

Project Description

CAREER: *Computing with Things Small, Wet, and Random* –

Design Automation for Digital Computation with Nanoscale Technologies and Biological Processes

1 Introduction

“Listen to the technology; find out what it’s telling you.” – Carver Mead, 1981

The successful formula in circuit integration has been both top-down and rigidly hierarchical, with sharp boundaries in the levels of abstractions: from transistors, to gates, to functional units, to processors. Perhaps the most fundamental abstraction is the one that separates the physical level from the logical level, namely that all digital computation consists of a deterministic sequence of zeros and ones. From the logic level up, the precise Boolean functionality of a circuit is prescribed; it is up to the physical layer to produce voltage values that can be interpreted as the exact logical values that are called for. This abstraction is firmly entrenched yet costly: variability, uncertainty, and noise in parameters and signals – all must be masked and compensated for through physical design and manufacturing [?].

As the semiconductor industry contemplates the end of Moore’s Law, there has been a groundswell of interest in technologies that offer a path to scaling beyond the limits of the current complementary metal-oxide-semiconductor (CMOS) technology [?]. **Nanoscale technologies** such as carbon nanotubes, nanowire arrays, and molecular switches present both challenges and opportunities for design automation. On the one hand, such technologies promise unprecedented densities, which will translate into vast numbers of switches, logic gates, and bits. On the other hand, both the scale and the assembly techniques constrain the circuits, particularly if they are self-assembled. Most of the technologies are characterized by very high defect rates, as well as inherent randomness in their wiring [?]. We argue that the deterministic paradigm is untenable in the nanoscale regime. This project will develop a vertically integrated framework for circuit integration based on **probabilistic design** that captures and exploits the inherent randomness in the components, connectivity, and execution of nanoscale technologies [?].

“If I can’t create it, I don’t understand it.” – Richard Feynman, 1983

In the nascent field of **synthetic biology**, practitioners are striving to create new form and functionality in biological systems through genetic manipulations. Recent accomplishments portend of a coming revolution in the biosciences. From *Salmonella* that secretes spider silk proteins [?], to yeast that degrades biomass into ethanol [?], to *E. coli* that produces antimalarial drugs [?], the potential impacts are far-reaching. Still in its early stages, the field has been driven by experimental expertise; the remarkable exploits are attributable to the skill of the researchers in specific domains of biology. There has been a concerted effort to assemble repositories of standardized components [?]. However, creating and integrating synthetic components remains an ad hoc process.

The field of synthetic biology is now reaching a stage where it calls for computer-aided design tools. The circuit design community has unique expertise to contribute to this endeavor. As with integrated circuits, the key in synthetic biology is to develop design flows that systematically explore configurations at different levels of abstraction. Randomness is inherent to biochemistry: at each instant, the sequence of reactions that fires is a matter of chance. We argue that biochemical design problems can be cast in terms of **discrete, probabilistic computation** performed on protein quantities. This project will develop a modular and extensible design methodology for synthesizing biochemistry with robust probabilistic outcomes [?].

“Probability is expectation founded upon partial knowledge.” – George Boole, 1832

This project advocates a novel view for computation: instead of transforming definite inputs into definite outputs – say, Boolean, integer, or real values into the same – circuits and biological systems transform probability values into probability values; so, conceptually, real-valued probabilities are both the inputs and the outputs. This is illustrated in Figure ???. Note that the underlying behavior of such circuits is discrete and inherently random. Nevertheless, when cast in terms of probabilities, the computation is robust: inputs

and inputs are random in biology, statistical distribution of the signals and 5m bit streams; these are digital, consisting of zeros and ones; they are processed by ordinary logic gates, such as AND and OR. The computation is random at the bit level; nonetheless, in the aggregate it becomes exact and robust, since the accuracy depends only on the statistics of the bit streams.

- In the realm of biology, signaling pathways produce specific output types of proteins in response to input types. The exact sequence and timing of biochemical reactions that fire is random. However, the probability distribution on specific outcomes – for instance, the mutually exclusive production of different signaling molecules – is precise and robust.

The view is the basis of an efficient new methodology for synthesizing circuits with probabilistic behavior. Our approach produces circuits that are highly resistant to errors – both in the underlying components and in the signal values. If noise-related faults produce random bit flips, these result in fluctuations in the statistics; accuracy is regained through increased redundancy. Thus, the approach provides *tolerance of faults* that scales gracefully to large numbers of errors [?]. In synthetic biology, our approach produces a precise distribution of different outcomes across a population of organisms or in a sequence of trials. This gives us the ability to fine-tune the response – akin to hedging with a portfolio of investments – in spite of large uncertainties in the underlying mechanisms [?].

2 Related Work

The project builds upon work presented at top conferences in the fields of design automation and synthetic biology in 2007 and 2008: the Design Automation Conference [?, ?], the International Conference on Computer-Aided Design [?, ?], the Synthetic Biology Conference [?], and the Advances in Synthetic Biology Conference [?]. Expertise in logic synthesis stems from Marc Riedel’s dissertation work on feedback in combinational circuits [?, ?]. Experimental validation of the design methods for synthetic biology will be pursued collaboratively with the research group of Tim Mullins in the Life Sciences at IBM, through the joint Biomedical Informatics and Computational Biology Initiative with the Mayo Clinic and IBM Rochester.

3 Design Automation for Nanoscale Digital Circuits

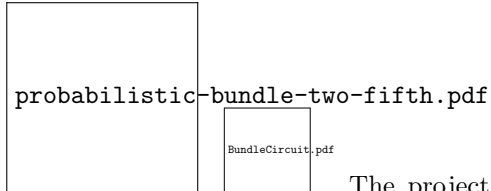
Statistical methods for performance analysis are well established in the realm of circuit design. These generally are applied to tighten performance bounds and to optimize for better nominal behavior. Techniques such as modular redundancy and majority voting have been investigated for fault tolerance. Such schemes require that the control circuitry for voting be fault-free. Fault tolerance through error correcting codes is widely applied for memory subsystems and communication links, both on-chip and off-chip. Again, the control circuitry for the error correction must operate reliably.

The approach taken in this project is fundamentally new. Instead of merely annotating signal values with statistical measures – that is, applying probabilistic techniques for analysis – the computation itself is structured probabilistically. This results in a dynamically tunable trade-off between computation time, accuracy, and reliability. Not only can variations and uncertainties be tolerated, *randomness can be exploited* in processor design and at the algorithmic level. Although the method entails redundancy in the encoding of signal values, complex operations can be performed using simple logic.

The project will develop, implement, and measure the performance of this stochastic paradigm from the level of logic through to the system architecture. A comprehensive methodology for logic synthesis and verification will be formulated; processor architectures that exploit the stochastic nature of the computation will be investigated; and architectures that are tailored to specific application domains, for instance randomized algorithms for scientific computing, will be explored.

