

Robustness and reproducibility of simple and complex synthetic logic circuit designs using a DBTL loop

Breschine Cummins^{1*}, Justin Vrana², Robert C. Moseley³, Hamed Eramian⁴, Anastasia Deckard⁵, Pedro Fontanarrosa⁶, Daniel Bryce⁶, Mark Weston⁴, George Zheng⁴, Joshua Nowak⁷, Francis C. Motta⁸, Mohammed Eslami⁴, Kara Layne Johnson¹, Robert P. Goldman⁶, Chris J. Myers⁹, Tessa Johnson⁵, Matthew W. Vaughn¹⁰, Niall Gaffney¹⁰, Joshua Urrutia¹⁰, Shweta Gopaulakrishnan¹⁰, Vanessa Biggers⁷, Trissha R. Higa⁷, Lorraine A. Mosqueda⁷, Marcio Gameiro¹¹, Tomáš Gedeon¹, Konstantin Mischaikow¹¹, Jacob Beal¹², Bryan Bartley¹², Tom Mitchell¹², Tramy T. Nguyen¹², Nicholas Roehner¹², and Steven B. Haase¹³

¹Department of Mathematical Sciences, Montana State University, Bozeman, MT, USA

²UW BIOFAB, Seattle, WA, USA

³Department of Biology, Duke University, Durham, NC, USA

⁴Netrias LLC, Annapolis, MD, USA

⁵Geometric Data Analytics, Inc., Durham, NC, USA

⁶Sift, LLC, Minneapolis, MN, USA

⁷Strateos, Inc., Menlo Park, CA, USA

⁸Department of Mathematical Sciences, Florida Atlantic University, Boca Raton, FL, USA

⁹Electrical, Computer & Energy Engineering, University of Colorado Boulder, Boulder, CO, USA

¹⁰Texas Advanced Computing Center, Austin, TX, USA

¹¹Department of Mathematics, Rutgers University, New Brunswick, NJ, USA

¹²Raytheon BBN, Cambridge, MA, USA

*Corresponding authors: E-mail: breschine.cummins@montana.edu

Abstract

Computational tools addressing various components of design-build-test-learn (DBTL) loops for the construction of synthetic genetic networks exist but do not generally cover the entire DBTL loop. This manuscript introduces an end-to-end sequence of tools that together form a DBTL loop called Design Assemble Round Trip (DART). DART provides rational selection and refinement of genetic parts to construct and test a circuit. Computational support for experimental process, metadata management, standardized data collection and reproducible data analysis is provided via the previously published Round Trip (RT) test-learn loop. The primary focus of this work is on the Design Assemble (DA) part of the tool chain, which improves on previous techniques by screening up to thousands of network topologies for robust performance using a novel robustness score derived from dynamical behavior based on circuit topology only. In addition, novel experimental support software is introduced for the assembly of genetic circuits. A complete design-through-analysis sequence is presented using several OR and NOR circuit designs, with and without structural redundancy, that are implemented in budding yeast. The execution of DART tested the predictions of the design tools, specifically with regard to robust and reproducible performance under different experimental conditions. The data analysis depended on a novel application of machine learning techniques to segment bimodal flow cytometry distributions. Evidence is presented that, in some cases, a more complex build may impart more robustness and reproducibility across experimental conditions.

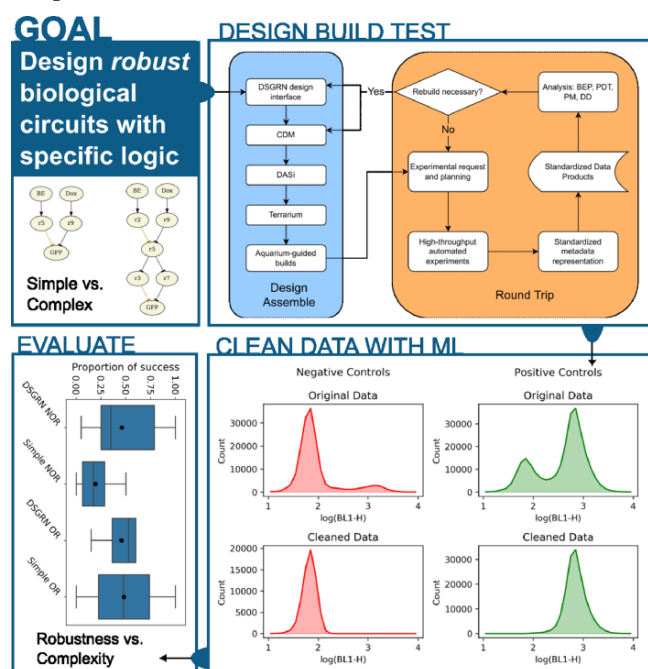
Key words: Synthetic logic circuits; design-build-test-learn; flow cytometry; CRISPR; yeast; genetic circuits; machine learning; automated experiments

Submitted: 30 May 2022; Received (in revised form): 22 February 2023; Accepted: 21 March 2023

© The Author(s) 2023. Published by Oxford University Press.

This is an Open Access article distributed under the terms of the Creative Commons Attribution-NonCommercial License (<https://creativecommons.org/licenses/by-nc/4.0/>), which permits non-commercial re-use, distribution, and reproduction in any medium, provided the original work is properly cited. For commercial re-use, please contact journals.permissions@oup.com

Graphical Abstract



1. Introduction

The construction of synthetic biology genetic circuits is a growing field that holds great promise for producing purpose-built cellular machines that perform important tasks, such as monitoring environmental conditions and producing materials or therapeutics (1–3). The reproducibility of these constructs is of critical importance. The most common approaches to achieving reproducibility in results seek to standardize protocols and analyses and tightly control experimental conditions. However, in the field of synthetic biology, the reproducible function of synthetic genetic circuits may be closely tied to the robustness of the construct design (4). By incorporating measures of robustness into the design principles of synthetic biology, one may be able to generate constructs that have functions that are robust to changes in genetic components or in experimental conditions, causing increased reproducibility across laboratories.

There are many stages in the design of synthetic constructs (5) including (i) the choice of circuit structure (also called topology in this work), (ii) the DNA sequence design of a genetic parts library, (iii) the choice of genetic parts (taken from the parts library) used within the modular circuit structure, (iv) the choice of insertion points or landing pads in the genome or plasmid and (v) the design of the experimental protocol used to assess the resulting design. All of these stages are facilitated by the rational application of computational tools to reduce experimental time and effort and to improve success rate.

Many computational tools are available for different parts of the design process. Examples include methods for DNA sequence design of genetic parts (6; 7), for parts choice and construction into a linear DNA sequence (8–10), for landing pad choice (11) and for circuit structure or topology (12). Experimental protocols are generally custom designed for each application, but tools exist to facilitate the reproducibility of protocols (13; 14). Some tools integrate several stages of the design process together; for example, the software tool Cello (15; 16) takes a logic function written in the Verilog language and identifies a single circuit

structure using design principles from electronic circuit design that employ NOT and NOR gates. Cello then optimizes the modular construction of the logic circuit from a parts library to create a linear DNA sequence with the desired circuit. Another method (17) exhaustively explores fan-out free circuit structures and jointly optimizes structure and parts assignment for a specific logic function.

Design methods are validated by the construction of the predicted optimal circuit design(s), which are then analyzed for performance. The conjunction of the design stage with the build, experimentation and analysis of a synthetically built genetic circuit forms a design–build–test–learn (DBTL) loop (18), in which the results of the analysis can be used to refine the experimental protocol or to tweak the design. The need for end-to-end tooling of DBTL loops, particularly when high-throughput data generation is available, is recognized (19).

In this work, a DBTL 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 (20) that are of interest to the synthetic biology community (21–23). DART is composed of tools for (i) the prediction of robust circuit topologies, (ii) prediction of the most effective choice of parts to construct the topology, (iii) sequence construction for selected designs, (iv) step-by-step instructions for build assembly and (v) reproducible experimental submission, data and metadata consolidation, data standardization and automated data analysis using a previously published test–learn loop called the Round Trip (RT, (24)).

The Design Assemble (DA) part of the tool chain is the primary focus in this work. DA, like Cello, starts with a logic function and a library of genetic parts characterized by dosage–response curves and ultimately produces a linear DNA sequence. The design stage differs from Cello by first screening alternative network topologies—possibly thousands depending on the allowed maximum size of the circuit—and scoring them for dynamically

robust performance to identify the topologies with the greatest predicted robustness in behavior. It was previously shown in (17) that varying the circuit topology can result in improved performance. The approach presented here differs from (17) by the choice of a robustness score based solely on circuit topology that reduces the number of parts optimizations required. In particular, (17) compute distances between simulated flow cytometry (FC) distributions for different parts and topology combinations, so that topology and parts scores are dependent, while the DA design stage has two independent parts. First, a topology robustness score is computed that is a global property of the topology independent of parts choice. Second, once the highest ranked topologies are identified, parts are assigned to each topology using a machine learning technique based on dosage-response FC data.

Each of the highest-scoring topologies with their associated parts assignments are referred to as 'designs' and are then passed to the assemble stage of DA. In this stage, the assigned parts for each assembled into a linear sequence using automated build software. Automated lab software then increases the reproducibility of the build stage in DBTL.

Ideally, a genetic construct will function robustly under a variety of conditions, since in practice it can be difficult to reproduce experiments across labs (25). Robust genetic constructs make it easier to achieve reproducibility, since they make design performance less susceptible to differences between lab conditions. The primary purpose of DART is to produce synthetic logic circuit designs embedded in cells that perform adequately under the widest possible range of conditions and provide reproducible and consistent results.

1.1 Software tool chain

DART provides a systematic and standardized approach to building genetic constructs with desired functional properties. A computational framework was developed that supports the design and construction of synthetic logic circuits from a library of dosage-response characterized parts, DA (Design Assemble), and was attached to an existing tool chain RT (24; 26) that standardizes data collection, preprocessing and analysis. A diverse set of tools was identified, collected and unified to meet the needs of cellular circuit construction from design to analysis. See Figure 1 for a schematic of the connections between the tools.

The RT tool chain supports experimental metadata development and maintenance by automating tedious metadata design and encoding and by reacting and repairing as deviations arise. The RT connects experimental data and subsequent analyses with the experimental DA constructs via user-friendly construct names. For example, RT users develop experimental requests referencing the common names for the constructs developed by DA, which are automatically resolved against Synthetic Biology Open Language (SBOL, (27; 28)) representations of the DA constructs. The RT carries these resolved references through the experimental process to link constructs to experimental results. The resulting data and metadata represent a rich, AI-ready dataset that the RT can automatically analyze and present or package for alternative analysis tools.

The design and build aspects of DART have not been previously used in combination. The design/prediction tools are Dynamic Signatures Generated by Regulatory Networks (DSGRN, (20,29,30)) and Combinatorial Design Model (CDM, (31; 32)). The build tools are DNA Assembly (DASi, (33–35)), the computer-aided process planning tool Terrarium (36; 35) and the lab software Aquarium (14; 37).

DSGRN is a flexible and highly scalable tool for analyzing and predicting all possible long-term dynamics of a regulatory network. It requires only a network topology with annotations indicating whether regulatory interactions are activating or repressing. The computed dynamics exhaustively describes the possible long-term behaviors that the network is capable of exhibiting. Although used here to assess the equilibrium values of logic circuits, DSGRN is not limited to modeling and analysis of logic functions; its capabilities greatly exceed that specialized problem. Its integration into DA involved the implementation of a user-friendly DSGRN Design Interface (38) dedicated to the express purpose of designing synthetic logic circuits. The DSGRN Design Interface incorporates qualitative build constraints in a plain language input file. Interpretation of the output circuit topology scoring is readily accessible via figures and human-readable descriptions of constraints on the interaction of genetic parts, and there is also the option of output in machine-readable SBOL2 format (27).

