# Design of Asynchronous Genetic Circuits

Tramy Nguyen, *Member, IEEE,* Timothy S. Jones, Pedro Fontanarrosa, Jeanet V. Mante, Zach Zundel, Douglas Densmore, *Member, IEEE,* Chris J. Myers, *Fellow, IEEE*

***Abstract*—Most digital electronic circuits utilize a timing reference to synchronize the progression of signals and enable sequential memory elements. These designs may not be realizable in biological substrates due to the lack of a reliable high frequency clock signal. Asynchronous designs eliminate the need for a clock with data encodings and request/acknowledge handshake protocols. This paper proposes a workflow to automate the design of asynchronous genetic circuits. This workflow extends genetic design tools by leveraging asynchronous logic design methods customized for this technology. This workflow is demonstrated on a genetic sensor that uses filtering and cellular communication to improve its reliability.**

***Index Terms*—Asynchronous design, genetic circuits, genetic design automation, standards, synthetic biology, verification**

## I. Introduction

Synthetic biology researchers are exploring the potential of programming cells to perform useful functions. In particular, they are designing genetic circuits to produce useful biochemical and pharmaceutical products [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], to help cure genetic diseases and create therapeutic bacterial agents [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], and to produce plants that can sense and adapt to a wider range of environments [28], [29], [30], [31], [32], [33], [34].

Such circuits are often created from biological components that mimic the behavior of Boolean logic gates. Most genetic circuits that have been built are *combinational circuits* (i.e. the input signals map directly to the output signal). *Sequential circuits*, on the other hand, have internal state allowing them to map a sequence of input signals to a sequence of output signals. While several genetic memory circuits have been created [35], [36], [37], [38], [39], general methodologies for genetic sequential circuit design have not been developed.

In a sequential circuit, the internal state can be updated using either *synchronous* or *asynchronous* timing. Synchronous designs use a global clock signal to produce a fixed time schedule for updating the state. Asynchronous designs, on the other hand, determine when to update their state using data encodings and request/acknowledge handshaking protocols. Given the difficulties to create a precise timing reference using biological components and the fact that most biological systems are responsive immediately to changes in environmental conditions, it seems likely that synthetic genetic circuits should follow the asynchronous design paradigm.

Nature also makes uses of asynchronous timing. One such example of an asynchronous sequential circuit that occurs in nature is the filtering system utilized by the venus flytrap (*Dionaea muscipula*). It shows different behaviors depending on how often prey touches trigger hairs within a period of time (the circuit is asynchronous as the time element is simply due to molecular decay). At least two trigger events are required to close the trap, and at least three to start producing digestion enzymes [40]. It has been suggested that the memory of the system works by increasing cytosolic calcium levels in a quantized manner correlating with the number of triggers that have been seen [41]. Thus, here the calcium levels are proposed to act as state memory in an asynchronous sequential network.

In order to design complex asynchronous genetic circuits, an automated workflow is required. This workflow begins with a specification and yields a collection of genetic logic and memory gates to produce the desired function. While *genetic design automation* (GDA) tools exist for the design and technology mapping of combinational circuits, such as Cello [42] and iBioSim [43], [44], there are not currently any GDA tools that support sequential genetic circuits. The only biological design automation tool that we are aware of that targets sequential circuits is the one developed in [45]. Rather than genetic circuits, this work targets *chemical reaction networks* (CRN), which potentially can be mapped to *DNA strand displacement* circuits [46] that compute using the structural rather than the functional properties of DNA. In particular, these circuits are composed of bi-molecular reaction motifs that are capable of modeling asynchronous logic circuits.

This paper describes a new automated workflow for the design of asynchronous sequential genetic circuits. This workflow builds upon existing tools for asynchronous logic design, namely ATACS [47], and links them to genetic technology mapping tools, namely Cello and iBioSim. A key feature is a new compiler that can translate to/from Verilog to other standard formats required by the design tools. This workflow is demonstrated on a genetic sensor that uses filtering and cellular communication to improve its reliability.

This paper is organized as follows. Section II provides a brief review of asynchronous circuits, genetic circuits, data standards, and existing GDA tools. Section III presents our proposed workflow for asynchronous genetic circuit design. Section IV describes a genetic sensor case study. Finally, Section V presents our conclusions and future work.

T. Nguyen and C. Myers are with the Department of Electrical and Computer Engineering, University of Utah, Salt Lake City, UT 84112 USA e-mail: myers@ece.utah.edu.

T. Jones and D. Densmore are with the Department of Electrical and Computer Engineering, Boston University, Boston, MA 02215 USA.

P. Fontanarrosa, J. Mante, and Z. Zundel are with the Department of Bioengineering, University of Utah, Salt Lake City, UT 84112 USA.

Manuscript received MONTH DAY, YEAR; revised MONTH DAY, YEAR.

```
module sync_sensor(Clock, Sensor, Actuator);
    input Clock, Sensor;
    output reg Actuator;
    initial begin
        Actuator = 1'b0;
    end
    always @(posedge Clock) begin
        Actuator = Sensor;
    end
endmodule
```

(a)

```
module async_sensor(Start, Sensor, Actuator);
    input wire Start, Sensor;
    output reg Actuator;
    initial begin
        Actuator = 1'b0;
    end
    always begin
        wait (Start == 1'b1 && Sensor == 1'b1);
        #5 Actuator = 1'b1;
        wait (Sensor == 1'b0);
        #5 Actuator = 1'b0;
    end
endmodule
```

(b)

Fig. 1. Verilog examples for a simple sensor. (a) Synchronous sensor that samples the *Sensor* input on each positive edge of the *Clock* signal. (b) Asynchronous sensor that activates the *Actuator* after receiving both the *Start* and *Sensor* input signals and resets immediately when *Sensor* goes low.

## II. Background

This section provides some brief background on asynchronous circuits, genetic circuits, data standards utilized by our workflow, and existing GDA tools.

### A. Asynchronous Circuits

Sequential circuits can be designed synchronously or asynchronously. Synchronous circuits operate on a global timing reference, called a clock. The internal states of the system are updated based on a fixed-time schedule. A synchronous design of a sensor can be described using the Verilog language as shown in Figure 1(a). This circuit samples the Sensor input when the clock signal transitions from LOW to HIGH. The key requirement for synchronous design to operate correctly is that the period of this clock signal must be long enough that sufficient time is provided for the logic to respond to each input change and come to a new stable value before the clock signal changes again. Otherwise, there are *synchronization problems* and the circuit fails to operate correctly.

Glitches on synchronous circuits are not a major concern as long as the signal is stable before or after the clock signal is sampled to produce a valid result. However, because synchronous circuits rely on this global clock, the clock rate is determined based on the critical path (i.e., longest path from inputs to an output). As a result, synchronous circuits must operate at worst-case timing to ensure correctness.

An alternative means of sequential circuit design uses the asynchronous timing paradigm. In asynchronous design, the timing of state changes is handled using data encoding and request/acknowledge handshake protocols. A simple protocol is the *four-phase handshaking protocol*. In this protocol, a *request* signal starts the operation when it goes from a low to a high value. When the operation is complete, an *acknowledge* signal confirms this by going from a low to a high value. The protocol then resets to prepare for the next operation by setting the *request* signal low followed by the *acknowledge* signal going low. Using a similar protocol, a simple asynchronous sensor is described using the Verilog language in Figure 1(b). This circuit waits for the *Start* and *Sensor* inputs to go high (i.e., a request), then it sets the *Actuator* output to high (i.e., an acknowledgement). Finally, the *Sensor* signal going low immediately results in the *Actuator* going low. The advantage of this asynchronous timing paradigm is that no fixed timing reference must be generated, and the circuit is free to respond to changes in the *Sensor* signal that occur at anytime, without risk of a synchronization problem.

### B. Genetic Circuits

In biology, information is generally stored in DNA (*deoxyribonucleic acid*). The central dogma of biology is that DNA is used to create RNA (*ribonucleic acid*) through a process called *transcription*. The RNA is in turn used to create proteins though a process called *translation*. Some of the proteins created, called *transcription factors*, can regulate (i.e., *activate* or *repress*) further protein production. The sequence which is directly used to create a protein is called the *coding sequence* (CDS). Not all of a genetic sequence is directly used in the creation of proteins. For example, there are sequence regions called *promoters* where transcription is initiated and which include binding sites for transcription factors allowing them to regulate the speed of transcription. *Ribosome binding sites* are sequences where *ribosomes* can bind to the RNA to initiate translation. Finally, there are *terminators*, which are regions of the DNA that indicate where transcription should stop.

