Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Synthetic neuromorphic computing in living cells

Luna Rizik
Technion – Israel Institute of Technology
Loai Danial
Technion – Israel Institute of Technology
Mouna Habib
Technion – Israel Institute of Technology
Ron Weiss
Massachusetts Institute of Technology https://orcid.org/0000-0003-0396-2443
Ramez Daniel (ramizda@technion.ac.il)
Technion – Israel Institute of Technology https://orcid.org/0000-0001-8886-3668

Article

Keywords: biological regulatory networks, synthetic neuromorphic computing, neuronal networks

Posted Date: September 14th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-901884/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Version of Record: A version of this preprint was published at Nature Communications on September 24th, 2022. See the published version at https://doi.org/10.1038/s41467-022-33288-8.

Title: Synthetic neuromorphic computing in living cells

Authors: Luna Rizik[1]†, Loai Danial[2]†, Mouna Habib[1]†, Ron Weiss[3], and Ramez Daniel[1*]

Affiliations:

[1]*Department of Biomedical Engineering, Technion - Israel Institute of Technology, Haifa 3200003 Israel.*

[2] *Andrew and Erna Viterbi Faculty of Electrical Engineering*, *Technion - Israel Institute of Technology, Haifa 3200003 Israel.*

[3] *Synthetic Biology Center, Massachusetts Institute of Technology, Cambridge, USA, Department of Biological Engineering, Massachusetts Institute of Technology, Cambridge, USA*

*Corresponding author. Email: ramizda@bm.technion.ac.il

†These authors contributed equally to this work.

Abstract:

Biological regulatory networks in cells and neuronal networks employ complex circuit topologies with highly interconnected nodes to perform sophisticated information processing. Despite the complexity of neuronal networks, their information processing and computational capabilities can be recapitulated using simplified models comprising repeated connected nodes, e.g., perceptrons, with decision-making capabilities and flexible weighted links. Here, we argue that analogous to their revolutionary impact on computing, neuro-inspired models can similarly transform synthetic gene circuit design in a manner that is reliable, efficient in resource utilization, and can be readily reconfigurable for new tasks. We introduce neuromorphic design for synthetic gene circuits by first defining the perceptgene, a perceptron that computes in the logarithmic domain, which enables efficient implementation of artificial neural networks in the cellular milieu. Working in *Escherichia coli* cells, we experimentally demonstrated logarithmic scale analog multiplication using a single perceptgene. We modified perceptgene parameters (weights and biases) to create devices that compute a log-transformed negative rectifier encoding the minimum operation, log-transformed positive rectifier encoding the maximum operation, and log-transformed average of analog inputs. We then created multi-layer perceptgene circuits that compute a majority function, perform analog-to-digital conversion, and implement a ternary switch. Experimental and theoretical analysis showed that our approach enables circuit optimization via artificial intelligence algorithms such as gradient descent and backpropagation. Realizing neural-like computing in the noisy resource-limited environments of individual cells marks an important step towards synthetic biological systems that can be engineered effectively via supervised ANN optimization algorithms.

Main Text:

A central goal of synthetic biology (*1–9*) is to create large-scale genetic networks in living cells that implement sophisticated sensing, processing, and actuation (*10–14*). To date, both the digital and analog computing paradigms have been implemented in living cells in an attempt to design and build genetic circuits efficiently (*15*). The digital paradigm, which abstractly computes with two discrete binary-coded levels [0,1] (*16*), has inspired implementation of wide variety of genetic circuits, including logic gates (*17–19*), memory elements (*20–22*), a counter (*23*), state machines (*24*), a toggle switch (*25*), a digitizer (*26*), and highly complex logic functions (*27*, *28*). The analog paradigm, in contrast, computes on a continuous set of numbers and has been suggested as an alternative to the digital paradigm for tasks that don't require decision-making (*29–31*). For example, it was shown that complex analog processing functions such as linearization, addition and ratiometers can be implemented with only a few transcription factors in living cells (*29*). Efforts in synthetic biology have also focused on other aspects of circuit control, such as complex temporal dynamics (*32–34*), and integral feedback controllers for robust adaptation (*35*, *36*).

Despite the many successful accomplishments to-date, significant challenges hinder further scaling of synthetic biological systems based on digital and analog computing (*1*, *37*). Critical impediments include cellular resource limitations, high levels of random fluctuations, and undesirable interactions between synthetic parts and host cells (*1*, *37*, *38*). Furthermore, digital design is often not suitable for computing with graded biological signals, while analog circuits may accumulate noise as they scale in size (*38*). To overcome these challenges, we have sought to adapt abstract models of neural networks in the form of neuromorphic computing (*39–41*) into individual single cells. We also were motivated by theoretical analyses from the early days of

systems biology that demonstrated how certain gene regulatory networks execute neural-like computations (*42–45*).

Neuromorphic computing, which has become pervasive as of late, has been successfully applied to a wide range of fields including electronics (*41*, *46–48*), optics (*49*), software algorithms (*40*), and even *in-vitro* DNA computing (*50*). Artificial intelligence systems efficiently solve complex tasks such as content addressable memory, pattern classification, object recognition, optimization, and prediction through machine learning algorithms (*51*). An advantage of neuromorphic computing systems compared to their digital counterparts is that implementing a given task often requires fewer computational devices (*41*, *47*), which is especially important when resources are scarce (such as synthetic biology). Neuromorphic systems usually combine analog information processing with decision-making capabilities and support for a particular kind of iterative optimization strategies. In these optimization strategies, characterization of circuit behavior is interleaved with iterative changes in computing device parameters, for example based on prediction of how changes under consideration correlate with the derivative of an overall score function.

Bio-inspired information processing systems that implement artificial neural networks (ANNs) operate differently than conventional computing (*40*). ANNs use analog information processing units that collectively interact through interconnected non-linear functions (fig. S1.1). The fundamental building block of an ANN is a *perceptron* (*52*), which consists of a linear combination of weighted analog input signals (fig. S1.2). The analog computation result serves as an input to an activation function that asymptotically computes the perceptron's non-linear output levels (e.g., sigmoid function $z = \frac{e^y}{1+e^y}$, where the input to the activation function is y and the output z is between 0 and 1). In considering an adaptation of this model to gene regulatory networks inside

individual living cells, it is worthwhile to note that biological pathways often operate in a non-linear fashion and exhibit logarithmic and power law input-output relations, where outcomes are dictated by relative fold-changes rather than absolute levels (*29*, *53*, *54*). Therefore, to implement ANNs in living cells, we define a log-based version of a perceptron, termed the *perceptgene* (Fig. 1A), whose logarithmic input-output operation makes it more suitable for the non-linear nature of biochemical reactions and gene regulation. The perceptgene implements a *logarithmically-separable classifier* that usually partitions all input values into two classes of output data points, whereby for example for a 2-input perceptgene a line demarcating the two classes can be drawn in a logarithmic graph (Fig. 1A, right).

In the perceptgene design operating in the logarithmic domain, the perceptron's linear operations of scalar multiplication and summation are transformed to exponentiation (power-law) and multiplication, respectively (Fig. 1A). The perceptgene's sigmoid activation function is described by Michaelis–Menten kinetics at steady state (*29*, *38*) ($z = \frac{e^{ln(y)}}{1+e^{ln(y)}}$, where y is a scaled protein concentration) and also operates in the logarithmic domain. In order to make the perceptgene operation more compatible with the realities of gene expression, we added basal expression β to the activation function ($z = \frac{e^{ln(y)}+\beta}{1+e^{ln(y)}+\beta}$, $\beta << 1$) (Fig. 1A). As we discuss below, the perceptron's high-level operations, e.g. weighted multi-input functions, classification, and gradient descent for learning algorithms (*41*, *47*) are supported by the perceptgene's log-based computing.

The perceptgene output is computed as a linear combination in the logarithmic domain (i.e. multiplication) of the weighted inputs and the bias in comparison to the threshold of the activation function (Supplementary Information, Section 1 - Box 1). In the perceptgene implementation, the weights are mainly determined by the Hill coefficients and design topology (e.g., feedback loops). The bias is set by the logarithmic transform of the ratio between the maximum protein level and

the binding affinities of protein-protein/protein-DNA reactions. The fold change of perceptgene output is set by the basal level, which in turn, determines the threshold of the activation function. Practically, we demonstrate our ability to fine-tune perceptgene biological parameters, including Hill coefficients, using well-known strategies for modifying gene regulatory networks (Supplementary Information, Section 8).

To implement the perceptgene in living cells, we first created a synthetic gene circuit that combines power-law and multiplication functions. The power-law functions compute the weighted inputs and the multiplication function aggregates these analog values (Fig. 1B). Our circuit inputs are small molecule inducers isopropyl β-D-1-thiogalactopyranoside (IPTG) and anhydrotetracycline (aTc), which bind LacI and TetR repressors, respectively. The repressors regulate their own production with auto-negative feedback loops via the P_{lacO} and P_{tetO} promoters (*29*). These auto-negative feedback loops implement the input's power-law functions and increase the dynamic range (*29*). To implement the multiplication function, we connected combinatorial promoter ($P_{lacO/tetO}$) (*55*) encoding LacI and TetR operators to the auto-negative feedback loops. The IPTG/aTc regulation of $P_{lacO/tetO}$ promoter via constitutively expressed LacI and TetR implements a conventional Boolean AND logic gate (*55*), but the regulatory topology described here, auto-negative feedback, converts the promoter's operation into a logarithmically-separable classifier (Eq. S2.13).

