RESEARCH ARTICLE SUMMARY

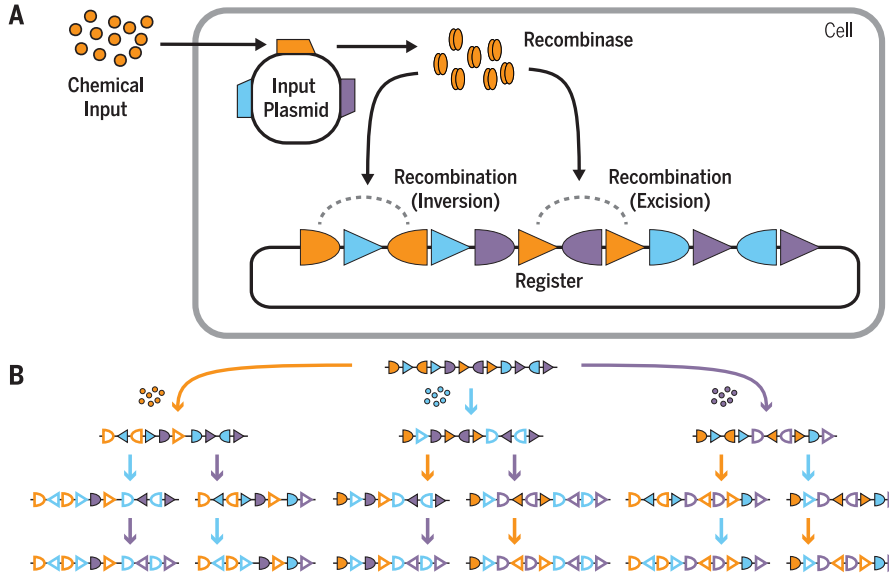
SYNTHETIC BIOLOGY

Synthetic recombinase-based state machines in living cells

Nathaniel Roquet, Ava P. Soleimany, Alyssa C. Ferris, Scott Aaronson, Timothy K. Lu*

INTRODUCTION: Living systems execute regulatory programs and exhibit specific phenotypes depending on the identity and timing of chemical signals, but general strategies for mimicking such behaviors with artificial genetic programs are lacking. Synthetic circuits that produce outputs only depending on simultaneous combinations of inputs are limited in their scale and their ability to recognize dynamics because they do not uniquely detect or respond to temporally ordered inputs. To address these limitations, we developed and experimentally validated a framework for implementing state machines that record and respond to all identities and orders of gene regulatory events in living cells.

RATIONALE: We built recombinase-based state machines (RSMs) that use input-driven recombinases to manipulate DNA registers made up of overlapping and orthogonal pairs of recombinase recognition sites. Specifically, chemical inputs express recombinases that can perform two types of irreversible operations on a register: excision if their recognition sites are aligned, or inversion if their recognition sites are anti-aligned. The registers are designed to adopt a distinct DNA sequence ("state") for every possible "permuted substring" of inputs—that is, every possible combination and ordering of inputs. The state persists even when inputs are removed and may be read with sequencing or by polymerase chain reaction. Using mathematical analysis to determine how the structure of a RSM relates to its scalability, we found that incorporating multiple orthogonal pairs of recognition sites per recombinase allows a RSM to outperform combinational circuits in scale.



Summary of a three-input, 16-state RSM. (**A**) The RSM mechanism. A chemical input induces the expression of a recombinase (from a gene on the input plasmid) that modifies a DNA register made up of overlapping and orthogonal recombinase recognition sites. Distinct recombinases can be controlled by distinct inputs. These recombinases each target multiple orthogonal pairs of their cognate recognition sites (shown as triangles and half-ovals) to catalyze inversion (when the sites are anti-aligned) or excision (when the sites are aligned). (**B**) The register is designed to adopt a distinct DNA state for every identity and order of inputs. Three different inputs—orange, blue, and purple—are represented by colored arrows, each of which expresses a distinct recombinase. Unrecombined recognition sites are shaded; recombined recognition sites are outlined.

Genetic parts (made up of promoters, terminators, and genes) may be interleaved into RSM registers to implement gene regulation programs capable of expressing unique combinations of genes in each state. In addition, we provide a computational tool that accepts a user-specified two-input multigene regulation program and returns corresponding registers that implement it. This searchable database enables facile creation of RSMs with desired behaviors without requiring detailed knowledge of gene circuit design.

ON OUR WEBSITE

Read the full article at http://dx.doi.org/10.1126/science.aad8559

RESULTS: We built two-input, five-state RSMs and three-input, 16-state RSMs capable of recording every permuted substring of their inputs. We tested the RSMs in *Escherichia coli* and used Sanger sequencing to measure performance. For the two-input, five-state RSM, at least 97% of cells treated with each permuted substring of inputs adopted their expected state. For the three-input, 16-state RSM, at least 88% of cells treated with each permuted substring of inputs adopted their expected state, although we observed 100% for most treatment conditions.

We used these two- and three-input RSMs to implement gene regulation programs by interleaving genetic parts into their registers. For the two-input, five-state system, we designed registers for various gene regulation programs using our computational database and search function. Four single-gene regulation programs and one multigene regulation program (which expressed a different set of fluorescent reports in each state) were successfully implemented in *E. coli*, with at least 94% of cells adopting their expected gene expression profile when treated with each permuted substring of inputs. Lastly, we successfully implemented two different three-input, 16-state gene regulation programs; one of these—a three-input passcode switch—performed with at least 97% of cells adopting the expected gene expression behavior.

CONCLUSION: Our work presents a powerful framework for implementing RSMs in living cells that are capable of recording and responding to all identities and orders of a set of chemical inputs. Depending on desired applications, the prototypical inducible systems used here to drive the RSMs can be replaced by sensors that correspond to desired input signals or gene regulation events. We anticipate that the integration of RSMs into complex living systems will transform our capacity to understand and engineer them. ■

The list of author affiliations is available in the full article online.
*Corresponding author. Email: timlu@mit.edu
Cite this article as N. Roquet *et al.*, *Science* 353, aad8559 (2016). DOI: 10.1126/science.aad8559

RESEARCH ARTICLE

SYNTHETIC BIOLOGY

Synthetic recombinase-based state machines in living cells

Nathaniel Roquet,[1,2,3,4] Ava P. Soleimany,[1,2,3] Alyssa C. Ferris,[1,2,3,5] Scott Aaronson,[3] Timothy K. Lu[1,2,3,4,6,*]

State machines underlie the sophisticated functionality behind human-made and natural computing systems that perform order-dependent information processing. We developed a recombinase-based framework for building state machines in living cells by leveraging chemically controlled DNA excision and inversion operations to encode states in DNA sequences. This strategy enables convenient readout of states (by sequencing and/or polymerase chain reaction) as well as complex regulation of gene expression. We validated our framework by engineering state machines in *Escherichia coli* that used one, two, or three chemical inputs to control up to 16 DNA states. These state machines were capable of recording the temporal order of all inputs and performing multi-input, multi-output control of gene expression. We also developed a computational tool for the automated design of gene regulation programs using recombinase-based state machines. Our scalable framework should enable new strategies for recording and studying how combinational and temporal events regulate complex cell functions and for programming sophisticated cell behaviors.

State machines are systems that exist in any of a number of states, in which transitions between states are controlled by inputs (*1*). The next state of a given state machine is determined not only by a particular input, but also by its current state. This state-dependent logic can be used to produce outputs that are dependent on the order of inputs, unlike in combinational logic circuits wherein the outputs are solely dependent on the current combination of inputs. Figure 1 depicts a state machine that enters a different state for each permuted substring of two inputs A and B, by which we refer to each distinct combination and ordering of those two inputs: {no input, A only, B only, A followed by B (A → B), B followed by A (B → A)}.

Synthetic state machines that record and respond to sequences of signaling and gene regulatory events within a cell could be transformative tools in the study and engineering of complex living systems. For example, in human development, progenitor cells differentiate into specific cell types with disparate functions determined by the timing and order of transcription factor (TF) activation (*2*, *3*). This information has allowed researchers to program human stem cells into differentiated cells (*4*, *5*), and conversely, reprogram differentiated cells into stem cells by means of exogenous, sequential TF activation (*6*, *7*). However, the temporal organization of TF cascades that drive different cell lineages remains largely unknown. State machines that record and actuate gene expression in response to the order of TF activation in individual cells would be useful for understanding and modulating these differentiation processes.

Such state machines may also improve our understanding of disease progression, which can also depend on the appearance and order of extracellular and intracellular factors. For example, in cancer, the temporal order of genetic mutations in a tumor can determine its phenotype (*8*). Similarly, in both somatic diseases and pathogenic infections, preadaptation of disease cells to different environmental conditions may affect the way the cells behave and respond to drug treatments (*9*–*12*). Integrating state machines into disease models and subsequently analyzing the history of cells that survive treatment would be useful for understanding how disease progression affects therapeutic response.

Despite their potential to transform the understanding and engineering of biological systems, complex functional state machines have yet to be implemented in living cells because of a lack of scalable and generalizable frameworks (*13*). Oishi *et al.* proposed a theoretical CRISPR interference-based strategy for building state machines in living cells, in which state is encoded epigenetically (*14*). In contrast, we developed a scalable recombinase-based strategy for implementing state machines in living cells, in which a given state is encoded in a particular DNA sequence. The direct storage of state information in the DNA sequence ensures that it is maintained stably and with minimal burden to the cell. Recombinases have been used to implement switches (*15*–*19*), chemical pulse counters (*20*), Boolean logic gates integrated with memory (*21*, *22*), and temporal logic (*23*). We used them to implement scalable state machines, such as those that can distinguish among all possible permuted substrings of a set of inputs with unique gene expression outputs. We refer to our state machine implementations as recombinase-based state machines (RSMs).

Recombinase-based state machine parts and operations

In a RSM, inputs are defined by chemical signals, and state is defined by the DNA sequence within a prescribed region of DNA, termed the register. Chemical signals mediate state transitions by inducing the expression of large serine recombinases that catalyze recombination events on the register, thereby changing the state. Specifically, each recombinase recognizes a cognate pair of DNA recognition sites on the register, *attP* (derived from a phage) and *attB* (derived from its bacterial host), and carries out a recombination reaction between them, yielding *attL* and *attR* sites (made up of conjoined halves of *attB* and *attP*) (*24*, *25*). In the absence of extra cofactors, this reaction is irreversible (fig. S1 and text S1) (*26*–*28*). Each site in a cognate *attB*-*attP* pair has a matching central dinucleotide that determines its polarity (*29*, *30*). If the two sites are anti-aligned (oriented with opposite polarity) on the register, then the result of their recombination is the inversion of the DNA between them (Fig. 2A and fig. S2A). Alternatively, if the two sites are aligned (oriented with the same polarity) on the register, then the result of their recombination is the excision of the DNA between them (Fig. 2B and fig. S2B). DNA segments that are excised from the register are assumed to be lost because of a lack of origin of replication.

When there are multiple inputs to a RSM, they can each drive distinct recombinases that operate only on their own *attB*-*attP* pairs. At least 25 (putatively orthogonal) large serine recombinases have been described and tested in the literature (*18*, *25*), and bioinformatics mining can be used to discover even more (*18*). Recognition sites for

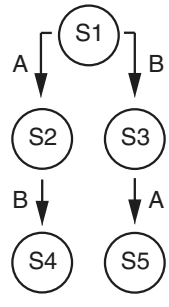


Fig. 1. Example of a state machine. Nodes represent states; arrows represent transitions between states mediated by inputs. Each of the possible permuted substrings of the two inputs A and B generates a unique state.

[1]Synthetic Biology Group, Research Laboratory of Electronics, Massachusetts Institute of Technology, Cambridge, MA 02139, USA. [2]Synthetic Biology Center, Department of Biological Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139, USA. [3]Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, USA. [4]Biophysics Program, Harvard University, Boston, MA 02115, USA. [5]Biochemistry Program, Wellesley College, Wellesley, MA 02481, USA. [6]Center for Microbiome Informatics and Therapeutics, Massachusetts Institute of Technology, Cambridge, MA 02139, USA.
*Corresponding author. Email: timlu@mit.edu

multiple recombinases may be arranged in several different ways on the register. If *attB-attP* pairs from different recombinases are nested or overlapping, then the operation of one recombinase can affect the operation of subsequent recombinases—either by rearranging the relative orientation of their *attB* and *attP* sites or by excising one or both sites in a pair from the register—thereby precluding any type of downstream operation on these sites. For example, if we consider the initial register design in Fig. 2C, applying input B → A leads to a unique DNA sequence, but applying A → B leads to the same DNA sequence we would expect if we only applied A, because the A-driven recombinase excises a site for the B-driven recombinase.

