nature chemical biology

Article

https://doi.org/10.1038/s41589-024-01730-1

Partitioning of a 2-bit hash function across 66 communicating cells

Received: 22 December 2023

Accepted: 14 August 2024

Published online: 24 September 2024

Check for updates

Jai P. Padmakumar[1,2,8], Jessica J. Sun[2,8], William Cho[3], Yangruirui Zhou[4], Christopher Krenz[4], Woo Zhong Han[5], Douglas Densmore[4,6], Eduardo D. Sontag[3,7] & Christopher A. Voigt[1,2]

Powerful distributed computing can be achieved by communicating cells that individually perform simple operations. Here, we report design software to divide a large genetic circuit across cells as well as the genetic parts to implement the subcircuits in their genomes. These tools were demonstrated using a 2-bit version of the MD5 hashing algorithm, which is an early predecessor to the cryptographic functions underlying cryptocurrency. One iteration requires 110 logic gates, which were partitioned across 66 *Escherichia coli* strains, requiring the introduction of a total of 1.1 Mb of recombinant DNA into their genomes. The strains were individually experimentally verified to integrate their assigned input signals, process this information correctly and propagate the result to the cell in the next layer. This work demonstrates the potential to obtain programable control of multicellular biological processes.

The complexity of the natural world, from the development of body plans to the computational power of the brain, arises from distributed computation performed by many communicating cells[1–7]. If the computational power of a cell population were harnessed, it could solve hard and energy-intensive problems, especially if they required repetitive operations[1,6,8,9]. Cryptographic hash functions, which are used in encryption and are well known from cryptocurrency, are one such example. They secure data by mapping data of arbitrary size ('a message') to a fixed size value (the 'hash'). Beyond solving computational problems, fully realizing the potential of engineered biology will require programming cell communities to coordinate their actions, such as by growing into a living structure.

Synthetic genetic circuits can be used to program a cell to execute a desired computational operation[10–12]. Their construction requires the balancing of interacting regulators and the selection of many genetic parts. This process was simplified by Cello automation software, in which a user specifies the operation using a high-level textual language (Verilog) that is mapped to a DNA sequence[13,14]. Logic minimization algorithms deconstruct the circuit into gates to which regulators are assigned. However, the size of a circuit that can be placed into one cell is limited because its function is performed by freely diffusing molecules that can cross-react[15,16]. In addition, the expression of many regulators burdens individual cells, leading to growth defects, circuit failures and evolutionary breakage[6,17–28]. Methods to reduce burden include integrating circuits into the genome and borrowing paradigms from control theory[29–37]. In practice, these constraints still limit the number of gates per cell to about ten[18].

Transcriptional NOR gates are often used to construct circuits because they require a single repressor and are easily encoded in DNA[38–43]. They are easy to connect to build different circuits by changing the pattern of promoters in front of each repressor gene. Libraries based on different repressor families have been built, but they have various problems restricting their use, such as sensitivity to ligands, large operators that must be inserted into promoters, repetitive domains and retroactivity[16,19,43–54]. The CI repressor from phage λ does not exhibit these problems and was used in many early synthetic biology projects[9,38–40,55–65]. While few homologs have been characterized, evidence indicates their orthogonality and the number of sequenced

[1]MIT Microbiology Program, Massachusetts Institute of Technology, Cambridge, MA, USA. [2]Department of Biological Engineering, Massachusetts Institute of Technology, Cambridge, MA, USA. [3]Department of Bioengineering, Northeastern University, Boston, MA, USA. [4]Department of Electrical and Computer Engineering, Boston University, Boston, MA, USA. [5]Department of Computer Science, Boston University, Boston, MA, USA. [6]Biological Design Center, Boston University, Boston, MA, USA. [7]Department of Electrical and Computer Engineering, Northeastern University, Boston, MA, USA. [8]These authors contributed equally: Jai P. Padmakumar, Jessica J. Sun. e-mail: cavoigt@gmail.com

viral genomes is growing rapidly. Coding theory predicts that up to 80 orthogonal CI repressor variants could be used in a cell[66].

Distributed computing is a powerful approach to problem-solving in which multiple cells collaborate by communicating the states of their circuits[42,67,68]. Information is transmitted by a 'sender device' from one cell that produces a diffusible chemical signal and a 'receiver device' in the next cell that responds to it[5,69–74]. The receiver can be connected to an input of a cellular genetic circuit, whose output can be connected to a sender. Up to four orthogonal sender–receiver pairs have been used together in a cell[41,75,76]. Distributed computing can be used to divide a circuit too large for a single cell across multiple cells[9,58,63,64,77–83]. This strategy reduces the burden on any one cell and improves robustness by requiring consensus. Communicating cells have been used to perform two-input Boolean operations, solve a maze, implement memory and function as a comparator, band-stop filter and adders[39,42,43,84–88]. These multicellular circuits were small, so gate partitioning could be performed easily by hand.

When designing electronic circuits, a common task is to divide a circuit into subcircuits, for example, to distribute circuits that are too large for one chassis (module, chip or board) across multiple chassis[89–92]. Partitioning algorithms convert the circuit into a graph and divide the nodes (gates) across a fixed number of chassis while minimizing the edges (wires) spanning chassis[93]. Many variations of these algorithms and corresponding software tools have been developed, but a shared feature is that they keep the number of chassis fixed[89]. In contrast, when dividing a genetic circuit across cells, the number of gates per cell and number of signaling molecules are constraints, whereas the number of cells (chassis) is variable.

Here, we demonstrate the partitioning of a hash algorithm into subcircuits encoded within *Escherichia coli* genomes, show that all the subcircuits function as designed and provide examples of the propagation of signals over two and three layers. The 128-bit MD5 ('message digest') algorithm has various roles, such as verifying data integrity after transfer, and is a predecessor to the SH256 algorithm underlying Bitcoin. Here, we used a 2-bit version of the MD5 function that was repeated to convert an input string (message) into an 8-bit hash. This MD5 function was converted to a circuit consisting of 110 NOR or NOT gates. Then, an algorithm was developed to partition the gates across strains while constraining the number of gates and communication channels per strain and allowing the total number of strains to vary. This algorithm partitioned the 110 gates into 66 strains. The subcircuits were computationally designed using Cello[13,14], a new library of phage repressors, inducible systems[75] and four sender–receiver devices[41,75,76]. The corresponding subcircuit DNA was introduced into the genome of each strain, requiring up to 41 genes (23 regulatory) and 31 kb. Collectively, this project required DNA construction on the scale of a small bacterial genome. The subcircuit functions were experimentally verified individually for correct information propagation between pairs of strains and in an example of a three-layer propagation culture between subcircuit strains.

Results

Wiring diagram design for a 2-bit MD5 hashing algorithm

The MD5 hashing algorithm was designed to run on a 32-bit computer, where the input is a 512-bit message and the output is a 128-bit hash. Here, we implemented a version designed to run on a 2-bit computer, where the input is a 32-bit message and the output is an 8-bit hash. The Verilog implementation is shown in Fig. 1a. The input is a binary message that is either padded to 32 bits (if shorter) or broken into 32-bit messages (if longer), after which the message is divided into 2-bit chunks. A different chunk serves as an input to 64 iterations, divided into four 16-iteration rounds, resulting in scrambling of the input message (Supplementary Fig. 1). The MD5 function calculations performed in each iteration are identical in the 32-bit and 2-bit implementations.

The Verilog implementation of the MD5 function was converted to a wiring diagram composed of NOT and NOR gates using the logic synthesis tool Yosys (Fig. 1b and Methods). The logic synthesis initially resulted in a total of 131 NOT and NOR gates connecting the inputs to the outputs, which was later reduced to 110 gates (see below). The wiring diagram has 16 binary inputs and 2 binary outputs; each iteration of the algorithm reuses this function with different input states. Each input and output variable has 2 bits, indicated by subscript 0 and 1. The 2-bit message chunk is represented by the inputs (m_0 m_1). Ultimately, the concatenation of the inputs (a_0 a_1), (b_0 b_1), (c_0 c_1) and (d_0 d_1) becomes the 8-bit hash; they are initialized as (00), (01), (10), (11) and are updated after each iteration (Supplementary Fig. 1). The inputs (s_0 s_1) and (t_0 t_1) are predefined constants that are updated after each iteration using a lookup table of 64 values. Each iteration involves a left shift in the value determined by s and an addition step using the value from t that increases the security of the hash. Inputs (r_0 r_1) represent the round number in binary. The MD5 wiring diagram integrates these eight inputs into a 2-bit output (o_0 o_1) that is used to update b for the next iteration, while a, c and d are updated using the previous values of d, b and c, respectively.

Circuit partitioning

A partitioning algorithm was developed to divide a large circuit into subcircuits carried by communicating cells (Fig. 1c and Methods). It seeks to minimize the required number of cells while conforming to a set of constraints. One constraint is the maximum number of gates in a subcircuit, which is set to avoid overburdening cells. The second constraint is the total number of available orthogonal cell–cell communication signals. At one extreme, if only one gate were allowed per cell, the solution would be to encode each gate in an independent cell, resulting in 131 strains. At the other extreme, if all 131 gates were allowed in a single cell, only one strain would be required to encode the complete circuit.

The partitioning algorithm is shown in Fig. 1c and is described in more depth in Supplementary Fig. 2. It differs from graph partitioning algorithms that fix the number of cells and divide gates among cells to minimize wire crossing. Instead, we implemented a greedy algorithm that seeks to group gates into cells without violating the constraints. Some steps are stochastic, so the process was repeated n times and the partition with the lowest number of cells was selected; in practice, $n = 1{,}000$ was sufficient to identify good partitions for the MD5 circuit. After partitioning, specific communication signals ('colors') must be assigned to wires ('edges') between partitions, a challenge known as the 'edge coloring' problem[94]. To simplify edge coloring, we mapped this task to a simpler 'node coloring problem' that we solved using the Welsh–Powell algorithm[95] (Supplementary Fig. 3). After the generic colors of the edges are computed, each color is randomly assigned to one of the available communication signals, which includes a specific device to send a signal and a specific device to receive the signal in the next cell (see below). The output of the partitioning algorithm was a set of Verilog files describing the logic operation and input/outputs required for the subcircuit in each participating cell. These files can be used by Cello to design the DNA sequences of all subcircuits to be carried in the genomes of the participating strains.

The partitioning algorithm was run on the MD5 circuit while constraining the maximum number of gates per cell to eight and the number of communication signals to four. A solution was found that partitioned the 131 gates across 66 subcircuits (Fig. 1b). Then, we reduced the number of gates per cell by rerunning logic minimization (Yosys) while including the possibility that an OR gate could be used in the last layer. OR gates can be easily implemented at this position using a tandem promoter. These changes reduced the total number of gates in the MD5 circuit to 110 (Methods).

The constraints were set based on our previous experiences with circuit design and the number of orthogonal signals that we could use simultaneously. However, the effect of changing these constraints on the number of cells required could be systematically explored using the partitioning algorithm. Interestingly, the benefit from adding

