Table of Contents

Chapter 1: Introduction to Biological Design Concepts

Suspendisse potenti.

1. Introduction

Over the last 40 years, dramatic improvements in biological measurement, computation, and genetic manipulation have largely elucidated the molecular basis for cellular and organismal function and behavior. It is now routine to sequence whole genomes and to measure expression of every gene and protein under various conditions. These innovations and others in imaging, mass spectroscopy, DNA synthesis and genetics allow scientists to rapidly identify and intervene in key mechanisms underlying disease, microbial survival, and the production of important biochemicals. Such endeavors have engendered a rapidly maturing set of technologies that promise a second revolution in engineering biology to solve societal problems in health, energy and the environment. Initial successes in this field include:

- Production of medically relevant proteins and small molecules
- Biosynthesis of important high-value chemicals, e.g. to replace petroleum based compounds
- Engineered crops with increased hardiness or nutritional value
- Engineered cells and viruses for medical therapy

To date, though, these solutions have been relatively few compared to the extensive accomplishments in electrical and mechanical engineering. To make the leap to a scaling engineering discipline, the technologies above must be organized into a formal design discipline supported by a robust infrastructure for computer-aided design and manufacture that support an array of advanced applications. Imagine for example:

- Treating cancer with viruses programmed to specifically recognize cancer cells for therapeutic applications
- Biosynthesizing any chemical that is physically realizable in a cell on demand
- Creating any tissue or organ by design, with programmed properties the individual host requires
- Designing a community of microbes to colonize the human gut and prevent infection, decrease inflammation, and balance nutrition
- Producing trustworthy organisms that clean our environment and support crop growth in the world's increasingly non-arable soils
- Engineering organisms that sustainably synthesize 3D nanostructured materials that can be used in optics, electronics, and construction

This book is about the fundamental techniques used to design and implement the subcellular biological networks that would allow these feats. Specifically, we need to engineer the pathways that control when and where biological molecules are produced (via transcription and translation), modified (for example, by phosphorylation and glycosylation), secreted, and degraded. If these processes can be controlled, a designer can affect key cellular behaviors such as feeding, biosynthesis, cell division, cell death, motility, and cell-cell signaling to reach the desired system formulation and outputs. The designer must also consider the interaction between the modified organisms and their environment. This overall approach is generally recognized as biochemical network engineering.

The central challenge, though, is transforming this cellular engineering into a true, scalable engineering discipline. Up to this point, most cellular engineering successes have resulted from herculean efforts to create a specific product, and the pieces used are therefore not necessarily compatible with other projects addressing other challenges. In other words, many engineers are working almost in a vacuum, such that they need to start almost every new project at the very beginning. To bring cellular engineering into compliance with the principles of other major engineering disciplines, we need to ensure scalability via large numbers of reliable small applications that can be built upon to produce large, complex biological systems that work correctly with minimal optimization required.

Biological engineering theory, manufacturing processes, and the physical elements from which systems are built are still being designed to work together as efficiently as they do in other disciplines. With this caveat in mind, in this book we will outline engineering and design approaches that have worked, examine their limitations, discuss the bottlenecks for creating complex and reliable biological systems, and identify areas where we believe there is the greatest potential for innovation and growth.

2. The Design Process

Design requires creating something we know we want via detailed descriptions of the problem to be solved and the solution implementation. A successful design will provide a clear, efficient route from a high-level description of the desired object and its goals (a formulation) to a solution via specification of a concrete implementation. This implementation must be based on a set of predictably and reliably operating primitive components connected by a standard manufacturing process. One of the chief goals of synthetic biology is to create such a formal framework, which does not yet exist in biological engineering.

The Design Process in Established Engineering Disciplines

Engineering disciplines have been hugely successful in part because of the ability to standardize inputs, outputs, and parts, and we aim to apply these principles to Synthetic Biology as well. We must acknowledge the potential difficulties this approach faces due to the inherently complex system within a cell or virus, inside of which we will be designing the majority of our systems, and the additional complexity introduced by their interaction with their environment. Much of this book will be dedicated to exploring and accounting for the resulting design challenges and uncertainties. Nonetheless, we will use the same design principles that have allowed other engineering pursuits to be so successful.

In most formal design disciplines, the requirements in a formulation are broken into subproblems that can be approached individually. This *hierarchical (top-down) decomposition* reduces the complexity of the overall problem by specifying loosely coupled subsystems, or modules. This approach promotes reusability of design elements, since independent modules can be redeployed in other applications, and allows the design work for each module to be distributed.

This initial high-level specification is the *system-level description,* which essentially provides a formal, unambiguous, comprehensible description of the desired system, including various inputs, outputs, and functions. The specific implementation is not included in this level of abstraction. To move toward implementation, the functions are converted to a *network-level description,* which specifies an abstract biological network that can carry out the desired activity.

Then, at the *physical level*, each element of the network is mapped to a possible physical instantiation.

Throughout the book we will often use analogies to the methods and practices in the electronics industry and while there are many truly analogous processes the metaphor is imperfect. Nonetheless, this field provides excellent examples of large-scale systems design enabled using hierarchical decomposition, as shown in the example below.

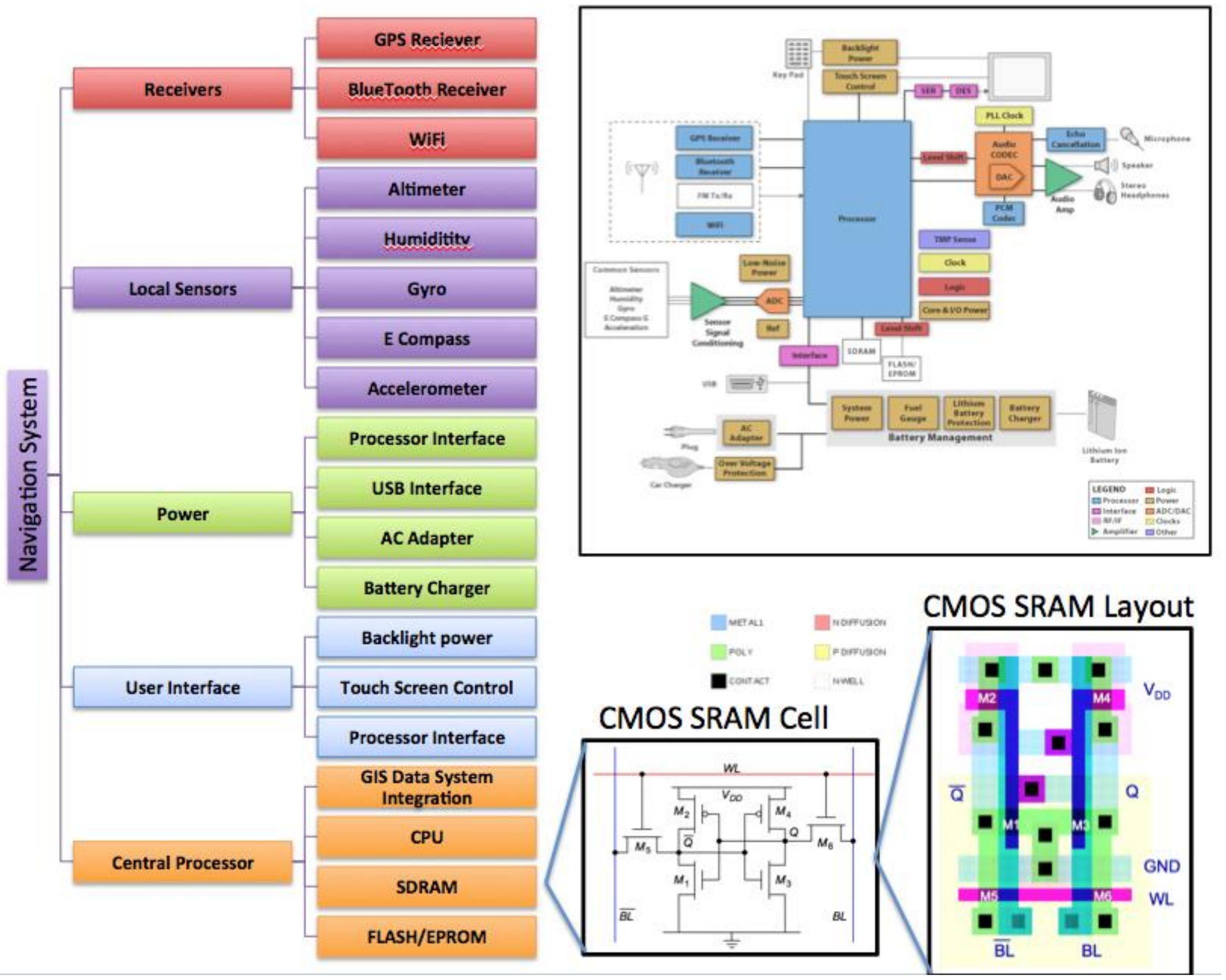


Figure 1. Hierarchical Decomposition of a GPS Navigation System with Block Network Diagram Inset.

For example, imagine a designer is sent the requirements for a navigation system. First, the designer can break the problem into a series of system functions that can each be individually optimized to achieve the goal. For example, the designer might break the problem into the need for a series of receivers, local sensors, power, user interface and central processing units. In a second step, the designer breaks these high-level functions into lower-level elements like Memory and Battery. Next, these elements are further decomposed into gate- and transistor-level designs. A network diagram links these modules together by their input/output dependencies in a logical layout (inset). This process is almost complete when the designer reaches the primitives for the design: reusable elementary parts or existing composite modules. For digital logic components, the low-level transistor design the goes through a process of floor-planning (physical layout). Other processes ensue for modules like the battery or display. The system is then optimized through modeling. Once achieved, the laid-out transistor-level design is sent for manufacture and the resultant device performance is evaluated against the desired goals.

Even in this mature, industrialized engineering field, multiple optimization cycles are generally necessary. Most system representations do not flawlessly transform a high-level specification into a perfectly functioning final physical implementation. Instead, human designers must frequently optimize the final system design and function through experience and experimentation, and after implementation, humans must debug the almost inevitable failures of the system and go through an iterative redesign process. Computational tools (so-called computer-aided design and manufacturing tools) can aid in this process; such programs are rather mature in electronic and mechanical engineering, but are just emerging in synthetic biology.

Design as Representation

To design a biological system one must represent *functionally* what you wish it to do, *structurally* how the system should accomplish this function using increasingly primitive components, and *physically* how one would instantiate these components. Ideally, one would have a formal language in which to state the desired high-level function. A compiler can then transform a program encoded in this high-level formal language into lower-level representations (possibly more than one), which would allow exploration of the validity of the design and design tradeoffs. Different representations of the design, at different levels of abstraction, could be used to explore different aspects of the design and its correctness in terms of achieving the desired high level goals. Ultimately, the design would have to be manifested in a particular physical implementation, and these would need to be consistent with the high level specification and work correctly within the biological constraints.

