Weiss. Accurate predictions of genetic circuit behavior from part characterization and modular composition. *ACS Synth. Biol.*, 4(6):673–681, 2015.

[40] H. de Jong. Modeling and simulation of genetic regulatory systems: A literature review. *J. Comp. Biol.*, 9(1):67–103, 2002.

[41] T. Decoene, B. D. Paepe, J. Maertens, P. Coussement, G. Peters, S. L. D. Maeseneire, and M. D. Mey. Standardization in synthetic biology: An engineering discipline coming of age. *Crit. Rev. Biotechnol.*, 38(5):647–656, 2018.

[42] R. W. Doran. The gray code. *J. UCS*, 13(11):1573–1597, 2007.

[43] K. E. Dray, J. J. Muldoon, N. M. Mangan, N. Bagheri, and J. N. Leonard. GAMES: A dynamic model development workflow for rigorous characterization of synthetic genetic systems. *ACS Synth. Biol.*, 11(2):1009–1029, 2022.

[44] D. A. Drubin, J. C. Way, and P. A. Silver. Designing biological systems. *Genes Dev.*, 21(3):242–254, 2007.

[45] E. B. Eichelberger. Hazard detection in combinational and sequential switching circuits. *IBM J. Res. Dev.*, 9(2):90–99, 1965.

[46] M. B. Elowitz and S. Leibler. A synthetic oscillatory network of transcriptional regulators. *Nature*, 403(6767):335–338, 2000.

[47] M. B. Elowitz, A. J. Levine, E. D. Siggia, and P. S. Swain. Stochastic gene expression in a single cell. *Science*, 297(5584):1183–1186, 2002.

[48] D. Endy. Foundations for engineering biology. *Nature*, 438(7067):449–453, 2005.

[49] V. Epshtein, F. Toulmé, A. R. Rahmouni, S. Borukhov, and E. Nudler. Transcription through the roadblocks: The role of RNA polymerase cooperation. *EMBO J.*, 22(18):4719–4727, 2003.

[50] M. Eslami, A. E. Borujeni, H. Doosthosseini, M. Vaughn, H. Eramian, K. Clowers, D. B. Gordon, N. Gaffney, M. Weston, D. Becker, Y. Dorfan, J. Fonner, J. Urrutia, C. Corbet, G. Zheng, J. Stubbs, A. Cristofaro, P. Maschhoff, J. Singer, C. A. Voigt, and E. Yeung. Prediction of whole-cell transcriptional response with machine learning, May 2021, bioRxiv: 2021.04.30.442142.

[51] A. Espah Borujeni, J. Zhang, H. Doosthosseini, A. A. K. Nielsen, and C. A. Voigt. Genetic circuit characterization by inferring RNA polymerase movement and ribosome usage. *Nat. Commun.*, 11(1):5001, 2020.

[52] R. P. Goldman et al. Highly-automated, high-throughput replication of yeast-based logic circuit design assessments, June 2022, bioRxiv: 2022.05.31.493627.

[53] E. Fehlberg. Low-order classical Runge-Kutta formulas with stepsize control and their application to some heat transfer problems. Technical Report NASA TR R-315, National Aeronautics and Space Administration, Washington, DC, July 1969.

[54] P. Fontanarrosa. Automated generation of dynamic models for genetic regulatory networks. Master's Thesis, University of Utah, Salt Lake City, Utah, Dec. 2019.

[55] P. Fontanarrosa, H. Doosthosseini, A. Espah Borujeni, Y. Dorfan, C. A. Voigt, and C. J. Myers. Genetic circuit dynamics: Hazard and glitch analysis. *ACS Synth. Biol.*, 9(9):2324–2338, 2020.

[56] M. Galdzicki, K. P. Clancy, E. Oberortner, M. Pocock, J. Y. Quinn, C. A. Rodriguez, N. Roehner, M. L. Wilson, L. Adam, J. C. Anderson, B. A. Bartley, J. Beal, D. Chandran, J. Chen, D. Densmore, D. Endy, R. Grünberg, J. Hallinan, N. J. Hillson, J. D. Johnson, A. Kuchinsky, M. Lux, G. Misirli, J. Peccoud, H. A. Plahar, E. Sirin, G.-B. Stan, A. Villalobos, A. Wipat, J. H. Gennari, C. J. Myers, and H. M. Sauro. The Synthetic Biology Open Language (SBOL) provides a community standard for communicating designs in synthetic biology. *Nat. Biotechnol.*, 32(6):545–550, 2014.