https://doi.org/10.1038/s41589-024-01730-1

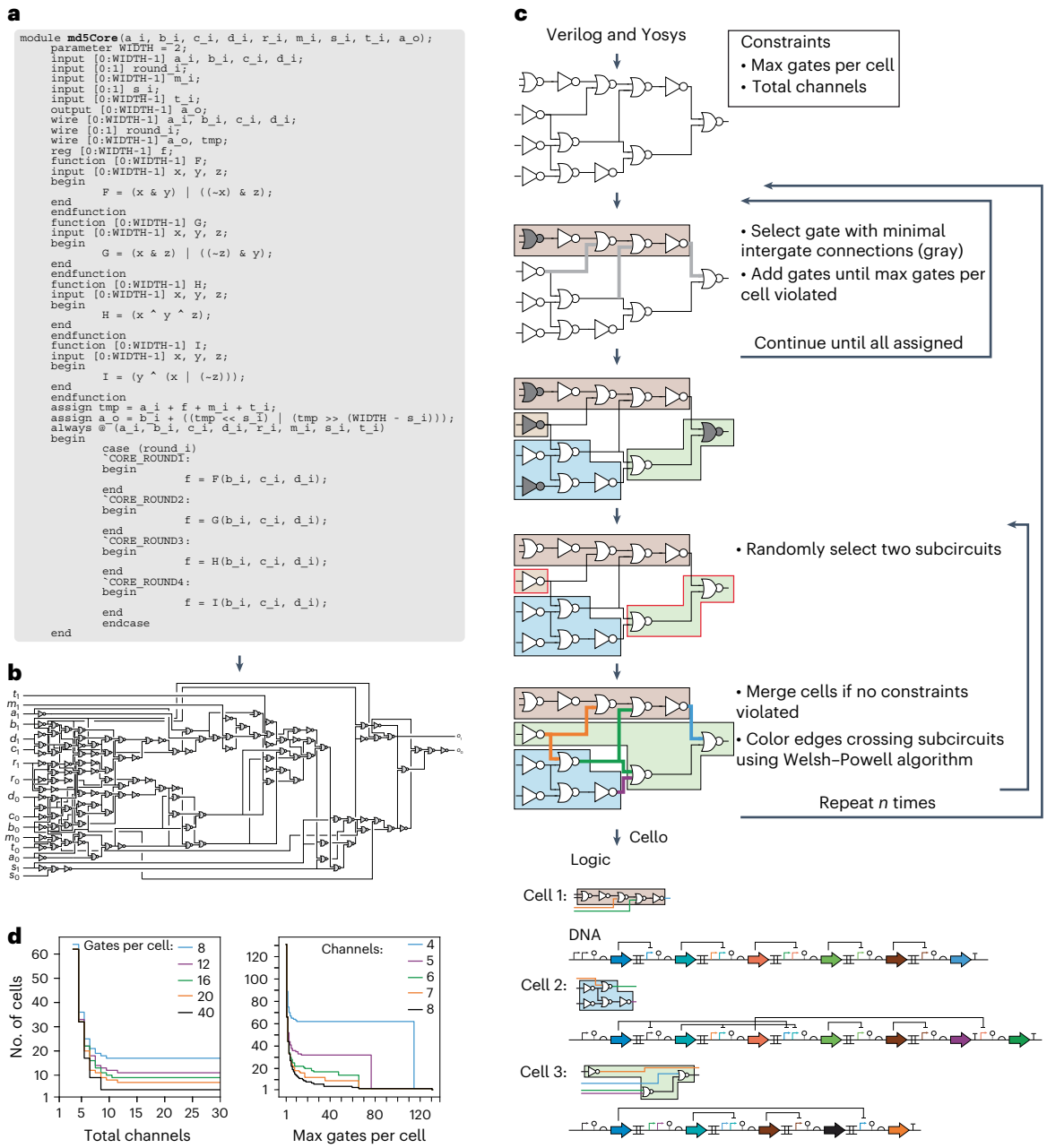


Fig. 1 | Multicellular implementation of the MD5 circuit. a, Verilog behavioral code used to create the circuit performing the MD5 function[116] (Methods). **b**, The Verilog file was used with Yosys to create the initial 131-gate circuit diagram comprising only NOR and NOT gates (Methods). The meanings of the variables describing the 16 inputs and 2 outputs are provided in the main text and Supplementary Fig. 1. **c**, The circuit partitioning function. The constraints used to divide the MD5 circuit were a maximum of five max gates per cell and four channels. Gray gates show the initial gate chosen for each group. After partitions were determined, Cello was used to map the subcircuits to DNA sequences to be inserted into the cell genomes. The algorithm is described in more detail in the Methods and Supplementary Figs. 2 and 3. **d**, Impact of changing the constraints on the number of cells required to encode the MD5 circuit. Effects are shown for increasing the number of total channels with various values for the maximum number of gates per cell (left) and gates per cell with various values for the total number of channels (right).

communication signals stopped at approximately eight, irrespective of the maximum number of gates per cell (Fig. 1d). Similarly, the benefit from increasing the number of gates per cell plateaued at ten, which is currently achievable but can lead to a higher probability of circuit failure (Fig. 1d). In this regime, the benefit from allowing more gates in a cell is incremental.

Gates based on phage repressors

We collated a set of phage repressors with the inclusion criteria of known promoters (P_R from the lysis–lysogeny switch) and unique operator sequences (Fig. 2a)[39]. This set initially contained 20 pairs of repressors and promoters (Supplementary Tables 1 and 2). Gates were constructed using these repressors. Initially, NOT gates were characterized using plasmids (Supplementary Methods). Two input promoters (aTc-inducible P_{Tet} and IPTG-inducible P_{Tac}) were placed in tandem to drive expression of the repressor. The output promoter was fused to the gene encoding yellow fluorescent protein (*yfp*). Repressor expression was controlled using a weak computationally designed ribosome-binding site (RBS)[96] (Supplementary Methods). The repressor gene cassette included insulators to reduce the impact of the

https://doi.org/10.1038/s41589-024-01730-1

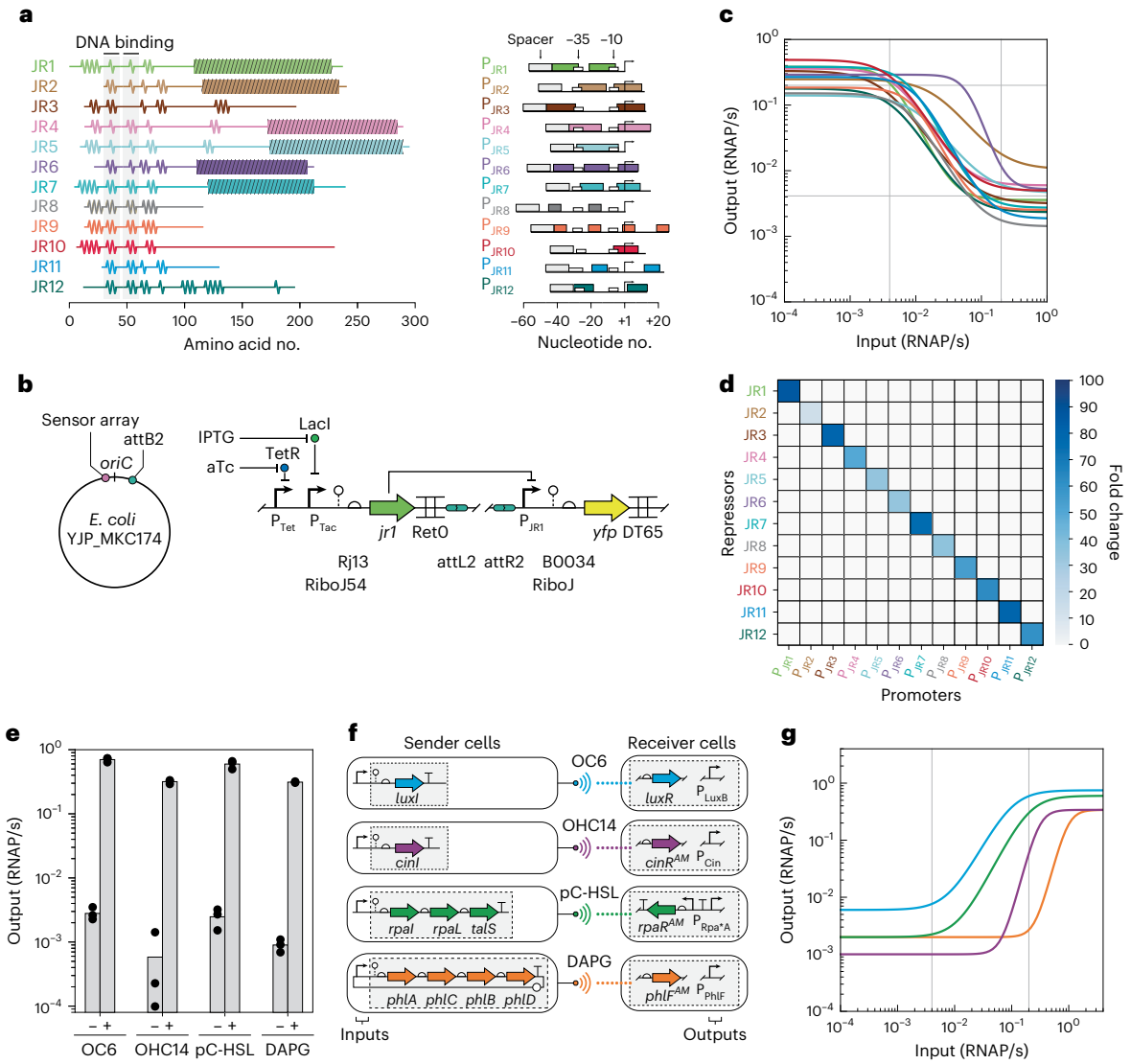


Fig. 2 | Logic gates and cell–cell communication used to build subcircuits. **a**, Library of phage repressors and their cognate promoters. The repressors are aligned by their DNA-binding domains; Pfam peptidase S24 domains are shown as dashed rectangles and α-helical regions were predicted computationally (Methods). The promoters are aligned by their transcription start site and operator sequences are shown as colored boxes. Sequences and references are provided in Supplementary Table 1. **b**, Genomic encoding of a NOT gate. Gate JR1-3 is shown as an example with the order of the repressor and output promoter/reporter reversed for clarity. Genetic parts and sequences are provided in Supplementary Table 11. **c**, NOT gate response functions. The line colors reflect the repressor colors from **a**. The lines were fitted to equation (2) using the parameters in Supplementary Table 3. Schematics of each gate, replicate data and growth impact are provided in the Supplementary Gate Datasheets. The light gray lines are the average outputs of the gates in ON/OFF states and show that the gates are 'impedance matched' and can be connected. **d**, Orthogonality of the repressor–promoter pairs. All combinations of repressors and output promoters were cloned to create 144 strains (Supplementary Fig. 4). Fold change was calculated as the ratio of the fluorescence of induced and uninduced cells, subtracting autofluorescence (*E. coli* JAI_MKC300). **e**, The OFF and ON responses used by Cello to design subcircuits (Supplementary Table 6). The data points represent three replicates performed on different days and the bar heights are means. The strains used were *E. coli* rLux, rCin, rRpa and rPhl (Supplementary Fig. 4). **f**, Genetic diagrams of the sender and receiver cells. Genetic parts are provided in Supplementary Table 11. **g**, The response functions of the sender–receiver devices. The *x* axis represents activity of the sender input promoter (P_{Tac}) and the *y* axis represents activity of the receiver output promoter. The response functions were obtained through fitting to equation (2) using three replicates performed on different days and the parameters in Supplementary Table 7. The light gray lines show the average output of sender–receiver pairs when they are OFF (left) and ON (right).

upstream promoter and to block transcriptional interference[14,33,97,98]. The response function of each gate was measured by inducing cells and measuring fluorescence using flow cytometry (Supplementary Gate Datasheets and Supplementary Methods). Of the original set, four were found to not produce sufficient repression, two caused growth defects and two exhibited crosstalk (Supplementary Table 2). From these experiments, 12 orthogonal gates were identified that yielded strong responses. In several cases, to improve the dynamic range or to change the threshold, either RBS libraries or synthetic output promoters were designed (Supplementary Methods). For simplicity, the final set of repressors and their cognate promoters were renamed JR1–JR12 and P_{JR1}–P_{JR12}, respectively (for example, CI was renamed JR1) (Fig. 2a and Supplementary Table 1).

The gates were then moved to the *E. coli* genome to measure their response functions in this context. The parent strain *E. coli* YJP_MKC174 contains three landing pads, each containing a phage integrase site (attB2, attB7 and attB5) to simplify the insertion of large DNA payloads[33]. Their genomic loci were empirically determined to produce high expression levels and are flanked by strong bi-directional terminators to insulate against incoming or outgoing transcription. This strain

also has the IPTG (LacI) and aTc (TetR) inducible systems in the attB5 landing pad. The gates were integrated into the attB2 landing pad and oriented with the output promoter first to avoid readthrough from strong input promoters (Fig. 2b).

When moving a gate from a plasmid to the genome, the strength of the RBS usually needs to be increased, which was necessary for all but one gate (JR6). To improve expression, we designed small RBS libraries using the RBS Library Calculator[96] and screened them in the attB2 landing pad (Supplementary Methods). From these libraries, we selected variants that yielded similar response functions across the full set of repressors. Uniform response functions simplify their connection by design automation software to build larger circuits.

The response functions of all the gates were then characterized. Cells were grown with different concentrations of inducers in M9 medium (Methods). These experiments produce response functions whose *y* axes represent arbitrary units of fluorescence and *x* axes represent the inducer concentration. Two steps were taken to convert the axes to absolute units of promoter activity (flux of RNA polymerase exiting the promoter per second, or RNAP/s). First, the fluorescence from the P_{Tet}–P_{Tac} tandem promoter was measured as a function of inducer concentration (*E. coli* JAI_MKC148), which could be used to convert the *x* axis of the response functions to fluorescence. Second, both axes were converted to RNAP/s using the reference promoter BBa_J23101 (*E. coli* YJP_MKC254) (Supplementary Fig. 4)[33,72,99]. These data were then fitted to

$$y = y_{\min} + (y_{\max} - y_{\min})\frac{K^n}{x^n + K^n} \quad (1)$$

where *y* is the activity of the output promoter, *x* is the activity of the input promoter, *K* is the threshold and *n* is the cooperativity. The response functions are shown in Fig. 2c with the parameters given in Supplementary Tables 3 and 4. The replicate information is given in the Supplementary Gate Datasheets. The uniformity of these response functions makes them easier to connect using Cello than previous TetR-family-based gates[49]. When the output of a gate switches from ON to OFF (horizontal lines in Fig. 2c), this crosses the range required to turn the next gate from OFF to ON (vertical lines in Fig. 2c). The expression of phage repressors from the genome had little impact on cell growth (Supplementary Gate Datasheets). To test for orthogonality, 144 strains were constructed by crossing the 12 repressors with the 12 promoters in their genomes. No crosstalk was observed (Fig. 2d and Supplementary Figs. 4 and 5).

The gates and their response functions were used to build a user constraint file (UCF) for Cello that can be used for automated circuit design (Eco2C1G5T1)[13,14]. Genome-encoded NOR gates use two copies of the same repressor gene, each of which is independently connected to an input promoter[33]. The UCF includes constraints that enforce separation of the repressor genes for one gate across the attB2 and attB7 landing pads. This prevents problems associated with RNAP roadblocking that can occur with tandem promoters[14] and avoids homologous recombination. For repressors JR1, JR7 and JR9, we included gates based on different RBSs that shifted the response function thresholds, providing more flexibility in finding solutions for circuit designs (Supplementary Gate Datasheets).

