Fast Simulation of Stochastic Biochemical Reaction Networks on Cytomorphic Chips

by

Sung Sik Woo

B.S., Electrical Engineering
KAIST (2009)
S.M., Electrical Engineering and Computer Science
Massachusetts Institute of Technology (2012)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2016

Author .....
Department of Electrical Engineering and Computer Science
August 29, 2016

Certified by .....
Rahul Sarpeshkar
Thomas E. Kurtz Professor, Dartmouth College, and Visiting Scientist,
Research Laboratory of Electronics, MIT
Thesis Supervisor

Accepted by .....
Leslie A. Kolodziejski
Chair, Department Committee on Graduate Students

Fast Simulation of Stochastic Biochemical Reaction Networks on Cytomorphic Chips

by

Sung Sik Woo

Submitted to the Department of Electrical Engineering and Computer Science
on August 29, 2016, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Electrical Engineering and Computer Science

Abstract

The large-scale simulation of biochemical reaction networks in cells is important in pathway discovery in medicine, in analyzing complex cell function in systems biology, and in the design of synthetic biological circuits in living cells. However, cells can undergo many trillions of reactions over just an hour with multi-scale interacting feedback loops that manifest complex dynamics; their pathways exhibit non-modular behavior or loading; they exhibit high levels of stochasticity (noise) that require expensive Gillespie algorithms and random-number generation for accurate simulations; and, they routinely operate with nonlinear statics and dynamics. Hence, such simulations are extremely computationally intensive and have remained an important bottleneck in computational biology over decades.

By exploiting common mathematical laws between electronics and chemistry, this thesis demonstrates that digitally programmable analog integrated-circuit 'cytomorphic' chips can efficiently run stochastic simulations of complex molecular reaction networks in cells. In a proof-of-concept demonstration, we show that 0.35 μm BiCMOS cytomorphic gene and protein chips that interact via molecular data packets with FPGAs (Field Programmable Gate Arrays) to simulate networks involving up to 1,400 biochemical reactions can achieve a 700x speedup over COPASI, an efficient biochemical network simulator. They can also achieve a 30,000x speedup over MATLAB. The cytomorphic chips operate over five orders of magnitude of input concentration; they enable low-copy-number stochastic simulations by amplifying analog thermal noise that is consistent with Gillespie simulations; they represent non-modular loading effects and complex dynamics; and, they simulate zeroth, first, and second-order linear and nonlinear gene-protein networks with arbitrary parameters and network connectivity that can be flexibly digitally programmed. We demonstrate successful stochastic simulation of a p53 cancer pathway and glycolytic oscillations that are consistent with results obtained from conventional digital computer simulations, which are based on experimental data.

We show that unlike conventional digital solutions, an increase in network scale or molecular population size does not compromise the simulation speed and accuracy of

our completely parallel cytomorphic system. Thus, commonly used circuit improvements to future chips in our digital-to-analog converters, noise generators, and biasing circuits can enable further orders of magnitude of speedup, estimated to be a million fold for large-scale networks.

Thesis Supervisor: Rahul Sarpeshkar
Title: Thomas E. Kurtz Professor, Dartmouth College, and Visiting Scientist, Research Laboratory of Electronics, MIT

Acknowledgments

First of all, I would like to express my sincere gratitude to my advisor, Professor Rahul Sarpeshkar. From the moment he first interviewed me during the MIT visit day, he has been nice, friendly, and caring. I could feel many times that he really cares about my well-being. Not only that, he is a man of extraordinary intelligence. It has been my pleasure to do research under his guidance, while having been exposed to and influenced by his unique insight, deep knowledge, and passion for pursuing real science and "high-hanging fruit". I can confidently say that he is the best advisor for me.

I would like to thank the members of my thesis committee: Professor Sanjoy Mitter, Professor Bruce Tidor, and Professor Jeremy England. They have given great advice and suggestions on my research and shown their willingness to help find ways to improve it. I feel privileged to have these prominent professors in my thesis committee.

I would also like to thank my academic advisor, Professor Jacob White. I have enjoyed visiting and talking with him about various subjects, from ordinary matters to scientific matters. He has always been willing to help anything he could, so that I can successfully complete the PhD program.

I am very grateful that I have been with such great colleagues in the Analog Circuits and Biological Systems Group: Current members including Areen Banerjee, Jaewook Kim, Ji Zeng, Jonathan Teo, and Susan Davco, and former members including Anne Ziegler, Isaac Weaver, Lorenzo Turicchia, Ramiz Daniel, and Soumyajit Mandal. Without them, I could simply not have gotten to this point. It has been intellectually and emotionally pleasing to interact with these electrical engineers and biologists on a daily basis. Special thanks to Jaewook, for his contribution to designing noise generators and ADCs in cytomorphic chips, fruitful discussions on diverse topics, and the refreshing times we had together at Fenway Park.

I would like to acknowledge the financial support from the Korea Foundation for Advanced Studies (KFAS), which made my research possible.

Next, I would like to thank D-Lab, MIT's program for international development, and all the people I have met in this field who do amazing work: Amy Smith, Libby Hsu, Amit Gandhi, Matt McCambridge, Elizabeth Moreno, Dan Sweeney, Jack Whipple, Nancy Adams, Debora Leal, Cheetiri Smith, and Hyung-joon Kim. They have given me huge motivation to hone my skills and be equipped as an engineer who can make positive changes in the world.

When I face hardships during the PhD program, several Korean friends at MIT have helped and encouraged me to persevere and not become exhausted. I am thankful to Young Gyu Yoon, Jin-hong Choi, Jiyoun Chang, Hyunryul Ryu, Dongsuk Jeon, Hyung-Min Lee, Hyun Ho Boo, Soohong Kim, Taehong Kwon, Seungbun Lim, Do Yeon Yoon, and Juho Kim.

Many thanks to Pastor Tae Whan Kim and my friends in First Korean Church in Cambridge, especially the members of CN1B and KOA Groups and the choir. Because of their support, I have been able to stand fast.

I am much obliged to Doctor Peter Maggs and the staff at Mount Auburn Hospital for taking good care of my health in 2015.

I am truly grateful for the unconditional support and countless hours of prayers of my family. I will never forget their sacrifice. My parents, Soon Koo Woo and Choon Hie Yu, and my sister, Youn Sik Woo, have allowed me to take this wonderful opportunity to study abroad. My father-in-law, Joo Min Lee, and mother-in-law, Kyung Hee Kim, have also shown their support, kindness, and love. I would also want to thank my relatives in Boston—my cousin brother, David Choi, and sister-in-law, Mimi Kim, and their cheerful daughters.

Ji Yung Lee, my lovely wife, has always been by my side, supporting, loving, caring, and understanding me in every way at every moment. I cannot be more grateful that I am with someone who shows me what true love, faith, and patience are. I love you.

Finally, this thesis is dedicated to my heavenly Father, who teaches me his ways and leads me in the right direction. I can do everything through him who gives me strength.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter presents the need for building a tool for high-speed simulations of biological networks and the summary of previous work to accomplish it. We describe how the deep similarities between chemistry and electronics enable us to devise efficient digitally programmable analog current-mode transistor circuits that quantitatively represent gene-protein networks in cells. We outline how our utilization of analog and digital computation combines analog efficiency with digital flexibility and robustness, thus addressing challenges faced by previous methods.

1.1 Motivation

Scientists have constantly yearned for unveiling the marvels and mechanisms of living organisms. Two emerging fields that take unique approaches towards an improved understanding of living systems are systems and synthetic biology: Systems biology is devoted to revealing complex interactions between the components of biological systems [13, 57, 71]. It uses a holistic view to explain causes and effects existing in those systems. On the other hand, synthetic biology is a field focusing on engineering biological circuits and systems [11, 49, 66, 110, 116, 131]. It encompasses various purposes, including manipulation of existing genetic networks to create useful functions, artificial construction of completely new biological systems, and design and analysis of small-scale synthetic systems to better understand the complex behaviors of larger

biological networks.

A powerful tool for modeling and simulation of biochemical reaction networks is highly important in both of the areas: It enables analysis and prediction of cellular functions of interest; by providing anticipated results and optimum design parameters, it may reduce the number of experiments that have to be conducted before attaining desired outcomes and thus save considerable resources; just like the creation of faithful models and simulators opened up a new era in semiconductor industry, it may facilitate the design of sophisticated synthetic circuits; and, by employing high-throughput searching and learning algorithms, it can contribute to filling the gap of unknowns in biology (effects of certain molecules, missing parameters, pathways, principles, etc.). Many pathways and parameters are still unknown even for relatively well-characterized reaction networks such as glycolysis. Several advantages of a circuits approach to systems and synthetic biology are discussed in [116] and in [111].

Several efforts have been made to create such simulators[1]. Software simulations have first become widely available, utilizing various math problem solvers, general-purpose programming languages, application-specific software for biological simulations, standardized representation formats, and biological databases. However, serial processing within the boundary of conventional von Neumann computing architecture poses formidable computational challenges. Simulation time grows prohibitively as the target networks contain multiscale, nonlinear, non-modular, stochastic, and feedback dynamical effects, which are common traits in gene-protein networks. Fortunately, it has been proven in many areas that utilizing special-purpose hardware can often be an effective solution to tackle such challenges. Yet, no promising solution has been developed so far in expediting the simulation of large-scale stochastic biochemical reaction networks.

It should be emphasized that both stochastic simulation and large-scale simulation are becoming increasingly important as the fields of systems and synthetic biology grow. For example, drug resistance in infectious diseases and cancer are often the results of stochastic effects, which cannot be emulated by deterministic simulations

[1]See Section 1.2 for details.

[69, 125]. Many other phenomena that are driven by random events, such as ageing, phenotypic heterogeneity, occurrence of certain diseases due to rare formation of malicious proteins, and cells' response to DNA damage, can only be studied with stochastic models as well [100, 101, 137]. On the other hand, large-scale simulations provide a holistic understanding of how organisms work. Just as how big data has turned out to be useful in various applications (e.g., optimizing delivery system or bus routes) shortly after becoming available, a wealth of knowledge extracted from large-scale biological systems will help us discover what has been unseen or elusive to date [60, 90].

Therefore, the goal of this thesis is to build a high-performance simulation, analysis, and design tool which can open up new opportunities for studying and constructing complex networks in systems and synthetic biology. This work can serve as a stepping-stone to the aspiration for simulating a whole human body someday in the future.

1.2 Prior Methods

This section outlines various approaches that have been taken so far by researchers for simulation of biological systems. Broadly speaking, they can be categorized into three groups: software implementations, digital implementations, and analog or hybrid analog-digital implementations.

When reviewing prior methods, we decided to include a few works related to simulation of neuronal networks. By modeling and simulating the functions of neurons, neuronal simulations aim at 1) learning the mechanisms of the brain and 2) creating a new brain-inspired computing paradigm called *cognitive computing* [90]. Neuronal networks and cellular networks are similar to each other in that they both compute with an enormous number of highly efficient units—cells and neurons—which construct massively parallel and complex structures. From a hybrid analog-digital computational point of view, they share 13 similarities, which have been outlined in [116]. Interestingly, the number of synapses in the human brain is estimated to be

on the same order (10^{14}) as the optimistic number of biochemical reactions per cell cycle in *E. coli* [28, 138]. Thus, it seems clear that both networks can take advantage of a high-performance simulation tool.

1.2.1 Software implementations

MATLAB provides a SimBiology toolbox[2] to aid in the modeling and simulation of biological systems. Graphical programming tools including Simulink[3] or LabVIEW[4] are useful for intuitive modeling and visualization. General-purpose programming languages like C/C++ and Python enable more efficient simulation than MATLAB at the expense of longer development time. For example, when it comes to executing loop statements or large-scale programs, C/C++ tends to be superior to MATLAB in terms of simulation speed. It is thus common to start a project with MATLAB to benefit from its ease of use and translate the code to C/C++ as it goes beyond the prototype stage, if performance matters.

Computational biology researchers often use software that is specifically developed to simulate biochemical reaction networks. COPASI[5] [54], SynBioSS[6] [52], and CellDesigner[7] [32] are examples of such software. Most of them are capable of running stochastic simulations as well. They are useful not only because they are faster and more efficient than typical math solvers, but also because they offer convenient graphical user interfaces, which alleviates the need for programming skills.

Compared to hardware realizations, software realizations enable shorter development time, faster modification, and easier scale-up. Scalability can be enhanced by modular design through object-oriented programming. Codes can mostly remain unchanged regardless of the type and the specifications of the operating system and computer hardware. Higher performance can easily be achieved by simply upgrading the hardware of the computer. On the contrary, custom hardware designs generally

[2] `http://www.mathworks.com/products/simbiology/`
[3] `http://www.mathworks.com/products/simulink/`
[4] `http://www.ni.com/labview/`
[5] `http://www.copasi.org/`
[6] `http://synbioss.sourceforge.net/`
[7] `http://www.celldesigner.org/`

have to undergo an extensive revision if their fabrication process, supply voltage, or clock speed changes.

Furthermore, more and more online databases such as BioNumbers[8] [88], RegulonDB[9] [34], BioModels[10] [59], and ProNIT[11] [65] are becoming available, from which users can download a large amount of data and manipulate and load them to a software application of their choice. An XML-based format called Systems Biology Markup Language (SBML[12]) [56] was also invented to represent biochemical network models in a unified way. As a result, models can be shared among researchers using different software tools, as long as the tools support SBML. This saves a tremendous amount of energy needed to rewrite models according to the file format of each software. Models represented in SBML and stored in databases have more chances to be influential and long-lasting. In fact, many investigators, when publishing a journal paper, publicize their models, databases, custom-developed softwares, and their manuals.

However, with a software-dependent solution, there is a prohibitive increase in simulation time when the complexity and the scale of biological models increase. Here we provide some examples: $\sim$200 hours of a 6-reaction system can easily take 20 hours to simulate with the exact stochastic simulation algorithm (SSA) on 6 CPU cores [74]; with faster stochastic methods, a day is required to simulate 100 minutes of a 100-reaction system on an 800 MHz Pentium III computer [28]; [118] and [119] estimated that, using a Java simulator engine on a 500 MHz Pentium II computer, and an efficient stochastic algorithm on highly simplified networks, stochastic simulation of the whole 10^{14} biochemical reactions during a cell cycle of *E. coli* would take about 12 years; in a recent prominent work, a whole-cell computational model of the bacterium *Mycoplasma genitalium* was produced [60]. It was shown that the model has an ability to generate results that both match existing experimental knowledge

[8] `http://bionumbers.hms.harvard.edu/`
[9] `http://regulondb.ccg.unam.mx/`
[10] `https://www.ebi.ac.uk/biomodels-main/`
[11] `http://www.abren.net/pronit/`
[12] `http://sbml.org/`

and possess predicting power which can bring about meaningful biological discovery. However, although *M. genitalium* has the smallest genome of any free-living organism (∼525 genes), the software implemented in MATLAB runs 10 hours on a 128-node Linux cluster running in parallel to simulate a single division of a cell, even with highly oversimplified primarily deterministic methods and no Poisson stochastic modeling. To fulfill their subsequent goal of extending their work to a bigger organism, such as *E. coli* (∼4400 genes) or a human cell (∼25,000 genes), substantially more computational power is needed. Software simulation is tractable for deterministic simulations of modest-size biological networks but infeasible for larger-scale networks or even for modest-size stochastic networks. In general, the cost of simulating a highly parallel, stochastic, stiff, collective analog bio-molecular network via a traditional serial digital von Neumann approach is not optimal.

Stochastic simulation focuses on randomness in chemical reactions—randomly moving molecules bumping into each other to create a chemical reaction event. It is unpredictable and probabilistic; thus, when the number of molecules is low, random fluctuation in molecular concentration manifests itself more evidently. To simulate such behavior, the Gillespie Stochastic Simulation Algorithm (SSA) [36, 40, 41] randomly selects a reaction to occur and the time it will occur in each iteration, based on the reaction rates of all reactions. Then, it updates the time and the molecule count of each species and moves on to the next iteration. This method is considered "exact" in the sense that it produces time course trajectories that are in exact accordance with the underlying chemical master equation. However, it is slow because it simulates only one reaction event at a time, requiring lots of iterations, and the tasks to be performed in each iteration are computationally expensive (especially random number generation). In addition, an increase in network scale, molecular population size, or stiffness inevitably increases the number of reaction events to be simulated and thus increases simulation time. Finally, several runs are needed to obtain statistical variances, which further increases computational cost. Therefore, for many practical applications, the cost simply becomes too high to be tractable. Although several algorithms have been developed to enhance simulation speed, it comes at the

expense of a loss of accuracy. See Section 4.4.1 for further discussions.

Biological systems exhibit a number of feedback, nonlinear, and non-modular dynamical behaviors. Due to these complexities, their simulations become mathematically hard problems, mostly having no analytic (or closed-form) solution. Consequently, numerical methods such as the Runge-Kutta method are essential. In addition, biochemical networks are often numerically stiff, owing to the coexistence of fast timescales (e.g., binding between inducers and transcription factors) and slow timescales (e.g., transcription, translation, and dilution). To avoid numerical stability, maintain Nyquist sampling rates, and ensure accuracy in such stiff systems, a small time step is essential, which drastically increases the time required for simulation. Solvers specialized in stiff differential equations exist (e.g., ode15s of MATLAB), but they sacrifice accuracy and are still expensive.

One can attain better performance by using state-of-the-art CPUs and memories, multi-core or multi-threading techniques, distributed computing with a cluster of computers, optimized algorithms, a compiled language instead of an interpreted language, or codes rewritten in lower level languages (e.g., from MATLAB to C, C to Assembly). In fact, these methods can certainly increase simulation speed, but their limitations are still apparent: For instance, a group of researchers of IBM demonstrated the phenomenal cat-scale cortical simulation—1.6 billion neurons and 8.87 trillion synapses—using the Dawn Blue Gene/P supercomputer with 147,456 CPUs and 144 TB of memory and the power consumption of 1.13 MW [4]. It took 643 seconds for the supercomputer to simulate one second of brain activity per Hz of mean neuronal firing rate. However, using a ton of cores is not a practical nor easily accessible solution. Essentially fast-and-serial modern digital CPUs with von Neumann bottlenecks and logic basis functions are not optimized for slow-and-massively-parallel noisy analog biological networks. The human cortex can be considered as a system with $\sim$22 billion locally interacting computing cores (neurons) that as a whole consumes only $\sim$14.6 W; the human body is a system with $\sim$100 trillion locally interacting computing cores (cells) consuming only $\sim$80 W [116]. This architectural

discrepancy implies that brute-force approaches may not be the best idea to solve the problem.

IBM researchers' endeavor to overcome this fundamental limit has been reflected in their recent project to develop a neuromorphic chip that contains one million neurons [87], which will be introduced in more detail in the next section. Another effort has been made by the researchers at the University of Illinois at Urbana-Champaign to achieve speedup by leveraging graphics processing units (GPUs) to run their original stochastic simulation software [47, 106]. The architectural benefits of GPUs have enabled parallel, faster execution of computationally intensive algorithms such as random number generation, offering up to two orders of magnitude speedup. As a result, their computing power has expanded from single-cell simulation to simultaneous simulation of multiple cells (colonies). Both cases described in this paragraph advocate the paradigm shift from software-only to hardware assisted methods. This thesis aims to extend this paradigm shift even further by creating digitally programmable analog chips with highly efficient hardware, thus enabling the best of the analog and digital worlds to be integrated in a scalable fashion.

1.2.2 Digital Hardware Implementations

In hardware realizations, hardware (analog or digital electronic circuits) undertakes the core computation; meanwhile, software may be involved in creating a user interface (UI), programming hardware, visualization of results, or less intensive calculations. A video card is a great example. Although CPUs can carry out any digital functions, such versatility is obtained by sacrificing performance. That is, CPUs are not optimized for manipulating computer graphics which requires high-throughput, highly parallel data processing. On the other hand, video cards are dedicated hardware designed specifically to accelerate such tasks.

When utilizing digital hardware, it is important to choose between general-purpose hardware such as Field Programmable Gate Arrays (FPGAs) or Complex Programmable Logic Devices (CPLDs) and special-purpose hardware, so called Application Specific Integrated Circuits (ASICs). Modern FPGAs are armed with remarkable capacity

and performance (20 nm process, >4.4 millions of logic cells, >130 Mb of RAM, and >5.8 Tb/s of total transceiver bandwidth for Virtex® UltraScale™). They serve as a great platform where a user can rapidly develop high-speed massively parallel digital functions, a capability that we utilize in our digitally programmable analog systems as well. The usability of FPGAs is thus expanding beyond the prototyping stage. Some prior work in [108] and [22] has not surprisingly, taken a purely-FPGA-based approach to accelerate the simulation of biological processes and has achieved modest speedups, but not as great as we have been able to achieve or is possible through our hybrid analog-digital approach.

Specialized hardware, through leveraging full customization, is used to offer maximum performance and efficiency for specific applications. By sacrificing flexibility, it gives a potential to exceed the speed and scale that general-purpose hardware can achieve. Anton by D. E. Shaw Research, for example, is a special-purpose supercomputing chip to accelerate simulation of molecular dynamics. It enabled millisecond-scale atomic-level simulations of molecules for the first time, for proteins with a relatively small number of amino acids, such as 100, and for a short protein folding time[13] [123]. TrueNorth created by IBM's scientists in 2013 is another exceptional integrated circuit which generates supercomputing power [87]. Having an architecture inspired by the brain, it has an ability to run large-scale neural applications in real-time, with one million neurons and 256 million synapses per chip. Compared to the Dawn Blue Gene/P supercomputer mentioned in Section 1.2.1, TrueNorth is demonstrated to be at least 100 times faster and 130,000 times more energy efficient. H. Park et al. built a proof-of-concept compiler to simulate how a network of dedicated stochastic processors could achieve speedup over traditional methods [97]. They showed that strongly coupled networks could achieve orders-of-magnitude speedup due to resource sharing but that this benefit did not port to loosely coupled networks as in many biological systems; nevertheless even these networks, achieved an order of magnitude improvement.

[13]Surprisingly, a 512-node Anton machine simulates a benchmark system at a rate of 16.4 us/day. It takes two months to reach a millisecond. Note this intensity of just a molecular simulation!

1.2.3 Analog Hardware Implementations

Although the benefits of an analog approach have been argued for years and proven in many high-performance systems, some even in patients [15, 86, 109, 115, 116], it is the imminent and widely-accepted end of Moore's law that has led to an increasing and new appreciation for analog hardware. For example, the U.S. Defense has recently articulated the possibility that analog probabilistic computing can offer orders of magnitude improvements in efficiency as well [84]. Theoretical proof that analog computation is more efficient than digital computation at low or moderate precision is provided in neural and electronic systems [109, 116] and in actual living cells [110]. The absolute necessity for computations in severely energy-limited cells to be analog and probabilistic has also recently been shown [110]. Furthermore, for multiscale biological models, it is difficult to achieve fast simulation using purely digital approaches, even with parallel processing techniques (see Section 4.4.1). In essence, the hybrid analog-digital nature of many biological systems implies that incorporating analog computation may provide great advantages in capturing certain behaviors in them.

Emulating biochemical reaction networks in cells using analog electronic circuits has been a topic of steady interest since 1950s [16, 50, 51, 55, 75, 78, 79, 83, 96, 104, 129]. Researchers have discovered in common that analog circuits are effective at quantitatively reproducing biological behaviors. However, they had applications limited to specific behaviors (usually mapping specific equations onto circuits) [50, 51, 75, 96], were power- and area-inefficient due to the use of power-hungry operational amplifiers or discrete components [50, 51, 75], presented less realistic circuit models and no physical hardware [16, 83, 96, 104, 129], adopted too abstracted models [55, 78, 129], or suffered from mismatch effects (no steady-state solutions) [75, 79]. None of them achieved the scale of more than 20 unit blocks (e.g., 6 in [78], 12 in [55], and 20 in [129]). It thus appears that no hardware simulation of gene-protein networks in a large scale has been demonstrated so far. Besides, none of the prior approaches seem to have exploited the fact that analog chemical circuits and analog electronic circuits are deeply similar with respect to stochastics, flows, and implementation due

to common Boltzmann laws [79, 116]. Thus, they have not built efficient transistor-based circuits as this thesis proposes to do. This may partly be because of the lack of unveiled biological networks in the past, resulting in less demand for simulations. Fortunately, although many effects in biology are still unknown, a wealth of models and parameters of large-scale biological networks are being unearthed every day.

Substantial efforts have also been made with respect to emulating neural networks with analog circuits in silicon, from relatively small scales [21, 23, 68, 77] to large scales [8, 117]. The pros and cons of analog vs. digital approaches are discussed in a rigorous fashion in [116]. As these attempts have produced meaningful results in neuroscience, so studies on modeling and large-scale integration of gene-protein networks on silicon wafers can have a major impact in systems and synthetic biology.

1.3 Our Approach

In a nutshell, we envision a proof-of-concept high-speed chipset to simulate large-scale gene-protein networks. Five lessons were learned from reviewing previous approaches:

1. Simulations of large-scale gene-protein networks have been performed with software but not with hardware. However, parallel computation using hardware is critical to boosting speed, and an efficient mapping between electronics and chemistry allows to achieve parallelism efficiently.