CDM is a neural-network-based model that makes *in silico* predictions of experiments 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.

Terrarium bridges the gap between synthetic biology design specification and the build process through computer-aided process planning. A biological design is encoded as a biological manufacturing file (BMF). Terrarium converts BMFs into executable networks of protocols, called workflows. These workflows are uploaded to the laboratory software Aquarium, which manages laboratory inventory, automates protocol execution and generates human-readable instructions to execute the workflows. Terrarium has an internal digital model of the laboratory that is periodically updated by metadata collected in Aquarium, specifically protocol execution time, inventory usage, experimental errors, success rates, materials and labor costs. Using this model, Terrarium can predict lead times and costs that inform future workflow process planning for increased accuracy, economy and efficiency. DASi is a subordinate algorithm of Terrarium released as stand-alone software that provides assembly instructions for a DNA sequence when that sequence does not already exist in lab inventory. It produces a BMF that is ingested by Terrarium to create a workflow for Aquarium.

1.2 Biological Scenario

The main hypothesis is that the benefits of genetic network redundancy can outweigh the costs of circuit build complexity through increased reproducibility and robustness across experimental conditions. In this work, we define *robustness* or robust performance to mean the ability of a genetic network to produce some desired behavior across multiple experimental conditions; we define *redundancy* to be the existence of parallel paths in a network structure (39); we define *network complexity* to be the number of parts used to construct the circuit, or similarly the number of nodes and edges in the schematic of a genetic network as shown, for example, in Figure 2; and lastly, we define *build complexity* to be the time and cost associated with constructing the synthetic network. An example of redundancy can be seen in Figure 2. The topology in the lower left quadrant demonstrates parallel paths; the paths $r1 \rightarrow r7 \rightarrow GFP$ and $r1 \rightarrow r10 \rightarrow GFP$ are redundant in that if one path is compromised, the other path is available to transmit the incoming signal. In short, a single part's failure may not impact circuit function. A network with built-in redundancy

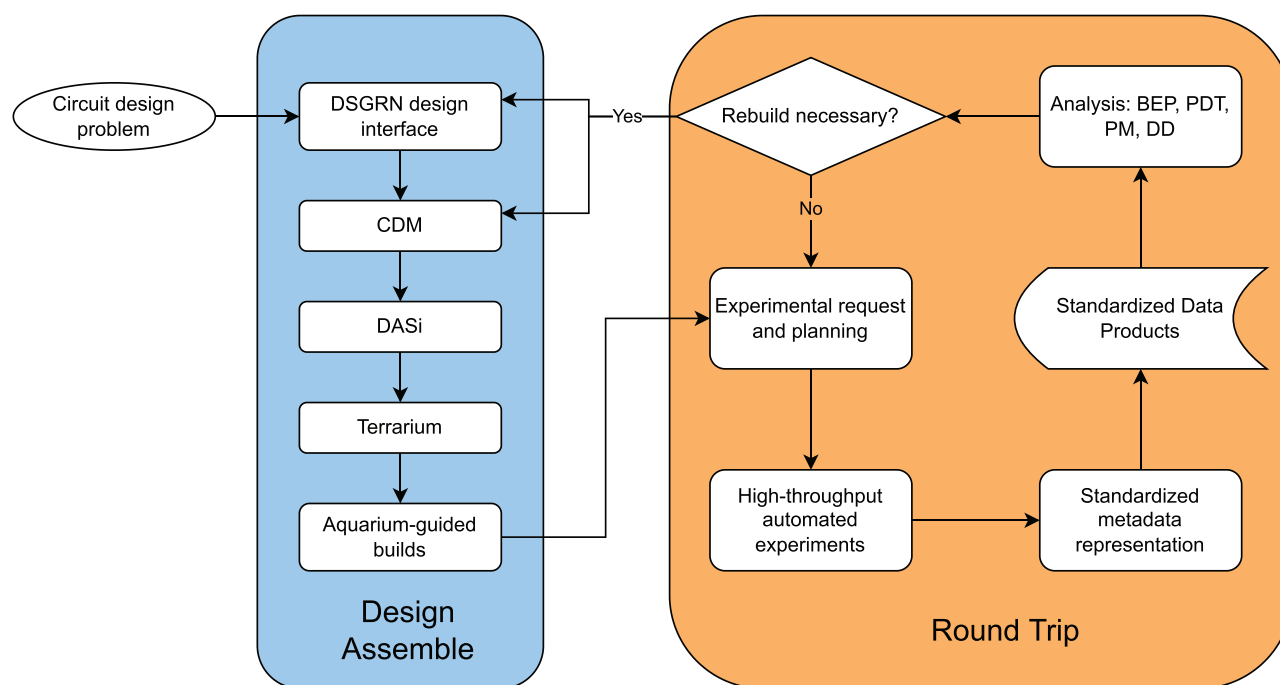


Figure 1. Flowchart showing the details of DART, particularly the integration between DA (introduced here) and RT (24).

has the capacity to exhibit more robust performance (17), but increased network complexity leads to greater build complexity.

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* via FC, 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 behavior were chosen based on preliminary data analysis in which previously built OR and NOR circuits performed poorly ((4; 40) and Figure 6 in (24)). See Figure 2 for 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 with constitutively expressed CRISPR gRNA gene products introduced in (41) 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).

The inducible and constitutively expressed gRNA parts were combined into circuits that act as sensors. For example, OR logic is realized when the absence of both inducers is associated with the absence of fluorescence while the presence of either inducer causes the production of GFP. In other words, the OR

circuit acts as a sensor to signal the presence of one or both molecules.

1.3 Major contributions

The assessment of circuit performance was based on FC data. The FC samples frequently expressed bimodal distributions spanning the ON and OFF fluorescence states. Given the lack of resources available to repeat the experiments, a machine learning method called Binary Event Prediction was developed to separate each bimodal distribution into an OFF distribution and ON distribution. The sample was then classified as primarily ON or OFF depending on which distribution had greater mass. Using this technique, useful information was extracted from suboptimal data.

After separation of bimodality, the analysis showed that most of the builds responded with better performance than a null model. However, the performance of the engineered switches was not consistent with the CDM predictions of which parts would lead to high-performing circuits and which parts would lead to low-performing circuits, likely due to assumptions that were placed on the CDM method for this application. DSGRN predictions were partially fulfilled in that DSGRN NOR topology outperformed the simple NOR topology, but no appreciable difference existed between the DSGRN and simple OR topologies. There is, therefore, evidence that a more complex circuit may at times exhibit greater success across experimental conditions.

2. Methods

2.1 Design tools

DSGRN Design Interface. DSGRN is a theoretical framework (20), and Python package (29) has been used in multiple applications to fully characterize the behavior of genetic networks (42–44). DSGRN permits a user to comprehensively describe

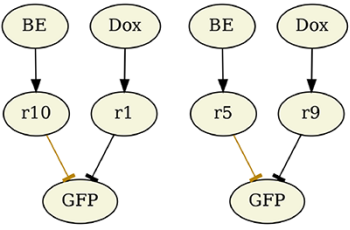
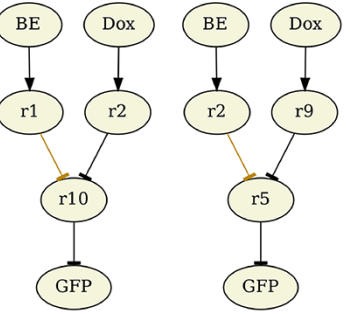
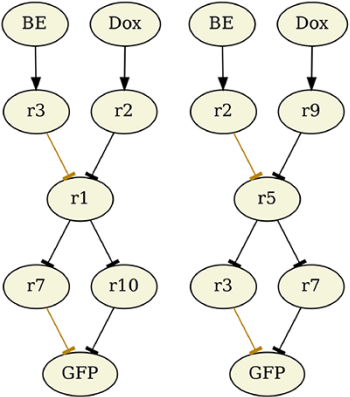
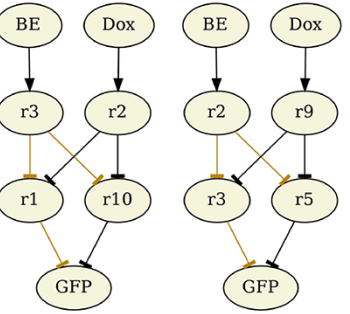
	NOR circuit		OR circuit	
	CDM low design	CDM high design	CDM low design	CDM high design
CDM score difference	1.49		2.54	
Simple topologies				
Topology Robustness Score	0.08		0.07	
CDM score difference	2.66		2.65	
DSGRN topologies				
Topology Robustness Score	0.22		0.28	

Figure 2. The designs for the built circuits, with one topology and two CDM designs in each quadrant. Pointed arrows \rightarrow denote an activating effect on transcription and blunt arrows \dashv denote a repressive effect on transcription. Upper left: The simple NOR topology published in (41), re-engineered with two different collections of guide RNA (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 (41), 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 interpretations.

the long-term dynamical behavior of a network topology. In the context of this paper, it predicts the equilibrium behavior of a logic circuit under DSGRN parameterizations. A DSGRN parameterization is actually a large region of high-dimensional parameter space across which equilibrium behavior is constant. This is a finite decomposition of high-dimensional parameter space, called the DSGRN parameter graph, so that DSGRN exhaustively predicts all behaviors available to a network topology. To predict the robustness of a dynamical behavior, DSGRN computes the percentage of the finite number of regions in the DSGRN parameter graph that exhibit the behavior of interest.

While powerful and flexible, DSGRN currently requires custom scripting to ask specific questions, such as the enumeration of the range of behaviors of a synthetic biology construct. A wrapper Python package `dsgnrn_design_interface` (38) was written to permit a plain language interface for a non-expert user to design a collection of feed-forward network topologies that satisfy a desired logic circuit behavior. Using this tool, thousands of network topologies can be screened for the desired logic behavior.

The input is a configuration file in JavaScript Object Notation (JSON) format, where the user specifies the desired circuit,

whether the inducible inputs are activated or repressed, the size of the network topology and the local logic of individual parts that are available to build the circuit (i.e. independent or dependent repressors or activators). The output is a collection of files accessible through an automatically generated Jupyter notebook that can be browsed for final choices of network topologies. An SBOL2 (28) formatted document can be created for any of the desired topologies. Comprehensive documentation is provided. Currently, the DSGRN Design Interface provides only limited access to the full functionality of DSGRN. An extension to support the design of larger circuits, such as those in (16), would be straightforward.

Given the user inputs, a DSGRN design parameter or set of parameters is identified that is consistent with the local logic of the parts. A successful output topology is one that exhibits the desired logic behavior at a design parameter. The neighboring parameters to the design parameters in the DSGRN parameter graph are well-defined and are associated with small malfunctions in the intended operation of the circuit. Each successful output topology is accompanied by a numerical score that ranks networks according to their predicted robustness based on these neighbors. For some successful network, let D be the number of design parameters with the correct equilibrium behavior, let N be the number of neighbors of all design parameters, let N_c be the number of these neighbors that exhibit the correct logic and let E be the number of essential DSGRN parameters (42). Then the topology robustness score (e.g. Figure 2) assigned to the network is a value between 0 and 1 defined by

$$S = \frac{D + N_c}{E + N}.$$

Under this scoring paradigm, a network with a million parameters and $S = 0.5$ is ranked the same as a network with a thousand parameters and $S = 0.5$. This normalization over parameter space permits the direct comparison of networks with vastly different complexity. See (30) for a similar scoring method in another design application of DSGRN.