The basic process begins by defining the purpose, inputs, and outputs of the system. The designer must create a detailed, precise representation, for example a flow chart encapsulating the individual functional units to be assembled. These functional units must then be further refined in an iterative process until the designer reaches a level of detail that can be directly implemented using the primitive building blocks of our chosen technology - for example, the promoters, ribosome binding sites, and genes in the bacterium *Escherichia coli*.

In this book, our goal is to train the next generation of synthetic biologists to create the tools, primitive building blocks and manufacturing processes that will support a similar level of computer-aided design and manufacturing support found in other disciplines. That is, we wish to create a field where a suitably specified flow-chart encapsulating the desired logical, synthetic and electromechanical functions of a designed biological systems can be automatically translated into DNA to be synthesized, assembled and transplanted into a suitable host to implement these functions. Thus, our goal is to teach how to transform one representation of a biological design into another and to point to places wherein research may better enable the automatic translation among them.

High-level behavior to physical biological implementation

Our design begins with a high-level behavior or other phenotype we want our organism to exhibit. Through a series of decompositions into modular functions, we can implement our system via abstract biological networks that ultimately will be reduced to manufacturable pieces like known enzymes, promoters, small functional RNAs, and specific metabolites encoded in DNA (See Figure below).

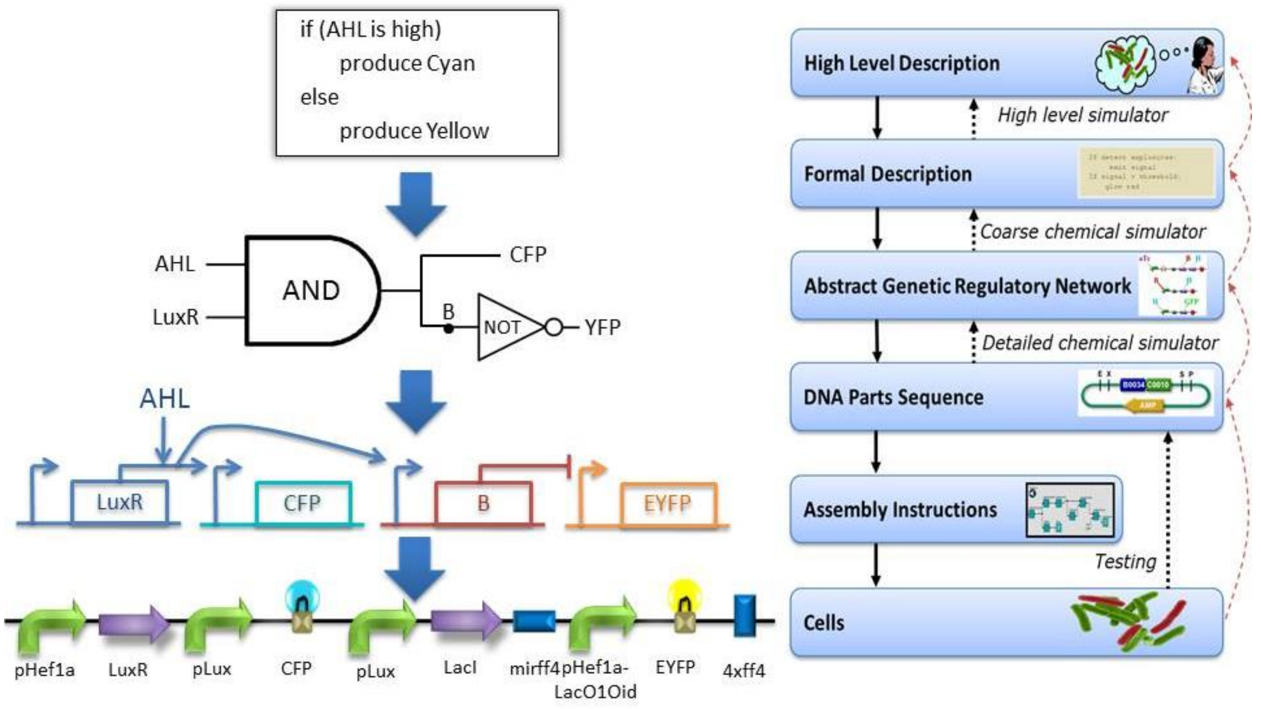


Figure 2: Mapping from high-level functional descriptions to gene networks with intermediate steps. In the high level program, AHL is a small molecule input, while Cyan and Yellow outputs can be implemented using fluorescent proteins (CFP and YFP respectively). LuxR is a protein that binds AHL and activates a corresponding pLux promoter, while B is an abstract transcriptional repressor.

In this book we will focus on the design principles of cellular networks with biological components - the cells themselves and the molecules that make them up - without detailed consideration of the process pipeline in which they might be inserted. We aim to elucidate a "basis set" of such componentry that satisfy requirements and constraints. These requirements and constraints might be chemical, such as solubility and thermostability, or they may relate to interoperativity, such as requiring that these parts be drawn from a family of well-modelled, compatible, homogeneously functioning elements with known properties of their composition into higher-order networks.

There will likely be other constraints that will affect the specification of which we are not aware at the time of the initial design; for example, what is necessary for cell survival in the reactor, in which growth stage will production occur, and many more things. Thus, the design process will include several series of iterations until the desired output is achieved.

3. Design in a Biological Environment

At a physical level, cellular networks are governed by molecules interacting asynchronously in a spatially inhomogeneous environment via non-linear and stochastic mechanisms. This inherent randomness makes it very difficult to fully map and understand natural systems. Nevertheless, there are a limited set of such mechanisms, and many can be abstracted to facilitate modeling and analysis. In addition, one sub-discipline developing in Synthetic Biology involves designing biological molecules and mechanisms such that the networks they create operate predictably with the endogenous host functions, making them easier to model and understand.

Despite these efforts, most designs will not produce the predicted output the first time they are implemented because of undescribed or poorly understood constraints. The design process will almost always be an iterative cycle: create a specification based on current knowledge, manufacture an object based on the specification, measure the object's behaviors and compare to the desired goals, and learn from the mistakes and successes to design a new specification. Good engineering reduces the time and cost of each cycle and the number of cycles until an effective object is created.

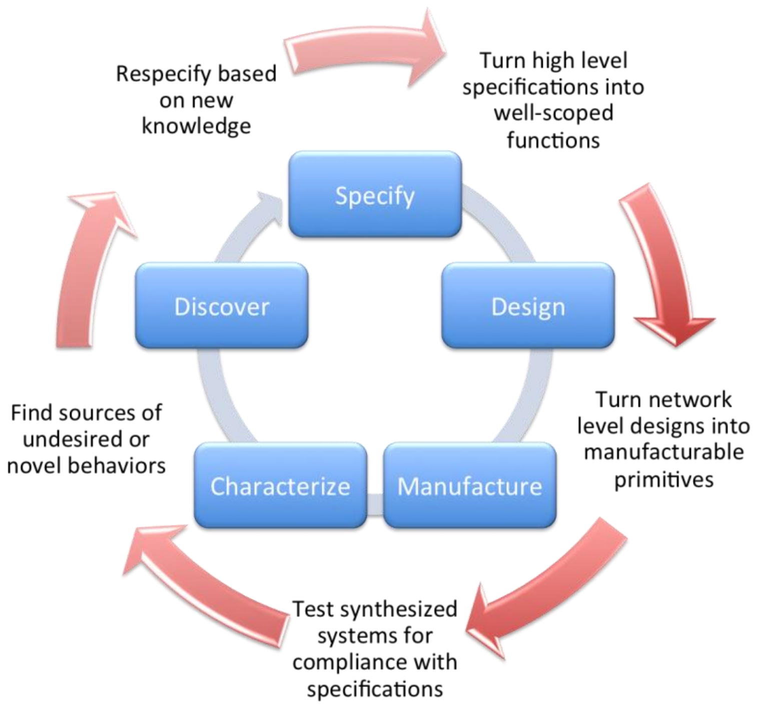

Respecify based on new knowledge
Turn high level specifications into well-scoped functions
Specify
Discover
Design
Turn network level designs into manufacturable primitives
Find sources of undesired or novel behaviors
Characterize
Manufacture
Test synthesized systems for compliance with specifications

Figure 3: The design, build, test and learn cycle.

Example: Expressing a protein to a particular level

Constitutive expression of an enzyme to a steady-state (constant) concentration per cell is one of the fundamental design goals in biological engineering. Achieving such constant expression levels can improve economically favorable production of a valuable small molecule, for example, but there are constraints on this goal arising from the practical considerations for the process and the limits of cellular physiology and biophysics.

A typical start to a high-level formulation for this specific design would be:

Goal: Reach a target cytoplasmic concentration of protein X of 200 proteins/cell (+/- 50 proteins/cell)
Constraints: Implement the system in *Escherichia coli* BW25113 (a particular strain of *E. coli*) and express the protein from the chromosome. Assume the culture will be grown in a 1 liter volume in a 37°C turbidostat (a continuous culture device) maintaining the cells in exponential growth and an optical density of 0.3 in a rich, defined, MOPS media.

There is a fair amount of technical information here in addition to the target protein concentration. There are requirements on host, temperature, growth phase, what the cell is fed, and that the target protein concentration is met within a certain error.

Despite these many specifications, the relative simplicity of the partial formulation above means a top-down decomposition is not entirely necessary. Everything has been constrained except for the genetic elements that express the protein.

The top-level module is the genetic cassette that expresses the protein. We may decompose this module by considering a simple model of gene expression described by four key parameters:

- the transcription initiation rate, xi
- the translation initiation rate, ti
- the mRNA degradation rate, md
- the protein degradation rate, pd

These parameters are affected by temperature and in some cases by the media composition.

In this simplified description, the mean steady-state level of protein is (xi/md)*(ti/pd). (See Chapters 2 and 3 for the derivation of this expression.) There are many possible biologically realizable combinations of parameters that yield the correct steady state levels, so the designer must choose the best combination of rates for the system. These choices can affect other performance measures of the system, such as temporal response and variability, which must be taken into account. An ideal modular decomposition of the gene expression cassette might specify modules that independently control each parameter, since these would encapsulate conceptually (if not physically) independent factors affecting expression.

To complete the above formulation, we have to identify the primitives from which our object will be built. For gene expression in *E. coli*, ideally we would have a set of well-characterized part families:

- promoters, different instances of which might initiate transcription at different rates (xi)
- sequence variants for the region between the promoter and the coding sequence for the protein X, which largely affect translation initiation and mRNA degradation (ti, md)
- variants of the coding sequence for protein X which may encode different protein degradation properties (pd)
- a transcriptional terminator

Well-characterized parts in this case means there are models of their behavior that predict their activity (the rate constants above) given both the context variables and their configuration with other parts.

Once the formulation is complete, a specification outlines the modules above and uses the part models to select the elements that are predicted to meet the formulation goals; in other words, it chooses a concrete implementation for each module. In fact, it is likely there will be a set of possible specifications, each with a different set of module instances, all of which meet the goal of 200 molecules/cell.

