

Published in final edited form as:

Curr Opin Biotechnol. 2010 August ; 21(4): 572–581. doi:10.1016/j.copbio.2010.07.005.

Programming Cells: Towards an automated “Genetic Compiler”

Kevin Clancy¹ and Christopher A. Voigt²

Christopher A. Voigt: cavoigt@gmail.com

¹ Life Technologies, 5791 Van Allen Way, Carlsbad, CA, 90028

² Department of Pharmaceutical Chemistry, University of California-San Francisco, MC 2540, Room 408C, 1700 4th Street, San Francisco, CA 94158

I. Summary

The increasing scale and sophistication of genetic engineering will necessitate a new generation of computer-aided design (CAD). For large genetic programs, keeping track of the DNA on the level of nucleotides becomes tedious and error prone. To push the size of projects, it is important to abstract the designer from the process of part selection and optimization. The vision is to specify genetic programs in a higher-level language, which a genetic compiler could automatically convert into a DNA sequence. Steps towards this goal include: defining the semantics of the higher-level language, algorithms to select and assemble parts, and biophysical methods to link DNA sequence to function. These will be coupled to graphic design interfaces and simulation packages to aid in the prediction of program dynamics, optimize genes, and scan projects for errors.

Keywords

Computer-aided design; systems biology; synthetic biology; design automation

II. Introduction

Numerous genetic circuits have been built that encode functions that are analogous to electronic circuits [1–3]. Genetic programs have been built by combining multiple circuits. For example, we constructed an “edge detector” program that combines a ANDN gate, light sensor, and cell-cell communication that give bacteria the ability to draw the edge between light and dark regions of an image projected onto a plate [4]. Other genetic programs have been built that combine circuits to produce a push-on/push-off circuit [5], implement a counter [6], and reproduce predator-prey dynamics [7]. These represent toysystems, but the implementation of such programs in applications for industrial biotechnology is inevitable.

Automated DNA synthesis gives genetic engineers an unprecedented design capacity [8]. This technology enables the specification of every basepair for long sequences, without having to be concerned about the path to construction. Together with methods to rapidly combine genetic parts [9,10] and assembly methods that scale to whole genomes [11–13], the problem of DNA construction has far outpaced our capacity for design [14]. A good example of this is the 2006 UCSF iGEM team to build a “remote-controlled bacterium.” DNA synthesis was used to build the first construct (requiring a few weeks), but after four years of additional tinkering, the paper will be submitted in 2010.

Publisher's Disclaimer: This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final citable form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

Our ability to design programs has been hampered by three problems. First, there is a lack of good, robust genetic circuits that can be easily connected. Second, there are few design rules that are sufficiently quantitative to be carried out by a design algorithm. Modeling can be helpful before the experiments to determine the topologies and parameter regimes required to obtain a particular function. However, simulations cannot be used to “reach down” to the DNA and suggest a specific mutation or select a part. Third, mistakes in the DNA sequence scale quickly with size. Currently, to scan for potential errors (*e.g.*, transposon insertion sites or putative internal promoters), it requires the running of multiple, (usually web-based) programs. There is no unified software package to date that addresses all of these issues.

The creation of a simulation environment for genetic engineering is complicated by the diversity of cellular functions. When studying natural networks, there is a feeling of “peeling an onion,” where there are seemingly endless redundancies and classes of biochemical interactions. Even within the Registry of Standard Biological Parts (www.partsregistry.org), there are a wide variety of cellular functions: from enzymes and transcription factors to multi-gene gas vesicles and secretion chaperones. Each specific problem requires its own style of simulation; a dynamic program may be well satisfied by sets of differential equations, pattern formation by cellular automata, and enzymes by metabolic flux analysis. It would be daunting to create a simulation package that could encompass all of this.

To reduce the problem complexity and to frame recent computational work, we introduce the concept of a “Genetic Compiler,” whose inputs are high-level instructions (equivalent to VHDL or Verilog) and whose output is a DNA sequence. The sequence can be sent to a company for DNA synthesis or a robot for automated assembly. The problem is constrained by focusing on genetic programs that encode a desired logical or dynamical function, which can be integrated into many applications in biotechnology (Figure 1). This avoids the application-specific portions of the problem; for example, building a butanol sensor a particular metabolic pathway. It is distinct from tools for protein or metabolic engineering [15].

The scope of this review is on the underlying algorithms and biophysical methods that would power such a compiler (Figure 2). Realizing this goal will require: 1. Libraries of reliable genetic circuits designed specifically to be part of a CAD program, 2. the definition of a higher-level language, 3. algorithms to assemble circuits according to a specified program, 3. biophysical methods to connect and optimize circuits, 4. simulation programs to debug the program dynamics, 5. algorithms for DNA assembly and experimental design. The scope has been limited to exclude several topics that are critical to synthetic biology, but have been well-reviewed elsewhere, notably codon optimization and tools from systems biology and metabolic engineering [15–17].

III. Main Text

Robust Combinatorial Logic

Combinatorial logic is implemented by Boolean circuits and is the basis for digital computing. It is used to build circuits that apply Boolean algebra on a set of inputs to transform them into a set of desired outputs. Simple circuits can be layered in different configurations in order to achieve a computational operation. This has enabled the automated design that underlies VLSI. The ability for digital circuits to be flexibly used and easily captured by CAD comes at a cost of speed, design size, and power [18]. There are ongoing efforts to design the core biochemistry that would act as genetic logic gates [4,19–29]. There are several important considerations in designing genetic logic gates to be used in CAD:

- *Scalability.* Each circuit in a design needs to be orthogonal because all of the circuits operate within the cell based on biochemical interactions. A circuit is scalable when the underlying genetic parts can be swapped to create orthogonal circuits. For example, orthogonal NOT gates could be built based on new repressor-operator pairs [30]. In contrast, an AND gate that we constructed is not scalable because its underlying parts do not lend themselves to making multiple gates [27].
- *Extensibility.* In order to layer genetic circuits, the inputs and outputs of the circuit have to have the same signal carrier [31]. Put simply, for circuits to be connected, the inputs and the outputs need to be the same biochemical form (phosphorylation, transcription, etc). For transcriptional circuits, it has been proposed that the flux of RNAPs on DNA could serve this purpose [31]. In practice, this can be implemented by making the circuit inputs and the outputs be promoters. Several examples of this are a NOT gate [30] and an AND gate [27]. This is in contrast to circuits where, for example, the inputs are small molecules and the output is the activation of a riboswitch [25].
- *Modularity.* The inputs and outputs of the circuits need to be able to be easily changed. Considering CAD, the ease has to be sufficient for a computer to be able to reliably perform this function. Promoters, protein-protein interaction domains, and RNA are sufficiently modular for this purpose [25]. Some biological systems that are considered modular – such as the protein domains of bacterial two-component systems – are not sufficiently modular for CAD. This is because small libraries are required to identify a functional chimera, as opposed to design rules that can be implemented by a machine.
- *Robustness.* The circuits must remain functional over large regions of parameter space. This is particularly important to be able to connect circuits in series, where robust circuits will allow for more uncertainty while still producing a functional connection [4,32]. The circuits also have to be non-toxic and have minimal impact on host processes [33].
- *Speed.* The circuits also need to be reasonably fast. For example, a three-layer cascade of NOT gates required 20 minutes for each layer to compute [34], but a DNA inversion switch requires 4hours [2]. Even 20 minutes may be too slow for some applications, thus requiring logic to be implemented by phosphorelays or protein-protein interactions, which can occur in milliseconds [35].

Higher-Level Languages and their Deconstruction

A higher-level language abstracts the designer from the details of the machine. In electronics, it enables the CPU operations to be hidden. In synthetic biology, it would hide the details of the molecular biology and DNA sequence, and possibly even the choice of circuits. Impeding the development of a higher-level language in synthetic biology is the diversity of cellular functions. This can be partially overcome by limiting the language to describe problems that are common to many applications, while leaving the application-specific aspects unabstracted (Figure 1). There are two areas that have shown promise in moving towards a higher-level language in synthetic biology. First, vocabularies and grammars are being developed that implement design rules and can be used to describe genetic parts and their combination. This will form the backbone of the higher-level language. Second, there are a number of algorithms that can be borrowed from electrical engineering, notably logic minimization, which can convert the language into integrated circuits.

The Semantics of Genetic Programs—A higher-level language in synthetic biology needs a vocabulary and grammar to represent genetic parts and the rules underlying how they are combined to build circuits and programs [36]. A vocabulary captures how genetic functions are described and how the data underlying a part is stored and accessed. A rule is an enforced relationship between parts. For example, an expression cassette is defined as needing a promoter, cistron, and terminator and a cistron requires a RBS sequence and gene [37].

An example of a program that describes a well-known genetic oscillator (the “repressilator”) is shown in Figure 3. At first glance, this appears unnecessarily complex compared to the more familiar plasmid visualization. However, it becomes advantageous as the designs get larger and more complex. Plasmid construction programs (like Vector NTI or ApE) toggle between the plasmid map for visualization and the DNA sequence for design. Both are difficult for debugging large programs, whereas formal grammars give an intermediate level of abstraction for writing programs and the rules enable automated troubleshooting [37]. Rules also allow for the permutation of parts in order to create multiple designs [38].

An emerging group of tools (GenoCAD, Eugene, and GEC) propose formalisms for vocabularies and provide a means to embed domain expertise in software allowing less experienced users to benefit from it [39–41]. Domain experts can express problem-specific design strategies as formal languages using a controlled vocabulary to represent the types of genetic parts usable in a particular organism or even for a particular application [40]. Formal languages provide a means to develop a knowledge base on a collection of defined terms organized with respect to their structural relationships to each other in the DNA sequence [36]. The advantage of this approach is that the relationship of the individual parts to each other are defined in the software and can provide a means of indexing and retrieving knowledge about parts and designs associated with defined terms. When expressing design strategies with formal languages it is possible to use parsers to verify that parts used in a design are properly positioned with respect to each other [42]. Thus, users have a level of *in silico* validation through the use of such systems. Beyond this syntactic or structural layer, languages can be augmented by adding a semantic layer to derive dynamics encoded in complex DNA sequences composed of interacting parts using algorithms originally developed to compile the source code of computer programs [37].

Instead of simulating the function of a DNA sequence, it is possible to perform a semantic analysis of DNA sequences using expert systems capable of reasoning using the relationships between defined terms to infer many aspects of their use, assembly based on data present in the part or associated with the biology from which the part was derived. An expert system would scan *in silico* designs for logical errors in construction, identify constraints that would preclude use of particular combinations of parts. Such an expert system can be constructed using ontologies – a rigorous organization of knowledge concerning a particular domain of knowledge [43]. Ontologies are particularly good examples of expert systems in that they permit the development of a model of the entities and their relationships within the domain [44]. Such definitions are logically consistent and can be used by reasoner softwares to verify known information on entities tagged by the ontology and to infer new information based upon these definitions. Softwares developed to use ontologies have the advantage that as data models change in the ontology, the software simply needs to adapt to the ontology, simplifying many aspects of software data model development and design [44]. The synthetic biology community has started the Synthetic Biology Ontology Language to represent concepts tied to synthetic biology [45]. Ongoing efforts aim clarifying how the SBOL approach and the languages used in GenoCAD complement each other.

A larger model that the synthetic biology community can take advantage of is the National Center for Biomedical Ontologies [46]. The NCBO supports researchers by providing resources to access, review and integrate many biomedical ontologies. The Ontology for Biomedical Research (OBI) is especially interesting as it has been developed to describe the disparate elements and their relations that are necessary and sufficient to describe an experiment [47]. Many of the concepts from OBI are very applicable to the description of synthetic biology workflows, protocols, parts and device usage. NCBO also promotes standards for sharing data across ontologies, most notably the Minimum Information to Reference an External Ontology [48]. Use of MIREOT guidelines during the development of an ontology will support the consistent inclusion of terms from ontologies as diverse as Gene, Chemical Information and Pathway Information and Phenotypic Information.