[57] M. Gameiro, T. Gedeon, S. Kepley, and K. Mischaikow. Rational design of complex phenotype via network models. *PLoS Comput. Biol.*, 17(7):e1009189, 2021.

[58] M. W. Gander, J. D. Vrana, W. E. Voje, J. M. Carothers, and E. Klavins. Digital logic circuits in yeast with CRISPR-dCas9 NOR gates. *Nat. Commun.*, 8:15459, 2017.

[59] M. W. Gander, J. D. Vrana, W. E. Voje, J. M. Carothers, and E. Klavins. Digital logic circuits in yeast with CRISPR-dCas9 NOR gates. *Nat. Commun.*, 8:15459, 2017.

[60] T. S. Gardner, C. R. Cantor, and J. J. Collins. Construction of a genetic toggle switch in *Escherichia coli*. *Nature*, 403(6767):339–342, 2000.

[61] K. L. Garner. Principles of synthetic biology. *Essays Biochem.*, 65(5):791–811, 2021.

[62] S. Ghaemmaghami, W.-K. Huh, K. Bower, R. W. Howson, A. Belle, N. Dephoure, E. K. O'Shea, and J. S. Weissman. Global analysis of protein expression in yeast. *Nature*, 425(6959):737, Oct. 2003.

[63] D. T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.*, 81(25):2340–2361, Dec. 1977.

[64] T. E. Gorochowski, A. E. Borujeni, Y. Park, A. A. Nielsen, J. Zhang, B. S. Der, D. B. Gordon, and C. A. Voigt. Genetic circuit characterization and debugging using RNA-seq. *Mol. Syst. Biol.*, 13(11):952, Nov. 2017.

[65] A. Gupta and M. Khammash. An efficient and unbiased method for sensitivity analysis of stochastic reaction networks. *J. R. Soc. Interface*, 11(101):20140979, 2014.

[66] R. Hackbart and D. Dietmeyer. The avoidance and elimination of function hazards in asynchronous sequential circuits. *IEEE Trans. Comput.*, C-20(2):184–189, 1971.

[67] T. S. Ham, Z. Dmytriv, H. Plahar, J. Chen, N. J. Hillson, and J. D. Keasling. Design, implementation and practice of JBEI-ICE: An open source biological part registry platform and tools. *Nucleic Acids Res.*, 40(18):e141–e141, Oct. 2012.

[68] A. Hasnain, S. Sinha, Y. Dorfan, A. E. Borujeni, Y. Park, P. Maschhoff, U. Saxena, J. Urrutia, N. Gaffney, D. Becker, A. Siba, N. Maheshri, B. Gordon, C. Voigt, and E. Yeung. A data-driven method for quantifying the impact of a genetic circuit on its host. In *2019 IEEE Biomed. Circ. Syst. Conf.*, BioCAS '19, pages 1–4, Nara, Japan, 2019. IEEE.

[69] E. C. Hayden. Synthetic biology called to order: Meeting launches effort to develop standards for fast-moving field. *Nature*, 520(7546):141–143, 2015.

[70] M. Heinemann and S. Panke. Synthetic biology—Putting engineering into biology. *Bioinformatics*, 22(22):2790–2799, 2006.

[71] V. Hsiao, A. Swaminathan, and R. M. Murray. Control theory for synthetic biology: Recent advances in system characterization, control design, and controller implementation for synthetic biology. *IEEE Control Syst. Mag.*, 38(3):32–62, 2018.