Characterization of cell–cell communication channels

Each channel was based on a sender device that produces the chemical signal and a receiver device that responds to it. We selected four channels known to not cross-react with each other's signals: 3-oxohexanoyl-homoserine lactone (OC6), 3-hydroxytetradecanoyl-homoserine lactone (OHC14), *para*-coumaroyl-homoserine lactone (pC-HSL) and 2,4-diacetylphloroglucinol (DAPG)[41,75,76]. First, we characterized the receiver devices that respond to each of these small molecules (based on LuxR, $CinI^{AM}$, $RpaR^{AM}$ and $PhlF^{AM}$)[75,76]. These regulators were expressed from a contiguous 'sensor array' that we inserted into the landing pad strain to generate *E. coli* JAI_MKC300 (Supplementary Fig. 6 and Supplementary Methods). When needed, $rpaR^{4M}$ was encoded with the DNA containing the circuit.

We measured the response functions of the four receiver devices. The output promoter P_{LuxB}, P_{Cin}, P_{Rpa*A} or P_{PhlF} was fused to *yfp* and inserted into the attB2 landing pad to create the *E. coli* rLux, rCin, rRpa and rPhl 'receiver cells', respectively (Supplementary Fig. 4). Each strain was grown at different concentrations of the exogenously added communication signal (Supplementary Methods). The fluorescence output was measured by cytometry and was converted to units of RNAP/s. These data were fitted to the response function

$$y = y_{\min} + (y_{\max} - y_{\min})\frac{c^n}{c^n + K^n} \quad (2)$$

where *c* is the concentration of the signaling molecule. The full response functions of the four receivers are shown in Supplementary Fig. 7 with parameters provided in Supplementary Table 5. Because Cello designs digital logic circuits, it requires only the activities of the receiver output promoters in the OFF and ON states (Fig. 2e).

The sender devices were then constructed. The input to a sender device is defined as a promoter and the output is the expression of the enzyme(s) that produce the communication signal[72] (Supplementary Subcircuit Datasheets). The biosynthetic enzymes comprising the sender devices were as follows: LuxI (OC6), CinI (OHC14), RpaIL/TalS[41] (pC-HSL) and PhlACBD[41] (DAPG). To characterize the devices, the IPTG-inducible P_{Tac} promoter was selected as the input. The first three were inserted into the attB2 landing pad to create the following 'sender cells': *E. coli* sLux, sCin and sRpa (Fig. 2f). The DAPG biosynthetic pathway was carried on a plasmid to obtain higher production levels (plJAI_617), which was carried by the *E. coli* sPhl sender cell.

The circuit design algorithm was modified to incorporate transmittal of the signal between cells. Previously, Cello predicted the activity of a circuit output promoter only if it was fused to a fluorescent reporter gene. Instead, we sought to predict how its activity would propagate and induce the receiver promoter in the next cell in the multicellular circuit. Performing this calculation requires a response function whose *x* axis represents the activity of the output promoter of the upstream cell and *y* axis represents the input promoter activity of the receiver in the downstream cell.

We empirically measured these functions using sender and receiver strains grown in liquid culture (Fig. 2g). Sender strains were grown in M9 medium as previously described with different concentrations of IPTG (Methods). The supernatants were collected, filtered and used to induce the receiver cells. The receiver cells were grown separately for 16 h and then cultured for 3 h in the sender's supernatant (Methods). The data from these experiments were used to fit response functions that capture transmission of the signal from the sender to receiver cells (equation (2)), where *c* was replaced by the promoter activity *x* of the sender device). The response functions are shown in Fig. 2g, the parameters are provided in Supplementary Table 6 and replicate information is provided in Supplementary Fig. 8. Each communication channel had a similar 50-fold dynamic range with varying activities in the OFF state. The average outputs of the NOT and NOR gates spanned the ranges required to turn on the sender device to induce a response in the receiver cells (horizontal lines in Fig. 2g). Therefore, the subcircuit output(s) could be reliably connected to the sender devices to transmit the signal to the next layer of the multicellular circuit.

Subcircuit design

Cello was modified to design the needed subcircuits (Supplementary Methods). First, the logic minimization and gate assignment algorithms had to be changed to design circuits with multiple outputs. Second,

the software was extended to automate the connection of outputs that are not fluorescent reporters. This change enabled the cell–cell communication devices to be connected and logic gates to be selected based on their response functions (Fig. 2g). Cello 2.1 software was used to design the DNA sequences for the subcircuits using the phage repressor library (Supplementary Methods).

The output of the partitioning algorithm specified 66 subcircuits, but some were identical, so only 41 strains needed to be constructed (for example, *E. coli* sc21 is used nine times) (Supplementary Fig. 9). The 41 Verilog files produced by the partitioning algorithm were provided to Cello along with the information defining the sensors and actuators (DNA sequences and responses) (Supplementary Table 7). There are 16 inputs to the MD5 circuit, the states of which were reported to the subcircuits using the IPTG, Ara, Cuma, aTc and Van inducible systems present in the *E. coli* Marionette sensor array[75]. The specific assignment of sensors to each MD5 input was based on their ability to connect to the subcircuit. Note that the same input to the MD5 circuit can be reported to subcircuits with different inducible systems; for example, input d_1 is reported by Ara in subcircuit sc2 by aTc in subcircuit sc3. Similarly, the same input to the MD5 circuit can be represented by different inducers for different subcircuits. The outputs of the subcircuits were specified as sender devices, which were randomly assigned to each 'color' from the partitioning algorithm. They consisted of the biosynthetic pathways for the communication signals as well as genes encoding YFP, red fluorescent protein (RFP), or blue fluorescent protein (BFP) so that their induction could be visualized. When the output of the subcircuit corresponded to an output of the MD5 circuit (o_0 and o_1), only the fluorescent reporters were the outputs (*E. coli* sc39 and sc41).

Cello 2.1 provided the subcircuit DNA sequences to be inserted into the landing pads of the genome, as well as their predicted responses. The median number of regulator genes required in a cell was eight and the median DNA size was 25 kb (Fig. 3a). The largest subcircuit was sc5, which has two inputs, eight gates and three communication outputs, requiring 31 kb of DNA to encode the subcircuit alone (the total amount of recombinant DNA in *E. coli* sc5 is 43 kb) (Fig. 3b,c). Following the Cello specifications exactly, the DNA sequences for the 41 subcircuits were constructed and inserted into the attB2 and attB7 landing pads of *E. coli* JAI_MKC300 to create strains *E. coli* sc1 to sc41 (Supplementary Subcircuit Datasheets and Supplementary Methods). The sender devices were encoded in the genome in the attB5 landing pad, except for the DAPG sender, which was carried on a p15a plasmid. For subcircuits sc5, sc6, sc15 and sc27, we observed toxicity due to the DAPG sender device, which was resolved by replacing the origin with that from a lower-copy pSC101 plasmid. Across the 41 strains, the total recombinant DNA required for introduction was 1.1 Mb, including the DNA inserted into the landing pads, the sensor array and the plasmid.

First, we used fluorescent reporters to characterize the responses of the subcircuits carried by the 41 strains. Cells were induced with different combinations of their inputs, including exogenously added communication signals if needed (DAPG, OC6, OHC14 or pC-HSL) (Methods). Cells were induced for 16 h in M9 medium at 37 °C using multicolor flow cytometry (Methods). The activities of the output promoters were characterized in RNAP/s using reference strains containing the BBa_J23101 promoter fused to *yfp*, *rfp* or *bfp* (*E. coli* YJP_MKC274, JAI_MKC399, JAI_MKC400) (Supplementary Fig. 10). For the largest subcircuit (sc5), the responses of the output promoters to all input combinations are shown in Fig. 3d. The data for all cellular circuits are compiled into Fig. 3e for all subcircuits and combinations of inputs (complete data are shown in the Supplementary Subcircuit Datasheets). Cello accurately predicted whether the circuit would be ON or OFF across all combinations of inputs, but the fine-tuned expression levels predicted in both states were more variable.

Only 3 of the 41 subcircuits failed in the initial attempt and had to be redesigned. Subcircuits *E. coli* sc39 and sc40 failed for the same reason and were fixed with the same modification. They failed in two states (−/−/+ and +/+/−), which we corrected by redefining an input to be the Cuma sensor (in place of the aTc sensor) and rerunning Cello to obtain new gate assignments. These changes improved the circuit performance; however, while the +/+/− state was technically OFF, the signal was still too high for the sender device to work properly. We hypothesized that this problem was caused by the weak RpoC terminator, so we moved the JR3 gate to the 3′ end, which corrected the problem. This was added as a constraint to the UCF for future designs. The third failed subcircuit was *E. coli* sc21, which was OFF in the +/− state when it should have been ON. Crosstalk has previously been observed between RpaR and P_{LuxB} and we suspected that this was also true for $RpaR^{AM}$. Therefore, we replaced P_{LuxB} with P_{Lux*TA}, which corrected the problem[76]. Despite these minor failures, the extent to which this large design project could be automated and worked in the first pass is remarkable.

Communication between MD5 subcircuits

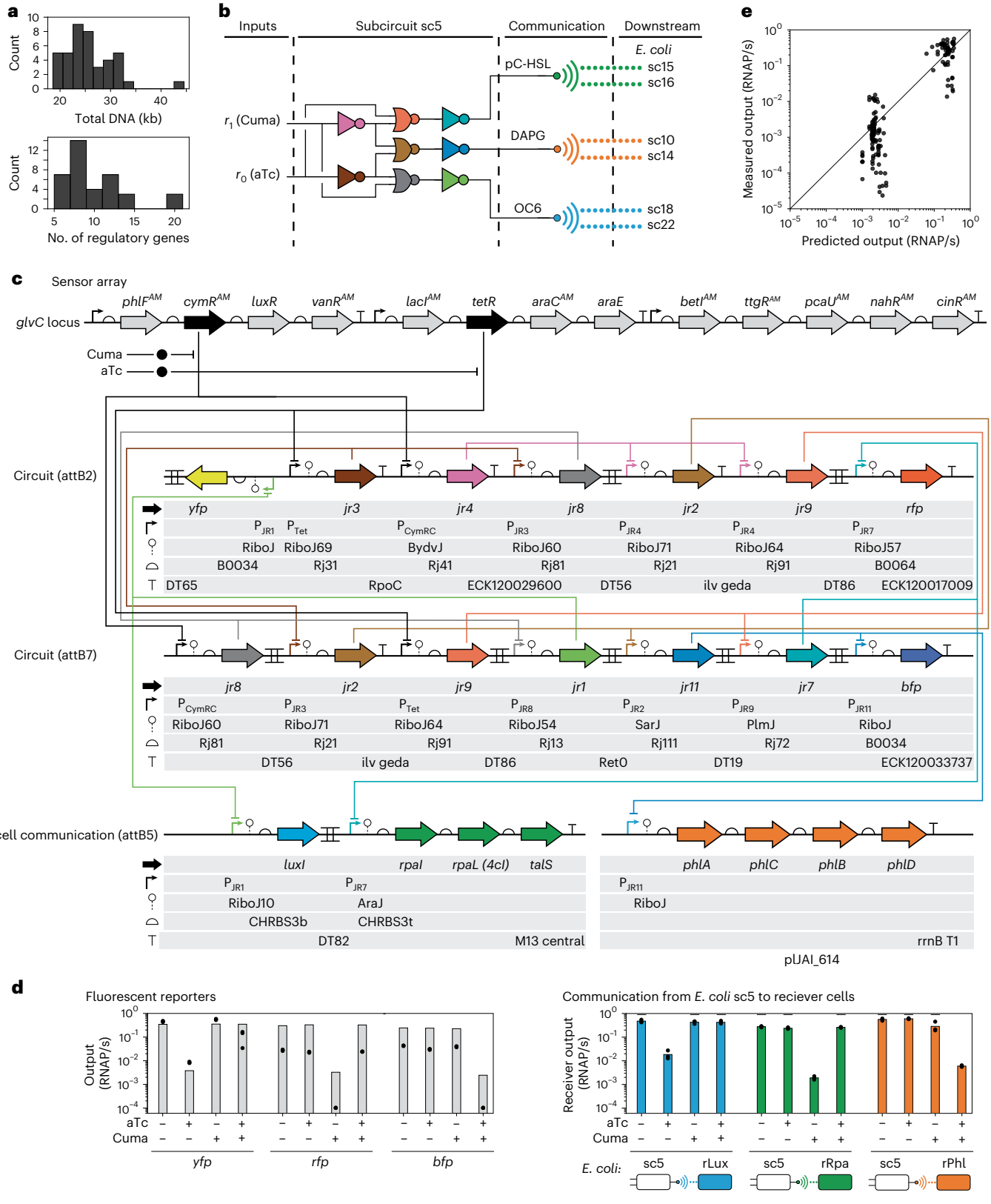
We characterized the ability of each cell carrying a subcircuit to communicate its state to the next layer of cells in the MD5 circuit. These experiments were performed independently by growing each subcircuit-containing cell with different combinations of inducers and measuring propagation of the communication signals to the receiver cells (*E. coli* rLux, rCin, rRpa and rPhl). First, cells containing a subcircuit were inoculated into M9 medium containing inducers and the supernatant was collected (Methods). The receiver cells were grown separately, diluted, added directly to the supernatant and cultured for 3 h. All cells communicated to the next layer as expected, producing the correct response. These data are shown for the largest subcircuit (sc5) in Fig. 3d. In all four states, the subcircuit could transmit its state to the next layer. Although *E. coli* sc5 carries up to 41 recombinant genes (23 regulatory genes), this had little impact on the growth rate (Supplementary Fig. 12). This is in striking contrast to previous circuit designs, where we observed that smaller circuits decreased the growth rate by up to 30% over 8 h, causing evolutionary breakage within a day[18].

