AFRL-RI-RS-TR-2022-130

GENETIC CIRCUIT DESIGN FOR EXTREME ENVIRONMENTS ENABLED BY MODELS EXTRACTED FROM PETABYTE-SCALE PERTURBATION ANALYSES

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

SEPTEMBER 2022

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nations. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2022-130 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /
WILMAR W. SIFRE
Work Unit Manager

/ S /
GREGORY J. HADYNSKI
Assistant Technical Advisor
Computing and Communications Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
		START DATE	END DATE
SEPTEMBER 2022	FINAL TECHNICAL REPORT	AUGUST 2017	MARCH 2022

4. TITLE AND SUBTITLE

GENETIC CIRCUIT DESIGN FOR EXTREME ENVIRONMENTS ENABLED BY MODELS EXTRACTED FROM PETABYTE-SCALE PERTURBATION ANALYSES

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
FA8750-17-C-0229	N/A	61101E

5d. PROJECT NUMBER	5e. TASK NUMBER	5f. WORK UNIT NUMBER
		R2CR

6. AUTHOR(S)

Amin Espah Borujeni, Hamid Doosthosseini, Yuval Dorfan, Alexander J. Triassi, Daniel A. Anderson, YongJin Park, Christopher A. Voigt, Benjamin A. Garcia, Una Saxena, Alexander W. Cristofaro, David B. Gordon, Timothy S. Jones, Dany Fu, Samuel Oliveira, Elijah Jones, Rita Chen, Douglas M. Densmore, Chris J. Myers, Pedro Fontanarossa, Zach Zundel, James Scholz, Benjamin Hatch, Eric Yu

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Massachusetts Institute of Technology, Dept of Biological Engineering
21 Ames St #56-651
Cambridge MA 02139

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)	11. SPONSOR/MONITOR'S REPORT NUMBER(S)
Air Force Research Laboratory/RITA 525 Brooks Road Rome NY 13441-4505	AFRL/ RI	AFRL-RI-RS-TR-2022-130

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Design and testing of genetic circuits is a laborious process requiring multiple design-test-build cycles whose processes, metadata, analyses, and naming conventions vary widely across groups performing synthetic biology. To alleviate the low-throughput and data-discrepancy challenges, we developed tools and techniques for automating high-throughput standardization, design, modeling, and sharing of genetic circuits and computational tools. SBOL and SynBioHub to facilitate the standardization, storing, and sharing of biological designs and relevant synthetic terms. Cello to standardize circuit design and modeling, and utilize SBOL for standardization. A full RNASeq pipeline from quality control, to mapping, to analysis, that leads to reproducible transcriptomics to help inform design and evaluation. A Landing Pad Selector, Genetic Circuit Analyzer, Genetic Circuit Simulator, and Whole Genome Simulator to analyze and simulate design of circuits. The resulting work for this group led to both novel computational tools and novel organisms that can be used in designs.

15. SUBJECT TERMS

Transcriptomics, Metabolomics, Genetic Circuit Design, Strain Engineering, Dynamic Modeling, Stochastic Analysis, Computational Chemistry, Bayesian Optimization, Perovskites, Computer-Aided Design Tools, Liquid Handling Robots, High-throughput DNA Assembly Methods, Data Standardization and Storage.

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	C. THIS PAGE		
U	U	U	SAR	162

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code)*
WILMAR W. SIFRE	N/A

PREVIOUS EDITION IS OBSOLETE.

STANDARD FORM 298 (REV. 5/2020)
Prescribed by ANSI Std. Z39.18

TABLE OF CONTENTS

List of Figures

List of Tables

1 SUBGROUP 1: VOIGT LAB (MIT)

1.1 Summary

Creating functional genetic circuits is frustrated by a lack of accurate predictive modeling of circuit function and computational tools to identify possible circuit failures. This slows design and results in a need for many design-build-test cycles before a functional genetic circuit is built. In this work, we present a high-throughput data-driven parametrization and modeling approach based on omics data that overcomes these challenges. The computational tools developed are able to identify genetic components, parametrize their function, simulate gene circuit behavior, predict circuit dynamics, simulate noise propagation, measure circuit sensitivity, debug circuit failures, and identify optimal locations for circuit integration. In addition, we demonstrate the applicability of what we have learnt by engineering multiple new *Bacillus* strains as part of a pressure test, and a therapeutically relevant strain *E. coli nissle* with functional genetic circuits.

1.2 Introduction

Accurate modeling of translation requires an understanding of transcription and translation as well as protein binding and biomolecular mechanisms fundamental to genetic regulation. Despite modeling efforts for biomolecular systems, characterizations require extensive experiments for each individual component and models rely heavily on parameters obtained from literature, making them inaccurate and not applicable to new conditions or host strains. Identifying and characterizing components in a high-throughput and data-driven way using omics can overcome these limitations of genetic circuit modeling.

1.3 Methods, Assumptions, and Procedures

Transcriptomics experiments: RNA sequencing and ribosomal profiling were conducted in the Voigt lab, these methods collect mRNA and quantitatively measure the abundance of each region on the genome, where ribosomal profiling measures only those regions protected by a ribosome. End-enriched RNA sequencing was used to improve accuracy of data near transcription initiation sites. Further high-throughput RNA sequencing and all proteomics experiments were conducted at Ginkgo Bioworks.

All modeling was written in python, using common packages, and simulations were ran on iPython / Jupyter notebooks at TACC. In collaboration with the Myers group, the models were integrated into iBioSim and simulations were made available there.

1.4 Results and Discussions

1.4.1 Part performance evaluation and error discovery

We have performed a combination of RNA-seq and ribosome profiling (Ribo-seq) on a large-scale genetic circuit in *E. coli* DH10B as the host cell. The circuit consists of 7 gates and >50 biological parts (Figure 1). The primary goal of the project was to determine whether the genetic circuit performed as predicted by Cello (Nielsen et al., 2016), and also quantify the performance of all biological parts in the context of the genetic circuit. RNA-seq and Ribo-seq provide a bird's

eye view of the activity of RNA polymerase and ribosome (cellular expression machineries) across the circuit (Figure 2). More importantly, RNA-seq profiles confirm that genetic circuit performs as predicted by Cello (Figure 3).

In addition to evaluating the performance of genetic circuit as a whole, RNA-seq and Ribo-seq can be used to evaluate the performance of individual biological parts in the circuit. In particular, we can learn how promoters, ribozymes, ribosome binding sites (RBSs), protein coding sections, terminators, and genetic gates perform in the context of a genetic circuit. An example of such evaluations is shown in Figure 4 for one of the gates in the circuit.

Although both the circuit and its biological parts have performed as expected, we hypothesized that other types of unexpected failures and hidden errors may have been present across the circuit. Importantly, these hidden errors are silent in the condition that circuit was tested, but could be harmful to the circuit if activated in other conditions. To address this hypothesis, we scanned the RNA-seq and Ribo-seq profiles across the circuit, and found a collection of diverse errors which was quite surprising. These errors were both at transcription and translation levels, as shown in Figure 5. The most prevalent errors were cryptic promoters in both sense and antisense strands (Figure 6). For example, two of the repressors used in the circuit had one or two strong cryptic promoters inside their coding section. These cryptic promoters were so strong that they were active even when the upstream annotated promoter of the repressors was active. Another surprising finding was the presence of a cryptic promoter in antisense strand of all ribozymes in the circuit (Figure 7). It is noteworthy that cryptic promoters that were present in protein coding section (e.g. Figure 6) are naturally occurring errors which are carried by the repressor proteins and are not the side-effect of the engineering genetic parts. However, cryptic promoters in the ribozymes (e.g. Figure 7) were man-made errors and were the side-effect of engineering genetic parts and can be avoided in future designs.

A potential problem with cryptic promoters is the possibility of expressing undesired proteins from mRNAs originated from cryptic promoters. We can see this effect first-hand using RNA-seq and Ribo-seq profiles (Figure 8). HlyIIR protein has a naturally occurring antisense cryptic promoter. Ribo-seq showed that this cryptic promoter generated an mRNA that carries a strong RBS (Figure 8) that expresses an unknown protein. This was confirmed by the observed high ribosome occupancy in the antisense of HlyIIR.

Finally, all the gates in the circuit were previously designed with the premise that they operate orthogonal to the host cell. That means that they do not directly repress or activate native genes on the genome. This assumption was made by the fact that none of the repressors' operator binding sites were present in *E. coli* genome. However, what this assumption is not taking into account is the sub-optimal operator binding sites. That is, a repressor may bind to its sub-optimal binding sites that contain a few mismatches, if its concentration is high enough to overcome reduced affinity toward the sub-optimal binding sites. This phenomenon was observed for one repressor (HlyIIR), where we discovered three off-target genes across the genome (Figure 9). Analysis of DNA sequence around the promoter region of one of the off-target genes shows an example of sub-optimal binding site of HlyIIR (Figure 9). This work was published as an article in Nature Communications.

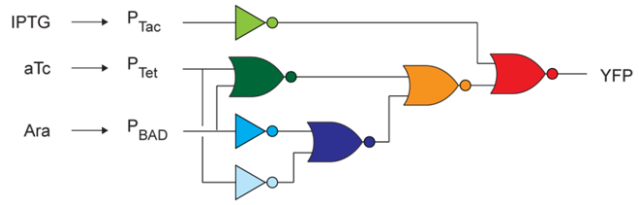


Figure 1. Circuit diagram of the genetic circuit used in this work.

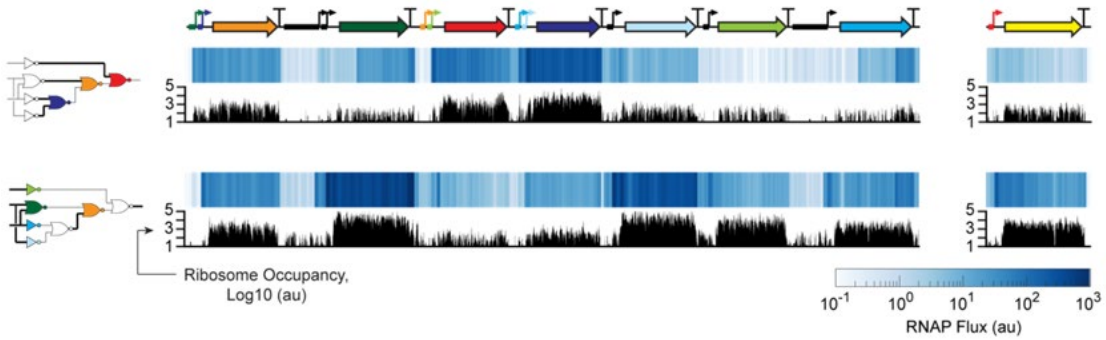


Figure 2. Transcription and translation profiles across a genetic circuit. The profile of RNA polymerase flux can be seen across genetic circuit using RNA-seq. Ribosome occupancy on each protein coding section corresponds very well with higher flux of RNA polymerase. Profiles are from two separate induction states of the circuit.

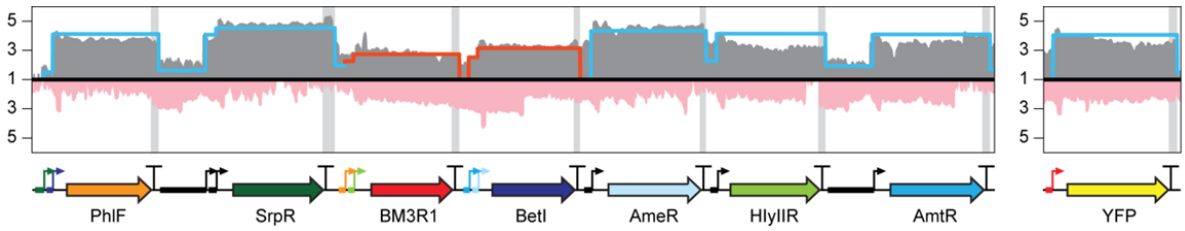


Figure 3. Validation of Cello using RNA-seq profile. RNA-seq confirms that the circuit expression follows Cello predictions as shown by blue and red lines.

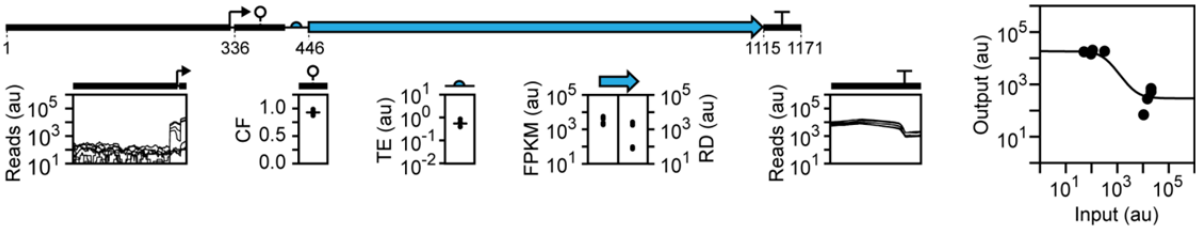


Figure 4. Full characterization of a gate's biological parts using RNA-seq and Ribo-seq. From left to right: 1- promoter activity and transcription start site are shown using RNA-seq, 2- ribozyme cleavage fraction are quantified using RNA-seq, 3- translation efficiency of RBS is quantified using Ribo-seq, 4- FPKM and ribosome density of each protein is calculated using RNA-seq and Ribo-seq, respectively, 5- terminator strength and transcription termination site are shown using RNA-seq, 6- input-output response of each gate is quantified using RNA-seq.

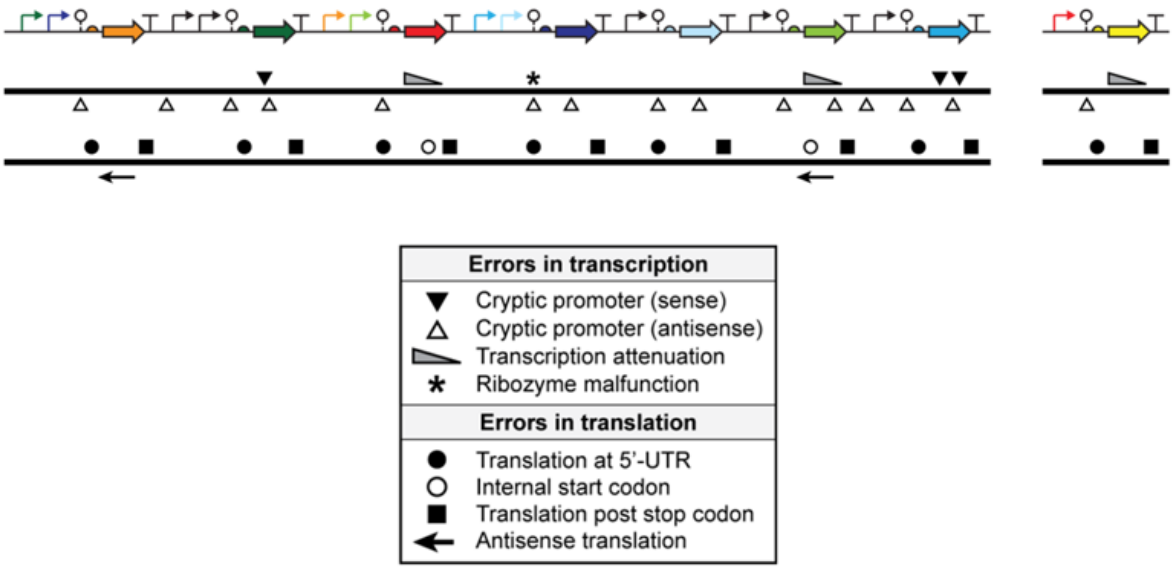


Figure 5. A map of all hidden errors identified across the genetic circuit. (Top) transcriptional errors (Bottom) translational errors. Legends are described in the box.

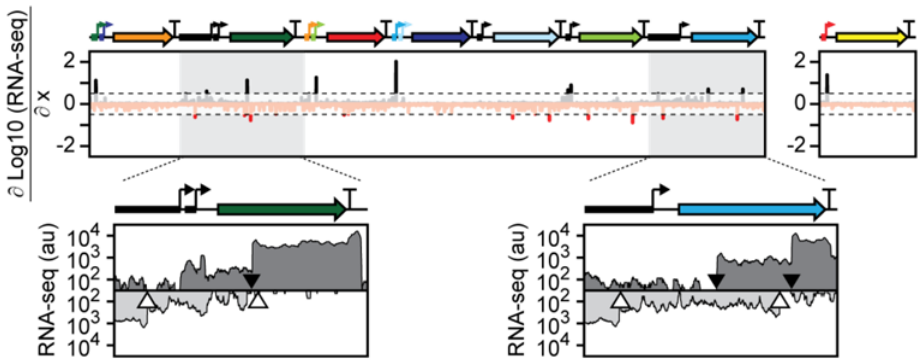


Figure 6. RNA-seq can identify cryptic promoters in both sense and antisense strands.

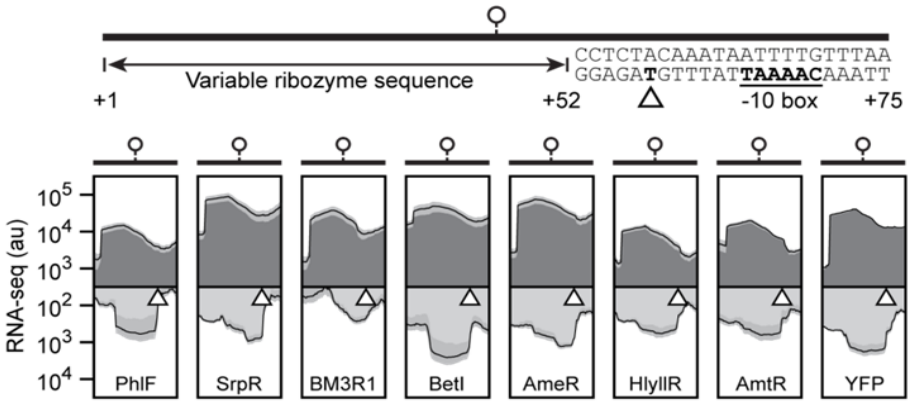


Figure 7. Cryptic antisense promoter inside ribozymes. A cryptic antisense promoter is hidden in all ribozyme sequences in the circuit, which causes about 5-fold reduction in sense transcription.

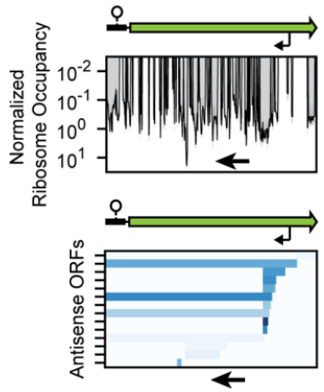


Figure 8. A cryptic antisense promoter leads to antisense translation. Ribo-seq profiles revealed a highly active antisense translation within the coding section of HlyIIR repressor, which is consistent with the predictions of RBS Calculator (bottom).

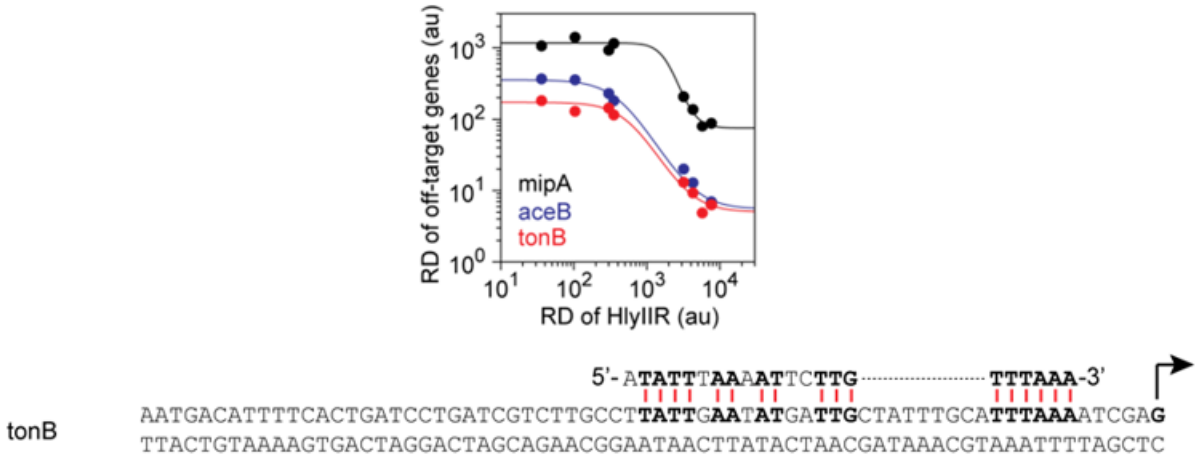


Figure 9. Off-target activity of HlyIIR repressor. RNA-seq showed that HlyIIR has off-target activity toward genes on the genome. Top plot shows the repression curves of three off-target gene on the genome. Bottom plot shows the potential binding site of HlyIIR on tonB promoter on the genome.

1.4.2 Identifying and parametrizing genetic components

Initially, a machine learning algorithm was developed to classify genetic components (promoters), i.e. to determine whether a short region of DNA either contains a promoter(+), or doesn't(−) using RNA sequencing data. A transcription profile is a mapping of RNA sequencing reads to genomic location at nucleotide resolution, see profiles in Figure 10 A in grey. Intuitively, a region containing a promoter has a profile that increases in value abruptly at the site of the promoter. So far, all research is focused on models to predict promoter strength from the DNA sequence. The DNA sequences of promoters vary in different strains because RNAP (the protein that recognizes them) is not the same. However, the behavior of all RNAP, i.e. its function that is to begin transcription, is consistent across all strains and manifests in RNA sequencing data. This means a tool to predict promoters from RNA sequencing data could be applicable across a variety of strains, even those that have not been well characterized or previously studied. As such, the goal is to train a model on a dataset from a well-studied strain and determine whether promoters can be identified in data from a different strain.

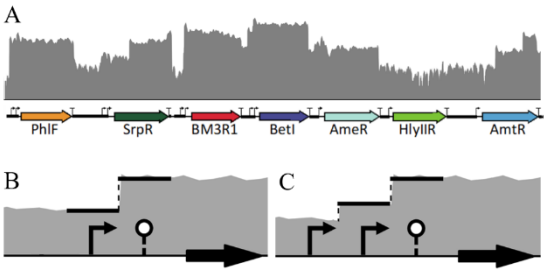


Figure 10. Transcription profiles showing curated RNA sequencing data (A) a synthetic DNA segment containing circuit elements and expected profiles around a single promoter (B) or tandem promoters (C)

Transcription profiles were produced from completed experiments in two strains (*E coli* and *B subtilis*) each with a genome larger than 4 million base pairs (bp). The *E coli* data was obtained from NAND 2.0 and the *B subtilis* was from Ginkgo of the 168 wild type. The data in its original format is sparse, however, classifying a region of 30 bp allows accurate determination of promoter locations. In the *E coli* data set, 1843 promoters are labelled of which 1757 have exact known locations. This renders 1757 data points with $(+)$ labels (each data point is the transcription profile of a region of 30 bp i.e. a vector of 30 features which are real numbers greater than or equal to zero). Across the rest of the genome, a further 10000 points are available with known $(-)$ labels. These points were selected at random with the constraints that (a) they be annotated and no promoter exist within them and (b) no two regions overlap. This creates an imbalanced dataset in two classes, with more $(-)$ labelled points than $(+)$ labeled points. Similarly for *B subtilis*, all 867 promoters are labelled of which 599 points have known locations. Thus 599 data points with $(+)$ labels and 10000 data points with $(-)$ labels are also available from that strain.

Not all the data points are usable. Often, in RNA sequencing experiments, certain RNA fragments are lost during the chemical processes involved and certain regions have low coverage. In other words, certain data points will have a feature vector of all zeros. The data was cleaned by removing regions with low coverage. Furthermore, each entire dataset was normalized to an average value of 1 read per bp which would allow comparable data across strains with different genome lengths. Data was not centered because a value of 0 should have the same connotation (i.e. no transcription) regardless of strain, and centering would render regions with value 0 to a different (negative) value for each strains. The data from each dataset was then organized into matrices.

The datasets are imbalanced and under sampling of the majority class would result in the elimination of a variety of dissimilar feature vectors that the model would never see. Oversampling the minority class could result in overfitting. As such, I chose to address the imbalanced dataset by modifying the objective function.

θ will be a 1×30 vector of weights and θ_0 will be a 1×1. The guess $g^{(i)}$ is obtained via the linear logistic classifier equation, and the objective function will be a weighted negative log-likelihood for a linear logistic classifier.

The weights w_+, w_- were calculated based on proportion of labels such that getting the same portion of each class wrong would result in the same loss. The objective function was not regularized because the features are a set of values at 30 consecutive base pairs, all values are relevant as all features are directly impacted by whether or not the region contains a promoter.

$$g^{(i)} = \sigma\left(\theta^T x^{(i)} + \theta_0\right) \quad (1)$$

$$J_{lr}(\theta, \theta_0; D) = \left(\frac{1}{n} \sum_{\{i=1\}}^{n} \widehat{L_{nll}}\left(g^{(i)}, y^{(i)}\right)\right) \quad (2)$$

$$\widehat{L_{nll}}\left(g^{(i)}, y^{(i)}\right) = \mathrm{w}_+ y^{(i)} \log\left(g^{(i)}\right) + \mathrm{w}_-\left(1 - y^{(i)}\right) \log\left(1 - \ g^{(i)}\right) \quad (3)$$

$$\mathrm{w}_+ = \frac{n_+ + n_-}{2n_+}\ , \mathrm{w}_- = \frac{n_+ + n_-}{2n_-} \quad (4)$$

$$\eta = \frac{1}{i^{0.1}} \quad (5)$$

Where n_+ is the number of positive-label data points and n_- is the number of negative-label data points, η is the step size and i is the iteration number. An initial test was done with 10 steps to find values such that the objective function would not oscillate. This was done manually.

In addition to the loss, two other measures are of interest, i.e. precision and recall. Precision describes the ratio of correctly classified positive-label points (true positives) to all data points classified as positive. Recall describes the ratio of correctly classified positive-label points (true positives) to all positive-labelled points. These are of importance since the hope is that, given a region of 30bp that is of interest, the model will let us know if it contains a promoter. If the model classifies it as containing a promoter, precision gives us a measure of how likely it is the model is right and recall gives us a measure of how many of the promoters given it will find. We expect recall to be low since a lot of regions may contain promoters that are not very active, where the features would be similar to a noisy non-promoter region. However, high precision is paramount for our goal.

The results presented below were obtained by training with *E coli* data and testing on the *B subtilis* data. The goal is to test whether a model trained on a well-characterized strain would render useful classification for a different strain. All codes were written in Jupyterhub on the sd2e platform.

The training was conducted until the change in loss was lower than 0.1%. It can be seen that as the model is trained on the training data, and the training loss decreases, the loss on the test data also decreases. This demonstrates that what the model is learning from the training data, is applicable to the test dataset.

The results demonstrate that the model achieves relatively high precision, 87.7% on the training dataset and 64.9% on the test data set. This is a significant finding since it shows that assuming an absolute cutoff of 0.5 for classification and disregarding how certain the model is about its decision, 64.9% of the regions in a new strain the model identifies as promoters, it does so correctly. To put this in context, the mechanistic model that is currently used in research in my lab has a precision of 35.4% and recall of 36.4%. A completely random coin-flip classification would give us a precision of 6.26%.

There is an evident trade-off between precision and recall; the more conservatively the model classifies as positive (containing a promoter) the lower the recall. A low recall is intuitively expected; cells contain promoters for a variety of functions which are inactive when the cell doesn't need that function (e.g. metabolizing a sugar that isn't present in its media). Therefore, a majority of promoters are expected to be inactive and recall is expected to be low. Further work to obtain experimental data from varying conditions (i.e. varying sugar molecules used) would be required to overcome this obstacle. It would be interesting to see if recall of the model can be improved by using multiple such datasets from the same strain. Figure 11 A shows the results of the trained model and B contains examples demonstrating the nuances of identifying promoter from RNA-seq data and potentially why the ML model outperforms previous models. This figure demonstrates the nuances and similarities between promoters and arbitrary noise from the experiment, this is a pitfall for previous models used and one reason the ML model preforms better.

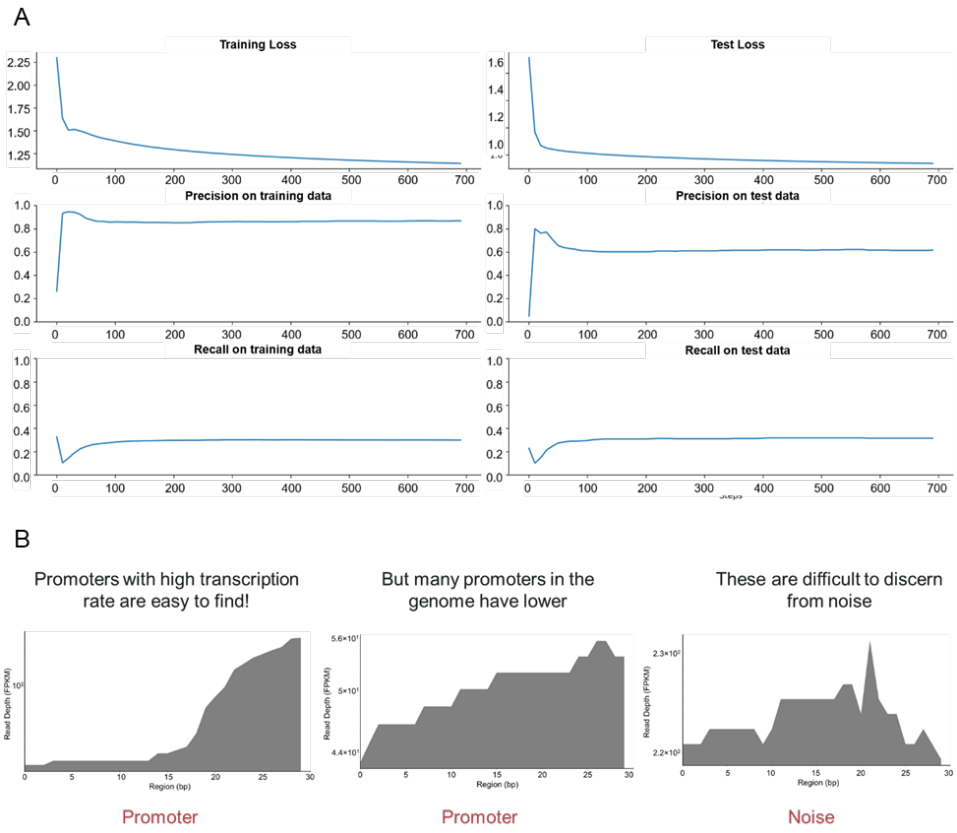


Figure 11. Results of genetic part identification model
(A) Results of the ML model on both training and test data sets. (B) Transcription profiles of 30bp regions – first two are promoters and third isn't.

Transcription profiles at nucleotide resolution contain a wealth of information, in addition to characterizing promoters, the activity of a promoter can also be determined, and the activity of tandem promoters can be calculated individually. End-enriching method of RNA sequencing (Espah Borujeni et al., 2020) has shorter reads and distinguishes the increase in RNAP flux from two tandem promoters, see Figure 10.

In addition to transcription, accurate measurements of translation and protein levels are required to produce an accurate model. I worked on a pipeline to grab and normalize proteomics data, analyze quality control based on the experimental confidence score and compile output. This was made into a pipeline with the help of TACC.

Initially, around 2000 proteins were observed in each sample as shown by the distribution below. However, upon further analysis, roughly a tenth of these were of high enough quality (confidence score >10) to be considered for analysis which is also demonstrated by the distributions in Figure 12.

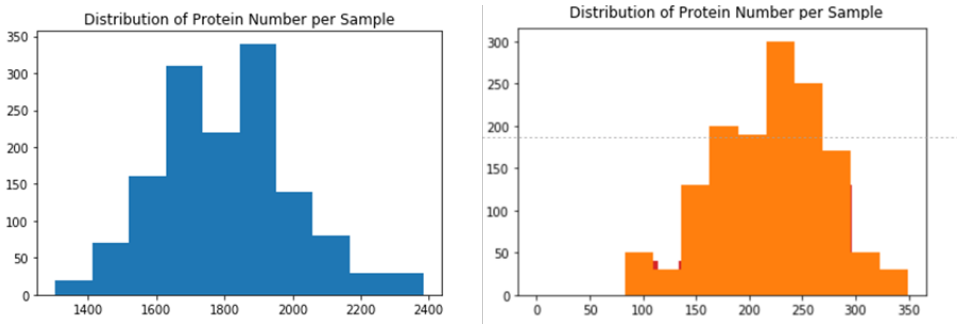


Figure 12. Results of high throughput proteomics experiments (A) Distribution of proteins with non-zero abundance value across samples in the proteomics data for NAND iterate and (B) same distribution for proteins that passed QC i.e. had confidence score > 10

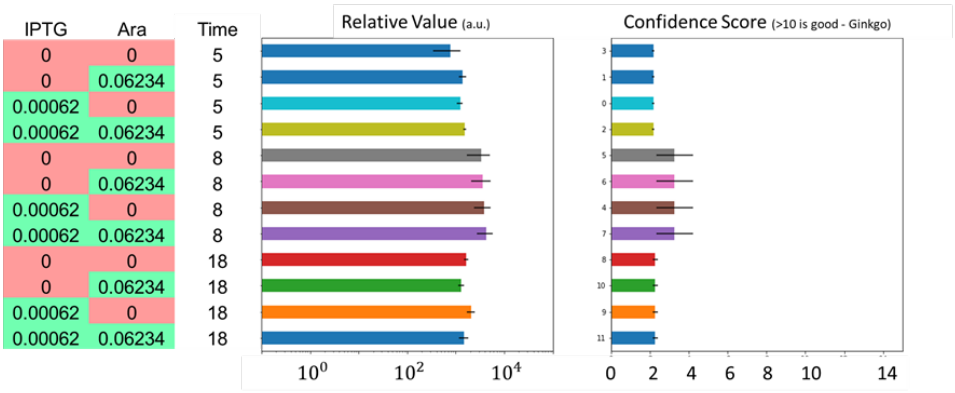
One consideration of using proteomics data was to resolve the question of unidentified IcaR protein in the RNAseq results is the NAND-circuit strain. However, as shown in Figure 13, only the IcaR gate on the plasmid had high enough quality data to conclusively quantify IcaR protein abundance. As a result, no conclusive result can be given from the proteomics data about the presence of IcaR in the NAND_circuit strain.

Given the low quality of proteomics data it was deemed unsuitable for such modeling, and ribosomal profiling, which obtains accurate data on actively translated mRNA, was used instead. Previous work in the lab has led to automated design of 3-input 1-output combinational circuits. Inducers for which sensors have been developed and characterized, such Arabinose, IPTG, and aTc, can be used as inputs. A variety of these circuits have previously been constructed and tested17. For the circuit 0x41 (Espah Borujeni et al., 2020), RNAseq and Ribo-seq data is available for all 8 combinatorial induction states and a control.

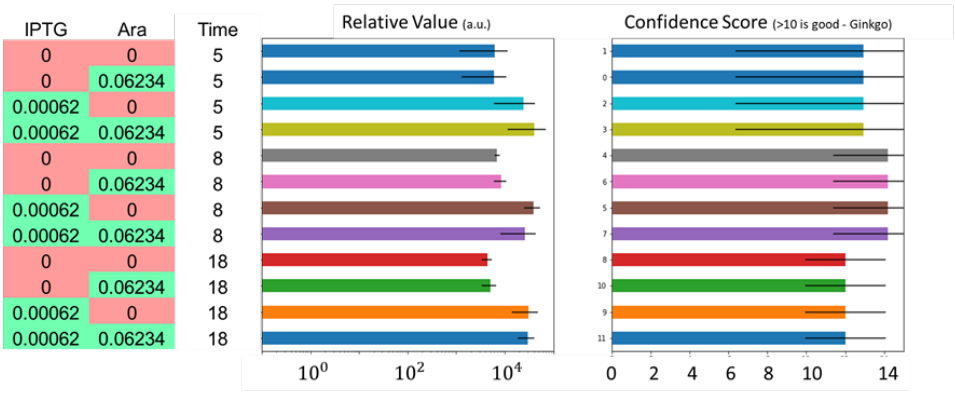
Form this data, translation efficiency can be calculated as the ratio of ribosome density to transcript abundance (Espah Borujeni et al., 2020). This value will be relative considering sequencing results are in arbitrary units. Analyzing the data at hand, the relative translation efficiency (RTE) varies greatly across synthetic genes but remains relatively constant for each when varying transcription level (Figure 14 B – V). In comparison to endogenous genes, RTE is lower and transcript abundance of synthetic genes is higher, suggesting more efficient translation would lower the transcriptional burden of synthetic genes.

Promoter maximum activity (α) and transcription factor binding coefficient (κ) are obtainable from the data by plotting promoter activity against transcription factor concentration and fitting, see Figure 14 C. The cooperativity of a repressor (n) is also measurable by fitting the hill equation to the data. Similar to promoter activity, the strength of the terminators and efficiency of the ribozymes can also be calculated. In addition, dilution rate and mRNA degradation rate shown as "delta"s can be obtained for a host strain from growth assays and RNA experiments, respectively. This data can be used to covert all relative units to biophysical units and obtain RNAP flux (k) in units of RNAP per second per DNA. Maximum promoter activity (α) when converted to physical units is denoted as k_{max}. Figure 14 C-D show clear agreement between original data and reconstruction proving state-independent parameters are suitable for describing variation in expression and protein production across induction states.

A. *MG1655_IcaR_Gate*



B. *MG1655_pJS007_LALT__I1__IcaRA*



A. *MG1655_NAND_Circuit*

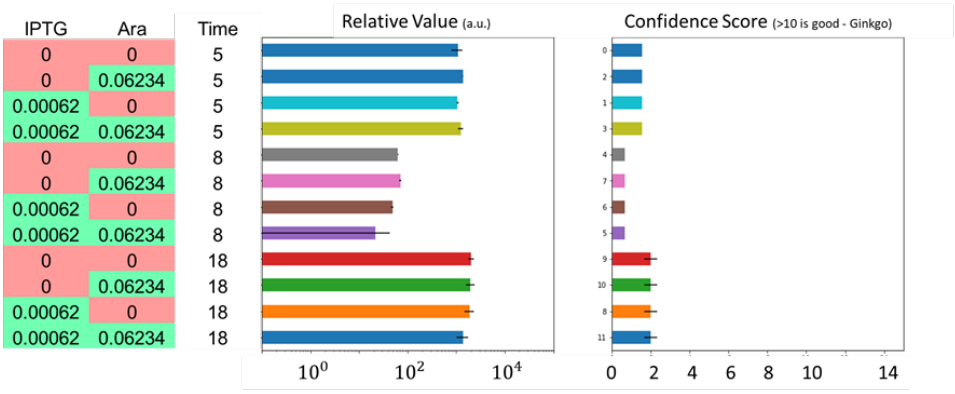


Figure 13. IcaR expression for three strains

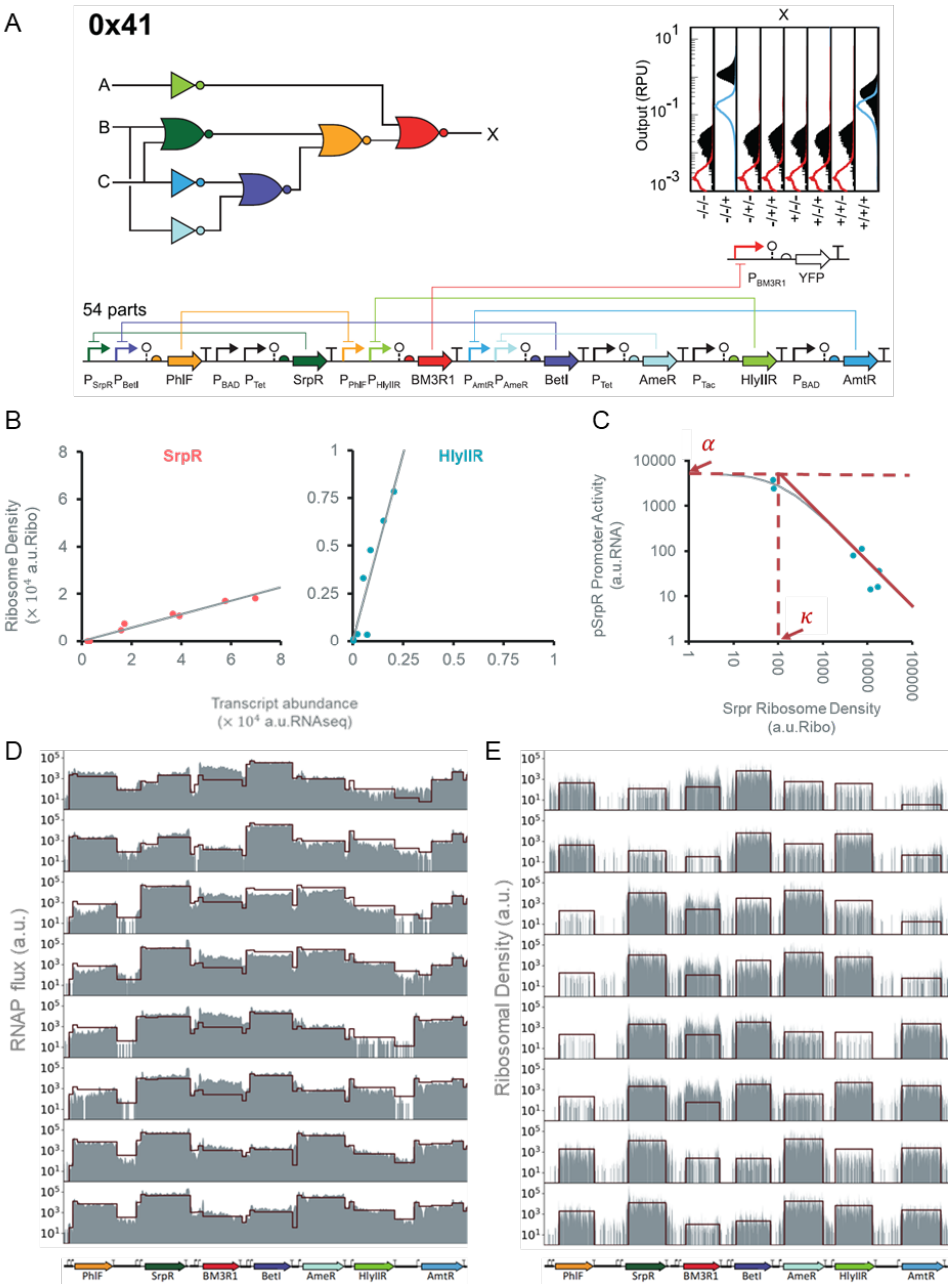


Figure 14. Part characterization using genetic circuit analyzer and steady state simulations (A) Cello designed circuit 0x41 showing the logic representation (top left), the genetic parts representation (bottom), and the predicted and experimental YFP output (top right). Reproduced from Nielsen et al., 2016. (B) Plot of ribosome density against transcript abundance whereby linear correlation demonstrates constant translation efficiency despite differential expression. (C) Plot of promoter activity against ribosome density, or equivalently protein abundance at steady state. Plot demonstrates suitability of equations employed to capture repression and parameters obtained from data. (D) Transcription and (E) translation profiles with reconstruction using model fully parametrized by sequencing data alone.

1.4.3 Constructing a dynamic model from data-driven parameters

A biophysical model of transcription and regulation yields ODEs capable of describing RNAP flux across the genetic construct shown in equations 6-10. Indices number parts along the genetic sequence and equations are written for a sample gate with parts in order of promoter "i-1", gene "i", and terminator "i+1". The transcription factor denoted by gene "j" represses promoter "i-1".

∀ promoter:

$$k_{downstream,i-1} = k_{upstream,i-1} + \frac{k_{max,i-1}}{1+\left(\frac{[P_j]}{\kappa_j}\right)^{n_i}} \quad (6)$$

∀ gene:

$$\frac{\boldsymbol{d}[mRNA_i]}{\boldsymbol{dt}} = k_{upstream,i} - \delta_{mRNA}[mRNA_i] \quad (7)$$

$$\frac{\boldsymbol{d}[P_i]}{\boldsymbol{dt}} = k_{translation,i}[mRNA_i] - \delta_{dilution}[P_i] \quad (8)$$

$$k_{downstream,i} = k_{upstream,i} \quad (9)$$

∀ terminator

$$k_{downstream,i+1} = \frac{k_{upstream,i+1}}{S_{i+1}} \quad (10)$$

$k_{upstream,i}$ and $k_{downstream,i}$ are the RNAP flux upstream and downstream of a part respectively, hence $k_{upstream,i} = k_{downstream,i-1}$ always. δ_{mRNA} is the mRNA degradation rate, $\delta_{dilution}$ cell dilution rate, $k_{translation,i}$ ribosome flux on a transcript, $k_{max,i}$ maximum promoter activity, κ_i binding coefficient of a transcription factor, and S_i terminator strength. Experiments were conducted 5h after induction, therefore equations can be collapsed assuming steady state and thus the data allows for direct calculation of parameters κ_i, n_i, $RTE = \frac{k_{translation,i}}{\delta_{dilution}}$, and $\alpha = \frac{k_{max,i}}{\delta_{mRNA}}$
.

To evaluate whether the parameters fully capture details of genetic circuit function, transcription and translation profiles at steady state were reconstructed across 8 combinatorial induction states using this initial model and are shown in Figure 14 D-E. State-independent parameters alone were used in the reconstruction and the surprisingly good agreement demonstrates their applicability in fully parametrizing a model able to capture the behavior of the genetic circuit.

With the dynamic model a temporal profile for any genetic circuit can be obtained, a set of parameters are required for each circuit (same as the Hill equation parameters). The temporal profile of the genetic circuit 0x41 from the Cello paper is shown below as an example. The plots clearly show delayed responses of gates to their inputs.

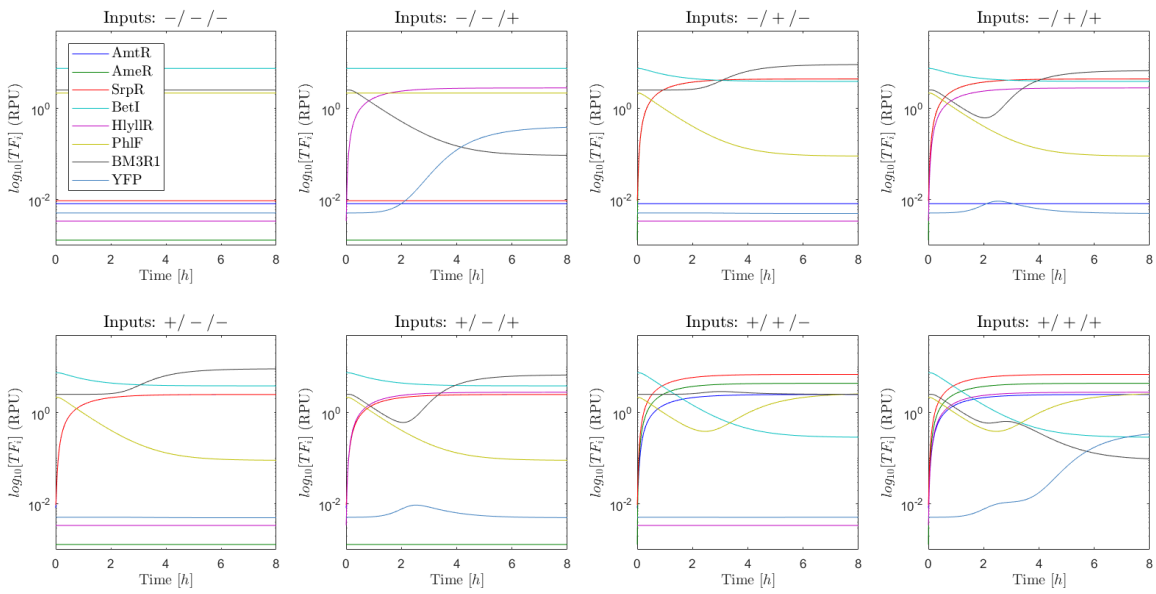


Figure 15. Predicted temporal profile of gene expression

Using this methodology allowed for the complete parameterization of the system with accurate predictions in almost all steady states and promoter activities. The prediction plots can be found below. The model has difficulty in accurately predicting the response when the two promoters PhlF and HlyIIR appear in tandem. Interestingly, using this method of characterization to identify the source of the issue we found that from the RNA-seq data the second promoter is shown to have higher activity even when fully repressed if the first promoter is on. This could potentially be the roadblocking phenomenon, which, had not been previously characterized or modeled. This elucidates sources of error and noise with RNA-seq data in general and is observed in all pair of tandem promoters, even though only in the case given above the error is significant enough to result in poor prediction. The plots following the prediction below show this effect in 3 pairs of tandem promoters.

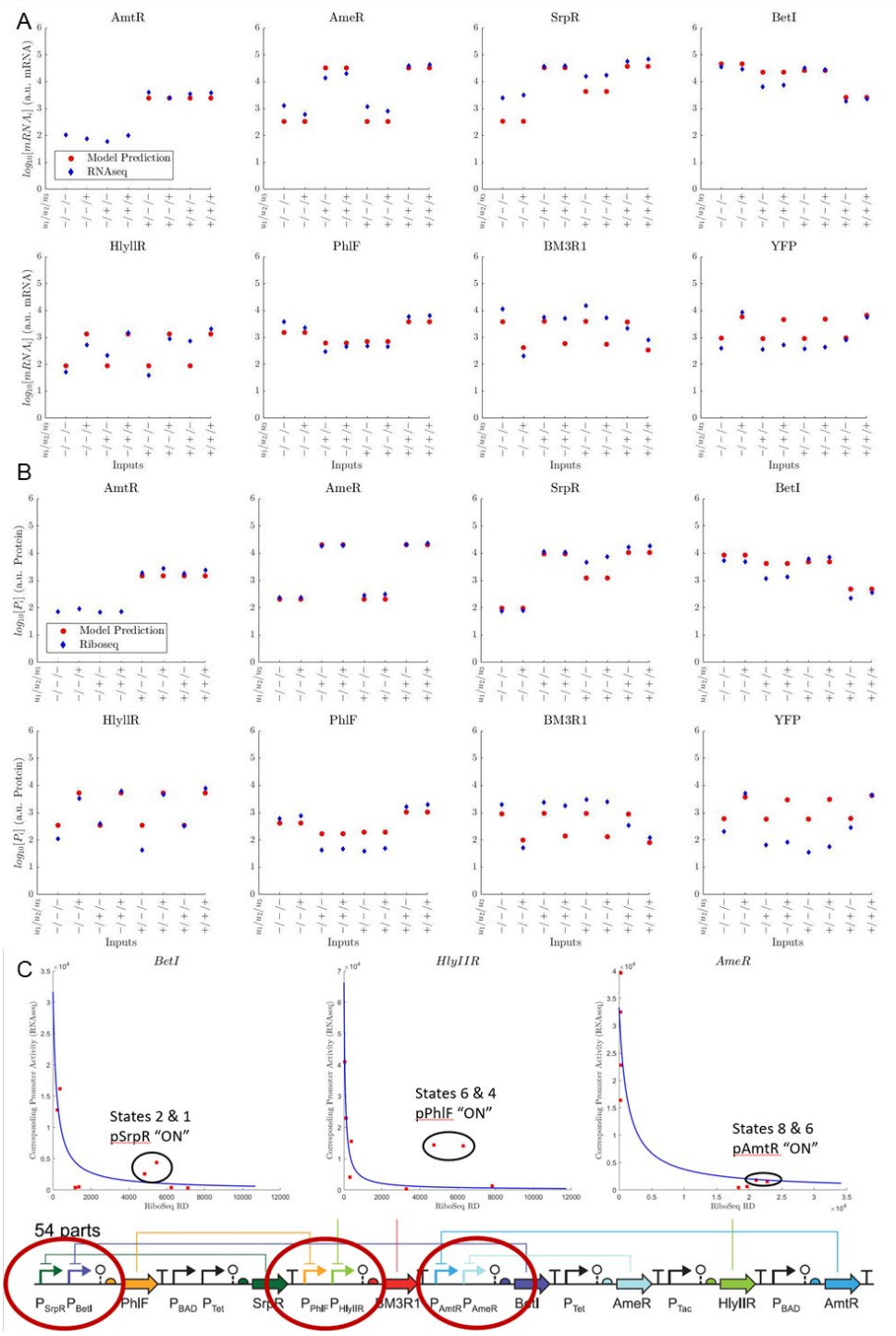


Figure 16. Predicted expression levels using the genetic circuit simulator (A) Prediction vs. Measured RNA-seq (B) Prediction vs. Measured protein expression (C) Tandem promoter effect

1.4.4 Modeling noise propagation through genetic circuits

Stochastic gene expression is a major contributor to variation in expression across a population. The predominant mechanism by which stochasticity appears in gene expression is transcriptional bursting, whereby a series of transcription events render the DNA transiently 'off' due to folding and coiling changes. These effects are resolved by enzymes such as topoisomerases allowing another burst to proceed. This mechanism is well understood but poorly parametrized to understand cell-to-cell variation within a population. In the past 3 months I have worked to produce stochastic simulations that can simulate the propagation of stochasticity through a gate to predict circuit malfunction due to loss of distinct expression states.

Figure 17 demonstrates a full two-state model describing the expression of a gene driven by promoters of different strength. Though mRNA levels are dynamic and vary significantly, protein levels are much less variable due to their relatively high stability and the predominance of dilution as the rate determining step in their depletion. This means circuits constructed using repressor proteins without degradation tags will have a much tighter distribution across a population as compared with more rapidly degrading proteins or mRNA only. The parametrization of the model is as follows with k_1, k_{-1}, μ, σ from literature and J, α can be obtained from RNAseq as explained in our recently published paper.

J: transcription rate (from mean transcription rate of specified promoter)
α: translation efficiency
k_1, k_{-1}: (from cell mean doubling time and burst frequency)
γ: mRNA degradation rate
μ, σ: dilution rate & partitioning variance

Error is defined as the difference between a single cell's expression level and the deterministic value assuming no stochasticity. This type of stochastic modeling will allow us to study the impact on larger circuits, improve circuit connection designs by Cello and determine sizes at which the circuit may fail due to stochastic variation across cells.

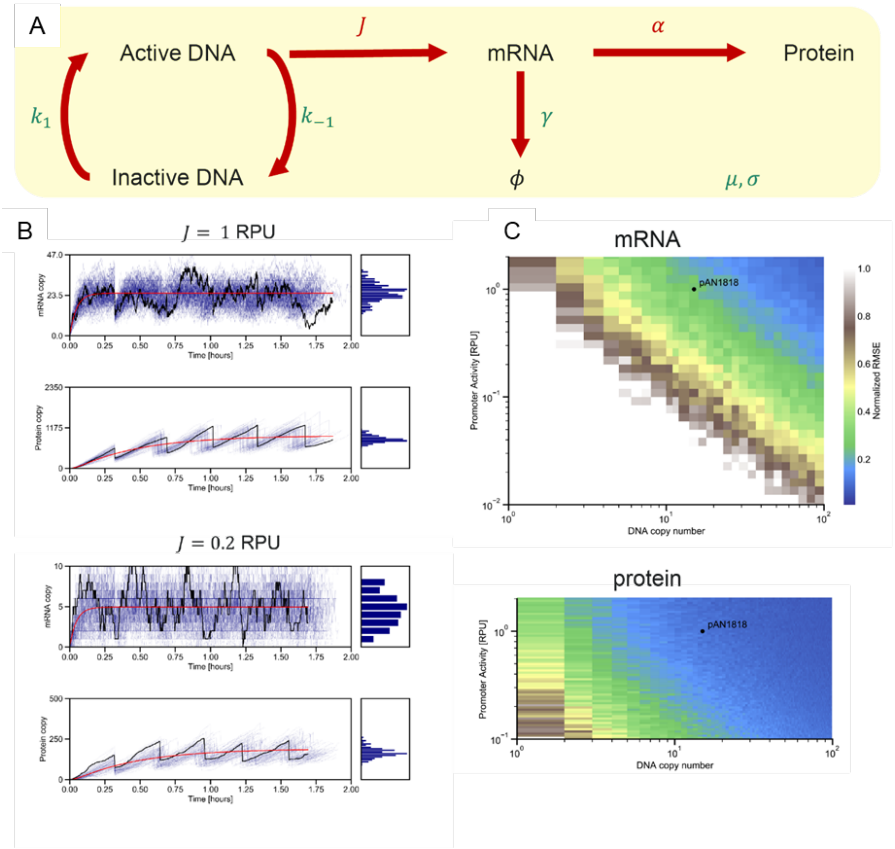


Figure 17. Stochastic simulation results
(A) Schematic of two-state model describing transcriptional bursting (B) results of stochastic simulation for two promoters of different strength on a low copy plasmid. (C) root mean squared error of an entire population of cells for a range of DNA copy and promoter activity levels.

Genetic circuits built on plasmids often have relatively tight distributions of expression level within a cell population, they owe this to the multiple copies of the plasmid and the high expression levels due to part designs used. However, plasmid-based gene circuits have lower evolutionary stability and often impact host growth by drawing significant cellular resources. This has incentivized researchers to construct gene circuits on the genome, which has lower copy, with lower expression designs. This addresses the challenges mentioned but raises the question of how low circuit expression can be tuned before it loses functionality due to wide variations within a cell population. If the binary expression states are no longer discernible due to the stochastic gene expression, the circuit is no longer usable.

To answer this question, we developed a stochasticity model to predict noise levels for a single gene expressed by a promoter, and studies have been conducted to understand the change in expression noise levels with changing promoter activity. Over the past three months I have worked to expand our model to simulate how noise would propagate through each genetic gate, with the goal of elucidating whether low expression gates can be used on the genome. Further by running these simulations for larger circuits, we can now show that noise levels do not increase due to propagation through sequential logic gates. As such, if the gates are designed individually within

parameters such that they're functional on a single DNA copy, the combined circuits should not fail because of noise propagation and expression distributions should still be discernible.

Figure 18 demonstrates the changes observed in simulated gates on a plasmid vs on the genome and when propagated through a large circuit (0x41), shows the output distributions are still discernible. Despite some intermediate gate inputs not being at the true high or low state, the simulation shows the final circuit output is not significantly impacted.

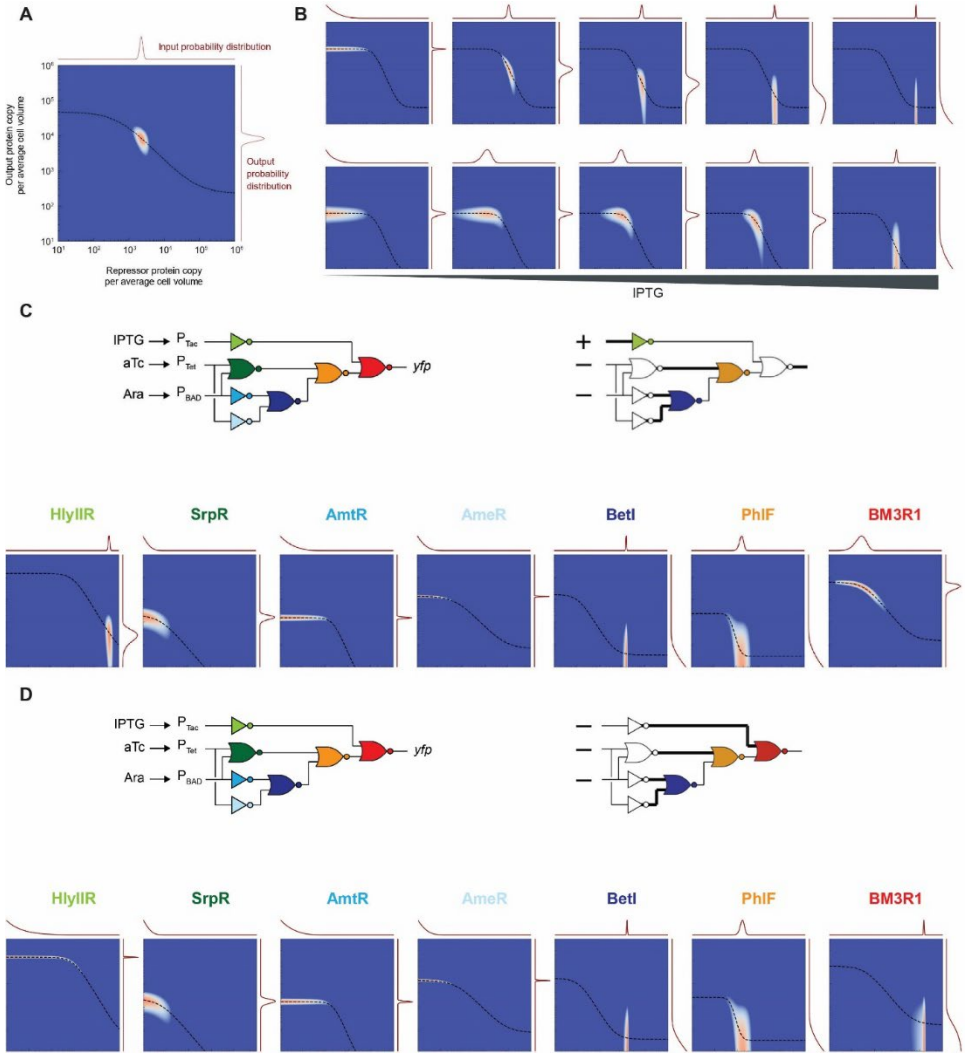


Figure 18. Simulations of noise propagation using stochastic gene expression model (A) Noise propagation through an example gate; x-axis is the repressor protein copy normalized for cell volume, y-axis is the output protein copy also normalized, the dashed black line shows the expected deterministic response function, the heat map shows the predicted distribution of cells within a population (blue to white to orange showing increasing fraction – the blue-white boundary can be thought of as likelihood equivalent to a single cell in 1ml culture at OD=1.0). The normalized probability distributions of expression level of the repressor is shown at the top and the output protein at the right. (B) Noise propagation plots for the AmeR gate on a plasmid (top row) and on the genome (bottom row). (C,D) Noise propagation plots for the 0x41 Circuit in an on state and off state

Plasmid-based gene circuits have low evolutionary stability and often impact host growth by drawing significant cellular resources. Many engineered bacterial strains require synthetic regulatory networks on their genome to ensure functionality. Genomic gene circuits have lower DNA copy and are designed for lower expression. However, low expression can result in loss of functionality due to wide variations within a cell population. If the binary expression states are no longer discernible due to stochastic gene expression, the circuit is no longer usable.

