# How to Make a Synthetic Multicellular Computer

**Javier Macia**[1,2]*, **Ricard Sole**[1,2,3]*

1 ICREA-Complex Systems Lab, Universitat Pompeu Fabra, Barcelona, Spain, 2 Institut de Biologia Evolutiva, UPF-CSIC, Barcelona, Spain, 3 Santa Fe Institute, Santa Fe, New Mexico, United States of America

## Abstract

Biological systems perform computations at multiple scales and they do so in a robust way. Engineering metaphors have often been used in order to provide a rationale for modeling cellular and molecular computing networks and as the basis for their synthetic design. However, a major constraint in this mapping between electronic and wet computational circuits is the wiring problem. Although wires are identical within electronic devices, they must be different when using synthetic biology designs. Moreover, in most cases the designed molecular systems cannot be reused for other functions. A new approximation allows us to simplify the problem by using synthetic cellular consortia where the output of the computation is distributed over multiple engineered cells. By evolving circuits in silico, we can obtain the minimal sets of Boolean units required to solve the given problem at the lowest cost using cellular consortia. Our analysis reveals that the basic set of logic units is typically non-standard. Among the most common units, the so called inverted IMPLIES (N-Implies) appears to be one of the most important elements along with the NOT and AND functions. Although NOR and NAND gates are widely used in electronics, evolved circuits based on combinations of these gates are rare, thus suggesting that the strategy of combining the same basic logic gates might be inappropriate in order to easily implement synthetic computational constructs. The implications for future synthetic designs, the general view of synthetic biology as a standard engineering domain, as well as potencial drawbacks are outlined.

**Citation:** Macia J, Sole R (2014) How to Make a Synthetic Multicellular Computer. PLoS ONE 9(2): e81248. doi:10.1371/journal.pone.0081248

**Editor:** Mark Isalan, Imperial College London, United Kingdom

**Received** July 12, 2013; **Accepted** October 10, 2013; **Published** February 19, 2014

**Funding:** This work has been supported by and EU ERC Advanced Grant, the James S. McDonnell Foundation, the Fundaci'on Botin and the Santa Fe Institute. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

**Competing Interests:** I have read the journal's policy and have the following conflicts: co-author Ricard Solé is a PLOS ONE Editorial Board member. This does not alter the authors' adherence to all the PLOS ONE policies on sharing data and materials.

* E-mail: javier.macia@upf.edu (JM); ricard.sole@upf.edu (RS)

## Introduction

A fundamental trait of biological systems is their capacity to perform computations [1]. Although cells are composed of molecules and their viability relies on extracting and using energy to maintain them, they are not "just" matter and energy. Information, and how it is processed and used, is an essential ingredient of biology. Adaptation to environmental signals requires the processing and proper output to incoming information. This is of no surprise when we consider that life is strongly tied to genetic information [2]. Similarly, a computational picture of biological systems is at the core of important, unanswered questions on how organisms behave [3].

How do biological systems compute? Computation is present at multiple scales, from molecules to collective decisions [4–10]. Developmental processes [11,12], collective intelligence [13,14] and complex decision-making in cells [15–18] can be mapped into some class of formal computational framework. Early works in theoretical biology, cybernetics, and Boolean dynamical systems widely emphasized the view of molecular phenomena within cells as the likely result of computational processes [19–22]. But beyond the classical theoretical approach to computation, one especially important avenue involves the engineering of cellular circuits in order to construct given computational functions or devices performing computations [23–31]. An example is given in figure 1, where we show the potential implementation of a simple logic gate using engineered regulatory networks. Here a NAND gate is built by combining a few basic components. Two input molecules ($\alpha$ and $\beta$) can be sensed by appropriate receptors or simply diffuse into the cell where they interact with operator sites. Only in the absence of both signals the output is produced.

In this context, it has been suggested that complex computational tasks might be obtained by engineering biological structures (molecules and cells) in such a way that they can respond to given sets of inputs and generate a pre-defined output response. Using synthetic biology techniques, a great deal of examples involving logic gates and simple combinations of them have been obtained and some specific computational problems addressed (see [24,32,33] and references cited). Much is expected from these developments towards new approaches to complex diseases, for example. But the promise of a reliable, scalable, reusable, robust and predictable life-based technology that could allow constructing complex living machines has been shown to be much more limited than expected [34]. After a successful first wave of important results, the promise of arbitrarily complex constructs obtained in a LEGO-like fashion is far from achieved. A flexible toolbox of reusable elements is yet to be developed and all synthetic designs so far devised are lmited to specific tasks and can not be applied to other problems. In particular, the combinatorial potential implicit in standard circuit engineering has not yet been explored.

Although it is known from the basic theory of combinatorial circuits that some particular logic gates (such as the NOR or NAND) can be used to build any conceivable circuit, this extrapolation has failed to succeed when applied to synthetic biological designs. In that sense, although several authors have been able to build such special gates and claimed that they could *in*

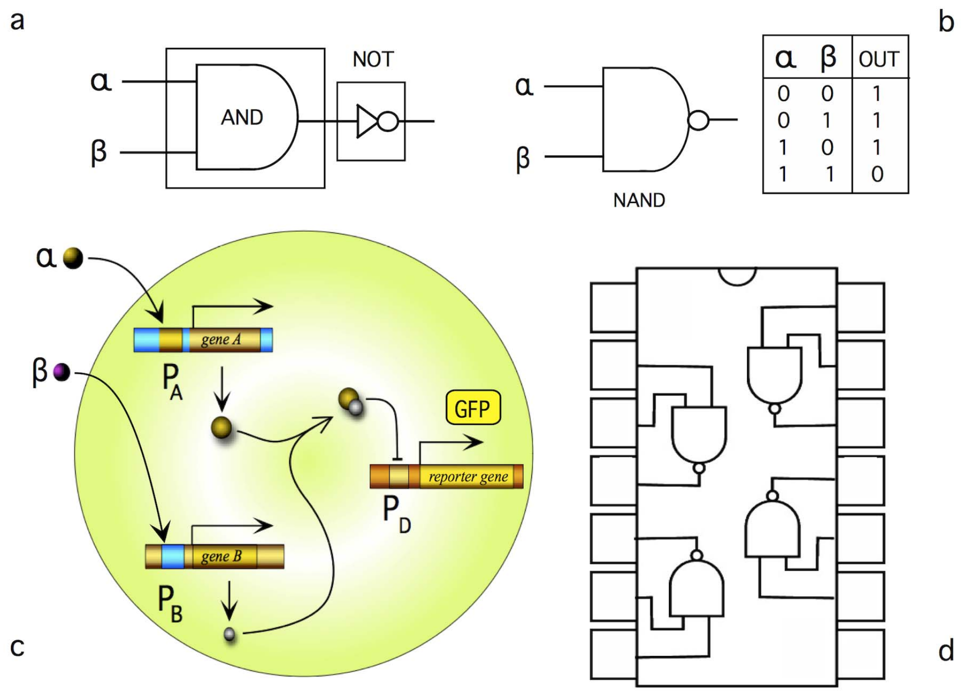
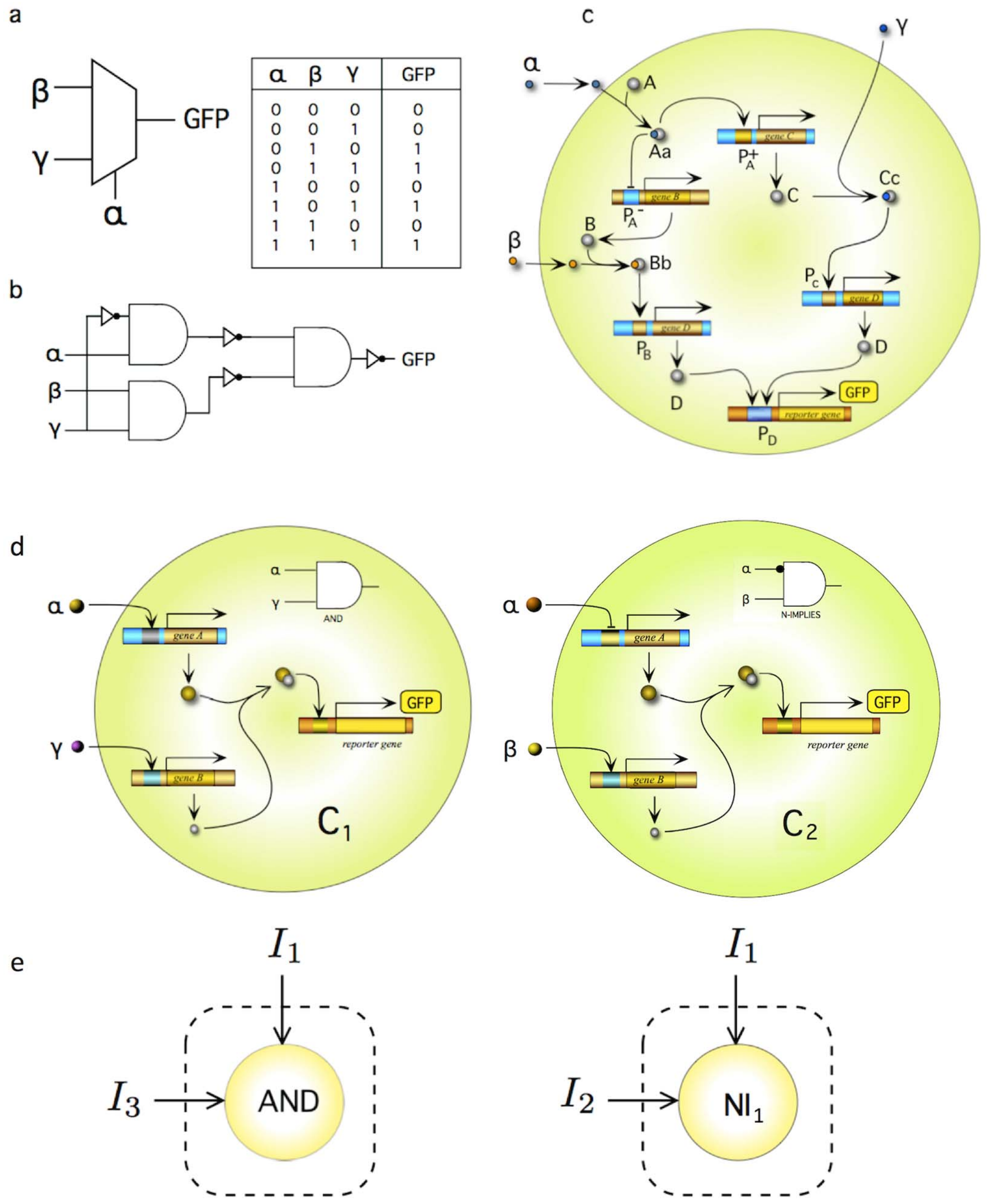