The partition of the MD5 circuit is shown in Fig. 4a, including the small molecules used to communicate the subcircuit states between

Fig. 3 | Division of the 2-bit MD5 circuit into subcircuits. a, Distribution of subcircuit designs after running the partitioning algorithm and Cello. The total DNA encompasses the amount of recombinant DNA that must be added to the genome of each cell, as defined in the Methods. The number of regulatory genes includes sensors and gates. The distributions are for 41 cells; datasheets are provided in Supplementary Subcircuit Datasheets. **b**, The largest subcircuit carried by one cell. Gates are colored by the repressor assigned. **c**, Genetic design of the largest subcircuit carried by *E. coli* sc5. The complete sensor array is shown, but only two sensors serve as inputs to this subcircuit. Most of these constructs are carried in the genome, but the DAPG sender device is carried on the pSC101 plasmid (Supplementary Fig. 4). Genetic part sequences are provided in Supplementary Table 11. **d**, Characterization of the largest subcircuit carried by *E. coli* sc5 (Methods). Left, activity of the sc5 output promoters for all combinations of inputs (200 nM aTc and 100 μM Cuma); bars represent the computational values predicted by Cello and points represent three biological replicates performed on different days. Right, communication of the subcircuit state of *E. coli* sc5 to the next layers of cells; bars represent the mean activities of the output promoters of the receiver devices and points represent three replicates performed on different days. The horizontal marks at the top of the graph indicate the states where the receiver device should be ON. Cytometry distributions are provided in Supplementary Fig. 11. **e**, Measured versus Cello-predicted activities of the output promoters for all 41 strains containing subcircuits (*E. coli* sc1 to sc41). The points represent the outputs for all combinations of inducers in all subcircuits. Details and replicate information are provided in the Supplementary Subcircuit Datasheets.

https://doi.org/10.1038/s41589-024-01730-1

cells. The empirical responses of all 41 subcircuits are shown in Fig. 4b. Each graph shows the response of the subcircuit to exogenously added small molecules, which is representative of either the inputs to the MD5 circuit or the communication signals. The outputs were the transmittal of the subcircuit state to the next layer of the multicellular circuit and were read out using *E. coli* rLux, rCin, rRpa or rPhl. All subcircuits performed as predicted, with those that should be in the ON state marked with an overbar in Fig. 4b. None of the cells containing the subcircuits exhibited a statistically significant growth defect, in contrast to our previous experiences[10,18]. The robustness of the MD5 subcircuits in this work speaks to the careful design and screening of gates to avoid toxicity and the impact of incorporating the circuits into the genome.



https://doi.org/10.1038/s41589-024-01730-1

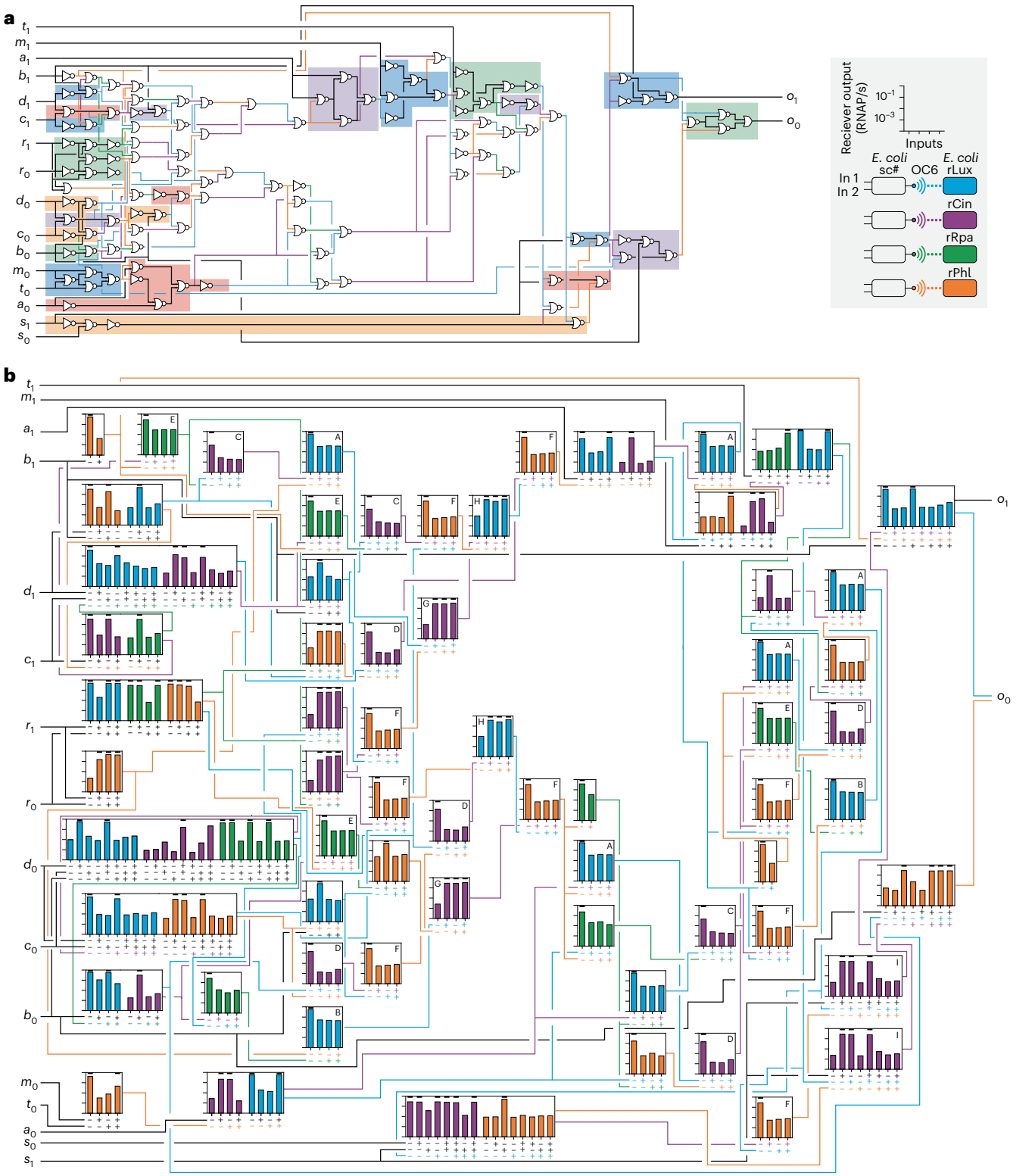


Fig. 4 | Multicellular computation of the 2-bit MD5 circuit. a, Circuit diagram for the MD5 function (110 gates). The subcircuit partitions are shown by the background colors and colored lines indicate the cell–cell communication signal (blue, OC6; purple, OHC14; green, pC-HSL; orange, DAPG). Supplementary Fig. 10 provides the subcircuit number for each partition. **b**, All MD5 subcircuits communicating to the next layer of cells (legend). The graphs are organized spatially to mimic 4a. The bars represent the means of three experiments performed on different days and the horizontal marks indicate the states where the output should be ON. The bar colors correspond to the cells used to measure the activity of the receiver devices (*E. coli* rLux, rCin, rRpa or rPhl). Complete data, including replicates, are provided in the Supplementary Subcircuit Datasheets along with the concentrations of the inducers used. Representative cytometry distributions are provided in Supplementary Fig. 11. The lines between graphs mark the cell–cell communication channels. The chemicals used for communication signals were exogenously added: 10 µM OC6, 10 µM OHC14, 10 µM pC-HSL, 25 µM DAPG. The letters indicate cells that were repeated at different positions in the MD5 circuit because they are identical in inputs, outputs and logic function (A, *E. coli* sc19; B, *E. coli* sc20; C, *E. coli* sc11; D, *E. coli* sc18; E, *E. coli* sc10; F, *E. coli* sc22; G, *E. coli* sc23; H, *E. coli* sc24; I, *E. coli* sc38).

The complete set of strains containing subcircuits cannot be connected to each other to build the full MD5 circuit. This limitation was due to the number of orthogonal cell–cell communication signals that were available. To demonstrate propagation of the signal through the three layers of the circuit in a co-culture, we selected two subcircuits (*E. coli* sc2 and sc4) and receiver strains for the outputs (*E. coli* rCin and rRpa) (Supplementary Fig. 13). Cultures of *E. coli* sc2 and sc4 were grown and then mixed in media with different concentrations of the inducers. After growth in the co-culture, the supernatants were used to induce receiver cells. Using this protocol, *E. coli* sc2 correctly transmitted its signals to *E. coli* sc4 followed by *E. coli* rRpa (and *E. coli* rCin) across all input and output states. Note that, even if more cell–cell communication signals were available, it would remain difficult to connect the subcircuits because the cells would need to be synchronized and when a signal skips layers, this can lead to faults (transient incorrect outputs).

The full MD5 hash function was calculated using the empirical responses measured for all strains containing subcircuits (Supplementary Fig. 14). Each strain was treated individually as performing the calculation and transmitting its signal to the next layer of the MD5 circuit. The data for the induction of a receiver cell by one cell containing a subcircuit were used for these calculations. A simulation was performed in which the signal was propagated through the MD5 circuit for 64 iterations to complete the hash. These simulations performed the correct hashing of 'MIT', indicating that the fuzzy logic implemented by strains carrying individual subcircuits is sufficient for performing the binary 2-bit hash.

Discussion

This work demonstrates how a circuit function that is too large and complex to be performed by a single cell can be divided across a set of communicating cells. This feat required the development of new design automation algorithms and genomically encoded gates. These tools allowed us to increase both the scale of individual circuits—to our knowledge, subcircuit sc5 is the largest constructed to date and the number of cells that can be part of a larger design project. This MD5 circuit design is a marked increase in complexity over earlier work, in 2020 to encode an LCD calculator display chip (Texas Instruments SN74LS49) across seven strains of *E. coli* (0.1 Mb)[18], in 2011 to encode an XOR gate across four strains (0.03 Mb)[39] and in 2009 to encode an edge detector in one strain (0.018 Mb)[9]. Note that the primary goal of genetic circuit design is to gain control over the capabilities of biology—evidenced in the natural world—not to beat electronics at computing tasks. However, there may be a point at which computing by living cells can outperform electronic circuits for some classes of problems[1,8].

Building more powerful biological computers requires larger circuits in individual cells. Information theory limits the number of DNA-binding regulators to hundreds and, even with burden-mitigating strategies, heterologous protein expression eventually overburdens the cell[23,24,66]. However, it is possible to be more computationally efficient with this capacity than with our two-input NOR gates. The computational complexity of a single cell could be improved by using multi-input logic, gate compression and analog circuits to make orders-of-magnitude improvements in computational complexity[6,52,100–104].

The more difficult remaining challenge is to connect many cells to perform distributed computing collectively. The human brain has 10^{15} connections between 10^{11} cells that passage information at the 1-ms timescale, representing a frustratingly high water mark for what is possible via biology[105]. The programable passage of information quickly and specifically between cells remains limiting. Relying on chemical signals to perform this function in liquid cultures or between colonies on a plate is slow (hours), requires a large cell density to make sufficient titers and is limited by the number of orthogonal channels, and it is difficult to remove signal once it has been produced. Various proposals have been made to use microfluidic devices or three-dimensional (3D) printed cells in hydrogels to arrange communicating cells[11,39,41,42,76,79,80,88,106–109]. However, these approaches require encoding the circuit function in the physical device itself (for example, molding liquid channels between cells) and are constrained in terms of the potential connectivity between cells, particularly if a circuit requires a wire that bypasses gate layers. Brains overcome this limitation through neurons extending their axons and dendrites to make contact with many distant cells. Fully realizing the computational potential of a cell population will require the ability to grow or print 'brain-like' structures that can transmit information rapidly through physical contacts[110].

Cryptographic problems may be suitable for biological computers, particularly problems requiring repetitive independent calculations. Cryptocurrencies use an estimated 1% of global electricity[111]. Here, we have begun to show how cells could be programmed to perform a simple MD5 hash function, and scaling the approach to the SHA256 algorithm underlying Bitcoin is theoretically possible. However, circuit design based on digital layered gates is not ideal for cellular regulatory networks. For one, our circuits are slow. Based on a 6-h cell-to-cell transmission time and the longest path through the MD5 circuit, we estimate that it would take 200 days to complete the 'MIT' hash (Supplementary Fig. 14) if done one step at a time. If performed in culture, synchronization is also a problem where there is no intrinsic 'clock' in cellular regulatory networks. Rather, biological computation thrives on amorphous and asynchronous analog computing, for which few design automation tools are currently available[112–115]. There are dozens of hash algorithms, in addition to MD5 and SH256, associated with various cryptocurrencies, all of which have been designed for use with electronic CPUs. One can imagine designing cryptography algorithms specific for cell-based computers that use their highly parallelized, asynchronous and amorphous structure[1].

Online content

Any methods, additional references, Nature Portfolio reporting summaries, source data, extended data, supplementary information, acknowledgements, peer review information; details of author contributions and competing interests; and statements of data and code availability are available at https://doi.org/10.1038/s41589-024-01730-1.