With a set of complete specifications in hand, they must be experimentally characterized and evaluated for how well they meet the goals of the formulation. In this case, because of the constraint on the variation of the protein concentration, it would be necessary to characterize the result by measuring the expression of the protein in single cells over time to see if the variability criterion is met. (This is no easy feat in itself.)

If the requirements are met, we are done. If not, diagnosis of the failure might identify a new constraint to be added to the formulation or a constraint that cannot be physically met and therefore must be modified. It is possible that the source of error cannot be identified. In all cases, a new specification must be created. In the former cases, the formulation is changed informed by the new data. In the latter case, new variants consistent with the original formulation are specified. Thence the cycle begins again.

4. Biological Systems Engineering

There are two main approaches for implementing a design: top-down decomposition and bottom-up assembly. We will use the example of a tissue homeostasis system to illustrate these two approaches.

Top-down decomposition

Biologists will be familiar with the reductionist paradigm, whereby a system is thought to be best understood by elucidating the elementary mechanisms of its parts and their interactions. For example, one understands a cell’s metabolism by first understanding the transformation of various carbon and nitrogen sources to biomass and outputs like carbon dioxide and water. These processes are then assigned to pathways (e.g. glycolysis, tryptophan biosynthesis), which are further decomposed into elementary chemical transformations catalyzed by enzymes encoded by genes. This hierarchical decomposition of the cell is necessary for understanding and ultimately for designing new functions. How a cell uses sugar and oxygen to propagate itself is at first opaque, but we gain increasing understanding as we refine our functional description into increasingly primitive function. At the bottom, we are left with an explicit physical mechanism for a particular reaction, which is easier to comprehend than the function of the whole.

In engineering, we explicitly design systems so that a complex function can be broken down into smaller pieces that can be implemented using a divide-and-conquer strategy. A good design allows each such operation to be designed without detailed consideration of the implementation of the others, except for the interactions between them.

Motivating Example: Artificial Tissue Homeostasis

In this section, we introduce a case study to illustrate this top-down design process: a relatively complex system to control tissue homeostasis. Tissue homeostasis is broadly defined as the process of balancing growth, death, and differentiation of multiple cell-types within a multicellular community, and represents an important class of problems in biology. Understanding and controlling tissue homeostasis is fundamental to the success of a wide range of tissue engineering goals, and the ability to create and analyze such a system may provide insight into mechanisms of endogenous tissue homeostasis and its misregulation in diseases such as cancer and diabetes.

For example, misregulation of tissue homeostasis plays a central role in Type I diabetes, in which natural populations of insulin-producing beta-cells are destroyed due to autoimmune defects. Automated mechanical systems have been proposed for insulin control in diabetes but still face significant challenges including long-term efficacy. Stem cell and beta-cell transplantations have also been studied as possible solutions, but recent results suggest that the transplanted cells fail to maintain homeostasis and become either tumorigenic or depleted within months. Our goal in this section is to begin with a high level specification and examine various possible system designs at increasing levels of biological detail. Ultimately, we hope to be able to design and implement an artificial tissue homeostasis system that would be therapeutically relevant for diseases such as Type I diabetes.

We begin with the highest level representation: Indefinitely maintain a population of cells committed to differentiate to a particular terminal cell type. Note that we have already made at least one design choice: we wish to maintain a steady population of cells committed to differentiate, as opposed to a population of already differentiated cells. This design choice, which relies on the assumption that the differentiation process is relatively robust, making the committed cell population a good proxy for the terminally differentiated cell population, creates a more sensitive, responsive system than would responding directly to the already differentiated cells. The actual differentiation process can take days to weeks, while the process of simply committing to differentiate can occur quite quickly. Therefore, if the design relied on the population of already differentiated cells, the long delay between the observation that additional cells need to differentiate and the time when these cells actually complete the differentiation process would result in a high latency negative feedback control loop, which would likely be unstable and difficult to regulate. On the other hand, when cells commit to differentiate, which occurs quite quickly, this information can be broadcast to the rest of the population, or at least the cells' neighbors, creating a much faster negative feedback control loop.

We begin with this population-level specification, but we will ultimately need to translate this into a representation of genetic circuitry inside a cell. Toward this end, a slightly lower-level specification is the following:

1) Differentiate individuals of an implanted stem cell population to a final committed cell fate at a rate sufficient to maintain the population given the committed cell's death rate
2) Maintain a population of undifferentiated and uncommitted stem cells sufficient to provide a steady-stream of committed cells at the rate required by (1) given the stem cells' death rate

The flowchart in Figure 4 (below) shows a population-level translation of the high-level design into a lower-level flow chart. The flowchart shows we need five control units:

- sensor for the number of uncommitted stem cells
- sensor for the number of committed cells
- stem cell growth trigger (activated when the number of stem cells is too low)
- unit that chooses a subset of the stem cells to commit to the differentiated state (activated when the number of committed cells is too low)
- differentiator that executes the choice and causes the chosen cells to actually differentiate into the desired cell type

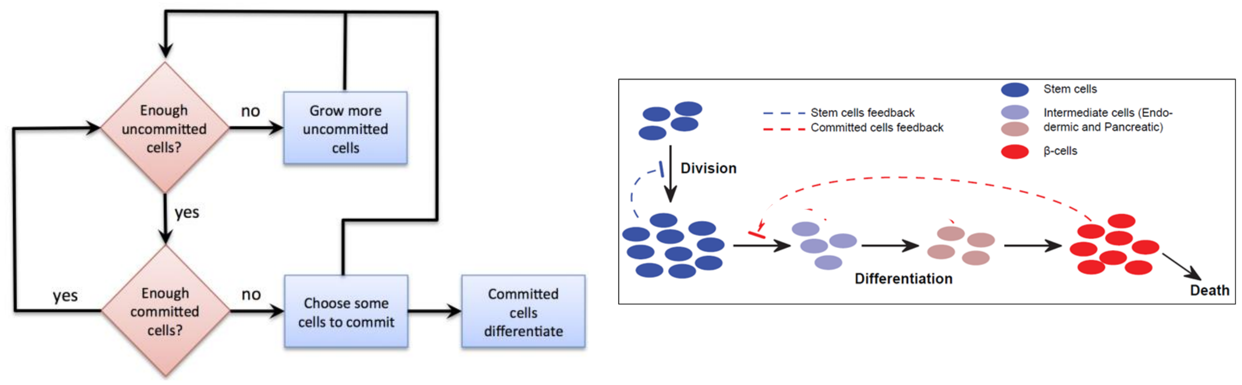

Enough uncommitted cells?
no
Grow more uncommitted cells
yes
Enough committed cells?
yes
no
Choose some cells to commit
Committed cells differentiate
Stem cells
Stem cells feedback
Committed cells feedback
Intermediate cells (Endo-dermic and Pancreatic)
β-cells
Division
Differentiation
Death

Figure 4: Population-level flowchart. The goal is to maintain a committed cell population within some range.

Once we have this population-level flowchart, we decompose it further into a flowchart that describes the behaviors that must be encoded in individual cells. Initially, all cells are undifferentiated and uncommitted, and each cell must estimate the number (or density) of committed and uncommitted cell subpopulations around it to determine its own actions.

An uncommitted cell decides whether to grow and whether to commit to differentiate based on its estimate of the number of uncommitted cells in its operational environment. If there are few uncommitted cells in its environment, it should grow to increase that population; if there are enough uncommitted cells, it will either remain unchanged or commit to differentiation, depending on the number of committed cells.

Both uncommitted and committed cells emit specific biochemical signals, and with reasonably fast diffusion, these signal concentrations are related to the local density of the uncommitted and committed cell population. Uncommitted cells first sense the concentration of the uncommitted cell signal. If it is below a certain threshold, the uncommitted cell will grow; if it is above the threshold, the cell must determine whether it should commit to differentiation, depending on the number of committed cells in its environment. If the committed cell signal is above its threshold, the uncommitted cell simply continues to monitor both these subpopulation levels. Alternatively, if the signal is below the threshold, the uncommitted cell commits to differentiation.

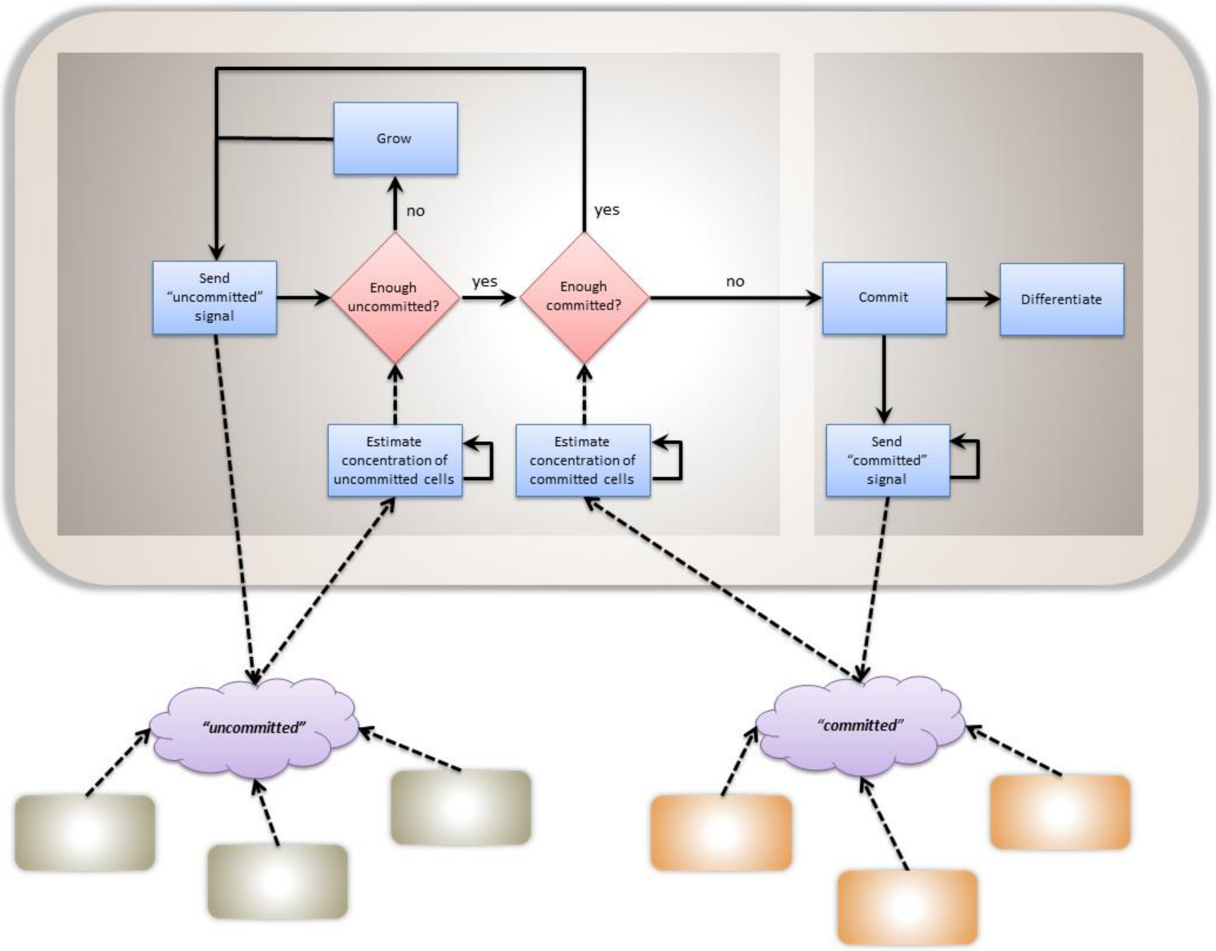


Figure 5: Initial artificial tissue homeostasis design.