**Figure 1. Simple logic gates can be implemented out from minimal sets of logic units.** The NAND gate (a) is obtained as a sequential combination of AND and NOT gates. The compressed symbol is shown in (b) along with the truth table. An example of a synthetic implementation of the NAND logic can be made (c) using genetic regulatory elements A and B forming a regulatory heterodimer complex that prevents the expression of the reporter gene. In conventional electronics, combinations of such gates allow to construct more complex circuits and chips (d), which are then used as basic modules for further circuit designs.
doi:10.1371/journal.pone.0081248.g001

*principle* implement any potential, complex cellular circuit, the extra engineering required makes that claim far fetched. More specifically, the idea of constructing complex devices by just combining logic gates in a standard manner fails due to the so-called *wiring problem*. In contrast to electronics, where all wires are identical but physically isolated, in a cellular context each connection must be implemented by a different biochemical element, e.g. proteins. Even relatively simple devices, such is a MUX circuit (figure 2a), are difficult to obtain [35]. This circuit involves three inputs, one of which ($\alpha$) is the so called *selector signal*. As can be seen, the state of this selector element determines which one of the two inputs ($\beta$ and $\gamma$) is "chosen" as the final output. Its use is widespread in electronics and it is part of many different applications. In figure 2b we show a standard implementation of this circuit obtained by connecting several NAND gates (here shown as AND+NOT elements). Despite its simplicity, this circuit requires a considerable engineering effort in order to follow standard circuit design principles within a single cell. For illustrative purposes we show a possible implementation of this circuit in figure 2c. We can easily appreciate that the internal logic of our proposed circuit requires several promoters to be connected through different molecular "wires". Such limitations pose immediate constraints to the possibility of creating robust, scalable, and flexible devices with higher computational capacities. Hence, the development of decision-making circuits performing complex functions and, in general, the path towards living computers seems compromised by the failure of standard design principles.

One way of approaching the problem of complex wiring is to use a cellular consortium, where different types of engineered cells are at work. Such cellular consortia are common in nature [37–39] and provide a more flexible scenario for building complex synthetic circuits [23,40–43]. Once a library of engineered cells has been constructed, it is possible to combine them in different ways to obtain different circuits. This is illustrated by the work of Tamsir et al. [44] where the authors used a set of NOR-like gates constructed on *Escherichia coli* by using two specific promoters, where the inputs and outputs are controlled by expression of different quorum-sensing molecules without cross-talk among the different cells. Here, as in conventional circuits, the colonies were spatially distributed on agar plates and connected through quorum-sensing wires. A reporter colony is used to indicate the final output response. By arranging the colonies in different spatial configurations, all of the elementary two-input logic gates can be implemented. This example provides an instance of the application of standard engineering principles to synthetic multicellular systems. However, because of its construction, it is once again limited in complexity, scalability, and flexibility.

In [45,46] we proposed a very different approach, which we named *distributed multicellular computation* (DMC). Under this approach, circuits are also divided into different cell types, but the similarities with standard electronics ends here. Roughly speaking, we allow circuits to be broken into pieces with the component indicating output scattered over the different pieces. Several types of reporter cells can be present and do not need to be connected. Upon this assumptions, it was shown [46] that complex circuits can be built from very simple cellular consortia. Each cell requires only a small amount of engineering and additional cell-cell communication molecules can be used (but are not always needed) to exchange information among cells. Multicellular implementation is conceptually appealing. It conceals the implementation details of each encapsulated logic gate, which can be individually designed and optimized. As such, it can facilitate circuit implementation and reduce interference with the host cell's physiology by minimizing the number of components introduced



| α | β | γ | GFP |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

**Figure 2. Standard and non-standard circuit design.** Combinatorial circuits are constructed in conventional electronic design by using predefined gates and wiring them in order to execute a given input-output table. This is illustrated by the so called multiplexer (MUX) circuit, whose representation and logic table is shown in (a). Using AND and NOT gates, a standard implementation is displayed in (b). In (c) we show an example of a synthetic gene network implementing a single-cell multiplexer. The output signal is a GFP reporter. A very different design of the MUX system is shown in (d). Here the circuit can be easily designed by splitting the computation into two separated *and disconnected* engineered cells, both able to display the output signal. A simplified diagram that summarized the logic of (d) is shown in (e).
doi:10.1371/journal.pone.0081248.g002

into each cell strain. Therefore, to assemble a complex multicellular circuit, the experimenter needs to be concerned with only two factors: the input-output function of each cellular gate and the output-input matching between layers. Moreover, another benefit of multicellular computing is that it allows the suppression of noise in each layer. Because the wiring-molecule output from each gate is mixed, and represents the sum over a population, spurious or 'noisy' responses within a small proportion of cells can be filtered out in subsequent layers [47].

Instead of using NAND gates, the reduction of wiring requirements was achieved by combining the standard AND and NOT gates with the non-standard logic gates called inverted implies (N-implies, NI). There are two possible NI logic gates (hereafter indicated as $NI_1$ and $NI_2$) defined in tables 1 and 2 which describe a Boolean function where the underlying circuit decides which of two bits is (strictly) larger than the other. In that case the first table would correspond to $I_1 < I_2$ whereas the second defines $I_1 > I_2$. Despite these gates not being commonly used in standard designs, they have a clear biological meaning found in

**Table 1.** Truth table for a $NI_1$ Boolean function.

| $I_1$ | $I_2$ | *GFP* |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

doi:10.1371/journal.pone.0081248.t001

many regulatory genetic networks, i.e. one of the inputs triggers the expression of an output gene whereas the other blocks this expression.

Additionally, under this new approach, the gates (NOT, AND and NI) are connected in a new way. Whereas in electronics design rules try to minimize the number of gates but do not address to the number of wires or the complexity of the resulting network of connections, here the number of wires is minimized and the network of connections is reduced to a simple, fixed topology. This was experimentally implemented using engineered yeast cells, which allowed building a library of cell types that could be combined in multiple ways in order to create different types of combinatorial circuits [45,46].

As a example, in figure 2d we present an alternative implementation for the MUX circuit based in DMC with *distributed output production*. As we can see, the two-cell implementation is made by using two *disconnected* elements, *both* being able to express the reporter gene. In electronics, it would mean breaking the circuit into two pieces; each one having a light bulb to indicate the output. Even though it makes little sense for electronic engineering, it solves a problem when dealing with a living computational device.

Considering these previous results, the goal of this paper is two-fold. On the one had we want to present a general picture of optimized circuits implementing arbitrary Boolean functions based on the combination of DMC and *distributed output production*. Such a picture would be helpful in guiding the choice and development of components and wires in synthetic constructs. On the other hand, we also want to see how far the analogies made between standard electronic circuits and their cellular counterparts can be stretched.

The previous choices make the potential set of designed circuits simple by construction. As shown below, searching for circuit designs compatible with our proposal leads to simple solutions. Such solutions largely combine a subset of logic gates that departs from the standard designs in several ways. Previous work has obtained optimal solutions in different systems [48] by using evolutionary algorithms, and the networks which evolved were simpler in some ways to hand-designed synthetic biology networks. Similarly, here we perform our analysis by means of an evolutionary algorithm exploring the space of possible designs for logic circuits. It is important to mention that our approach implicitly assumes that the underlying engineering associated with these multicellular consortia is not affected by a number of relevant problems, including cross-talk, noise, and population dynamics. Some of these problems were addressed in [45,46] where it was shown that the DMC approach is able to overcome some of these problems. The analog, noisy, and population-dependent extensions of the work presented here will be explored elsewhere.