To answer this question, a stochasticity model has been developed to predict noise levels for a single gene expressed by a promoter, and studies have been conducted to understand the change in expression noise levels with changing promoter activity. The following experiments were conductd to validate the model's predictions and study the impact of stochastic expression on genetic gates on the genome. By constructing multiple gates with the same promoter activities but varying ribosomal binding site (RBS) strength, design rules to construct robust low expression genetic circuits can be derived.

Figure 19 shows that as the expression level of these gates increase, the noise level drops. Importantly, the noise level of a circuit output is actually lower than the noise level of the input promoter; demonstrating that noise levels do not increase as a result of signal propagation. Further, when expressed at intermediate levels (i.e. the repressible promoter is neither fully active nor fully repressed), noise levels vary between the three constructs. This shows that by tuning RBS activity, noise levels can be adjusted while still achieving similar expression levels.

Each dot is a sample grown separately and analyzed on the flow cytometer to obtain population values of fluorescence, legend entries denote the name of the strain in each samples. "R1", "R2", and "R3" denote three strains containing the same gate but with different ribosome binding sites. "AutoF" represents samples with only autofluorescence (the strain contains no fluorescent protein) as a negative control. A strain containing the yellow fluorescent protein (YFP) expressed by a standard promoter was used as a positive control, this promoter expresses YFP at a level of 1 relative promoter unit (RPU) and the strain is denoted as "RPUs" i.e. RPU standard. "Input" is the strain containing the gate's input promoter expressing YFP. Each sample was tested in two biological replicates grown separately.

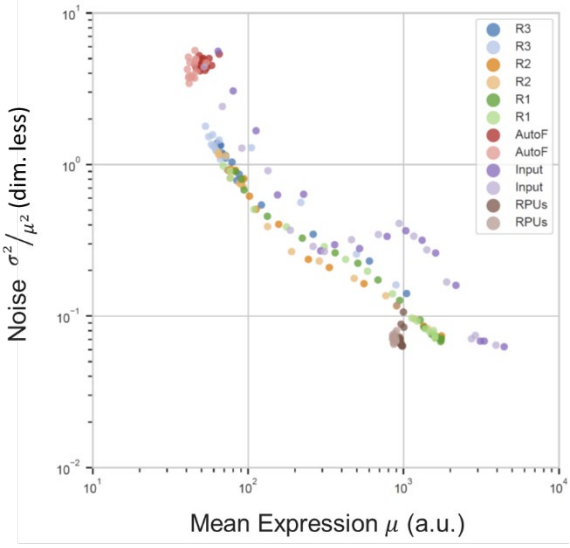


Figure 19. Noise levels within a cell population

The low evolutionary stability and high resource usage of plasmid-based circuits encourages the idea of moving genetic circuits onto the genome. However, genome circuits risk losing functionality due to low expression or wide variations (noisy expression) within a cell population. If the binary expression states are no longer discernible due to noise, the circuit is no longer usable.

Building on the stochasticity model, we proceeded to conduct a meta-analysis of flow cytometry data of genome-based gates in *E. coli*. The first set of data showed that varying the RBS upstream of the repressor did not change the noise level.

This allowed us to identify the most influential factors that determine the noise level of different genetic gates. Figure 20a shows that the majority of gates constructed on the genome fall on a lower bound of extrinsic noise, however, certain gates have significantly higher noise levels at the same mean expression. By investigating the main differences among these gates, the maximum strength of the repressible promoter (i.e. when repressor is uninduced) driving the output fluorescent protein gene is the determining factor in noise level. This is shown in Figure 20b where the noise level is scaled by the output promoter's max strength. Using this data in conjunction with the mechanistic model, optimal gates for each position in a gene circuit can be selected to minimize noise propagation through a circuit. Each dot is a sample grown separately and analyzed on the flow cytometer to obtain population values of fluorescence.

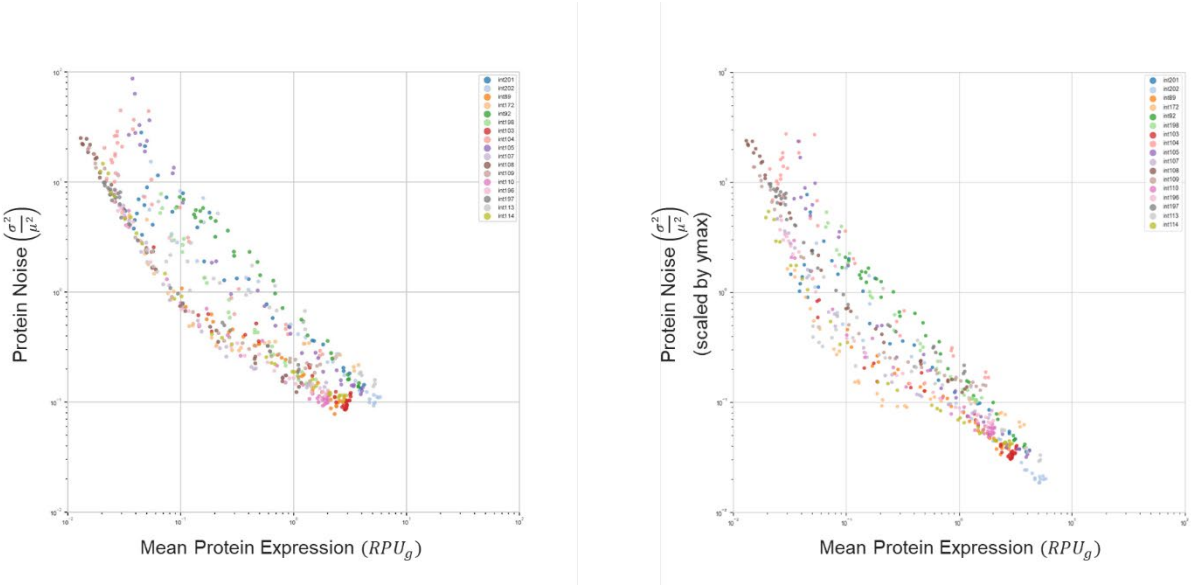


Figure 20. Noise levels across different genomic gates in *E. coli* left (a) Noise (as variance over mean squared) within a cell population as a function of the mean expression level for 17 strains containing a genomic gate each, represented by different colors. (right) same data scaled in noise by the output promoter's maximum strength

1.4.5 Landing pad selection

A further effort was made this quarter to develop a methodology to select landing pads based on expression in the vicinity of that region. RNA sequencing data from Ginkgo on the wild-type Bacillus 168 Marburg strain was used to initialize a model and 15 positions were selected and submitted. The expression in the region is depicted in Figure 21 below. The hope is that with OD

data and expression level (fluorescence) from the results of the landing pads, and similar data from E. coli, the model can be built to select optimal landing pad position in a strain-agnostic manner.

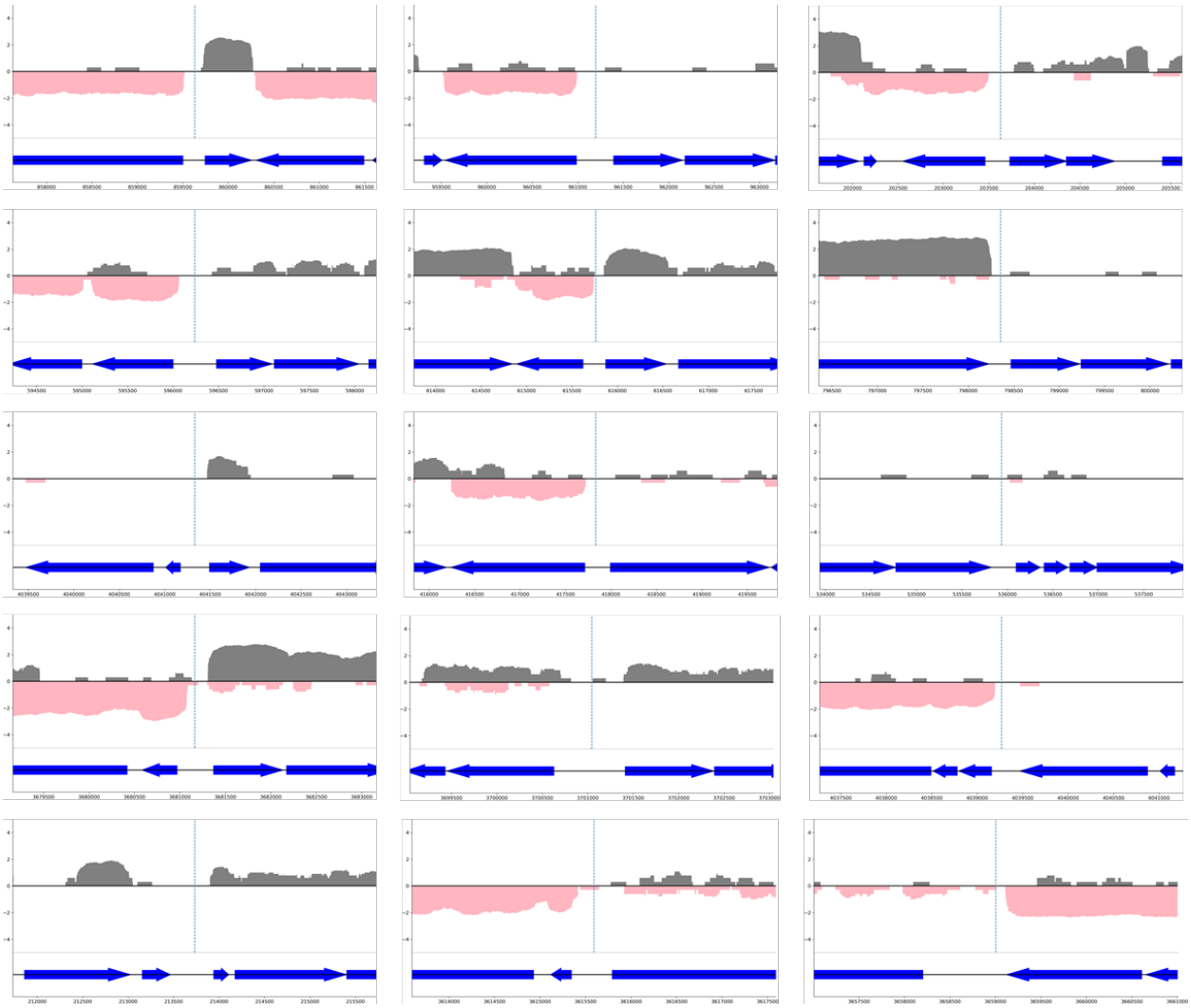


Figure 21. Automatic landing pads selected using transcription profiles. Positions are shown with vertical dashed line and expression profiles are grey (forward strand) and pink (reverse). Genes are shown as blue arrows below each plot

1.4.6 Characterizing genomic circuit promoter activities using RNA-seq

Cello is an automated software for the design of genetic circuits. It uses genetic logic gates to build circuits with user-defined inputs, outputs, and truth tables. Logic gates are composed of a repressor protein and a cognate repressible promoter. Up until the beginning of SD2 program, Cello had only been used and tested to design plasmid-based circuits. In SD2, we attempted to move circuits into the genome. Therefore, in Voigt lab, we began moving logic gates into the genome of *E. coli* MG1655 strain within specific locations known as landing pads. We then characterized these gates using flow cytometry and build first generation of circuits using Cello.

These circuits were very simple and consisted of only two gates. The accuracy of Cello-designed circuits depended on the activity of their synthetic promoters. Each promoter had a well-characterized activity that had been previously measured using flow cytometry and in standard conditions (at 37 ^{0}C and 5 hours after induction at log-phase of growth) under a plasmid-based system. However, due to very low copy number of genomic DNA inside the cells (1 copy), flow cytometry data often reached limit of detection, and thus resulted in limited dynamic range for gate activities. To alleviate this problem, we were interested in characterizing these genomic promoter activities using transcriptomic data (RNA-seq). We wanted to use RNA-seq as a tool for promoter characterization, and compare them with flow cytometry measurements with the ultimate goal of replacing all flow cytometry measurements with RNA-seq. To demonstrate this concept, we characterized our genome-integrated two-gate circuits using both RNA-seq and flow cytometry measurements in Ginkgo.

Figure 22 shows the transcription profile of genome-integrated YFP reporter across 4 biological replicates under standard condition (37 ^{0}C and log-phase) without IPTG induction. Figure 23 shows the same measurements but under IPTG induction. Figure 24 shows the region around pTACmin promoter for the YFP reporter, with and without IPTG induction, revealing excellent consistency across 4 replicates. Therefore, promoter activities were calculated by averaging the profiles before and after the promoter's start site and then subtracting the two averaged values. The measured promoter activities and their OFF to ON dynamic range are shown in Table 1, where both RNA-seq and flow cytometry measurements provide relatively similar dynamic ranges.

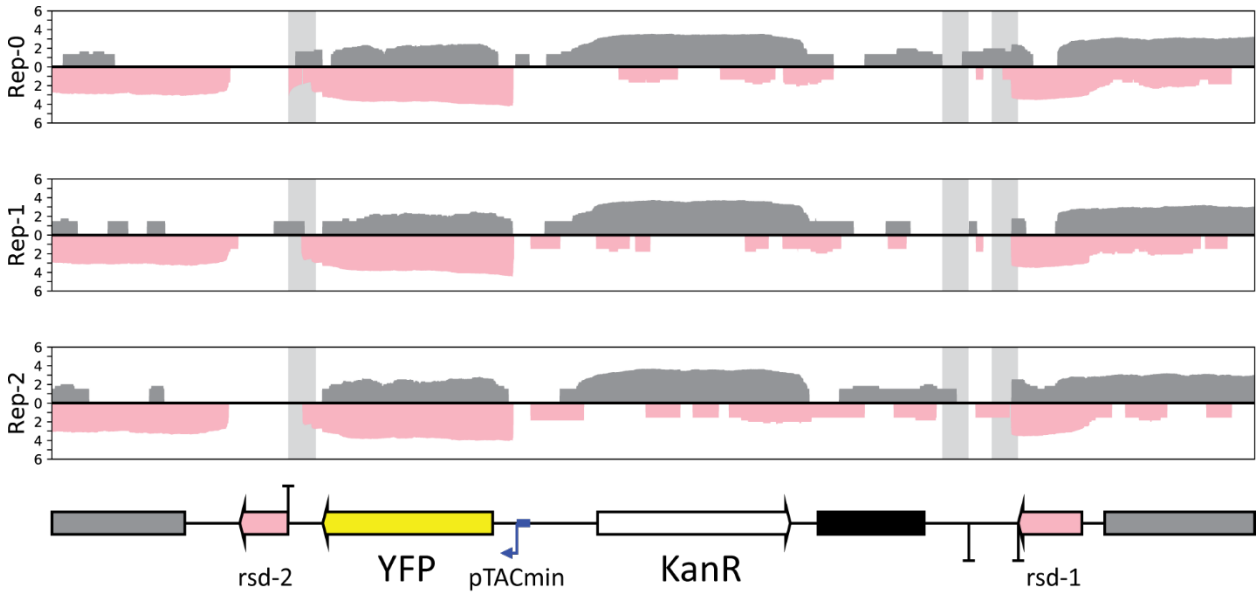


Figure 22. Transcription profile of genome-integrated promoters in uninduced conditions Transcription profiles of the genome-integrated YFP reporter across 4 biological replicates are shown under standard condition (37 ^{0}C and log-phase) without IPTG induction. One of the replicates was lost due to robot handling issue. The DNA annotation is shown at the bottom. Gray and pink regions in the transcription profiles represent sense and antisense transcription, respectively. The grey columns are the location of terminators that were used to insulate the synthetic gates from the neighboring genomic regions by splitting the native gene rsd. The location of promoter of interest (pTACmin) is also shown.

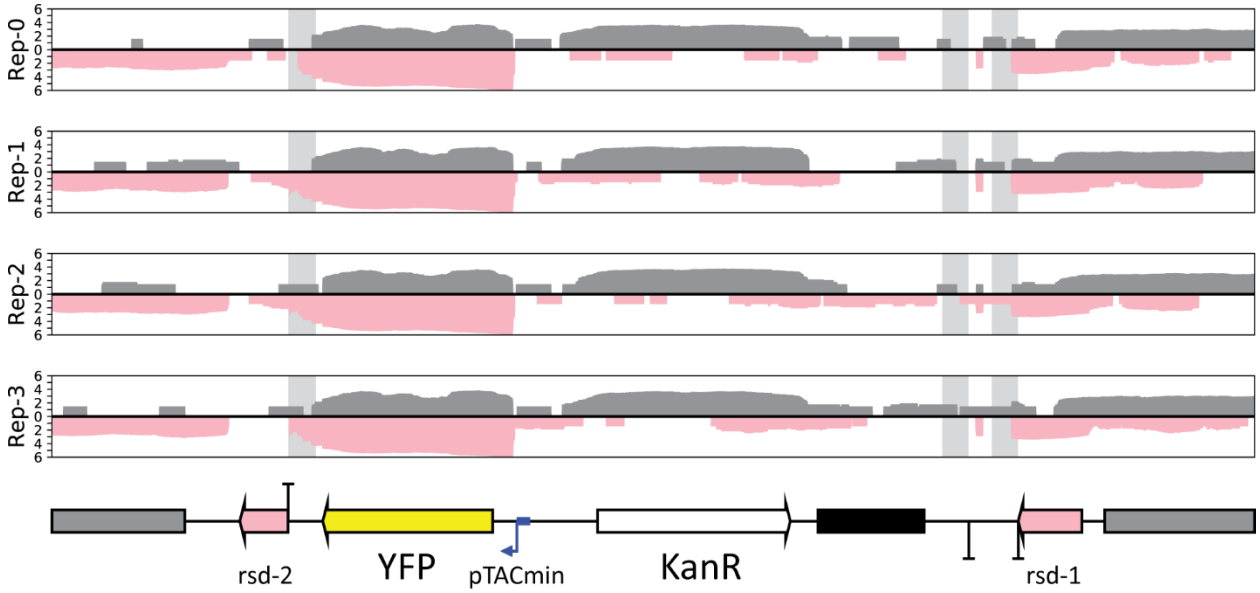


Figure 23. Transcription profile of genome-integrated promoters under induced conditions. Transcription profiles of the genome-integrated YFP reporter across 4 biological replicates are shown under standard condition (37 ^{0}C and log-phase) with IPTG induction.

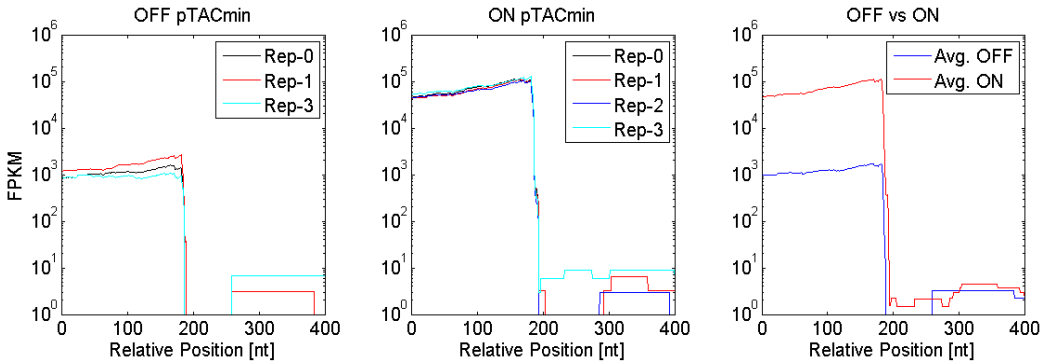


Figure 24. Transcription profiles around promoter start sites.
Transcription profiles of the 4 biological replicates shown in Figure 22Figure 23 are overlaid to show the regions around 200 nucleotides before and after the pTACmin promoter. ON and OFF plots are samples with and without IPTG, respectively.

Table 1. Measured activity of pTACmin promoter. RPU is Relative Promoter Unit

	OFF activity (- IPTG)	ON activity (+ IPTG)	Dynamic Range (ON / OFF)
RNA-Seq (FPKM)	1990	114730	57
Cello (RPU)	0.024	0.967	40

Using the above method, we calculated the activity of all the genomic synthetic promoters designed in this experiment. Overall, after normalizing the effect of copy number on promoter activity (dividing promoter activities by the expression level of their antibiotic resistance markers), we found a clear correlation between the RNA-seq values and the values used in Cello (measured using flow cytometry) (Figure 25). Interestingly, the correlation for promoters on plasmids is one to one, demonstrating the power of RNA-seq to characterize promoter activities in a direct way. Finally, using the RNA-seq measurements under different growth conditions (different temperature and different growth times) we can directly measure how promoter activities are changing compared to the standard conditions (Figure 26). These findings are quite valuable because they can give us a hint as to how much Cello-designed circuit are expected to perform properly in non-standard conditions.

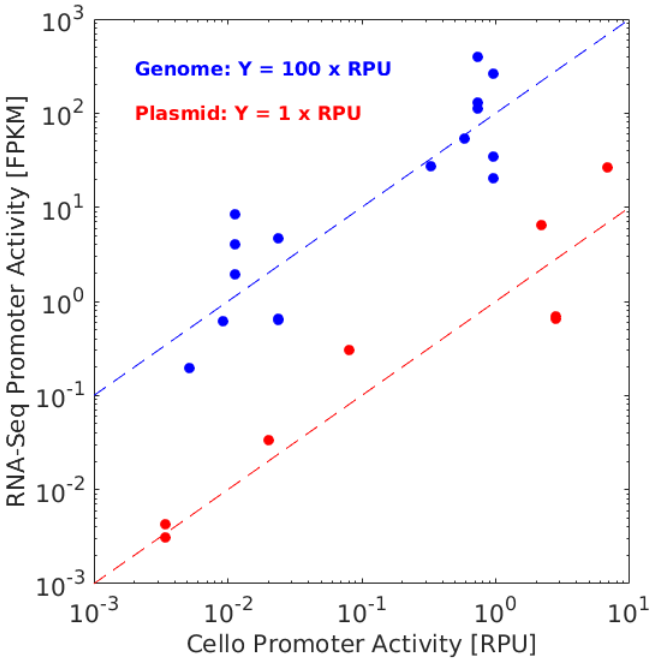


Figure 25. Comparison of promoter activities using RNA-seq and Cello, values derived from flow cytometry

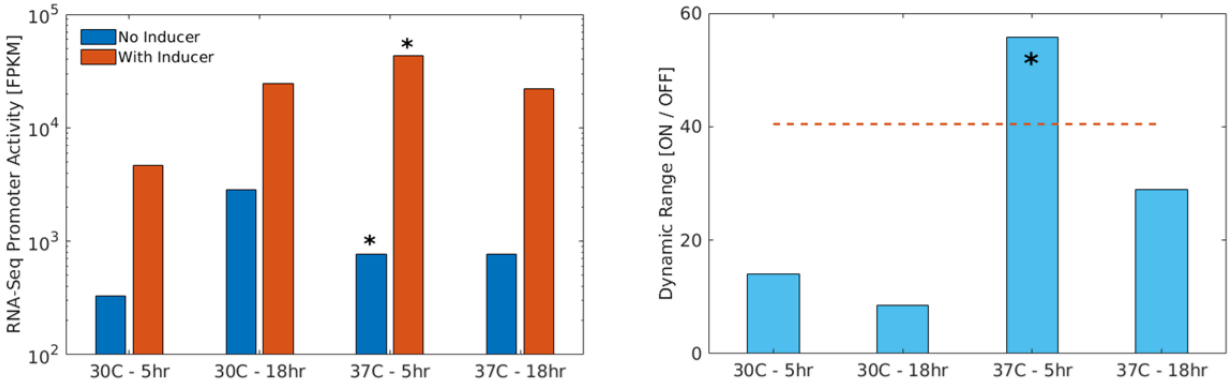


Figure 26. Effect of growth conditions on the activity of pTACmin promoter. Left plot shows the ON and OFF activities and right plot shows the dynamic ranges. The growth conditions are shown under x-axis. The horizontal dashed line is Cello value at standard condition (37 ^{0}C and 5 hours), and asterisk symbol shows RNA-seq value at standard condition.

1.4.7 Designing the next generation of circuits in *E. coli* genome with proper controls

The first version of our genomic-integrated gates and circuits in Voigt Lab had couple of problems that needed to be addressed. First, two of the integration locations (landing pads) were found to split two native genes into half, which could cause undesired impacts on the host cell. Second, there were a few genes (including couple of sensors and antibiotic resistant markers) that were not used by the circuit and could impose additional metabolic load on the host cell. In order

to accurately measure the impact of our gates and circuit on the host cell, these problematic issues must had been resolved. Therefore, we generated a new version of *E. coli* MG1655 strains with corrected integration locations (problem #1 resolved). Next, starting from this corrected strain, we designed and built 7 new strains that each contain only one sensor with no antibiotic resistant marker (problem #2 resolved). Out of these 7 sensors, we chose LacI as our default sensor, which responds to IPTG and activates pTac promoter. We integrated 9 separate gates into the strain with LacI sensor, so that the only external genes in these strains were LacI sensor, the gate's repressor protein, and the gate's YFP reporter. Importantly, antibiotics resistant markers were removed from all these strains, which were subsequently sent to Ginkgo for DNA sequencing and full characterization using plate reader, proteomics, flow cytometry, and RNA-seq.

We also fully characterized these 16 strains (7 sensors and 9 gates) in Voigt lab according to the Voigt lab standard protocol. This included measuring the activity of sensor output promoters in the OFF (absence of inducer) and ON (presence of inducer) conditions. Promoter activity was measured in the unit of RPU by normalizing the fluorescence of YFP reporter protein in each strain with the fluorescence of a standard promoter strain. The resulting sensor activities are shown in Figure 27. They all had variable OFF and ON promoter activities, and different dynamic ranges. Then, this information was used by Cello to design and build couple of NAND circuits that allow us to investigate how two separate gates will impact the host cell when combined in a NAND circuit. For the purpose of NAND circuit design, we chose LacI and AraC as our two input sensors.

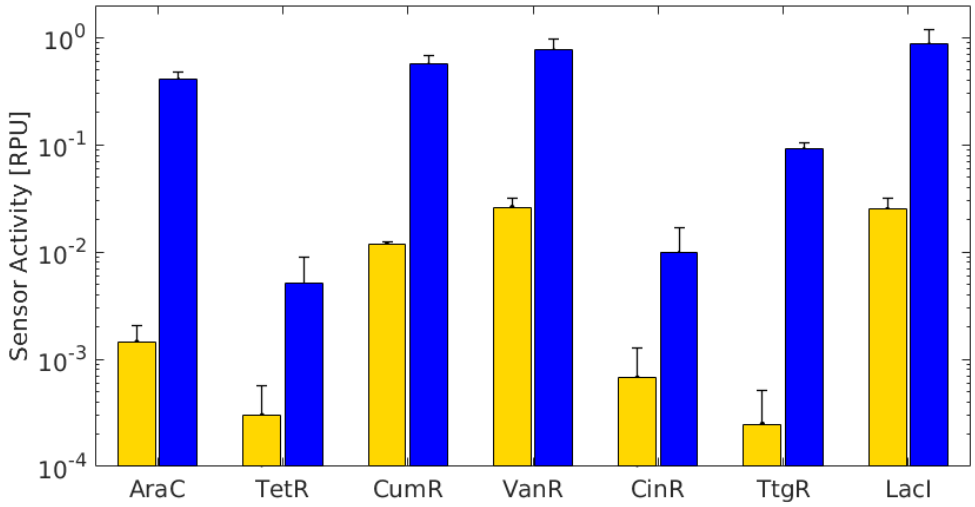


Figure 27. Genomic sensors characterizations. The output promoter activity of each sensor in the absence (yellow bar – OFF state) and presence (blue bar – ON state) of its corresponding inducer. For AraC and LacI sensor, inducers are arabinose and IPTG, respectively.

We also fully characterized all 9 genomically-integrated gates, which includes both the gates' response functions (Figure 28) and gates' toxicities (Figure 29). The response function of each gate relates the activity of gate's input promoter to the activity of gate's output promoter (both in RPU unit). Note that the input promoter for all these 9 gates is pTac promoter, which responds to IPTG. To obtain the output activities of these response functions (Y-axis values in Figure 28), each gate

was treated with a titration of IPTG (12 points between 0 and 1 mM). To obtain the input activities (X-axis in Figure 28), the strain with LacI sensor alone was treated with the same IPTG titration.

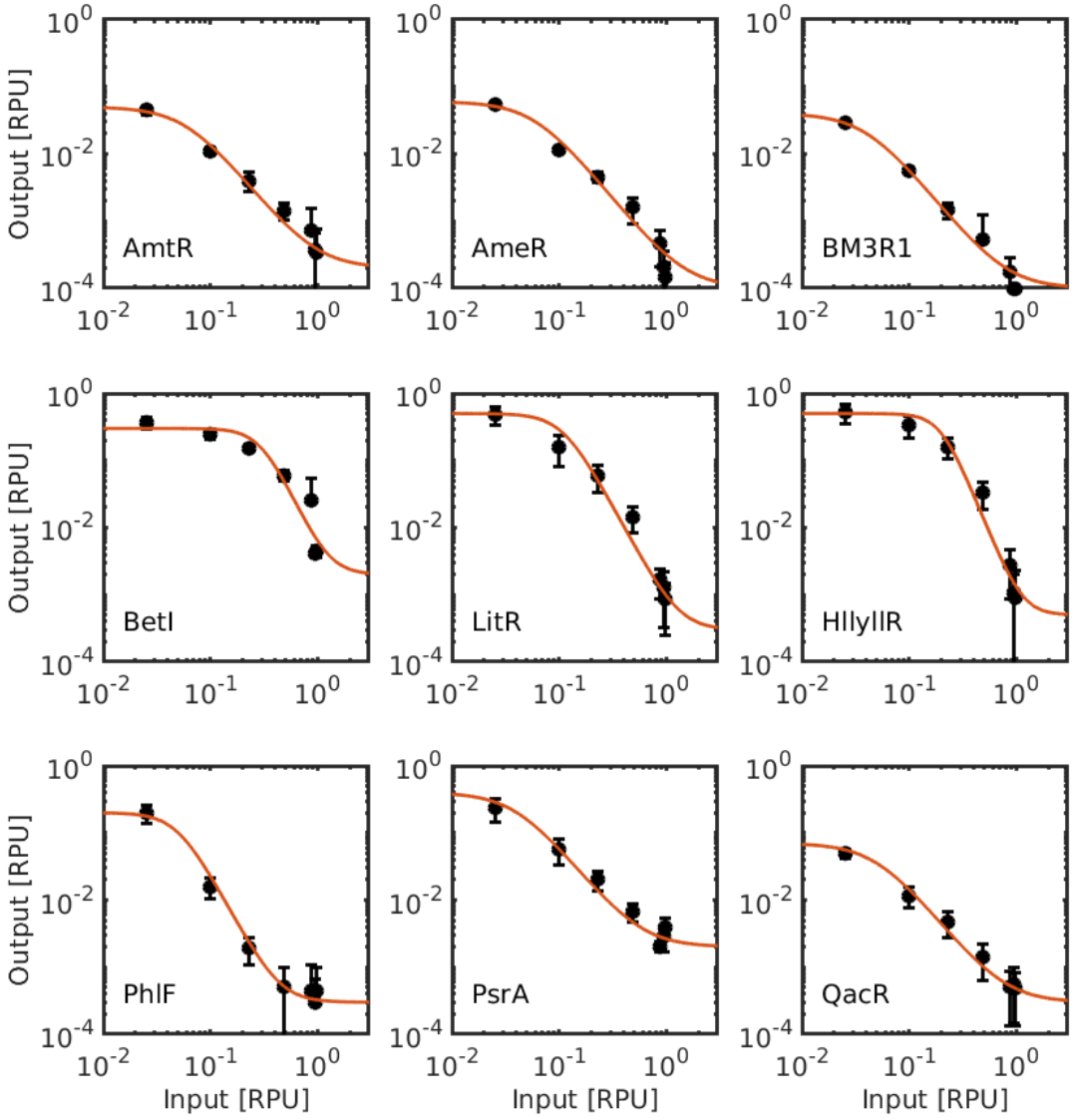


Figure 28. Response function of 9 genomically-integrated gates.
Data is the average of 3 measurements in separate days. Red line is the best fit using a Hill equation.

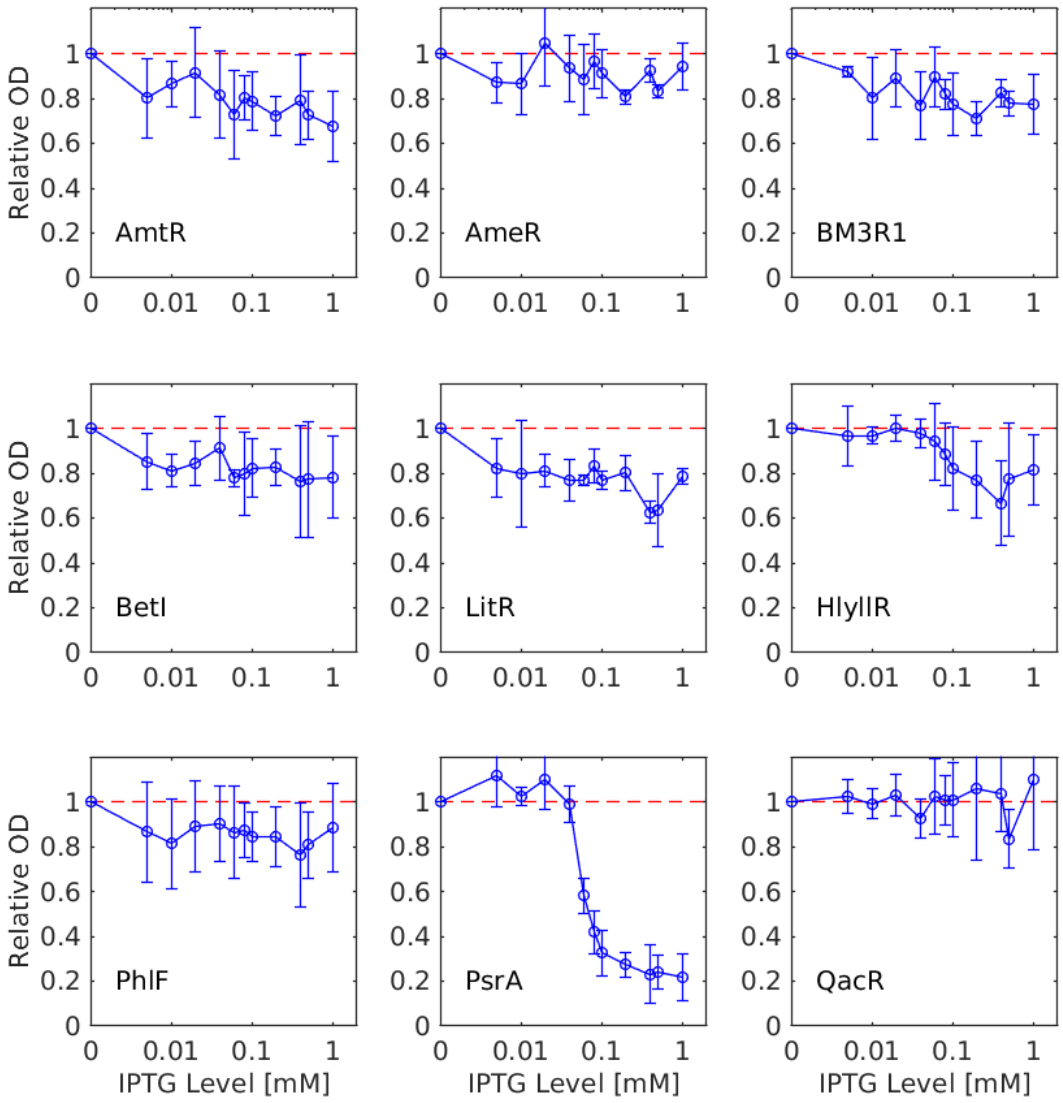


Figure 29. Toxicity of 9 genomically-integrated gates under different IPTG induction. PsrA is a gate that becomes highly toxic after high dose of IPTG.

Having above information about the activity of sensors and gates, we used Cello to design several NAND circuits. To do that, we first created a User-Constraints-File (UCF) specific to these genomically-integrated gates, and Cello predicted 8 versions of NAND circuits. We selected the top three versions with the highest dynamics range of the output YFP reporter: a NAND with PhlF and AmeR gates (50-fold), a NAND with PsrA and AmeR gates (44-fold), and a NAND with PhlF and BM3R1 gates (36-fold). Interestingly, PsrA gate which is toxic, is chosen by Cello in one of the NAND circuits, which gave us an opportunity to study the high impact of that NAND on the host cell. These three NAND circuits were constructed, integrated into the genome, and cured for the antibiotic resistant marker. More importantly, in order to properly quantify the impact of the

NAND circuits and their individual parts on the host cell, we also designed up to 9 additional strains for each NAND circuit, in which different combinations of parts were present in the genome. Overall, in this work, we designed and built 34 genome-integrated strains in *E. coli* MG1655. After cloning was finished, all these 34 strains were characterized in Voigt lab before strains were sent to Ginkgo for DNA-sequencing verification and final characterization at Ginkgo.

1.4.8 Investigating the additive impact of individual gates in NAND circuits

In our NAND circuit RNA-seq data (also known as NAND2.0 dataset), we measured the impact of 4 individual gates and 3 NAND circuits on the native gene expression levels. To measure the exclusive impact of each genetic part, we took a reductionist approach and broke down the gates into their building blocks (sensors, sensor+repressor, sensor+repressor+YFP). This allowed us to measure the impact of individual parts more accurately. However, given the architecture of these gates, the exclusive impact of repressor proteins cannot be directly measured. This is because a repressor protein must be expressed in the presence of a sensor and by adding an inducer. Therefore, the measured impact is confounded by the impact from sensors and inducers. To address this problem, we indirectly quantify the exclusive impact of repressor protein using a simple arithmetic approach, where we subtracted (or divided) the FPKM of sensor+inducer (e.g. shown by strain #7 in Figure 30) from the FPKM of sensor +inducer+ repressor (e.g. shown by strain #11 in Figure 30).

We used a multiplicative method for this task. Importantly, our 34 engineered strains built in the work provided excellent opportunity to test this multiplicative hypothesis. The goal was to quantify the exclusive impact of each repressor and create an impact library. Then, when building a circuit, the impact of all the repressors used in the circuit will be added to predict the impact of the full circuit. Our analysis showed that each repressor had a unique footprint on the genome, especially when it was expressed (Figure 30 for BM3R1 and Figure 31 for PhlF). However, we noticed that when a double-sensor strain was used for subtraction (division), the result was different from single-sensor reference for PhlF repressor (Figure 32 for BM3R1 and Figure 33 for PhlF). This was an interesting observation. We collaborated with MIT probabilistic computing group to further investigate this finding using their machine learning approaches (see Section 2.4.12).

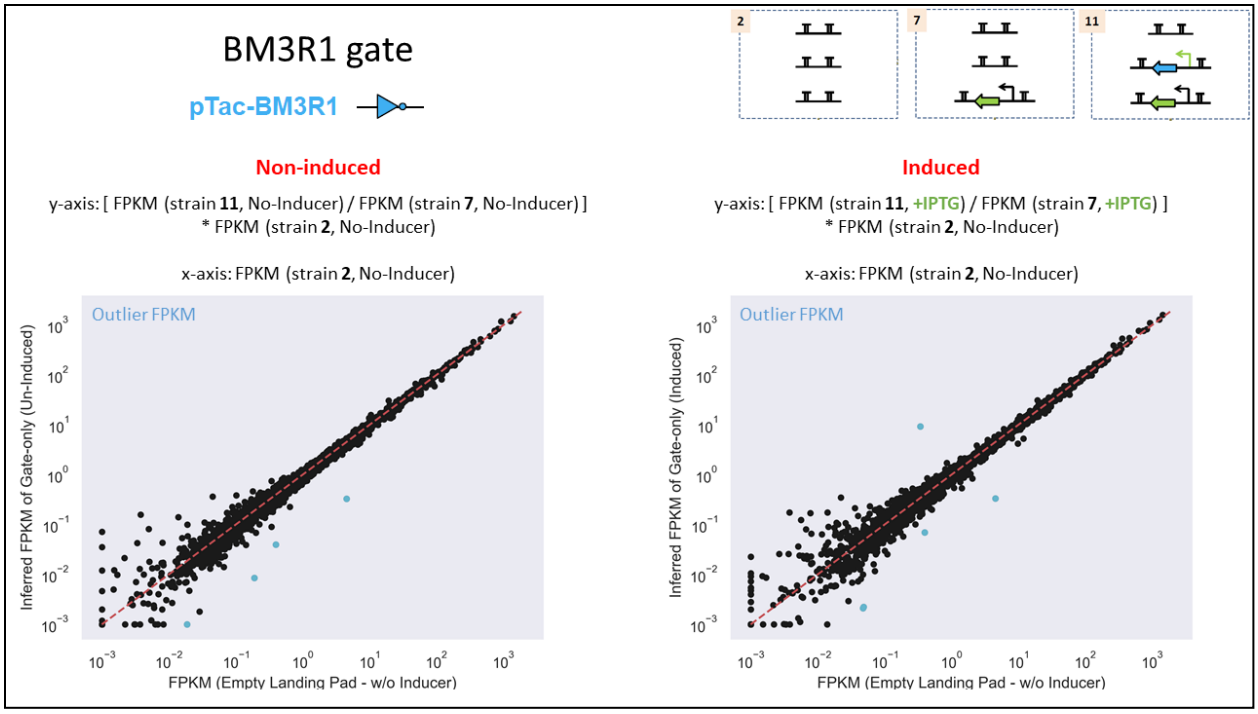


Figure 30. Exclusive impact of BM3R1 gate under induced and non-induced conditions. The quantification formula is shown. Schematics of strains used in the formula are shown in top right. Each box represents a strain with three landing pads in the genome (from bottom to top: sensor landing pad, repressor landing pad, YFP landing pad). The x-axis is the measured genomic FPKM data from the control strain 2 (empty landing pads), and y-axis data is the predicted genomic FPKM of the repressor. Outlier genes with significant deviation from x=y are marked with blue color.

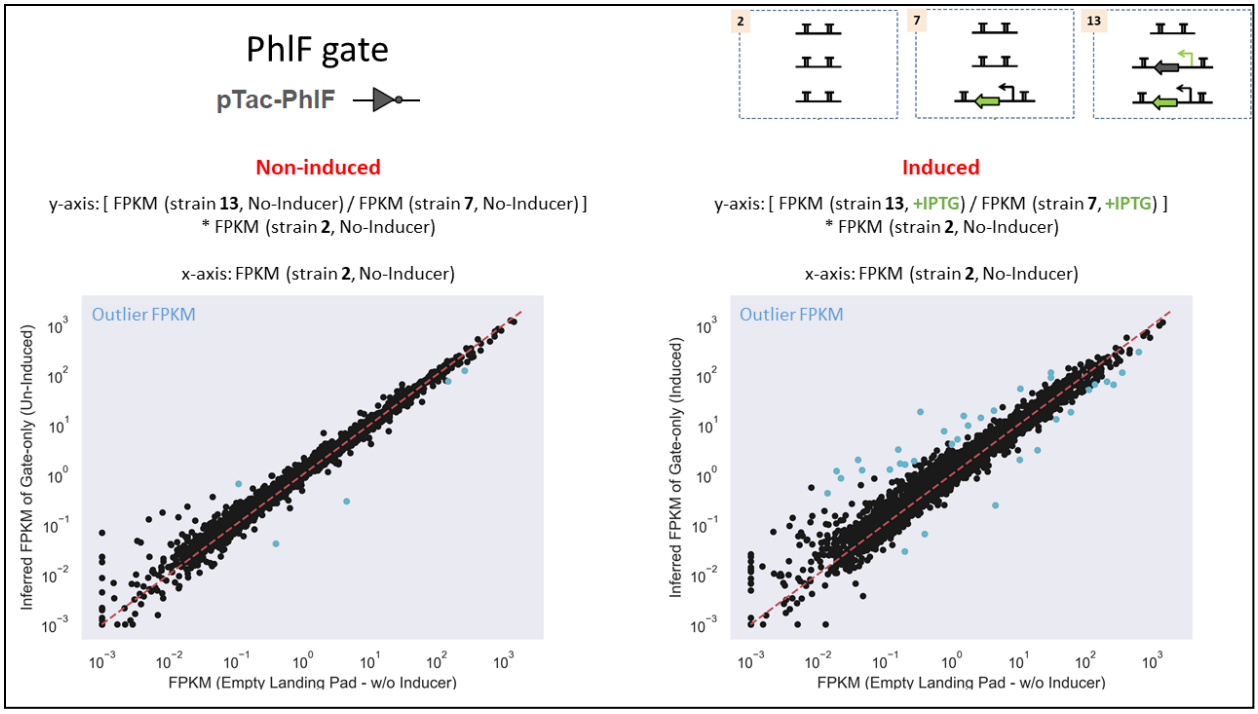


Figure 31. Exclusive impact of PhlF gate under induced and non-induced conditions. The quantification formula is shown. Schematics of strains used in the formula are shown in top right. Each box represents a strain with three landing pads in the genome (from bottom to top: sensor landing pad, repressor landing pad, YFP landing pad). The x-axis is the measured genomic FPKM data from the control strain 2 (empty landing pads), and y-axis data is the predicted genomic FPKM of the repressor. A long list of outlier genes with significant deviation from x=y are marked with blue color in the induced-state of PhlF.

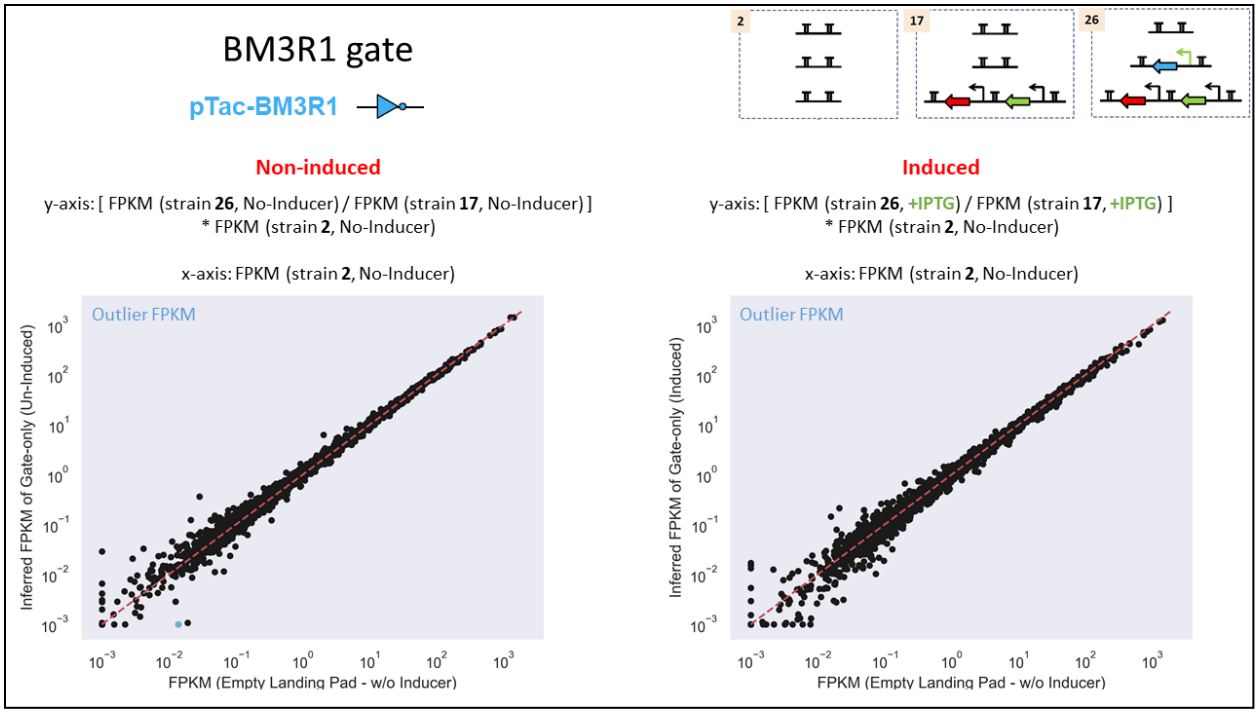


Figure 32. Exclusive impact of BM3R1 gate under induced and non-induced conditions. In this plot, a strain with double sensors (strain 17) is used for removing the effect of sensors. The quantification formula is shown. Schematics of strains used in the formula are shown in top right. Each box represents a strain with three landing pads in the genome (from bottom to top: sensor landing pad, repressor landing pad, YFP landing pad). The x-axis is the measured genomic FPKM data from the control strain 2 (empty landing pads), and y-axis data is the predicted genomic FPKM of the repressor. Outlier genes with significant deviation from x=y are marked with blue color (no outlier was identified in this case).

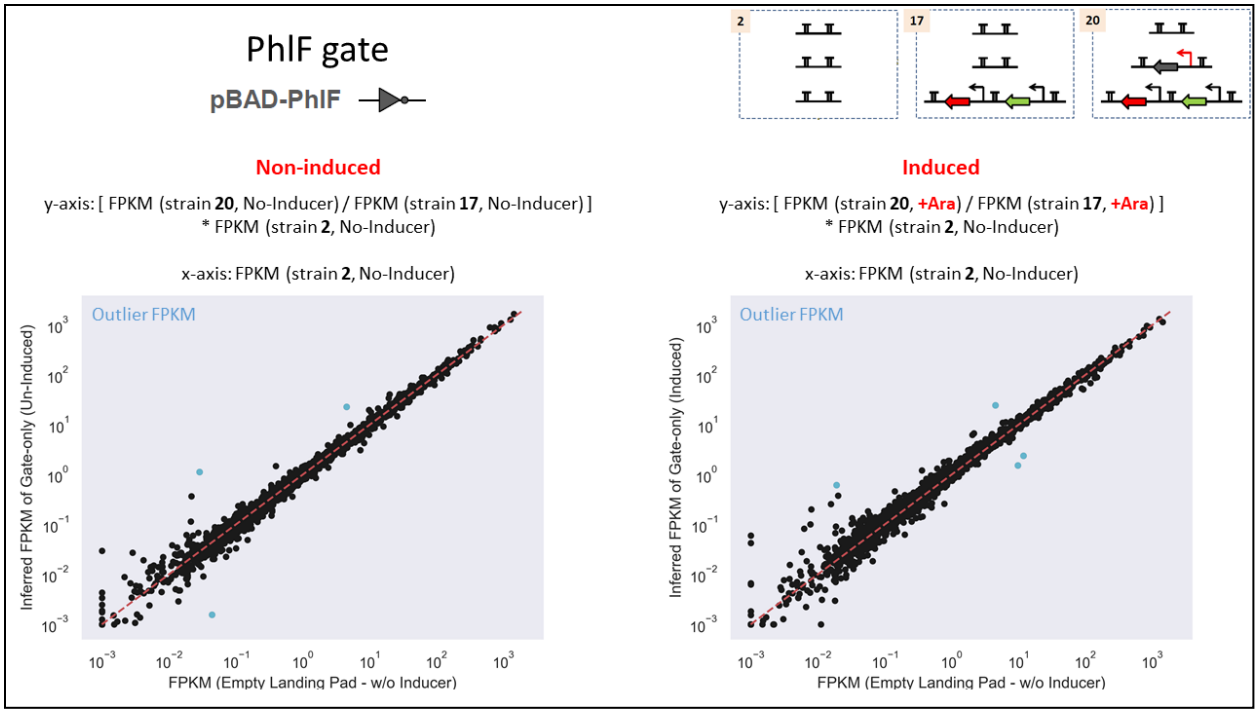


Figure 33. Exclusive impact of PhlF gate under induced and non-induced conditions. In this plot, a strain with double sensors (strain 17) was used for removing the effect of sensors. The quantification formula is shown. Schematics of strains used in the formula are shown in top right. Each box represents a strain with three landing pads in the genome (from bottom to top: sensor landing pad, repressor landing pad, YFP landing pad). The x-axis is the measured genomic FPKM data from the control strain 2 (empty landing pads), and y-axis data is the predicted genomic FPKM of the repressor. Outlier genes with significant deviation from x=y are marked with blue color (much fewer outlier genes were identified in this case).

1.4.9 Identifying exclusive genomic targets for each repressor in NAND circuits

After analyzing RNA-seq data, we identified a set of exclusive genomic targets for each repressor in NAND2.0 dataset. To do that, we used the measured RNA-seq data of strains that carry two sensors and one repressor only (Strain #20, #22, #24, and #26 for PhlF, PsrA, AmeR, and BM3R1 repressors, respectively). We then found those genes that have more than 2-fold change in FPKM (average of 4 replicates) with respect to a reference strain (strain #17 which includes only two sensors AraC and lacI) under appropriate induction conditions. We repeated this calculation for three strains (#28, #30, and #32) carrying three different NAND circuits (combination of two repressors where both are expressed upon double induction). For each NAND circuit, we created a list of genes that were regulated (both up and down) by the circuit and its individual repressors (Figure 34, Figure 35, and Figure 36). We colored each target gene by the color of its corresponding repressor. However, target genes were colored only if they were exclusively regulated by only one repressor and were also correctly regulated in the NAND circuit. Genes that were regulated by individual gates and not by the NAND circuit were colored black.

We found multiple target genes that did not follow additivity assumption of the impact (black colored genes), suggesting other hidden factors in their regulation. We also found that BM3R1 and PsrA repressors result in the least and most target genes in the genome, respectively.

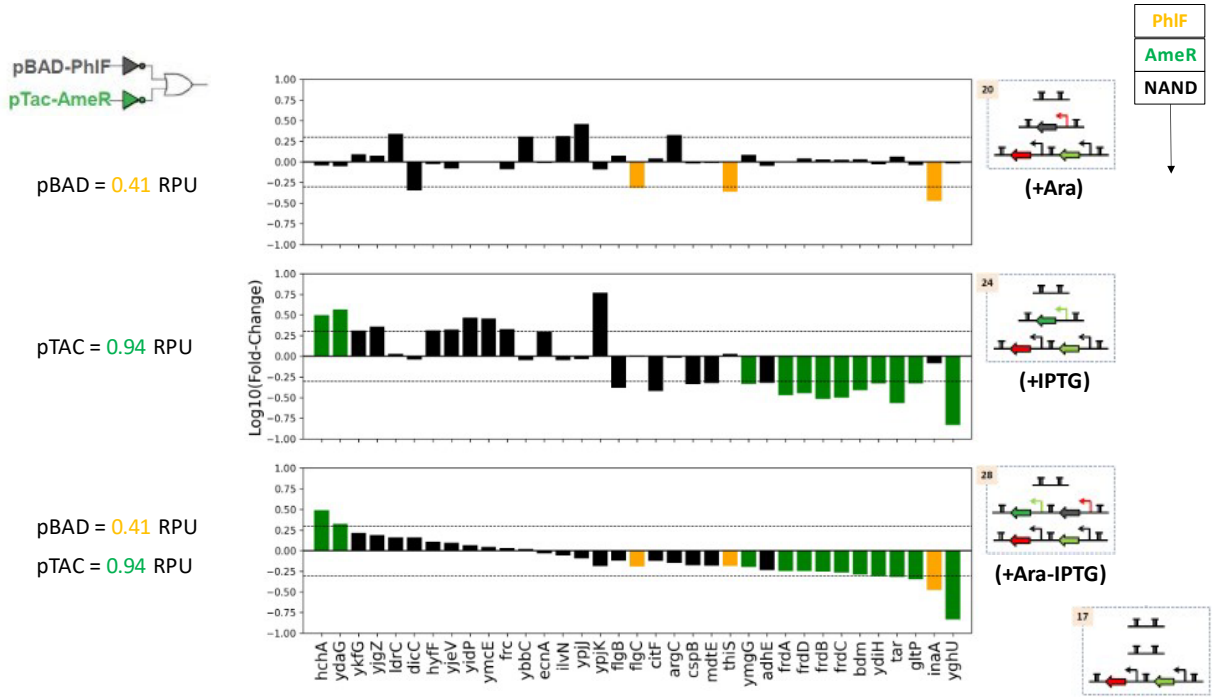


Figure 34. List of regulated genes in PhlF-AmeR NAND circuit. Orange and green colors represent exclusive target genes for PhlF and AmeR, respectively. The strains used to identify the targets genes are shown as dashed box next to each plot, and the reference strain is shown in bottom right corner. The strength of promoters expressing each repressor is also shown in RPU.

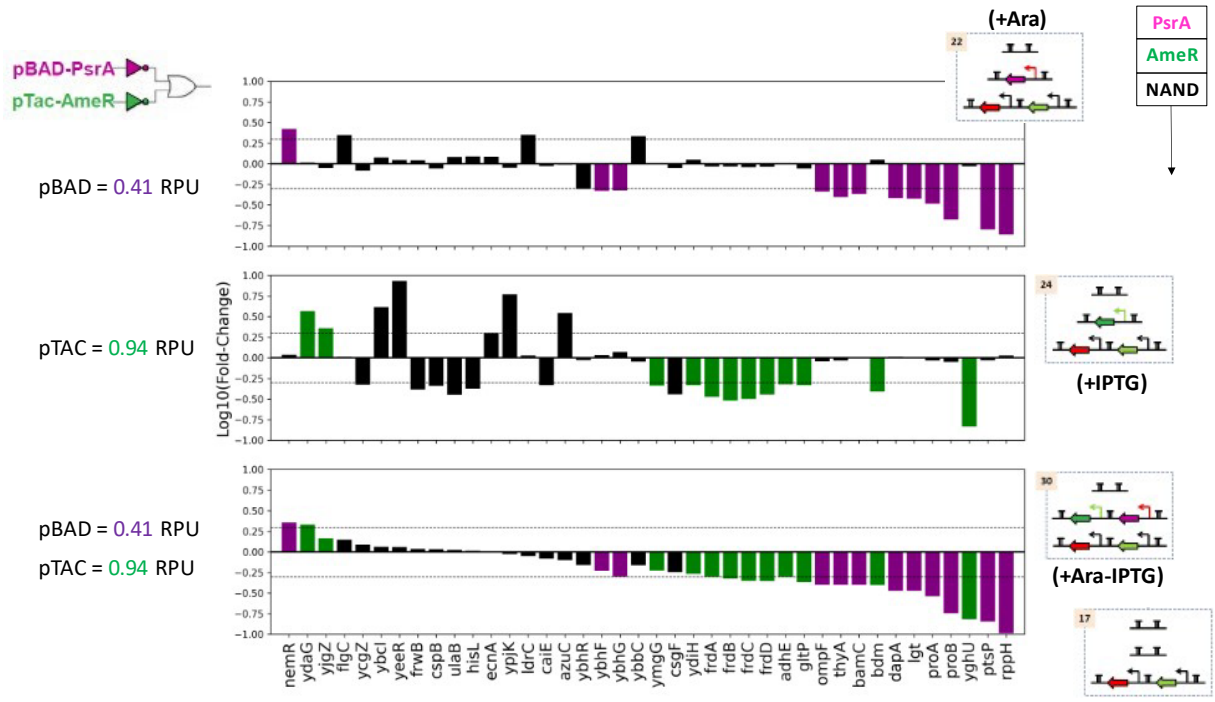


Figure 35. List of regulated genes in PsrA-AmeR NAND circuit.
Purple and green colors represent exclusive target genes for PsrA and AmeR, respectively. The strains used to identify the targets genes are shown as dashed box next to each plot, and the reference strain is shown in bottom right corner. The strength of promoters expressing each repressor is also shown in RPU.