Simulating Artificial Tissue Homeostasis Design

At this point in the design process, it is usually beneficial to create a computational model of the system, simulate it, and then analyze whether and how this particular design could achieve the desired function. The transition in representation from a block diagram / flowchart to a computer model forces us to create a formal description and make certain design choices and assumptions. It is essentially always the case that a particular system design only functions correctly under certain conditions, where these conditions include parameter values for the processes that govern system operation. Such processes can include basic biochemical reactions, for instance chemical diffusion of molecular signals between cells. But in a computer model, we can also model more abstract, 'lumped' processes, such as cell growth and division or multi-input logic functions that regulate gene expression.

We chose to simulate the system using the GRO computational environment (Jang et al.), which provides convenient abstractions to model the spatiotemporal evolution of multicellular systems. It’s important to note that the GRO simulations are high level, and do not directly keep track of all the relevant basic biological processes. However, they are quite useful in terms of investigating the overall characteristics of the design and associated algorithm. Simulating in GRO consists of writing a program file that encodes the various aspects of the system. In the file, included in the Appendix, we encode the flowchart behavior of each cell (Figure 5) as a set of variables and guarded rules that can execute simultaneously. For instance, one variable called ‘commit_mode’ keeps track of the state of this particular cell, with 0 signifying an uncommitted state and 1 a committed state. A guarded rule then continuously examines the value of ‘commit_mode’. If the value of ‘commit_mode’ is 0, the cell produces a diffusible signal to signify its uncommitted state. (GR1 below). On the other hand, if the value of ‘commit_mode’ is 1, a second guarded rule results in secretion of a different diffusible signal (GR2 below). These rules are written as follows:

```
// GR1

commit_mode = 0 : {
      emit_signal ( sig_qs,  signal_qs_synth );
}

// GR2
commit_mode = 1 : {
      emit_signal ( sig_cmt, signal_cmt_synth );
      gfp := 50;
}
```

The first rule, which is executed when ‘commit_mode’ is 0, results in production of signal ‘sig_qs,’ which signifies that the cell is uncommitted, at a synthesis rate of ‘signal_qs_synth’. The second rule, which is executed when ‘commit_mode’ is 1, results in production of signal ‘sig_cmt,’ which signifies that the cell is committed, at a synthesis rate of ‘signal_cmt_synth’. In the second guard rule you can see that we have also added a visual indicator for committed cells in the form of green fluorescent protein (GFP), which is relatively bright and provides a nice, easy way to visually monitor our system’s behavior. The symbol “:=” signifies an assignment of the value 50 to the variable gfp, in contrast to checking for equality with the symbol “=”.

The cell continuously monitors the local concentrations of the two diffusible signal molecules, ‘sig_qs’ and ‘sig_cmt’, and uses this information to approximate the level of both the uncommitted and committed cell subpopulations in its environments. To fully define the system, additional guard rules are needed to encode how a cell should respond to different environmental levels of ‘sig_qs’ and ‘sig_cmt’.

For example, the following rule describes an uncommitted cell’s continued proliferation if it senses that there are enough committed cells:

```
// GR4
(commit_mode = 0) & (sense_sig_qs < min_uncommitted ): {
       set ("ecoli_growth_rate", 0.05);
       gfp := 0;
}
```

This rule states that, if the cell is uncommitted (“commit_mode = 0”) and the cell senses that the level of chemical signal associated with uncommitted cells is lower than some threshold (“sense_sig_qs < min_uncommitted”), the cell will grow at the “ecoli_growth_rate” without producing any gfp (“gfp := 0”). Note that the parameter related to sig_qs, the uncommitted signal, is not the absolute value of this signal, but the value sensed by the cell.

Another rule is needed to describe how an uncommitted cell responds if there are enough uncommitted cells (“sense_sig_qs > min_uncommitted”) and not enough committed cells (“sense_sig_cmt < need_comitted”):

```
// GR6
(sense_sig_qs > min_uncommitted) & (sense_sig_cmt < need_committed) : {
   commit_mode := 1;
   set ("ecoli_growth_rate", 0);
}
```

If the preconditions are satisfied, the cell commits to differentiation (“commit_mode := 1”). In addition, committed cells don’t grow, so when the cell commits, the growth rate must be set to zero (“set ("ecoli_growth_rate", 0)”). The set of commitment preconditions represents a particular design decision to implement tissue homeostasis as described above. Additional code needed to perform the simulation, included in the Appendix, allows us to explore the behavior of the system. Figure 6 provides snapshots of the simulation state from three different time points and a time-series graph showing the total number of uncommitted and committed cells over time. In this example, all uncommitted cells in the system quickly commit, which is not the aim of our system. The reservoir of uncommitted cells is completely depleted, and the system then slowly decays as the committed cells die and no uncommitted cells are available to replenish their population. This result highlights the importance of evaluating a design using computational tools prior to the physical construction of the corresponding DNA and experimental testing.

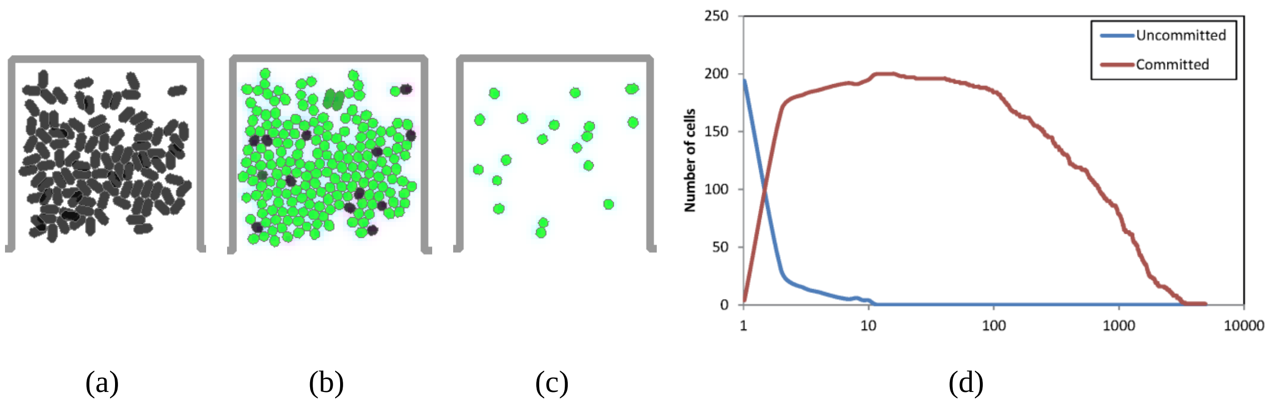
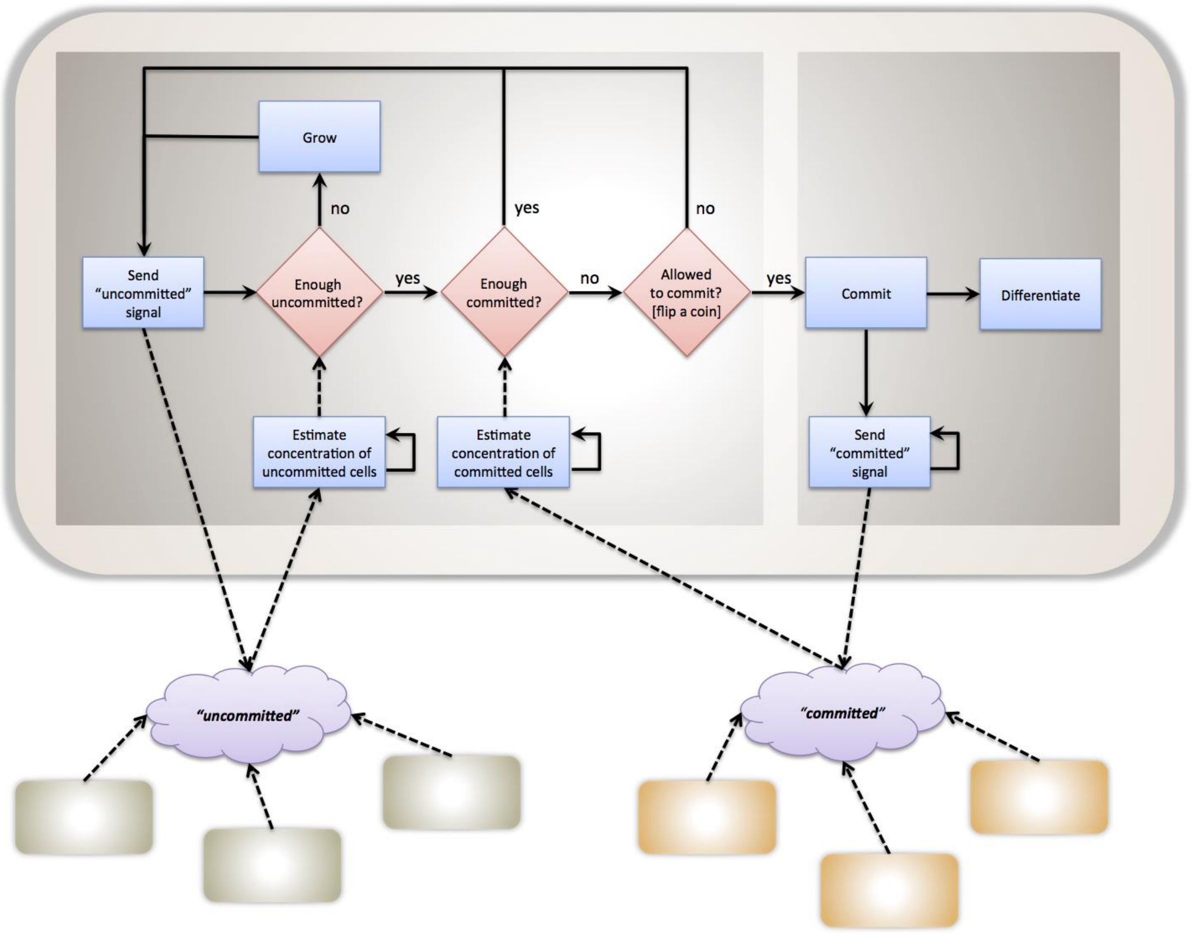


Figure 6: Time-series simulations of artificial tissue homeostasis failure. (a-c) Snapshots of the simulation at times 0, 2, and 2000 hours. The black cells are proliferating uncommitted cells, the dark green cells are non-proliferating uncommitted cells, and the bright green cells are committed to differentiation. (d) Graph depicting the number of committed and uncommitted cells during the simulation.