**Table 2.** Truth table for a $NI_2$ Boolean function.

| $I_1$ | $I_2$ | *GFP* |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

doi:10.1371/journal.pone.0081248.t002

## Materials and Methods

### 0.1 Boolean Models of Cellular Computation

The simplest theoretical framework to define biological computation is a Boolean approximation. In such framework, the set of possible states to be observed is limited to two, i. e. $\Sigma = \{0,1\}$. A given input string I made of zeros and ones can be written as an element of

$$\Sigma^N = \underbrace{\{0,1\} \times \ldots \times \{0,1\}}_{n} \qquad (1)$$

Such an input string would correspond to a set of present or absent input signals, which can be molecules but also physical variables. Let us focus on a given Boolean function $\Phi$ involving $N$ inputs and one output. This is formally indicated as a mapping

$$\Phi : \Sigma^N \rightarrow \Sigma^1 \qquad (2)$$

This function defines a input-output mapping between any given binary string of $N$ bits $\mathbf{I} \in \{0,1\}^N$ and the two possible output values $\{0,1\}$. Complex circuits can be obtained out from a combination of several smaller sub-circuits called *logic gates*, and this can be done in multiple ways. These logic gates are two particularly relevant subsets of Boolean functions, namely i) the 16 two-input one-output logic gates defining the set $\mathcal{G}^{(2,1)} = \{g_{ij}^k\}$ where $g_{ij}^k$ represents the *k-st* gate responding to inputs $i$ and $j$, and ii) the one-input one-output gates, i. e. the set $\mathcal{G}^{(1,1)} = \{g_{i0}^1, g_{i0}^2\}$ where $g_{i0}^1$ is the negation (NOT) and $g_{i0}^2$ is the identity function (Id) in response to i *-st* input. It is well known that multiple subsets of logic gates can be used to implement *any* possible combinatorial circuit. These are known as *functionally complete sets* (FCSs). Typical examples of these sets are the pairs {AND,NOT} and {OR,-NOT} but since the NAND and NOR gates are obtained from the combination of these previous pairs, it actually occurs that the single-function sets {NAND} and {NOR} are themselves FCSs. This statement can be proven [49] using the rules of Boolean algebra.

As mentioned above, the single-cell implementation, although possible in principle, has two drawbacks. The first involves the unavoidable design problem associated with the use of several molecular wires associated with each gene-gene connection. The second is the limited flexibility of a single-purpose design. Most typical designs cannot be recycled in any way, but a flexible and scalable system should allow for combination among components such that multiple functions could be implemented. We must keep in mind, as we have pointed out before, that the standard methods for circuit design focus on the minimization of the number of logic gates but do not pay attention to the number of wires or to the complexity of the pattern of connections. This becomes a

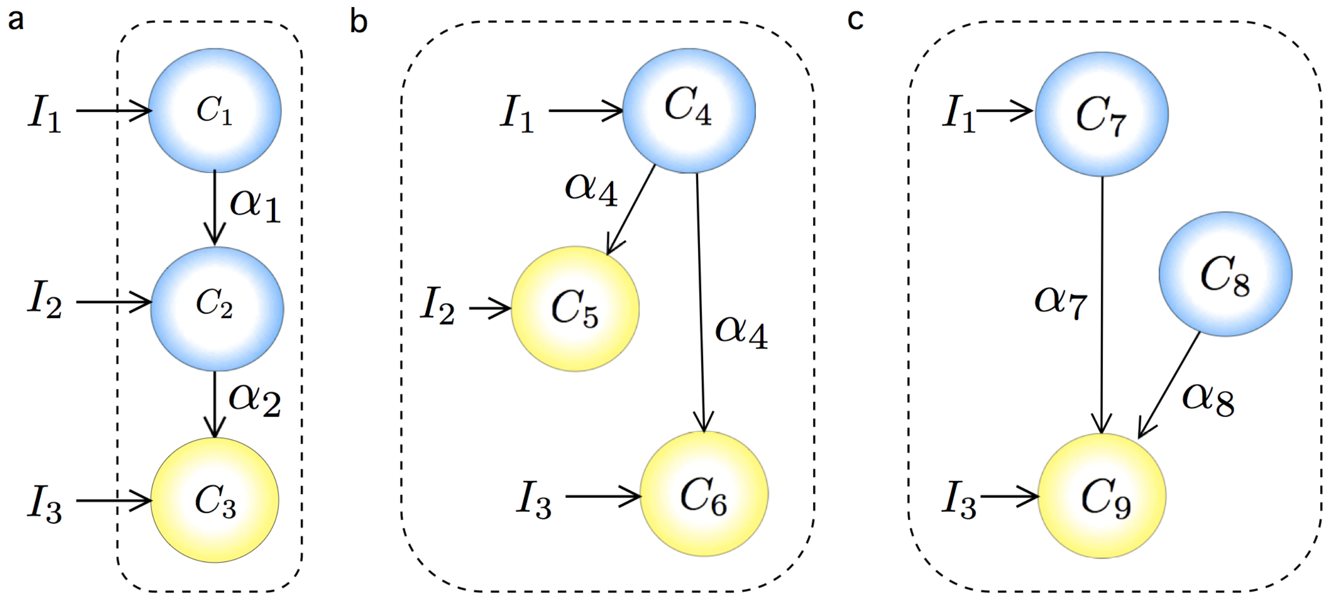


**Figure 3. A general circuit design can be obtained by starting from a multicellular system where each virtual cell is a given logic gate.** Here each engineered cell is indicated as $C_q$ and wires generated by $C_q$ are indicated as $\alpha_q$. Cell $C_q$ will produce an output according with the logic defined by $g_{ij}^k$. Here $g_{ij}^k$ represents the $k$-th logic gate responding to two inputs, the external one $I_i$ and the internal wire $\alpha_j$ secreted by cell $C_j$. The upper layer involves single-input gates, i.e. $g_{i0}^k$ (thus only the identity or NOT are possible). Different motifs of connections can emerge according with the criteria introduced, such as independent strings of connected cells, where each cell responds to different wire (a), the same wire can be sensed by more than one cell (b), or a given cell responds to wires produced in more than one cell (c). In this last case, due to cells only can sense two inputs (one external and one internal) all wires produced in different parts of the circuit but sensed by the same cell must be implemented using the same diffusible molecule, i.e. wire $\alpha_7$ and $\alpha_8$ are implemented by the same molecule, which can be produced in cells $C_7$ and $C_8$ independently. This situation corresponds to an implicit implementation in $C_9$ of the OR logic with respect to wires $\alpha_7$ and $\alpha_8$. Yellow cells ($C_3$, $C_5$, $C_6$ and $C_9$) can produce independently the final output signal e.g. a GFP.
doi:10.1371/journal.pone.0081248.g003

limitation for the application of standard rules for cellular circuit design, and hence a novel methodology for cellular circuit designs seems necessary.

To reach this goal, we propose a different approach according to the following criteria.

1. the circuit can be distributed in a network of different engineered cell types (*distributed computation*),
2. the output production can take place in differently cell types simultaneously (*distributed output production*),
3. the set of *wires* connecting the different cell types must be minimal, and
4. the pattern of connections between cells must be as simple as possible.

Distributed computation allows for a minimization of circuit complexity, since each cell carries a small amount of engineering, limited to implementing one given logic gate. Logic gates (cells) generate an output responding to an external input or at the most to an external input and to the output produced by other different cell, i.e. a wire, according to the logic of one of the possible functions from the set $\mathcal{G}^{(1,1)} \cup \mathcal{G}^{(2,1)}$. This output can be the final output of the circuit or a new wire. Of note, this assumption implies a significant difference with respect to standard methodologies because there are no hidden gates, i.e. gates responding only to internal signals (wires), typically present in standard circuits. Every feed-forward circuit implemented by our system is based on three basic motifs, shown in Figure 3. These motifs involve engineered cells (colour balls) and links $\alpha_i$ connecting them. By splitting different parts of the circuitry over different cells, we can take advantage of the intrinsic modularity of cells as units. Furthermore, distributed output production allows for a strong relaxation of wiring requirements. The connections between different cell types can be implemented by producing small diffusible molecules that can be secreted by a given cell type and sensed by another cell type. Our analysis is centred on the minimal scenario for combinational digital circuits, where feedback connections are not allowed.

## 0.2 Evolving Distributed Circuits

The efficient design of synthetic biocomputers faces a complex optimization problem. As the number of potential elements grows with circuit complexity, so does the potential number of solutions. Such combinatorial explosion can be managed by using automated methods of design [50]. This has been done in some special cases, most of which consider the analog nature of genetic systems. They include small memory devices (flip-flops, [51]), pulses and bandwidth detectors [52], and are generalized to different scenarios through simulated annealing [53], non-linear programming methods [54] standard growth and selection procedures [55], evolutionary optimization of a set of independent circuits [56], and in silico automated design inspired by standard minimization techniques borrowed from electronic design [57]. All of these methods exhibit advantages and limitations, and the predicted circuits, especially when dealing with a large number of biological parts, involve complex wiring diagrams and are not expected to be reused for multiple functions.