References

1. Abelson, H. et al. Amorphous computing. *Commun. ACM* **43**, 74–82 (2000).
2. Davidson, E. H. *Genomic Regulatory Systems* (Academic Press, 2001).
3. Turing, A. M. The chemical basis of morphogenesis. *Philos. Trans. R. Soc. Lond., Ser. B* **237**, 37–72 (1952).
4. Wolfram, S. *A New Kind of Science* (Wolfram Media, 2002).
5. Barcena Menendez, D., Senthivel, V. R. & Isalan, M. Sender–receiver systems and applying information theory for quantitative synthetic biology. *Curr. Opin. Biotechnol.* **31**, 101–107 (2015).
6. Karkaria, B. D., Treloar, N. J., Barnes, C. P. & Fedorec, A. J. H. From microbial communities to distributed computing systems. *Front. Bioeng. Biotechnol.* **8**, 834 (2020).
7. Zhang, Y. et al. A system hierarchy for brain-inspired computing. *Nature* **586**, 378–384 (2020).
8. Grozinger, L. et al. Pathways to cellular supremacy in biocomputing. *Nat. Commun.* **10**, 5250 (2019).
9. Tabor, J. J. et al. A synthetic genetic edge detection program. *Cell* **137**, 1272–1281 (2009).
10. Brophy, J. A. & Voigt, C. A. Principles of genetic circuit design. *Nat. Methods* **11**, 508–520 (2014).
11. Hasty, J., McMillen, D. & Collins, J. J. Engineered gene circuits. *Nature* **420**, 224–230 (2002).
12. McAdams, H. H. & Arkin, A. Gene regulation: towards a circuit engineering discipline. *Curr. Biol.* **10**, R318–R320 (2000).

13. Jones, T. S., Oliveira, S. M. D., Myers, C. J., Voigt, C. A. & Densmore, D. Genetic circuit design automation with Cello 2.0. *Nat. Protoc.* **17**, 1097–1113 (2022).
14. Nielsen, A. A. et al. Genetic circuit design automation. *Science* **352**, aac7341 (2016).
15. Lucks, J. B., Qi, L., Whitaker, W. R. & Arkin, A. P. Toward scalable parts families for predictable design of biological circuits. *Curr. Opin. Microbiol.* **11**, 567–573 (2008).
16. Nielsen, A. A., Segall-Shapiro, T. H. & Voigt, C. A. Advances in genetic circuit design: novel biochemistries, deep part mining, and precision gene expression. *Curr. Opin. Chem. Biol.* **17**, 878–892 (2013).
17. Fernandez-Rodriguez, J., Yang, L., Gorochowski, T. E., Gordon, D. B. & Voigt, C. A. Memory and combinatorial logic based on DNA inversions: dynamics and evolutionary stability. *ACS Synth. Biol.* **4**, 1361–1372 (2015).
18. Shin, J., Zhang, S., Der, B. S., Nielsen, A. A. & Voigt, C. A. Programming *Escherichia coli* to function as a digital display. *Mol. Syst. Biol.* **16**, e9401 (2020).
19. Bragdon, M. D. J. et al. Cooperative assembly confers regulatory specificity and long-term genetic circuit stability. *Cell* **186**, 3810–3825 (2023).
20. Sleight, S. C., Bartley, B. A., Lieviant, J. A. & Sauro, H. M. Designing and engineering evolutionary robust genetic circuits. *J. Biol. Eng.* **4**, 12 (2010).
21. Ceroni, F., Algar, R., Stan, G. B. & Ellis, T. Quantifying cellular capacity identifies gene expression designs with reduced burden. *Nat. Methods* **12**, 415–418 (2015).
22. Huang, H. H. et al. dCas9 regulator to neutralize competition in CRISPRi circuits. *Nat. Commun.* **12**, 1692 (2021).
23. McBride, C. D., Grunberg, T. W. & Del Vecchio, D. Design of genetic circuits that are robust to resource competition. *Curr. Opin. Syst. Biol.* https://doi.org/10.1016/j.coisb.2021.100357 (2021).
24. Scott, M., Gunderson, C. W., Mateescu, E. M., Zhang, Z. & Hwa, T. Interdependence of cell growth and gene expression: origins and consequences. *Science* **330**, 1099–1102 (2010).
25. Tan, C., Marguet, P. & You, L. Emergent bistability by a growth-modulating positive feedback circuit. *Nat. Chem. Biol.* **5**, 842–848 (2009).
26. Şimşek, E., Yao, Y., Lee, D. & You, L. Toward predictive engineering of gene circuits. *Trends Biotechnol.* **41**, 760–768 (2023).
27. Zhang, R. et al. Topology-dependent interference of synthetic gene circuit function by growth feedback. *Nat. Chem. Biol.* **16**, 695–701 (2020).
28. Zhang, R. et al. Winner-takes-all resource competition redirects cascading cell fate transitions. *Nat. Commun.* https://doi.org/10.1038/s41467-021-21125-3 (2021).
29. Barajas, C., Huang, H. H., Gibson, J., Sandoval, L. & Del Vecchio, D. Feedforward growth rate control mitigates gene activation burden. *Nat. Commun.* **13**, 7054 (2022).
30. Chen, Y. et al. Genetic circuit design automation for yeast. *Nat. Microbiol.* **5**, 1349–1360 (2020).
31. Guan, Y. et al. Mitigating host burden of genetic circuits by engineering autonegatively regulated parts and improving functional prediction. *ACS Synth. Biol.* **11**, 2361–2371 (2022).
32. Liu, Q., Schumacher, J., Wan, X., Lou, C. & Wang, B. Orthogonality and burdens of heterologous AND gate gene circuits in *E. coli*. *ACS Synth. Biol.* **7**, 553–564 (2018).
33. Park, Y., Espah Borujeni, A., Gorochowski, T. E., Shin, J. & Voigt, C. A. Precision design of stable genetic circuits carried in highly-insulated *E. coli* genomic landing pads. *Mol. Syst. Biol.* **16**, e9584 (2020).
34. Barajas, C. & Del Vecchio, D. Synthetic biology by controller design. *Curr. Opin. Biotechnol.* **78**, 102837 (2022).
35. Grob, A., Di Blasi, R. & Ceroni, F. Experimental tools to reduce the burden of bacterial synthetic biology. *Curr. Opin. Syst. Biol.* **28**, 100393 (2021).
36. Son, H. I., Weiss, A. & You, L. Design patterns for engineering genetic stability. *Curr. Opin. Biomed. Eng.* **19**, 100297 (2021).
37. Ceroni, F. et al. Burden-driven feedback control of gene expression. *Nat. Methods* **15**, 387–393 (2018).
38. Lou, C. et al. Synthesizing a novel genetic sequential logic circuit: a push-on push-off switch. *Mol. Syst. Biol.* **6**, 350 (2010).
39. Tamsir, A., Tabor, J. J. & Voigt, C. A. Robust multicellular computing using genetically encoded NOR gates and chemical 'wires'. *Nature* **469**, 212–215 (2011).
40. Yokobayashi, Y., Weiss, R. & Arnold, F. H. Directed evolution of a genetic circuit. *Proc. Natl Acad. Sci. USA* **99**, 16587–16591 (2002).
41. Du, P. et al. De novo design of an intercellular signaling toolbox for multi-channel cell–cell communication and biological computation. *Nat. Commun.* **11**, 4226 (2020).
42. Macia, J. et al. Implementation of complex biological logic circuits using spatially distributed multicellular consortia. *PLoS Comput. Biol.* **12**, e1004685 (2016).
43. Sexton, J. T. & Tabor, J. J. Multiplexing cell–cell communication. *Mol. Syst. Biol.* **16**, e9618 (2020).
44. Garg, A., Lohmueller, J. J., Silver, P. A. & Armel, T. Z. Engineering synthetic TAL effectors with orthogonal target sites. *Nucleic Acids Res.* **40**, 7584–7595 (2012).
45. Green, A. A. et al. Complex cellular logic computation using ribocomputing devices. *Nature* **548**, 117–121 (2017).
46. Hsia, J., Holtz, W. J., Maharbiz, M. M., Arcak, M. & Keasling, J. D. Modular synthetic inverters from zinc finger proteins and small RNAs. *PLoS ONE* **11**, e0149483 (2016).
47. Jusiak, B., Cleto, S., Perez-Pinera, P. & Lu, T. K. Engineering synthetic gene circuits in living cells with CRISPR technology. *Trends Biotechnol.* **34**, 535–547 (2016).
48. Nielsen, A. A. & Voigt, C. A. Multi-input CRISPR/Cas genetic circuits that interface host regulatory networks. *Mol. Syst. Biol.* **10**, 763 (2014).
49. Stanton, B. C. et al. Genomic mining of prokaryotic repressors for orthogonal logic gates. *Nat. Chem. Biol.* **10**, 99–105 (2014).
50. Taketani, M. et al. Genetic circuit design automation for the gut resident species *Bacteroides thetaiotaomicron*. *Nat. Biotechnol.* **38**, 962–969 (2020).
51. Didovyk, A., Borek, B., Hasty, J. & Tsimring, L. Orthogonal modular gene repression in *Escherichia coli* using engineered CRISPR/Cas9. *ACS Synth. Biol.* **5**, 81–88 (2016).
52. Rondon, R. E., Groseclose, T. M., Short, A. E. & Wilson, C. J. Transcriptional programming using engineered systems of transcription factors and genetic architectures. *Nat. Commun.* **10**, 4784 (2019).
53. Bonnet, J., Yin, P., Ortiz, M. E., Subsoontorn, P. & Endy, D. Amplifying genetic logic gates. *Science* **340**, 599–603 (2013).
54. Zhang, S. & Voigt, C. A. Engineered dCas9 with reduced toxicity in bacteria: implications for genetic circuit design. *Nucleic Acids Res.* **46**, 11115–11125 (2018).
55. Basu, S., Mehreja, R., Thiberge, S., Chen, M. T. & Weiss, R. Spatiotemporal control of gene expression with pulse-generating networks. *Proc. Natl Acad. Sci. USA* **101**, 6355–6360 (2004).
56. Kobayashi, H. et al. Programmable cells: interfacing natural and engineered gene networks. *Proc. Natl Acad. Sci. USA* **101**, 8414–8419 (2004).
57. Ackers, G. K., Johnson, A. D. & Shea, M. A. Quantitative model for gene regulation by lambda phage repressor. *Proc. Natl Acad. Sci. USA* **79**, 1129–1133 (1982).
58. Basu, S., Gerchman, Y., Collins, C. H., Arnold, F. H. & Weiss, R. A synthetic multicellular system for programmed pattern formation. *Nature* **434**, 1130–1134 (2005).