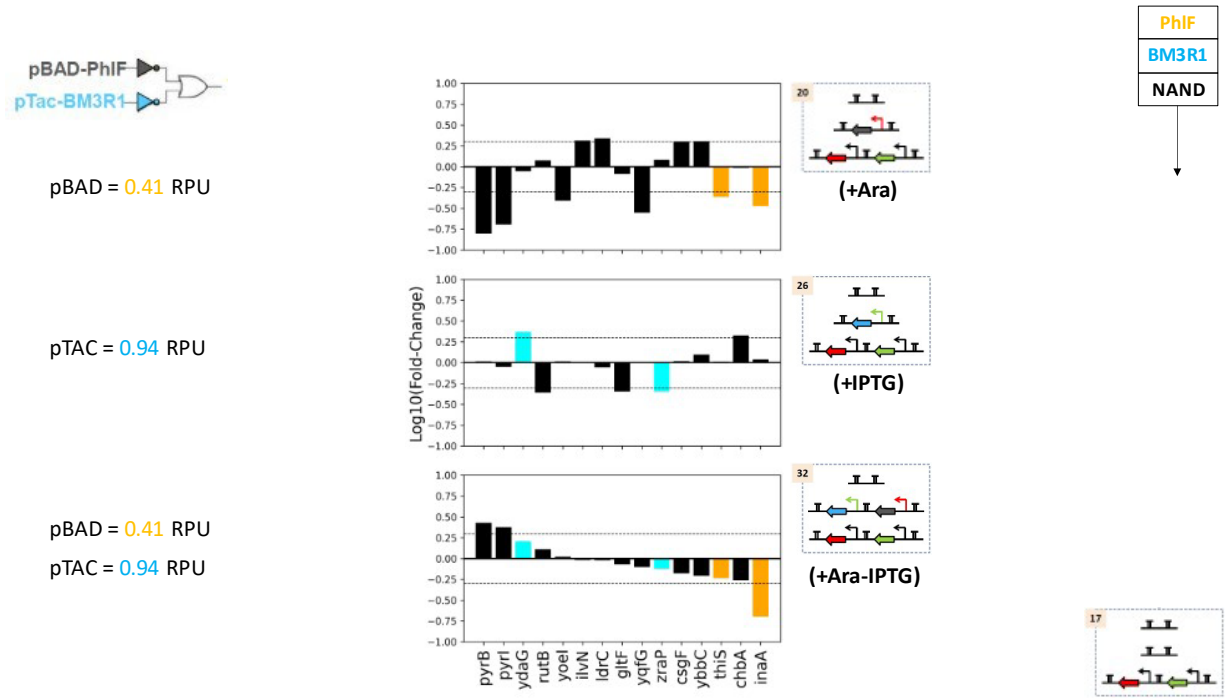


Figure 36. List of regulated genes in PhlF-BM3R1 NAND circuit. Orange and light blue colors represent exclusive target genes for PhlF and BM3R1, respectively. The strains used to identify the targets genes are shown as dashed box next to each plot, and the reference strain is shown in bottom right corner. The strength of promoters expressing each repressor is also shown in RPU.

1.4.10 Mapping non-specific impacts of circuits on the host cell using Ribo-seq

The ribosome profiling data reported above can be used to generate the complete map of non-specific impacts of the circuit on the host cell. Using the ribosome density values (which is a measure of protein expression levels), we can search for the correlations between the repressor proteins in the circuit and the native genes on the genome. Our analysis showed that the non-specific impacts are stronger than expected and exist for almost all the repressors in the circuit (Figure 37). Interestingly, the correlations follow a Hill function, suggesting that the non-specific impacts have a DNA-binding nature. Using this information, we can generate a genome-wide map of circuit impacts on the host cell (Figure 38). This information can guide the future design of genetic circuits in order to have fewer non-specific interactions with the host cell.

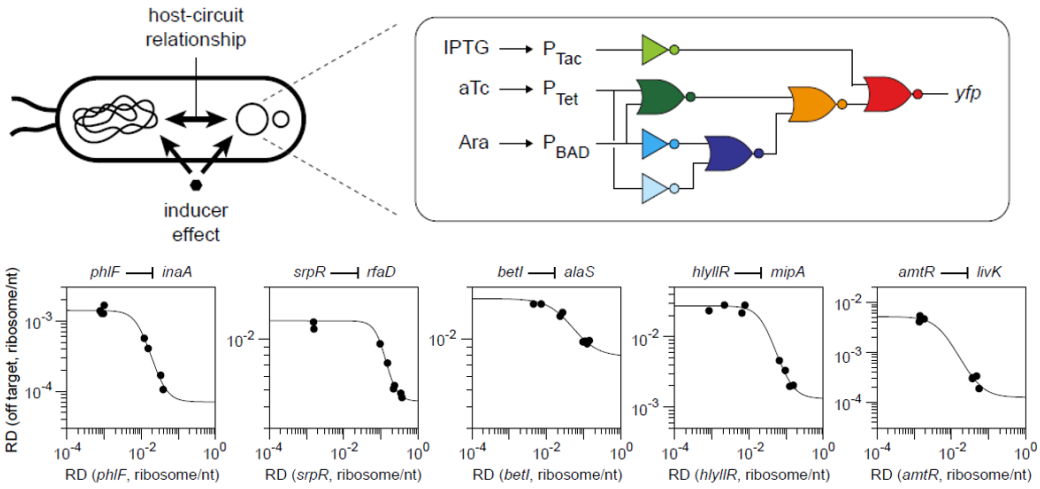


Figure 37. Off-target activity of all repressors in the circuit. Examples of non-specific interactions between circuit's repressor proteins and the native genes on E. coli genome. Genes are correlated according to a Hill function, suggesting a regression mode of gene regulation.

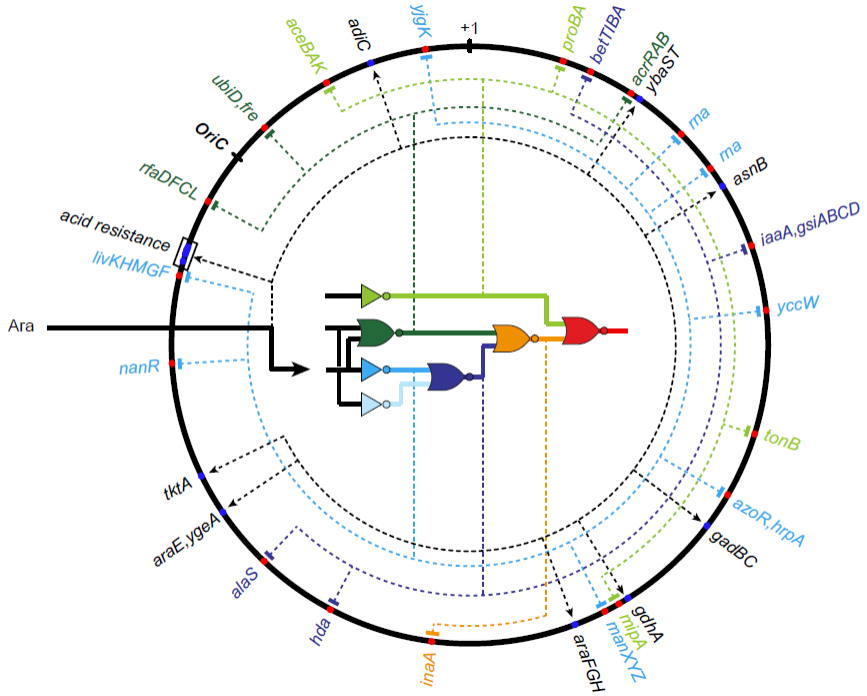


Figure 38. A complete map of circuit's non-specific impacts on E. coli genome. Dashed lines with arrow heads mean activation, while dashed lines with stopping-line head represent repression.

1.4.11 Visualizing circuit's impact on *E. coli* metabolism using metabolomics data

Our RNA-seq and Ribos-seq data showed large changes in the expression of endogenous enzymes involved in the host cell metabolism, especially in the TCA cycle, which is a main source of energy (ATP) for the cell through oxidative phosphorylation. Our results showed that as the expression of circuit genes increased, the expression of genes in the TCA cycle decreased, likely to avoid unnecessary gene expression and to save cellular resources (Figure 39).

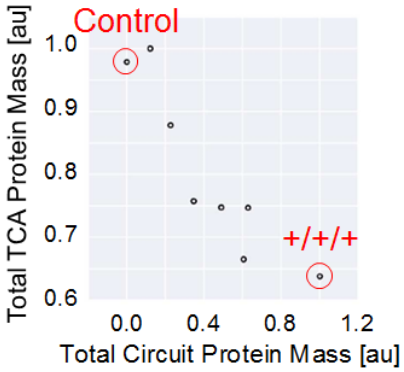


Figure 39. Impact of the circuit on TCA cycle enzyme levels.
The total protein mass going toward TCA cycle decreases upon increase in the total protein mass in the genetic circuit. +/+/+ is the state of the circuit with the maximal gene expression (with IPTG/ aTc/ arabinose induction). Protein mass is calculated by multiplying the Ribo-seq expression of each gene by the molecular weight of its protein product. This value is then normalized across all the genes in the genome and the circuit.

To further corroborate these results and to investigate the true impact of the circuit on host cell metabolism, we generated the first ever metabolomics dataset for a large genetic circuit (same circuit as above). This information-rich dataset provided data for >600 metabolites both inside the cell and in the media, including peptides, nucleotides, energy-related metabolites, cofactor, lipids, etc., (Figure 40).

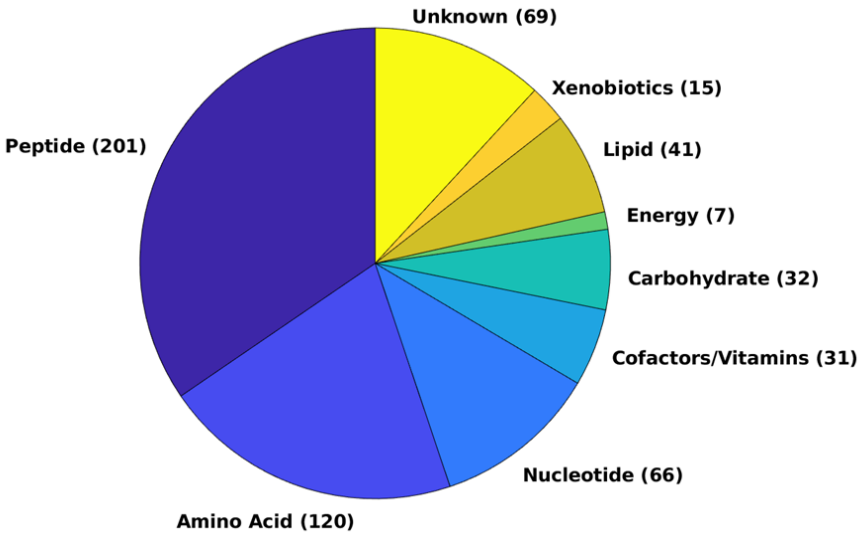


Figure 40. Breakdown of the metabolites detected in our metabolomics study.

The most interesting results in this metabolomics dataset was that the concentration of key metabolites involved in the TCA cycle were all reduced upon maximally expressing genetic circuit, which is consistent with our RNA/Ribo-Seq findings (Figure 41).

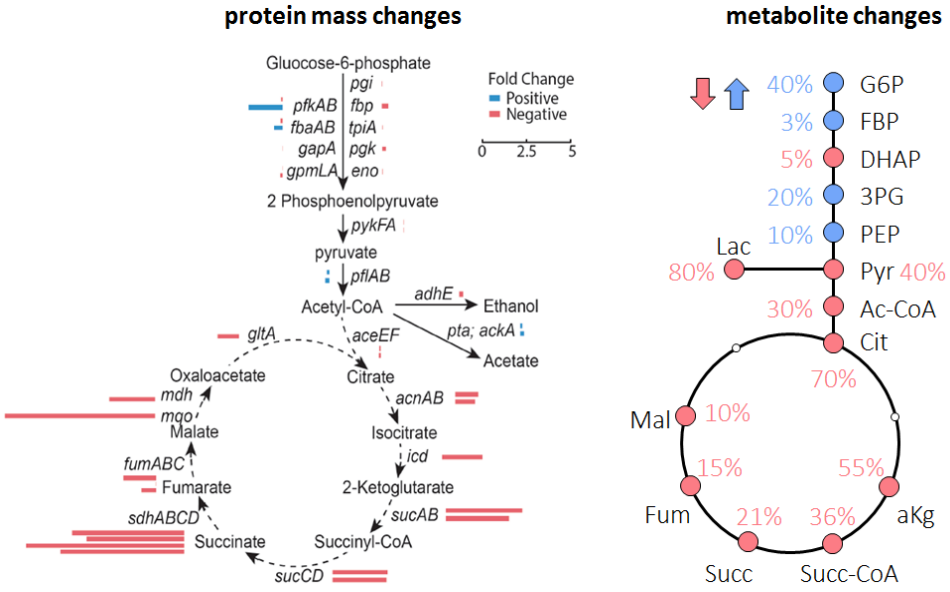


Figure 41. Metabolomics data confirms the Ribo-seq results.
As the expression of enzymes in TCA decreases, the concentration of metabolites involved in their enzymatic reaction also decreases. Plot are generated by comparing +/+/+ state with the control.

The above result motivated us to develop a whole-cell metabolic network of *E. coli* (Figure 42). The purpose of this map was to illustrate the distribution of metabolic fluxes inside the cell under different conditions. In addition, this map can be used to visualize the distribution of gene expression levels between metabolic pathways. This map will be very useful for researchers who use transcriptomics and metabolomics data. There are more than 1000 metabolic reactions and 1000 metabolites shown in this map. Not all the metabolites are present in our metabolomics data (white circles). Only those colored with red and blue are present, and the size of the circle shows the fold-change of the metabolite levels with respect to a control sample (red and blue colors mean increase and decrease, respectively). We used FBA (flux balance analysis) to infer the metabolic fluxes across the network and calculate the difference of predicted fluxes between the circuit and control sample (shown by blue and red lines). The thickness of the lines means higher difference in fluxes (red and blue means increase and decrease in flux, respectively). Reactions that have zero flux are shown with white color. To calculate these fluxes, we used our metabolomics data to quantify the rate of exchange of metabolites between inside and outside the cell, and that was used in FBA to quantify the intracellular fluxes that maximize the biomass production. For the maximally induced state of the circuit (+/+/+), FBA also predicted a reduced growth rate than the control sample, which is similar to our measurement. It is clear from this map that majority of fluxes going through TCA cycle, respiration, amino acids biosynthesis pathways, and nucleotide biosynthesis pathways are reduced in state +/+/+ of the circuit, whereas fatty acid production is activated. Interestingly, most of amino acid and nucleotide biosynthesis pathways are not highly active, which could be because we added excess amount of amino acids to the media (in the form of casamino acids) and cells don't see a need to activate these energy-intensive biosynthesis pathways.

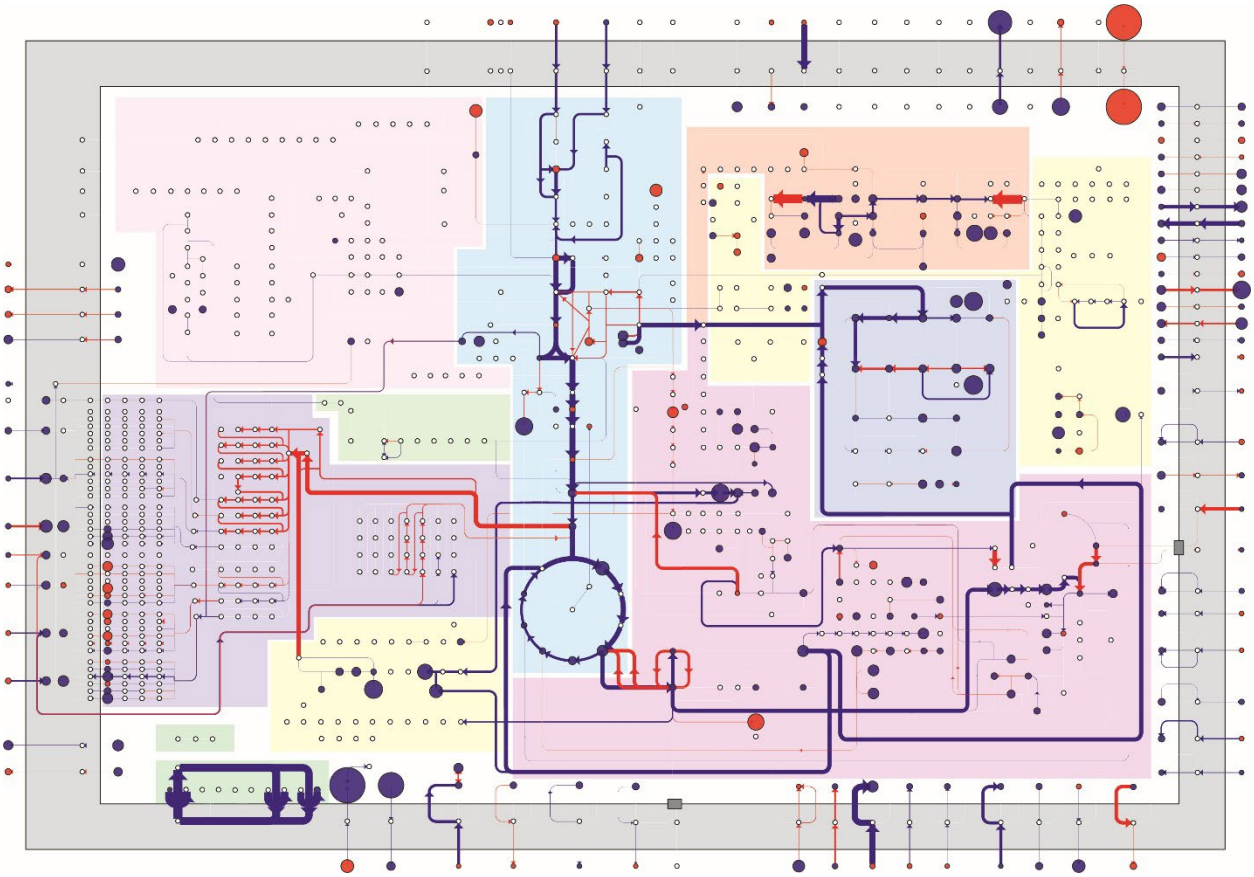


Figure 42. The metabolic network of E. coli.
Red and blue lines are the difference (increase and decrease) in metabolic fluxes of circuit versus control sample, inferred by FBA. Red and blue circles are metabolites and show the fold-increase and decreases in metabolite levels with respect to a control sample, respectively. The white lines are those with zero predicted fluxes or no flux changes. The white circles are those metabolites that do not exist in our metabolomic data.

1.4.12 **Whole genome simulator**, a data-driven probabilistic model of circuit impact on host

We analyzed all RNA-seq data from NAND 2.0 experiment in a collaboration with MIT probabilistic computing group (point of contact: Ulrich Schaechtle). The objective of this collaboration was to be able to develop a probabilistic model that is trained solely using RNA-seq data and without any prior knowledge of *E. coli* network. We therefore developed the *whole genome simulator* software that can learn gene-gene interactions using a Bayesian data-driven approach *without the prior knowledge of E. coli network*. The software was trained on all RNA-seq data in NAND2.0, except for three NAND circuits that were held-out for evaluation purposes. The Bayesian inference model was proven to be highly accurate (Figure 43).

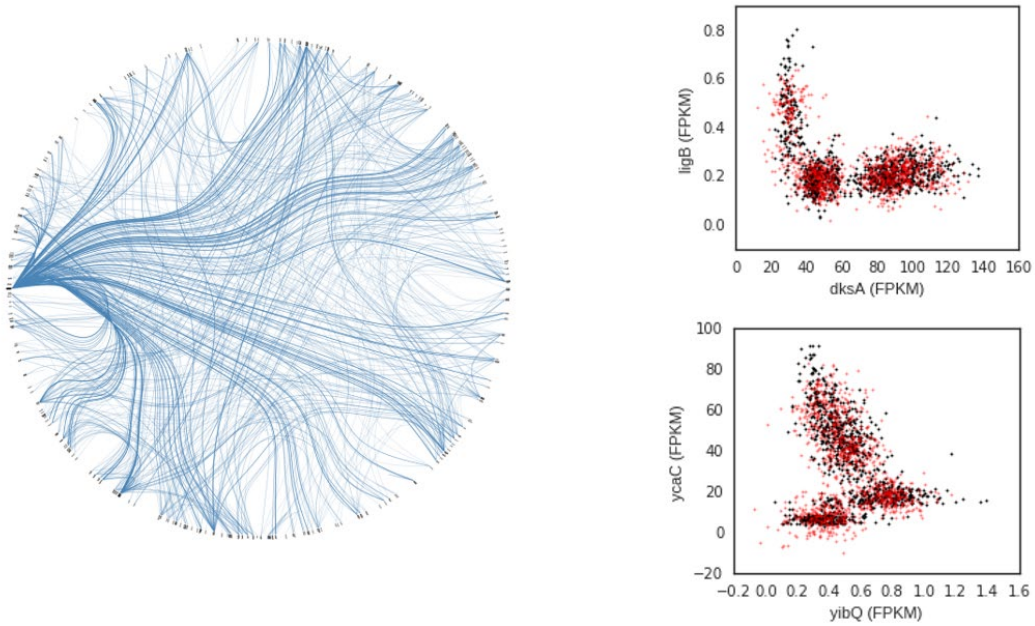


Figure 43. Inferred gene-gene interactions of whole genome simulator. (left) The gene-gene interaction network of E. coli MG1655 is shown in a circular format. Nodes are the exact location of genes in the genome. (right) Two examples of model-predicted gene expressions from samples used in training. Black and red dots are the experimental and predicted FPKM values, respectively.

The automated Bayesian model-generator of whole-genome simulator computes the probability of gene-gene interactions based on a Markov-Chain Monte-Carlo algorithm (Figure 44). The variables in this model included the expression of all the genes (both native *E. coli* genes as well as synthetic genes LacI, AraC, AmeR, BM3R1, PhlF, PsrA, YFP), the presence or absence of inducer chemicals (arabinose, IPTG), and growth time points (exponential log phase, stationary phase). We had more than 1000 RNA-seq samples in NAND 2.0 experiment each containing more than 4000 genes expression, thus giving us more than 4 million data points for training our probabilistic model. An important feature of this model is that it is based on conditional probabilities, meaning that it can predict new gene expressions in *E. coli* if the growth conditions

are changed. For example, we can predict the expression of all native *E. coli* genes in a scenario that arabinose is present and synthetic genes LacI, AraC, and PhlF are highly expressed. As a result, this capability allows us to predict the impact of NAND circuit expression on all native *E. coli* genes.

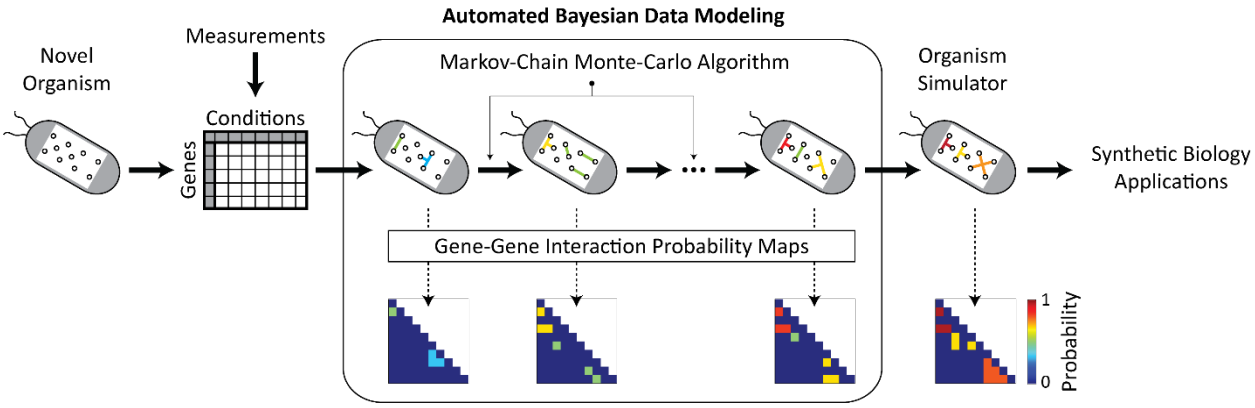


Figure 44. Overview of the internal working of whole genome simulator.

After completing the model training step, we conducted series of calculations in which we simulated the impact of NAND circuits on *E. coli*. Figure 45 shows the decomposition of one of the NAND circuits (strain #29) into its individual building blocks. This circuit consists of two sensors (LacI, AraC), two repressors (PhlF, AmeR), and a reporter (YFP). We predicted the impact of the NAND circuit (strain #29) under all combinatorial induction conditions (arabinose, IPTG) and two time points (4x2 = 8 conditions). Overall, model was able to predict the expression of all native *E. coli* genes with relatively high accuracy. For example, when both arabinose and IPTG were present, and when growth was at exponential log phase, the model was able to predict 85% of native genes within 2-fold error (Figure 46). This is similar to the accuracy of the model when predicting the expression of native genes in the Wild-Type strain of *E. coli* under the exact same conditions (Figure 47).

However, when we switched the growth conditions to the stationary phase of growth, model accuracy dropped significantly for the NAND circuit (strain #29) with only 58% of native genes were predicted within 2-fold error (Figure 48). This error margin is much larger than Wild-Type *E. coli* which showed 75% accuracy (Figure 49). These results demonstrated to us that model was not trained well enough when circuit components were expressed at stationary phase of growth. Therefore, predicting circuit impact with high accuracy under these conditions required further optimization of model parameters.

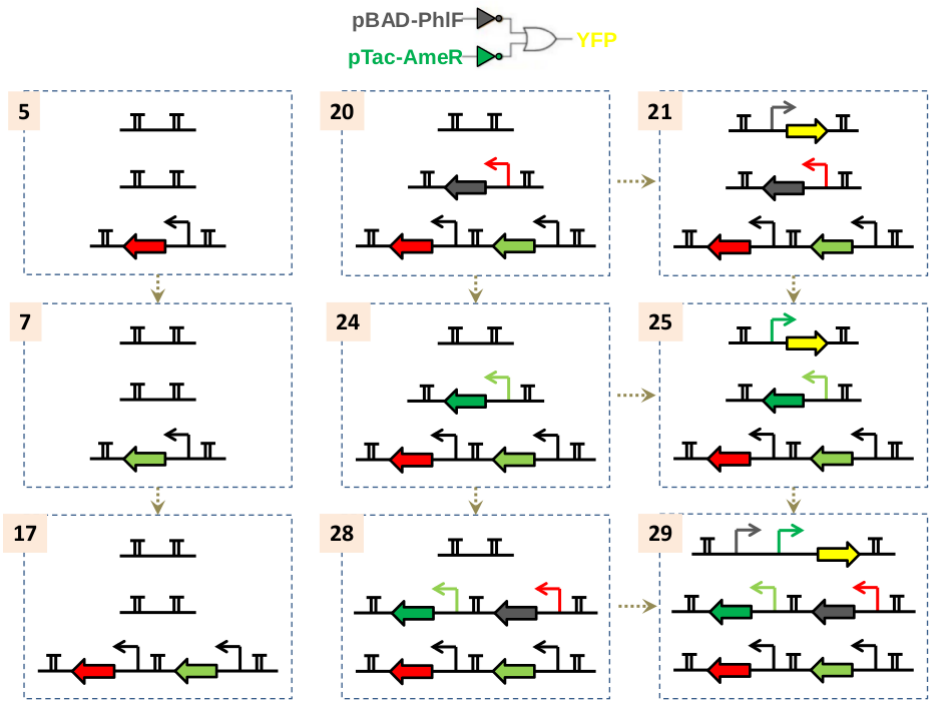


Figure 45. Decomposition of a NAND circuit into its building block components.

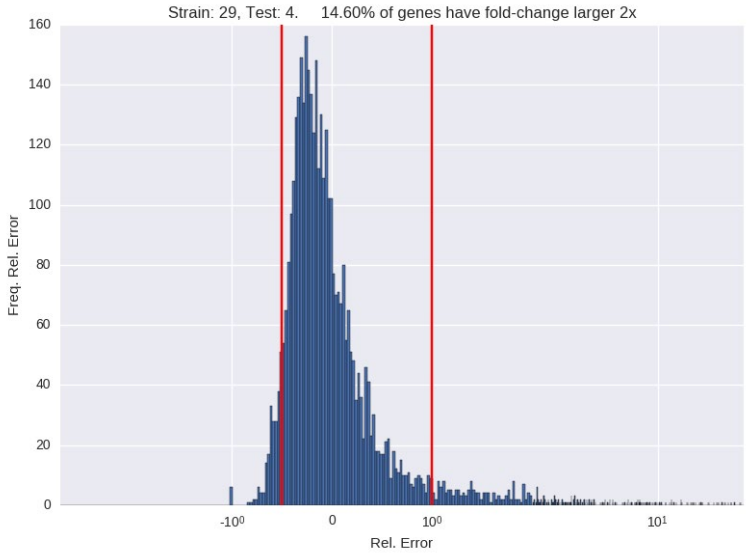


Figure 46. Accuracy of whole genome simulator for a NAND circuit in exponential growth. The distribution of relative errors for predicted expression of all native genes in strain 29 when both IPTG and arabinose are present and growth is at exponential log phase. Red lines depict the 2-fold error thresholds.

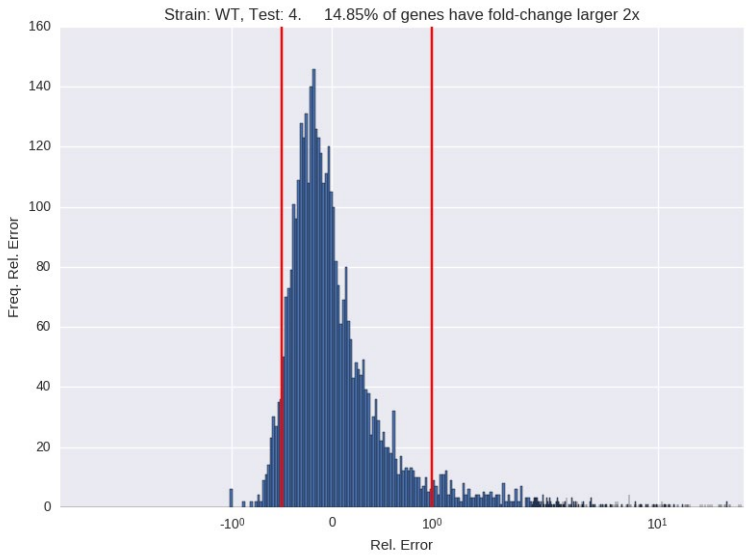


Figure 47. Accuracy of whole genome simulator for wild type E. coli in exponential growth. The distribution of relative errors for predicted expression of all native genes in WT strain when both IPTG and arabinose are present and growth is at exponential log phase. Red lines depict the 2-fold error thresholds.

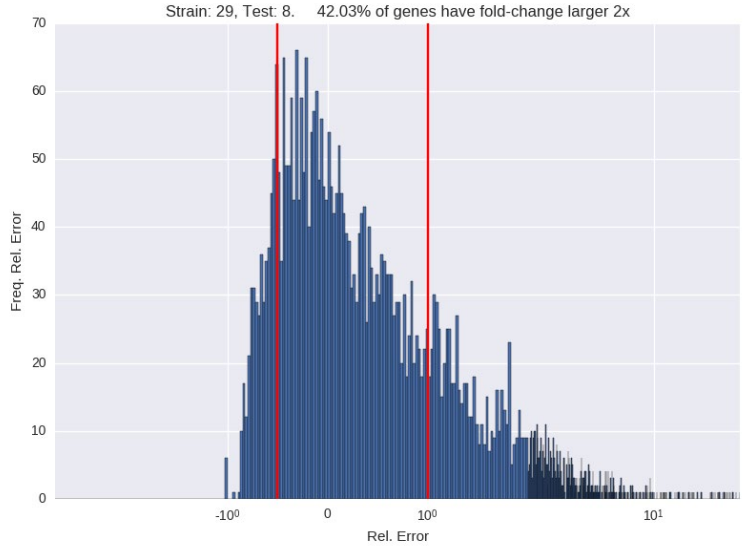


Figure 48. Accuracy of whole genome simulator for a NAND circuit at stationary phase. The distribution of relative errors for predicted expression of all native genes in strain 29 when both IPTG and arabinose are present and growth is at stationary phase. Red lines depict the 2-fold error thresholds.

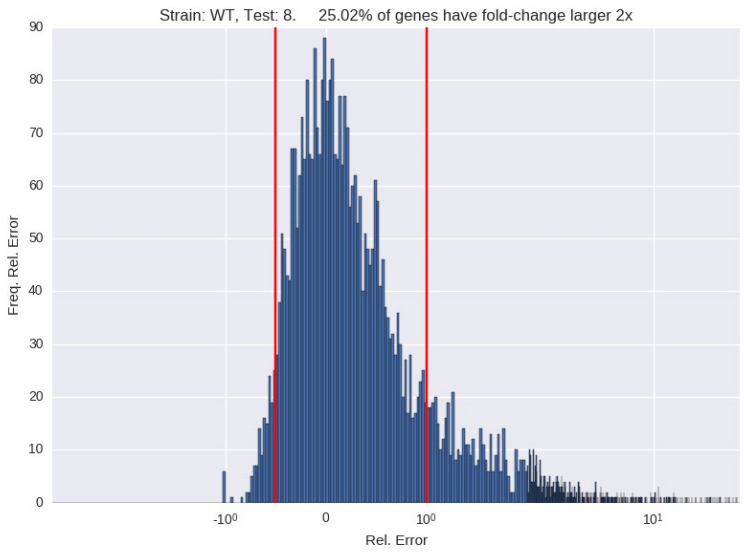


Figure 49. Accuracy of whole genome simulator for wild type E. coli at stationary phase. The distribution of relative errors for predicted expression of all native genes in WT strain when both IPTG and arabinose are present and growth is at stationary phase. Red lines depict the 2-fold error thresholds.

Whole genome simulator can also predict the conditional gene expression between two conditions or strains. By simulating >1000 iterations, the model can generate a distribution of gene expression for any specific gene, where the mean of the distribution determines if the gene is up-regulated or down-regulated with respect to the reference strain. Figure 50 shows two examples of predicted gene regulation in one of the held-out strains. Overall, with a probability of >83%, our whole genome simulator can correctly predict the state of gene regulations (up or down) for genes within the held-out circuit strains (Figure 51).

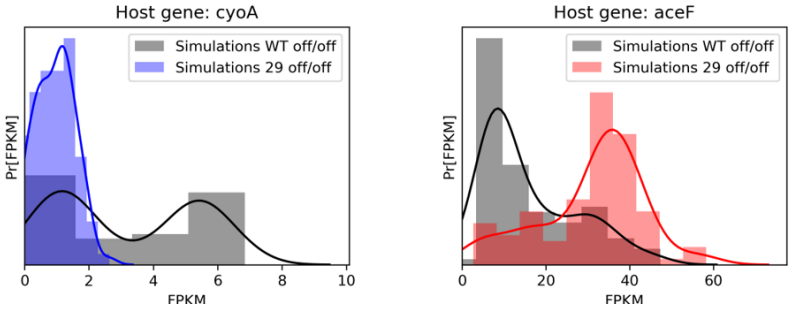


Figure 50. Simulating gene regulation by whole genome simulator. (left) An example of predicted down-regulated gene in the held-out circuit #29 with respect to wild type E. coli MG1655 (reference strain) in the absence of both inducers (IPTG and arabinose). (right) An example of predicted up-regulated gene in the held-out circuit #29.

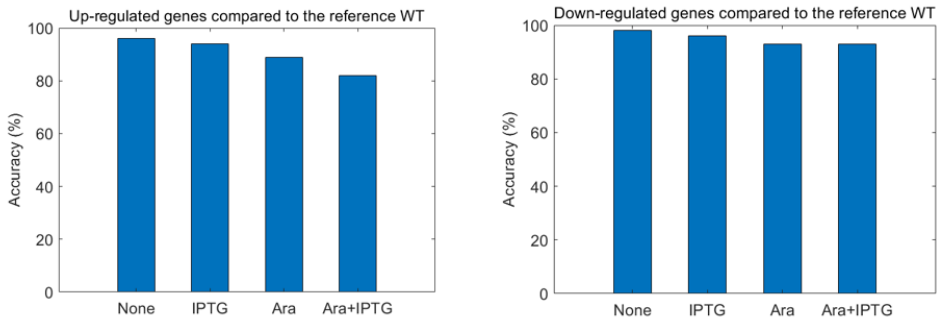


Figure 51. Accuracy of gene regulation prediction by whole genome simulator. Whole genome simulator can predict (with >83% accuracy) the up-regulation (left) and down-regulation (right) state of all the genes in the held-out circuits across all inducer conditions. The reference strain is wild type E. coli MG1655.

1.4.13 Building new type of Phage-based gates in *Bacillus subtilis*

We started a project for building new type of Phage-based gates in Bacillus subtilis. We began to transfer repressors to Bacillus subtilis to form a new set of orthogonal NOT gates. The repressors we chose to transfer were a set of phage repressors that we developed in Voigt lab for use in E. coli. Our first-pass attempt was to wholly transfer the repressor cassette from E. coli into B. subtilis. From previous experience with poor expression in B. subtilis, we did some mild engineering of the translational control elements (swapped in B. subtilis RBSs), but left the repressible-promoter the same as in E. coli. All gates were recombined into the AmyE locus in B. subtilis PY79. In the gate constructs, pCym (a cumate-inducible promoter) was used to drive phage repressor production. Downstream of the repressor, a phage-repressible promoter was responsible to drive GFP production. To characterize the gates, cells were brought into exponential phase, then induced and allowed to grow for another ~3 hours. Included controls were pCym (cumate-inducible GFP) and PY79 (cells without any genetic construct in them). The raw flow cytometry data for each of the strains are shown in Figure 52.

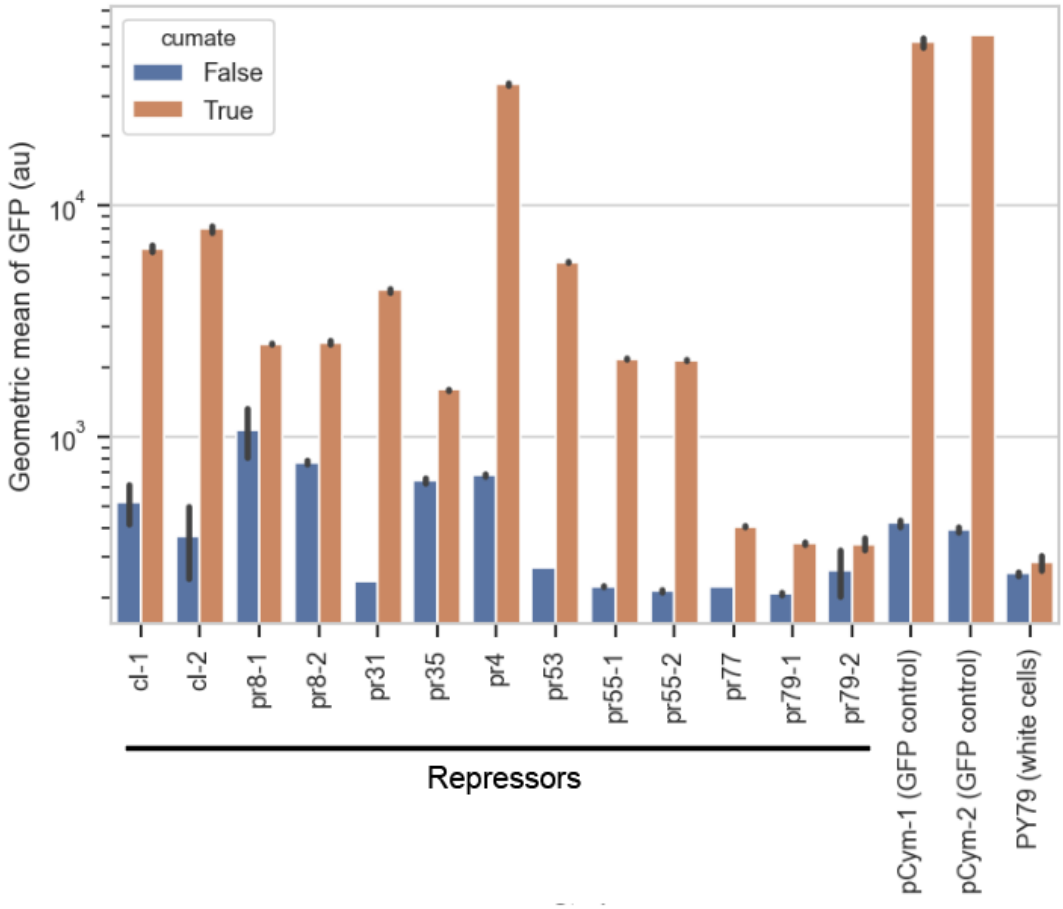


Figure 52. Characterization of Phage-based gates in B. subtilis. The measured fluorescence of sensor and phage-based gates in B. subtilis in the presence and absence of cumate inducer.

From the data above, it is apparent that none of the gates worked properly in this first-pass result. If they worked properly, we would see repression of the output upon induction via cumate. However, from these results, we have gained two key insights for engineering gates in B. subtilis:

(1) Phage-repressible promoters ported directly from E. coli have poor constitutive expression in B. subtilis. This can be seen in the similarity of GFP values between the uninduced repressors and the white cells (PY79). This indicates that E. coli promoters are not always likely to express well in B. subtilis despite their well-conserved sequence constraints.

(2) Transcriptional read through of E. coli terminators is a serious concern when profiling gates in B. subtilis. This is apparent from the high induction of the repressed promoters in the presence of cumate. Since the induced repressors lie directly upstream of the repressed promoters, it is likely that readthrough from the repressor is going directly into the repressed promoter.

1.4.14 Building circuits in *B. subtilis* using conjugation and obstacle course testing

The 4-day obstacle course as defined by DARPA was implemented at the Broad institute to test circuits in-house as well as to debug any process issues with the obstacle course experiments.

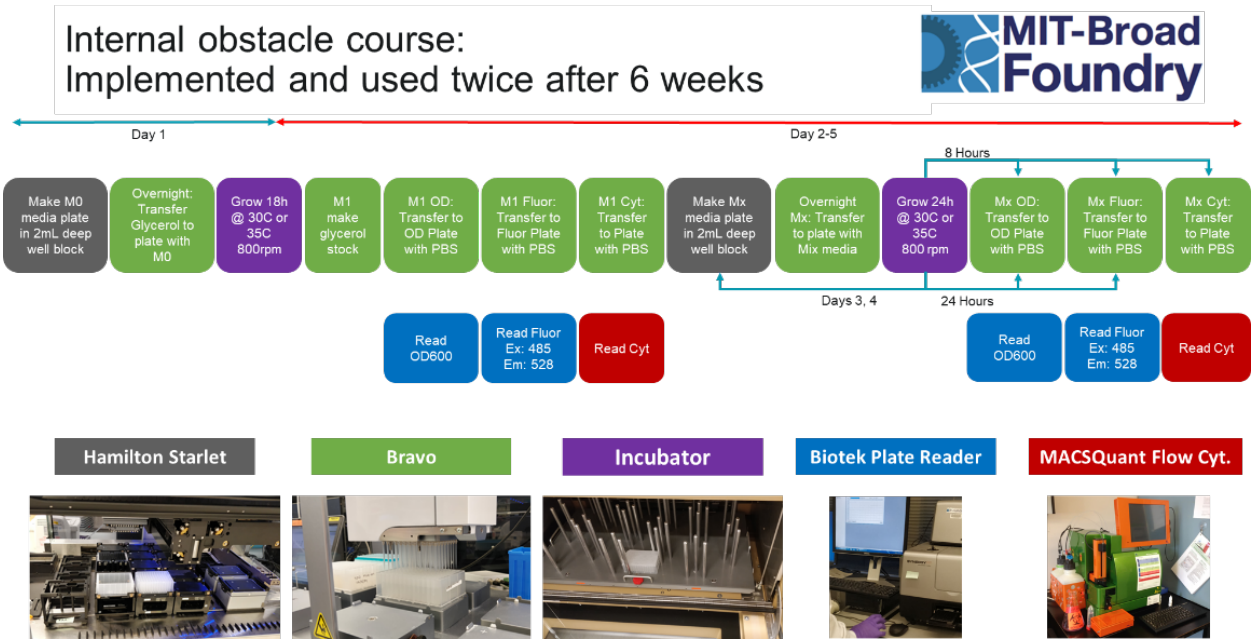


Figure 53. Schematic of 4-day obstacle course as implemented at the Broad institute.

Five circuits were constructed in *B. subtilis "donor"* derived from parent *B. subtilis 168*. These circuits included three OR functioning circuits and two IMPLY circuits, with their performance in the obstacle course shown below in Figure 54. Results were obtained using plate reader fluorescence 485nm / 530nm. The color of each bar denotes the expected state (green being ON and pink being OFF). The growth rate of the donor white cells is shown measuring plate reader OD600. Values written in green above each bar represent fold change in raw data compared with measured OFF state value. Due to the low growth rate, a selection of 8 other host strains or "recipients" were selected to trans-conjugate the circuits into and test in the modified M9 media. Figure 55 shows the circuit-recipient combinations selected for further testing based on initial results.

The circuits were tested according to the obstacle course protocol, which was immortalized in the form an Aquarium protocol to be reproducible in a third lab for potential testing in future. The tests were conducted at 35 ^{0}C at the BROAD institute and the results are shown in Figure 56, demonstrating multiple circuits passed the test, with some passing in more than one host. This shows the efficacy of this method in producing multiple viable circuit-hosting strains.

However, when testing was conducted at Strateos at 35 ^{0}C, the results had higher variability with white cells showing fluorescence. This suggests potential cross contamination across wells. The high variation in fluorescence values of technical replicates also suggests technical issues such inadequate mixing of a well or pipetting too low such that aggregates are collected and a source of error in measurement. These results can be shown in Figure 57.

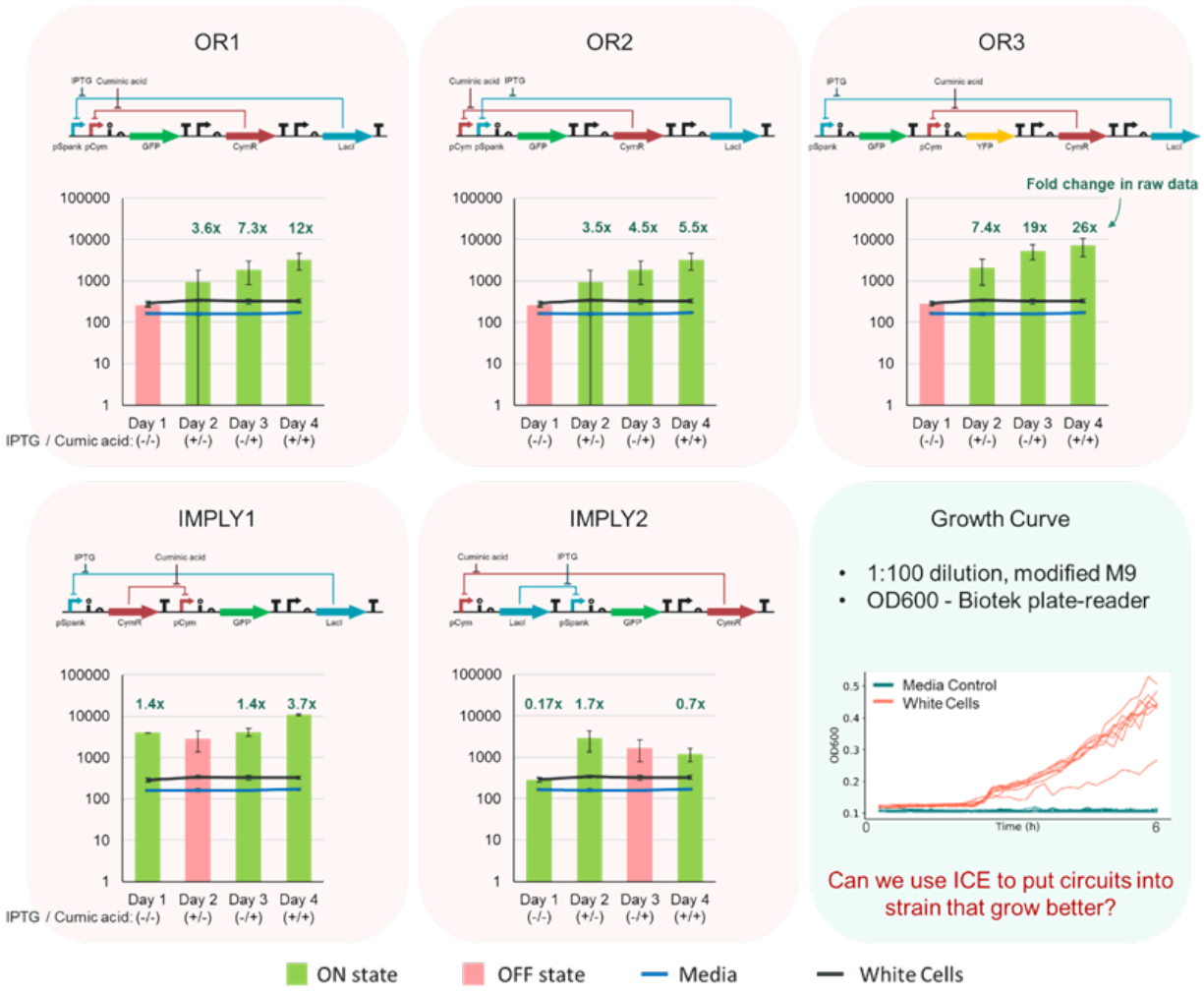


Figure 54. Performance of 5 circuits built in B. subtilis donor.

- Colonies were obtained from **all** pairs
- ✓ Selected based on initial results
- G design fail & constitutively on
- White cells for each strain was used as negative control

Recipient / Circuit	JAB 395	JAB 396	JAB 397	JAB 546	JAB 547	JAB 549	JAB 575	PY 79
OR_1		✓		✓				✓
OR_2		✓						✓
OR_3		G				✓		✓
IMPLY_1								✓
IMPLY_2				✓	✓		✓	✓
IPTG sensor				✓	✓			✓

Figure 55. Circuit-recipient strain combinations selected for obstacle course testing.

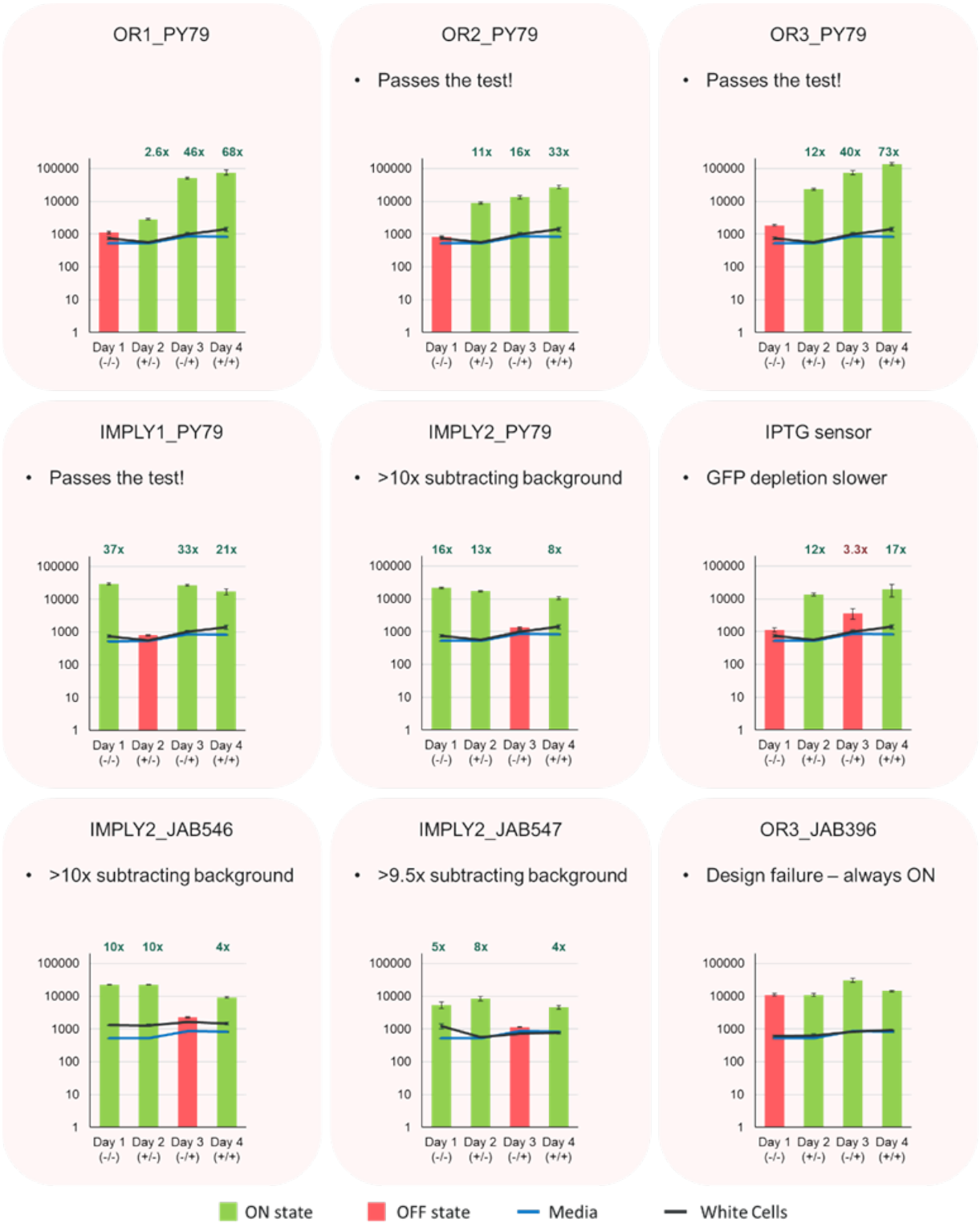


Figure 56. Plate reader fluorescence from obstacle course run at the BROAD institute at 35C

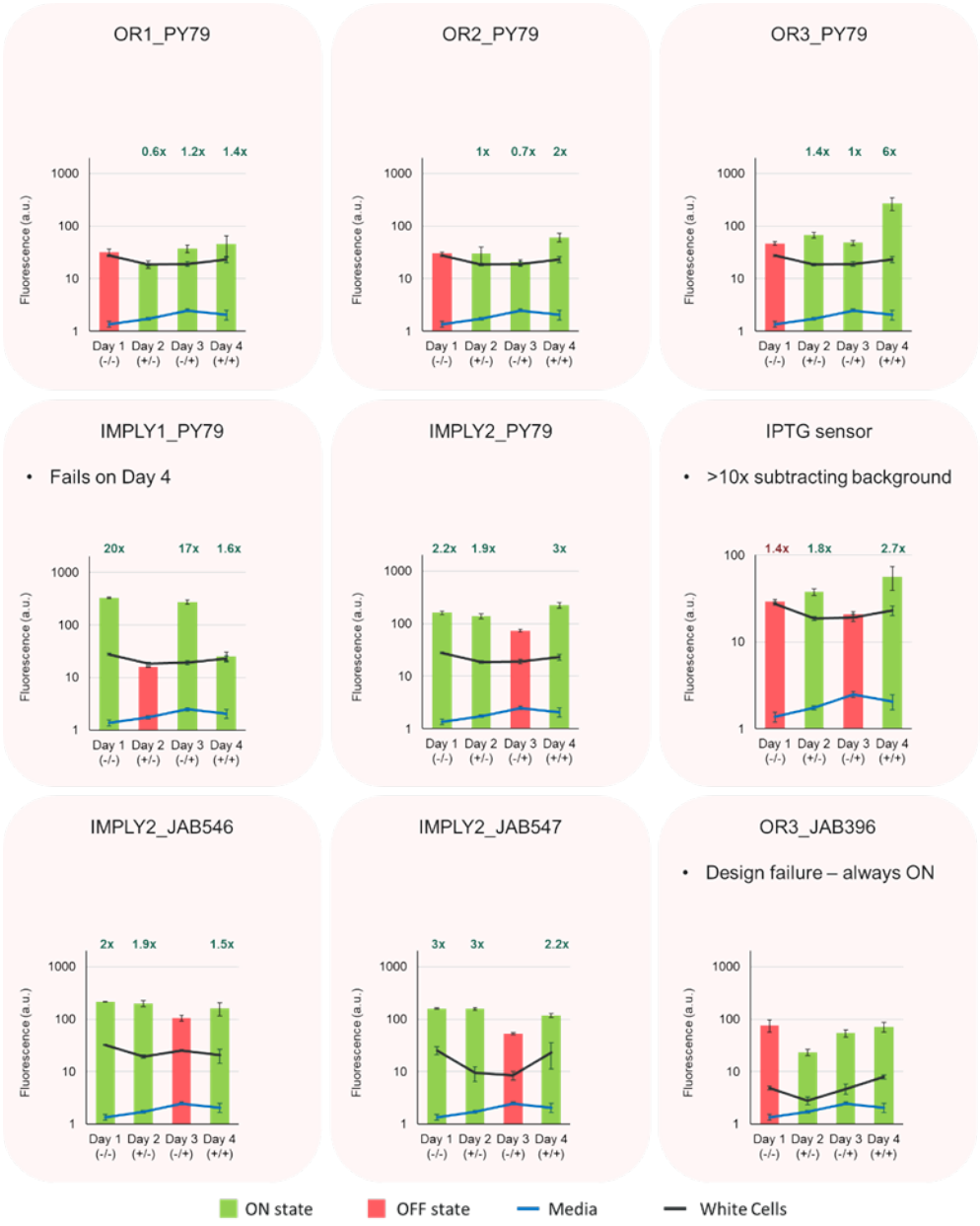


Figure 57. Plate reader fluorescence from obstacle course run at Strateos at 35C.

The 4 days obstacle course can be summarized as shown below in Figure 58. We highlight 3 circuits constructed in *B. subtilis "donor"* derived from parent *B. subtilis 168*. These circuits included one OR and two IMPLY circuits were transformed using mini-ICE to other soil strains and tested internally and externally. The circuits were tested according to the obstacle course protocol, which was immortalized in the form an Aquarium protocol to be reproducible in a third lab for potential testing in future.

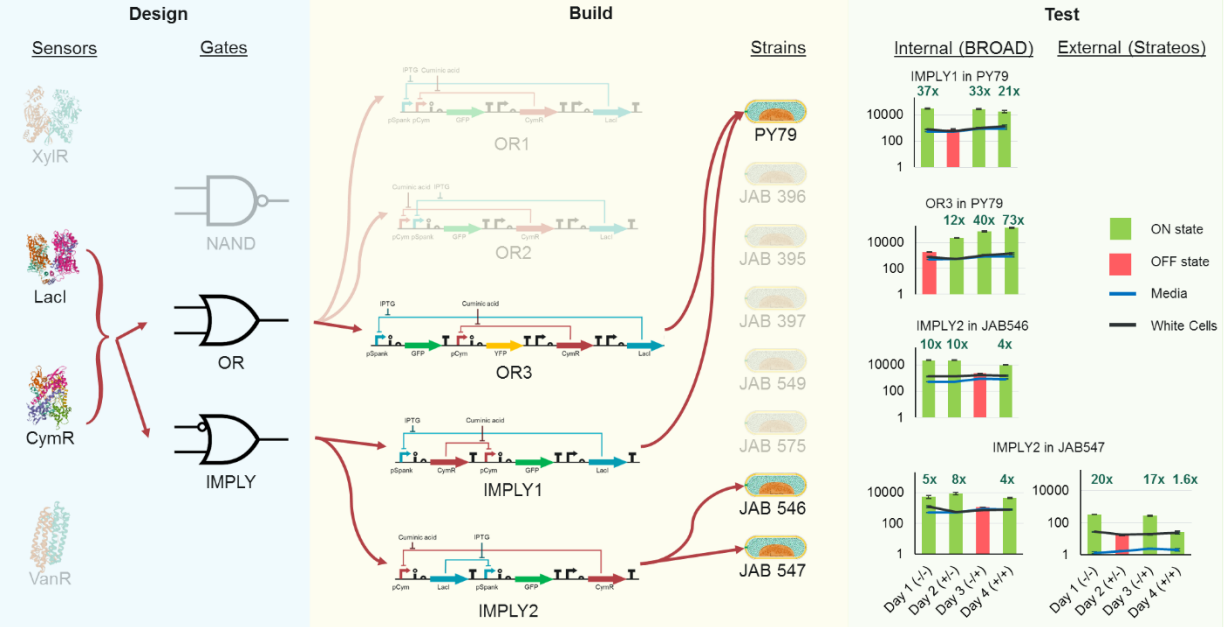


Figure 58. Creating 4 functional constructs in new hosts. Pressure test accomplishments after 4 months

1.4.15 Onboarding *E. coli* Nissle for genomic circuit design

So far, most of our studies were done using a common strain of *E. coli* (MG1655). In SD2, we were interested in expanding the host chassis to non-model organisms. We therefore attempted to onboard *E. coli* Nissle, as an important therapeutical bacterial strain. To do that, 9 transcriptional NOT gates were built in the genome of *E. coli* Nissle (Figure 59). To build these gates, three landing pads were first generated in *E. coli* Nissle genome (one for the sensor array, one for the circuit, and one for the actuator). Sensor array is composed of 7 small-molecule-responsive proteins transcribed from two separate prompters. For all NOT gates, *LacI* sensor, which responds to IPTG, regulates the transcriptional activity of pTac promoter. pTac promoter generates a DNA-binding protein (repressor) which binds to its cognate promoter in the actuator landing pad and represses the transcription of output YFP fluorescent reporter protein. The input-output response function of each NOT gate was characterized in Voigt lab using flow cytometry, as shown in Figure 60. The input to each gate is the activity of pTac promoter (converted to RPU units), and the output of the gate is the activity of the gate's output promoter (converted to RPU units). These data will be used to generate the user constraint file (UCF) that will be used in Cello to build larger circuits in *E. coli* Nissle.