As an example, consider the *genetic toggle switch* circuit shown in Figure 2(a), which was one of the two seminal synthetic biology circuits designs published in 2000 [35] (the other was the *repressilator* [48]). The genetic toggle switch shown in this example is composed of two transcriptional units; one that produces LacI and another that produces TetR and GFP (*green fluorescent protein*). The GFP protein serves as an output reporter to indicate when the TetR protein is produced. The output of each of these transcriptional units represses the production of the other transcriptional unit. TetR is a protein that can bind to pTet, inhibiting the production of LacI. Similarly, LacI can bind to the promoter pLac, inhibiting the production of TetR. TetR and LacI can bind to small molecules (aTc[1] and IPTG[2], respectively) to form complexes that are unable to bind to promoters. This means that when aTc is present, it acts as an inhibitor of TetR and therefore, TetR can no longer repress the production of LacI. Similarly, when IPTG is present, it binds and inhibits LacI, so that LacI is not able to repress the production of TetR. If neither IPTG or aTc are present, then the genetic toggle switch holds its state and continues to produce the TetR if IPTG was the last molecule present or LacI if aTc was the last molecule present. In other words, this circuit implements a simple sequential memory

[1] Anhydrotetracycline

[2] Isopropyl $\beta$-D-1-thiogalactopyranoside

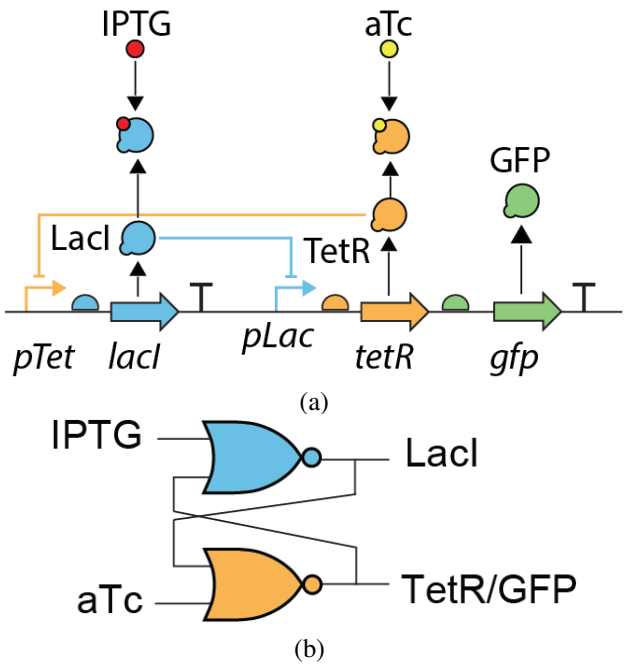



Fig. 2. The genetic toggle switch. (a) SBOL Visual 2 compliant representation of the genetic toggle switch. The promotors (bent arrows) indicate areas where transcription starts. They can be repressed by the binding of transcription factors (indicated with flat headed arrows), which prevents the expression of downstream proteins. The ribosome binding sites (half circles) indicate where ribosomes bind to transcripts to start reading the RNA sequence during translation. The coding sequence (fat arrow) indicates the region of DNA directly read during translation to create proteins (rounded blobs). The terminators (T) indicate where a transcriptional unit ends. The diagram shows that LacI represses the blue promoter (pLac) preventing TetR and GFP expression (so the cells do not fluoresce). However, the addition of IPTG leads to LacI forming a complex with the IPTG molecule preventing it from repressing the blue promoter (pLac) thus allowing TetR and GFP production (the cell fluoresces green). TetR represses the yellow promoter (pTet) preventing the production of LacI. Upon the addition of aTC, TetR forms a complex with aTc lifting the repression on the yellow promoter (pTet) allowing the production of LacI which represses TetR production. Thus, IPTG or aTc are required to change state, but once a state is achieved, the circuit maintains that state. (b) The equivalent electronic circuit diagram composed of cross-coupled NOR gates forming a set-reset latch.

circuit, which exhibits a bistable behavior. It retains its state until it detects the addition of a small molecule inducer (IPTG or aTc). Therefore, the behavior of the genetic toggle switch is essentially that of a set-reset latch created by cross-coupled NOR gates as shown in Figure 2(b).

### C. Data Standards

In order to design software applications that can automate the process of designing a genetic circuit and retrieving information about biological parts, this information must be encoded using data standards. Normally, a data standard is depicted in a format that has a core representation and agreed terminologies to represent the data of interest. Data standards play a big role in this workflow as they enable the transfer of data between different stages of our workflow. In particular, our workflow uses two biological data standards, the *Synthetic Biology Open Language* (SBOL) [49], [50] and the *Systems Biology Markup Language* (SBML) [51].

SBOL is a data standard that is used to represent genetic designs *in silico* (i.e. on a computer). This standard allows the specification of the structure of the design, such as the DNA sequences for the DNA components (e.g. pTet and pLac), protein sequences for the protein components (e.g. LacI and TetR), and chemical makeup of the small molecules (e.g. IPTG and aTc). SBOL can also define the qualitative behavior of a genetic design. For example, it can encode the interactions between the components, such as the repression of the pTet promoter by the TetR protein and that the lacI CDS encodes the LacI protein. This standard is used within our workflow to represent the essential structure and function of our genetic circuits. Furthermore, many biological parts have been encoded into this format and are publicly available to view and use from online repositories, such as SynBioHub [52]. Finally, another complimentary standard, SBOL Visual, provides a means for visualizing a genetic design [53], such as in Figure 2(a).

SBML is a data standard used to describe computational models of biological processes and pathways. The SBML data model compliments the SBOL data model in that it allows us to verify the behavior of our design. Different than SBOL, SBML has agreed terminologies and rules that define the quantitative aspect of a biological system. Genetic designs translated into SBML can express information about biological entities as *chemical species*. These species can represent both DNA and non-DNA components described in SBOL. SBML represents the processes that create, modify, or destroy these species as *chemical reactions*. These reactions quantitatively model the interactions expressed in SBOL.

In the past, we have developed converters between SBOL and SBML to navigate between these two data standards [54], [55]. In particular, our workflow utilizes the SBOL to SBML converter to create a dynamic model of our design to allow for verification through simulation.

### D. Genetic Design Automation

The challenge with any design process is to efficiently explore the space of possible designs to find one that functions correctly and performs the best for some given criteria. Similar to how *electronic design automation* (EDA) evolved to address this challenge, researchers are creating GDA tools with this goal in mind. Some of the added challenges that these GDA tools must address for genetic logic devices include *crosstalk*, *signal mismatch*, *roadblocking*, and *genetic context effects* [56]. As shown in Figure 3(a), crosstalk occurs in genetic circuits when the product of one gate has unintended interactions with another gate. In design, this is avoided by requiring that a particular signal carrier (transcription factor) is only assigned to one node in the circuit. Unlike electrical circuits, the physical output of each gate is unique. As shown in Figure 3(b), signal mismatch occurs when the level of product produced by one gate does not meet the threshold necessary to cause the correct response in another downstream gate. The problem of finding a design that implements a specified function can be defined as finding a set of biological gates such that the switching threshold of any gate lies within the output range of the upstream gate. As shown in Figure 3(c), roadblocking can

occur in gates that use tandem promoters, and the downstream promoter interferes in transcription initiated at the upstream promoter [42]. Roadblocking is avoided by finding all such instances of promoter interference during the gate library characterization, and eliminating these problematic permutations from the ensemble of DNA sequences that are considered to fulfill the specification. Finally, as shown in Figure 3(d), genetic context effects occur when the ordering of neighboring components changes the behavior of the components. In such cases where context effects are known to be strong, gates may be integrated into the host genome at predetermined locations where each gate's transcription rate has already been characterized.

Given that each operation in a Boolean function must receive a unique biological implementation and that the ordering of the biological gates or their sub-components may be constrained, a GDA tool chooses a particular assignment and ordering of biological gates from a combinatorial space. For example, implementing the AND function as NOR(NOT(p),NOT(q)) in a library of 5 gates, there are $5 \cdot 4 \cdot 3$ choices of unique gate assignments, and 3! choices of the ordering of the choice of gates in a DNA sequence. Within the NOR gate itself, the inputs may also be permuted in two ways in the sequence. Some of these assignments or orderings will be eliminated according to the constraints described above.

Some of the GDA tools that have been developed to search the design space while addressing these genetic constraints are shown in Table I. Each of these tools begins with a specification of the desired behavior. In MatchMaker [57], SBROME [58], and iBioSim [43], this specification is given in the form of a graphical representation of an abstract *genetic regulatory network* (GRN). GeneTech [59] allows the behavior to be specified using a set of Boolean expressions. Cello [42] supports a subset of the Verilog *hardware description language* (HDL) for specification. The second input to each of these tools is a library of genetic logic gates. The SBROME tool begins with simply a set of genetic parts, while the Cello and GeneTech tools require a library of NOT and NOR gates. MatchMaker and iBioSim support a library composed of any arbitrary logic gates. The goal of each of these GDA tools is to select from the library a collection of elements that when composed together produce the specified behavior. Furthermore, the tool should also attempt to address the genetic design constraints to increase the likelihood that the selected design actually performs as desired after it is constructed. Most of these tools at a minimum address the signal crosstalk issue, but Cello is the only tool that attempts to address three of the four identified genetic constraints. Additionally, we would like to highlight the data standards that the tools support. Unfortunately, SBROME and GeneTech do not support standard representations for their genetic designs, making them difficult to connect into a complete workflow. MatchMaker supports only the out-of-date SBOL1 standard. iBioSim and Cello support SBOL2, the latest version, and iBioSim also supports SBML for computational models that can be used to validate the produced designs. It should be mentioned that all of these GDA tools are currently restricted to produce combinational circuits only.

Our workflow extends the Cello and iBioSim tools to support asynchronous sequential circuits. In this initial workflow, the Cello tool is leveraged for mapping the next state and output logic to produce an implementation that satisfies all of the genetic design constraints. The iBioSim tool is utilized to produce dynamic computational models encoded in SBML to validate the designs produced.

## III. Methodology

The proposed workflow for asynchronous genetic circuit design is shown in Figure 4. The workflow begins with a high-level specification encoded using the Verilog language. Verilog was chosen due to its ubiquity in existing electrical circuit design workflows. There are many tools that exist to validate and simulate Verilog designs, which can be readily integrated into this workflow. This high-level specification is then compiled to a *labeled Petri net* (LPN) [60] following a fairly direct syntax-directed translation. This LPN can then be simulated to check behavior using the iBioSim software [61]. At this point, the asynchronous synthesis tool ATACS is used to produce logic equations [47], which can also be expressed in the Verilog language. This synthesis process is described in Section III-A. This workflow requires a library of characterized parts described in SBOL [50] and made available via the SynBioHub repository [52]. These parts are composed to form the genetic logic gates that are used in the composite design. This library is described in Section III-B. Next, the process of *technology mapping* applies a matching and covering procedure to select parts from the library to implement the genetic design while addressing the genetic design constraints. The technology mapping step can either use the graph-based covering approach described in [62] or Cello's simulated annealing algorithm described in [42]. The technology mapping step is described in Section III-C. Finally, the resulting design can then be converted from SBOL to a computational model expressed in SBML using the model generation procedure described in [63], [64]. The resulting simulation of the design is then compared to the simulation of the specification to verify that the circuit behavior is as desired. This verification procedure is described in Section III-D.