59. Kotula, J. W. et al. Programmable bacteria detect and record an environmental signal in the mammalian gut. *Proc. Natl Acad. Sci. USA* **111**, 4838–4843 (2014).
60. Elowitz, M. B. & Leibler, S. A synthetic oscillatory network of transcriptional regulators. *Nature* **403**, 335–338 (2000).
61. Hooshangi, S., Thiberge, S. & Weiss, R. Ultrasensitivity and noise propagation in a synthetic transcriptional cascade. *Proc. Natl Acad. Sci. USA* **102**, 3581–3586 (2005).
62. Xiong, L. L., Garrett, M. A., Buss, M. T., Kornfield, J. A. & Shapiro, M. G. Tunable temperature-sensitive transcriptional activation based on lambda repressor. *ACS Synth. Biol.* **11**, 2518–2522 (2022).
63. Karig, D. et al. Stochastic turing patterns in a synthetic bacterial population. *Proc. Natl Acad. Sci. USA* **115**, 6572–6577 (2018).
64. Liu, C. et al. Sequential establishment of stripe patterns in an expanding cell population. *Science* **334**, 238–241 (2011).
65. Ptashne, M. *A Genetic Switch: Phage Lambda Revisited*. 3rd ed. (Cold Spring Harbor Laboratory Press, 2004).
66. Itzkovitz, S., Tlusty, T. & Alon, U. Coding limits on the number of transcription factors. *BMC Genomics* **7**, 239 (2006).
67. Payne, S. & You, L. Engineered cell–cell communication and its applications. *Adv. Biochem Eng. Biotechnol.* **146**, 97–121 (2014).
68. Duncker, K. E., Holmes, Z. A. & You, L. Engineered microbial consortia: strategies and applications. *Microb. Cell Fact.* **20**, 211 (2021).
69. Kylilis, N., Tuza, Z. A., Stan, G.-B. & Polizzi, K. M. Tools for engineering coordinated system behaviour in synthetic microbial consortia. *Nat. Commun.* **9**, 2677 (2018).
70. Weber, W., Daoud-El Baba, M. & Fussenegger, M. Synthetic ecosystems based on airborne inter- and intrakingdom communication. *Proc. Natl Acad. Sci. USA* **104**, 10435–10440 (2007).
71. Bacchus, W. & Fussenegger, M. Engineering of synthetic intercellular communication systems. *Metab. Eng.* **16**, 33–41 (2013).
72. Canton, B., Labno, A. & Endy, D. Refinement and standardization of synthetic biological parts and devices. *Nat. Biotechnol.* **26**, 787–793 (2008).
73. Weiss, R. & Knight, T. F. Engineered communications for microbial robotics. In *Revised Papers from the 6th International Workshop on DNA-Based Computers: DNA Computing* (eds. Condon, A. & Rozenberg, G.) 1–16 (Springer-Verlag, 2001).
74. Kong, W., Celik, V., Liao, C., Hua, Q. & Lu, T. Programming the group behaviors of bacterial communities with synthetic cellular communication. *Bioresour. Bioprocess.* **1**, 24 (2014).
75. Meyer, A. J., Segall-Shapiro, T. H., Glassey, E., Zhang, J. & Voigt, C. A. *Escherichia coli* 'Marionette' strains with 12 highly optimized small-molecule sensors. *Nat. Chem. Biol.* **15**, 196–204 (2019).
76. Vaiana, C. A. et al. Characterizing chemical signaling between engineered 'microbial sentinels' in porous microplates. *Mol. Syst. Biol.* **18**, e10785 (2022).
77. Chen, T., Ali Al-Radhawi, M., Voigt, C. A. & Sontag, E. D. A synthetic distributed genetic multi-bit counter. *iScience* **24**, 103526 (2021).
78. Al-Radhawi, M. A. et al. Distributed implementation of Boolean functions by transcriptional synthetic circuits. *ACS Synth. Biol.* **9**, 2172–2187 (2020).
79. Balagaddé, F. K. et al. A synthetic *Escherichia coli* predator–prey ecosystem. *Mol. Syst. Biol.* **4**, 187 (2008).
80. Danino, T., Mondragón-Palomino, O., Tsimring, L. & Hasty, J. A synchronized quorum of genetic clocks. *Nature* **463**, 326–330 (2010).
81. Payne, S. et al. Temporal control of self-organized pattern formation without morphogen gradients in bacteria. *Mol. Syst. Biol.* **9**, 697 (2013).
82. Alnahhas, R. N. et al. Majority sensing in synthetic microbial consortia. *Nat. Commun.* **11**, 3659 (2020).
83. Cao, Y. et al. Collective space-sensing coordinates pattern scaling in engineered bacteria. *Cell* **165**, 620–630 (2016).
84. Ausländer, D. et al. Programmable full-adder computations in communicating three-dimensional cell cultures. *Nat. Methods* **15**, 57–60 (2018).
85. Regot, S. et al. Distributed biological computation with multicellular engineered networks. *Nature* **469**, 207–211 (2011).
86. Sarkar, K., Chakraborty, S., Bonnerjee, D. & Bagh, S. Distributed computing with engineered bacteria and its application in solving chemically generated 2 × 2 maze problems. *ACS Synth. Biol.* **10**, 2456–2464 (2021).
87. Carignano, A. et al. Modular, robust, and extendible multicellular circuit design in yeast. *eLife* **11**, e74540 (2022).
88. Urrios, A. et al. A synthetic multicellular memory device. *ACS Synth. Biol.* **5**, 862–873 (2016).
89. Buluç, A., Meyerhenke, H., Safro, I., Sanders, P. & Schulz, C. Recent advances in graph partitioning. *Algorithm Engineering* (eds Kliemann, L. & Sanders, P.) 117–158 (Springer, 2016).
90. Hendrickson, B. & Kolda, T. G. Graph partitioning models for parallel computing. *Parallel Comput.* **26**, 1519–1534 (2000).
91. Augeri, C. J. & Ali, H. H. New graph-based algorithms for partitioning VLSI circuits. In *2004 IEEE International Symposium on Circuits and Systems (ISCAS)* Vol. 4, 521–524 (IEEE, 2004).
92. Chen, Y. P., Wang, T. C. & Wong, D. F. A graph partitioning problem for multi-chip design. In *1993 IEEE International Symposium on Circuits and Systems (ISCAS)* 1778–1781 (IEEE, 1993).
93. Perl, Y. & Snir, M. Circuit partitioning with size and connection constraints. *Networks* **13**, 365–375 (1983).
94. Diestel, R. *Graph Theory* 5th edn (Springer-Verlag, 2017).
95. Matula, D. W. & Beck, L. L. Smallest-last ordering and clustering and graph coloring algorithms. *J. ACM* **30**, 417–427 (1983).
96. Salis, H. M., Mirsky, E. A. & Voigt, C. A. Automated design of synthetic ribosome binding sites to control protein expression. *Nat. Biotechnol.* **27**, 946–950 (2009).
97. Lou, C., Stanton, B., Chen, Y. J., Munsky, B. & Voigt, C. A. Ribozyme-based insulator parts buffer synthetic circuits from genetic context. *Nat. Biotechnol.* **30**, 1137–1142 (2012).
98. Chen, Y.-J. et al. Characterization of 582 natural and synthetic terminators and quantification of their design constraints. *Nat. Methods* **10**, 659–664 (2013).
99. Shao, B. et al. Single-cell measurement of plasmid copy number and promoter activity. *Nat. Commun.* **12**, 1475 (2021).
100. Macia, J. & Sole, R. How to make a synthetic multicellular computer. *PLoS ONE* **9**, e81248 (2014).
101. Ausländer, S., Ausländer, D., Lang, P. F., Kemi, M. & Fussenegger, M. Design of multipartite transcription factors for multiplexed logic genome integration control in mammalian cells. *ACS Synth. Biol.* **9**, 2964–2970 (2020).
102. Groseclose, T. M., Rondon, R. E., Herde, Z. D., Aldrete, C. A. & Wilson, C. J. Engineered systems of inducible anti-repressors for the next generation of biological programming. *Nat. Commun.* **11**, 4440 (2020).
103. Groseclose, T. M. et al. Biomolecular systems engineering: unlocking the potential of engineered allostery via the lactose repressor topology. *Annu. Rev. Biophys.* **50**, 303–321 (2021).
104. Daniel, R., Rubens, J. R., Sarpeshkar, R. & Lu, T. K. Synthetic analog computation in living cells. *Nature* **497**, 619–623 (2013).
105. DeWeerdt, S. How to map the brain. *Nature* **571**, S6–S8 (2019).
106. Prindle, A. et al. A sensing array of radically coupled genetic 'biopixels'. *Nature* **481**, 39–44 (2011).
107. Ben Said, S., Tecon, R., Borer, B. & Or, D. The engineering of spatially linked microbial consortia—potential and perspectives. *Curr. Opin. Biotechnol.* **62**, 137–145 (2020).
108. Osmekhina, E. et al. Controlled communication between physically separated bacterial populations in a microfluidic device. *Commun. Biol.* **1**, 97 (2018).

109. Sardanyés, J., Bonforti, A., Conde, N., Solé, R. & Macia, J. Computational implementation of a tunable multicellular memory circuit for engineered eukaryotic consortia. *Front. Physiol.* **6**, 281 (2015).
110. Toda, S., Blauch, L. R., Tang, S. K. Y., Morsut, L. & Lim, W. A. Programming self-organizing multicellular structures with synthetic cell–cell signaling. *Science* **361**, 156–162 (2018).
111. Shirriff, K. Mining Bitcoin with pencil and paper: 0.67 hashes per day. http://www.righto.com/2014/09/mining-bitcoin-with-pencil-and-paper.html *Ken Shirriff's Blog* (2014).
112. Goñi-Moreno, A. & Amos, M. DiSCUS: a simulation platform for conjugation computing. In *Unconventional Computation and Natural Computation* (eds. Calude, C. S. & Dinneen, M. J.) 181–191 (Springer International Publishing, 2015).
113. Gutiérrez, M. et al. A new improved and extended version of the multicell bacterial simulator gro. *ACS Synth. Biol.* **6**, 1496–1508 (2017).
114. Gorochowski, T. E. Agent-based modelling in synthetic biology. *Essays Biochem.* **60**, 325–336 (2016).
115. Naylor, J. et al. Simbiotics: a multiscale integrative platform for 3D modeling of bacterial populations. *ACS Synth. Biol.* **6**, 1194–1210 (2017).
116. Rivest, R. The MD5 message-digest algorithm. *RFC* 10.17487/RFC1321 (1992).

Methods

Computational methods

All files are available at https://github.com/VoigtLab/MD5_circuit. The Verilog code for the MD5 algorithm was adapted from https://github.com/stass/md5_core/blob/master/md5_core.v. The code was modified to reduce the size of the inputs to 2-bits. Yosys[117] was used to synthesize the circuit and minimize the number of NOT and NOR gates using the control file 'md5_opt.ys'. This version of the complete circuit diagram was used for partitioning. After partitioning, wiring diagrams for individual cells were generated by writing individual structural Verilog files for each subcircuit, which were further minimized using Yosys if possible. Verilog files can be found at https://github.com/CIDARLAB/Cello-v2-1-Core/tree/main/library. Repressor secondary structures were annotated using Interpro[118] to identify Pfam domains. Individual helices within the helix-turn-helix domain were mapped using the Jpred 4 online web server[119] and further validated visually by multiple-sequence alignment to confirm predictions using MUSCLE[120] with the default parameters.

Partitioning algorithm

The algorithm aims to minimize the total number of cells required to implement a given circuit under two user-specified constraints: the maximum number of gates per cell and the total number of communication channels available. Circuit partitioning was done in two stages: subcircuit assignment and merging (Supplementary Fig. 2). In the subcircuit assignment stage, each gate was partitioned into a particular subcircuit. First, the number of intergate connections for each gate was determined. An intergate connection was defined as the total number of gates to which a given gate is connected. A subcircuit was initialized by randomly selecting a gate with the least number of intergate connections. Next, all gates connected to the current subcircuit (defined as any gate connected to any gate in the current subcircuit) were identified to generate a list of 'candidate' gates. From this list, a gate was randomly selected and added to the subcircuit. If adding the gate caused any of the initial constraints to be violated, this gate was removed from both the subcircuit and the list of candidate gates and another gate was randomly chosen from the list. This process was repeated until the subcircuit contained the user-specified maximum number of gates per cell or no more candidate gates were available to add to the subcircuit. This procedure was repeated until all gates were assigned to a subcircuit. During the merging stage, the goal was to optimize partitioning by combining smaller subcircuits. The subcircuits were merged, if possible, by randomly combining pairs of subcircuits while ensuring that the constraints were satisfied. First, a subcircuit was randomly chosen and all other subcircuits were placed into a randomly ordered list. The chosen subcircuit was merged with each subcircuit in the list until a merge that satisfied the constraints was found. If a merge was completed or no merge was found, another subcircuit was randomly chosen and all other subcircuits were placed into a randomly ordered list, with the process repeated. These iterations were continued until no additional merges were found. After all gates were partitioned into subcircuits (nodes), the wires (edges) between subcircuits were 'colored', where each color was abstractly associated with a chemical signal (undetermined at this point). Edges were required to be colored such that all edges sharing a node had a unique color (an 'edge coloring' problem). From the partitioned circuit diagram, a new graph was constructed to convert the task into a node coloring problem, where each node was colored such that connected nodes were assigned different colors (Supplementary Fig. 3). First, all gates containing an output that moves from one subcircuit to another were numbered and a graph was constructed with each numbered gate as a node. For each subcircuit, the gates that output a signal to the given subcircuit and all gates that output a signal from the given subcircuit were identified. Edges were drawn between all such gates and this process was repeated for each subcircuit. Note that, because they are connected to the same subcircuit, these gates require unique colors; thus, the constructed graph transforms the edge coloring problem into a node coloring problem. Node coloring was performed using the Welsh–Powell algorithm[95]. The vertices were ordered by the number of edges and each node was assigned a unique color such that no two connected nodes were assigned the same color (Supplementary Fig. 3). The algorithm sought to minimize the number of colors used. Once the nodes were colored, they were mapped back to the original gates and the edges were colored according to the node color. This process was repeated a minimum of $n = 1{,}000$ times and the partition with the smallest number of subcircuits was chosen.

Strains, DNA constructs, media and chemicals

Plasmid cloning was performed in *E. coli* NEB10β competent cells (NEB, C3019I). When the plasmid contained an R6K origin, cloning was performed using *E. coli* JTK164A or *E. coli* TransforMax EC100D pir^+ (Lucigen, CP09500). Strains modified to contain sensors or circuits were based on *E. coli* MG1655: *E. coli* YJP_MKC174 (containing plYJP064-Sensor) or *E. coli* JAI_MKC300 (a Δ*araC* derivative of *E. coli* YJP_MKC173 containing the Marionette sensor array). Supplementary Tables 8–10 list all strains used in this study. DNA sequences for all constructs except those used to generate carrying *E. coli* strains are provided in Supplementary Table 11. Subcircuit constructs are provided as GenBank files at https://doi.org/10.5281/zenodo.13247698. Plasmid maps are provided in Supplementary Fig. 16. LB medium (Difco, 244620) and LB medium + 2% Bacto-agar (Difco, 244620) plates were used for all routine cloning. Minimal M9 medium was used for all assays (unless otherwise noted): 1× M9 salts (Difco, 248510), 2 mM $MgSO_4$ (Affymetrix, 18651), 100 µM $CaCl_2$ (Sigma, C1016), 0.2% Casamino acids (BD, 223050), 0.4% glucose (Fisher Chemical, D16-1), 0.34 mg ml^{-1} Vitamin B_1 (Alfa Aesar, A19560). SOC recovery medium (NEB, B9020S) was used for recovery after transformation. Antibiotics were used at the following concentrations: 50 µg ml^{-1} kanamycin (Kan, GoldBio, K-120-10), 100 µg ml^{-1} carbenicillin (Carb, GoldBio, C-103-5), 5 µg ml^{-1} tetracycline (Tet, GoldBio, T-101-25), 50 µg ml^{-1} spectinomycin (Spec, GoldBio, S-140-5), 20 µg ml^{-1} gentamicin (Gm, Enzo Lifesciences, no. 380-003-G001) and 25 µg ml^{-1} chloramphenicol (Cm, Alfa Aesar, B20841). Cells were induced with the following: isopropyl β-d-1-thiogalactopyranoside (IPTG, GoldBio, I2481C), anhydrotetracycline (aTc, Sigma, 37919), cuminic acid (Cuma, Sigma, 268402), vanillic acid (Van, Sigma, 94770), L-arabinose (Ara, Sigma, A3256), 3-oxohexanoyl-homoserine lactone (OC6, Millipore Sigma, K3007), 3-hydroxytetradecanoyl-homoserine lactone (OHC14, Sigma, 51481), 2,4-diacetylphophloroglucinol (DAPG, Santa Cruz Biotechnology, sc-206518), *p*-coumaroyl-homoserine lactone (pC-HSL, Millipore Sigma, 07077) and *p*-coumarate (Sigma, C9008). For cytometry, cells were diluted in PBS (EMD Millipore, 6505). The *yfp*, *rfp* and *bfp* genes were *eYFP*, *mRFP* and *mTagBFP2** (a derivative of *mTagBFP2* with restriction sites removed).

