RESEARCH ARTICLE SUMMARY

SYNTHETIC BIOLOGY

Genetic circuit design automation

Alec A. K. Nielsen, Bryan S. Der, Jonghyeon Shin, Prashant Vaidyanathan, Vanya Paralanov, Elizabeth A. Strychalski, David Ross, Douglas Densmore, Christopher A. Voigt*

INTRODUCTION: Cells respond to their environment, make decisions, build structures, and coordinate tasks. Underlying these processes are computational operations performed by networks of regulatory proteins that integrate signals and control the timing of gene expression. Harnessing this capability is critical for biotechnology projects that require decision-making, control, sensing, or spatial organization. It has been shown that cells can be programmed using synthetic genetic circuits composed of regulators organized to generate a desired operation. However, the construction of even simple circuits is time-intensive and unreliable.

ON OUR WEBSITE

Read the full article at http://dx.doi.org/10.1126/science.aac7341

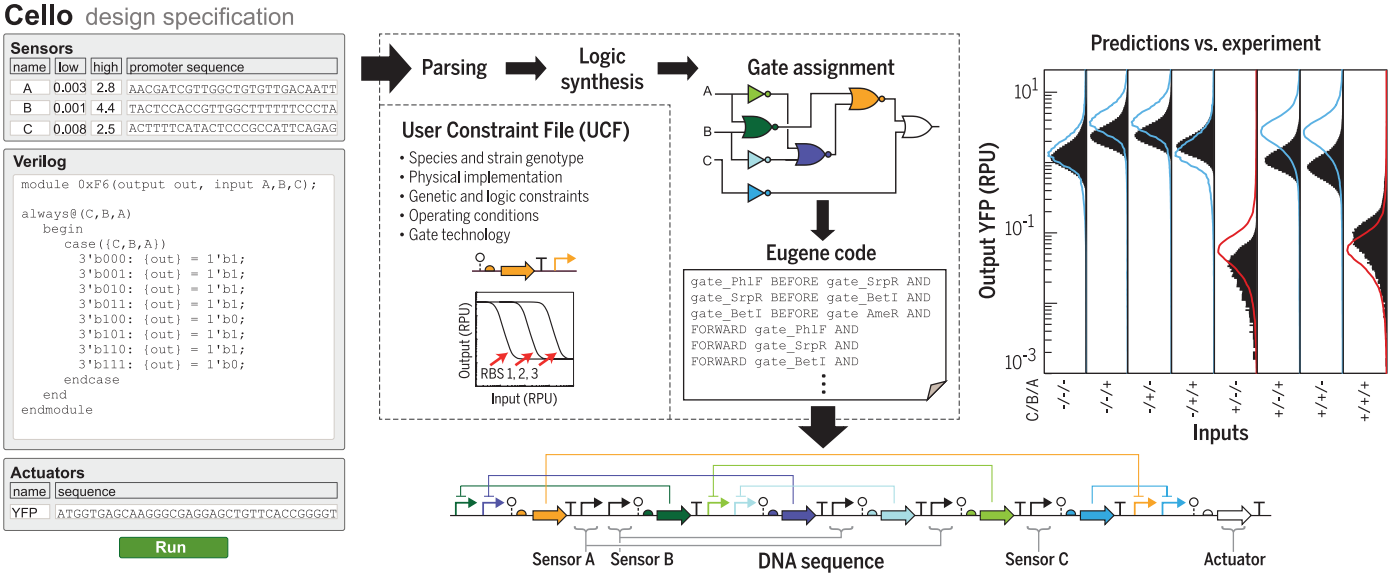
RATIONALE: Electronic design automation (EDA) was developed to aid engineers in the design of semiconductor-based electronics. In an effort to accelerate genetic circuit design, we applied principles from EDA to enable increased circuit complexity and to simplify the incorporation of synthetic gene regulation into genetic engineering projects. We used the hardware description language Verilog to enable a user to describe a circuit function. The user also specifies the sensors, actuators, and "user constraints file" (UCF), which defines the organism, gate technology, and valid operating conditions. Cello (www.cellocad.org) uses this information to automatically design a DNA sequence encoding the desired circuit. This is done via a set of algorithms that parse the Verilog text, create the circuit diagram, assign gates, balance constraints to build the DNA, and simulate performance.

RESULTS: Cello designs circuits by drawing upon a library of Boolean logic gates. Here, the gate technology consists of NOT/NOR logic based on repressors. Gate connection is simplified by defining the input and output signals as RNA polymerase (RNAP) fluxes. We found that the gates need to be insulated from their genetic context to function reliably in the context of different circuits. Each gate is isolated using strong terminators to block RNAP leakage, and input interchangeability is improved using ribozymes and promoter spacers. These parts are varied for each gate to avoid breakage due to recombination. Measuring the load of each gate and incorporating this into the optimization algorithms further reduces evolutionary pressure.

Cello was applied to the design of 60 circuits for *Escherichia coli*, where the circuit function was specified using Verilog code and transformed to a DNA sequence. The DNA sequences were built as specified with no additional tuning, requiring 880,000 base pairs of DNA assembly. Of these, 45 circuits performed correctly in every output state (up to 10 regulators and 55 parts). Across all circuits, 92% of the 412 output states functioned as predicted.

CONCLUSION: Our work constitutes a hardware description language for programming living cells. This required the co-development of design algorithms with gates that are sufficiently simple and robust to be connected by automated algorithms. We demonstrate that engineering principles can be applied to identify and suppress errors that complicate the compositions of larger systems. This approach leads to highly repetitive and modular genetics, in stark contrast to the encoding of natural regulatory networks. The use of a hardware-independent language and the creation of additional UCFs will allow a single design to be transformed into DNA for different organisms, genetic endpoints, operating conditions, and gate technologies. ■

The list of author affiliations is available in the full article online.
*Corresponding author. E-mail: cavoigt@gmail.com
Cite this article as A. A. K. Nielsen *et al.*, *Science* **352**, aac7341 (2016). DOI: 10.1126/science.aac7341



Genetic programming using Cello. A user specifies the desired circuit function in Verilog code, and this is transformed into a DNA sequence. An example circuit is shown (0xF6); red and blue curves are predicted output states for populations of cells, and solid black distributions are experimental flow cytometry data. The outputs are shown for all combinations of sensor states; plus and minus signs indicate the presence or absence of input signal. RBS, ribosome binding site; RPU, relative promoter unit; YFP, yellow fluorescent protein.

RESEARCH ARTICLE

SYNTHETIC BIOLOGY

Genetic circuit design automation

Alec A. K. Nielsen,[1] Bryan S. Der,[1,2] Jonghyeon Shin,[1] Prashant Vaidyanathan,[2] Vanya Paralanov,[3] Elizabeth A. Strychalski,[3] David Ross,[3] Douglas Densmore,[2] Christopher A. Voigt[1]*

Computation can be performed in living cells by DNA-encoded circuits that process sensory information and control biological functions. Their construction is time-intensive, requiring manual part assembly and balancing of regulator expression. We describe a design environment, Cello, in which a user writes Verilog code that is automatically transformed into a DNA sequence. Algorithms build a circuit diagram, assign and connect gates, and simulate performance. Reliable circuit design requires the insulation of gates from genetic context, so that they function identically when used in different circuits. We used Cello to design 60 circuits for *Escherichia coli* (880,000 base pairs of DNA), for which each DNA sequence was built as predicted by the software with no additional tuning. Of these, 45 circuits performed correctly in every output state (up to 10 regulators and 55 parts), and across all circuits 92% of the output states functioned as predicted. Design automation simplifies the incorporation of genetic circuits into biotechnology projects that require decision-making, control, sensing, or spatial organization.

Electronic design automation (EDA) software tools aid engineers in the design and analysis of semiconductor-based electronics. Prior to EDA, integrated circuit design was a manual process performed by hand. This was accelerated by the development of hardware description languages (e.g., Verilog) that enabled a user to design an electronic system through textual commands that are transformed into a circuit patterned on silicon. We applied this approach to genetic circuits, so that a Verilog design is transformed into a linear DNA sequence that can be constructed and run in living cells. The design environment, referred to as Cello (an abbreviation of Cellular Logic), implements algorithms that derive the detailed physical design from the textual specification (Fig. 1). Cello requires genetic logic gates that are sufficiently modular and reliable such that their interconnection can be automated.

Moving computing into cells enables programmable control over biological functions (*1*–*5*). This is crucial for fully realizing the potential of engineering biology, where applications require that different sets of genes be active under different conditions (*6*–*9*). Cells are naturally able to respond to their environment, make decisions, construct intricate structures, and coordinate to distribute tasks. These functions are controlled by a regulatory network of interacting proteins, RNA, and DNA. Patterns of such interactions generate computational operations analogous to those used in electronic circuits (*10*–*14*), and regulators can be combined to build synthetic genetic circuits (*15*–*18*). This approach has led to digital logic gates (*19*–*24*), memory devices (*25*–*29*), analog computation (*30*), and dynamic circuits (e.g., timers and oscillators) (*15*, *16*, *31*–*33*). These have begun to be integrated into biotechnological applications (*34*, *35*)—for example, to implement feedback control in a metabolic pathway (*36*). However, the construction of simple circuits consisting of only a few regulators remains a time-consuming task, and this has limited their widespread implementation.

Genetic circuit design is challenging for several reasons (*37*, *38*). First, circuits require precise balancing of regulator expression (*22*, *39*). Second, many parts are combined to build a circuit and their function can vary depending on genetic context, strain, and growth conditions (*40*–*46*). Third, circuits are defined by many states (their response to different inputs or how they change over time), and this can be cumbersome to characterize (*15*, *47*–*49*). Finally, many regulators are toxic when overexpressed, and even mild effects can combine to drive negative selection against the circuit (*50*). Balancing these issues is difficult to do by hand. Thus, computational tools have been developed for the study of natural networks and to aid circuit design by predicting how parts or devices will perform when connected (*51*–*58*).

We developed Cello to accelerate circuit design, allow increased complexity of circuits, and enable non-experts to incorporate synthetic gene regulation into genetic engineering projects (Fig. 1). The focus is on the design of a circuit that performs a desired computational operation, which connects to cell-based sensors and cellular functions (actuators). A user provides three specifications to Cello. The first are the DNA sequences for the sensors: the sequences of their output promoters and data for their ON/OFF signal strengths in standardized units (see below) (*59*). The second is the "user constraints file" (UCF), which contains the functional details of the gate library, the layout of the genetic system, the organism and strain, and the operating conditions for which the circuit design is valid. The third is Verilog code that captures the desired computational operation. Cello uses this information to automatically design a DNA sequence encoding the desired genetic circuit by connecting a set of simpler gates that implement Boolean logic to the sensors and each other. The output of the circuit can be connected to cellular processes by directing the output promoter to control a cellular function (e.g., a metabolic pathway), either directly or through an intermediate (e.g., a phage RNA polymerase) (*60*, *61*). The sensors, circuit, and actuator are inserted into specific genetic locations and transformed into a strain, both of which are defined in the UCF (Fig. 1B).