### A. Synthesis

The first step in our workflow is the creation of a behavioral specification for the genetic design. This specification describes the behavior of the desired circuit independent of the specific biological parts that are needed to build the circuit. Similar to the Cello tool, our specification is also constructed using the Verilog language. In order to express asynchronous behavior, our workflow makes use of blocking assignments and wait statements to describe handshaking protocols. In the example shown in Figure 4, the toggle switch must wait for IPTG to be applied, which changes the toggle switch to the high state and produces GFP. Next, it must wait for aTc to be applied, which changes the toggle switch to the low state and stops the production of GFP.

In addition to the circuit specification, our workflow also requires a testbench (not shown) that describes the behavior

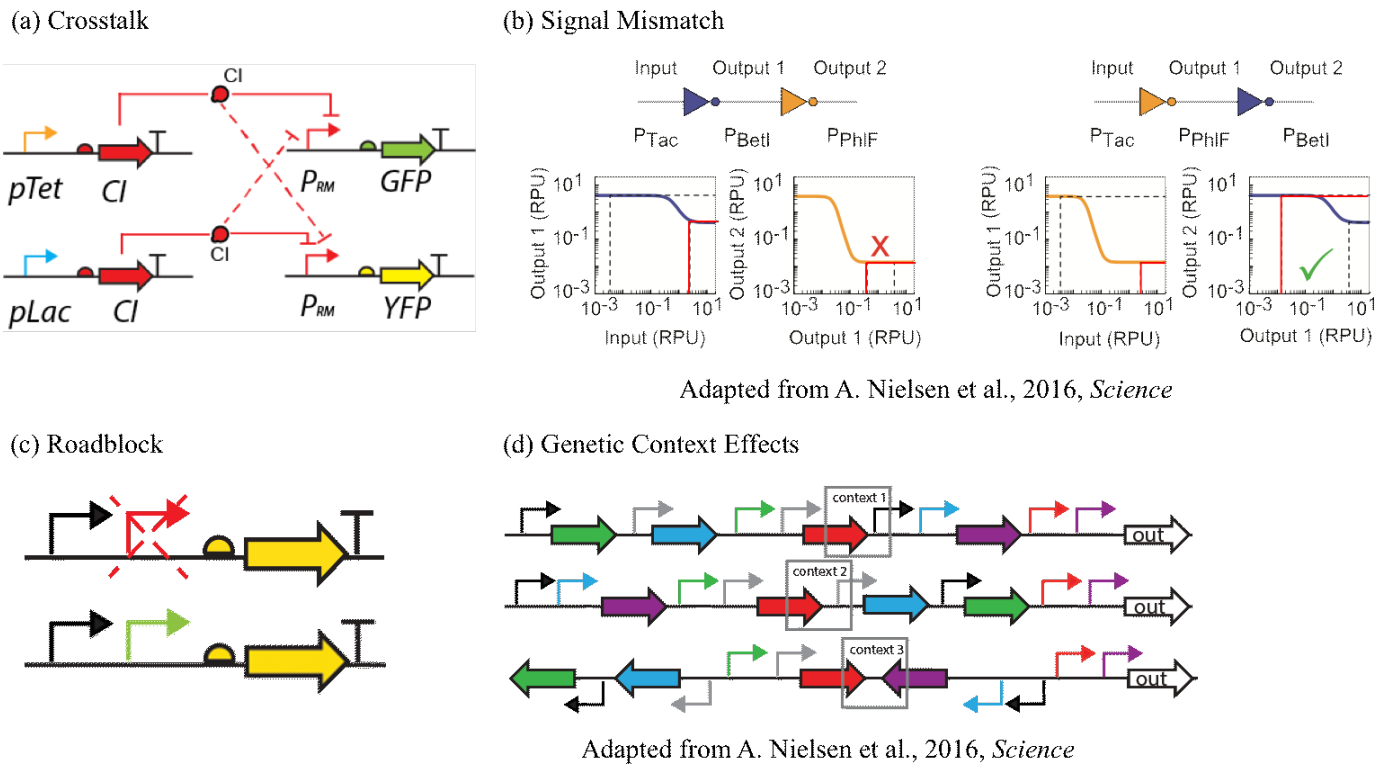



Fig. 3. Four genetic constraints that occur in genetic circuits. (a) Crosstalk: interference of circuit components with each other or the host circuitry. (b) Signal mismatch: incompatible signal levels of gates composed in series. (c) Roadblock: initiation of transcription from the upstream promoter in a tandem promoter is impeded by the presence of a transcription factor bound to the downstream promoter. (d) Genetic context effects: the same circuit can act differently based on the ordering of neighboring components.

TABLE I
THIS TABLE SUMMARIZES THE DIFFERENT EXISTING TECHNOLOGY MAPPING TOOLS. IT SHOWS THE TOOL'S INPUT SPECIFICATION LANGUAGE, STANDARDS SUPPORTED, THE TYPES OF LIBRARY PARTS USED, AND THE GENETIC CONSTRAINTS ADDRESSED.

| | MatchMaker [57] | SBROME [58] | iBioSim [43] | Cello [42] | GeneTech [59] |
|---|---|---|---|---|---|
| Specification language | Abstract GRN | Abstract GRN | Abstract GRN | Verilog | Boolean expressions |
| Library | Arbitrary logic gates | Parts | Arbitrary logic gates | NOT and NOR gates | NOT and NOR gates |
| Signal Mismatch | ✓ | ✗ | ✗ | ✓ | ✗ |
| Crosstalk | ✗ | ✓ | ✓ | ✓ | ✓ |
| Context Effects | ✗ | ✗ | ✗ | ✗ | ✗ |
| Roadblock | ✗ | ✗ | ✗ | ✓ | ✗ |
| Standard Support | SBOL1 | ✗ | SBOL2 and SBML | SBOL2 | ✗ |

of the environment. The environment must provide IPTG, wait for GFP to go high, remove IPTG, add aTc, wait for GFP to go low, then remove aTc, and repeat. This series of actions exercises the circuit through the normal steps of operation. Our synthesis procedure requires that both the circuit and the environment are fully specified.

Once the specification and the testbench have been tested for the desired functionality using a Verilog simulator, our workflow uses our Verilog compiler to produce equivalent SBML models, which can also be simulated to validate functionality. These SBML models can then flattened and complied into the LPN model, which is required by the asynchronous logic synthesis tool, ATACS [47], that our workflow uses.

The goal of logic synthesis is to derive Boolean logic equations that implement the specified behavior. ATACS creates a state graph from the given LPN model. Next, the state graph is analyzed to determine the logic necessary to implement any state and output signals. A key challenge with asynchronous logic design is the avoidance of *logic hazards*. A logic hazard occurs if a signal that is supposed to remain stable, momentarily changes value, or a signal that is supposed to change, does so non-monotonically [65]. Unlike electronic circuit designs where the cost of errors is circuit failure, in genetic circuits, errors can be tolerated. Since genetic circuits are deployed in a population of cells, there are multiple copies of a genetic circuit available. Therefore, if one genetic circuit fails within a cell due to glitches, it may not highly affect the entire system if other cells exhibit the correct behavior. Nikolaev et al. [66] has shown from modeling a genetic toggle switch that a large population of cells performing the same task will help average out the expected behavior for the majority of the genetic toggle switch circuits. While it is important to

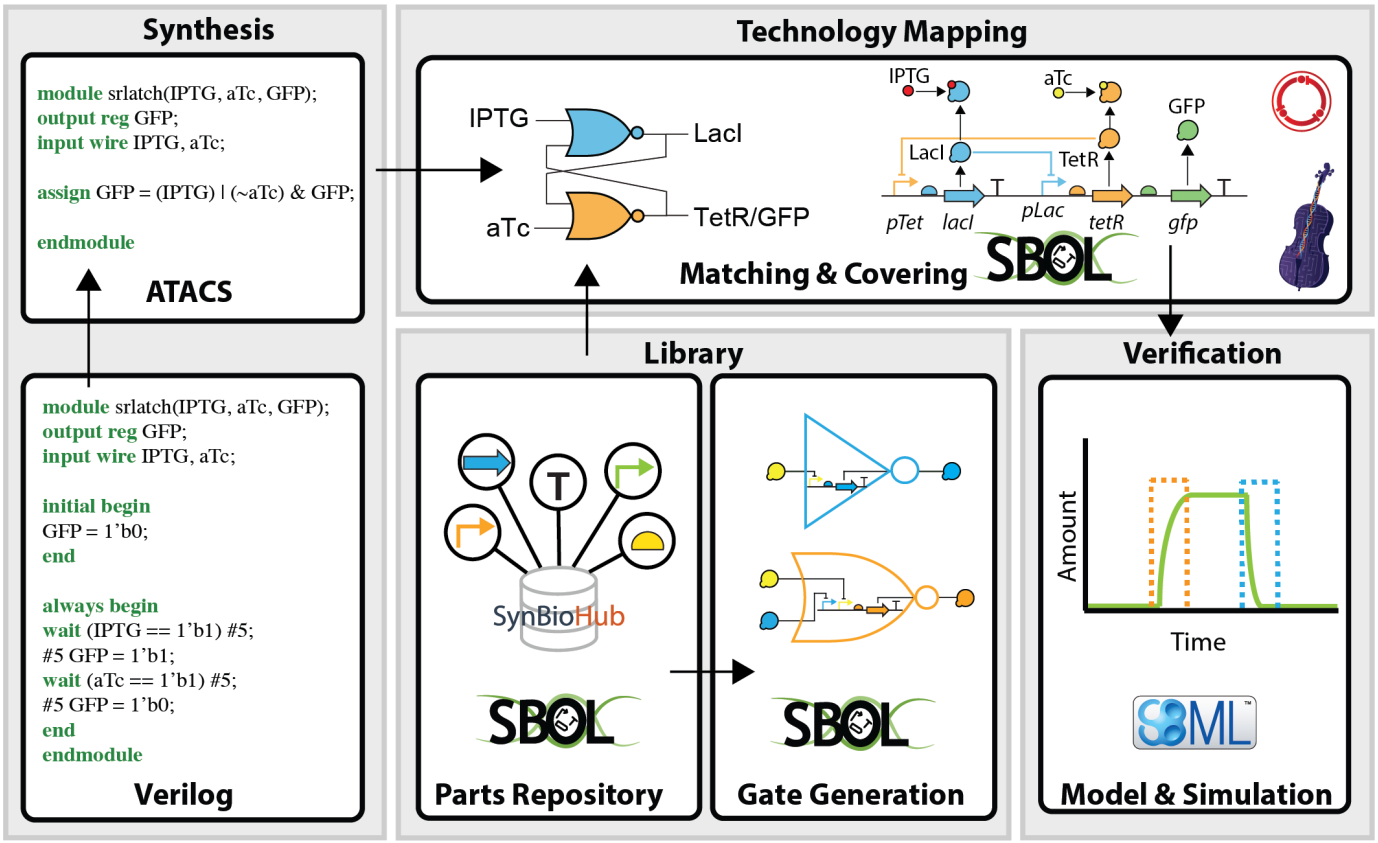



Fig. 4. The proposed workflow illustrated using the genetic toggle switch design. First, the asynchronous design is specified using behavioral Verilog. Next, the ATACS asynchronous design tool synthesizes logic equations represented using structural Verilog. The synthesized circuit is then realized as a physical design by a technology mapping procedure that selects gates from a SBOL encoded gate library stored in a SynBioHub repository. In order to verify the workflow, a model is generated and simulations are performed to verify that the design displays the expected behavior.