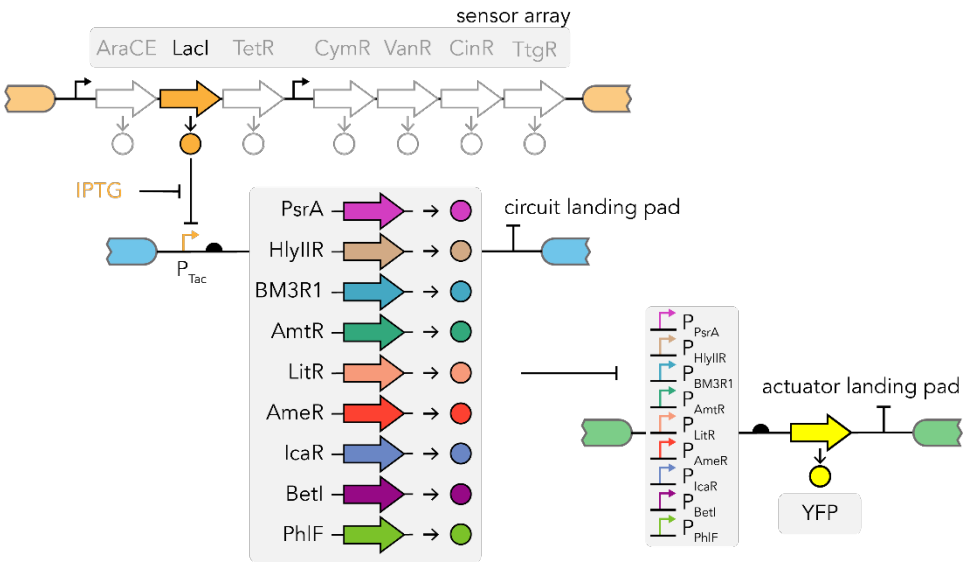


Figure 59. 9 separate genome-integrated NOT gates in *E. coli* Nissle.

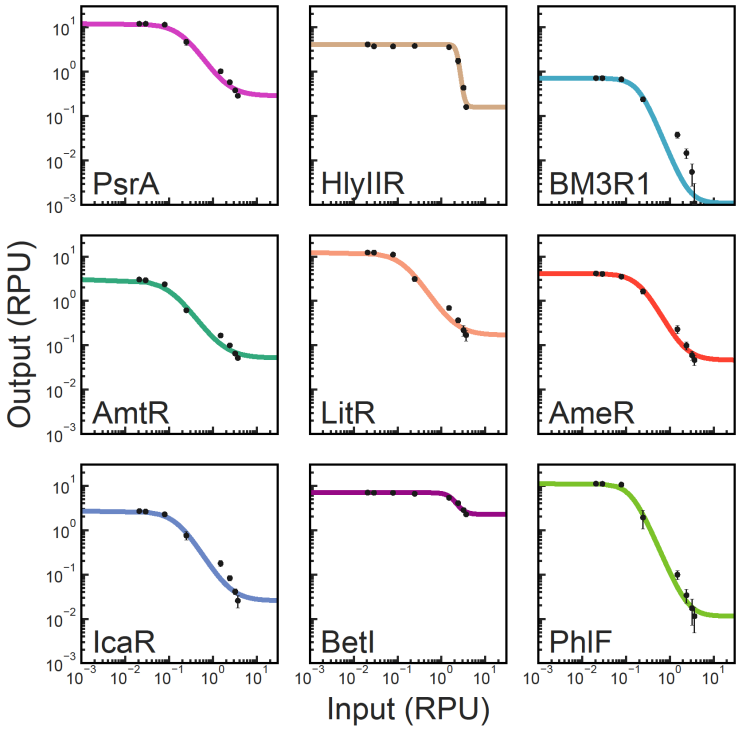


Figure 60. Response function of genomic-integrated NOT gates in *E. coli* Nissle. Lines are best fits to a Hill function.

1.4.16 Testing circuits built *E. coli Nissle*

RNA sequencing data was conducted on *E. coli* Nissle at TA3 lab (Ginkgo), we submitted 12 strains including the wild type, a sensor array, and 10 NOT gates. Among these gates we expected 1 failure (QacR) from internal flow cytometry data. RNAseq provides an ideal platform to study the internal function of these genetic circuits and the impact on the host. We had also previously observed one gene (HlyIIR) to be slightly toxic. The RNAseq data revealed HlyIIR expression results in significant variation in the transcription of endogenous genes, especially those with higher natural expression in the wild type. Further study needs to be conducted to understand the pathways and mechanisms of this impact.

Figure 61 demonstrates two RNAseq profiles from the gates (a) PhlF and (b) QacR, respectively. It is evident that the YFP expression is not impacted by QacR despite appearing downstream of the repressible promoter, confirming the failure of this gate.

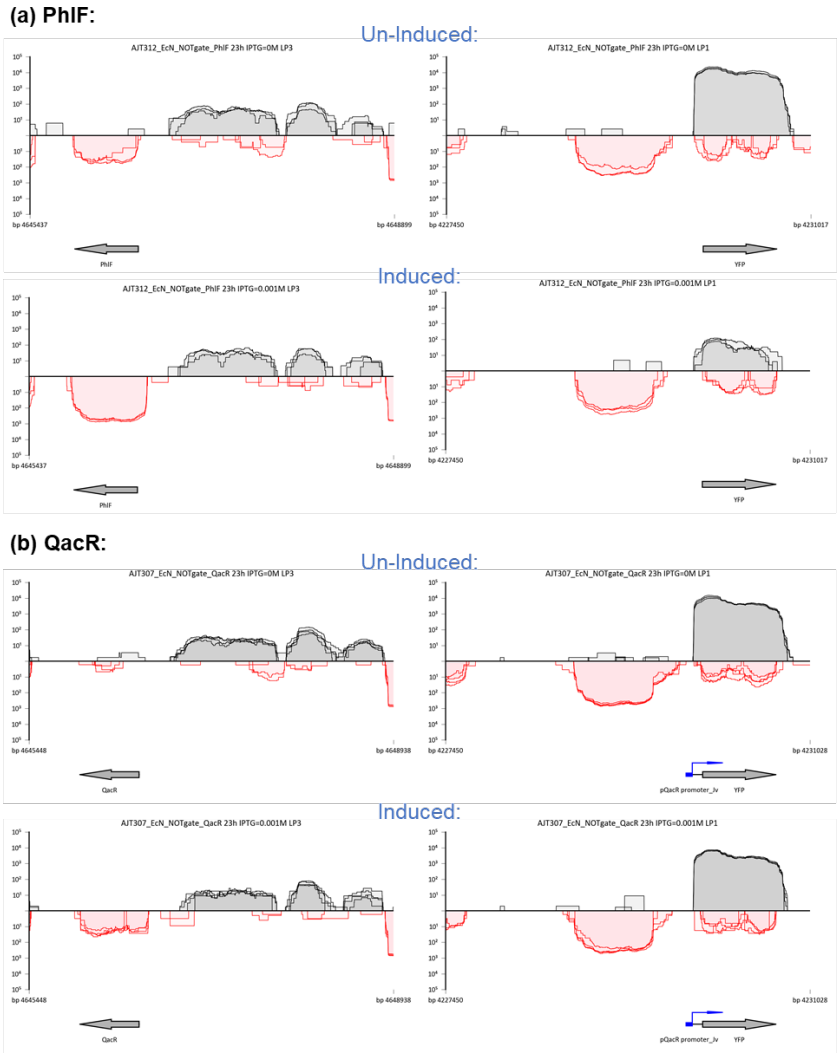


Figure 61. RNA sequencing profiles of engineered *E. coli Nissle*. Left panels show the gate region where the repressor is expressed under induction, right panels show output region where YFP is expressed by a repressible promoter. (A) PhlF and (b) QacR gate.

1.5 Conclusion

This work serves to improve the reproducibility and accuracy of genetic circuit design. The tools developed here fit within a pipeline of modular computational components to create an automated design-build-test-analyze cycle within SD2. Our use of transcriptomics provides a strain-agnostic, reproducible, and affordable approach to discover and parametrize genetic parts, evaluate part performance, discover errors, and model genetic circuit function. The data-driven models allow us to predict the dynamic and stochastic behavior of circuit designs and to derive design heuristics. Furthermore, the genome-wide nature of the data enabled mapping of circuit components to changes in host genome expression. By using this data to study the additivity of circuit impact, identify exclusive genomic targets, and build simulator tools; we have advanced analysis tools with which we can understand the broader impact of a circuit design and predict the comprehensive health and functionality of a strain containing a particular circuit. Lastly, this work has yielded experimental approaches to constructing a new generation of circuits, onboarding novel organisms of industrial relevance as circuit hosts, and rapid, standardized, and reproducible testing of circuits within the SD2 framework.

2 SUBGROUP 2: GORDON LAB (THE MIT-BROAD FOUNDRY)

2.1 Summary

The main focus of the foundry was centered around identifying and implementing methodology to facilitate the automation of data analysis, to increase the understanding of novel chasses, and to be able to perform predictive modeling. To achieve this, we worked with multiple groups and stakeholders across SD2 to both provide automated processing and novel insights into synthetic biology, RNASeq, and perovskites.

2.2 Introduction

We had four main projects in SD2 that can be broken up into smaller tasks. The first is the development of RNASeq analysis. For this, we helped develop pipelines that utilized TACC resources to integrate metadata into a full computational RNASeq pipeline of read mapping, counts, differential expression and analysis. Omics tools has been used to analyze multiple datasets from E. coli, E. coli Nissle, and Bacillus to better understand how inducers, circuits, and hosts impact RNASeq. An extension of the RNASeq pipeline is an analysis of the reproducibility of RNASeq data across different organisms. Such an analysis is important to understand not only how well your experimental and computational pipelines perform, but also how variable they are both for a given organism and between different organisms. We found that our pipelines generally worked well for E. coli, but had more difficulty with organisms such as Bacillus (where it was hypothesized that not using single culture resulted in larger differences in initial growth states).

In addition to RNASeq, we also work on additional problems of drug effectiveness and mixed media growth. Both of these problems used high throughput fluorescence to identify the effects that drugs and media had on growth. The goal of the project was to identify drugs that worked well for selecting of synthetic organisms (which is very organism specific), and how much different medias impacted log-phased aerobic growth and carrying capacity. Much of the tool creation was limited to facilitating methods development on the wet-lab side and analyzing results. While it was potentially beneficial for creating automated conclusions, the scale of the project meant that way more time would be spent creating the process than any timed gained analyzing results.

Lastly, we worked with the SD2 perovskite team to help predict crystal formation across a variety of amines. This started out from model selection, to development of a convex hull for limiting erroneous predictions, to taking part in a challenge to determine which model performed the best at predicting crystal formation across a bunch of amines. The project started with an initial training set, and active learning phase, followed by a prediction phase. Ultimately, our model performed well utilizing only 3 parameters that were variable across conditions.

2.3 Methods, Assumptions, and Procedures

The RNASeq utilized a set of standard tools in a unique pipeline that facilitated RNASeq analysis. For our portion of the tool, we utilized edgeR with TMM normalization in a way that helped facilitate user desired cross sample comparisons. We did this by creating a python package that takes in counts data and a json containing the types of desired comparisons. While this tool is part of a larger pipeline, it is agnostic to the type of mapping performed and only cares about the data format. We made no assumptions about the data, instead using quality control thresholds for

data inclusion. We also performed a reproducibility study in E. coli and Bacillus to validate how well our wet-lab side performed.

The Minimum inhibitory and mixed growth project was a simple project utilizing python as a means for processing and analyzing data. One of the major challenges to creating a universal tool for this process was both the lack of standardization of data formats across labs and time to create a standard (this was near the end of the program), which meant that essentially each lab performing the same experiment would result in a different data format, limiting the scale and useability of the tools. As such, this was most an exercise in helping collaborators perform initial drug and media screens.

The perovskite challenge utilized a gaussian process bayesian optimization model that took in a small subset of the metadata to perform a regression analysis. Our assumptions were that the metadata that was variable would be the most useful to prediction and that the degree of crystallization was relevant. By in large, our assumptions help true as depending on the metric, our model either was the top model or 2nd model.

2.4 Results and Discussions

2.4.1 Automated RNASeq Analysis

We performed a survey of the current state of data and metadata standardization and compliance within the SD2 program. This was done in partnership with the other team (BBN/Nick Roehner). We identified several severe gaps in the way metadata is propagated across groups, creating a situation wherein too little information survives the trip from the design group all the way to SynBioHub. As such, it is currently not yet possible to ask even simple questions like "How many RNA-seq experiments have been performed by SD2?" or "What yeast RNA-seq experiments have been performed in galactose?" or "What organisms have we used?" To address this situation, we have been actively participating in the Data Representation working group to ensure that SD2 processes record critical metadata information, including: media composition, growth conditions, characterized parts (and their documentation with SBOL image), links to relevant datasets in the repository, links to related experiments, reference sequences and results (Table 2). So far in Q3, the working group has added the metadata information for previously analyzed experiments which was hitherto missing such as conditions, strain and sequence information according to each challenge problems. Integration of XPlan with SynBioHub was facilitated by TA4. Due to our involvement, XPlan now captures additional metadata information and updates to previously analyzed challenge problem datasets.

While awaiting the transition to a functional metadata repository, we have initiated development of APIs and client to query and parse experimental information. A test GUI is implemented (by BNL) on TACC to facilitate testing of SynBioHub queries. GUI: https://hub-api.sd2e.org/sparql. This enables SynBioHub to be central database to capturing all metadata information, relevant planning details and results location.

Table 2. Metadata Information, captured in SynBioHub. Each of the target represents SynBioHub classes that provides information about the experiment and sample level information.

Goal: Data captured from both planning stages and post analysis results			
Target 1.A	Study ID	1.1	Autogenerate OR assigned study ID
	Challenge Problem	1.2	(Any) Novel Chassis, Riboswitches, Yeaststates, Immortality or Protein Stability
	Principle Investigator(s)	1.3	Principle investigators and groups involved
	Study Design with documentation	1.4	Circuit design with documentation or Treatment design
	Experiment Details	1.5	Additional experimental details
	Study Objectives	1.6	Goal and sub tasks involved for the study
	Study Outcome	1.7	Summary or results snapshot
	Protocol Details	1.8	Culture methods or library preparation and sequencing information for NGS
	Raw Data Location	1.9	Parent directory of all raw data
	Results Data Location	1.10	Parent directory of all results data
	Genome: Version and File Path	1.11	Location of reference sequences
	Annotation: Version and File Path	1.12	Location of annotation files
	Additional Description	1.13	Any other additional details
	Study Reports Location	1.14	Location of all reports and/or presentation details
	References	1.15	References based on which the study was designed
Goal: Elaborate details on Experiments for the study			
Target 1.B	Sequencing Technolgoy	1.16	Protocols and Results Path (depending on short or long read sequencing)
	Metabolomics	1.17	Protocols and Results Path
	Proteomics	1.18	Protocols and Results Path
	Metabolomics	1.19	Protocols and Results Path
	Flow Cytometry	1.20	Protocols and Results Path
Goal: Sample Level Information for each Target 1.B			
Target 2.A	Sample ID	2.1	Corresponds to raw data file or user defined name
	Sample File and Location	2.2	Sample location with file name
	Sample Group	2.3	Parts Introduced from Circuit Design or Sample Group
	Collection Date	2.4	Sample data collection date
	Replicate Number	2.5	Sample replicates or no replicates
	Organism	2.6	Test or model organism (Scientific Name)
	Strain	2.7	Strain information
	Genome File Location	2.8	Genome file and version
	Annotation File Location	2.9	Annotation file and version
	Adapter Sequence	2.10	Sequencing adapter sequence
	Media/Culture	2.11	Media or culture information. Also captures bacterial abundance values.
	Growth Conditions	2.12	Additional growth conditions
	Time Point	2.13	Growth time that may indicate growth phase
	Inducer	2.14	Inducers added to individual sample
	pH	2.15	pH value in growth conditions
	OD	2.16	
	QC metrics Wet Lab	2.17	Example include bioanalyzer or nano drop results for NGS sequences
	QC metrics Computational	2.18	Example includes FastQC or RNAseq QC metrics and contamination estimation
	Nutrient Concentration	2.19	Raw spectrometric or fluorometric readings converted to concentrations

To establish proof-of-concept results, and to provide a platform to explore how to best automated specific aspects of RNA-seq analysis, we allocated a considerable portion of Q3 activities toward manual stewardship and analysis of SD2 RNA-seq data.

The first major accomplishment was that we (manually) assembled a meta-data database of all completed and planned RNA-seq experiments for the program as of Q3. This was largely a communications activity (Slack/email/phone calls), as none of the information had yet been centralized, and (as we learned) was not even well documented by originating/proposing labs. Our resulting database is the only existing comprehensive overview of SD2 RNA-seq data, and we have started using it as the basis for our proof-of-concept analyses. As of 7/1, there are six experiments completed, three planned/proposed, spanning four different species and three different circuit designs.

Second, we have established several components of the RNA-seq analysis pipeline. For example, there is now a Docker container at TACC that automatically performs mapping and

generates read count matrix per experiment for suitable for downstream differential expression analysis using DeSEQ analysis and Gene Set Enrichment analysis. Moreover, we have applied these tools to perform cross-experiment and within-experiment RNA-seq analysis of differentially expressed genes, KEGG pathways, and GO categories. For example, in comparing the YeastGates 1.0 data to a dataset from GEO grown under the same conditions (GSE88952), we have provisional findings that cell introduction of sgRNA-based gates has side effects of upregulating wall biogenesis and cell cycle processes (Figure 62). In comparing the YeastGates 1.0 datasets to each other, we found that there are some subtle side-effects between the "01" and "10" states that may be of interest. In the coming quarter, we will extend these analyses to other RNA-seq datasets, focusing initially on novel chassis (*E. coli.*).

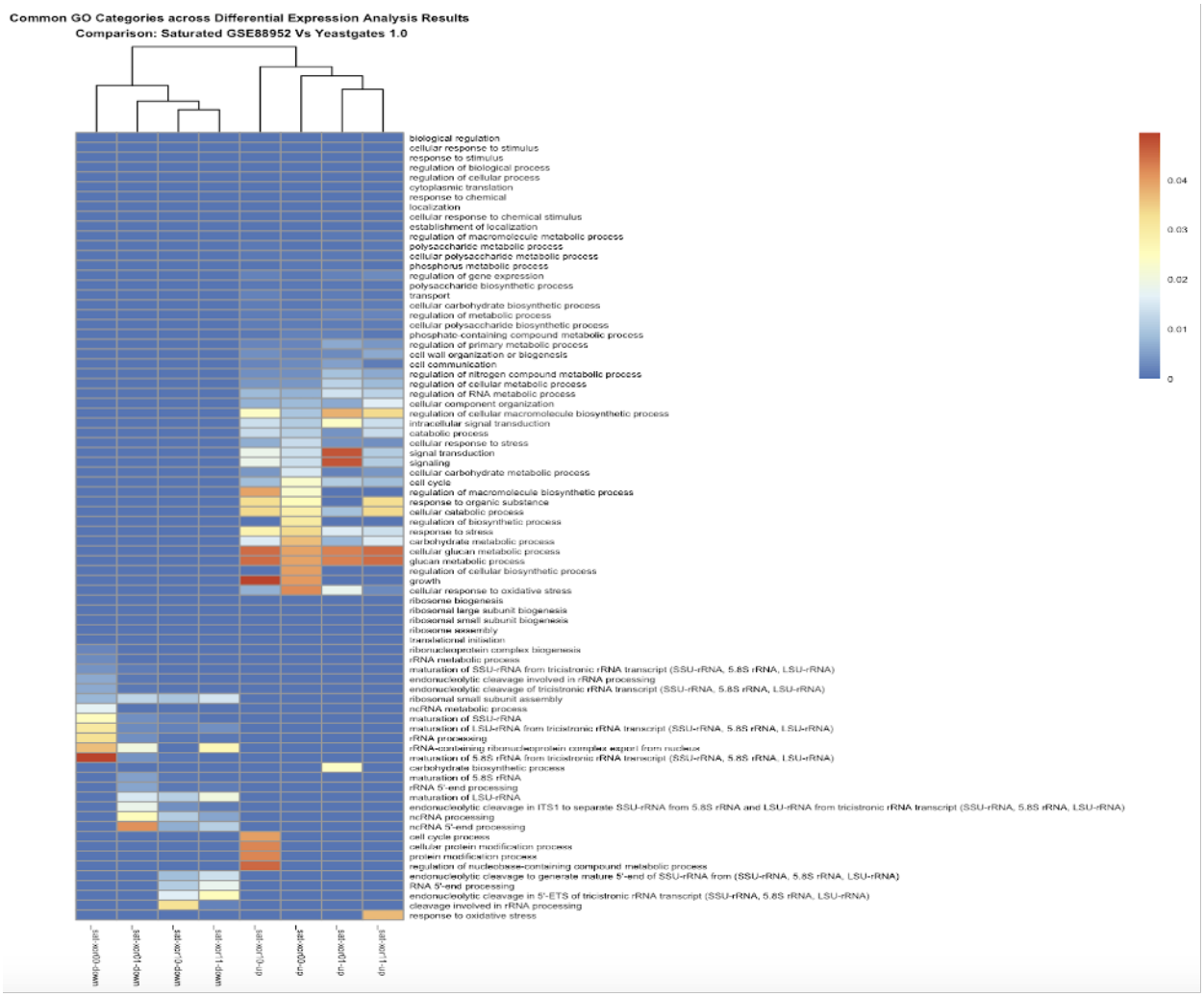


Figure 62. GSE88952 Vs Yeastgates 1.0: Common GO Categories for differentially expressed genes from various comparisons: The gene set enrichment analysis was performed on upregulated and downregulated genes from differential expression analysis.

The GO categories were corrected by Bonferroni-Hochberg (BH) (p adjusted values <0.05) multiple testing correction. GSE88952 samples grown in saturated nutrient media were compared with the four Yeaststates 1.0 circuit groups. Gene set enrichment analysis results were compared across all the above comparisons to identify any functional patterns that were significantly enriched. For example cell cycle and signaling process is consistently down-regulated in circuit gates when compared against samples grown in saturated media. Going forward we intend to make more robust analysis based on SD2 internal datasets based on consistent query rules rather than relying on open sourced data.

Using both RNAseq we compared quantified data across the various data across different labs i.e. Gingko Bioworks, BioFab and Transcriptic. Due to sequencing quality issues, Transcriptic dataset was not further analyzed. Within Gingko Bioworks, two library preparation kits data set was compared i.e. normal library preparation and minimal (mini) library preparation (Figure 63). We find the within Gingko Bioworks library preparation, we find that normal library kit to yield data that had adequate quality and must be used for future work. All comparative analysis was performed mostly using read count data (Figure 64). From the minimal library preparation, we performed comparison of data across different temperatures and different time points (Figure 65 and Figure 66).

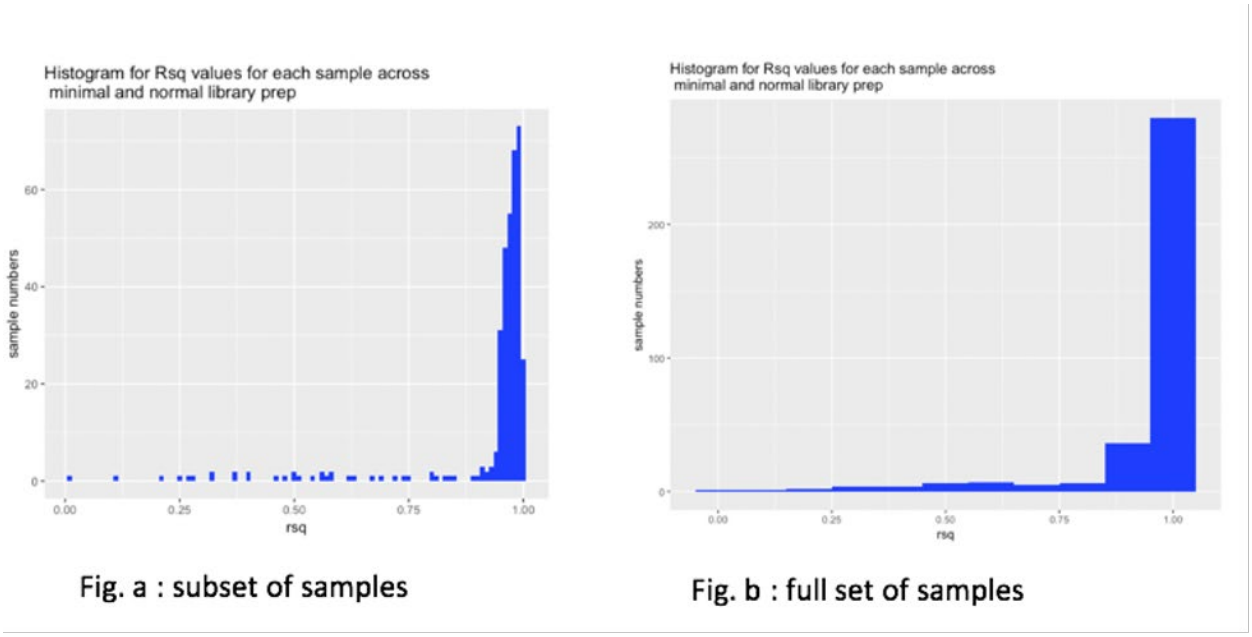


Figure 63. Comparative analysis of mini and normal preparation kit. Here R^2 values was calculated from read count data set for individual samples.

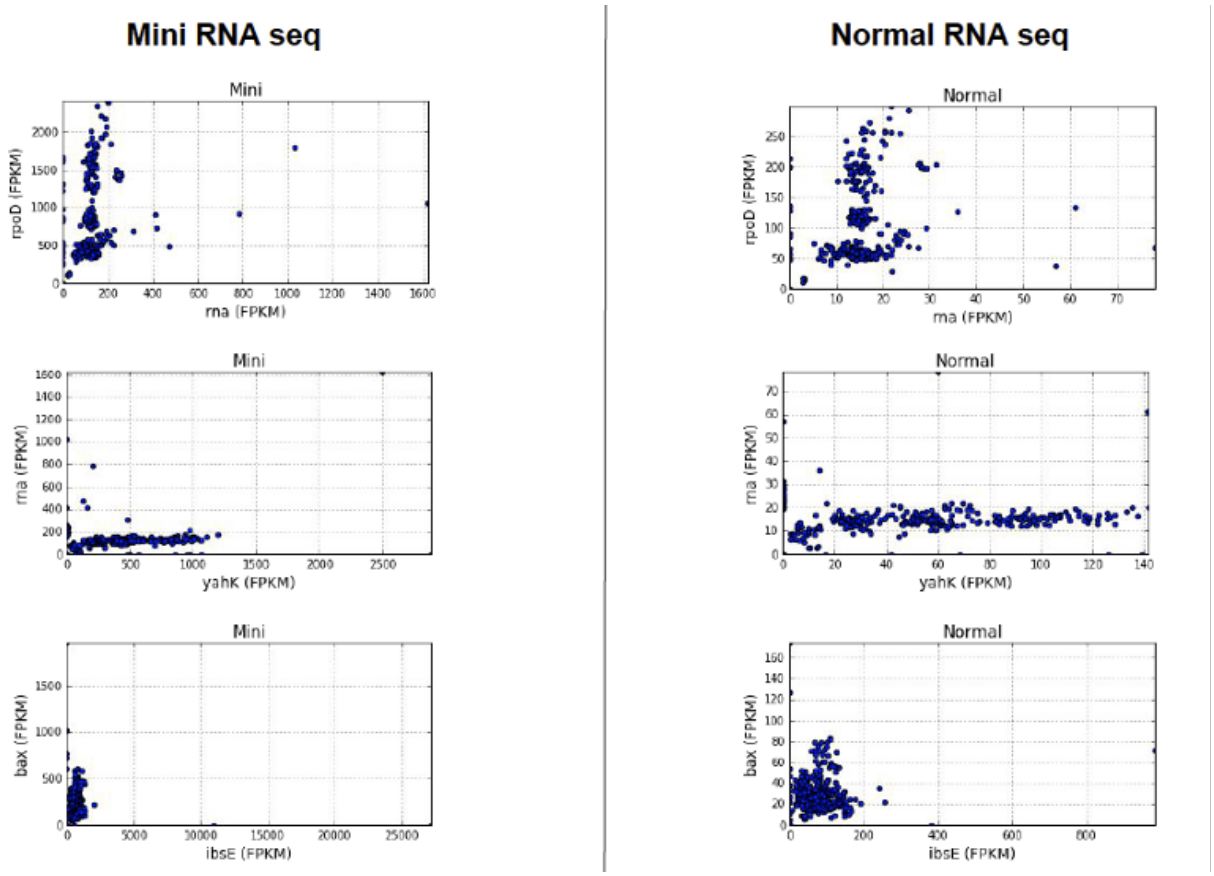


Figure 64. Genes comparison of mini and normal preparation kit using FPKM values

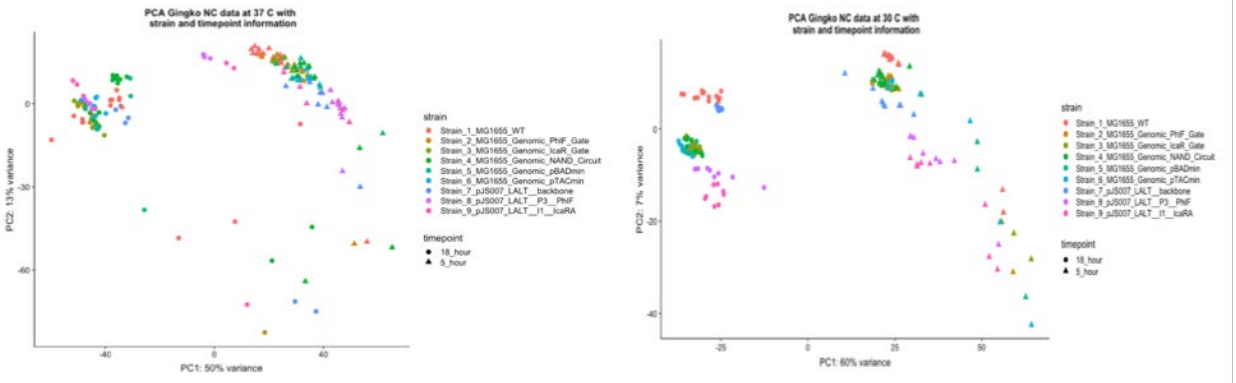


Figure 65. Comparing Gingko generated data across 37°C and 30°C

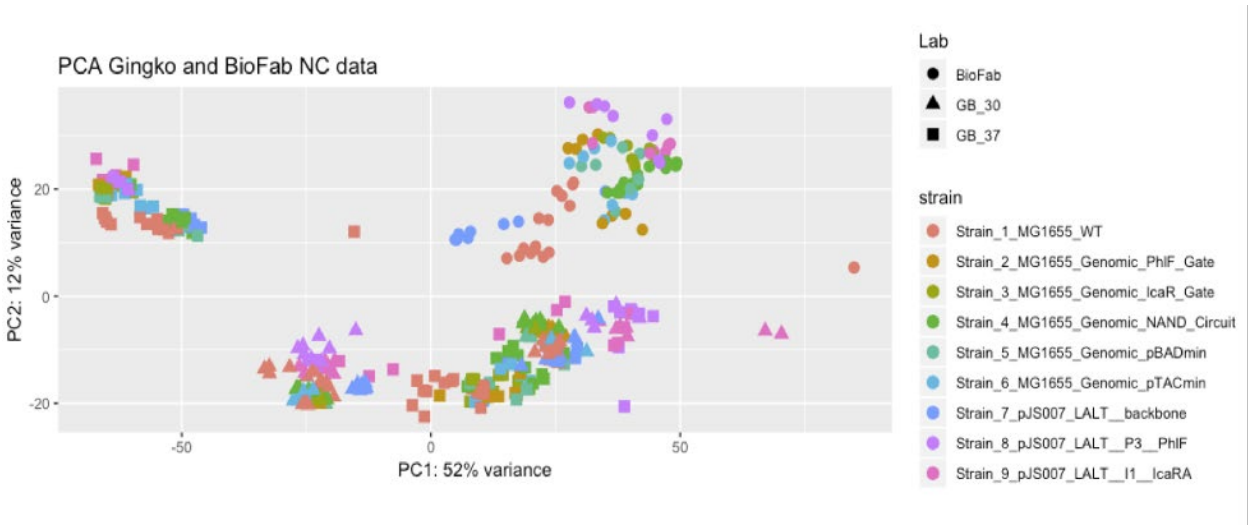


Figure 66. Comparing Gingko and BioFab generated data across 37ºC and 30ºC

The results of the comparison show the impact of temperature on host genes are largely dependent on strain and circuits. The impact of inducer is largely circuit dependent. And finally, impact growth time is slightly circuit dependent.

An integral component of RNASeq analysis requires constructing all of the different meaningful comparisons between experimental conditions. We created a python package that accomplishes this task. The package has four key functions: it can be used to construct all meaningful comparisons with a specific number of degrees of freedom (typically one), perform multi-core differential expression analysis with appropriate QC and cross-library normalization, annotate the differential test results with GO and KEGG biological terms, and offer a user-friendly question & answer interface to a visualization of the differential expression results.

We integrated the NAND v1 raw count dataframes from Ginkgo, Transcriptic, and UW Biofab. The QC filter was to remove samples with less than 1 million mapped reads. This filter ended up removing 42 of 352 samples from Ginkgo's dataframe, 3 of 176 samples from Transcriptic's dataframe, and 78 of 88 samples from UW Biofab's dataframe. The 1 million mapped reads threshold was taken as a recommendation from Ginkgo and UW Biofab. The final dataframe included 492 samples. (Figure 67)

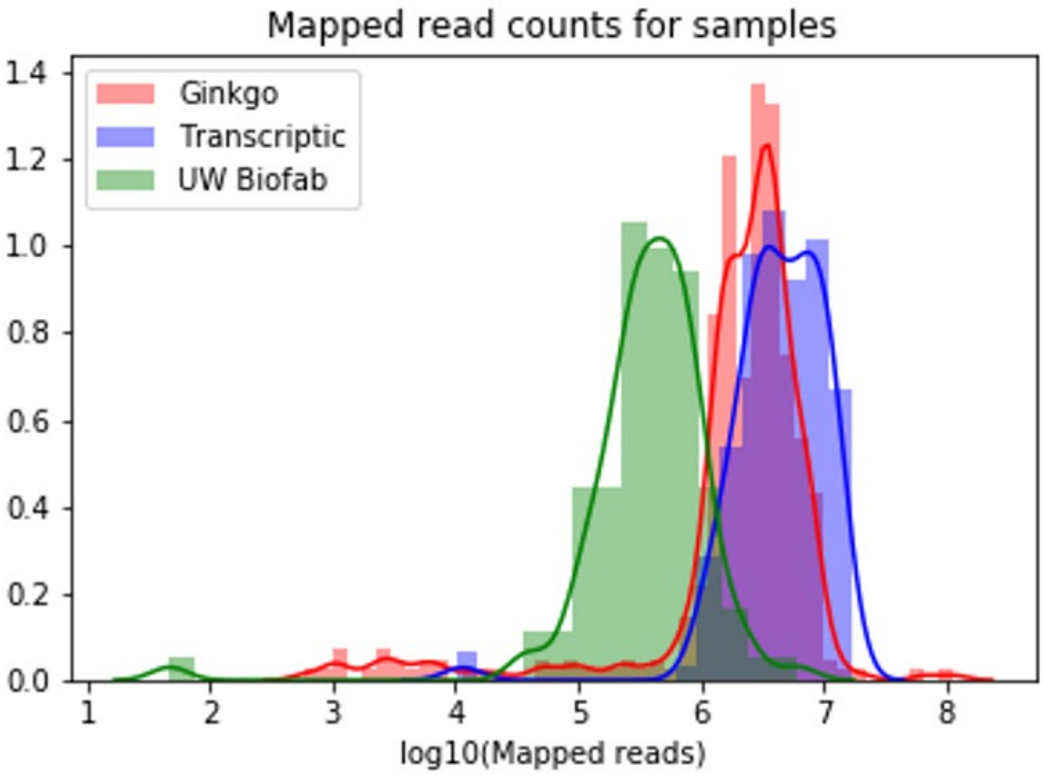


Figure 67. The number of mapped reads for each lab is log10 scaled and shows many samples pass the QC threshold of 1 million mapped reads. Ginkgo and Transcriptic had a majority of samples above the threshold while UW Biofab only had 10 samples remaining for downstream testing.

The first application of the tool is used here in the context of NAND v1 RNA Seq data but the logic can be extrapolated to other experiment types with many conditions requiring differential testing. The full set of meaningful differential test factors is constructed from a chosen list of strain pairs, the IPTG and Arabinose induction conditions, timepoints, temperatures, and laboratory of origin. The tool was used to generate 200 differential comparison tests when grouping replicates irrespective of laboratory of origin and then also 266 differential comparison tests when including laboratory of origin as a factor for differential testing.

The second application allows two modes of use, either in a Jupyter Notebook instance or batch job submission with a compute cluster. We designed a simple python interface for running rigorous statistical tests of differential gene expression in EdgeR without needing to know R or edgeR. Running the tests in Jupyter session itself took about six hours for 266 differential comparison tests on an eight thread machine. The Jupyter option to run the differential tests scales linearly with the number of available cores. If the user has access to an HPC they can take advantage of the tool's capability of generating Rscript files for the differential tests.

Finding higher-order biological significance from gene IDs is a known way of getting actionable insights from expression data. I included GO and KEGG as simple calls to a function in the tool, requiring only the user looks up the appropriate organism identifier. All the differential

test results were annotated with GO and KEGG then applied multiple comparison statistical test correction by using the Benjamini-Hochberg procedure for false discovery rate.

To visualize and quickly browse the dataset while being able to ask biological questions is of key importance. The dataframe containing all the annotated differential test results can be difficult to parse by eye (Figure 68). However, it can be easily subset via a method in the python tool that allows for building up questions by pivoting or constraining certain categories. For example, if we wanted to know if there were any differences in the data obtained from different laboratories we could pivot the display to only show differential tests where laboratories are the only differential factor, holding all others constant (Figure 69). The visualization can also be viewed by uploading the exported dataframe to a website http://amp.pharm.mssm.edu/clustergrammer/ however there is no option here to subset the dataframe via python.

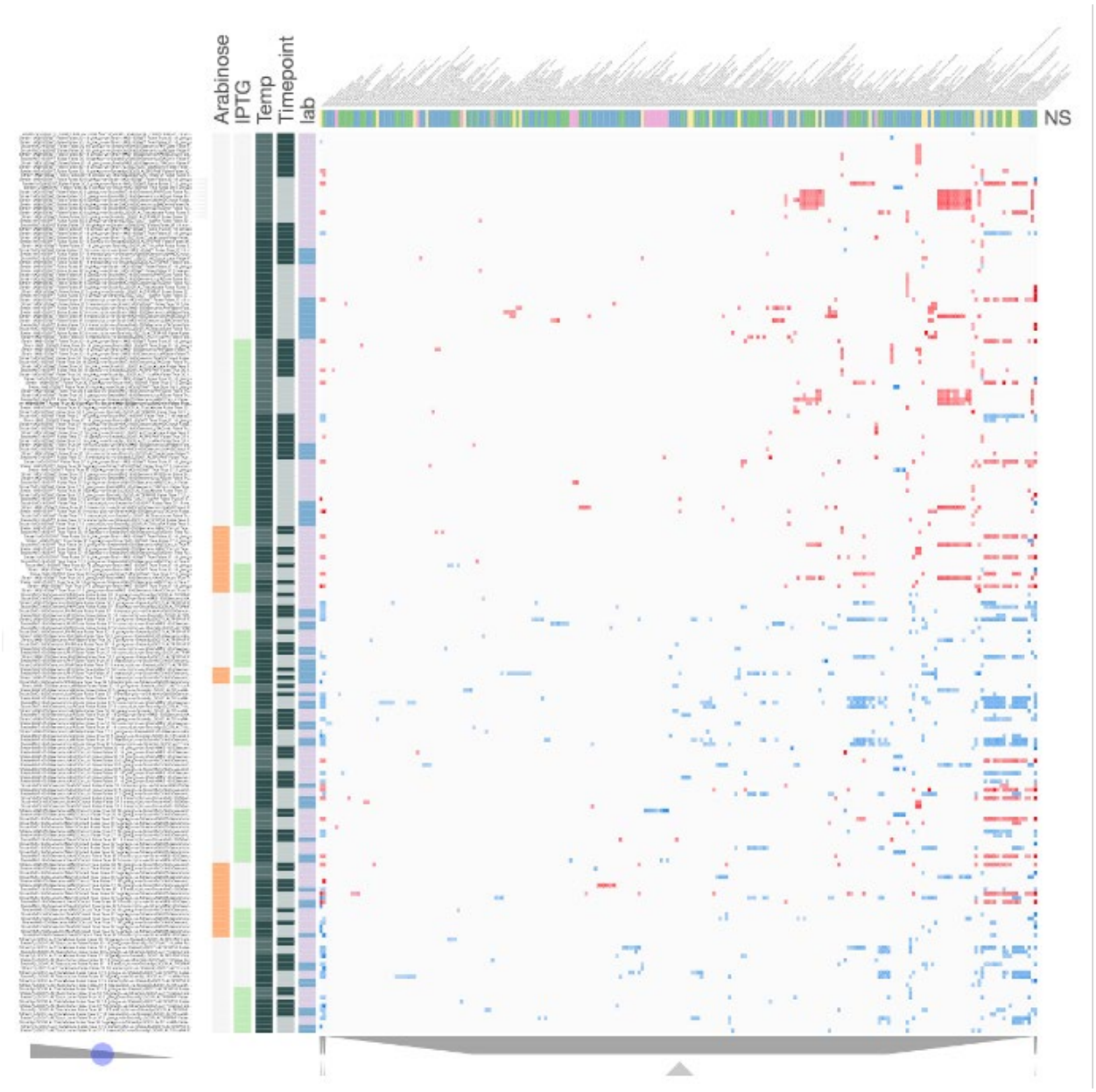


Figure 68. A matrix visualization of all annotated up-regulated and down-regulated GO and KEGG terms for differential tests using factors Arabinose, IPTG, Temp, Timepoint, and lab. The GO and KEGG terms are on the top as columns and the differential tests are displayed as rows. The column and row information is displayed when hovering over a specific cell in the matrix for quick look-up. Categories on the rows and columns can be re-sorted by double-clicking the corresponding category term. The visualization can be cropped to zoom in on an interesting region. A slider (below row names) offers the ability to change which depth of the clustering dendrogram is displayed beneath the x-axis, used to quickly crop the matrix to a cluster.

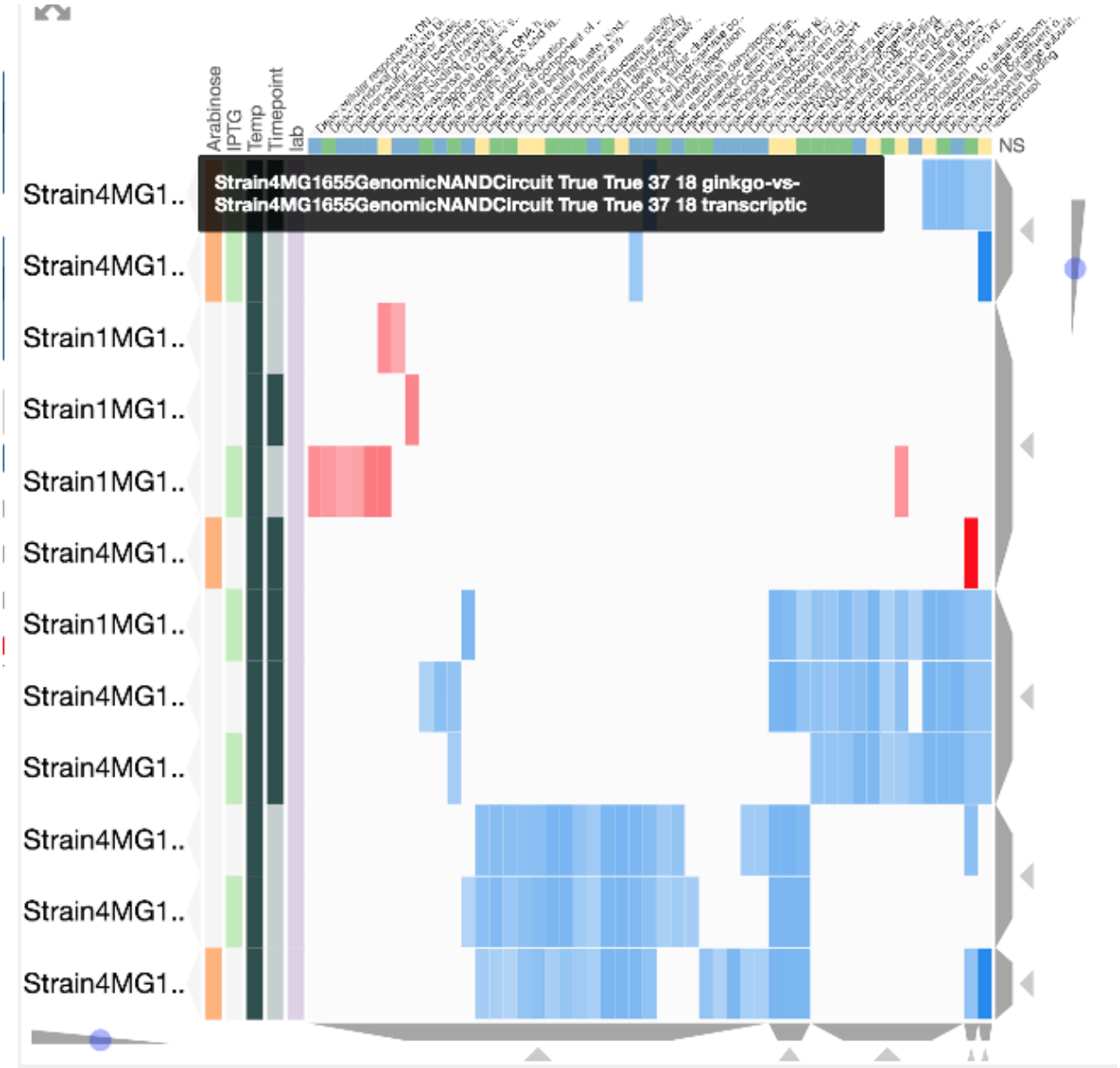


Figure 69 Fig 2.4.1.8 - Visualization for the differential tests relevant to a specific experimental question: are there any differential genes that arise due to samples with identical conditions but prepared in different labs? After using the subset query feature of the python tool the dataframe is reduced to display the samples of the particular question of interest. In this case the answer is yes, there are substantial differences between Ginkgo and Transcriptic data, limited to strain 1 and 4.

We additionally developed a pipeline for analyzing the impact that circuits have on the cell host via RNASeq. Figure 70 illustrates the pipeline with the end goal of integrating findings into Cello to improve design decisions. NAND 2.0 RNASeq data was preprocessed and run through Omics tools to identifying differentially expressed genes between modified strains and the wildtype control under the same growth and induction conditions. We then ran Gene Ontology

(GO) enrichments to identify functional changes caused by the integration of circuits into the cell, and how these functional changes were dependent on induction and growth conditions.

Iterative Random Forests (iRF) were then ran a GO enrichment matrix against a feature matrix of circuit components, growth conditions, and induction conditions to identify which features were important for functional changes within the cell. Random Intersection Trees (RIT) were then run on the iRF results to estimate which combinations of features are important for functional changes within the cell. The results of iRF and RIT can be utilized to inform strain design or be integrated into Cello's for selecting which circuits have the lowest impact on the cell host.

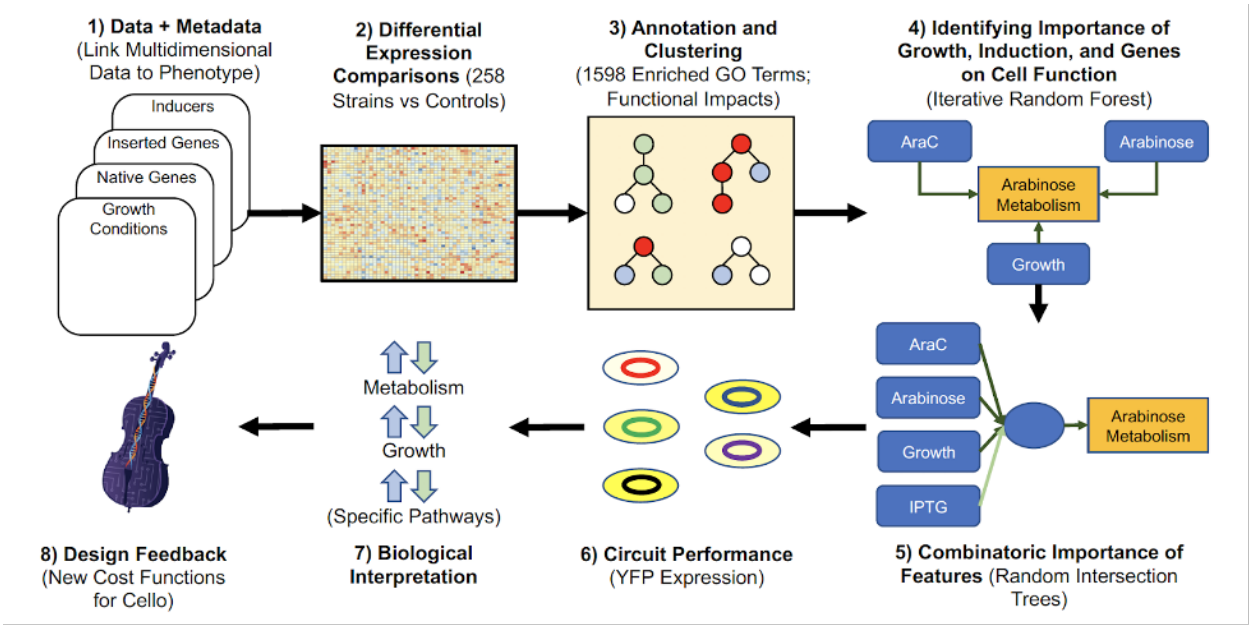


Figure 70. RNASeq pipeline for identifying functional impacts of circuits on cell hosts

We then expanded upon the RNA-seq analysis pipeline to identify circuit components with the highest impact on markers for host-stress (Figure 71). We can integrate the results into Cello, by selecting components with the lowest stress out of circuits with equivalent performances (potentially leading to less performance loss over time). Additionally, we can identify pathways of interest that can be modified to try and mitigate some of the stress imparted onto the host organism. Finally, we can use the tool for either novel scientific discovery or by providing evidence for interactions that may not be fully characterized. Interactions here offer a starting point for either future experimental design or for literature searches to better understand or hypothesize any given interaction or combination of interactions.

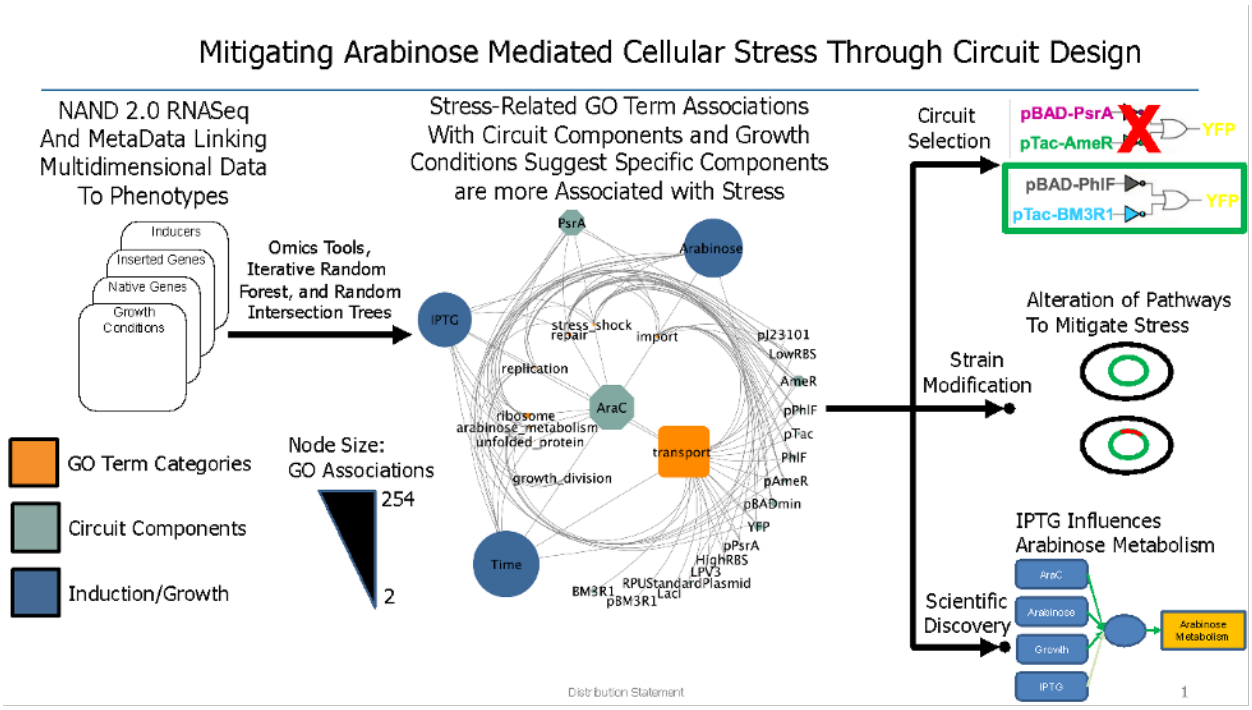


Figure 71. Results of RNA-seq analysis on NAND2.0 dataset. We start with an RNA-seq dataset that contains circuits and/or gates subjected to multiple different conditions (growth, inducers, combinations of gates/circuits, sub selection of gate components). From here we run the data through omics tools to get differential expression and gene ontology functional changes. We then run the resulting feature matrix (matrix of components/conditions and matrix of GO terms changes) through iterative random forest to get associations between features and GO terms and through Random Intersection Tress to identify combinations of features that affect GO terms. The result is relations and combinatorics of relationships between components/conditions and functional changes. From here we can identify which components have a greater impact on the host, which can be integrated into Cello for helping decide between equivalent circuits. We can also identify which components and pathways have large impacts on the host, which represents targets for modification to mitigate impact. Finally, we can identify how different components interact leading to identifying areas of novel discovery.

The RNASeq pipeline was also run on E. coli Nissle. Unfortunately, sample quality was an issue for many of the samples and controls, which only allowed for analysis of the 23 hour timepoints (equivalent of 5 hr). A differential expression analysis was run comparing each gate to the wildtype control and each gate to the sensor-only control under both non-induction and IPTG induction. Comparisons against the wildtype were highly correlated to each other (Figure 74); whereas, comparisons against the sensor were a lot more variable. This suggests that most of the effects of the sensor plus gate are fairly generic (except for gates such as HlyllR and AmeR). Unsurprisingly, induction condition had a greater impact on the sensor correlations than on the wildtype correlations.

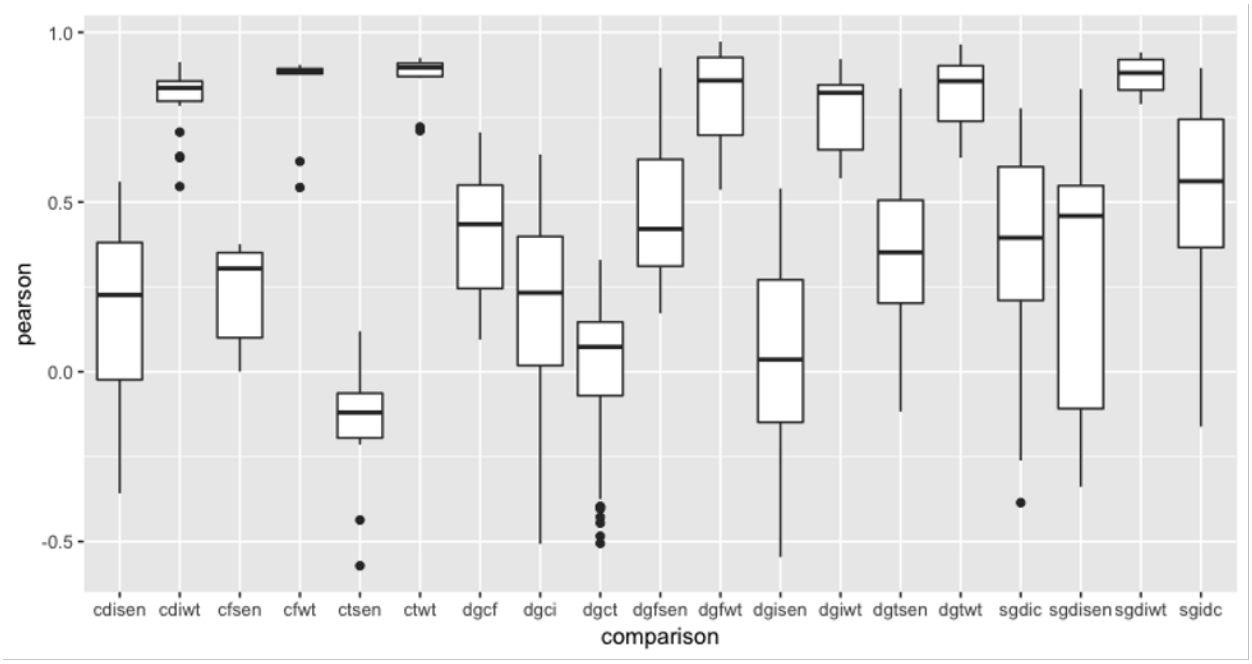


Figure 72. Pearson values for the different fold change comparisons. C – control, sen – sensor, wt -wild type, d – different, f – false (non-induction), t – true (induction), g – gate, i – induction (for different induction condition), s – same. Having a gate/sensor is the largest contributor to fold changes in the wildtype, with induction having a much smaller effect. However, in comparisons to the sensor, there are much bigger differences between gates and induction conditions.

An FDR of 0.05 was applied to the p-values to get differentially expressed genes. Most gates had a slightly higher number of differentially expressed genes versus the wildtype compared to the sensors versus the wildtype (Figure 73), while two gates, AmeR and HlyllR, had over 500 more differentially expressed genes. Similarly, these two gates were least like the sensors (Figure 74), with the number of differentially expressed genes versus the sensors being at similar levels to versus the wildtype. Most other gates had very little differential versus the sensors. One explanation for the high levels of differential expression in HlyllR is because a good transfection could not be achieved and thus it had a much lower growth rate compared to all other gates and the sensor. More explorations need to be done to identify why AmeR had similar differences and if there are any functional trends between the gates.