Cello builds circuits by connecting transcriptional gates, whose common signal carrier is RNA polymerase (RNAP) flux on DNA (*62*). This conversion allows gates to be layered by having the output promoter from one gate serve as the input to the next. This modularizes the design, so that a circuit is defined by a pattern of promoters in front of regulators on a linear DNA strand (Fig. 2A). Within this paradigm, the regulators performing the gate biochemistry could be transcription factors (*21*, *63*), RNA-based regulation (*64*–*66*), protein-protein interactions (*19*, *67*), CRISPR/Cas-based regulation (*68*–*71*), or recombinases (*25*, *26*, *72*).

We developed a set of insulated NOT and NOR gates based on prokaryotic repressors (*21*). These repressor-based gates were characterized in isolation as NOT gates. To facilitate the connection of gates and sensors, we adopted the BBa_J23101 constitutive promoter as a standard (*59*). The output of an insulated version of this promoter (*73*) is defined as 1 RPU (relative promoter unit); working with National Institute of Standards and Technology (NIST) collaborators, this output was measured to correspond to 24.7 ± 5.7 mRNAs per cell, which is approximately 0.02 RNAP s^{-1} promoter^{-1} (fig. S33). These data were used by Cello to automatically generate a large set of circuits. The sequences were built as specified by the software output with no additional tuning, which facilitates the iterative improvement of the quality of the gates and design rules.

Cello design environment

Verilog is a commonly used hardware description language for electronic system design (*74*). It is hardware-independent, meaning that a circuit can be described by abstract textual commands and then transformed into different physical implementations (i.e., chip types). Verilog is often accompanied by a simulation package that aids the evaluation of a design in silico before building the system. Verilog code has a hierarchical organization centered on modules that communicate

[1]Synthetic Biology Center, Department of Biological Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139, USA. [2]Biological Design Center, Department of Biomedical Engineering, Department of Electrical and Computer Engineering, Boston University, Boston, MA 02215, USA. [3]Biosystems and Biomaterials Division, Material Measurement Laboratory, National Institute of Standards and Technology, Gaithersburg, MD 20817, USA.
*Corresponding author. E-mail: cavoigt@gmail.com

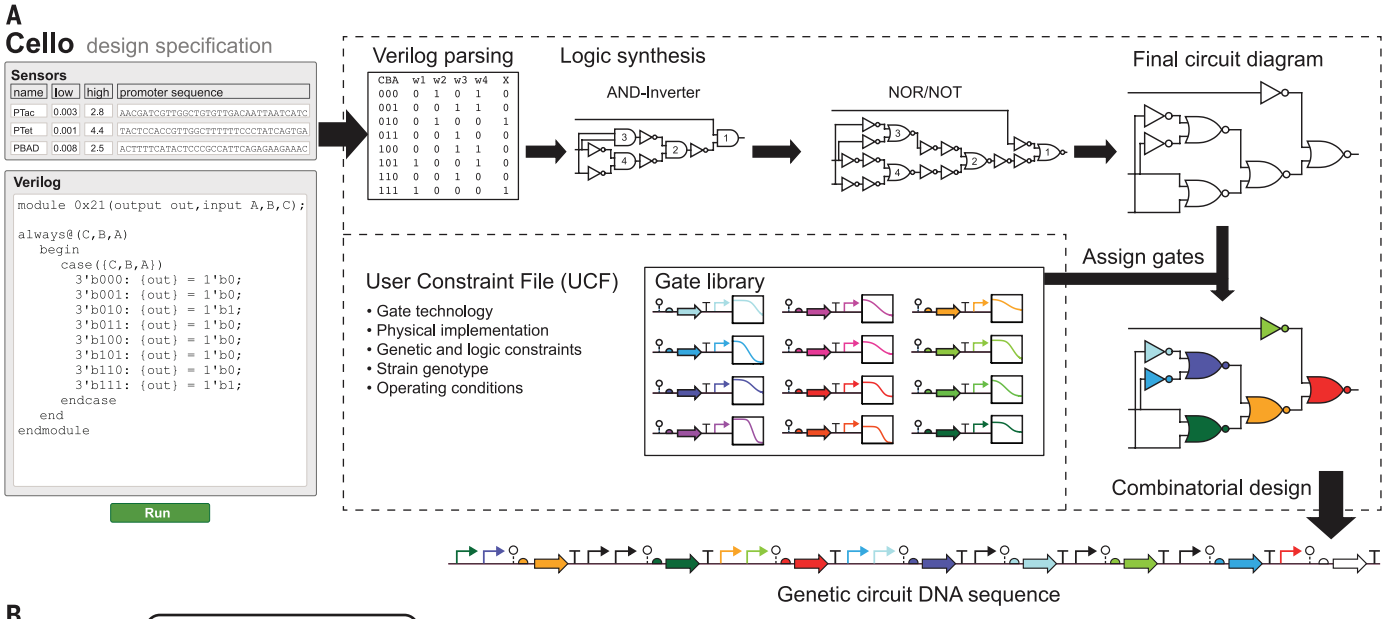


Fig. 1. Overview of Cello. (**A**) Cello users write Verilog code and select or upload sensors and a UCF. On the basis of the Verilog design, a truth table is constructed from which a circuit diagram is synthesized. Regulators are assigned from a library to each gate (each color is a different repressor). Combinatorial design is then used to concatenate parts into a linear DNA sequence. SBOL Visual (*101*) is used for the part symbols. Raised arrows are promoters, circles on stems are ribozyme insulators, hemispheres are RBSs, large arrows are protein-coding sequences, and "T"s are terminators. Part colors correspond to physical gates. (**B**) The physical specification for the Eco1C1G1T1 UCF. The circuit and sensors are inserted into one plasmid; the other plasmid contains the circuit output promoter, which can be used to drive the expression of a fluorescent protein or other actuator. Both plasmids must be present in the specified strain for the design to be valid.

through wires to propagate signals. In our implementation, circuit function can be defined by Case, Assign, or Structural statements within modules (fig. S23). Initially, our focus with Cello is on the creation of asynchronous combinational logic without feedback. This is useful in the design of genetic circuits that can process multiple environmental sensors in order to choose among different cellular functions. However, Verilog provides the framework to extend the designs to include more complex circuits, including those with specified timing and signal strengths as well as analog (Verilog-AMS) functions.

The philosophy behind Cello is to generate circuits for highly specified physical systems and operating conditions. This paradigm is captured by the UCF, which specifies (i) the gate technology, including DNA sequences and functional data; (ii) defined physical locations for the circuit (e.g., plasmid or genomic locus); (iii) the organism, strain, and genotype; (iv) operating conditions where the circuit design is valid; (v) architectural rules to constrain the part arrangement; and (vi) preferred logic motifs to be incorporated during logic synthesis.

The UCF follows the JSON (JavaScript Object Notation) standard (*75*), which is both human- and machine-readable and is convertible with SBOL (Synthetic Biology Open Language) (*76*). We developed the Eco1C1G1T1 UCF (data file S1) for *E. coli* (NEB 10-beta) and gate technology based on a set of 12 prokaryotic repressors (*21*). The development of additional UCFs would enable a circuit design to be transferred to other organisms, conditions, or gate technologies.

When a user selects a UCF and synthesizes a circuit from Verilog code, the corresponding DNA sequence is designed in three steps (Fig. 1). First, the textual commands are converted to a circuit diagram. Algorithms parse the Verilog code and derive a truth table (fig. S23), which is converted to an initial circuit diagram by the logic synthesis program ABC (*77*) and subsequently modified to only contain logic operations for gates available in the UCF (fig. S25). The second step is to assign specific regulators to each gate in the diagram. Functionally connecting gates requires that the outputs from the first gate span the input threshold of the second gate (Fig. 2B). Because gates based on different regulators have different response functions, not all pairs can be functionally connected (Fig. 2C). Identifying the optimal assignment is an NP-complete problem (Fig. 2D) (*78*–*81*). We implemented a Monte Carlo simulated annealing algorithm to rapidly identify an assignment that produces the desired response (Fig. 2E and fig. S28). The third step is to create the linear DNA sequence based on the circuit diagram and gate assignment. The assignment is converted into a set of parts and constraints between the parts [written with the Eugene language (*82*)]. The UCF can also include additional constraints on the genetic architecture—for example, to forbid a particular combination of parts. A combinatorial design algorithm (*83*) is used to build a genetic construct that conforms to the constraints (fig. S29). This allows a user to design multiple constructs containing the same circuit function and genetic constraints while varying