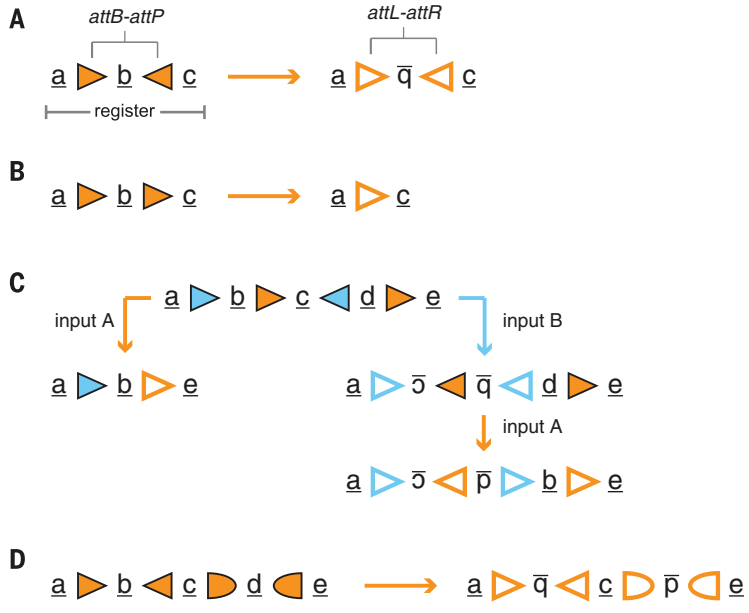


Fig. 2. Rules of recombination on a register. The register is depicted as an array of underscored alphabet symbols (arbitrary DNA) and shape symbols (recognition sites). (**A**) If sites in an *attB-attP* pair are anti-aligned, then the DNA between them is inverted during recombination. (**B**) If sites in an *attB-attP* pair are aligned, then the DNA between them is excised during recombination. (**C**) Multiple inputs can drive distinct recombinases that operate on their own *attB-attP* pairs. In this example, input A drives the orange recombinase and input B drives the blue recombinase. (**D**) Multiple orthogonal *attB-attP* pairs for a given recombinase can be placed on a register. Here, distinct shapes denote two pairs of *attB-attP*. Up to six orthogonal and directional *attB-attP* pairs can be created per large serine recombinase (*31*). Figure S2 gives more detail on the recombination reactions shown here.

We measure the "information capacity" of a RSM by the number of distinct states it can access, and hence the number of permuted substrings of inputs it can distinguish. Given the noncommutative nature of recombinase operations on a register, one might naïvely believe that the information capacity of RSMs would behave like $N!$ for N inputs. But if a RSM is designed such that each input-driven recombinase only has one *attB-attP* pair on the register, the information capacity of the RSM never exceeds 2^N, which is the result we would expect if recombinase operations were commutative (Box 1 and text S2). To circumvent this information bottleneck, registers must be designed with multiple orthogonal *attB-attP* pairs per recombinase. Orthogonal *attB-attP* pairs for a large serine recombinase can be engineered by mutating the central dinucleotide of each site in the native *attB-attP* pair (*29*–*31*). Pairs of sites with the same central dinucleotide sequence should recombine, but they should not recombine if the central dinucleotide sequences do not match (Fig. 2D and fig. S2C).

Building a two-input, five-state RSM

To implement a RSM that enters a different state (five in total) for every permuted substring of two inputs, it was sufficient to use two orthogonal *attB-attP* pairs for one recombinase and one *attB-attP* pair for the other recombinase. Figure 3A shows the RSM design and a detailed representation of its state diagram. This RSM is composed of two plasmids: an input plasmid and an output plasmid. The input plasmid, at a high copy number, expresses two large serine recombinases, BxbI and TP901, from the anhydrotetracycline (ATc)–inducible P_{LtetO} promoter and the arabinose (Ara)–inducible P_{BAD} promoter, respectively. The output plasmid, at a single copy number, contains the register that is modified by the recombinases expressed from the input plasmid.

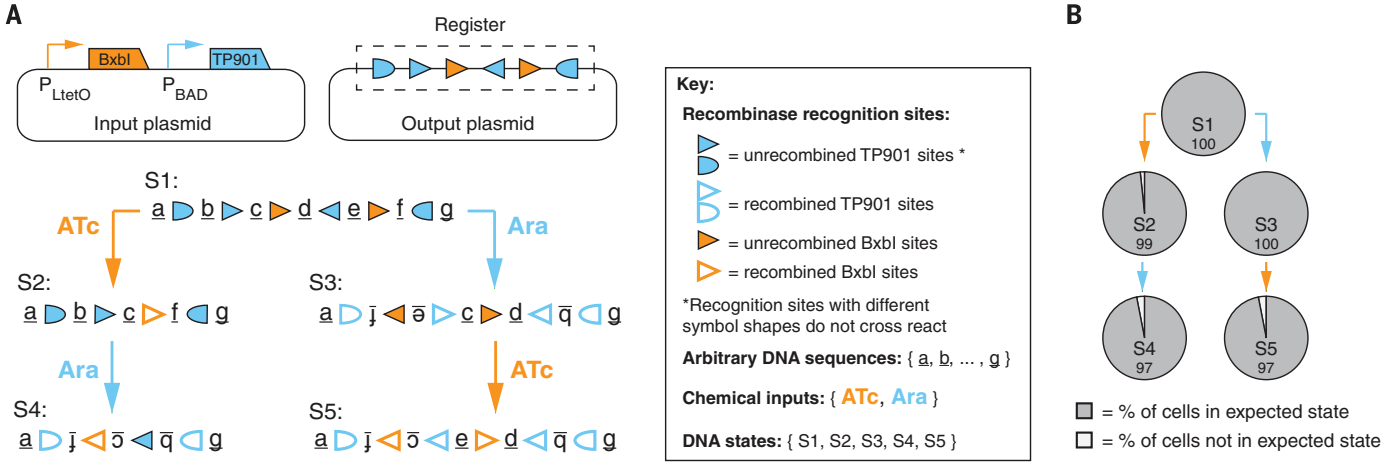


Fig. 3. Designing and validating a two-input, five-state RSM. (**A**) The two plasmids used to implement the RSM (top) and a detailed state diagram demonstrating the resulting register for each permuted substring of the two inputs (ATc and Ara; bottom). (**B**) The performance of the RSM in *E. coli*. Nodes represent populations of cells induced with permuted substrings of the inputs ATc (orange arrow) and Ara (blue arrow). Cultures were treated with saturating concentrations of each input (ATc, 250 ng/ml; Ara, 1% w/v) at 30°C for 18 hours in three biological replicates. Node labels indicate the expected state [corresponding to (A)] and the percentage of cells in that state as determined by Sanger sequencing of colonies from individual cells in each population (at least 66 cells totaled over all three biological replicates).

The register is initially composed of an aligned BxbI *attB-attP* pair and two anti-aligned and orthogonal TP901 *attB-attP* pairs. If ATc is introduced first to the system, then BxbI is expressed and excises the DNA inside of its cognate recognition site pair, which includes a recognition site for TP901. Subsequent introduction of Ara to the system induces the expression of TP901, which recombines its cognate recognition sites on the outer edge of the register, thus inverting everything in between. Conversely, if Ara is introduced first to the system, then the outer TP901 sites invert everything between the edges of the register and the inner TP901 sites invert an inner portion of the register, thus setting the BxbI recognition sites into an anti-aligned configuration. Subsequent application of ATc to the system inverts the sequence of DNA between the BxbI sites. As a result, each permuted substring of the inputs yields a distinct DNA sequence on the register.

To evaluate the performance of the RSM in *Escherichia coli*, we grew five populations of cells that were treated with all five permuted substrings of the inputs ATc and Ara (no input, ATc only, Ara only, ATc → Ara, and Ara → ATc). We Sanger-sequenced the register in colonies of at least 22 cells from each population in each of three biological replicates to determine the percent of cells with the expected DNA sequence (Fig. 3B) (*32*). At least 97% of all cells treated with each permuted substring of inputs adopted the expected state, thus confirming the fidelity of our RSM. Table S1 provides information for the sequenced registers that were not in the expected state.

Because our Sanger sequencing readout of state was low-throughput, we also developed a quantitative polymerase chain reaction (qPCR)–based method to conveniently interrogate state on a population-wide level. The excision and inversion of DNA segments in our register permitted the design of primer pairs that were amplified in some states but not others. We created a computer program, the PCR-based state interrogation tool (PSIT), to identify all possible sets of primer pairs that uniquely identify each state of a given register (fig. S3 and appendix S2). For our two-input, five-state RSM, we chose a set of three primer pairs and performed qPCR on DNA that was isolated from each population of cells treated with all possible permuted substrings of the ATc and Ara inputs. The fractional amount of register DNA amplified was calculated for each primer pair in our set and was compared to what we would expect if all cells in each population adopted just one of the five possible states (*32*). In agreement with our sequencing results, the qPCR measurements of all experimental populations were most similar to what we would expect if all cells in each population adopted their expected state (fig. S4).

Scaling RSMs

We developed a modular register design strategy for building RSMs that enter a distinct state for every permuted substring of inputs (approximately $eN!$ states for N inputs; see table S2 and texts S4 and S5). For N inputs, the design strategy uses $N-1$ recognition sites per recombinase, and hence is limited to register designs for up to seven inputs (13,700 states) because only six orthogonal and directional *attB-attP* pairs can be created per large serine recombinase (*31*).

Because the two-input, five-state RSM shown in Fig. 3A represents only a marginal improvement in information capacity over two-input, four-state systems achievable by combinational computation, we sought to further demonstrate the information capacity enabled by our RSM framework by scaling to a three-input, 16-state RSM (Fig. 4A and fig. S5). The input plasmid for this state machine expresses an additional recombinase, A118, under a 2,4-diacetylphloroglucinol (DAPG)–inducible P_{PhlF} promoter system, and its register uses two orthogonal *attB-attP* pairs for each of the three recombinases (following the design strategy in text S5).

To evaluate the performance of this RSM in *E. coli*, we grew 16 populations of cells that were treated with all 16 permuted substrings of the inputs ATc, Ara, and DAPG. We sequenced the register in colonies of five or six cells from each population in each of three biological replicates to determine the percentage of cells with the expected DNA sequence (Fig. 4B) (*32*). In most populations, 100% of the cells adopted their expected state, and even in the worst-performing population (ATc → Ara → DAPG), 88% of cells adopted their expected state. Table S1 provides information for the sequenced registers that were not in the expected state. We also measured the predominant state of each population by qPCR with a set of six primer pairs elucidated by PSIT (*32*). In agreement with the sequencing results, the qPCR measurements for all experimental populations were most similar to what we would expect if all cells in each population adopted their expected state (fig. S6).

Gene-regulatory RSMs

Our state machine framework enables the creation of state-dependent gene regulation programs that specify which genes should be expressed or not expressed in each state. This could be useful for a wide range of biological applications, such as programming synthetic differentiation cascades,

Box 1. Mathematical discussion of RSMs.

If a RSM with *N* inputs is designed such that each input-driven recombinase only has one *attB-attP* pair on the register, the number of states cannot exceed 2^N. To prove this important claim, we first introduce the concept of irreducibility. An irreducible string of recombinases is one in which, when the recombinases are applied to a register in the given order, each recombinase performs an operation (excision or inversion) on the register. We can make the following two statements about irreducible strings:

Statement 1: Every possible state of a register must be accessible by the application of some irreducible string of recombinases. This follows from considering that (i) each state is the result of a string of recombination operations, and (ii) the string of recombinases corresponding to that string of recombination operations is irreducible by definition.

Statement 2: Assuming a register with one *attB-attP* pair per recombinase, all irreducible strings from the same subset of recombinases generate the same state on the register. This follows from considering that (i) all rearrangeable DNA segments on the register are flanked on both sides by *attB* and/or *attP* sites belonging to the subset of recombinases being applied; (ii) by the definition of irreducibility, each recombinase in the irreducible string will catalyze recombination between its *attB-attP* pair; and (iii) when recombination between *attB* and *attP* sites occurs, they always form the same junctions: The back end of the *attB* will join the front end of the *attP*, and the front end of the *attB* will join the back end of the *attP*. Therefore, all rearrangeable DNA segments will form the same junctions after an irreducible string of recombinases is applied, regardless of the order in which those recombinases are applied.