[72] M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, A. P. Arkin, B. J. Bornstein, D. Bray, A. Cornish-Bowden, A. A. Cuellar, S. Dronov, E. D. Gilles, M. Ginkel, V. Gor, I. I. Goryanin, W. J. Hedley, T. C. Hodgman, J.-H. Hofmeyr, P. J. Hunter, N. S. Juty, J. L. Kasberger, A. Kremling, U. Kummer, N. Le Novère, L. M. Loew, D. Lucio, P. Mendes, E. Minch, E. D. Mjolsness, Y. Nakayama, M. R. Nelson, P. F. Nielsen, T. Sakurada, J. C. Schaff, B. E. Shapiro, T. S. Shimizu, H. D. Spence, J. Stelling, K. Takahashi, M. Tomita, J. Wagner, and J. Wang. The systems biology markup language (SBML): A medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531, 2003.

[73] A. D. Irving. Stochastic sensitivity analysis. *Appl. Math. Model.*, 16(1):3–15, 1992.

[74] M. M. Jessop-Fabre and N. Sonnenschein. Improving reproducibility in synthetic biology. *Front. Bioeng. Biotechnol.*, 7:18, 2019.

[75] T. S. Jones, S. M. D. Oliveira, C. J. Myers, C. A. Voigt, and D. Densmore. Genetic circuit design automation with Cello 2.0. *Nat. Protoc*, 17:1097–1113, 2022.

[76] E. M. Judd, M. T. Laub, and H. H. McAdams. Toggles and oscillators: New genetic circuit designs. *BioEssays*, 22(6):507–509, 2000.

[77] G. Karlebach and R. Shamir. Modelling and analysis of gene regulatory networks. *Nat. Rev. Mol. Cell Biol.*, 9(10):770–780, 2008.

[78] M. Karnaugh. The map method for synthesis of combinational logic circuits. *Trans. Am. Inst. Electr. Eng. Part 1*, 72(5):593–599, 1953.

[79] Y. N. Kaznessis. Models for synthetic biology. *BMC Syst. Biol.*, 1(1):47, 2007.

[80] J. R. Kelly, A. J. Rubin, J. H. Davis, C. M. Ajo-Franklin, J. Cumbers, M. J. Czar, K. de Mora, A. L. Glieberman, D. D. Monie, and D. Endy. Measuring the activity of BioBrick promoters using an in vivo reference standard. *J. Biol. Eng.*, 3(1):4, 2009.

[81] E. Klipp, W. Liebermeister, A. Helbig, A. Kowald, and J. Schaber. Systems biology standards—The community speaks. *Nat. Biotechnol.*, 25:390–391, 2007.

[82] T. Knight. DARPA BioComp plasmid distribution 1.00 of standard Biobrick components. Technical Report, Massachusetts Institute of Technology Artificial Intelligence Lab, Cambridge, MA, May 2002.

[83] T. Knight. Idempotent vector design for standard assembly of biobricks. Techincal Report, Massachusetts Institute of Technology Artificial Intelligence Lab, Cambridge, MA, Jan. 2003.

[84] S. Kosuri, D. B. Goodman, G. Cambray, V. K. Mutalik, Y. Gao, A. P. Arkin, D. Endy, and G. M. Church. Composability of regulatory sequences controlling transcription and translation in *Escherichia coli*. *PNAS*, 110(34):14024–14029, 2013.

[85] R. Kwok. Five hard truths for synthetic biology. *Nature*, 463(7279):288–290, 2010.

[86] N. Le Novère, M. Hucka, N. Anwar, G. D. Bader, E. Demir, S. Moodie, and A. Sorokin. Meeting report from the first meetings of the Computational Modeling in Biology Network (COMBINE). *Stand. Genomic. Sci.*, 5(2):230–242, 2011.

[87] P. L. Lee and B. N. Miles. Autoprotocol driven robotic cloud lab enables systematic machine learning approaches to designing, optimizing, and discovering novel biological synthesis pathways. In *SIMB Annu. Meeting*, SIMB 2018, Chicago, IL, 2018.

[88] Y.-S. Lee, O. Z. Liu, H. S. Hwang, B. C. Knollmann, and E. A. Sobie. Parameter sensitivity analysis of stochastic models provides insights into cardiac calcium sparks. *Biophys. J.*, 104(5):1142–1150, 2013.