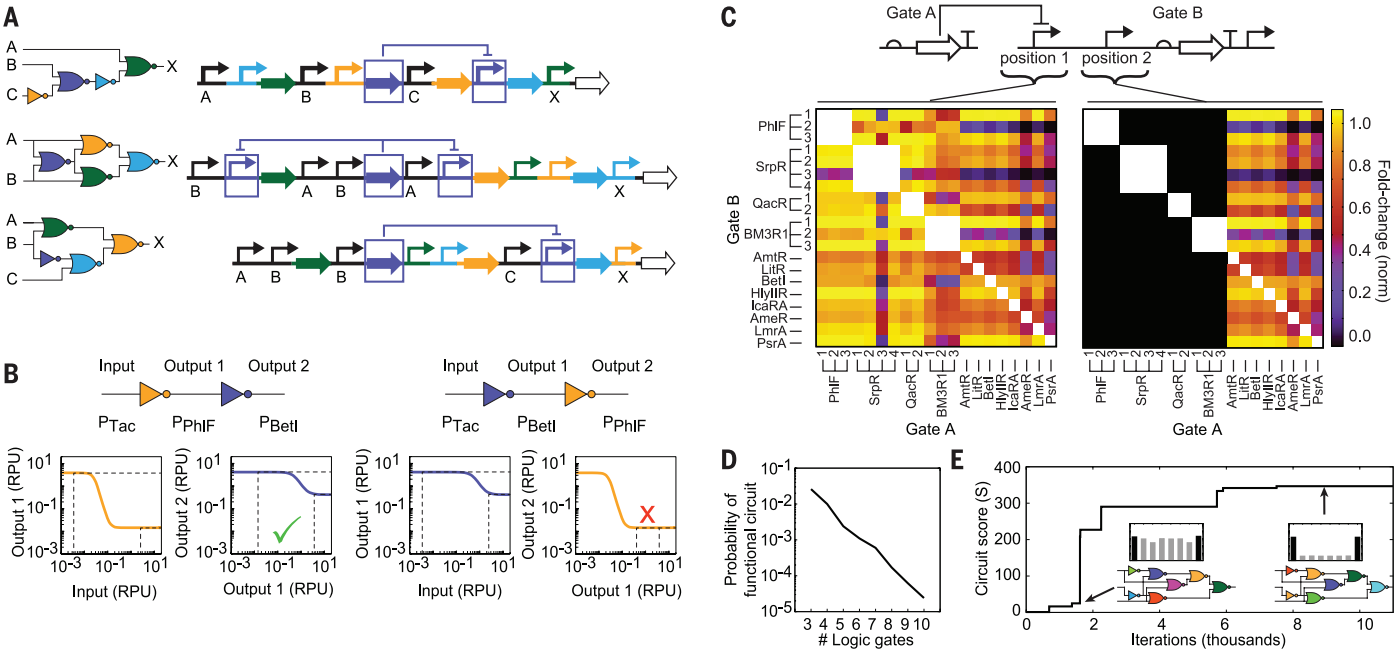


Fig. 2. Assignment of genetic gates to the circuit diagram. (**A**) A set of four gates based on different repressors (colors) connected in various permutations to build different circuit functions. The inputs (A, B, and C) are sensor input promoters; the circuit output promoter (X) controls the actuating gene. (**B**) The shapes of the gate response functions determine whether they can be functionally connected. The orange gate (PhlF) has a large dynamic range (dashed lines) that spans the threshold of the purple gate (BetI). However, in the reverse order, the gates do not functionally connect. (**C**) Combinatorial relations of repressors from the insulated gate library are shown in the upstream (gate A) and downstream (gate B) positions. Color scale at right indicates relative change (normalized), calculated as the maximum output range that can be achieved by connecting gate A to gate B. Numbers denote different RBSs. The left and right graphs show when gate A regulates position 1 and position 2, respectively. Gates that are excluded from position 2 because of roadblocking are shown in black (fig. S9). (**D**) The probability of finding a functional circuit versus the number of logic gates. The probability of a functional circuit is defined as the likelihood that a random assignment passes input threshold analysis (fig. S31) and has no roadblocking combinations. (**E**) The convergence of the simulated annealing gate assignment algorithm (fig. S28). Bar graph insets: Black bars should be ON, gray bars should be OFF; the *y* axis is the output in RPU on a log scale, and the *x* axis is the input state (from left to right: 000, 001, 010, 011, 110, 101, 110, 111). The circuit score (*S*) is defined as the ratio of the lowest predicted ON state to the highest predicted OFF state (fig. S26 and eq. S2). An example search is shown for the circuit diagram in the insets; colors correspond to repressors assigned to each gate (Fig. 3B).

unconstrained design elements to build a library that can be screened.

Cello then simulates the performance of the genetic circuit. When flow cytometry data are provided in the UCF for the gates, this provides the cell-to-cell variation in the response for a population of cells. We developed a computational approach to quantify how population variability propagates from the sensors through the gates to the output promoters (fig. S30). Cello applies a simple algorithm to determine how signals propagate from the sensors through the gates to the output promoters. This generates predicted cytometry distribution for all combinations of input states, which can be directly compared to experiments. Finally, for each gate, the load on the cell for carrying the gates is estimated on the basis of their impact on growth [percent reduction of optical density at 600 nm (OD_{600})] as a function of the activity of the input promoter (*84*). For any combination of inputs, if the predicted growth reduction exceeds a threshold, this information can guide multiobjective circuit optimization or be provided as a warning to the user (fig. S27).

Initial gate assembly and failure modes

We constructed a gate library based on a set of 16 Tet repressor (TetR) homologs that are orthogonal; that is, they do not bind to each other's promoters (*21*). These can be converted into simple NOT/NOR gates by having the input promoter(s) drive the expression of the repressor, after which there is a terminator and output promoter. Because of a lack of strong terminators when these gates were built, the same terminator (BBa_B0015) was reused for each one. Each repressor had a different ribosome binding site (RBS), chosen to maximize the dynamic range.

These gates could be connected to form simple functional circuits; however, in each case additional tuning was required and the dynamic range of the output was low (*21*). We tested the ability of the response functions of the gates to predict circuit behavior as a whole with no additional tuning. We designed a set of eight simple circuits from these gates that required between one and four repressors (Fig. 3A) (*84*). Nearly all of the circuits generated an incorrect response. Only the (A NIMPLY B) gate functioned properly, and six of eight circuits had their output states either all OFF or all ON for every input condition. Across all the circuits, 13 of 32 output states were correct, which is comparable to what would be expected from a process that generates random outputs.

We used this test set to determine common causes for circuit failure (figs. S6 to S10). When paired with different promoters, gates often generated an unpredictable response, and this was apparent even for circuits based on a single repressor (*42*). The promoters generated transcripts with different 5′-untranslated regions (5′UTRs), which can strongly influence gene expression (*40*, *85*). A second problem was that some promoters in the downstream "position 2" of a NOR gate (Fig. 2C) can reduce transcription from the upstream promoter, a phenomenon we refer to as "roadblocking." Third, some circuits had growth defects, which were caused by repressors that become toxic when expressed past a threshold (Fig. 3D). Fourth, several circuits were genetically unstable because of homologous recombination of parts reused in the same circuit (*86*, *87*).

Insulated gates

A second generation of gates was constructed to address the observed failure modes (Fig. 3B). Changes took two forms: (i) New parts were added to gates to insulate them from genetic context, and (ii) rules were included in the UCF that disallow certain parts, positions, and part combinations that lead to unpredictable behavior. Transcriptional insulation was achieved for gates by adding a different strong terminator with sufficiently diverse sequences to avoid homologous recombination

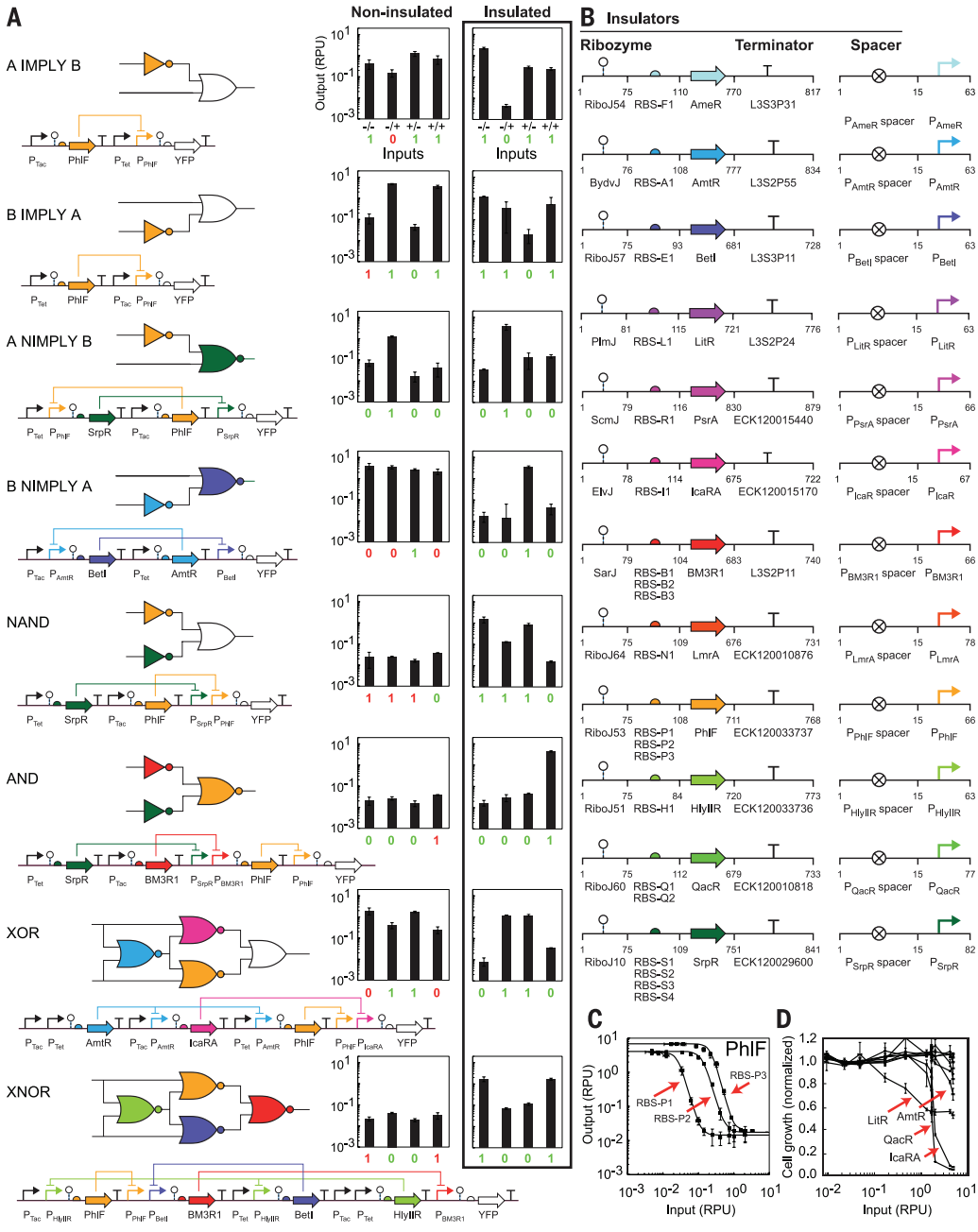


Fig. 3. Impact of gate insulation. (**A**) The logic function, circuit diagram, and DNA construct are shown for each genetic circuit. Only the insulated circuit schematics are shown; the equivalent information for the non-insulated circuits is shown in fig. S5. The expected output for each circuit is shown at the bottom of each bar graph as 1 for ON and 0 for OFF. The numbers are colored according to whether the state is predicted correctly (green) or incorrectly (red). For noninsulated circuits, inputs correspond to the absence or presence of 1 mM isopropyl-β-D-thiogalactopyranoside (IPTG; right −/+) and 20 μM 3-oxohexanoyl-homoserine lactone (3OC6HSL; left −/+). For insulated circuits, inputs correspond to the absence or presence of 1 mM IPTG (right −/+) and aTc (2 ng/ml; left −/+) (*84*). (**B**) The architectures of the insulated gates. Some gates have multiple versions with different RBS sequences. The gate DNA sequences are provided in table S8. (**C**) An example of a response function for a NOT gate based on the PhlF repressor. The change in the threshold for the three RBSs is shown. Data for all insulated gates are shown in fig. S4. (**D**) The impact of each gate on cell growth as a function of its input promoter activity. Cell growth was measured as OD_{600} and normalized by the growth of the no-inducer control 6 hours after induction (*84*). The four gates that reduced growth by >20% are indicated. Error bars are one SD of normalized cell growth [*y* axis in (D)] and the median [*y* axis in (A) and (C); *x* axis in (C) and (D)] for three independent experiments performed on the same day.