The simulation reveals that our designed system is not behaving as needed: instead of homeostasis (maintenance of a certain level of committed and uncommitted cells), we saw en-masse cell commitment in response to an insufficient number of committed cells, resulting in a crash of both the uncommitted and committed cell numbers. Closer inspection of system dynamics during homeostasis failure (not shown here) reveals the cause: the sense_sig_qs signal decays more slowly than the commitment rate of the cells, so the uncommitted cells temporarily "believe" that there is still a sufficiently large number of them even after their numbers dwindle. In other words, the uncommitted cell signal is an estimation, not a direct measurement, of the uncommitted cell population, and may in fact sometimes represent out-of-date information. This is not uncommon in biological system, but as designers, we must find methods to handle such incomplete information.

In our particular case, one possible solution is to curtail the commitment rate such that only a portion of the uncommitted cells are "allowed" to commit during a given time frame, thereby limiting the impact of out-of-date information and hopefully preventing this type of catastrophic system failure. Within the biological and physical constraints of our system, it is not possible to immediately and exactly determine which portion of the cells would be allowed to make the differentiation decision, but we can approximate this approach using statistical means. Specifically, each uncommitted cell that senses lower-than-needed committed cell signal and higher-than-needed uncommitted cell signal will flip a weighted coin to determine whether it is allowed to commit, instead of committing automatically (Figure 7). Adding this statistical requirement will effectively set an upper-bound on the overall rate of commitment for the population as a whole, which should address the reason for system failure in our first simulation.



Grow
no
yes
no
Send "uncommitted" signal
Enough uncommitted?
yes
Enough committed?
no
Allowed to commit? [flip a coin]
yes
Commit
Differentiate
Estimate concentration of uncommitted cells
Estimate concentration of committed cells
Send "committed" signal
"uncommitted"
"committed"

Figure 7: Revised design of artificial tissue homeostasis flowchart in individual cells based on a coin flip before commitment, to prevent en-masse commitment of cells.

In our simulation environment, this behavior can be encoded using the following modification to the earlier guarded rule GR6:

```
// GR6
(sense_sig_qs  >  min_uncommitted)  &  (sense_sig_cmt  <  need_committed)  &
(rand(10000) > commit_coin_flip) : {
    commit_mode := 1;
    set ("ecoli_growth_rate", 0);
}
```

This rule states that, if the cell “should” commit based on the levels of uncommitted and committed signal, ie the uncommitted signal (sense_sig_qs) is sufficiently high and the committed signal (sense_sig_cmt) is too low, the cell essentially flips a coin to determine whether it commits. The simulated coin flip is conducted by comparing a number chosen at random, rand(10000), to a predetermined threshold (commit_coin_flip).

Upon running our simulation with this added requirement, you can see that this probabilistic decision-making process provides a steady supply of uncommitted-to-committed cell transitions and, if the right system parameters are achieved, can maintain tissue homeostasis, perhaps indefinitely (Figure 8).

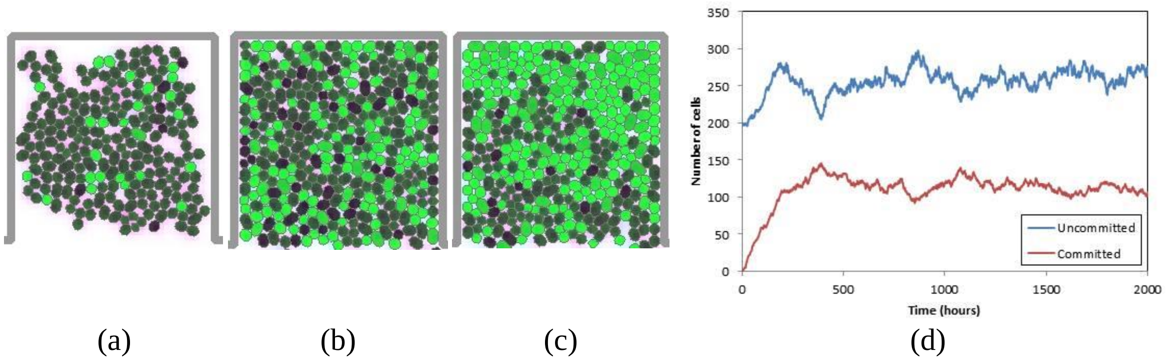


Figure 8: Time series simulation of artificial tissue homeostasis with coin flip to determine commitment. (a-c) Snapshots of the simulation at times 50, 200, and 1200 hours. (d) Graph depicting the number of committed and uncommitted cells for the duration of the simulation.

These types of simulations allow designers to analyze their system logic to determine whether there are fatal design flaws that need to be addressed, and whether there are aspects of the system that make it extremely sensitive to certain parameters. In other words, creating a detailed simulation before building the physical design components greatly increases the chance of success when the design is ultimately implemented.

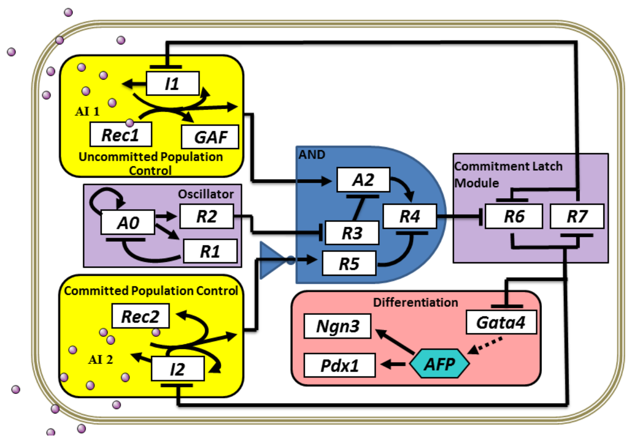
Bottom-up Assembly: Gate-level designs and part specifications

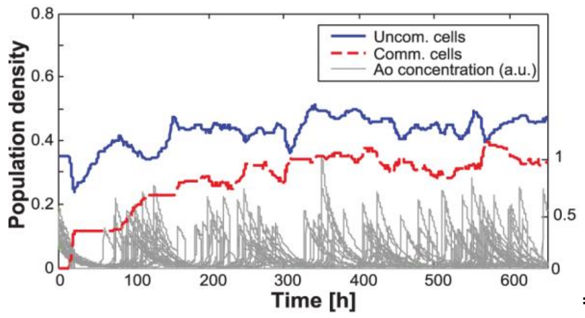
The design above is at a level of abstraction above both the biological network layer and the physical implementation layer. We have not stipulated the physical requirements for operation, such as generic properties of the biological mechanisms such a cooperativity or specific requirements for concentrations and kinetics; we simply know the specific high-level functions we wish to achieve, as defined by the population-level flowcharts and refined by the cell-based simulations. Once we believe that the high-level logic is potentially sound, we must begin to map the design to more realistic biological processes, for which we will introduce a third approach: discussing the system in terms of logic and gates used in fields like electrical engineering.

We will start working on a bottom-up assembly approach to engineer specific required components, as defined by our top-down design. Such a bottom-up assembly process requires gate-level designs, followed by specification of the properties necessary for robust function and finalized by a specification of actual DNA that fulfills these requirements.

Bottom-up Assembly: Tissue Homeostasis Example

The GRO simulations from the previous section provided insight into the general design, but the guarded rules that guided those simulations are not directly implementable using genetic circuit in cells. Instead, we need to go back to the revised flow chart design and map this representation to a system of interactions between biological entities within a cell. We will organize these regulatory interactions into interconnected modules that each perform a particular function, as shown in Figure 9.





(a) (b)

Figure 9: (a) A module-centric abstract gene regulatory network schematic of the tissue homeostasis system, also showing interactions between basic regulatory elements. Each set of basic elements grouped into a module performs a particular task, and modules interface with each other through specific regulatory interactions. Yellow modules involve cell-cell communication. Purple modules contain feedback regulation while the blue module serves as a logic function, but does not include feedback regulation. The peach module is for differentiation of stem cells into beta cells. (b) Stochastic simulation of the abstract gene regulatory network representation of the artificial tissue homeostasis design.

At this point we will provide just the broad summary of this modular design, without going into all of the details. The “Uncommitted Population Control” (UPC) and “Committed Population Control” (CPC) modules provide the mechanism by which cells estimate the density of uncommitted and committed cells and control commitment to differentiation depending on those densities. The module labeled “Oscillator” provides the coin flip that determines whether a cell is “allowed” to commit to differentiate; remember we identified the importance of this element in our GRO simulations. These three signals feed into a Commitment Permission Module (labeled as “AND”) that determines the appropriate behavior (commit or don’t commit) and sends the signal to a toggle switch (the “Commitment Latch Module”), which helps minimize feedback delay that would otherwise occur due to the very slow stem cell differentiation process, which can take several weeks to complete (Leon-Quinto et al.). The toggle switch in the Commitment Latch Module both controls differentiation and feeds back to the UPC and CPC, thereby decoupling the UPC and CPC modules from the slow differentiation process and placing the feedback control immediately downstream of the toggle switch rather than after the full differentiation process. Consequently, we gain a faster feedback response in exchange for assuming that a relatively constant fraction of cells successfully differentiate upon commitment.

This representation is closer to a biologically realizable system than our high-level flowchart - we have called out specific genes and proteins, like Rec2 and GAF - but implementation will require an even more detailed and realistic analysis of the biological processes underlying the proposed design, which we will demonstrate below for one circuit piece: the toggle switch.

A Logic Gate Example: The toggle switch design

A toggle switch is a memory circuit that can be switched between two states via signal pulses that switch the toggle to “on” and reset signal pulses that switch it back to “off.” Note that neither the set signal nor the reset signal need to remain at a constant level for the toggle to remain either “on” or “off”; the switch requires just a single pulse that reaches a threshold level to indefinitely switch the output state.

A simple, abstract biological network implementation of a toggle switch is a cross-repressive genetic regulatory network, in which one transcription factor represses the expression of another, and vice versa. (This design is similar to one designed in a seminal paper in the synthetic biology field (Gardner et al.), but differs in key ways.) In the system depicted above, R5 acts as the “SET” signal, specifying that cells should commit to differentiate, and R6 and R7 are the two cross-repressive transcription factors. The “RESET” signal is not represented, but would likely be an externally applied molecule, which would provide the designer with greater control of the system.

The system is broken down into further detail in Figure 10, below, and can be thought of as two gates, G1 and G2, two external inputs, the "SET" (R5) and "RESET" signals, and the two internal cross-repressive components, R6 and R7.