3.1 Stochastic Logic

Although there has been intense research into new nanoscale devices and technologies, most are still in the exploratory phases, still years or decades from the point when they will be actualized [?]. Accordingly, the development of software tools and techniques for logic synthesis remains speculative. And yet, for some types of new technologies, we can identify broad traits that will likely impinge upon synthesis. For instance, nanowire arrays are stochastically self-assembled in tightly pitched bundles. Accordingly, they exhibit the following [?]: 1) minimal control during assembly; 2) inherent randomness in the interconnect schemes; and 3) high defect rates.



The project will develop a new strategy for synthesizing logic in nanoscale systems, targeting randomness and defects. Instead of computing with deterministic signals, in our approach operations at the logic level are performed on randomized values in serial streams or in parallel “bundles” of wires (an approach originally suggested by von Neumann [?]). When serially streaming, the signals are probabilistic in *time*; in parallel, they are probabilistic in *space*. Figure ?? illustrates the latter. The wires in the bundles are digital, carrying zeros and ones; they are processed by ordinary logic gates, such as AND and OR. However, the signal is conveyed through *statistical distribution* of the logical values. Real values in the interval $[0, 1]$ correspond to the fraction of wires with value one. For example, if half the wires in a bundle X have value one, then the signal is $p = 0.5$. If two-fifths are one, then it is $p = 0.4$.

With physical uncertainty, the fractional numbers correspond to *probability of occurrence* of logical one versus logical zero in an observation interval. In this way, computations in the **deterministic Boolean domain** are transformed into **probabilistic computations in the real domain**. This is illustrated in Figure ?. Our synthesis strategy is to cast logic computations as arithmetic operations in the probabilistic domain and implement these directly as stochastic operations on data-paths. For instance, two simple arithmetic operations – multiplication and scaled addition – are illustrated in Figures ?? and ??.

Given a specific Boolean function $y = f(x_1, x_2, \dots, x_n)$, stochastic logic is implemented as follows. We generate n independent stochastic bit streams X_1, X_2, \dots, X_n , each consisting of N bits. Each bit in the stream X_i equals 1 with independent probability p_{X_i} . The stream is fed into the corresponding input x_i . Thus, in a statistical sense, each bit stream represents a random Boolean variable. In this way, when we measure the rate of the occurrence of 1 in the output bit stream, it gives us an estimate of p_Y . If the bit stream is sufficiently long, this estimate becomes accurate. We assume that the input and the output of the circuit are directly usable in this fractionally weighted form and so be interpreted as probabilistic signals. For instance, in sensor applications, analog voltage discriminating circuits are used to transform real values to and from appropriated weighted sets of random bits on the wire bundles.

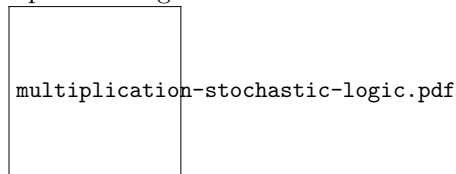


Figure 1: **Multiplication with stochastic logic.** The AND operation is performed on randomly shuffled pairs of wires from two bundles A and B , with the output assembled into a bundle C . Assuming that these signals are independent, it performs the *multiplication* of the corresponding signals a and b .

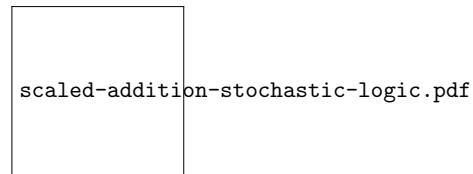


Figure 2: **Scaled addition with stochastic logic.** The MUX operation is performed on randomly shuffled pairs of wires from two bundles X and Y based on a selecting bundle S , with the output assembled into a bundle Z . It performs the *scaled addition* of the corresponding probability signals: x and y based on s .

3.2 Stochastic Datapaths

r3.0in

Polynomial.pdf

We will devise a general methodology for synthesizing stochastic logic for the computation of polynomial arithmetic functions, a category that is important for applications such as digital signal processing. Consider such an application that calls for arithmetic operations on datapaths (either integer or floating point values). In the design, we first scale the magnitude of the values on the datapath to the real interval $[0, 1]$. We translate the inputs into probability values, that is to say, we randomly assign bits on bundles at the correct fractional weighting. Now arithmetic operations can be performed directly on the probabilistic signals. This is illustrated in Figure ??.

Given a specification of a circuit in the Boolean domain, we compute a real-valued implementation of the arithmetic operation in the probabilistic domain. For the polynomials, multiplication and addition are synthesized with conjunctive operations (i.e., AND gates) and multiplexing operations (i.e., MUX gates) in the logical domain. Our methodology is based on converting polynomials into a particular mathematical form – Bernstein polynomials – and then implementing the computation with stochastic multiplexing [?]. (Recall that

$$B^n(t) = \sum_{i=0}^n b_i^n B_i^n(t), \quad \text{where}$$
$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}, \quad i = 0, \dots, n$$

is a *Bernstein polynomial* of degree n . The b_i^n 's are called *Bernstein coefficients*.)

converting-coefficients-berstein-polynomial-unit-interval.pdf

Figure 3: **Example of converting a power-form polynomial into Bernstein form.** It is well known that *any* power-form polynomial can be converted into an equivalent Bernstein polynomial. We have shown a new mathematical result: any Bernstein polynomial can be converted into an equivalent form *with all coefficients in the unit interval*. The conversion provides a constructive means of implementing stochastic logic through multiplexing.

graph-mcclaurin-stochastic-deterministic.pdf

Figure 4: **The error tolerance of deterministic vs. stochastic computation on evaluations of polynomials.** Sixth-order Maclaurin polynomial approximations of $\sin(x)$, $\tan(x)$, $\arcsin(x)$, $\arctan(x)$, $\sinh(x)$, $\tanh(x)$, $\operatorname{arcsinh}(x)$, $\cos(x)$, $\cosh(x)$, $\exp(x)$, and $\ln(x+1)$ with 10 bit resolution were synthesized. The error ratio ϵ is set to 0, 0.001, 0.002, 0.005, 0.01, 0.02, 0.05, and 0.1; this is the fraction of random bit flips that are injected into the input data. The figure shows the relative error in terms bit flips in the output.

In preliminary experiments, we have evaluated the cost and performance of our stochastic design methodology [?]. We compared the hardware cost of stochastic implementations to that of conventional deterministic implementations, as measured by the area-delay product of polynomial calculations. We also compared the performance of these two implementations on noisy input data. The experiments show that the stochastic approach produces circuits that are *much more tolerant of errors* in the input stream, while the area-delay product of the circuit is comparable to that of deterministic implementations. This is illustrated by the data in Figure ??.