Combinatorial Design Model. Experiment and circuit designs are often additive, namely, single conditions are combined to elicit a collective response. For a circuit constructed of multiple parts, the challenge is to select the best combination of parts that produces the best response (e.g. the largest dynamic range). While there are tools that exist to support this task, such as Cello (16), they require a thorough characterization of every part (e.g. a titration of inducer for each part) to construct the circuit. The CDM is a neural-network-based model that is trained with a subset of conditions and predicts the left out conditions (31; 32). The conditions need not cover a full range of titrations, but the predictions will then be limited to only the combinations of inducer concentrations and gates tested. For this effort, ON/OFF conditions for each part were tested and the FC distributions were predicted over all combinations of inducers and parts (e.g. Figure S9 in the Supplementary Data).

Currently, the best use of CDM is to have tested a full topology and only swap out parts. In this manuscript, we side-stepped that assumption by analytically combining independent parts predictions into a normalization procedure. An end-to-end model would have returned results more consistent with the original intent of CDM. The original usage case for CDM also does not consider the order of parts, so swapping the positions of $r1$ and $r2$ does not impact CDM results. The same normalization procedure was used here to overcome this assumption.

The normalization procedure altering CDM output for full topologies is reported in detail in [Supplementary Section 6 \(Figures S10–S20\)](#). Briefly, CDM predictions are combined in multiple stages, each stage aligned with a given topology. For example, the six-node DSGRN NOR topology with two input nodes, an intermediate node and two nodes with promoters at the GFP coding region had a three-stage model corresponding to each of these layers. At each layer, a measure of deviation of the predicted fold change was computed, with details of normalization varying at each stage. Since CDM scores change with topology size, it is more comparable to look at differences between high and low scores as in [Figure 2](#), rather than to interpret absolute scores.

2.2 Build tools

Aquarium. For plasmid and yeast construction, Aquarium software (14; 37) was used to manage inventory, experimental workflows and protocol execution. The input to Aquarium is a workflow or a linked set of protocols. During the execution of a workflow, data and metadata are logged and tracked to a database, which is used for future automated planning. Aquarium workflows are executed by technicians following on-screen instructions generated by Aquarium. Occasionally, the same protocol is executed by multiple technicians at the same time or sequentially (one technician leaving and another starting where the other left off). Technicians were trained beforehand, but had various skill levels and experience. During the execution of the protocols for the build steps and parts library dosage-response experiments conducted at the UW BIOFAB (45), technicians were unaware of project goals and were instructed to exactly follow the steps and instructions provided by Aquarium.

Terrarium. There are two inputs to the computer-assisted process planning software Terrarium (36): metadata from previously run Aquarium workflows and a biological design encoded in a BMF using the JSON standard. A BMF defines three components: (i) the laboratory configuration, (ii) relationships between biological sub-components in a given design and (iii) biological definitions, which include the type of biological sample (yeast, bacteria, DNA, sequences, etc.). Terrarium uses lab inventory information from the BMF plus metadata from previously executed Aquarium workflows to generate a model of the laboratory. This enables predictions for lead time, costs, labor and predictions of experimental errors for proposed workflows that permit the creation of an optimized Aquarium workflow.

DASi. In some cases, the sequence provided in a BMF is not available in the laboratory as inventory. In these cases, DNA sequences can be created in the laboratory by a combination of DNA sub-cloning or DNA synthesis from outside vendors. For this type of planning, Terrarium uses a planning subroutine, DASi (33), to automatically generate an economical DNA cloning assembly plan from sequences, using available lab inventory whenever possible. DASi has flexible input requirements for a DNA sequence and may be either a string of characters, a GenBank file, an SBOL file or a BMF. Unlike other cloning automation software, DASi does not require further design specifications beyond the DNA sequence to effectively create DNA assembly plans. Once an assembly is returned from DASi, the output file can be converted into a BMF that is processed by Terrarium into a validated Aquarium workflow.

2.3 Round Trip

DART relies upon the RT tool chain (24) to test assembled designs and then process experimental data. The RT accomplishes this by accepting a semi-structured Experimental Request (ER). The ER represents human-readable descriptions of the experiment and a set of structured tables that describe controls, builds, reagents, conditions and measurements. The RT uses a tool called the Intent Parser (46) to resolve human-readable terms to their representations in SBOL, including their design and assembly. The Intent Parser validates the ER and converts the experimental description into a machine-readable format that goes through experimental planning. The experimental planner, XPlan (13), partitions and allocates the requested builds, conditions and measurements to experimental runs that are launched at a laboratory with compatible software.

The RT processes the data through three main steps: (i) database ingest, (ii) standardization and versioning and (iii) automated analysis. The database ingest, a process called Extract-Transform-Load (ETL), checks that the measurements requested in the ER are fulfilled by each experimental run and then stores the data and metadata in a database called the data catalog. The Data Converge (DC) tool converts the data catalog records for each experiment into several standardized and versioned data products. These versioned products, such as data frames of log-transformed FC events or per sample absorbance measurements, provide a standard schema across experiments that use different builds, reagents and protocols. The standardization provides a common format, which enables generic analysis tools that can apply to many different experiments without extensive per-experiment customization. The versioning supports more effective analyst-to-analyst comparisons that are based upon the same version of the data. The RT uses a final tool called the Precomputed Data Table (PDT) (47) to apply a suite of analysis tools to the standardized dataframes. Each analysis tool (described in Methods) processes the data to pre-compute (rather than have the user manually compute) the resulting data. The resulting analyses, in conjunction with any *ad hoc* analyses performed by the user, can be used to inform additional experiments with the RT or inform new designs with the DA components.

In this work, the experimental planning tools automatically launched experimental runs at the Strateos (48) robotic cloud laboratory. Each run produced data that Strateos uploaded to the Texas Advanced Computing Center (TACC) (49) infrastructure for RT processing and analysis. TACC provided the Synergistic Discovery and Design Environment (SD2E; (50)) on which the various components of the RT are integrated. RT packages (and other Synergistic Discovery and Design (SD2) packages) are also freely available at (51).

2.4 Experimental process

Build complexity. The genetic circuits required a series of sequential genomic events to integrate the genetic parts into the yeast strains, where each event is called a 'layer'. Each layer required the construction of a new integrant (either linear or plasmid DNA), followed by a yeast transformation event, a selection event (selecting the right transformants from the transformed yeast on antibiotic or auxotrophic media) and finally a quality control (QC) validation step (polymerase chain reaction (PCR) using a primer specific for the genomic DNA just outside the insertion location and validating that the size of the fragment DNA is correct). It took about 1–2 working weeks to complete each step, depending on whether the integration was successful and

the availability of Aquarium research technicians. To save on build time, sequential transformations were completed in haploid strains and the circuit was assembled by 'mating' alpha and A strains to create a final diploid strain. The mating itself required a selection step, creating the need for at least two other selection markers to select for the new diploid strain (for example, the A strain may contain a HIS⁺ marker and the alpha may contain a LEU⁺ marker; selection will occur with HIS⁻ and LEU⁻ media). Part of build complexity is that there were a limited number of selectable markers available (four auxotrophic markers that are very good selectable markers: LEU, HIS, TRP and URA and three antibiotic markers that are fairly poor selectable markers, in that it is difficult to get the correct concentration of antibiotic to eliminate false positives and false negatives when selecting for colonies on a plate).

Beyond the build time and selection marker limitations, there was also the issue of DNA similarity. A number of unexpected and unsuccessful integration events were encountered during this and other projects that used the CRISPR gRNA parts. The hypothesis is that this is due to the extreme similarity between the parts, i.e. different promoters and gRNAs that are similar by ~2 kb and only differ by ~60 bp. As more and more highly similar synthetic parts in the haploid strain are incorporated, the likelihood of successful integration event validated by QC seemed to decrease as the number of layers increased, possibly because the process relied on homologous recombination to accomplish the genomic integration; the more highly similar DNA available in the genome, the more likely the DNA integrate at the wrong locus. Because of this known difficulty, three different build plans for each circuit were created with shuffled transformation order for the haploids before diploid mating. Most of the time, only one of these build plans was successful.

Plasmid construction. Backbone and insert fragments were amplified with PCR, gel extracted, purified and assembled using Gibson assembly (52) using standardized assembly linkers. Backbones contained a high-copy *Escherichia coli* origin of replication and ampicillin resistance for propagation. The yeast expression cassettes were flanked upstream and downstream by approximately 500 bases of chromosomal homology to the yeast genome and PmeI restriction sites for linearization before transformations. Plasmids were sequence-verified using Sanger sequencing.

Build construction. Yeast builds were constructed using genomic integration from linearized DNA into the CEN.PK113-7D strain of *S. cerevisiae*. Integrative plasmids were linearized using PmeI digestion (37°C, 30 min) to cut upstream and downstream of the chromosomal homology. Unpurified, linearized DNA was transformed into yeast cells using a standard lithium acetate protocol (53). Build selection was performed on solid synthetic-complete (SC) using auxotrophic or antibiotic markers. Diagnostic colony PCR was performed to verify integration into the proper locus. Builds were picked from single colonies and stored long-term at -80°C in a sterilized 30% glycerol and media solution. Build retrieval was performed by plating glycerol stocks onto solid media plates (YPAD) grown for 2–3 days at 30°C and picking single colonies for liquid culture. All yeast cultures and assays were grown at 30°C with shaking at 275 RPM. Builds consisted of both individual inducible parts and full circuits. Individual parts were characterized via dosage-response FC experiments to confirm functional response to inducer presence.

Plate layout for circuits. Each 96-well plate had eight data rows, one for each build. The columns had five different inducer concentrations, two replicates for each, as well as two columns for controls. Each plate had one of three media: standard (Synthetic Complete, SC), rich (Yeast Extract Peptone with 2% Dextrose, YEP) or slow (Synthetic Complete with 1% Sorbitol, SB) and one logical input transition (see Tables S1–S2 in the [Supplementary Data](#)).

Automated Laboratory. The Strateos robotic cloud lab (48) provides three main functions to DART: (i) the development of replicable, robotically executed experimental protocols, (ii) the execution of requested experimental runs and (iii) the delivery of the measurement data to the SD2E platform on TACC.

The protocols developed as part of this and related projects are variations of high-throughput screening with 96-well plates. Each protocol involves commanding a robotic workcell to incubate samples overnight and then dilute, apply reagents and gather a time series of plate reader and FC measurements. Strateos executes experimental runs that it receives from the RT. Each run references a container that includes the samples that include the various designs developed by DA components. Users ship these containers to Strateos and create container metadata describing each sample. The RT's XPlan planner allocates the requested samples in each ER to the container aliquots using this metadata. The Strateos platform validates each requested experimental run, constructs a sequence of Autoprotocol (54) instructions and informs the RT of the run identifier. After completing the protocol, the Strateos platform uploads the experimental data and metadata referenced by the run identifier. The RT then ingests the run data as described above.

2.5 Analysis tools

Precomputed Data Table. The PDT is a component of RT and is responsible for executing a suite of analyses on data products generated by DC (briefly described in Round Trip) and storing and versioning the respective results. The PDT provides DART with (i) rapid, consistent analysis of data of different types and from different experiments and (ii) the ability to automatically add additional data features (results from analyses) for higher-level analyses and modeling. These two features enable an increase in the rate at which DART can be iterated as actionable results are available within hours of data availability and more advanced analyses can be applied sooner. The PDT contains several types of analyses and not all are executed on all available data due to incompatibility of analysis types and data types. The next three subsections are descriptions of the analyses that the PDT was programmed to execute for the experiments at hand. The full suite of analyses is listed in (24).

Binary Event Prediction. The Binary Event Prediction (BEP) tool (47; 55) employs the random forest classifier implemented in the Python package scikit-learn (56) to predict whether an event within a FC sample corresponds to a 'high' or a 'low' signal, in the same sense as digital logic. Properly interpreting FC data collected from cells expressing a fluorescent protein requires the use of a positive and a negative control. In the case of the current experiments, the positive controls are cells that constitutively express GFP and the negative controls are cells that do not express GFP due to the lack of the GFP coding sequence. Using these controls, BEP develops the classifier for each pair of high/positive and low/negative controls. The tool trains the classifier on all FC events for the high/positive and low/negative controls, labeling each respective

event as either 0 (low) or 1 (high), and then uses it to predict for each event within each sample whether the event is low or high.