In this paper we show the potential of DMC for evolving complex decision-making circuits that would be very difficult to implement using inspiration from standard electronics. Along with

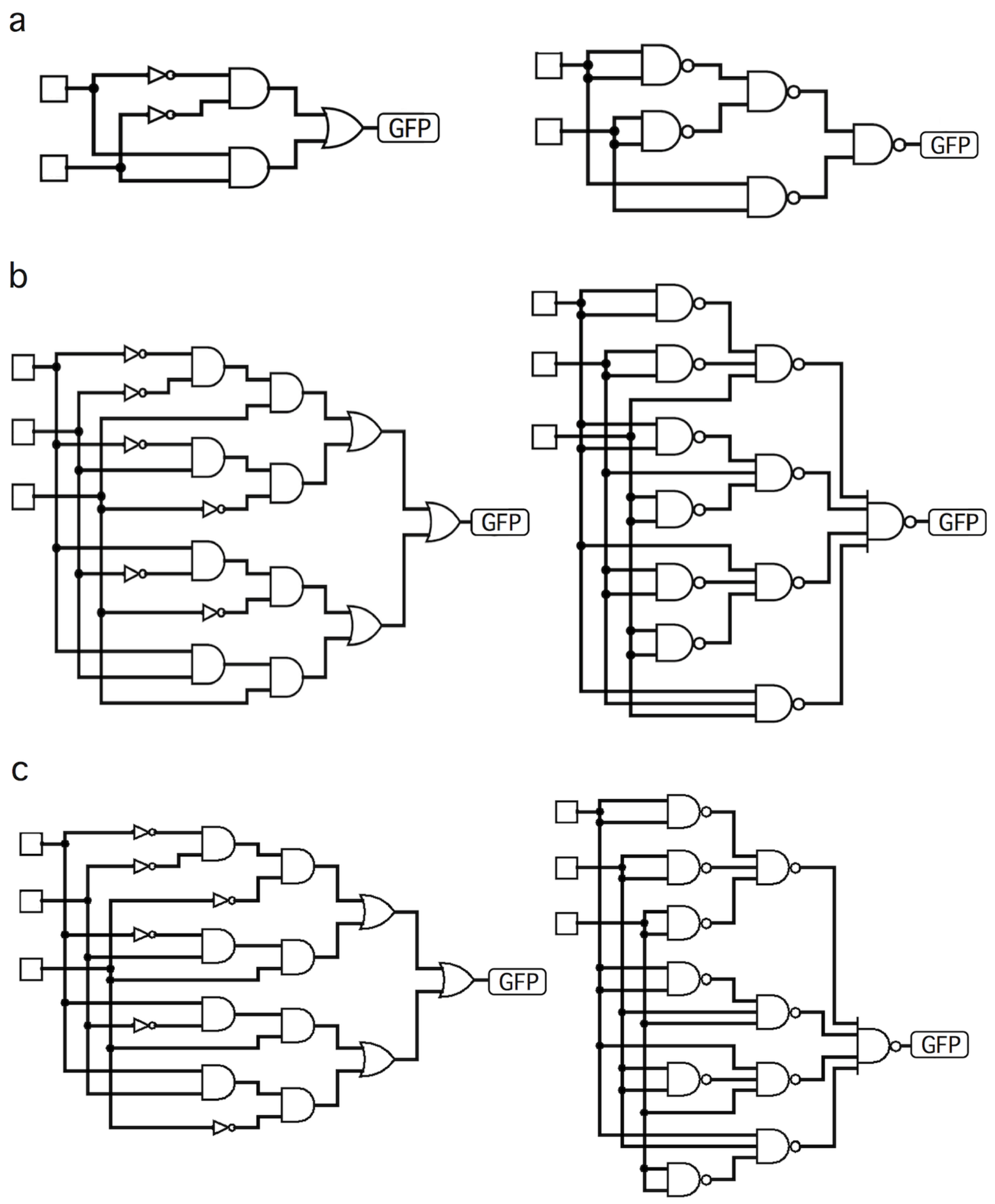


**Figure 4. Examples of standard engineering designs of three cases studies.** (a) Two-bit comparator (b) the three-bit adder and (c) a 3-bit parity circuit. Here the set of inputs appears indicated as open squares and the single-output element is marked as GFP. The left and right columns are different implementations of the same circuits design (Boolean table) but they have been constructed used diverse logic gates (left) or only using NAND gates with variable numbers of inputs. These circuits have been generated using the *Logisim* software package. Once the truth table is provided, it builds the logic circuits, either choosing the appropriate set of two-input logic gates or using just NAND gates. If the NAND gates were chosen such that they only include two inputs, the circuits would be much more complex.
doi:10.1371/journal.pone.0081248.g004

**Table 3.** Truth table for a Two-bits magnitude comparator.

| $I_1$ | $I_2$ | *GFP* |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

doi:10.1371/journal.pone.0081248.t003

the previously mentioned MUX circuit, we show the result of evolving five different standard circuits of increasing complexity. These are a binary comparator, a three-bit adder, the 3-bit parity circuit (see description below), and two 4-input circuits (see below). In order to compare the expected networks resulting from electronic design principles with our (much less complex) proposed DMC constructs, in figures 4a, b, c we show their traditional implementation using logic gates. The left and right columns correspond to the implemented wiring diagrams using different one- and two-input logic gates and NAND gates with arbitrary inputs, respectively. We can clearly appreciate how rapidly the

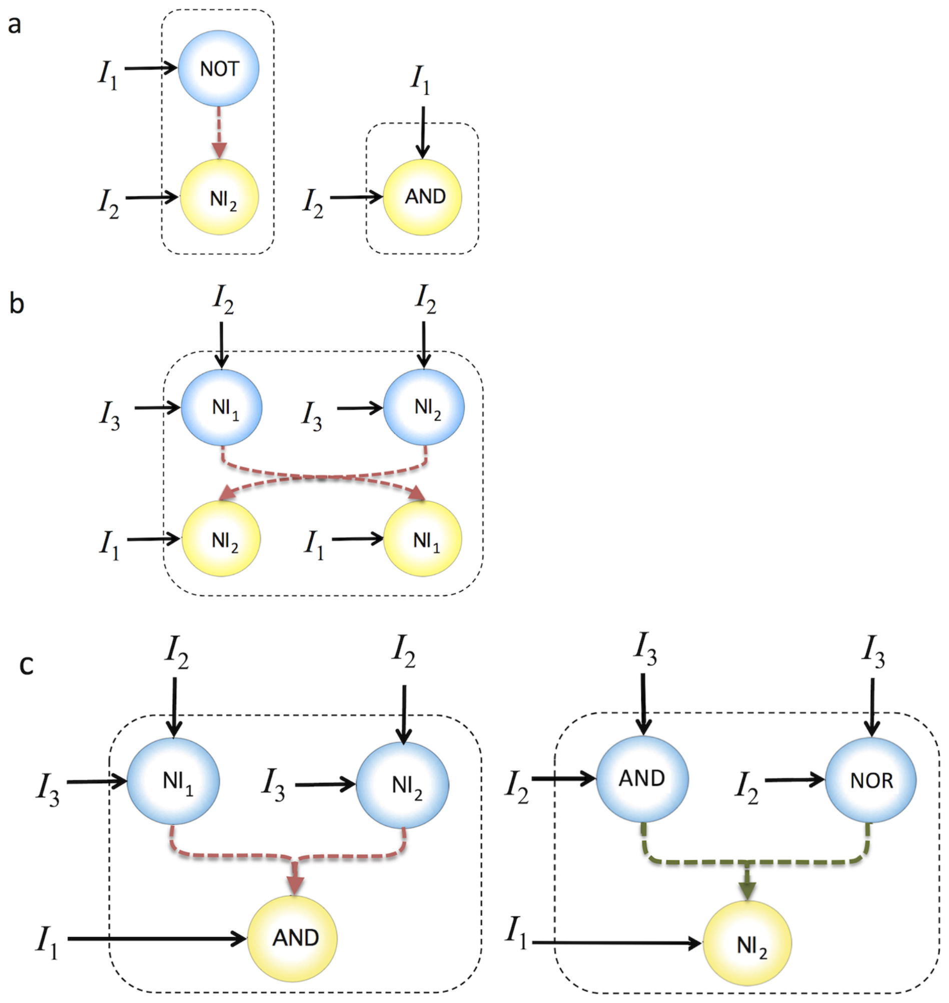


**Figure 5. Examples DMC designs obtained by using an evolutionary algorithm to find solutions to given computational functions, as defined by Boolean tables of the examples shown in figure 4.** The coloured balls represent the basic set of engineered cells indicating their internal Boolean functions. The dashed boxes indicate subsets of cells linked through the same communication signal (wire) represented by coloured dashed arrows. In (a) the binary comparator circuit is shown, after all simplifications have been performed. Figure (b) shows a DMC circuit implementing a binary three-bits adder using distributed logic. Finally, in (c) we display the minimal three-bit parity circuit is shown. Although the standard circuit is quite complex, a cell consortia involving six different cell types is enough to implement this complex function. Of note, these examples involve two different wires at the most in the most complex circuit and hence a real wet lab implementation is feasible.
doi:10.1371/journal.pone.0081248.g005

**Table 4.** Truth table for a Three-bit adder.

| $I_1$ | $I_2$ | $I_3$ | $I_1 + I_2 + I_3$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

doi:10.1371/journal.pone.0081248.t004

**Table 5.** Truth table for a 3-Parity bit circuit.

| $I_1$ | $I_2$ | $I_3$ | Paritybit |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

doi:10.1371/journal.pone.0081248.t005

wiring complexity increases once we consider three input functions. The dramatic increase in circuit complexity is especially well illustrated by the expected designs based on NAND gates (right column). Here we allow these gates to receive multiple inputs. If we force them to include only two inputs, the number of gates and links rapidly explodes.

In order to design a given circuit implementing an arbitrary Boolean function $\Phi$, we start with a simple feed-forward architecture. We define a set of input signals $\{I_1,...,I_N\}$ with $I_i \in \{0,1\}$ and a library $\Omega = \{C_q(g_{ij}^k)\}$ of different cells. Each cell implements the k *-st* Boolean function $g_{ij}^k$, responding to both an external input $I_i$ and to an internal wire $\alpha_j$, or only responding to an external input, i.e. $g_{i0}^k$.