Now to prove the claim: Given a RSM with N input-driven recombinases and one pair of *attB-attP* per recombinase on its register, all states must be accessible by some irreducible string of recombinases (statement 1), and all irreducible strings from the same subset of the N recombinases must generate the same state (statement 2). Therefore, there cannot be more states than there are subsets of recombinases, which is 2^N (see text S2 for a more detailed version of this proof).

More generally, this proof can be expanded to show that, given k pairs of orthogonal *attB-attP* pairs per recombinase on a register, the number of states it can access will never exceed 2^{kN} (see text S3). For large serine recombinases, there is a limit of $k = 6$ orthogonal and directional *attB-attP* pairs for a given recombinase (*31*). Therefore, the information capacity of RSMs using large serine recombinases is intrinsically bound exponentially.

There are still many unanswered mathematical questions regarding RSM structure. For example, given a DNA sequence, what is the computational difficulty of deciding whether it admits an irreducible ordering of a set of recombinases? Is this problem NP-hard? Also, how can we decide whether an irreducible string of recombinases has minimal length, or whether there might be a shorter irreducible string that produces the same DNA sequence?

encoding the identities and order of biological events into selectable or sortable reporters, or targeting genetic perturbations to cells that experience a particular order of biological events. Gene regulation programs can be implemented by incorporating genetic regulatory elements, such as promoters, terminators, and genes, into the registers of our RSMs. The rearrangement of these elements in each state should then alter gene expression in a predictable manner. Such RSMs are a biological realization of Moore machines from automata theory, where each state is associated with a set of outputs (*1*). We refer to them as gene-regulatory RSMs (GRSMs).

To help researchers design circuits for desired gene regulation programs, we created a large, searchable database of two-input, five-state GRSM registers. To compile this GRSM database (Fig. 5), we first enumerated all possible registers that could result from interleaving functionally distinct parts (made from terminators, constitutive promoters, and genes; see text S6 for more details) before and after each recombinase recognition site in our validated five-state register from Fig. 3A. We evaluated each state of each register for gene transcription, and aggregated registers that implement the same gene regulation program. During this evaluation step, we assumed that all genes had bidirectional terminators on their 3′ ends, thus disallowing the possibility of an RNA polymerase traversing a gene (in either direction) to transcribe another gene. We also assumed that each gene in a register was distinct. These assumptions were made to simplify register designs and keep the database at a manageable size for fast computational search.

To avoid redundancy in the database, we removed any register with superfluous parts (containing terminators, promoters, or genes that do not affect gene regulation in any state) if its "parent" register [the same register except without the superfluous part(s)] was also represented in the database. Moreover, all registers that transcribed either no gene or the same gene in every state were removed from the database, as this gene regulation is trivial to implement.

The resulting database (database S1) contains a total of 5,192,819 GRSM registers that implement 174,264 gene regulation programs. Each register is different in the sense that no two registers have all of the same parts in all of the same positions. Registers in the database regulate the transcription of 1 to 14 genes (fig. S7A). A register for any desired program that regulates up to three genes is likely to be in the database, which comprises 100% of possible single-gene regulation programs, 95% of possible two-gene regulation programs, and 61% of possible three-gene regulation programs (fig. S7B). Moreover, 27% of possible four-gene regulation programs are represented in the database, but the percentage drops off steeply beyond that, as the number of possible gene regulation programs grows exponentially with each additional gene (text S7). One could apply straightforward gene replacement principles to go beyond the scope of regulation programs represented in the database—for example, by replacing multiple distinct genes on a register with copies of the same gene, or replacing a gene with a multicistronic operon (fig. S8). To conveniently use the GRSM database for design or exploration, we created a search function that accepts a user-specified gene regulation program and returns all registers from the database that may be used to implement it (Fig. 5 and appendix S1).

To create functional GRSMs in *E. coli*, we implemented the same input-output plasmid scheme as our two-input, five-state RSM (Fig. 3A), except that we substituted registers from our database on the output plasmid. Fluorescent protein (FP) genes were built on the registers to evaluate gene regulation performance. We grew populations of cells treated with all five permuted substrings of the inputs ATc and Ara, and then used flow cytometry on each population to measure the percentage of cells with distinct FP expression profiles (*32*). We successfully implemented four single-gene regulation programs (Fig. 6, A to D) and one multigene regulation program (in which unique subsets of three distinct FPs were expressed in each state;

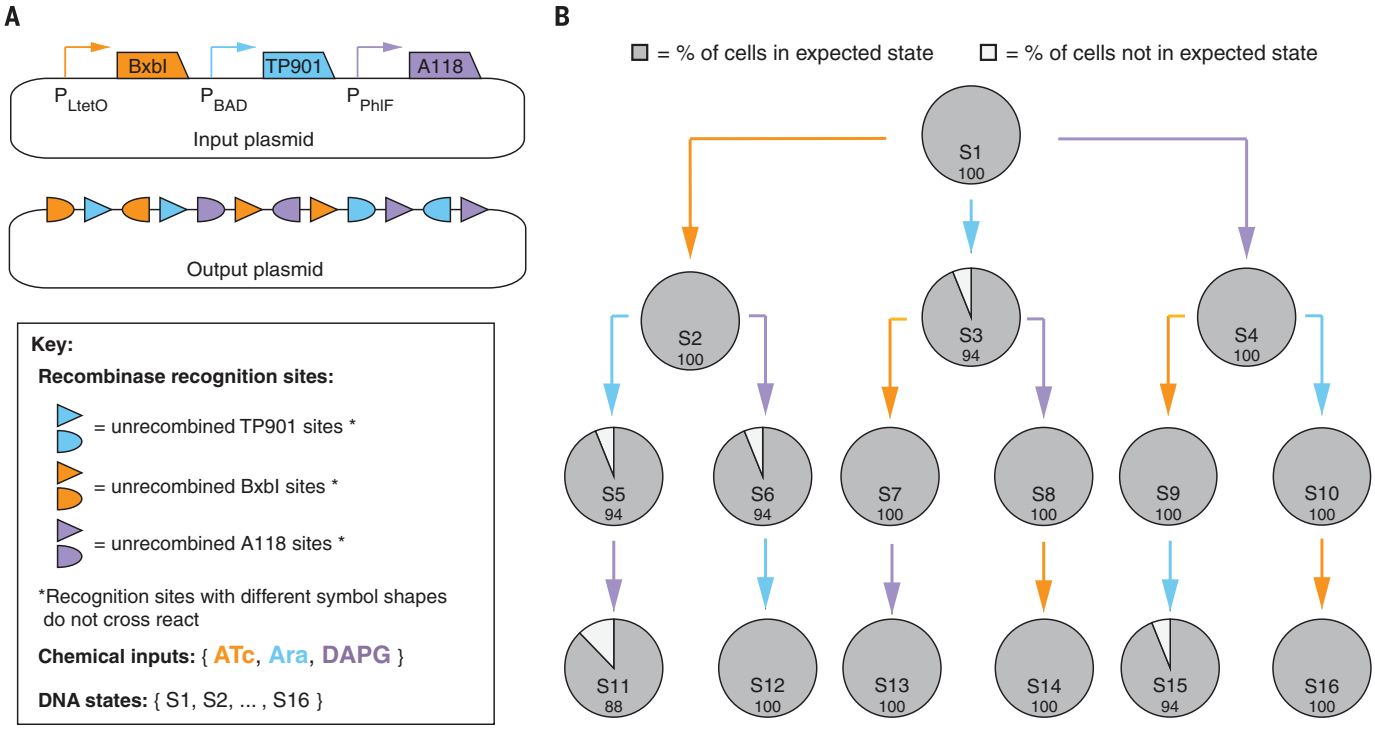


Fig. 4. Scaling to a three-input, 16-state RSM. (**A**) The two plasmids used to implement the RSM. ATc, Ara, and DAPG induce expression of BxbI, TP901, and A118 recombinases, respectively. A detailed state diagram of the register on the output plasmid is shown in fig. S5. (**B**) The performance of the RSM in *E. coli*. Nodes represent populations of cells induced with all permuted substrings of the inputs ATc (orange arrow), Ara (blue arrow), and DAPG (purple arrow). Cultures were treated with saturating concentrations of each input (ATc, 250 ng/ml; Ara, 1% w/v; DAPG, 25 μM) at 30°C for 24 hours in three biological replicates. Node labels indicate the expected state (corresponding to fig. S5) and the percentage of cells in that state as determined by Sanger sequencing of colonies from individual cells in each population (at least 17 cells totaled over all three biological replicates).

Fig. 6E), with at least 94% of cells from each experimental population adopting the expected FP expression profile. These GRSMs enable convenient fluorescent-based reporting on the identity and order of cellular events. For example, the GRSM from Fig. 6E allowed us to evaluate the performance of the underlying RSM with increasing input time durations (by 1-hour steps) by means of flow cytometry (fig. S9). Our findings demonstrated that input durations of 2 hours were sufficient for a majority of cells to adopt their expected state.

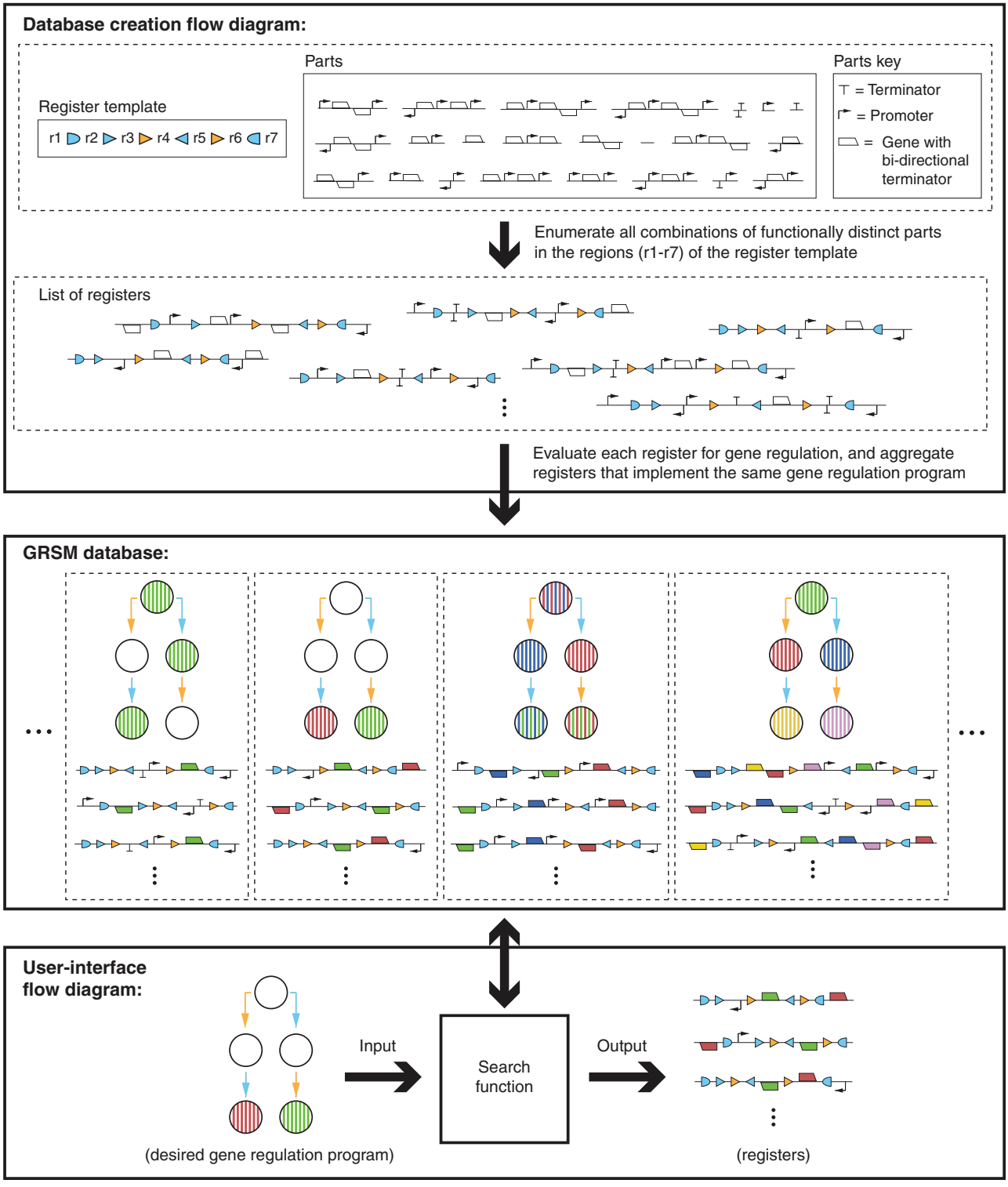


Fig. 5. The GRSM database. (**Top**) Flow diagram depicting how the database was created. (**Middle**) The database has a precompiled list of GRSM registers for distinct gene regulation programs. State diagrams represent gene regulation programs, with each node containing stripes of different colors corresponding to which genes are expressed in that state (no stripes implies no expression of any gene). (**Bottom**) A search function accepts a user-specified gene regulation program and returns registers from the database capable of implementing it.