Compared to techniques that define a linear threshold to a single FC channel measuring GFP to separate low and high, BEP constructs a multidimensional nonlinear threshold (represented as a random forest). The classifier not only subsumes the threshold method but also incorporates all FC channels, such as forward scatter, side scatter and other color channels. The inputs to BEP are all FC channel measurements for each event and the output of BEP is the proportion of events per sample that are predicted to be ON vs OFF.

In the redesign effort described above, bimodality was observed in GFP-positive cytometer channels for the positive and negative controls. This caused ambiguity in determining what events are high and low by the classifier as subpopulations of events in one control were consistent with subpopulations in the opposite control. BEP was used to overcome this bimodality in the positive and negative controls. The poor control data were culled using the classifier's training/test split of the controls during cross-validation to identify the probability of each label predicted in the test set of each cross-validation fold. The control data were then 'cleaned', where for each control type, a threshold on probability was applied that dropped events found below the threshold but maintained a minimal total event count of 10,000 events. In this way, the model assesses the probability that each control data point (event) should be labeled according to the control group from which it came and retains only the most probable, i.e. those points that the model determines have a high probability of being correctly labeled. A new classifier was constructed on the remaining 'clean' data and used to predict signals for the non-control samples. If a non-control sample contained more than 50% predicted high events, then every event predicted low by BEP was dropped from the sample population. This modified population was then used in downstream analysis. A similar procedure occurred for distributions with more than 50% predicted low events.

Performance Metrics. Performance Metrics (PM) (57) measures the performance of experiments where the samples should be in two different states (e.g. ON and OFF states) and should have a separation in experimental measurements between these two states. The user defines how to aggregate samples (e.g. by build, condition and time) and the experimental measurement (e.g. fluorescence), and the tool computes a suite of metrics that compare the measured values for the two states. The tool returns metrics of how large the separation between the two defined states are for each group in the aggregation.

The metrics compare the distributions of aggregates of samples in each state (ON versus OFF) to compute the differences (ON–OFF) and ratios (ON/OFF) between the two states. The tool compares the left- versus right-hand side of the distributions via percentiles and standard deviations at different thresholds. For example, PM will aggregate samples by build, condition and time point and then for each group, it computes the difference between the 50% percentile of the ON samples and the 50% percentile of the OFF samples. It then computes per-sample metrics by comparing each individual sample in one state to the opposite state's distribution. For example, for each group, for each individual ON sample, it computes the difference and ratio to the median of the OFF states and then vice versa for each of the OFF samples.

PM reads in a configuration file and data and metadata files. The configuration file specifies the column name of the experiment output, information about the states (e.g. ON and OFF, or any other identifier), as well as a dictionary of what columns to use

to make aggregates (e.g. group by build and group by build, condition and time). The per-sample metrics for aggregates of build, condition and time can be used as inputs by Data Diagnosis (DD).

Data Diagnosis. Data Diagnosis (DD) (58) performs diagnostic tests for both experimental design and experimental performance. These test for variables that are associated with variations in performance and identify which values of the variables are associated with good or bad performance. Additionally, our tests identify if there appears to be any dependence between variables, for example, two variables contain the same information and should not be treated separately.

DD has two tests for performance and one test for dependence. Our first performance test is the Kruskal–Wallis H-test for categorical variables. This completes a non-parametric analysis of variance for the categorical variables grouped by a user-selected variable. The continuous variable analysis uses the Spearman correlation coefficient to measure the association between the performance variable and each continuous variable. Finally, there is the dependence test for randomization. This is primarily to study whether the experimental design has redundancies or has missed combinations of variables. It should be run prior to the experiment but can also be run afterward to aid in analysis and troubleshooting of the experiment. This test determines if the dependent variables were properly randomized by running a Chi-Squared Test for Independence between all pairs of categorical features. If the experiment was properly randomized, then none of the pairs should be dependent. Together all of these tests provide researchers valuable information on how their experiment performed and most importantly if certain variables are associated with good or poor performance.

DD reads in a configuration file and data and metadata files. The configuration file specifies the column name of the performance values (in this case, coming from PM), as well as a dictionary of what columns to use to perform analysis on subsets of the metadata.

Parameter value estimations and ODE modeling. A data-fitting algorithm using a Nelder–Mead minimization method (59) was implemented to determine the Hill function (60) 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 FC data distributions from the dosage-response experiments used to train CDM. The experimental data were fitted to Eq. 1 (for activations) and Eq. 2 (for repressions) derived from Cello (16):

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

$$y_{i_{SS}} = y_{i_{min}} + (y_{i_{max}} - y_{i_{min}}) \frac{1}{\left(\frac{y_{i-1_{SS}}}{\kappa_i^{n_i}}\right)^{n_i} + 1}, \quad (2)$$

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 Eqs. 1 or 2). Using the Hill function parameter value estimations a resulting ODE model is then analyzed using the Runge–Kutta–Fehlberg (4,5) method (61)

implemented in iBioSim (62) to obtain steady-state output predictions for each design under different input concentrations (shown in Figures S7–S8 in the Supplementary Data).

3. Results

3.1 Predictions

DSGRN predictions of circuit topology performance. The DSGRN Design Interface provides robustness scores for a sample of network topologies exhibiting the desired logic within a user-specified size range. Higher robustness scores correspond to predictions of greater success across experimental conditions. In principle, the robustness score can achieve a maximum of 1, but in reality this score is highly improbable. In Figure 2, the DSGRN NOR topology has a score of 0.22, more than twice the simple NOR topology with a score of 0.08. Similarly, the DSGRN OR topology has a 4-fold score of 0.28 compared to the simple design at 0.07. According to these predictions, the DSGRN topologies should exhibit better performance than the simple topologies over condition space, which consists of media conditions and inducer concentrations in the experiments performed here.

It is important to realize that DSGRN robustness scores are not to be taken as quantitative predictions. The robustness scores for the OR topologies would naively indicate that the DSGRN topology should successfully show the correct logic four times as often as the simple topology. However, that is at best an indicator of relative performance across all condition space (empirical numerical evidence for this can be found in (63)), not the limited condition space that is available experimentally. Therefore, only ordinal interpretation of the robustness scores is justified.

It is not always advantageous to build the circuit topology with the highest robustness score. As robustness increases, the redundancy in the network and therefore the complexity of the network is also increasing. There is a trade-off between predicted robustness and the ease of building the circuit. For example, there were two NOR designs with the same number of nodes as the DSGRN topology in Figure 2 with a score of 0.30 that were rejected due to doubts about build feasibility. This decision involved the number of parts of the build, i.e. the number of edges (promoters) in the network. The number of nodes is directly controlled through a parameter choice in the DSGRN Design Interface. The number of edges is not directly controlled but is influenced by a choice of the maximum number of promoters permitted at each node. This flexibility in the number of edges is useful, since directly limiting the number of edges may result in an inability to locate any network with the desired dynamics.

Other networks were rejected due to the number of 'layers' in the network. A layer is a collection of nodes that are designed to be induced at the same time. For example, in Figure 2, the simple NOR topology in the upper left is a single layer network; the inducer and GFP nodes are separated only by a single set of simultaneously (in theory) induced nodes. Similarly, the OR topologies in the right column are double-layer networks, and the DSGRN NOR topology in the lower left is a triple-layer topology.

As the number of layers increases, the build complexity of the circuit increases due to the need for sequential genomic transformations required to complete the genetic circuit within the yeast strains (see Methods for further discussion). Not only do the number of layers increase build complexity, they also increase the total time required for experiments, since the induction signal takes longer to propagate through the network. We also remark that the total number of parts, which tends to increase with the number of layers, increases the metabolic burden on the cell. The feasible

number of layers is domain-specific knowledge depending on the organism and the nature of the parts.

CDM predictions of various parts assignments. Given a circuit topology, it remains to assign parts to each node in the network. This was accomplished using a modification of the CDM, a neural network model designed to use partial data to predict the performance of genetic constructs across untested experimental conditions. The data used to train the model were 14 FC dosage-response curves, a BE-inducible and Dox-inducible version for each of the seven gRNA promoter regions evaluated in this study: r1, r2, r3, r5, r7, r9 and r10. Like all machine learning methods, sufficient data are needed to perform model training. FC data are ideal for this usage case given their tens of thousands of data points. CDM can be naturally used to predict the performance of simple NOR circuits; that is, it predicts the outcome of combining two inducible parts together when information is only known about individual parts. It is not explicitly designed to choose parts to construct larger circuits and so assumptions had to be made in order to apply it. First, it was assumed that the independent CDM scores for NOR units could be combined in a way that leads to an accurate prediction of whole circuit behavior. Second, the model was trained on data for inducible parts only, but there were both inducible and constitutively expressed parts in the parts library. It was assumed that the rank ordering induced by CDM scores on combinations of inducible parts was preserved for the combinations of constitutively expressed parts.

Using CDM predictions, two parts assignments were made for each circuit topology: one that was expected to be a better performer and another that was expected to be a poorer performer. These parts assignments are shown in [Figure 2](#), along with the difference between the high and low design CDM scores. While not interpretable in a quantitative manner, the difference in CDM scores are roughly comparable across circuit topologies. Again, only ordinal rankings are appropriate in the interpretation of these performance scores. As an example, compare the 1.49 difference in low and high simple NOR designs to the 2.54 difference in simple OR designs. The interpretation is that a larger difference between low and high designs should be seen in the simple OR topology versus the simple NOR topology. As a side effect of parts assignment, CDM was also able to reduce the intervals of inducer concentrations to be tested.

3.2 Data Overview

The eight builds in [Figure 2](#) were grown in three media types, YEP 2% Dextrose (rich media), Synthetic Complete (standard media) and Synthetic Complete containing 1% Sorbitol (slow growth media), which were assumed to present a nontrivial range of cellular conditions. The three media will be referred to as YEP, SC and SB, respectively. The cultures were used to create fifteen 96-well plates, with five plates per media condition. Each plate had titrated inducer concentrations for one or both of BE and Dox, see [Table S1](#) in the [Supplementary Data](#). The five titration experiments were BE titration into base media, Dox titration into base media, BE and Dox simultaneous titration into base media, Dox titration into BE spike-in media and BE titration into Dox spike-in media (the term spike-in means that the inducer was added to the media before the incubation period).

FC data were collected at three time points: 12, 24 and 36 h after an incubation period of 16 h. These conservative times were chosen based on previous experience with the constitutive CRISPR gRNA parts and circuit constructs from [\(41\)](#) for time to full induction. In [Figure 5](#) of that paper, induction times are shown for

variable-length linear cascades of NOT gates. From the figure, one can estimate that a 3-layer cascade takes about 12 h to reach half-maximal response. Above and beyond these experiments, there may be a further delay in the circuits pictured in [Figure 2](#) due to the BE- and Dox-inducible parts. Moreover, it is unclear if the reported induction times for linear cascades are conserved for more complex circuits with multiple interaction points in each layer. In addition, experiments preceding this one using circuit constructs from [\(41\)](#) showed inconsistency in cell growth after the freeze/thaw cycle, which necessitated longer recovery times. Therefore, sample time points were chosen up to 36 h to (hopefully) ensure full induction, GFP decay and cell growth for all circuits.

An event refers to a single FC measurement, and a sample is the collection of events associated to a well (a fixed location on a plate) at a single time point. The presence/absence of the inducers determines the expected outcome of the GFP signal in the FC measurements for the OR and NOR circuits; see [Table S2](#) in the [Supplementary Data](#) for a key to the media and expected outcomes of the circuits for each plate.