Logic Minimization—Several programs developed in electrical engineering have the ability to take a truth table as an input and then output a wiring diagram from simpler circuits (Figure 4). The ESPRESSO program, developed in the early 1980s, has been used extensively for logic minimization in VLSI design [49] and it is embedded along with other tools in the abc program that is currently maintained by UC-Berkeley [50]. The output of the logic minimization tools feeds into programs, such as Logic Friday [51], which both act as a visualization tool and enable constraints to be applied to the construction of a circuit diagram. The minimum identified by these programs is not guaranteed to be the global minimum. The logic minimization programs will have to be modified to include biological constraints, including the size of the DNA, and availability and orthogonality of the gates. Two additional considerations in choosing a wiring diagram are:

- *Fault tolerance with respect to asynchronous computing.* Genetic circuits are asynchronous because there is no clock controlling when each layer of the computation is performed [52]. Delays in the progression of the signal can lead to faults in the calculation. One approach to this problem is to build a genetic clock, and progress towards a robust oscillator [1,53] and counter [6] indicate that this may be possible. The design of asynchronous logic blocks that are robust to faults is also an active area of research, especially with respect to minimizing power requirements [54–56]. The principles from this work could be applied to the construction of integrated genetic logic, although unlike simple logic minimization, there has been historically a lack of CAD tools for asynchronous circuit design [9,57]. Recently, a system for asynchronous system design (RALA) has been developed and this framework is particularly applicable for biological systems [58].
- *Robustness to perturbations.* The wiring diagrams need to be robust with respect to perturbations due to fluctuations in molecules and environmental conditions [59]. In the design of electronic circuits, it has been noted that asynchronous circuits are also more robust [9,56]. Sole and co-workers enumerated digital systems composed of layered NAND gates and found that fault tolerance resulted from designs that contained a high degree of degeneracy, defined as occurring when multiple subsets of a circuit diagram that are structurally different perform the same function [60].

Biophysical Models of Part Function

The automated selection of genetic parts is one of the most difficult aspects of synthetic biology CAD [61]. Biophysical models can map the DNA sequence of a genetic part to its function. This enables the impact of an individual mutation to be calculated for a part and then coupled to a dynamical model to determine the effect on circuit function. In other words, they facilitate linking a simulation of reaction kinetics to physical DNA. Biophysical models are also useful in screening parts selected from a registry for context effects and can scan the

DNA sequence of programs to identify unintended functional sequences (*e.g.*, internal promoters or terminators) and correct potential errors.

Cellular regulation is highly redundant and there are often many ways to control the same parameter. For example, there are many options in controlling the concentration of a protein, from plasmid copy number to protein stability. This gives flexibility to the designer to take whatever route is convenient. In terms of design automation, it will be important to choose those routes that are the most reliable when modeled on the computer. For example, the interactions between RNA basepairs is relatively straightforward to calculate and efficient algorithms have been developed to calculate the folding of RNA secondary structure [62,63]. This has formed the core of predictive methods to capture how changes in the sequence affect the strength of ribosome binding sites [32,64,65], the efficiency of terminators [66–68], the shape of the transfer function of an shRNA switch controlled by a small molecule [69], and to create orthogonal rRNA-mRNA binding sequences [70]. In each of these models, the effect of arbitrary mutations on part function can be quantitatively calculated.

The activity of promoters have been successfully modeled using position weight matrixes (PWMs) [71,72]. These models predict the free energy by which a transcription factor binds to its operator sequence in a promoter. They make an additive assumption, where each base in the operator is assumed to be independent and a relatively small data set is used to parameterize a model. PWMs have suffered from high false positive rates, which has been a challenge in using them to scan genomes for promoters. However, they have been very successful at modeling how mutations affect the strength of a given promoter [71]. In this sense, they are ideal for applications where it is necessary to link how mutations will affect circuit dynamics by impacting promoter function.

Biophysical models are particularly applicable to a common problem that occurs when connecting genetic circuits in series (Figure 5). If the output of the first circuit is not in the correct dynamic range to trigger the second circuit, then the combined circuits will not function properly. One way to accomplish this is to vary the ribosome binding site (RBS) linking the circuits, typically by substituting different sites from a part library [73,74] or through random mutagenesis [27,75]. More recently, a biophysical model of RBS function was developed and shown to be able to identify *de novo* RBS sequences that can connect two previously-characterized circuits [32]. This approach could accelerate the automated connection and optimization of many genetic circuits.

Simulation and Design Environments

Kinetic simulations are commonly used to study the dynamics of molecules in the cells and there are a number of packages from systems biology that aid the modeling of signaling and regulatory networks [76–79]. A limitation in applying these tools to synthetic biology is the need to connect the output of a simulation to a specific design choice that is on the level of the DNA sequence, whether it be a mutation or the choice of a part. In the context of a compiler, the role of simulations would be to predict the dynamics of the assembled genetic circuits and serve as a debugging tool that would aid the user in writing the higher-level language (Figure 2).

A number of software tools have been developed to facilitate the selection and combination of parts gleaned from the Registry of Standard Biological Parts. For example, BioJade and SynBioSS enable the import of BioBricks along with descriptive parameters sets that can form the basis for simulations or algorithms for part selection [80,81]. Several algorithms enable the selection of parts to match an objective function that describes a desired circuit, such as an oscillator or toggle switch or logic gate [59,82]. All of these simulation packages

suffer from a lack of parts and circuits whose kinetics are well-documented and easily imported.

An underlying assumption of this approach is that parts will quantitatively identical kinetic parameters when measured individually or in the context of different genetic programs. There are several elegant examples where this assumption has been shown to hold, including in the construction of AND gates [4], toggle switches, and feedforward loops [83] through the assembly of different permutations of underlying parts. In our work with the edge detector, we found that the behavior of the program could be explained based on the component circuits [4]. However, it is not clear how often this is true and there are many published (as well as unpublished!) reports where the final behavior could not be predicted based on the components [33,84,85]. How to quantitatively measure and report this information is still an open question.

Genetic circuits usually consist of small numbers of molecules, and stochastic effects can dominate. There has been much work performed in the area of stochastic modeling and this has been applied to understanding natural and synthetic genetic networks [4]. A challenge is to convert the results of a stochastic simulation to a design choice; for example, the selection of a promoter. Noise can be controlled through the position of an operator in a promoter [85], and it is influenced by the number of layers in a genetic program [86]. In a theoretical study, Hasty and co-workers demonstrated that the variability of the circuit response can be tuned by independently controlling the transcription and translation rates and DNA copy number [87]. Cytometry data is the most common form of data produced when characterized by a genetic circuit and this contains significant and often unutilized information about the variability in the circuits. Munsky et al. have developed a method that enables the full cytometry distributions to be used to parameterize a genetic circuit, including its stochastic behavior [88].

Optimizing DNA Assembly and Experimental Design

Once the DNA sequence is debugged on the computer, it needs to be constructed. This can either be by automated DNA synthesis or by the robotic assembly of parts. Despite the declining cost of DNA synthesis, the ability to automatically assemble parts using BioBricks and related cloning strategies will be an advantage when it is desired to make many combinatorial variants of a genetic system. This is useful when there is potential degeneracy in the design, allowing multiple constructs to be tested for function. Often, only a subset of the permutations function as expected.