It is not surprising that the deterministic implementation is so sensitive to errors, given that the representation used is binary radix. In a noisy environment, bit flips afflict all the bits with equal probability. In the worst case, the most significant bit gets flipped, resulting in relative error of $2^{M-1}/2^M = 1/2$ on the input value. In contrast, in a stochastic implementation, the data is represented as the fractional weight on

a bit stream of length 2^M . Thus, a single bit flip only changes the input value by $1/2^M$, which is minuscule in comparison.

3.3 Nanowire Crossbar Arrays

schemes similar to those used for field-programmable gate arrays (FPGAs). These rely on probing the circuit and programming interconnects after fabrication. In this project, we will develop a general method for synthesizing nanoscale logic that exploits both the parallelism and the random effects of the self-assembly, obviating the need for such post-fabrication configuration.

General features of existing strategies for synthesizing logic Figure 23 will be discussed below. Field-effect transistor (FET) like junctions are formed through a specific doping of nanowire arrays, controlled through doping during self-assembly [?]. When high or low voltages are applied to input nanowires, the FET-like junctions that cross these develop a high or low impedance, respectively. Because the doping regions for the junctions are randomly placed across the crossbar, the horizontal and vertical connections are random.

We exploit this randomness to implement arithmetic in the probabilistic domain: “shuffled AND” operations for multiplication and “bundleplexing” for scaled addition, corresponding to the operations in Figure ?? and ??, respectively. Preliminary synthesis trials show that our technique is effective in implementing designs with nanowire arrays, with a measured trade-off between the degree of redundancy and the accuracy of the computation [?].

3.4 Stochastic Architectures

At the system level, computing with unreliable components presents significant challenges. While modular redundancy and majority-voting can be used to mitigate errors, this method is only applicable when the modules have small numbers of components characterized by low error rates. Indeed, with large numbers of components and high error rates, nearly every module is likely to experience errors that no amount of voting can compensate for. Furthermore, the system is vulnerable to faults in the majority-voting circuitry [?, ?]. With our probabilistic representation, failure resistance can be achieved much more transparently.

In preliminary work, we have studied the implementation of image processing algorithms such as edge detection and the Discrete Fourier Transform with stochastic processing. We measured the percent output (in pixels on the resulting image) that are in error with the stochastic implementation compared to a deterministic one. As expected, the error introduced by the stochastic computation decreases as the number of bits used per pixel increases. If we inject faults as random bit flips in the logic, the error increases, but proportionately, as shown in Figure ??.

At the application level, we will study hardware/software codesign techniques that leverage the probabilistic nature of the computation. We will use Verilog HDL-based simulations to study the trade-offs between computation time, accuracy, and reliability for stochastic implementations. We will first evaluate operations from benchmark circuits, say for digital signal processing applications. Next we will implement and evaluate a complete OpenRISC processor in this paradigm. Finally, we will implement a behavioral high-level synthesis flow for stochastic processing.

3.5 Randomized Algorithms

Randomness in the underlying architecture presents both challenges and oppor-

tunities for the application layer: on the one hand, uncertainty in the execution flow precludes applications with outcomes that hinge on particular branch points (for instance, controllers); on the other hand, if properly characterized and circumscribed, probabilistic execution can be leveraged for applications that are data-intensive yet statistical in nature.

Randomized algorithms such as the fingerprinting method will be investigated. Consider an architecture in which unreliable cores perform redundant computations very fast – say, $O(n^3)$ matrix multiplication

operations $C = A \times B$ – and a slower, reliable core performs a “gather” operation using fingerprinting methods – requiring, say $O(n)$ time. An example of fingerprinting for the $C = A \times B$ computation is to randomly choose a vector $\alpha \in \{0, 1\}^n$, and compute $\beta = A(B\alpha)$ and $\gamma = C\alpha$. If $\beta \neq \gamma$, output “ $C \neq A \times B$.” By repeating this process K times, we get a high confidence in detecting errors.

We will study the implications of stochastic processing for algorithms at the application layer and develop methods for hardware/software co-design. We will aim to provide bounds on the accuracy and quality of solutions generated by specialized algorithms run on machines characterized by permanent and transient error rates. This will include a study of the trade-offs in resources, time complexity, and the quality of solutions.

4 Design Automation for Synthetic Biology

The scope of recent research in synthetic biology is nothing short of Promethean. The J. Craig Venter Institute’s team has made significant progress toward the goal of artificial life: a living bacterial cell with fully synthetic DNA [?, ?]. In engineering terms, the objective is to assemble a machine (a synthetic bacterium) in which the functionality of all the parts (the genes, the proteins that they code for, and how these interact biochemically) are understood. If the machine works, this vindicates the scientific understanding; if it doesn’t – and surely it won’t at first – then new understanding can be achieved by examining where and how it breaks. Of course, with a working blueprint for a synthetic machine, new functionality can be engineered robustly and effectively.

The set of constitutive parts that can be used for genetic manipulation in synthetic systems is vast. Comprehensive repositories of genetic data have been assembled – some public, some commercial – cataloging genes, their DNA sequences, and their products [?]. A concerted effort has been made to assemble repositories of standardized and interoperable parts for synthetic applications [?]. The platforms used will depend on the application, but the technology for synthesizing DNA is becoming routine: in late 2007, firms started offering custom-gene synthesis through e-commerce websites; the going rate is \$0.49 per base pair [?].

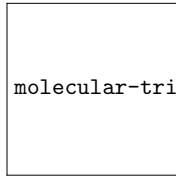
So, in a real sense, the **hardware** for synthetic biology exists, i.e., the technology and infrastructure for obtaining cells with custom-designed genes. The **instruction set** is, to a large extent, known, i.e., genes and their function, cataloged in libraries. The challenge is: *how can we write code with these instructions on this type of hardware?*

4.1 Design Abstractions

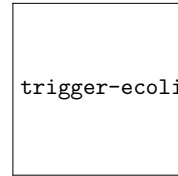
One of the great successes of integrated circuit design has been in *abstracting* and *scaling* the design problem. The physical behavior of transistors is understood in terms of differential equations – say, with models found in tools such as SPICE [?]. However, the design of circuits proceeds at a more abstract level – in terms of switches, gates, and functional units. This modular approach makes design tractable; furthermore, it permits a systematic exploration of different configurations, leading to optimal designs.

Although driven by experimental expertise, synthetic biology has reached a stage where it calls for a similar automated design flow. This would allow for virtual experimentation: one could vary the inputs and parameters of synthetic designs and observe the outputs – in a manner analogous to traditional *in vitro* and *in vivo* experimentation. (This has been dubbed, somewhat facetiously, as “*in silico*” experimentation [?].) Beyond analysis, by deliberately applying design methodologies, one could engineer **computational control** over biological processes, designing pathways that produce specific outputs in response to different combinations of inputs.