To explore the potential sets of gates to be used in designing arbitrary circuits, we have used an evolutionary algorithm as our search engine for optimal synthetic circuits. We start with a set $\Gamma = \{\Gamma_1,...,\Gamma_s\}$ of randomly wired circuits. Here $s$ different potential solutions are considered and each circuit will have $N$ inputs and one output. The total number of possible functions implementable by this circuits is thus $2^{2^N}$. Each circuit will be composed by a random number of logic gates, which are also randomly chosen from the set $\mathcal{G}^{(1,1)} \cup \mathcal{G}^{(2,1)}$ which includes $2^1 + 2^4 = 18$ different potential choices (for general definitions, further information and classification, see [36] ). Once a given input-output function is defined, many possible combinations can implement it.

**0.2.1 Fitness function.** Each of these circuits is characterized by a fitness function $F(\Gamma_m)$. This fitness function is defined as the combination of two different terms, i.e. $F(\Gamma_m) = f(F_\Phi(\Gamma_m), F_c(\Gamma_m))$. The first term, $F_\Phi(\Gamma_m)$, measures how good the computation of $\Phi$ performed by the circuit $\Gamma_m$ is. However, good computations are not enough but it is necessary minimize the complexity of the circuit's design. This can be achieved in several ways. The second term of the fitness function, $F_c(\Gamma_m)$, accounts for additional evolutionary scenarios satisfying the criteria presented in section 2 in order to safely translate them into an experimentally feasible construct.

Specifically, $F_\Phi(\Gamma_m)$ is defined as the normalized distance

$$F_\Phi(\Gamma_m) = 1 - \frac{1}{2^N} \sum_{j=1}^{N} |O_j^m - \phi_j| \tag{3}$$

where $O_j^m$ and $\phi_j$ are the observed and expected outputs associated to the $j$-th input string (i. e. $\phi_j = \Phi(\mathrm{I}_j)$), respectively. The highest value, $F_\Phi(\Gamma_m) = 1$, will be obtained when perfect matching is achieved.

We have explored different scenarios imposing a selection force described by the second term of the fitness function, $F_c(\Gamma_m)$, namely:

**0.2.2 Evolutionary algorithm.** The evolutionary process starts from a set $\Gamma$ of $s$ circuits. Each one of these circuits is formed by a random number $T$ of logic gates, with $T \in \{1, N \cdot 2^N\}$, where $N$ is the number of external inputs. These gates are randomly wired and the process follows several different steps.

Step 1: For each circuit the computational term of the fitness ($F_\Phi(\Gamma_m)$) is evaluated according to expression (3) upon the different input strings $I_j$.

Step 2: For all circuits with highest $F_\Phi(\Gamma_m)$ values, the second term $F_c(\Gamma_m)$ is evaluated. This criterion prevents possible biases associated with the constrained component $F_c(\Gamma_m)$, favouring good computation.

Step 3: The total fitness is calculated according to

$$F(\Gamma_m) = \begin{cases} F_c(\Gamma_m) & \text{if } F_\Phi(\Gamma_m) = \max\{F_\Phi(\Gamma_i)\} \\ \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where $F_\Phi(\Gamma_i) = \{F_\Phi(\Gamma_1),...,F_\Phi(\Gamma_s)\}$. Each on of these circuits has a probability to pass to the next round proportional to the total fitness, i.e.

$$\wp(\Gamma_m) = \frac{F(\Gamma_m)}{\sum_{i=1}^{s} F(\Gamma_i)} \tag{5}$$

Step 4: Random mutations are introduced in the circuits. Random mutations can occur at different levels, namely i) addition of a new wire with probability $\mu_a$, ii) deletion of a wire with probability $\mu_d$, iii) addition of a new logic gate with probability $\theta_a$, iv) deletion of an existing gate with probability $\theta_d$, and v) modification of a logic gate with probability $\gamma$. More specifically, modifications of a logic gate are implemented by inverting a randomly chosen output bit in the truth table that defines the logic function.

Step 5: In order to maintain a constant population $s$ of circuits in each round, the set of circuits coming from the previous round is completed by new randomly generated circuits.

Step 6: Go to to Step 1.

In order to illustrate the potential for constructing simple, nonstandard circuits from the DMC metaphor, let us consider several representative examples. These examples belong to the most standard set of components used within electronic devices, but they are also relevant to potential applications of synthetic consortia. The circuits shown below are, among others, the minimal designs obtained simultaneously under the different scenarios described by the function $F_c(\Gamma_m)$ in section 3.1. Simulations were run 500 times for each $F_c(\Gamma_m)$ conditions with the set of parameters $\{\mu_a, \mu_d, \theta_a, \theta_d, \gamma\}$ randomly chosen in the interval $(0.001, 0.1)$ in a population of $s = 1000$ circuits. The logic circuits obtained have been tested using the ***Logisim*** software package (see http://ozark.hendrix.edu/~burch/logisim/ for detailed information).

1. Minimal gate number: Given a library of cells to be used, our goal is to make the total number of circuit elements as small as possible. Of note, thinking in terms of wet lab implementation, minimization of gate diversity can be useful in order to reduce the library of engineered cells. Here, $F_c(\Gamma_m)$ is defined as

$$F_c(\Gamma_m) = \frac{1}{\xi_m} \tag{6}$$

where $\xi_m$ is the total number of gates forming in the circuit $\Gamma_m$.

2. Minimal wire number: The implementation of wires, i.e. connections between elements (e.g. cells), is one of the strongest limitations for synthetic designs [34,46]. Here we minimize the total number of wires used in the circuit by defining $F_c(\Gamma_m)$ as

$$F_c(\Gamma_m) = \frac{1}{1 + \omega_m} \tag{7}$$

where $\omega_m$ is the total number of different wires present in $\Gamma_m$.

3. Minimal complexity: This optimization procedure looks for a *structurally* minimal wiring pattern [46,58] as the fitness function, in terms of information processed by the circuit. Roughly speaking, this complexity measure $C_m$ weights the contribution of modularity versus integration [58]. We define $F_c(\Gamma_m)$ as

$$F_c(\Gamma_m) = \frac{1}{1+C_m} \quad (8)$$

where $C_m$ is defined as [41].

$$C_m = \frac{1}{2}\sum_{i=1}^{Z} \langle I(X_i^q, X - X_i^q) \rangle \quad (9)$$

Here, $Z$ is the number of gates in the circuit and $I(X_i^q, X - X_i^q)$ is the mutual information between the q−*st* subset of the circuit formed by $i$ gates, i.e. $X_i^q$, and the rest of the circuit (i. e. $X - X_i^q$). The symbols $\langle\rangle$ indicate the average among all possible subsets formed by $i$ gates. Finally, the index $i$ covers all possible sizes, form subsets formed by one gate ($i=1$) to the whole circuit ($i=Z$).

Interestingly, as shown below, all these different constraints converge into essentially the same basic space of possible solutions.

## Results

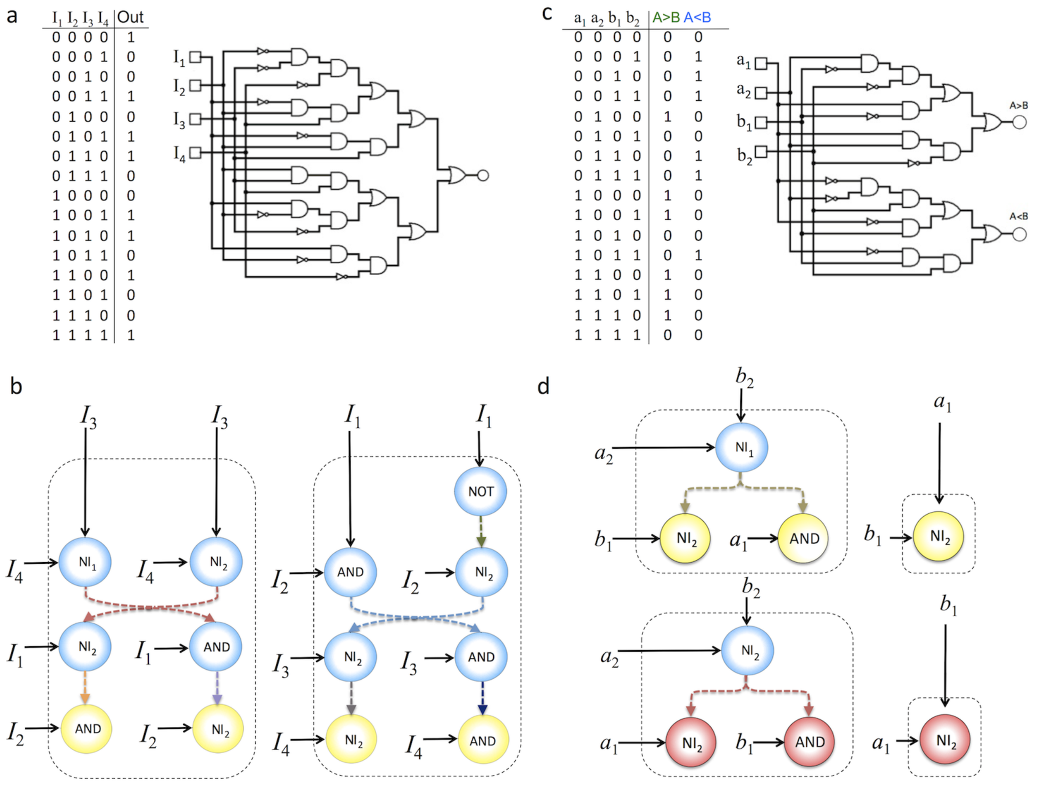
### 0.3 Multiplexer

Our first example was the MUX circuit. The MUX circuit obtained following our method is extremely simple. The resulting construct requires just two different engineered cell types. Instead of a complex set of engineered regulatory interactions within a single cell, we can create a consortium involving just two cells. The most important feature to be noticed (aside from the simple design) is that the cells *are not connected* as a consequence of the distributed output. The whole system performs the computation and thus both pieces are required, but there is no need to couple them.