address high probability hazards, genetic circuits can likely tolerate low probability hazards. Exploiting this observation is an area of future research that we plan to explore.

Synthesis produces Boolean logic functions described in a structural Verilog representation. This representation contains assign statements expressed in the form shown below:

$$\text{Output} \quad = \quad \text{Set} \mid (\overline{\text{Reset}} \ \& \ \text{Output})$$

In the example shown in Figure 4, for the output GFP, the Set function is IPTG, and the Reset function is aTc. It should be noted that this workflow produces expressions that use feedback to produce state. Many well known genetic circuits have been built using negative feedback and have been shown to exhibit bistablity that results in cellular memory capable of holding state [35], [36], [37], [38].

### B. Library

A genetic design workflow requires a collection of well-characterized parts, and GDA tools require programmatic access to such part libraries to produce circuit designs. In particular, part and device model parameters, sequence information, and part placement constraints must all be encoded in a machine-readable library. In the case of the Cello CAD software [42], [67], the entire library is stored as a "user constraints file" (UCF) in the JSON format.

Different tools may have different design strategies, but if a genetic circuit design is to be fabricated and placed in a host organism, the complete DNA sequence of every part must exist in the library. If parts and devices are integrated into the cell together on a plasmid, they may have constraints as to their placement on the plasmid. For example, in a transcriptional unit engineered with tandem promoters, one promoter when repressed may be more likely to impede transcription beginning at the upstream promoter (i.e., roadblocking), inhibiting the promoters' ability to act independently, e.g. to exhibit NOR logic, and indicating that the two promoters should not be placed in this configuration. Certain genetic devices may be more demanding on the host organism than others, and a library may encode optical density measurements for a host inoculated with every device in the library, one by one, as a measure of the load on the host. Basic structural information about what parts (promoters, coding sequences, etc.) comprise the functional devices in the circuit are also part of a design library. Finally, as design tools employ some biological model to compose devices into a circuit that realizes a given function, a library stores parameterizations of this model for each of the devices. Cello, for example, stores Hill function parameters for every gate in the library. The parameters are fit to dose-response data that has been transformed into a common unit (i.e., *relative promoter units*

(RPUs)) to model signal propagation from one gate to another.

For libraries to be shared and even utilized by different tools, their format and storage are also important to consider. The SynBioHub repository is a storage solution not only for design libraries, but also for assembly plans for circuits, networks of interactions in circuits, and other information [52]. All data stored in a SynBioHub instance exists in the SBOL format, and thus anything that can be represented in SBOL can be stored in SynBioHub. Furthermore, any data that is not readily expressed in SBOL can be attached as supplemental files. The library of TetR homolog based NOR gates in *E. coli* that were originally distributed with Cello was translated to SBOL and exists in the *National Science Foundation* (NSF) *Living Computing Program* (LCP) instance of SynBioHub (https://synbiohub.programmingbiology.org). SynBioHub exposes an HTTP REST API for programmatic access by design tools, allowing libraries to be retrieved and used on demand.

### C. Technology Mapping

As shown in Fig. 4, the logic equations expressed in structural Verilog must be matched to genetic gates from the gate library using technology mapping. Our workflow can use either Cello [42] or iBioSim [43] to perform technology mapping.

Cello takes the synthesized Verilog file produced from ATACS and decomposes the expressions into NOT and 2-input NOR gates. Next, the decomposed expression is mapped directly to a library of gates that is limited to NOT and NOR gates. During this matching step, the characterization data stored within the library is used by Cello to solve the genetic constraints to increase the likelihood that the design operates as desired when constructed in the wet lab. Once all Verilog expressions have been mapped, the physical design can be exported into the SBOL data format and verified using the model generation procedure described in [68] and also the next section.

iBioSim, on the other hand, uses the Verilog compiler that was built for this workflow to parse and perform decomposition on the Verilog expressions. The decomposed network is represented as a *genetic regulatory network* (GRN) using SBOL components and interactions. This GRN is equivalent to the NOT and 2-input NOR decomposition. iBioSim uses a matching and covering procedure that allows for multiple nodes to be covered by a single gate, allowing for arbitrary logic gates to be selected for the final implementation [54]. Like Cello, the final design is also produced in the SBOL data format, and it can be verified using the same model generation procedure [68].

### D. Verification

The workflow described in this paper progresses through several procedures to translate a high-level description to a physical design composed of biological parts. The description of the design uses different forms and formats to perform each task. To ensure that the behavior that was initially described in the design specification stays the same throughout the data format conversions, verification must be performed at each stage of the workflow as shown in Figure 5. The left-hand-side of the figure shows the different modeling formats used throughout the workflow, which include behavioral Verilog, LPNs, structural Verilog, SBOL, and SBML. The right-hand-side of the figure shows the simulation results at each step of the workflow, produced by both Verilog simulators and the iBioSim simulators. The outcome of this verification process is to ensure that the behavior that is observed in the simulation is the same in each step of the workflow. If the behavior observed in the simulation results changes in one of these verification steps, then this is a clear indication that the intended behavior of the user's design has changed during our workflow. At this point, the Verilog compiler and/or converters must be fixed to ensure that information stays consistent and the data are producing valid results.

Verilog simulations can be done by using any available Verilog simulation tool. For our workflow, we used ModelSim to observe the behavior of our Verilog designs. Whereas for genetic circuits, there are a variety of formal methods and computational tools for their analysis [69], [70]. For this workflow, we used iBioSim as the simulation tool to model and simulate our genetic circuits, since it supports the SBML data format that is used within our workflow. From the array of simulation methods supported in iBioSim, we opted for stochastic simulation for analyzing the behavior of our circuit since the molecule counts in these circuits are low and the behavior of these genetic asynchronous circuits must be analyzed in the presence of noise and hazards.

The final step of our verification procedure leverages the model generation procedure described in [68]. This procedure begins with the SBOL representation of the DNA-level design produced by the technology mapping step. It then enriches this representation with interaction data stored in SynBioHub. The resulting qualitative SBOL representation can then be converted into a dynamic SBML model using the SBOL to SBML converter described in [54]. This SBML representation can then be simulated and compared with the original simulation of the Verilog specification.

## IV. Case Study

This section presents a case study demonstrating our workflow on a genetic sensor that uses filtering and communication to make a more reliable detection decision. In particular, the sensor is constructed from three sensors as described in the Verilog specification shown in Figure 1(b). These sensors are connected such that the Actuator of the first sensor is connected to the Start signal of the second sensor, and the Actuator of the second is connected to the Start signal of the third sensor. All three sensors share the same Sensor input. This genetic sensor circuit could be used, for example, to release pharmaceuticals in specific tissues, after sensing a cancer-related signal for a prolonged period of time [16], [71]. Examples of tissue specific molecules, or the enzymes that produce them, are curated by the Human Protein Atlas and include surfactant protein A1 in the lungs, thyroglobulin in the thyroid, and uromodulin in the kidney [72], [73]. There are many examples of possible cancer markers, though often

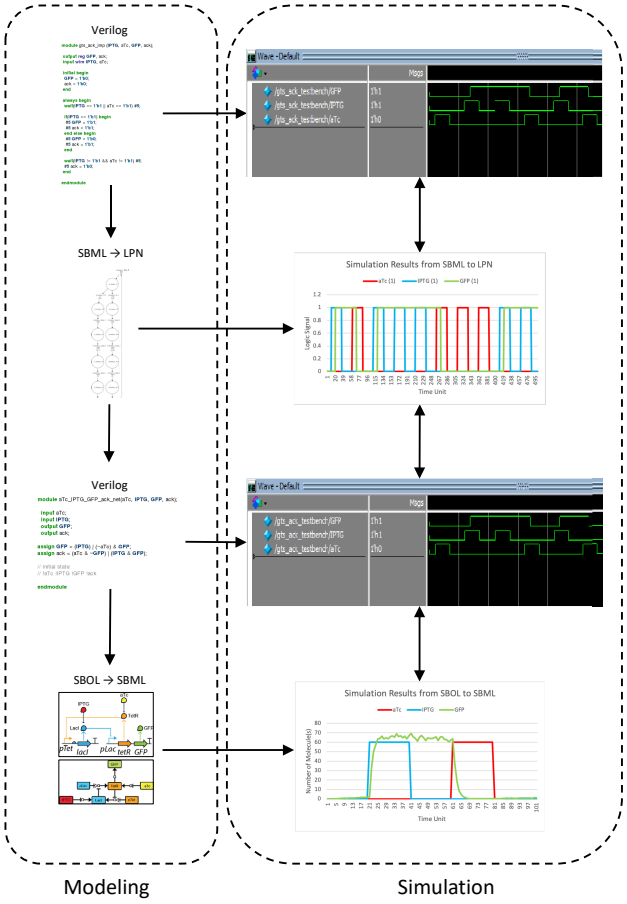



Fig. 5. An example of a verification process applied within our workflow. Arrows from the modeling box to the simulation box depict output simulations and arrows between simulations depict comparison. The Verilog simulations are produced from a Verilog simulator called ModelSim. The two simulations showing the results produced from the SBML to LPN converter and the SBOL to SBML converter are simulated in iBioSim. The result of iBioSim's simulation was exported to a .csv file and then plotted using Excel. The goal of our verification process is to ensure that the behavior that is described in the initial design remains consistent as the design specification goes through the different levels of our workflow. The first Verilog simulation is created from the testbench modeling the design specification. The SBML to LPN simulation is the compiled version of the initial Verilog design before it is synthesized in ATACS. The third simulation is verified from the logic expression produced from ATACS. The last simulation is produced from the technology mapping tools that is used within our workflow; specifically iBioSim or Cello. As exhibited in the figure, all four simulations are producing the same behavior that is described in the specification. Notably, GFP turns HIGH when IPTG is HIGH and GFP goes LOW when aTc is HIGH.