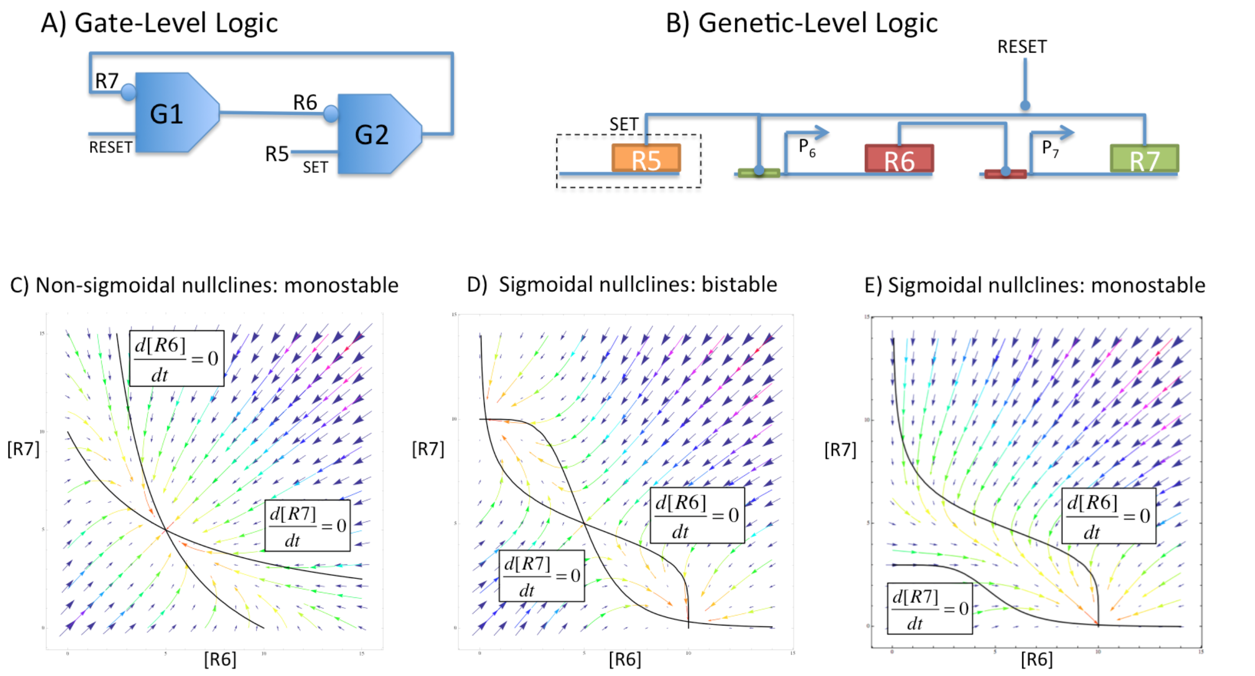


Figure 10: Requirements and circuit diagrams for the commitment toggle. (A). Digital logic representation of the Commitment Latch Module. (B) Abstract genetic circuit representation of the Commitment Latch Module. R5 and R7 are the same repressor protein but expressed from different promoters. R5 comes from the Commitment Decision Module promoter and R7 is expressed from an R6 repressible promoter P7; otherwise the repressors are identical. (C) Phase plane of the theoretical dynamics of the repressible promoters in B where the repression is non-cooperative. The resultant non-sigmoidal nullclines can intersect at only one point so the system can't be a toggle switch. (D) As in C but with cooperative repression of the promoters and resultant sigmoidal nullclines that intersect at three locations, two of which are stable steady-states. This would create a functional toggle-switch. (E) as in D but where the dynamic range of R7 production is not sufficient to intersect at with the R7 nullcline at the appropriate points. This is a so-called gate-matching failure.

We can see this circuit is a toggle by following the logic. First, we assume that both the SET and RESET inputs are initially low. Then, if the level of the first repressor, R6, is high, the output of G2, R7, is low. When R7 is low, the output of G1, R6, is allowed to remain at a high level, and the logic is consistent and creates a stable state of the system. Alternatively, if the R7 level is initially high, R6 is forced to decrease, which allows R7 to remain high, resulting in an alternative stable state.

If both repressors start at either low or high levels, the system is in a race condition to determine which stable state the system reaches. If both repressors begin low, the race is to see which factor increases quickly enough to repress the other; the result is often determined stochastically. If both repressors begin high, the race condition depends on which factor first fluctuates to a lower level, which can result from noisy gene expression.

Once the system reaches a stable state, it will only change to the alternate state if there is a change in either the SET or RESET inputs. For example, if the toggle is in the R6 ON state and receives the SET (R5) signal, R7 is forced on by G2, and G1 then forces R6 off. This new state is then maintained even when the system is no longer receiving the SET signal. Analogously, the RESET signal switches the R6 “off” state back to R6 “on.” We must ensure that the initial state of the toggle is OFF so that the cell will not commit to its final fate early. A simple way to do this is to have the RESET signal under the control of an externally applied small molecule such that a researcher can ensure the system starts in the right initial state.

To work biologically, this circuit must meet certain physical constraints. We have established that a toggle switch requires two stable states, and that the system reaches stable states when the concentrations of R6 and R7 are not changing; in other words, when the rates of change of the concentrations are 0, or d[R6]/dt=d[R7]/dt=0. In our system, d[R6]/dt is a function only of R7, so we can plot the locus of concentrations of R7 for which d[R6]/dt is zero, i.e. its nullcline, as well as a similar curve for the values of R6 that make d[R7]/dt=0. To produce two steady states, these curves must intersect in at least two places. The in-depth discussion of this requirement is saved for Chapter XX, but we will briefly explore the concept here.

The figures above show two cases: hyperbolic nullclines and sigmoidal nullclines. Superimposed on the nullclines is the vector field: the direction in the R6, R7 concentration space in which the differential equations are moving the system. If the nullclines are hyperbolic (Figure 10.C), they cannot intersect more than once, and all vector field lines converge to the single point of intersection: the single stable stationary state. In the sigmoidal case (Figure 10.D), on the other hand, the nullclines intersect in three places, and the vector field lines converge mainly to the two outer intersections, which represent the stable states. Biologically, sigmoidal nullclines result from cooperative binding, and the hyperbolic nullclines reflect non-cooperative binding (See Chapter/Appendix XX for more detail). Therefore, for an effective toggle, each repressor must inhibit its target promoter cooperatively. In addition, the gate-matching condition must be met; the dynamic range of R6 output must spans the switching region of R7, and vice versa. Figure 10.E demonstrates a situation where this condition is not met.

With such basic constraints in hand, we now need to specify a number of other requirements that are affected by how this toggle is to be connected to the rest of the circuit. For the moment, assume that the curves in Figure 10.D are realistic representations of the needed behavior of the circuit. This assumes that key physical parameters such as binding constants and degradation rates have been set for all elements within the Commitment Latch Module. For R5 to act as a SET signal with the toggle system depicted in Figure 10.B, its "OFF" state has to be well below 5 concentration units and its ON state well above this threshold. To drive the Commitment Actuation Module, the ON state of R6 (here, about 10 units) would have to be well above and the OFF state well below the repression threshold of the target promoter. Finally, the switching time of the toggle must be faster than the switching time of the oscillator driving the Commitment Decision Module. The design requirements of this module are driven in part by the interface requirement to the connected modules; all three modules must adhere to consistent specifications to be compatible.

Once we have specified the abstract genetic network and the physical requirements for the promoters and repressors such that the nullcline and gate-matching conditions are met, we can look for actual biological parts that meet these needs. Ideally, we would have a database of standard biological parts that have been carefully characterized in conditions that match those in which we expect our circuit to operate. In this case we might find, for example, that a designed family of well-understood promoter/repressor systems meet our specifications: the Talen-based TalR system.

Talens are a designable class of DNA binding proteins wherein a sequence of amino acids can be designed to target a cognate sequence of DNA. In this case, Talens targeted to different sequences bind upstream of the promoter, interfering with RNA polymerase elongation and repressing transcription. Talens are desirable not only for their tunable specificity, but also because using a designer family of repressors engineered from a common "parent" design means they are likely to have similar properties to each other and therefore will be more easily uniformly characterized and modeled.

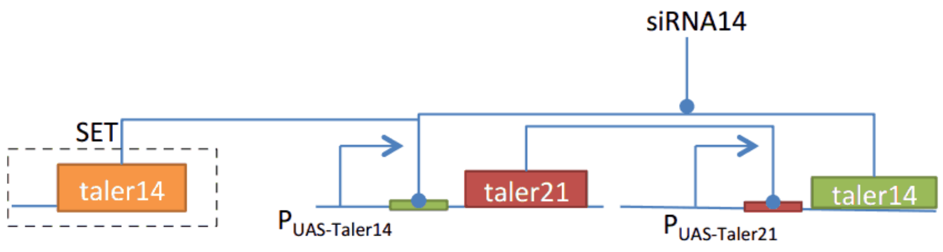


Figure 11: Mapping of physical elements to the abstract biological network for the Commitment Latch Module.

We can associate R6 and R7 with "orthogonal" members of this family (those that do not bind to each other's sites) TALER14 and TALER21, which repress their respective strong promoters ($P6=P_{UAS\text{-}Taler14}$ and $P7=P_{UAS\text{-}Taler21}$) in a sigmoidal fashion, to achieve the desired response. Expression of TALER14, our SET signal, is inhibited by the small inhibitory RNA siRNA14, which becomes our RESET signal. This siRNA targets a unique sequence in the TALER14 mRNA and triggers degradation of the transcript, thereby reducing TALER14 concentration. The measured transfer functions, which map input to output, can be used to assess whether these factors will meet our requirements; this will be discussed in more detail in Chapter XX.

As designers of this module, we must negotiate with the owner of the Commitment Permission Module to use TALER14 as R5 and our SET signal. Then, we must also ask that no other members of the team use TALER14 and TALER21 in their designs, nor can they use the respective promoters without checking that doing so doesn't destroy the overall logic of the system. In a perfect world, there might be many independent repressors and cognate promoters with similar properties from which the design could be implemented, making these negotiations easier. However, when they are available there are usually tradeoffs involved with the choices, and often there are simply too few options available for effective, large-scale design. Therefore, one of the thrusts in synthetic biology is to expand the number of independent parts that have particular properties or controlled variations thereof. Talen-based repressors are a possible type of scalable part family. We will discuss the discovery and design of scalable part families in Chapter XX.

The choices made above provide most of the necessary information for manufacturing, though some details remain unspecified, including particulars for the 5' and 3' UTR regions, the location in which these fragments of DNA will reside in the cell (in the genome, on an extrachromosomal elements, etc), and how they will be delivered (virally, via plasmid). Once it is manufactured (you can find a more detailed discussion of this process in the next section), ideally we would test this module design in the target host cell to confirm that it works as we induce the SET signal from a standard inducible promoter and that siRNA21 expression is an effective RESET signal. If the system behavior matches our physical specifications, we are done. If not, we must diagnose what went wrong and decide how to modify the system to bring it into compliance. Often at this point a library approach is used, wherein a number of variants of the promoter, protein coding sequence and 5' UTR are synthesized and screened for desired function.