[89] I. Lestas, J. Paulsson, N. E. Ross, and G. Vinnicombe. Noise in gene regulatory networks. *IEEE Trans. Autom. Control*, 53:189–200, 2008.

[90] P. Li, J. O. Dada, D. Jameson, I. Spasic, N. Swainston, K. Carroll, W. Dunn, F. Khan, N. Malys, H. L. Messiha, E. Simeonidis, D. Weichart, C. Winder, J. Wishart, D. S. Broomhead, C. A. Goble, S. J. Gaskell, D. B. Kell, H. V. Westerhoff, P. Mendes, and N. W. Paton. Systematic integration of experimental data and models in systems biology. *BMC Bioinf.*, 11:582, 2010.

[91] Z. Li and Q. Yang. Systems and synthetic biology approaches in understanding biological oscillators. *Quantit. Biol.*, 6(1):1–14, 2018.

[92] W. A. Lim. Designing customized cell signalling circuits. *Nat. Rev. Mol. Cell Biol.*, 11(6):393–403, 2010.

[93] G. Linshiz, N. Stawski, S. Poust, C. Bi, J. D. Keasling, and N. J. Hillson. PaR-PaR laboratory automation platform. *ACS Synth. Biol.*, 2(5):216–222, 2013.

[94] T. K. Lu, A. S. Khalil, and J. J. Collins. Next-generation synthetic gene networks. *Nat. Biotechnol.*, 27(12):1139–1150, 2009.

[95] E. Lukas, R. Nicolas, J. Nick, C. Vijayalakshmi, L. Camille, L. Chen, and L. N. Nicolas. Designing and encoding models for synthetic biology. *J. R. Soc. Interface*, 6:S405–S417, 2009.

[96] M. W. Lux, B. W. Bramlett, D. A. Ball, and J. Peccoud. Genetic design automation: Engineering fantasy or scientific renewal? *Trends Biotechnol.*, 30(2):120–126, 2012.

[97] J. T. MacDonald, C. Barnes, R. I. Kitney, P. S. Freemont, and G.-B. V. Stan. Computational design approaches and tools for synthetic biology. *Integr. Biol.*, 3(2):97–108, 2011.

[98] C. Madsen, C. J. Myers, T. Patterson, N. Roehner, J. T. Stevens, and C. Winstead. Design and test of genetic circuits using iBioSim. *IEEE Des. Test Comput.*, 29(3):32–39, 2012.

[99] J. Mante, Y. Hao, J. Jett, U. Joshi, K. Keating, X. Lu, G. Nakum, N. E. Rodriguez, J. Tang, L. Terry, X. Wu, E. Yu, J. S. Downie, B. T. McInnes, M. H. Nguyen, B. Sepulvado, E. M. Young, and C. J. Myers. Synthetic biology knowledge system. *ACS Synth. Biol.*, 10(9):2276–2285, 2021.

[100] M. A. Marchisio. Parts & Pools: A framework for modular design of synthetic gene circuits. *Front. Bioeng. Biotechnol.*, 2:42, 2014.

[101] M. A. Marchisio, editor. *Computational methods in synthetic biology*. Humana Press, New York, NY, 2015.

[102] M. A. Marchisio. *Introduction in synthetic biology: About modeling, computation, and circuit design*. Springer Berlin Heidelberg, New York, NY, 2018.

[103] M. A. Marchisio and J. Stelling. Computational design of synthetic gene circuits with composable parts. *Bioinformatics*, 24(17):1903–1910, 2008.

[104] M. A. Marchisio and J. Stelling. Automatic design of digital synthetic gene circuits. *PLoS Comput. Biol.*, 7(2):e1001083, 2011.

[105] S. Marino, I. B. Hogue, C. J. Ray, and D. E. Kirschner. A methodology for performing global uncertainty and sensitivity analysis in systems biology. *J. Theor. Biol.*, 254(1):178–196, 2008.

[106] A. Mazo, J. W. Hodgson, S. Petruk, Y. Sedkov, and H. W. Brock. Transcriptional interference: An unexpected layer of complexity in gene regulation. *J. Cell Sci.*, 120(16):2755–2761, 2007.