3.3 Data Processing

The RT provided standardized FC data products for visualization and analysis. Plots of these data showed significant bimodal distributions in both the OR and NOR builds, see [Figure 3](#) for an example plate. A possible explanation for bimodality in the data is that induction times for the circuits are slow, and therefore there are populations of both sufficiently and insufficiently induced cells.

However, there is also bimodality in the positive and negative controls, see the first column in [Figure 3](#) as well as the first row of [Figure 4](#). In the positive controls, this suggests potential mutations where the GFP coding region has been compromised. In the negative controls, this suggests possible contamination of the plates with GFP-producing cells. Given these issues with the controls, it would be ideal to redo the experiments.

In many situations, it is not feasible to redo experiments either due to lack of resources or inability to obtain more samples. This is the case for the data presented here; another set of experiments was not possible to obtain. The challenge was to see how much information could be drawn from data with anomalous controls. One solution would be to presume that the GFP-positive population in the negative control is uniform throughout the plate and subtract that portion of the population from all of the wells. This approach has the disadvantages that (i) the assumption of uniform contamination across wells may be incorrect and (ii) it has only one channel out of 16 to judge the identity of problematic events, a substantial reduction in the amount of available information. Our approach is to instead use prediction-confidence of a fitted model that was trained to discriminate between positive and negative controls, which used all available channel data.

To address these issues, a machine learning technique referred to as BEP [\(47\)](#) was developed to separate every bimodal distribution into low and high (or OFF and ON) distributions on a per-plate basis using 16 FC channels, each with width, height and area measurements. BEP uses a random forest classifier in two stages, an initial stage that 'cleans' the bimodal controls and then a second stage that classifies each FC event as either low or high GFP. In the first stage, the pooled replicates of the positive and negative controls on a plate are used to train a random forest classifier. The trained classifiers, one per plate, produce prediction probabilities for each event in the pooled positive and negative controls. Those

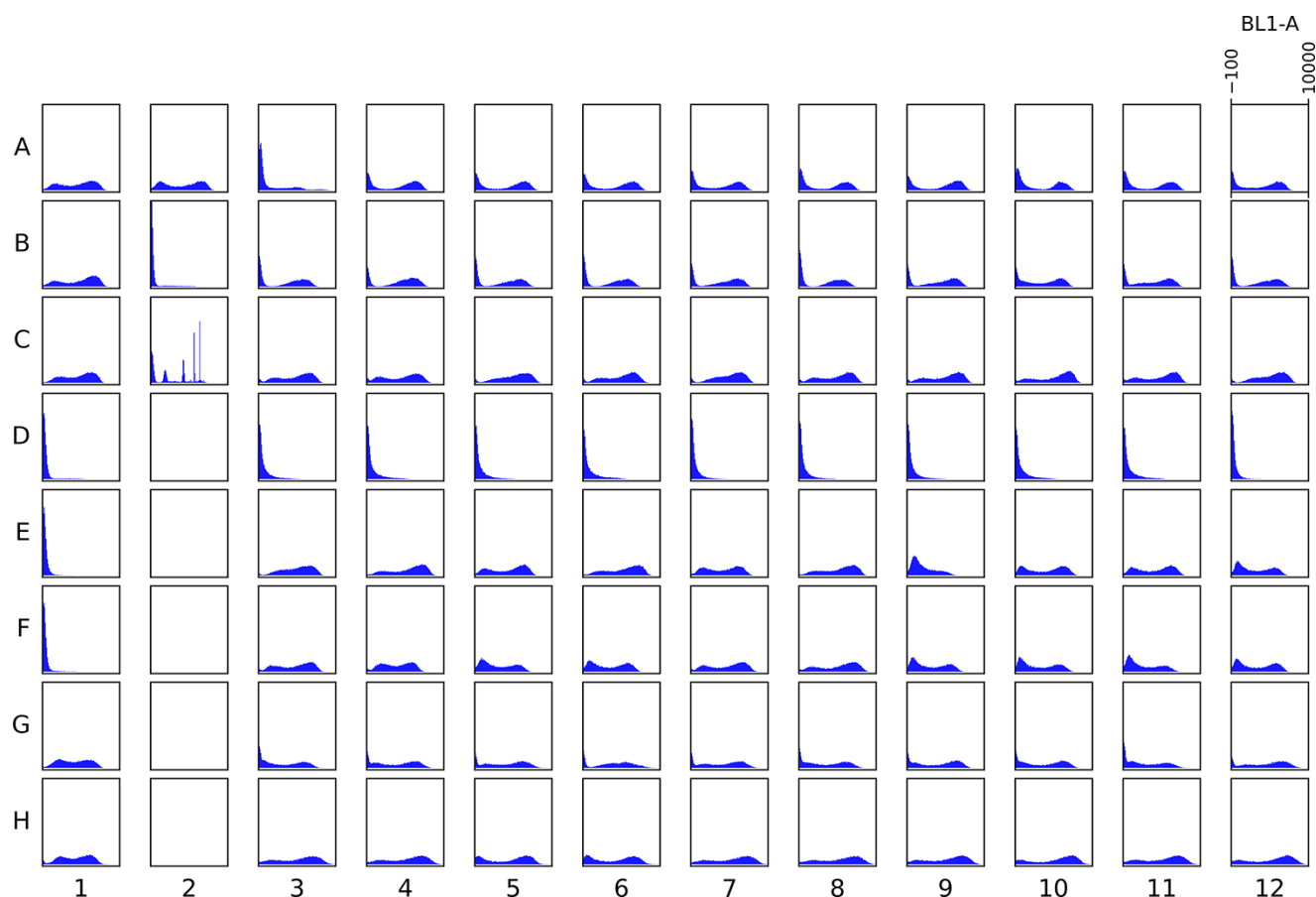


Figure 3. An example 96-well plate in SC media (plate id *r1fqsfkwxccc*v6, see [Table S2](#)) sampled 12 h post-incubation showing fluorescence area distributions in arbitrary units. Controls (positive, negative and bead) are in the first two columns. The remainder of the columns are data for the eight builds, one per row, with varying inducer concentrations.

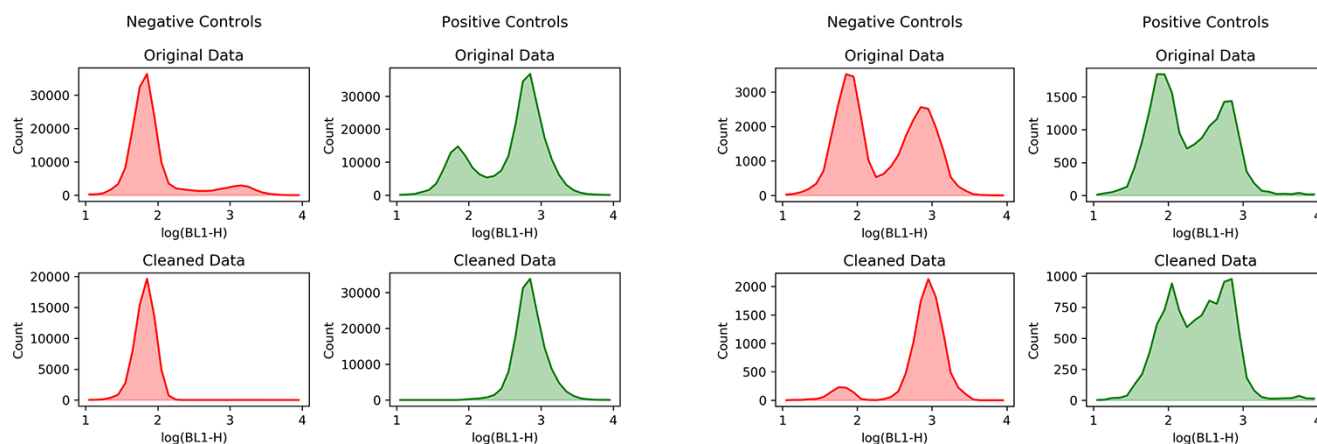


Figure 4. Examples of effective (left two columns) and ineffective (right two columns) cleaning of controls after stage one of BEP. Effective cleaning results in unimodal distributions, a low one for the negative control and a high one for the positive control. Ineffective cleaning either does not eliminate bimodality or does not result in the low (high) distribution being associated with the negative (positive) control. Negative controls are shown in red (first and third columns) and positive controls are shown in green (second and fourth columns). The histograms are GFP height measurements in log arbitrary units (horizontal axis). Top row: Unmodified controls pooled over replicates. Bottom row: Cleaned controls pooled over replicates.

events that exceed an adaptive probability threshold (see [Methods](#)) for either the high distribution in the positive control or a low distribution in the negative control were kept, producing ‘clean’ controls that frequently achieved reduced bimodality; see [Figure 4](#) for examples of effective versus ineffective cleaning. [Figure S1](#) in the [Supplementary Data](#) shows the fluorescence distributions for

the original and cleaned controls for each of the 15 plates in the experiments. On the left are nine plates with effective cleaning of the controls, i.e. they have achieved unimodality after cleaning, and on the right are the remaining six plates that show ineffective cleaning. Overall, there was substantial improvement from bimodal to unimodal distributions in the controls. The number of

plates with suitable positive and negative controls nearly doubled, and almost all of the positive controls achieved unimodal distributions. Given this performance, using a random forest classifier is a viable avenue for cleaning problematic bimodal data.

After the controls were cleaned, the second stage of BEP used these clean controls to train a new random forest classifier that was applied to the test data, e.g. the remaining wells on the plate that encompassed the experimental conditions. The result was two pools of events for each sample, those predicted to be ON and those predicted to be OFF. The mass of these two distributions was compared, and the distribution with the greater mass was taken to be the BEP-cleaned version of the experimental data. Subsequent analyses used this cleaned dataset.

The BEP methodology drew conclusions using all 16 channels and all three time points of the FC data without assuming a uniform distribution of contamination. As is intuitive, the GFP channel was the most important contributor to the model according to SHapley Additive exPlanations (SHAP) values (64), see Figure S2 in the Supplementary Data.

A comparison was performed between the original log-transformed data (all events) and the subset of cleaned data identified by BEP in order to assess BEP performance. Both the original and modified datasets were analyzed via the RT tools (i) PM to assess fold change between expected OFF and expected ON circuit states and (ii) DD to assess variability in the fold change as function of experimental variables. PM computes the fold change in median between all samples that are expected to be ON according to circuit function and all of those that are expected to be OFF. The distributions of the median fold change per plate for the original and modified datasets are shown in Figure S3 in the Supplementary Data. The distributions are seen to be very similar, indicating that in aggregate the fold changes between circuit ON and circuit OFF states are not improved by the BEP technology. However, DD shows that even though the average fold change hovers around 1 for both datasets, the BEP-cleaned data have many more outliers at higher fold changes (≥ 5) than the original data; compare Figures S4 and S5. This indicates that circuit performance measured as fold change is greatly improved for a number of samples.

Moreover, as will be demonstrated in the next section, the BEP event predictions of ON versus OFF state permit the evaluation of circuit performance, which is otherwise problematic when the fold change between expected ON and OFF distributions is approximately 1.

3.4 Data Analysis

Circuit performance was assessed by examining the three time point FC distributions for GFP for each well in a plate. Each FC distribution was required to meet conservative criteria for cell presence in order for a well to be assessed for success. The total number of events in the distribution had to be at least 10 000, and there had to be a cell density of at least 500 000 cells/ml. If any of the three time points did not meet these criteria, then the well was classified as a failure, with the following exception: one plate (r1fw4stb38ewsh, see Table S2 in the Supplementary Data) with SB media had such widespread insufficiency in cell density and number of events that the whole plate was excluded from analysis. This plate was one of the plates with ineffectively cleaned controls (see first row, right columns in Figure S1). The remaining 14 plates consist of five plates for each of SC and YEP media and four with SB.