Discrete Biochemistry: Interesting biochemistry typically involves complex molecules such as proteins and enzymes. Within the confines of a cell, the *quantities* of such molecules are often surprisingly small: on the order of tens, hundreds, or thousands of molecules of each type. At this scale, individual reactions matter, and the problem must be analyzed discretely [?].



molecular-triggers-ecoli-molecular-products.pdf

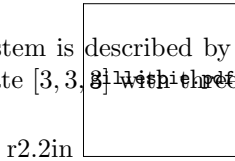


trigger-ecoli-probability.pdf

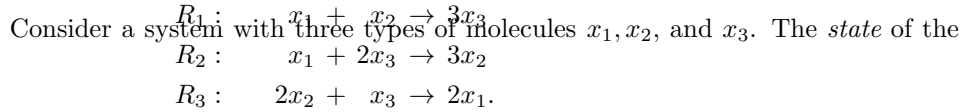
Figure 5: A illustration of biochemistry that computes output *quantities* of molecular products, such as proteins, in response to input *quantities* of molecular triggers.

Figure 6: A illustration of biochemistry that computes *probabilities* on the production of *fixed* quantities of products in response to input *quantities* of molecular triggers.

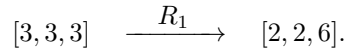
system is described by the number of molecules: $[|x_1|, |x_2|, |x_3|]$, For instance, the system might be in the state $[3, 3, 3]$ with three molecules of each type. Consider the three reactions:



r2.2in



Note that these reactions are *coupled*: the types appear both as reactants and as products in different reactions. Suppose that the system is in the state $[3, 3, 3]$ and reaction R_1 fires. One molecule of type x_1 and one of type x_2 are consumed; three of type x_3 are produced. This results in the state transition:



As reactions fire, a cellular process follows a sequence of such transitions. Figure ?? illustrates the trajectory taken from the state $[3, 3, 3]$ by the sequence R_1, R_2 , and R_3 .

Probabilistic Biochemistry: Randomness is inherent: at each instant, the exact sequence of reactions that fires next is a matter of chance. Indeed, ignoring environmental changes outside the cell, one can assume cellular biochemistry behaves as a *Markov process*: the probability of future events depends only on the present state of the cell. At each point in time, the probability of a given reaction firing is a function of the quantities of different types of molecules present. Specifically, it is proportional to: 1) the number of ways that the reactants can come together; and 2) and the reaction *rate*. Suppose that the system in the example above is in the state $S = [3, 4, 5]$. There are

$$3 \times 4 = 12, \quad 3 \times \binom{5}{2} = 30, \quad \binom{4}{2} \times 5 = 30$$

ways to choose the reactants of R_1, R_2 , and R_3 , respectively. Suppose that the rates of reactions R_1, R_2 , and R_3 are 1, 2 and 3, respectively. Then the firing probabilities for R_1, R_2 , and R_3 are

$$\frac{12 \times 1}{162} = 0.074, \quad \frac{30 \times 2}{162} = 0.370, \quad \frac{30 \times 3}{162} = 0.556, \quad \text{respectively.}$$

Although we will often refer to rates in relative and qualitative terms – e.g., “fast” vs. “slow” – these are, in fact, real-valued parameters that are either deduced from biochemical principles or measured experimentally [?].

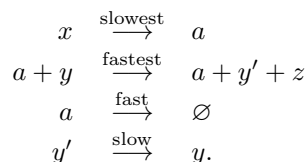
Computationally, such discrete probabilistic biochemical systems are characterized through Monte Carlo simulation [?], [?], [?]. Beginning from an initial state, reactions are chosen at random, based on propensity calculations. As reactions fire, the quantities of the different species change by integer amounts. Repeated trials are performed and the probability distribution of different outcomes is estimated by averaging the results.

4.2 A Biochemical Toolkit

This project will develop a modular and extensible design flow for implementing biochemical computation. Synthesis first will be performed at a conceptual level, in terms of abstract biochemical reactions – a task

analogous to **technology-independent logic synthesis**. Then the results will be mapped onto specific biochemical components, selected from libraries – a task analogous to **technology mapping**. A flexible toolkit of functional modules will be assembled: these include modules for standard arithmetic operations (analogous to those performed by an arithmetic-logic unit in a microprocessor system) as well as regulatory functions (analogous to those performed by control circuitry). Arithmetic computations will be both deterministic, as illustrated in Figure ??, and probabilistic, as illustrated in Figure ??.

r3.0in



Example: multiplication. Suppose that we wish to design a module that produces an output quantity of a type z that is proportional to the input quantities of both type x and type y , $|z| = |x| |y|$. The set of reactions in Figure ?? accomplish this.

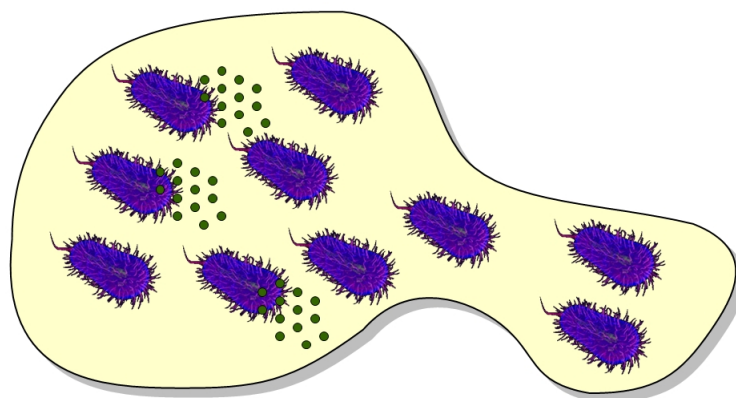
To see that these reactions implement multiplication, note that no reaction can fire until the first one does, producing a molecule of type a . When it does, it initiates an iteration of a *loop*: the quantity of z increases as the second reaction fires repeatedly until there is no more y remaining. Once this process terminates, the third and fourth reactions fire, ending the iteration and restoring y to its initial value. In each iteration, the quantity of x is decremented by one and the quantity of z is incremented by y . The final result is a quantity of z equal to the initial quantity of x times the initial quantity of y .

We have developed modules for computing a variety of functions: **multiplication**, **exponentiation**, **logarithms**, etc. [?]. With our linear and raising-to-a-power modules, our scheme can be used to implement *arbitrary* polynomial functions; hence, in principle, it could be used to approximate complex functions through Taylor series expansions. In this project, we will extend the methodology to arbitrary curve fitting of data points [?].