2. Large-scale hardware simulations have been performed for neural networks but not for gene-protein networks.

3. When simulating a network which has low-precision and multiscale characteristics (as it is in many biological systems), analog hardware may be more efficient than digital hardware.

4. To build a flexible system amenable to the simulation of a wide range of biochemical networks, elementary blocks should be made sufficiently general and programmable.

5. To build a physical chip that integrates many genes and proteins, it is required to thoroughly address various design requirements (e.g., area, power, number of pins, on-chip connectivity, etc.) and non-idealities (e.g., mismatch, noise, crosstalk, process variation, parasitic capacitance, leakage, etc.), from component level to system level. Practical and realistic design choices must be made.

Hence, the very first design choice we make is that we use analog hardware as a primary method for computation. As opposed to digital computation that discards all information except "0" and "1", analog computation keeps and uses all intermediate values. Thus analog computation can extract more information from a transistor, becoming inherently more efficient. Since nature is full of analog functions and exhibits incredible efficiency, our approach resembles that of nature. In fact, mappings between molecular flux and electron flow, molecular noise and transistor noise, and the conservation law for molecules (flux balance analysis) and electrons (Kirchhoff's current law) are transparent and intuitive with analog circuits but not with digital circuits. Power and time for computation may be saved by the clever use of such natural mappings based on physical basis functions existing in analog circuits [79, 116]. In this sense, the term "cytomorphic" that compose the title of this thesis reflects the fact that our chip not only emulates the operations of cells but also is inspired by the underlying computation principle of cells. The principle is, in a phrase, *collective analog computation* [109, 116], where many imprecise analog units collectively compute to perform complex or precise functions [140]. For example, recent work has shown that analog computation can also be made arbitrarily precise like digital computation: Four 4-bit-precise spiking-neuron-like analog adders adding via Kirchhoff's current law and interacting via a pulsatile carry implemented 16-bit-precise analog addition, which could be made scalably and arbitrarily precise.

Digital circuits can offer robustness, scalability, and programmability to our system. Although analog circuits are efficient in nonlinear, feedback, and stochastic computations, they are susceptible to corruption due to noise. Since the noise accumulates over cascaded stages and over time, at some point, analog signals should

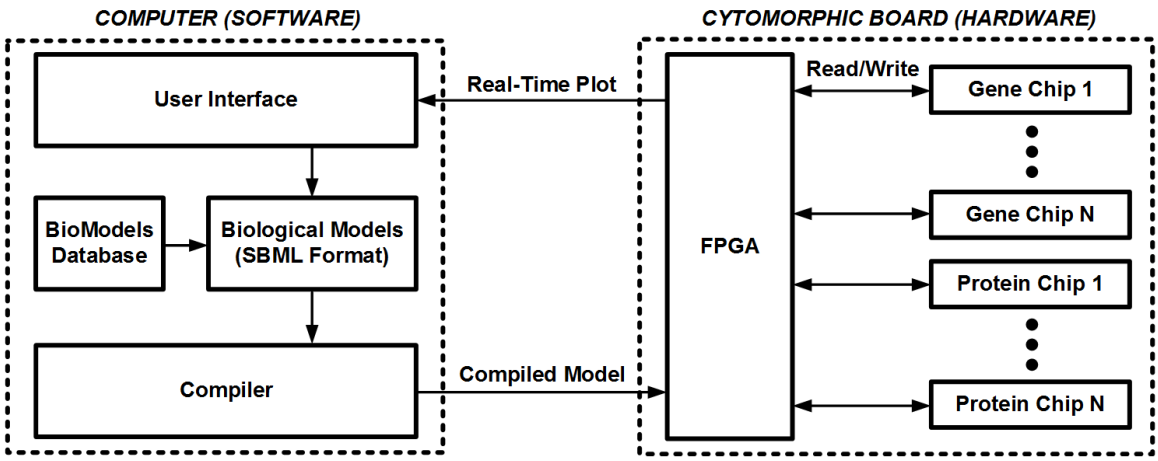


Figure 1-1: High-level block diagram of the cytomorphic system.

be converted to a digital format to maintain the information, as rigorously analyzed in [116]. Similarly, as for long-term storage of data, digital memory (e.g., SRAM) is more appropriate than analog memory (e.g., capacitors). It is thus not surprising that biology also employs digital-like schemes, e.g., all-or-none spikes of neurons (communication), on/off states of genes (signal restoration), or lysis-lysogeny decision (decision making) [110].

Our system is built upon this hybrid analog-digital design methodology. To maximize efficiency, tasks are optimally assigned to analog and digital domains: Analog circuits constitute efficient parallel processing cores; digital circuits implement reconfigurability of parameters and connections, memory, etc.; FPGAs digitally control and program cytomorphic chips, establish data communication throughout the system, and implement error correction and other digital signal processing; and software offers a convenient user interface, graphically presents chip data, compiles a model, and analyzes data. Analog-to-digital converters (ADCs) and digital-to-analog converters (DACs) incorporated in cytomorphic chips allow to cross between analog and digital domains.

Figures 1-1 and 1-2 are pictorial representations of our overall methodology (see Section 4.1 to find detailed descriptions). To realize our goal, we first conceive and implement seven building-block circuits to quantitatively model fundamental biomolecular circuits in cells. These blocks include 1) the mass-action fundamental

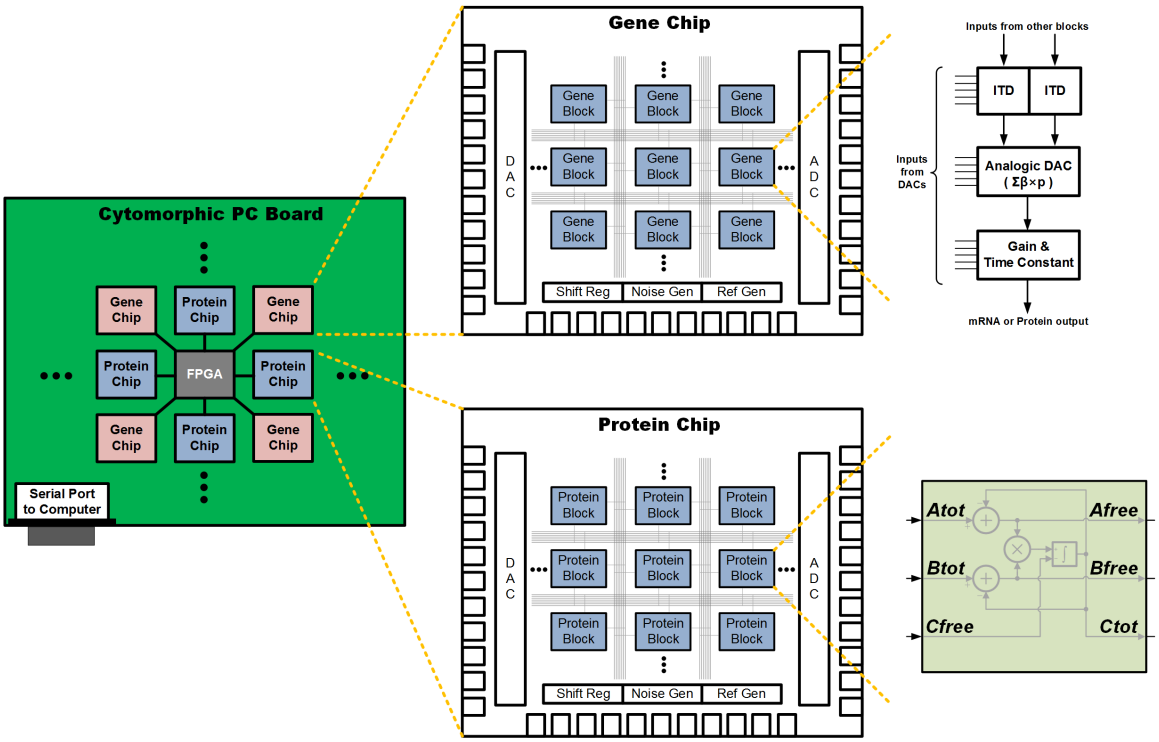


Figure 1-2: Overall architecture of the cytomorphic PC board and the cytomorphic chips.

reaction block, 2) the Hill block, 3) the Inducer-Transcription-factor-DNA (ITD) binding block, 4) the analogic DAC, 5) the gain and time constant block, 6) the noise generator, and 7) the ADC and the DAC (see Sections 2.1 and 2.2). These building-block circuits are then composed and scaled to construct the "gene chip" (Chapter 2) and the "protein chip" (Chapter 3), specialized in modeling and simulating genetic networks and protein networks in cells, respectively.

Finally, for fast and parallel simulation of large-scale networks, an array of these chips are laid out on a printed circuit board (PCB) with an FPGA-like architecture, i.e., an array of gene and protein blocks interacting via routing channels, within and amongst the chips (Chapter 4). This work lays a foundation for massively parallel simulations of biological systems ranging from relatively small synthetic genetic networks such as repressilator [26] and genetic toggle switch [35] to metabolic pathways such as glycolysis and Krebs cycle, to whole-cell scale networks such as 525 genes in *M. genitalium*, and to multi-cell scale networks such as a colony, a tissue, an organ,

and ultimately a whole body. High-throughput simulations will eventually lead to the discovery of unknown parameters and pathways via machine learning algorithms, which may highly influence the field of medicine.

1.4 Mapping Cells to Electronics

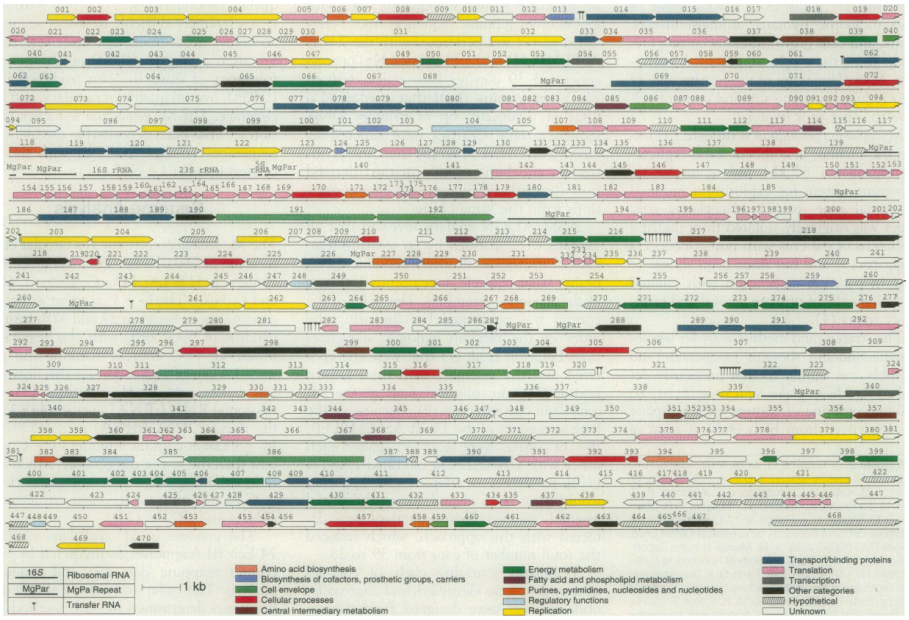
This section aims at bridging the gap between cells and electronics. Our unique approach, originated from the deep similarities between chemistry and electronics [79, 116], leads us to leverage analog translinear circuits to build transistor models of gene-protein networks. We show how these models quantitatively represent the steady-state behaviors of genetic activator and repressor circuits in *E. coli*.

1.4.1 Gene-Protein Networks in Cells

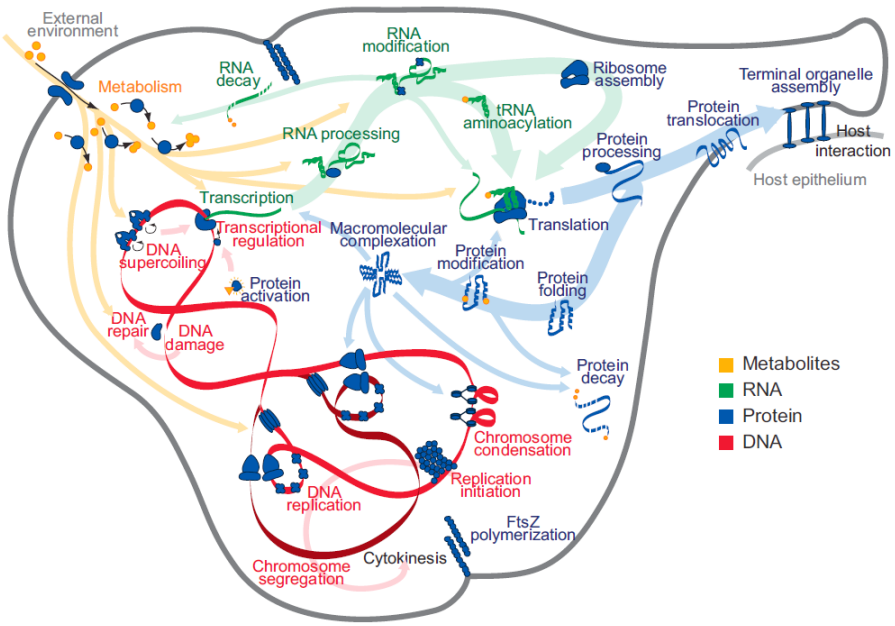
Cells are indeed miraculous devices, of which all creatures are composed. Despite their tiny size (see Table 1.1), they contain diverse sophisticated biological networks such as metabolic, protein-protein, and gene regulatory networks, which process an astounding amount of information. Nonetheless, each cell only consumes less than 1 pW of power [116]. Figure 1-3 shows the entire genome and various cellular processes of *M. genitalium*, the smallest known free-living organism [30, 60]. As the figure illustrates, many genes, proteins, metabolites, and other small molecules collectively operate to maintain the life and functionality of organisms. Our goal is to study such mechanisms in a holistic approach, through enabling large-scale simulations.

Table 1.1 shows typical values of parameters in cellular networks that help us set the design requirements of our system [3, 27, 92]. For example, to simulate a bacterial genetic network, programmable timescales of 70 dB dynamic range, at least 40 dB dynamic range of protein concentration, and 20 dB of signal-to-noise ratio are necessary. When studying the dynamics of transcriptional networks, the dynamics of inducer-transcription factor binding can be well approximated to be at steady state [3].

Figure 1-4 shows a simplified overview of gene-protein interactions in cells, where



(a)



(b)

Figure 1-3: (a) The whole genome map [30] and (b) various cellular processes [60] of *Mycoplasma genitalium*. (a) From [Claire M. Fraser et al. The Minimal Gene Complement of *Mycoplasma genitalium*. *Science*, 270(5235):397–404, 1995.]. Reprinted with permission from AAAS. (b) Reprinted from [Jonathan R. Karr et al. A whole-cell computational model predicts phenotype from genotype. *Cell*, 150(2):389–401, July 2012.], Copyright 2012, with permission from Elsevier.

Table 1.1: Typical Parameter Values in Cellular Networks

Parameter	**Bacteria** (***E. coli***)	**Yeast** (***S. cerevisiae***)	**Mammalian cell** (***Human Fibroblast***)
Cell volume	0.5–5 μm^3	20–160 μm^3	100–10000 μm^3
Concentration of one protein/cell	∼1 nM	∼1 pM	∼0.1 pM
Cell cycle time	20–40 min	70–140 min	15–30 hr
Transcription time	∼1 min	∼1 min	∼30 min
Translation time	∼2 min	∼2 min	∼30 min
mRNA lifetime	2–5 min	10 min–1 hr	10 min–10 hr
Inducer-transcription factor binding time	∼1 ms	∼1 s	∼1 s
Transcription factor-DNA binding time	∼1 s		
Concentration for a signaling protein	10 nM–1 μM	10 nM–1 μM	10 nM–1 μM
Signal-to-noise ratio	0–20 dB	0–20 dB	0–20 dB

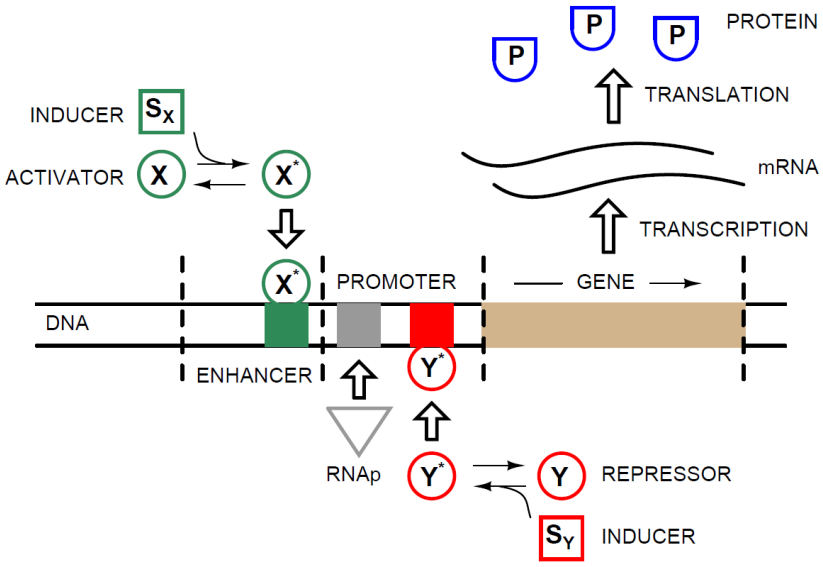


Figure 1-4: A simplified overview of gene-protein interactions in cells [78]. Copyright 2009 IEEE.

the production of a protein molecule is controlled by the reactions among inducers, transcription factors, and DNA. S_X is a kind of small inducer molecules which comes into the cell and binds to activator molecules (X) to change their state from X to X^*. This binding reaction can be written as $S_X + X \underset{k_r}{\overset{k_f}{\rightleftharpoons}} X^*$, where k_f and k_r are the forward and reverse reaction rate constants, respectively. The mathematical representation of this reaction is given by

$$\frac{d[X^*]}{dt} = k_f[S_X][X] - k_r[X^*] \tag{1.1}$$

$$[X_{tot}] = [X] + [X^*] \tag{1.2}$$

where $[X_{tot}]$ refers to the total concentration of the activator molecules. From the above equations, the concentration of X^* at steady state can be derived as

$$0 = k_f[S_X]\left([X_{tot}] - [X^*]\right) - k_r[X^*] \tag{1.3}$$

$$[X^*] = [X_{tot}]\left(\frac{[S_X]/K_D}{[S_X]/K_D + 1}\right) \tag{1.4}$$

where $K_D = k_r/k_f$ is the dissociation constant of the reaction.

When the activated transcription factor X^* is formed, it binds to the enhancer site of DNA with a high binding affinity. Then, it recruits the enzyme RNA polymerase (RNAp in Figure 1-4), which binds to the promoter region of DNA and initiates *transcription*—a process where a particular gene in the DNA is copied into a corresponding messenger RNA (mRNA) transcript. This mRNA is in turn used as a templete to produce a protein through the *translation* process.

On the other hand, S_Y is a kind of inducer molecules which binds to repressor molecules (Y) to change their state from Y to Y^*. These Y^* molecules then bind to a particular site of DNA to prevent the RNA polymerase from binding to DNA. Thus, as opposed to the activator molecules, the binding of the repressor molecules decreases the rate of transcription. This mechanism of regulating the expression levels of mRNA or protein by using transcription factors and other proteins is called *transcriptional regulation*. It is a vital and clever strategy of a cell to sense and

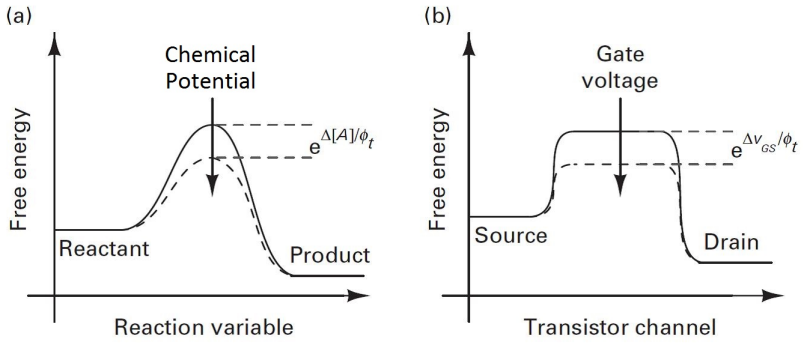


Figure 1-5: Analogies between (a) molecular flux in chemical reactions and (b) electronic current flow in subthreshold transistors [79, 116]. Copyright 2009 IEEE.

respond to environmental signals and conduct various housekeeping activities.

1.4.2 Similarities Between Chemical Reactions and Transistor Operations

This section describes the "cytomorphic mapping" which serves as the fundamental motivation and idea to port biochemical reaction networks to electronic circuits. Figure 1-5 shows the deep analogies between molecular flux in chemical reactions and electronic current flow in the transistors operating in the subthreshold regime [116]: Reactant and product concentrations in chemical reactions are analogous to electron concentrations at the source and the drain in transistors, respectively; forward and reverse reaction rates are analogous to forward and reverse electronic current flows, respectively; as the presense of enzymes exponentially changes reaction rates by lowering the activation energy, so the gate voltage exponentially changes current flow rates; and, interestingly, the stochastics of molecular shot noise is mapped into the stochastics of Poisson shot noise in transistors. Several constraints present in chemistry such as flux balance analysis and thermodynamic energy balance are also present in electronics, in the form of Kirchhoff's current law and Kirchhoff's voltage law, respectively.

Another important similarity is that the electrochemical potential exists in the form of "log(molecular concentration)+energy" in chemistry and "log(electronic cur-

rent)+voltage" in transistors [110]. This arises from the logarithmic and exponential basis functions that analog transistors naturally have. As such, it is easy to use log-domain analog circuits to create any dynamical systems of the form

$$\frac{d\mathbf{x}}{dt} = \mathbf{C} + \mathbf{D}\mathbf{x} + \mathbf{E}(\mathbf{x} \otimes \mathbf{x}) + \mathbf{F}\mathbf{u} + \mathbf{G}(\mathbf{x} \otimes \mathbf{u}) \tag{1.5}$$

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{K}\mathbf{u} \tag{1.6}$$

where $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{u}$ denote the vectors for reactant concentrations, output concentrations (linear combinations of all species), and input variables, respectively [79]. The matrix coefficients represent chemical-reaction kinetic parameters and $\otimes$ indicates the outer product. The above equation is general enough to create any combination of zeroth-, first-, and second-order chemical reactions.

Therefore, log-domain cytomorphic circuits built on top of this principle can be very effective in making a quantitative connection between chemistry and electronics [16, 78, 79]. They will serve as the fundamental building blocks to construct a rapid and efficient tool to run highly parallel stochastic simulations, as will be shown in the following chapters. Furthermore, the analogy in Figure 1-5 reveals that analog circuit design principles accumulated over decades can be utilized to create highly efficient log-domain synthetic circuits in cells [17] or to symbolically represent molecular circuits as a unifying language to help analyze and design complex biological circuits [131].

The next two sections describe the basic ideas to build log-domain current-mode circuits and how they can be used to create highly compact 8-transistor analog circuits to model transcriptional regulation depicted in Section 1.4.1.

1.4.3 Current-Mode Circuits

Contrary to voltage-mode circuits which represent information by node voltages, current-mode circuits represent information by current flowing through the branches of the circuit. Since current levels represent the dynamic range of signals, current-mode circuits are suitable for low-power, low-voltage, and wide-dynamic-range ap-

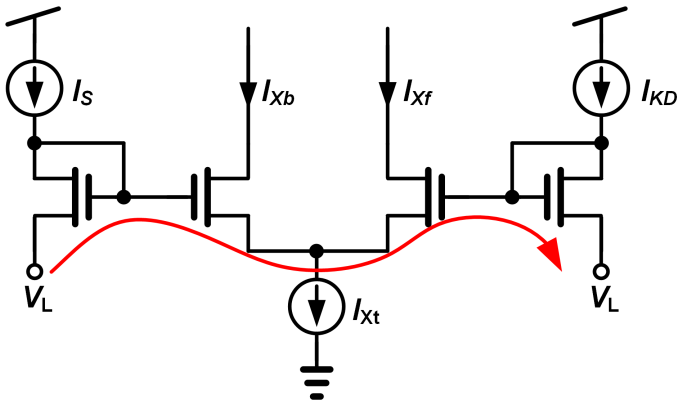


Figure 1-6: Schematic of a typical current-mode circuit used throughout our system. It is assumed that all transistors are equally sized and the well is tied to the source for all transistors.

plications. Some functions such as multiplication, integration, filtering, and other nonlinear operations can often be easily implemented in the current-mode domain. Several current-mode circuits logarithmically compress an input current to a voltage, process it in the log domain to yield desired behaviors, and exponentially expand it to create an output current. Thus, such current-mode signal processing is often referred to as log-domain signal processing.