Experimentally, the IPTG/aTc transfer function has an input dynamic range of two orders of magnitude for each input (Fig. 1C). The output exhibits a power-law and multiplication function with respect to these inputs that spans more than two orders of magnitude (fig. S2.1), with the line defining the logarithmically separable classifier shown in Fig. 1C. Our minimized biochemical model reveals that the power-law coefficients are determined mainly by Hill coefficients

describing inducers binding to transcription factors and repressors binding to promoters (Eq. S2.17). Motivated by this analysis, we modified one of the auto-negative feedback loops by replacing P_{lacO} (which has two LacI binding sites) with P_{lacO1} (which has only one LacI binding site) (fig. S2.5b). The reduced cooperativity of P_{lacO1} resulted in a measured 50% increase in IPTG's power-law coefficient, i.e. its input weight (Fig. 1D, fig. S2.6). This result was due to a combination of interactions, including the impact of the negative feedback loops, different plasmid copy numbers, and the combinatorial promoter, as explained by our mathematical model (Fig.1, E and F, figs. S2.7 and S2.8). For a negative feedback loop, our model shows that the IPTG Hill coefficient is inversely proportional to the number of repressor binding sites in P_{lacO} or P_{lacO1} (Eq. S2.20.9). Experimentally, the behavior of this optimized power law and multiplication circuit remained stable over the course of approximately ten hours (figs. S9.1 and fig. S9.2).

Next, to implement the full perceptgene, we connected the power-law and multiplication function to a customizable activation function. Specifically, we selected AraC/P_{BAD} activation, and hence encoded AraC downstream of $P_{lacO/tetO}$, which in turn regulates perceptgene output via promoter P_{BAD} and arabinose inducer (Fig. 1G). A perceptgene with AraC/ P_{BAD} activation can be readily customized to perform different computational tasks by modifying various parameters, including arabinose levels (fig. S2.9b). Arabinose levels tune the AraC/P_{BAD} Hill-coefficient by converting the transcriptional repression function of free AraC to transcriptional activation of the AraC/arabinose complex (*56*). Experimentally, with low arabinose levels, the circuit converts an analog pattern of two inputs (IPTG and aTc) into a non-linear function that performs analog classification (Fig. 1H, fig. S2.10) and correlates well with a detailed biochemical model (Fig. 1I, fig. S2.11). We then analyzed the computational capabilities of this perceptgene and closely related variants. By extending existing neural computing analysis of perceptrons (*57*), we first proved that

the smooth minimum, maximum, and average functions can theoretically be encoded in the operation of perceptgenes using a log-transformed negative rectifier, log-transformed positive rectifier and log-transformed linear activation functions, respectively (Supplementary Information, Section 3, fig. S3.1).

In terms of the perceptgene activation function, for low arabinose induction levels, P_{BAD} promoter exhibits a log-transformed sigmoidal response (fig. S2.9d). Within our input dynamic range for P_{BAD}, we observe an approximately shifted and biased log-transformed negative rectifier activation function whereby $\log(AraC)$ concentrations lower than a high threshold (u_{01}) result in promoter expression proportional to $\log(AraC)$, while concentrations above this u_{01} threshold result in an asymptotically high output (fig. S3.1c). Numerically, the log-transformed negative rectifier function of P_{BAD} induced with low arabinose can be approximated as $2 \cdot [min(u_{01}, \log(AraC)) + f_{max}]$ when the $2 \cdot [u_{01} + f_{max}]$ is the highest output. In this design, the perceptgene input to this activation function which determines $\log(AraC)$ is a linear combination of weighted analog values for two input signals (IPTG, aTc), with weights n_3 and n_4, and maximum protein level ($AraC_{max}$) such the log transformation yields $\log(AraC) = n_3 \cdot \log(IPTG/IDR_3) + n_4 \cdot \log(aTc/IDR_4) + \log(AraC_{max})$ (BOX1, Supplementary Information, Section 1, Fig. 1D, fig. S2.6). Empirically, we observed that f_{max} depends on AraC and it can be estimated as $f_{max} = w_1 \cdot \log(aTc/IDR_4) + const$, when the log-transformed perceptgene output is presented as function of $\log(IPTG/IDR_3)$ (fig. S3.2a). Using the mathematical identities (Supplementary Information, Section 3)

$$\min\{u_{01}, x + y\} = \min\{u_{01} - y, x\} + y \quad (1)$$

the empirical f_{max} observation, and defining $x = n_3 \cdot \log(IPTG/IDR_3)$ and $y = (n_4 - w_1) \cdot \log(aTc/IDR_4) + \log(AraC_{max})$ for equation 1 above, we obtain that the perceptgene computes

$$\min\left\{\frac{u_{01}}{2}-\frac{\log(AraC_{max})}{2}-\frac{(n_4-w_1)}{2}\cdot\log\left(\frac{aTc}{IDR_4}\right),\frac{n_3}{2}\cdot\log\left(\frac{IPTG}{IDR_3}\right)\right\}+\frac{n_4}{2}\cdot\log\left(\frac{aTc}{IDR_4}\right)+\frac{const_1}{2}+\frac{\log(AraC_{max})}{2}-\frac{u_{01}}{2} \quad (2)$$

(fig. S3.2a). The term $\log(AraC_{max})-u_{01}$ is equal to the $\log(B_2)$, where B_2 is the bias (Eq. S3.3.3). The factor ½ that appears besides the mathematical terms in equation 2 is included because the P_{BAD} activity equals twice the negative rectifier (fig. S3.1c)**.** Hence, this perceptgene computes the minimum between two log-transformed analog numbers that are related to aTc and IPTG which is then offset by an analog value that is related to aTc only (fig. S3.2a). The smooth minimum computation between $\frac{n_3}{2}\cdot\log(IPTG/IDR_3)$ and $\frac{\log(B_2)}{2}-\frac{(n_4-w_1)}{2}\cdot\log(aTc/IDR_4)$ can be extracted by subtracting the normalized $\log(GFP)$ output signal by $0.5\cdot\log(aTc/IDR_4)-0.4$ since $n_4/2=0.5$, and $\frac{const_1}{2}+\frac{\log(B_2)}{2}=-0.4$. The experimental observations reveal that this smooth minimum computation exhibits a standard error of 10% (Fig. 1J, fig. S3.2, Table S3.8).

Next, we implemented a perceptgene encoding a shifted and biased log-transformed positive rectifier activation function using a modification of the AraC/P_{BAD} system (fig. S2.9d). In particular, P_{BAD} implements a positive rectifier function for high arabinose concentrations and low AraC levels (fig. S3.1c) that computes $2\cdot[\max(u_{02},\log(AraC))+f_{min}]$ with $2\cdot[u_0+f_{min}]$ signifying the lowest output. For this positive rectifier, $\log(AraC)$ concentrations lower than a u_{02} threshold result in a low constant P_{BAD} activity, while $\log(AraC)$ concentrations above this value elicit promoter expression proportional to $\log(AraC)$. Here, we control AraC with the Lux and Tet systems (Fig. 2, A to D), since we wanted to expand the set of regulatory elements that could be incorporated into multi-perceptgene networks. The inputs to this circuit are aTc and acyl

homoserine lactone (AHL). aTc induces expression as above, while AHL binds transcriptional activator LuxR, forms AHL-LuxR complex, and activates promoter P_{lux} (Fig. 2A).

We focused on finding the best AHL/aTc regulation that matches the input dynamic range of P_{BAD}/AraC for implementing the positive rectifier function. For AraC expression, we incorporated $P_{lux/tetO}$, a combinatorial promoter with LuxR and TetR operators (*55*). For TetR/aTc regulation, we used the same topology as above. To obtain a LuxR/AHL power-law response, we built a graded auto-positive feedback loop (*29*) using a weak P_{lux} mutant that broadens the input dynamic range (fig. S2.13). Similar to the analysis of auto-negative feedback, our biochemical model of auto-positive feedback revealed that the power-law coefficients are determined by the Hill coefficient and binding affinity of LuxR to promoter P_{lux} (Eq. S2.37). We built a library comprising seven different P_{lux} mutant by introducing random mutations to the LuxR operator, which alter LuxR binding affinity to P_{lux} (fig. S2.14, fig. S8.8). The mutant P_{luxTGT} achieved the best match with the input dynamic range of P_{BAD}/AraC under high arabinose levels. Experimentally, the measured AHL/aTc transfer function of $P_{lux/tetO}$ exhibits a power-law and multiplication output response with an input dynamic range for both inputs roughly of two orders of magnitude (Fig. 2B, fig. S2.15). This transfer function correlated well with our detailed biochemical model (Fig. 2C, fig. S2.16). To create the full perceptgene, we encoded AraC activator under $P_{lux/tetO}$, which in turn regulated the P_{BAD} promoter (Fig. 2D). Experimentally, with high arabinose levels, the perceptgene converts an analog pattern of two inputs (aTc and AHL) into a non-linear function (Fig. 2E, fig. S2.18).