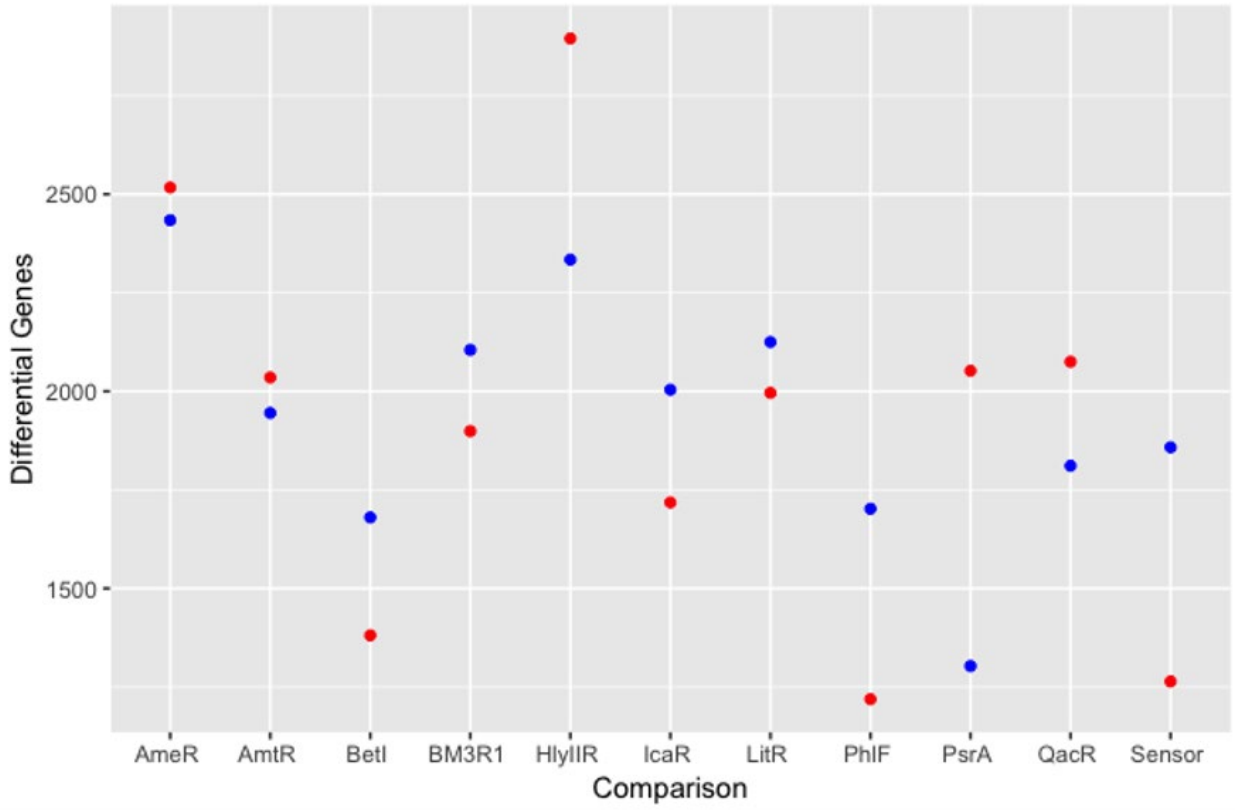


Figure 73. Differential expression of NOT gates to wildtype. Red - no induction, blue - induction. HlyllR had the highest differential expression followed by AmeR; the two gates least similar to the other gates. Most gates had slightly higher differential compared to the sensors alone.

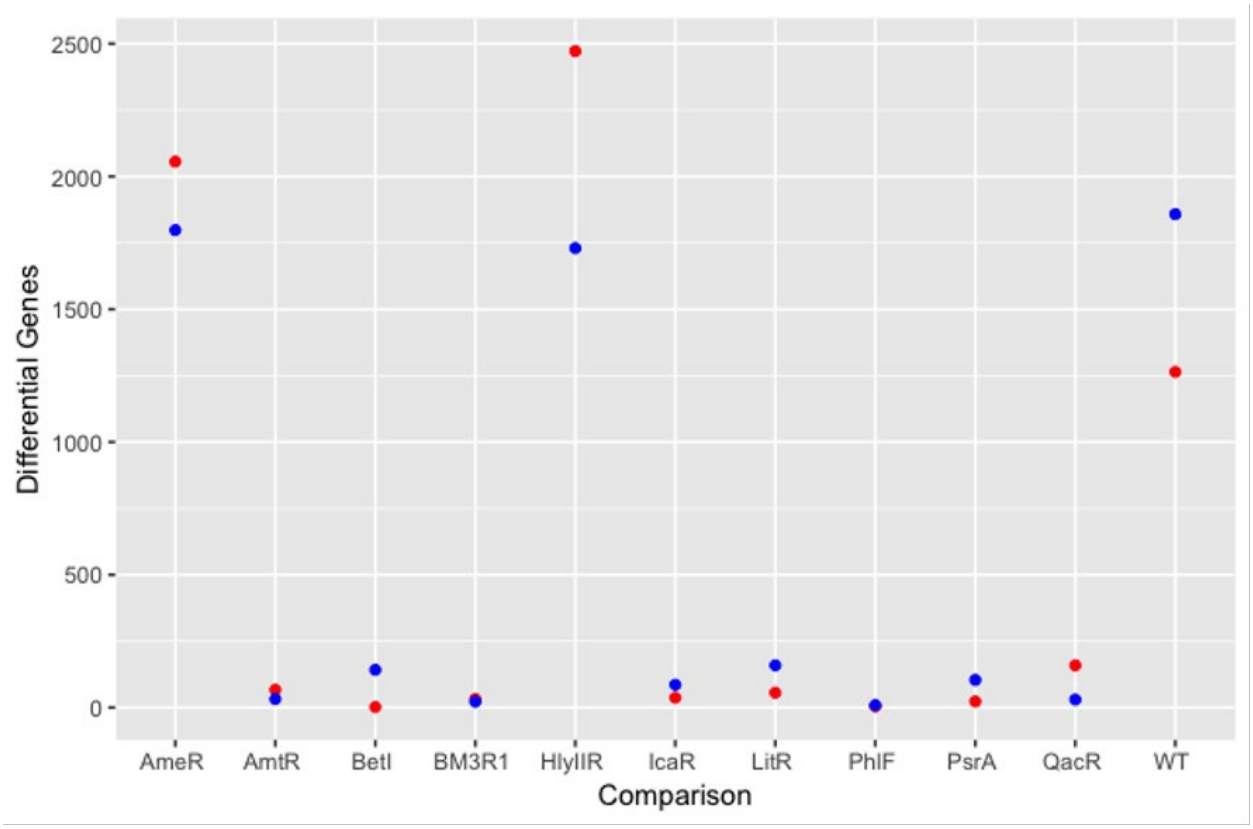


Figure 74. Differential expression of NOT gates to sensors. Red - no induction, blue - induction. HlyllR and AmeR are least like the sensors, with differential levels similar to that of the wildtype. Most of the other gates were very similar to the sensors.

2.4.2 Reproducibility in RNASeq data

Reproducibility is a necessary part of research, ensuring that identified phenotypes are the result of accurately measurable properties rather than unmeasured variability in the system. To help understand the reproducibility of our research pipelines (specifically at ginkgo), we analyzed control data from two sets of organisms (E. coli MG1655 and B. subtilis), run through the same experimental protocol. Additionally, to help facilitate the understanding of the data and to ensure technical reproducibility, all our RNASeq data was run through the computational sequencing pipeline and omics_tools.

First, we analyzed the MG1655 data at three different timepoints (5, 8, and 15 hours), and on two different days. We utilized the timepoints to assess common growth stages in E. coli (log phased, transition, and stationary). Differential expression analysis (FDR < 0.05 and log2FC > 2 with both TMM normalization and FPKMs) and correlations (FPKMs and Figure 75) were performed across all possible comparisons. Additionally, the two different days were chosen to identify how repeatable experiments are when run at different times. Unsurprisingly, the different day, same hour comparisons were the most highly similar, with an average number differential genes of 10.8 per comparison. There were minor differences between the significant genes from omics tools and FPKM (82.6% overlap for different hour comparisons), likely due to the

intersample normalization from TMM. The data suggest that the experimental pipeline is highly reproducible for E. coli.

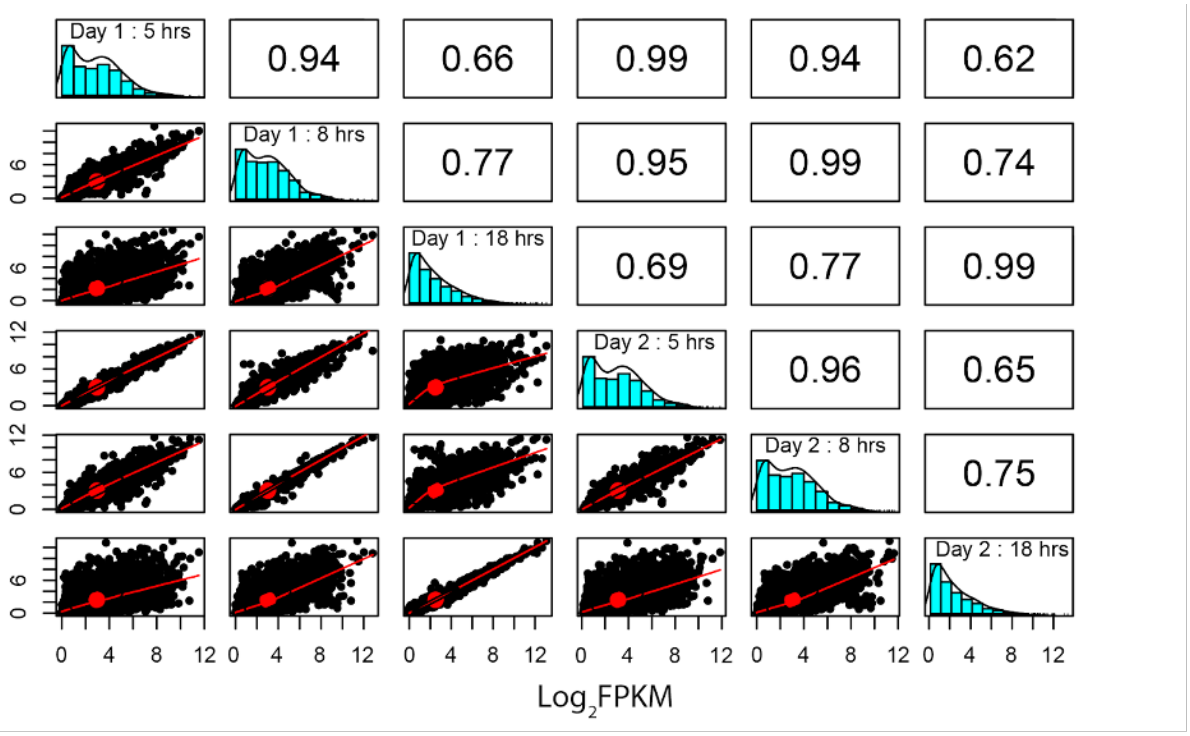


Figure 75. Pearson's correlations for E. coli (upper right numbers) between log2 FPKM samples at each of the measured timepoints on both days. Samples taken on different days, at the same hour, are much more similar (avg .99) than at different hours. The 5 hour and 18 hour have high correlations (avg 0.95), suggesting they are in similar growth states, while the 5 hour and 18 hour have the least similar expression profiles (avg. 0.64) due to their differences in growth states. Scatterplots are gene-gene Log_2 FPKM comparisons, with a red linear regression line. Blue plots are histograms for frequencies of gene FPKMs.

However, when we applied the same pipeline to B. subtilis control data (5 hour timepoint on 3 different days), we get slightly different results. For 2/3 of the 5 hour comparisons (Figure 76), correlations are lower than that of the 5-8 hour E. coli comparisons. The average percent of significant genes was also much higher for B. subtilis (4.87%) compared to E. coli at the same 5 hour comparison (0.27%). Both suggest that there are slight differences in growth among the samples. It is hypothesized that the B. Subtilis was more impacted by the overnight growth in wells as well as higher potential dilution effects on growth and sporulation compared to E. coli. We also show that conditions that are highly reproducible in one organism do not necessarily produce reproducible results in another organism.

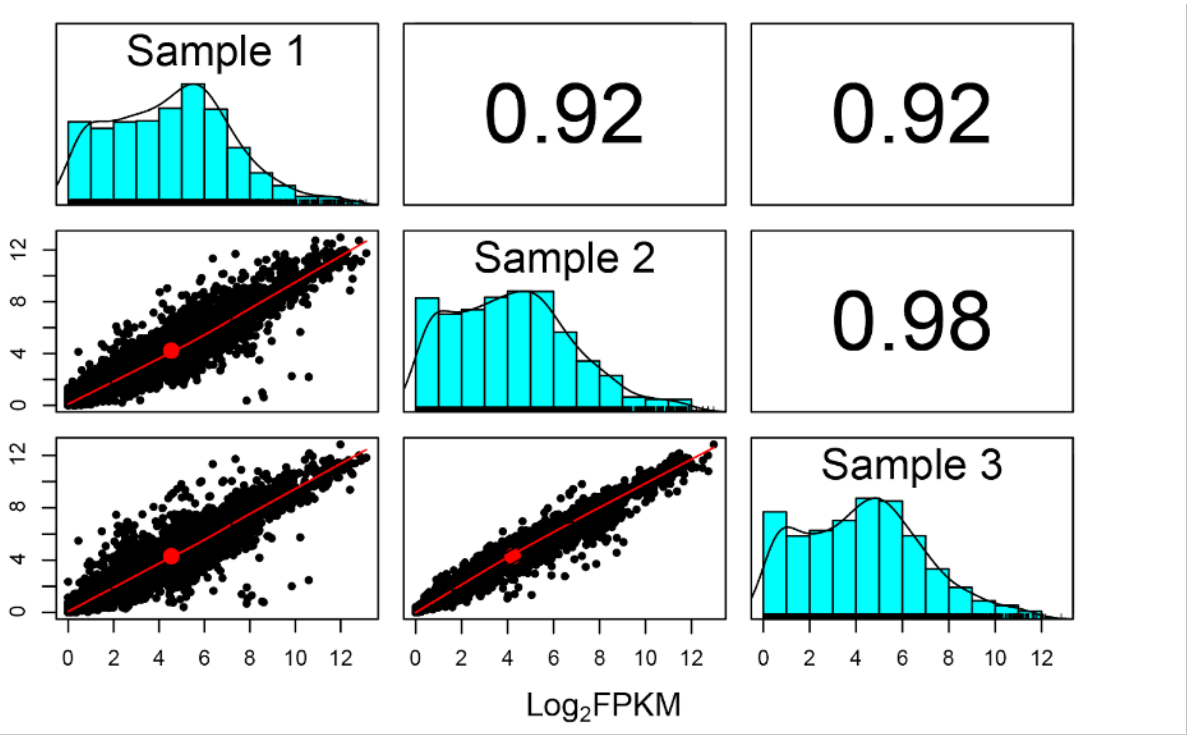


Figure 76 Pearson's correlations for B. subtilis (upper right numbers) between log2 FPKM of 5 hour time points on each of the three different days. Samples 2 and 3 have much more similar patterns compared to 1-2 and 1-3. Additionally, these two comparisons have a lower correlation than the 5-8 hour E. coli comparisons, suggesting greater variability in the growth/biological conditions for B. subtilis. Scatterplots are gene-gene Log_2 FPKM comparisons, with a red linear regression line. Blue plots are histograms for frequencies of gene FPKMs.

2.4.3 Minimum Inhibitory Concentrations and Mixed Media Growth

We designed minimum inhibitory concentration experiments designed for E. coli Nissle and Pseudomonas protogens PF5. There were some contamination issues with the initial Nissle run, so that organism had to be rerun. In addition to the work at Strateos, an MIC analysis pipeline was created to automatically QC and analyze results from the MIC experiments. A software program was created that takes in the output from Strateos and outputs MIC results, as well as figures for QC validation. Results can be seen for Nissle (Figure 77) and protogens (Figure 78). From the results, recommendations can be made for which antibiotics to use for host selection. For example, in Nissle, tetracycline, ampicillin, and chloramphenicol would be good choices. Whereas, in protogens, kanamycin and tetracycline are ideal choices.

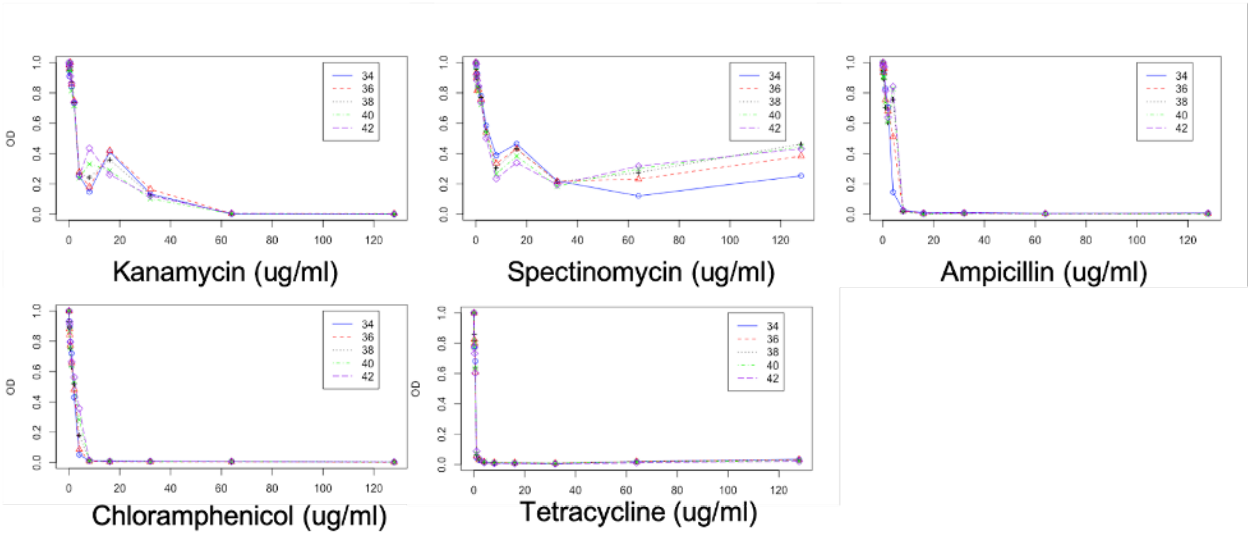


Figure 77. MIC results for E. coli Nissle. Chloramphenicol, tetracycline, and ampicillin are ideal choices for selection of modified E. coli Nissle. E. coli Nissle showed some tolerance to spectinomycin at higher concentrations that increased with time. While kanamycin might be ok at higher doses, the dose response curves are less than ideal, making other drugs better initial choices.

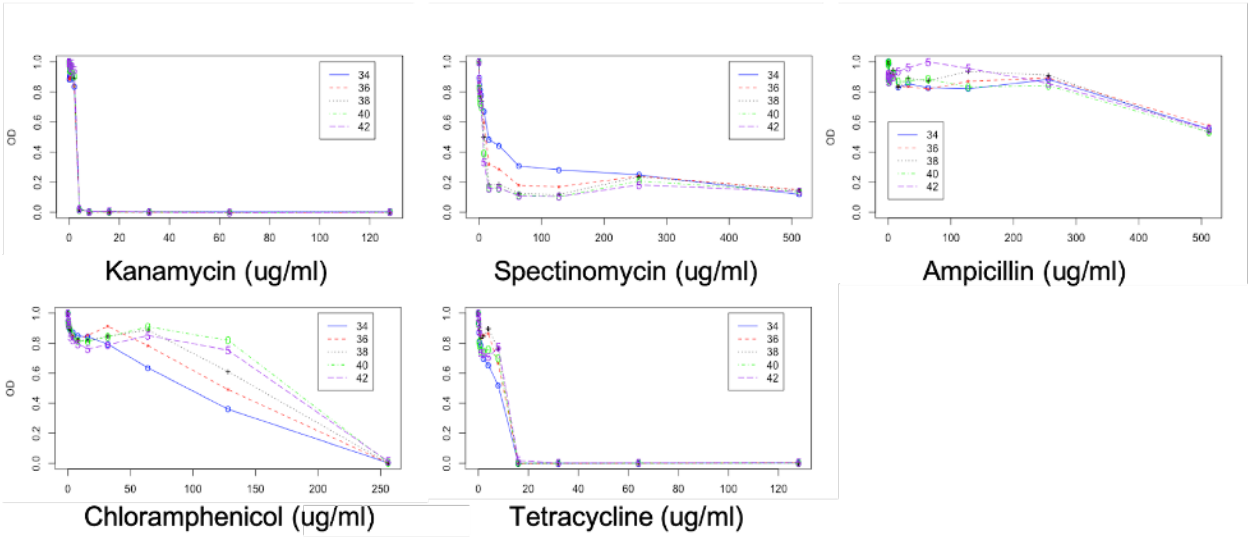


Figure 78. MIC results for Pseudomonas protogens PF5. Kanamycin, and tetracycline are ideal choices for selection of modified protogens. Protogens showed some tolerance to spectinomycin at higher concentrations that slightly decreased with time. MIC50 for chloramphenicol greatly increased with time, suggesting an increased tolerance with exposure. Protogens was resistant to Ampicillin, which is common for Pseudomonads.

Minimum inhibitory concentration experiments were run for six additional different organisms including *B. magaterium, L.* planeterum*, S. pasteurii, P. brenneri, P. chloraphis,* and *V. natriegens*. Each organism was tested with five drugs: gentamicin, spectinomycin, ampicillin, tetracycline,

and chloramphenicol under log phased aerobic growth over a 36-hour time period. We are currently having problems getting repeatable growth in *L. planterum*, a facultative anaerobe. While growth rates are consistent in the first few hours, the controls diverge with a ~2x difference in total growth, which makes any drug effects unreliable. Additionally, *L. Plantarum* had death shortly after stationary began, something that no other organism exhibited. However, the other 4 organisms had highly repeatable growth curves. An ideal drug for selection (shown in Figure 79), is a drug that has a low MIC and is consistent with time. A drug that is ineffective is one where the organism is resistant over part, if not all of the times measured (Figure 80)

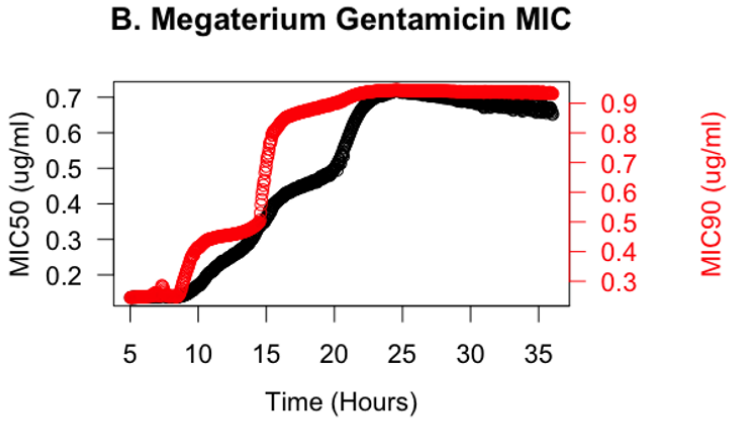


Figure 79. MIC50 (shown in black) and MIC90 (shown in red) are consistent (less and a 2x difference in MIC) for most of the times measured. Gentamicin would be one of the preferential drugs for synthetic selection in B. megaterium.

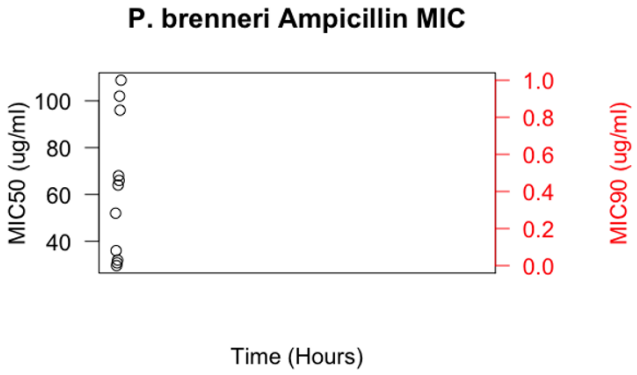


Figure 80. MIC50 (shown in black) and MIC90 (shown in red) are nonexistent for most of the times measured. The lack of MICs suggests resistance, which is not surprising for ampicillin and Pseudomonas. Ampicillin would not be a good candidate for synthetic selection in P. brenneri

Mixed media experiments were also run in 5 organisms: *B. megaterium, E. Nissle, P. protogens, P. chloraphis, and P. brenneri*. Organisms were different then the MIC experiments due to requirements for the organism to be able to grow well in both LB and M9CA and to have another organism in the same growth temperature (so as to not waste plates). While M9CA is a minimal media, it supplemented with nutrients to promote growth and DNA replication. However, the expectation was that M9CA would have both lower growth rate and lower carrying capacity. Surprisingly, M9CA only had a lower carrying capacity. Media had minimal effect on growth rate with a fairly large (depending on organism) effect on carrying capacity and time in log-phased growth (Figure 81). While M9CA tended to have the lowest carrying capacity, mixed media tended to have the highest, likely due to the supplemental nutrients.

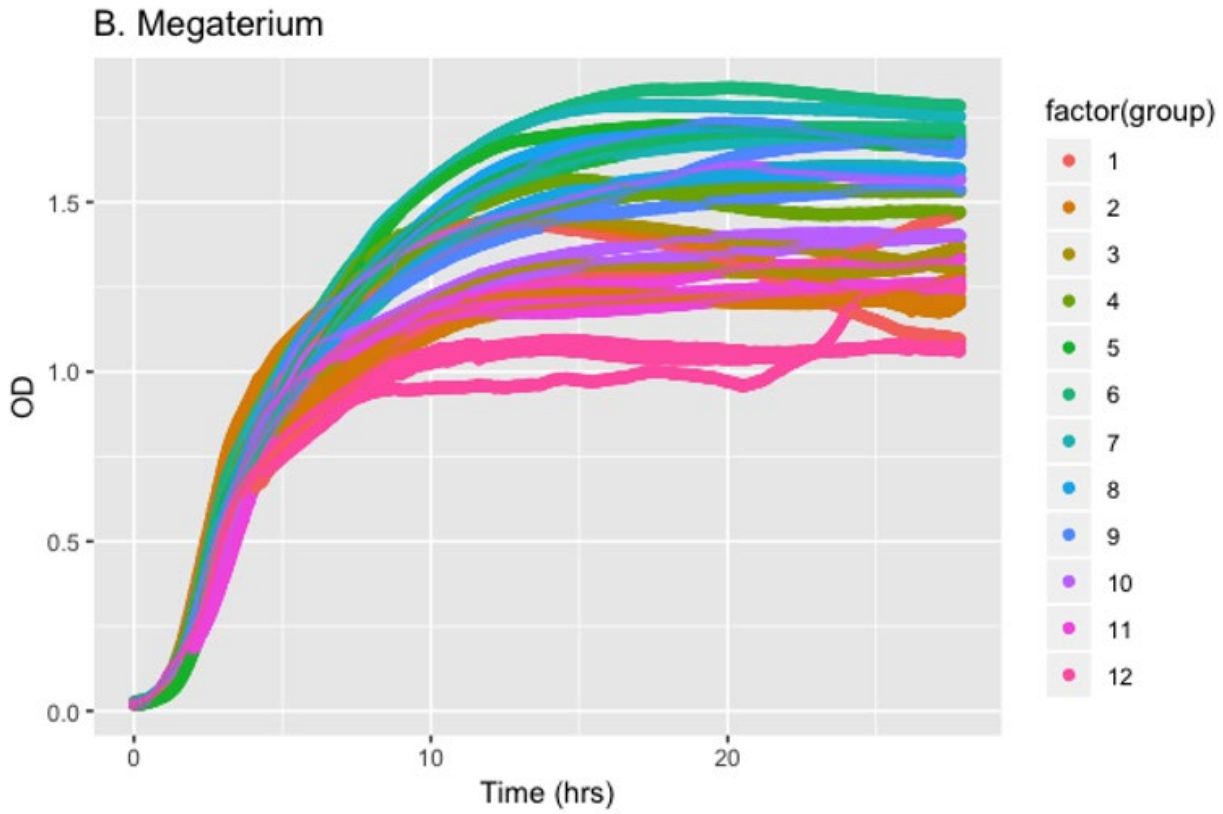


Figure 81 Mixed media growth in B megaterium. Group 1 is 100% LB, group 12 is 100% M9CA with intermittent groups representing 10% shift from LB to M9CA (2 is an exception as it is 5% due to having 1 extra column). Media composition between LB and M9CA had minimal effect on the growth rate, while having a much bigger effect on carrying capacity and time in log-phased growth. This is ideal for selecting times to compare media as the growth rates will be similar (as long as a time is selected while in log-phased growth).

2.4.4 Perovskites

Due to their O(N^3) calculation time, Gaussian-processes (GPs) that underlie BO's have historically been employed only in cases where data is scarce. This is unlike the ProStab problem, which provides >10^5 datapoints. We therefore tested several variants of so-called "sparse-GP's," which have been reported to get around the cubic data dependency. Unfortunately, we found that the majority of these did not converge when provided the ProStab data, so we redirected our efforts to more suitable algorithms. We first focused on developing methods for automatic selection of features to be used in a complex model. We started by employing regularization methods to linear models (using the 110 features provided by UW), and over the course of the quarter increase the sophistication to quadratic models comprising thousands of terms, yet that are not at risk of overfitting by enforcing regularization. The feature importance of the quadratic terms can be seen in Figure 82 in the score column, interestingly the quadratic terms all out-ranked the linear form. These models (still under development) are competitive (R^2 = 0.422 on new train/test split and R^2 = 0.375 on consistent TA1 train/test split) against some of the new models proposed by TA1 groups, even though we do not utilize any new observations beyond the 110 original features.

index	term	coef
4979	hphob_sc_contacts,netcharge	0.2794
5848	netcharge,alacount	-0.1837
120	alacount,buried_minus_exposed	0.1680
1641	buried_minus_exposed,abego_res_profile_penalty	0.1675
1010	abego_res_profile_penalty,hydrophobicity	0.1595
5115	hydrophobicity,chymo_with_lm_cut_sites	0.1588
2310	chymo_with_lm_cut_sites,avg_all_frags	0.1500
1082	avg_all_frags,avg_all_frags	0.1496
1056	avg_all_frags,avg_all_frags	0.1383
1101	avg_all_frags,abego_res_profile	-0.1350
902	abego_res_profile,avg_all_frags	-0.1322
1089	avg_all_frags,buried_np_afilmvwy	-0.1242

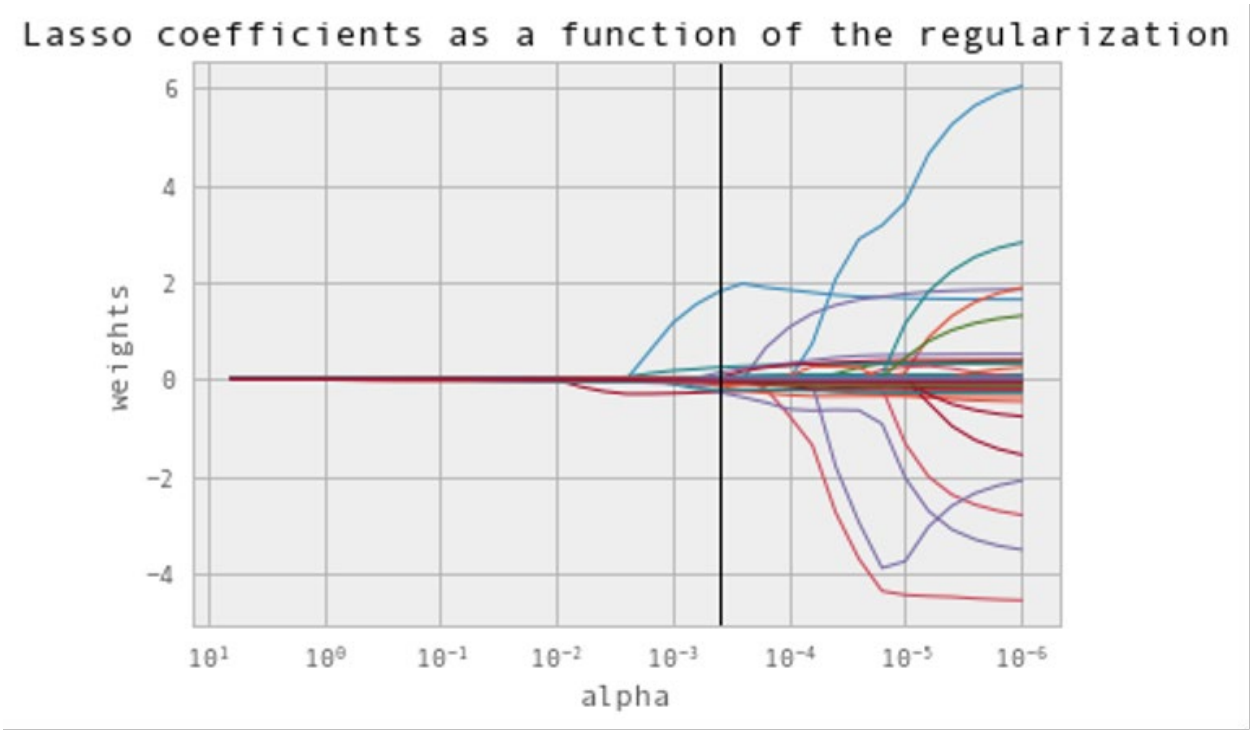


Figure 82. Lasso computer Rosetta terms and alpha value. Top: The Lasso computed Rosetta terms that are major contributors to ProStab stability score. Bottom: The computed alpha value which thresholds the scores that get minimized to zero during significance evaluation.

Perhaps more significant is that our approach will be able to assess and integrate findings from TA1 teams. For example, we have initial findings (still to be confirmed) that the Prot2Vec approach does not capture more new information beyond the original 110 measurements (though it still certainly has utility, e.g. it could serve as a proxy that may be faster to compute). In contrast, we find the topological data analysis (TDA) metric adds significant (R2 = 0.48 on new train/test, R2 = 0.40 on consistent TA1 train/test) information to the model, which indicates that it captures protein-folding information not already represented within the 110 original features. When the regression was done on the consistent train/test set from Hamed the performance was generally worse across the board as compared to performance on a new train/test split. If a model performs significantly better when tested on its own training data, it is likely overfit. Ultimately, our framework can incorporate all prediction methods from all TA1 teams and automatically determine which ones should be combined to form the best predictive model. This will be a focus for Q4.

This analysis also highlighted subtler aspects of the prediction. For example, we learned that certain features of the 110 were not predictive when used alone, but improve the prediction (a little) when multiplied by other features, indicating the importance of modeling interdependencies. We also identified structural families that were "easier" and "harder" for models to predict. (For example, 'HHHH or 4h' is easy, and 'coil' is hard, based on individualized cross-validated R^2. This has implications for future experimental designs: Looking forward, the biggest payoffs may result from focusing on improving our ability to make predictions in the "harder" structural classes.

In parallel, we have also started developing our deep neural network (DNN) model, to ascertain whether DNNs with automatically-inferred hyperparameters and embeddings could perform better than polynomial models. Initial results indicate that DNNs are not superior to polynomial models if we limit the inputs to the 110 features, but that they may perform better if the DNNs are also provided with the underlying protein sequence information.

Given the propensity of the model to predict location outside the statespace for a given amine, we implement constraints into the BO-GP model, requiring calculation of the plane equations comprising a convex hull of a given amine's State Set and propagate the constraints through the model's acquisition function and prediction criteria. This effectively reduced the potential testing points by up to 62% for the ethyl amine (Figure 83).

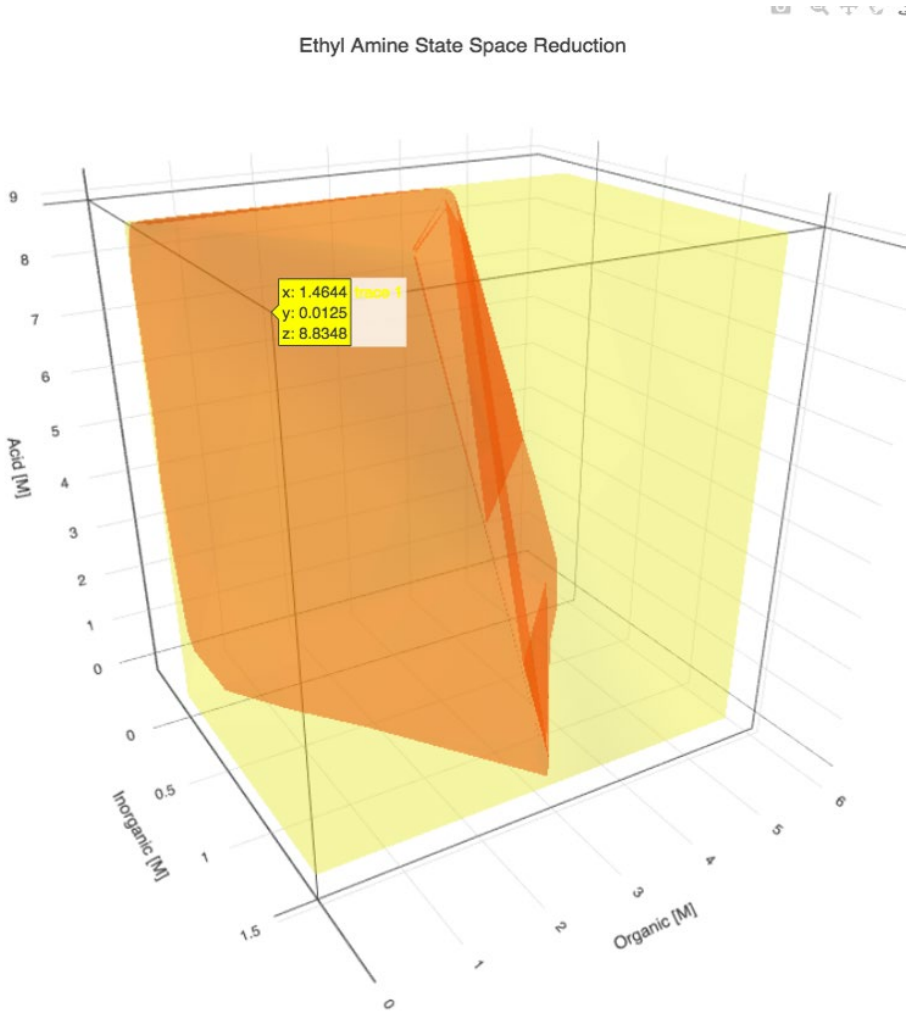


Figure 83. Convex hull limiting the statespace of the ethyl amine predictions

We simulated what the GP posterior variance would be if randomly sampling the State Space for ethyl amine. A GP model was trained using the Training Set for each experiment cycle, inclusive of all ethyl amine data generated up to that date's experiment. To create an adequate coverage of the ethyl amine domain for evaluating the GP model, we generated a grid of ~2940 points within the State Space. The trained GP model was then used to predict the mean and likelihood variance of the grid, and the average likelihood variance of the grid used as the overall posterior likelihood variance (PLV) of the model. Using this grid evaluation technique, I tested batches of 86 and 48 random samples and the actual samples over 9 iterations observing a total GP variance change of 0.169, .133, and 0.404 respectively. Also, the average change in run-to-run PLV is -0.006, 0.007, and -0.047 for 86 random, 48 random, and actual samples respectively, clearly showing that random sampling does not improve (reduce) the models PLV nearly as well as the Bayesian method (Figure 84). At experiment iteration 7 the convex hull of ethyl amine increased significantly due to improvement of the State Space generation workflow, which

correspondingly increased the model variance when evaluated with the grid technique. It is noteworthy that the PLV can be seen to be quickly decreasing after two experiment iterations, showing how the BO-GP rapidly identifies new regions of high uncertainty and suggests samples in that space.

The rate of change of PLV may be a viable method to determine whether an amine's State Space has been sufficiently explored or not, or at least as a means of improving the model's predictive capacity. Once the rate of PLV trends to zero there will be no further improvement in model predictive power. The interpretation here is that the GP for the model could still be used for that amine in order to make predictions, but the Bayesian sample selection technique would no longer have a purpose as the expected model improvement is zero.

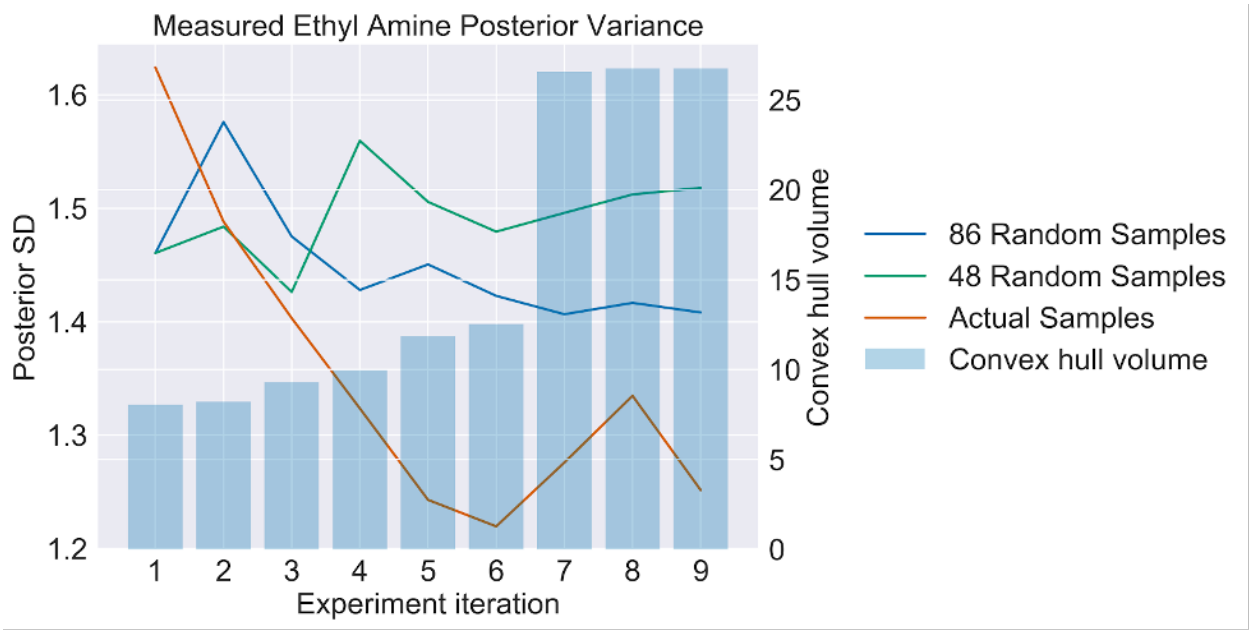


Figure 84. Rate of change in the convex hull volume explored based on the number of additional experimental points seen

We reduced all amine State Spaces by up to 62% (eg. ethyl amine) by implementing a Convex Hull into the BO-GP model, have produced metrics for evaluating BO-GP model performance in terms of improvement and success, and defined a stopping criteria for the optimization cycle where further sampling would no longer improve the GP model. The next steps will be to continue optimizing all possible amines and initiate BO-GP on new amines, establish our proposed metrics for model evaluation on the ethyl amine by continuing the current experiment cycle until we reach a reasonable stopping point, either defined by our metric or agreed upon within the CP. We will expand the BO-GP optimization to the time and temperature dimensions for ethyl amine when experiments are available, and potentially integrating more of the chemical features defined by Haverford. Finally, we have an optimization script uploaded to Gitlab

(https://gitlab.sd2e.org/acristo/perovskite-TA2/tree/master) that is parameterized to run the BO-GP prediction model for any amine ready for TACC integration.

If there are no improvements in the posterior likelihood variance (PLV) after a run, possibly two runs, then the model is not gaining any more knowledge about the state space. When the potential knowledge gain is below a threshold, which is some value close to zero, the suggestions can follow a 100% exploitation protocol. If reproducibility is not 100% then the PLV increases which can be explained as there is more uncertainty that a given reaction condition results in an expected outcome.

The reproducibility for Ethyl amine in the 65-70 Celsius range was calculated by binning samples by a precision value and counting how many of these distinct groups with more than 2 samples contained identical outcomes or not (Figure 85). If we use decimal precision of 0.1M for the three reagents 'organic', 'inorganic', 'acid' then crystals with score 1 have 88% reproducibility and crystals of score 4 have 55% reproducibility. We can evaluate how precision affects reproducibility by scanning the range of precisions and calculating number of groups and samples per group (Figure 86).

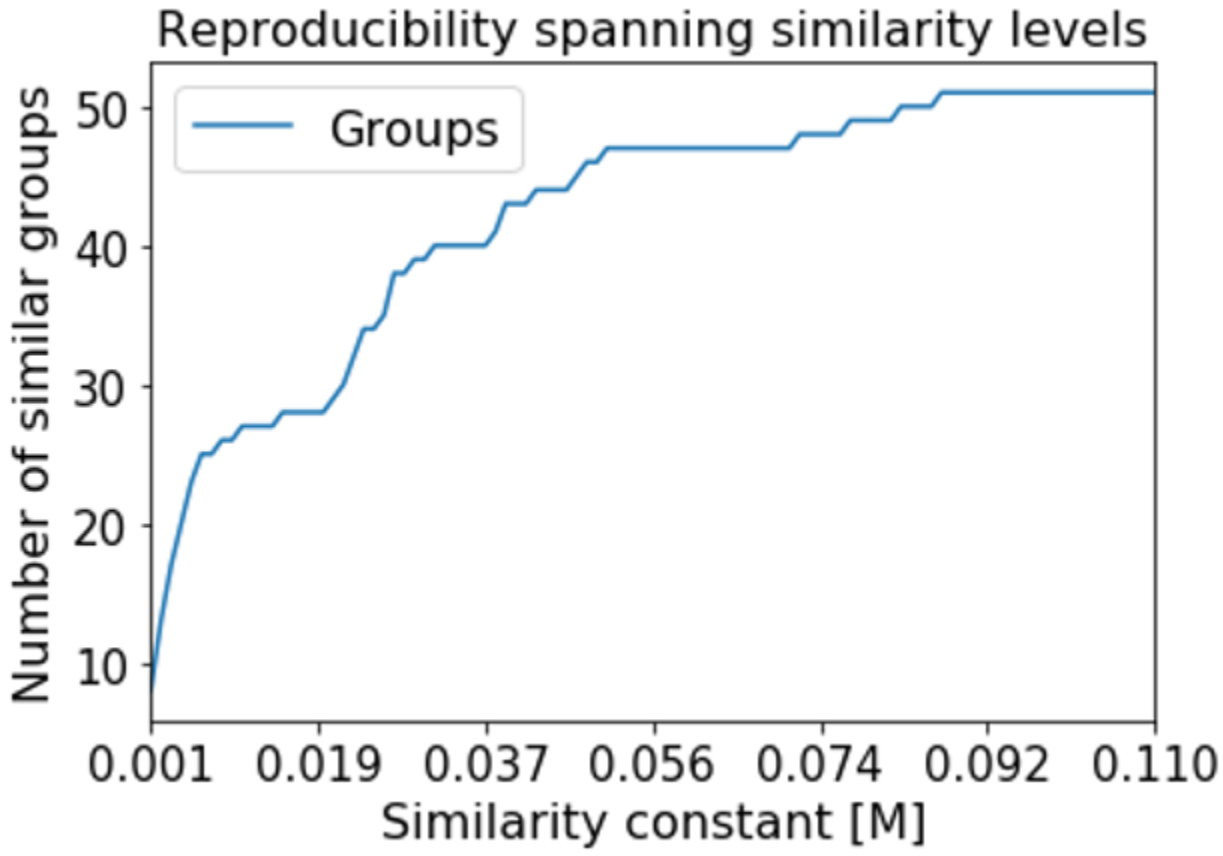


Figure 85. Groups of replicate tests. Using a decimal precision of 0.1 for the ethyl amine between 65-70 C we observe 51 groups of replicate tests comprised of 174 samples. The distribution of groups containing a given number of samples is shown, a majority of groups contain two samples.

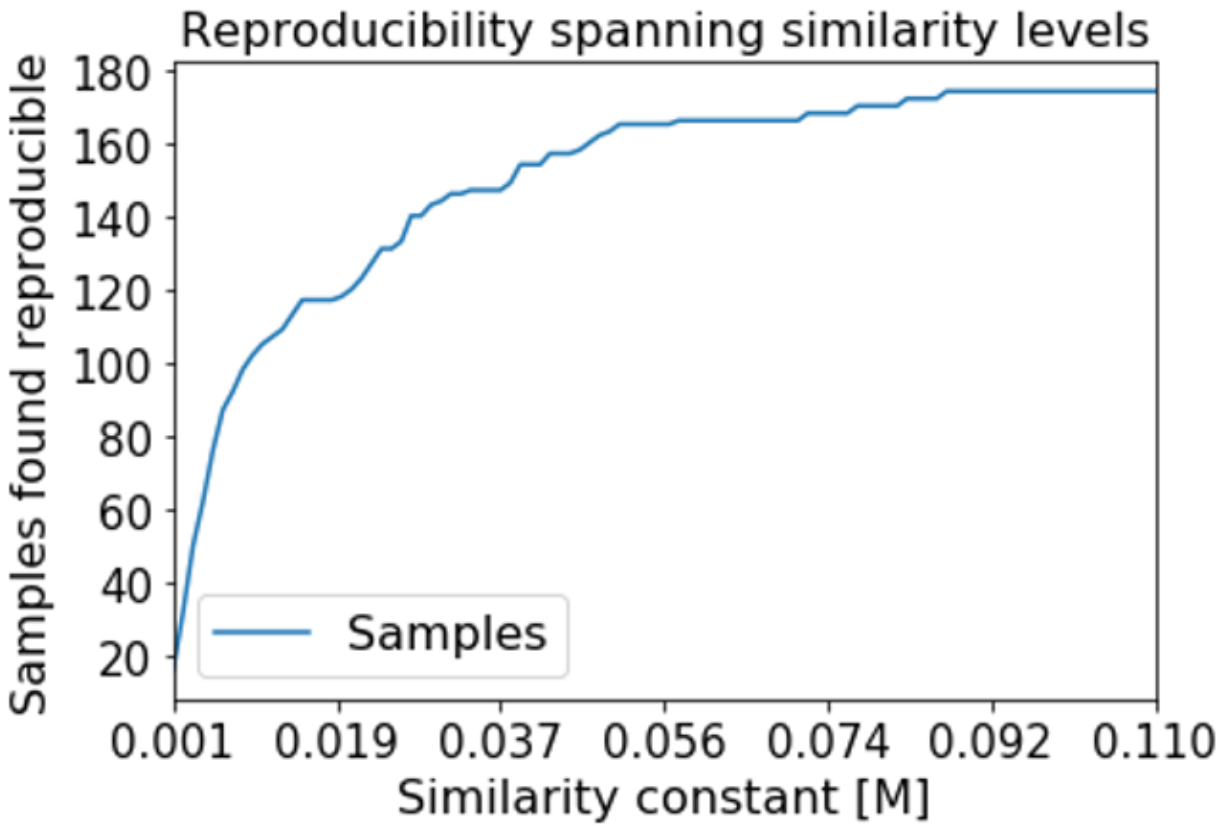


Figure 86. Scanning the precision constant from 0.001 to 0.11 we observe how samples begin forming groups. This can be expanded upon by using the percent reproducibility per group to get an idea of how precise a reproducible result is, but this idea needs more data to assess.

The next step is to work on predicting experimental conditions that produce high quality crystals. We start out with an initial run of conditions that explore the experimental space for producing crystals (Figure 87). Amines with at least one high quality crystal will be explored further with multiple different machine learning techniques. To predict crystal scores, we utilize a Bayesian optimization method that first predicts conditions with high uncertainty that could lead to production of a crystal. These points will then be tested to improve the model for a final prediction. The final prediction then focuses on predicting areas where crystal scores might produce a local maxima. The scores are then filtered to only include conditions that produce high quality crystals, which will then get validated for accuracy. The above is part of a challenge of competing methods.

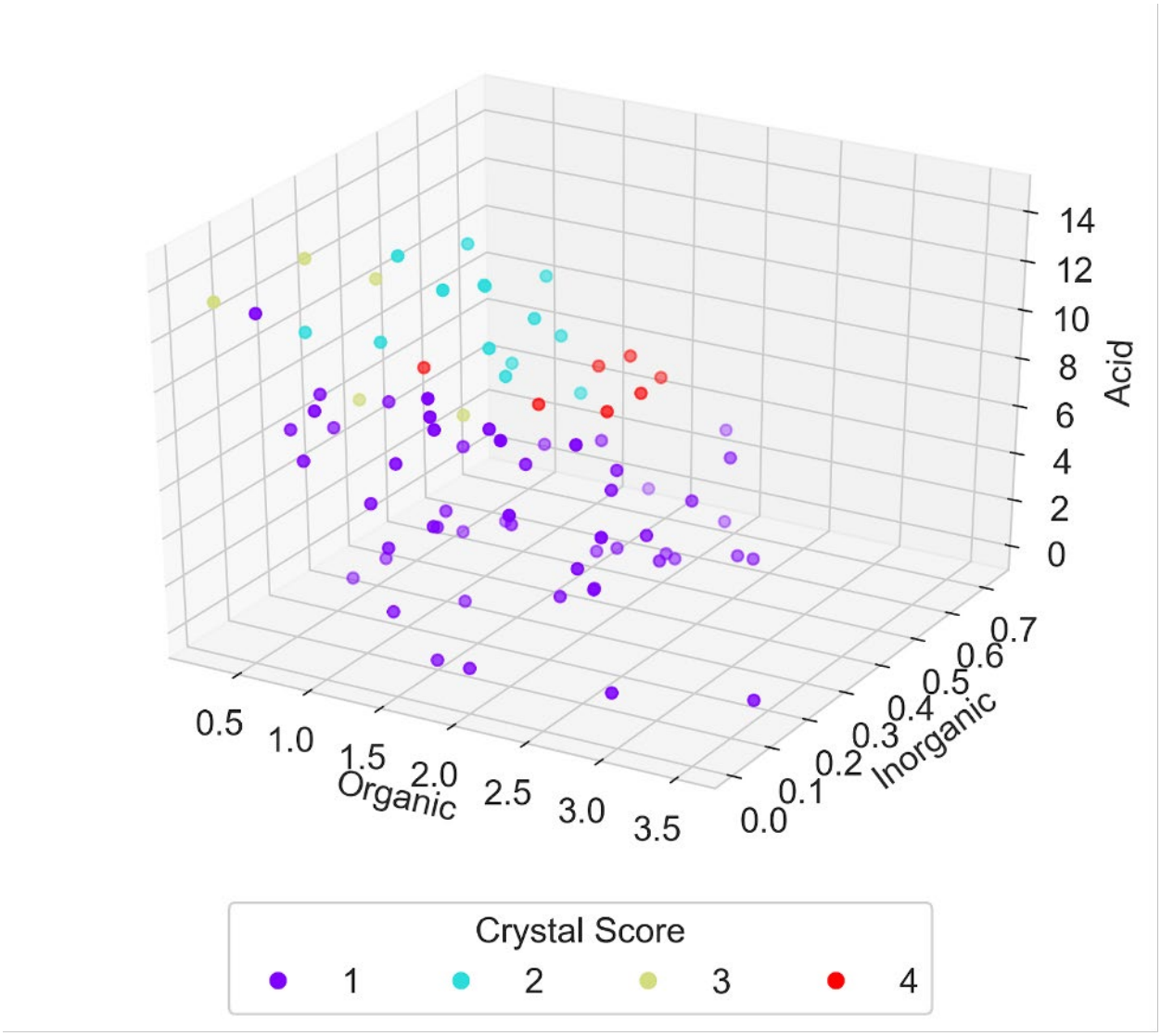


Figure 87. Experimental conditions for an amine of interest that explores much of the desired experimental space. Crystal scores of 3+ are desirable and only appear in a small fraction of the experimental space. The data in the run will be used by a Bayesian optimization method to estimate areas of uncertainty to help improve the model. After exploring these areas, a prediction will be made to identify areas of high crystal scores for testing and evaluation.

Identifying model strengths and weaknesses is necessary to both evaluate and improve the model. For the perovskite challenge, we developed a Bayesian Optimization model. A Gaussian process-based Bayesian Optimization (BO) algorithm utilizing spatial constraints was implemented using the GPyOpt package with the following parameters: Matern32 kernel, local penalization, automatic relevance determination, jitter, and the expected improvement acquisition function. Constraints for the BO were generated by computing a convex hull from the stateset for each amine using the "ConvexHull" function from scipy. For optimization, the independent variables were: molar organic concentration, molar inorganic concentration, and molar acid

concentration. The crystal score was considered as the dependent variable, and the values were treated as negatives for minimization within the BO. For each round of active learning, a location within the stateset was chosen based on the lowest Euclidean distance to the location suggested by the "suggest_next_locations" function. Resulting crystal scores for the tested location were then incorporated into the training data to compute a suggestion for the next round. In the model validation round, CMA-ES was utilized to identify points in the stateset likely to exhibit high-quality crystals. CMA-ES was run 20 times, seeding each search with the 20 points measured in the training and active learning rounds (1 point per CMA-ES run). As different seeds would often converge to the same minima, CMA-ES produced too few distinct final guesses. To fill in the remaining guesses, "suggest_next_locations" was executed 100 times. The resulting locations were ranked based on expected crystal score, and the remaining guesses were filled in with the top-scoring points. As before, the locations were then mapped to their closest locations in the stateset.

To visually evaluate how the models changed during the active learning stages, we created a 2d representation of the model regression with standard deviation. This representation was necessary as each of the other initial models were classification-based models, and the active learning could not be directly compared across the different model types. Each of the amines produced a different quality of models at the end of the active learning stage (Figure 88). What we learned from this process is that the model does not have to be representative of the actual real results to be able to predict likely perovskite formations (ie. measured values were between 1-4; despite this, one amine had unrealistically high predictions in one iteration, 1e16, and despite this still have high accuracy). Model correlation to actual measured states were more important for serendipity, where models with low standard deviation and closer match to expected range of 1-4 were better able to accurate identify crystals across the entire space, rather than in just the most likely areas.

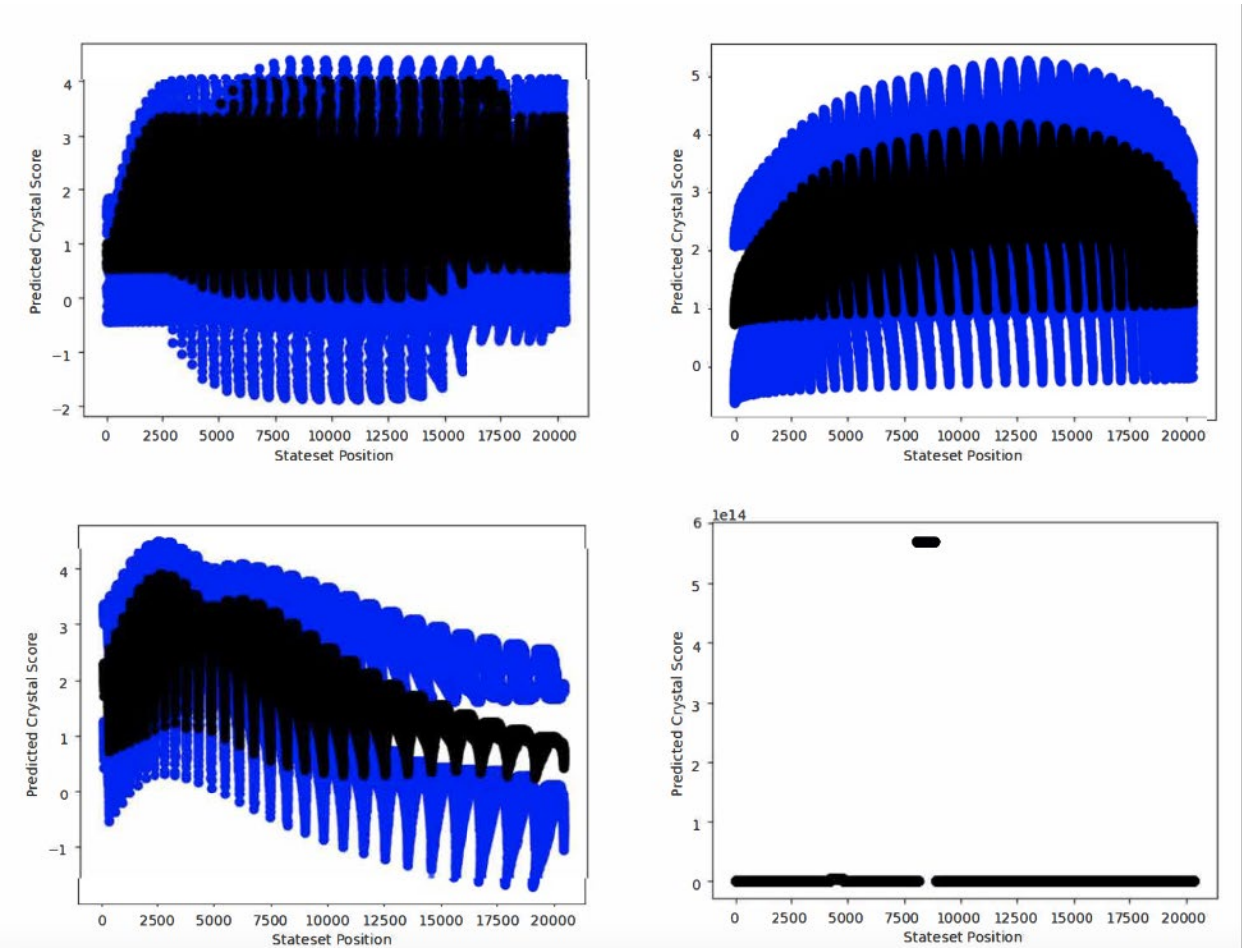


Figure 88. 2d representation of four different models after the last active learning stage. Black represents the predicted crystal score for the set of acid, organic, and inorganic concentrations at a given stateset point, while blue represents the standard deviation of that score. Higher values tend to have lower standard deviations due to the model intentionally over exploring areas that are likely to produce crystals. The model representing the underlying space is not necessary to accurately identify areas that are likely to produce crystals as maxima are explored rather than raw values. However, this is not the case for serendipity where more of the space is explored.