In 1975, Barrie Gilbert invented a useful subclass of current-mode circuits called translinear circuits [37]. They leverage the exponential current-voltage (I-V) characteristic of bipolar junction transistors (BJTs) or MOS transistors in weak inversion. The I-V characteristic is given by

$$I = I_S e^{V_{BE}/\phi_t} \text{ (BJT)} \tag{1.7}$$

$$I = I_{0S} e^{\kappa_S V_{GS}/\phi_t} \text{ (MOS transistor in weak inversion)} \tag{1.8}$$

where I_S is the reverse saturation current of a BJT, ϕ_t is the thermal voltage, I_{0S} is the pre-exponential current of an MOS transistor in weak inversion, and κ_S is the constant representing the channel divider. For subthreshold MOS transistors, the well should be tied to the source to remove the undesirable body effect.

Figure 1-6 shows a typical current-mode circuit which consists of 4 transistors. This simple block and the underlying principle are widely utilized throughout our

system to construct various types of current-mode circuits. If we apply Kirchhoff's voltage law around the closed loop indicated by the red line, it simply results in

$$\sum_{n \in CW} V_n = \sum_{n \in CCW} V_n. \tag{1.9}$$

This equation tells that the sum of clockwise voltages (i.e., voltage rise) is equal to the sum of counter-clockwise voltages (i.e., voltage drop). If the 4 transistors have the same dimensions, it is straighforward to show that because of the exponential I-V relationship in equation (1.8), the above equation becomes

$$\prod_{n \in CW} I_n = \prod_{n \in CCW} I_n. \tag{1.10}$$

That is, the product of clockwise currents equals to the product of counter-clockwise currents in the loop. This is called the *translinear principle* [37]. Hence, the two equations that describe the behavior of the circuit in Figure 1-6 are given by

$$I_S I_{Xf} = I_{Xb} I_{KD} \tag{1.11}$$

$$I_{Xt} = I_{Xb} + I_{Xf}. \tag{1.12}$$

Solving these two equations for I_{Xb} and I_{Xf} yields

$$I_S \left(I_{Xt} - I_{Xb}\right) = I_{Xb} I_{KD} \tag{1.13}$$

$$I_{Xb} \left(I_S + I_{KD}\right) = I_S I_{Xt} \tag{1.14}$$

$$I_{Xb} = I_{Xt} \left(\frac{I_S/I_{KD}}{I_S/I_{KD} + 1}\right) \tag{1.15}$$

$$I_{Xf} = I_{Xt} \left(\frac{1}{I_S/I_{KD} + 1}\right). \tag{1.16}$$

Note that equation (1.15) is equivalent to equation (1.4), where $[X_{tot}]$, $[X^*]$, $[S_X]$, and K_D in equation (1.4) correspond to I_{Xt}, I_{Xb}, I_S, and I_{KD} in equation (1.15), respectively. Thus, it can be seen that the 4-transistor circuit in Figure 1-6 efficiently represents the steady-state behavior of the binding reaction, where molecular

concentrations are mapped into electric current levels.

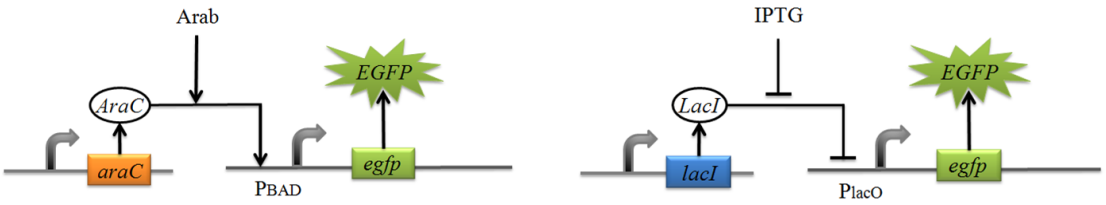
1.4.4 Transistor Models of Activator / Repressor Circuits

This section shows that current-mode circuits similar to the one in Figure 1-6 can be repeatedly used to create compact analog transistor models of synthetic activactor and repressor circuits in *E. coli*. The models are capable of producing input-output characteristics that quantitatively represent the mathematical models and experimental biological data for those bacterial genetic circuits. Such quantitative agreement gave us motivation to proceed to the rest of the work in the thesis, with current-mode circuits as the primary computing devices.

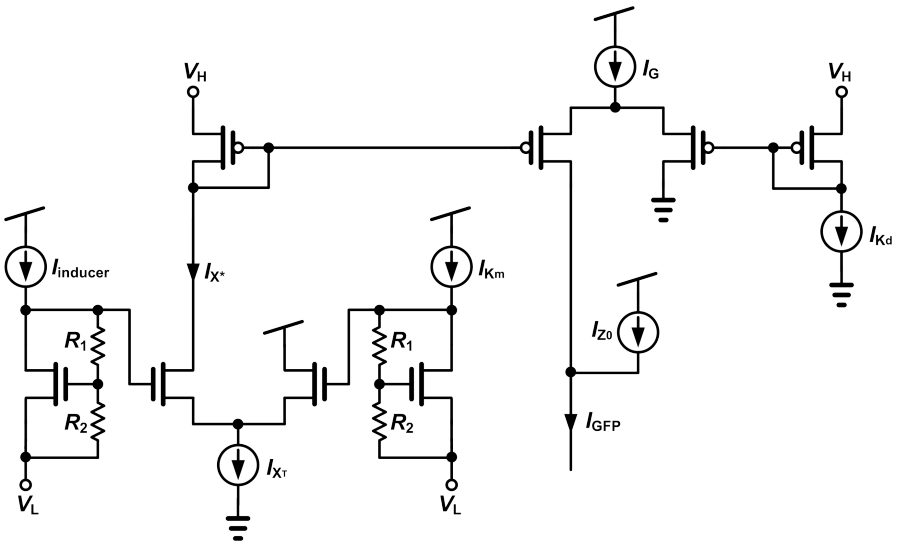
This section contains the work that was previously published as a paper in the *Proceedings of the 2011 IEEE Biological Circuits and Systems Conference* [16], Copyright 2011 IEEE. Reprinted with permission from the publisher. All biological experiments are conducted by Ramiz Daniel.

First, Figure 1-7(a) is a representation of a simple synthetic activator circuit. AraC is a transcriptional activator that is constitutively produced. When it binds to an Arabinoise (Arab) inducer, it changes to an activated form and binds to the DNA. Then, it enhances the rate of transcription via the PBAD promoter. EGFP is a reporter protein which allows us to monitor the change in the expression level from the PBAD promoter, by using a standard fluorescence technique.

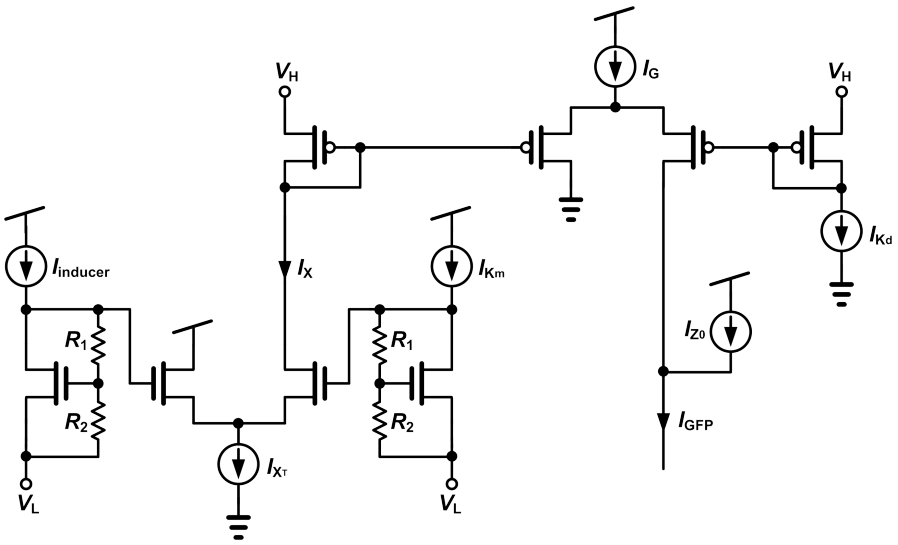
Figure 1-7(c) shows a subthreshold transistor circuit model of the activator circuit in Figure 1-7(a). It is basically a cascade of the two current-mode circuits presented in Figure 1-6, one modeling the inducer-activator binding and the other modeling the activator-DNA binding. The resistive divider composed of R_1 and R_2 models the Hill coefficient ($m = (R_1 + R_2)/R_2$) of the inducer-activator binding. $I_{inducer}$, I_{K_m}, I_{X_T}, and I_{X^*} represent the inducer concentration, the dissociation constant for the inducer-activator binding, the total activator concentration, and the inducer-activator complex concentration, respectively. I_{K_d}, I_G, I_{Z_0}, and I_{GFP} correspond to the dissociation constant for the activator-DNA binding, a combined rate of transcription-factor-dependent protein production based on RNA polymerase and ribosomal activities, the



(a) (b)



(c)



(d)

Figure 1-7: Schematic representation of (a) the activator and (b) the repressor circuit in *E. coli*. The corresponding 8-transistor circuits to model (c) the activator and (d) the repressor circuit [16]. Copyright 2011 IEEE.

basal level of fluorescence, and the level of (green) fluorescence which is proportional to the level of EGFP.

Similar to the derivation process of equation (1.15), the equation for each of the two binding reactions can be derived:

$$I_{X^*} = I_{X_T}\left(\frac{(I_{inducer}/I_{K_m})^m}{(I_{inducer}/I_{K_m})^m + 1}\right) \tag{1.17}$$

$$I_{GFP} = I_G\left(\frac{I_{X^*}/I_{K_d}}{I_{X^*}/I_{K_d} + 1}\right) + I_{Z_0}. \tag{1.18}$$

If we substitute equation (1.17) into equation (1.18), we obtain

$$I_{GFP} = I_G\left(\frac{I_{X_T}\left(\frac{(I_{inducer}/I_{K_m})^m}{(I_{inducer}/I_{K_m})^m+1}\right)/I_{K_d}}{I_{X_T}\left(\frac{(I_{inducer}/I_{K_m})^m}{(I_{inducer}/I_{K_m})^m+1}\right)/I_{K_d} + 1}\right) + I_{Z_0} \tag{1.19}$$

$$= \frac{I_G}{1 + \frac{I_{K_d}}{I_{X_T}}\left(1 + \left(\frac{1}{I_{inducer}/I_{K_m}}\right)^m\right)} + I_{Z_0}. \tag{1.20}$$

Remarkably, the above equation has the form that is identical to the equation derived for the observed fluorescence level in biological experiments, described in prior modeling work in [2]:

$$Z = \frac{G}{1 + \frac{1}{x}\left(1 + \left(\frac{1}{I}\right)^m\right)} + Z_0 \tag{1.21}$$

where Z, G, x, I, and Z_0 correspond to I_{GFP}, I_G, I_{X_T}/I_{K_d}, $I_{inducer}/I_{K_m}$, and I_{Z_0} in equation (1.20), respectively.

In a similar fashion, the behavior of the genetic repressor circuit in Figure 1-7(b) is modeled by the subthreshold transistor circuit in Figure 1-7(d). For this circuit, solving for the level of I_{GFP} yields

$$I_{GFP} = \frac{I_G}{1 + \frac{I_{X_T}/I_{K_d}}{1+(I_{inducer}/I_{K_m})^m}} + I_{Z_0} \tag{1.22}$$

and this is identical to the equation describing the observed fluorescence of EGFP

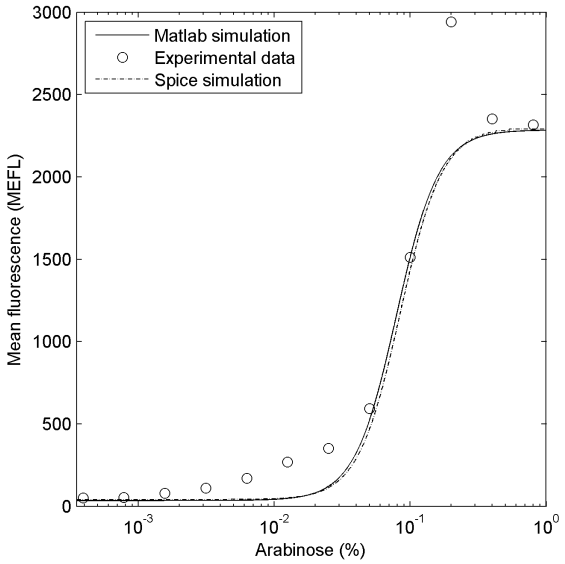
derived by [2]:

$$Z = \frac{G}{1 + \frac{x}{1+I^m}} + Z_0. \tag{1.23}$$

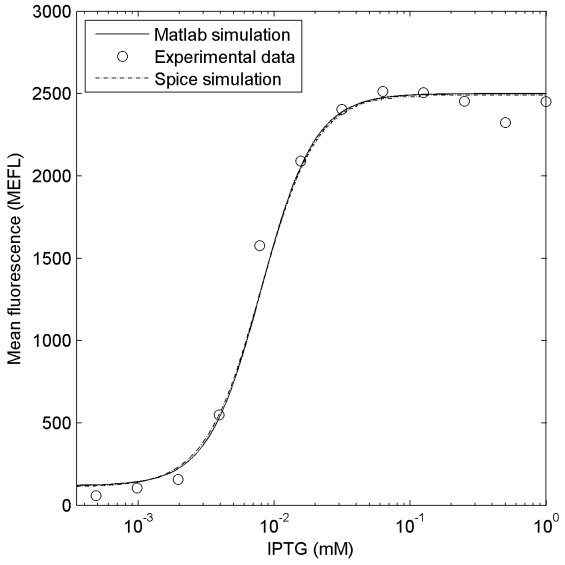
Note that there exists a physical symmetry between the electronic citcuit in Figure 1-7(c) and Figure 1-7(d). For the former circuit, the inducer-activator complex can bind to the DNA and the binding leads to higher expression of GFP. Thus, between the two arm currents of the two differential pairs, the left arm currents are chosen. On the other hand, for the repressor circuit, the "free" repressor molecules can bind to the DNA, and it leads to lower expression of GFP. Thus, between the two arm currents, the right arm currents are chosen. Due to the activation effect of Arabinose and de-repression effect of IPTG, the expression level of EGFP increaes as the inducer level increases for both circuits. This symmetry suggest that the same electronic circuit can be used to model both the activator and repressor genetic circuits, by simply implementing digital programmability which allows to choose which arm current to take.

Finally, the activator and repressor genetic circuits in Figure 1-7(a) and Figure 1-7(b) are built on a plasmid and transfected into *E. coli* using standard genetic experimental techniques [73], whose measurement results are shown in Figure 1-8. The MATLAB fit to equation (1.21) and (1.23) and the SPICE simulation result obtained from the electronic circuit in Figure 1-7(c) and 1-7(d) are also plotted in Figure 1-8. The plots illustrate that both the MATLAB and SPICE simulation generate identical fits to experimental data for a wide range of inducer levels.

Overall, this modeling and experimental work suggests that the use of current-mode analog transistor circuits can be an effective method to model and simulate the behavior of biological mechanisms. Although this section only presented circuits to capture the static (steady-state) behvaiors of biological models, the following chapters will show various current-mode circuits that quantitatively reproduce the dynamic behaviors as well.



(a)



(b)

Figure 1-8: Biological fluorescence data for the genetic circuits of (a) Figure 1-7(a) and (b) Figure 1-7(b), plotted with fits to the data by MATLAB and SPICE simulations of the circuit of (a) Figure 1-7(c) and (b) Figure 1-7(d) [16]. Copyright 2011 IEEE.

1.5 Thesis Organization

This thesis is organized as follows. Chapter 2 describes the cytomorphic current-mode building-block circuits that faithfully represent fundamental bio-molecular functions in cells. These blocks are composed and laid out on a silicon chip that we call the "gene chip". The capability of the gene chip to run deterministic and stochastic simulations of well-known synthetic genetic networks is illustrated.

In Chapter 3, the "protein chip" designed to capture complex dynamic behaviors of protein networks is presented. The protein chip is mainly composed of multiple copies of a versatile "protein" block which can be configured to model mass-action kinetics of any combination of zeroth, first, and second-order reactions, with wide-dynamic-range parameters. The chip simulation results of a p53 signaling pathway and a glycolysis pathway are shown, which quantitatively match with software simulation data.

Chapter 4 describes how these cytomorphic chips can be utilized to create a system which enables fast simulation of large-scale biochemical reaction networks. We show that the scalable architecture and parallelization incorporated in our system allow us to achieve speedup over conventional methods without compromising simulation accuracy, especially for simulation of multiscale networks. A proof-of-concept demonstration is carried out on a board capable of simulating up to 1,400 reactions using 20 cytomorphic chips. Three factors that may hinder simulation speed of the system are characterized and several ways to address them are discussed.

Finally, Chapter 5 concludes the thesis with the summary of author's contributions and future work.

Chapter 2

The Gene Chip

Gene regulatory networks are one of the primary decision making machinery in cells. They sense a great variety of environmental signals such as temperature, levels of metabolites, and availability of nutrients and respond accordingly [3]. Such environmental conditions are encoded by active or inactive states of transcription factors. Active transcription factors can bind DNA to regulate gene expression. Because of their importance as a controller, a sensor, a regulator, and a conveyor of information, gene regulatory networks must be studied to better understand cellular behaviors. Our gene chip is designed to fulfill this purpose.

In this chapter, we shall illustrate the blueprint of the gene chip in the following organization: We start with introducing the seven building-block circuits of the chip to model fundamental bio-molecular functions and our motivation to build them in Section 2.1. Section 2.2 describes the implementation of the building-block circuits along with chip measurement data. Section 2.3 discusses design considerations that are important in BiCMOS cytomorphic chip design. Section 2.4 discusses how cytomorphic building-block circuits may be composed to simulate synthetic biochemical reaction networks and architects this composition for three concrete biological examples. Section 2.5 summarizes the overall specifications of the chip. Section 2.6 concludes the chapter by summarizing our contributions.

Reprinted with permission from the publisher.

2.1 Introduction

The modeling and simulation of biochemical reaction networks in living cells that involve small molecules, DNA, RNA, and proteins is challenging for at least two reasons: 1) We still have a vast ignorance of the pathways and parameters of such networks, though our knowledge about them is rapidly increasing every day; 2) A fast, high-throughput simulation-and-modeling tool that can rapidly enable exploration, validation, learning, and constraining of the vast parameter and connectivity spaces with known experimental data is lacking. Just as Google helps us search for solutions amongst big-data spaces today, constantly learning to do better, if such a tool existed for the vast space of reaction networks, it could help us predict and design new experiments for further discovery.

However, it is computationally intensive and daunting to faithfully model cell-to-cell variability; Poisson stochastics due to modest molecular counts or high Fano factors in multiple state variables; host and environmental context; the feedback "loading" of downstream state variables on upstream state variables; diffusion and compartmentalization; cell-cycle and cell-division effects; unpredictable "cross talk" via shared polymerase, ribosome, ATP, protease, and RNAase resources; molecular toxicity due to high-copy-number, cross reactive, or drug effects; and interactions between cell-cell communication, regulatory, metabolic, developmental, and signaling pathways. Yet, many of these effects are important in systems and synthetic biology and in actual disease: For example, stochastics is important for tumor drug resistance [125] and for antibiotic drug resistance of "persister cells" [69]. Metabolic loading and molecular toxicity [110, 136], loading of one circuit by another [19], the breakdown of logic abstractions, and interactions between cellular resources [12, 110], have all prevented synthetic circuits from exceeding even six logic parts in one cell after almost two decades of research [102]. Therefore, there is motivation to create a fast-and-flexible simulation, modeling, and design tool where the addition of complexity in

modeling does not drastically compromise the speed of the simulation as it often does today.

Fortunately, the exponential Boltzmann equations that govern stochastic electron flow in a transistor and stochastic biochemical reaction flux in a chemical reaction are deeply mathematically similar as described in Section 1.4.2. This similarity enables us to map biochemical circuits to log-domain transistor circuits with a few handfuls of transistors per gene or protein [16, 78–80, 110, 116]. In turn, log-domain circuit motifs for linearization [130] have been mapped to create highly part-count efficient circuits in living cells [17]. The "cytomorphic" mapping of Figure 1-5, can enable rapid and highly parallel stochastic simulations of a single cell on a few chips and speedup multi-cell simulations on multi-chip electronic boards [116]. Therefore, we were motivated to create a few fundamental molecular circuits that could be composed and scaled to model large biochemical reaction networks. This chapter describes these fundamental building-block cytomorphic circuits. This chapter also focuses on establishing quantitative agreement of the cytomorphic chip measurements with prior biological measurements and models.

The building-block circuits may be classified into seven important categories:

1. Basic BiCMOS current-mode analog circuits exploit the log-domain cytomorphic mapping to capture the exact dynamics of **fundamental mass-action molecular kinetics** such as association, dissociation, and degradation. These fundamental circuits are general enough to capture subtle effects such as loading, fan-out, feedback, and substrate depletion through the use of a few explicit connections and Kirchhoff's current law. A wide dynamic range of operation and low power consumption are achieved through the use of bipolar and subthreshold MOS transistors that function at low current levels.

2. The ability to model cooperative binding is enabled by **tunable Hill-function** building-block circuits.

3. An "ITD" block built by a composition of above current-mode circuits enables mapping of the exact differential equations of **inducer-transcription-factor**

binding and transcription-factor-DNA binding including forward and reverse reactions, degradation, protection from degradation of transcription factors bound to DNA, and the change in DNA binding affinity of transcription factors when bound by an inducer.

4. An "analogic" current-mode circuit determines the transcription rate of genes based on the probability of multiple transcription-factor DNA binding sites being occupied or unoccupied in a combinatorial fashion with the relative mRNA production rate of each such combinatorial state programmable by the user. This strategy enables us to implement any arbitrary **"analogic promoter function"** that is capable of complex saturating digital logic or probabilistic analog behavior depending on the molecular concentration.

5. A current-mode low pass filter (LPF) circuit enables **the gain and dynamics of mRNA and protein production and degradation** to be represented.

6. **A stochastics circuit** intentionally amplifies analog Poisson noise in transistors to represent biological fluctuations in mRNA and protein concentrations at very low copy numbers and at relatively high noise levels. The Poisson nature of biochemical reaction fluxes automatically maps biological noise to electronic noise at high copy numbers and relatively low noise levels [116]. Thus, these circuits are most useful for reliably modeling highly stochastic and relatively low signal-to-noise-ratios in biological cells.

7. **ADCs and DACs** convert between analog currents and digital bits to enable our chips to communicate with each other via digital input/output (I/O), and with off-chip digital processors and computers.

Off-chip digital processors synergistically interact with our chips to carry out various functions: reading digital data from the chips; decoding the data; performing high-speed digital signal processing as necessary (e.g., scaling, diffusion, time delay, and error correction); encoding the data to create or modify molecular data packets via address and data strings; storing the programmable address connectivity amongst

gene and protein circuits; and communicating data to other chips or to a computer. For simplicity, the data in this chapter were collected with a data-acquisition board (NI PXI-6541) that interacted with our chip and with MATLAB on a computer. A high-performance FPGA (e.g., from the Xilinx Spartan family) could perform all of these functions as well. **Shift registers, SRAM blocks, and switches on the chip** enable programmability of parameters (e.g., reaction rates, dissociation constants, Hill coefficients, and time constants) as well as connectivity.

2.2 Building-Block Circuits

This section describes the details of the seven building blocks that can be programmed to capture the essential dynamics of molecular basis-function circuits in cells. These blocks were implemented in a proof-of-concept VLSI chip fabricated in an AMS 0.35 μm BiCMOS process. As shown in Figure 2-1, the chip contains identical gene blocks. To prove the functionality of on-chip building block circuits, the input-output characteristics for each basis-function circuit created on the chip was compared with ideal simulation results produced by MATLAB. MATLAB was also used to perform chip programming and readout of chip data via a NI PXI-6541 data-acquisition board. To obtain values of chip data shown in this section, the outputs of ADC's from the chip were converted into actual current levels via measured scale factors.