elevation of a marker above normal is also used, examples include: HURP in prostate cancer cells [74], Carcinoembryonic antigen (CEA) in colorectal cancer [75], and CA 19-9 in pancreatic cancer [76]. The initial Start signal would be a tissue-specific chemical signature and the sensing input would be a cancer-related chemical. The output Actuator for the third sensor would be a chemotherapeutic agent. The bacteria would only produce the payload if they are in the correct tissue, and if they consistently sense the cancer-related input, thus filtering and decreasing the chance of false positives. Additionally, this prevents the release of the chemotherapeutic agent in cancer-free tissues reducing unwanted side effects.

To exemplify this work, a case study circuit is designed that uses IPTG as the Sensor input, aTc as the Start input, and YFP (*yellow fluorescent protein*) as the output. Each sensor is going to be transformed into a different cell type to reduce crosstalk concerns allowing for gate reuse. Essentially, our circuit is composed of a population of three types of cells (note that the host cells may be identical but they each include a different circuit). The cells communicate using quorum sensing molecules LasI and RhlI, which are molecules that can diffuse between the cells.

The result of asynchronous logic synthesis is shown in Figure 6(a). This circuit is composed of three *generalized Muller C-elements* (gC), each of which is implemented as an independent circuit in a different cell. This gC behaves as follows, its output goes high when both inputs are high, and its output goes low only after the input not marked with a "+" goes low. In other words, if the input marked with a "+" goes low first, it remains high. The complete circuit behaves as follows. If the first circuit receives the Start signal (aTc) and the Sensor input (IPTG), it will produce a quorum sensing signal (LasI). If the Sensor input (IPTG) is still present, the second circuit would receive the first quorum sensing signal and the sensor input, thus producing a second quorum sensing molecule (RhlI). The third circuit, if it receives this second quorum sensing signal and the Sensor input (IPTG) is still present, it can produce the Actuator output (YFP). The purpose of this circuit is to work as a low-pass filter, as the Sensor input has to be present during the whole process in order to produce the Actuator. Otherwise, if the Sensor input disappears before the Actuator is produced, all the gC gates reset to low and LasI, RhlI, and YFP are no longer produced and are degraded away. This means that the circuit does not produce an Actuator output if the Sensor input is present only briefly, "filtering out" noise on the Sensor input.

The result of technology mapping using Cello is shown in Figure 6(b). The technology mapping actually produces three distinct genetic circuit sequences. Each sequence can be used to construct a plasmid that can be transformed into a separate culture of cells. The result would be three different cell types, which when mixed together form a population of cells that would implement the entire composite circuit.

Using iBioSim, an ODE simulation of the circuit was performed indicating the filtering behavior of the circuit shown in Figure 7. The simulation produces predictions of in RPUs [77], since the genetic parts from Cello's SynBioHub repository uses this unit for parameterization. In each case, the circuit was exposed to varying amounts of times in which the signal IPTG is high, which produces different responses of the system as a whole (see Figure 7). In those cases where the exposure time to IPTG is not sufficient, the system acts as a filter and does not produce the YFP reporter protein (see Figure 7(a-c)). The YFP reporter protein is only produced when the system has been exposed to IPTG for sufficient time so that each subsystem (cell) produces it's corresponding output before the signal IPTG is removed (see Figure 7(d)).

Separating the circuits into separate cells has multiple advantages. First, it reduces crosstalk problems and allows the

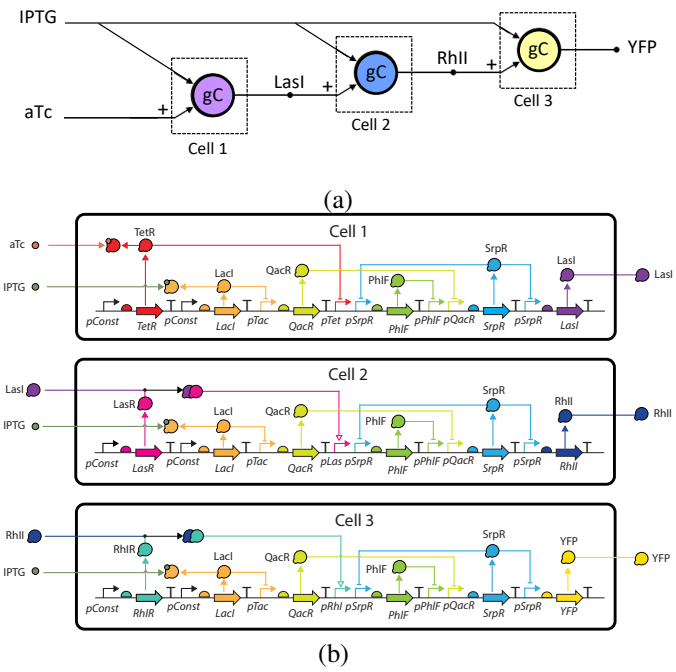



Fig. 6. A genetic sensor that uses filtering and communication to improve its reliability. (a) The logic diagram produced by logic synthesis. It is composed of three gC gates which go high when both inputs are high and go low when the input not marked with a "+" goes low. The output of the second and third gate are connected to the "+" input of the next gate. The detection begins when both IPTG and aTc go high, activating Cell 1. This creates the quorum signal LasI to diffuse to Cell 2, which then activates Cell 2 to produce the quorum signal RhlI. The RhlI signal diffuses to Cell 3 to activate YFP production. However, if IPTG goes low at any point during this chain reaction, the whole circuit resets. (b) The genetic design produced by Cello, which is composed of three genetic sequences that can be put onto separate plasmids and transformed into cells to create three cell types.

same gates to be reused. Second, it increases the delay of the circuit to improve the filtering performance. Third, it enables further robustness as errors caused by hazards or other noise sources in a small number of cells can be tolerated, since the overall behavior is determined by the population dynamics. The decision to split the circuit across three cells is to create a delay to filter out momentary pulses of IPTG. In order to create a realistic design, however, we were limited no more than three cells, since the number of known orthogonal quorum sensing molecules is limited (namely, LasI, and RhlI). Even limiting to three cell types, there are additional challenges in engineering microbial consortia. Some of these challenges are shared with engineering of homogeneous cell populations. Others, like maintaining stable cell proportions, avoiding horizontal gene-transfer [78] or engineering stable cell-cell communications [79] are specific for multi-cellular engineering and require special attention [80].

## V. Conclusion

This paper demonstrates a workflow to construct asynchronous genetic circuits by leveraging asynchronous logic design methods. This workflow leverages data standards such as SBOL and SBML, allowing different tools to be easily composed and potentially replaced with alternatives in the future. This proposed workflow allows asynchronous genetic circuits to be described and the resulting designs can be verified through simulation. The workflow starts with a Verilog design to describe the specification and a testbench to verify the specification's behavior. The Verilog design is then compiled into an LPN that is fed to ATACS to perform hazard-free asynchronous logic synthesis. ATACS synthesizes the LPN design into a logic design also expressed in Verilog. This Verilog design can then be imported into Cello or iBioSim for technology mapping to produce a physical design. Finally, the SBOL data model can be enriched with interaction data stored in SynBioHub to enable conversion into SBML to validate that the behavior of the physical design is consistent with the original Verilog design. The proposed workflow is demonstrated on an asynchronous genetic sensor design that can potentially be substantially more reliable than a combinational sensor.

While this workflow is promising, there is still a lot of work that needs to be done. Verilog templates should be developed to help designers who are not familiar with Verilog and/or asynchronous design. The library should also be expanded to include gates and characterization data for more types of logic families and host organisms. Next, while all steps in the workflow exist, they still need to be completely connected into a seamless and automated procedure. In particular, the multi-cellular design optimization is not currently supported by the tools, but must be created by hand. Also, the dynamic model generation work needs some refinement to better support modeling of the diffusion of quorum sensing molecules. Furthermore, the modeling could be improved through the support of stochastic analysis and population dynamics by leveraging techniques such as stochastic model checking [81], [82]. Such analysis would enable a better understanding of the effects of hazards in genetic circuits and indeed when and if they need to be avoided. Finally, some designs produced by this workflow should be built and tested in the laboratory.

## Acknowledgment

The authors of this work are supported by the National Science Foundation under Grant No., 1522074 (T.J., C.M., and D.D.), CCF-1748200 (T.N., J.M., and C.M.), DBI-1356041 (T.N. and C.M.), and DARPA FA8750-17-C-0229 (T.J., P.F., Z.Z., D.D., and C.M.). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding agencies.

## References

[1] G. Schendzielorz, M. Dippong, A. Grnberger, D. Kohlheyer, A. Yoshida, S. Binder, C. Nishiyama, M. Nishiyama, M. Bott, and L. Eggeling, "Taking control over control: use of product sensing in single cells to remove flux control at key enzymes in biosynthesis pathways," *ACS synthetic biology*, vol. 3, no. 1, pp. 21–29, 2013.