### 0.4 Comparator

In many relevant applications, a given device might need to evaluate when two or more signals are equal or not. A comparator circuit performs such an evaluation. Table 3 describes the Boolean function for a problem involving two bits (corresponding to the XNOR logic).

The minimal implementation in terms of number of logic gates is a single XNOR gate. However, in a biological context, implementation of complex gates such as XOR and XNOR require the layering of multiple genetic circuits, thus necessitating substantial efforts in circuit construction and tuning [59]. As a standard alternative, this circuit can be built by combining several NOT, AND and OR gates. Comparators are a widespread



| $I_1$ | $I_2$ | $I_3$ | $I_4$ | Out |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

| $a_1$ | $a_2$ | $b_1$ | $b_2$ | A>B | A<B |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 |

**Figure 6. Scaling up DMC.** In (a) we consider a complex, nontrivial 4-input 1-output Boolean function analysed in Marchisio and Stelling 2011 [57]. Figure shows the truth table and the standard design using two-input gates along with NOT gates. Below (b) we display a minimal circuit implementing this function. In (c) a two-bits comparator circuit is implemented according with the standard methodology, whereas in (d) an alternative design obtained by evolution is shown involving less gates an wires Only two communication signals (wires) are required for this implementation. Two colours (red and yellow) are used to indicate the two reporter molecules associated to each possible output (either $A>B$ or $A<B$).
doi:10.1371/journal.pone.0081248.g006

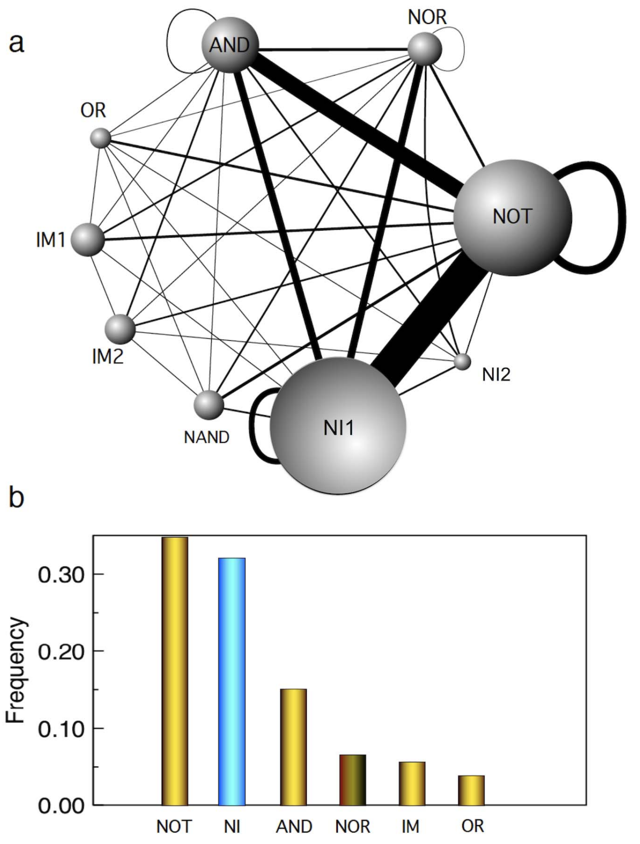


**Figure 7. Graphs showing the frequency of gates in a MUX circuit.** (a) Weighted graph showing the frequency of gates used in the generation of evolved MUX circuits under DMC. The diameter of the nodes is proportional to the frequency of the gate. Links indicate that two gates have been wired together within a circuit. The weights provide a measure of how frequently a given pair has been used. Note the disproportionate frequency of N-implies ($NI_1$) which are typically connected to NOT and AND gates, much less with NOR gates. Note also the vanishing frequency of NOR-NOR links (see text). In (b) we summarize the relative frequencies of different gates. The two N-Implies gates (here NI = N1+IN2) have been added together. These and the NOR abundances are highlighted with distinctive colours. Circuits were evolved imposing minimal number of wires.
doi:10.1371/journal.pone.0081248.g007

component in most electronic devices and are commonly used in converters, detectors, and oscillators. Their potential for synthetic biology applications seems clear if we consider that most potential decision-making circuits performing complex tasks are likely to require this type of operation.

Here, we have evolved this circuit imposing a minimization of circuit complexity. The minimal configuration obtained involves three different cell types implementing the AND, NOT and NI gates, and one wire (see figure 5a). As it occurs with the MUX system, two subsets of cells that do not exchange signals can be used to implement the Boolean function, whereas five NAND gates would be needed in the NAND-based logic. Of note, implementation of logic gates such as NOT, AND and NI are easier than implementation of XNOR gates [59], and the circuit implementation combining these gates (despite one wire being required) can be easier from an experimental point of view.

## 0.5 Three-bit Adder

An additional operation that is essential in most devices is the binary addition of several input numbers described by the truth table. Let us consider the three-bit adder, as defined in table 4.

This circuit requires a large number of gates (in both scenarios) and wires that make it almost prohibitive for real scenarios. Because of this, it provides a perfect illustration of the potential for strongly reducing circuit complexity by combining non-standard gates with distributed output production. Here the standard implementation would be highly difficult to implement, whereas the consortium solution is completely feasible (see figure 5b).

## 0.6 3-Parity Bit Circuit

Our fourth example is another three-input, one-output circuit that implements the parity bit circuit, described in table 5. An even parity bit circuit generates an output of 0 if the number of 1 *s* in the input sequence is even and 1 if the number of 1 *s* in the input sequence is odd. Because of its simplicity, parity is used in many applications where an operation can be repeated in case of difficulty, or where simply detecting an error is helpful. Parity can be used as a control to check whether the input string is the expected one or some error (wrong bit) is present. Figure 5c shows the minimal implementation found by the evolutionary algorithm. In this case two different wires are required.

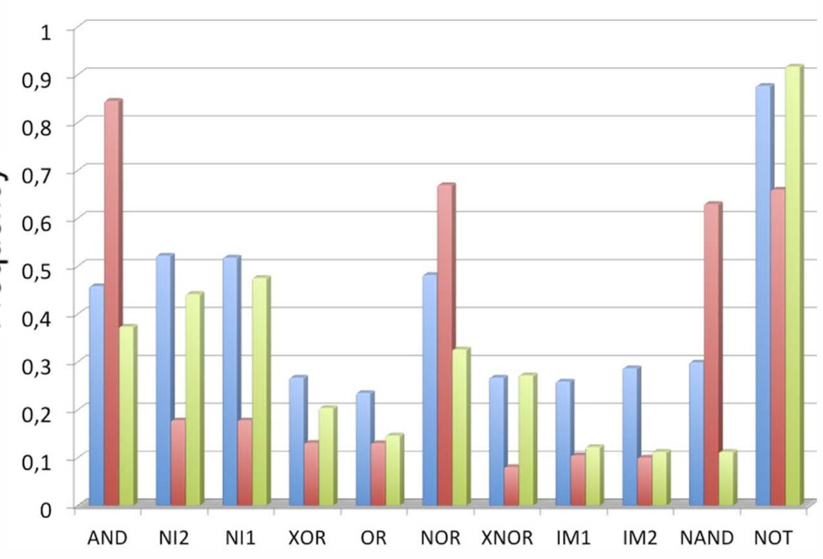
## 0.7 4-Inputs: Two Illustrative Examples

In order to emphasize the advantages of DMC combined with distributed output production, we have analyzed one of the most complex functions involving four inputs. This function, described by the truth table shown in figure 6a, is complex due to the fact that the canonical form does not allow for Boolean simplifications, as analysed in [57]. The function involves a rather complex computation, which cannot be reduced nor decomposed in simpler components and, as a result, has a complex associated standard circuit. Here 17 two-input gates (and four NOT gates) are needed to construct the circuit based on electronic design rules. As we can see, the number of wires rapidly explodes. Such circuit complexity makes a mapping between these designs and a synthetic construct highly unlikely to even be possible.

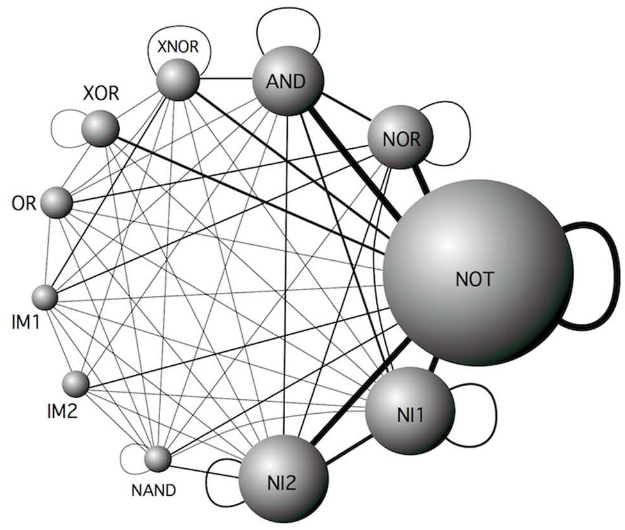
Again the solutions found by using our evolutionary algorithm minimizing the number of wires involve a reduced number of wires and cells. Figure 6b shows the minimal circuit obtained, involving seven wires and thirteen cells, all of them with the minimal engineering required for our method. This is the worst case scenario that can be found. Another circuit of great importance in electronic engineering, the so called two-bit comparator, which processes 4-bit input strings and has a two-bit output, is shown in figure 6c. The standard circuit is again very large, with 14 two-input gates and four inverters. In figure 6d the corresponding DMC design is shown, requiring only eight cells. Of note, it is interesting to see that the minimal circuits are formed by a non-standard combination of AND and N-Implies gates. The large number of the later type is a characteristic trait of DMC evolved circuits, as shown below.