2.2.1 Mass Action and Michaelis-Menten Reaction Block

Among various different approaches to represent gene-protein networks, including directed graphs, Bayesian networks, Boolean networks, differential equations, and master equations [18], our circuit schematics are based on ordinary differential equations (ODEs) that automatically incorporate Poisson stochastics into electron current fluxes [116]. Electron copy number is analogous to molecular copy number. We assume that the laws of mass action determine chemical reaction kinetics. The most fundamental molecular basis functions that need to be implemented are the rate equations of the following four elementary reactions:

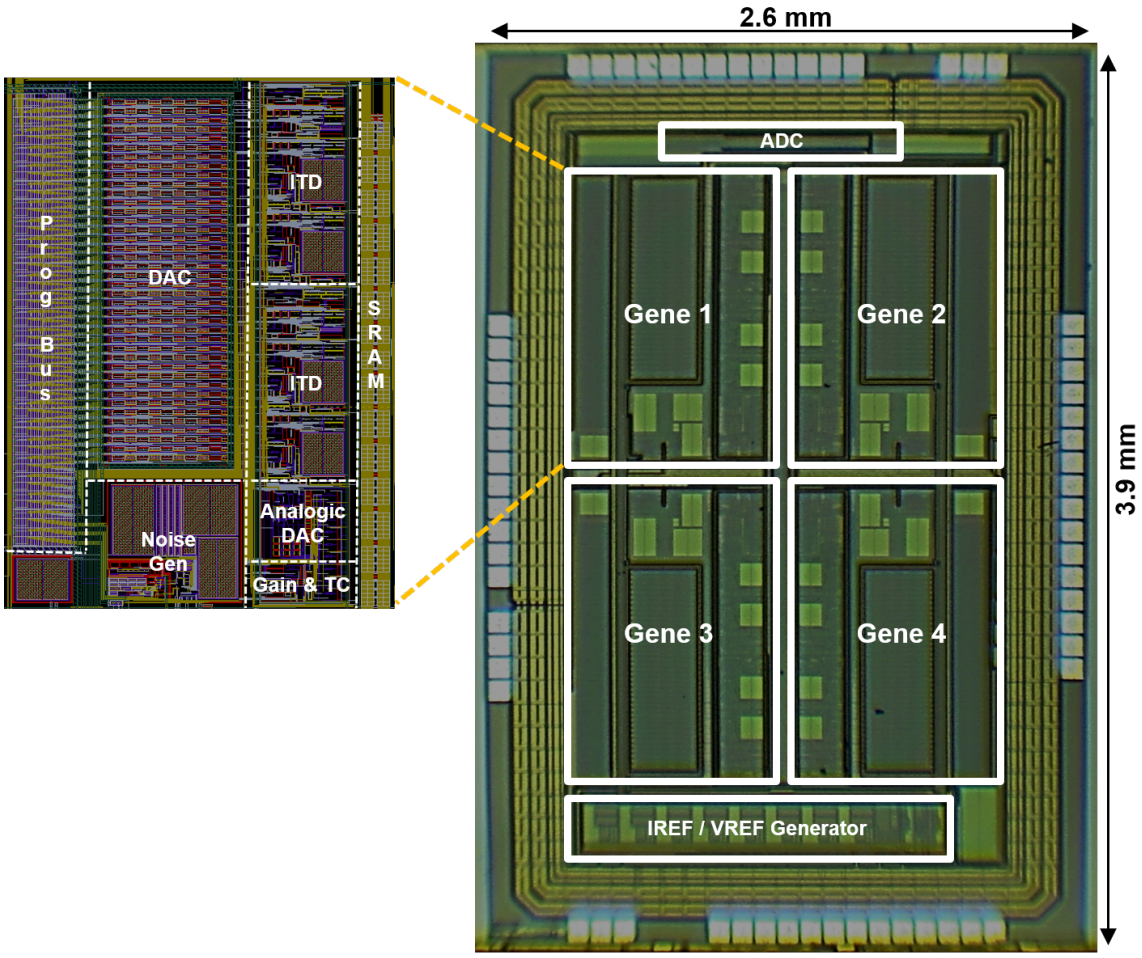
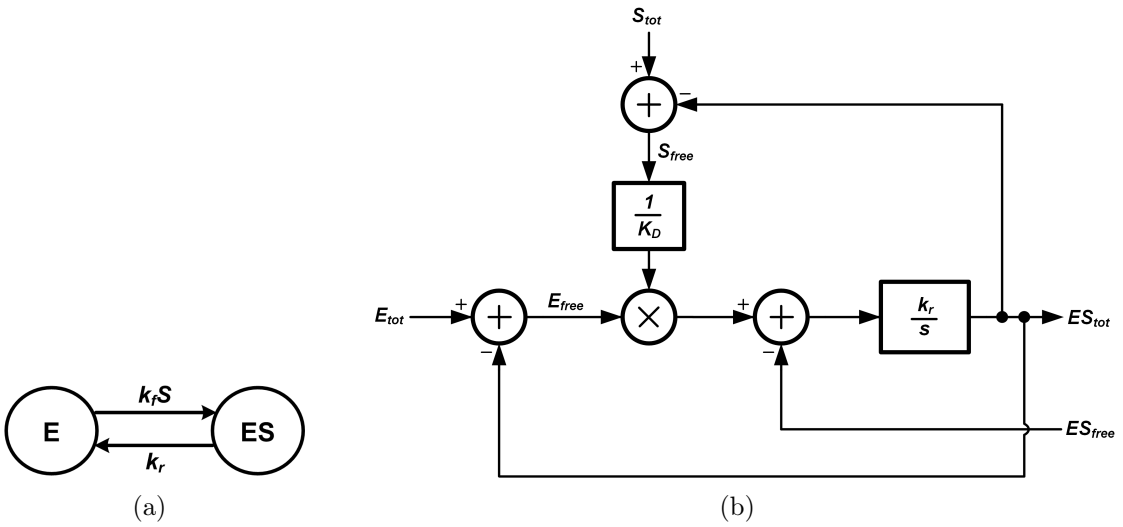


Figure 2-1: Die micrograph of the 2.6 mm × 3.9 mm cytomorphic chip fabricated in an AMS 0.35 μm BiCMOS process. The left inset is a layout screen capture of one gene block (2x magnification).



(a) (b)

Figure 2-2: (a) Simple representation of an enzyme-substrate binding reaction, $E + S \rightleftharpoons ES$. (b) Block diagram representation of the same reaction.

1. Production: $\phi \rightarrow E$

2. Degradation: $E \rightarrow \phi$

3. Association: $E + S \rightarrow ES$

4. Dissociation: $ES \rightarrow E + S$

All complicated biochemical reaction networks can be decomposed into a series of these reactions. Figure 2-2(a) shows a simple diagram to depict a network with an association (forward) and a dissociation (reverse) reaction. A block diagram we use to model the same network is shown in Figure 2-2(b). For the two diagrams, typical symbols to describe Michaelis-Menten kinetics are used — E for enzyme, S for substrate, ES for enzyme-substrate complex, k_f and k_r for rate constants of forward and reverse reactions, respectively, and $K_D = k_r/k_f$ for the dissociation constant of ES. This block is the main building block of our system, which is capable of simulating all of the four elementary reactions listed above. The production and degradation reactions can often be more efficiently simulated by simpler blocks as well.

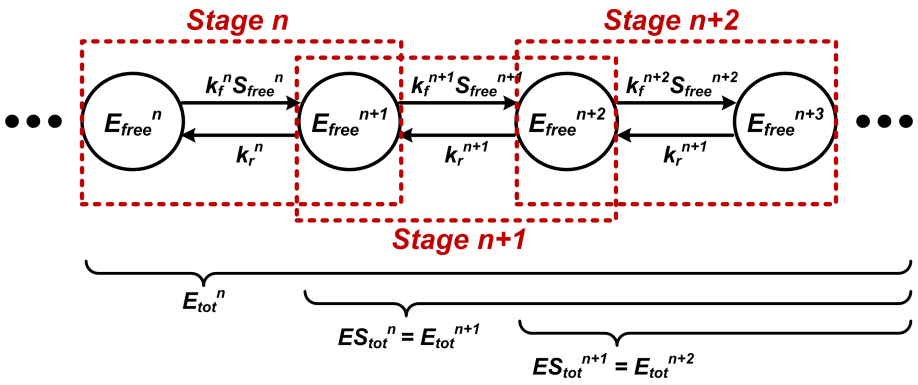


Figure 2-3: A cascade reaction network to depict how "total" and "free" variables are defined.

The block diagram of Figure 2-2(b) basically solves the following three differential equations:

$$\frac{d[ES_{tot}]}{dt} = k_f[E_{free}][S_{free}] - k_r[ES_{free}] \tag{2.1}$$

$$\frac{d[E_{free}]}{dt} = -k_f[E_{free}][S_{free}] + k_r[ES_{free}] \tag{2.2}$$

$$\frac{d[S_{free}]}{dt} = -k_f[E_{free}][S_{free}] + k_r[ES_{free}]. \tag{2.3}$$

Note that E_{tot}, S_{tot}, and ES_{tot} in Figure 2-2(b) are "total" variables that include all downstream quantities necessary for "loading." They are used to compute the "free" variables, E_{free} and S_{free}. ES_{free} is the free amount of ES available that is not bound up in any downstream reactions. If ES is not used anywhere, ES_{tot} becomes equal to ES_{free}, a special case in which the steady state output of this network is given by

$$[ES_{tot}] = [E_{tot}]\left(\frac{[S_{free}]/K_D}{1 + [S_{free}]/K_D}\right). \tag{2.4}$$

Figure 2-3 is drawn to further illustrate the notion of "total" and "free" variables. Let's consider a cascade reaction network where multiple substrates bind to an enzyme in sequence. In stage n, the inputs are denoted as E_{tot}^n and S_{tot}^n, from which E_{free}^n and S_{free}^n are obtained and in turn ES_{tot}^n is computed. This ES_{tot}^n becomes E_{tot} of

the next stage. That is,

$$ES_{tot}^{n} = E_{tot}^{n+1} \tag{2.5}$$

$$ES_{tot}^{n+1} = E_{tot}^{n+2} \tag{2.6}$$

$$ES_{tot}^{n+2} = \cdots \tag{2.7}$$

Besides, as the braces in Figure 2-3 indicate, E_{tot} of each stage is the sum of E_{free} values of the current stage and all subsequent stages. Accordingly, E_{free} of the current stage is given by

$$E_{free}^{n} = E_{tot}^{n} - ES_{tot}^{n} = E_{tot}^{n} - E_{tot}^{n+1}. \tag{2.8}$$

It should be emphasized that the use of "total" variables allows the solutions of differential equations to reach their desired steady-state values. For example, consider a simple transformation reaction $\mathrm{A} \underset{k_r}{\overset{k_f}{\rightleftharpoons}} \mathrm{B}$. A conventional way of solving this in analog hardware is to implement two separate differential integrator circuits to represent the time derivative of [A] and [B], respectively, as has been demonstrated previously [79]. The two time derivatives are given by

$$\frac{d[A]}{dt} = -k_f[A] + k_r[B] \tag{2.9}$$

$$\frac{d[B]}{dt} = \alpha_1 k_f[A] - \alpha_2 k_r[B] \tag{2.10}$$

where α_1 and α_2 are the scale factors to account for inevitable mismatch present in any analog transistor circuits. The above equations have the solution of the form

$$[A] = c_1 e^{\lambda_1 t} + c_2 e^{\lambda_2 t} \tag{2.11}$$

$$[B] = c_3 e^{\lambda_1 t} + c_4 e^{\lambda_2 t} \tag{2.12}$$

where λ_1 and λ_2 are the eigenvalues of the Jacobian matrix of the network, given by

$$\mathbf{J} = \begin{bmatrix} -k_f & k_r \\ \alpha_1 k_f & -\alpha_2 k_r \end{bmatrix}. \tag{2.13}$$

The eigenvalues of this matrix can be obtained by solving

$$\lambda^2 + (k_f + \alpha_2 k_r)\lambda + (\alpha_2 k_f k_r - \alpha_1 k_f k_r) = 0. \tag{2.14}$$

Equations (2.11) and (2.12) show that unless one eigenvalue is zero, the steady-state solutions go to 0 or ∞. From (2.14), we see that a zero eigenvalue arises when $\alpha_2 k_f k_r - \alpha_1 k_f k_r = 0$, i.e., $\alpha_1 = \alpha_2$. Since transistor mismatch is largely determined by "random" process variations, it is practically impossible to meet this condition.

On the other hand, with our scheme using "total" variables, the network is described by

$$[A] = [A_{tot}] - \alpha[B] \tag{2.15}$$

$$\frac{d[B]}{dt} = k_f[A] - k_r[B] \tag{2.16}$$

where α is a scale factor to represent transistor mismatch. Deriving the time derivative of [A] from (2.15) yields

$$\frac{d[A]}{dt} = -\alpha \cdot \frac{d[B]}{dt} \tag{2.17}$$

$$= -\alpha k_f[A] + \alpha k_r[B]. \tag{2.18}$$

Thus, the Jacobian matrix of the network given by

$$\mathbf{J} = \begin{bmatrix} -\alpha k_f & \alpha k_r \\ k_f & -k_r \end{bmatrix} \tag{2.19}$$

has an eigenvalue of zero, which indicate that our circuit can produce a steady-state solution (other than 0 or ∞). Intuitively, the former method requires two balances (equations (2.9) and (2.10)) to be satisfied at the same time (i.e., both derivatives reaching zero). However, they are essentially the same equations except that mismatch is present. Hence, they cannot be satisfied. On the other hand, our method requires only one balance (equation (2.16)) to be satisfied, which is possible.

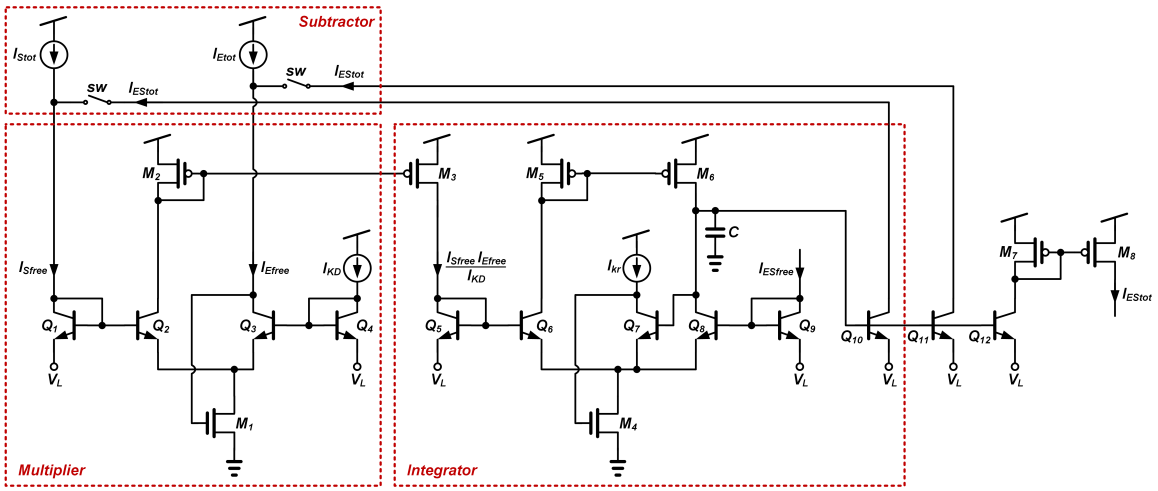


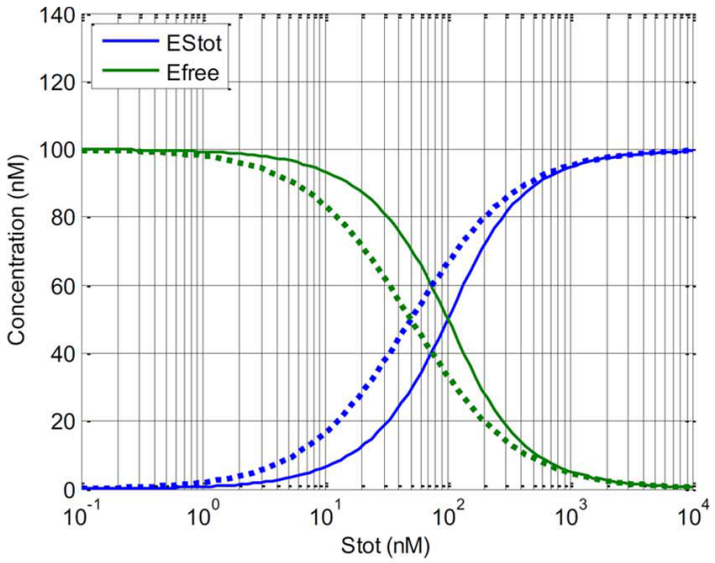
Figure 2-4: Transistor schematic of the reaction block of Figure 2-2(b).

Figure 2-4 shows the transistor schematic of the block diagram in Figure 2-2(b), constructed based on BiCMOS current-mode circuits [116]. The circuit consists of a multiplier, an integrator, and two subtractors. The multiplier block composed of Q_1–Q_4 and M_1 calculates $I_{Sfree}I_{Efree}/I_{KD}$ from two variables I_{Efree} and I_{Sfree} and one parameter I_{KD}. The differential integrator block formed by Q_5–Q_{10} and M_4–M_6 gives

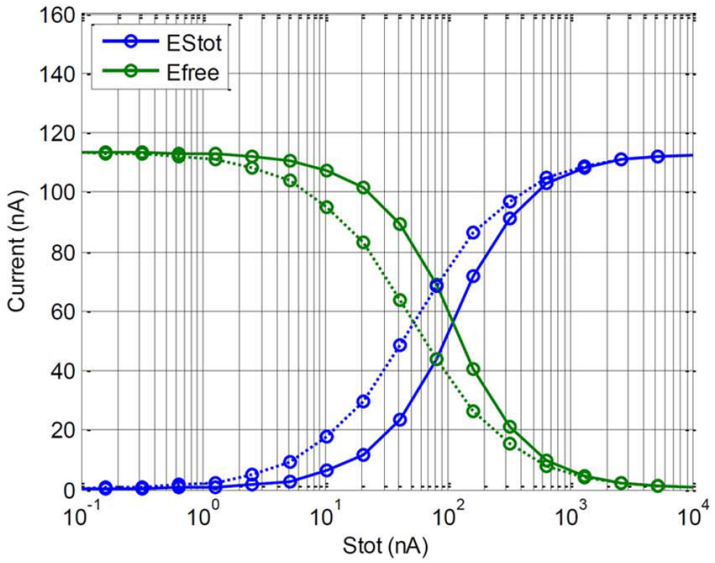
$$\frac{dI_{EStot}}{dt} = \left(\frac{I_{Sfree}I_{Efree}}{I_{KD}} - I_{ESfree}\right) \cdot \frac{I_{kr}}{C\phi_t} \tag{2.20}$$

which is equivalent to (2.1). Multiple copies of its output current are created, which are used for subtraction to compute E_{free} and S_{free} or sent to other blocks. Each of the two subtractors is easily created by the two currents gathering into a node. Thus, current-mode circuits effectively implement multiplication and integration through Kirchhoff's voltage law (KVL) and addition and subtraction through Kirchhoff's current law (KCL). The switch shown in Figure 2-4 is useful in experimentally illustrating the effects of loading in biochemical circuits as discussed later.

Figure 2-5(a) and 2-5(b) show the MATLAB simulation results and the chip data, respectively, of the block in Figure 2-2(b). It was assumed that $ES_{free} = ES_{tot}$, since we are at first, modeling a single biochemical reaction, which is not part of a cascade network with loading. We discuss loading effects that affect ES_{free} later. The steady-state values of E_{free} and ES_{tot} are plotted, while S_{tot} is varied from 100



(a)



(b)

Figure 2-5: (a) MATLAB and (b) chip data for the Michaelis-Menten reaction block in steady state, with (solid lines) and without (dotted lines) substrate depletion, which represents a typical loading effect. The switches in Figure 2-4 were used to experimentally introduce the effects of loading (or not). Lines shown in (b) are connection between points, not any fits, unlike in Figure 2-6, where MATLAB fits and chip data are explicitly compared. Note that chip data are ADC outputs divided by the scale factors to obtain current levels. Chip parameters: $I_{Etot} = 100$ nA, $I_{Stot} =$ 100 pA–10 μA, and $I_{KD} = 50$ nA.

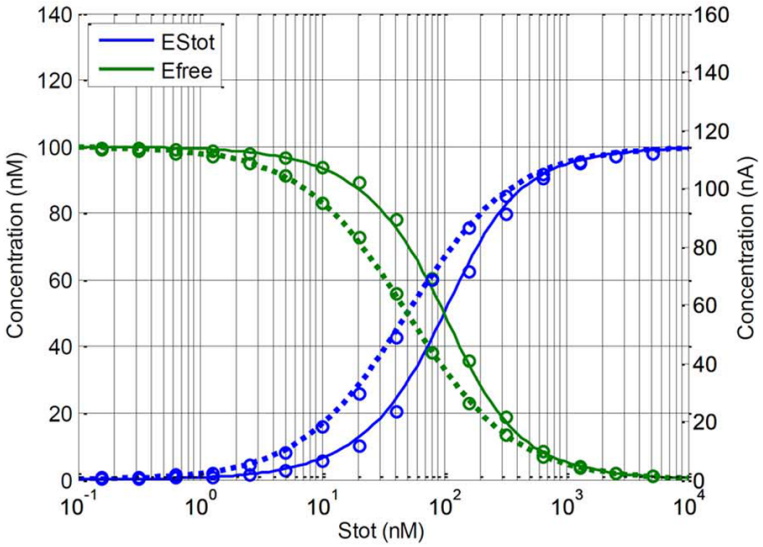


Figure 2-6: Chip data (circles) plotted on top of MATLAB data (lines).

pA to 1 μA and E_{tot} and K_D are fixed at 100 nA and 50 nA, respectively. It can be seen from Figure 2-6 that with the concentration scale mapped from 100 nM to 115 nA for ES_{tot}, the MATLAB plot becomes a reasonable fit to the chip data. Although not shown, this Michaelis-Menten reaction block in the chip mimics the dynamics of the MATLAB model as well.

The dotted traces of Figure 2-5 correspond to the results when substrate depletion (substrate being used up whenever it binds to an enzyme) is ignored, i.e., when the switch in Figure 2-4 to subtract ES from S_{tot} is programmed as "off" and loading effects such as substrate depletion are ignored. By comparing the dotted and solid traces, the effect of substrate depletion can be clearly observed. Generally, this effect is ignored by many researchers, under the assumption that substrate is much more abundant than enzyme, or to simplify analytic solutions. However, substrate depletion may exert a noticeable effect when the level of substrate is comparable to enzyme. Common protease, RNAase, ATPase, polymerase, or ribosome resources that are shared amongst many circuits exhibit "resource depletion" that manifests in an analogous fashion to the substrate-depletion example shown here. Figure 2-5 shows that such depletion manifests as an increase in the effective K_D and Hill coefficient values [17]. It should be noted that, since physical cytomorphic circuits represent

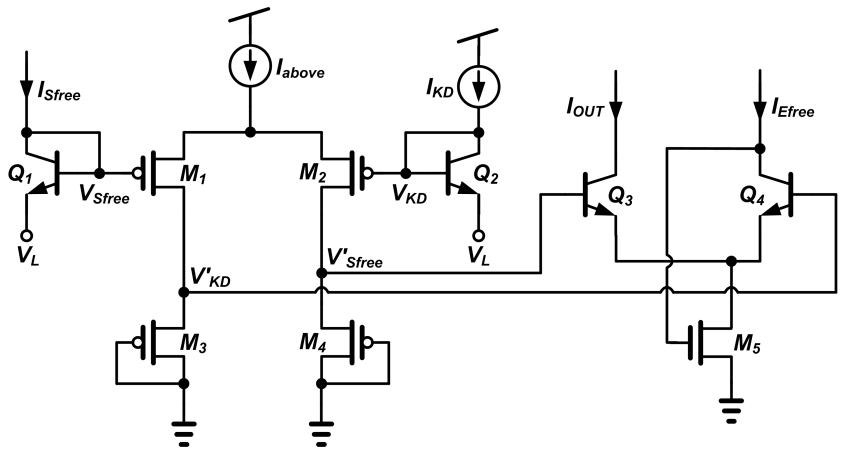


Figure 2-7: A cytomorphic circuit implements a Hill coefficient greater than 1 by amplifying the voltage difference between V_{Sfree} and V_{KD}, using hybrid bipolar-and-above-threshold circuits.

biomolecular circuits efficiently, this loading effect is captured by the simple addition of a wire and a switch in Figure 2-4.

2.2.2 Hill Block

The Hill coefficient is a parameter used in biochemistry to characterize the effect of cooperative binding. Note that (2.4), which is realized by the circuit of Figure 2-4 has a Hill coefficient of 1. Figure 2-7 shows how we can build steeper Hill-coefficient circuits. The voltage difference between the logarithm of I_{Sfree} and I_{KD} is amplified to yield

$$V'_{Sfree} - V'_{KD} = n\,(V_{Sfree} - V_{KD}) \tag{2.21}$$

that then gives the term $(I_{Sfree}/I_{KD})^n$ in place of I_{Sfree}/I_{KD} in (2.4). This amplification is done by a differential amplifier operating in the above-threshold regime (created by the abovethreshold current I_{above}), where its gain is set by the ratio between $\sqrt{W/L}$ of upper (M_1 and M_2) and lower (M_3 and M_4) PMOS transistors. By digitally programming the effective size of the upper transistors between 1x to 16x through switching bits, we can alter the Hill coefficient between 1 and 4. This range of Hill coefficients enables us to represent almost all cases of biological operation.