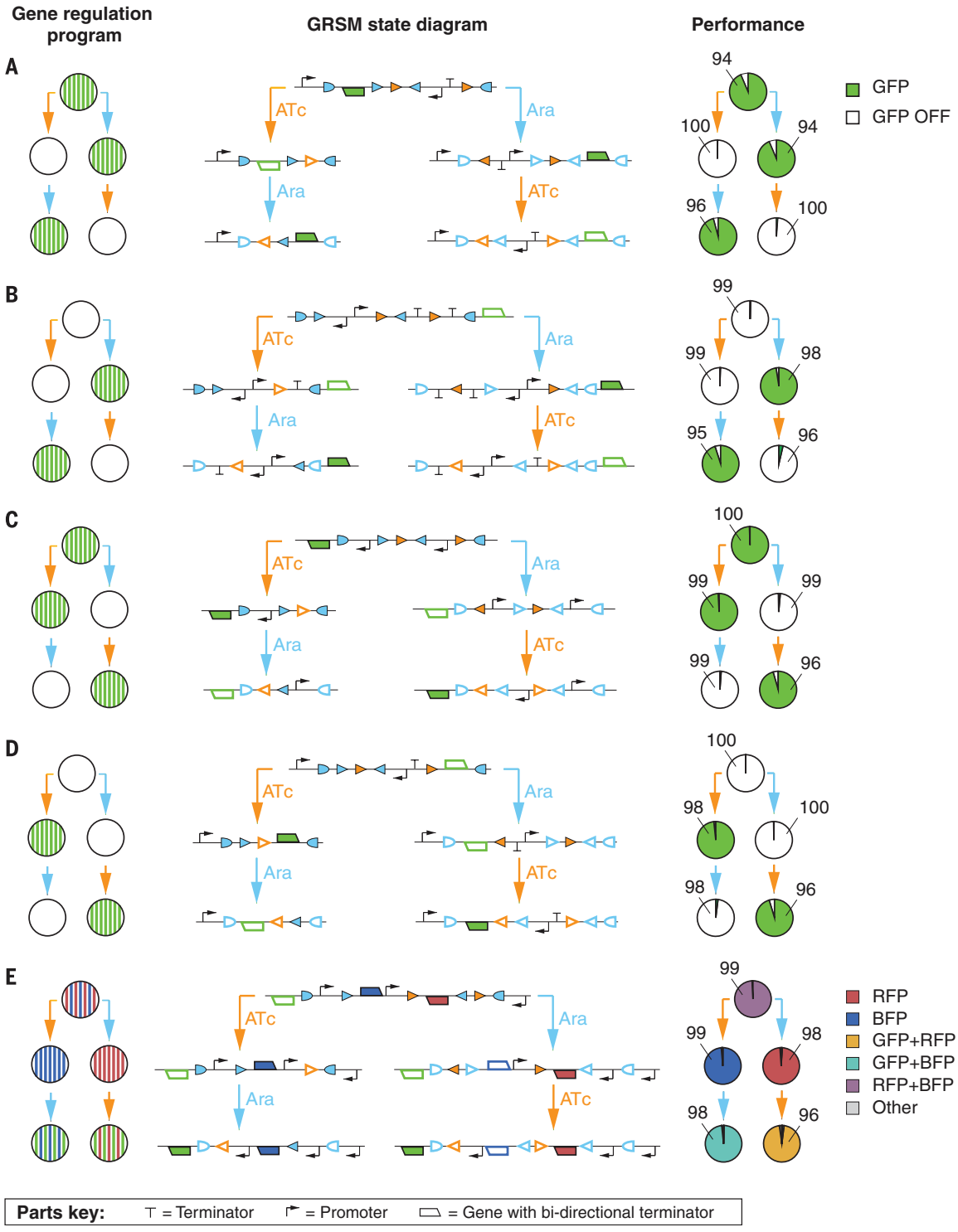


Fig. 6. Implementing two-input, five-state GRSMs. (**A** to **E**) We built GRSMs (one for each panel) in *E. coli* to implement the gene regulation programs depicted at the left, with each node containing stripes of different colors corresponding to which gene products (green, GFP; red, RFP; blue, BFP) are expressed in that state (no stripes implies no expression of any gene). The corresponding GRSM state diagrams are depicted in the middle column, with expressed (ON) fluorescent reporters represented by shaded genes and non-expressed (OFF) fluorescent reporters represented by outlined genes. In the right column, nodes represent populations of cells induced with all permuted substrings of the inputs ATc (orange arrow) and Ara (blue arrow). Cultures were treated with saturating concentrations of each input (ATc, 250 ng/ml; Ara, 1% w/v) at 30°C for 24 hours in three biological replicates. The nodes are shaded according to the percent of cells with different gene expression profiles (ON/OFF combinations of the fluorescent reporters) as measured by flow cytometry. Node labels show the percentage of cells with the expected gene expression profile (averaged over all three biological replicates).

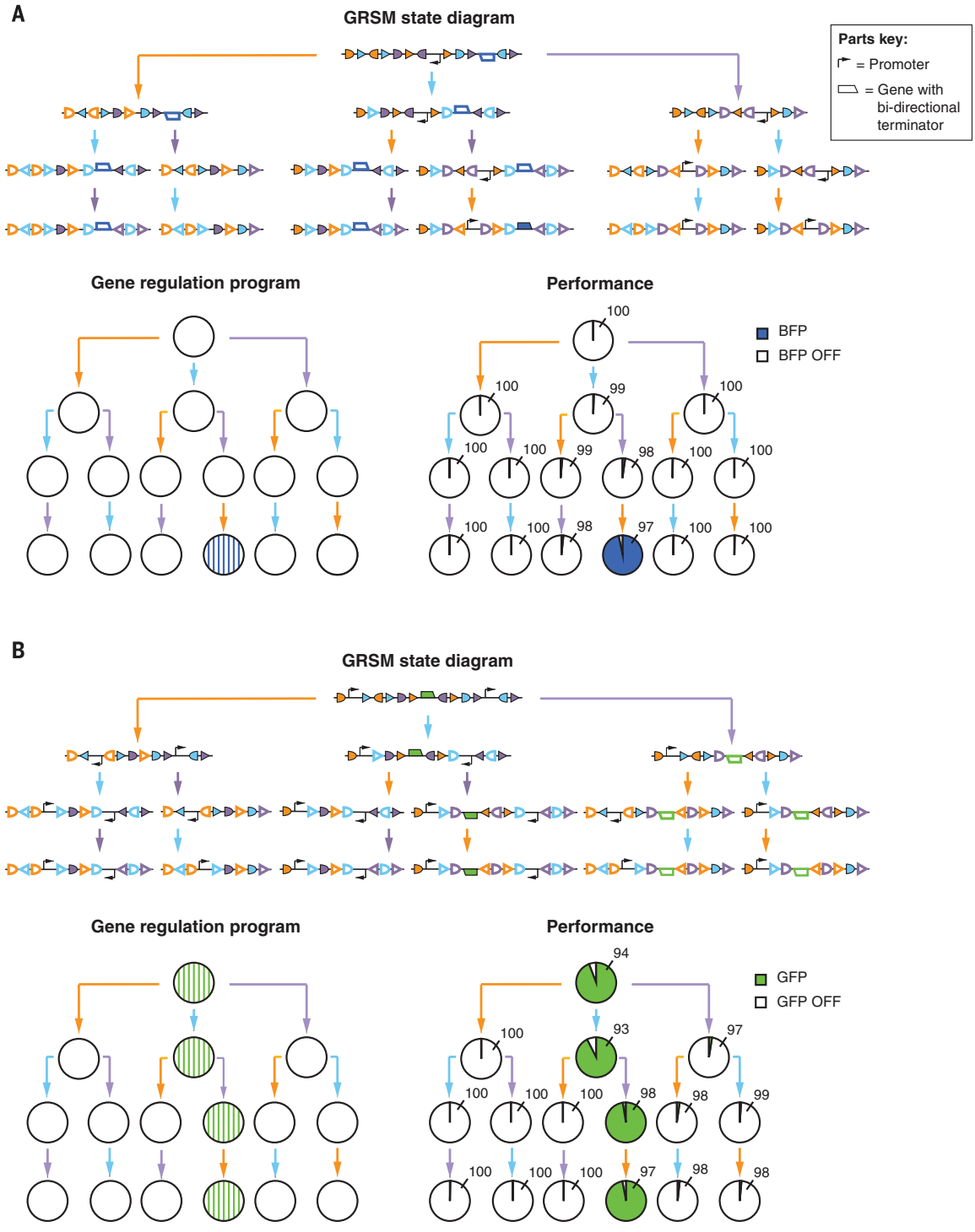


Fig. 7. Implementing three-input, 16-state GRSMs. (**A** and **B**) We built GRSMs in *E. coli* to implement the gene regulation programs depicted at the lower left of each panel, with each node containing stripes of different colors corresponding to which gene products (blue, BFP; green, GFP) are expressed in that state (no stripes implies no expression of any gene). The corresponding GRSM state diagrams are depicted at the top of each panel, with expressed (ON) fluorescent reporters represented by shaded genes and non-expressed (OFF) fluorescent reporters represented by outlined genes. At the lower right of each panel, nodes represent populations of cells induced with all permuted substrings of the inputs ATc (orange arrow), Ara (blue arrow), and DAPG (purple arrow). Cultures were treated with saturating concentrations of each input (ATc, 250 ng/ml; Ara, 1% w/v; DAPG, 25 μM) at 30°C for 24 hours in three biological replicates. The nodes are shaded according to the percentage of cells with or without gene expression as measured by flow cytometry. Node labels show the percentage of cells with the expected gene expression profile (averaged over all three biological replicates).

Because unpredictable behaviors can result when gene regulatory parts are assembled into specific arrangements, certain GRSMs may not implement gene regulation programs as expected. Indeed, this was the case when we initially tested a GRSM that was expected to express green fluorescent protein (GFP) after being exposed to one of two inputs, Ara only or ATc → Ara (fig. S10A) (*32*). Rather than debugging, we constructed two alternative GRSMs using different registers from our database (fig. S10, B and C) that performed better than the initial GRSM, one of which had at least 95% of cells with the expected gene expression profile for each experimental population (fig. S10C). In general, many gene regulation programs represented in our database have multiple possible registers that can implement them (fig. S11). For example, most single-gene regulation programs have at least 373 possible registers, most two-gene regulation programs have at least 55 possible registers, and most three-gene regulation programs have at least 14 possible registers. Even for programs in the database that regulate up to 14 genes, most have at least four possible registers that can implement them. This highly degenerate design space offers a range of GRSM registers that can act as alternatives for one another in the event that a particular register fails to perform to a certain standard. Additional computationally and experimentally derived rules might enable ranking of candidate registers for their likelihood of successful gene regulation function.

To demonstrate the scalability of GRSMs, we built two different three-input, 16-state GRSMs by interleaving genetic parts into the register from Fig. 4A. One GRSM functions as a three-input passcode switch that turns on the expression of a gene (encoding blue fluorescent protein) only when it receives the input Ara → DAPG → ATc (Fig. 7A). The other GRSM expresses a gene (encoding GFP) by default and turns it off if it receives any input that is not along the Ara → DAPG → ATc trajectory (Fig. 7B). Both GRSMs were implemented in *E. coli* and tested with all 16 permuted substrings of the inputs ATc, Ara, and DAPG (*32*). Flow cytometry revealed that at least 93% of cells from each experimental population adopted the expected gene expression profile. Thus, scalable GRSMs that function efficiently can be implemented using our design framework.