Analogous to the minimum function analysis above, we use the mathematical identity (Supplementary Information, Section 3)

$$\max(u_{02}, x + y) = \max(u_{02} - y, x) + y \qquad (3)$$

We then define $x = n_6 \cdot \log(aTc/IDR_6)$ and $= (n_5 - w_2) \cdot \log(AHL/IDR_5) + \log(AraC_{max})$, and empirically observe that $f_{min} = w_2 \cdot \log(AHL/IDR_5) + const_2$. This yields a perceptgene that computes

$$\max\left(\frac{u_{02}}{2} - \frac{\log(AraC_{max})}{2} - \frac{(n_5 - w_2)}{2} \cdot \log\left(\frac{AHL}{IDR_5}\right), \frac{n_6}{2} \cdot \log\left(\frac{aTc}{IDR_6}\right)\right) + \frac{n_5}{2} \cdot \log\left(\frac{AHL}{IDR_5}\right) + \frac{const_2}{2} + \frac{\log(AraC_{max})}{2} - \frac{u_{02}}{2} \quad (4)$$

(fig. S3.3.2A). The term $\log(AraC_{max}) - u_{02}$ is equal to the $\log(B_3)$, where B_3 is the bias (Eq. S3.8). The factor ½ that appears beside the mathematical terms in equation 4 is included because the P_{BAD} activity equals twice the positive rectifier (fig. S3.1c). Hence, this perceptgene computes the maximum between two log-transformed analog numbers that are related to AHL and aTc, which is then offset by an analog value that is related to AHL only (fig. S3.3.2). The smooth maximum computation between $\frac{n_6}{2} \cdot \log(aTc/IDR_6)$ and $\frac{\log(B_3)}{2} - \frac{(n_5 - w_2)}{2} \cdot \log(AHL/IDR_5)$ can be extracted by subtracting the normalized $\log(GFP)$ output signal by $0.46 \cdot \log(AHL/IDR_5) - 0.1$ since $n_5/2 \approx 0.46$ and $\frac{const_2}{2} + \frac{\log(AraC_{max})}{2} - \frac{u_{02}}{2} = -0.1$. The experimental observations reveal that this smooth maximum computation exhibits a standard error of 23% (Fig. 2F, fig. S3.3.2C, and Table S3.9).

The third classification function that we implemented with a single perceptgene was a log-transformed average of two analog inputs IPTG and AHL offset by a constant bias (Fig. 2G). The IPTG and AHL inputs simultaneously regulate combinatorial promoter $P_{lux/lacO}$ (*55*) via graded auto-negative and auto-positive feedback loops. The average operation can theoretically be implemented using power-law and multiplication on $P_{lux/lacO}$ with weights of 0.5 for both inputs, without the need for an AraC non-linear activation function. However, we faced a challenge in reducing the IPTG and aTc weights of the combinatorial promoter to lower than one. Rather than

further reducing IPTG input weight directly, we designed the perceptgene's AraC activation function to be linear over the input dynamic range with very small slope (fig. S3.5). For the IPTG input, we used LacI/P_{lacO1} auto-negative feedback with an input weight of 1.9 (figs. S3.6 and S3.7). To ensure that AHL's weight matches the IPTG weight, we created a new mutant lux promoter P_{luxAAT} (fig. S2.14) and incorporated it into a graded positive feedback system. The resulting AHL input weight is 1.65, and closely matches that of IPTG (fig. S3.7). We then had to compensate for the high IPTG and AHL input weights by fine-tuning the AraC activation function to exhibit a sufficiently shallow slope in the log-log domain. First, we used very high arabinose concentrations to obtain a P_{BAD}/AraC activation function with a low Hill coefficient (fig. S2.9). Then, our mathematical analysis revealed that for AraC levels slightly higher than the binding dissociation constant of AraC to P_{BAD}, along with high arabinose concentrations, the activation function's slope is approximately 0.33 (Eq. S3.23.). Experimentally, with these AraC levels and a high arabinose concentration, the circuit indeed calculates the log-transformed average of AHL and IPTG (Fig. 2, H and I, fig. S3.8) offset by a normalized value of -¼ with a standard error of 9% (Table S3.10). Further analysis validating the smooth minimum, maximum, and average functions is provided in Supplementary Information, Section 4 (Table S4.2). As with the power law and multiplication circuit, the average circuit output also remained stable over the course of approximately ten hours (fig. S9.3). We also quantified the signal-to-noise ratio (SNR) for the three circuits based on single-cell measurements (Supplementary Information, Section 10). We observed that SNR for the power law and multiplication stage is reduced when replacing auto-negative feedback regulation with auto-positive feedback regulation, and that the addition of the activation function (AraC) tends to coalesce the SNR distributions of all three circuits to roughly the same values (fig. S10.1).

We then assembled combinations of the above perceptgenes into more complex circuits that implement higher-order functions using principles of deep ANNs, including feedforward networks (*40*). We first designed a two-layer perceptron network that implements a three-input majority function, whose output is "1" when two or more of its three inputs are "1" (Fig. 3A). Our simplified mathematical analysis showed that when considering input values of 0s and 1s, we can evaluate the design parameters of this perceptron network using a set of linear equations (Table 1). For this analysis, we use linear-domain perceptron activation functions that are approximated as piecewise linear with three regimes (constant low level when perceptgene input is lower than γ_L, linear as a function of input, and constant high level when perceptgene input is higher than γ_H). When the inputs to the first perceptron are both low, its output should be low enough such the second perceptron cannot be activated regardless of its input value (Table 1, row 1+2). Similarly, when both inputs to the first perceptron are high, its output should be high enough to activate the second perceptron regardless of its input value (Table 1, row 7+8). However, when only one of the first perceptron inputs is high, its output should be insufficient to activate the second perceptron by itself (Table 1, row 3+5) but high enough to jointly activate the second perceptron if its input is high (Table 1, row 4+6). A detailed explanation of the constraints on perceptgene input weights, biases, and activation function thresholds is included in the Supplementary Information, Table S5.1.

The mapping from a perceptron network design to a perceptgene network implementation of the majority function involves transformation to the logarithmic domain, as done above for the single perceptgene devices (Fig. 3, B and C, fig. S5.3d). The implementation comprises two cascaded perceptgenes and a GFP output (Fig. 3C). The perceptgene of the first layer of the cascade has AHL and IPTG inputs, a topology similar to the perceptgene in Fig. 2G, and T7 RNA

polymerase output. The second layer perceptgene inputs are T7 RNA polymerase from the first layer perceptgene output and the majority function's third input (aTc). T7 RNA polymerase is modified to include two amber stop codons, which normally block translation (*58*). Expression of amber suppressor tRNA supD, which is regulated by aTc, unblocks T7 RNA polymerase translation and activates T7 promoter (Supplementary Information, Section 5). T7 promoter activation of GFP signifies that a majority of the three inputs (AHL, IPTG, and aTc) is high.

Our choices of specific weights for implementing the majority function were guided by the simplified linear-domain analysis in Table 1, a conversion of this analysis into the log-domain (Table S5.2), and computational analysis of various log-domain tradeoffs (Supplementary Information, Section 5.1). These analyses essentially yielded the same constraints. We determined that the circuit can compute majority even with asymmetric weights for AHL and IPTG (fig. S5.3e, with error of 11%). For P_{lacO1}, the IPTG measured input weight was 1.8 (fig. S5.7). For wild type P_{lux}, the AHL input weight was 2 (fig. S8.10) but the input dynamic range was too narrow (less than one order of magnitude) for defining robust 0 and 1. Through random mutations, we found a mutant lux promoter P_{luxM56} which when incorporated into a graded positive feedback system (fig. S8.9) exhibited AHL input weight of 0.9 (fig. S5.7) and a sufficiently wide input dynamic range (larger than three orders of magnitude). This resulted in a design constraint for bias B_1 such that $\gamma_{L1}\text{-}min(0.9,1.8)<B_1<\gamma_{L1}$ (Table S5.1 caption). Empirically, B_1 is determined by the ratio between the maximal level of AraC and the binding dissociation constant of AraC to P_{BAD}. Hence, to satisfy the above constraint we fused an ssrA degradation tag (*59*) to AraC, resulting in a decrease in B_1 by two orders of magnitude from 0.15 to 0.0025 based on simulation results. For the constraint on the input dynamic range of the fist perceptgene activation function, we find that its level should be less than the combined total weight for the AHL and IPTG inputs $(\gamma_{H1}\text{-}\gamma_{L1}<n_1+n_2)$.

For the parameter constraints on the second perceptron, we find that the system has a solution only when the internal weight of T7 RNA polymerase is greater than the weight of P_{tetO}/aTc (Table S5.1, $m>n_3$), reflecting the intuition that T7 RNA polymerase represents the accumulated value of two inputs to the majority function. With TetR-based negative feedback regulation, the experimentally measured aTc input weight is 0.7 (fig. S5.8), while T7 RNA polymerase is a monomer activator with weight of 1. However, as our analysis shows, on the one hand the weight of aTc should be larger than the input dynamic range of the second perceptgene activation function ($n_3>\gamma_{H2}-\gamma_{L2}$), which is approximately equal to 1 (i.e. P_T promoter has a single binding site for T7 RNA polymerase). On the other hand, the weight of aTc should not exceed the weight of T7 RNA polymerase, satisfying the constraint $m>n_3$. Therefore, we incorporated dual SupD binding sites on T7 RNA polymerase (*58*), which increases aTc input weight by approximately 1.5 to reach a weight of 1.05 (fig. S5.3). Finally, to ensure that the bias and a single high input of the second perceptgene could not activate its output, we reduced the bias by incorporating a low affinity ribosome binding sequence for T7 RNA polymerase (Supplementary Information, Section 5).