## 0.8 Bias Towards Non-standard Gates

A more general analysis has been performed by considering the whole set of possible random 3-input, 1-output functions. Our evolutionary algorithm was used to search for evolved DMC designs minimizing circuit size and circuit complexity. The resulting optimized circuits involve different abundances of different gates and have different topological arrangements. Our results are summarized by means of a graph that captures both the a


Frequency
1
0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0
AND
NI2
NI1
XOR
OR
NOR
XNOR
IM1
IM2
NAND
NOT

b


XNOR
AND
XOR
NOR
OR
NOT
IM1
IM2
NI1
NAND
NI2

c

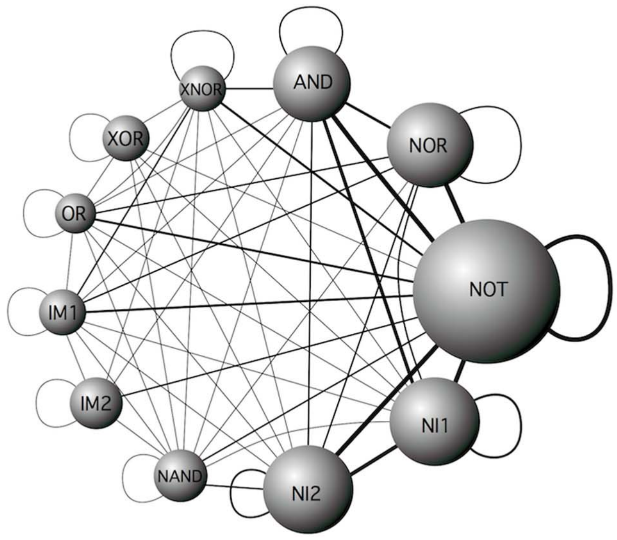

XNOR
AND
XOR
NOR
OR
NOT
IM1
IM2
NI1
NAND
NI2

**Figure 8. Frequency of appearance of different logic gates for circuits evolved without evolutionary pressure.** In (a) the graph displays the frequency of appearance of different logic gates for circuits evolved without any specific evolutionary pressure (red), imposing minimum size (green), and under minimal circuits complexity (blue). These frequencies have been calculated as the average results for 500 runs. In (b) the graph displays the average results for 500 runs of an evolutionary pressure towards minimum size. The frequency of a given gate is proportional to node diameter, whereas the probability of finding two gates in the same circuit is given by the link weights. In (c) we show the corresponding results for evolved networks under minimal circuits complexity. In both cases, the most represented gates are NOT, AND, $NI_1$, $NI_2$ and NOR.
doi:10.1371/journal.pone.0081248.g008

frequency of gates used in the final design and how often they are found together within a given solution.

Again, let us start with the MUX circuit, which is a good representation of a complex design. Figure 7a shows the weighted graph associated to the frequency of gates used in the generation of evolved MUX circuits imposing wires minimization. These frequencies are the average of 500 independent runs of the evolutionary algorithm evolving MUX circuits.

The diameter of the nodes is proportional to the frequency of its abundance and the thickness of the links represents the probability that two gates are present in the same circuit. The graph allows us to draw to important conclusions.

We can observe an overabundance of a few gates. The histogram shown in figure 7b allows comparing these frequencies easily. The most common 2-input 1-output gate is clearly non-standard from the point of view of engineering principles. The second is that neither combinations of the standard gates NOR or NAND seem to be very relevant here. Their frequency is much smaller than other gates, particularly the non-conventional N-IMPLIES ($NI_1$), which is the most common choice, followed by AND and NOT gates. The fact that the most common gates define a non-standard set of logic functions (which has been shown to be complete, see [45]) suggests that we must consider a whole alternative landscape of combinatorial logic designs under our distributed computation paradigm.

This analysis can be extended to the whole set of 3-input 1-output functions, i.e. for each possible function the evolutionary algorithm has found a possible solution under different constraints. This experiment has been repeated 500 times and the frequencies shown in figure 8 are the average of these independent runs of the evolutionary algorithm. Figure 8a shows the different frequency of appearance of logic gates in circuits evolved without additional constrains (red bars), i.e. the unique imposition is a proper computation, and the corresponding results obtained by evolution imposing minimal size (green bars) and minimal complexity (blue bars). As the figure shows, the patterns are significantly different. In the absence of additional constraints, the most frequent gates are NOT, AND, and not surprisingly NOR and NAND gates. However, in the presence of additional constrains, patterns change and the abundance of NOR and NAND gates is reduced, whereas the $NI_1$ and $NI_2$ gates emerge as an alternative for improved circuits.

Despite the NOR gate is a functional complete set by itself, circuits involving only, or dominantly, NOR gates are not the optimal solutions found as the thin auto-link in the NOR node of the graphs indicate (figures 8b–c). Similar arguments can be applied to the NAND gate. Of note, the similarity between graphs obtained imposing minimal size and minimal informational complexity is remarkable, suggesting that the optimization of one aspect could be related with an implicit optimization of the other. Future work should be devoted to analyse these implications carefully.

## Discussion

The continuous advance towards the design and synthesis of biocomputers shows that there is an enormous potential for innovation [60,61]. One of the goals of ongoing synthetic designs is the construction of complex living circuits able to perform complex decision-making tasks. Here we present a promising approach to this problem, which requires reliable and flexible approximations and the potential for extensive reuse and combination of basic units.

We have explored the landscape of circuit designs associated to a number of complex decision-making constructs based on distributed multicellular computation [45,46]. Such circuits include a large number of potentially important scenarios, including among them many related to biomedical applications, bioremediation scenarios, and several kinds of bioengineering problems. Engineering biofilms, tissue architecture, and growing biomaterials all deal with multiple interacting cell types and it is precisely its multicellular character that makes our method scalable and easy to implement. Our results suggest that the potential associated with multicellular consortia, which have been explored in different scenarios [62–65], can be used for designing complex computational devices.

The results described above suggest that, according to the design rules proposed, other functional complete sets, such as AND and NI, can be more optimal for cellular implementation than the standard ones, indicating that the optimal functional complete set dependents on constraints imposed on the circuit topology. Our approximation strongly departs from standard electronic design and in doing so we are able to greatly reduce the connectivity requirements as well as provide a source of flexible combinatorial power.

Our work provides a robust approach to building more complex computations, is predictable and scalable, and opens the door to the future design of multicellular chips. It also allows for re-thinking the way cells and tissues process information beyond man-made metaphors. Its general nature makes it easily extendable to other forms of molecular interactions and model organisms or even protocellular systems [25,66] and can take advantage of additional techniques, such as microencapsulation [67,68]. Our results provide a proof of concept that the expectations of synthetic biology in terms of creating complex computational functions from simple ones can be reached, and allows for the exploration novel forms of optimization beyond standard engineering [69].

Despite the promising previous results obtained so far under the multicellular consortia approximation [44,45], future work should be devoted to analyse possible drawbacks that can limit the potentiality of multicellular circuits, such as stochastic effects on low cell number populations, the impact of different growth rates in a multicellular consortia, circuit stability in long term experiments, and experimental implementation of different wire molecules. These limitations will be strongly dependent upon the specific embodiment of the future synthetic devices. For instance, the same logic circuit can be implemented by using transcriptional or post-transcriptional regulatory genetic circuits hosted in prokaryotes or eukaryotes organisms, and depending on these aspects the same circuit can behave in a different way.

Moreover, other layers of the hierarchy of computational systems, including memory and more complex integrated circuits, should also be explored. Although still far from a complex architecture that we could identify with a "computer", our method shows enormous potential to achieve such goal.

## Acknowledgments

We thank Alfred Borden and the members of the Complex Systems Lab for useful discussions.

## Author Contributions

Conceived and designed the experiments: JM RS. Performed the experiments: JM RS. Analyzed the data: JM RS. Contributed reagents/materials/analysis tools: JM RS. Wrote the paper: JM RS.