Different applications in synthetic biology require different types of bottom-up specification. Metabolic engineering applications may require knowledge of enzymatic activities, allosteric regulation, metabolic fluxes, transport reactions, host metabolite pools, and intermediate toxicity. Viral gene therapy applications might require knowledge of immunogenic factors on the viral capsid, cell targeting efficiencies and factors that control location of integration into the chromosome. Some of the design requirements in these cases can be explored via simulation and formal design, and others require trying a number of variations and screening or selecting for the correct properties. For most of this book we will focus on methods of more formal design; in later chapters we will visit using the design cycle when it is necessary to use larger libraries or laboratory evolution.

5. A Brief Overview of Manufacturing and Characterization

Once a system has been designed, tested in a simulation environment, and perhaps adjusted based on any issues revealed in the simulation, the system must then be manufactured so it can be tested in real life, and implemented if successful. Manufacturing of synthetic biology systems consists mainly of creating DNA fragments (primarily genes and regulatory elements) and appropriately connecting or inserting them into existing sequences, as depicted in Figure 12 below.

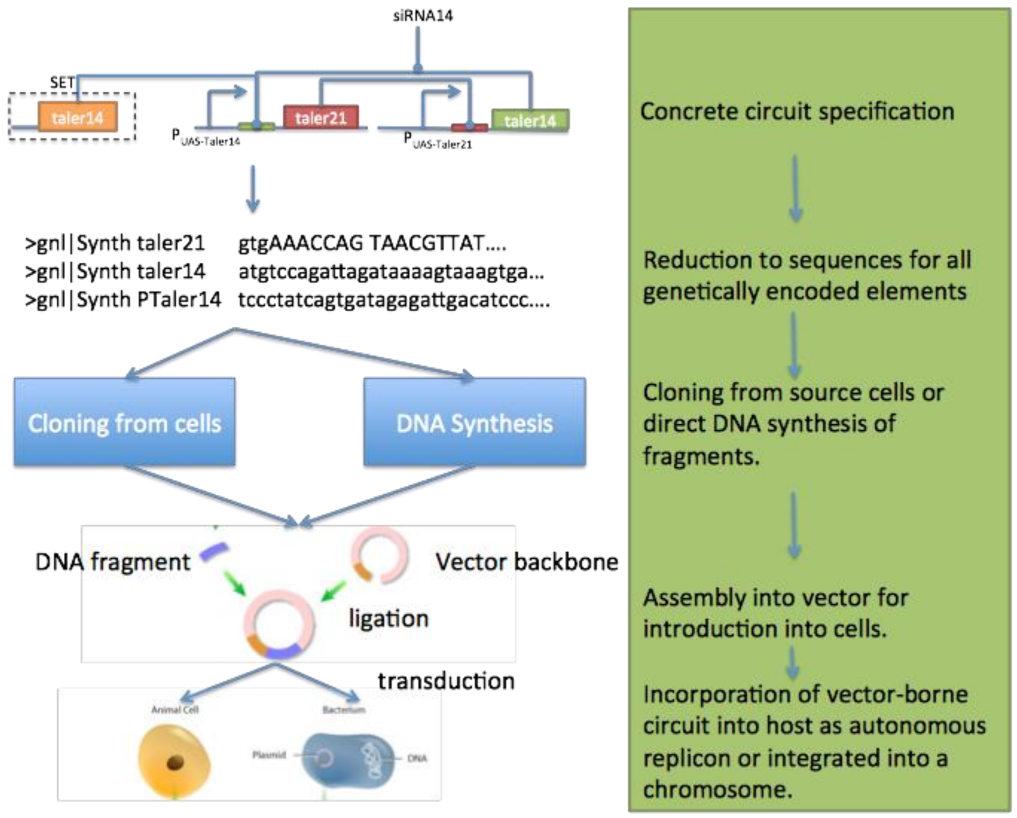

siRNA14
SET
taler14
$P_{UAS\text{-}Taler14}$
taler21
$P_{UAS\text{-}Taler21}$
taler14
>gnl|Synth taler21 gtgAAACCAG TAACGTTAT....
>gnl|Synth taler14 atgtccagattagataaaagtaaagtga...
>gnl|Synth PTaler14 tccctatcagtgatagagattgacatccc....
Cloning from cells
DNA Synthesis
DNA fragment
Vector backbone
ligation
transduction
Animal Cell
Bacterium
Plasmid
DNA
Concrete circuit specification
Reduction to sequences for all genetically encoded elements
Cloning from source cells or direct DNA synthesis of fragments.
Assembly into vector for introduction into cells.
Incorporation of vector-borne circuit into host as autonomous replicon or integrated into a chromosome.

Figure 12. Basic workflow for manufacture. This process may be supported by computer-aided manufacturing software that plans optima synthesis and assembly routes (both for efficient and error free assembly and cost). There are emerging platforms also for automated this assembly and transformation of DNA into the host.

During the design process, we determined which biological molecules and regulatory elements we wish to include in our system. When we are ready to manufacture, we can search various databases to find the DNA sequences that encode them. Then, we can either use cloning techniques to copy these target elements from existing genetic material, or we can *de novo* synthesize increasingly long fragments of the desired DNA, which is becoming more common as synthesis technology improves. These fragments are then assembled, often in a series of steps, to create larger and larger DNA molecules until the desired design fragment is complete. Finally, these fragments are ligated into a DNA backbone (usually a plasmid of some sort) that will be transferred into the target cell or virus. At this point the host expression machinery takes over and, if all goes well, the circuit will begin to function. (For a review of synthesis, cloning, assembly and packaging see Appendix A.)

The resulting cells and their populations may be measured with various tools, including microscopy, mass spectrometry, RNA sequencing, and growth measurements, to characterize circuit function and its interaction with host physiology and fitness. In the case of a direct design, the results can be compared to the original specifications; if the manufactured system doesn't meet the specifications, several diagnostic strategies can be used to determine how the design should be adjusted (see Chapter XX for a more detailed description of this process).

In some cases, we may have manufactured a library of circuit versions to identify the one(s) that perform best. In these situations, we transfer a heterogenous DNA mixture to our target cells so that each individual cell takes up a single variant in the library, and statistically each variant is operating in at least one cell. We can then screen this cell population; that is, we can separate and individually observe the cells to find the variants that best meet specification. This process is most efficient if the circuit function is linked to a selectable marker. For example, if we are testing a library of inducible promoters, we could use a selectable marker to easily isolate versions that are "off" below a certain concentration of inducer and "on" above it. In this case, we might place an antibiotic resistance gene downstream of the promoter. We can then induce the promoter, grow the cells in the presence of antibiotic, and keep the cells that survive, since their survival means that their promoters have turned "on" when induced; this process is called negative selection.

This initial screen only tests for half of the desired function though; it identifies promoters that turn "on" above a certain threshold, but it does not insure that they are "off" below that threshold. To find the promoters that also meet this specification, we need to isolate the negative selection survivors that, in the absence of inducer, do **not** survive if antibiotic is present. If the library resulting from the negative selection is small enough, we can test for this property individually; otherwise we may need to devise a second construct for additional testing. Once we have identified those cells with the appropriate behavior, we must characterize the clones to identify the successful design sequences. Further analysis of the survivors relative to the failures can also provide insight about our design that can be used to work toward a more rational design.

With working circuits in hand, we next begin strain optimization, wherein we test either genetic or growth condition variants to bring the system into optimal compliance with specification. This might entail modifying the host so that less of the sugar it consumes is directed to creating more cells and more is directed towards biosynthesis of a target molecule, or it might mean changing expression levels of key circuit proteins so the heterologous circuit isn't so great a load on the host. Some aspects of this will be covered in Chapters XX.

6. Challenges to Design and the Real Biology

The design and manufacturing processes described above might not seem exactly simple, but they may seem straightforward enough to ask what the roadblocks are for creating the advanced applications proposed at the beginning of this chapter. The example of tissue homeostasis uncovers a series of technological, managerial and perhaps social challenges to achieving the scalable engineering discipline we set as a central goal of synthetic biology, which we describe below.

The Challenge of Uncertainty

Current approaches to address Type I Diabetes through stem cell transplantation are hampered by the lack of complete system characterization. Biological systems are complex, and introducing variations to the components and behaviors often has unexpected, and potentially undesirable, effects. For example, cell transplantation may result in interactions between the transplanted cells and the host that lead to the conversion of transplanted stem cells into uncontrolled tumorigenic descendants or host death. When we do not fully understand the mechanisms by which these issues arise, which is often the case, it is exceedingly difficult to adjust the design to address them.

Synthetic biology begins to provide the opportunity to overcome this uncertainty. Specifically, if designers can create a starting cell line that can be transplanted safely and add only the circuitry that does exactly what the engineer specifies and no more, such undesirable side effects can hopefully be limited.

However, due to our uncertainty about the mechanisms that lead to failure within a host, we know the specifications for our synthetic circuit are incomplete and may suffer from some of the same issues as the original, less rationally engineered, solutions. From an engineering perspective, this uncertainty is expected, and an iterative design cycle is known to be necessary. The goal is not to eliminate failure completely, but to make it easier to diagnose the reasons for failure and thereby determine the variations that are most likely to solve the problem. Rapid diagnosis of design failures and iteration are critical in applications whose proper functions can only be tested in vivo, and is especially important in applications constructed with many biological components, for example, long metabolic pathways, where the number of possible variants is so large they cannot be explored through library approaches. Rationalizing the design and creating the system from parts of known properties is a crucial step in this direction.

There is an emerging classification of types of uncertainties that arise which together are referred to as context issues. These include environmental context, in which interactions between the engineered biological system and the world in which it lives lead to system malfunction and undesirable environmental changes; host context issues, in which a heterologous circuit or part behaves unpredictably in different host genetic backgrounds and physiologies; genetic context issues, which stem from uncertainties in synthetic circuit/part behavior when physically or functionally connected to other parts; and evolutionary context, which is uncertainty in how the system fails or is affected by mutations that arise during its function.

In addition, uncertainty about a design's ultimate behavior raises social and ethical questions, including:

- What regulation may be appropriate for the engineering design process, and ultimately the application itself?
- What risks are acceptable during the design process?
- To what extent must engineers demonstrate the safety of the final application given the variation in host environment for each individual case?

Synthetic biology aims to address these issues as well as the purely technical hurdles to creating reliably engineered biological systems.

The Challenge of Paucity of Parts

Once a top-level specification is made, there are usually many different possible routes for implementation. There may be conceptually similar approaches that have different top-down decompositions and thus are composed of different functional modules, or there may be conceptually different approaches that lead to entirely different functional solutions. There may be many different ways of implementing a given module, including logically equivalent circuits implementing the same regulatory system using different gates; pathways using the same starting compound and delivering the same end product through different series of chemical transformations, with differing enzymatic steps, cofactor requirements, and yields; and similar designs implemented with different versions of the molecular functions, such as different enzyme variants or repressor types.