Discussion

We created state machines by using recombinases to manipulate DNA registers assembled from overlapping and orthogonal recombinase recognition sites. We used a mathematical framework to analyze the information capacity and scalability of our state machines and understand their limits. For a fixed number of inputs, the information capacity enabled by RSMs is much greater than that of traditional combinational circuits. Furthermore, we created a rich database accessible to the scientific community (database S1 and appendix S1) to enable the automatic design of GRSM registers that implement two-input, five-state gene regulation programs.

We validated our RSM framework by building two-input, five-state and three-input, 16-state RSMs, testing them with Sanger sequencing and qPCR, and applying them to build state-dependent gene regulation programs. Our state machines differ from other strategies for genetic programming, such as combinational Boolean logic gates that are stateless (*33*–*44*), cell counters that do not integrate multiple inputs (*20*), temporal logic circuits that are unable to report on all possible input identities and permutations in a single circuit (*23*), and other multi-input recombinase-based circuits that do not use overlapping recombinase recognition sites and thus cannot perform order-dependent input processing (*21*, *22*).

Although we implemented RSMs in bacteria, we anticipate that our framework will be extensible to other organisms in which recombinases are functional. For example, the large serine recombinases used here (BxbI, TP901, and A118), as well as ϕC31, ϕFC1, ϕRV1, U153, and R4, catalyze recombination in mammalian cells (*45*–*48*). Identification of additional recombinases that function in different organisms should expand the applicability of our framework. The incorporation of reversible recombination events through proteins such as recombination directionality factors could also enable reversible transitions between gene regulatory states (*15*). Depending on desired applications, the prototypical inducible promoters we used here to drive the RSMs could be replaced by sensors that correspond to the desired signals to be recorded. Such sensors need not be based on transcriptional regulation, as long as they can control recombinase activity.

The integration of RSMs into complex systems should enable researchers to investigate temporally distributed events without the need for continual monitoring and/or sampling. For example, by incorporating RSMs into tumor models, scientists may record the identity and order of oncogene activation and tumor suppressor deactivation events in individual cancer cells, and further correlate this information to phenotypic data from transcriptomic analysis or drug assays. In a recent study of myeloproliferative neoplasms containing mutations in both TET2 (a tumor suppressor) and JAK2 (a proto-oncogene), it was discovered that the order in which the mutations occurred played a role in determining disease phenotype, including sensitivity to therapy (*8*). This research underscores the potential impact of order dependencies in other malignancies and the importance of studying them. Cell sorting based on reporter gene expression from GRSMs could be used to separate cells exposed to different identities and orders of gene regulatory perturbations, which could then be further studied to determine functional cellular differences.

Aside from recording and responding to naturally occurring signals, RSMs have potential applications when the signals that control them are applied by a user. For example, RSMs can generate gene expression based not only on simultaneous combinations of inputs, but also on orders of inputs. Thus, they may be useful to bioengineers for programming multiple functions in cell strains for which there are limited numbers of control signals. For example, they could be used to program cell differentiation down many different cell fate paths based on the order and identities of just a few inputs.

Beyond applications in biological research and engineering, our work has also revealed an interesting mathematical structure to recombinase systems. At first glance, the noncommutative behavior of recombinase operations suggests that there might be a superexponential relationship between the number of possible states in a RSM and the number of recombinases it incorporates. Instead, our results show that the number of states is bound exponentially given a finite number of *attB-attP* pairs per recombinase (Box 1 and texts S2 and S3). Many open mathematical problems remain. For example, what is the minimum number of recognition sites on a register needed to implement a particular state machine? Given a gene regulation program of arbitrary scale and complexity, how can we decide whether there exists a corresponding GRSM? We anticipate that solving such problems will be of interest to mathematicians and biologists alike.

Materials and methods

Strains, media, antibiotics, and inducers

All plasmids were implemented and tested in *E. coli* strain DH5αPRO [F-Φ80*lacZ*ΔM15 Δ(*lacZYA-argF*)U169 *deo*R *rec*A1 *end*A1 *hsd*R17(rk^-, mk^+) *pho*A *sup*E44 *thi*-1 *gyr*A96 *rel*A1 λ^-, P_{N25}/tet^R, P_{laciq}/lacI, Sp^r]. All experiments were performed in Azure Hi-Def medium (Teknova, Hollister, CA) supplemented with 0.4% glycerol. For cloning, we used *E. coli* strains DH5αPRO or EPI300 [F-*mcr*A Δ(*mrr-hsd*RMS-*mcr*BC) Φ80lacZM15 Δ*lacX*74 *rec*A1 *end*A1 *ara*D139 Δ(*ara*, *leu*)7697 *gal*U *gal*K λ^- *rps*L (Str^R) *nup*G *trf*A *dhfr*], as indicated below. All cloning was done in Luria-Bertani (LB)–Miller medium (BD Difco) or Azure Hi-Def medium, as indicated below. LB plates were made by mixing LB with agar (1.5% w/v; Apex). For both cloning and experiments, the antibiotics used were chloramphenicol (25 μg/ml) and kanamycin (30 μg/ml). For experiments, the inducers used were ATc (250 ng/ml), Ara (1% w/v), and DAPG (25 μM).

Plasmid construction and cloning

All plasmids were constructed using basic molecular cloning techniques and Gibson assembly (*49*, *50*). Figure S12 shows all plasmids and their relevant parts. Tables S3 and S4 give a list of relevant parts, their sequences, and the sources from which they were derived.

All input plasmids (pNR64 and pNR220) have a kanamycin resistance cassette (*kanR*) and a ColE1 (high copy) origin of replication. The input plasmid pNR64 was adapted from the dual recombinase controller from (*22*) (Addgene #44456). We replaced the chloramphenicol resistance cassette in this dual recombinase controller with *kanR* to make pNR64. To make pNR220, we inserted the PhlF promoter system from (*36*) onto pNR64 to drive the expression of the A118 recombinase, a gift from J. Thomson (USDA-ARS

WRRC, Albany, CA). To control A118 tightly in the absence of any input, we expressed the *phlF* gene (responsible for suppressing transcription from P_{PhlF}) from the strong constitutive proD promoter (*51*). All input plasmids were transformed into chemically competent *E. coli* strain DH5αPRO, and subsequently isolated using the Qiagen QIAprep Spin Miniprep Kit and verified with Sanger sequencing (Quintara Biosciences).

All output plasmids (pNR160, pNR163, pNR164, pNR165, pNR166, pNR186, pNR187, pNR188, pNR291, pNR292, and pNR284) have a chloramphenicol resistance cassette (*camR*) and are built on a bacterial artificial chromosome (BAC) vector backbone to ensure low copy number, as we ideally want ~1 register per cell. The BAC we used is derived from (*52*) and is capable of being induced to a higher copy number with Copy Control (Epicentre) in EPI300 cells. Strings of *attB* and *attP* recognition sites for pNR160 and pNR188 were synthesized from Integrated DNA Technologies and cloned into their respective backbones. For the construction of all GRSM output plasmids (pNR163, pNR164, pNR165, pNR166, pNR186, pNR187, pNR291, pNR292, and pNR284), we interleaved the array of recognition sites on pNR160 (for two-input, five-state) and pNR188 (for three-input, 16-state) with promoters, terminators, and genes using Gibson assembly. In order to prevent unwanted recombination on our plasmids, we avoided reusing identical part sequences on the same plasmid. For promoters, we used proD, BBa_R0051, and BBa_J54200, which have all been previously characterized to have strong expression (*53*). The proD promoter is an insulated promoter, which helps with consistent performance across varying contexts (*51*). We fused the two promoters, BBa_R0051 and BBa_J54200, upstream of 20-nucleotide initial transcribed sequences (ATATAGTGAACAAGGATTAA and ATAGGTTAAAAGCCAGACAT, respectively) characterized in (*54*), and named the concatenated parts proNR3 and proNR4, respectively. We chose terminators for our GRSMs from among the set of validated strong and sequence diverse terminators characterized in (*55*). We often constructed terminators in tandem to increase termination efficiency. Lastly, we used the fluorescent reporter genes *gfpmut3b* (*56*), *mrfp* (*57*), and *mtagbfp* (*58*) to produce outputs. The ribosome binding site (RBS) of each gene was optimized using the Salis Lab RBS calculator (*59*). Upstream of each RBS, we fused a self-cleaving hammerhead ribozyme to prevent the upstream 5′ untranslated transcript region from interfering with translation of the downstream gene (*60*). All output plasmids were transformed into chemically competent *E. coli* strain EPI300 or DH5αPRO, and subsequently isolated using the Qiagen QIAprep Spin Miniprep Kit and verified with Sanger sequencing (Quintara Biosciences).

Like the output plasmids, all plasmids to test the forward (*attB-attP* → *attL-attR*) and reverse (*attL-attR* → *attB-attP*) recombination efficiencies for each recombinase used in this study (see fig. S1) have *camR* and are built on a BAC. The forward reaction test plasmids (pNR230 for BxbI, pNR239 for A118, and pNR276 for TP901) were each constructed with a reverse-oriented *gfpmut3b* (attached to the same RBS and ribozyme as on the output plasmids described above) downstream of a forward-oriented proD promoter, and with anti-aligned *attB* and *attP* sites for the cognate recombinase flanking the gene. Each forward reaction test plasmid was transformed into chemically competent *E. coli* strain DH5αPRO, and subsequently isolated using the Qiagen QIAprep Spin Miniprep Kit and verified with Sanger sequencing (Quintara Biosciences). To generate the reverse reaction test plasmids (pNR279 for BxbI, pNR280 for A118, and pNR287 for TP901), we transformed each forward reaction test plasmid into chemically competent *E. coli* strain DH5αPRO containing the pNR220 input plasmid, induced the cognate recombinase for each test plasmid, and isolated the recombined plasmid from cells using the Qiagen QIAprep Spin Miniprep Kit. Each reverse reaction test plasmid was then transformed into chemically competent *E. coli* strain DH5αPRO, and subsequently isolated again using the Qiagen QIAprep Spin Miniprep Kit and verified with Sanger sequencing (Quintara Biosciences). The second transformation and isolation step for these test plasmids was done to separate them from the pNR220 plasmid, which inevitably was present in the purified DNA solution after the first isolation step.

RSM implementation

All RSMs were implemented with a two-plasmid system (an input plasmid and an output plasmid). Table S5 shows each RSM and the names of the input and output plasmids used to implement them. All two-input RSMs used the pNR64 input plasmid with various output plasmids depending on the desired gene regulation program. All three-input RSMs used the pNR220 input plasmid with various output plasmids depending on the desired gene regulation program.

For the two-input, five-state RSMs, the input plasmid (pNR64) and the output plasmid were simultaneously transformed into chemically competent *E. coli* DH5αPRO cells. Post-transformation, the cells were plated on LB plates with chloramphenicol and kanamycin. Colonies from these plates were used to initiate RSM testing experiments (see below).

For the three-input, 16-state RSMs, we first transformed the input plasmid (pNR220) into chemically competent *E. coli* DH5αPRO cells and plated the transformants onto LB plates with kanamycin. Subsequently, we inoculated a colony in Azure Hi-Def medium (with kanamycin) and grew it overnight at 37°C, then diluted it 1:2000 into fresh medium (same as the overnight) and let it regrow at 37°C to an OD_{600} of 0.2 to 0.5. The cells from this culture were then made chemically competent and transformed with the output plasmid. The purpose for the sequential transformation in this case was to allow time for the *phlF* gene (on the input plasmid) to be expressed at a high enough level to suppress expression of the A118 recombinase from the P_{PhlF} promoter (also on the input plasmid). This was to ensure minimal recombinase levels when the output plasmid was introduced into the system; otherwise the register on the output plasmid could have falsely recorded a chemical induction event prior to its actual occurrence. After transformation of the output plasmid, the cells were plated on an LB plate with chloramphenicol and kanamycin. Colonies from these plates were used to initiate RSM testing experiments (see below).