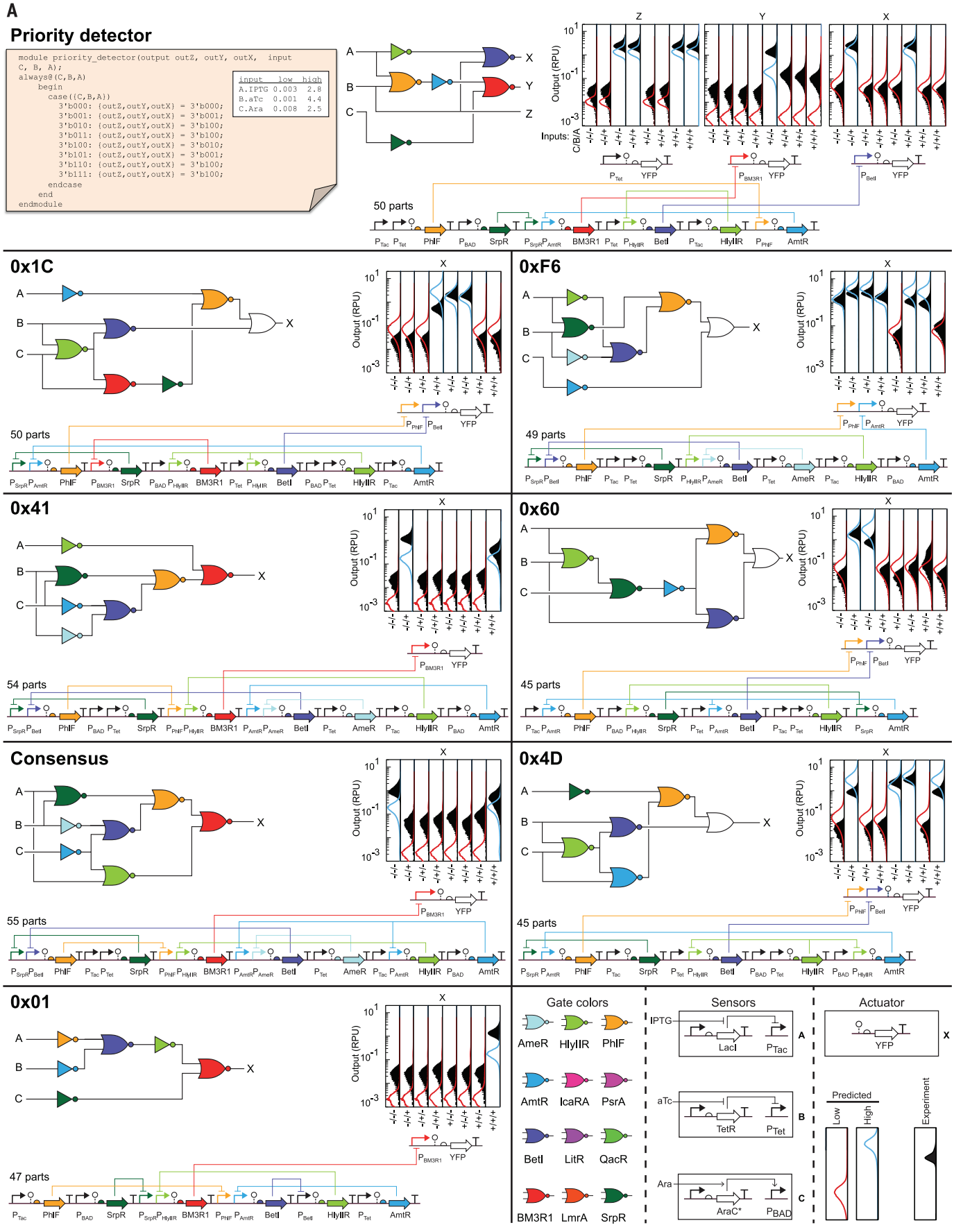
(table S2) (*87*). The output promoters were also insulated on both sides from changes to their upstream and downstream context. Insulators consisting of a hammerhead ribozyme and downstream hairpin (RiboJ) ensure that a promoter generates the same response function irrespective of the downstream gene (*42*). As with the terminators, to avoid recombination we had to create a library of RiboJ variants that are functionally identical but sequence-diverse so that each gate had a unique insulator sequence (fig. S3 and table S1). To insulate the promoter from the upstream sequence, we added 15 nucleotides (nt) of randomly generated DNA to extend the promoters to −50 so as to include regions with an impact on strength (table S8) (*88*). Finally, the propensity for repressible promoters to roadblock was measured (fig. S9) and these data were used to create Eugene rules in the UCF that disallow these promoters from position 2 in NOR gates (Fig. 2C).

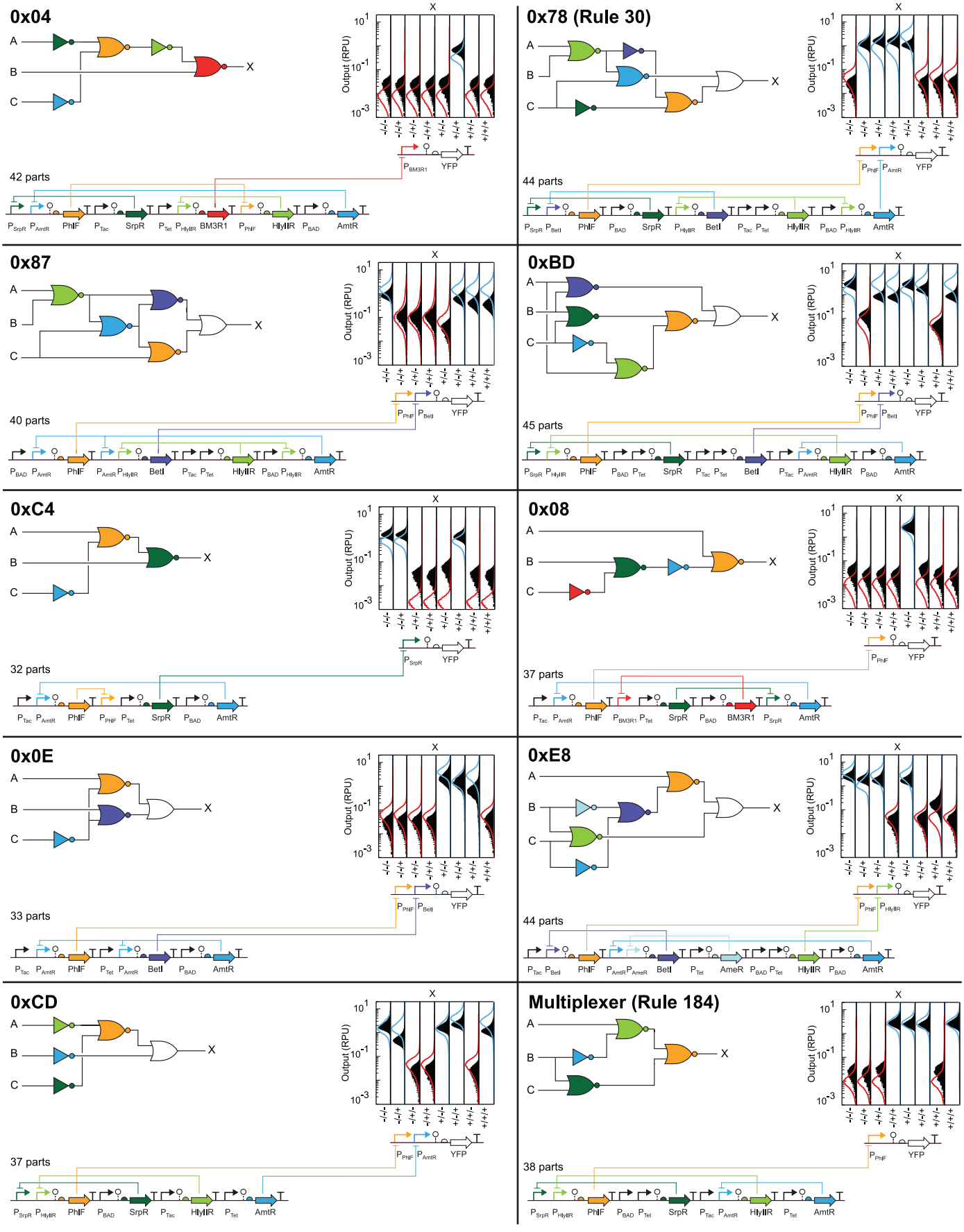
The response functions were then experimentally measured for all the gates (Fig. 3C, fig. S4, and table S4). Several of the first-version gates had response functions that were difficult to connect functionally to sensors or other gates (Fig. 2B). To increase the likelihood of finding a connection, we made versions of the gates with different RBSs that shift the response threshold (table S3). The growth impact of each gate was then measured as a function of the input promoter activity to determine whether there is a toxicity threshold that should be avoided (Fig. 3D and fig. S10). To eliminate toxic or cross-reacting repressors, we reduced the original set of 16 TetR homologs to 12. Only four of these caused a growth defect at high inputs, and this could be avoided by the assignment algorithm (fig. S27).

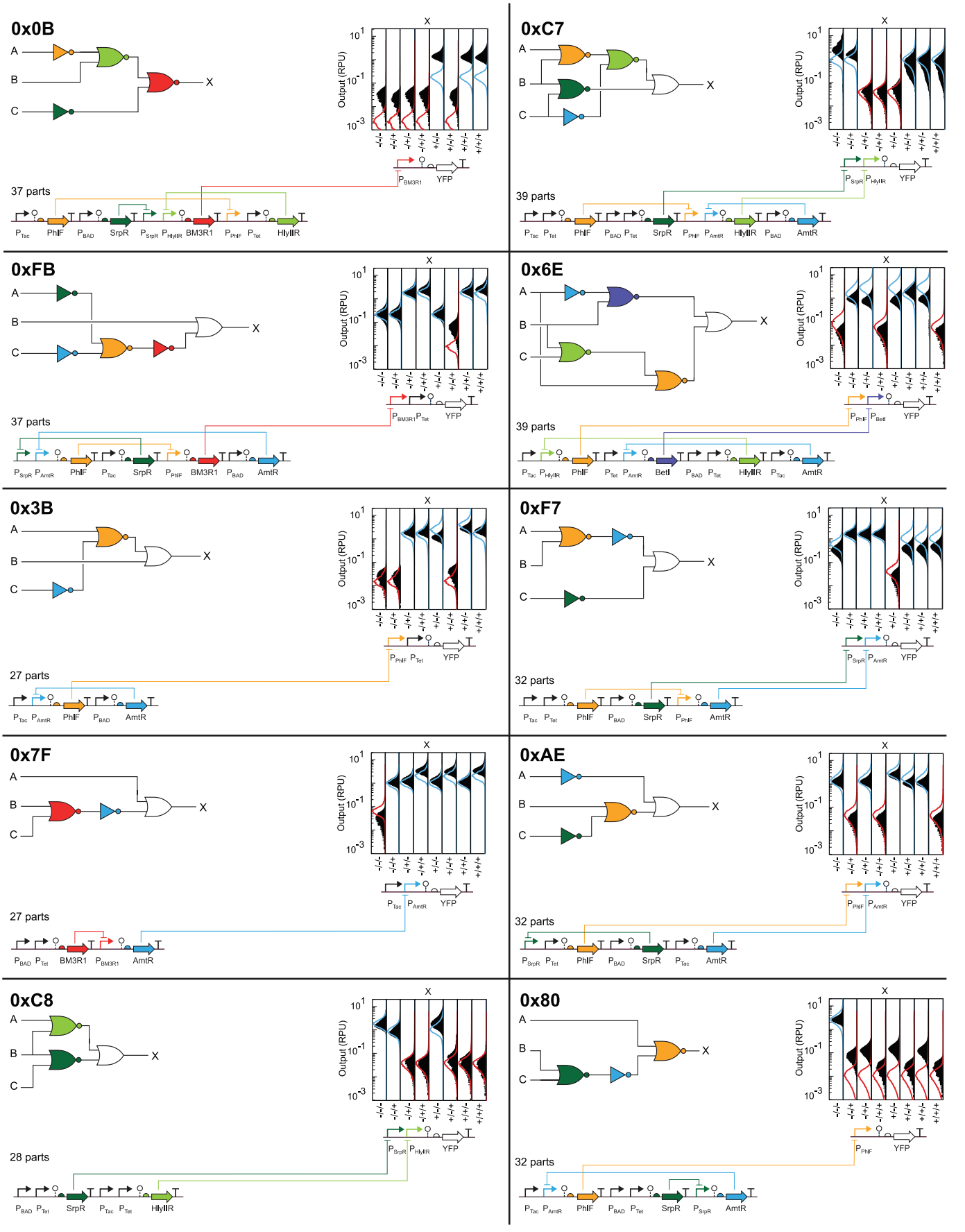
The eight simple circuits were redesigned with the new gate library (*84*). The sequences were constructed as designed, with no post-design tuning. All of the circuits functioned correctly, corresponding to a total of 32/32 correct output states (Fig. 3A).