Successful performance was assessed as a ‘match’ between the observed behavior of the circuit, ON or OFF, and the ON/OFF

behavior predicted by the digital logic the circuit was designed to produce. Each well in a plate was designated as an overall success or failure using information from all three time points. Since BEP exhibited variable performance on a per-plate basis, successes for each build were aggregated on the plate level. A *plate performance score* for a build is the number of successes on the plate divided by 10, since each build was measured under five inducer conditions with two replicates on each plate.

It remains to describe what constituted a match. The observed behavior of the circuit was classified as ON, OFF or indeterminate (called the *observed state*) based on the relative masses of the ON and OFF distributions for each sample that were predicted by BEP. For each time point, there was a proportion of events in the FC distribution that was predicted by BEP to be ON (p) or OFF ($1 - p$), where p will be called the BEP ratio. The algorithm for assigning an observed state to a well had two parts. First, if a well satisfied $p > 0.6$ ($p < 0.4$) at all three time points, then the build was declared to be exhibiting ON (OFF) behavior. Additionally, if neither condition above was met but p increased (decreased) over time, then the build was said to exhibit ON (OFF) behavior. This latter condition was chosen since the hypothesis that some of the builds may still have been undergoing the temporal processes of induction and GFP decay over the course of the experiment could not be excluded. Any well that did not meet one of these criteria was said to have indeterminate behavior.

A success for a well was defined to be any case where the observed state matched the intended ON or OFF state of the circuit given the presence or absence of the inducers, otherwise it was classified as a failure (see Table S1 for the logical inputs corresponding to each combination of inducer concentrations). In particular, any sample with an indeterminate observed state was classified as a failure.

The 14 plate performance scores form the distributions plotted in Figure 5(a). The category labels indicate the topology, circuit and CDM prediction in order. Median performance shows a wide range of values and the interquartile ranges are generally large; therefore, it is unclear whether or not any of these circuits can be declared successful on the whole. To address this, a baseline distribution was created where each plate had its BEP ratios permuted over all eight builds and three time points 1000 times. The baseline plate performance score was computed for each circuit at each iteration and the distributions are shown in Figure 5(b). Although there is an overlap in the interquartile differences, the means and medians of the true scores for DSGRN NOR, DSGRN OR/CDM low designs and simple OR topologies are all higher than those of the randomized empirical null.

In Figure 5(c), the distributions in Figure 5(a) are split over media condition. The distributions only have 4–5 points each, but the boxplot shows suggestive patterns. First, the DSGRN NOR topologies perform either better or similarly to the simple NOR topologies in all three media conditions, with particularly high performance in the SB media condition. Second, the underperformance of the DSGRN OR/CDM high design seems to be due primarily to the SC media. Disappointingly, no build seems to exhibit similar, and therefore reproducible, performance across media conditions. The closest is the simple OR/CDM high design.

To provide some statistical quantification of the observed trends, the probability that a new observation will be a success for a given build and media condition was estimated using a logistic regression model (Table S4 and Figure S6 in the Supplementary Data). Figure 5(d) shows these predicted probabilities along with a 95% confidence interval for the true probability of success. Where the confidence intervals do not overlap, that is suggestive of,

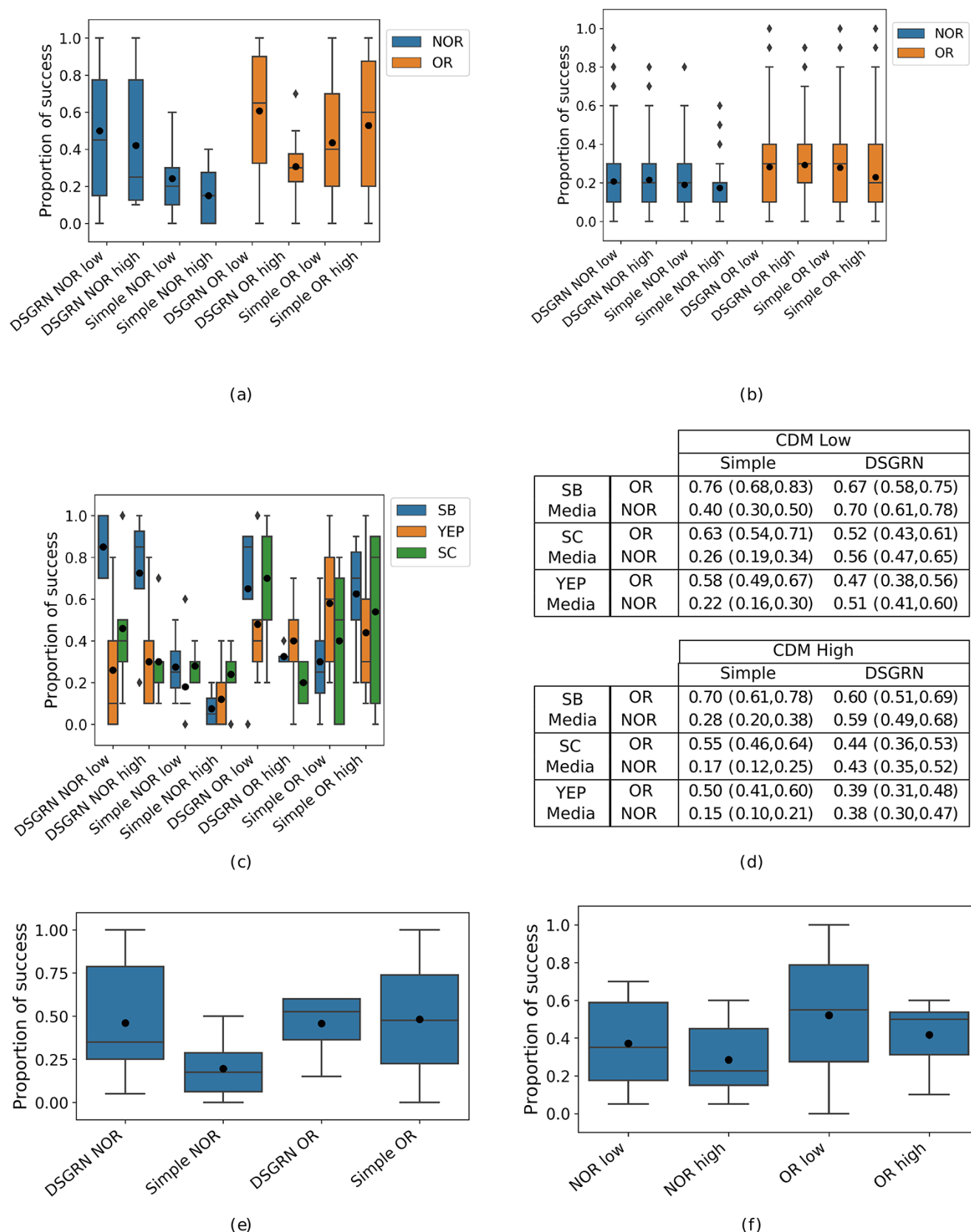


Figure 5. (a) Distributions of 14 plate performance scores (proportion of successes per build per plate) across the eight designs. The vertical bars indicate the last datapoint within a window of 1.5 times the interquartile distance with outliers as diamonds. The rectangles show 25–75 quartiles, with the horizontal bar showing the median and the black dot showing the mean. (b) Baseline distributions for plate performance scores; each plate had its BEP ratios permuted over all wells and time points 1000 times. (c) Plate performance scores split by media (five plates each for SC and YEP and four plates for SB). (d) Predicted probabilities of success for each build with 95% confidence intervals for the true probabilities of success using a logit model (see Section 4 in the Supplementary Data). (e)–(f) Distributions of plate performance scores pooled across media and CDM design (e) to assess DSGRN predictions, and scores pooled across media and topology (f) to assess CDM predictions.

though not a guarantee of, significantly different probabilities of success. Given a media condition and circuit, the only time there is non-overlapping confidence intervals, and therefore possible significance, is between the simple NOR topologies with lower predicted success and DSGRN NOR topologies with higher predicted success. This can be seen by choosing a NOR circuit row and pairwise comparing the confidence intervals between the simple and DSGRN topologies. Remarkably, the DSGRN NOR topology outperforms the simple NOR topology for all media conditions and both builds. By examining the top half and bottom half of the table, it can be seen that the CDM low designs have higher probabilities of success than the CDM high designs, but there is overlap in the confidence intervals.

The accuracy of the DSGRN robustness predictions and the CDM performance predictions are further assessed by pooling media conditions and builds together, as shown in Figure 5(e)–(f). In Figure 5(e), the plate performance scores for the high and low CDM builds are pooled to compare the DSGRN topologies against the simple topologies. The DSGRN NOR topology outperforms the simple NOR topology, indicating that that circuit redundancy may indeed produce more robust performance, as also supported by the statistical analysis in Figure 5(d). On the other hand, the two OR topologies are not substantially different performers (in mean and median). However, Figure 5(a) and (d) show that the DSGRN OR designs show a large differential performance from each other, whereas the two simple OR designs do not. A possible conclusion is that the DSGRN OR topologies are not as reproducible as the simple OR designs in terms of the specific biological parts deployed, which is counter to DSGRN predictions. In Figure 5(f), the plate performance scores for the DSGRN and simple topologies are pooled to compare the CDM low-performance predictions to the CDM high-performance predictions. The CDM low designs show a trend of higher performance than CDM high designs, consistent with Figure 5(d).

The simple NOR design is an anomalously poor performer compared to all other designs. This is interesting because all the other builds contain similar NOR gates, although they use different gRNA parts (see Figure 2). 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.

In general, the Hill model predicted that circuits should respond more strongly to Dox (Figure 6(a)), but that the dosage–response to BE was acceptable (Figure 6(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(c)). Figure 6(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 a.u. (arbitrary units), 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. See Figures S7 and S8 in the [Supplementary Data](#) for comprehensive predictions across all builds.

Figure 6(b)–(c) shows the differential Hill model predicted performance of the DSGRN and simple NOR topologies using parameters for the CDM low designs. In this case, BE successfully

represses the signal of the DSGRN NOR design to the 1500–2000 a.u. low steady state but does not suppress that of the simple NOR design. In the absence of both inducers (first bar), both designs are predicted to exhibit a high fluorescence signal, but in the subsequent titrations, the simple design is predicted to have a GFP signal more than twice the low steady state. The BE-inducible parts r10 and r5 are used only in the simple NOR designs and not in the other designs (Figure 2). One cannot exclude the hypothesis that the inclusion of the BE-inducible parts r5 and r10 into the three other topologies would also result in degraded performance.

4. Discussion

This manuscript introduces a set of tools for the design of circuit topology through performance prediction and a set of tools to improve the efficiency and accuracy of building synthetic circuits. Together this DA toolchain is melded with a previously published toolchain, the RT. During the DA portion of this sequence, OR and NOR logical circuit topologies and parts assignment were designed using the predictive tools DSGRN (29; 38) and CDM (32) and built using the laboratory software tools DASi (33; 34), Terrarium (36) and Aquarium (37). The RT (24) portion of the toolchain enhanced the reproducibility of experimental results through the automation of experimental specification, subsequent data handling and standardized analysis.

The built circuits were two variants each of networks that should exhibit OR and NOR logic: the standard, simple OR and NOR topologies, plus an OR and a NOR topology predicted by DSGRN to show robustness across experimental conditions. Although the simplest topologies for OR and NOR circuits are well-known to synthetic biologists, there are many circuit topologies that also exhibit the correct logical behavior. DSGRN predicts that many OR and NOR circuit topologies with redundancy should show more consistent and accurate performance across experimental conditions. DSGRN scoring is designed to be agnostic to organism chassis or parts choice and therefore does not incorporate constraints specific to the yeast chassis or gRNA regulators. User-chosen constraints of the DSGRN Design Interface include the number of logic gates, the number of binding sites per gate, the logic function to be realized at each gate (in this case NOT and NOR) and a threshold for a robustness score. These constraints together reduce the number of considered networks to a number amenable to manual inspection that takes advantage of subject matter expertise. A large lesson learned in this effort is that subject matter expertise cannot simply be encoded by automation in a generalizable manner. Once the desired topologies were identified, CDM predicted groupings of parts that should show better versus worse circuit performance for each topology.