Figure 2-8(a) and 2-8(b) show the MATLAB simulation results and the chip data,

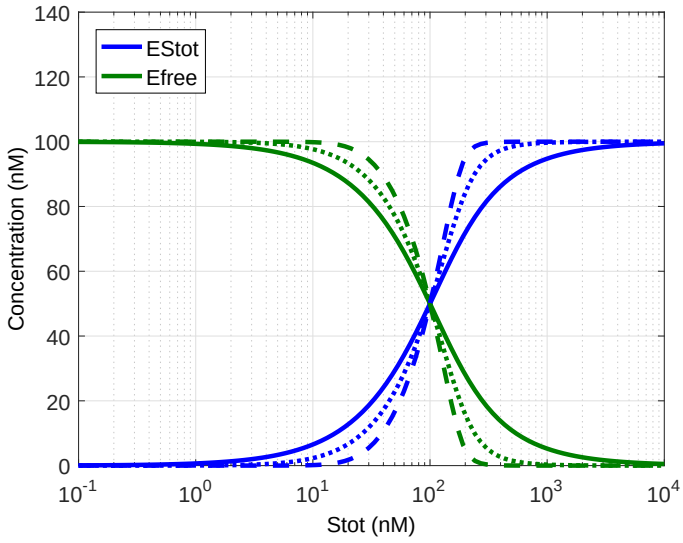
respectively, of the Michaelis-Menten reaction block in Figure 2-2(b), for different Hill coefficients. All variables are the same as in Section 2.2.1, except that the Hill coefficient is set as 1, 2, or 4 respectively. As Figure 2-8 reveals, the chip data exhibit the same behavior as the MATLAB model.

2.2.3 ITD Block

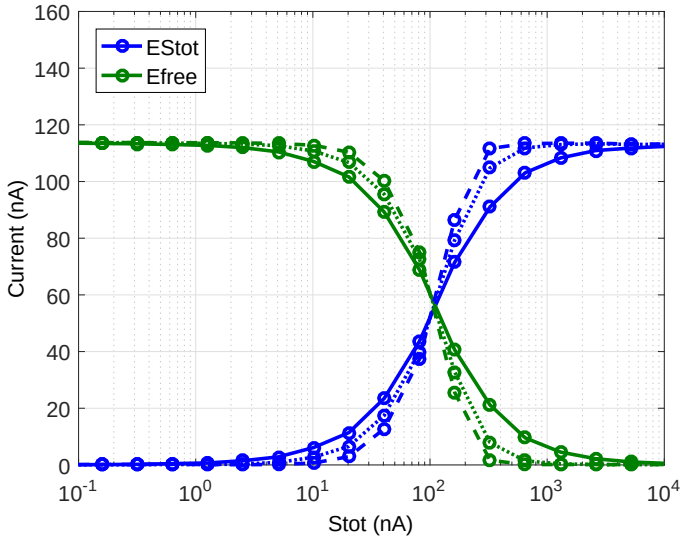
Each of the two ITD blocks shown in Figure 2-1 simulates the dynamics of inducer-transcription factor binding and transcription factor-DNA binding. Figure 2-9(a) is a simplified diagram of the ITD block depicting the key reactions among inducer molecules (I), transcription factors (TF), and DNA binding sites (DNA). The names of the molecules and bound complexes are self-explanatory. Both TF and I-TF are allowed to bind to DNA with programmably different binding affinities. The binding of an inducer to a transcription factor causes its binding affinity to DNA to typically change by a factor of 10 to 100 in cells. Some transcription factors (e.g., AraC) can act both as an activator and as a repressor, depending on whether inducers are bound or not. All of these cases can be modeled by our ITD and analogic DAC blocks (described in the next section).

A more detailed block diagram representation of the same network is shown in Figure 2-9(b). First, the MM_Static block models inducer-transcription factor binding and is a modified version of the circuit described in [16]. Since the dynamics of inducer and transcription factor binding is usually a few orders of magnitude faster than the dynamics between transcription factors and DNA binding [3, 92], such binding is assumed to reach steady state nearly instantaneously, and is therefore neglected, as in most models.

The two MM_Basic blocks represent the basic Michaelis-Menten reaction blocks shown in Figure 2-2 and 2-4. The S_{tot} input for the two blocks is represented by TF_{bnd} and TF_{free}, equivalent to I-TF and TF in Figure 2-9(a) respectively. Since a DNA binding site can be bound with either I-TF or TF, TF_{bnd} and TF_{free} represent "total substrate variables" for the DNA, and Figure 2-9(a) and 2-9(b) effectively implement a competitive fan-out condition for the DNA binding site. The amount of TF-DNA

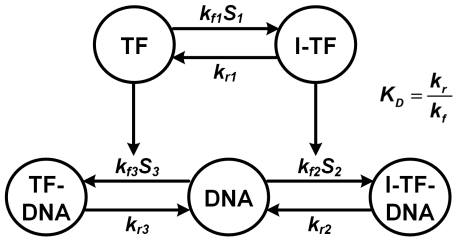


(a)

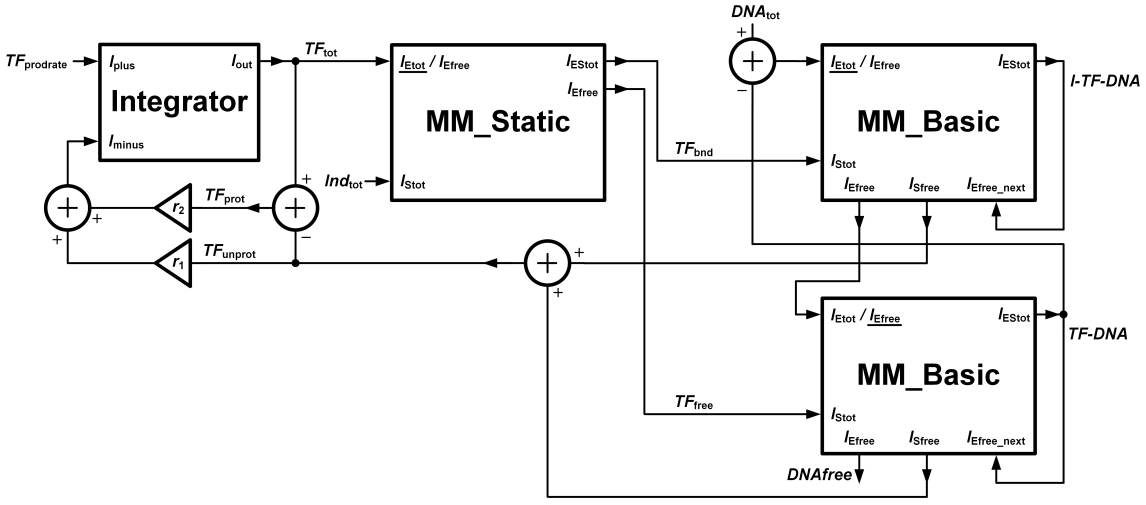


(b)

Figure 2-8: (a) MATLAB and (b) chip data for the Hill block in steady state, for three Hill coefficients, 1 (solid lines), 2 (dotted lines), and 4 (dashed lines). Chip parameters: $I_{Etot} = 100$ nA, $I_{Stot} = 100$ pA–10 μA, and $I_{KD} = 50$ nA.



(a)



(b)

Figure 2-9: (a) Simplified diagram showing the key reactions of the ITD block. (b) Block diagram of the ITD block. The integrator models the production and degradation of transcription factor; MM_Static models the steady-state behavior of inducer-transcription factor binding; the two MM_Basic blocks model the dynamics of transcription factor-DNA binding when the transcription factor is bound to an inducer (TF_{bnd} input) or not (TF_{free} input).

from the bottom MM_Basic block is subtracted from DNA_{tot} to yield the total number of DNA sites available for $I\text{-}TF$ binding in the top MM_Basic block. The net unbound DNA from the top MM_Basic block is the E_{free} input to the bottom MM_Basic block. It is worth noting that the top MM_Basic block explicitly uses the switch in Figure 2-4 to achieve correct loading since its input is an E_{tot} input. In contrast, the bottom MM_Basic block, does not use this switch since its input is already an E_{free} input.

The integrator block in Figure 2-9(b) is a differential integrator with production-rate and degradation-rate inputs that determine the total amount of the transcription factor. A circuit similar to the integrator included in the MM_Basic block and described in [116] is used. We modeled the protection effect of DNA on transcription factor degradation [17]: The degradation rate of a transcription factor can drop when it binds to DNA because a certain part of it is hidden and it becomes harder for a protease to digest the transcription factor. Thus, in Figure 2-9(b) different degradation rate constants (r_1 and r_2) multiply the amount of transcription factor that is unbound (TF_{unprot}) or bound (TF_{prot}) to DNA, respectively. The sum of the two multiplied results is the negative input of the integrator block.

Our ITD model was chosen because it captures important gene-protein dynamics seen in practice. Figure 2-9(b) shows that it is constructed by a straightforward arrangement and wiring of blocks in Figure 2-4 and 2-9. Here we list the assumptions inherent in our ITD model, which are explicitly or implicitly used by most other models that fit biological data as well, e.g., those described in [3]:

1. The binding of inducers to transcription factors occurs at a much faster rate than the rest of the network and can thus be assumed to be nearly instantaneous.

2. The affinity of inducers to transcription factors remains the same, regardless of whether transcription factors are bound to DNA or not.

3. The degradation of inducers is ignored; when a transcription factor degrades, its associated inducer is automatically freed.

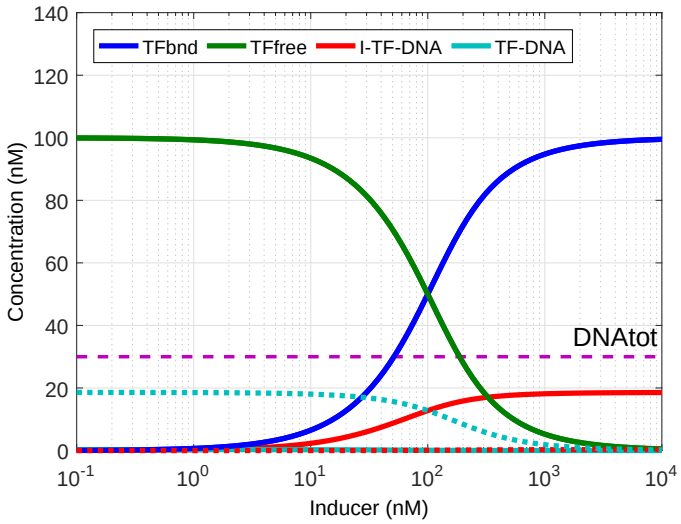
4. When multiple binding sites for the same transcription factor exist on the DNA, each binding event is independent of other events.

Though it is not very useful for fitting biological data in most situations, it is possible to not make any of the above assumptions and to design more complex cytomorphic ITD circuits that capture other scenarios.

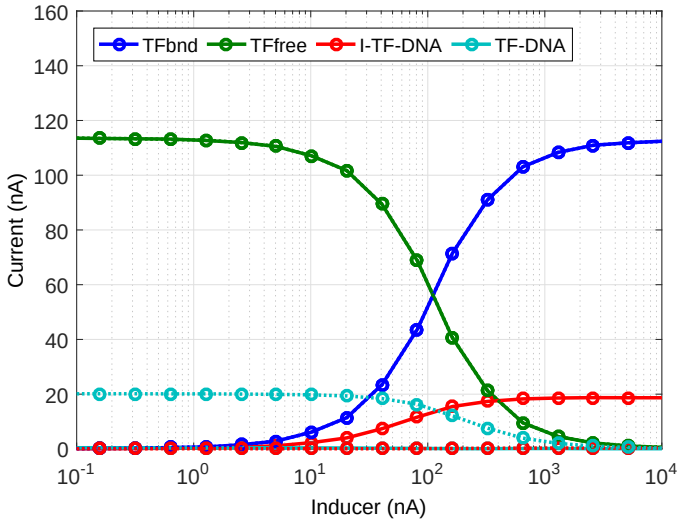
Figure 2-10(a) and 2-10(b) show the steady-state simulation results of the ITD block using MATLAB and the chip, respectively, for equivalent parameter sets. As for the inducer-transcription factor binding, the same variables are used as in Section 2.2.1. As for the transcription factor-DNA binding, $DNA_{tot} = 30$ nA, and two different parameter sets are used for K_{D2} and K_{D3}: 1) $K_{D2} = 50$ nA, $K_{D3} = 10$ μA (solid traces) and 2) $K_{D2} = 10$ μA, $K_{D3} = 50$ nA (dotted traces). The former is the case when only TF_{bnd} can bind DNA, and the latter is when only TF_{free} can bind DNA. The amount of DNA_{tot} is drawn in Figure 2-10(a) as a dashed line, so readers can estimate the amount of $DNA_{free} = (DNA_{tot} - I\text{-}TF\text{-}DNA - TF\text{-}DNA)$. It can be seen that the chip produces outputs that are very similar to the plots generated by MATLAB.

2.2.4 Analogic DAC

Our models assume two sites in a DNA promoter where transcription factors can bind, which is typical for most microbes. Different transcription factors can bind different binding sites. The rate of mRNA synthesis varies depending on the probabilistic state of each binding site. In the ITD model in Figure 2-9(a), binding sites can be bound by either $I\text{-}TF$ or TF, or by neither. Correspondingly, there are three outputs of each ITD block, $I\text{-}TF\text{-}DNA$, $TF\text{-}DNA$, and DNA_{free}, which are output to the analogic DAC. Then, using the circuit shown in Figure 2-11, the three binding probabilities of



(a)



(b)

Figure 2-10: (a) MATLAB and (b) chip data for the ITD block in steady state. Chip parameters: $I_{Etot1}(TF_{tot}) = 100$ nA, $I_{Stot1}(Ind_{tot}) = 100$ pA–10 μA, $I_{KD1} = 50$ nA, and $I_{Etot2}(DNA_{tot}) = 30$ nA. Also, $I_{KD2} = 50$ nA and $I_{KD3} = 10$ μA for solid lines, and $I_{KD2} = 10$ μA and $I_{KD3} = 50$ nA for dotted lines. The dashed line in (a) indicates the level of DNA_{tot}, from which DNA_{free} can be estimated.

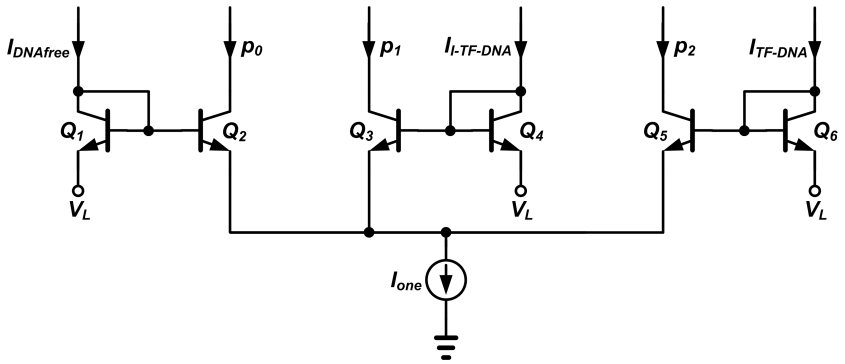


Figure 2-11: Circuit used in the analogic DAC to compute the three probabilities (p_0, p_1, and p_2) of a DNA binding site.

a site, p_0, p_1, and p_2, are computed as

$$p_0 = I_{one}\left(\frac{I_{DNAfree}}{I_{DNAfree} + I_{I-TF-DNA} + I_{TF-DNA}}\right) = I_{one}\left(\frac{I_{DNAfree}}{I_{DNAtot}}\right) \tag{2.22}$$

$$p_1 = I_{one}\left(\frac{I_{I-TF-DNA}}{I_{DNAfree} + I_{I-TF-DNA} + I_{TF-DNA}}\right) = I_{one}\left(\frac{I_{I-TF-DNA}}{I_{DNAtot}}\right) \tag{2.23}$$

$$p_2 = I_{one}\left(\frac{I_{TF-DNA}}{I_{DNAfree} + I_{I-TF-DNA} + I_{TF-DNA}}\right) = I_{one}\left(\frac{I_{TF-DNA}}{I_{DNAtot}}\right) \tag{2.24}$$

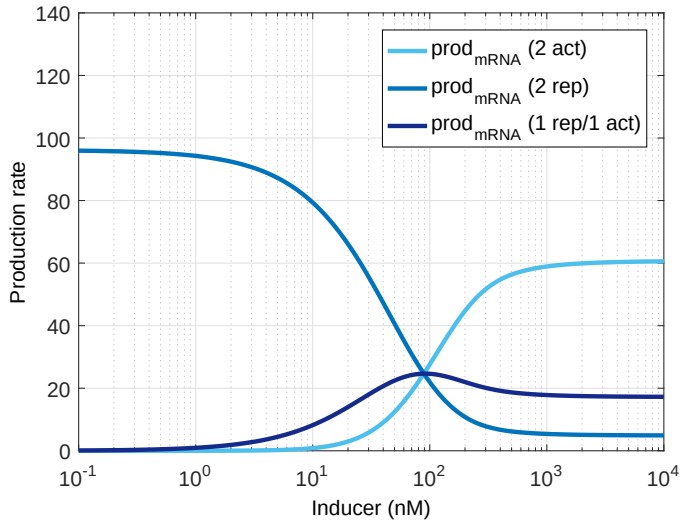
where I_{one} is a constant reference current which corresponds to 100% probability. An identical circuit is used for a second binding site, with p_0, p_1, and p_2 used in place of I_{one} such that all nine probabilities, p_{00}, p_{01}, p_{02}, p_{10}, p_{11}, p_{12}, p_{20}, p_{21}, and p_{22}, for two binding sites can be computed. For example, p_{00} is given by

$$p_{00} = I_{one}\left(\frac{I_{DNAfree1}}{I_{DNAtot1}}\right)\left(\frac{I_{DNAfree2}}{I_{DNAtot2}}\right). \tag{2.25}$$

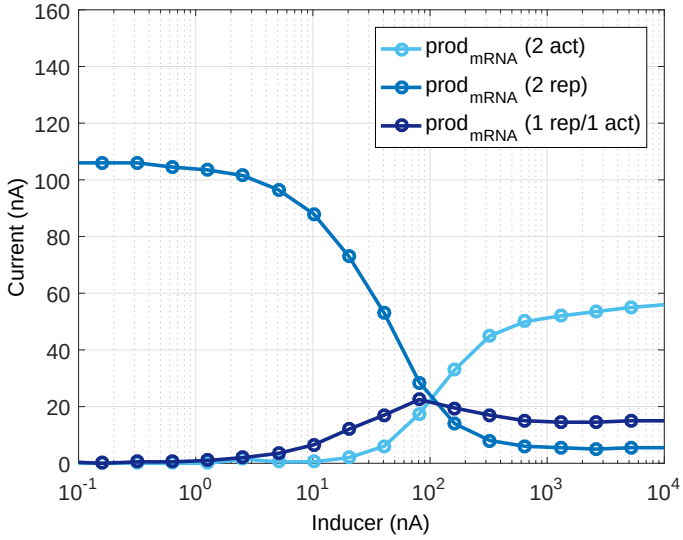
The net mRNA synthesis rate is determined by the sum of the probabilities of the promoter being in each of the nine states multiplied by the relative mRNA synthesis rate, from β_{00} to β_{22}, corresponding to that state [45]. That is,

$$prod_{mRNA} = \sum_{i=0}^{2}\sum_{j=0}^{2}\frac{p_{ij}\beta_{ij}}{I_{one}} \tag{2.26}$$

where $\frac{p_{ij}\beta_{ij}}{I_{one}}$ can be obtained by using the multiplier included in Figure 2-4.



(a)



(b)

Figure 2-12: (a) MATLAB and (b) chip data for the analogic DAC in steady state. Chip parameters (two ITD blocks): $I_{Etot1}(TF_{tot}) = 200$ nA, $I_{Stot1}(Ind_{tot}) = 100$ pA–10 μA, $I_{KD1} = 50$ nA, $I_{Etot2}(DNA_{tot}) = 30$ nA, $I_{KD2} = 50$ nA and $I_{KD3} = 10$ μA. Chip parameters (Analogic DAC): 1) $\beta_{11} = 100$ nA, other $\beta_{ij} = 0$ (two activators), 2) $\beta_{00} = 100$ nA, other $\beta_{ij} = 0$ (two repressors), and 3) $\beta_{01} = 100$ nA, other $\beta_{ij} = 0$, (one repressor and one activator).

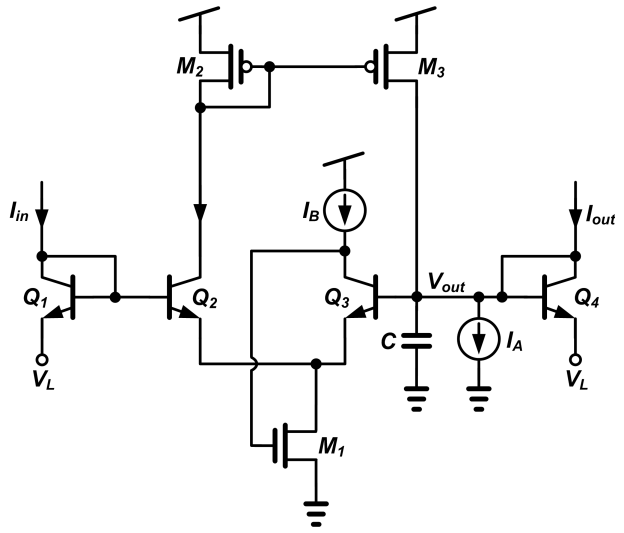


Figure 2-13: Current-mode low-pass filter (LPF) circuit used to set the gain and the time constant of transcription (and translation).

Figure 2-12(a) and 2-12(b) show the plots of the mRNA production rate simulated by MATLAB and the chip, respectively. Three different configurations were made for the analogic DAC: 1) every $\beta_{ij} = 0$ except $\beta_{11} = 100$ nA (i.e., two activators), 2) every $\beta_{ij} = 0$ except $\beta_{00} = 100$ nA (i.e., two repressors), and 3) every $\beta_{ij} = 0$ except $\beta_{01} = 100$ nA (i.e., one repressor and one activator). Since the parameters for the two ITD blocks were set such that $I\text{-}TF\text{-}DNA$ increases as inducer level increases, for each configuration, the level of $prod_{mRNA}$ 1) increases; 2) decreases; and 3) increases and then decreases, as shown in both Figure 2-12(a) (MATLAB) and Figure 2-12(b) (Chip).

2.2.5 Gain & Time Constant Block

Figure 2-13 shows a current-mode low-pass filter capable of setting the gain and time constant of transcription. The input current I_{in} is mRNA synthesis rate generated by the analogic DAC, the output current I_{out} is mRNA concentration, and the two programmable currents I_A and I_B set the gain (I_B/I_A) and the time constant ($C\phi_t/I_A$) [116]. That is,

$$\frac{I_{out}(s)}{I_{in}(s)} = \frac{I_B/I_A}{1 + s\left(\frac{C\phi_t}{I_A}\right)} \tag{2.27}$$

which is equivalent to the following differential equation:

$$\frac{d[mRNA]}{dt} = \alpha_{mRNA} - \gamma_{mRNA}[mRNA] \tag{2.28}$$

where $\alpha_{mRNA} = I_{in}I_B/(C\phi_t)$ and $\gamma_{mRNA} = I_A/(C\phi_t)$ are the production and degradation rate of mRNA, respectively. The gain term may also include a protein production-rate constant (whose unit is proteins/mRNA·s). In this case, I_{out} becomes the rate at which a protein is produced, which can directly be used as the $TF_{prodrate}$ input of an ITD block. Occasionally, the kinetics of mRNA and protein molecules are lumped into a single production rate and a degradation rate, under the condition that the time scale of protein dynamics is much slower than mRNA. Then, I_{out} corresponds to the protein concentration.

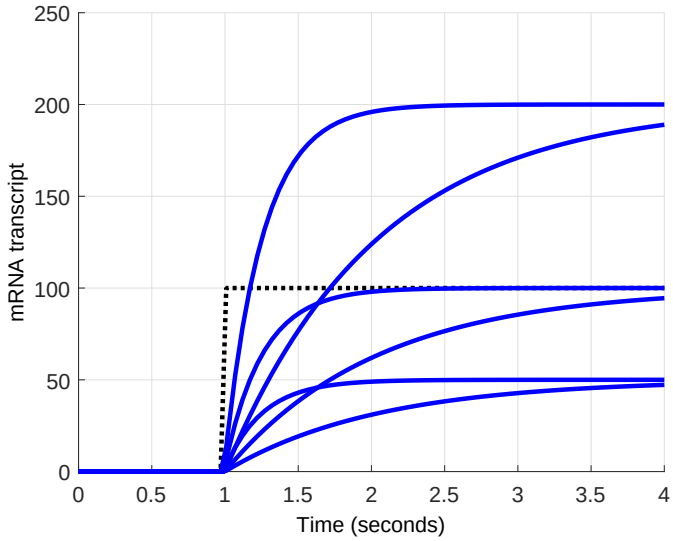
Figure 2-14(a) and 2-14(b) show the MATLAB simulation data and the chip data, respectively. The gain & time constant block were tested for three different gains, 0.5, 1, and 2, and two time constants, 0.25 s and 1 s, for a step input [shown as the dotted trace in Figure 2-14(a)].