[107] J. A. McLaughlin, C. J. Myers, Z. Zundel, G. Mısırlı, M. Zhang, I. D. Ofiteru, A. Goñi-Moreno, and A. Wipat. SynBioHub: A standards-enabled design repository for synthetic biology. *ACS Synth. Biol.*, 7(2):682–688, 2018.

[108] G. Mısırlı, J. Hallinan, M. Pocock, P. Lord, J. A. McLaughlin, H. Sauro, and A. Wipat. Data integration and mining for synthetic biology design. *ACS Synth. Biol.*, 5(10):1086–1097, 2016.

[109] G. Misirli, J. Hallinan, and A. Wipat. Composable modular models for synthetic biology. *ACM J. Emerg. Technol. Comput. Syst.*, 11(3):22:1–22:19, 2014.

[110] G. Misirli, T. Nguyen, J. A. McLaughlin, P. Vaidyanathan, T. S. Jones, D. Densmore, C. Myers, and A. Wipat. A computational workflow for the automated generation of models of genetic designs. *ACS Synth. Biol.*, 11(2):1548–1559, 2018.

[111] G. Misirli, A. Wipat, J. Mullen, K. James, M. Pocock, W. Smith, N. Allenby, and J. S. Hallinan. BacillOndex: An integrated data resource for systems and synthetic biology. *J. Integr. Bioinform.*, 10(2):224, 2013.

[112] T. S. Moon, C. Lou, A. Tamsir, B. C. Stanton, and C. A. Voigt. Genetic programs constructed from layered logic gates in single cells. *Nature*, 491(7423):249–253, 2012.

[113] F. Moser, A. E. Borujeni, A. N. Ghodasara, E. Cameron, Y. Park, and C. A. Voigt. Dynamic control of endogenous metabolism with combinatorial logic circuits. *Mol. Syst. Biol.*, 14(11):e8605, 2018.

[114] S. Mukherji and A. van Oudenaarden. Synthetic biology: Understanding biological design from synthetic circuits. *Nat. Rev. Genet.*, 10(12):859–871, 2009.

[115] V. K. Mutalik, J. C. Guimaraes, G. Cambray, C. Lam, M. J. Christoffersen, Q.-A. Mai, A. B. Tran, M. Paull, J. D. Keasling, A. P. Arkin, and D. Endy. Precise and reliable gene expression via standard transcription and translation initiation elements. *Nat. Methods*, 10(4):354–360, 2013.

[116] V. K. Mutalik, J. C. Guimaraes, G. Cambray, Q.-A. Mai, M. J. Christoffersen, L. Martin, A. Yu, C. Lam, C. Rodriguez, G. Bennett, J. D. Keasling, D. Endy, and A. P. Arkin. Quantitative estimation of activity and quality for collections of functional genetic elements. *Nat. Methods*, 10(4):347–353, 2013.

[117] V. K. Mutalik, L. Qi, J. C. Guimaraes, J. B. Lucks, and A. P. Arkin. Rationally designed families of orthogonal RNA regulators of translation. *Nat. Chem. Biol.*, 8(5):447–454, 2012.

[118] C. J. Myers. *Asynchronous circuit design*. John Wiley & Sons, New York, NY, 2001.

[119] C. J. Myers. *Engineering genetic circuits*. Chapman and Hall/CRC, Boca Raton, FL, 2016.

[120] C. J. Myers. Computational synthetic biology: Progress and the road ahead. *IEEE Trans. Multi-Scale Comput. Syst.*, 1(1):19–32, 2015.

[121] C. J. Myers, N. Barker, K. Jones, H. Kuwahara, C. Madsen, and N.-P. D. Nguyen. iBioSim: A tool for the analysis and design of genetic circuits. *Bioinformatics*, 25(21):2848–2849, 2009.

[122] C. J. Myers, J. Beal, T. E. Gorochowski, H. Kuwahara, C. Madsen, J. A. McLaughlin, G. Mısırlı, T. Nguyen, E. Oberortner, M. Samineni, A. Wipat, M. Zhang, and Z. Zundel. A standard-enabled workflow for synthetic biology. *Biochem. Soc. Trans.*, 45(3):793–803, 2017.