Circuit design automation using Cello

Cello was used to design a large set of 52 additional circuits based on the insulated gates (Fig. 4) (*84*). These circuits include a Priority Detector (that prioritizes the inputs and selects which output is ON based on the highest-priority input that is ON), well-known functions (e.g., a multiplexer), and logic underlying cellular automaton pattern







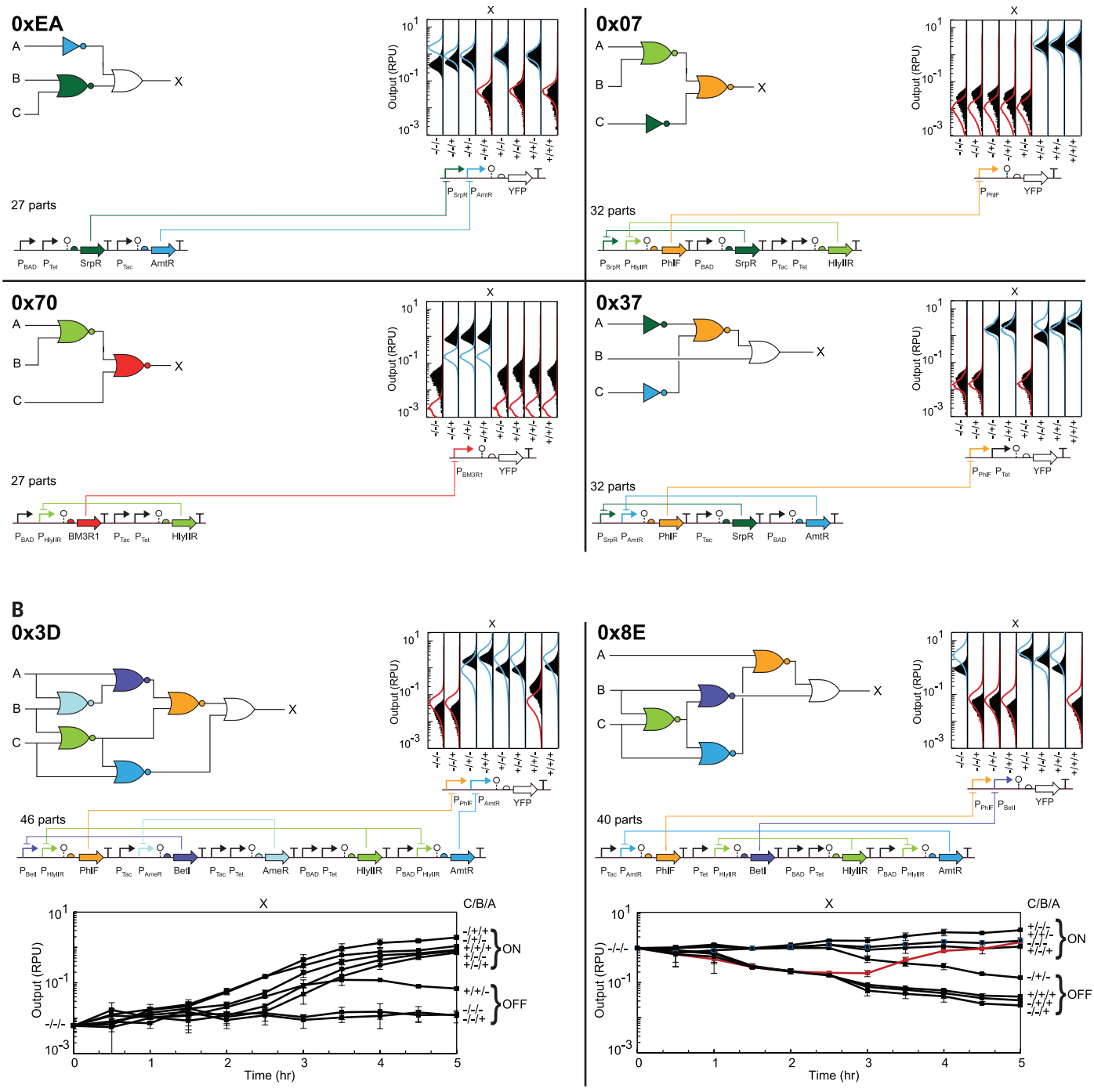


Fig. 4. Automated design of circuits by Cello. (**A**) An example of the code along with the input states (in RPU) is shown for the Priority Detector circuit. All circuits designed by Cello that had correct output states are shown along with their genetic schematics, output predictions, and experimental measurements. The inputs A, B, and C correspond to the P_{Tac}, P_{Tet}, and P_{BAD} sensor promoter activities; their corresponding regulators (LacI, TetR, and AraC*) are not shown in the schematics. The outputs (X, Y, and Z) correspond to YFP driven from output promoters in separate experiments. Solid black distributions are experimental data; blue and red line distributions are computational predictions from Cello (fig. S30). The number of parts for each circuit includes all functional DNA parts in the circuit (promoters, ribozymes, RBSs, protein-coding sequences, and terminators), plus eight parts for the sensor block and two parts for the plasmid backbones. Inputs correspond to the absence or presence of 1 mM IPTG, aTc (2 ng/ml), and 5 mM L-arabinose (Ara). Replicates are provided in fig. S19. The DNA sequence for each circuit is provided in data file S2. When the circuit does not have a common name (e.g., Priority Detector), a hexadecimal naming system is used [e.g., 0x41 (*84*)]. The names starting with "Rule" refer to Wolfram's cellular automaton convention (*89*). (**B**) Time-course data are included for two circuits. The circuits are maintained in the −/−/− state for 3 hours before induction and then switched to the other eight possible states at time = 0 hours (*84*). Error bars are one SD of RPU median performed on three separate days.

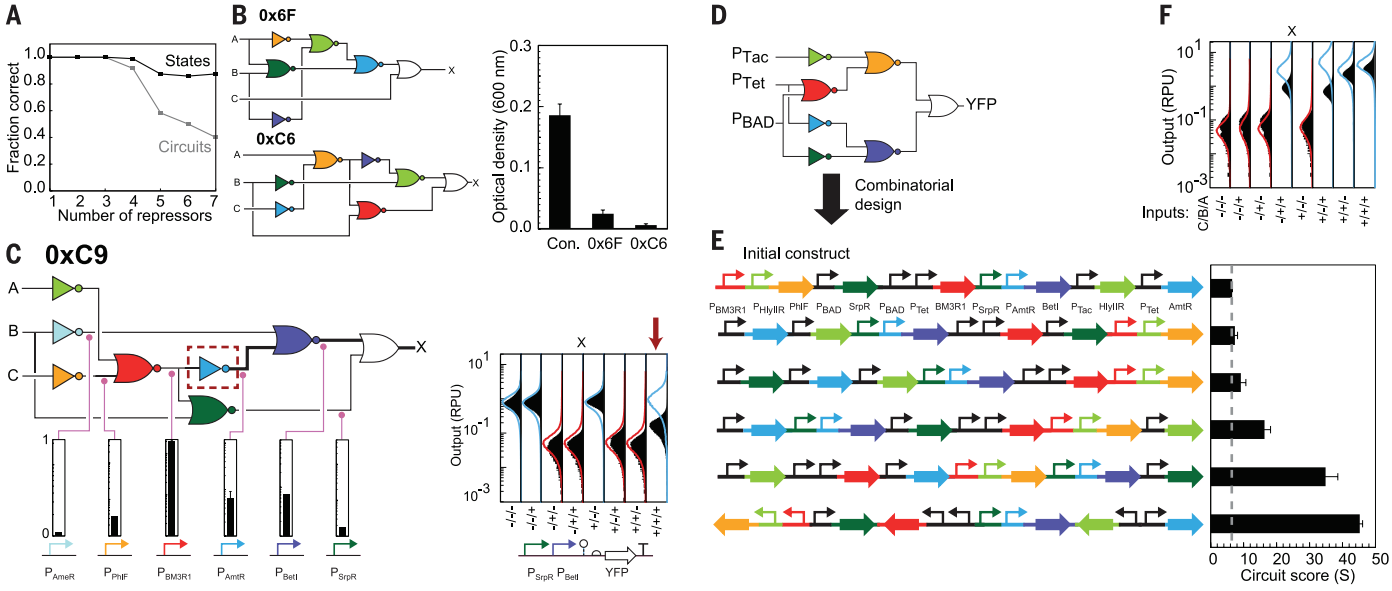


Fig. 5. Analysis of circuit failures and the design of multiple constructs by combinatorial design. (**A**) For the library of 60 circuits (Fig. 3A, Fig. 4, and figs. S12, S13, S14, and S17), the fraction of correct states (black) and the fraction of fully correct circuits (gray) are shown versus the number of repressors in the circuit. (**B**) Impact on cell growth for the two circuits that failed because of toxicity. The control bar is for cells containing the RPU standard plasmid only. Data are average values from three experiments performed on different days; error bars denote SD. (**C**) An example of circuit debugging. All combinations of inputs for all wires were tested; for clarity, only a subset of debugging for the failed state (+/+/+) is shown. The data are normalized to [0, 1] to correct for the dynamic range across gates. In this case, the failure originated when the AmtR gate produced an intermediate response that then propagated through the circuit. (**D**) The circuit diagram for a Majority circuit. Colors correspond to repressors. (**E**) Six layouts for this circuit were designed that maintain the same repressor assignments but allow the order and orientation of the gates to vary. The circuit score (*S*) is defined as the ratio of the lowest ON state's median to the highest OFF state's median. Error bars are one SD for two experiments performed on different days. Cytometry distributions for each design are shown in fig. S15. The dashed line marks the lowest circuit score in the library. (**F**) Predictions and cytometry distributions for the final design. The format and inducer concentrations are as described in Fig. 4. The red line is a transient fault state.