2.2.6 Noise Generator

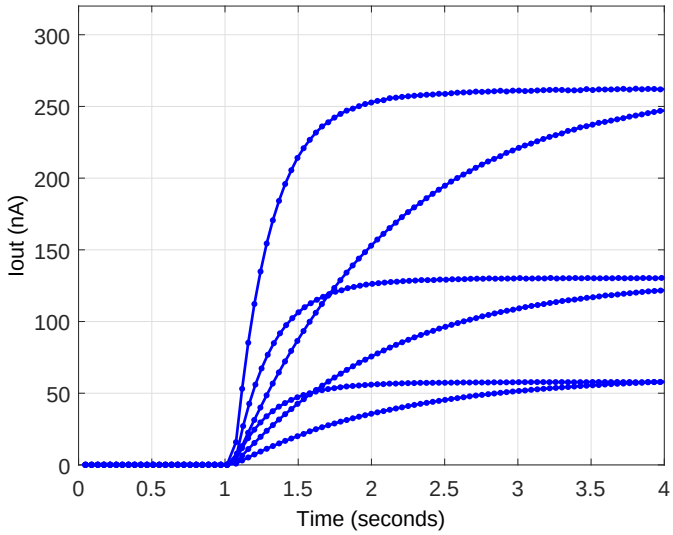
Intrinsic noise in cells, also known as molecular noise, arises from the probabilistic arrivals and collisions of molecules in biochemical reactions. As a result, genetically identical cells respond differently to the same environmental conditions. This cell-to-cell variability can be simulated by solving a set of equations that determine the probability that a system with a given initial condition will have each of the discrete states of molecular populations at a given time. This equation is known as the "chemical master equation" [85], which is written as

$$\frac{\partial P(\mathbf{x}, t|\mathbf{x}_0, t_0)}{\partial t} = \sum_{j=1}^{M}[a_j(\mathbf{x} - \mathbf{v}_j)P(\mathbf{x} - \mathbf{v}_j, t|\mathbf{x}_0, t_0) - a_j(\mathbf{x})P(\mathbf{x}, t|\mathbf{x}_0, t_0)] \tag{2.29}$$

where $P(\mathbf{x}, t|\mathbf{x}_0, t_0)$ is the probability that the system will have a state $\mathbf{x}$ at time t, given that its state at time t_0 is $\mathbf{x}_0$, and $a_j(\mathbf{x})$ is the propensity function for each



(a)



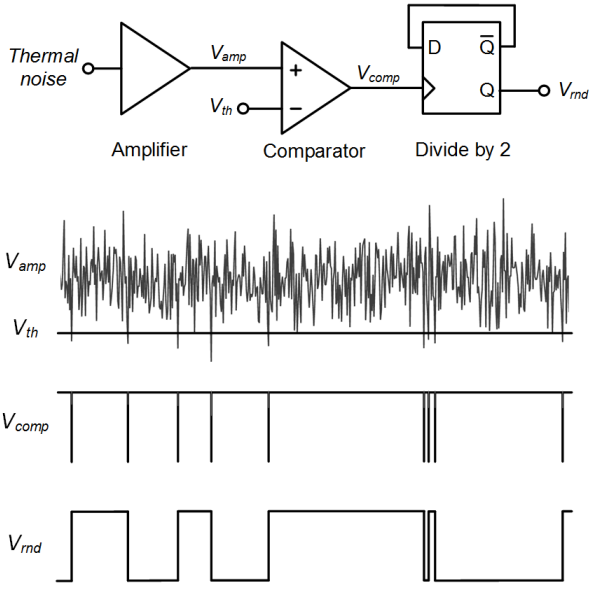
(b)

Figure 2-14: (a) MATLAB and (b) chip data over time for the gain & time constant block, for three gains (0.5, 1, and 2) and two time constants (0.25s and 1s). The step function shown in (a) is the input to the block. Chip parameters: $C = 1$ μF and $(I_A, I_B) = (25, 12.5)$, $(25, 25)$, $(25, 50)$, $(100, 50)$, $(100, 100)$, and $(100, 200)$ (units are nA).

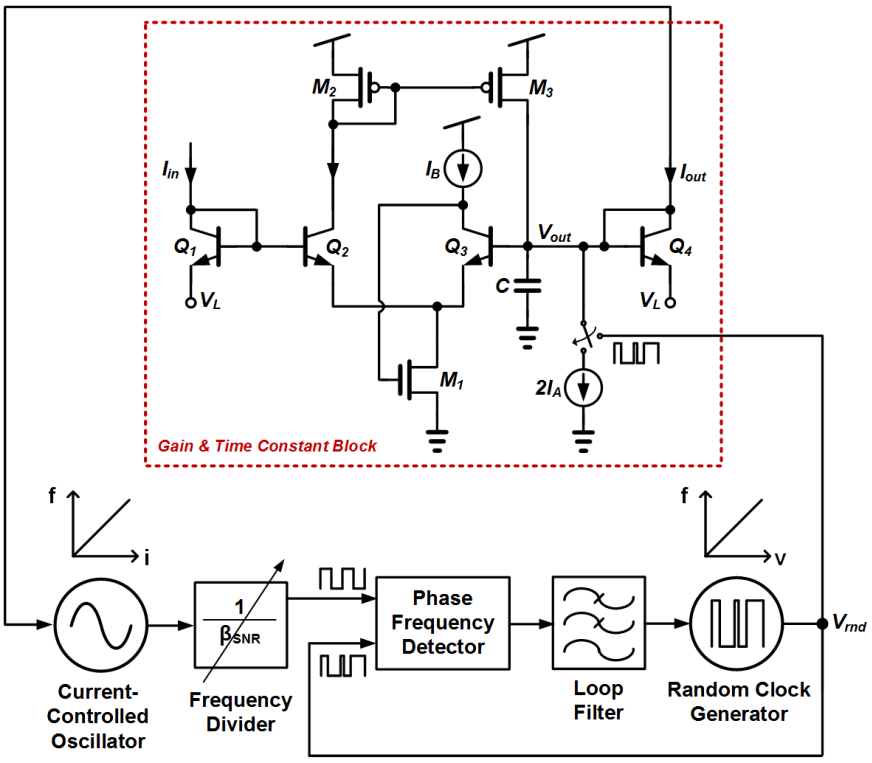
reaction [42]. However, since the above equation has to be written for each of the possible states, solving the chemical master equation easily becomes intractable as the number of state variables increases. Thus, instead of computing the whole probability distribution of a system, we can create individual instantiations of stochastic time trajectories of molecular populations using a Monte Carlo method called the Gillespie algorithm. In this method, randomness is embedded in trajectories themselves. That is, each instantiation generates different stochastic trajectories, and statistical characteristics such as mean and variance at a given time can be obtained from the result of many instantiations.

The original Gillespie algorithm is considered "exact" since it is derived from the same fundamental premise as the chemical master equation, without any approximation [42]: the probability that a reaction with a propensity function $a_j(\mathbf{x})$ will occur in some infinitesimal time interval dt is $a_j(\mathbf{x})dt$. The resulting stochasticity in biochemical reactions is well modeled by the Poisson shot noise of electronic current in subthreshold transistors with electron copy number analogous to molecular copy number [116]. Important stochastic characteristics including the "burst factor" observed in genetic networks [95] are faithfully reproduced in cytomorphic circuits [78,110,116] as an effective current gain. In the gain & time constant block presented in Section 2.2.5, the signal-to-noise ratio (SNR) at the output (I_{out}) is proportional to the size of the capacitor C and the current level; by adjusting the capacitor or current, any desired SNR can be achieved [116]. However, for SNR's below 15–25 dB (which is the case when the number of molecules is relatively small) capacitor sizes and the current levels can be small in electronics deteriorating the reliability and controllability of the noise level [78, 116]. A noise generator which can create artificially high levels of noise is therefore desirable. The basic idea is to probabilistically generate reaction events according to the rate constants and the number of reactant molecules for each reaction, which is analogous to the fundamental premise of the chemical master equation stated above. This enables simulation of stochasticity that is inherent in any reactions, e.g., plus and minus reactions in equations (2.1) and (2.28).

Figure 2-15 depicts the operating mechanism of the noise generator we created on



(a)

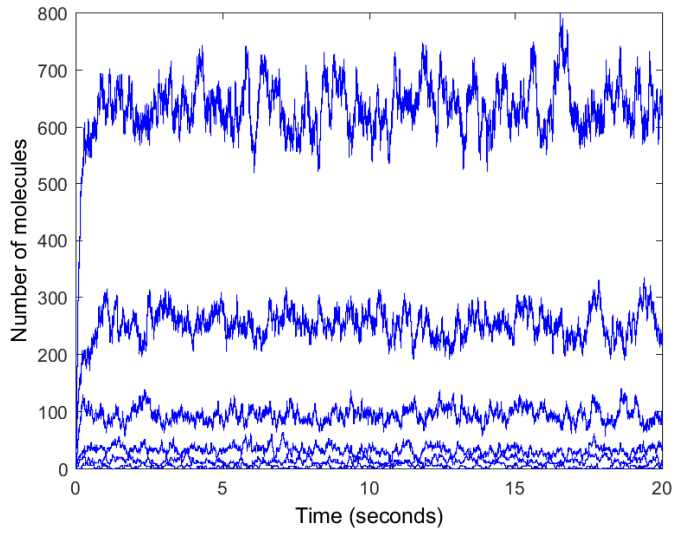


(b)

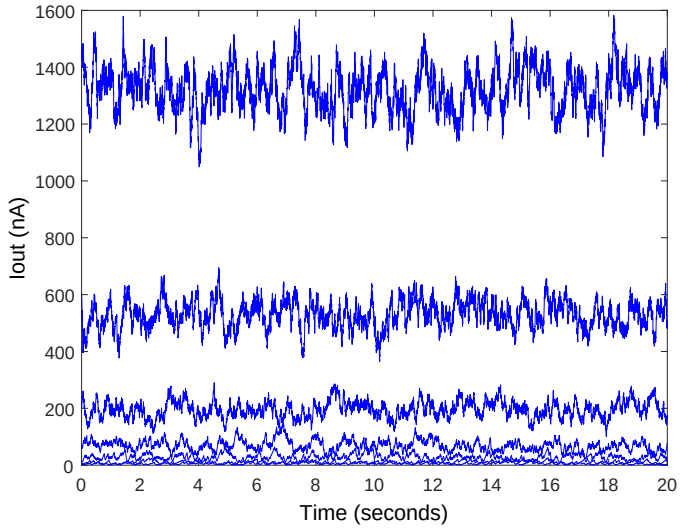
Figure 2-15: Operating mechanism of the noise generator [63]. (a) A "random" clock is generated by using a thermal-noise amplifier, a comparator, and a divide-by-2 circuit. (b) A current-controlled oscillator, a frequency divider, and a frequency-locked loop operate to regulate the mean frequency of the random clock, which is used to turn on and off I_A of the the gain & time constant block.

the chip [63], which is similar to that described in [116] except that it uses amplified analog thermal noise instead of a pseudo-random number generator. As shown in Figure 2-15(a), our circuit amplifies analog thermal noise that inherently exists in any transistor and compares it with a threshold voltage (V_{th}). Then, the comparator output (V_{comp}) goes through a divide-by-2 circuit to create a "random" clock (V_{rnd}) which exhibits pure Poisson characteristics with a mean duty cycle of 0.5. As illustrated in Figure 2-15(b), this random clock is used to turn on and off I_A of the the gain & time constant block to generate noise, via charging and discharging the capacitor node [78, 116]. A current-controlled oscillator (CCO) is used to produce a reference clock signal whose frequency is proportional to the output current I_{out}. The feedback loop around the phase frequency detector functions such that the mean frequency of the random clock is equal to that of the reference clock, by automatically finding V_{th} which produces the desired frequency. Thus, the mean frequency of the random clock is proportional to I_{out}. In addition, the mean frequency is inversely proportional to the gain factor β_{SNR}. By increasing β_{SNR}, artificially generated shot noise is increased, which mimics the high levels of noise seen in biology [78,116]. [116] provides a mathematical explanation of how the charge on the electron is effectively increased by this mechanism to amplify inherent shot noise in transistors.

Figure 2-16(a) and 2-16(b) show noise generated in MATLAB using a Gillespie algorithm for molecular production and degradation [40] and using the noise generator in the chip, respectively, for relatively low numbers of molecules. In this simulation, we arranged for 1 nA of I_{out} to correspond to approximately two molecules (per cell). The comparison between the SNR obtained from the MATLAB data and the chip data is shown in Figure 2-17. It can be seen that the artificial noise generator implemented in the chip generates noise that is very similar to that from a Gillespie algorithm, both visually and quantitatively. The good agreement with the Gillespie algorithm, a widely used method to model biological noise, is in accord with noise measurements and theories of Poisson noise in transistors and in cells [95,110,113,116]. Such work allows us to achieve fast-and-accurate stochastic Gillespie simulations on custom analog chips. Thus we can enable improvements over methods such as tau



(a)



(b)

Figure 2-16: Noise generated by (a) a MATLAB simulation using the Gillespie algorithm and (b) the noise generator block in the chip, for relatively low numbers of molecules. 1 nA of I_{out} is mapped to correspond to approximately two molecules.

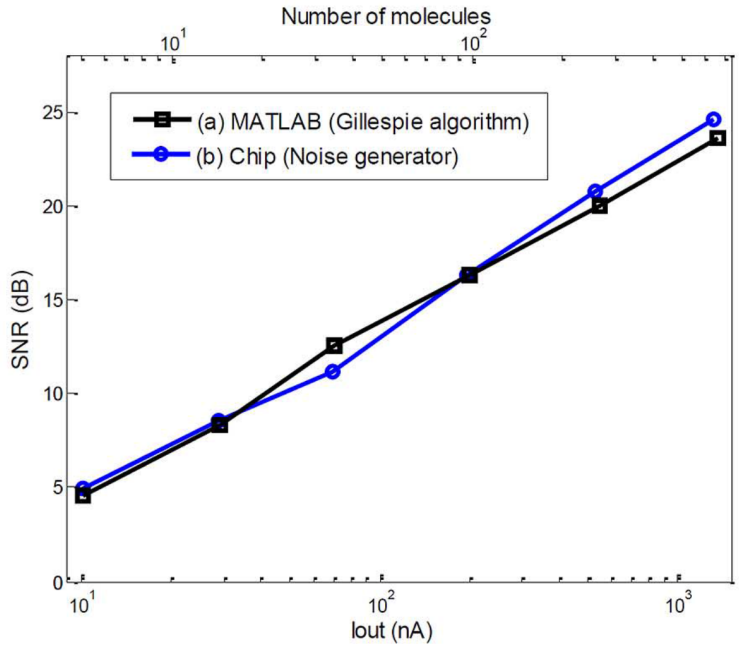


Figure 2-17: Comparison of SNR obtained from the MATLAB data and the chip data.

leaping [41], Langevin-noise addition, or limited-moment simulations of actual Fokker-Planck equations [29, 99] that necessarily require accuracy to be compromised for speed on general-purpose digital computers.

2.2.7 DAC and ADC

The analog circuits inside the cytomorphic chip primarily use currents to represent variables and parameters. On the other hand, chip-to-processor and chip-to-computer communication is done digitally. It is thus important to have a scheme to convert between analog currents and digital bits.

The gene block in Figure 2-1 includes 40 digital-to-analog converters (DACs), which requires us to design a power- and area-efficient DAC, while preserving reasonable precision, dynamic range, and linearity. To this end, we first took a stable bias current of 11.5 μA created by an on-chip current generator and then used a current splitter similar to that described in [20] to divide the current into halves, 13 times in succession; the 1^{st}, 3^{rd}, 5^{th}, 7^{th}, 9^{th}, 11^{th}, and 13^{th} bias voltage outputs served as global reference voltages for the whole chip.

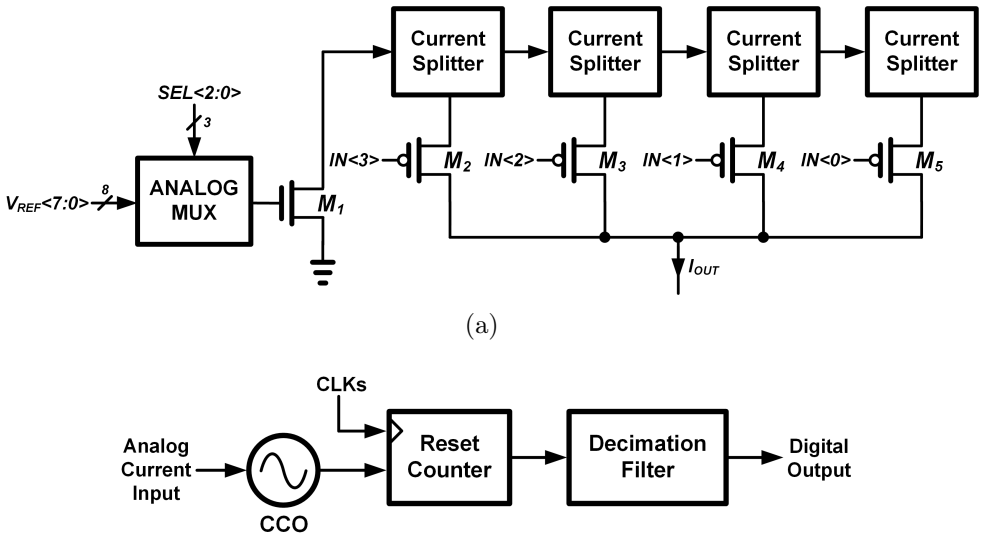


Figure 2-18: Block diagram of (a) the digital-to-analog converter (DAC) and (b) the analog-to-digital converter (ADC).

Figure 2-18(a) shows the block diagram of a single DAC. It selects one of the eight reference voltages (V_{REF}<7:0>) with a 3-bit digital input (SEL<2:0>) and an 8-to-1 analog multiplexer, applies it to the gate of M_1 to convert it back to a current, and additionally divides this current into halves, four times in succession. The sum of the four resulting currents, turned on or off by another 4-bit digital input (IN<3:0>), is the output current of the DAC. In summary, each DAC uses seven bits (stored in SRAM or shift registers) to determine its output current level—three bits to select a range in an exponential fashion and four bits to select a value in a linear fashion. This mechanism of coarse and fine selection enables an experimentally measured output dynamic range of the DAC from 29 pA to 21 μA; we only use a range of 100 dB from 100 pA to 10 μA in our circuits to ensure robustness to leakage and voltage-headroom effects. Furthermore, distributing reference voltages instead of currents allows us to save area and power which would otherwise be consumed by numerous current mirrors and distributing wires. We were careful about using sufficiently large transistors for converting between reference voltages and currents to mitigate the effect of mismatch, and about using thick-and-wide metal wires to minimize voltage drops.

The number of analog-to-digital converters (ADCs) depends on the maximum number of variables that need to be monitored or digitally processed off chip. For example, one may want to monitor 10 to 50 variables per chip. Hence, the top consideration when choosing an ADC topology is to minimize the number of output bits, so that the pin count devoted to this ADC on the chip is minimized. Therefore, we choose a current controlled oscillator-based (CCO-based) ADC shown in Figure 2-18(b), which is a first-order noise-shaping oversampling ADC [64]. The advantages of this ADC include compactness that comes from its simple architecture and its inherent noise shaping property. The current-to-frequency nonlinearity in the CCO, the major drawback of this architecture, can be easily compensated for by the use of a mapping table that is stored off chip. In our implementation, each CCO-based ADC operates at a sampling frequency of 5 MHz, and generates one-bit digital outputs that are averaged and down-sampled off chip to achieve analog-to-digital conversion.

2.3 Design Considerations in BiCMOS Cytomorphic Design

We chose to use a BiCMOS process technology to leverage good matching, high Early voltage, and most importantly, the nearly ideal wide-dynamic-range exponential current-voltage (I-V) characteristics of bipolar transistors. As explained in Section 1.4.3, translinear circuits utilize the exponential characteristics as an essential basis function, which both bipolar transistors and subthreshold MOS transistors have. However, in the case of MOS transistors, the saturation current follows the form of

$$i_{DS} = Ke^{\kappa_S V_{GS}/\phi_t} \tag{2.30}$$

in subthreshold operation (low levels of current) and

$$i_{DS} = K(V_{GS} - V_t)^2 \tag{2.31}$$

in above-threshold operation (high levels of current).

Unfortunately, switching from one operation regime to the other does not occur abruptly at a certain point. It rather gradually changes the behavior, and in the middle of the change (called the moderate inversion region) appears a mixed behavior between the two. This implies that the distortion due to the square term of equation (2.31) may cause a perceptible amount of error particularly around the upper edge of the subthreshold region. Typically, current levels between 100 pA to 1 μA are considered as subthreshold current levels. Below 100 pA is less reliable owing to leakage and noise, and above 1 μA is dominated by the above-threshold square-law relation. However, simulation results showed that even at 100 nA and 1 μA, a reasonably sized MOS transistor already yields as high as $\sim$26% and $\sim$78% of error, respectively. This suggests that the actual usable operating range of current is smaller.

On the contrary, bipolar transistors exhibit exponential I-V behavior over a very wide range of current. This is greatly helpful in increasing the dynamic range of variables, which in turn brings about several benefits (see Section 4.4.2). For example, we can exploit this wider current range by increasing all current levels ten times. Then, the effect of leakage current and parasitic poles is reduced by ten times, leading to less error and better stability. Thus, we implemented 100 dB dynamic range, from 100 pA to 10 μA, for most on-chip variables represented as current levels.

Extra care is needed to avoid potentially negative effects of bipolar transistors due to finite base currents effects and forward biasing of the base-collector junction. For example, we used the modified circuit shown in Figure 2-19 to solve practical problems that we encountered in the circuit of Figure 2-13: In this circuit, M_3 is inserted as a buffer (source follower) which supplies the base currents of Q_1 and Q_2 on behalf of I_{in}. This buffer is also useful in that it can rapidly charge the base node, thereby reducing parasitic-capacitance effects if any. The npn transistor Q_5 minimizes base-current effects in Q_3 and Q_4 without any body-effect voltage losses, while the PMOS transistor M_4 (whose source is tied to the well) minimizes base-current losses in Q_5 itself, while drawing no current from the V_C node. The overall cascaded buffer

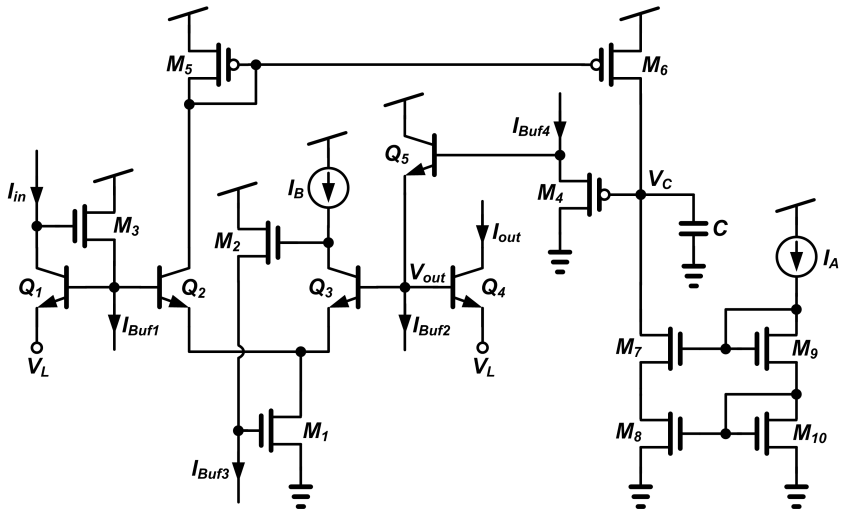


Figure 2-19: Modified LPF to solve practical problems in the circuit of Figure 2-13.

formed by Q_5 and M_4 also keeps V_C at a good DC operating point, which is necessary for the current mirror coupled to V_C to function in saturation. The cascode current mirror formed by M_7–M_{10} changes the direction of I_A since every current on chip is assumed to be supplied as a source current. Finally, M_2 serves as a buffer that enables Q_3 to always operate in its forward active region. The techniques described in this paragraph were also applied to all current-mode circuits in the chip, e.g., multipliers and integrators shown in Figure 2-4.

2.4 Simulation of Synthetic Genetic Circuits

To show the potential of our chip as a simulation tool for quantitatively modeling bio-molecular circuits, we emulated the models of several well-known synthetic genetic networks. To do so, we composed and programmed the basis-function circuits described in Section 2.2 to architect the network. This section illustrates how the simulations of three bio-molecular networks—the repressilator [26], a feed-forward loop network [25], and a delay-induced oscillator [82]—can be implemented with the chip.