4.3 Engineering Stochasticity

Randomness has been well characterized in biological systems [?], [?]. Certain biochemical systems appear to exploit randomness, choosing between different outcomes with a probability distribution – in effect, hedging their bets with a portfolio of responses. Examples include the *pap pili* epigenetic response of bacteria [?], the lentiviral positive-feedback loop in the HIV virus [?], and the lysis/lysogeny switch of the *lambda* bacteriophage [?, ?]. As in natural systems, randomness plays a pivotal role in synthetically engineered systems.

r3.5in



cell-programmable-probability-response.pdf

A specific aim of

the project is to develop methods for synthesizing biochemistry with robust probabilistic outcomes: the production of molecular products according to specified probability distributions [?]. Consider the following design problem, in the vein of recent research by Chris Voigt’s group at UCSF [?]. Suppose that bacteria are engineered to invade tumors and produce a drug to kill the cancer cells. The bacteria are engineered

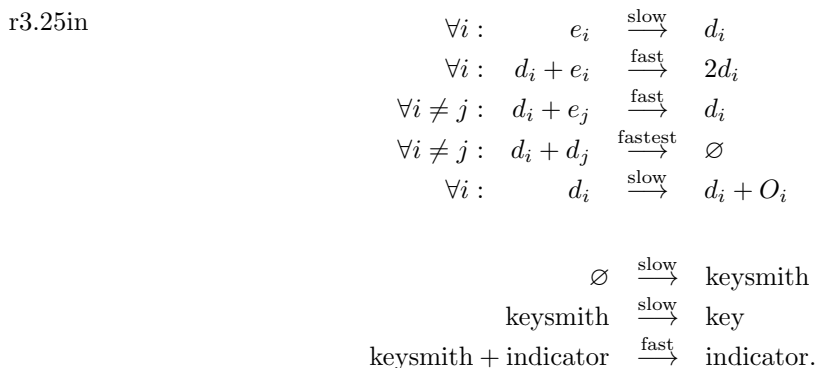
to produce the drug in response to a triggering compound that is injected into the cancerous tissue. Each bacterium produces a fixed quantity of the drug. We cannot control the population density nor what fraction of the bacteria are exposed to the triggering compound. How can we calibrate the amount of drug that is produced?

The solution is to engineer a **fractional** response across the population. If say, m out of the population of n respond, then a fraction $\frac{m}{n}$ of the total amount of drug will be produced. This can be accomplished if we engineer a **probabilistic** response: each bacterium produces the drug with *probability* $\frac{m}{n}$. This is illustrated in Figure ???. This need not be a fixed probability; rather it can have a functional dependence on the input quantity. More generally, we can engineer logical combinations of outcomes according to such programmable probability distributions. This is illustrated in Figure ???.

We will develop methods for designing biochemical reactions that produce different combinations of molecular types according to specified probability distributions [?]. An abbreviated description of a stochastic module is given in Figure ???. The response to this system is precise and robust to perturbations. Furthermore, it is programmable: the probability distribution is a function of the quantities of input types [?]. Our contribution is design automation for biochemical synthesis at the level of abstract arbitrary types (a, b, c , etc.) – “technology-independent synthesis.” The solution is then mapped to specific types and reactions from a suitable toolkit [?], [?].

4.4 Rate-Independent Biochemical Synthesis

An important constraint in our design methodology is the *timing*, captured in the relative rates of the biochemical reactions. Both for intra- and intermodule computation, strict synchronization is often required: all the outputs of a given phase must be produced before the next phase can begin consuming them. To achieve this synchronization, the reaction rates must sometimes be separated by orders of magnitude: some much faster than others, some much slower. This might be costly or infeasible given a specific library of biochemical reactions.



In this project, we will explore a more efficient scheme for synchronization that we call *locking* [?]. It entails adding reactions involving a specific molecular type to each module – the module’s *key*. Without the key, the sequence of reactions in the module is prevented from firing. Each module’s key is produced by a reaction involving another type – the module’s *keysmith*. The keysmiths for the different modules are produced slowly and randomly. When one pops into existence, it is generally consumed by another reaction involving types called *indicators*. If none of these indicators are present, then it is this module’s turn to execute. Only then does the keysmith produce the key for the corresponding module. If there are multiple input dependencies in a module, we will need an indicator and an associated reaction for each. The set of reactions is shown in Figure ???.

This mechanism for locking is, in fact, analogous to *handshaking* mechanisms in asynchronous circuit design, with the creation of a keysmith acting as a probe and the generation of a key being an affirmative response [?]. With locking, our method synthesizes robust computation that is nearly rate independent, requiring at most two speeds (“fast” and “slow”). The trade-off is with respect to the size of the solution: more reactions are needed.

The first step for module locking is to identify all molecular types that indicate that the locked module should not be firing. Then, the reactions outlined in Equation ?? are added. Finally, the loop-initiator reaction is changed so that it requires the appropriate key, that is, one that will not be produced in the

presence of the specified indicators. This is illustrated with an example in Figures ?? and ?. Curves of the probabilistic response for both the locked and unlocked versions are plotted for different rates Figure ?. We see how effective locking is, even if “fast” is only twice as fast as “slow.”

4.5 Information-Theoretic Measures for Biology

Although Monte Carlo simulations faithfully reproduce the physical behavior of a system, they are computationally prohibitive. Furthermore, by averaging the results, such simulations produce only coarse-grained statistics that provide little insight into the dynamics of the stochastic behavior of biological systems.

In this project, we will explore metrics for characterizing the state space. In preliminary work, we have described a conceptually simple, yet effective, metric: the *probability gradient* [?]. For each point in the state space, the probability gradient is a vector corresponding to the *expected value* of transitions. More precisely, consider a system consisting of N types of molecules $\{X_1, \dots, X_N\}$. The state of the system is

$$[x_1, \dots, x_N] \in \mathbb{N}^N,$$

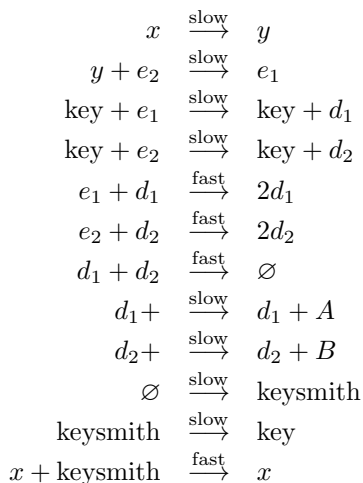
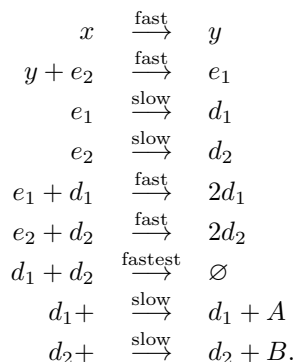
where x_i is the number of molecules of type X_i . Suppose that, from a given state \vec{S} , there are M possible transitions. Suppose that the j -th transition occurs with probability p_j and is characterized by the vector

$$\vec{V}_j = [v_{1,j}, \dots, v_{N,j}] \in \mathbb{Z}^N,$$

where $v_{i,j}$ is the change in the number of molecules of type X_i that this transition produces. The gradient is computed as