formation (e.g., "Rule 30") (*89*). Additional three-input, one-output logic circuits were built that demonstrate the ability to integrate inputs in different ways. These circuits could be used to turn a cellular function on or off in response to an environment defined by multiple signals. Each of the 52 circuits was specified either by using behavioral Verilog (with Cello performing the logic minimization step) or by performing a separate enumeration to identify the global minimum number of gates and specifying the circuit diagram using structural Verilog (*90*). Subsequently, the global minimum three-input logic gates were included in the UCF so that they could be incorporated as motifs in larger circuits in future designs (fig. S25). For each circuit, the sensor promoters and ON/OFF values were specified, the Eco1C1G1T1 UCF was selected, and a DNA sequence was automatically generated by Cello. DNA synthesis (*91*) and assembly were used to build each sequence; sequences contained up to 10 regulators and 55 parts. The output states of each circuit were measured by flow cytometry and compared with the Cello predictions. No additional tuning was done to diverge from the Cello-predicted sequence.

Of these 52 circuits, 37 functioned as predicted, such that all of the output states matched desired ON and OFF levels (Fig. 4A). Further, the predicted cytometry distributions closely matched those measured experimentally. Of 412 output states across all circuits we built, 92% were correct. The Consensus circuit (output is ON only when all three inputs agree) is the largest, containing 10 regulatory proteins (seven repressors from NOT/NOR gates and three from the inducible systems) and 55 genetic parts. Two of the circuits with four layers (0x3D and 0x8E) were selected to characterize the switching dynamics between states (Fig. 4B). Interestingly, 0x8E shows a transient incorrect state, known as a "fault" in electronics, when the inputs are changed from −/−/− to +/−/+. This is consistent with the last NOR gate transiently receiving ON/OFF inputs until one of the signals transits two layers.

Of the 52 circuits, seven were incorrect in one state, two were incorrect in two states, and five had ≥3 failures (figs. S12 to S14 and fig. S20). As more gates were included in a design, there was a higher probability of failure (Fig. 5A). Two circuits were found to cause a growth defect (Fig. 5B). The circuits that failed in a few states tended to match the remaining states closely, so the initial design could be used as the basis for further rounds of optimization. To this end, debugging experiments were performed to determine which gates failed. This was done by creating a set of plasmids that contained each gate's output promoter fused to yellow fluorescent protein (YFP). These plasmids were transformed with the circuit in lieu of the output plasmid, and the response of the internal gate was measured for all combinations of inputs. An example of this is shown in Fig. 5C; several other examples are shown in fig. S21. From this analysis, most of the circuit failures point to unexpected behavior from the anhydrotetracycline (aTc) sensor (seven circuits) or AmtR gate (two circuits).

Screening design variants has the potential to increase the probability of success, particularly for larger circuits. To do this, Cello outputs a Eugene file that contains architectural rules from the UCF as well as constraints to enforce the circuit diagram and repressor assignments (fig. S29). The user can specify the size of the library, and a combinatorial design algorithm (*83*) generates the target number of constructs. Although all of the systems should be functionally equivalent, subtle changes in their composition may affect circuit function through hidden effects (e.g., transcriptional read-through or promoter interference). We tested this approach by designing a Majority circuit (Fig. 5D), whose output is ON when a majority of its inputs are ON. We built a small library of six constructs that maintained the same circuit diagram and repressor assignments, but in which the order and orientation of genes was allowed to vary (Fig. 5E). Several of these circuits functioned correctly; the response of the best of these is shown in Fig. 5F.

Discussion

The design of synthetic regulatory networks has been dominated by manual trial-and-error tinkering at the nucleotide level. Cello automates the selection and concatenation of parts and balancing

the associated constraints. By doing this, it enables more rapid design of larger multipart systems; the circuits that we present here are larger and more complex than most that have been built by hand. Of 60 circuits designed automatically, 45 functioned as designed (Fig. 2A, Figs. 4 and 5, and fig. S17). Our largest circuit has 12 regulated promoters, doubling a plateau first noted in 2009 regarding a limit on the complexity of circuits that could be designed by hand (*38*). The DNA sequence output represents a testable prediction that either validates the underlying theory or reveals failure modes that can be addressed in the gate design. Previous experiments in which repressors were used to build synthetic logic gates showed that this often led to nonsensical functions that could not be predicted from the known interactions (*92*). Quantifying why predictions fail, where systems break, and how the host evolves provides failure modes that can be addressed through engineering. Iterative co-development of robust gates and software converged on genetic systems that are highly repetitive and modular, in stark contrast to the encoding of natural networks.

The future of engineering biology will require integrated design across many subcellular systems, including the creation of sensors that can process many stimuli, management of resources and metabolites, and control over multiple cellular functions (communication, stress response, chemotaxis, etc.). Within this greater framework, our approach is to separate the design and construction of synthetic circuits from engineering considerations for other cellular processes (e.g., metabolic engineering). Working with transcriptional circuits establishes a discrete boundary that other methods can engage to create a desired circuit to specification. For example, circuits could be built for which the sensors had been designed using all-atom biophysical models (*93*–*96*) and the outputs used to control enzyme expression levels, as determined via metabolic flux models (*97*, *98*). Integration with amorphous computing would enable spatial and community design (*58*, *99*, *100*). Integrating across these computer-aided design (CAD) tools in a way that automates design choices and balances constraints will be critical to advancing the complexity of genetic engineering projects.