Experiment for testing the two-input, five-state RSM from Fig. 3A

To test the two-input, five-state RSM for one biological replicate, a colony of *E. coli* cells containing input plasmid pNR64 and output plasmid pNR160 was inoculated into medium with kanamycin and chloramphenicol, grown overnight (~18 hours) at 37°C, and subjected to two rounds of induction followed by a round of outgrowth. For the first round of induction, the overnight culture was diluted 1:250 into medium with no inducer, medium with ATc, and medium with Ara, and grown at 30°C for 18 hours. For the second round of induction, these three cultures were then diluted again 1:250 into fresh medium; the non-induced culture was diluted into medium with no inducer again, the ATc-induced culture was diluted into medium with no inducer and medium with Ara, and the Ara-induced culture was diluted into medium with no inducer and medium with ATc. These cultures were again grown at 30°C for 18 hours. The resulting cultures represented five populations of cells treated with all five permuted substrings of the inputs ATc and Ara. Lastly, for the outgrowth, these cultures were diluted 1:250 into medium with no inducer and grown at 37°C for ~18 hours. The purpose of this final outgrowth was to allow all cell populations to normalize to conditions without inducer, such that detected differences between populations could be attributed to their history of inputs rather than their current environment. This experiment was repeated with a different starting colony for each biological replicate. All cultures were grown in 250 μl of medium (in 96-well plates) shaken at 900 rpm. All media contained chloramphenicol and kanamycin. Final populations from the experiment were analyzed with sequencing assays and qPCR assays (see below).

Sequencing assay for testing the two-input, five-state RSM from Fig. 3A

For the sequencing assay, each of the five experimental populations described above (from each of three biological replicates) were diluted $1:10^6$, plated (100 μl) onto LB plates with chloramphenicol and kanamycin, and grown overnight at 37°C such that each resulting colony represented the clonal population of a single cell from each experimental population. The register region on the output plasmid for around 24 (at least 22) colonies from each plate (experimental population) was amplified with colony PCR and sent for Sanger sequencing (Quintara Biosciences). Chromatograms from the sequencing reactions were aligned to the expected register sequence to determine whether they matched. Results from all three replicates were totaled, and the percent

of cells matching their expected sequence is displayed in Fig. 3B.

qPCR assay for testing the two-input, five-state RSM from Fig. 3A

For the qPCR assay, plasmids from each of the five experimental populations described above (from each of three biological replicates) were isolated with the QIAprep Spin Miniprep Kit and used as template in qPCR reactions. All qPCR reactions were performed on the Roche LightCycler 96 Real-Time System using KAPA SYBR FAST Master Mix and according to Kapa Biosystems' recommended protocol (200 nM each primer, 10 μl of 2× master mix, and no more than 20 ng of template in a 20-μl reaction). Each template was qPCR-amplified with each of three primer pairs (pp1, pp2, and pp3) elucidated by PSIT (described below; see appendix S2 for the program), as well as a normalizing primer pair (ppN) that amplified the backbone of the output plasmid. Figure S13 shows the regions on the register to which the three PSIT primer pairs bind and the register states that they are supposed to amplify. Table S6 gives the primer sequences. Along with the experimental templates, we also ran qPCR reactions of each primer pair with control template made up entirely of output plasmid containing register state S3 (fig. S13) that would get amplified by each primer pair. We isolated this output plasmid from our Ara-treated *E. coli* population and sequence-verified it to make sure that the register state matched S3. We calculated the "fractional amount" of output plasmid amplified by each primer pair (pp1, pp2, or pp3) for each experimental template (t1, t2, t3, t4, or t5) as

$$f_{tx,\,ppy} = 2^{(Cq_{tx,\,ppn} - Cq_{tc,\,ppn}) - (Cq_{tx,\,ppy} - Cq_{tc,\,ppy})}$$

where *tx* is the experimental template of interest (t1, t2, t3, t4, or t5), *ppy* is the primer pair of interest (pp1, pp2, or pp3), *tc* is the control template (output plasmid in S3), *ppn* is the normalizing primer pair (ppN), and *Cq* is the Cq value from the qPCR reaction of the template and primer pair indicated in its subscript.

From these $f_{tx,\,ppy}$ values, we created a qPCR result vector for each experimental template, $\mathbf{f}_{tx}$:

$$\mathbf{f}_{tx} = [f_{tx,\,pp1},\, f_{tx,\,pp2},\, f_{tx,\,pp3}]$$

This result vector was compared to the theoretical result vector that we would get if the template were made up entirely of a register from one particular state in our RSM, $\mathbf{f}_{ts}$:

$$\mathbf{f}_{ts} = [f_{ts,\,pp1},\, f_{ts,\,pp2},\, f_{ts,\,pp3}]$$

where *ts* is the template made entirely of register from one state (S1, S2, S3, S4, or S5). The $f_{ts,\,ppy}$ values are 0 or 1 depending on whether the particular primer pair *ppy* amplifies that state (fig. S13). The similarity of $\mathbf{f}_{tx}$ to $\mathbf{f}_{ts}$ was quantified by Euclidean distance, $D_{tx,\,ts}$:

$$D_{tx,\,ts} = |\mathbf{f}_{tx} - \mathbf{f}_{ts}|$$

$$= \sqrt{(f_{tx,\,pp1} - f_{ts,\,pp1})^2 + (f_{tx,\,pp2} - f_{ts,\,pp2})^2 + (f_{tx,\,pp3} - f_{ts,\,pp3})^2}$$

The Euclidean distances between the qPCR result vectors of each experimentally derived template and the theoretical qPCR result vectors of each state are displayed in a heat map in fig. S4 for each of three biological replicates.

Experiment for testing the three-input, 16-state RSM from Fig. 4A

To test the three-input, 16-state RSM for one biological replicate, a colony of *E. coli* cells containing input plasmid pNR220 and output plasmid pNR188 was inoculated into medium with kanamycin and chloramphenicol, grown overnight (~18 hours) at 37°C, and subjected to three rounds of induction followed by a round of outgrowth. For the first round of induction, the overnight culture was diluted 1:250 into medium with no inducer, medium with ATc, medium with Ara, and medium with DAPG, and grown at 30°C for 24 hours. For the second round of induction, these four cultures were then diluted again 1:250 into fresh media: The noninduced culture was diluted into medium with no inducer; the ATc-induced culture was diluted into medium with no inducer, medium with Ara, and medium with DAPG; the Ara-induced culture was diluted into medium with no inducer, medium with ATc, and medium with DAPG; and the DAPG-induced culture was diluted into medium with no inducer, medium with ATc, and medium with Ara. These cultures were again grown at 30°C for 24 hours. For the third round of induction, each of these 10 cultures were diluted again 1:250 into fresh media: The noninduced → noninduced, ATc → noninduced, Ara → noninduced, and DAPG → noninduced cultures were diluted into medium with no inducer; the ATc → Ara and Ara → ATc cultures were diluted into medium with no inducer and medium with DAPG; the ATc → DAPG and DAPG → ATc cultures were diluted into medium with no inducer and medium with Ara; and the Ara → DAPG and DAPG → Ara cultures were diluted into medium with no inducer and medium with ATc. These cultures were again grown at 30°C for 24 hours. The resulting cultures represented 16 populations of cells treated with all 16 permuted substrings of the inputs ATc, Ara, and DAPG. Lastly, for the outgrowth, these cultures were diluted 1:250 into medium with no inducer and grown at 37°C for 18 hours. This experiment was repeated with a different starting colony for each biological replicate. All cultures were grown in 250 μl of medium (in 96-well plates) shaken at 900 rpm. All media contained chloramphenicol and kanamycin. Final populations from the experiment were analyzed with sequencing assays and qPCR assays (see below).

Sequencing assay for testing the three-input, 16-state RSM from Fig. 4A

For the sequencing assay, each of the 16 experimental populations described above (from each of three biological replicates) were diluted $1{:}10^6$, plated (100 μl) onto LB plates with chloramphenicol and kanamycin, and grown overnight at 37°C such that each resulting colony represented the clonal population of a single cell from each experimental population. The register region on the output plasmid for five or six colonies from each plate (experimental population) was amplified with colony PCR and sent for Sanger sequencing (Quintara Biosciences). Chromatograms from the sequencing reactions were aligned to the expected register sequence to determine whether they matched. Results from all three biological replicates were totaled, and the percent of cells matching their expected sequence is displayed in Fig. 4B.

qPCR assay for testing the three-input, 16-state RSM from Fig. 4A

For the qPCR assay, plasmids from each of the 16 experimental populations described above (from each of three biological replicates) were isolated with the Qiagen QIAprep Spin Miniprep Kit and used as template in qPCR reactions. As with the two-input, five-state RSM testing, all qPCR reactions were performed on the Roche LightCycler 96 Real-Time System using KAPA SYBR FAST Master Mix and according to the Kapa Biosystems recommended protocol (200 nM each primer, 10 μl of 2× master mix, and no more than 20 ng of template in a 20-μl reaction). Each template was qPCR-amplified with each of six primer pairs (pp1, pp2, pp3, pp4, pp5, and pp6) elucidated by PSIT as well as a normalizing primer pair (ppN) that amplified the backbone of the output plasmid. Figure S14 shows the regions on the register to which the six PSIT primer pairs bind and the register states that they are supposed to amplify. Table S7 gives the actual primer sequences. Similar to the two-input, five-state system, we also ran qPCR reactions of each primer pair with control template made up entirely of output plasmid containing a register that would get amplified by each primer pair. Unlike with the two-input, five-state RSM, however, there was no single register state that would get amplified by each primer pair. So we ended up using an output plasmid in state S2 as a control template for pp1, pp4, and pp5 and an output plasmid in state S8 as a control template for pp2, pp3, and pp6 (fig. S14). The plasmid with register state S2 was isolated from our ATc-treated *E. coli* population (and sequence-verified), and the plasmid with register state S8 was isolated from our Ara → DAPG–treated *E. coli* population (and sequence-verified). We proceeded with calculating the fractional amount of plasmid amplified by each primer pair for each experimental template, and then comparing the data for each template to each theoretical state (with Euclidean distance) the same way as we did for the two-input, five-state RSM, except generalized to six primer pairs and 16 states. That is,

$$\mathbf{f}_{tx} = [f_{tx,\,pp1},\, f_{tx,\,pp2},\, \ldots,\, f_{tx,\,pp6}]$$

$$\mathbf{f}_{ts} = [f_{ts,\,pp1},\, f_{ts,\,pp2},\, \ldots,\, f_{ts,\,pp6}]$$

$$D_{tx,\,ts} = |\mathbf{f}_{tx} - \mathbf{f}_{ts}|$$

$$= \sqrt{(f_{tx,\,pp1} - f_{ts,\,pp1})^2 + \ldots + (f_{tx,\,pp6} - f_{ts,\,pp6})^2}$$

The Euclidean distances between the qPCR result vectors of each experimentally derived template

and the theoretical qPCR result vectors of each state are displayed in a heat map in fig. S6 for each of three biological replicates.

Designing the GRSM registers from Fig. 6 and fig. S10

We inputted our desired gene regulation programs into the database search function [coded in MATLAB R2013b (Mathworks, Natick, MA); appendix S1], and received an output list of registers, from which we chose our candidates for implementation. Table S8 shows the MATLAB search function input matrix we used to specify our desired gene regulation programs, as well as the search function output vectors that we chose as our registers to implement the gene regulation programs, as per the instructions on how to use the search function (appendix S1).

Testing the GRSMs from Fig. 6 and fig. S10

The experiments to test the two-input, five-state GRSMs followed the same format as the experiment to test the two-input, five-state RSM from Fig. 3A, except that we used 24-hour inductions instead of 18-hour inductions for the induction rounds, and instead of analyzing the experimental populations with sequencing and qPCR assays, we used a fluorescence assay (see below).

Testing the GRSMs from Fig. 7

The experiments to test the three-input, 16-state GRSMs followed the same format as the experiment to test the three-input, 16-state RSM from Fig. 4A, except that instead of analyzing the experimental populations with sequencing and qPCR assays, we used a fluorescence assay (see below).

Testing the reversibility of BxbI, TP901, and A118 in fig. S1

For each recombinase in our study (BxbI, TP901, and A118), we isolated two plasmids that were recombined versions of each other: one with *attB-attP* and no GFP expression (pNR230 for BxbI, pNR239 for A118, and pNR276 for TP901), and the other with *attL-attR* and GFP expression (pNR279 for BxbI, pNR280 for A118, and pNR287 for TP901). We transformed each of these plasmids into chemically competent *E. coli* DH5αPRO containing the input plasmid pNR220 (prepared as described above). To measure recombination for each transformant, a colony was inoculated into media with kanamycin and chloramphenicol, grown overnight (~18 hours) at 37°C, and subjected to a round of induction followed by a round of outgrowth. For the induction, the overnight culture was diluted 1:250 into medium with no inducer and medium with inducer (ATc for BxbI, Ara for TP901, or DAPG for A118) and grown at 30°C for 16 hours. For the outgrowth, these cultures were diluted 1:250 into medium with no inducer and grown at 37°C for 18 hours. This experiment was repeated with a different starting colony for each of three biological replicates. All cultures were grown in 250 μl of medium (in 96-well plates) shaken at 900 rpm. We measured the percentage of cells from each population expressing GFP, as described below.