After satisfying the various constraint-driven design decisions discussed above by using a high concentration of arabinose of 0.25 mM that supports near-saturating levels of P_{BAD} activation, the initial version of the majority function yielded a system where output for two of the eight AHL/IPTG/aTc input cases were incorrect (left side of Fig. 3D). To address this problem, we focused on improving the performance of the majority function by fine-tuning arabinose, a readily accessible method to alter mainly circuit weight but also bias, specifically modifying P_{BAD} response (fig. S2.9b). We measured the performance of the majority function under administration of a set of lower arabinose concentrations. For each individual arabinose level, we measured the

response to eight different AHL/IPTG/aTc combinations (Fig. 3D, fig. S5.9), which allowed us to compute the overall cost function (*60*) (i.e. error) for that P_{BAD}/AraC weight (Fig. 3E). Here, the cost function computes the logarithmically mean squared error ($\langle C \rangle = \frac{1}{2N}\sum_{i=1}^{N}\left(\log(\frac{z_{Di}}{z_i})\right)^2$) (Eq. S6.18) where N is the number of samples (also called the batch size), z_{Di} is the desired output for each state, and z_i is the observed output for these states.The best performance was observed for a lower level of arabinose (0.03125mM) and corresponding intermediate P_{BAD}/AraC weight (Fig. 3E, right side). The experimental results correlated well with our computational models of the majority (fig. S5.4) and cost functions (Fig. 3F, Eqs. S6.16-6.17).

Next, we studied whether our circuit's performance may potentially be optimized using a customized backpropagation learning algorithm based on gradient descent (*60*, *61*). Our backpropagation algorithm evaluates how incremental changes in weights for any perceptgene in the network affect overall system performance (i.e. cost function). This evaluation is used iteratively to determine how to update weights in a gradient descent fashion (Fig. 3G). In the first step of the algorithm, the outputs for each of the eight majority function input conditions for both the first layer perceptgene as well as the second layer (i.e. full circuit) are measured experimentally (fig. S6.5 and fig. S6.7). The eight output values for each perceptgene are normalized based on the highest level measured and basal expression (Table S6.1). Then, these experimentally derived output values are used in conjunction with a chain rule formula to determine the derivatives of the cost function with respect to changes in the two current weights. These derivatives yield suggestions for the next weights to test (Eq. S621-S622, Supplementary Information, Section 6.2). The algorithm cycles through the weights one at a time by evaluating the sign of the partial derivatives of the cost function $\left(\frac{\partial \mathrm{C}}{\partial \mathrm{n}}\right)$ with respect to this particular weight *n.* Based on the sign of the partial derivative, the algorithm updates the weight to the next nearest available value (*62*).

The algorithm repeats this process until either all output values reach their desired binary values or the cost function reaches a local minima.

Using this backpropagation algorithm, we followed the trajectories in a two-dimensional weight space with the P_{BAD}/AraC and P_{lux}/AHL weights. The P_{BAD}/AraC weight is chosen from a set of six different pre-selected Arabinose levels and the weight of P_{lux}/AHL is determined by selecting one of a small library of four genetic variants of the LuxR operator (fig. S5.7 and fig. S6.4). The four P_{lux} mutations exhibit different weights (0.25, 0.45, 0.65, 0.9) within the operating dynamic range of P_{lux}/AHL [0.1875-3μM]. We exhaustively measured the majority function response to eight different AHL/IPTG/aTc combinations across the six different arabinose concentrations and four P_{lux} mutations, in triplicates (a total of 8×6×4×3=576 samples). This allowed us to pre-compute an overall cost function for each P_{BAD}/AraC and Plux/LuxR weight combination (Fig. 3H). In emulating a backpropagation algorithm, we started at the corner of the weight space with the lowest values of P_{BAD}/AraC and P_{lux}/AHL weights and iteratively updated these weights. Based on the sign of the weight derivatives, the available higher or lower weight was chosen. The next weight value is either a P_{lux} genetic variant available in our pre-existing library or an arabinose inducer concentration from an *a priori* determined set of inducer values. The optimization trajectory culminated in a solution that provides the desired majority function binary output values after three iterations of P_{lux} and arabinose weight tuning, using information from 8×(3+3)=48 samples. Further experiments and analysis validating the backpropagation algorithm is provided in Supplementary Information, Section 6 (Table S6.2 and fig. S6.8).

For our final perceptgene network, we implemented an analog-to-digital converter (ADC; fig. S7.1), useful for a variety of intracellular and extracellular biosensing applications. The conversion of analog information into digital encoding is a classification problem that is efficiently solved by

ANN architectures (*63*) (Supplementary Information, Section 7.1). We evaluated three designs, starting from a perceptgene adaptation of a classical ADC perceptron network (*63*), a second design that adds two inhibitory regulatory links, and a third design that improves the fidelity of the digital output signals. In our first design (Fig. 4A), individual perceptgenes convert the analog input into digital bits starting from the most significant bit (MSB) to the least significant bit (LSB). While the analog input signal is routed to all perceptgenes, the computation from the MSB also contributes a negative weight to the LSB. This effectively subtracts the higher order bit from the LSB's analog input signal (figs. S7.2 and S7.3). We designed a gene network that uses transcriptional interference (*64*) to perform subtraction (fig. S7.5). We first checked whether P_{LUX} promoter activity can be subtracted from P_{BAD} promoter activity by arranging the promoters in a convergent orientation (*65*). However, experimental analysis of this subtraction method revealed that in addition to observing the desired increase in the input threshold required for output promoter activation, there was also undesirable repression of maximal promoter activation (fig. S7.5). Such repression can affect ADC performance for high input by displaying (1,0) instead of (1,1). To alleviate repression of maximal promoter activation caused by the transcriptional interference, we introduce a second regulatory element TetR, which represses a hybrid version of the convergent P_{lux} promoter ($P_{lux/tetO}$, fig. S7.8). In the revised ADC design, TetR is regulated by the MSB to nullify the subtraction of the MSB from the LSB only for high input levels.

The revised ADC gene circuit comprises four main elements: the AHL input stage, the MSB, the MSB subtractor, and the LSB (Fig. 4B and C). For the AHL input, graded positive feedback with LuxR increases the input's dynamic range, as above (fig. S2.13). To compute the MSB, AHL activates expression from P_{lux} encoded on a medium copy number. Computation of LSB involves AraC activation of P_{BAD} as well as down-regulation by transcriptional interference

(*64*) from convergent promoter $P_{lux/tetO}$, which is oriented in the opposite direction to P_{BAD}. This transcriptional interference implements subtraction of MSB from the LSB for intermediate levels of AHL, but not for high levels of AHL. This resulted in four distinct outputs states for the two-input ADC (green and red lines in Fig. 4D). An experiment using the initial circuit design in which P_{lux} promoter (without tet operator) replaces transcriptional interference promoter $P_{lux/tetO}$ shows low LSB levels under high AHL concentrations (purple line in Fig. 4D). Further experiments are provided in Supplementary Information, Section 7 (figs. S7.4 to S7.10).

To improve the accuracy of the LSB output for the (1,0) state, we revised the ADC design to compute LSB by including two separate perceptgenes (LSB_{low}, and LSB_{high}) with the same GFP output (figs. 4, E to F). This technique of aggregating circuit output from two promoters has been used successfully in previous synthetic gene circuit designs (*20*, *29*). The LSB_{low} perceptgene responsible for exhibiting high LSB output for the (0,1) state is configured similarly to the LSB perceptgene used in the previous ADC design. One difference is removal of TetR and use of P_{lux} instead of $P_{lux/tetO}$ for transcriptional interference in order to regulate LSB by MSB even for high AHL (figs. S7.7 and S7.8). The LSB_{high} perceptgene responsible for exhibiting high LSB output for the (1,1) case consists of promoter P_{rhl} that is directly regulated by AHL-LuxR but requires high levels of AHL for activation (fig. S7.13). This revised design yielded four distinct digital states with improved performance for the (1,0) state, albeit with inclusion of an undesirable narrow transitional state between (0,1) and (1,0) (Fig. 4G). The revised ADC maintained stable output for approximately ten hours (figs. S9.4). In comparison to previous synthetic data converters (*26*, *66*), our logarithmic domain analog-to-digital converter performs a complex computation that encodes the digital output value using multiple bits of information. The circuit in (*26*) implemented a digitizer, where one single analog input is converted to three discrete outputs, rather than a 2-bit

ADC. The circuits in (*66*) integrate multiple analog inputs and display multiple digitals outputs using logic gate design. In terms of efficiency, the computation in this study requires only two transcription factors.

Emerging applications in synthetic biology benefit from implementations that achieve complex regulatory functions using a minimal number of parts, based on designs that can be optimized with efficient algorithms. The existing paradigms of digital and analog biological circuits often fail in these regards. Instead, here we introduced the notion of genetic circuits that implement logarithmic domain ANNs, with an approach that supports flexible choices of weights and biases for circuit optimization (Supplementary Information, Section 8). Logarithmic operation, which is based on fold-change regulation rather than absolute changes, is suitable for biological information sensing and transduction in both natural and synthetic systems (*29*, *54*, *67*) (Supplementary Information, Section 11).