$$\vec{G} = \sum_{j=1}^M p_j \vec{V}_j.$$

r3.0in



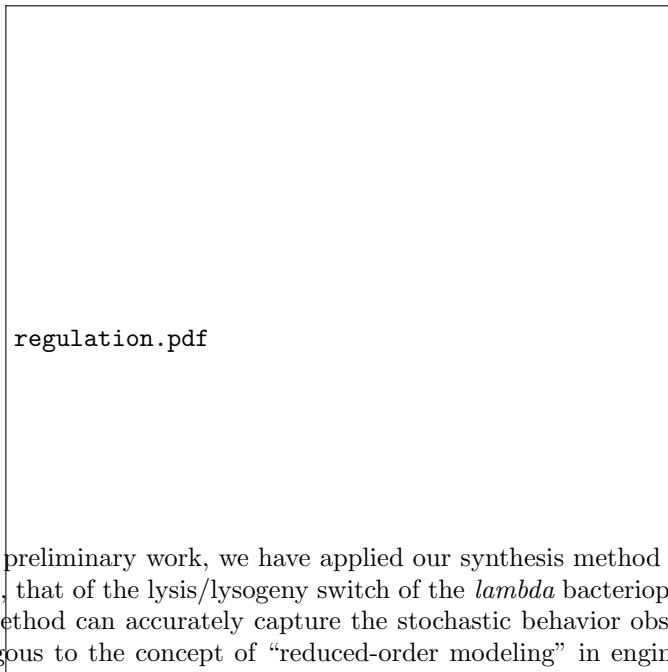
order gradient, we consider the transitions resulting from all possible sequences of m reactions firing.

For the *first-order* gradient, we consider the transitions resulting from a single reaction firing; for the m -th

The gradient allows us to characterize the topology of the state space. If the gradient is small in magnitude, then we are likely in a “flat” region with considerable uncertainty in the outcome; if it is large in magnitude, then we are likely in a “canyon” progressing rapidly toward a decision. If the outcome is nearly certain, then a trajectory can be terminated early, reducing the computation time. Furthermore, profiling the probability gradient allows us to detect abrupt changes in the topology – indicating the decision points of the system.

Many biological networks – metabolic pathways, protein-protein interactions, and transcriptional mechanisms – are commonly modeled as random graphs. Here, the degree distribution $P(k)$ is the probability that a “node” (gene) of the network is connected to exactly k other nodes. Networks with this property are known as scale-free networks, as the relative probability for two different connectivities k depends only on the ratio of the connectivities, rather than on their absolute values. Percolation theory for such random graphs tells us that there is sharp non-linearity in the global connectivity as a function of the local connectivity. We will study this phenomenon from the perspective of *digital computation* with biological networks; we will be drawing on the analogy with integrated circuits, where non-linearities in semiconductors serve as the basis for switching functions for digital computation.

4.6 Applications: Modeling the λ Bacteriophage



In preliminary work, we have applied our synthesis method to fit the data from a well-known biological model, that of the lysis/lysogeny switch of the *lambda* bacteriophage [?, ?]. Our goal is to demonstrate that our method can accurately capture the stochastic behavior observed in natural biological systems. This is analogous to the concept of “reduced-order modeling” in engineering analysis: the input/output behavior is maintained while the internal dynamics are lost. We do not present any biological interpretation of the results. However, we point out that our synthetic model is more compact and perhaps more robust to specific perturbations than the natural model.

Natural Model: The *lambda* bacteriophage is a virus that infects the *E. coli* bacteria. It chooses one of two survival strategies: either it integrates its genetic material with that of its host and then replicates when the bacterium divides (termed *lysogeny*); or it manipulates the molecular machinery of its host to make many copies of itself, killing the bacterium in the process, and thereby releasing its progeny into the environment (termed *lysis*) [?].

The biological model for this behavior consists of an elaborate set of 117 reactions in 61 molecular types [?]. We focus on a single input type, *MOI*, that plays a crucial role in the viral response. (It correlates with the number of copies of the virus that have infected the cell.) We analyze the probabilistic response for the production of two output types, *cro2* and *ci2*. The production of the former corresponds to lysis; the production of the latter to lysogeny. (The outcomes are judged according to threshold values: 55 for *cro2* and 145 for *ci2*.)

We characterized the probabilistic response of the model with Monte Carlo simulations. Sweeping the input type *MOI* across a range of values, we recorded the percentage of trials that resulted in each outcome. From this data, we performed a curve fit. The result:

$$P(\textit{lysis}) = 15 + 6 \log_2(\textit{MOI}) + \frac{\textit{MOI}}{6}.$$

The data points and the curve fit are shown in Figure ???. We assume that one or the other outcome always occurs, so $P(\textit{lysogeny}) = 1 - P(\textit{lysis})$.

obtain a model with 19 reactions in 17 types. Deterministic modules produce the linear and logarithmic dependence. A stochastic module produces the probabilistic response. Some simple additional reactions are used to glue the modules together. We characterized the probabilistic response of the synthetic model with Monte Carlo simulations. The results are shown in Figure ???. As can be seen, it implements a close fit to the natural model.

In this project, we will explore the implementation of our synthesis methodology with parts from the MIT BioBricks repository [?]. In addition to the potential applications in synthetic biology, we are interested in characterizing the stochastic dynamics of natural biological systems in a rigorous and systematic way – applying engineering concepts such as abstraction and reduced-order modeling. We will study a number of other biological models, including the lentiviral positive-feedback loop in the HIV virus [?] and the pheromone-response pathway in yeast [?].

4.7 High-Performance Computing for Synthetic Biology

Both analysis and synthesis are computationally intensive tasks. Biochemical systems are typically characterized through Monte Carlo simulations that track the minutiae of the reactions that are executing [?, ?]. Such simulations can be very lengthy because there are a multitude of reactions happening simultaneously in a cell. Each trial might consist of millions of reaction events; thousands of such trials must be performed in order to get an accurate estimate. For synthesis problems, mapping conceptual designs onto specific biochemical reactions entails a combinatorial search. An exhaustive search of the combinatorial space is an intractable proposition; heuristics and high-performance computing are needed to formulate optimal designs.

At present, computation time severely limits the application of stochastic simulation [?]. The computational burden stems from the physical nature of the problem: in a cell, vast numbers of chemical reactions happen nearly simultaneously. Stochastic simulation tracks each and every one of these reactions. If implemented sequentially, each trajectory in a simulation becomes impossibly long. We will consider two approaches to parallelizing stochastic simulation: 1) Run separate trajectories on each processor and communicate and aggregate the statistics on a central node. 2) Partition the state space dynamically into regions and assign these to different processors, handing off trajectories from processor to processor.

Molecular processes are not always independent of spatial constraints; they sometimes occur on the surface of membranes or in highly confined areas, in the presence of chemical gradients. We will explore simulation techniques that incorporate *reaction-diffusion* gradients. With this approach, parallelization could be applied *spatially*, with the cell volume divided into compartments, each assigned to a different processor. Gradients could be modeled without the computational blow-up that is incurred when tracking individual molecules [?].