Grammars have been modified to include information for assembly [40]. The inclusion of rules can significantly reduce the construction cost by reducing the number of variants that need to be assembled [38]. This can be further coupled to kinetic simulations to reduce the permutations to those that are likely to perform a given circuit function. For example, Peccoud and co-workers have applied grammars to permutations of the toggle switch in order to identify variants that are likely to perform the correct function [37]. After rules and simulations constrain the set of permutations, a path to fabrication needs to be determined. Densmore and co-workers have developed algorithmic methods to reduce the path length in the number of assembly steps when constructing a genetic system [8]. The same approach can be applied to reducing the cloning steps when using the same set of underlying parts to assemble many systems.

Methods from the statistical design of experiments (DOE) could also be integrated into a CAD program [89,90]. Of particular interest is the design of sequential experiments, where the design of the next round of DNA constructs is influenced by the results from the previous round. This has been applied to highly-dimensional problems, such as the

optimization of fermentation, where there are many variables [pH, feed rate, oxygen, media, etc] that need to be optimized. In the context of genetic programs, the rigorous application of DOE methods would randomize an initial set of constructs. After the constructs were measured for the desired function, this would be fed back to the algorithm to identify the next set of constructs. The process is iterated until the function is optimized. This is a particularly powerful approach when coupled with optimization algorithm, such as Nelder-Mead or ant colonization [91].

IV. Conclusions

Genetic engineering is moving towards becoming an information science. The model of storing and distributing genetic material is slowly losing relevance. It is routine to outsource the task of constructing DNA from the designer to synthesis facilities. This has created a strong need for computer aided design programs that are able to facilitate the organization and construction of large projects. Once the parts are experimentally characterized, it is unnecessary to distribute the DNA. Rather, the CAD program can access the sequence and function information to design a genetic program. The sequence information corresponding to the desired program is sent to a synthesis/assembly facility for construction.

Existing genetic circuits have not been designed to be sufficiently robust to be automatically connected using CAD. These circuits and the genetic parts on which they are based need to be constructed specifically with the intent of coupling with a CAD program. One of the key problems is that there simply are not enough robust circuits to feed into a CAD program. There are significant efforts underway to develop computational methodologies specifically to create the orthogonal parts that would underlie such programs [70,92,93]. What is particularly needed are sets of orthogonal transcription factors that bind to different operator sequences. It is the co-development of fundamental circuits with the computational algorithms to assemble them that will allow us to move towards the vision of being able to program living cells.

Acknowledgments

The authors thank Ron Weiss (MIT), Rahul Sarpeshkar (MIT), Alan Mishchenko (UC-Berkeley), Jean Peccoud (VPI), Costas Maranas (Penn State), and Douglas Densmore (BU) for helpful discussions. CAV is supported by Life Technologies, ONR, Packard Foundation, NIH, NSF (synBERC: Synthetic Biology Engineering Research Center, www.synberg.org) and a Sandpit on Synthetic Biology hosted by EPSRC/NSF.

VII. References

1. Stricker J, Cookson S, Bennett MR, Mather WH, Tsimring LS, Hasty J. A fast, robust and tunable synthetic gene oscillator. *Nature*. 2008; 456:516–519. [PubMed: 18971928]
2. Ham TS, Lee SK, Keasling JD, Arkin AP. A tightly regulated inducible expression system utilizing the fim inversion recombination switch. *Biotech Bioeng*. 2006; 94:1–4.
3. Voigt CA. Genetic parts to program bacteria. *Curr Opin Biotech*. 2006; 17:548–557. [PubMed: 16978856]
4. Salis H, Kaznessis YN. Computer-aided design of modular protein devices: Boolean AND gene activation. *Physical Biology*. 2006; 3:395–310.
5. Lou C, Liu Xili, Ni M, Huang Y, Huang Q, Huang L, Jiang L, Lu D, Wang M, Liu C, Chen D, Chen C, Chen X, Yang L, Ma H, Chen J, Ouyang Q. Synthesizing a novel genetic sequential logic circuit: a push-on push-off switch. *Molecular Systems Biology*. 2009; 6:1–11.
- 6**. Friedland AE, Lu TK, Wang X, Shi D, Church G, Collins JJ. Synthetic gene networks that count. *Science*. 2009; 324:1199–1202. Genetic programs based on DNA recombinases and riboswitches are implemented that respond after counting a series of pulses. [PubMed: 19478183]