During data inspection, it became clear that contamination, of the negative controls with GFP-producing cells and of the positive controls with mutated cells lacking GFP production, was a real concern—appearing as strong bimodality in the FC distributions. However, there were insufficient resources to repeat this suite of experiments and no guarantee that the issues would be resolved through replicated experiments.

A major thrust of this paper and of the SD2 program that funded this study is to address the question: To what extent can all collected data, regardless of quality, be used to draw conclusions and suggest further experimentation? It is a challenging goal to develop methods that can use all data, whether optimal or sub-optimal, to inform future experiments. To this end, a machine learning technique called BEP (47) was developed to separate

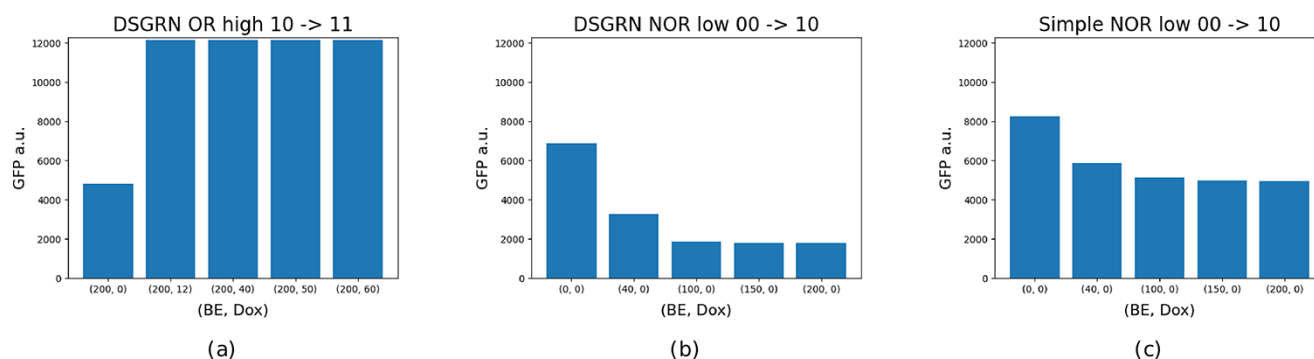


Figure 6. Example predicted steady-state values of the geometric mean of the FC distribution of GFP a.u. from Hill function models. Horizontal axis: Inducer concentrations in nM in the order (BE, Dox). Vertical axis: Steady-state GFP values in arbitrary units as predicted by the parameterized Hill models. (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).

bimodal distributions based on 16 FC channels. The intent was to identify true ON/OFF versus false ON/OFF events in order to assess circuit performance. Aside from this, BEP provided extra information that FC channels correlated with cell size were not very important to the classification (see Figure S2 in the Supplementary Data). After treatment with BEP, the number of plates with suitable positive and negative controls nearly doubled, and almost all of the positive controls were cleaned from bimodal to unimodal high distributions (see Figure S1), permitting the cleaning of the experimental condition data.

The simple and DSGRN NOR circuit performance exhibited a trend consistent with DSGRN predictions of circuit robustness, namely that a circuit design with redundancy may indeed be more robust with regard to diverse experimental conditions. However, there are mixed conclusions to be drawn for the OR circuit. On the one hand, DSGRN OR topologies show a smaller interquartile range, consistent with greater average reproducibility. On the other hand, the CDM-predicted high and low builds of the DSGRN OR topology show a much greater difference in median than the analogous builds for the simple OR designs, indicating lower reproducibility.

CDM predictions were generally not fulfilled. This is likely due to faulty assumptions when creating a purpose-built version of CDM for this application. Either the scores for combinations of NOR genetic subunits were incorrectly calibrated or the assumption that genetic parts with gRNA binding sites exhibited similar behavior to their BE- and Dox-inducible counterparts was incorrect. An alternative future route is to combine DSGRN topology generation with stochastic parts assignment optimization, such as that in Cello (15), or with the Wasserstein metric-based parts optimization in (17).

The trends in performance are suggestive rather than significant. These trends can be used to hypothesize the most promising directions for future experimentation. Possible avenues of exploration involve the choice of topology, particularly for the OR circuit, the choice of parts assignment, particularly for the simple NOR designs, the choice of experimental conditions or the experimental platforms and protocols used. From the work here, better experimental controls and different assignments of parts are clear choices. A wider range of parts assignments and experimental conditions would lead to a better exploration of the operational envelope of the construct and would therefore put

higher confidence in the topology rankings produced by DSGRN. Lastly, alternative topologies for OR logic could provide further evidence for or against increased robustness of performance when comparing structural redundancy to build complexity.

In conclusion, synthetic logic circuits predicted to exhibit OR and NOR logic behavior were designed, built, observed using FC and analyzed for performance using a toolchain designed to enhance robustness and reproducibility. A machine learning technique was developed and trained on all measured FC channels to mitigate unexpected experimental complications. The results indicate that more complex circuits exhibiting structural redundancy may provide more robust and reproducible behavior in a synthetic biology setting despite increased build difficulty.

Supplementary data

Supplementary data are available at SYNBIO Online.

Data availability

GenBank accession numbers for the strain sequences are in Table S3 in the Supplementary Data. All flow cytometry fcs files were uploaded to flowrepository.org under experiments FR-FCM-Z5SA, FR-FCM-Z5S9, FR-FCM-Z5SC, FR-FCM-Z5SD, FR-FCM-Z5SF, FR-FCM-Z5SH, FR-FCM-Z5SW, FR-FCM-Z5SJ, FR-FCM-Z5SN, FR-FCM-Z5SP, FR-FCM-Z5SQ, FR-FCM-Z5SR, FR-FCM-Z5SS, FR-FCM-Z5SE, FR-FCM-Z6YJ, FR-FCM-Z5S5 and FR-FCM-Z5S7. Processed data and scripts used in the generation of figures in this manuscript are available in the public repository <https://gitlab.com/breecummins/2022-sd2-yeaststates-data-analysis.git>. The code packages cited in the manuscript are all open source.

Material availability statement

No materials are available.

Funding

This work was supported by Air Force Research Laboratory and Defense Advanced Research Projects Agency (DARPA) contracts HR001117C0092, HR001117C0094, HR001117C0095, FA875017C0184, FA875017C0229, FA875017C0231 and FA875017C0054 as

part of the Synergistic Discovery and Design program. This document does not contain technology or technical data controlled under either US International Traffic in Arms Regulation or US Export Administration Regulations. Views, opinions and/or findings expressed are those of the author(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the US Government. Approved for public release, distribution unlimited (DISTAR Case #36 307).

Author contributions statement

Conceptualization was performed by B.C., J.V., R.C.M., M.G., T.G., K.M. and S.B.H. Data curation was performed by B.C., J.V., R.C.M., A.D., D.B., T.J., M.W., G.Z., J.N., C.J.M., J.B., B.B., T.M., T.T.N. and N.R. Formal analysis was performed by B.C., R.C.M., A.D., P.F., T.J., M.W., G.Z., K.L.J. and R.P.G. Funding acquisition was performed by D.B., M.W.V., C.J.M., T.G., K.M. and S.B.H. Investigation was performed by J.V., R.C.M., J.N., V.B., T.R.H. and L.A.M. Methodology was performed by B.C., J.V., R.C.M., H.E., A.D., P.F., F.C.M., D.B., M.E., K.L.J., M.G. and J.B. Project administration was performed by D.B., J.B., M.W., M.W.V. and S.B.H. Resources were analyzed by J.V., J.N., N.G., M.W.V. and J.U. Software were analyzed by B.C., J.V., R.C.M., H.E., A.D., P.F., F.C.M., D.B., T.J., M.W., G.Z., M.E., S.G., R.P.G., C.J.M., M.G., N.G., J.U., J.B., B.B., T.M., T.T.N. and N.R. Supervision was performed by D.B., J.B., M.W., T.G., K.M. and S.B.H. Validation was performed by J.V., A.D., M.W., G.Z. and J.N. Visualization was performed by B.C., R.C.M., H.E., A.D., P.F., M.W. and K.L.J. Writing of original draft was performed by B.C., J.V., R.C.M., A.D., P.F., D.B., M.E., K.L.J. and C.J.M. Writing, review and editing were performed by B.C., P.F., D.B., F.C.M., M.E., R.P.G., J.B., T.G., K.M. and S.B.H.

Competing interests. Some of the authors are employed by companies that may benefit or be perceived to benefit from this publication.