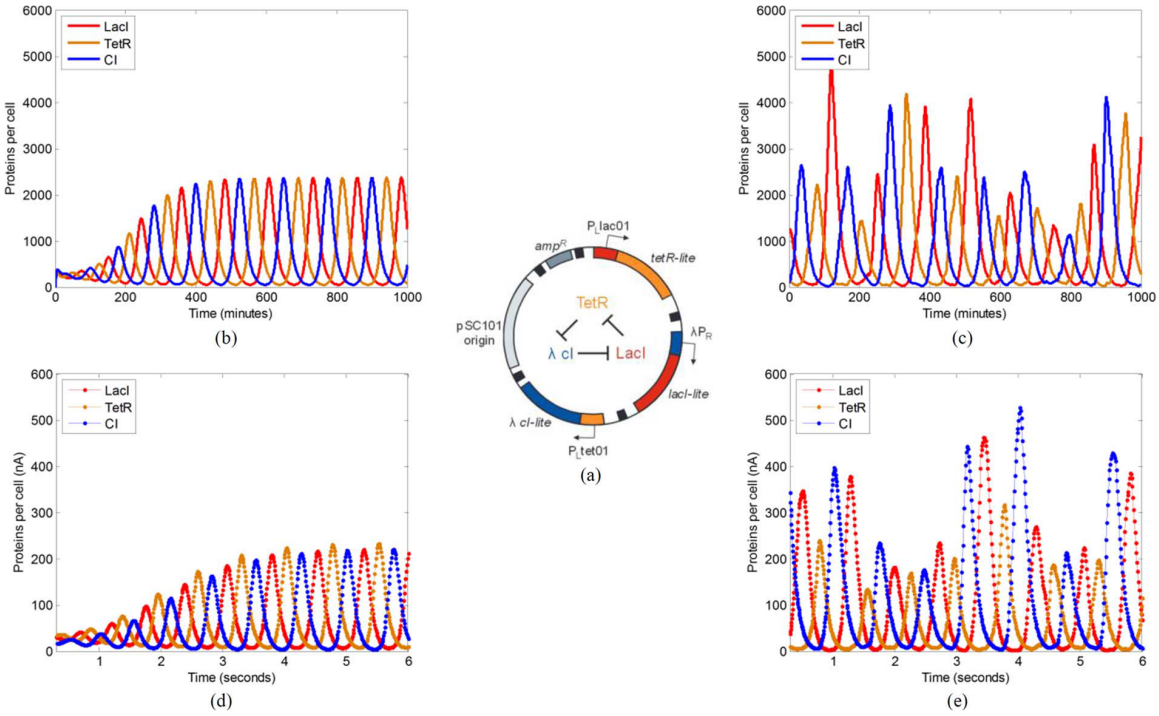


Figure 2-20: (a) Repressilator circuit [26]. (b) Deterministic and (c) stochastic simulation results of MATLAB, using the mathematical model provided in [26]. (d) Deterministic and (e) stochastic simulation results of the chip, which is programmed using the parameters in Table 2.1. (a) Reprinted by permission from Macmillan Publishers Ltd: [*Nature*] (M. B. Elowitz and S. Leibler, "A synthetic oscillatory network of transcriptional regulators," *Nature*, vol. 403, no. 6767, pp. 335338, Jan. 2000), Copyright 2000.

2.4.1 Repressilator

The repressilator network shown in Figure 2-20(a) is a circuit that helped pioneer the field of synthetic biology [26]. It creates a self-sustaining oscillation by using three repressors in a ring form, similar to a ring oscillator made with three cascaded inverters in electronic systems.

First, from the two dimensionless equations and parameters given in Box 1 of [26], we derived the original differential equations that determine the dynamic behavior of mRNA and protein concentrations of each repressor species:

$$\frac{dm_i}{dt} = -5.8 \times 10^{-3} m_i + \frac{0.5}{1 + \left(\frac{p_j}{40}\right)^2} + 5 \times 10^{-4} \tag{2.32}$$

$$\frac{dp_i}{dt} = -1.2 \times 10^{-3} p_i + 116 \times 10^{-3} m_i. \tag{2.33}$$

Next, scale factors for time and concentration scaling were selected. The time constant for transcription factor-DNA binding is ∼1.25 ms in the chip (when $C = 50$ pF and $I_{kr} = 1$ nA are used for the MM_Basic blocks in the ITD circuit), and the mRNA time constant is assumed to be 20 times bigger (25 ms) such that the former dynamics can be seen as nearly instantaneous. Then, $1/(5.8 \times 10^{-3}) = 172$ s of biological time is mapped into 25 ms of electronic time, corresponding to approximately 7000x speedup. Concentration scaling was performed such that 10 monomers of protein or mRNA were mapped into 1 nA of current in the chip. Based on these scaling parameters, all other biological parameters were scaled to electronic chip parameters, as summarized in Table 2.1.

After programming these parameters and the connectivity of the circuit into the SRAM and shift registers of the chip, we simulated the network on the chip. Figure 2-20(b) and 2-20(c) show the deterministic and stochastic simulation results of MATLAB, respectively, using the biological values given in Table 2.1; Figure 2-20(d) and 2-20(e) show the deterministic and stochastic simulations results of the chip, respectively, using the chip parameters given in Table 2.1. For the stochastic simulation, the noise generator described in Section 2.2.6 was turned on to produce artificial noise

Table 2.1: Parameter Mappings for Repressilator Simulation

Parameter	Biological Value		Chip Value
mRNA degradation rate	5.8×10^{-3} s^{-1}	→	40 s^{-1}
Protein degradation rate	1.2×10^{-3} s^{-1}	→	8 s^{-1}
Transcription rate (fully induced)	0.5 transcripts/s	→	346 nA/s
Transcription rate (repressed)	0.5×10^{-4} transcripts/s	→	346 pA/s
Translation rate	116×10^{-3} proteins/(mRNA·s)	→	800 s^{-1}
Dissociation constant	40 proteins	→	4 nA
Hill coefficient	2	→	2

Block	Chip Parameters
ITD – Integrator	I_{plus} = mRNA output of adjacent repressor, I_{r1} = 200 nA, I_{r2} = 200 nA, C = 1 µF
ITD – MM_Static	I_{Stot1} = 0, I_{Etot1} = TF_{tot}, I_{KD1} = 10 µA, $Hill1$ = 1
ITD – MM_Basic (upper)	I_{Stot2} = TF_{bnd}, I_{Etot2} = 1 µA, I_{KD2} = 10 µA, I_{kr} = 1 nA, C = 50 pF, $Hill2$ = 1
ITD – MM_Basic (lower)	I_{Stot3} = TF_{free}, I_{Etot3} = I_{Efree} of MM_Basic (upper), I_{KD3} = 4 nA, I_{kr} = 1 nA, C = 50 pF, $Hill3$ = 2
Analogic DAC	β_{00} = 865 nA, β_{20} = 865 pA, other β_{ij} = 0
Gain & time constant	I_A = 1 µA, I_B = 1 µA, C = 1 µF

in each mRNA concentration. It can be seen that the waveforms from MATLAB and the chip are highly correlated, with the scale factors close to those described in the previous paragraph. As opposed to software simulations, running stochastic simulations does not increase the simulation time of our highly parallel circuits.

2.4.2 Feed-Forward Loop Network

The feed-forward loop network shown in Figure 2-21(a) has two repressors, LacI and TetR, that repress the expression of yEGFP, and TetR also represses the production of LacI [25]. The inputs of the network that regulate yEGFP expression are ATc and IPTG inducers. Experiments were conducted with three different TetR-regulated promoters to study the effect of promoter strength.

Just as in the repressilator simulation of Section 2.4.1, concentration scale factors and chip parameters were determined from mathematical models and parameters described in the supplementary material of [25]. 1 ng/ml of ATc, 1 μM of IPTG, 1 AU (arbitrary unit for fluorescence level) of yEGFP, and 1 nM of repressor protein concentration were mapped to 1 nA of current. Each biological model that represented inducer-repressor binding or repressor-DNA binding was approximated on the chip using a dissociation constant and a Hill coefficient. In addition, to prevent conditions where LacI tetramer concentration becomes excessively high due to the $[LacI]^4$ term in the mathematical model, we assumed an upper concentration bound of 1 μM (which is consistent with the typical values given in [92]). To accomplish these goals, MATLAB read the value of LacI monomer concentration from a first gene block on our chip, processed it with tetramerization and concentration-limiting functions, and fed the result back to the chip as the I_{Etot} input of the MM_Static portion of a second gene block. This example illustrates how our chip can be synergistically used with off-chip tools such as MATLAB or an FPGA that can extend its capabilities via traditional software if needed. Mathematical models yield better fits of biological data if concentration-limiting functions are not applied, but only at unphysically high values of LacI that are not implementable either in biology or on our chips over a dynamic range of even 100 dB. Thus, our chip automatically enables discovery of

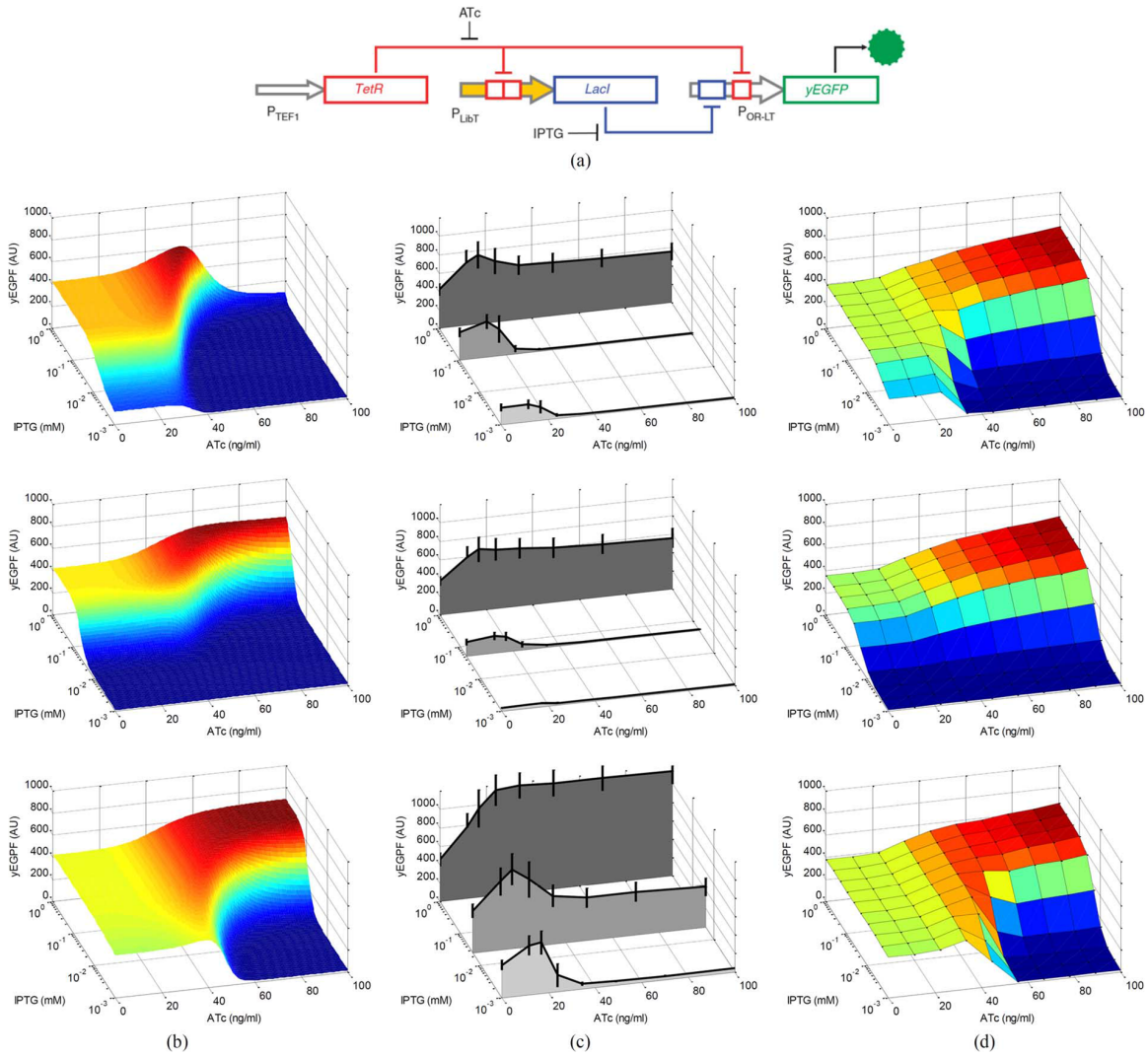


Figure 2-21: (a) Feed-forward loop network [25]. (b) MATLAB simulation results of the mathematical model and (c) experimental data from the feed-forward loop network constructed in *S. cerevisiae*, for three different promoters, TX, T8, and T18 [25]. (d) Simulation results of the chip; the tetramerization and concentration-limiting functions were implemented in MATLAB using molecular data packets from the chip. (a)-(c) Reprinted by permission from Macmillan Publishers Ltd: [*Nature Biotechnology*] (T. Ellis, X. Wang, and J. J. Collins, "Diversity-based, model-guided construction of synthetic gene networks with predicted functions," *Nature Biotechnology*, vol. 27, no. 5, pp. 465471, May 2009.), Copyright 2009.

unphysical parameters in mathematical models.

Figure 2-21(b)–(d) show MATLAB simulation results from a mathematical model, experimental results from the network constructed in *S. cerevisiae*, and chip simulation results. In all cases, where concentrations of molecules were at physically plausible levels, our chip produced results that matched the biological data at least as well as the published mathematical model (the average percentage error for six cases was 64% in the published mathematical model [25] versus 62% in our chips). In one case where concentration limiting functions were applied, the chip produced lower average percentage error than the mathematical model (24% versus 34% for the top row of Figure 2-21(b)–(d) at an IPTG concentration of 1 mM). In other cases where the concentration limiting functions were applied, both the chip and mathematical models performed poorly (errors that exceeded 200% in both cases).

2.4.3 Delay-Induced Oscillator

A pure delay is hard to implement efficiently with analog continuous-time systems since it requires an infinite number of state variables or a Pade approximant circuit. However, a pure delay is extremely easy to implement with digital systems. Although our system exploits analog and stochastic computation for its computationally intensive portions, our chips communicate digitally with external devices via on-chip ADCs and DACs. Thus, we have the flexibility to add digital basis functions like delays to supplement our on-chip analog basis functions. To demonstrate such flexibility, we simulated a delay-induced oscillator shown in Figure 2-22(a), where the time delay in the negative feedback loop due to transcription, translation, protein folding, and multimerization causes oscillations in the autorepression network [82]. The model in [82] is described by

$$\frac{dr}{dt} = \frac{\alpha}{1 + \left(\frac{r_\tau}{C_0}\right)^2} - \frac{\gamma_r}{1 + \frac{r}{R_0}} - \beta r \tag{2.34}$$

where r is the number of repressor molecules, r_τ is a delayed version of r (i.e., $r_\tau(t) = r(t - \tau)$), τ is the delay time, α is the production rate of the repressor, C_0 is the dissociation constant for repressor-DNA binding, γ_r is the maximum rate

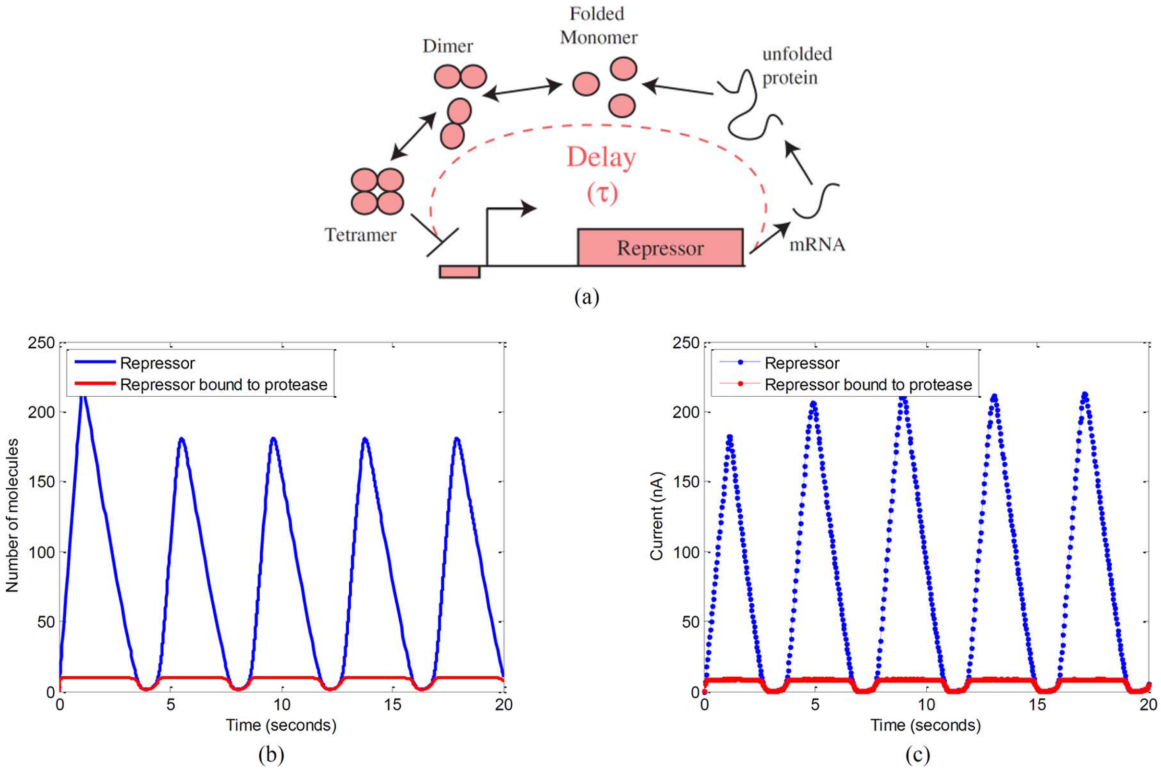


Figure 2-22: (a) Delay-Induced Oscillator Network [82]. (b) MATLAB simulation results of the mathematical model in equation (2.34) and (c) chip simulation results. (a) Reprinted figure with permission from [W. Mather, M. Bennett, J. Hasty, and L. Tsimring, "Delay-induced degrade-and-fire oscillations in small genetic circuits," *Phys. Rev. Lett.*, vol. 102, p. 068105, Feb 2009.] Copyright 2009 by the American Physical Society.

of degradation due to protease, R_0 is the dissociation constant for repressor-protease binding, and β is the rate of degradation due to dilution.

This network was simulated by the chip with the following configuration: One ITD block in Figure 2-9(b) provides an integrator to model production and degradation of repressor. The I_{r1} and I_{r2} inputs of the integrator correspond to γ_r and β, respectively. MM_Static is utilized to mimic the binding between repressor and protease and TF_{bnd} and $(TF_{tot} - TF_{bnd})$ are directed to the I_{r1} and I_{r2} sides of the integrator, respectively. The output of the integrator generates r, which is conveyed off chip as digital bits via the operation of the ADC. MATLAB reads the value of r, delays it, and programs the chip to send the delayed value to another ITD block, where repressor-DNA binding can be computed via an MM_Static circuit within the ITD. Finally, the TF_{free} output of the latter ITD block is sent to the $TF_{prodrate}$ input of the first ITD block to complete the auto-repressory feedback loop.

Figure 2-22(b) and 2-22(c) show the MATLAB and chip simulation results, respectively. The parameters used in MATLAB are $\tau = 1$ s, $\alpha = 300$, $C_0 = 10$, $\gamma_r = 80$, $R_0 = 1$, and $\beta = 0.1$ (same as Fig. 2 presented in [82]). Their corresponding chip parameters were also selected, with a one-to-one time mapping and 1 molecule-to-1 nA concentration mapping. Taking these scale factors into account, it can be seen that fairly similar results were produced by MATLAB and by our chip.

2.5 Specifications of the Chip

Table 2.2 summarizes the important performance characteristics of the chip. The values of the parameters suggest that our chip can model the wide range of parameters found in cells (listed in Table 1.1).

Digitally programmable analog systems have been reported [5, 6, 15, 24, 31, 38, 46, 67, 89, 96, 112, 115, 120, 121, 133–135], and have shown tremendous potential in medical and computing applications [116]. Work described in [109, 110, 116] contains an extensive discussion of the pros and cons of analog versus digital computation including issues relevant to speed, precision, power, flexibility, and programmability.

Purely custom digital implementations have also been used to model protein folding [123] or to model visual neuronal networks [122]. Our work suggests that prior work on deterministic analog computation could be efficiently extended to deterministic and stochastic simulations of living cells via our digitally programmable cytomorphic circuits.

2.6 Conclusion

In this chapter, we described a 0.35 μm BiCMOS silicon chip that quantitatively models fundamental molecular circuits via efficient log-domain cytomorphic transistor equivalents. These circuits include those for biochemical binding with automatic representation of non-modular and loading behavior, e.g., in cascade and fan-out topologies; for representing variable Hill-coefficient operation and cooperative binding; for representing inducer, transcription-factor, and DNA binding; for probabilistic gene transcription with analogic representations of log-linear and saturating operation; for gain, degradation, and dynamics of mRNA and protein variables in tran-

Table 2.2: Performance Characteristics of the Gene Chip

Parameter	**Value**
Technology	AMS 0.35 μm BiCMOS
Supply voltage	3.3 V
Dynamic range of variables (DAC output)	100 dB (100 pA–10 μA)
Number of DACs per gene block	40
Hill coefficient	1–4
Time constant of TF-DNA binding	1.25 ms–12.5 ms
Time constant of mRNA, protein	1.25 ms–12.5 s
Programming clock frequency	1 MHz
Programming time	0.3 ms
ADC sampling clock frequency	5 MHz
ADC readout time	2–20 ms
ADC input range	1 nA–10 μA
Number of ADCs per chip	12
Signal-to-noise ratio	4–35 dB
Power consumption	$<$ 10 mW
Chip size	2.6 mm $\times$ 3.9 mm

scription and translation; and, for faithfully representing biological noise via tunable stochastic transistor circuits. The use of on-chip DACs and ADCs enables multiple chips to interact via incoming and outgoing molecular digital data packets and thus create scalable biochemical reaction networks. The use of off-chip digital processors and on-chip digital memory enables programmable connectivity and parameter storage.

We showed that published static and dynamic MATLAB models of synthetic biological circuits including repressilators, feed-forward loops, and feedback oscillators are in excellent quantitative agreement with those from transistor circuits on the chip. Computationally intensive stochastic Gillespie simulations of molecular production are also reproduced by the chip and can be reliably tuned over the range of signal-to-noise ratios observed in biological cells.

Our work suggests that biological design, simulation, and analysis of circuits in living cells, which is very important in synthetic and systems biology, can greatly benefit from a cytomorphic transistor circuit approach. Such an approach captures the analog, digital, probabilistic, dynamic, stochastic, nonlinear, non-modular, and complex network and circuit behavior of biochemical reaction networks on compact transistor circuits very efficiently and naturally. Our chip's circuits function over more than five orders of magnitude of molecular concentration, which is more than adequate for representing the dynamic range of any given protein, DNA, RNA, or small-molecule state variable in cells.

Chapter 3

The Protein Chip

Our second cytomorphic chip, namely the protein chip, is built to model protein-protein interaction networks within the cell. They play a vital role in various cellular processes such as metabolism, signal transduction, and transcriptional regulation. For example, in the p53 signaling model described in Section 3.5.1, a signal indicating DNA damage is passed along through a biochemical chain of events (i.e., protein signaling), which eventually regulates the activity of p53 and Mdm2 proteins. The p53 protein acts as a transcription factor to activate the transcription of Mdm2, and the Mdm2 protein product in turn enhances the degradation of p53, forming a negative feedback loop. The latter interaction occurs on the protein level and thus on a much faster timescale than the former transcriptional interaction. This enhances the stability of the loop. Because of such an advantage, this type of feedback is commonly found in cellular networks [3].