REFERENCES AND NOTES

1. J. Hasty, D. McMillen, J. J. Collins, Engineered gene circuits. *Nature* **420**, 224–230 (2002). doi: 10.1038/nature01257; pmid: 12432407
2. D. Sprinzak, M. B. Elowitz, Reconstruction of genetic circuits. *Nature* **438**, 443–448 (2005). doi: 10.1038/nature04335; pmid: 16306982
3. D. A. Drubin, J. C. Way, P. A. Silver, Designing biological systems. *Genes Dev.* **21**, 242–254 (2007). doi: 10.1101/gad.1507207; pmid: 17289915
4. D. Endy, "Taking Faster and Smarter to New Physical Frontiers." *N.Y. Times*, 6 December 2011; www.nytimes.com/2011/12/06/science/drew-endy-better-computing-for-the-things-we-care-about-most.html.
5. D. G. Gibson *et al*., Creation of a bacterial cell controlled by a chemically synthesized genome. *Science* **329**, 52–56 (2010). doi: 10.1126/science.1190719; pmid: 20488990
6. W. Weber, M. Fussenegger, Emerging biomedical applications of synthetic biology. *Nat. Rev. Genet.* **13**, 21–35 (2012). pmid: 22124480
7. W. C. Ruder, T. Lu, J. J. Collins, Synthetic biology moving into the clinic. *Science* **333**, 1248–1252 (2011). doi: 10.1126/science.1206843; pmid: 21885773
8. P. M. Boyle, P. A. Silver, Parts plus pipes: Synthetic biology approaches to metabolic engineering. *Metab. Eng.* **14**, 223–232 (2012). doi: 10.1016/j.ymben.2011.10.003; pmid: 22037345
9. W. J. Holtz, J. D. Keasling, Engineering static and dynamic control of synthetic pathways. *Cell* **140**, 19–23 (2010). doi: 10.1016/j.cell.2009.12.029; pmid: 20085699
10. H. H. McAdams, A. Arkin, Simulation of prokaryotic genetic circuits. *Annu. Rev. Biophys. Biomol. Struct.* **27**, 199–224 (1998). doi: 10.1146/annurev.biophys.27.1.199; pmid: 9646867
11. H. H. McAdams, L. Shapiro, Circuit simulation of genetic networks. *Science* **269**, 650–656 (1995). doi: 10.1126/science.7624793; pmid: 7624793
12. M. Ptashne, A genetic switch: Gene control and phage. lambda (1986) (available at www.osti.gov/scitech/biblio/5413898).
13. U. Alon, *An Introduction to Systems Biology: Design Principles of Biological Circuits* (CRC Press, 2007).
14. N. E. Buchler, U. Gerland, T. Hwa, On schemes of combinatorial transcription logic. *Proc. Natl. Acad. Sci. U.S.A.* **100**, 5136–5141 (2003). doi: 10.1073/pnas.0930314100; pmid: 12702751
15. M. B. Elowitz, S. Leibler, A synthetic oscillatory network of transcriptional regulators. *Nature* **403**, 335–338 (2000). doi: 10.1038/35002125; pmid: 10659856
16. J. Stricker *et al*., A fast, robust and tunable synthetic gene oscillator. *Nature* **456**, 516–519 (2008). doi: 10.1038/nature07389; pmid: 18971928
17. S. Basu, Y. Gerchman, C. H. Collins, F. H. Arnold, R. Weiss, A synthetic multicellular system for programmed pattern formation. *Nature* **434**, 1130–1134 (2005). doi: 10.1038/nature03461; pmid: 15858574
18. W. A. Lim, Designing customized cell signalling circuits. *Nat. Rev. Mol. Cell Biol.* **11**, 393–403 (2010). doi: 10.1038/nrm2904; pmid: 20485291
19. T. S. Moon, C. Lou, A. Tamsir, B. C. Stanton, C. A. Voigt, Genetic programs constructed from layered logic gates in single cells. *Nature* **491**, 249–253 (2012). doi: 10.1038/nature11516; pmid: 23041931
20. A. Tamsir, J. J. Tabor, C. A. Voigt, Robust multicellular computing using genetically encoded NOR gates and chemical 'wires'. *Nature* **469**, 212–215 (2011). doi: 10.1038/nature09565; pmid: 21150903
21. B. C. Stanton *et al*., Genomic mining of prokaryotic repressors for orthogonal logic gates. *Nat. Chem. Biol.* **10**, 99–105 (2014). doi: 10.1038/nchembio.1411; pmid: 24316737
22. J. C. Anderson, C. A. Voigt, A. P. Arkin, Environmental signal integration by a modular AND gate. *Mol. Syst. Biol.* **3**, 133 (2007). doi: 10.1038/msb4100173; pmid: 17700541
23. S. Ausländer, D. Ausländer, M. Müller, M. Wieland, M. Fussenegger, Programmable single-cell mammalian biocomputers. *Nature* **487**, 123–127 (2012). pmid: 22722847
24. W. S. Teo, M. W. Chang, Development and characterization of AND-gate dynamic controllers with a modular synthetic GAL1 core promoter in *Saccharomyces cerevisiae*. *Biotechnol. Bioeng.* **111**, 144–151 (2014). doi: 10.1002/bit.25001; pmid: 23860786
25. P. Siuti, J. Yazbek, T. K. Lu, Synthetic circuits integrating logic and memory in living cells. *Nat. Biotechnol.* **31**, 448–452 (2013). doi: 10.1038/nbt.2510; pmid: 23396014
26. J. Bonnet, P. Yin, M. E. Ortiz, P. Subsoontorn, D. Endy, Amplifying genetic logic gates. *Science* **340**, 599–603 (2013). doi: 10.1126/science.1232758; pmid: 23539178
27. J. Bonnet, P. Subsoontorn, D. Endy, Rewritable digital data storage in live cells via engineered control of recombination directionality. *Proc. Natl. Acad. Sci. U.S.A.* **109**, 8884–8889 (2012). doi: 10.1073/pnas.1202344109; pmid: 22615351
28. L. Yang *et al*., Permanent genetic memory with >1-byte capacity. *Nat. Methods* **11**, 1261–1266 (2014). doi: 10.1038/nmeth.3147; pmid: 25344638
29. B. P. Kramer, M. Fussenegger, Hysteresis in a synthetic mammalian gene network. *Proc. Natl. Acad. Sci. U.S.A.* **102**, 9517–9522 (2005). doi: 10.1073/pnas.0500345102; pmid: 15972812
30. R. Daniel, J. R. Rubens, R. Sarpeshkar, T. K. Lu, Synthetic analog computation in living cells. *Nature* **497**, 619–623 (2013). doi: 10.1038/nature12148; pmid: 23676681
31. T. Danino, O. Mondragón-Palomino, L. Tsimring, J. Hasty, A synchronized quorum of genetic clocks. *Nature* **463**, 326–330 (2010). doi: 10.1038/nature08753; pmid: 20090747
32. S. Basu, R. Mehreja, S. Thiberge, M.-T. Chen, R. Weiss, Spatiotemporal control of gene expression with pulse-generating networks. *Proc. Natl. Acad. Sci. U.S.A.* **101**, 6355–6360 (2004). doi: 10.1073/pnas.0307571101; pmid: 15096621
33. M. Tigges, T. T. Marquez-Lago, J. Stelling, M. Fussenegger, A tunable synthetic mammalian oscillator. *Nature* **457**, 309–312 (2009). doi: 10.1038/nature07616; pmid: 19148099
34. T.-M. Lo, M. H. Tan, I. Y. Hwang, M. W. Chang, Designing a synthetic genetic circuit that enables cell density-dependent auto-regulatory lysis for macromolecule release. *Chem. Eng. Sci.* **103**, 29–35 (2013). doi: 10.1016/j.ces.2013.03.021
35. T. Ellis, X. Wang, J. J. Collins, Diversity-based, model-guided construction of synthetic gene networks with predicted functions. *Nat. Biotechnol.* **27**, 465–471 (2009). doi: 10.1038/nbt.1536; pmid: 19377462
36. F. Zhang, J. M. Carothers, J. D. Keasling, Design of a dynamic sensor-regulator system for production of chemicals and fuels derived from fatty acids. *Nat. Biotechnol.* **30**, 354–359 (2012). doi: 10.1038/nbt.2149; pmid: 22446695
37. R. Kwok, Five hard truths for synthetic biology. *Nature* **463**, 288–290 (2010). doi: 10.1038/463288a; pmid: 20090726
38. P. E. M. Purnick, R. Weiss, The second wave of synthetic biology: From modules to systems. *Nat. Rev. Mol. Cell Biol.* **10**, 410–422 (2009). doi: 10.1038/nrm2698; pmid: 19461664
39. Y. Yokobayashi, R. Weiss, F. H. Arnold, Directed evolution of a genetic circuit. *Proc. Natl. Acad. Sci. U.S.A.* **99**, 16587–16591 (2002). doi: 10.1073/pnas.252535999; pmid: 12451174
40. S. Kosuri *et al*., Composability of regulatory sequences controlling transcription and translation in *Escherichia coli*. *Proc. Natl. Acad. Sci. U.S.A.* **110**, 14024–14029 (2013). doi: 10.1073/pnas.1301301110; pmid: 23924614
41. D. B. Goodman, G. M. Church, S. Kosuri, Causes and effects of N-terminal codon bias in bacterial genes. *Science* **342**, 475–479 (2013). doi: 10.1126/science.1241934; pmid: 24072823
42. C. Lou, B. Stanton, Y.-J. Chen, B. Munsky, C. A. Voigt, Ribozyme-based insulator parts buffer synthetic circuits from genetic context. *Nat. Biotechnol.* **30**, 1137–1142 (2012). doi: 10.1038/nbt.2401; pmid: 23034349
43. V. K. Mutalik *et al*., Precise and reliable gene expression via standard transcription and translation initiation elements. *Nat. Methods* **10**, 354–360 (2013). doi: 10.1038/nmeth.2404; pmid: 23474465
44. F. Moser *et al*., Genetic circuit performance under conditions relevant for industrial bioreactors. *ACS Synth. Biol.* **1**, 555–564 (2012). doi: 10.1021/sb3000832; pmid: 23656232
45. B. Yordanov *et al*., A computational method for automated characterization of genetic components. *ACS Synth. Biol.* **3**, 578–588 (2014). doi: 10.1021/sb400152n; pmid: 24628037
46. S. Cardinale, M. P. Joachimiak, A. P. Arkin, Effects of genetic variation on the *E. coli* host-circuit interface. *Cell Rep.* **4**, 231–237 (2013). doi: 10.1016/j.celrep.2013.06.023; pmid: 23871664
47. N. Rosenfeld, J. W. Young, U. Alon, P. S. Swain, M. B. Elowitz, Gene regulation at the single-cell level. *Science* **307**, 1962–1965 (2005). doi: 10.1126/science.1106914; pmid: 15790856
48. T. S. Gardner, C. R. Cantor, J. J. Collins, Construction of a genetic toggle switch in *Escherichia coli*. *Nature* **403**, 339–342 (2000). doi: 10.1038/35002131; pmid: 10659857
49. F. K. Balagaddé, L. You, C. L. Hansen, F. H. Arnold, S. R. Quake, Long-term monitoring of bacteria undergoing programmed population control in a microchemostat. *Science* **309**, 137–140 (2005). doi: 10.1126/science.1109173; pmid: 15994559
50. A. P. Arkin, D. A. Fletcher, Fast, cheap and somewhat in control. *Genome Biol.* **7**, 114 (2006). doi: 10.1186/gb-2006-7-8-114; pmid: 16942631
51. R. Ghamari, thesis, Boston University (2011).
52. D. Chandran, F. T. Bergmann, H. M. Sauro, TinkerCell: Modular CAD tool for synthetic biology. *J. Biol. Eng.* **3**, 19 (2009). doi: 10.1186/1754-1611-3-19; pmid: 19874625
53. C. J. Myers *et al*., iBioSim: A tool for the analysis and design of genetic circuits. *Bioinformatics* **25**, 2848–2849 (2009). doi: 10.1093/bioinformatics/btp457; pmid: 19628507
54. J. Beal, T. Lu, R. Weiss, Automatic compilation from high-level biologically-oriented programming language to genetic regulatory networks. *PLOS ONE* **6**, e22490 (2011). doi: 10.1371/journal.pone.0022490; pmid: 21850228
55. M. J. Czar, Y. Cai, J. Peccoud, Writing DNA with GenoCAD. *Nucleic Acids Res.* **37**, W40–W47 (2009). doi: 10.1093/nar/gkp361; pmid: 19429897