[123] N. Nandagopal and M. B. Elowitz. Synthetic biology: Integrated gene circuits. *Science*, 333(6047):1244–1248, 2011.

[124] J. A. Nelder and R. Mead. A simplex method for function minimization. *Comput. J.*, 7(4):308–313, 1965.

[125] T. Neupane, C. J. Myers, C. Madsen, H. Zheng, and Z. Zhang. STAMINA: STochastic Approximate Model-Checker for INfinite-State Analysis. In I. Dillig and S. Tasiran, editors, *Computer aided verification*, pages 540–549, Springer International Publishing, Cham, Switzerland, 2019.

[126] M. Newville, R. Otten, A. Nelson, A. Ingargiola, T. Stensitzki, D. Allan, A. Fox, F. Carter, Michał, R. Osborn, D. Pustakhod, lneuhaus, S. Weigand, Glenn, C. Deil, Mark, A. L. R. Hansen, G. Pasquevich, L. Foks, N. Zobrist, O. Frost, A. Beelen, Stuermer, azelcer, A. Hannum, A. Polloreno, J. H. Nielsen, S. Caldwell, A. Almarza, and A. Persaud. Lmfit/lmfit-py: 1.0.3. Zenodo, Oct. 2021.

[127] T. Nguyen, T. S. Jones, P. Fontanarrosa, J. V. Mante, Z. Zundel, D. Densmore, and C. J. Myers. Design of asynchronous genetic circuits. *Proc. IEEE*, 107(7):1356–1368, 2019.

[128] T. Nguyen, N. Roehner, Z. Zundel, and C. J. Myers. A converter from the systems biology markup language to the synthetic biology open language. *ACS Synth. Biol.*, 5(6):479–486, 2016.

[129] A. A. K. Nielsen, B. S. Der, J. Shin, P. Vaidyanathan, V. Paralanov, E. A. Strychalski, D. Ross, D. Densmore, and C. A. Voigt. Genetic circuit design automation. *Science*, 352(6281):aac7341, 2016.

[130] E.-M. Nikolados, A. Y. Weiße, F. Ceroni, and D. A. Oyarzún. Growth defects and loss-of-function in synthetic gene circuits. *ACS Synth. Biol.*, 8(6):1231–1240, 2019.

[131] M. Padidam and Y. Cao. Elimination of transcriptional interference between tandem genes in plant cells. *BioTechniques*, 31(2):328–334, 2001.

[132] J. Paulsson. Summing up the noise in gene networks. *Nature*, 427(6973):415–418, 2004.

[133] W. R. Pearson and D. J. Lipman. Improved tools for biological sequence comparison. *PNAS*, 85(8):2444–2448, 1988.

[134] J. Peccoud. Synthetic biology: Fostering the cyber-biological revolution. *Synth. Biol.*, 1(1):ysw001, 2016.

[135] I. Phillips and P. Silver. A new biobrick assembly strategy designed for facile protein engineering. Technical Report, Massachusetts Institute of Technology, Cambridge, MA, Apr. 2006.

[136] A. Polynikis, S. J. Hogan, and M. di Bernardo. Comparing different ODE modelling approaches for gene regulatory networks. *J. Theor. Biol.*, 261(4):511–530, 2009.

[137] P. E. M. Purnick and R. Weiss. The second wave of synthetic biology: From modules to systems. *Nat. Rev. Mol. Cell Biol.*, 10(6):410–422, 2009.

[138] J. M. Raser and E. K. O'Shea. Noise in gene expression: Origins, consequences, and control. *Science*, 309(5743):2010–2013, 2005.

[139] R. Roberts, T. Neupane, L. Buecherl, C. J. Myers, and Z. Zhang. STAMINA 2.0: Improving Scalability of Infinite-State Stochastic Model Checking. In B. Finkbeiner and T. Wies, editors, *Verification, model checking, and abstract interpretation*, pages 319–

331, Springer International Publishing, Cham, Switerzland, 2022.