[2] F. Zhang, J. M. Carothers, and J. D. Keasling, "Design of a dynamic sensor-regulator system for production of chemicals and fuels derived from fatty acids," *Nature biotechnology*, vol. 30, no. 4, pp. 354–359, 2012.

[3] T.-M. Yi, Y. Huang, M. I. Simon, and J. Doyle, "Robust perfect adaptation in bacterial chemotaxis through integral feedback control," *Proceedings of the National Academy of Sciences*, vol. 97, no. 9, pp. 4649–4653, 2000.

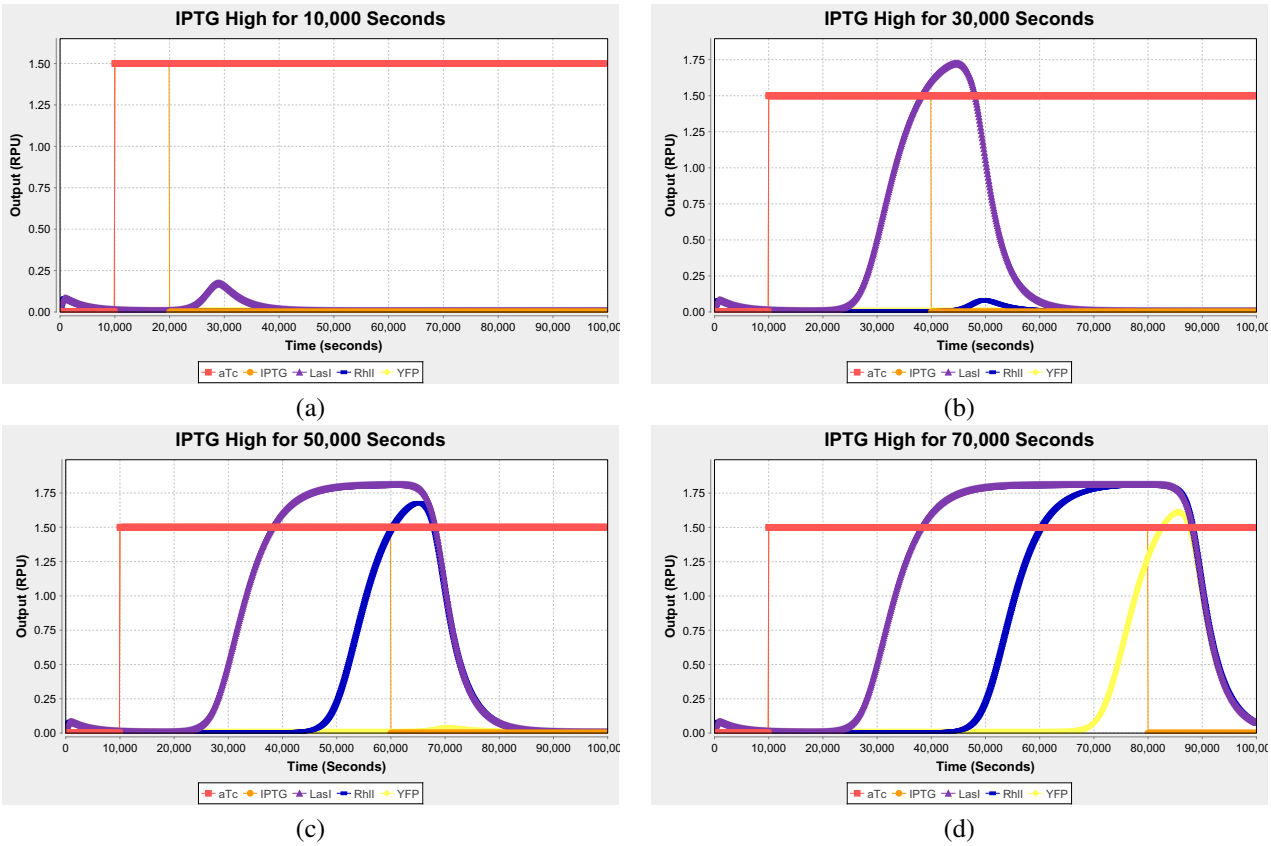



Fig. 7. iBioSim ODE Runge-Kutta simulation of the genetic sensor demonstrating the filtering behavior. Note that all results are measured in RPUs. For the circuit outputs to turn on (LasI, RhlI, and YFP for the three cells respectively) both aTc and IPTG inputs need to be present, however if the IPTG signal is only briefly present then the circuit filters this and does not show the YFP output. This behavior can be seen in the four different panels where the IPTG is present for differing amounts of time. (a) IPTG is present for 10,000 seconds and almost no LasI is produced. (b) IPTG is present for 30,000 seconds and LasI is produced by the first cell but only a little RhlI is produced by the second cell. (c) IPTG is present for 50,000 seconds and both LasI and RhlI are produced with a clear delay between the two peaks. (d) IPTG is present for 70,000 seconds providing sufficient time for the final circuit output (YFP) to be produced. Although not shown in these plots, if aTc is removed after about 40,000 seconds, there is no significant change in the overall behavior.


[4] K. Krishnanathan, S. R. Anderson, S. A. Billings, and V. Kadirkamanathan, "A data-driven framework for identifying nonlinear dynamic models of genetic parts," *ACS synthetic biology*, vol. 1, no. 8, pp. 375–384, 2012.

[5] P. Carbonell, P. Parutto, C. Baudier, C. Junot, and J.-L. Faulon, "Retropath: automated pipeline for embedded metabolic circuits," *ACS synthetic biology*, vol. 3, no. 8, pp. 565–577, 2013.

[6] B. L. Adams, K. K. Carter, M. Guo, H.-C. Wu, C.-Y. Tsao, H. O. Sintim, J. J. Valdes, and W. E. Bentley, "Evolved quorum sensing regulator, lsrr, for altered switching functions," *ACS synthetic biology*, vol. 3, no. 4, pp. 210–219, 2013.

[7] T. Umeyama, S. Okada, and T. Ito, "Synthetic gene circuit-mediated monitoring of endogenous metabolites: identification of gal11 as a novel multicopy enhancer of s-adenosylmethionine level in yeast," *ACS synthetic biology*, vol. 2, no. 8, pp. 425–430, 2013.

[8] J. A. Stapleton, K. Endo, Y. Fujita, K. Hayashi, M. Takinoue, H. Saito, and T. Inoue, "Feedback control of protein expression in mammalian cells by tunable synthetic translational inhibition," *ACS synthetic biology*, vol. 1, no. 3, pp. 83–88, 2011.

[9] M. H. Medema, R. Breitling, R. Bovenberg, and E. Takano, "Exploiting plug-and-play synthetic biology for drug discovery and production in microorganisms," *Nature reviews. Microbiology*, vol. 9, no. 2, p. 131, 2011.

[10] M. Fischbach and C. A. Voigt, "Prokaryotic gene clusters: a rich toolbox for synthetic biology," *Biotechnology journal*, vol. 5, no. 12, pp. 1277–1296, 2010.

[11] H.-J. Frasch, M. H. Medema, E. Takano, and R. Breitling, "Design-based re-engineering of biosynthetic gene clusters: plug-and-play in practice," *Current opinion in biotechnology*, vol. 24, no. 6, pp. 1144–1150, 2013.

[12] K. Temme, D. Zhao, and C. A. Voigt, "Refactoring the nitrogen fixation gene cluster from klebsiella oxytoca," *Proceedings of the National Academy of Sciences*, vol. 109, no. 18, pp. 7085–7090, 2012.

[13] Z. Shao, G. Rao, C. Li, Z. Abil, Y. Luo, and H. Zhao, "Refactoring the silent spectinabilin gene cluster using a plug-and-play scaffold," *ACS synthetic biology*, vol. 2, no. 11, pp. 662–669, 2013.

[14] C. Oßwald, G. Zipf, G. Schmidt, J. Maier, H. S. Bernauer, R. Mller, and S. C. Wenzel, "Modular construction of a functional artificial epothilone polyketide pathway," *ACS synthetic biology*, vol. 3, no. 10, pp. 759–772, 2012.

[15] L. Steidler, W. Hans, L. Schotte, S. Neirynck, F. Obermeier, W. Falk, W. Fiers, and E. Remaut, "Treatment of murine colitis by lactococcus lactis secreting interleukin-10," *Science*, vol. 289, no. 5483, pp. 1352–1355, 2000.

[16] J. C. Anderson, E. J. Clarke, A. P. Arkin, and C. A. Voigt, "Environmentally controlled invasion of cancer cells by engineered bacteria," *Journal of molecular biology*, vol. 355, no. 4, pp. 619–627, 2006.

[17] W. C. Ruder, T. Lu, and J. J. Collins, "Synthetic biology moving into the clinic," *Science*, vol. 333, no. 6047, pp. 1248–1252, 2011.

[18] J.-P. Motta, L. G. Bermúdez-Humarán, C. Deraison, L. Martin, C. Rolland, P. Rousset, J. Boue, G. Dietrich, K. Chapman, P. Kharrat *et al.*, "Food-grade bacteria expressing elafin protect against inflammation and restore colon homeostasis," *Science translational medicine*, vol. 4, no. 158, pp. 158ra144–158ra144, 2012.

[19] S. Wang, Q. Kong, and R. Curtiss, "New technologies in developing recombinant attenuated salmonella vaccine vectors," *Microbial pathogenesis*, vol. 58, pp. 17–28, 2013.

[20] J. H. Huh, J. T. Kittleson, A. P. Arkin, and J. C. Anderson, "Modular design of a synthetic payload delivery device," *ACS synthetic biology*, vol. 2, no. 8, pp. 418–424, 2013.

[21] S. Gupta, E. E. Bram, and R. Weiss, "Genetically programmable

pathogen sense and destroy," *ACS synthetic biology*, vol. 2, no. 12, pp. 715–723, 2013.