2.5 Conclusion

Our work within the greater SD2 helped facilitate reproducibility on the computational side as well as providing valuable insight into the biology of circuits and synthetic parts. Our RNASeq tools are part of pipelines that are publicly available for use by anyone that has an interest in an automated pipeline. Additionally, given the modular nature of the pipeline, users can pick and choose which parts they wish to use, as well as replace the ones they do not want to use. In addition to the creation of the pipeline, we have also used this pipeline to analyze a wide array of RNASeq from E. coli and Bacillus and to perform a reproducibility study across these two organisms that could be utilized to improve wet-lab procedures. We have also worked to perform predictive modeling on perovskite crystal formation as well as take part in a group competition to determine the best model.

3 SUBGROUP 3: DENSMORE LAB (CIDAR GROUP, BOSTON UNIVERSITY)

3.1 Summary

The major activities and contributions of the CIDAR group to the SD2 Project were centered around developing and implementing software and hardware tools for the forward design and construction of biological systems in an automated, standard, multiplexed, high-throughput fashion, allowing many different configurations to be tested simultaneously. We developed hardware automation scripts and software tools to help investigate and optimize DNA design and assembly through quantifiable metrics in a standard, modular and automated fashion. In addition, a collection of standard physical DNA fragments and its online repository were developed to test the functionality and efficiency of automated workflows, methodologies, and the number of samples for the construction of relatively complex, large DNA constructs.

3.2 Introduction

We improved the genetic circuit design automation tool (CELLO, cellocad.org) to support the design of DNA circuits for five new organisms (e.g., Bacillus subtilis), the splitting of a tandem promoter (present in NOR gates) into unique landing pads in the genome, and flexibility for gate structure representation in the user-constraint file (UCF). Next, we investigated the formulation of the stability metric of SR-latches (e.g., sequential logic circuits) to expand the predictive design of more extensive, complex sequential logic circuits using the Cello tool. Finally, we refined the collection of universal metrics to describe genetic devices with different functions.

The DAMP lab provided DNA design and construction tasks to the SD2 project. We developed automated laboratory workflows, such as DNA assembly, bacterial cell transformation, cellular plating and picking, and functional assays, including Cello's modular genetic circuits. The DNA circuits built can be transformed into bacterial cells and tested for their assembly performance and expected biological function (as determined by Cello models). One example is the construction of complex genetic constructs, such as sequential logic circuits (e.g., the above SR latches required for expanding Cello capabilities). Finally, we used liquid handling robot-based workflows to construct and test the DNA circuits designed by Cello and other software tools.

Next, we expanded the library of modular DNA fragments and created Cello-adapted parts to enable the fast synthesis of new generation Cello circuits, such as simple SR-latches. We believe this can allow the assembly and assay of complex Cello-circuits in a modular and fast fashion. The final goal was to use these interactions with other SD2 needs to increase the number of projects and customers engaged with the DNA facility at Boston University.

3.3 Methods, Assumptions, and Procedures

Cello uses a fast iterative method (simulated annealing) for searching the possible assignment space. DSGRN can accept a specification of a regulatory network and a description of a behavior of interest, e.g., oscillations, bi-stability, or conditions on an output node given the behavior of input nodes (equivalent to a truth table). DSGRN was investigated to guide the gate assignment procedure in Cello.

Experimental data for gates, sensors, outputs, strains (i.e., chassis) in UCF, Input, and Output files, such as those for empirical metrics for each transcriptional unit (e.g., a NOT gate or an Input-sensor) and output signal (e.g., YFP fluorescence signal), such as cell growth used a plate-reader, and fluorescence intensities used a flow-cytometer.

For Puppeteer development, the Software & Application Innovation Lab (SAIL) team created python modules, which handle calls to the Trident API and Autoprotocol conversion. The new NGINX and Gunicorn servers offer greater scalability and security. SAIL also created scripts to deploy these servers on the Massachusetts Open Cloud (MOC) and docker containers. SAIL team also updated Puppeteer's user interface. The design was guided and reviewed by a UI/UX expert from Redhat and contained more explicit instructions and a more intuitive workflow for biologists. The team has also designed mock-ups for a complete user-interface overhaul of Puppeteer.

We used the Hamilton Star Liquid handling system and the Opentrons OT2 liquid handler to create automated workflows. The scripts written in Python and Ruby on Rail were used for automating laboratory protocols, i.e., a modular DNA assembly and bacterial transformation. Both protocols are hosted on private GitHub repositories and can be provided upon request.

To solve the bootstrapping problem of the foundry database, we used probability models (e.g., Markov model) and deep learning (e.g., neural network/LSTM) to learn generic representations of the planning process. For example, we wanted to split the ~10,000 plans from BIOFAB into two disjoint sets to mimic two labs to have a train and test set for our model. We used a modified Karger's algorithm and ran it multiple times over a cardinality range. In the context of deep learning, we implemented an LSTM which can learn to generalize plans by utilizing the contextual data of the signatures, and alternatively, graph neural networks (e.g., graph CNN, generalized adversarial network, etc.) that make use of the implicit graph structure utilized in Aquarium plans.

3.4 Results and Discussions

3.4.1 Developing a new version of the Cello tool (cellocad.org)

A discussion with Bree Cummins, Rob Moseley, Justin Vrana, and Dan Bryce led to planned modifications to Cellov2 to support assignment with new versions of the UW BioFab yeast gates that include inducible repressible promoters. Additionally, a new web-based UI was created for the software tool Cello. The interface supports connection to SynBioHub: a user may specify any public SynBioHub collection to use as a gate library instead of a UCF, and a user may submit a Cello-designed circuit to their personal SynBioHub account, all within the Cello2 interface. The Cello2 UI also features a settings page where aspects of each stage of the design process (logic synthesis, technology mapping, placement, SBOL generation, etc.) may be configured. Specification of settings in this manner was absent from the original version of Cello. Cello2 also features continuous integration, a modern token-based login scheme, and per-user log file routing. Finally, the Cello2 SBOL output was adjusted such that gates are instantiated as singular objects in a plasmid. Previously, the notion of a gate object was absent in the SBOL representation, and a "gate" would be recognized only by a human. This adjustment allows proper assignment of Hill function parameters as features of the gate object itself, not any of its parts.

Cellov2 was presented in a preliminary demo to the Voigt lab, along with a roadmap for the future of Cello development. It was demonstrated before members of the Voigt lab. A roadmap for future releases of Cello was also prepared and discussed. The roadmap is in Figure 89.

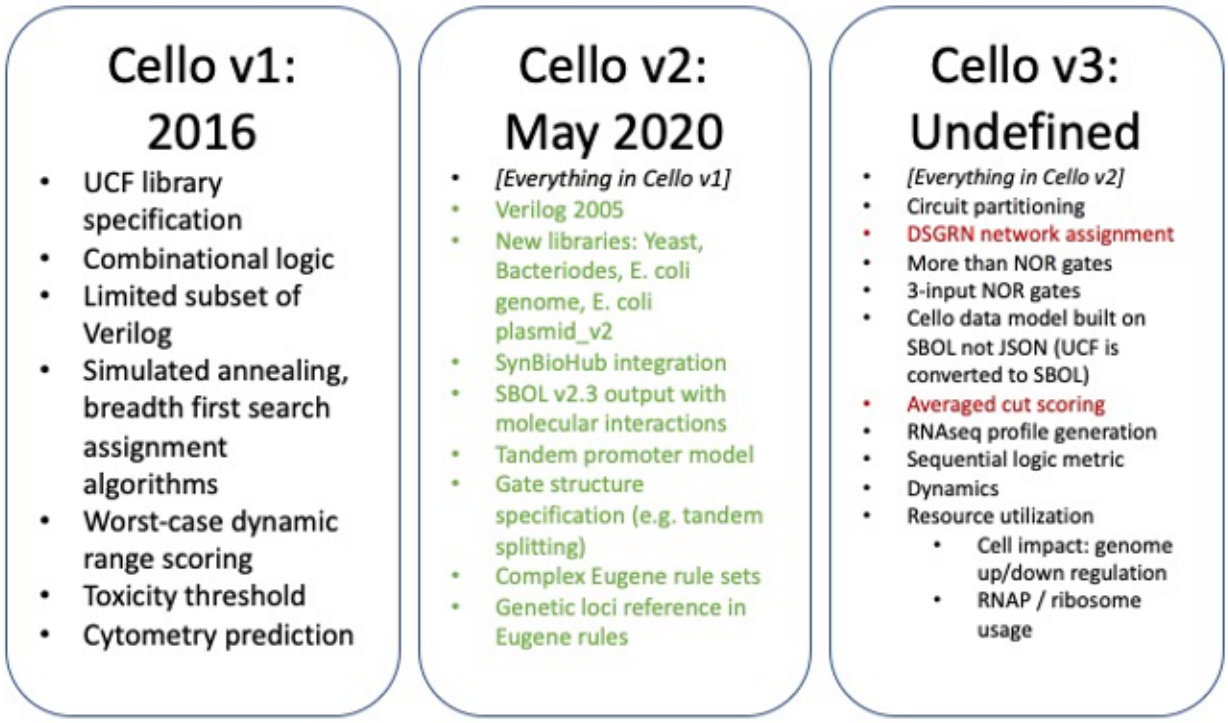


Figure 89. Roadmap of Cello releases.

It was provided enhanced deployment and testing infrastructure and is now outfitted with enhanced software infrastructure. Continuous integration via travis-ci.org is enabled on the Cellov2 repository, located at https://github.com/CIDARLAB/Cello-v2 and on the web app and GUI repositories, located at https://github.com/CIDARLAB/Cello-v2-webapp and https://github.com/CIDARLAB/Cello-v2-webapp-gui, respectively. More details about each of these modifications are found in the following subsections.

3.4.2 Enhanced Cello output

Cello now uses the Virtual Parts Repository API to query a parts collection and a set of molecular interactions stored in SynBioHub. These queries find and encode gate-to-gate interactions in SBOL. The interaction network is returned in the same SBOL file that the structural specification of a circuit design, e.g., a plasmid, is written by Cello. An example set of interactions displayed by SynBioHub is shown in Figure 90.

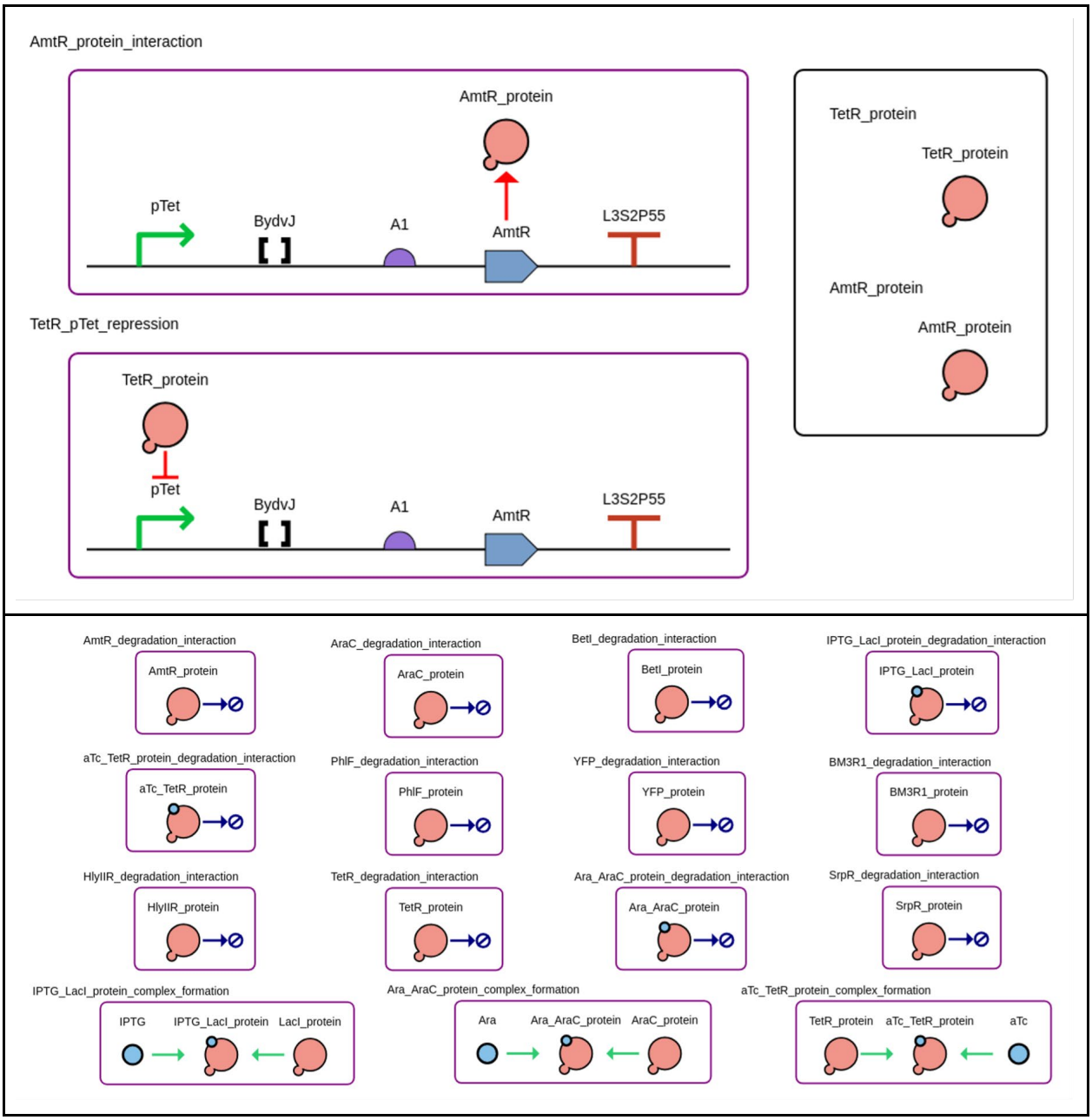
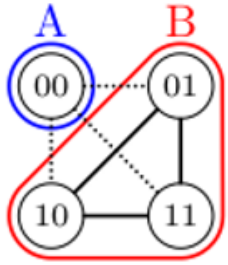


Figure 90. VisBOL representations of the molecular interactions at the level of the transcriptional unit and the entire circuit. One protein interacts with a promoter in the transcriptional unit, and a coding sequence produces another. The circuit-level interactions are composed of protein degradation and complex formation. All interactions are included in a circuit's SBOL representation generated by Cello and the Virtual Parts API.

3.4.3 Circuit Performance Prediction

Bree Cummins has identified two mechanisms to score a genetic circuit's truth table based on fluorescence distributions of an output reporter measured by flow cytometry for each input/output state of the circuit. Scoring of predicted and actual circuit performance was made based on Bree

Cummins' normalized cut and averaged cut truth table metrics. The two scores, the normalized cut and averaged cut metrics, are rooted in a graph-theoretic description of a truth table and differ from Cello's standard lowest-ON-by-highest-OFF ratio score in that all output states are considered, not just the minimum or maximum values of the on- or off-states (Figure 91).



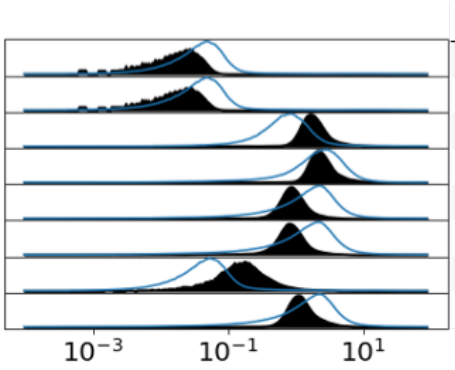
$$V = A \cup B$$

$$\mathrm{Ncut}(A,B) = \frac{\mathrm{assoc}(A,B)}{\mathrm{assoc}(A,V)} + \frac{\mathrm{assoc}(A,B)}{\mathrm{assoc}(B,V)}$$

$$\mathrm{assoc}(X,Y) = \sum_{x \in X, y \in Y} w_{x,y}$$

Figure 91. The specification of the normalized cut score. A graph with nodes representing the different states of the circuit is first fully connected then partitioned to match other truth tables. The edge weights correspond to the Wasserstein distances between cytometry distributions. A lower score indicates a more faithful implementation of a particular truth table.

The normalized cut score was computed for several circuits reported in the original Cello publication (REF: Neilsen et al., 2016). The circuits were reported as a success or a failure in the publication based on whether they were deemed to implement the truth tables specified in the circuits' designs. Computing the normalized cut score for these circuits led to a few interesting cases where circuits marked as success in the publication (presumably by qualitative assessment of the distances between the cytometry distributions) would have been marked as failures according to the normalized cut score. An example is shown in Figure 92.



	Wolfram	min Ncut
000	0	0
001	0	0
010	1	1
011	1	1
100	1	1
101	1	1
110	0	1
111	1	1

Figure 92. A circuit marked as successful in the Cello publication (Nielsen et al., 2016), but where the best-normalized cut score was lower than the specified truth table.By the normalized cut metric, the circuit would be marked as a "failure." The column denoted "Wolfram" is the truth table of the desired circuit behavior in the table. The column marked "min Ncut" is the truth table with the minimum normalized cut score of all possible 3-input truth tables.

Finally, a histogram of the normalized cut scores for the scored circuits, categorized by their being reported as successful or failed, is shown in Figure 93.

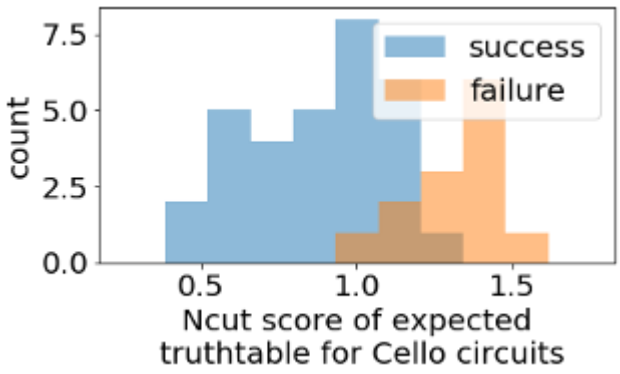


Figure 93. Histogram of normalized cut scores for 44 circuits scored from the Cello publication (Nielsen et al. 2016). "Success" or "failure" denotes whether a circuit was reported in the publication as a faithful implementation of the truth table specified during design.

3.4.4 DSGRN integration with Cello

DSGRN can accept a specification of a regulatory network and a description of a behavior of interest, e.g., oscillations, bi-stability, or conditions on an output node given the behavior of input nodes (equivalent to a truth table). Discussion occurred in SD2 2021 PI meetings about the relationship between Cello's design process and DSGRN. A few avenues for integrating these two tools were investigated in this project. First, DSGRN was analyzed to guide the gate assignment procedure in Cello. Given a regulatory network and a truth table, DSGRN will specify regions of "parameter space," i.e., the space of free Hill-function parameters corresponding to the nodes in the network, which could realize the given truth table (Figure 94).

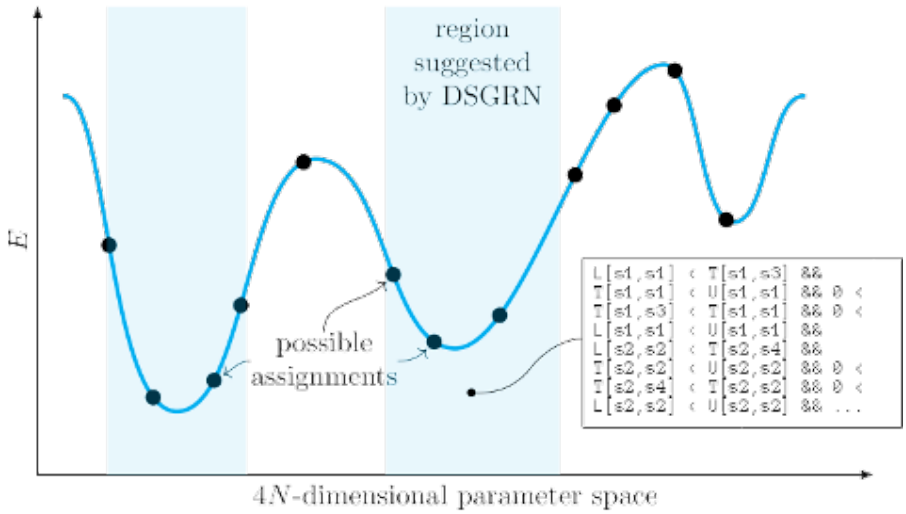


Figure 94. DSGRN suggests regions of parameter space for Cello to investigate. A region of parameter space corresponds to a set of inequalities on Hill-function parameters that would implement the given truth table.

Cello can now call DSGRN and read back sets of inequalities on Hill-function parameters corresponding to these regions. What remains to be implemented is searching these regions for discrete sets of gates within a library (Cello UCF) with parameters within these regions. Cello already uses a fast iterative method (simulated annealing) for searching the possible assignment space, and so reducing the space with DSGRN and then employing the simulated annealing search on each of these regions is unlikely to reap many benefits in time or efficiency of gate assignment, at least given the size of the circuits currently being designed (usually not more than ten gates).

Another possible means of integrating DSGRN and Cello uses DSGRN to rank crosstalk paths in each regulatory network. WheFor example, when spidering the YeastStates gates originating from the Klavins lab, Cello picks guide RNAs that minimize the net crosstalk (according to the matrix appearing in REF: Gander et al. 2017) to assign to the abstract regulatory network. Still, it then randomly assigns those guide RNA gates to the nodes in the network. However, given knowledge of which possible parasitic (crosstalk) paths would be more likely to hamper circuit function—information it was thought that DSGRN might be able to provide—a design tool like Cello could make a more intelligent decision about which guide RNA gate to assign to which abstract node in the network. For example, see Figure 95.

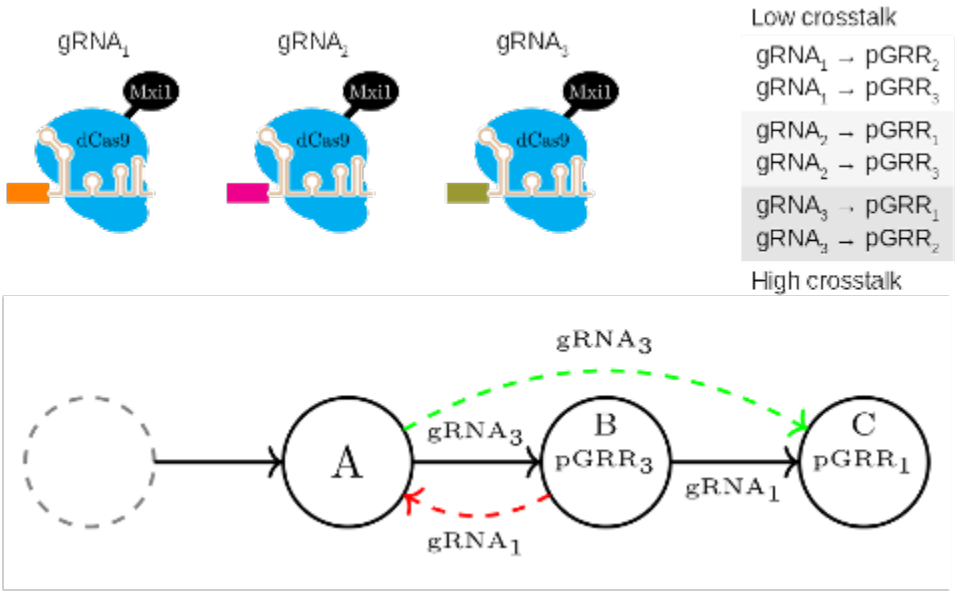


Figure 95. Example assignment of guide RNAs according to knowledge of crosstalk effects. The path from A to C is assumed to be least likely to hurt circuit function, and so it is assigned the guide RNA with the worst crosstalk behavior. The path from B to A is assumed to be the most likely to hurt circuit function, and so it is assigned the guide RNA with the best crosstalk behavior.

The suggested implementation of a crosstalk scoring was to add all possible crosstalk paths between node pairs, one at a time, then call DSGRN to assess whether regions of parameter space still exist in the circuit with a crosstalk path that might implement the desired function. The issue with this approach is that Cello would like to find a particular point in the parameter space as a gate assignment. So merely knowing that regions of parameter space associated with a network with crosstalk paths exist are not enough. We would like to know whether the point in the parameter space previously selected also lies within any newly suggested regions. However, this comparison is not well formulated due to differences in the dimensionality of the regions to be compared. This is an open question for the DSGRN developers and discussed with Bree Cummins.

DSGRN could function to evaluate graphs (regulatory network topologies) implementing a given function that Cello's logic synthesis stage might not have produced. Cello's logic synthesis stage is based on a tool that makes optimal gate-level designs to be implemented with transistors. Still, given the non-uniform response of the different biological gates in a library, i.e., responses without consistent transition thresholds or on/off expression levels, it is conceivable that a non-standard topology leads to an implementation that performs better than an implementation based on standard topology.

3.4.5 Tandem promoter splitting

To prevent recombination, the Voigt lab has begun physically splitting transcriptional units with tandem promoters into two transcriptional units located in different regions of the host organism's genome (REF: Park et al., 2020). According to gate type, the gates are ordinarily placed at one of three landing pads in the genome: Input, output, or logic. After a split, the second of the

two transcriptional units is placed on the landing pad with the copy number most similar to that of the landing pad originally targeted before the split. For example, if the "input," "output," and "logic" landing pads have copy numbers 1.4, 1.2, and 1.5, respectively, a NOR gate split into two units would see one of its units go to the "input" landing pad. The splitting rule is shown visually in Figure 96.

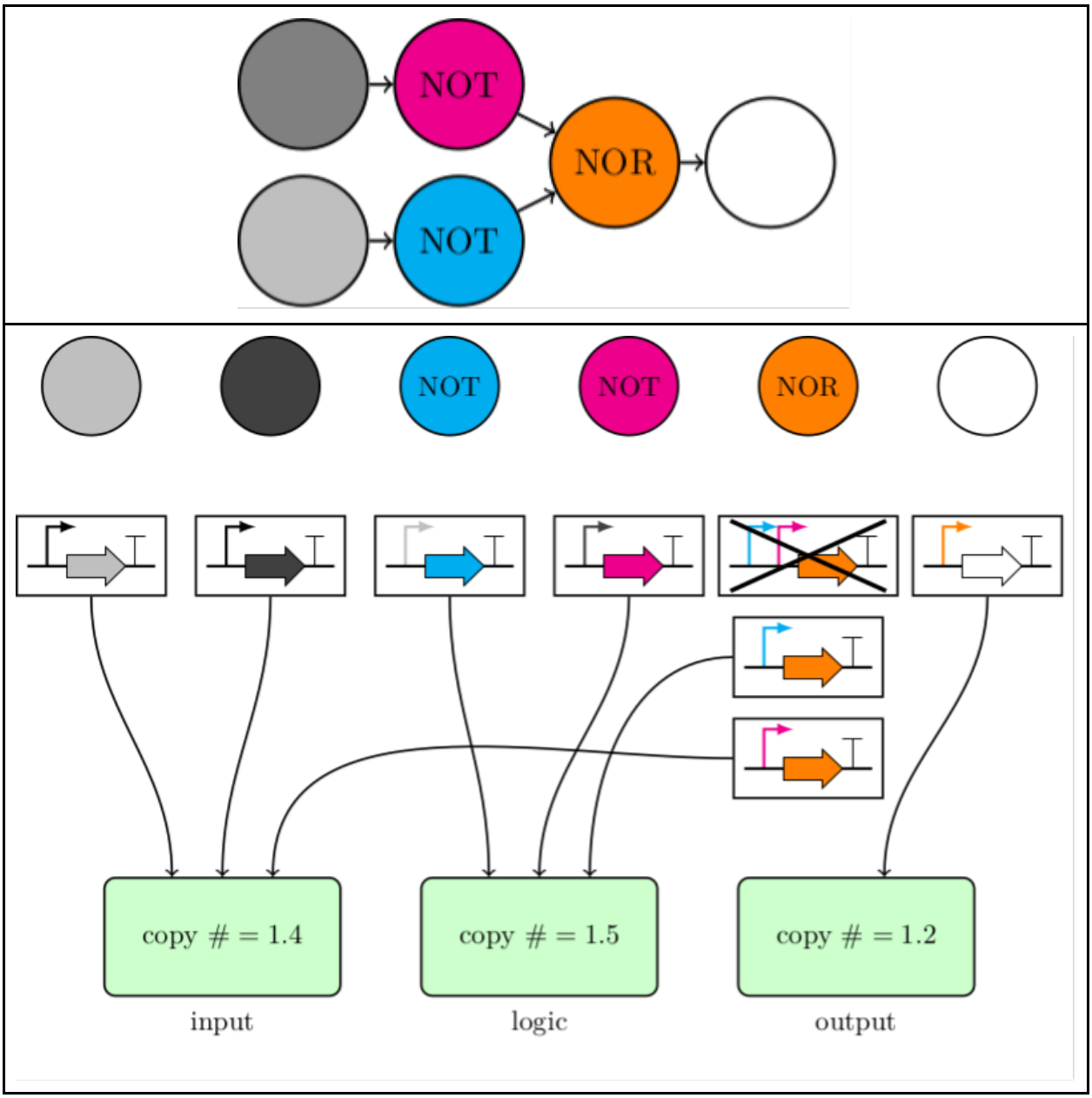


Figure 96. Tandem promoter splitting. The NOR gate transcriptional unit with tandem promoters is split into two, and one unit is reassigned to a different location.

Ordinarily, the transcriptional activity of tandem promoters is assumed to add in a simple linear combination, as in Figure 97.

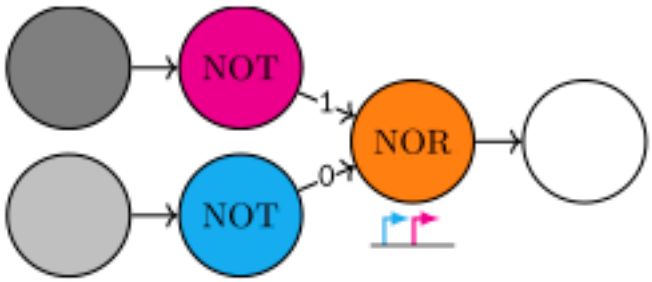


Figure 97. Using the tandem promoter model, edges are assigned a weight (an index) signifying the promoter order in the gate assignment procedure.

In reality, a repressed promoter in a tandem pair will diminish the transcriptional activity of the otherwise unrepressed, upstream promoter in the pair. Jonghyeon Shin in the Voigt lab has developed a model to account for this change in transcriptional activity at the upstream promoter due to the downstream promoter. In this project, we made this transcriptional unit splitting implementable in the new version of Cello.

3.4.6 New, flexible gate structure representation in the UCF

The gate structure was primarily fixed in Cello version 1, then implemented in version 2. While a designer creating a UCF could choose which parts made up a gate, the basic structure was assumed to be one group of parts per Gate in one location of a genome or plasmid. As a result, the Voigt lab has moved to splitting a two-input-promoter transcriptional unit into two units and placing the units in disparate regions of the genome to avoid recombination.

Initial solutions to allow a designer to specify this gate structure in Cello2 involved the use of flags in the UCF or on the command line that indicates, for example, that all two-promoter transcriptional units should be split. However, as each host organism requires unique variations to gate structure, encoding each possible structure to be selected with a flag becomes a burdensome effort to Cello developers. Instead, we have implemented a gate_structure collection in the UCF to flexibly deal with almost any Gate structure. For example, a designer might specify structures to define the regions where circuit sequences are to be inserted and rules for the placement of the fractional gate structures into those regions. For example, to specify a gate that can have up to two transcriptional units, where each is to be split, one would utilize the following structure:

```
{
    "collection": "gate_structure",
    "gate_name": "P1_PhlF",
    "devices": [
        {
            "name": "P1_PhlF_a_cassette",
            "components": [
                "RiboJ53",
                "P1",
                "PhlF_a",
                "DT5"
            ]
        },
        {
            "name": "P1_PhlF_a",
            "components": [
                "#promoter?",
                "P1_PhlF_a_cassette"
            ],
            "maps_to_variable": "x",
            "output": "pPhlF_a"
        },
        {
            "name": "P1_PhlF_b_cassette",
            "components": [
                "RiboJ53",
                "P1",
                "PhlF_b",
                "DT5"
            ]
        },
        {
            "name": "P1_PhlF_b",
            "components": [
```


The Gate is divided into two "device" objects, "P1_PhlF_a" and "P1_PhlF_b." The actual expression cassette for each device is another nested device. If a gate has only one Input (a NOT gate), only one of "P1_PhlF_a" and "P1_PhlF_b" will be used in the circuit. To specify, for example, three possible genetic locations, one would use the following structure:


```
{
    "collection": "genetic_locations",
    "symbol": "L1",
    "locus": 4196314,
    "sequence": "NCBI U00096.3"
},
{
    "collection": "genetic_locations",
    "symbol": "L2",
    "locus": 4506858,
    "sequence": "NCBI U00096.3"
},
{
    "collection": "genetic_locations",
    "symbol": "L3",
```


The "symbol" fields specify a marker to be used in the rules for locations of the gates or their sub-devices. To determine that each transcriptional unit of a split gate should occur at a different site (either in "L1" or one of "L2" and "L3", one would write:

```
{
    "collection": "circuit_rules",
    "rules": [
        {
            "function": "AND",
            "rules": [
                {
                    "function": "OR",
                    "rules": [
                        {
                            "function": "AND",
                            "rules": [
                                "P1_PhlF_a BEFORE L2",
                                "P1_PhlF_b AFTER L2"
                            ]
                        },
                        {
                            "function": "AND",
                            "rules": [
                                "P1_PhlF_b BEFORE L2",
                                "P1_PhlF_a AFTER L2"
                            ]
                        }
                    ]
```


User constraints file (UCF) resources were centralized, standardized, and subjected to automatic, data-driven tests. This includes a UCF for the Voigt lab yeast gates.

In addition, all the canonical UCF resources supported by Cello v2 are now stored in a repository, located at https://github.com/CIDARLAB/Cello-UCF. JSON Schema files have also been prepared to describe the UCF format and validate specific files. All the UCFs in the repository are validated against the schemas via continuous integration on travis-ci.org.

3.4.1.6. Automatic RNA-seq profile generation

Previously, RNA-seq profile generation was semi-automatic and required some user input and manipulation of Python scripts. Cello2 can now generate an RNA-seq profile for a circuit design implemented in a supported library. A library is supported for RNA-seq profile generation if it includes ribozyme efficiency measurements and terminator strengths. Currently, only the E. coli version 1 library is kept. A sample RNA-seq profile is shown in Figure 98.

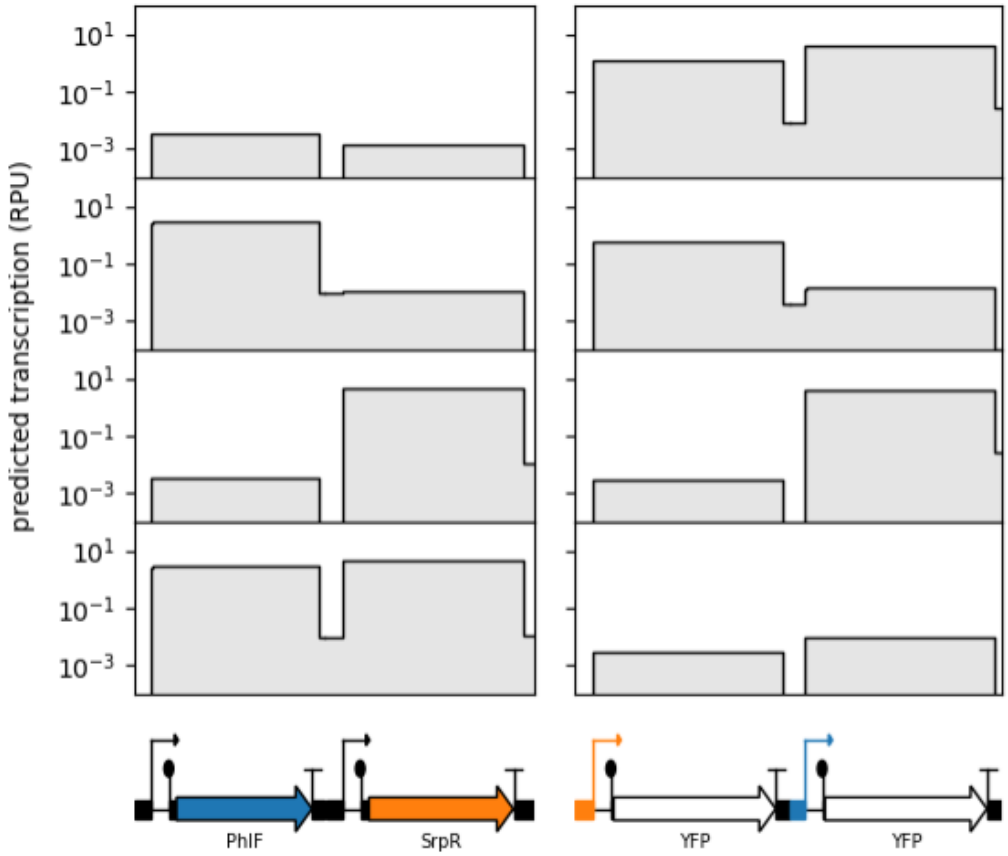


Figure 98. Cello2 generated a sample RNA-seq profile. The profile is for a NAND circuit. Each row of plots represents a particular input state. Each column represents a specific plasmid.

3.4.7 Generating UCF for E. coli Nissle

New User Constraint File (UCF), Input and Output files, and Cello-type DNA circuits were manually adapted to attend E. coli Nissel's implementations. We designed the Cello-input files for E. coli Nissel (UCF library, Input, and Output files) based on the experimental data provided by the MIT team. Note that Cello input files should be in JSON format. We used the same UCF library structure (REF: Park et al., 2020) for S. cerevisiae, where gates carry promoters in split configuration. Then, while the UCF libraries should contain metrics for multiple gates and the input-sensor files should contain characterized data for sensors, output-device files should have described data for actuators or signals used as final outputs of designed circuits.

The files' preparation started with the decision for gates, sensors, outputs, strains (i.e., chassis), and experimental conditions for genetic circuit design. Namely, it began collecting empirical metrics for each transcriptional unit (e.g., a NOT gate or an Input-sensor) and output signal (e.g., YFP fluorescence signal) in each experimental condition. These metrics, such as cell growth optical density and flow-cytometer fluorescence measurements, were obtained from Voigt's lab's experiments.

Finally, those new chassis files ('Eco2C1G4T1' set) were uploaded on the Cello 2.0 web app to design custom DNA circuits for E. coli Nissel. We used them to create and refine the DNA designs of example circuits with max. 4 gates (following subsections). We used these tasks and results to add content and revise the reviewers' points of the Manuscript we are writing for Nature Protocols. Finally, we again improved the existing troubleshooting table and fixed more software bugs (more details in the submission files).

3.4.1.8. Cello Design and Construction of logic circuits for the *E. coli* Nissle genome

We wanted to test the new UCF file created to design realistic DNA circuits that could be physically built. Our goal was to use the DAMP lab's automation facility and protocols to make them robust. In addition, we created three batches of DNA designs that could be tested, such as combinatorial logic circuits (AND Gate) for genome integration (Figure 99).

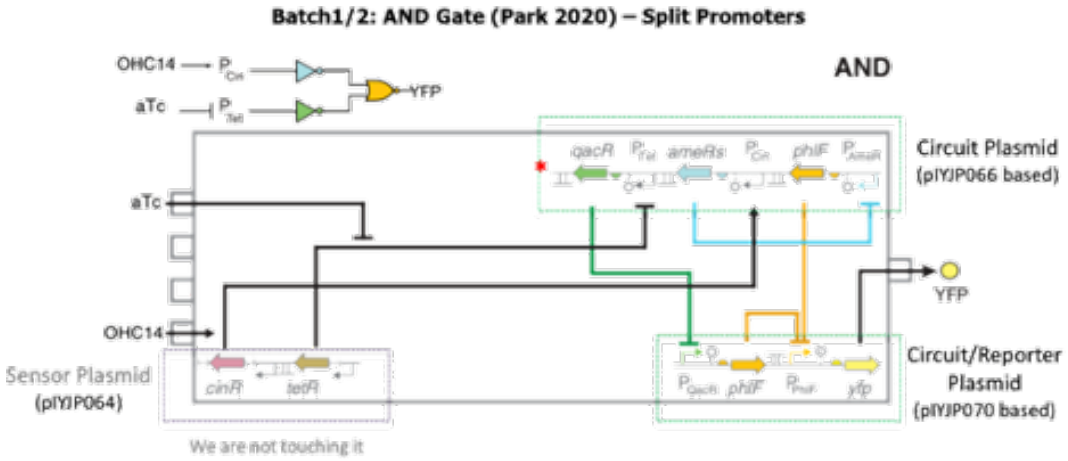


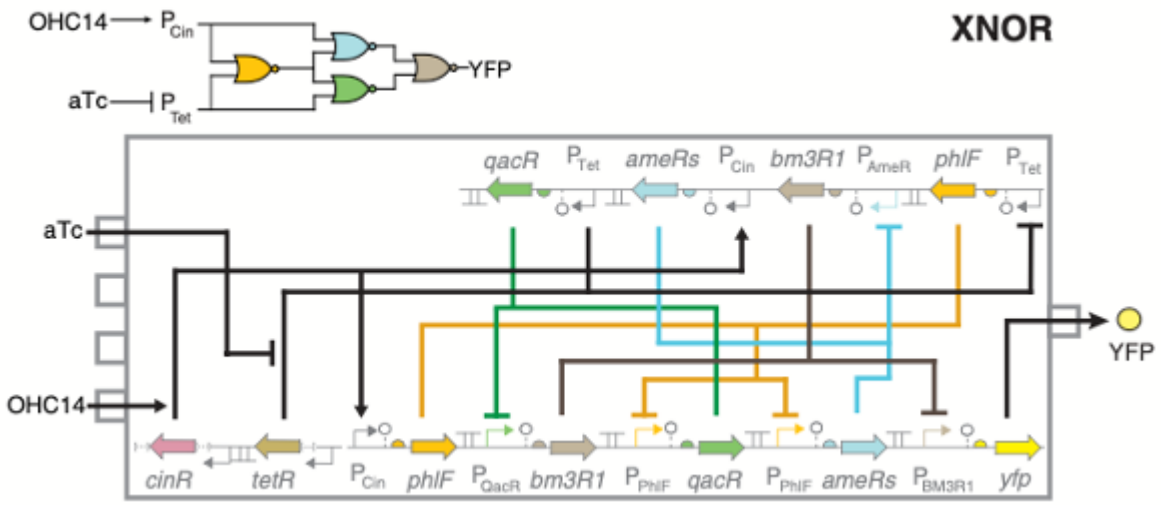
Figure 99. The schematic of the AND logic Gate (from Park et al., 2020) will receive MoClo-adapted parts before the rapid, modular assembly of genome-integration plasmids. The two circuit plasmids (pIYJP066 and piYJP070) have been designed to carry level-1 and level-2 fragments flanked with the appropriate MoClo scars. The circuit pIYJP066 (red asterisk) adaptation requires a new gate design from scratch since it is not present in the CIDAR MoClo library.

We also considered the construction of a sequential logic circuit (SR Latch) to check for our automated platform's potential to rapidly design, build, and test functional circuits (previously tested in plasmids) in the genome-integrated approach. In addition, we designed and built functional NOR gates (once integrated into plasmids for E. coli K-12) that could be used as building blocks of custom NIssle logic circuits.

Initially, we intended to include sequential logic gates as a candidate for genome integration into both E. coli wild type and E. coli Nissle chassis. However, Cello 2.0 cannot (up to date) support Verilog files of sequential logic circuits. Second, with extensive expertise and considering preliminary results of this project, for higher success rates with constructions in the lab, the MIT team strongly suggested that we work with gates carrying split-promoters instead of tandem's and implement them using genome-integrations instead of plasmids. This was particularly useful to test the expected results of batch-1 and -2 and served during the revisions of the Cello 2.0 manuscript (published in Nature Protocols, REF: Jones et al., 2022).

Finally, the DNA circuits designed for E. coli Nissel's implementations were AND Gate, 0X0, and the XNOR circuit (Figure 100) from the (REF: Park et al., 2020).

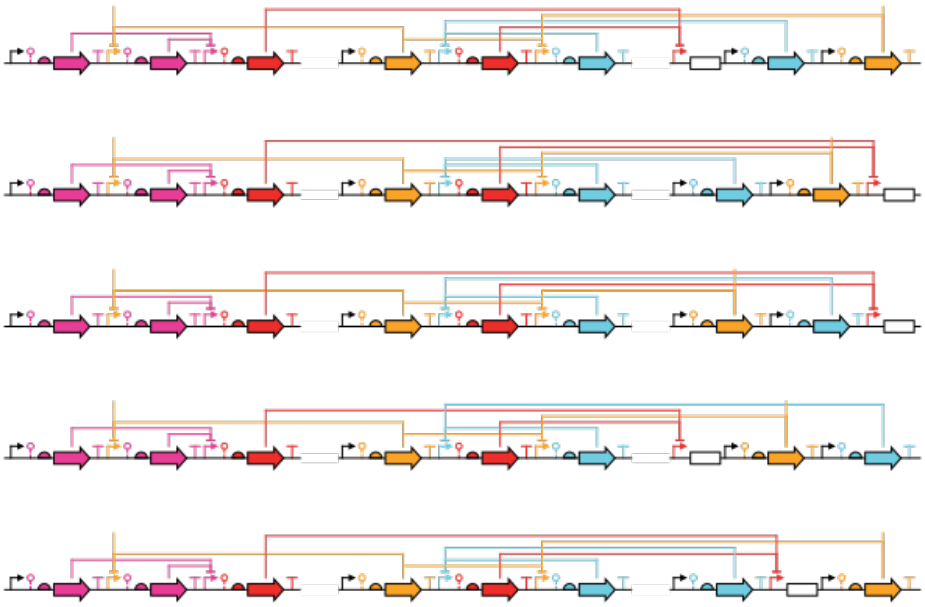
Genetic Circuit Schematic (from Park et al., 2020)



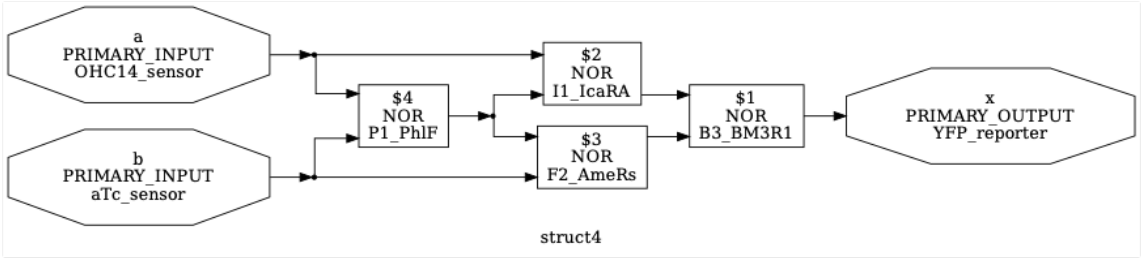
Verilog File

```
module struct4(output x, input a, b);
  wire w1, w2, w3, w4;
  nor (w1, a, b);
  nor (w2, w1, a);
  nor (w3, w1, b);
  nor (x, w2, w3);
endmodule
```

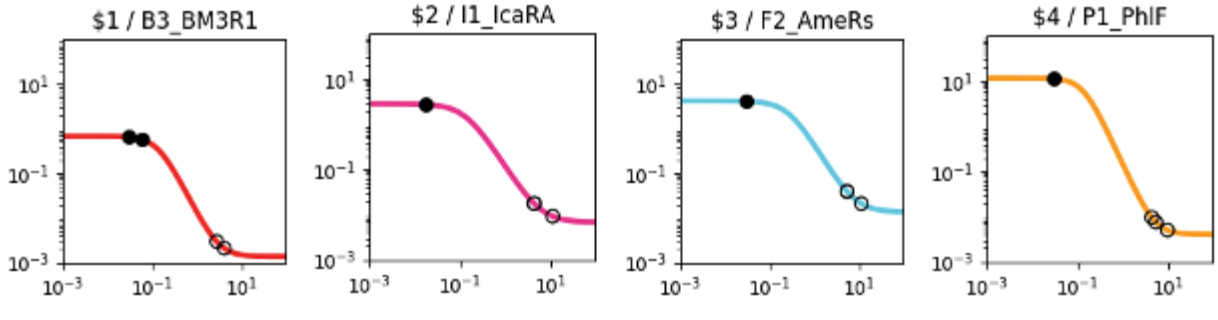
Possible DNA Designs (split-promoters in three landing pads)



DNA Logic Synthesis with Placing

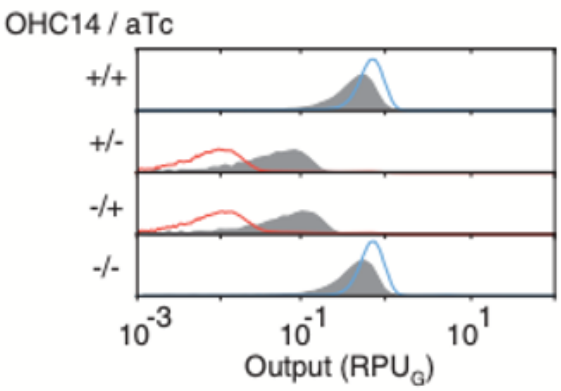


Predicted Response Curves



Performance Analysis

Expected Performance (Park et al., 2020)



Predicted Performance (From Cello)

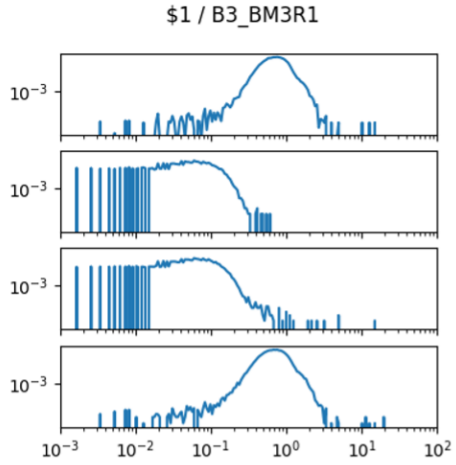
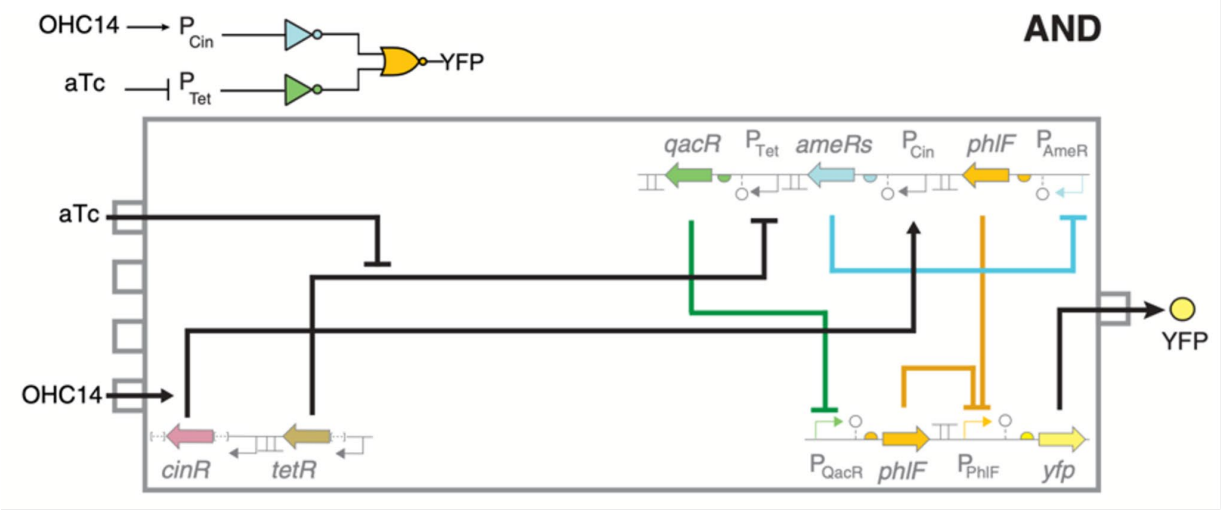


Figure 100. The circuit schematic to be implemented in E. coli Nissel, the Verilog file used in Cello, and the new set of input files for genome integration. Output files and predictions resulted from Cello. It presents the rationale and the result files for the design of the latter (XNOR Circuit), along with the information of input sensors (OHC14 and aTc) and set of gates (F2_AmeRs, I1_IcaRA, P1_PhIF, B3_BM3R1) selected by Cello for the desired performance (truth table).

Then, we used Nissle's UCF library in the Cello2.0 webtool to compile the genetic sequences for implementation in E. coli Nissle. Overall, we created the construction plan for an AND Circuit (Figure 101) intended for constructing and implementation into E. coli NIssle's genome (split-approach in landing pads of Eco2C1G4T1 E. coli Nissel). The input and output files of this design by Cello can be found in Figure 101.

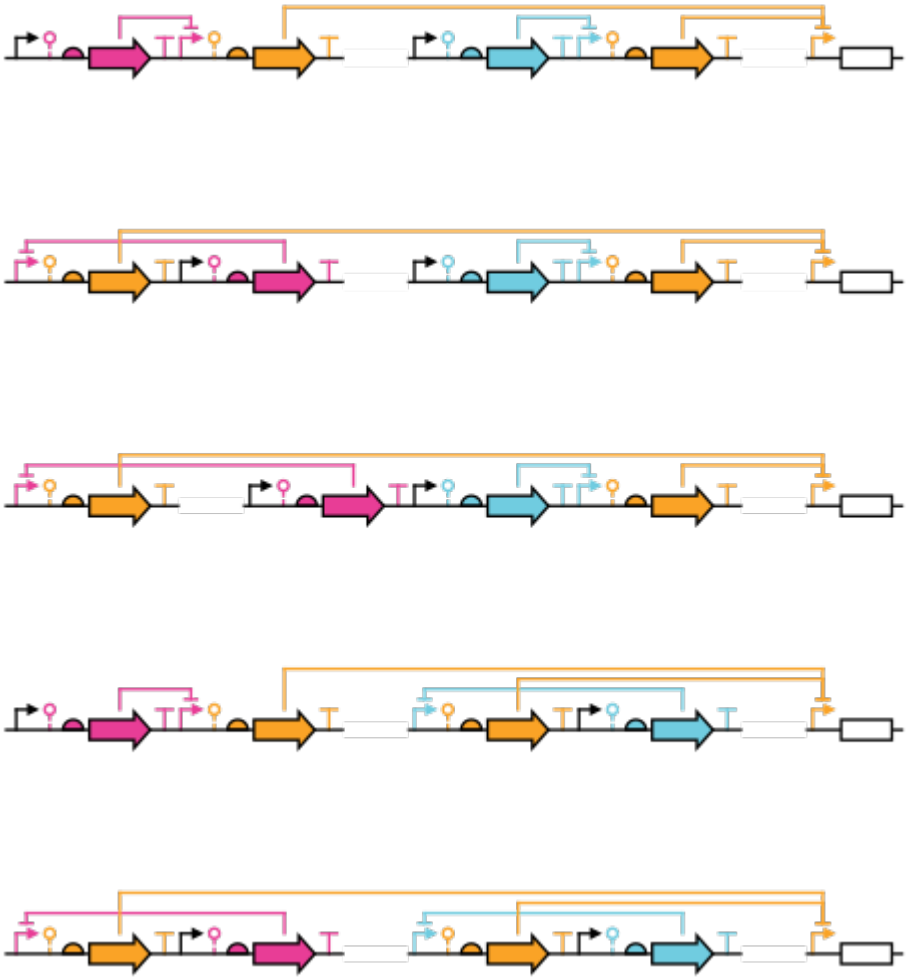
Genetic Circuit Schematic (from Park et al., 2020)



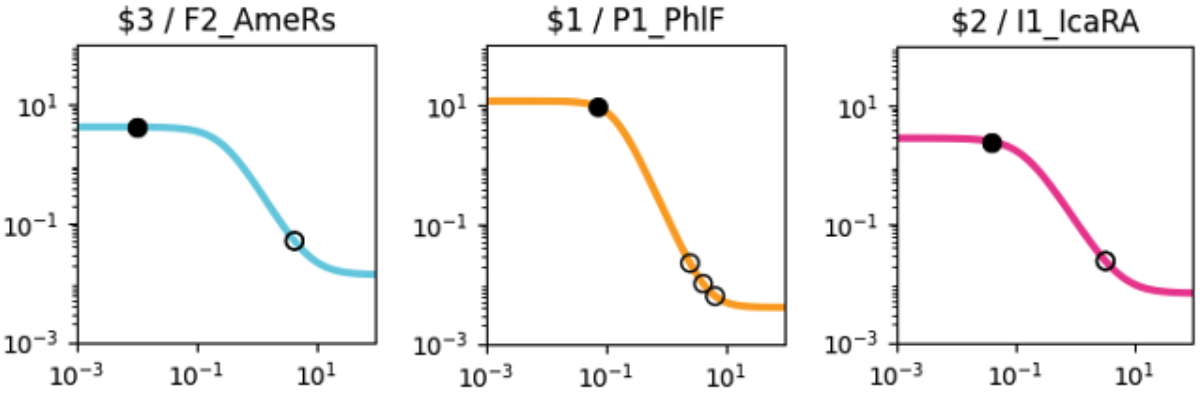
Verilog File

```
module struct5(output x, input a, b);
  wire w1, w2;
  not (w1, a);
  not (w2, b);
  nor (x, w1, w2);
endmodule
```

Possible DNA Designs (split-promoters in three landing pads)



Predicted Response Curves



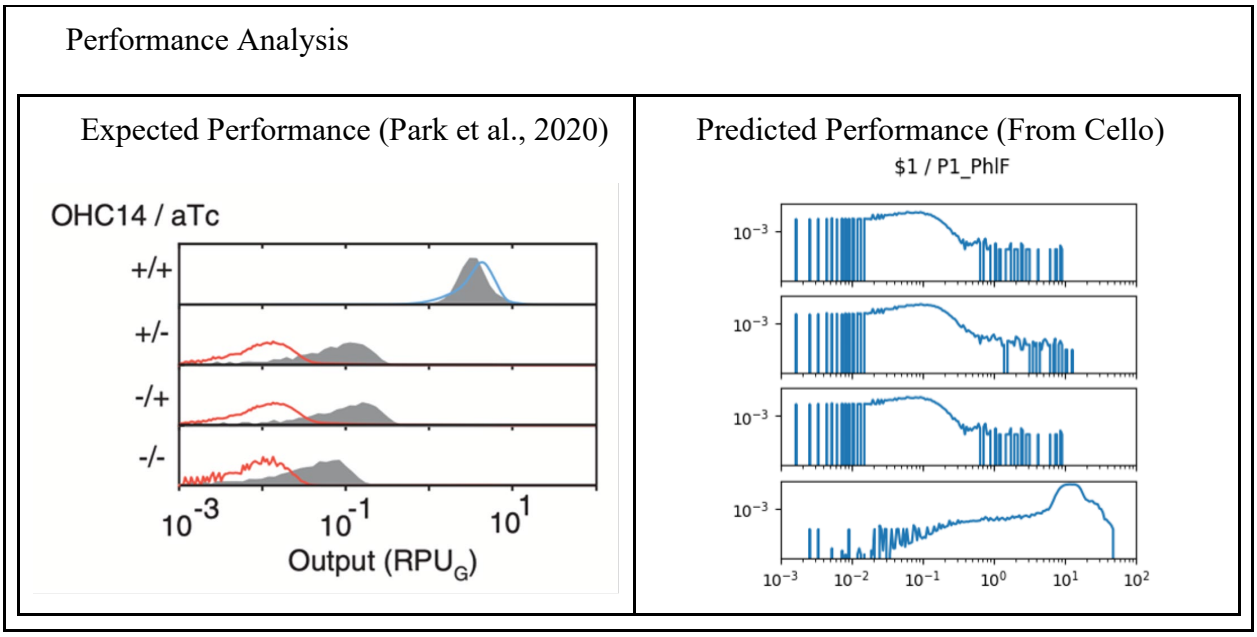


Figure 101. The AND circuit schematic to be implemented in E. coli Nissel, the Verilog file used in Cello, and the new set of input files for genome integration. Output files and predictions resulted from Cello.

Using the designed files generated by Cello, we prepared the necessary plan for constructing level-0, level-1, and sometimes level-2 MoClo parts and devices. Those DNA elements are used to assemble the required intermediate plasmids of future genome integrations (Figure 102). In the following steps, we constructed a series of MoClo parts and devices in an automated way using the DAMP lab's liquid handling robots (Opentron and Hamilton).