[140] N. Roehner, J. Beal, K. Clancy, B. Bartley, G. Misirli, R. Grünberg, E. Oberortner, M. Pocock, M. Bissell, C. Madsen, T. Nguyen, M. Zhang, Z. Zhang, Z. Zundel, D. Densmore, J. H. Gennari, A. Wipat, H. M. Sauro, and C. J. Myers. Sharing structure and function in biological design with SBOL 2.0. *ACS Synth. Biol.*, 5(6):498–506, 2016.

[141] N. Roehner and C. J. Myers. Directed acyclic graph-based technology mapping of genetic circuit models. *ACS Synth. Biol.*, 3(8):543–555, 2014.

[142] N. Roehner, Z. Zhang, T. Nguyen, and C. J. Myers. Generating systems biology markup language models from the synthetic biology open language. *ACS Synth. Biol.*, 4(8):873–879, 2015.

[143] S. Rollié, M. Mangold, and K. Sundmacher. Designing biological systems: Systems engineering meets synthetic biology. *Chem. Eng. Sci.*, 69(1):1–29, 2012.

[144] A. Sanchez, S. Choubey, and J. Kondev. Stochastic models of transcription: From single molecules to single cells. *Methods*, 62(1):13–25, 2013.

[145] M. Santillán. On the use of the Hill functions in mathematical models of gene regulatory networks. *Math. Model. Nat. Phenom.*, 3(2):85–97, 2008.

[146] T. Schladt, N. Engelmann, E. Kubaczka, C. Hochberger, and H. Koeppl. Automated design of robust genetic circuits: Structural variants and parameter uncertainty. *ACS Synth. Biol.*, 10(12):3316–3329, 2021.

[147] T. Schlitt. Approaches to modeling gene regulatory networks: A gentle introduction. *Methods Mol. Biol.*, 1021:13–35, 2013.

[148] T. Schlitt and A. Brazma. Current approaches to gene regulatory network modelling. *BMC Bioinf.*, 8(6):S9, 2007.

[149] M. Scott, B. Ingalls, and M. Kærn. Estimations of intrinsic and extrinsic noise in models of nonlinear genetic networks. *Chaos*, 16(2):026107, 2006.

[150] K. E. Shearwin, B. P. Callen, and J. B. Egan. Transcriptional interference – A crash course. *Trends Genet.*, 21(6):339–345, 2005.

[151] R. P. Shetty, D. Endy, and T. F. Knight. Engineering BioBrick vectors from BioBrick parts. *J. Biol. Eng.*, 2(1):5, 2008.

[152] J. Shin, S. Zhang, B. S. Der, A. A. Nielsen, and C. A. Voigt. Programming *Escherichia coli* to function as a digital display. *Mol. Syst. Biol.*, 16(3):e9401, 2020.

[153] A. Singh and M. Soltani. Quantifying intrinsic and extrinsic variability in stochastic gene expression models. *PLoS One*, 8(12):e84301, 2013.

[154] A. L. Slusarczyk, A. Lin, and R. Weiss. Foundations for the design and implementation of synthetic genetic circuits. *Nat. Rev. Genet.*, 13(6):406–420, 2012.

[155] K. Sneppen, I. B. Dodd, K. E. Shearwin, A. C. Palmer, R. A. Schubert, B. P. Callen, and J. B. Egan. A mathematical model for transcriptional interference by RNA polymerase traffic in *Escherichia coli*. *J. Mol. Biol.*, 346(2):399–409, 2005.

[156] D. Sprinzak and M. B. Elowitz. Reconstruction of genetic circuits. *Nature*, 438(7067):443, 2005.

[157] B. C. Stanton, A. A. K. Nielsen, A. Tamsir, K. Clancy, T. Peterson, and C. A. Voigt. Genomic mining of prokaryotic repressors for orthogonal logic gates. *Nat. Chem. Biol.*, 10(2):99–105, 2014.

[158] P. S. Swain, M. B. Elowitz, and E. D. Siggia. Intrinsic and extrinsic contributions to stochasticity in gene expression. *PNAS*, 99(20):12795–12800, 2002.

[159] A. Tamsir, J. J. Tabor, and C. A. Voigt. Robust multicellular computing using genetically encoded NOR gates and chemical 'wires'. *Nature*, 469(7329):212–215, 2011.