Flow cytometry

Fluorescence was measured using a BD LSRII Fortessa flow cytometer with an HTS attachment running BD FACSDIVA v8.0 software. At least 30,000 events were recorded for each sample. The FITC, PE-Texas Red and Pacific Blue channels were used to collect data for YFP, RFP and BFP, respectively. The Cytoflow Python package was used to process FCS 3.0 files and gate cells. The FSC, SSC, FITC, PE-Texas Red and Pacific Blue voltages were set to 750, 300, 450, 600 and 418 V, respectively. The medians of the distributions are reported. Fluorescence in arbitrary units was converted to RNAP/s as follows. First, *E. coli* YJP_MKC254 (containing the reference promoter BBa_J23101 fused to *yfp* in the attB2 landing pad) was cultured under the same conditions as the sample of interest. Autofluorescence was measured using *E. coli* YJP_MKC174 (for NOT gate strains) or *E. coli* JAI_MKC300 (for all other strains) grown under the same conditions as the sample of interest. The BBa_J23101

promoter was measured previously to generate 0.029 RNAP/s/DNA[99] and the copy number of the attB2 landing pad was measured to be 3.5 under similar growth conditions[33], yielding a total RNAP flux of 0.102 RNAP/s. Arbitrary units were converted using the following equation: $(0.102)((\langle YFP\rangle_{measured}) - (\langle YFP\rangle_{blank}))/((\langle YFP\rangle_{BBa\_J23101}) - (\langle YFP\rangle_{blank}))$, where $(\langle YFP\rangle_{measured})$, $(\langle YFP\rangle_{BBa\_J23101})$ and $(\langle YFP\rangle_{blank})$ are the median fluorescence values (in arbitrary units) obtained from the sample of interest, reference promoter and appropriate autofluorescence control, respectively. Two subcircuits (sc5 and sc7) had multiple outputs that need to be characterized with reporters whose signals were distinct from YFP. To convert the fluorescence values (in arbitrary units) for RFP and BFP to units of RNAP/s, the following protocol was used. The BBa_J23101 promoter was fused to the *rfp* or *bfp* gene with the BBa_B0064 and BBa_B0034 RBSs and the ECK120017009 and ECK120033737 terminators, respectively, to create expression cassettes. Note that the RBS for the *yfp* cassette was BBa_B0034 and the terminator was DT3 in *E. coli* strain YJP_MKC254. These were inserted into the attB2 landing pad of *E. coli* JAI_MKC300 to create *E. coli* JAI_MKC399 and *E. coli* JAI_MKC400 (Supplementary Fig. 5). Autofluorescence was measured in the relevant channels. Fluorescence values (in arbitrary units) were then converted to RNAP/s using the equation described above except that <YFP> was replaced with the fluorescence of the corresponding reporter. To convert plasmid-based NOT gates to RNAP/s, the same protocol was used, except that the BBa_J23101 reference promoter was fused to *yfp* with BBa_B0064 as the RBS and L3S2P21 as the terminator and placed onto a p15a plasmid (pJSBS_RPU). The plasmid was carried in *E. coli* NEB10β. The copy number of p15a in *E. coli* NEB10β was estimated to be nine under similar growth conditions[33]. Multiplying this value by 0.029 RNAP/s/DNA[99] yielded a total RNAP flux of 0.261 RNAP/s for plasmid-borne BBa_J23101. Fluorescence (in a.u.) was then converted to RNAP/s using the equation described above except that the 0.102 value was replaced with 0.261. Distributions were converted to RNAP/s using the same protocol, except that unit conversion was performed on a per-cell basis rather than using median values (Supplementary Fig. 10).

NOT gate characterization for genome-encoded gates

Strains were streaked from glycerol stocks onto LB-agar plates with Kan and grown overnight. Single colonies were picked and cultured overnight in 400 µl M9 medium in 2-ml 96-deep-well plates (USA Scientific, 1896-2000) covered with AeraSeal film (Excel Scientific) and grown at 37 °C and 900 rpm (InforsHT Multitron Pro shaker incubator). The cultures were diluted 1:100 into 400 µl M9 medium and grown for 1.5 h under the same conditions. The cultures were diluted 1:1,000 into 400 µl M9 medium containing 0, 10, 20, 30, 40, 50, 70, 100, 150, 200 or 1,000 µM IPTG or 1,000 µM IPTG + 200 nM aTc and grown for 4.5 h under the same conditions. A 50-µl aliquot of each culture was then diluted into 180 µl PBS containing 1 mg ml^{-1} Kan for flow cytometry analysis. To obtain the response functions, the data were fitted to equation (1) using the SciPy Python package scipy.optimize.curve_fit(). *E. coli* JAI_MKC148 carrying an integrated cassette (attB2 landing pad) with a P_{Tet}–P_{Tac} promoter fused to *yfp* was run in parallel under the same conditions to convert the *x* axis into arbitrary units.

Growth impact of NOT gates carried in the genome

Strains were streaked from glycerol stocks onto LB-agar plates with Kan and grown overnight. Single colonies were inoculated into 400 µl M9 medium in 2-ml 96-deep-well plates (USA Scientific, 1896-2000), covered with AeraSeal film and grown overnight at 37 °C and 900 rpm (InforsHT Multitron Pro shaker incubator). The cultures were then diluted 1:100 into 400 µl M9 medium and grown for 1.5 h. Cultures were then diluted 1:1,000 in M9 medium with appropriate inducer(s) and grown under the same conditions for 5.5 h. The OD_{600} was measured by taking a 200-µl aliquot of the culture and transferring it to a Nunc 96-well plate with an optically clear bottom (Thermo Scientific, 165305). The OD_{600} was then measured using a Synergy H1 plate reader (BioTek Instruments), from which the OD_{600} of the M9 medium alone was subtracted. To normalize these data, they were divided by the OD_{600} obtained when the repressor was not expressed (no inducer), also subtracting the OD_{600} of the M9 medium.

Characterization of crosstalk between phage repressors and promoters

The 144 *E. coli* crosstalk strains (Supplementary Fig. 4) were streaked from glycerol stocks onto LB-agar plates with Kan and Carb and grown overnight. Each crosstalk strain contained a different combination of repressor and output promoter (constructed using plasmids plJAI_JR(1–12)-cross and plJAI_pJR(1–12); Supplementary Fig. 14). Individual colonies were picked, inoculated into 150 µl M9 medium in shallow-bottom 96-well plates (Thermo Scientific, 249662) and cultured overnight at 37 °C and 1,000 rpm in an ELMI shaker. Aliquots were then diluted 1:100 into 150 µl M9 medium and cultured for 1 h under the same conditions. Aliquots were then diluted 1:1,000 into 150 µl M9 medium with and without 1 mM IPTG and grown for 4.5 h under the same conditions. A 50-µl aliquot of the culture was then diluted into 180 µl PBS with 1 mg ml^{-1} Kan and fluorescence was measured by flow cytometry.

Calculation of total recombinant DNA and number of regulators

The following counting methods were used to compute the distributions shown in Fig. 3a. For each cell, the 'total DNA' counts all the synthetic DNA added to the cell, including the entire sensor array, plasmid backbones, etc. The number of regulator genes counts all synthetic genes added to the cell, including two repressor genes for each NOR gate, the sensors required for the circuit and the genes that produce communication signals. Antibiotic markers and the *repA* gene required for replication of the pSC101 origin were excluded from the count. In the Supplementary Subcircuit Datasheets, 'subcircuit DNA alone' counts the DNA integrated into the attB2 and attB7 landing pads.

Characterization of the ON/OFF states of sensors for genetic circuit design

Sensor strains (*E. coli* JAI_MKC269, JAI_MKC322, JAI_MKC323, JAI_MKC334, JAI_MKC335, JAI_MKC336, JAI_MKC337, JAI_MKC338, JAI_MKC340, JAI_MKC342) were streaked from glycerol stocks onto LB-agar plates with Kan. Single colonies of each strain were inoculated into 150 µl M9 medium and grown for 16 h in shallow-bottom plates (Thermo Scientific, 249662) at 37 °C and 1,000 rpm (ELMI plate shaker). Cells were then diluted 1:100 into 150 µl M9 medium and cultured for 1.5 h under the same conditions. Then, cells were diluted 1:2,000 in M9 medium with and without inducer and cultured for 5 h under the same conditions. Lastly, 50 µl aliquots were diluted into 180 µl PBS with 1 mg ml^{-1} Kan, analyzed via flow cytometry and converted to RNAP/s. Inducers and their concentrations were as follows: 25 µM DAPG, 100 µM Cuma, 10 µM OC6, 100 µM Van, 1,000 µM IPTG, 200 nM aTc, 4,000 µM Ara, 100 µM Sal, 10 µM pC-HSL and 10 µM OHC14. The sensor strain diagrams are shown in Supplementary Fig. 4.

Characterization of sender–receiver response functions

Sender strains (*E. coli* sLux, sCin, sRpa, sPhl, Kan or Gm as appropriate) and receiver strains (*E. coli* rLux, rCin, rRpa or rPhl) were streaked from glycerol stocks onto LB-agar plates. Single colonies of the sender cells were picked into 100 µl M9 medium (with Gm for *E. coli* sPhl and no antibiotics otherwise) and grown for 8 h in shallow-bottom plates (Thermo Scientific, 249662) at 37 °C and 1,000 rpm (ELMI plate shaker). Cells were then diluted 1:2,500 into 150 µl M9 medium containing 0, 10, 20, 30, 40, 50, 70, 100, 150, 200 or 1,000 µM IPTG and cultured for 16 h under the same conditions. Cells were then diluted 1:1,000 into 1 ml M9 medium (for *E. coli* sRpa, the medium also contained 100 nM

p-Coum) and cultured for 4 h at 37 °C and 900 rpm (InforsHT Multitron Pro shaker incubator) in 2 ml deep-well plates (USA Scientific, 1896-2000) under the same induction conditions as before. The plates were then spun at 4,500*g* for 10 min at room temperature to pellet the cells. A 400-µl aliquot of the supernatant medium was filter sterilized using a 0.2-µm regenerated cellulose filter (Chrom Tech, 96F-RC020). Single colonies of receiver cells were cultured for 16 h in M9 medium containing Gm. Receiver cells were diluted 1:1,000 into 150 µl of the sender supernatant and cultured for 3 h in shallow-bottom plates (Thermo Scientific, 249662) at 37 °C and 1,000 rpm (ELMI plate shaker). Aliquots (50 µl) of the culture were then diluted into 180 µl PBS with 1 mg ml^{-1} Kan, analyzed via flow cytometry, converted to RNAP/s and fitted to equation (2) (Supplementary Fig. 8).

Circuit characterization (fluorescence)

Strains were streaked from glycerol stocks onto LB-agar plates containing appropriate antibiotics and incubated at 37 °C overnight. Single colonies were inoculated into 100 µl M9 medium (with 20 µg ml^{-1} Gm if appropriate) and grown in shallow-bottom plates (Thermo Scientific, 249662) at 37 °C and 1,000 rpm (ELMI plate shaker) for 8 h. The cultures were then diluted 1:2,500 into 150 µl M9 medium (with Gm if appropriate) with appropriate inducers and grown in shallow-bottom plates at 37 °C and 1,000 rpm for 16 h (ELMI plate shaker). Then, a 1-µl aliquot of the culture was diluted into 300 µl PBS containing 1 mg ml^{-1} Kan for flow cytometry. The medians of the resulting distributions were used to calculate the activities of the output promoters.