The principal investigator has an existing collaboration with the Blue Gene Development team at IBM Rochester, Minnesota. IBM will contribute both infrastructure and associated software expertise to code development on high-performance computing platforms. Stochastic simulation will be implemented on IBM's Cell Broadband Engine (Cell BE) processor as well as on IBM's Blue Gene Supercomputer. Emphasis will be placed on accelerating key mathematical routines that best leverage Cell BE's capability. Design alternatives will be considered for code hotspots. The code will be designed to leverage hardware architectural advantages such as parallelism, pipelining, and vector processing fundamentals.

5 Educational and Outreach Plan

“This isn’t jargon – it’s what things are called!” – David Baltimore, 2001

The project will communicate the goals and the impetus for interdisciplinary research to a wide audience, including students in the sciences and engineering as well as the general public. This will be achieved through curriculum development, writing, seminars, and workshops.

5.1 ECE/BICB Graduate Course and Book

A new graduate-level course titled “**Circuits, Computation, and Biology**” will be developed and offered jointly through the Electrical and Computer Engineering Department and the Biomedical Informatics and Computational Biology Program at the University of Minnesota and the Mayo Clinic. It will explore connections between engineering concepts — circuit theory, digital computation, and distributed computing in particular — and biological systems. A broad theme is the application of expertise from the former to the latter — specifically, the application of algorithmic and computational expertise from circuit design to the analysis and synthesis of biochemical and neural systems. It will be aimed at a wide audience; no prior knowledge of engineering or biology will be assumed. It will investigate a variety of topics from disparate fields but will not attempt to survey the research exhaustively. Rather, it will strive for depth and mathematical rigor in select areas. In parallel with the course, a book with same title will be written and published by Pan Stanford Publishing in 2010.

5.2 Collaborative Workshop on Technical Writing

The project includes an annual workshop on writing skills that will be organized annually at the University of Minnesota. The workshop is titled “SoC/ASIC, Induction of Apoptosis, . . . , *what the heck?* Impenetrable Jargon in Interdisciplinary Research.” (The former is an example of a vapid acronym from circuit design: it stands for “system-on-a-chip/application-specific integrated circuit”; the latter is an example of vapid jargon in biology: it means “causes cell death.”) For the workshop, researchers writing in interdisciplinary areas will be asked to submit technical papers. These will be critiqued by students in the journalism program at Minnesota State University.

5.3 Seminar Series on Public Engagement in Science

Engagement will also include public seminars discussing the nature of interdisciplinary research in a **Café Scientifique** series hosted by the Bell Museum of Natural History in Minneapolis. The format permits in-depth presentations but also encourages informal dialogue. The seminars will discuss design automation for nanotechnology and synthetic biology, as they are seen through the narrow lens of our research activities. They will also discuss these topics more broadly, as they are seen through the wide lens of public interest in science. We will discuss the ethical and security concerns swirling around research in synthetic biology; the social climate of research in biology versus engineering versus mathematics; and ways that we can interpret circuit design practices as a window into the potential achievements and limitations of human cognition.

5.4 K–12 Teaching

The activities of this project will be pursued in conjunction with a program called “Research Experiences for Teachers” (RET) at the University of Minnesota, which provides funding to support K–12 teachers to work on research projects in the summer. Through this program, K–12 teachers will be introduced to computational research in biology, including computer-aided design and simulation tools.

5.5 Undergraduate Research

The activities of this project will be pursued in conjunction with a program called “Research Experiences for Undergraduates” (REU) through the ECE Department and through the Minnesota Supercomputing Institute’s summer internship program. Undergraduates will work on “front-end” database systems of biochemical models as well as designing “back-end” number-crunching algorithms for analysis and synthesis on a farm of high-performance processors. Specifically, they will structure a relational database of biochemical parts, from MIT’s BioBricks repository; also, they will implement routines for mapping conceptual designs for synthetic biology onto this library of parts. These routines will be parallelized for IBM’s Cell BE chip and implemented on a Sony Playstation 3, as a precursor to higher powered parallel platforms such as IBM’s Blue Gene Supercomputer.

5.6 Additional Outreach to Underrepresented Groups

To ensure the participation in this research of students from traditionally underrepresented groups, we will work closely with existing recruitment programs at the University of Minnesota. In particular, we will recruit through college-wide program **Academic Programs for EXcellence in Engineering and Sciences** (APEXES).

6 Prior NSF Support

None.

References

- [1] J. Anderson, E. Clarke, A. Arkin, and C. Voigt, "Environmentally Controlled Invasion of Cancer Cells by Engineered Bacteria," *Journal of Molecular Biology*, Vol. 355, No. 4, pp. 619–627, 2006.
- [2] A. Arkin, J. Ross, and H. McAdams, "Stochastic Kinetic Analysis of Developmental Pathway Bifurcation in Phage λ -Infected *E. coli* Cells," *Genetics*, Vol. 149, No. 1633, 1998.
- [3] Z. Asgar, S. Kodakara and D. Lilja, "Fault-tolerant image processing using stochastic logic," *UMN ECE Technical Report*, 2005.
- [4] J. Backes, B. Fett, and M. Riedel, "The Analysis of Cyclic Circuits with Boolean Satisfiability," *International Conference on Computer-Aided Design*, 2008.
- [5] Y. Benenson, B. Gil, U. Ben-Dor, R. Adar, and E. Shapiro, "An Autonomous Molecular Computer for Logical Control of Gene Expression," *Nature*, Vol. 429, pp. 423–429, 2004.
- [6] MIT Registry of Standard Biological Parts.
- [7] The BioCyc Collection of Pathway/Genome Databases.
- [8] C. Bishop, *Neural networks for pattern recognition*, Clarendon Press, 1995.
- [9] B. Brown and H. Card, "Stochastic neural computation I: computational elements," *IEEE Transactions on Computers*, Vol. 50, No. 9, pp. 891–905, 2001.
- [10] A. DeHon, "Nanowire-Based Programmable Architectures," *ACM Journal on Emerging Technologies in Computing Systems*, Vol. 1, No. 2, pp. 109–162, 2005.
- [11] S. Deiss, R. Douglas and A. Whatley, "A pulse coded communications infrastructure for neuromorphic systems," *Pulsed Neural Networks*, W. Maass and C.M. Bishop, eds., Chapter 6, MIT Press, 1999.
- [12] D. Endy and R. Brent, "Modelling Cellular Behaviour," *Nature*, Vol. 409, pp. 391–395, 2001.
- [13] D. Endy, "Foundations for Engineering Biology," *Nature*, Vol. 438, pp. 449–453, 2005.
- [14] FENA Mission Statement, MARCO Center Functional Engineered Nano Architectonics (FENA), 2008.
- [15] B. Fett and M. Riedel, "Characterizing the Decisions of Biochemical Systems with Probability Gradients," *International Conference on Systems Biology*, Dec. 2006.
- [16] B. Fett, J. Bruck, and M. Riedel, "Synthesizing Stochasticity in Biochemical Systems," *Design Automation Conference*, Vol. 44, pp. 640–645, 2007.
- [17] B. Fett and M. Riedel, "Module Locking in Biochemical Synthesis," *International Conference on Computer-Aided Design*, 2008.
- [18] Firms performing DNA synthesis commercially: Invitrogen, Genscript, and Mr Gene.
- [19] M. Gibson and J. Bruck, "Efficient Exact Stochastic Simulation of Chemical Systems with Many Species and Many Channels," *Journal of Physical Chemistry A*, No. 104, pp. 1876–1889, 2000.
- [20] M. Gibson, "Computational Methods for Stochastic Biological Systems," *Ph.D. Dissertation*, California Institute of Technology, 2001.
- [21] D. Gibson, G. Benders, C. Andrews-Pfannkoch, E. Denisova, H. Baden-Tillson, J. Zaveri, T. Stockwell, A. Brownley, D. Thomas, M. Algire, C. Merryman, L. Young, V. Noskov, J. Glass, J. Venter, C. Hutchison, and H. Smith, "Complete Chemical Synthesis, Assembly, and Cloning of a *Mycoplasma genitalium* Genome," *Science*, Vol. 319, No. 5867, pp. 1215–1220, 2008.