[160] M. Thattai and A. van Oudenaarden. Intrinsic noise in gene regulatory networks. *PNAS*, 98(15):8614–8619, 2001.

[161] J. J. Tyson, K. C. Chen, and B. Novak. Sniffers, buzzers, toggles and blinkers: Dynamics of regulatory and signaling pathways in the cell. *Curr. Opin. Cell Biol.*, 15(2):221–231, 2003.

[162] J. L. Villanueva-Ca ñas, E. Gonzalez-Roca, A. G. Unanue, E. Titos, M. J. M. Yoldi, A. V. Gómez, and J. A. P. Butillé. ROBOCOV: An affordable open-source robotic platform for SARS-CoV-2 testing by RT-qPCR, bioRxiv: 2020.06.11.140285, June 2020.

[163] C. A. Voigt. Genetic parts to program bacteria. *Curr. Opin. Biotech.*, 17(5):548–557, 2006.

[164] D. I. Walsh III, M. Pavan, L. Ortiz, S. Wick, J. Bobrow, N. J. Guido, S. Leinicke, D. Fu, S. Pandit, L. Qin, et al. Standardizing automated DNA assembly: Best practices, metrics, and protocols using robots. *SLAS Tecnol.*, 24(3):282–290, 2019.

[165] D. Waltemath, R. Adams, F. T. Bergmann, M. Hucka, F. Kolpakov, A. K. Miller, I. I. Moraru, D. Nickerson, S. Sahle, J. L. Snoep, and N. Le Novère. Reproducible computational biology experiments with SED-ML - The simulation experiment description markup language. *BMC Syst. Biol.*, 5(1):198, 2011.

[166] D. Waltemath, R. Henkel, F. Winter, and O. Wolkenhauer. Reproducibility of model-based results in systems biology. In A. Prokop and B. Csukás, editors, *Systems biology: Integrative biology and simulation tools*, pages 301–320. Springer Netherlands, Dordrecht, 2013.

[167] B. Wang, R. I. Kitney, N. Joly, and M. Buck. Engineering modular and orthogonal genetic logic gates for robust digital-like synthetic biology. *Nat. Commun.*, 2:508, 2011.

[168] Z. Wang, M. Gerstein, and M. Snyder. RNA-Seq: A revolutionary tool for transcriptomics. *Nat. Rev. Genet.*, 10(1):57–63, 2009.

[169] L. Watanabe, T. Nguyen, M. Zhang, Z. Zundel, Z. Zhang, C. Madsen, N. Roehner, and C. Myers. iBioSim 3: A tool for model-based genetic circuit design. *ACS Synth. Biol.*, 8(7):1560–1563, 2018.

[170] Y. Xiang, N. Dalchau, and B. Wang. Scaling up genetic circuit design for cellular computing: Advances and prospects. *Nat. Comput.*, 17(4):833–853, 2018.

[171] E. Yeung, J. Kim, Y. Yuan, J. Gonçalves, and R. M. Murray. Data-driven network models for genetic circuits from time-series data with incomplete measurements. *J. R. Soc. Interface*, 18(182):20210413, 2021.

[172] E. Yeung, S. Kundu, and N. Hodas. Learning deep neural network representations for Koopman operators of nonlinear dynamical systems. In *2019 Am. Cont. Conf.*, ACC '19, pages 4832–4839, Philadelphia, PA, 2019. IEEE.

[173] R. A. Young and J. A. Steitz. Tandem promoters direct *E. coli* ribosomal RNA synthesis. *Cell*, 17(1):225–234, 1979.

[174] M. Zhang, J. A. McLaughlin, A. Wipat, and C. J. Myers. SBOLDesigner 2: An intuitive tool for structural genetic design. *ACS Synth. Biol.*, 6(7):1150–1160, 2017.

[175] S. Zilberzwige-Tal, P. Fontanarrosa, D. Bychenko, Y. Dorfan, E. Gazit, and C. J. Myers. Investigating and modeling the factors that affect genetic circuit performance, bioRxiv: 2022.05.16.492150, May 2022.