7. Balagadde FK, Song H, Ozaki J, Collins CH, Barnet M, Arnold FH, Quake SR, You L. A synthetic *Escherichia coli* predator-prey system. *Molecular Systems Biology*. 2008; 4:1–8.
8. Czar MJ, Anderson JC, Bader JS, Peccoud J. Gene synthesis demystified. *Trends in Biotechnology*. 2008; 27:64–72.
- 9*. Gibson DG, Young L, Chuang R-Y, Venter JC, Hutchinson CA, Smith HO. Enzymatic assembly of DNA molecules up to several hundred kilobases. *Nature Methods*. 2009; 6:343–345. A method is presented for the highly efficient single reaction assembly of many genetic parts. This eliminates the need for restriction enzymes in combining parts. [PubMed: 19363495]
10. Shetty RP, Endy D, Knight TF. Engineering BioBrick vectors from BioBrick parts. *J Biol Eng*. 2008; 2:1–12. [PubMed: 18271947]
11. Gibson DG, Benders GA, Axelrod KC, Zaveri J, Algire MA, Moodle M, Montague MG, Venter JC, Smith HO, Hutchison CA. One-step assembly in yeast of 25 overlapping DNA fragments to form a complete synthetic *Mycoplasma genitalium* genome. *Proc Natl Acad Sci USA*. 2008; 105:20404–20409. [PubMed: 19073939]
12. Gibson DG, Glass JI, Lartigue C, Noskov VN, Chuang RY, Algire MA, Benders GA, Montague MG, Ma L, Moodie MM, Merryman C, Vashee S, Krishnakumar R, Assad-Garcia N, Andrews-Pfannkoch C, Denisova EA, Young L, Qi ZQ, Segall-Shapiro TH, Calvey CH, Parmar PP, Hutchinson CA, Smith HO, Venter JC. Creation of a bacterial cell controlled by a chemically synthesized genome. *Science*. 2010:1–8.
13. Wang HH, Isaacs FJ, Carr PA, Sun ZZ, Xu G, Forest CR, Church GM. Programming cells by multiplex genome engineering and accelerated evolution. *Nature*. 2009; 460:894–898. [PubMed: 19633652]
14. Purnick PE, Weiss R. The second wave of synthetic biology: from modules to systems. *Nature Reviews*. 2009; 10:410–421. [14] A very interesting review that outlines the difficulties that underlie increasing the complexity of genetic programs.
15. Cho B-K, Charusanti P, Herrgard MJ, Palsson BO. Microbial regulatory and metabolic networks. *Curr Opin Biotech*. 2007; 18:360–364. [PubMed: 17719767]
16. Richardson SM, Nunley PW, Yarrington RM, Boeke JD, Bader JS. GeneDesign 3.0 is an updated synthetic biology toolkit. *Nucleic Acids Res*. 2010; 38:2603–2606. [PubMed: 20211837]
17. Villalobos A, Ness JE, Gustafsson C, Minshull J, Govindarajan S. Gene Designer: a synthetic biology tool for constructing artificial DNA segments. *BMC Bioinformatics*. 2006; 7:285. [PubMed: 16756672]
- 18*. Sarpeshkar, R. *Ultra Low Power Bioelectronics*. Cambridge, MA: Cambridge University Press; 2010. Some outstanding discussions comparing digital and analog circuits and the interpretation of genetic regulation from the perspective of electrical engineering. Read Chapters 1, 2, 7, 9, and 22–24
19. Ramalingam KI, Tomshine JR, Maynard JA, Kaznessis YN. Forward engineering of synthetic biological AND gates. *Biochem Eng J*. 2009; 47:38–47.
20. Kinkhabwala A, Guet CC. Uncovering cis regulatory codes using synthetic promoter shuffling. *PLOS One*. 2008; 3:1–10.
21. Cox RS, Surette M, Elowitz MB. Reprogramming gene expression with combinatorial promoters. *Molecular Systems Biology*. 2007; 3:1–11.
22. Rackham O, Chin JW. Synthesizing cellular networks from evolved ribosome-mRNA pairs. *Biochem Soc Trans*. 2006; 34:328–329. [PubMed: 16545106]
23. Kramer BP, Fischer M, Fussenegger M. Semi-synthetic mammalian gene regulatory networks. *Metabolic Engineering*. 2005; 7:241–250. [PubMed: 16140238]
24. Benenson Y. RNA-based computation in live cells. *Curr Opin Biotech*. 2009; 20:471–478. [PubMed: 19720518]
- 25*. Sharma V, Nomura Y, Yokobayashi Y. Engineering complex riboswitch regulation by dual genetic selection. *J Amer Chem Soc*. 2008; 130:16310–16315. A selection method is proposed that optimizes genetic AND/NAND gates composed of riboswitches that respond to small molecules. This approach is broadly useful in optimizing circuit function and creating orthogonal gates. [PubMed: 18998646]

26. Dueber JE, Yeh BJ, Chak K, Lim WA. Reprogramming control of an allosteric signaling switch through modular recombination. *Science*. 2003; 301:1904–1908. [PubMed: 14512628]
27. Anderson JC, Voigt CA, Arkin AP. Environmental signal integration by a modular AND gate. *Molecular Systems Biology*. 2007; 3:1–8.
28. Rinaudo K, Bleris L, Maddamsetti R, Subramanian S, Weiss R, Benenson Y. A universal RNAi-based logic evaluator that operates in mammalian cells. *Nature Biotechnology*. 2007; 25:795–801.
29. Buchler NE, Gerland U, Hwa T. On schemes of combinatorial transcription logic. *Proc Natl Acad Sci USA*. 2003; 100:5136–5141.
30. Yokobayashi Y, Weiss R, Arnold FH. Directed evolution of a genetic circuit. *Proc Natl Acad Sci USA*. 2002; 99:16587–16591. [PubMed: 12451174]
31. Canton B, Labno A, Endy D. Refinement and standardization of synthetic biological parts and devices. *Nature Biotechnology*. 2008; 26:787–793.
32. Salis HM, Mirsky EA, Voigt CA. Automated design of synthetic ribosome binding sites to control protein expression. *Nature Biotechnology*. 2009; 27:946–951.
- 33*. Tan C, Marguet P, You LC. Emergent bistability by a growth-modulating positive feedback circuit. *Nature Chemical Biology*. 2009; 5:842–848. A positive feedback loop constructed using T7 is shown to impact host processes, which then changes the dynamics of the circuit.
34. Hooshangi S, Thiberge S, Weiss R. Ultrasensitivity and noise propagation in a synthetic transcriptional cascade. *Proc Natl Acad Sci USA*. 2005; 102:3581–3586. [PubMed: 15738412]
35. Groban ES, Clarke EJ, Salis HM, Miller SM, Voigt CA. Kinetic buffering cross talk between bacterial two-component systems. *J Mol Biol*. 2009; 390:380–393. [PubMed: 19445950]
36. Cai Y, Hartnett B, Gustafsson C, Peccoud J. A syntactic model to design and verify synthetic genetic constructs derived from standard biological parts. *Bioinformatics*. 2007; 23:2760–2767. [PubMed: 17804435]
- 37*. Cai Y, Lux MW, Adam L, Peccoud J. Modeling structure-function relationships in synthetic DNA sequences using attribute grammars. *PLOS Computational Biology*. 2009; 5:1–10. A description of a grammar that captures synthetic systems. The grammar was used to constrain the computational enumeration of toggle switches by varying the underlying genetic parts.
38. Densmore D, Kittleson JT, Bilitchenko L, Liu A, Anderson JC. Rule based constraints for the construction of genetic devices. 2010
39. Czar MJ, Cai Y, Peccoud J. Writing DNA with GenoCAD. *Nucleic Acids Research*. 2009; 37:W40–W47. [PubMed: 19429897]
40. Cai Y, Wilson ML, Peccoud J. GenoCAD for iGEM: a grammatical approach to the design of standard-compliant constructs. *Nucleic Acids Research*. 2010; 38:2637–2644. [PubMed: 20167639]
- 41*. Bilitchenko L, Liu A, Chung S, Weeding E, Xia B, Leguia M, Anderson JC, Densmore D. Eugene - A domain specific language for specifying and constraining synthetic biological parts, devices, and systems. *PLOS One*. 2010 Eugene provides a vocabulary and grammar for the description of.
42. Goler JA, Bramlett BW, Peccoud J. Genetic design: rising above the sequence. *Trends in Biotechnology*. 2008; 26:538–544. [PubMed: 18687496]
43. Arp R, Smith B. Function, role, and disposition in basic formal ontology. *Nature Preceedings*. 2008:1–4. hdl:10101/npre.2008.1941.1.
44. Shah NH, Jonquet C, Chiang AP, Butte AJ, Chen R, Musen MA. Ontology-driven indexing of public datasets for translational bioinformatics. *BMC Bioinformatics*. 2009; 10:S1. [PubMed: 19208184]
45. Jiang J, Hui C. Hedgehog signaling in development and cancer. *Dev Cell*. 2008; 15:801–812. [PubMed: 19081070]
46. Clevers H. Wnt/beta-catenin signaling in development and disease. *Cell*. 2006; 127:469–480. [PubMed: 17081971]
47. Smith B, Ashburner M, Rosse C, Bard J, Bug W, Ceusters W, Goldberg LJ, Eilbeck K, Ireland A, Mungall CJ, Rocca-Serra P, Ruttenberg A, Sansone S-A, Scheuermann RH, Shah N, Whetzel PL, Lewis S. the OBI Consortium Leontis N. The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature Biotechnology*. 2007; 25:1251–1255.