RSM time course experiment in fig. S9

For one biological replicate, a colony of *E. coli* DH5αPRO cells containing input plasmid pNR64 and output plasmid pNR291 was inoculated into medium with kanamycin and chloramphenicol, grown overnight (~18 hours) at 37°C, rediluted 1:75 into fresh medium, split into 11 cultures, and grown at 30°C. When cells reached an OD_{600} of 0.1, we rediluted cells from one culture 1:125 into fresh medium and let them outgrow at 37°C. This (uninduced) population would become the 0-hour time point in fig. S9, C to E. All other cultures were subjected to induction prior to outgrowth. Ara was directly added to five of the cultures, and ATc was directly added to the other five and they were allowed to continue growing at 30°C. Each of the five cultures for each input would become induction time points separated by 1-hour steps (for each input); we refer to them as input seed cultures. After 1 hour, we diluted cells from one ATc seed culture 1:125 into fresh medium and let them outgrow at 37°C. This would become the 1-hour time point for ATc in fig. S9C. From the same seed culture, we also diluted cells 1:25 into medium with Ara and let them grow for the equivalent amount of input exposure time (1 hour) at 30°C before diluting 1:125 into fresh medium and letting them outgrow at 37°C. This would become the 1-hour time point for ATc → Ara in fig. S9E. Then, for the same seed culture, we directly added Ara and let the cells grow for the equivalent amount of input exposure time (1 hour) at 30°C before diluting 1:125 into fresh medium and letting them outgrow at 37°C. This would become the 1-hour time point for ATc → Ara in fig. S9D. The same procedure was done for an Ara seed culture after 1 hour, except with ATc as the sequentially added input. This process was subsequently repeated at 2 hours with different ATc and Ara seed cultures, and so on for 3, 4, and 5 hours. The outgrowth for all cell populations continued for 16 hours after the final cells were diluted for outgrowth (10 hours after the initial induction began). This experiment was repeated for three biological replicates. All cultures were grown in 250 μl of medium (in 96-well plates) shaken at 900 rpm. All media contained chloramphenicol and kanamycin. Final populations from the experiment were analyzed with flow cytometry (see below).

Fluorescence assay

For all experiments with a fluorescence assay, we diluted cells 1:125 into phosphate-buffered solution (PBS, Research Products International) and ran them on a BD-FACS LSRFortessa-HTS cell analyzer (BD Biosciences). We measured 30,000 cells for each sample and consistently gated by forward scatter and side scatter for all cells in an experiment. GFP (product of *gfpmut3b*) intensity was measured on the FITC channel (488-nm excitation laser, 530/30 detection filter), RFP (product of *mrfp*) intensity was measured on the PE–Texas Red channel (561-nm excitation laser, 610/20 detection filter), and BFP (product of *mtagbfp*) intensity was measured on the PacBlue channel (405-nm excitation laser, 450/50 detection filter). A fluorescence threshold was applied in each channel to determine the percent of cells with expressed (ON) versus not expressed (OFF) fluorescent proteins. The threshold was based on a negative control (*E. coli* DH5αPRO containing pNR64 and a BAC with no fluorescent reporter genes) population, such that 0.1% of these negative control cells were considered to have ON fluorescent protein expression in each channel (corresponding to a 0.1% false positive rate).

All fluorescence-based experiments had three biological replicates. For the recombinase reversibility experiment (fig. S1) and the RSM time course experiment (fig. S9), the data for all three replicates is shown. For the GRSM experiments (Figs. 6 and 7 and fig. S10), the data from all three replicates are averaged. For these experiments, the largest standard error for the percent of any fluorescent subpopulation was 1.22%.

GRSM database and search function

The GRSM database was constructed (as discussed in the main text) using MATLAB R2013b (Mathworks), partly run on the Odyssey cluster supported by the FAS Division of Science, Research Computing Group at Harvard University.

The database contains three arrays: *registerArray* (an array of GRSM registers), *grpArray* (an array of gene regulation programs), and *register2grp* (an array that maps each register in *registerArray* to its corresponding gene regulation program in *grpArray*, by index).

Each gene regulation program in *grpArray* is represented by a 70-element vector of 0s and 1s. Each contiguous stretch of 14 elements belongs to a state—S1, S2, S3, S4, and S5, respectively—corresponding to the states in Fig. 3A. And within each state, each element (1 to 14) represents a gene (G1 to G14, respectively). For example, given a vector in *grpArray*, element 1 represents G1 in S1, element 15 represents G1 in S2, element 29 represent G1 in S3, element 43 represent G1 in S4, element 57 represents G1 in S5, element 2 represents G2 in S1, element 16 represents G2 in S2, and so on. The binary value of each element indicates whether that gene in that particular state is OFF (0) or ON (1). If the value of any given gene in every state in a gene regulation program is 0, then that gene does not exist in the regulation program.

Each register in *registerArray* is represented by a seven-element vector of numbers 1 through 25. Each element of the vector corresponds to a DNA region (a to g) interleaving the recognition sites of the register shown in Fig. 3A. The value of each element (1 to 25) represents a part, as defined in table S9. Each part is made up of genes, terminators, and constitutive promoters, arranged such that each part is functionally distinct (see text S6). Nonpalindromic parts (as indicated in table S9) can appear inverted on the register, in which case they take on a negative value. For example, part 1 is a gene, which is a nonpalindromic part. If it appears as a "1" on an element of a register vector, then it is facing left to right (5′ to 3′), and if it appears as a "–1" on an element of a register vector, then it is facing right to left (5′ to 3′).

Note that all explicitly depicted terminators in the parts (table S9) are unidirectional; thus, transcription can move through them in the reverse direction. However, the unidirectional terminator in part 3 can be replaced by a bidirectional terminator without changing the function of the part. This is because placing an additional terminator upstream of the promoter in part 3 would only terminate transcription that would subsequently be reinitiated in the same direction. Also, the unidirectional nature of part 7 is not always necessary to the gene regulation program of the underlying register. That is, sometimes part 7 (a unidirectional terminator by itself) can be replaced by part 4 (a bidirectional terminator by itself) without affecting the gene regulation implemented by the underlying register. To make this distinction clear to database users, we parsed all occurrences of part 7 in the *registerArray* and replaced it with a special identifier, part 15, if its unidirectional nature is not important to the gene regulation program of the underlying register. Therefore, all occurrences of part 7 in *registerArray* now represent parts that necessitate "terminator read-through" (transcription through their unidirectional terminators in the reverse direction) for the gene regulation program of the underlying register. Likewise, because convergent (face-to-face) promoters can destructively interfere with each other (*61*), we made a special distinction for parts with promoters that necessitate "promoter read-through" (transcription through their promoters in the reverse direction; table S9). Because part 10 (a promoter by itself), depending on its register context, can sometimes necessitate read-through and sometimes not, we parsed all occurrences of part 10 in *registerArray* and replaced it with a special identifier, part 14, if it does not necessitate read-through for the gene regulation program of the underlying register. Therefore, all occurrences of part 10 in *registerArray* now represent parts that necessitate promoter read-through for the proper gene regulation program of the underlying register.

All parts with genes in *registerArray* also have bidirectional terminators on the 3′ ends of those genes. These terminators are not explicitly depicted in table S9. Although the database has otherwise been reduced to avoid superfluous terminators, promoters, and genes, the implicit terminators on the 3′ ends of genes may sometimes be superfluous. That is, they may not be necessary for the proper gene regulation program of the underlying register.

Lastly, the array *register2grp* has the same number of elements as *registerArray*. It maps each register in *registerArray* to a value that is the index of its corresponding gene regulation program in *grpArray*.

We present the database as a MATLAB MAT-file (database S1), where each array is stored in a MATLAB variable. The search function for this MAT-file database was also created in MATLAB R2013b and requires MATLAB software to run. Code for the MATLAB search function and more information on how it works are included in appendix S1.

PCR-based state interrogation tool (PSIT)

The PSIT algorithm uses an abstract data type—the class *DNARegister*—to represent registers. To determine what sets of primer pairs may be used to uniquely detect an inputted *DNARegister* and all of its recombined states, the algorithm (i) "recombines" the input register, generating *DNARegister* instances for all states that result from any permuted substring of inputs; (ii) generates a list of primer pairs made up of all possible primers that bind to each region between recognition sites and on the terminal ends of the recognition site arrays; (iii) narrows the list to primer pairs that only amplify in any given state when they are on adjacent regions; and (iv) determines all subsets of this final list of primer pairs that can be used to uniquely identify each possible state of the DNA register. This final list of primer pair subsets is then returned as output along with details regarding which primer pairs amplify in which states. For qPCR compatibility purposes, step iii ensures that every amplicon is short and that every primer pair always yields the same amplicon when it amplifies (regardless of state). The PSIT program was implemented in Python 2.7. Code for the PSIT program and more information on how it works are included in appendix S2.