Circuit characterization (to receiver cells)

To induce the receiver cells via cell–cell communication signals, the above 'circuit characterization' assay was continued as follows. Aliquots of the 16-h culture were diluted 1:1,000 into 1 ml M9 medium with appropriate inducers (and Gm if appropriate) in 2-ml deep-well plates (USA Scientific, 1896-2000) and cultured for 4 h at 37 °C and 900 rpm (InforsHT Multitron Pro shaker incubator). For strains producing pC-HSL, the medium also contained 100 nM *p*-Coum. The plates were then spun at 4,500*g* for 10 min at room temperature to pellet cells. From the plates, 500 µl of the supernatant was aspirated and filtered to remove cells. When the circuit outputs led to the production of OC6, pC-HSL or DAPG, the samples were filtered using either cellulose acetate 96-well filter plates (Cytiva Life Sciences, 7700-2808) or regenerated cellulose 96-well filter plates (Chrom Tech, 96F-RC020). OHC14-producing strains were processed using the Chrom Tech filters because we found that OHC14 does not pass through cellulose acetate filters. The filtered supernatant was then used to induce the appropriate receiver cells: *E. coli* rLux, *E. coli* rCin, *E. coli* rRpa and *E. coli* rPhl. The receiver cells were prepared by streaking from glycerol stocks onto LB-agar plates with 20 µg ml^{-1} Gm followed by incubation at 37 °C overnight. Single colonies were picked into 150 µl M9 medium and grown for 16 h in shallow-bottom plates at 37 °C and 1,000 rpm. Aliquots were taken and diluted 1:1,000 into 150 µl of the supernatant collected from the circuit and incubated for 3 h at 37 °C and 1,000 rpm in shallow-bottom plates. Aliquots (50 µl) of the culture were then diluted into 180 µl PBS containing 1 mg ml^{-1} Kan and analyzed via flow cytometry.

Growth impact of subcircuit sc5

Strains of *E. coli* G6 and *E. coli* JAI_MKC300 were streaked from glycerol stocks onto LB-agar plates (with Gm as appropriate) and grown overnight. Single colonies were picked into 100 µl M9 medium (with Gm as appropriate) and cultured in shallow-bottom Nunc 96-well plates (Thermo Scientific, 249662) at 37 °C and 1,000 rpm in an ELMI shaker. The strains were then diluted 1:2,500 in 1 ml M9 medium and appropriate inducers (with Gm if necessary) and cultured for 16 h at 37 °C and 900 rpm (InforsHT Multitron Pro shaker incubator) in 2-ml deep-well plates (USA Scientific, 1896-2000). The cells were then diluted 1:100 in 1 ml M9 medium and cultured for 2 h under the same conditions. Then 900 µl of medium was used to measure the OD_{600} in a spectrophotometer (Agilent Cary 60 UV-Vis). Each sample was diluted to an OD_{600} of 0.1 and grown under the same conditions for 2 h before measuring the OD_{600} again. Doubling time was calculated assuming exponential growth by multiplying the elapsed time (in minutes) divided by the number of doublings in that time ($\log_2$(final OD_{600}/initial OD_{600})) (Supplementary Fig. 12).

Co-culture of *E. coli* subcircuits

These experiments correspond to Supplementary Fig. 14. Subcircuit strains were streaked from glycerol stocks onto LB-agar plates containing appropriate antibiotics and incubated at 37 °C overnight. Single colonies were inoculated into 100 µl M9 medium and grown in shallow-bottom plates (Thermo Scientific, 249662) at 37 °C and 1,000 rpm (ELMI plate shaker) for 8 h. The two subcircuit cultures were combined in a co-culture, diluted 1:2,500 in 150 µl M9 medium with appropriate inducers and grown in shallow-bottom plates at 37 °C and 1,000 rpm (ELMI plate shaker) for 16 h. Aliquots of the 16-h culture were diluted 1:1,000 into 1 ml M9 medium with different combinations of inducers in 2-ml deep-well plates (USA Scientific, 1896-2000) and cultured for 4 h at 37 °C and 900 rpm (InforsHT Multitron Pro shaker incubator). For strains producing pC-HSL, 100 nM Coum was added to the medium. The plates were then spun at 4,500*g* for 10 min at room temperature to pellet cells. From the plates, 600 µl of the supernatant was aspirated and filtered to remove cells. The samples were filtered through regenerated cellulose 96-well filter plates (Chrom Tech, 96F-RC020). The filtered supernatant was then used to induce the receiver cells. The receiver cells were prepared by streaking from glycerol stocks onto LB-agar plates with 20 µg ml^{-1} Gm and incubating at 37 °C overnight. Single colonies were picked into 150 µl M9 medium and grown for 16 h in shallow-bottom plates at 37 °C and 1,000 rpm. Aliquots were taken and diluted 1:1,000 into 150 µl of the supernatant collected from the circuit and incubated for 3 h at 37 °C and 1,000 rpm in shallow-bottom plates. Aliquots (50 µl) of the culture were then diluted in 180 µl PBS with 1 mg ml^{-1} Kan and analyzed via flow cytometry.

Reporting summary

Further information on research design is available in the Nature Portfolio Reporting Summary linked to this article.

Data availability

Sequences for strains and plasmids used in this work are included in the Supplementary Information file. GenBank files of full constructs for each subcircuit can be found at https://doi.org/10.5281/zenodo.13247698 ref. 121. Additional data are available from the corresponding author upon reasonable request. Source data are provided with this paper.

Code availability

Cello 2.1 is available at cellocad.org and can be accessed via Google account. All files for Cello 2.1 can be found at https://github.com/CIDARLAB/Cello-v2-1-Core/tree/main/library. The script used to simulate the MD5 algorithm can be found at https://github.com/VoigtLab/MD5_Circuit. The manual for Cello 2.1 is provided as Supplementary Software.

References

117. Wolf, C. *Design and Implementation of the Yosys Open SYnthesis Suite* https://yosyshq.net/yosys/files/yosys_manual.pdf (2013).
118. Paysan-Lafosse, T. et al. InterPro in 2022. *Nucleic Acids Res.* **51**, D418–D427 (2022).
119. Drozdetskiy, A., Cole, C., Procter, J. & Barton, G. J. JPred4: a protein secondary structure prediction server. *Nucleic Acids Res.* **43**, W389–W394 (2015).

120. Edgar, R. C. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res.* **32**, 1792–1797 (2004).
121. Voight, C & Sun, J. Subcircuit genome files. *Zenodo* https://doi.org/10.5281/zenodo.13247698 (2004).

Acknowledgements

We thank J. Roberts (Boston University) and S. Oliveira (North Carolina A&T State University) for their help in developing Cello 2.1. This work was supported by funding from the National Science Foundation SemiSynBio program awards CCF-1807575 (J.P., J.S., C.A.V.) and CCF-1849588 (W.C., E.D.S., C.A.V.); DARPA Synergistic Discovery and Design program (SD2) award FA8750-17-C-0229 (J.P., J.S., C.A.V.); an award from the Schmidt Innovation Fellows Program (J.P., J.S., C.A.V.); Air Force Office of Scientific Research award FA9550-22-1-0316 (W.C., E.D.S); National Science Foundation award 2211040 (Y.Z., D.D.) and National Science Foundation's Semiconductor Synthetic Biology for Information Storage and Retrieval award 2027045 (C.K., W.Z.H.).

Author contributions

J.P., J.S. and C.A.V. conceived the study and designed the experiments. J.P. and J.S. performed the experiments and analyzed the data. W.C., Y.Z., D.D. and E.S. implemented the partitioning and edge coloring algorithm. C.K., W.Z.H. and D.D. developed Cello 2.1. J.P., J.S. and C.A.V. wrote the manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41589-024-01730-1.

Correspondence and requests for materials should be addressed to Christopher A. Voigt.

Peer review information *Nature Chemical Biology* thanks Irene Otero-Muras, Xiao-Jun Tian and the other, anonymous, reviewer(s) for their contribution to the peer review of this work.

nature portfolio

Corresponding author(s): Christopher Voigt

Last updated by author(s): Aug 8, 2024

Reporting Summary

Nature Portfolio wishes to improve the reproducibility of the work that we publish. This form provides structure for consistency and transparency in reporting. For further information on Nature Portfolio policies, see our Editorial Policies and the Editorial Policy Checklist.

Statistics

For all statistical analyses, confirm that the following items are present in the figure legend, table legend, main text, or Methods section.

n/a	Confirmed
☐	☒ The exact sample size (*n*) for each experimental group/condition, given as a discrete number and unit of measurement
☐	☒ A statement on whether measurements were taken from distinct samples or whether the same sample was measured repeatedly
☒	☐ The statistical test(s) used AND whether they are one- or two-sided *Only common tests should be described solely by name; describe more complex techniques in the Methods section.*
☒	☐ A description of all covariates tested
☒	☐ A description of any assumptions or corrections, such as tests of normality and adjustment for multiple comparisons
☐	☒ A full description of the statistical parameters including central tendency (e.g. means) or other basic estimates (e.g. regression coefficient) AND variation (e.g. standard deviation) or associated estimates of uncertainty (e.g. confidence intervals)
☒	☐ For null hypothesis testing, the test statistic (e.g. *F*, *t*, *r*) with confidence intervals, effect sizes, degrees of freedom and *P* value noted *Give P values as exact values whenever suitable.*
☒	☐ For Bayesian analysis, information on the choice of priors and Markov chain Monte Carlo settings
☒	☐ For hierarchical and complex designs, identification of the appropriate level for tests and full reporting of outcomes
☒	☐ Estimates of effect sizes (e.g. Cohen's *d*, Pearson's *r*), indicating how they were calculated

Our web collection on statistics for biologists contains articles on many of the points above.

Software and code

Policy information about availability of computer code

Data collection	BD FACSDIVA software v8.0 was used to collect cytometry data.
Data analysis	Genetic circuit design was performed with Cello 2.1. Flow cytometry data was analyzed using the Cytoflow package v1.2 and scipy v1.7.3. The MD5 circuit was simulated using a custom Python 3.9 script which can be found at https://github.com/VoigtLab/MD5_circuit.

For manuscripts utilizing custom algorithms or software that are central to the research but not yet described in published literature, software must be made available to editors and reviewers. We strongly encourage code deposition in a community repository (e.g. GitHub). See the Nature Portfolio guidelines for submitting code & software for further information.

Data

Policy information about availability of data

All manuscripts must include a data availability statement. This statement should provide the following information, where applicable:

- Accession codes, unique identifiers, or web links for publicly available datasets
- A description of any restrictions on data availability
- For clinical datasets or third party data, please ensure that the statement adheres to our policy

Sequences for strains and plasmids used in this work are included in the Supplementary Information file. GenBank files of full constructs for each subcircuit can be found at doi:10.5281/zenodo.13247698. Medians from flow cytometry experiments can be found in the Source Data and Supplementary Source data files. Additional data available from the corresponding author upon reasonable request.

Research involving human participants, their data, or biological material

Policy information about studies with human participants or human data. See also policy information about sex, gender (identity/presentation), and sexual orientation and race, ethnicity and racism.

Reporting on sex and gender	N/A
Reporting on race, ethnicity, or other socially relevant groupings	N/A
Population characteristics	N/A
Recruitment	N/A
Ethics oversight	N/A

Note that full information on the approval of the study protocol must also be provided in the manuscript.

Field-specific reporting

Please select the one below that is the best fit for your research. If you are not sure, read the appropriate sections before making your selection.

☒ Life sciences ☐ Behavioural & social sciences ☐ Ecological, evolutionary & environmental sciences

For a reference copy of the document with all sections, see nature.com/documents/nr-reporting-summary-flat.pdf

Life sciences study design

All studies must disclose on these points even when the disclosure is negative.

Sample size	No statistical methods were used to predetermine sample size. At least three biological replicates were performed in separate days for all experiments. Three replicates was sufficient for consistent and reproducible results to support claims made in this paper.
Data exclusions	All data was included.
Replication	All experimental claims have been tested at least three times on different days, following the reported methods. Results were repeatable across replicates.
Randomization	Randomization was not relevant from this work as the control and experimental samples used the same starting culture and materials, experiments were performed simultaneously with the same procedure, and samples were analyzed individually then compared.
Blinding	Blinding was not applicable for our study as data collection utilized objective methods like flow cytometry. Samples were individually analyzed using identical methods.

Reporting for specific materials, systems and methods

We require information from authors about some types of materials, experimental systems and methods used in many studies. Here, indicate whether each material, system or method listed is relevant to your study. If you are not sure if a list item applies to your research, read the appropriate section before selecting a response.

Materials & experimental systems

n/a	Involved in the study
☒	☐ Antibodies
☒	☐ Eukaryotic cell lines
☒	☐ Palaeontology and archaeology
☒	☐ Animals and other organisms
☒	☐ Clinical data
☒	☐ Dual use research of concern
☒	☐ Plants

Methods

n/a	Involved in the study
☒	☐ ChIP-seq
☐	☒ Flow cytometry
☒	☐ MRI-based neuroimaging

Plants

Seed stocks	N/A
Novel plant genotypes	N/A
Authentication	N/A

Flow Cytometry

Plots

Confirm that:

☒ The axis labels state the marker and fluorochrome used (e.g. CD4-FITC).

☒ The axis scales are clearly visible. Include numbers along axes only for bottom left plot of group (a 'group' is an analysis of identical markers).

☒ All plots are contour plots with outliers or pseudocolor plots.

☒ A numerical value for number of cells or percentage (with statistics) is provided.

Methodology

Sample preparation	Bacterial cells were prepared as described in the Methods and diluted into PBS before running in the flow cytometer.
Instrument	BD LSRII Fortessa flow cytometer
Software	Data was collected using FACS Diva Software 8.0; Scipy v1.7.3 and Cytoflow vl.2 was used for data analysis.
Cell population abundance	30,000 gates cells per sample.
Gating strategy	The following gating strategy was used to separate E.coli cells from debris: 1,500-10,000 FSC-H and 5,000-10,000 SSC-H for stationary phase cells and 3,000-11,500 FSC-H and 10,000-10,000 SSC-H for mid-exponential cells.

☒ Tick this box to confirm that a figure exemplifying the gating strategy is provided in the Supplementary Information.