48. Xiang Z, Courtot M, Brinkman RR, Rutenberg A, He Y. OntoFox: web-based support for ontology reuse. *BMC Research Notes*. 2010; 3:175. [PubMed: 20569493]
49. Rudell, RL. Multiple-Valued Logic Minimization for PLA Synthesis. UC-Berkeley; Berkeley, CA: 1986.
50. Mishchenko, A. ABC A system for sequential synthesis and verification. 2010. Available from: <http://www.eecs.berkeley.edu/~alanmi/abc/>
- 51*. Wu Y, Frey D, Lungu OI, Jaehrig A, Schlichting I, Kuhlman B, Hahn KM. A genetically encoded photoactivatable Rac controls the motility of living cells. *Nature*. 2009; 461:104–108. A simple graphic interface to minimize logic operations, apply constraints, and display the circuit diagram. The minimization procedure is based on the ESPRESSO algorithm. [PubMed: 19693014]
52. Seitz, CL. Introduction to VLSI systems. C.M.a.L. Conway; 1979.
- 53**. Danino T, Mondragon-Palomino O, Tsimring L, Hasty J. A synchronized quorum of genetic clocks. *Nature*. 2010; 463:326–329. A robust genetic oscillator is connected to cell-cell communication to synchronize cells within a population. [PubMed: 20090747]
54. Cortadella J, Kondratyev A, Lavago L, Sotiriou CP. Desynchronization: Synthesis of asynchronous circuits from synchronous specification. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. 2006; 25:1904–1921.
55. Manohar, R. Reconfigurable asynchronous logic. *IEEE Custom Integrated Circuits Conference*; 2006.
56. Rahbaran B, Steininger A. Is asynchronous logic more robust than synchronous logic? *IEEE Transactions on Dependable and Secure Computing*. 2009; 6:282–294.
57. Josephs, MB.; Furey, DP. A programming approach to the design of asynchronous logic blocks. In: JC, et al., editors. *Concurrency and Hardware Design*. Springer-Verlag; Berlin: 2002. p. 33–60.
- 58**. Gershenfeld, N.; Dalrymple, D.; Chen, K.; Knaian, A.; Green, F.; Demaine, ED.; Greenwald, S.; Schmidt-Nielsen, P. *POPL*. Madrid; Spain: 2010. Reconfigurable Asynchronous Logic Automata (RALA). A cell-based CAD program is developed for the automated design of asynchronous circuits. The operations and framework are simple and should be directly applicable to genetic circuits
59. Rodrigo G, Jaramillo A>. Computational design of digital and memory biological devices. *Syst Synth Biol*. 2008
- 60*. Macia J, Sole RV. Distributed robustness in cellular networks: insights from synthetic evolved circuits. *J R Soc Interface*. 2009; 6:393–400. Various wiring diagrams consisting of NAND gates are analyzed for fault tolerance. The measures of robustness applied here could be broadly applied to designing integrated genetic circuits. [PubMed: 18796402]
61. Kazenesis YN. Models for synthetic biology. *BMC Systems Biology*. 2007; 1:1–4. [PubMed: 17408505]
62. Dirks RM, Bois JS, Schaeffer JM, Winfree E, Pierce NA. Thermodynamics analysis of interacting nucleic acid strands. *SIAM Rev*. 2007; 49:65–68.
63. Markham NR, Zuker M. UNAFold - Software for nucleic acid folding and hybridization. *Methods in Molecular Biology*. 2008; II:3–31. Structure, Function, and Applications. [PubMed: 18712296]
64. De Smit MH, van Duin J. Secondary structure of the ribosome binding site determines translational efficiency: A quantitative analysis. *Proc Natl Acad Sci USA*. 1990; 87:7668–7672. [PubMed: 2217199]
65. Na D, Lee S, Lee D. Mathematical modeling of translocation initiation for the estimation of its efficiency to computationally design mRNA sequences with desired expression levels in prokaryotes. *BMC Systems Biology*. 2010; 4:71. [PubMed: 20504310]
66. Carafa, Yd; Brody, E.; Thermes, C. Prediction of rho-independent Escherichia coli transcription terminators. *J Mol Biol*. 1990; 216:835–858. [PubMed: 1702475]
67. Yager TD, von Hippel PH. A thermodynamic analysis of RNA transcript elongation and termination in Escherichia coli. *Biochemistry*. 1991; 30:1097–1118. [PubMed: 1703438]
- 68*. Kingsford CL, Ayanbule K, Salzberg SL. Rapid, accurate, computational discovery of Rho-independent transcription terminators illuminates their relationship to DNA uptake. *Genome Biology*. 2007; 8:R22. The authors describe a computational method to scan genomes for

terminators (TransTermHP). This is also useful to scan synthetic designs for inadvertent terminator sequences. [PubMed: 17313685]