REFERENCES AND NOTES

1. J. E. Hopcroft, J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation* (Addison-Wesley, ed. 1, 1979).
2. S. M. Kaech, W. Cui, Transcriptional control of effector and memory $CD8^+$ T cell differentiation. *Nat. Rev. Immunol.* **12**, 749–761 (2012). doi: 10.1038/nri3307; pmid: 23080391
3. N. Yosef *et al*., Dynamic regulatory network controlling TH17 cell differentiation. *Nature* **496**, 461–468 (2013). doi: 10.1038/nature11981; pmid: 23467089
4. S. Agarwal, K. L. Holton, R. Lanza, Efficient differentiation of functional hepatocytes from human embryonic stem cells. *Stem Cells* **26**, 1117–1127 (2008). doi: 10.1634/stemcells.2007-1102; pmid: 18292207
5. C. E. Murry, G. Keller, Differentiation of embryonic stem cells to clinically relevant populations: Lessons from embryonic development. *Cell* **132**, 661–680 (2008). doi: 10.1016/j.cell.2008.02.008; pmid: 18295582
6. T. Brambrink *et al*., Sequential expression of pluripotency markers during direct reprogramming of mouse somatic cells. *Cell Stem Cell* **2**, 151–159 (2008). doi: 10.1016/j.stem.2008.01.004; pmid: 18371436
7. R. Jaenisch, R. Young, Stem cells, the molecular circuitry of pluripotency and nuclear reprogramming. *Cell* **132**, 567–582 (2008). doi: 10.1016/j.cell.2008.01.015; pmid: 18295576
8. C. A. Ortmann *et al*., Effect of mutation order on myeloproliferative neoplasms. *N. Engl. J. Med.* **372**, 601–612 (2015). doi: 10.1056/NEJMoa1412098; pmid: 25671252
9. E. Fokas, W. G. McKenna, R. J. Muschel, The impact of tumor microenvironment on cancer treatment and its modulation by direct and indirect antivascular strategies. *Cancer Metastasis Rev.* **31**, 823–842 (2012). doi: 10.1007/s10555-012-9394-4; pmid: 22825313
10. A. N. Hata *et al*., Tumor cells can follow distinct evolutionary paths to become resistant to epidermal growth factor receptor inhibition. *Nat. Med.* **22**, 262–269 (2016). doi: 10.1038/nm.4040; pmid: 26828195
11. J. Shah, P. T. Desai, D. Chen, J. R. Stevens, B. C. Weimer, Preadaptation to cold stress in *Salmonella enterica* serovar Typhimurium increases survival during subsequent acid stress exposure. *Appl. Environ. Microbiol.* **79**, 7281–7289 (2013). doi: 10.1128/AEM.02621-13; pmid: 24056458
12. R. Roemhild, C. Barbosa, R. E. Beardmore, G. Jansen, H. Schulenburg, Temporal variation in antibiotic environments slows down resistance evolution in pathogenic *Pseudomonas aeruginosa*. *Evol. Appl.* **8**, 945–955 (2015). doi: 10.1111/eva.12330; pmid: 26640520
13. Y. Benenson, Biomolecular computing systems: Principles, progress and potential. *Nat. Rev. Genet.* **13**, 455–468 (2012). doi: 10.1038/nrg3197; pmid: 22688678
14. K. Oishi, E. Klavins, Framework for engineering finite state machines in gene regulatory networks. *ACS Synth. Biol.* **3**, 652–665 (2014). doi: 10.1021/sb4001799; pmid: 24932713
15. 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
16. 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
17. T. S. Ham, S. K. Lee, J. D. Keasling, A. P. Arkin, A tightly regulated inducible expression system utilizing the fim inversion recombination switch. *Biotechnol. Bioeng.* **94**, 1–4 (2006). doi: 10.1002/bit.20916; pmid: 16534780
18. L. Yang *et al*., Permanent genetic memory with >1-byte capacity. *Nat. Methods* **11**, 1261–1266 (2014). doi: 10.1038/nmeth.3147; pmid: 25344638
19. L. Prochazka, B. Angelici, B. Haefliger, Y. Benenson, Highly modular bow-tie gene circuits with programmable dynamic behaviour. *Nat. Commun.* **5**, 4729 (2014). doi: 10.1038/ncomms5729; pmid: 25311543
20. A. E. Friedland *et al*., Synthetic gene networks that count. *Science* **324**, 1199–1202 (2009). doi: 10.1126/science.1172005; pmid: 19478183
21. 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
22. J. Bonnet, P. Yin, M. E. Ortiz, P. Subsoontorn, D. Endy, Amplifying genetic logic gates. *Science* **340**, 599–603 (2013). pmid: 23539178
23. V. Hsiao, Y. Hori, P. W. K. Rothemund, R. M. Murray, A population-based temporal logic gate for timing and recording chemical events. *Mol. Syst. Biol.* **12**, 869 (2016). doi: 10.15252/msb.20156663; pmid: 23539178
24. N. D. F. Grindley, K. L. Whiteson, P. A. Rice, Mechanisms of site-specific recombination. *Annu. Rev. Biochem.* **75**, 567–605 (2006). doi: 10.1146/annurev.biochem.73.011303.073908; pmid: 16756503
25. W. R. A. Brown, N. C. O. Lee, Z. Xu, M. C. M. Smith, Serine recombinases as tools for genome engineering. *Methods* **53**, 372–379 (2011). doi: 10.1016/j.ymeth.2010.12.031; pmid: 21195181
26. H. M. Thorpe, M. C. Smith, *In vitro* site-specific integration of bacteriophage DNA catalyzed by a recombinase of the resolvase/invertase family. *Proc. Natl. Acad. Sci. U.S.A.* **95**, 5505–5510 (1998). doi: 10.1073/pnas.95.10.5505; pmid: 9576912
27. P. Ghosh, N. R. Pannunzio, G. F. Hatfull, Synapsis in phage Bxb1 integration: Selection mechanism for the correct pair of recombination sites. *J. Mol. Biol.* **349**, 331–348 (2005). doi: 10.1016/j.jmb.2005.03.043; pmid: 15890199
28. P. A. Rowley, M. C. A. Smith, E. Younger, M. C. M. Smith, A motif in the C-terminal domain of ϕC31 integrase controls the directionality of recombination. *Nucleic Acids Res.* **36**, 3879–3891 (2008). doi: 10.1093/nar/gkn269; pmid: 18502775
29. M. C. A. Smith, R. Till, M. C. M. Smith, Switching the polarity of a bacteriophage integration system. *Mol. Microbiol.* **51**, 1719–1728 (2004). doi: 10.1111/j.1365-2958.2003.03942.x; pmid: 15009897
30. P. Ghosh, L. A. Bibb, G. F. Hatfull, Two-step site selection for serine-integrase-mediated excision: DNA-directed integrase conformation and central dinucleotide proofreading. *Proc. Natl. Acad. Sci. U.S.A.* **105**, 3238–3243 (2008). doi: 10.1073/pnas.0711649105; pmid: 18299577
31. S. D. Colloms *et al*., Rapid metabolic pathway assembly and modification using serine integrase site-specific recombination. *Nucleic Acids Res.* **42**, e23 (2014). doi: 10.1093/nar/gkt1101; pmid: 24225316
32. See the Materials and Methods section.
33. B. Wang, R. I. Kitney, N. Joly, M. Buck, Engineering modular and orthogonal genetic logic gates for robust digital-like synthetic biology. *Nat. Commun.* **2**, 508 (2011). doi: 10.1038/ncomms1516; pmid: 22009040
34. R. Gaber *et al*., Designable DNA-binding domains enable construction of logic circuits in mammalian cells. *Nat. Chem. Biol.* **10**, 203–208 (2014). doi: 10.1038/nchembio.1433; pmid: 24413461

35. J. J. Lohmueller, T. Z. Armel, P. A. Silver, A tunable zinc finger-based framework for Boolean logic computation in mammalian cells. *Nucleic Acids Res.* **40**, 5180–5187 (2012). doi: 10.1093/nar/gks142; pmid: 22323524
36. 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
37. S. Regot *et al*., Distributed biological computation with multicellular engineered networks. *Nature* **469**, 207–211 (2011). doi: 10.1038/nature09679; pmid: 21150900
38. 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
39. 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
40. M. N. Win, C. D. Smolke, Higher-order cellular information processing with synthetic RNA devices. *Science* **322**, 456–460 (2008). doi: 10.1126/science.1160311; pmid: 18927397
41. 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
42. S. Ausländer, D. Ausländer, M. Müller, M. Wieland, M. Fussenegger, Programmable single-cell mammalian biocomputers. *Nature* **487**, 123–127 (2012).pmid: 22722847
43. J. M. Callura, D. J. Dwyer, F. J. Isaacs, C. R. Cantor, J. J. Collins, Tracking, tuning, and terminating microbial physiology using synthetic riboregulators. *Proc. Natl. Acad. Sci. U.S.A.* **107**, 15898–15903 (2010). doi: 10.1073/pnas.1009747107; pmid: 20713708
44. J. Hasty, D. McMillen, J. J. Collins, Engineered gene circuits. *Nature* **420**, 224–230 (2002). doi: 10.1038/nature01257; pmid: 12432407
45. A. C. Groth, E. C. Olivares, B. Thyagarajan, M. P. Calos, A phage integrase directs efficient site-specific integration in human cells. *Proc. Natl. Acad. Sci. U.S.A.* **97**, 5995–6000 (2000). doi: 10.1073/pnas.090527097; pmid: 10801973
46. E. C. Olivares, R. P. Hollis, M. P. Calos, Phage R4 integrase mediates site-specific integration in human cells. *Gene* **278**, 167–176 (2001). doi: 10.1016/S0378-1119(01)00711-9; pmid: 11707334
47. S. M. Stoll, D. S. Ginsburg, M. P. Calos, Phage TP901-1 site-specific integrase functions in human cells. *J. Bacteriol.* **184**, 3657–3663 (2002). doi: 10.1128/JB.184.13.3657-3663.2002; pmid: 12057961
48. A. Keravala *et al*., A diversity of serine phage integrases mediate site-specific recombination in mammalian cells. *Mol. Genet. Genomics* **276**, 135–146 (2006). doi: 10.1007/s00438-006-0129-5; pmid: 16699779
49. J. Sambrook, E. Fritsch, T. Maniatis, *Molecular Cloning: A Labratory Manual* (Cold Spring Harbor Laboratory Press, ed. 2, 1989).
50. D. G. Gibson *et al*., Enzymatic assembly of DNA molecules up to several hundred kilobases. *Nat. Methods* **6**, 343–345 (2009). doi: 10.1038/nmeth.1318; pmid: 19363495
51. J. H. Davis, A. J. Rubin, R. T. Sauer, Design, construction and characterization of a set of insulated bacterial promoters. *Nucleic Acids Res.* **39**, 1131–1141 (2011). doi: 10.1093/nar/gkq810; pmid: 20843779
52. J. Wild, Z. Hradecna, W. Szybalski, Conditionally amplifiable BACs: Switching from single-copy to high-copy vectors and genomic clones. *Genome Res.* **12**, 1434–1444 (2002). doi: 10.1101/gr.130502; pmid: 12213781
53. iGem Registry of Standard Biological Parts (parts.igem.org).
54. L. M. Hsu *et al*., Initial transcribed sequence mutations specifically affect promoter escape properties. *Biochemistry* **45**, 8841–8854 (2006). doi: 10.1021/bi060247u; pmid: 16846227
55. 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
56. B. P. Cormack, R. H. Valdivia, S. Falkow, FACS-optimized mutants of the green fluorescent protein (GFP). *Gene* **173**, 33–38 (1996). doi: 10.1016/0378-1119(95)00685-0; pmid: 8707053
57. R. E. Campbell *et al*., A monomeric red fluorescent protein. *Proc. Natl. Acad. Sci. U.S.A.* **99**, 7877–7882 (2002). doi: 10.1073/pnas.082243699; pmid: 12060735
58. O. M. Subach *et al*., Conversion of red fluorescent protein into a bright blue probe. *Chem. Biol.* **15**, 1116–1124 (2008). doi: 10.1016/j.chembiol.2008.08.006; pmid: 18940671
59. 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
60. 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
61. B. P. Callen, K. E. Shearwin, J. B. Egan, Transcriptional interference between convergent promoters caused by elongation over the promoter. *Mol. Cell* **14**, 647–656 (2004). doi: 10.1016/j.molcel.2004.05.010; pmid: 15175159

ACKNOWLEDGMENTS

The dual recombinase controller was a gift from D. Endy (Addgene plasmid #44456) and is available at Addgene under a material transfer agreement. All plasmids created in this project are also available on Addgene. We thank J. Thomson (USDA-ARS WRRC, Albany, CA) for the *a118* gene, A. A. K. Nielsen and C. A. Voigt for the *phlF* gene, J. Shin and C. A. Voigt for hammerhead ribozyme parts, C. J. McClune and C. A. Voigt for helpful discussions on recombinase systems, and the FAS Division of Science, Research Computing Group, at Harvard University for supporting computations involved in the creation of the GRSM database on the Odyssey cluster. A MAT file containing the GRSM database is provided in database S1. Supported by the Ford Foundation Predoctoral Fellowships Program and Molecular Biophysics Training Grant NIH/NIGMS T32 GM008313 (N.R.); the NSF Alan T. Waterman Award, grant 1249349 (S.A.); the Center for Microbiome Informatics and Therapeutics; and NIH grants DP2 OD008435 and P50 GM098792, Office of Naval Research grants N00014-13-1-0424 and N0001411110725, NSF grant MCB-1350625, and an NSF Expeditions in Computing Program Award (1522074) as part of the Living Computing Project. The project or effort depicted was or is also sponsored by the Defense Advanced Research Projects Agency (HR0011-15-C-0091). The content of the information does not necessarily reflect the position or the policy of the U.S. government. T.K.L. and N.R. have filed a patent application based on this work with the US Patent and Trademark Office. Author contributions: N.R. and T.K.L. conceived the work; N.R., A.P.S., and A.C.F. performed the cloning and experiments; N.R. performed data analysis and developed the GRSM database and search function; A.P.S. developed the PSIT program; S.A. provided the mathematical analysis; and all authors reviewed the paper.

SUPPLEMENTARY MATERIALS

www.sciencemag.org/content/353/6297/aad8559/suppl/DC1
Texts S1 to S8
Figs. S1 to S14
Tables S1 to S10
Database S1
Appendices S1 and S2
Reference (*62*)

13 November 2015; accepted 2 June 2016
10.1126/science.aad8559

Synthetic recombinase-based state machines in living cells

Nathaniel Roquet, Ava P. Soleimany, Alyssa C. Ferris, Scott Aaronson and Timothy K. Lu

Science **353** (6297), aad8559.
DOI: 10.1126/science.aad8559

Building a computing system in bacteria

Finite state machines are logic circuits with a predetermined sequence of actions that are triggered depending on the starting conditions. They are used for a variety of devices and biological systems, from vending machines to neural circuits. Roquet *et al.* have taken a finite state machine approach to control the expression of integrases, or enzymes that insert or excise phage DNA into or out of bacterial chromosomes. The integrases altered the DNA sequence of a plasmid to record all five possible combinations of two inputs. Such circuits can be used to record the states that the cell experienced over time and can be deployed in state-dependent gene expression programs.

Science, this issue p. 363