To gain further insight into the benefits of performing log-domain perceptgene computing, we compared the perceptgene simulated behavior to a linear-domain perceptron that uses a biologically relevant Michaelis-Menten activation function instead of a sigmoid function (fig. S1.3b). First, the analysis showed that the perceptgene successfully classified an analog signal into low and high levels, whereas the modified perceptron failed (fig. S1.5), implying that logarithmic domain computing is more suitable than linear domain computing for our neuromorphic gene circuits. Second, our theoretical sensitivity analysis also showed that logarithmic domain neuromorphic computing is robust to noise at low signal concentrations, whereas the performance of analog circuits is often poor for such signals (fig. S1.10). Third, both the perceptgene and perceptron designs support only a limited number of distinct inputs. While the perceptgene is mainly limited by the ability to integrate multiple analog signals in regulating gene expression, the

perceptron is limited by the intrinsic noise generated during signal aggregation (Table S1.1, Supplementary Information, Section 1.5). Given these constraints, an implementation of a linear-domain perceptron in living cells may nonetheless be practical for applications that focus on a single layer with multiple inputs of highly expressed analog signals such as cell-free systems (*68*) and microbial consortia (*69*). The observed transfer functions obtain high sharpness in the case of cell-free systems via substrate saturation and in the case of the microbial consortia via multicellular positive feedback using secreted small molecule inducers that bind transcription factors. However, while a few examples exist, engineering a large library of synthetic biology devices that exhibit sharp responses when operating inside individual cells remains a challenge.

The design principles of logarithmic domain ANNs exhibit collective resilient properties, offer efficient parallel execution of complex functions, require a small number of components to produce required results, and provide scalability for deep networks. These properties enable construction of systems that can be readily adapted to customized functions by supervised optimization algorithms (*41*, *47*). We began to explore these properties experimentally by creating several neuro-inspired genetic circuits that encode minimum, maximum, and average functions, each comprising a single perceptron with analog outputs. The minimum and maximum functions are widely used in fuzzy logic computing to implement conjunction, disjunction, implication, equivalence, and negation (*70*) (fig. S4.7). For biological systems, these functions can, for example, be useful for situations that require graded expression levels that are precisely determined by multiple inputs, as opposed to simple ON/OFF expression. The minimum operation may improve safety and efficacy of genetic circuits for cancer immunotherapy because it enables recognition of cancer biomarkers in a manner reminiscent of an AND logic gate (*71*, *72*) But in contrast to AND logic, the minimum operation will activate the immunotherapy at levels

proportional to the biomarkers detected, which may reduce undesirable effects such as cytokine storms. The average function is useful for engineering biological systems able to tolerate noise and compensate for distortion of biological signals.

Our neural gene circuits may also provide an alternative framework for building logic functions that reduce usage of cellular resources. For example, the implementation of our three-bit majority function requires 15 biological parts (i.e. promoters and genes), in comparison to 22 parts used for the state-of-the-art three-bit majority implementation based on the digital abstraction (*2*). The design of a two-bit ADC in mixed-signal computation required three logic stages (*73*) and approximately ten biological parts (*26*), whereas our neuro-inspired two-bit ADC in which the MSB regulates the LSB requires only two transcription factors. In terms of a relevant but more complex computation, theoretical analysis of a 2-bit Full Adder implementation comparing the neuromorphic versus digital approaches is provided in Supplementary Information, Section 8 (fig. S8.12). Besides minimizing circuit sizes, our perceptgene networks also operate with low expression levels, mainly in order to maintain low bias levels. For instance, our circuits' proteins are expressed at low levels either via usage of weak ribosome binding sites, weak RNA polymerase binding sites, or by fusion with ssrA degradation tags. In contrast, digital systems often attempt to operate with significant noise margins, and hence high expression levels for ON values. Further discussion on the benefits of neuromorphic versus digital circuit design is provided in Supplementary Information, Section 12.

Another important property of ANNs is the ability to efficiently fine-tune and even repurpose their function, e.g. by changing weights and biases via a gradient descent algorithm. For example, by increasing the bias of a perceptgene from low to high, its computation can be modified from OR to AND (fig. S7.15). As another example, the ADC circuit can be easily reconfigured to

function as an AHL-induced ternary switch with distinct low, medium, and high output states (Supplementary Information, Section 7.4). Analysis of the model suggested this can be achieved by enhancing activation of the LSB perceptgene (fig. S7.16). Accordingly, we experimentally administered high levels of arabinose, and observed the desired ternary response to AHL (Fig. 4H). As yet another example, we analyzed the behavior and topology of the majority circuit and experimentally demonstrated that replacement of promoter P_{tetO} with combinatorial promoter $P_{luxM56/tet}$ yields a new logic function "AHL OR [IPTG AND aTc]" (fig. S7.18). Furthermore, the neuro-inspired design of the three-input majority circuit allowed us to optimize the error by changing the weights of P_{BAD}/AraC and AHL/P_{lux} in similar manner to backpropagation algorithm. Supplementary Information, Section 8 provides experimental demonstration of modulating weights and biases using a variety of additional biological mechanisms including transcription factor sequestration, steric hinderance, and operator sequences.

Our theoretical and experimental study shows that perceptgene networks can utilize a broad range of biological regulatory mechanisms (Supplementary Information, Section 8), and accordingly, we expect to be able to implement these networks with other modalities of biological regulation (e.g., protein-protein interactions (*74*, *75*) and RNA devices (*76*, *77*)) and in different organisms. Flexibility in implementation approaches will help employ such neural networks to address a wide range of industrial, diagnostic, and therapeutic applications (Supplementary Information, Section 13). For example, bioengineers could use analog-to-digital converters for selecting which specific combinations of several genes to express based on administration of a single inducer. Also, with a single inducer, bioengineers could harness the ternary switch to select between one of multiple expression levels for a given gene in a robust and noise resistant fashion. These circuits could also form the basis for engineering more sophisticated cellular biosensing,

e.g. multi-bit classifiers with higher precision than their digital counterparts and increased robustness relative to their analog counterparts. These regulation and sensing capabilities are valuable both in biomanufacturing and therapeutic contexts.

We anticipate that the new framework described here constitutes a first step towards implementing supervised machine learning optimization algorithms in individual living cells. Future efforts will include codifying the design principles of neuromorphic gene networks such as development of effective mechanisms to combine coarse-grain and fine-grain control over weights and biases (Supplementary Information, Section 8). ANNs are also compatible with the digital and analog computing platforms. One can leverage the specific advantages of each of these three platforms in a synergistic fashion to create an efficient, accurate, and scalable hybrid approach for robust genetic engineering of living cells (fig. S13). Future efforts will also focus on developing computer-aided design tools that combine ANN design principles with linear equations and logic gates (e.g., Cello (*2*)) to automate biological engineering. Analogous to the manner by which natural signaling pathways combine a variety of regulatory modalities, we anticipate that future synthetic gene networks integrate the different approaches mentioned above in ways that are particularly suitable for the biological substrate.