69. Beisel CL, Bayer TS, Hoff KG, Smolke CD. Model-guided design of ligand-regulated RNAi for programmable control of gene expression. *Molecular Systems Biology*. 2008; 4:1–12.
70. Chubiz LM, Rao CV. Computational design of orthogonal ribosomes. *Nucleic Acids Research*. 2008; 36:4038–4046. [PubMed: 18522973]
- 71**. Rhodius VA, Mutalik VK. Predicting strength and function for promoters of the *Escherichia coli* alternative sigma factor, sigma_E. *Proc Natl Acad Sci USA*. 2010; 107:2854–2859. A predictive model of promoter strength is developed using position weighted matrixes and data from a comprehensive set of promoters activated by the σ_E sigma factor. They also demonstrate equivalence between *in vitro* and *in vivo* measurement methods. This approach could be broadly applied to modeling of the impact of basepair changes on promoter strength. [PubMed: 20133665]
72. Li H, Rhodius V, Gross C, Siggia ED. Identification of the binding sites of regulatory proteins in bacterial genomes. *Proc Natl Acad Sci USA*. 2002; 99:11772–11777. [PubMed: 12181488]
73. Tabor JJ, Salis HM, Simpson ZB, Chevalier AA, Levskaya A, Marcotte EM, Voigt CA, Ellington AD. A synthetic genetic edge detector program. *Cell*. 2009; 137:1272–1281. [PubMed: 19563759]
74. Temme K, Salis H, Tullman-Ercek D, Levskaya A, Hong SH, Voigt CA. Induction and relaxation dynamics of the regulatory network controlling the type III secretion system encoded within *Salmonella* pathogenicity island 1. *J Mol Biol*. 2008; 377:47–61. [PubMed: 18242639]
75. Anderson JC, Clarke EJ, Arkin AP, Voigt CA. Environmentally controlled invasion of cancer cells by engineered bacteria. *J Mol Biol*. 2006; 355:619–627. [PubMed: 16330045]
76. Andrews SS, Addy NJ, Brent R, Arkin AP. Detailed simulations of cell biology with Smoldyn 2.1. *PLOS Computational Biology*. 2010; 6:1–10.
77. Tomita M, Hashimoto K, Takahashi K, Shimizu TS, Matsuzaki Y, Miyoshi F, Saito K, Tanida S, Yugi K, Venter JC, Hutchison CA. E-Cell: software environment for whole-cell simulation. *Bioinformatics*. 1999; 15:72–84. [PubMed: 10068694]
78. Casanova H, Berman F, Bartol T, Gokcay E, Sejnowski T, Brinbaum A, Dongarra J, Miller M, Ellisman M, Faerman M, Obertelli G, Wolski R, Pomerantz S, Stiles J. The Virtual Instrument: Support for Grid-Enabled MCell Simulations. *Intl J of High Perf Comp App*. 2004; 18:3–17.
79. Ander M, Beltrao P, Di Ventura B, Ferkinghoff-Borg J, Foglierini M, Kaplan A, Lemerle C, Tomas-Oliveira I, Serrano L. SmartCell, a framework to simulate cellular processes that combines stochastic approximation with diffusion and localization: analysis of simple networks. *Sys Biol*. 2004; 1:129–138.
80. Hill AD, Tomshine JR, Weeding EMB, Sotiropoulos V, Kaznessis YN. SynBioSS: the synthetic biology modeling suite. *Bioinformatics*. 2008; 24:2551–2553. [PubMed: 18757873]
81. Goler, JA. Masters Thesis. Massachusetts Institute of Technology; Cambridge, MA: 2004.
82. Dasika MS, Maranas CD. OptCircuit: An optimization based method for computational design of genetic circuits. *BMC Systems Biology*. 2008; 2:2. [PubMed: 18173842]
- 83**. Ellis T, Wang X, Collins JJ. Diversity-based, model-guided construction of synthetic gene networks with predicted functions. *Nature Biotechnology*. 2009; 27:465–471. A parameterized set of parts was used to construct a large library of feedforward and toggle switch circuits that implement logic and time delay functions. Mathematical models of the combinations of parts accurately predicted the circuit dynamics.
84. Guet CC, Elowitz MB, Hsing W, Leibler S. Combinatorial synthesis of genetic networks. *Science*. 2002; 296:1466–1470. [PubMed: 12029133]
85. Murphy KF, Balazsi G, Collins JJ. Combinatorial promoter design for engineering noisy gene expression. *Proc Natl Acad USA*. 2007; 104:12726–12731.
86. Pedraza JM, van Oudenaarden A. Noise propagation in gene networks. *Science*. 2005; 307:1965–1969. [PubMed: 15790857]
87. Lu T, Ferry M, Weiss R, Hasty J. A molecular noise generator. *Physical Biology*. 2008; 8:1–8.
- 88**. Munsky B, Trinh B, Khammash M. Listening to the noise: random fluctuations reveal network parameters. *Molecular Systems Biology*. 2009; 5:1–7. A method is presented to fit cytometry data and use it estimate the parameters for a genetic circuit.

89. Bailey, RA. Design of Comparative Experiments. In: Gill, BDRR.; Ross, S.; Silverman, BW.; Stein, M., editors. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge: Cambridge University Press; 2008.
90. Atkinson, A.; Doney, A.; Tobias, R. Optimum Experimental Designs, with SAS. In: Atkinson, RJCAC.; Hand, DJ.; Pierce, DA.; Schervish, MJ.; Titterington, DM., editors. Oxford Statistical Science Series. Oxford: Oxford University Press; 2007.
91. Dorigo M, Birattari M, Stutzle T. Ant Colony Optimization. IEEE Computational Intelligence Magazine. 2006;28–49.
92. Mandell JG, Barbas CF. Zinc finger tools: custom DNA-binding domains for transcription factors and nucleases. Nucleic Acids Research. 2006; 34:W516–W523. [PubMed: 16845061]
- 93**. Desai TA, Rodionov DA, Gelfand MS, Alm EJ, Rao CV. Engineering transcription factors with novel DNA-binding specificity using comparative genomics. Nucleic Acids Research. 2009; 37:2493–2503. An approach using comparative genomics is used to create an orthogonal set of CRP mutants that bind to different DNA sequences. This approach could be used to create many classes of orthogonal transcription factors. [PubMed: 19264798]
94. Elowitz MB, Leibler S. A synthetic oscillatory network of transcriptional regulators. Nature. 2000; 403:335–338. [PubMed: 10659856]
- 95*. Densmore D, Hsiao THC, Kittleson JT, DeLoache W, Batten C, Anderson JC. Algorithms for automated DNA assembly. Nucleic Acids Research. 2010; 38:2607–2616. Graph searching algorithms are applied to identify the optimal path in assembling constructs using BioBrick and related cloning strategies. The method is particularly suited for robotic automation. [PubMed: 20335162]