56. M. A. Marchisio, J. Stelling, Automatic design of digital synthetic gene circuits. *PLOS Comput. Biol.* **7**, e1001083 (2011). doi: 10.1371/journal.pcbi.1001083; pmid: 21399700
57. H. M. Sauro *et al*., Next generation simulation tools: The Systems Biology Workbench and BioSPICE integration. *OMICS* **7**, 355–372 (2003). doi: 10.1089/153623103322637670; pmid: 14683609
58. T. J. Rudge, P. J. Steiner, A. Phillips, J. Haseloff, Computational modeling of synthetic microbial biofilms. *ACS Synth. Biol.* **1**, 345–352 (2012). doi: 10.1021/sb300031n; pmid: 23651288
59. J. R. Kelly *et al*., Measuring the activity of BioBrick promoters using an in vivo reference standard. *J. Biol. Eng.* **3**, 4 (2009). doi: 10.1186/1754-1611-3-4; pmid: 19298678
60. T. H. Segall-Shapiro, A. J. Meyer, A. D. Ellington, E. D. Sontag, C. A. Voigt, A 'resource allocator' for transcription based on a highly fragmented T7 RNA polymerase. *Mol. Syst. Biol.* **10**, 742 (2014). doi: 10.15252/msb.20145299; pmid: 25080493
61. H. H. Wang *et al*., Genome-scale promoter engineering by coselection MAGE. *Nat. Methods* **9**, 591–593 (2012). doi: 10.1038/nmeth.1971; pmid: 22484848
62. B. Canton, A. Labno, D. Endy, Refinement and standardization of synthetic biological parts and devices. *Nat. Biotechnol.* **26**, 787–793 (2008). doi: 10.1038/nbt1413; pmid: 18612302
63. R. Weiss, thesis, Massachusetts Institute of Technology (2001).
64. A. A. Green, P. A. Silver, J. J. Collins, P. Yin, Toehold switches: De-novo-designed regulators of gene expression. *Cell* **159**, 925–939 (2014). doi: 10.1016/j.cell.2014.10.002; pmid: 25417166
65. V. K. Mutalik, L. Qi, J. C. Guimaraes, J. B. Lucks, A. P. Arkin, Rationally designed families of orthogonal RNA regulators of translation. *Nat. Chem. Biol.* **8**, 447–454 (2012). doi: 10.1038/nchembio.919; pmid: 22446835
66. J. Chappell, M. K. Takahashi, J. B. Lucks, Creating small transcription activating RNAs. *Nat. Chem. Biol.* **11**, 214–220 (2015). doi: 10.1038/nchembio.1737; pmid: 25643173
67. D. Chen, A. P. Arkin, Sequestration-based bistability enables tuning of the switching boundaries and design of a latch. *Mol. Syst. Biol.* **8**, 620 (2012). doi: 10.1038/msb.2012.52; pmid: 23089683
68. L. S. Qi *et al*., Repurposing CRISPR as an RNA-guided platform for sequence-specific control of gene expression. *Cell* **152**, 1173–1183 (2013). doi: 10.1016/j.cell.2013.02.022; pmid: 23452860
69. D. Bikard *et al*., Programmable repression and activation of bacterial gene expression using an engineered CRISPR-Cas system. *Nucleic Acids Res.* **41**, 7429–7437 (2013). doi: 10.1093/nar/gkt520; pmid: 23761437
70. A. A. Nielsen, C. A. Voigt, Multi-input CRISPR/Cas genetic circuits that interface host regulatory networks. *Mol. Syst. Biol.* **10**, 763 (2014). doi: 10.15252/msb.20145735; pmid: 25422271
71. K. M. Esvelt *et al*., Orthogonal Cas9 proteins for RNA-guided gene regulation and editing. *Nat. Methods* **10**, 1116–1121 (2013). doi: 10.1038/nmeth.2681; pmid: 24076762
72. T. S. Ham, S. K. Lee, J. D. Keasling, A. P. Arkin, Design and construction of a double inversion recombination switch for heritable sequential genetic memory. *PLOS ONE* **3**, e2815 (2008). doi: 10.1371/journal.pone.0002815; pmid: 18665232
73. The standard differs from the Kelly standard and contains an insulating upstream terminator, a different spacer upstream of the promoter (as opposed to a BioBricks prefix), RiboJ, RBS B0064, a different terminator, and three silent mutations to YFP.
74. D. E. Thomas, P. R. Moorby, *The Verilog Hardware Description Language* (Springer Science & Business Media, 2002).
75. D. C. Crockford, "The application/json Media Type for JavaScript Object Notation (JSON)"; https://tools.ietf.org/html/rfc4627.
76. M. Galdzicki *et al*., The Synthetic Biology Open Language (SBOL) provides a community standard for communicating designs in synthetic biology. *Nat. Biotechnol.* **32**, 545–550 (2014). doi: 10.1038/nbt.2891; pmid: 24911500
77. R. Brayton, A. Mishchenko, in *Computer Aided Verification*, T. Touili, B. Cook, P. Jackson, Eds. (Springer, 2010), pp. 24–40; http://link.springer.com/chapter/10.1007/978-3-642-14295-6_5.
78. N. Roehner, C. J. Myers, Directed acyclic graph-based technology mapping of genetic circuit models. *ACS Synth. Biol.* **3**, 543–555 (2014). doi: 10.1021/sb400135t; pmid: 24650240
79. F. Yaman, S. Bhatia, A. Adler, D. Densmore, J. Beal, Automated selection of synthetic biology parts for genetic regulatory networks. *ACS Synth. Biol.* **1**, 332–344 (2012). doi: 10.1021/sb300032y; pmid: 23651287
80. G. Rodrigo, A. Jaramillo, AutoBioCAD: Full biodesign automation of genetic circuits. *ACS Synth. Biol.* **2**, 230–236 (2013). doi: 10.1021/sb300084h; pmid: 23654253
81. L. Huynh, I. Tagkopoulos, Optimal part and module selection for synthetic gene circuit design automation. *ACS Synth. Biol.* **3**, 556–564 (2014). doi: 10.1021/sb400139h; pmid: 24933033
82. E. Oberortner, S. Bhatia, E. Lindgren, D. Densmore, A rule-based design specification language for synthetic biology. *J. Emerg. Technol. Comput. Syst.* **11**, 25:1–25:19 (2014). doi: 10.1021/sb400139h; pmid: 24933033
83. M. J. Smanski *et al*., Functional optimization of gene clusters by combinatorial design and assembly. *Nat. Biotechnol.* **32**, 1241–1249 (2014). doi: 10.1038/nbt.3063; pmid: 25419741
84. See supplementary materials on *Science* Online.
85. H. M. Salis, E. A. Mirsky, C. A. Voigt, Automated design of synthetic ribosome binding sites to control protein expression. *Nat. Biotechnol.* **27**, 946–950 (2009). doi: 10.1038/nbt.1568; pmid: 19801975
86. S. C. Sleight, H. M. Sauro, Visualization of evolutionary stability dynamics and competitive fitness of *Escherichia coli* engineered with randomized multigene circuits. *ACS Synth. Biol.* **2**, 519–528 (2013). doi: 10.1021/sb400055h; pmid: 24004180
87. Y.-J. Chen *et al*., Characterization of 582 natural and synthetic terminators and quantification of their design constraints. *Nat. Methods* **10**, 659–664 (2013). doi: 10.1038/nmeth.2515; pmid: 23727987
88. V. A. Rhodius, V. K. Mutalik, C. A. Gross, Predicting the strength of UP-elements and full-length *E. coli* σ^E promoters. *Nucleic Acids Res.* **40**, 2907–2924 (2012). doi: 10.1093/nar/gkr1190; pmid: 22156164
89. S. Wolfram, *A New Kind of Science* (Wolfram Media, Champaign, IL, 2002), vol. 5.
90. This was done for 0x0E, 0x19, 0x1C, 0x38, 0x3D, 0x6E, 0x81, 0xB9, 0xC6, 0xC7, 0xBD, and 0xC8.
91. S. Kosuri, G. M. Church, Large-scale de novo DNA synthesis: Technologies and applications. *Nat. Methods* **11**, 499–507 (2014). doi: 10.1038/nmeth.2918; pmid: 24781323
92. R. S. Cox, M. G. Surette, M. B. Elowitz, Programming gene expression with combinatorial promoters. *Mol. Syst. Biol.* **3**, 145 (2007). doi: 10.1038/msb4100187
93. B. I. Dahiyat, S. L. Mayo, De novo protein design: Fully automated sequence selection. *Science* **278**, 82–87 (1997). doi: 10.1126/science.278.5335.82; pmid: 9311930
94. L. L. Looger, M. A. Dwyer, J. J. Smith, H. W. Hellinga, Computational design of receptor and sensor proteins with novel functions. *Nature* **423**, 185–190 (2003). doi: 10.1038/nature01556; pmid: 12736688
95. C. E. Tinberg *et al*., Computational design of ligand-binding proteins with high affinity and selectivity. *Nature* **501**, 212–216 (2013). doi: 10.1038/nature12443; pmid: 24005320
96. W. R. Whitaker, S. A. Davis, A. P. Arkin, J. E. Dueber, Engineering robust control of two-component system phosphotransfer using modular scaffolds. *Proc. Natl. Acad. Sci. U.S.A.* **109**, 18090–18095 (2012). doi: 10.1073/pnas.1209230109; pmid: 23071327
97. C. S. Henry, L. J. Broadbelt, V. Hatzimanikatis, Thermodynamics-based metabolic flux analysis. *Biophys. J.* **92**, 1792–1805 (2007). doi: 10.1529/biophysj.106.093138; pmid: 17172310
98. A. M. Feist *et al*., A genome-scale metabolic reconstruction for *Escherichia coli* K-12 MG1655 that accounts for 1260 ORFs and thermodynamic information. *Mol. Syst. Biol.* **3**, 121 (2007). doi: 10.1038/msb4100155
99. S. S. Jang, K. T. Oishi, R. G. Egbert, E. Klavins, Specification and simulation of synthetic multicelled behaviors. *ACS Synth. Biol.* **1**, 365–374 (2012). doi: 10.1021/sb300034m; pmid: 23651290
100. A. E. Blanchard, V. Celik, T. Lu, Extinction, coexistence, and localized patterns of a bacterial population with contact-dependent inhibition. *BMC Syst. Biol.* **8**, 23 (2014). pmid: 24576330
101. J. Y. Quinn *et al*., SBOL Visual: A graphical language for genetic designs. *PLOS Biol.* **13**, e1002310 (2015). doi: 10.1371/journal.pbio.1002310; pmid: 26633141

ACKNOWLEDGMENTS

All data described in this paper are presented in the supplementary materials. Cello can be accessed at www.cellocad.org; the code is open source and maintained via the Nona Research Foundation (www.nonasoftware.org). The insulated gates, circuits, and plasmids defined in the UCF can be obtained via Addgene (www.addgene.org). The *E. coli* strain used in this UCF is available from New England Biolabs (C3019). D.D. is president and part owner of Lattice Automation, which works on software for synthetic biology. Supported by Office of Naval Research, Multidisciplinary University Research Initiative grant N00014-13-1-0074 (C.A.V. and A.A.K.N.); a Siebel Scholarship (Class of 2015) and Air Force Office of Scientific Research, National Defense Science and Engineering Graduate fellowship FA9550-11-C-0028 (A.A.K.N.); National Institute of General Medical Sciences grant P50 GM098792 and NSF Synthetic Biology Engineering Research Center grant SynBERC EEC0540879 (C.A.V. and J.S.); NSF grant 1147158 (D.D. and P.V.); the NIST Strategic and Emerging Research Initiative (V.P., E.A.S., and D.R.); the National Research Council postdoctoral program (V.P.); an AWS in Education Grant; and Thermo Fischer (Life Technologies, A114510). We thank S. Bhatia (Boston University) for his work to enumerate the three-input, one-output logic motifs; R. Ghamari (Boston University) for early work on the Cello software foundation and design approach; N. Roehner for technical assistance with SBOL; and E. Oberortner for technical assistance with Eugene. The National Institute of Standards and Technology notes that certain commercial equipment, instruments, and materials are identified in this paper to specify an experimental procedure as completely as possible. In no case does the identification of particular equipment or materials imply a recommendation or endorsement by NIST, nor does it imply that the materials, instruments, or equipment are necessarily the best available for the purpose. The University of California, San Francisco, holds a patent application associated with this work (US20130005590 A1). The set of insulated gates, the plasmid backbones for the circuit and output plasmids (Fig. 1), and the standard RPU plasmid are available via Addgene (www.addgene.org/Christopher_Voigt/). Cello can be accessed as a web-based application (www.cellocad.org), the code for which will be open source and provided on Github (https://github.com/CIDARLAB/cello).

SUPPLEMENTARY MATERIALS

www.sciencemag.org/content/352/6281/aac7341/suppl/DC1
Supplementary Text
Materials and Methods
Figs. S1 to S44
Tables S1 to S10
Supplementary References
Data Files S1 to S4

4 June 2015; accepted 21 January 2016
10.1126/science.aac7341

Genetic circuit design automation

Alec A. K. NielsenBryan S. DerJonghyeon ShinPrashant VaidyanathanVanya ParalanovElizabeth A. StrychalskiDavid RossDouglas DensmoreChristopher A. Voigt

Science, 352 (6281), aac7341.

Programming circuitry for synthetic biology

As synthetic biology techniques become more powerful, researchers are anticipating a future in which the design of biological circuits will be similar to the design of integrated circuits in electronics. Nielsen *et al.* describe what is essentially a programming language to design computational circuits in living cells. The circuits generated on plasmids expressed in *Escherichia coli* required careful insulation from their genetic context, but primarily functioned as specified. The circuits could, for example, regulate cellular functions in response to multiple environmental signals. Such a strategy can facilitate the development of more complex circuits by genetic engineering.

Science, this issue p. 10.1126/science.aac7341