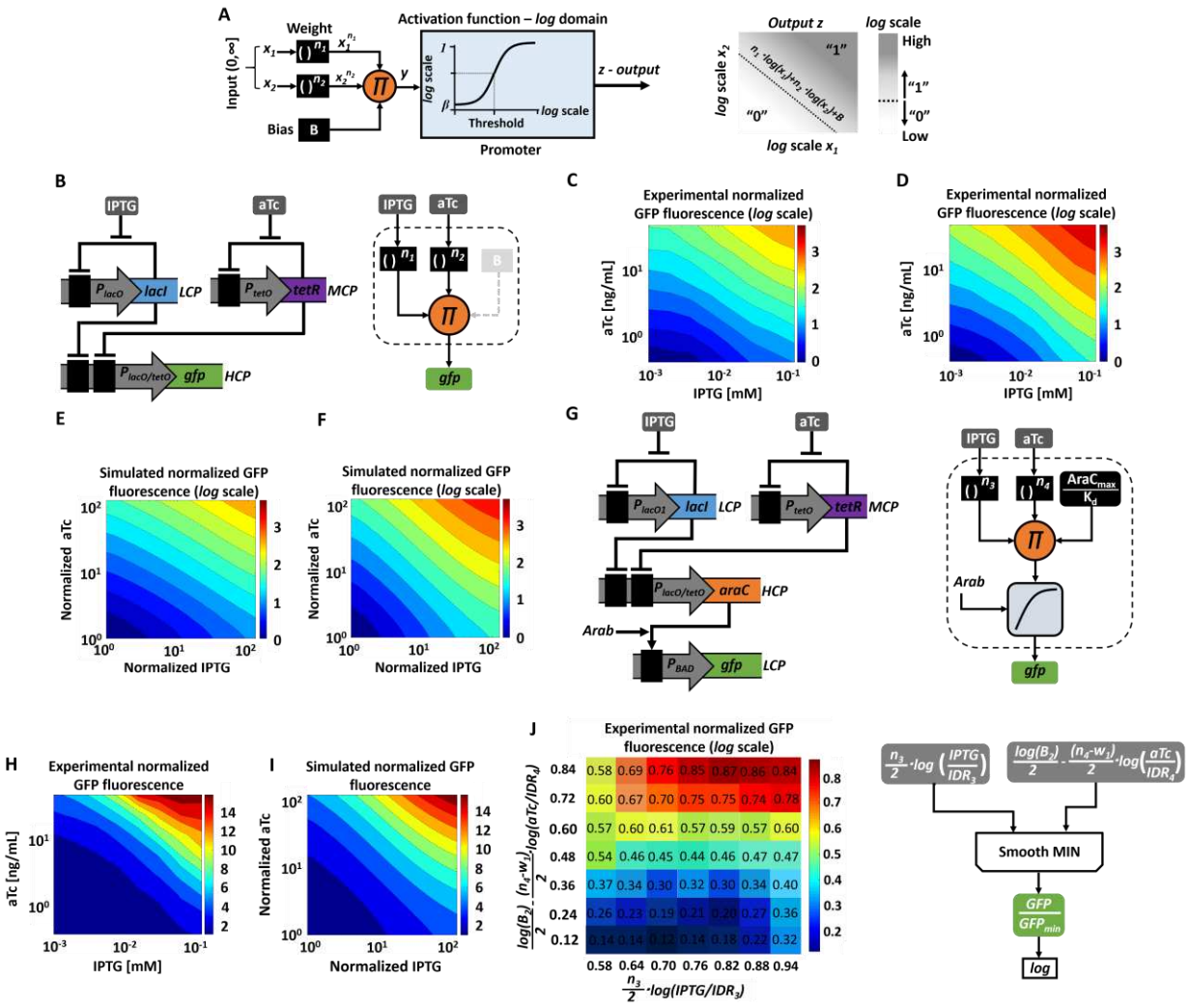

Fig. 1. Theory and implementation of the perceptgene. **(A)** Mathematical model of the perceptgene. The perceptgene has multiple analog inputs with positive values and (I) raises each input x_i to the power of its corresponding weight n_i, (II) multiplies these power-law products to obtain $y = B \cdot \prod x_i^{n_i}$ where B is a bias that shifts the computation of y into the desired range, and (III) applies to the product y a sigmoid activation function that operates in the logarithmic domain $\left(z = \frac{\beta + e^{\ln(y)}}{1+\beta+e^{\ln(y)}}\right)$, with $z \in [\beta, 1]$, where β is a minimum (*i.e.*, basal) level. Depicted on the right is the resulting logarithmically separable classification of analog inputs x_1 and x_2 (Supplementary information, BOX 1). **(B)** The power-law and multiplication circuit for inputs isopropyl β-D-1-thiogalactopyranoside (IPTG) and anhydrotetracycline (aTc). Combinatorial promoter ($P_{lacO/tetO}$) is encoded on a high-copy-number plasmid (HCP) and is regulated by LacI and TetR repressors.

The P_{lacO} promoter, encoded on a low-copy-number plasmid (LCP), is regulated by LacI via an auto-negative feedback loop and induced by IPTG. The P_{tetO} promoter, encoded on a medium-copy-number plasmid (MCP), is regulated by TetR through an auto-negative feedback loop and induced by aTc. Depicted on the right is a block diagram for the genetic circuit operation; IPTG and aTc inputs are raised to their respective powers n_1 and n_2 and multiplied. The bias B is introduced later. **(C)** The experimentally measured transfer function shows output green fluorescent protein (GFP) as a function of IPTG and aTc inputs at steady state. We considered input values for IPTG and aTc only above their dissociation constant. **(D)** Measured transfer function for a modified circuit where P_{lacO} within the auto-negative feedback loop was replaced by P_{lacO1}, a promoter that consists of only one LacI binding site. **(E-F)** Computed transfer functions based on detailed biochemical models of power-law and multiplication circuits with Placo and PlacO1, respectively. IPTG and aTc levels are normalized by their dissociation constants of IPTG-LacI binding and aTc-TetR binding, respectively. **(G)** A perceptgene was implemented by adding the bias and the activation function. We replaced GFP output of the power-law and multiplication function with AraC activator, which in turn regulates the P_{BAD} promoter. AraC is the input and GFP is the output for the activation function. An ssrA degradation tag (LAA) was added to AraC to reduce the maximum protein level. Depicted on the right is a block diagram for the genetic circuit operation. The bias is computed with $AraC_{max}$=Transcription rate × Translation rate × mRNA half-life × AraC half-life, and K_d as the dissociation constant of AraC binding P_{BAD}. Arabinose controls the threshold and slope of the activation function. **(H-I)** Measured and simulated transfer functions of the perceptgene with arabinose concentration of 0.04 mM. **(J)** A log-transformed and normalized smooth minimum computation derived from the experimental results using the function depicted to the right of the graph. The numerical values in the graph provide help in terms of careful analysis of the computation of minimum. $n_3=0.75$, $n_4=1$, $w_1=0.3$, $log(B_2)/2=-1$, $const_1/2=0.6$, $log(IDR_3)=2.1$, $log(IDR_3)=2.1$, where IDR_3 and IDR_4 are the input dynamic ranges of IPTG and aTc, respectively. The lowest IPTG and aTc levels are not depicted as part of the minimum function because the fluorescence output for these basal expression levels is not a reliable measure of actual protein expression. All experimental data represent the average of three experiments. Output data is normalized by the measured minimum.

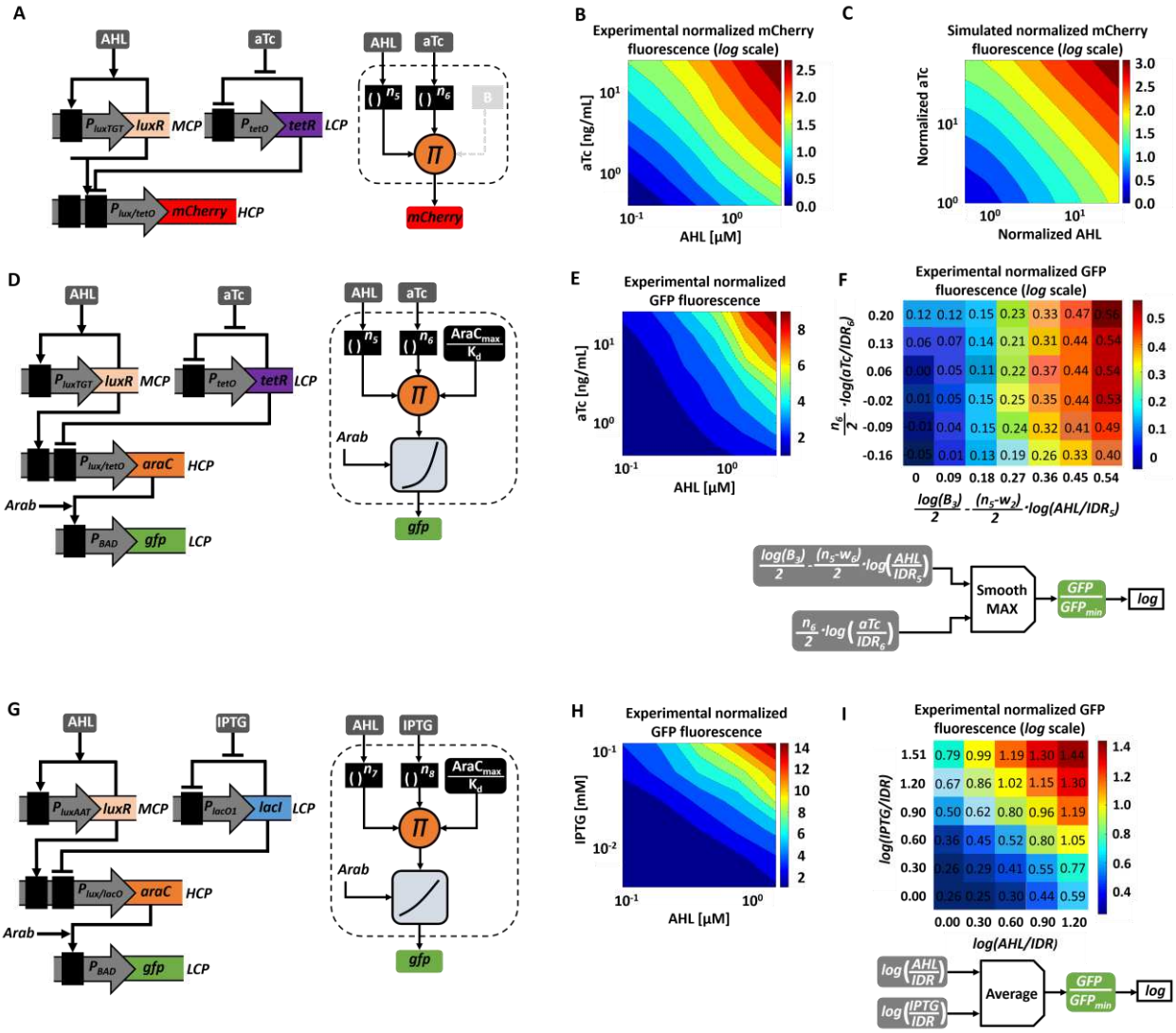

Fig. 2. Perceptgenes based on auto-negative and auto-positive feedback loops. **(A)** The power-law and multiplication function circuit for inputs acyl-homoserine-lactone (AHL) and aTc. Mutant P_{luxTGT} promoter, encoded on an MCP, is regulated by LuxR activator through an auto-positive feedback loop and is induced by AHL. The P_{tetO} promoter, encoded on an LCP, is regulated by TetR through an auto-negative feedback loop and is induced by aTc. A combinatorial promoter $P_{lux/tetO}$, encoded on an HCP, is regulated by LuxR and TetR with output mCherry. Depicted on the right is a block diagram for the genetic circuit operation with inputs AHL and aTc raised to their respective powers n_5 and n_6. The bias computation is introduced later. **(B)** Measured AHL/aTc transfer function. We considered input values for AHL and aTc only above their dissociation constant. **(C)** Simulated AHL/aTc transfer function based on a detailed biochemical model. The IPTG and AHL are normalized by their dissociation constants of IPTG-LacI binding