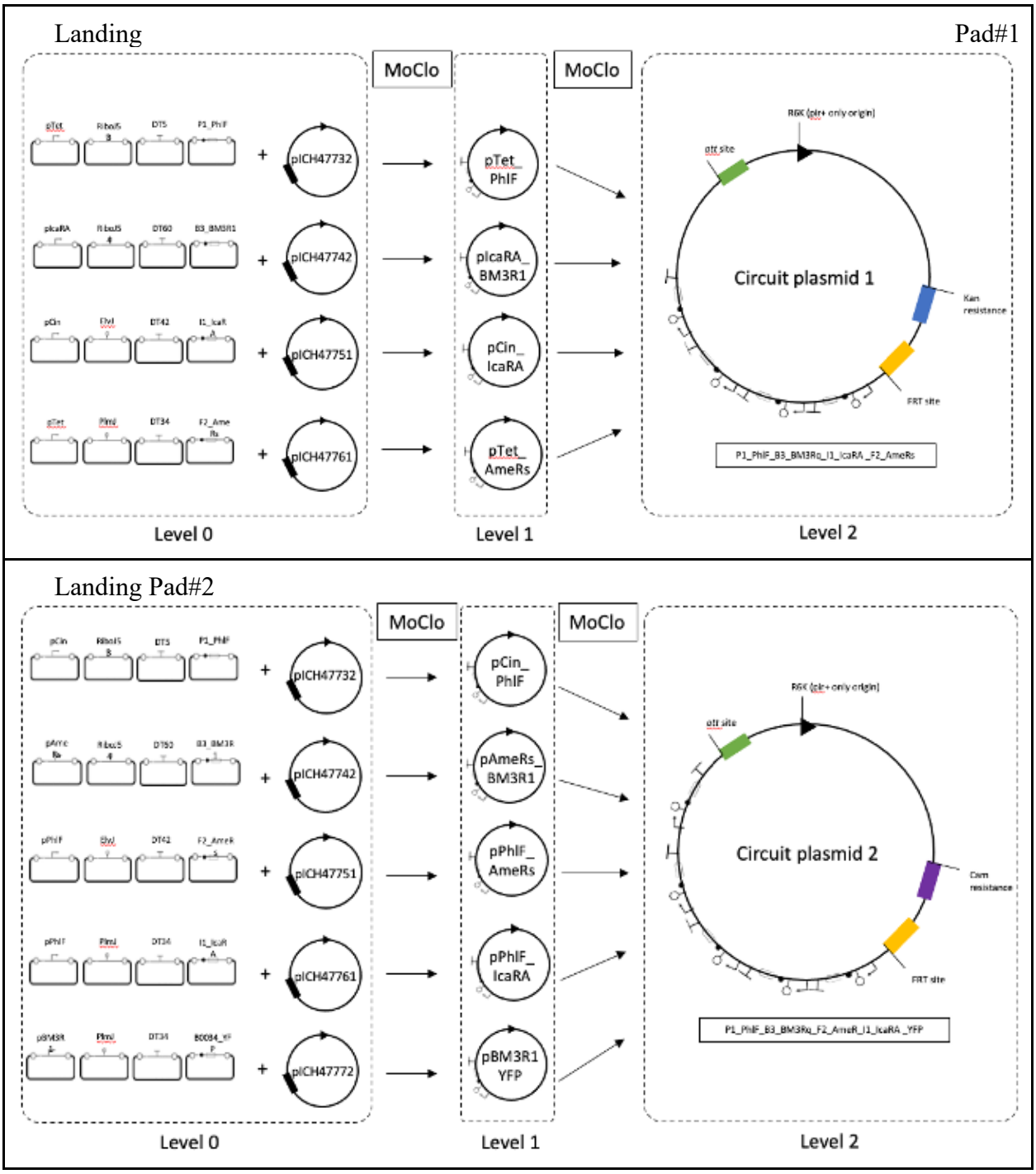
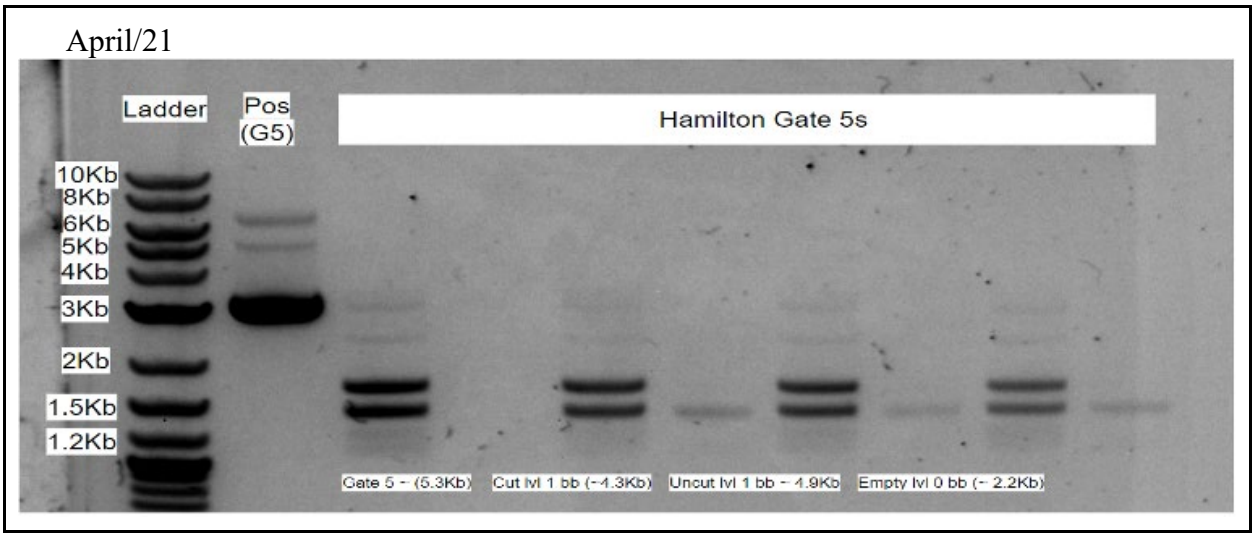


Figure 102. The two construction plans for the XNOR circuit presented in Figure 101. It considers two independent plasmid assemblies associated with their corresponding landing pad (1, 2, or 3) and MoClo parts for modular and automation assembly protocol, such as the MoClo assembly strategy.

Before implementing the plan entirely, we selected which automation approach yields the best results regarding part assembly success rate sequence matching. We used the two liquid handling robots to design and choose an optimal script and protocol to generate functional modular MoClo parts and devices.

We first troubleshot both robots for pipetting accuracy to guarantee that any problem we may face in the construction step could not be related to the robot's ability to generate reproducible results. The OT2-Opentron was ruled out from this test due to a lack of confidence in pipetting small volumes. In that, not all parts were being assembled/pipetted correctly (or accurately) in the individual wells, notice by the different total volume remaining in each well after sequential pipetting steps, a pipetting pattern of skipping wells after a run. Similarly, but with less significant differences, the Hamilton robot also presented pipetting issues. However, adjusted protocols and pipetting controls for this robot ensured a standard volume of 400uL in each stock tube prevented liquid level detecting failure from affecting pipetting (aspirating air bubbles, etc.). Moreover, DI-H2O solutions were proven to increase the differences in volumes in wells, thus reducing the method's accuracy, especially when using a multi-aliquot protocol. Instead, when the water was replaced with Qiagen Elution Buffer, all pipetting inaccuracies ceased, implying that the issue lies with Hamilton's software's liquid settings and correction curve.

We created an SOP for running Protocols in Hamilton from the tests and conclusions (Figure 103). We determined a standard volume (250uL and up) that could work without LLD (low-level detection) sensor failures and no consequence on pipetting accuracy. All elution buffer volumes, even those below the LLD failure threshold, consistently pipetted accurately with minimal error (>4 runs). Therefore, with future Hamilton experiments, all runs were defined in elution buffer with a minimum volume of 250uL in source tubes. As proof of concept, we present in the figure below the previous and latest results of a gate construction using the Hamilton robot before and after creating its SOP.



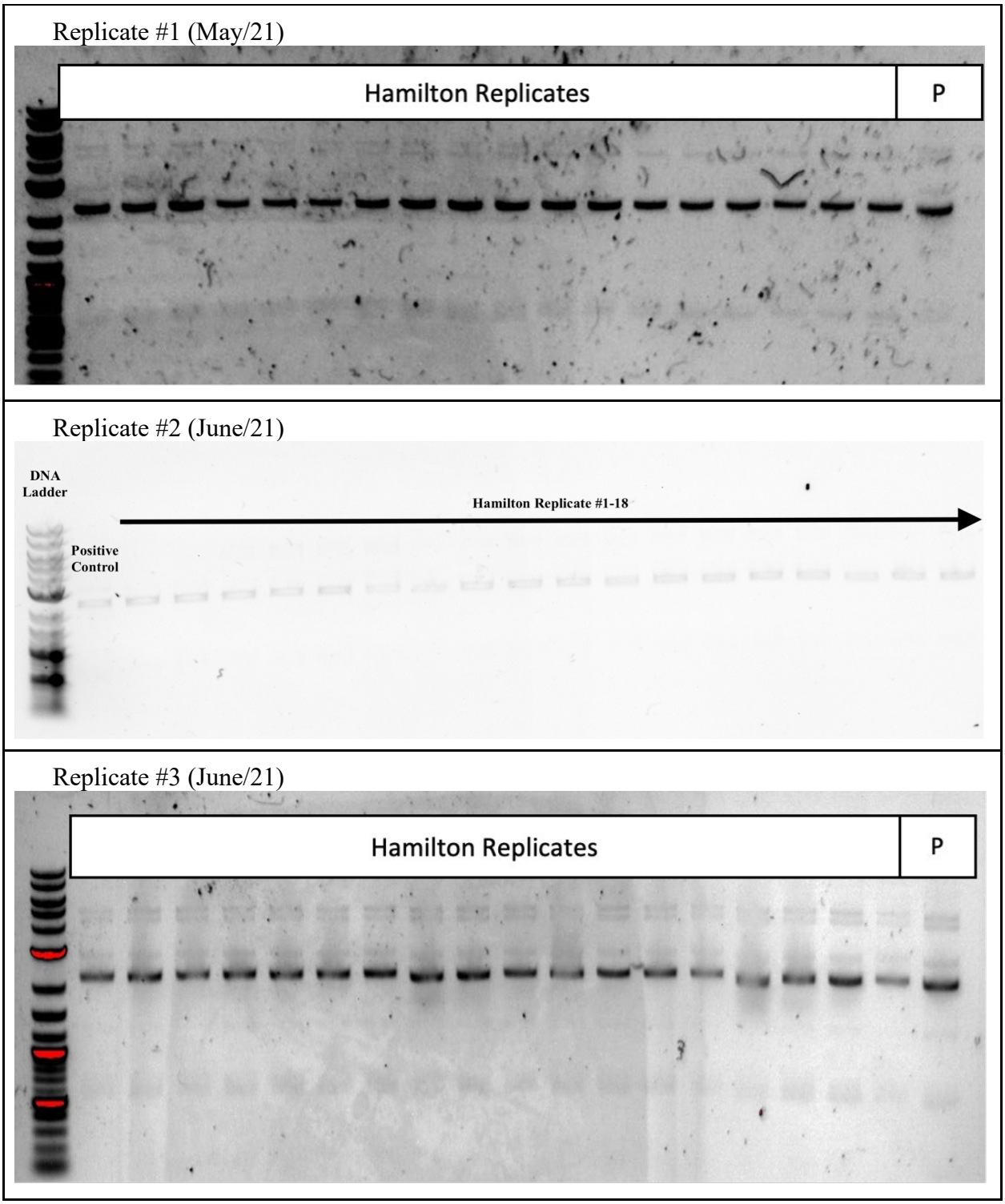


Figure 103. Results from electrophoresis gels presenting the molecular weight approximation of the construction of a NOR gate performed in the Hamilton robot and a MoClo reaction using a PCR device on the benchtop. We run three biological replicates of Hamilton's same NOR gate construction protocol. Results show a protocol with both 100% reproducibility and replicability.

Finally, due to changing staff members in the group and lack of knowledge transfer, unfortunately, the project was halted, and a new pipeline was created. Namely, we used a semi-automated construction workflow consisting of tasks partially done by a technician on the bench and the reaming using the robots (Figure 104). In addition, the automatic construction of elements into the transcription units and the final AND gate system follow the manual step to create a components library.

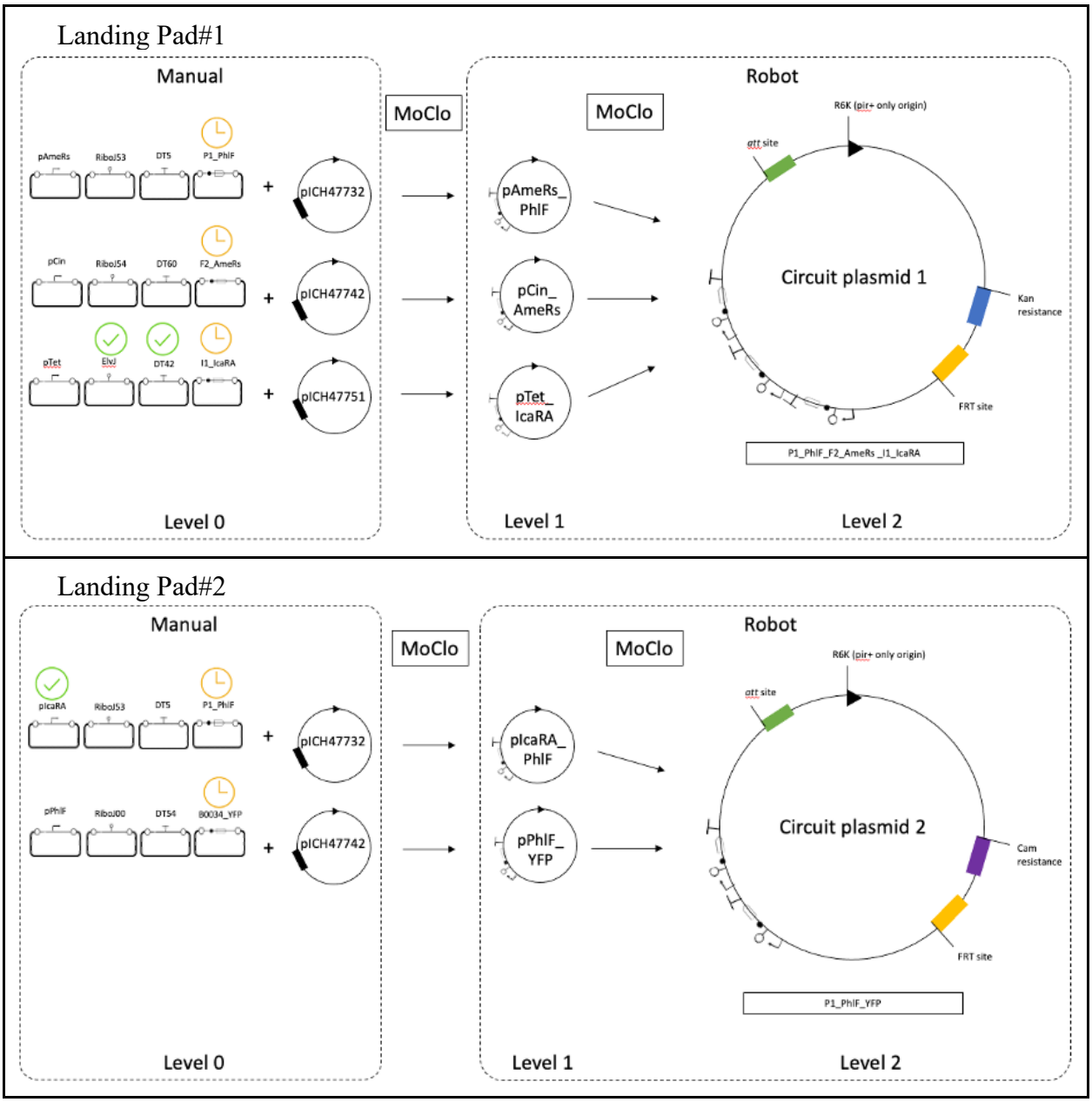


Figure 104. The two construction plans for the AND circuit presented in Figure 101. It considers two independent plasmid assemblies associated with their corresponding landing pad (#1 and #2). Modular parts are assembled manually or automatically using a liquid handling robot depending on the construction level of the MoClo assembly strategy.

As partial results, level-0 parts DT42+IcaRa+ElvJ and pAJT256-PTac-ElvJ-I1-IcaRA-DT42 have been successfully constructed. The construction of the remaining fragments of the library has initially failed for 5 CDS in Level-0 plasmids. The assembly attempts yielded (i) low correct CFU (1 colony in a plate) for F2_AmeRs and R1_PsrA; (ii) no correct CFU for P1_PhlF; and (iii) correct CFU for B0034_YFP and I1_IcaR, but no correct sequencing confirmed. After completing the

manual construction of the level-0 library, the automated construction of levels-1 and -2 will be performed using a liquid handling robot.

3.4.1.9. Software Release, Writing Cello's Manuscript, and its Deployment to cellocad.org

At the end of the project, after many improvements listed above, Cello 2.0 was finally released and deployed to cellocad.org in March/2022. It comes with 5 UCFs: the original E. coli plasmid gates, E. coli plasmid-integrated gates with a tandem promoter model, genome integrated E. coli gates, genome integrated S. cerevisiae gates, and genome integrated B. thetaiotamicron gates. In addition, Cello has the following features:

Improved Verilog support: Cello 2.0 supports the Verilog 2005 specification almost in its entirety. This allows users to utilize control structures that will be important for describing sequential logic circuits (circuits with memory or state).

Gate architecture specification: Cello 1.0 was hardwired for NOR gates with two input promoters in series. The new UCF structure specification lets users define gate architecture based on a collection of part types. This supports the yeast and other gate systems that require gates with two inputs to be split into two transcriptional units (Figure 105).

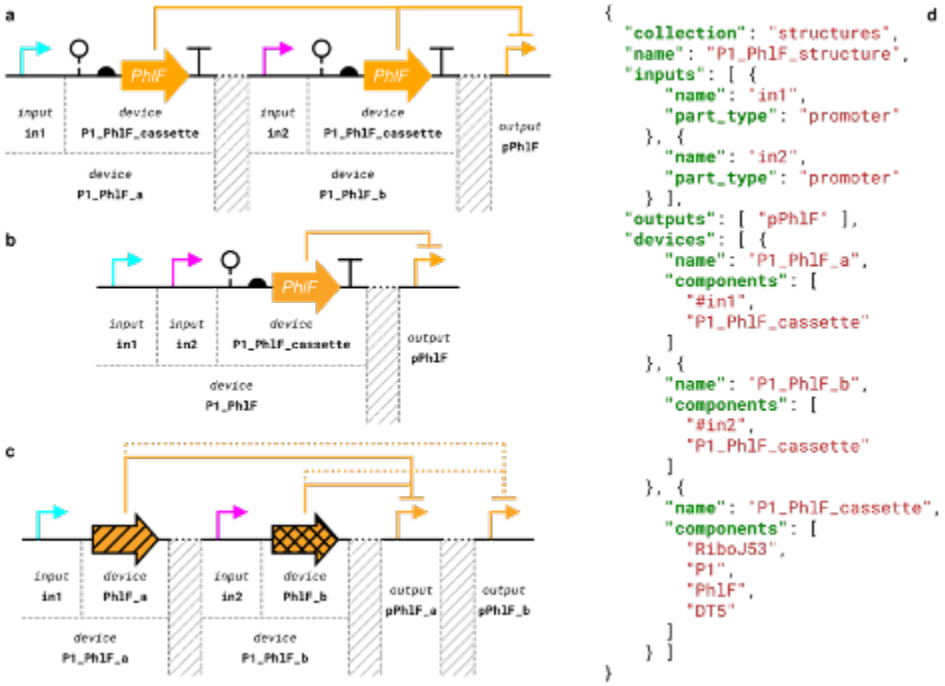


Figure 105. Three gate structures and a JSON implementation. (a) implementation of a gate with split transcriptional units. (b) Tandem promoters. (c) Split transcriptional units with multiple variant output promoters and genes. (d) The performance of the Gate in a.

The genetic_locations collection allows the specification of integration sites referred to in the UCF rules by name. According to regulations, gates and components can be distributed across different locations. For example, in Cello 1.0, all gates were lumped into a single location by default.

Gate model flexibility: Previously, the response functions were defined to have a single mathematical form and associated fit parameters. However, different gate architectures and

different classes of regulators require alternative models. The new UCF model structure allows a user to define arbitrary functions for each Gate.

Expansion of rule sets for DNA mapping: Cello uses rules, defined in the UCF, to map the circuit design to a linear DNA sequence, for example, the order and orientation of repressor genes on a plasmid. These rules were expanded to encompass the needs for gate designs in new specie and their encoding and the genome. For example, a new object type in the UCF defines the host genome, either by a link to a sequence in the NCBI database or by completely embedding the host genome sequence in the UCF and the insert locations that can be referenced symbolically in the Eugene rule sets.

SBOL 2.3 and SynBioHub support: Cello 2.0 provides an SBOL 2.3 description of the circuit as an output. This output optionally includes the molecular interactions that occur between gates. This information is not included in the UCF, as it is not needed for a gate assignment. Instead, the molecular interactions are mined from SynBioHub. The SBOL output can also be uploaded to SynBioHub from the GUI, and a user can optionally load a library from SynBioHub instead of selecting a UCF.

3.4.8 Developing Puppeteer, a tool for automating DNA assembly protocols

The SAIL team at Boston University designed and implemented new graph data structures for Puppeteer, a DNA assembly software tool. We created the graphs as part of a long-term effort to build Puppeteer's capacity to conduct optimization tasks and other computational analyses. The graph stores data about parts, such as DNA sequences and overhangs, and parent relationships to other parts. The parent relationships indicate which parts constitute composite parts, such as transcriptional units or plasmids.

The SAIL team also wrote an API for computing over the data structure and navigating and retrieving data from protocol graphs. The protocol graph adds operation steps to an underlying assembly graph structure and can generate overhang sequences. The operation steps and overhangs depend on a user's requested build method, such as "modular cloning" and instruction type. The instruction type options (Aquarium Plan, Autoprotocol, or Tecan robot directions) define Puppeteer's output format. Next, the new data structures impacted the entire Puppeteer codebase, from parsing circuit designs in SBOL to creating assembly directions. Finally, SAIL engineers refactored Puppeteer to use the assembly and protocol graph structures, wrote new unit tests to promote software robustness, and linked it to Aquarium, a protocol automation tool.

A central internal data structure within Puppeteer has been updated from a "protocol" graph to an "operations" graph. These data structures store all the information necessary that turns an assembly graph (assembly and lab agnostic) into a lab and assembly method-specific instructions. The new "operations" graph is a directed acyclic graph where each node is an operation, and each edge indicates the next operation in the plan (Figure 106). This structure is more flexible than the protocol graph and addresses the limitations above.

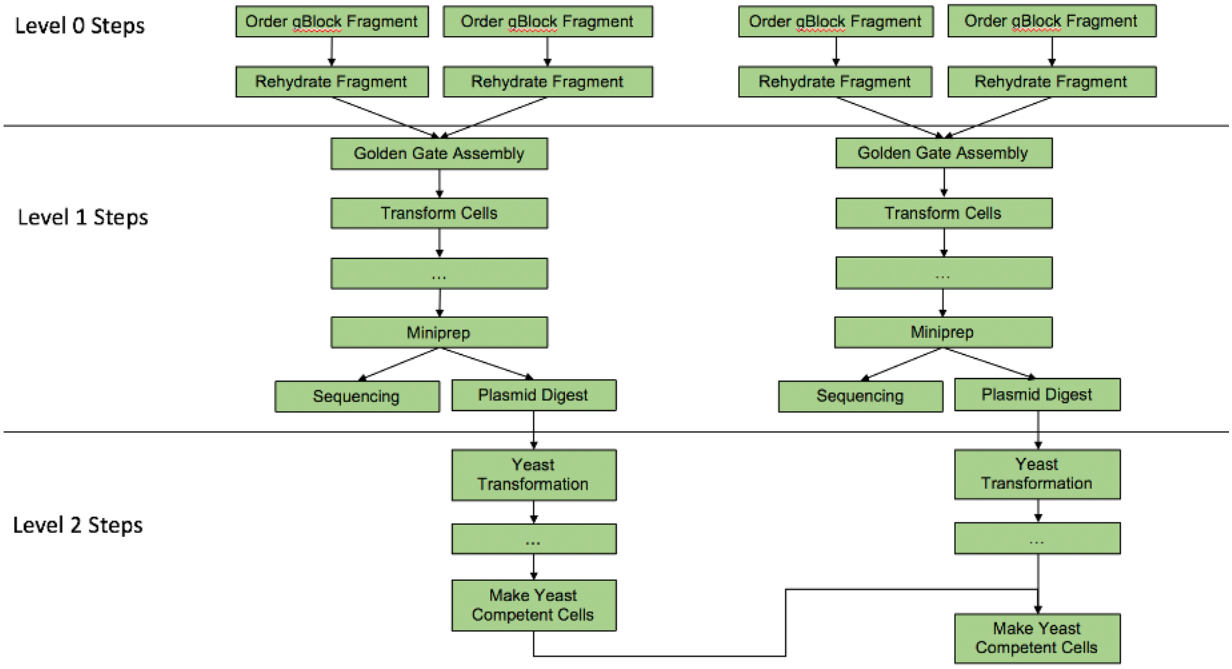


Figure 106. Visualization of an Operations Graph within Puppeteer. Each operation (green box) stores input parts, output parts, and a set of previous operations and subsequent operations.

One of the goals was to conduct an end-to-end test eventually. A researcher uses Puppeteer to retrieve a Cello circuit design from SynBioHub, create an assembly plan, and send the request to the Aquarium lab. SAIL engineers updated the Puppeteer codebase to process the SBOL 2.2 UW yeast gate circuit with this aim in mind. Implementing the integrated DNA assembly software pipeline using the UW yeast gates circuit is particularly relevant for the SD2 yeast gates challenge problem (Figure 107).

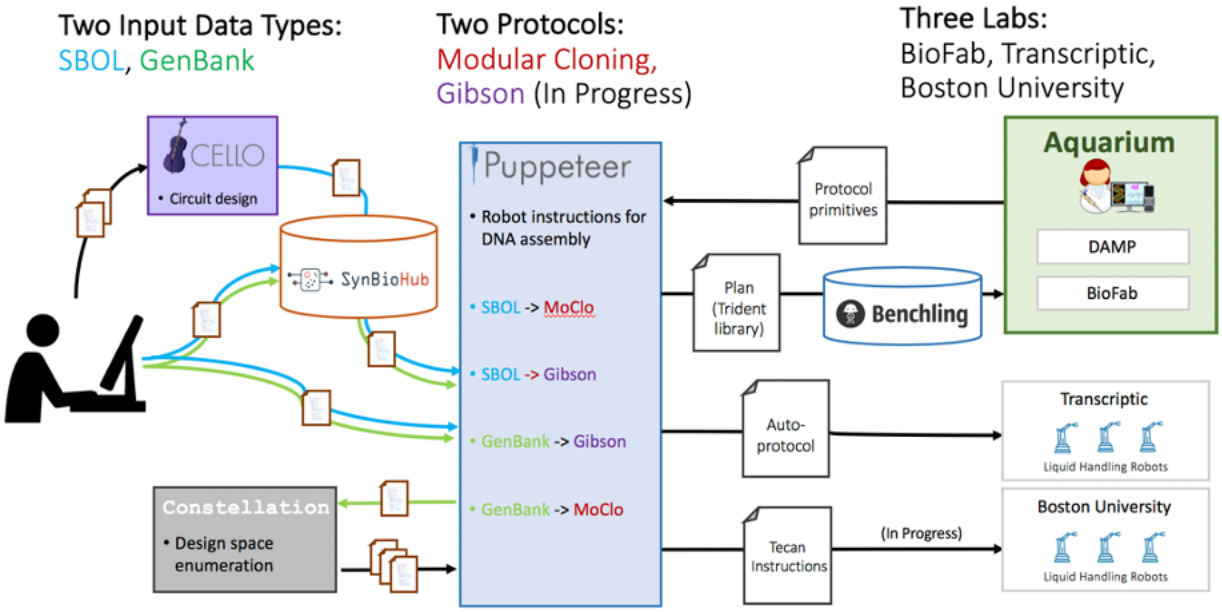


Figure 107. An architecture diagram shows Puppeteer's workflow and integrations with other software tools. The design process starts with a user creating a circuit design in Cello and uploading the SBOL output to SynBioHub. Puppeteer retrieves and parses the SBOL. Then, a user can select from a list of available cloning methods, including Modular Cloning and Gibson, and an output type, which can be Aquarium operations (BU DampLab or UW Biofab) or liquid handling robot instructions (Autoprotocol or Tecan). DNA sequence data is also uploaded to Benchling for Aquarium operations.

The SAIL team created a new, more robust server for Puppeteer's python modules, which handle calls to the Trident API and Autoprotocol conversion. In addition, the new NGINX and Gunicorn servers offer greater scalability and security. In addition, SAIL created scripts to deploy these servers on the Massachusetts Open Cloud (MOC) and docker containers.

Puppeteer has always been intended as a web-based tool and thus requires a user-friendly and intuitive interface. The previous version of the UI was adopted from an existing project at SAIL and not customized for genetic assembly plans. SAIL team also updated Puppeteer's user interface. The new UI, implemented this quarter, is designed explicitly. The design was guided and reviewed by a UI/UX expert from Redhat and contained clearer instructions and a more intuitive workflow for biologists. Users can now view and download well plate visualizations that depict part names and well placements. SAIL implemented this new functionality based on feedback from BU lab technicians, one of whom will use a well plate visualization to prepare a source plate for an upcoming test with Transcriptic. The team has also designed mock-ups for a complete user-interface overhaul of Puppeteer.

The accumulation of work on both Puppeteer and Cello in the past year has allowed us to automate the generation and planning of genetic circuits. In this project, Cello was used to design a 3-node circuit in yeast that exhibits bistable behavior, and Puppeteer translated this design into an Aquarium plan at the UW Biofab. The circuit was constructed, and initial sequencing results showed successful cloning and transformation of three individual gates in E.coli. The construction of this circuit demonstrated a successful proof-of-concept of the Cello-Puppeteer-Aquarium

automation pipeline. In the future, Puppeteer will generate plans that include PCR operations using preexisting parts at Biofab, decreasing the overall cost.

3.4.9 Bootstrapping a lab with Aquarium

University of Washington (UW) BIOFAB (Biofoundry) has almost 10,000 experimental plans stored in the Aquarium database. Each plan encodes information such as status, user, operations involved, the status of each operation (e.g., error), samples involved, and dates (created, updated). When an operation fails, it is marked as an "error" with no additional explanation. Therefore, parsing the plans alone cannot tell us whether it is a technician error (e.g., dropping the test tube), bad sample (e.g., contamination), defective protocol, or other reason.

The user then determines what to do next – repeat the experiment exactly as it is, repeats with a different sample or protocol, etc. The ability to determine the best course of action after an error comes with domain knowledge and familiarity with the lab and its protocols. This information presumably is encoded in the next set of plans they create to correct the error. In this report, we held discussions with BIOFAB around developing software to automate the process of laboratory error detection and correction and formalize the types of errors.

We obtained a copy of all production data from the BIOFAB's instance of Aquarium and determined the subset of data needed to cluster plans by similarity. We wanted to develop a strategy to best cluster these operations and find similar plans by using parameters such as the order of operations, the category of each operation (e.g., cloning, tissue culture), user, and dates. The UW BIOFAB created a Terrarium tool, which automates the planning of Aquarium experiments based on past plan data. However, a solution does not exist currently to automatically generate plans when Aquarium is newly introduced to a lab, which we've dubbed the "bootstrapping problem."

Therefore, the SAIL team at Boston University proposed an idea to solve the bootstrapping problem by abstracting plan operations into a set of "signatures," which include properties of the input and output items. The goal is to explore and use probability models (e.g., Markov model) and deep learning (e.g., neural network/LSTM) to learn generic representations of the planning process. For example, we wanted to split the ~10,000 plans from BIOFAB into two disjoint sets to mimic two labs to have a train and test set for our model.

We used a modified Karger's algorithm that chooses an edge at random and merges the "left" and "right" vertices until there are only two vertices with n number of edges between the two, where n is estimated to be twice the number of the minimal cut in the worst-case scenario. We also weighted the edges with the number of times the connection occurs in real experimental plans. We ran this algorithm multiple times over a cardinality range, then plotted the size of the cuts as the cardinality changes and visually picked a reasonable split (Figure 108).

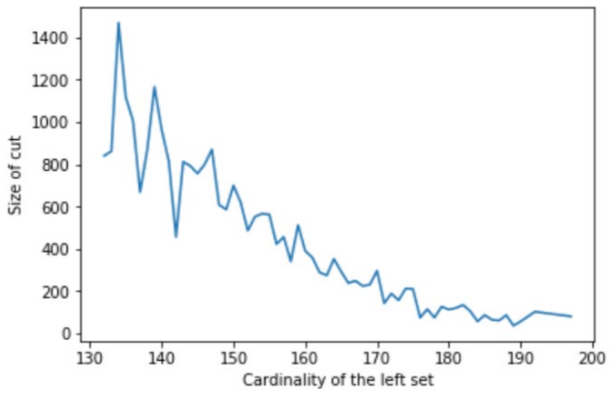


Figure 108. Graph shows the number of cuts vs. cardinality of one of the sets using the modified Karger's algorithm. Out of 264 total operation types, cardinality 177 means that the left set will contain 67% of all operation types, and the right retains 33%.

In the context of deep learning, we considered implementing an LSTM which can learn to generalize plans by utilizing the contextual data of the signatures, and alternatively, graph neural networks (e.g., graph CNN, generalized adversarial network, etc.) that makes use of the implicit graph structure utilized in Aquarium plans. We opted to table the deep learning approach favoring a modified Steiner tree algorithm currently implemented in Terrarium. However, the current algorithm has several shortcomings, including the lack of ability to produce multiple outputs and handle missing AllowableFieldTypes. Our proposed solution is to implement a more generalized solution for the problem of plan synthesis with various goals and optimized handling of missing inputs. An input agnostic solution means we can directly run it over plans generated with signatures instead of AllowableFieldTypes.

3.4.10 Launching a high throughput SARS-CoV2 testing facility

Plans began in May at Boston University to launch a high throughput SARS-CoV2 testing facility, and SD2 members from BU are also heavily involved in this COVID testing effort. We collaborate with UW and Duke by contributing the Aquarium code, CDC protocols, and BU's protocols. CDC protocols that we have contributed include positive control aliquoting and sample aliquoting. We have so far completed the Aquarium code for BU protocols for positive control aliquoting, sample aliquoting, and RNA extraction. In addition, we have plans to create Aquarium protocols for the RT-PCR process, including RT-PCR prep, RT-PCR assay, and RT-PCR analysis. In addition, we are sharing information learned from the launch of the test facility to the extent possible.

3.4.11 Collaborating with the development of VisBOL

In addition to Puppeteer developments, SAIL engineers collaborated with Chris Myers' team (University of Colorado, Boulder) to make significant contributions to VisBOL, a software tool that SynBioHub uses to visualize genetic parts and designs. They closed over a dozen GitHub issues on the VisBOL codebase, bringing the tool closer to SBOL 2.0 compatibility. In total, SAIL has helped to close 26 issues on Github since our involvement in the development of VisBOL

began in April 2018. Once deployed to SynBioHub, our latest enhancements will greatly aid SD2 performers in visualizing genetic circuits. Example VisBOL improvements include rendering non-DNA glyphs, showing interactions between non-DNA components, and representing composite glyphs (nested circuit parts). Next, Glyphs were added that represent RNA, DNA, protein, and complexes (Figure 109).

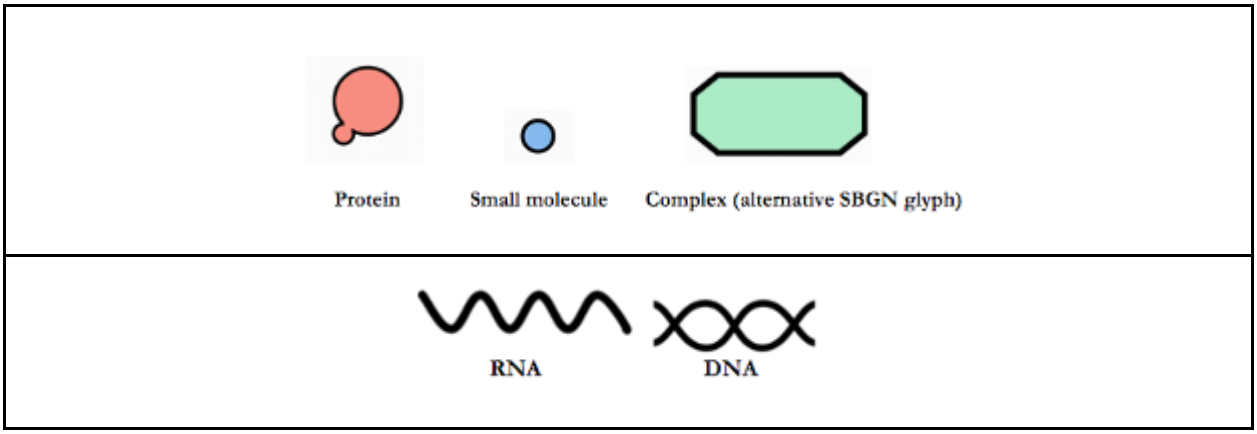


Figure 109. New glyphs added to VisBOL

Initial support for detecting molecule and DNA interactions was completed, including visualization of repression, activation, degradation, non-covalent binding (between molecule and protein), and production (of protein).

Next, a more efficient and accurate method for composite structure detection has been identified and implemented. Our previous approach also relied on display names, leading to erroneous detection of composite parts when names were not unique. Namely, we updated the data structure for storing rendering relevant information for interactions and modified the visualization (Figure 110).

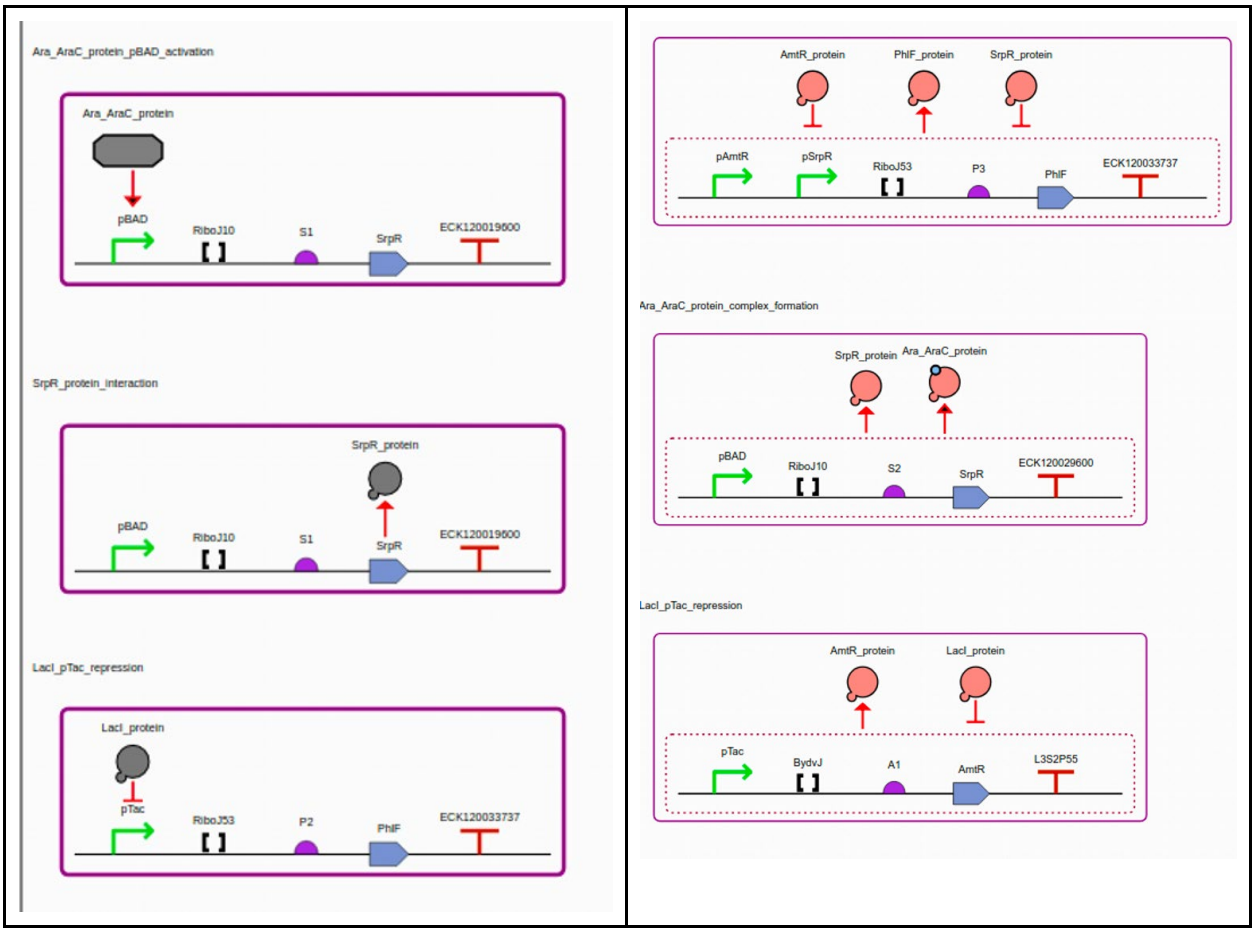


Figure 110. Previous vs. a current rendering of interactions in VisBOL. On the left is the previous implementation, where the non-DNA component sat directly on top of the part of the DNA where the interaction occurred. Unfortunately, the assumptions made in this approach often led to erroneous renderings and cannot currently be fixed due to SBOL limitations. On the right is the current implementation, where the non-DNA components are shown above the entire circuit.

Previously, interactions between non-DNA components and parts of the DNA were matched by the display name. The non-DNA component was rendered directly on top of the DNA part where the interaction occurred. However, there are no formal constraints on display names, and this approach often led to erroneous rendering when display names were not the same. However, due to the limitations in the SBOL structure, we cannot currently fix this issue and maintain the same visualization. Now, non-DNA components that have a functional relationship with a part of a DNA strand can be identified and rendered in one single diagram on top of the corresponding circuit instead of a single part. This approach is also more efficient as it decreases the total number of renderings.

3.5 Conclusion

One of the goals of our group was to conduct an end-to-end test in which a researcher uses Puppeteer to retrieve a Cello circuit design from SynBioHub, create an assembly plan, and send the request to the Aquarium lab. SAIL engineers updated the Puppeteer codebase to process the SBOL 2.2 UW yeast gate circuit with this aim in mind. Implementing the integrated DNA assembly software pipeline using the UW yeast gates circuit was particularly relevant for the SD2 yeast gates challenge problem. In the end, a demo and presentation were prepared to show the integrated solution. The demo showed Puppeteer's enhancements and demonstrated its ability to: (i) process multiple input types - SBOLs retrieved from SBH or dragged into the Puppeteer UI, as well as GenBank files; (ii) implement Modular Cloning or Gibson DNA assembly protocol; (iii) retrieve protocol primitives from the UW BioFab or BU DAMPlab instances of Aquarium and create plans for either lab; (iv) support E. coli and yeast operations; (v) support PCR operations with auto-generated forward and reverse primers; (vi) generate assembly plans in Autoprotocol using acoustic transfer operations; (vii) produce source and destination plate visualizations for DNA assembly plans; and (viii) upload sequencing data onto Benchling. Another goal was to improve the CelloV2 software tool. As a result, we published the Manuscript in Nature Protocols. All authors of the Manuscript are SD2 participants: Timothy Jones, Samuel Oliveira, Chris Myers, Chris Voigt, and Doug Densmore. The Manuscript is the most comprehensive description of the new UCF format available now. The most significant additions to the UCF format are: (i) the models' collection; (ii) the structures collection; (iii) the functions collection; and (iv) the expansion of the genetic locations and rules collections.

4 SUBGROUP 4: MYERS GROUP (UNIVERSITY OF COLORADO, BOULDER)

4.1 Summary

Facilitating genetic circuit and parts data sharing is critical for any collaborative work, especially for a big program like SD2 where different research groups have different design tools. Therefore, it was decided that the Synthetic Biology Open Language (SBOL) data standard would be used for the representation of biological designs and the electronic exchange of information on the structural and functional aspects of biological designs. Furthermore, the online repository SynBioHub was to be used as an online repository.

4.2 Introduction

Some of the tools and methods developed in this program include, but are not limited to:

4.2.1 SynBioHub

SynBioHub (J. A. McLaughlin et al., ACS Synth. Biol., 2018) is a design repository for people designing biological constructs. It enables DNA and protein designs to be uploaded, then provides a shareable link to allow others to view them. SynBioHub also facilitates searching for information about existing useful parts and designs by combining data from a variety of sources.

4.2.2 Synthetic Biology Open Language (SBOL)

SBOL (M. Galdzicki et al., Nat. Biotechnol., 2014) is a free and open-source standard for the representation of biological designs. The SBOL standard was developed by the synthetic biology community to create a standardized format for the electronic exchange of information on the structural and functional aspects of biological designs.

4.2.3 VisBOL

VisBOL (J. A. McLaughlin et al., ACS Synth. Biol., 2016) is a JavaScript software library to visualize DNA features from SBOL2 documents using the SBOL Visual standard. A Web interface is also provided to enable end-user access to generate diagrams for presentations and publications.

4.2.4 Dynamic modeling

Dynamic modeling allows for the prediction of output states *between* steady-states. This is critical for the correct prediction of dynamic behavior of the different parts and circuits designed in this program. We used new dynamic models to predict the robustness and glitch probabilities of various genetic circuits and parts designed throughout the SD2 program.

4.3 Methods, Assumptions, and Procedures

The facilitation of data standards and data sharing of genetic circuit and part design was developed using SynBioHub, SBOL, and VisBOL. Different tools and scripts were also created to help with the migration from a wide range of data formats into SBOL and VisBOL, in order to upload, store and share using SynBioHub. Furthermore, several groups and workshops were created and to maintain and help different challenge problem groups migrate their results into the sharable data standards used for this program, and get direct feedback on problems/opinions/critiques/suggestions from users in the program.

The dynamic simulation and analysis of robustness/glitch propensities of the different designs from various challenge problems was done using SBOL and iBioSim (C. J. Myers et al., Bioinformatics, 2009).

4.4 Results and Discussion

4.4.1 SBOL development/support

A critical component of the DARPA SD2 project is the use of the *Synthetic Biology Open Language* (SBOL) and SynBioHub (see below) for the encoding and storage of design and experiment metadata. Throughout the course of this project, numerous enhancements to SBOL and SynBioHub were executed to support these efforts. Ultimately, these efforts led to the development of SBOL Version 3, a substantial redesign of the SBOL data standard to improve representation efficiency. The SD2 project also exposed numerous limitations in the SynBioHub user interface as well as scalability issues that are not easily addressed in the current software architecture used by SynBioHub. This lead to a complete redesign of SynBioHub that provides support for the following topics:

- A completely new and interactive graphical user interface that was based on the requirements of synthetic biologists from academic, government, and industrial research labs.
- Enhanced authentication and security capabilities enabling its use on government, national laboratory, and commercial servers.
- A more robust backend that supports alternative database storage systems including commercially available data stores.
- Native support for the SBOL Version 3 data model.

Initial development and prototyping of this new version of SynBioHub conduced to the development of this project, SBOL Version 2.3. This version of SBOL included several new features including ones that were motivated by the SD2 project. These include a new Measure class to allow the specification of characterization parameters, new experiment and experimental data classes, and methods to references partial sub-sequences of parts. PI Myers led the finalization of this version during the Harmony Workshop at Caltech in March. This work included updating the specification, the software libraries, and the generation of test cases. These changes will allowed us to recreate our Cello part libraries with parameters in a standard way using

the new Measure class, so they could become more readily exchangeable. The new experiment and experimental data classes will allowed us to link these parameters to the experiments that determined them.

4.4.2 Converting design data to SBOL

Converting and sharing design information using SBOL is of paramount importance for the SD2 program, as it enables standardized data storage and sharing protocols through the different research groups. However, converting different data files and sources to SBOL is not easy without some scripting and effort.

During the program we provided support to groups needing to convert their challenge problem designs into SBOL and deposit them in SynBioHub. We have continued to provide support for SynBioHub and SBOL to SD2 Data Representation Working Group. We have been updating SynBioHub, as necessary, to support these activities. In particular, SynBioHub 1.5.4 includes a radically redesigned and documented API that greatly enhances programmatic access capabilities of SynBioHub, complete support for submit, visualization, and download plugins, a major refactoring of the submission process, support for combined GFF3/FASTA files, and several minor bug fixes. Some of these efforts included:

4.4.3 UCF to SBOL

UCF to SBOL (Chris, Pedro & Tim): Converted Voigt yeast gate to SBOL, and a prototype library for the UW yeast gates. In collaboration with Tim Jones, created an SBOL gate library for Klavin's yeast gates (https://synbiohub.programmingbiology.org/public/UWYeast1/UWYeast1_collection/1).

4.4.4 Other Design Libraries to SBOL

Converted two design libraries into SBOL and uploaded them to SynBioHub. In particular, we converted the Novel Chassis Group GFF files for 33 NAND 2.0 strains into SBOL and uploaded to SynBioHub: https://hub.sd2e.org/user/sd2e/design/novel_chassis_strains/1.

Worked with Hamid from Voigt's group to capture the sequence-level designs of their B. Subtilis circuits.

4.4.5 Software Converters to SBOL

We also wrote a converter to SBOL from a JSON file that Howard Salis provided that described 30,000+ characterized non-repetitive promoters. This promoter library has also been uploaded to SynBioHub: https://hub.sd2e.org/user/sd2e/SalisLabNonRepetitivePromoters2019Bacterial/SalisLabNonRepetitivePromoters2019Bacterial_collection/1.

4.4.6 VisBol updates

VisBOL is another standard intimately related to the SD2 program. VisBOL is a JavaScript software library to visualize DNA features from SBOL2 documents using the SBOL Visual

standard. A Web interface is also provided to enable end-user access to generate diagrams for presentations and publications. We provided support and development of this standard for the different challenge problems in the SD2 program.

VisBol updates (Chris & BU team): The SBOL visualization tool, VisBol, has been extended to support all new DNA glyphs for SBOL Visual 2, along with the new non-DNA glyphs.

In collaboration with Dany Fu and Arezoo Sadeghi from the Boston University SAIL team, added visualization capabilities to the VisBol software for functional information such as the interactions between device components.

We conducted a complete redesign of the VisBOL visualization tool that SynBioHub uses to display genetic design schematics (see below).

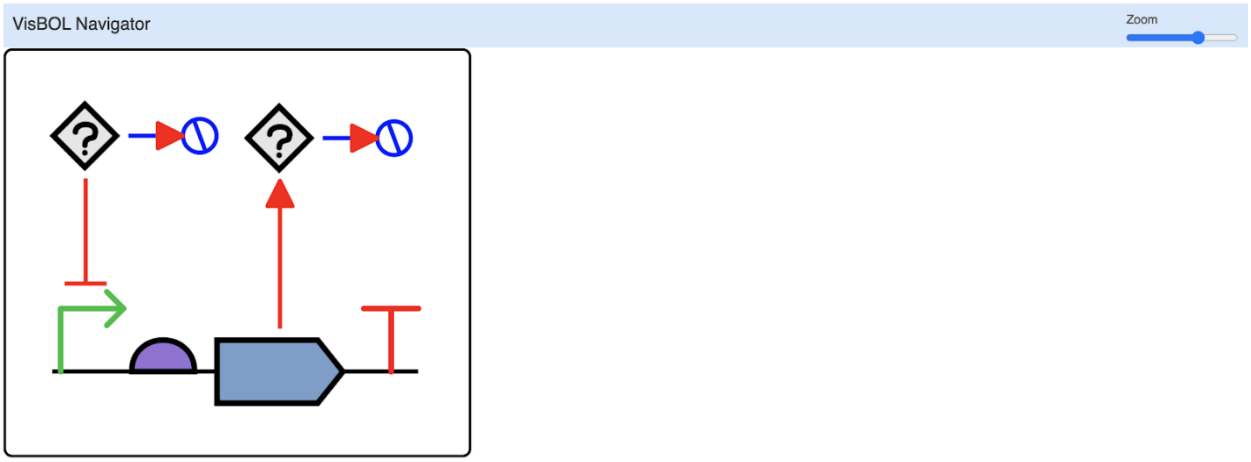


Figure 111. Complete redesign of the Visbol visualization tool that SynBioHub plugins use to display design schematics.

4.4.7 SynBioHub Updates

SynBioHub was chosen to be the online repository to store and share data throughout the SD2 program. During the entire program, we provided support for both uploading data into this online repository, as well as handling bugs, conversion efforts, and enhancements for the effective use of SynBioHub for this challenge. In particular, some of the changes where:

- Implementation of a number of critical bug fixes in SynBioHub required for the SD2 dictionary and SynBioHubAdapter. We also worked with the Roundtrip team to test these bug fixes, and the TACC team to first deploy the update on the staging and once confirmed functionality on the production SynBioHub server.
- In Q2-2019, we released SynBioHub 1.4, which included among other things: performance improvements for submission, retrieval, and rendering of data, and enhancements to page content and visualizations. This release included prototype plugin-in support to enable others to create customized data renderings. With the help of TACC, this release was deployed using our new docker containers for SynBioHub, Virtuoso, SBOLExplorer, and ElasticSearch. We have also developed a new testing and continuous integration methodology for SynBioHub, which should make future releases smoother. Shortly after the February 2019 PI meeting, we released SynBioHub

1.4.1, which included several bug fixes and enhancements identified during the meeting. During the February 2019 PI meeting, we also developed a plan with TACC that will enable nearly automatic updates for both the staging and production SynBioHub servers.

- In Q3-2019, we released SynBioHub 1.5. This version included the new SBOL 2.3 feature support, as well as support for authenticated data updates via SPARQL.
- We have improved our deployment workflow for new releases of SynBioHub. Now, whenever new code is merged into the master branch for SynBioHub, a new snapshot Docker container is created and automatically deployed on the staging instance of SynBioHub. This process has greatly accelerated the testing of new features. As a result, during this quarter we have successfully deployed three new versions of SynBioHub. SynBioHub 1.5.0 was deployed to production, which, among other things, provided full SBOL 2.3 support. Shortly thereafter, we released SynBioHub 1.5.1, which, among other things, provided the ability to edit some metadata on SynBioHub directly.
- Released SynBioHub 1.5.2 to the production server. This version included the following:
    - Preliminary GFF3 support.
    - Substantial update to the advanced search capabilities.
    - Created a full SPARQL interface for admin bulk editing.
    - Further improvements to logging.
    - Fixed a few issues with field editing.

- We have completed the new authentication module. This new module will enable more fine grain permissions to be granted, as well as new features like groups. Share links will also continue to be supported. The other major thrust will be a generalization of the types of plugins that will be supported. In particular, new plugins capabilities will be added for processing non-SBOL data for in-take into SynBioHub and data curation, as well as support for creating plugins to export or visualize data in non-SBOL formats. At IWBDA 2019, we also gave a talk on new visualization methods that were enabled by plugins, and published a journal paper for ACS Synthetic Biology about these and other SynBioHub plugins.
- In Q2-2020, we released SynBioHub 1.5.5, which included the following:
    - New Features
        - Support for sequence-based search
        - Additional improvements to the programmatic API support
        - Finer grain types on the collection pages and support to filter by these types
        - Ability to add and remove members of collections
    - Changes
        - Updated API error messages
        - Change to use sbols.org for SBOL terms
    - Bug fixes
        - Fixed several unhandled promise exceptions
        - Fixed bug with log lost on docker restarts
        - Fixed bug that caused crash when attachURL has not type selected

- Fixed several bugs in the add/edit/remove fields functionality

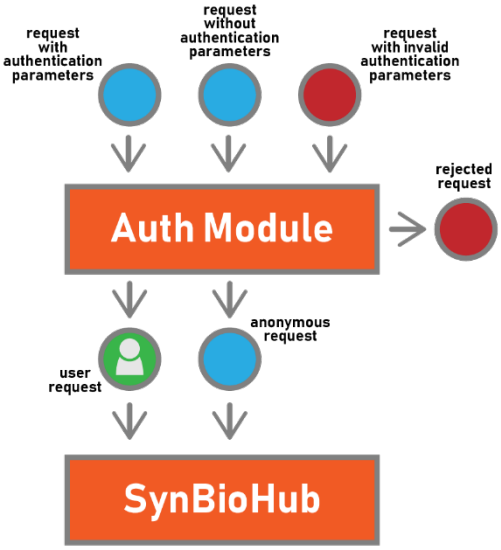


Figure 112. New authentication module for SynBioHub. All requests to access SynBioHub will come through this module. This will support more fine grain user and group permissions. It will also allow anonymous access to support share links.

- We completed and released support for SynBioHub plugins, which enable third parties to extend the SynBioHub core architecture. As shown in Figure 113, SynBioHub now has support for submit, visualization, and download plugins. Submit plugins enable preprocessing of data being submitted to SynBioHub. For example, this could be used by SD2 performers for processing spreadsheets, sequence files in alternative formats (GFF3, GenBank, FASTA), etc. Visualization plugins enable the development of new ways to render data stored in SynBioHub. For example, this could be used by SD2 performers integrate new tools to visualize provenance history. Finally, download plugins allow conversion to new file formats. For example, we have developed a plugin to transform annotated sequences into the SnapGene DNA format. We published a journal paper in the ACS Synthetic Biology journal describing plugins.

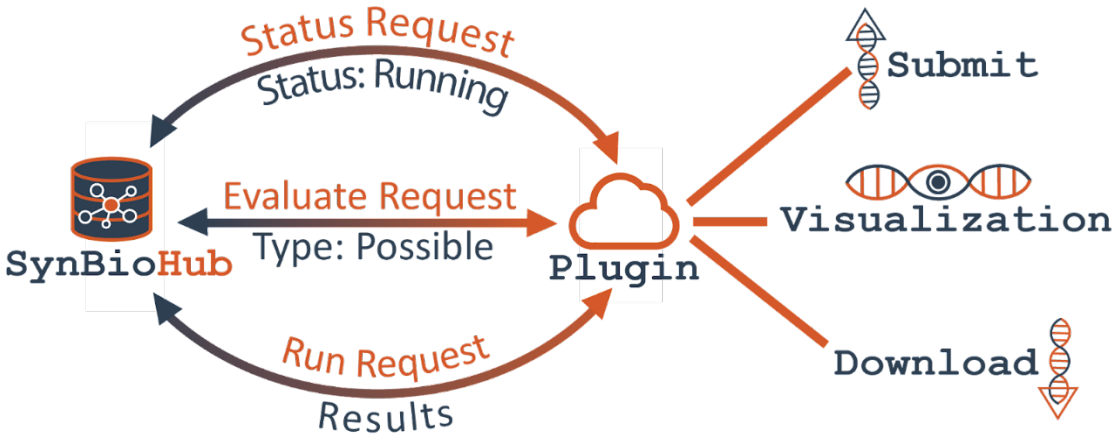


Figure 113. SynBioHub Plugin Architecture.

- Many bug fixes have been detected using the testing infrastructure being developed by an undergraduate on this project (James Scholz), who has also written an extensive documentation for SynBioHub's API (see https://synbiohub.github.io/api-docs/). We created a project with a Google Season of Docs student to develop a SynBioHub user's documentation to improve the SynBioHub user's experience. The final documentation can be found here (https://synbiohub.github.io).
-

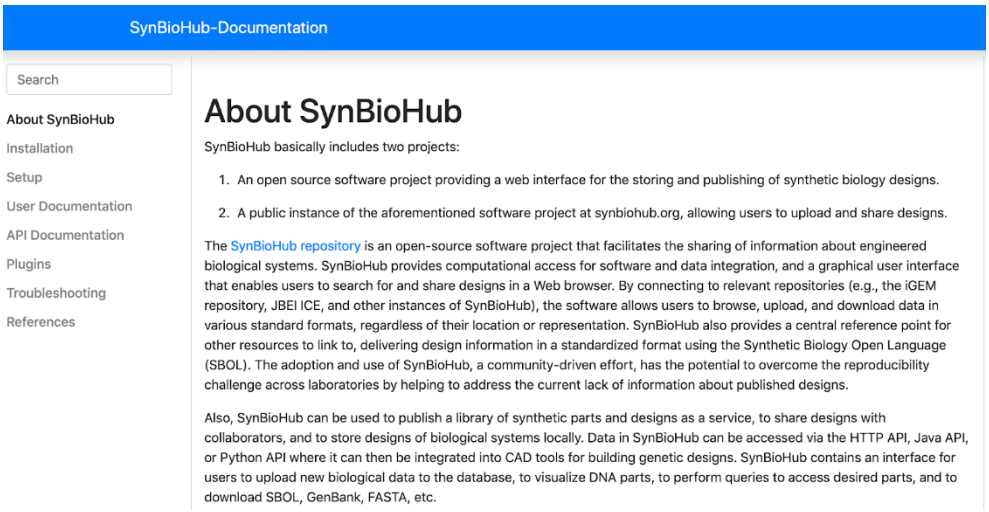


Figure 114. SynBioHub's new documentation found at https://synbiohub.github.io

- In Q3-2020, we completed a major release of SynBioHub (version 1.6.0). This version included a vastly improved and more robust API, many bug fixes detected using the testing infrastructure being developed by an undergraduate on this project (James Scholz), and an improved genetic circuit visualization tool, VisBOL2 developed by another undergraduate on this project (Ben Hatch). In addition, this new version supported spreadsheets for submitting libraries and build requests via a plugin developed by a Google Summer of Code student, as well as an improved sequence visualization capability developed by another Google Summer of Code student.