References

- Del Valle, I., Fulk, E.M., Kalvapalle, P., Silberg, J.J., Masiello, C.A. and Stadler, L.B. (2021) Translating new synthetic biology advances for biosensing into the Earth and environmental sciences. *Front. Microbiol.*, **11**, 3513.
- Khalil, A.S. and Collins, J.J. (2010) Synthetic biology: applications come of age. *Nat. Rev. Genet.*, **11**, 367–379.
- Cameron, D.E., Bashor, C.J. and Collins, J.J. (2014) A brief history of synthetic biology. *Nat. Rev. Microbiol.*, **12**, 381–390.
- Cummins, B., Moseley, R.C., Deckard, A., Weston, M., Zheng, G., Bryce, D., Gopaulakrishnan, S., Johnson, T., Nowak, J., Gameiro, M. et al. (2022) Computational prediction of synthetic circuit function across growth conditions. *10.1101/2022.06.13.495701*.
- Marchisio, M.A. and Stelling, J. (2009) Computational design tools for synthetic biology. *Curr. Opin. Biotechnol.*, **20**, 479–485.
- Villalobos, A., Ness, J.E., Gustafsson, C., Minshull, J. and Govindarajan, S. (2006) Gene designer: a synthetic biology tool for constructing artificial DNA segments. *BMC Bioinform.*, **7**, 1–8.
- Swainston, N., Dunstan, M., Jervis, A.J., Robinson, C.J., Carbonell, P., Williams, A.R., Faulon, J.-L., Scrutton, N.S., Kell, D.B. and Kelso, J. (2018) PartsGenie: an integrated tool for optimizing and sharing synthetic biology parts. *Bioinformatics*, **34**, 2327–2329.
- Czar, M.J., Cai, Y. and Peccoud, J. (2009) Writing DNA with GenoCAD™. *Nucleic Acids Res.*, **37**, W40–W47.
- Misirli, G., Hallinan, J.S., Yu, T., Lawson, J.R., Wimalaratne, S.M., Cooling, M.T. and Wipat, A. (2011) Model annotation for synthetic biology: automating model to nucleotide sequence conversion. *Bioinformatics*, **27**, 973–979.
- Huynh, L., Tsoukalas, A., Köppe, M. and Tagkopoulos, I. (2013) Sbrone: A scalable optimization and module matching framework for automated biosystems design. *ACS Synth. Biol.*, **2**, 263–273.
- Park, Y., Espah Borujeni, A., Gorochowski, T.E., Shin, J. and Voigt, C.A. (2020) Precision design of stable genetic circuits carried in highly-insulated *E. coli* genomic landing pads. *Mol. Syst. Biol.*, **16**, e9584.
- François, P. and Hakim, V. (2004) Design of genetic networks with specified functions by evolution in silico. *Proc. Natl. Acad. Sci.*, **101**, 580–585.
- Kuter, U. et al. (2018) XPLAN: Experiment planning for synthetic biology. In: Pascal B, Daniel H, Susanne B and Ron A (eds). *ICAPS Workshop on Hierarchical Planning*.
- Vrana, J., de Lange, O., Yang, Y., Newman, G., Saleem, A., Miller, A., Cordray, C., Halabiya, S., Parks, M., Lopez, E. et al. (2021) Aquarium: open-source laboratory software for design, execution and data management. *Synth. Biol.*, **6**, ysab006.
- Jones, T.S., Oliveira, S., Myers, C.J., Voigt, C.A. and Densmore, D. (2022) Genetic circuit design automation with Cello 2.0. *Nat. Protoc.*, **17**, 1–17.
- Nielsen, A.A.K., Der, B.S., Shin, J., Vaidyanathan, P., Paralanov, V., Strychalski, E.A., Ross, D., Densmore, D. and Voigt, C.A. (2016) Genetic circuit design automation. *Science*, **352**, aac7341.
- Schladt, T., Engelmann, N., Kubaczka, E., Hochberger, C. and Koeppl, H. (2021) Automated design of robust genetic circuits: Structural variants and parameter uncertainty. *ACS Synth. Biol.*, **10**, 3316–3329.
- Appleton, E., Madsen, C., Roehner, N. and Densmore, D. (2017) Design automation in synthetic biology. *Cold Spring Harb. Perspect. Biol.*, **9**, a023978.
- Yeoh, J.W., Swainston, N., Vegh, P., Zulkower, V., Carbonell, P., Holowko, M.B., Peddinti, G. and Poh, C.L. (2021) SynBiopython: an open-source software library for synthetic biology. *Synth. Biol.*, **6**, ysab001.
- Cummins, B., Gedeon, T., Harker, S., Mischaikow, K. and Mok, K. (2016) Combinatorial representation of parameter space for switching networks. *SIAM J. Appl. Dyn. Syst.*, **15**, 2176–2212.
- Yeung, E., Kim, J., Yuan, Y., Gonçalves, J. and Murray, R.M. (2021) Data-driven network models for genetic circuits from time-series data with incomplete measurements. *J. R. Soc. Interface*, **18**, 20210413.
- Yeung, E., Kundu, S. and Hodas, N. (2019) Learning deep neural network representations for Koopman operators of nonlinear dynamical systems. In: *2019 American Control Conference (ACC)*, IEEE, Philadelphia, PA, USA, pp. 4832–4839.
- Li, Z. and Yang, Q. (2018) Systems and synthetic biology approaches in understanding biological oscillators. *Quant. Biol.*, **6**, 1–14.
- Bryce, D., Goldman, R.P., DeHaven, M., Beal, J., Bartley, B., Nguyen, T.T., Walczak, N., Weston, M., Zheng, G., Nowak, J. et al. (2022) Round Trip: An automated pipeline for experimental design, execution, and analysis. *ACS Syn. Bio.*, **11**, 608–622.
- Jessop-Fabre, M.M. and Sonnenschein, N. (2019) Improving reproducibility in synthetic biology. *Front. Bioeng. Biotechnol.*, **7**, 1–18.
- Bryce, D., Goldman, R.P., DeHaven, M., Beal, J., Nguyen, T., Walczak, N., Weston, M., Zheng, G., Nowak, J., Stubbs, J. et al. (2020) Round-trip: An automated pipeline for experimental design, execution, and analysis. In: *Proceedings of the 12th International Workshop on Bio-Design Automation (IWDA-20)*, pp. 29–30.
- Roehner, N., Beal, J., Clancy, K., Bartley, B., Misirli, G., Grünberg, R., Oberortner, E., Pocock, M., Bissell, M., Madsen, C. et al. (2016) Sharing structure and function in biological design with SBOL 2.0. *ACS Syn. Bio.*, **5**, 498–506.

28. Roehner,N., Mante,J., Myers,C.J. and Beal,J. (2021) Synthetic biology curation tools (SYNBIC). *ACS Syn. Bio.*, **10**, 3200–3204.
29. Gameiro,M. (2022) DSGRN Software. <https://github.com/marciogameiro/DSGRN> (31 December 2021, date last accessed).
30. Gameiro,M., Gedeon,T., Kepley,S. and Mischaikow,K. (2021) Rational design of complex phenotype via network models. *PLoS Comput. Biol.*, **17**, e1009189.
31. Eslami,M., Borujeni,A.E., Doosthosseini,H., Vaughn,M., Eramian,H., Clowers,K., Gordon,D.B., Gaffney,N., Weston,M., Becker,D. et al. (2021) Prediction of whole-cell transcriptional response with machine learning. *bioRxiv*, <https://www.biorxiv.org/content/early/2021/05/01/2021.04.30.442142>.
32. Eramian,H. and Eslami,M. (2021) Combinatorial Design Model. <https://github.com/SD2E/CDM> (31 December 2020, date last accessed).
33. Vrana,J. (2021) DASi DNA Design. <https://github.com/jvrana/DASi-DNA-Design.git> (1 January 2020, date last accessed).
34. Vrana,J., (2021) DASi DNA Design Documentation. <https://jvrana.github.io/DASi-DNA-Design/> (6 May 2022, date last accessed).
35. Vrana,J., (2021) Software systems for automated manufacturing of engineered organisms. Ph.D. dissertation, University of Washington, copyright - Database copyright ProQuest LLC; ProQuest does not claim copyright in the individual underlying works; Last updated - 2021-11-24. <https://www.proquest.com/dissertations-theses/software-systems-automated-manufacturing/docview/2594492440/se-2?accountid=28148>.
36. Vrana,J., (2021) Terrarium. <https://github.com/jvrana/Terrarium.git> (6 May 2022, date last accessed).
37. Vrana,J., (2022) Aquarium: The Laboratory Operating System. <https://www.aquarium.bio/> (9 May 2022, date last accessed).
38. Cummins,B. (2021) DSGRN Design Interface Software. https://gitlab.com/breecummins/dsgrn_design_interface.git (31 December 2021, date last accessed).
39. Baggs,J.E., Price,T.S., DiTacchio,L., Panda,S., FitzGerald,G.A. and Hogenesch,J.B. (2009) Network features of the mammalian circadian clock. *PLoS Biol.*, **7**, e1000052.
40. Goldman,R.P., Moseley,R., Roehner,N., Cummins,B., Vrana,J.D., Clowers,K.J., Bryce,D., Beal,J., DeHaven,M., Nowak,J. et al. (2022) Highly-automated, high-throughput replication of yeast-based logic circuit design assessments. *bioRxiv*, <https://www.biorxiv.org/content/early/2022/06/01/2022.05.31.493627>.
41. Gander,M.W., Vrana,J.D., Voje,W.E., Carothers,J.M. and Klavins,E. (2017) Digital logic circuits in yeast with CRISPR-dCas9 NOR gates. *Nat. Commun.*, **8**, 15459.
42. Gedeon,T., Cummins,B., Harker,S. and Mischaikow,K. and You,L. (2018) Identifying robust hysteresis in networks. *PLoS Comput. Biol.*, **14**, e1006121.
43. Xin,Y., Cummins,B. and Gedeon,T. (2020) Multistability in the epithelial-mesenchymal transition network. *BMC Bioinform.*, **21**.
44. Fox,E., Cummins,B., Duncan,W. and Gedeon,T. (2022) Modeling transport regulation in gene regulatory networks. *Bull. Math. Biol.*, **84**, 1–42.
45. Fox,E., Cummins,B., Duncan,W. and Gedeon,T. (2022) University of Washington Biofabrication Center. <http://www.uwbiofab.org/> (6 May 2022, date last accessed).
46. Nguyen,T., Walczak,N., Sumorok,D., Weston,M. and Beal,J. (2021) Intent parser: A tool for codification and sharing of experimental design. *ACS Syn. Bio.*, **11**, 502–507.
47. Zheng,G., Moseley,R.C., Bryce,D., Deckard,A., Cummins,B., Goldman,R., Eramian,H., Johnson,T. and Weston,M. (2021) Pre-Computed Data Table. <https://github.com/SD2E/precomputed-data-table.git> (6 May 2022, date last accessed).
48. Zheng,G., Moseley,R.C., Bryce,D., Deckard,A., Cummins,B., Goldman,R., Eramian,H., Johnson,T. and Weston,M. (2022) Strateos, Cloud Lab Automation-as-a-Service. <https://strateos.com> (6 May 2022, date last accessed).
49. Zheng,G., Moseley,R.C., Bryce,D., Deckard,A., Cummins,B., Goldman,R., Eramian,H., Johnson,T. and Weston,M. (2022) Texas Advanced Computing Center. <https://www.tacc.utexas.edu/> (6 May 2022, date last accessed).
50. Zheng,G., Moseley,R.C., Bryce,D., Deckard,A., Cummins,B., Goldman,R., Eramian,H., Johnson,T. and Weston,M. (2022) Synergistic Discovery and Design Environment. <https://www.tacc.utexas.edu/research-development/tacc-projects/sd2e> (23 May 2022, date last accessed).
51. Zheng,G., Moseley,R.C., Bryce,D., Deckard,A., Cummins,B., Goldman,R., Eramian,H., Johnson,T. and Weston,M. (2022) Synergistic Discovery and Design Github. <https://github.com/SD2E> (23 May 2022, date last accessed).
52. Gibson,D.G., Young,L., Chuang,R.-Y., Venter,J.C., Hutchison,C.A. and Smith,H.O. (2009) Enzymatic assembly of DNA molecules up to several hundred kilobases. *Nat. Methods*, **6**, 343–345.
53. Gietz,R.D. and Woods,R.A. (2002) Transformation of yeast by lithium acetate/single-stranded carrier DNA/polyethylene glycol method. *Methods in enzymology*. Elsevier, **350**, 87–96.
54. Gietz,R.D. and Woods,R.A. (2022) Autoprotocol: An open standard for Scientific Experimental Design and Automation. <https://autoprotocol.org> (10 May 2022, date last accessed).
55. Bryce,D., Moseley,R. and Ladwig,J. (2022) Python SD2 Circuit Analysis Tool. <https://github.com/SD2E/pysd2cat.git> (6 May 2022, date last accessed).
56. Pedregosa,F., Varoquaux,G., Gramfort,A., Michel,V., Thirion,B., Grisel,O., Blondel,M., Prettenhofer,P., Weiss,R., Dubourg,V. et al. (2011) Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.*, **12**, 2825–2830.
57. Deckard,A. and Johnson,T. (2019) Performance Metrics. <https://github.com/SD2E/performance-metrics.git> (31 December 2021, date last accessed).
58. Deckard,A. and Johnson,T., (2019) Data Diagnosis. <https://github.com/SD2E/diagnose.git> (31 December 2021, date last accessed).
59. Nelder,J.A. and Mead,R. (1965) A simplex method for function minimization. *The Computer journal*, **7**, 308–313.
60. Santillán,M. (2008) On the use of the hill functions in mathematical models of gene regulatory networks. *Math. Model. Nat. Phenom.*, **3**, 85–97.
61. Fehlberg,E. (1969) “Low-order classical Runge-Kutta formulas with stepsize control and their application to some heat transfer problems,” National aeronautics and space administration. *Tech. Rep.* **315**, 00572.
62. Watanabe,L., Nguyen,T., Zhang,M., Zundel,Z., Zhang,Z., Madsen,C., Roehner,N. and Myers,C. (2018) iBioSim 3: A tool for model-based genetic circuit design. *ACS Syn. Bio.* **8**, 1560–1563.
63. Kepley,S., Mischaikow,K. and Queirolo,E. (2022) Global Analysis of Regulatory Network Dynamics: Equilibria and Saddle-Node Bifurcations. <https://arxiv.org/abs/2204.13739> (6 May 2022, date last accessed).
64. Lundberg,S. (2018) Shapley Additive Explanations (shap). <https://github.com/slundberg/shap> (1 March 2022, date last accessed).