and AHL-LuxR binding, respectively. **(D)** Perceptgene with P_{BAD}/AraC activation. An ssrA degradation tag (LAA) was added to AraC to reduce the maximum protein level. **(E)** Measured transfer function of the perceptgene circuit with AHL and aTc analog inputs. Arabinose concentration is 0.5 mM**. (F)** A log-transformed and normalized smooth maximum computation perceptgene circuit derived from the experimental results using the function depicted below the graph. The numerical values in the graph provide help in terms of careful analysis of the computation of smooth maximum. $n_5=1$, $n_6=0.8$, $w_2=0.22$, $log(B_3)/2=-0.2$, $const_2/2=0.1$, $log(IDR_5)=1.5$, $log(IDR_6)=1.8$, where IDR_5 and IDR_6 are the input dynamic ranges of AHL and aTc, respectively. **(G)** Perceptgene for computing a log-transformed average function of AHL and IPTG inputs. Most regulatory elements are previously described, except for mutant P_{luxAAT} promoter, which is encoded on an MCP and regulated by LuxR through an auto-positive feedback loop. **(H)** Measured transfer function of the perceptgene circuit with AHL and IPTG analog inputs. Arabinose concentration is 0.5mM. **(I)** A log-transformed and normalized smooth average computation perceptgene circuit derived from the experimental results using the function depicted below the graph. $log(IDR_{IPTG})=1.5$, $log(IDR_{aTc})=1.2$. All experimental data represent the average of three experiments. Output data is normalized by the measured minimum.

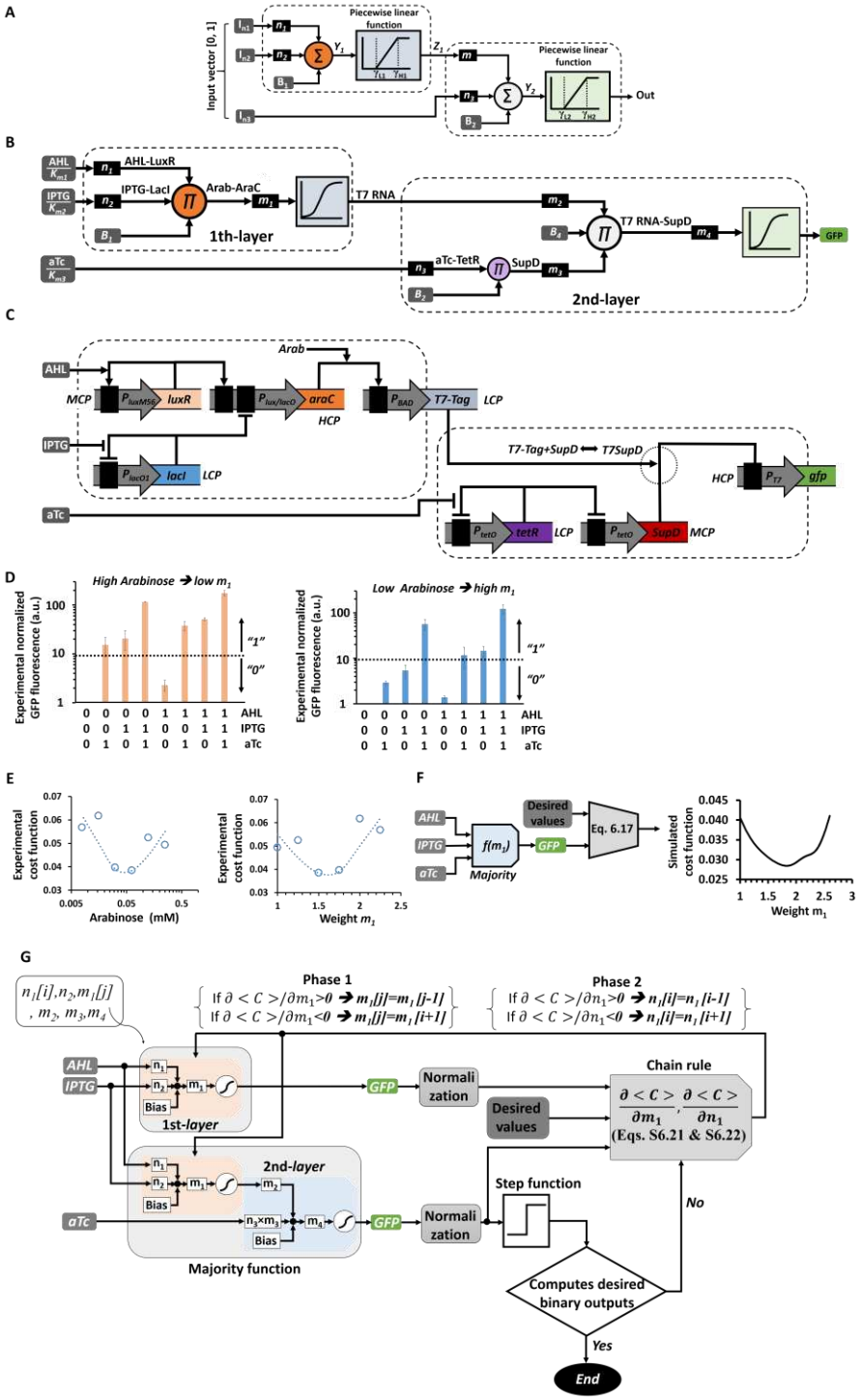
A
Input vector [0, 1]
Piecewise linear function
Piecewise linear function
Out
B
AHL-LuxR
IPTG-LacI
Arab-AraC
1th-layer
T7 RNA
T7 RNA-SupD
aTc-TetR
SupD
2nd-layer
GFP
C
Arab
AHL
MCP
luxR
araC
HCP
T7-Tag
LCP
IPTG
lacI
LCP
T7-Tag+SupD ⟷ T7SupD
HCP
gfp
aTc
tetR
LCP
SupD
MCP
D
High Arabinose → low m1
Low Arabinose → high m1
Experimental normalized GFP fluorescence (a.u.)
"1"
"0"
AHL
IPTG
aTc
E
Experimental cost function
Arabinose (mM)
Weight m1
F
AHL
IPTG
aTc
f(m1)
Majority
Desired values
GFP
Eq. 6.17
Simulated cost function
Weight m1
G
Phase 1
If ∂<C>/∂m1>0 → m1[j]=m1[j-1]
If ∂<C>/∂m1<0 → m1[j]=m1[j+1]
Phase 2
If ∂<C>/∂n1>0 → n1[i]=n1[i-1]
If ∂<C>/∂n1<0 → n1[i]=n1[i+1]
AHL
IPTG
Bias
1st-layer
GFP
Normalization
Chain rule
Desired values
(Eqs. S6.21 & S6.22)
2nd-layer
aTc
Majority function
Step function
No
Computes desired binary outputs
Yes
End
H
Experimental cost function
Weight m1
Weight n1

Fig. 3. Three input majority function: multilayer perceptgene network and backpropagation algorithm. **(A)** The design of a majority function with two-layer cascaded perceptrons (dashed boxes). **(B-C)** Conversion of the biophysical model of the majority function into a two-layer perceptgene network with inputs AHL, IPTG, and aTc. Network operation is determined by (1) weights (m_i, n_i), which are derived from Hill coefficients, (2) biases, which are proportional to the ratio between maximum protein expression and promoter binding affinity, and (3) activation functions, which are described by promoter activity. The inputs to the first layer perceptgene are AHL and IPTG and output T7 RNA polymerase is regulated by P_{BAD} promoter. The second layer perceptgene inputs are T7 RNA polymerase and aTc. Multiplication of these two inputs is achieved via expression of aTc-regulated SupD and the binding reaction $T7_{RNA} + supD \leftrightarrow T7_{RNA}SupD$. This complex activates the T7 promoter and expresses GFP. To achieve the required bias levels, AraC is fused with ssrA degradation tag (LAA) and encoded on an LCP, while T7 RNA polymerase is regulated by a low binding affinity ribosome-binding sequence (RBS; Supplementary Information, Section 5). **(D)** Measured response of the majority gene circuit for all eight low/high combinations of the three inputs: AHL [0.1875, 0.3μM], IPTG [7.8125, 125μM] and aTc [1.5625, 25ng/mL]. High arabinose = 0.25mM and low arabinose = 0.03125mM. **(E)** The cost function for the operation of the majority circuit under various arabinose levels was estimated using experimental results based on fig. S5.9 and Eq. S6.18, shown here both as a function of arabinose and its corresponding weight. Dashed lines provide a visual guide. Each cost function was calculated using averaged output of three experiments. **(F)** The simulated majority circuit cost function was estimated using a logarithmic backpropagation algorithm, as shown in in fig.S6.3 and Eq. S6.17. **(G)** Backpropagation algorithm for two weights, based on measuring the first and second layer perceptgenes and then computing partial derivatives for the weights by applying the chain rule using equations S6.21 and S6.22. **(H)** Experimental cost function for two-dimensional weight space with the weights of P_{BAD}/AraC and P_{lux}/AHL. The P_{BAD}/AraC weight is regulated by six different pre-selected Arabinose levels (0.25mM, 0.125mM, 0.062mM, 0.031mM, 0.015mM and 0.007mM). The weight of P_{lux}/AHL is determined by selecting one of a small library of four genetic variants of the LuxR operator (TCTA, GTTG, GAGC, and TGGG). The backpropagation algorithm path starts from the point with the lowest weights, alternates in modifying one of the two weights to improve the cost function, and ultimately converges to a local minimum. All experimental data represent the average of three experiments.