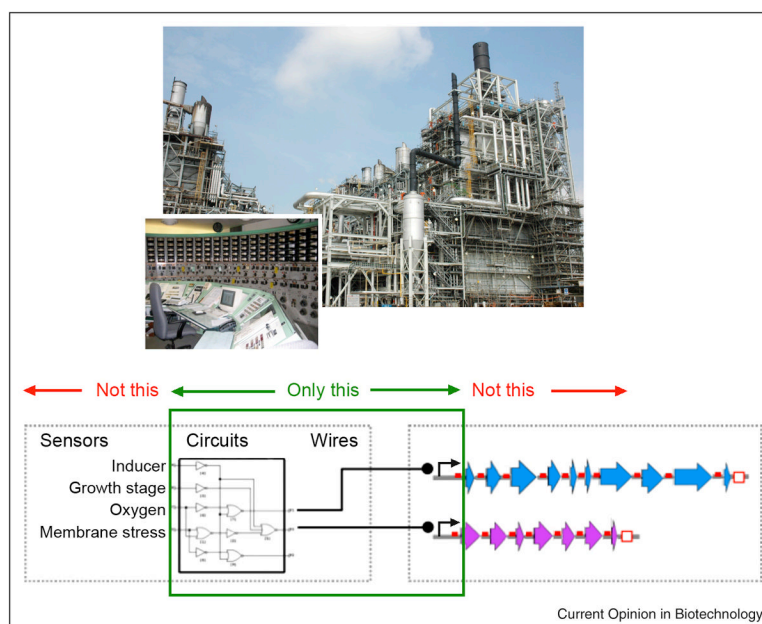


Figure 1. The compiler is focused on assembling the circuitry that links the inputs and outputs of a larger project

The inputs include genetic sensors that can respond to diverse signals, such as temperature, light, stress, or metabolites. The circuitry encodes the logic and dynamics. The output of the circuits control actuators, such as a metabolic pathway, the activation of cell-cell communication, or a stress response. The inputs and outputs tend to be problem-specific and the diversity of biological applications makes this difficult to encompass in a single simulation package. In contrast the circuitry can be reconfigured to build programs relevant to diverse problems in biotechnology.

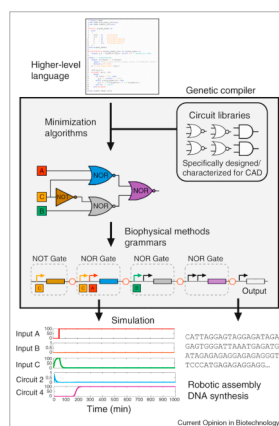


Figure 2. A Genetic Compiler

The compiler automatically converts a higher-level language to a DNA sequence. Ideally, the designer would be completely blind to these steps (gray box). Simulations aid the debugging of the program, but the debugging would occur within the higher-level language.



Figure 3. Semantics of genetic programs

The plasmid map (A) and Eugene code (B) for a genetic oscillator [94] is shown (B). The author of the Eugene code is Adam Liu [41].

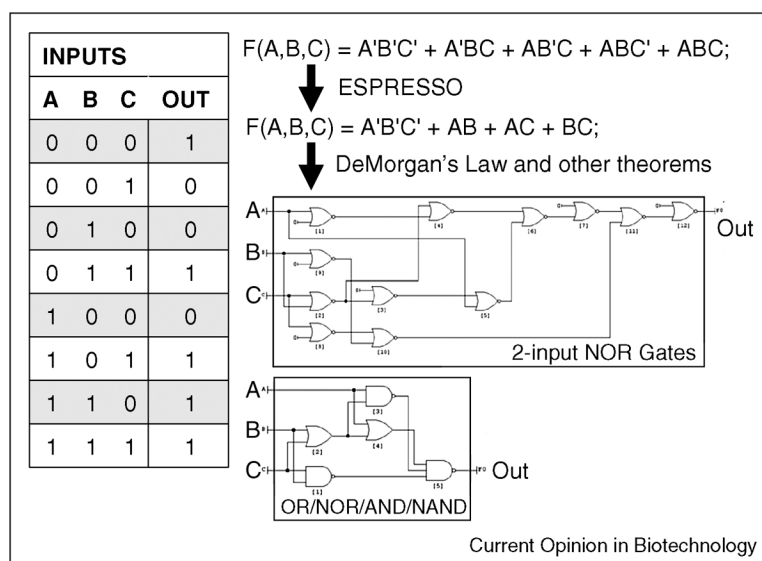


Figure 4. Automated program design using logic minimization algorithms

An example of a multi-input single-output truth table is shown. The truth table is converted to an equation F , which is a function of the four inputs (a, b, c, d). Each term corresponds to each row of the truth table where the output is 1 and the prime (') is shorthand for the NOT function. Logic minimization algorithms, such as ESPRESSO [49], can be used to simplify the full equation to its minimal form (simplest sum of products). This equation is then converted to circuit diagrams using programs such as Logic Friday [51], which can also implement constraints. For example, the large wiring diagram consists of only 2-input NOR gates (top), whereas the smaller wiring diagram was built allowing for multiple-input OR/NOR/AND/NAND gates. Biological constraints, such as gate availability and DNA size can then be applied to search for the optimal diagram.

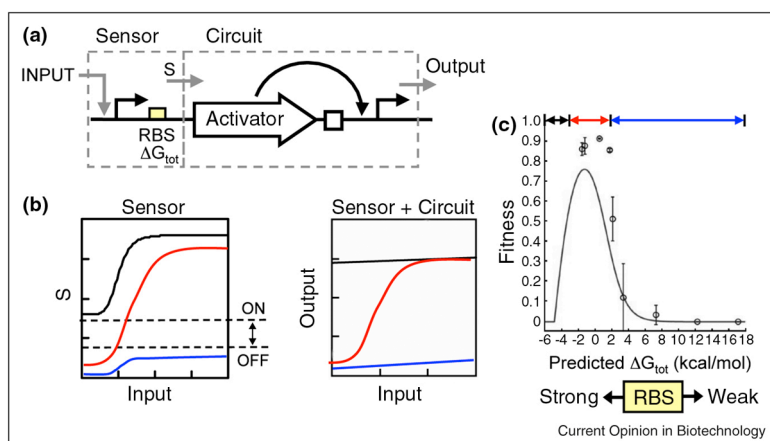


Figure 5. Connecting genetic circuits

(A) A simple program is shown consisting of a sensor and circuit. The sensor turns on a promoter in response to an INPUT. A RBS controls a translation initiation rate S that then serves as an input to the circuit. The circuit consists of an activator that turns on a promoter that serves as the output. (B) If the transfer function of the sensor is too low (blue) or the basal level is too high (black), then it will not cross the dynamic range to turn on the circuit (dotted lines). When the sensor output crosses that which is required to trigger the circuit, then the program is functional (red). (C) Experimental data is shown where the RBS is modified in the connection of a sensor to an AND gate. The solid line is the predicted fitness curve derived from the transfer functions of the sensor and circuit measured in isolation of the program.