Generally, this variety is a good thing. Different design choices may result in a variety of useful properties of the designed behaviors, as well as different unavoidable off-design effects that may be more or less advantageous for a given application. Often it will not be clear which are tolerable or good until after the system is tested, so it is important to have options.

As the field currently stands, however, only a tiny fraction of the sequences we have collected from natural systems has been functionally characterized to the extent that they can be modeled sufficiently for use in a formal design process, or at all. To date, most synthetic biological applications have been constructed from parts derived either from a very limited set of transcription factors and promoters that operate generally in a narrow range of cell types, including certain bacteria, particular yeasts and, more rarely, mammals, or from a more diverse but still relatively small number of enzymes known to be important to the biosynthesis of important classes of molecules. Thus, the options currently available for implementing a given design are far more limited than might be expected. Simply finding a part with the necessary function, let alone the physical parameters, to meet the integrated design specification can be nearly impossible.

Synthetic biology also requires a much greater diversity of parts than does electronics, compounding the problems described above. In electronics, identical copies of a transistor can be spatially separated and connected with actual metal wires to create multiple independent circuits with a relatively small supply of unique parts. In cells, on the other hand, such spatial separation usually does not exist, so “wiring” circuits requires chemical specificity, and wiring complex systems with multiple circuits requires a large supply of orthogonal parts that all meet these specificity requirements. We cannot reuse the same repressor like we reuse the “same” transistor in electronic circuits, so we need *variants* of the same repressor, with nearly the same characterized properties (e.g. degradation rate, DNA binding constant, cooperativity), but with different specificity for their target DNA sequences. The cognate promoters would also have identical behaviors, and all of these parameters would remain constant or vary predictably as the host and environment changed. With such a standardized set of parts, it would be easy to design and construct large, complex regulatory circuits.

However, it is rarely the case that we have such classes of well behaved, homogeneously operating, non-interfering, environmentally insensitive parts. Moving forward, finding and characterizing parts that form such a basis set or foundation for scalable design of diverse biological applications is a central challenge for synthetic biology. In the meantime, we can take advantage of the ability to construct and screen/select large libraries of circuit variants (for some applications). In addition, DNA sequencing and synthesis technologies and methods of assaying molecular function such as mass-spec are improving sufficiently to begin to permit extremely high-throughput assay of biomolecules in different configurations, from which we can begin to extract rules for design.

The Challenge of Composition

In other engineering disciplines, parts are commonly designed for easy and predictable assembly. Formal frameworks state how parts can be combined to form higher-level objects, and when followed ensure that a given design yields a functionally correct implementation upon manufacture (these notions are referred to as “rules of composition” and “correctness by construction”).

For biological systems, achieving such a framework is challenging because of the uncertainties and uncharacterized contextual effects that arise in biology. While far from complete, here is a short list of compositional issues synthetic biologists currently face:

1) **Variation of function when two “parts” are physically connected by residing on the same molecule (physical composition).** For example, in *Escherichia coli*, placement of sequences on the same chromosome can affect how they behave. Specifically, placing a gene expression cassette near the origin of replication can lead to a larger copy number during replication than if the cassette were placed near the chromosome terminus, leading to predictable but perhaps unwanted variation in expression levels. Another example occurs when different functional elements, such as genes and ribosome binding sites, are placed on the same mRNA transcript. On the mRNA, these sequences may form unpredicted base-pairing and tertiary structural interactions, resulting in three-dimensional RNA structures that could change the expression level of the genes or the apparent strength of the ribosome binding site.

2) **Variation of function due to level matching and retroactivity when the activity of one part drives that of another (functional composition).** If one part is designed to drive the activity of a downstream part, the levels must be matched; that is, the output of the first part must be at the appropriate level to trigger the appropriate action from the subsequent part. Imagine a repressor system, with the repressor output designed to vary between a “low” concentration X to

a high concentration Y. To be a functional repressor, the target must be appropriately calibrated to levels X and Y. If the target expects “low” to be less than X, or “high” to be greater than Y, these parts are not matched, and a change of state in the repressor - say, turns “on” to reach concentration Y - fails to propagate to the target, which only responds to concentrations **greater** than Y. It is often non-trivial to change the dynamic range (high and low values) of an upstream device or the threshold requirements of a downstream device, so level matching must be carefully considered at the point of initial design formulation.

Problems of retroactivity occur when connecting the output of an upstream device to a downstream part affects the first device’s function or its ability to connect to other devices. A simple example arises when a repressor is designed to drive multiple downstream target promoters. This fan-out configuration creates competition for the repressor such that downstream device A must compete for repressor with downstream device B. If A is alone in the cell with the input device it may function correctly, but when B is introduced, it also binds the repressor, drawing it away from A and causing a level matching problem.

3) **Variation of function due to functional parasitism and cross-talk when parts rely on common resources (Chassis composition).** Parasitics arise when a device exhibits activities outside the desired design. For example, an enzyme may demonstrate substrate promiscuity and operate on more than just a target metabolite, affecting cellular pathways other than those intended by the designer. Such parasitic activity can create additional metabolic load or be toxic to the host, changing its fitness or interfering with other parts of the designed metabolic pathway. At times such promiscuity may be useful and can be designed into our systems, but in other cases it can be problematic.

Parasitic activity is one source of cross-talk, such as when a repressor binds not only to its target sequence but also to the target sequence of another repressor. This direct parasitism leads to unwanted coupling between the two systems. Cross-talk can also arise when multiple devices use the same common resource, without parasitism. For example, when two proteins are degraded by the same host protease, increased production of the first protein will make it compete better for the protease, effectively slowing the second protein’s degradation rate. Competition for other common resources such as ATP, other co-factors, or ribosomes are further examples of this phenomenon.

To make biological design scalable, we must specify rules that can account for these varied compositional issues and guarantee correct compositional behavior. For example, we may want to limit designers to create and use families of parts designed to be orthogonal/specific so that parasitics are minimized (see Chapter XX). We may attempt to minimize cross-talk by requiring that parts be vetted for using the least common resources. We may create physical interconnection standards that insulate one part of a molecule from another to minimize unwanted interaction effects (Chapter XX). We might specify standards for the dynamic ranges for input and outputs and standard “insulators” for functional composition so that designers can easily use one another’s devices.

Even if such standards were implemented, though, they could not be obeyed universally. We cannot eliminate all dependence on common cellular resources or eliminate all parasitic activities from our parts. Even for common, ubiquitous parts such as gene expression regulators, it is not yet possible, and may not ever be possible, to gain perfect behavioral homogeneity across a transcription factor family, especially not over variable environmental conditions. In addition, many systems we wish to design with, including enzymes, molecular machines like flagella, and secretion systems, are fundamentally unsuited for standardization because in many ways their functions are, by their nature, unique and non-standard. Even so, if we compile enough examples of their uses and modes of failure, we can begin to use that knowledge in design.

There is still much to be learned about the mechanisms that govern the behavior of even a simple cell, including how it interacts with other cells and its environment. This uncertainty remains a serious challenge to scalable and trustworthy design. Abstraction may be key to a good design discipline, but we must also contend with the “real” biology of our systems.

7. Summary

In this chapter we have brought you through a way of approaching design that is relatively universal in engineering fields. You start with a high-level specification of what you want your system to do. You break the problem down into a series of subtasks and subfunctions that can be worked on separately. Possibly, you explore different variations of this high-level design. At this point simulations of the system may show you that design choices are incorrect or lead to trade-offs you must consider. As the concreteness of the designs increases and choices are made, lower-level abstractions are made that are closer to how the parts of the system would work in reality. In our tissue homeostasis example, we went from a high-level flow chart to a modular design for an abstract genetic network, where the simulations look more like real biology and allow the exploration of more refined, specific design choices.

At some point the system has become concrete enough that you can choose actual physical elements from which to construct your system. If they are sufficiently characterized you can test whether these elements will work in more detailed and measurement-constrained simulations to determine if they are likely to work. If the uncertainty is high, you might choose a set of physical parts to test and simulate the range of results. Ultimately, you have to experimentally test the subsystems and the integrated system for adherence to specification, then diagnose failures and suggest design modifications to fix them.

For synthetic biology, this design-build-test-learn cycle is not yet common practice. In part this derives from the lack of knowledge about the biological systems being engineered and therefore acceptable models and simulations to be used in design. In part, it derives from the fact that there are still few applications complex enough to require it. Bespoke design, where master builders with deep domain knowledge know from experience what to try and how things fail, is still the main method for constructing biological systems. Because it is relatively easy, and becoming easier, to construct and screen many variants of a system effectively, especially for the chemical biosynthesis applications that are the most popular, simply trying many variants constrained by relatively simple domain knowledge can be very effective.

However, if we wish to not only increase the complexity and reliability of the applications but also the speed with which even small applications can be productively built, we need to move to a more standardized manufacturing framework, which will be made possible by a more formal design process. In the following chapters we will take you through the rapidly evolving landscape of designing parts, hosts, and standard principles of interconnection that address the challenges in section 1.5 while also enabling the abstractions necessary for the rest of the design process. We will show that in the cases where this has been best achieved, it is possible to reliably design fairly complex circuitry or choose variations of circuits among which a working version will be found.

As a field, we are simultaneously inventing the “theoretical” engineering design framework for biology, designing and implementing components to make it easier to implement system designs, and trying to design useful systems. In this book, our goal is to teach the successes thus far and the approaches we believe will be most effective in creating the discipline in which biological design is served by highly predictive computer-aided design and manufacturing frameworks that permit the synthesis of biological systems with increasingly complex function to solve key societal issues.

Bibliography

Jang SS, Oishi KT, Egbert RG, Klavins E, Specification and simulation of synthetic multicelled behaviors. ACS Synth Biol. 2012 Aug 17;1(8):365-74

Gardner TS, Cantor CR, Collins JJ (2000) Construction of a genetic toggle switch in Escherichia coli. Nature 403: 339–342.

Tigges M, Marquez-Lago T, Stelling J, Fussenegger M (2009) A tunable synthetic mammalian oscillator. Nature 457: 309–312.

Stricker J, Cookson S, Bennett M, Mather W, Tsimring L, et al. (2008) A fast, robust and tunable synthetic gene oscillator. Nature 456: 516–519.

Leon-Quinto T, Jones J, Skoudy A, Burcin M, Soria B (2004) In vitro directed differentiation of mouse embryonic stem cells into insulin-producing cells. Diabetologia 47: 1442–1451.

Zhang D, Jiang W, Liu M, Sui X, Yin X, et al. (2009) Highly efficient differentiation of human ES cells and iPS cells into mature pancreatic insulin-producing cells. Cell Res 19: 429–438.

Weiss R, Homsy GE, Knight TF Jr (1999) Toward in vivo digital circuits. DIMACS Workshop on Evolution as Computation.