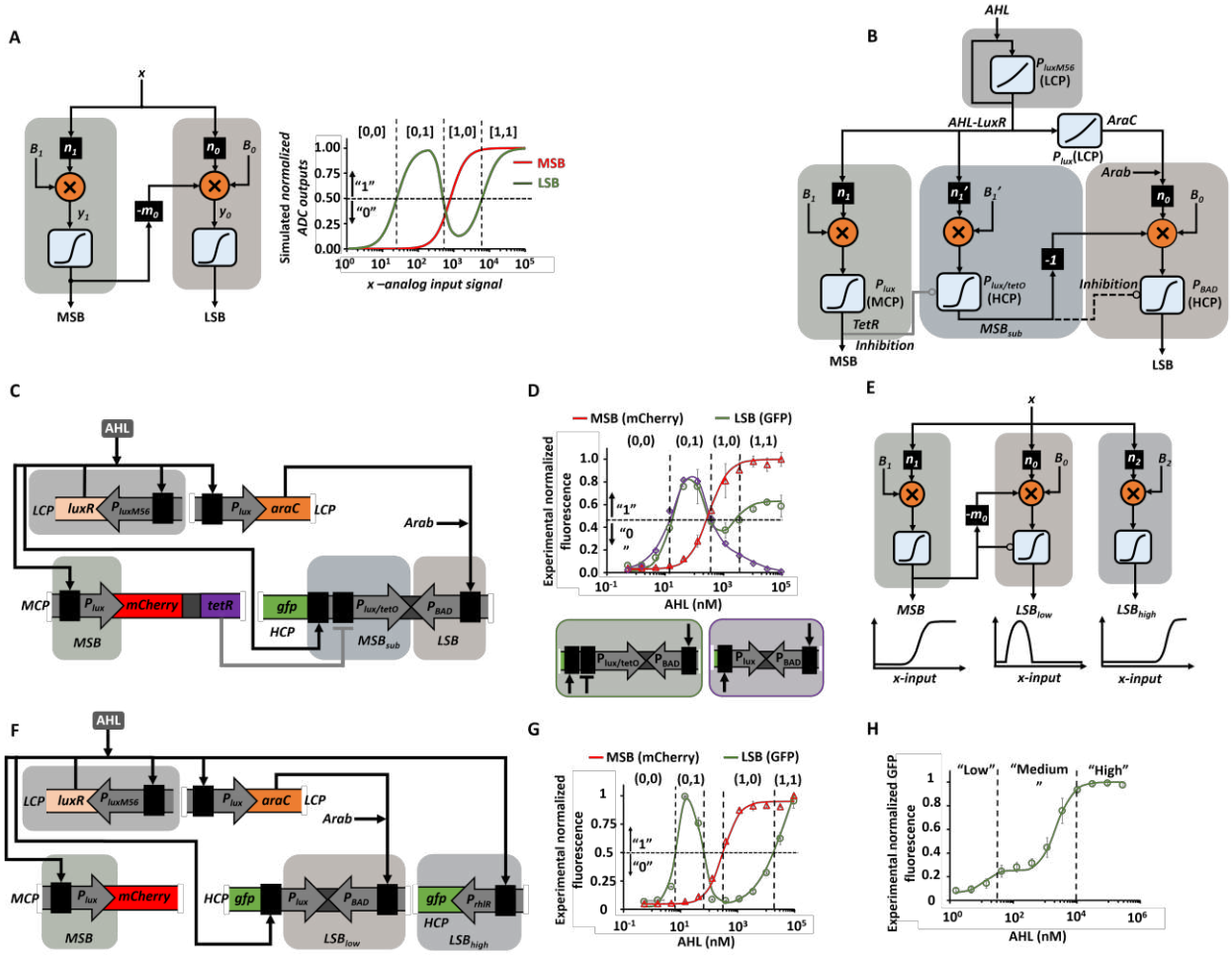

Fig. 4. Genetically encoded data converters based on neural network principles. **(A)** High level circuit design and simulation results of a two-bit analog-to-digital converter (ADC) using the perceptgene model. The architecture is based on transforming abstract design rules of a neural-network ADC (perceptron-based) from the linear (Eqs. S7.13–S7.15) to the logarithmic domain (Eqs. S7.30–S7.32, fig. S7.3). The most significant bit (MSB) and the least significant bit (LSB) are the ADC outputs. (**B**) The modified perceptgene design of 2-bit ADC as suggested by analyzing models of gene networks. More sophisticated regulation is required, including subtraction of the MSB from the computation of LSB using transcriptional interference (denoted by "-1") and inhibition. **(C)** Genetically encoded implementation of a two-bit ADC. The circuit converts AHL input concentration into two digital bits (LSB and MSB). Positive feedback regulation of AHL input via mutant P_{luxM56} promoter linearizes the response. The MSB, encoded on an MCP, is computed by a perceptgene comprising the P_{lux} promoter, which regulates mCherry and TetR. The LSB, encoded on an HCP, receives the linearized AHL input, which activates P_{BAD} promoter via

AraC and expresses GFP. The MSB subtractor ($MSB_{sub,}$ encoded on an HCP) regulates P_{BAD} promoter via transcriptional interference ($P_{lux/tetO}$, oriented in the opposite direction to P_{BAD}). **(D)** Measured and computationally simulated (Eqs. S7.44 and S7.47) responses of the LSB and MSB (arabinose=0.4mM). Red triangles show the measured MSB mCherry signal. Green circles show the LSB GFP signal with $P_{lux/tetO}$ promoter, purple diamonds show the GFP signal with P_{lux} (arabinose=0.4mM). The $P_{lux/tetO}$ and P_{lux} configurations of the convergent promoters are shown below the graph. **(E)** Design of a 2-bit ADC using three perceptgenes, two of which are used for computing the LSB. Perceptgene LSB_{low} is activated under low AHL concentrations, and inhibited for medium and high AHL concentrations. Perceptgene LSB_{high} is activated only under high AHL concentrations. The MSB perceptgene has a topology similar to the MSB in Fig. 4B. **(F)** Genetic implementation of the 2-bit ADC from Fig. 4E. In comparison to Fig. 4C, we removed TetR from the MSB perceptgene, replaced $P_{lux/tetO}$ with P_{lux} to obtain LSB_{low}, and implemented LSB_{high} using P_{rhl} promoter that is activated by AHL-LuxR but requires high levels of AHL. **(G)** Measured and computationally simulated response of the 2-bit ADC from Fig. 4F (arabinose=0.06mM). **(H)** Measured and computationally simulated response of the ternary switch (arabinose=0.4mM). All experimental data represent the average of three experiments, and is denoted by various data point markers and std. dev. Computational simulations are depicted by lines. Output data is normalized by the measured minimum.

Table 1: Truth table of the linear-domain perceptron-based 3-input majority function and evaluation of constraints on the design parameters. B_1 and B_2 are the biases of the first layer and second layer perceptgenes, respectively. The three input weights are n_1, n_2, and n_3, while m is the weight of the first layer perceptegene output (Z_1) that serves as an input to the second layer perceptgene. γ_{L1}, γ_{L2} and γ_{H1} and γ_{H2} are the low and high thresholds of the piecewise-linear first and the second activation functions f_{A1} and f_{A2}. The γ_{H2}- γ_{L2}, and γ_{H1}- γ_{L1} are defined as the input dynamic ranges of the activation functions.

I_{n1} I_{n2} I_{n3}	Out	Y_1	Z_1	Y_2	Constraints on Design parameters
0 0 0	0	Design constraints subsumed by 001 case			
0 0 1	0	B_1	0	B_2+n_3	$B_1 < \gamma_{L1}$ $B_2+n_3 < \gamma_{L2}$
0 1 0	0	B_1+n_2	$0 \le f_{A1}(B_1+n_2) < 1$	$B_2+m \times f_{A1}(B_1+n_2)$	$B_2+m \times f_{A1}(B_1+n_2) < \gamma_{L2}$
0 1 1	1	B_1+n_2	$0 < f_{A1}(B_1+n_2) \le 1$	$B_2+n_3+m \times f_{A1}(B_1+n_2)$	$B_1+n_2 > \gamma_{L1}$ $B_2+n_3+m \times f_{A1}(B_1+n_2) > \gamma_{H2}$
1 0 0	0	B_1+n_1	$0 \le f_{A1}(B_1+n_1) < 1$	$B_2+m \times f_{A1}(B_1+n_1)$	$B_2+m \times f_{A1}(B_1+n_1) < \gamma_{L2}$
1 0 1	1	B_1+n_1	$0 < f_{A1}(B_1+n_1) \le 1$	$B_2+n_3+m \times f_{A1}(B_1+n_1)$	$B_1+n_1 > \gamma_{L1}$ $B_2+n_3+m \times f_{A1}(B_1+n_1) > \gamma_{H2}$
1 1 0	1	$B_1+n_1+n_2$	1	B_2+m	$B_1+n_1+n_2 > \gamma_{H1}$ $B_2+m > \gamma_{22}$
1 1 1	1	Design constraints subsumed by 110, 101, 011 cases			

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- SupplementaryinformationNCB.pdf