## References

1. Hopfield M (1994) Physics, Computation, and Why Biology Looks so Different. J. Theor. Biol. 171: 53–60.
2. Nurse P (2008) Life, logic and information. Nature 454: 424–426.
3. Brenner S (2012) Turing centenary: Life's code script. Nature 482: 461.
4. Bray D (1995) Protein molecules as computational elements in living cells. Nature 376: 307–312.
5. Amos M (2004) *Cellular Computing.* Oxford University Press, New York.
6. Amos M (2005) Theoretical and Experimental DNA Computation. Springer, New York.
7. Reed MA, Tour MJ (2000) Computing with molecules. Sci. Am. 282: 86–93.
8. Simpson ML, Sayler GS, Fleming JT, Applegate B (2001) Whole-cell biocomputing. Trends Biotechnol. 19: 317–323.
9. Sauro HH, Khodolenko BN (2004) Quantitative analysis of signaling networks. Prog. Biophys. Mol. Biol. 86: 5.43.
10. Istrail S, Ben-Tabou S, Davidson EH (2007) The regulatory genome and the computer. Dev. Biol. 310: 187–195.
11. Hogeweg P (2002) Computing an organism: on the interface between informatic and dynamic processes. Biosystems 64: 97–109.
12. Deneubourg JL (1989) Collective patterns and decision-making. Ethol. Ecol. and Evol. 1: 295–311.
13. Solé RV, Delgado J (1996) Universal Computation in Fluid Neural Networks. Complexity 2: 49–56.
14. Fernando CT, Liekens AM, Bingle LE, Beck C, Lenser T, et al. (2009) Molecular circuits for associative learning in single-celled organisms. J. R. Soc. Interface 6: 463–469.
15. Perkins TJ, Swain PS (2009) Strategies for cellular decision-making. Mol. Syst. Biol. 5: 326.
16. Hjelmfelt A, Weinberger ED, Ross J (1991) Chemical implementation of neural networks and Turing machines. Proc Natl Acad Sci USA. 88: 10983–10987.
17. Wang B, Kitney RI, Joly N, Buck M (2011) Engineering modular and orthogonal genetic logic gates for robust digital-like synthetic biology. Nature Communications 2: 508. doi: 10.1038/ncomms1516.
18. Wang B, Barahona M, Buck M (2013) A modular cell-based biosensor using engineered genetic logic circuits to detect and integrate multiple environmental signals. Biosensors and Bioelectronics 40: 368–376.
19. Kauffman SA (1993) The origins of order. Oxford U. Press, New York.
20. Kauffman SA, Peterson C, Samuelsson S, Troein C (2003) Random Boolean network models and the yeast transcriptional network. Proc. Natl. Acad. Sci. USA 100: 14796–14799.
21. Klemm K, Bornholdt S (2005) Topology of biological networks and reliability of information processing, Proc. Natl. Acad. Sci. USA 102: 18414–18419.
22. Andrianantoandro E, Basu S, Karig D, Weiss R (2006) Synthetic biology: new engineering rules for an emerging discipline. Mol Syst Biol. 2: E1–E14.
23. Goni-Moreno A, Amos M, de la Cruz F (2013) Multicellular Computing Using Conjugation for Wiring. PLoS ONE 8(6): e65986.
24. Purnick PEM, Weiss R (2009) The second wave of synthetic biology: from modules to systems. Nature Reviews Molecular Cell Biology 10: 410–422.
25. Solé RV, Munteanu A, Rodriguez-Caso C, Macia J (2007) Synthetic protocell biology. From reproduction to computation. Phil. Trans. R Soc. B Biol Sci. 362: 1727–1739.
26. Kobayashi H, Kaern M, Araki M, Chung K, Gardner TS, et al. (2004) Programmable cells: Interfacing natural and engineered gene networks. Proc. Natl. Acad. Sci. USA 101: 8414–8419.
27. Kramer BP, Fischer C, Fussenegger M (2004) BioLogic gates enable logical transcription control in mammalian cells. Biotechnol. Bioeng. 87: 478–484.
28. Benenson Y, Gil B, Ben-Dor U, Adar R, Shapiro E (2004) An autonomous molecular computer for logical control of gene expression. Nature 429: 423–429.
29. Tan CM, Song H, Niemi J, You LC (2007) A synthetic biology challenge: making cells compute. Mol. Biosyst. 3: 343–353.
30. Benenson Y (2009) Biocomputers: from test tubes to live cells- Mol. BioSyst. 5: 675–685.
31. Friedland AE, Lu TK, Wang X, Shi D, Church G, et al. (2009) Synthetic gene networks that count. Science 324: 1199–1202.
32. Ruder WC, Lu TK, Collins JJ (2011) Synthetic Biology Moving into the Clinic. Science 333: 1248–1252.
33. Weber W, Fussenegger M (2012) Emerging biomedical applications of synthetic biology. Nature Rev. Gen. 13: 21–35.
34. Kwok R (2010) Five hard truths for synthetic biology. Nature 463: 288–290.
35. Moon TS, Clarke EJ, Groban ES, Tamsir A, Clark RM, et al. (2011) Construction of a genetic multiplexer to toggle between chemosensory pathways in Escherichia coli. J Mol Biol. 406: 215–27.
36. Tinder RF (2000) Engineering digital design: Revised Second Edition 317319. ISBN 0-12-691295-5.
37. VerBerkmoes NC, Denef VJ, Hettich RL, Banfield JF (2009) Systems Biology: Functional analysis of natural microbial consortia using community proteomics Nature Reviews Microbiology 7: 196–205.
38. Bernstein HC, Carlson RP (2012) Microbial Consortia Engineering for Cellular Factories: in vitro to in silico systems Computational and structural biotechnology journal 3: e201210017. Available: http://dx.doi.org/10.5936/csbj.201210017.
39. O'Toole G, Kaplan HB, Kolter R (2000) Biofilm formation as microbial development. Annu Rev Microbiol. 54: 49–79.
40. You L, Cox RS, Weiss R, Arnold FH (2004) Programmed population control by cell-cell communication and regulated killing. Nature 428: 868–871.
41. Basu S, Gerchman Y, Collins CH, Arnold FH, Weiss R (2005) A synthetic multicellular system for programmed pattern formation. Nature 434: 1130–1134.
42. Shou W, Ram S, Vilar JMG (2006) Synthetic cooperation in engineered yeast populations. Proc. Natl. Acad. Sci. USA 104: 1877–1882.
43. Brenner K, Karig DK, Weiss R, Arnold FH (2007) Engineered bidirectional communication mediates a consensus in a microbial biofilm consortium. Proc. Natl. Acad. Sci. USA 104: 17300–17304.
44. Tamsir A, Tabor JJ, Voigt CA (2010) Robust multicellular computing using genetically encoded NOR gates and chemical wires. Nature 469: 212–215.
45. Regot S, Macia J, Conde N, Furukawa K, Kjelln J, et al. (2011) Distributed biological computation with multicellular engineered networks. Nature 469: 207–211.
46. Macia J, Posas F, Solé RV (2012) Distributed Computation: The New Wave of Synthetic Biology Devices. Trends Biotech. 30: 342–349.
47. Li B, You L (2011) Division of logic labour. Nature 496: 171–172.
48. McGregor S, Vasas V, Husbands P, Fernando C (2012) Evolution of Associative Learning in Chemical Networks. PLoS Comput Biol 8(11): e1002739. doi:10.1371/journal.pcbi.1002739.
49. Enderton H (2001) A Mathematical Introduction to Logic, second edition. Harcourt Academic Press.
50. Marchisio MA, Stelling J (2009) Computational design tools for synthetic biology. Curr. Opin. Biotech. 20: 479–485.
51. Rodrigo G, Jaramillo A (2007) Computational design of digital and memory biological devices. Syst. Synth. Biol. 1: 183–195.
52. Cao H, Romero-Campero FJ, Heeb S, Camara S, Krasnogor N (2010) Evolving cell models for systems and synthetic biology. Syst. Synth. Biol. 4: 55–84.
53. Rodrigo G, Carrera J, Jaramillo A (2007) Genetdes: automatic design of transcriptional networks. Bioinformatics 23: 1857–1858.
54. Huynh L, Kececioglu J, Koppe M, Tagkopoulos I (2012) Automatic Design of Synthetic Gene Circuits through Mixed Integer Non-linear Programming. PLoS ONE 7(4): e35529.
55. François P, Hakim V (2004) Design of genetic networks with specified functions by evolution in silico. Proc. Natl. Acad. Sci. USA 101: 580–585.
56. Dasika MS, Maranas CD (2008) Optcircuit: an optimization based method for computational design of genetic circuits. BMC Syst Biol 2: 24.
57. Marchisio MA, Stelling J (2011) Automatic Design of Digital Synthetic Gene Circuits. PLoS Comput. Biol. 7: e1001083.
58. Tononi G, Sporns O, Edelman GM (1999) Measures of degeneracy and redundancy in biological networks. Proc. Natl. Acad. Sci. USA 96: 3257–3262.
59. Siuti P, Yazbek J, Lu TK (2013) Synthetic circuits integrating logic and memory in living cells. Nature Biotechnology. doi:10.1038/nbt.2510.
60. Benenson Y (2012) Biomolecular computing systems: principles, progress and potential. Nature Rev. Genet. 13: 455–468.
61. Ausländer S, Ausländer D, Muller M, Wieland M, Fussenegger M (2012) Programmable single-cell mammalian biocomputers. Nature 487: 123–127.
62. Weber W, Daoud-El Baba M, Fussenegger M (2007) Synthetic ecosystems based on airborne inter- and intrakingdom communication. Proc. Natl. Acad. Sci. USA 104: 10435–10440.
63. Brenner K, Arnold FH (2008) Engineering microbial consortia: a new frontier in synthetic biology. Trends Biotech. 28: 483–489.
64. Wintermute EH, Silver PA (2010) Dynamics in the mixed microbial concourse. Genes Dev. 24: 2603–2614.

65. Song H, Payne S, Gray M, You L (2009) Spatiotemporal modulation of biodiversity in a synthetic-mediated ecosystem. Nat. Chem. Biol. 5: 929–935.
66. Smaldon J, Romero-Campero FJ, Fernandez Trillo F, Gheorghe M, Alexander C, et al. (2010) A computational study of liposome logic: towards cellular computing from the bottom up. Syst. Synth. Biol. 4: 157–179.
67. Chang TMS (2005) Therapeutic applications of polymeric artificial cells. Nature Rev. Drug Discov. 4: 221–235.
68. Ausländer S, Wieland M, Fussenegger M (2012) Smart medication through combination of synthetic biology and cell microencapsulation. Metab. Eng. 14: 252–260.
69. Banga JR (2008) Optimization in computational systems biology. BMC Syst. Biol. 2: 47.