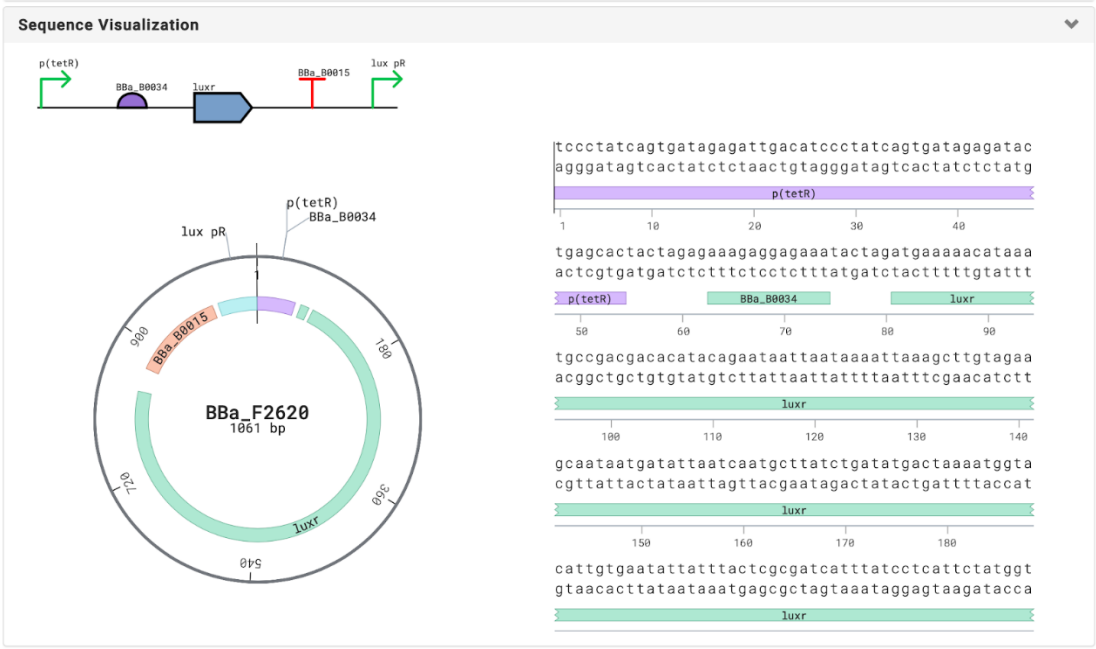


Figure 115. Sequence visualization using SynBioHub's plugin architecture.

- Throughout the program, we participated in office hours during the weekly SD2 Data Representation calls that have included various SD2 performers. For example, the Salis lab, with help from the data representation group, was able to create a collection of circuit designs found here: https://hub.sd2e.org/user/sd2e/SalisLabCircuitDesigns/SalisLabCircuitDesigns_collection/1
- Throughout the program, we worked on and developed different SynBioHub plugins, which enable third parties to extend the SynBioHub core architecture depicted in Figure 116. SynBioHub now has support for submit, visualization, and download plugins. Submit plugins enable preprocessing of data being submitted to SynBioHub. For example, this could be used by SD2 performers for processing spreadsheets, sequence files in alternative formats (GFF3, GenBank, FASTA), etc. Visualization plugins enable the development of new ways to render data stored in SynBioHub. For example, this could be used by SD2 performers integrate new tools to visualize provenance history. Finally, download plugins allow conversion to new file formats. For example, we have developed a plugin to transform annotated sequences into the SnapGene DNA format.

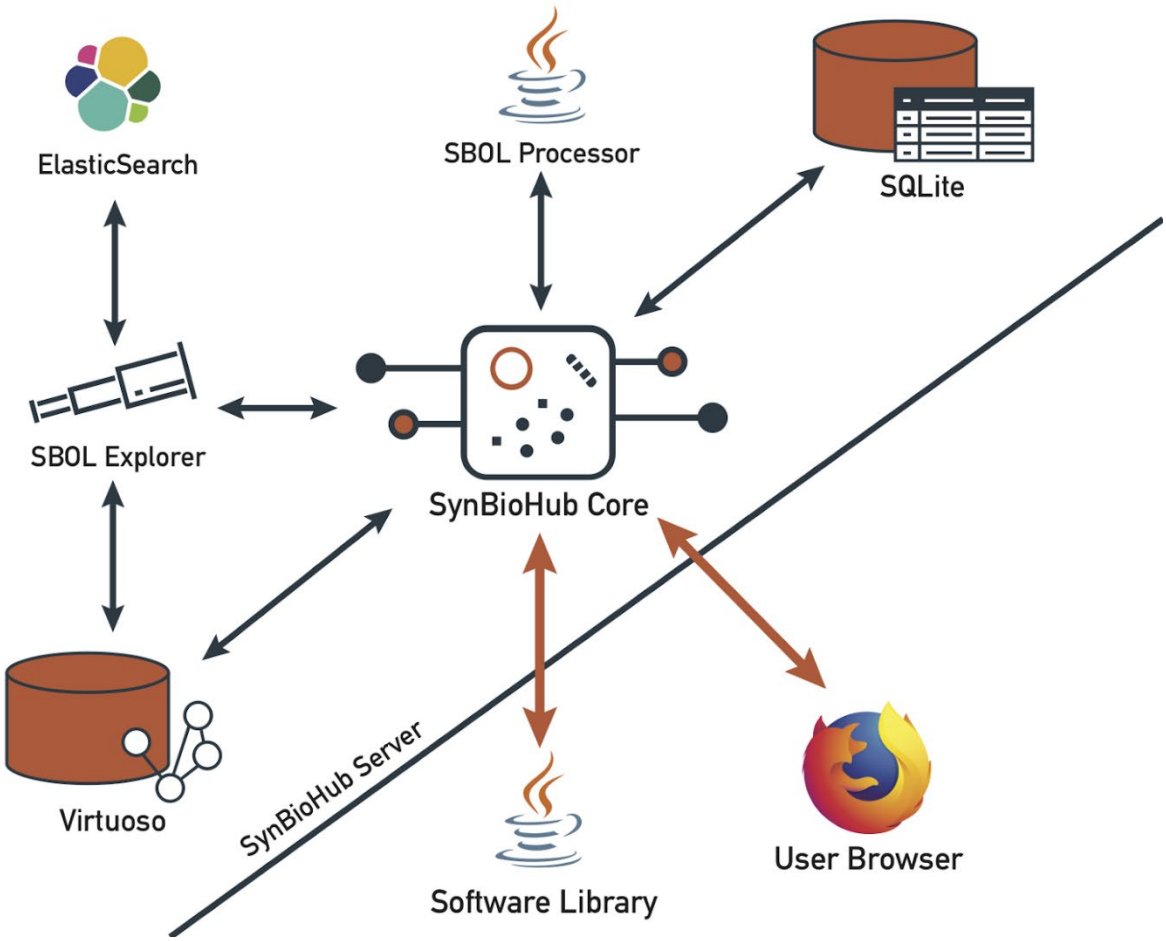


Figure 116. SynBioHub new backend and front-end architecture.

- We provided support for the data processing workflow that was worked out at the DARPA 2021 PI meeting in Boston last October. This included providing guidance to the DSGRN, UW Biofab, and Salis software teams who are now providing SBOL support within their tooling. We also updated SynBioHub, as necessary, to support these new activities. In particular, SynBioHub 1.5.3 included support for download of GFF3 files, improved support for GenBank files, and a couple of other small bug fixes. SynBioHub 1.5.4, which was released in 2021, included a radically redesigned API that greatly enhances programmatic access capabilities of SynBioHub.
- We focused on the front-end of SynBioHub to create what we are calling SynBioHub2. SynBioHub2 has a completely new browser-based user interface, but it communicates with the existing SynBioHub backend to fetch data. This work on SynBioHub2 including a demo was presented at IWBDA in September 2021.

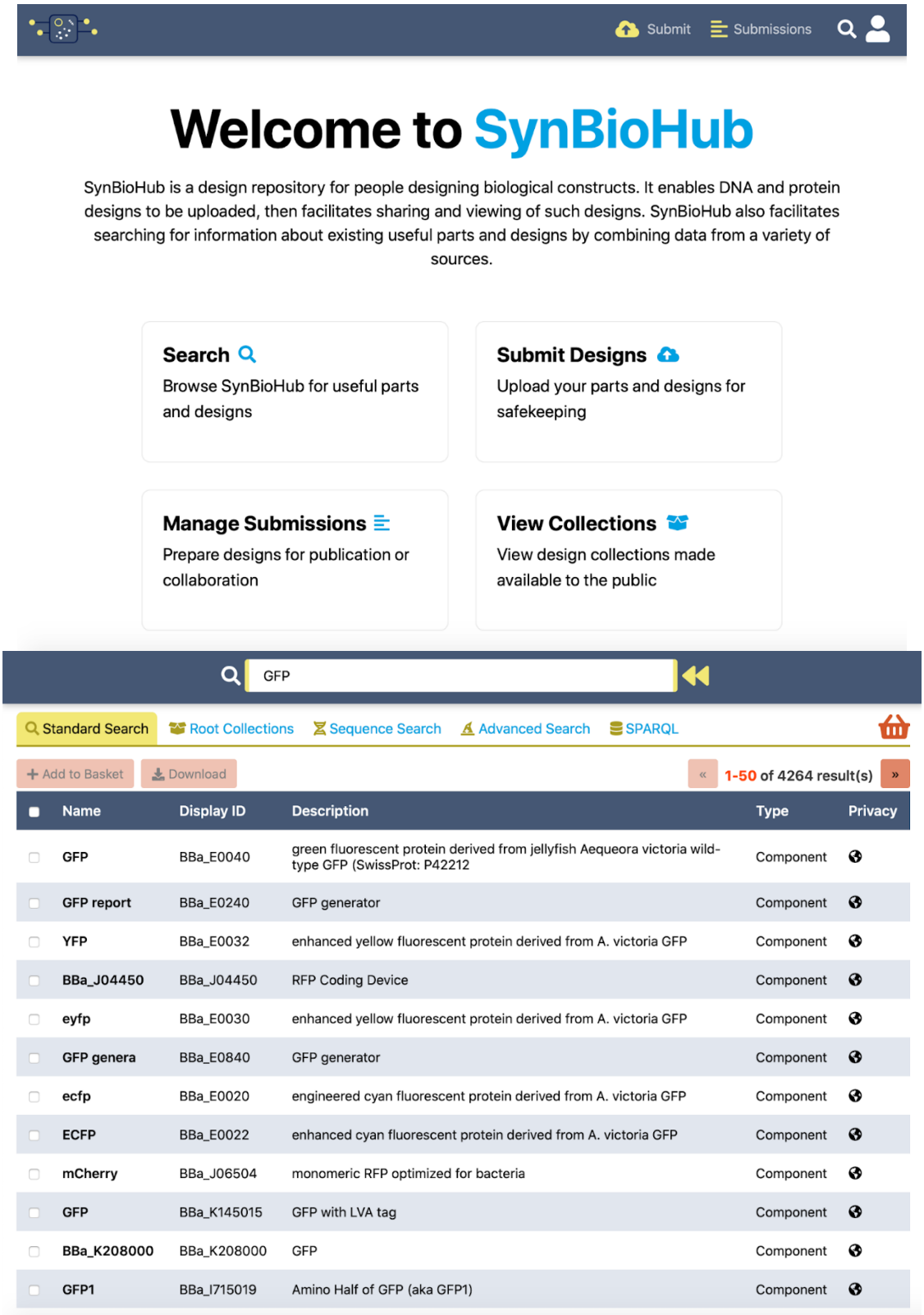


Figure 117. SynBioHub new front-end architecture.

We migrated all the core functionality over to this new front-end, and we are reached out for feedback from the SynBioHub user community.

- We received a new grant to support SynBioHub3 development from NIST. We met with our program managers to kickoff this project.
- We deployed this frontend to SynBioHub that we are calling SynBioHub2. SynBioHub2 has a completely new browser-based user interface, but it communicates with the existing SynBioHub backend to fetch data. One notable development was the creation of a new visualization plugin for SynBioHub that renders genetic circuits using VisBOL and SBOLCanvas renderings, as shown below.

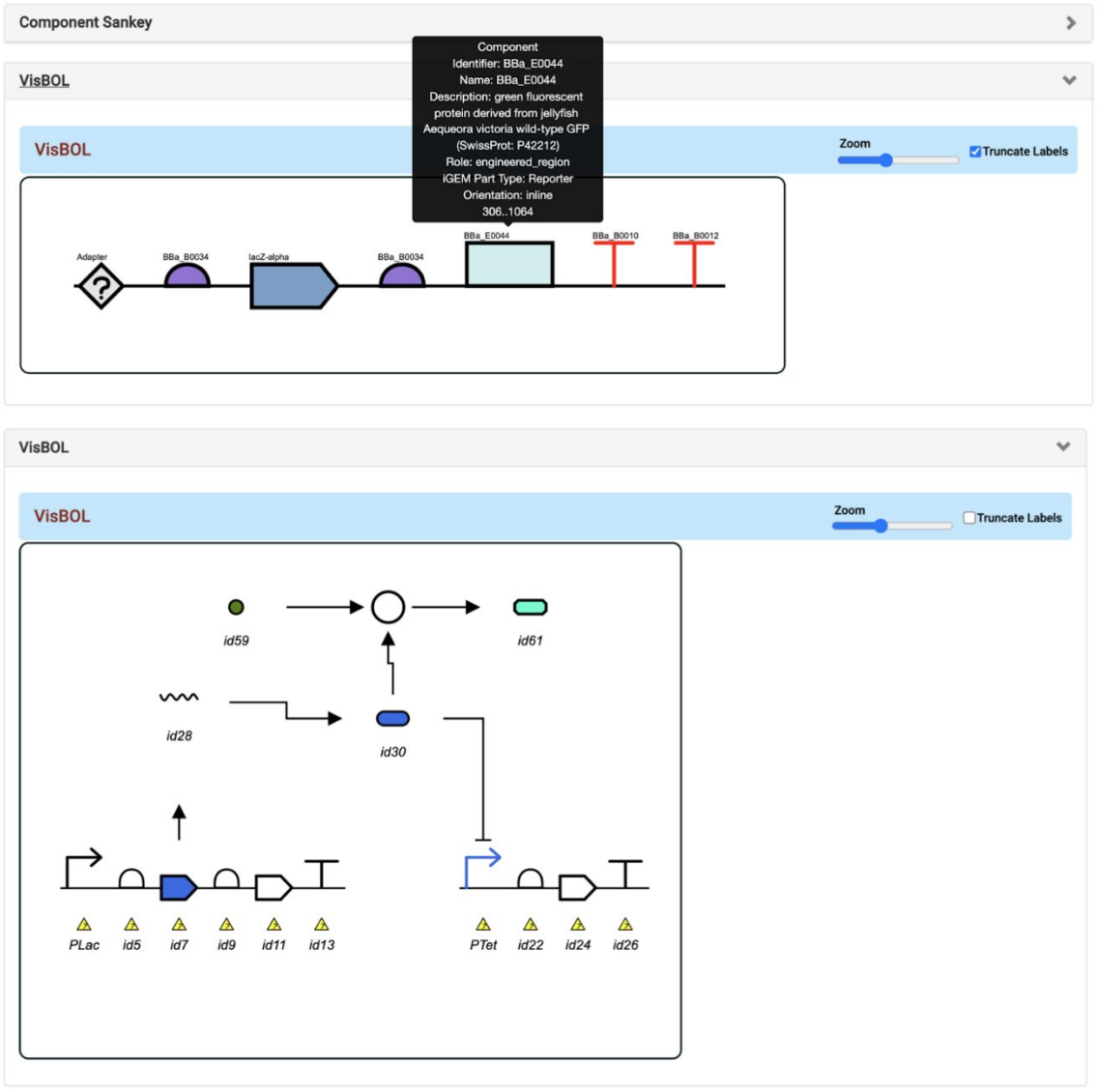


Figure 118. Further updates in genetic circuit design structure and function as part of SynBioHub's plugin architecture.

- We continued work on a new SynBioHub back-end. This back-end is written in Java, and it will be API compatible with the existing SynBioHub. We have successfully integrated the new back-end with the new front-end to create SynBioHub3. SynBioHub3 will be available for experimentation at https://dev3.synbiohub.org. We presented SynBioHub3 at the COMBINE Forum during the last quarter, and we are preparing a journal paper for ACS Synthetic Biology to be submitted this quarter.
- In Q3-2020, we released SynBioHub 1.6.0. This version included the following:

New Features

- Added sequence-based search
- Added support to edit annotations
- Added ability to add/remove members of collections
- Added ability to filter collections by type
- Added Virtuoso health check

Changes

- Refactored submission process code
- Added validation when editing roles and types
- Added ability to change more config options in admin panel
- Browse page now shows other SynBioHub's in Web-of-Registries

Bug fixes

- Fixed miscellaneous API issues
- Made collection icons persistent
- Fixed inconsistency with buttons
- Fixed issue with arrows on pull down menus
- Fixed issue where not all triples were deleted when collection removed

In Q3-2020, we released user documentation for SynBioHub (https://wiki.synbiohub.org).

We updated the SynBioHub documentation and aid to our test suite for the SynBioHub API.

We have continued development of SynBioHub 3. We have identified the following key design goals:

- Community involvement throughout the development process
- More intuitive front-end for biologists
- Improved integration of curation
- Ability to use different triplestore databases
- Support for SBOL3
- Preserve existing back-end API and plugin API

The core architecture is depicted below along with a couple screenshots for the prototype. In March, we led a breakout session at the HARMONY workshop to discuss SynBioHub3 with users and potential users. There were over 20 participants including academic, industrial, and government researchers. This discussion was very informative. The users particularly emphasized the need for effective means for search, curation, data sharing, and data conversion.

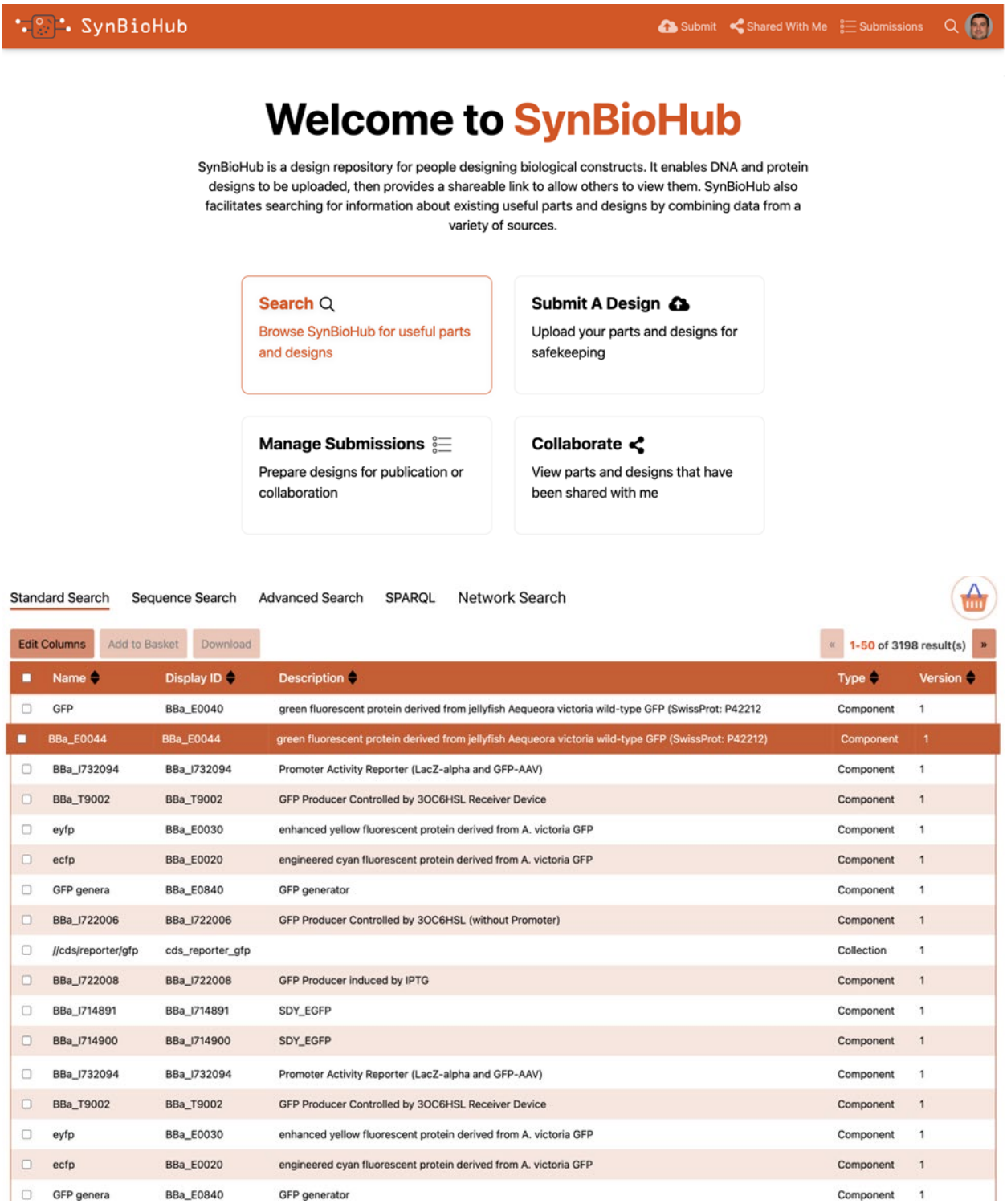


Figure 119. SynBioHub 3.0 new backend and front-end architecture.

- We have explored transition opportunities for SynBioHub. These include an Army center with the Voigt group and others, a DOE proposal with Jake Beal (BBN) and others, and finally a potential opportunity with NIST that came out of the HARMONY workshop discussions.
- We developed prototypes of SynBioHub3 that involved both a more intuitive and interactive front-end and a new backend that supports various performance, data encoding and storage, and security enhancements.
- We developed and submitted proposals for transition opportunities for SynBioHub. We submitted a proposal to the Army to create a Synthetic Biology Center with the Voigt group and others, and a proposal NIST to support SynBioHub3 development.
- We presented SynBioHub3 at the COMBINE Forum, and submitted a journal paper for ACS Synthetic Biology.
- For this project, we have also begun recruiting external advisory board members to help provide input on SynBioHub3 development. This board currently includes the original developers on SynBioHub (James McLaughlin and Zach Zundel), as well as David Ross (NIST) and Geoff Baldwin (Imperial).

4.4.8 Cello updates

During this project, we offered support to the development of Cello, and its integration with SBOL. Some changes include:

- Assisted in refining Cello's SBOL support for parts, designs, and build instructions.
- Continued collaboration with Tim Jones on Cello 2.0. Updates on this are reported in the CIDAR group section.
- Assisted with the new web API, which will tightly integrate with SynBioHub to fetch part information encode in SBOL and save designs also encoded in SBOL. We also plan to integrate our dynamic model generation procedure into Cello. Additional updates on this work are reported in the CIDAR group section.

4.4.9 Dynamic model generation

Using a dynamic model formulation from Hamid Hosseini in Voigt's group, we created an initial implementation of a dynamic model generation that incorporates parameters from part characterization data being produced by the SD2 project.

We also worked with Hamid Doosthosseini to use his dynamic model of genetic circuits for model generation and simulation in iBioSim tool. This allows us to model genetic circuits using a dynamic modeling, and simulate using various methods (ODEs, stochastic, etc) to predict mRNA and protein production over time for various genetic circuits.

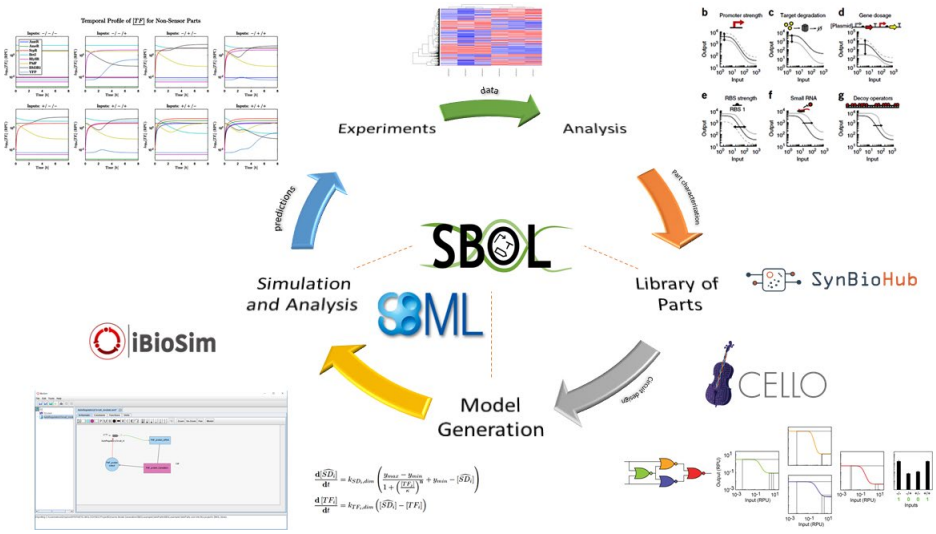


Figure 120. Updated proposed workflow for the integration of experimental data with data modeling/simulation using a new dynamic model for genetic circuits created by Hamid Doosthosseini, integrated to iBioSim to produce RNAseq and Protein concentration predictions to be compared with experimental data and to help de-bug genetic circuits.

This new dynamic modeling provided time-series predictions of productions. The mRNA output predictions can be used to compare it with RNAseq data obtained from experiments. This comparison can lead to better refining information about the parameters used in the model, and as a way to debug a genetic circuit by comparing the states of the internal genetic gates and the part performance with the predicted outcome of this model. At the same time, this can be used to determine termination failures, sensor malfunction or antisense promoters not detected in the original model.

We continued to develop our dynamic model generator to incorporate road blocking effects along with new measured parameters that represent the rate that gates switch on and off. This new model is depicted in Figure 120.

4.4.10 Circuit function hazard analysis

We continued to test and refine our new dynamic model generation procedure that incorporates Cello characterization. In order to demonstrate its utility over steady-state analysis, we have applied our method to analyze the problem of glitches in Cello designed circuits. One example is shown in Figure 121. In circuit 0x8E from the Cello Science paper, a glitch in the output was observed experimentally. To better understand this behavior, we created a dynamic model for this circuit using our new model generation procedure, and the simulation is shown in the figure. The glitch observed experimentally is marked in the simulation. The reason for this glitch is due to something down as a *function hazard*. A function hazard occurs when there is a race through the logic caused by two or more inputs changing simultaneously. The Karnaugh map at the top of the figure illustrates this particular function hazard. Namely, it occurs when A (IPTG) and C (Ara) are changed from a low to a high value simultaneously. If the change in C is felt before A, there will be no problem and the circuit will output a high continuously. However, if the change in A is felt before C, then the output will potentially go towards a low value before finally settling back

to the desired high value. This glitch is what we actually observe in the simulation. A function hazard is an inherent property of the Boolean function, and they cannot be avoided or removed by changing the combinational logic implementation of the function. The only way to eliminate this function hazard and the possibility of this glitch is to limit the allowed input changes. In this case, we would need to require that C is set to high first, and only after the circuit has stabilized would we be allowed to change A to high. Removal of this glitch may be critical if the glitch has the potential to enable an undesired downstream process such as a programmed cell death. Dynamic simulation allows us to observe this potential design flaws that would otherwise be missed in a steady-state analysis. Furthermore, in the future, we would like to extend this analysis further to look at the probability of the glitches actually manifesting. Do to the inherently stochastic nature of genetic circuits, some function hazards will produce glitches with a higher probability than others. It is these high probability function hazards that will need to be avoided.

Figure 121. Example of a function hazard in a genetic circuit design. In this circuit, when A (IPTG) and C (Ara) go high simultaneously, this creates a race through the logic. Namely, if the change on C is felt first, the circuit remains outputting a high value, but if the change on A is felt first, there will be a glitch to a low value on the way to the final high state. This is glitch is observed in the dynamic simulation generated by our model generation procedure. This race is known as a function hazard, and these are inherent to the Boolean function being implemented. There is no logic implementation that can eliminate this function hazard. The only solution is to restrict the order of input changes (for example, require C to be set high before A).

In July, we presented our hazard analysis results at the 2019 International Workshop on Bio-Design Automation (IWBDA) in Cambridge UK. We submitted a journal paper describing this work and our dynamic model generator for the IWBDA special issue of ACS Synthetic Biology (P. Fontanarrosa et. al., ACS Synth. Biol., 2020). This article was favorably reviewed, accepted, and published. We have extended this work to consider logic hazards, and we presented this extension during the January 2020 PI meeting. The key difference between function hazards and logic hazards is that function hazards can only be avoided by restricting the allowed input changes, but logic hazards can be avoided by changing the logic design. We have demonstrated that the circuit can be redesigned as shown in Figure 122 to be free of logic hazards. We are beginning to write a paper on this topic.

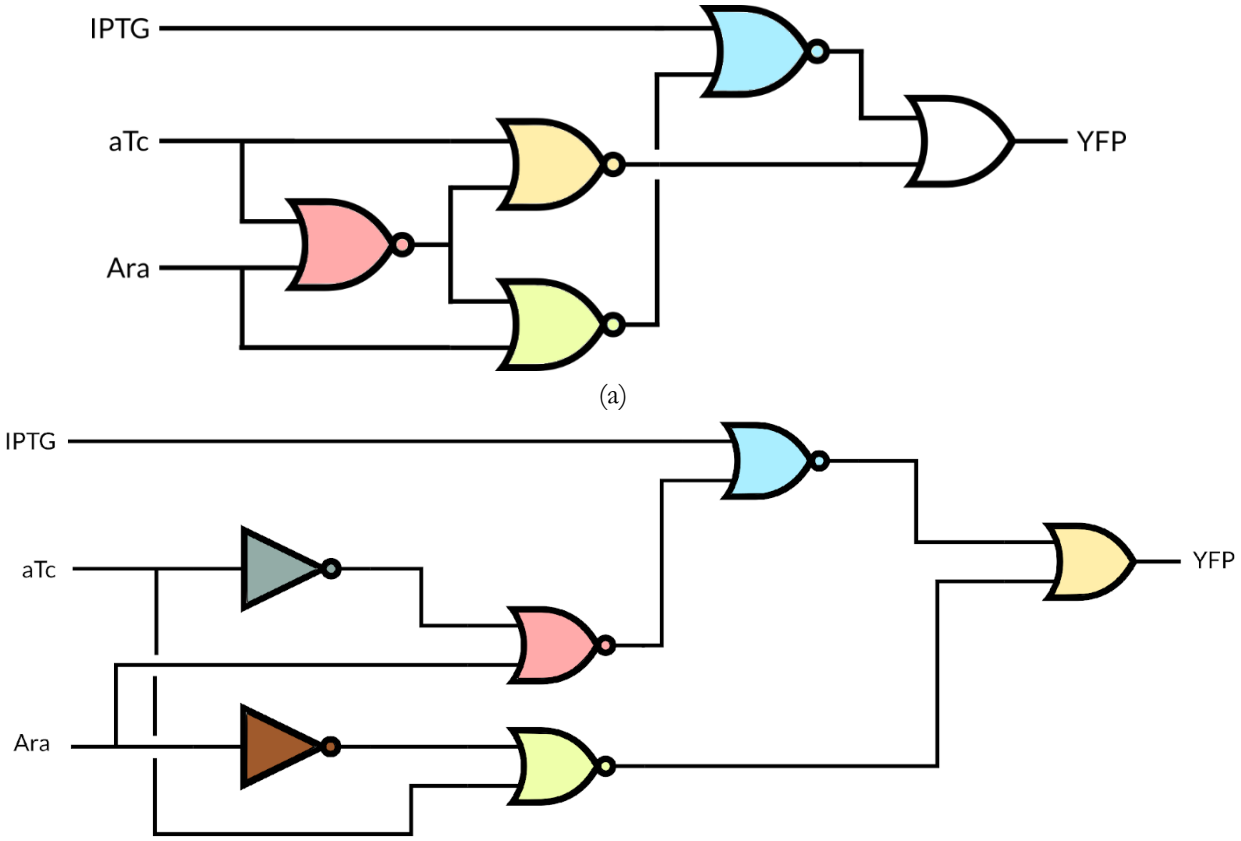
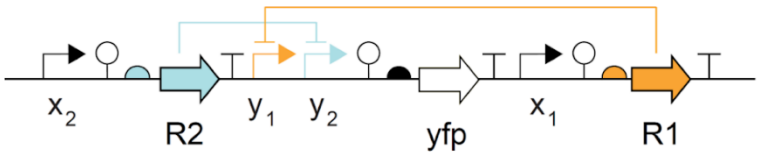


Figure 122. Original and Alternative Circuit Designs (a) Original Cello design of the circuit 0x8E. (b) Alternative design of circuit 0x8E that is free of logic hazards.



The non-specific suppression of the upstream promoter is captured by the parameter α

The degree of roadblocking is represented by the first term in parenthesis

$$y_{tot} = y_1 + y_2 = y_{min,2} + (y_{max,2} - y_{min,2})\frac{{K_2}^{n_2}}{{K_2}^{n_2} + {x_2}^{n_2}} + \alpha\left(\frac{{K_2}^{n_2} + \beta\,{x_2}^{n_2}}{{K_2}^{n_2} + {x_2}^{n_2}}\right)\left[y_{min,1} + (y_{max,1} - y_{min,1})\frac{{K_1}^{n_1}}{{K_1}^{n_1} + {x_1}^{n_1}}\right]$$

β captures competing effects between the kinetics of the repressor off rate and dissociation rate

y is the output

$$\frac{dy}{dt} = \begin{cases} \tau_y^{ON}(y_{ss} - y) & if\ y < y_{ss} \\ \tau_y^{OFF}(y_{ss} - y) & otherwise \end{cases}$$

y_{ss} is the steady-state response

Figure 123. Improved dynamic model formulation. The new steady-state model includes parameters to express the effect of roadblocking. The dynamics are now governed by measured rates of change to turn on and off.

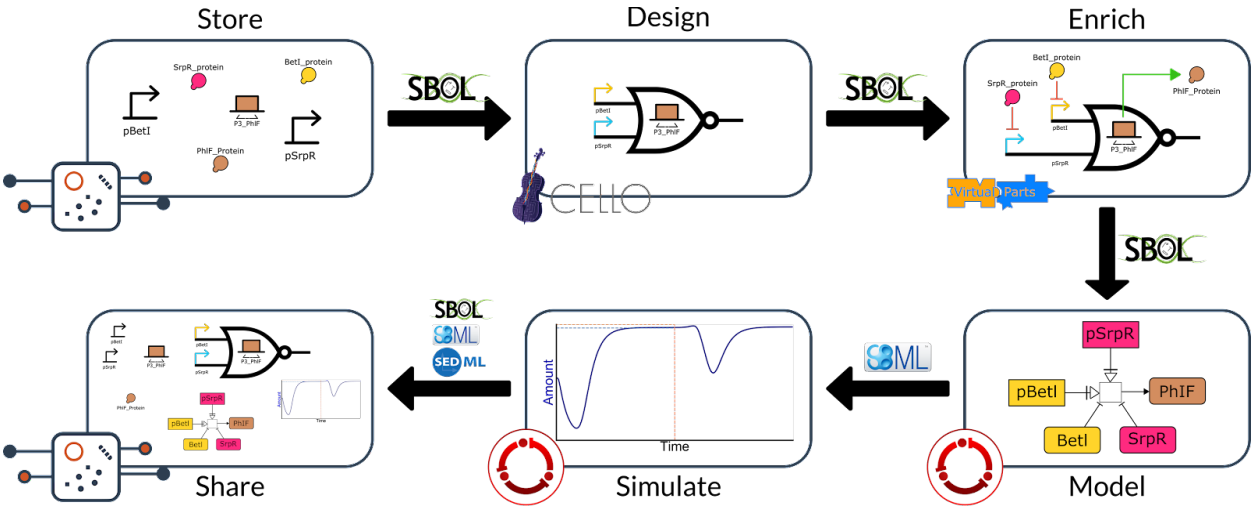


Figure 124. Design build test learn workflow for genetic circuit failure percent characterization and modeling. This workflow shows how a user can start with parts and designs in SynBioHub and end up with prediction models for the same designs, also using SynBioHub.

4.4.11 Stochastic analysis of genetic circuit

We continued to investigate models and analysis methods for genetic circuit hazards. In particular, we were interested in exploring the effect of intrinsic noise (dynamic variations of protein production and degradations rates within cells) to extrinsic noise (static variations of the protein production and degradation rates between different cells). In addition, in collaboration with Lukas Buecherl, a PhD student on another project, we have developed stochastic analysis techniques to explore the probability of glitches due to hazards, which ultimately resulted in a publication in ACS Synthetic Biology (L. Buecherl et al., ACS Synth. Biol., 2021).

We have applied these models and analysis techniques to both genetic circuits designed by Cello and genetic network topologies produced by DSGRN. In particular, we analyzed several DSGRN designs to evaluate their robustness using the model generation and hazard analysis methodology developed by Pedro Fontanarrosa. In particular, we compared the design of OR and NOR gates designed in Yeast by Gander et al. with DSGRN redesigns. Below is an example of some results in which we looked at the probability of these gates producing erroneous behavior (i.e. glitches) under both an intrinsic noise model (i.e. variation within a cell) and an extrinsic noise model (i.e. variation across different cells). The indication is that choice of which design produces the most robust operation appears to be different depending on which type of noise is dominant. In particular, the redesigned circuit appears to perform better when intrinsic noise is dominant, while the original circuit appears to perform better if extrinsic noise is dominant.

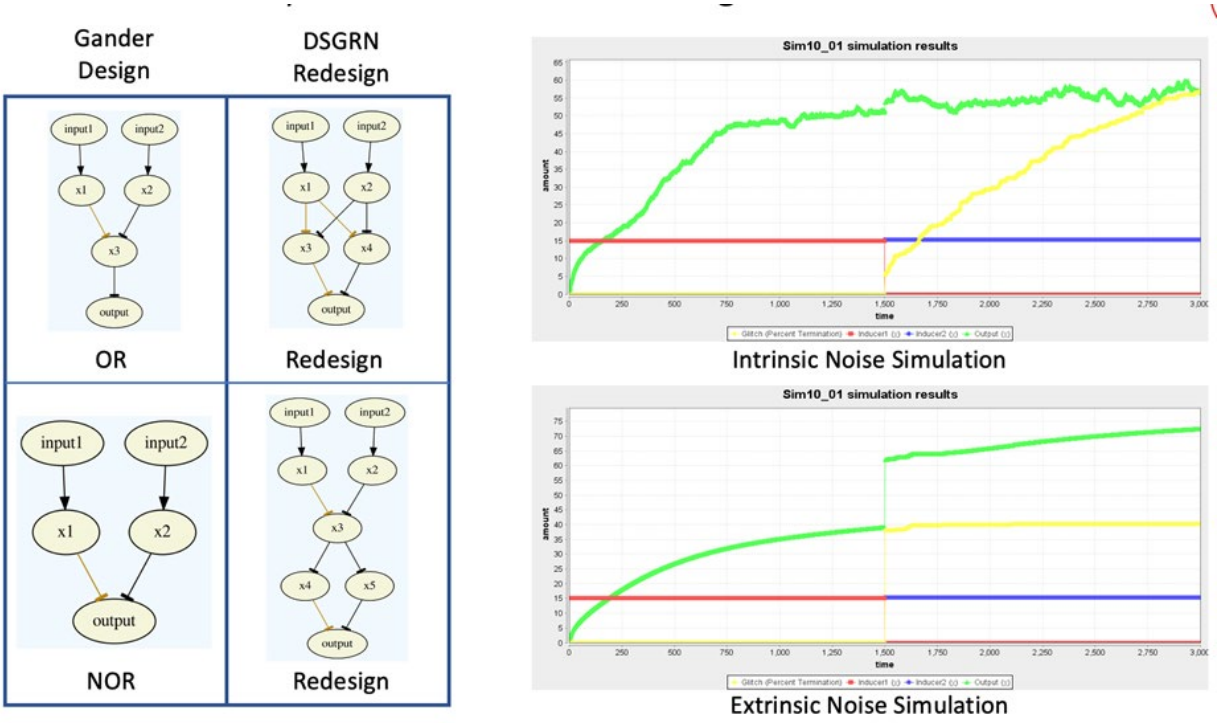


Figure 125. Extrinsic and intrinsic noise model genetic circuits failure predictions for two different DSGRN design layouts using iBioSim.

We also collaborated with Bree Cummins, to look at the hazard behavior in several alternative yeast circuit designs. We extended the development of stochastic analysis in order to determine the probability that these genetic circuit hazards will result in glitching behavior. We are applying this work to both genetic circuits produced by Cello and yeast circuit designs produced by DSGRN. Preliminary results are very encouraging.

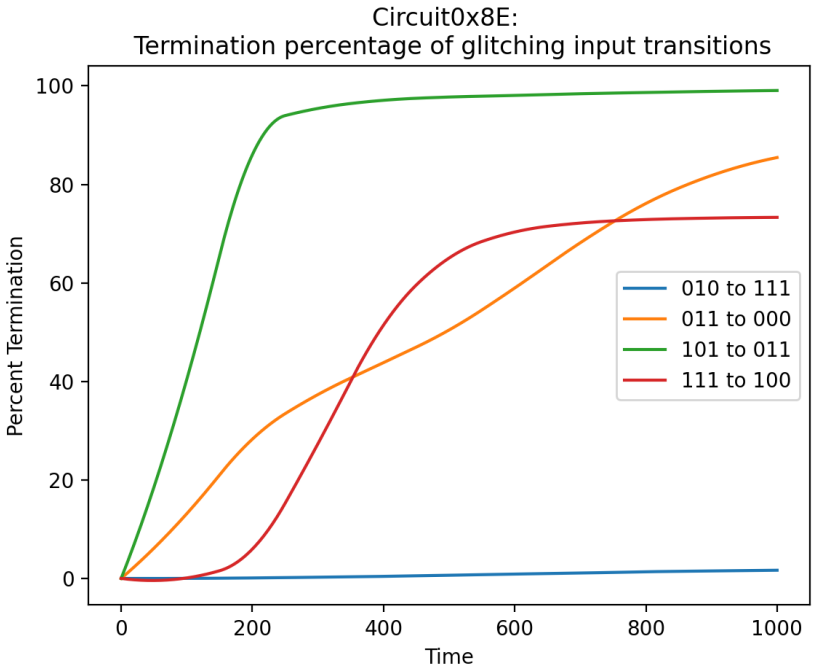


Figure 126. Circuit 0x8E intrinsic noise modeling predictions of circuit failure for different inducing concentrations.

We analyzed several DSGRN designs to evaluate their robustness using the model generation and hazard analysis methodology developed by Pedro Fontanarrosa. Below is an example of some DSGN designs that we have analyzed along with probabilities of different types of failures. These include function hazards where a glitch occurs upon the change of multiple inputs simultaneously, setup glitches where there is a glitch when the circuit starts with a given input initially and hold failures where the state is not properly maintained over time. These results provide useful information about the various potential sources for erroneous behavior for each alternative potentially aiding in design decisions.

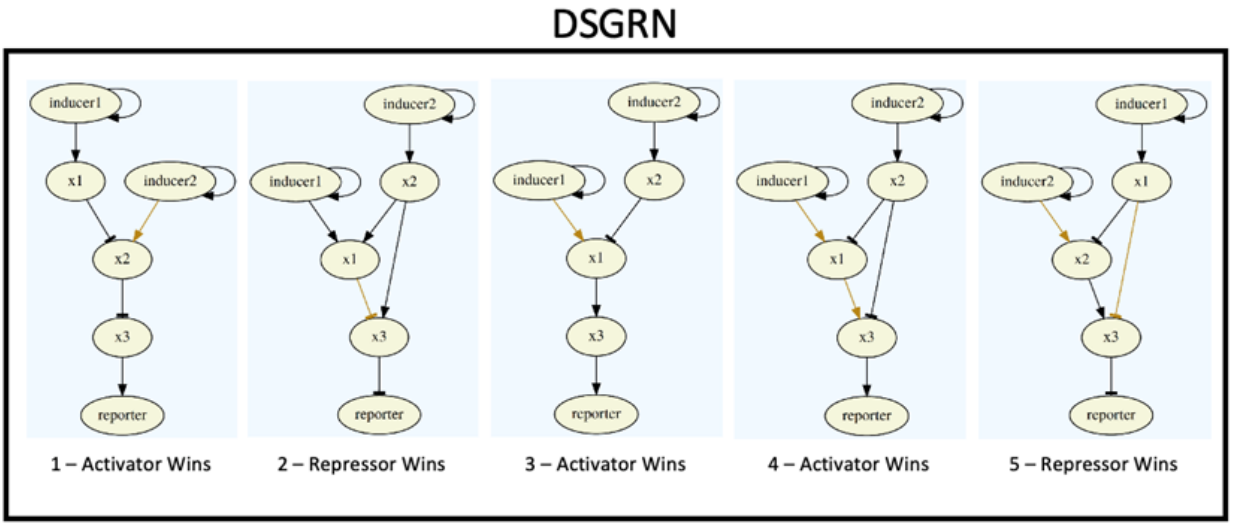


Figure 127. SGRN designed circuits

Table 3. Noise model predictions for DSGRN designed circuit failure percentages

Failure	Transition /State	1 – Activator Wins	2 – Repressor Wins	3 – Activator Wins	4 – Activator Wins	5 – Repressor Wins
	Percent Failures					
Function Hazard	00 -> 11	0	20	5	0	42
	11 -> 00	4	3	49	90	4
Set-up glitches	00	21	4	100	100	3
	01	10	2	50	49	0
	10	75	65	23	42	75
	11	9	32	75	44	55
Hold-State failures	00	2	4	100	10	2
	01	21	1	0	0	0
	10	55	25	100	20	23
	11	1	15	60	0	29

Percent failure predictions for Function hazards, set-up glitches, hold-state failures for a DSGRN designed circuit using intrinsic and extrinsic noise model simulations.

We have completed our analysis of several DSGRN designs to evaluate their robustness using the model generation and hazard analysis methodology developed by Pedro Fontanarrosa. In

particular, we compared the design of OR and NOR gates designed in Yeast by Gander et al. with DSGRN redesigns. Below are the results in which we looked at the probability of these gates producing erroneous behavior (i.e. glitches) under both an intrinsic noise model (i.e. variation within a cell) and an extrinsic noise model (i.e. variation across different cells). These results indicate that the choice of which design produces the most robust operation appears to be different depending on which type of noise is dominant. In particular, the redesigned circuit appears to perform better when intrinsic noise is dominant, while the original circuit appears to perform better if extrinsic noise is dominant. This work will be presented in a paper being led by Bree Cummins and the Yeast States working group.

Table 4. Intrinsic noise model predictions of circuit failure percentages

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

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

Table 5. Extrinsic noise model predictions of circuit failure percentages

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

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

Table 6. Intrinsic and extrinsic noise model predictions of circuit failure percentages

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

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

We also successfully worked on the re-parameterization experiments with the new CRISPR gates so that we can model, analyze, and predict the performance of NOR and OR re-designed circuits using DSGRN. We successfully characterized the gates used in the DSGRN designs using Cello modeling techniques (see parameters below). We then remodeled the DSGRN OR and NOR circuits (original and re-designed) using Cello modeling and characterized parameters. These results will be included in a forthcoming paper on the yeast states project.

Table 7. Hill function parameterization of CRISPR gates designed throughout the SD2 program in order to obtain model predictions before building the circuits.

Parameters	WBF_25784']	Parameters	WBF_36117']
n	-0.062106820874499195	n	1.1986575035065876
k	1.678092418893987e-15	k	6.850862092022009
ymax	3448.55947716817	ymax	12218.25921
ymin	356.078896572589	ymin	2672.652785

Parameters	['UWBF_36110']	['UWBF_36112']	['UWBF_36115']	['UWBF_36113']	['UWBF_36116']	['UWBF_36114']
n	3.482595636227894	4.159615347315064	3.747378529248974	3.490683222676764	4.154375448826994	4.122517072147784
k	4041.3645100526	4694.445288248454	4594.014744078124	4548.285272967804	5025.292681895934	4808.185674520564
ymax	8945.238114	11994.28668	11249.16229	11220.68686	10909.85502	12530.43814
ymin	1487.392097	2492.356539	1946.441898	2159.521548	2425.044833	2661.802854
	r9	r10	r3	r5	r2	r1

Parameters	['UWBF_24959']	['UWBF_24962']	['UWBF_24960']	['UWBF_24963']	['UWBF_24961']	['UWBF_24952']
n	2.3382116735988037	2.5515685334584552	2.4811777217540483	2.7525176253214774	2.563571013440848	2.59948820618263
k	620.0463752005587	411.5825766278883	472.3655059911373	379.4079548450835	635.7012161825055	456.27194445326984
ymax	15198.5784009186	17048.287724102	15127.2222926103	17055.9785206273	16583.3014828694	15413.1839583825
ymin	1415.72922511981	1161.47178854524	744.601338476396	834.098424424363	1509.41049466909	1036.5821944245
	r10	r3	r5	r2	r1	r7

4.4.12 Collaboration and Workshops

We hosted weekly meetings with the MIT/BU/Broad group, as well as weekly Data Representation Working Group meetings and bi-weekly Roundtrip Working Group meetings. Chris and Pedro also have a weekly meeting with Tim Jones (BU) to discuss Cello development. Finally, several other ad hoc meetings were held with various SD2 participants. Overall, the ramp-up is complete, and several areas have been identified where Myers group will be making contributions in the coming months.

4.5 Conclusion

The extensive use of the data standards proposed for this program propelled the development of both the tools that use these standards, as well as the standards themselves, since there was a direct contact between developers and users. This accelerated even more on each of the SD2 quarterly meetings as we could hear the critiques, concerns, suggestions, and comments of the users that used the data standards chosen to share design information through this program.

5 PUBLICATIONS

MIT Voigt Lab:

Jones, Timothy S., Samuel Oliveira, Chris J. Myers, Christopher A. Voigt, and Douglas Densmore. "Genetic circuit design automation with Cello 2.0." Nature Protocols (2022): 1-17.

Eslami, Mohammed, Amin Espah Borujeni, Hamed Eramian, Mark Weston, George Zheng, Joshua Urrutia, Carolyn Corbet et al. "Prediction of whole-cell transcriptional response with machine learning." Bioinformatics 38, no. 2 (2022): 404-409.

Dorfan, Yuval, Amin Espah Borujeni, YongJin Park, Uma Saxena, Ben Gondon, Christopher A. Voigt, and Enoch Yeung. "A data-driven method for quantifying the impact of a genetic circuit on its host." (2019).

Taketani, Mao, Jianbo Zhang, Shuyi Zhang, Alexander J. Triassi, Yu-Ja Huang, Linda G. Griffith, and Christopher A. Voigt. "Genetic circuit design automation for the gut resident species Bacteroides thetaiotaomicron." Nature biotechnology 38, no. 8 (2020): 962-969.

Chen, Ye, Shuyi Zhang, Eric M. Young, Timothy S. Jones, Douglas Densmore, and Christopher A. Voigt. "Genetic circuit design automation for yeast." Nature Microbiology 5, no. 11 (2020): 1349-1360.

Shin, Jonghyeon, Shuyi Zhang, Bryan S. Der, Alec AK Nielsen, and Christopher A. Voigt. "Programming Escherichia coli to function as a digital display." Molecular systems biology 16, no. 3 (2020): e9401.

Park, Yongjin, Amin Espah Borujeni, Thomas E. Gorochowski, Jonghyeon Shin, and Christopher A. Voigt. "Precision design of stable genetic circuits carried in highly-insulated E. coli genomic landing pads." Molecular systems biology 16, no. 8 (2020): e9584.

Espah Borujeni, Amin, Jing Zhang, Hamid Doosthosseini, Alec AK Nielsen, and Christopher A. Voigt. "Genetic circuit characterization by inferring RNA polymerase movement and ribosome usage." Nature communications 11, no. 1 (2020): 1-18.

MIT Broad Foundry:

Garcia BJ, Urrutia J, Zheng G, Becker D, Corbet C, Maschhoff P, Cristofaro A, Gaffney N, Vaughn M, Saxena U, Chen YP, Gordon DB, Eslami M. A toolkit to enhance reproducibility of RNASeq analysis for synthetic biologists. Synthetic Biology (Under Review)

Shekar V, Yu V, Garcia BJ, Gordon DB, Moran G, Roch L, Duran AG, Najeeb MA, Zeile M, Nega P, Li Z, Kim M, Chan E, Norquist A, Friedler S, Schrier J. Great perovskite bake off. (Being Written)

CIDAR Lab:

Jones, Timothy S., Samuel Oliveira, Chris J. Myers, Christopher A. Voigt, and Douglas Densmore. "Genetic circuit design automation with Cello 2.0." Nature Protocols (2022): 1-17.

Myers Group:

P. Fontanarrosa, L. Buecherl, and C. J. Myers, “Comparison of Extrinsic and Intrinsic Noise Model Predictions for Genetic Circuit Failures,” in 13th international workshop on bio-design automation, online, Sep. 2021, pp. 25–29.

L. Buecherl et al., “Stochastic Hazard Analysis of Genetic Circuits in iBioSim and STAMINA,” ACS Synth. Biol., Oct. 2021, doi: 10.1021/acssynbio.1c00159.

H. Baig et al., “Synthetic biology open language visual (SBOL visual) version 3.0,” Journal of Integrative Bioinformatics, Oct. 2021, doi: 10.1515/jib-2021-0013.

H. Baig et al., “Synthetic biology open language visual (SBOL Visual) version 2.3,” Journal of Integrative Bioinformatics, 2021, doi: 10.1515/jib-2020-0045.

J. A. McLaughlin et al., “The Synthetic Biology Open Language (SBOL) Version 3: Simplified Data Exchange for Bioengineering,” Front. Bioeng. Biotechnol., vol. 8, 2020, doi: 10.3389/fbioe.2020.01009.

P. Fontanarrosa, H. Doosthosseini, A. Espah Borujeni, Y. Dorfan, C. A. Voigt, and C. J. Myers, “Genetic Circuit Dynamics: Hazard and Glitch Analysis,” ACS Synth. Biol., Aug. 2020, doi: 10.1021/acssynbio.0c00055.

H. Baig et al., “Synthetic biology open language visual (SBOL visual) version 2.2,” Journal of Integrative Bioinformatics, vol. 17, no. 2–3, Jun. 2020, doi: 10.1515/jib-2020-0014.

H. Baig et al., “Synthetic biology open language (SBOL) version 3.0.0,” Journal of Integrative Bioinformatics, vol. 17, no. 2–3, Jun. 2020, doi: 10.1515/jib-2020-0017.

6 REFERENCES

Chen, Ye, Shuyi Zhang, Eric M. Young, Timothy S. Jones, Douglas Densmore, and Christopher A. Voigt. "Genetic circuit design automation for yeast." Nature Microbiology 5, no. 11 (2020): 1349-1360.

Jones, Timothy S., Samuel Oliveira, Chris J. Myers, Christopher A. Voigt, and Douglas Densmore. "Genetic circuit design automation with Cello 2.0." Nature Protocols (2022): 1-17.

Gander, Miles W., Justin D. Vrana, William E. Voje, James M. Carothers, and Eric Klavins. "Digital logic circuits in yeast with CRISPR-dCas9 NOR gates." Nature communications 8, no. 1 (2017): 1-11.

Nielsen, Alec AK, Bryan S. Der, Jonghyeon Shin, Prashant Vaidyanathan, Vanya Paralanov, Elizabeth A. Strychalski, David Ross, Douglas Densmore, and Christopher A. Voigt. "Genetic circuit design automation." Science 352, no. 6281 (2016): aac7341.

Lalanne, Jean-Benoît, James C. Taggart, Monica S. Guo, Lydia Herzel, Ariel Schieler, and Gene-Wei Li. "Evolutionary convergence of pathway-specific enzyme expression stoichiometry." Cell 173, no. 3 (2018): 749-761.

McLaughlin, James Alastair, Chris J. Myers, Zach Zundel, Göksel Mısırlı, Michael Zhang, Irina Dana Ofiteru, Angel Goni-Moreno, and Anil Wipat. "SynBioHub: a standards-enabled design repository for synthetic biology." ACS synthetic biology 7, no. 2 (2018): 682-688.

Galdzicki, Michal, Kevin P. Clancy, Ernst Oberortner, Matthew Pocock, Jacqueline Y. Quinn, Cesar A. Rodriguez, Nicholas Roehner et al. "The Synthetic Biology Open Language (SBOL) provides a community standard for communicating designs in synthetic biology." Nature biotechnology 32, no. 6 (2014): 545-550.

Venkat, Sumana, Jourdan Sturges, Alleigh Stahman, Caroline Gregory, Qinglei Gan, and Chenguang Fan. "Genetically incorporating two distinct post-translational modifications into one protein simultaneously." ACS synthetic biology 7, no. 2 (2018): 689-695.

Dona, Malathi SI, Luke A. Prendergast, Suresh Mathivanan, Shivakumar Keerthikumar, and Agus Salim. "Powerful differential expression analysis incorporating network topology for next-generation sequencing data." Bioinformatics 33, no. 10 (2017): 1505-1513.

Fontanarrosa, Pedro, Hamid Doosthosseini, Amin Espah Borujeni, Yuval Dorfan, Christopher A. Voigt, and Chris Myers. "Genetic circuit dynamics: Hazard and Glitch analysis." ACS Synthetic Biology 9, no. 9 (2020): 2324-2338.

Buecherl, Lukas, Riley Roberts, Pedro Fontanarrosa, Payton J. Thomas, Jeanet Mante, Zhen Zhang, and Chris J. Myers. "Stochastic Hazard Analysis of Genetic Circuits in iBioSim and STAMINA." ACS Synthetic Biology 10, no. 10 (2021): 2532-2540.

7 ACRONYM LIST

ACS	American Chemical Society
aTc	Anhydrotetracycline
API	Application Programming Interface
ATP	Adenosine Triphosphate
AutoF	Autofluorescence
BO	Bayesian Optimization
bp	Base Pairs
BU	Boston University
CDC	Centers for Disease Control and Prevention
CMA-ES	Covariance Matrix Adaptation Evolution Strategy
CNN	Convolutional Neural Network
CRISPR	Clustered Regularly Interspaced Short Palindromic Repeats
DARPA	Defense Advanced Research Projects Agency
DeSEQ	Differential gene expression analysis (Packge in R)
DNA	Deoxyribonucleic Acid
DNN	Deep Neural Network
DOE	Department of Energy
DSGRN	Dynamic Signatures Generated by Regulatory Networks
FBA	Flux Balance Analysis
FDR	false discovery rate
FPKM	fragments per kilobase per million mapped fragments
GEO	Gene Expression Omnibus
GFP	Green Fluorescent Protein
GO categories	Gene Ontology categories
GUI	Graphical User Interface
HlyIIR	name of a transcriptional regulator
IcaR	name of a transcriptional regulator
IPTG	Isopropyl β-D-1-thiogalactopyranoside
iRF	Iterative Random Forests
JSON	Javascript Object Notation
KEGG	Kyoto Encyclopedia of Genes and Genomes
LacI	Lipoprotein-Associated Coagulation Inhibitor
LLD	Low-Level Detection
LSTM	Long short-term memory
MIC	Minimal Inhibitory Concentration
MIT	Massachusetts Institute of Technology
ML	Machine Learning
mM	millimolar
MOC	Massachusetts Open Cloud

mRNA	messenger Ribonucleic Acid
NCBI	National Center for Biotechnology Information
NIST	National Institute of Standards and Technology
ODE	Ordinary Differential Equation
PCR	polymerase chain reaction
pCym	name of a Cumate-inducible promoter
PhlF	name of a transcriptional regulator
PI	Principal Investigator
PLV	posterior likelihood variance
pTac	name of an inducible promoter
RBS	Ribosome Binding Site
RIT	Random Intersection Trees
RNA	Ribonucleic Acid
RNAP	Ribonucleic Acid Polymerase
RPU	Relative Promoter Unit
RTE	Relative Translation Efficiency
RT-PCR	reverse transcription-polymerase chain reaction
SAIL	Software & Application Innovation Lab
SD2	Synergistic Design 2
SBOL	Synthetic Biology Open Language
SOP	Standard Operating Procedures
TACC	Texas Advanced Computing Center
TDA	Topological Data Analysis
TMM	Trimmed Mean of M values
TCA cycle	Tricarboxylic Acid cycle
UCF	User-Constraints-File
UI/UX	User Interface / User Experience
UW	University of Washington
YFP	Yellow Fluorescent Protein