[22] I. Y. Hwang, M. H. Tan, E. Koh, C. L. Ho, C. L. Poh, and M. W. Chang, "Reprogramming microbes to be pathogen-seeking killers," *ACS synthetic biology*, vol. 3, no. 4, pp. 228–237, 2013.

[23] A. Prindle, J. Selimkhanov, T. Danino, P. Samayoa, A. Goldberg, S. N. Bhatia, and J. Hasty, "Genetic circuits in salmonella typhimurium," *ACS synthetic biology*, vol. 1, no. 10, pp. 458–464, 2012.

[24] K. Volzing, J. Borrero, M. J. Sadowsky, and Y. N. Kaznessis, "Antimicrobial peptides targeting gram-negative pathogens, produced and delivered by lactic acid bacteria," *ACS synthetic biology*, vol. 2, no. 11, pp. 643–650, 2013.

[25] J. Hasty, "Engineered microbes for therapeutic applications," 2012.

[26] T. Danino, J. Lo, A. Prindle, J. Hasty, and S. N. Bhatia, "In vivo gene expression dynamics of tumor-targeted bacteria," *ACS synthetic biology*, vol. 1, no. 10, pp. 465–470, 2012.

[27] E. Engineered, "coli that detect and respond to gut inflammation through nitric oxide sensing archer, eric j.; robinson, andra b.; suel, gurol m," *ACS Synthetic Biology*, vol. 1, no. 10, pp. 451–457, 2012.

[28] M. S. Antunes, K. J. Morey, J. J. Smith, K. D. Albrecht, T. A. Bowen, J. K. Zdunek, J. F. Troupe, M. J. Cuneo, C. T. Webb, H. W. Hellinga *et al.*, "Programmable ligand detection system in plants through a synthetic signal transduction pathway," *PLoS One*, vol. 6, no. 1, p. e16292, 2011.

[29] D. M. Widmaier, D. Tullman-Ercek, E. A. Mirsky, R. Hill, S. Govindarajan, J. Minshull, and C. A. Voigt, "Engineering the salmonella type iii secretion system to export spider silk monomers," *Molecular Systems Biology*, vol. 5, no. 1, p. 309, 2009.

[30] K. Bernhardt, N. S. Chand, E. Carter, J. Lee, Y. Xu, X. Zhu, D. Rowe, J. W. Ajioka, J. Goncalves, J. Haseloff *et al.*, "New tools for self-organized pattern formation," *BMC Systems Biology*, vol. 1, no. 1, p. S10, 2007.

[31] X.-X. Xia, Z.-G. Qian, C. S. Ki, Y. H. Park, D. L. Kaplan, and S. Y. Lee, "Native-sized recombinant spider silk protein produced in metabolically engineered escherichia coli results in a strong fiber," *Proceedings of the National Academy of Sciences*, vol. 107, no. 32, pp. 14 059–14 063, 2010.

[32] D. M. Widmaier and C. A. Voigt, "Quantification of the physiochemical constraints on the export of spider silk proteins by salmonella type iii secretion," *Microbial cell factories*, vol. 9, no. 1, p. 78, 2010.

[33] F. Aquea, F. Federici, C. Moscoso, A. Vega, P. Jullian, J. Haseloff, and P. ARCE-JOHNSON, "A molecular framework for the inhibition of arabidopsis root growth in response to boron toxicity," *Plant, cell & environment*, vol. 35, no. 4, pp. 719–734, 2012.

[34] M. S. Antunes, S.-B. Ha, N. Tewari-Singh, K. J. Morey, A. M. Trofka, P. Kugrens, M. Deyholos, and J. I. Medford, "A synthetic de-greening gene circuit provides a reporting system that is remotely detectable and has a re-set capacity," *Plant biotechnology journal*, vol. 4, no. 6, pp. 605–622, 2006.

[35] T. S. Gardner, C. R. Cantor, and J. J. Collins, "Construction of a genetic toggle switch in *escherichia coli*," *Nature*, vol. 403, no. 6767, pp. 339–342, 2000.

[36] J. Sardanyés, A. Bonforti, N. Conde, R. Solé, and J. Macia, "Computational implementation of a tunable multicellular memory circuit for engineered eukaryotic consortia," *Frontiers in physiology*, vol. 6, 2015.

[37] M. C. Inniss and P. A. Silver, "Building synthetic memory," *Current Biology*, vol. 23, no. 17, pp. R812–R816, 2013.

[38] A. Urrios, J. Macia, R. Manzoni, N. Conde, A. Bonforti, E. de Nadal, F. Posas, and R. Sol, "A synthetic multicellular memory device," *ACS synthetic biology*, vol. 5, no. 8, pp. 862–873, 2016.

[39] L. Andrews, A. Nielsen, and C. Voigt, "Cellular checkpoint control using programmable sequential logic," *Science*, vol. 361, no. 6408, 2018. [Online]. Available: http://science.sciencemag.org/content/361/6408/eaap8987

[40] J. Böhm, S. Scherzer, E. Krol, I. Kreuzer, K. von Meyer, C. Lorey, T. D. Mueller, L. Shabala, I. Monte, R. Solano *et al.*, "The venus flytrap dionaea muscipula counts prey-induced action potentials to induce sodium uptake," *Current Biology*, vol. 26, no. 3, pp. 286–295, 2016.

[41] M. Escalante-Prez, E. Krol, A. Stange, D. Geiger, K. A. S. Al-Rasheid, B. Hause, E. Neher, and R. Hedrich, "A special pair of phytohormones controls excitability, slow closure, and external stomach formation in the venus flytrap," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 108, no. 37, p. 1549215497, Sep 2011. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3174645/

[42] 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*, vol. 352, no. 6281, 2016. [Online]. Available: http://science.sciencemag.org/content/352/6281/aac7341

[43] N. Roehner and C. J. Myers., "Directed acyclic graph-based technology mapping of genetic circuit models," *ACS Synthetic Biology*, vol. 3, no. 8, pp. 543–555, 2014.

[44] 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 synthetic biology*, 2018.

[45] L. Cardelli, M. Kwiatkowska, and M. Whitby, "Chemical reaction network designs for asynchronous logic circuits," in *International Conference on DNA-Based Computers*. Springer, 2016, pp. 67–81.

[46] Y.-J. Chen, S. D. Rao, and G. Seelig, "Plasmid-derived dna strand displacement gates for implementing chemical reaction networks," *Journal of Visualized Experiments: JoVE*, no. 105, Nov 2015. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4692756/

[47] C. J. Myers, W. Belluomini, K. Kallpack, E. Peskin, and H. Zheng, "Timed circuits: A new paradigm for high-speed design," in *Proceedings of the 2001 Asia and South Pacific Design Automation Conference*. ACM, 2001, pp. 335–340.

[48] M. B. Elowitz and S. Leibler, "A synthetic oscillatory network of transcriptional regulators," *Nature*, vol. 403, no. 6767, pp. 335–338, 2000.

[49] M. Galdzicki, K. Clancy, E. Oberortner, M. Pocock, J. Quinn, C. Rodriguez, N. Roehner, M. Wilson, L. Adam, C. Anderson *et al.*, "The synthetic biology open language (sbol) provides a community standard for communicating designs in synthetic biology," *Nature biotechnology*, vol. 32, no. 6, p. 545, 2014.

[50] N. Roehner, J. Beal, K. Clancy, B. Bartley, G. Mısırlı, R. Grünberg, E. Oberortner, M. Pocock, M. Bissell, C. Madsen, T. Nguyen, M. Zhang, Z. Zhang, Z. Zundel, D. Densmore, J. Gennari, A. Wipat, H. Sauro, and C. Myers, "Sharing structure and function in biological design with SBOL 2.0," *ACS synthetic biology*, vol. 5, no. 6, pp. 498–506, 2016.

[51] M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, and et al., "The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models." *Bioinformatics*, vol. 19, no. 4, pp. 524–531, Mar. 2003.

[52] J. McLaughlin, C. Myers, Z. Zundel, G. Mısırlı, M. Zhang, I. Ofiteru, A. Goñi Moreno, and A. Wipat, "Synbiohub: A standards-enabled design repository for synthetic biology," *ACS synthetic biology*, vol. 7, no. 2, pp. 682–688, 2018.

[53] J. Y. Quinn, R. S. Cox III, A. Adler, J. Beal, S. Bhatia, Y. Cai, J. Chen, K. Clancy, M. Galdzicki, N. J. Hillson *et al.*, "Sbol visual: a graphical language for genetic designs," *PLoS biology*, vol. 13, no. 12, p. e1002310, 2015.

[54] N. Roehner, Z. Zhang, T. Nguyen, and C. J. Myers, "Generating systems biology markup language models from the synthetic biology open language," *ACS synthetic biology*, vol. 4, no. 8, pp. 873–879, 2015.

[55] 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 synthetic biology*, vol. 5, no. 6, pp. 479–486, 2016.

[56] P. Vaidyanathan, B. S. Der, S. Bhatia, N. Roehner, R. Silva, C. A. Voigt, and D. Densmore, "A framework for genetic logic synthesis," *Proceedings of the IEEE*, vol. 103, no. 11, pp. 2196–2207, Nov 2015.

[57] F. Yaman, S. Bhatia, A. Adler, D. Densmore, and J. Beal, "Automated selection of synthetic biology parts for genetic regulatory networks," *ACS synthetic biology*, vol. 1, no. 8, pp. 332–344, 2012.

[58] L. Huynh, A. Tsoukalas, M. Kppe, and I. Tagkopoulos, "Sbrome: A scalable optimization and module matching framework for automated biosystems design," *ACS Synthetic Biology*, vol. 2, no. 5, pp. 263–273, 2013, pMID: 23654271. [Online]. Available: http://dx.doi.org/10.1021/sb300095m

[59] H. Baig and J. Madsen, "A top-down approach to genetic circuit synthesis and optimized technology mapping," 2017.

[60] S. Little, D. Walter, C. Myers, R. Thacker, S. Batchu, and T. Yoneda, "Verification of analog/mixed-signal circuits using labeled hybrid petri nets," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 4, pp. 617–630, April 2011.