- [22] D. Gillespie, "Exact Stochastic Simulation of Coupled Chemical Reactions," *Journal of Physical Chemistry*, Vol. 81, No. 25, pp. 2340–2361, 1977.
- [23] J. Glass, N. Assad-Garcia, N. Alperovich, S. Yooseph, M. Lewis, M. Maruf, C. Hutchison, H. Smith, and J. Craig Venter, "Essential Genes of a Minimal Bacterium," *Proceedings of the National Academy of Sciences*, Vol. 103, No. 2, pp. 425–430, 2006.
- [24] R. Heinrich and S. Shuster, "The Regulation of Cellular Systems," *Chapman and Hall*, 1996.
- [25] A. Hernday, B. Braaten, and D. Low, "The Intricate Workings of a Bacterial Epigenetic Switch," *Advances in Experimental Medicine & Biology*, Vol. 547, No. 83-9, 2004.
- [26] I. Herskowitz, "Life Cycle of the Budding Yeast *Saccharomyces cerevisiae*," *Microbiological Reviews*, Vol. 52, No. 4, pp. 536–553, 1988.
- [27] C. Hescott, D. Ness, D. Lilja, "Scaling Analytical Models for Soft Error Rate Estimation Under a Multiple-Fault Environment," *Euromicro Conference on Digital System Design*, 2007.
- [28] The International Technology Roadmap for Semiconductors.
- [29] E. Libby, T. Perkins, and P. Swain, "Noisy information processing through transcriptional regulation," *Proceedings of the National Academy of Sciences*, Vol. 104, No. 17, pp. 7151–7156, 2007.
- [30] L. Lok, "The Need for Speed in Stochastic Simulation," *Nature Biotechnology*, Vol. 22, No. 8, 2004.
- [31] L. Lok and R. Brent, "Automatic Generation of Cellular Reaction Networks with Molecuizer 1.0," *Nature Biotechnology*, Vol. 23, 2005.
- [32] H. McAdams and A. Arkin, "Stochastic Mechanisms in Gene Expression," *Proceedings of the National Academy of Sciences*, Vol. 94, No. 3, 1997.
- [33] C. Myers, "*Asynchronous Circuit Design*," John Wiley and Sons, 2001.
- [34] L. Nagel and D. Pederson, "Simulation Program with Integrated Circuit Emphasis," *Midwest Symposium on Circuit Theory*, 1973.
- [35] D. Ness and D. Lilja, "Guiding Circuit Level Fault-Tolerance Design with Statistical Methods," *Design, Automation, Test Europe (DATE)*, 2008.
- [36] S. Paliwal, P. Iglesias, K. Hilioti, A. Groisman, and A. Levchenko, "MAPK-mediated bimodal Gene Expression and Adaptive Gradient Sensing in Yeast," *Nature*, Vol. 446, Issue 7131, pp. 46–51, 2007.
- [37] W. Qian and M. Riedel, "The Synthesis of Stochastic Circuits for Nanoscale Computation," *International Workshop on Logic and Synthesis*, pp. 176–183, 2007.
- [38] W. Qian and M. Riedel, "The Synthesis of Robust Polynomial Arithmetic with Stochastic Logic," *Design Automation Conference*, pp. 648–653, 2008.
- [39] M. Riedel and J. Bruck, "The Synthesis of Cyclic Combinational Circuits," *Design Automation Conference*, **Best Paper Award**, 2003.
- [40] M. Riedel and J. Bruck, "Exact Stochastic Simulation with Event Leaping," *International Conference on Systems Biology*, Oct. 2005.
- [41] M. Riedel, "Synthesizing Stochasticity in Biochemical Systems," *Synthetic Biology 3.0*, 2007.
- [42] M. Riedel, "The Computer-Aided Synthesis of Stochasticity in Biochemical Systems," *Advances in Synthetic Biology*, 2008.
- [43] M. Riedel, "Cyclic Combinational Circuits," Ph.D. Dissertation, California Institute of Technology, **Best Dissertation in E.E. Award**, 2004.

- [44] D. Ro, E. Paradise, M. Ouellet, K. Fisher, K. Newman, J. Ndungu, K. Ho, R. Eachus, T. Ham, M. Chang, S. Withers, Y. Shiba, R. Sarpong, and J. Keasling, "Production of the Antimalarial Drug Precursor Artemisinic Acid in Engineered Yeast," *Nature*, Vol. 440, pp. 940–943, 2006.
- [45] M. Sedlak and N. Ho, "Production of Ethanol from Cellulosic Biomass Hydrolysate Using Genetically Engineered Yeast," *Applied Biochemistry & Biotechnology*, Vol. 114, No. 1–3, pp. 403–416, 2004.
- [46] E. Voit, "Computational Analysis of Biochemical Systems: A Practical Guide for Biochemists and Molecular Biologists," Cambridge University Press, 2000.
- [47] L. Weinberger, J. Burnett, J. Toettcher, A. Arkin, and D. Schaffer, "Stochastic Gene Expression in a Lentiviral Positive-Feedback Loop: HIV-1 Tat Fluctuations Drive Phenotypic Diversity," *Cell*, Vol. 122, pp. 169–182, 2005.
- [48] E. Zielinska, "Chris Voigt: Biology's Toy Maker," *The Scientist*, Vol. 21, No. 9, 2007.
- [49] J. von Neumann, "Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components," in *Automata Studies*, C.E. Shannon and J. McCarthy (Eds.), Princeton Univ. Press, pp. 329–378, 1956.