[61] C. Madsen, C. J. Myers, T. Patterson, N. Roehner, J. T. Stevens, and C. Winstead, "Design and test of genetic circuits using iBioSim," *IEEE Design Test of Computers*, vol. 29, no. 3, pp. 32–39, June 2012.

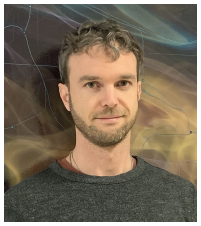
[62] N. Roehner and C. Myers, "Directed acyclic graph-based technology mapping of genetic circuit models," *ACS Synthetic Biology*, vol. 3, no. 8, pp. 543–555, 08 2014. [Online]. Available: https://doi.org/10.1021/sb400135t

[63] G. Mısırlı, T. Nguyen, , J. McLaughlin, , P. Vaidyanathan, T. Jones, D. Densmore, C. Myers, and A. Wipat, "A computational workflow for

the automated generation of models of genetic designs," *ACS synthetic biology*, 2018.
[64] G. Misirli, J. Hallinan, and A. Wipat, "Composable modular models for synthetic biology," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 11, no. 3, p. 22, 2014.
[65] C. J. Myers, *Asynchronous circuit design*. John Wiley & Sons, 2001.
[66] E. V. Nikolaev and E. D. Sontag, "Quorum-sensing synchronization of synthetic toggle switches: A design based on monotone dynamical systems theory," *PLoS computational biology*, vol. 12, no. 4, p. e1004881, 2016.
[67] P. Vaidyanathan, B. S. Der, S. Bhatia, N. Roehner, R. Silva, C. A. Voigt, and D. Densmore, "A framework for genetic logic synthesis," *Proceedings of the IEEE*, vol. 103, no. 11, pp. 2196–2207, 2015.
[68] G. Mısırlı, T. Nguyen, J. A. McLaughlin, P. Vaidyanathan, T. Jones, D. Densmore, C. J. Myers, and A. Wipat, "A computational workflow for the automated generation of models of genetic designs," *ACS synthetic biology*, 2018.
[69] C. J. Myers, *Engineering genetic circuits*. Chapman and Hall/CRC, 2009.
[70] ——, "Platforms for genetic design automation," in *Methods in Microbiology*. Elsevier, 2013, vol. 40, pp. 177–202.
[71] N. Nguyen, C. Myers, H. Kuwahara, C. W., and J. Keener, "Design and analysis of a robust genetic muller c-element," *Journal of Theoretical Biology*, vol. 264, no. 2, pp. 174 – 187, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0022519309005153
[72] "The human proteome the human protein atlas." [Online]. Available: https://www.proteinatlas.org/humanproteome/tissue/tissue+specific
[73] J. Zhu, G. Chen, S. Zhu, S. Li, Z. Wen, B. Li, Y. Zheng, and L. Shi, "Identification of tissue-specific protein-coding and noncoding transcripts across 14 human tissues using rna-seq," *Scientific Reports*, vol. 6, p. 28400, Jun 2016. [Online]. Available: https://www.nature.com/articles/srep28400
[74] I. Espinoza, M. J. Sakiyama, T. Ma, L. Fair, X. Zhou, M. Hassan, J. Zabaleta, and C. R. Gomez, "Hypoxia on the expression of hepatoma upregulated protein in prostate cancer cells," *Frontiers in Oncology*, vol. 6, Jun 2016. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4908134/
[75] M. J. Duffy, A. van Dalen, C. Haglund, L. Hansson, R. Klapdor, R. Lamerz, O. Nilsson, C. Sturgeon, and O. Topolcan, "Clinical utility of biochemical markers in colorectal cancer: European group on tumour markers (egtm) guidelines," *European Journal of Cancer*, vol. 39, no. 6, p. 718727, Apr 2003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0959804902008110
[76] K. S. Goonetilleke and A. K. Siriwardena, "Systematic review of carbohydrate antigen (ca 19-9) as a biochemical marker in the diagnosis of pancreatic cancer," *European Journal of Surgical Oncology (EJSO)*, vol. 33, no. 3, p. 266270, Apr 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0748798306003763
[77] 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," *Journal of biological engineering*, vol. 3, no. 1, p. 4, 2009.
[78] J. Davison, "Genetic exchange between bacteria in the environment," *Plasmid*, vol. 42, no. 2, pp. 73–91, 1999.
[79] A. Pai, Y. Tanouchi, C. H. Collins, and L. You, "Engineering multicellular systems by cell–cell communication," *Current opinion in biotechnology*, vol. 20, no. 4, pp. 461–470, 2009.
[80] K. Brenner, L. You, and F. H. Arnold, "Engineering microbial consortia: a new frontier in synthetic biology," *Trends in biotechnology*, vol. 26, no. 9, pp. 483–489, 2008.
[81] C. Madsen, Z. Zhang, N. Roehner, C. Winstead, and C. Myers, "Stochastic model checking of genetic circuits," *J. Emerg. Technol. Comput. Syst.*, vol. 11, no. 3, pp. 23:1–23:21, Dec. 2014. [Online]. Available: http://doi.acm.org/10.1145/2644817
[82] T. Neupane, Z. Zhang, C. Madsen, H. Zheng, and C. J. Myers, "Approximation Techniques for Stochastic Analysis of Biological Systems," *arXiv e-prints*, p. arXiv:1901.07857, Jan. 2019.

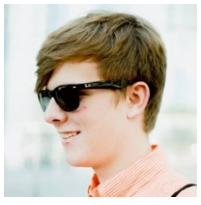
**Tramy Nguyen** received the B.S. and M.S. degrees in electrical and computer engineering at the University of Utah, Salt Lake City, UT, USA in 2015 and 2016, respectively. She is currently a Ph.D. student working in the Myers Research Group. Her research interest is applying engineering concepts to help develop CAD tools for automating the process of designing asynchronous genetic circuits. Her contributions have also taken place in the SBOL community where she has represented the community as an SBOL Editor from 2016 to 2018.

**Timothy S. Jones** received the B.S.E. degree in electrical engineering in 2010 and the Ph.D. degree in electrical engineering in 2016, both from the University of Pennsylvania, Philadelphia, PA. He is currently a postdoctoral research associate under Douglas Densmore at Boston University. His current research involves extensions to the genetic circuit design software Cello, as well as quantitative microscopy and modeling spatiotemporal behavior of cell-cell communication systems.

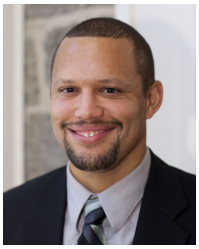


**Pedro Fontanarrosa** received his B.S. in Biological Sciences in 2008 and his M.S. in Evolutionary Biology in 2011 from the University of Buenos Aires, Argentina. He continued researching on a University of Buenos Aire's scholarship until 2014. His research was based mainly on speciation of indigenous species of *Drosophila* in Argentina. He received a Fulbright scholarship to pursue a Ph.D. Degree in Bioengineering and moved to Salt Lake City, Utah. As Fulbright Scholar at the University of Utah, he specialized in Synthetic Biology and works under Chris Myers' mentorship.

**Jeanet V. Mante** received her B.A. degree in Natural Science from the University of Cambridge, United Kingdom in 2018. Her primary research was on the quantification of transcription factor localisation in *Marchantia polymorpha* gemmae. She is currently a Ph.D. student working under Chris Myers at the University of Utah, Salt Lake City, UT, USA.



**Zach Zundel** is pursuing a dual-BEng in Computer Science and Biomedical Eningeering at the University of Utah. He is currently an undergraduate researcher in Chris Myers' group. His research interests include tooling for synthetic biologists, data standardization, and the semantic web.



**Douglas Densmore** (Senior Member, IEEE) received the B.S.E. degree in computer engineering from the University of Michigan, Ann Arbor, MI, USA, in 2001, and the M.S. and Ph.D. degrees in electrical engineering from the University of California, Berkeley, CA, USA, in 2004 and 2007, respectively.

He is a Kern Faculty Fellow at Hariri Institute for Computing and Computational Science and Engineering Junior Faculty Fellow and Associate Professor in the Department of Electrical and Computer Engineering at Boston University, Boston, MA, USA. His research focuses on the development of tools for the specification, design, and assembly of synthetic biological systems, drawing upon his experience with embedded system level design and electronic design automation (EDA). He is the director of the Cross-disciplinary Integration of Design Automation Research (CIDAR) group at Boston University, which develops computational and experimental tools for synthetic biology. His research interests include synthetic biology, bioelectronic systems, cyber physical systems, digital logic design, and system level design.

**Chris J. Myers** (Fellow, IEEE) received the B.S. degree in Electrical Engineering and Chinese history in 1991 from the California Institute of Technology, Pasadena, CA, and the M.S.E.E. and Ph.D. degrees from Stanford University, Stanford, CA, in 1993 and 1995, respectively. He is a Professor and Associate Chair in the Department of Electrical and Computer Engineering, University of Utah, Salt Lake City, UT. Dr. Myers is the author of over 170 technical papers and the textbooks Asynchronous Circuit Design and Engineering Genetic Circuits. He is also a co-inventor on 4 patents. His research interests include asynchronous circuit design, formal verification of analog/mixed signal circuits and cyber-physical systems, and modeling, analysis, and design of genetic circuits. Dr. Myers received an NSF Fellowship in 1991, an NSF CAREER award in 1996, and best paper awards at the 1999 and 2007 Symposiums on Asynchronous Circuits and Systems. Dr. Myers is a Fellow of the IEEE, and he is a Member of the Editorial Board for ACS Synthetic Biology, Engineering Biology, and Synthetic Biology, and has served on the Editorial Boards for the IEEE Transactions on VLSI Systems, IEEE Design & Test Magazine, IEEE Life Sciences Letters, and Formal Methods in System Design. Dr. Myers has also served as an editor for the Systems Biology Markup Language (SBML) standard and is on the steering committee for the Synthetic Biology Open Language (SBOL) standard.