Since protein networks exhibit complex dynamics and various forms of network topologies (e.g., cascade, feed-forward and feedback loop, fan-in, fan-out, and loading), an important design requirement of the protein chip is to have versatile computational units which can be flexibly programmed in terms of parameters, network connectivity, and initial conditions. Thus, we design the "protein block", a universal analog computational unit of the chip featuring several digitally programmable components. It is explained in Section 3.1 along with the overview of the protein chip. Section 3.3 shows how to configure multiple copies of the blocks to model diverse

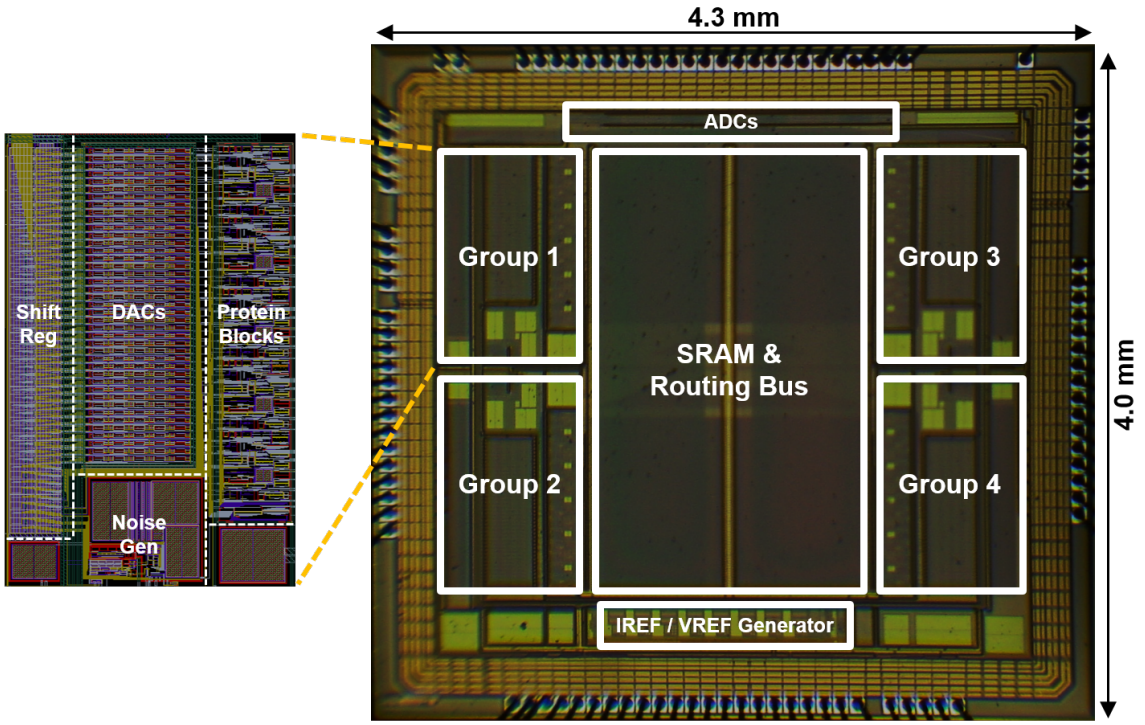


Figure 3-1: Die micrograph of the 4.3 mm × 4.0 mm cytomorphic chip fabricated in an AMS 0.35 μm BiCMOS process. The left inset is a layout screen capture of one of the four identical protein block groups (2x magnification).

types of reactions and network topologies. Section 3.5 describes how the protein chip can be used to simulate the computational models of biological networks, by providing simulation examples for the p53 signaling pathway and the glycolysis pathway. Section 3.6 summarizes the specifications of the chip, and Section 3.7 concludes the chapter.

3.1 Architecture of the Protein Chip

The primary purpose of the protein chip is to model and simulate the dynamics of general chemical reactions among various proteins and small molecules in biological systems. The current version of the chip is capable of modeling the mass-action kinetics of up to 60 unidirectional reactions with 60 state variables: 20 forward and 20 reverse reactions, both of which can be of zeroth, first, or second order, and 20 degradation reactions of first order. The transistor circuits to model these reactions

constitute 20 universal bidirectional reaction blocks, i.e., the protein blocks, which are the main computational units. Their details will be explained in the following sections. In the next chapter, we demonstrate how several protein and gene chips can be composed together with FPGAs to form scalably large digitally programmable biochemical reaction networks.

Figure 3-1 shows a die micrograph of the chip fabricated in an AMS 0.35 μm BiCMOS process, where essential components are labeled. The chip is divided into 4 identical groups, each of which contains 5 copies of protein blocks, a 350-bit shift register, 41 digital-to-analog converters (DACs), and a noise generator.

Within the chip, the protein blocks are connected to one another via a routing bus. The data to set the connectivity among the blocks is stored in SRAM, which is programmed by the operation of the shift registers fed with a bitstream created by the FPGA. The shift registers also hold various information including selection bits for switches, Hill coefficients, and the 7-bit numbers used by DACs to create analog currents. These electric currents serve as the primary means to represent variables and parameters of the protein block. When it is necessary to transmit certain variables off chip, ADCs convert them to digital bits and send to the output pins of the chip (1 bit per variable). The FPGA periodically reads and processes the ADC outputs, which may be transferred to other chips for computation or to the computer for creating real-time plots. Finally, the noise generator operates to artificially generate high levels of noise for stochastic simulations.

It is important to note that core computation takes place on the chip and in the continuous time and continuous signal domain, while off-chip communication and data processing take place in the discrete time and discrete signal domain. Furthermore, regardless of the number of variables, all on-chip and off-chip operations are performed simultaneously. This parallelization is the key to developing a system whose simulation time does not increase with the scale of the target network.

Detailed descriptions of some of the circuits, including the Hill circuit, the DAC, the ADC, and the noise generator can be found in Chapter 2.

3.2 The Protein Block

The protein block is a versatile computational unit capable of modeling a maximum of three reactions—a forward and a reverse reaction and a degradation reaction—with programmable mass action rate constants. Figure 3-2 shows the transistor schematic of the protein block. The block heavily utilizes current-mode circuits, and to make full use of them, they are implemented in a BiCMOS process technology which offer an exponential current-voltage relationship over a wide range of current levels. Basically, the block consists of two subtractors to model "loading" (see the following paragraphs), a Hill circuit to compute the forward rate with a Hill coefficient ($I_{Afree}(I_{Bfree}/I_{KDfw})^n$) (see Section 2.2.2), two multipliers to compute the reverse ($I_{Cfree}I_{Dfree}/I_{KDrv}$) and the degradation ($I_{Cfree}I_{ratC}/I_{One}$) rates, respectively, and an integrator to integrate these rates to create an analog current equivalence of molecular concentration.

This integrator circuit is a three-input differential current-mode integrator designed specifically for the protein block. According to the translinear principle applied to the three loops around 1) Q_1–Q_4, 2) Q_3–Q_6, and 3) Q_4 and Q_7–Q_9 [38], respectively, the time derivative of the output current is given by

$$\begin{aligned}\frac{dI_{Ctot}}{dt} &= \frac{I_{kr}}{C\phi_t}\cdot I_{fw} - \frac{I_{kr}}{C\phi_t}\cdot I_{rv} - \frac{I_{kdeg}}{C\phi_t}\cdot I_{deg} \\ &= \text{(forward rate) - (reverse rate) - (degradation rate)}\end{aligned} \tag{3.1}$$

where I_{fw} corresponds to $I_{Afree}(I_{Bfree}/I_{KDfw})^n$, I_{rv} to $I_{Cfree}I_{Dfree}/I_{KDrv}$, and I_{deg} to $I_{Cfree}I_{ratC}/I_{One}$. The transistors M_1, M_2, M_5, M_6, and M_8 serve as buffers (I_{Buf} = 1 μA), and another buffer composed of Q_5 and M_4 supplies the base current of Q_3 and Q_4 without causing any base-current loss from the capacitor node (V_C) or voltage error due to the body effect (see Section 2.3). CLK and $\overline{CLK}$ are digital random clock signals generated by the noise generator, which turn on and off the current charging the capacitor to induce fluctuations in the capacitor voltage, as described in the prior chapter or in [116].

Figure 3-3 is a block diagram of the circuit in Figure 3-2, showing its 7 inputs,

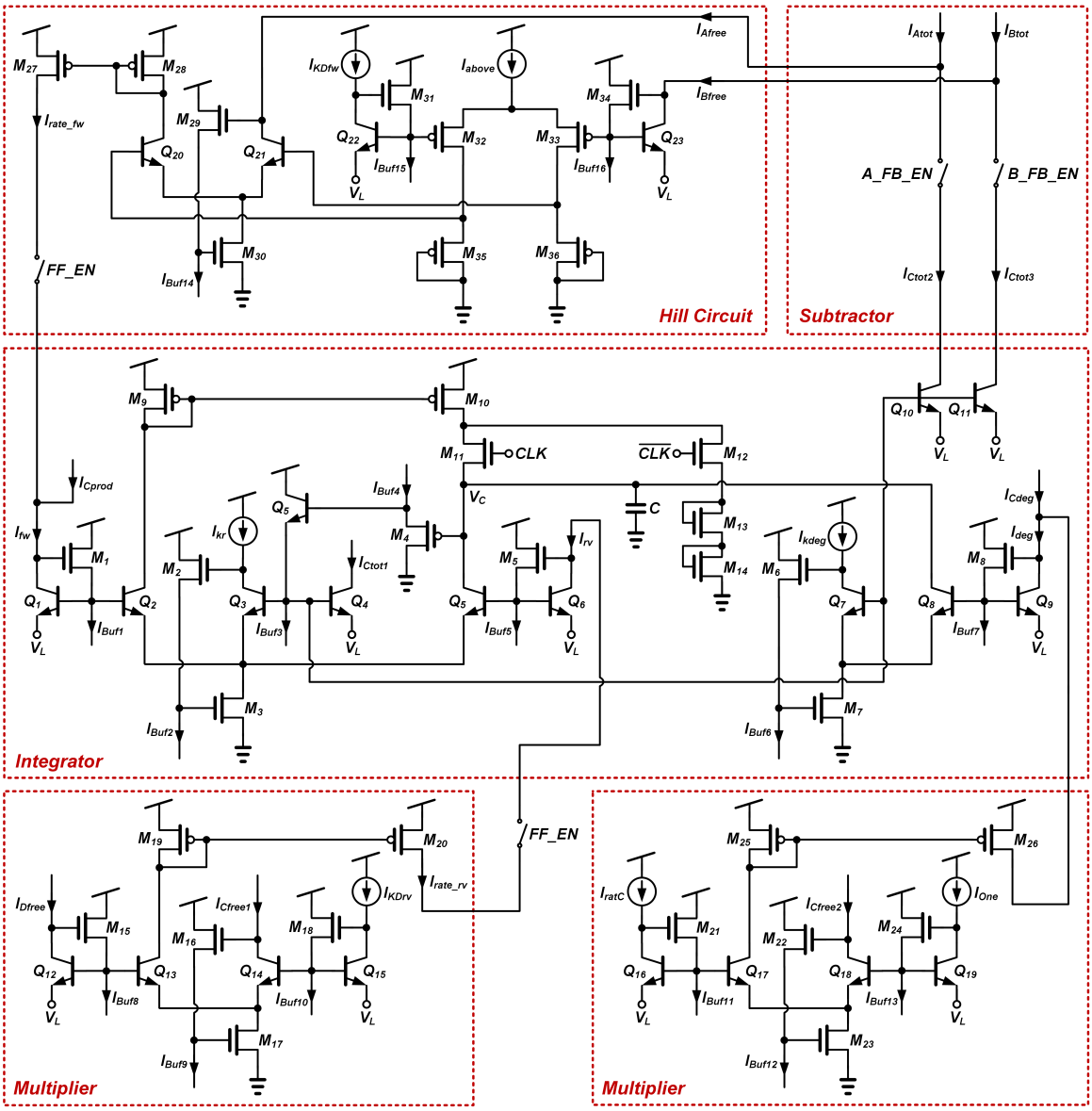


Figure 3-2: Transistor schematic of the protein block.

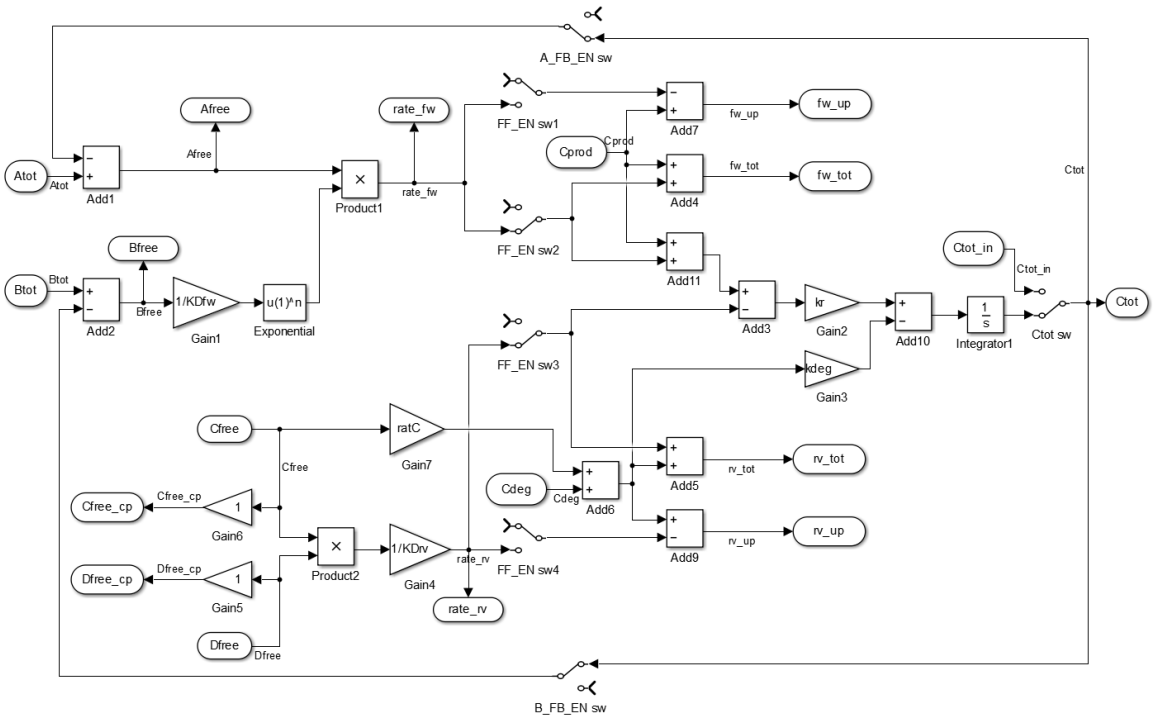


Figure 3-3: Block diagram of the protein block.

11 outputs, 6 parameters, and 3 switches (see Section 3.4 for details). Among them, here are the ones that are used most frequently: Atot / Btot / Cfree (inputs), Afree / Bfree / Ctot (outputs), and KDfw / kr (parameters). The rest are used for special occasions and for block-to-block connections. Note that some inputs, outputs, and switches, as well as current mirrors to create output variables are omitted in Figure 3-2 for simplicity. Note also that kr and kdeg in Figure 3-3 are mapped into the physical circuit parameters of $I_{kr}/C\phi_t$ and $I_{kdeg}/C\phi_t$, respectively. Our circuit has an additional degree of freedom in representing a D variable that interacts with C to effectively generate a reverse reaction current dependent on the concentrations of D and C. While this freedom is useful in the most general cases for composing arbitrary networks as we discuss later, for simplicity, we shall begin with a discussion of just A, B, and C in the discussion that follows.

In a typical configuration for a binding reaction, where Dfree/KDrv=1, ratC=kdeg=0,

and n=1, the model in Figure 3-3 solves

$$\frac{d[Ctot]}{dt} = kf[Afree][Bfree] - kr[Cfree] \tag{3.2}$$

$$[Afree] = [Atot] - [Ctot] \tag{3.3}$$

$$[Bfree] = [Btot] - [Ctot] \tag{3.4}$$

where kf=kr/KDfw. As seen from the above equations, the amount of the product Ctot is subtracted from the "total" amount of the reactants, Atot and Btot, to obtain the "free" amount of the reactants, Afree and Bfree. This models the effect of "loading" present in any binding reaction: Whenever two reactants A and B react to produce a product C, the number of A and B is decreased by the number of produced C. These two inherent feedback loops are explicitly represented in Figure 3-3.

Hence, as their names imply, Atot and Btot are "total" variables that include all downstream quantities (see Section 2.2.1). If Atot and Btot are constants and Ctot is not used in any downstream reaction (i.e., Cfree=Ctot), the equations above are simply equivalent to the following set of differential equations:

$$\frac{d[Cfree]}{dt} = kf[Afree][Bfree] - kr[Cfree] \tag{3.5}$$

$$\frac{d[Afree]}{dt} = -kf[Afree][Bfree] + kr[Cfree] \tag{3.6}$$

$$\frac{d[Bfree]}{dt} = -kf[Afree][Bfree] + kr[Cfree]. \tag{3.7}$$

As such, the use of these "total" variables and the subtraction mechanism allows us to compute at most three differential equations at the cost of one (since subtraction is essentially free with analog currents). Not only that, it also enables convergence of solutions by avoiding the divergence problem which may originate from circuit mismatches when the three equations are computed independently in three separate circuits, as in other programmable analog hardware to solve differential equations [15, 75, 79]. See Section 2.2.1 for further discussions.

Figure 3-4 shows the symbol of the protein block which displays the heart of

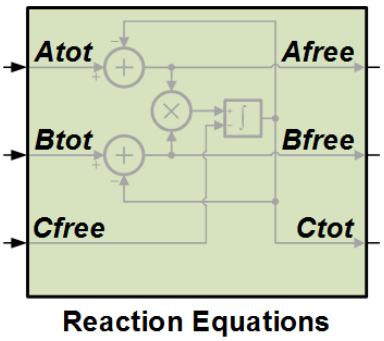


Figure 3-4: Block symbol of the protein block.

its computation and essential input and output ports. This symbol will be used throughout the rest of this article to describe the schemes to configure the protein blocks.

3.3 Protein Block Configurations for Various Network Topologies

This section demonstrates how connection among the ports and manipulation of switches can implement various types of reactions and network topologies found in chemistry. For explanatory purposes, configurations for simple networks are presented in Figures 3-5 and 3-6 as sample cases. In reality, the same principles can be used to construct complex reaction networks (e.g., a combination of multiple fan-out and fan-in with reversible reactions) in a scalable fashion.

Note that the distinction between "total" and "free" variables described in Section 3.2 calls for extra care in making connections between ports. For example, the degradation rate of a species is supposed to be proportional to its "free" amount; however, the Ctot output of a block corresponds to the "total" amount. Thus, if this Ctot is used as a reactant in a downstream block (as Atot or Btot), whatever is left (Afree or Bfree) is returned to the original block, as the Cfree input, and used to produce the degradation rate.

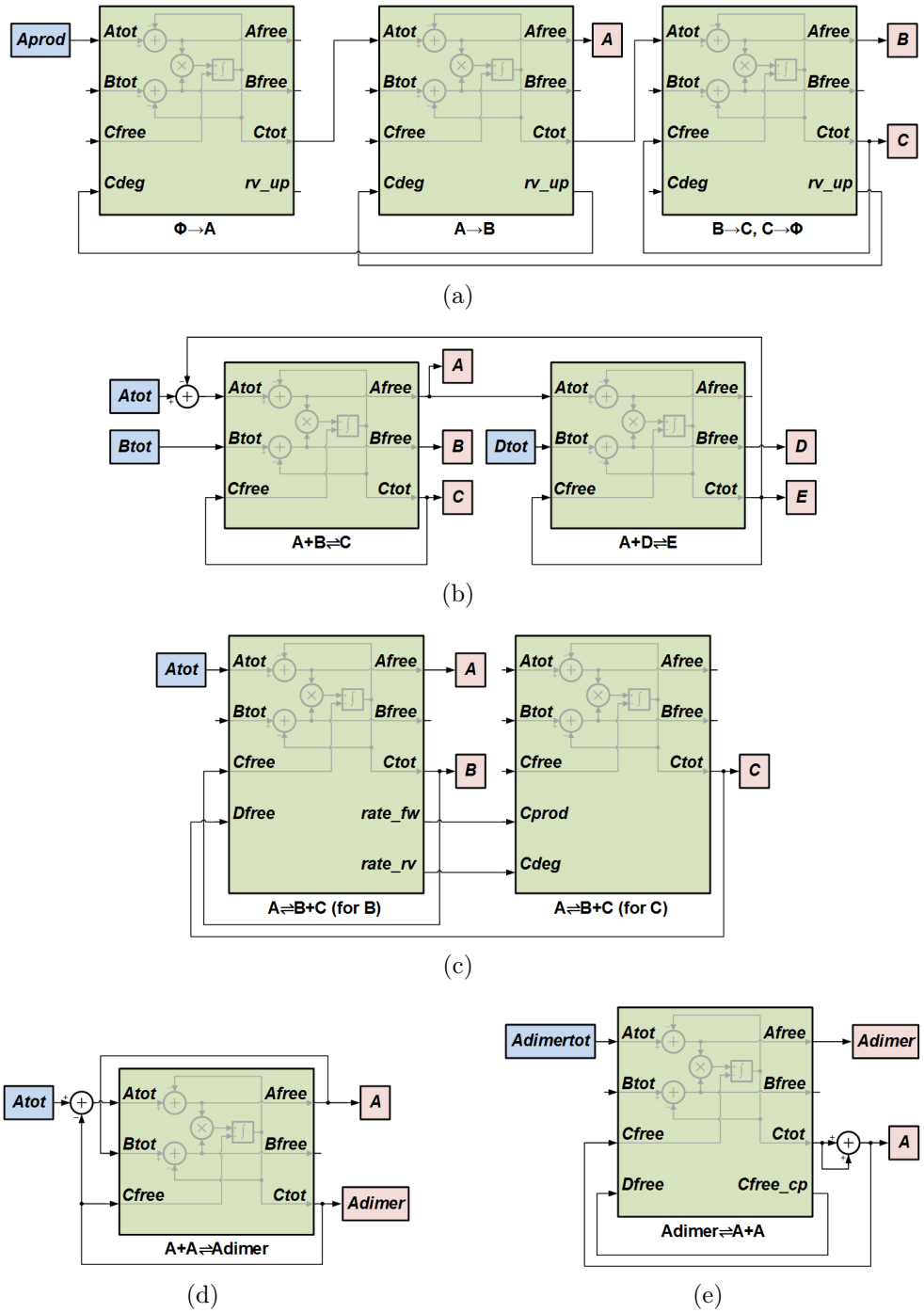
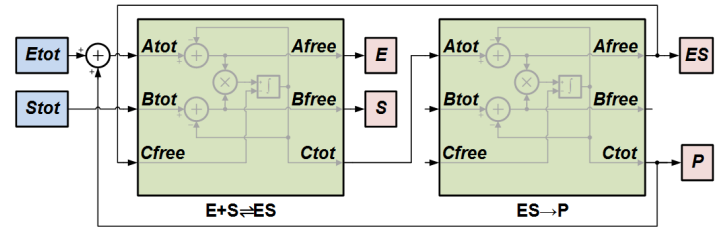
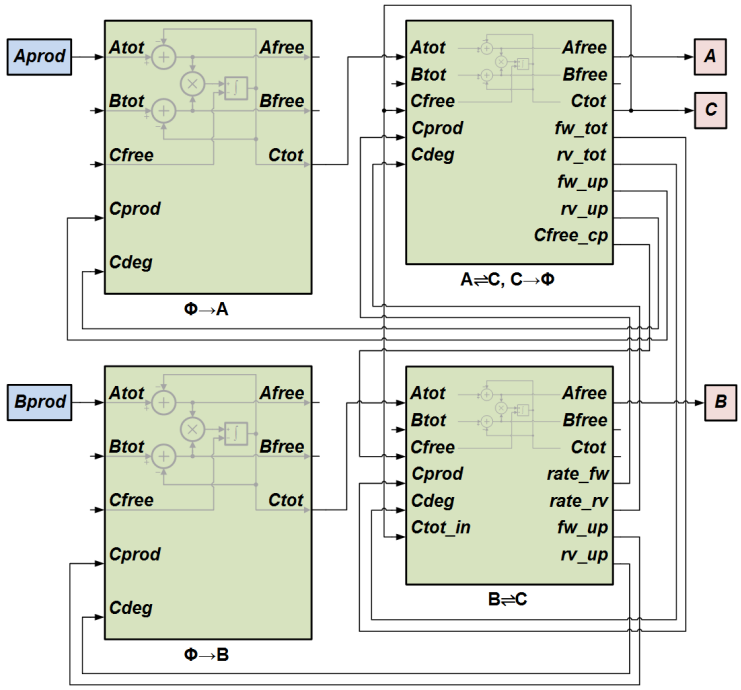


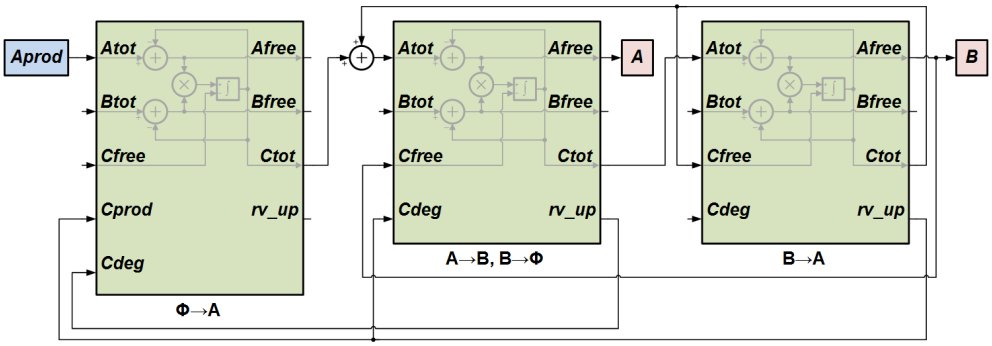
Figure 3-5: Configuration examples for various reactions and network topologies. (a) Cascade with degradation (∅→A, A→B→C, C→∅). (b) Fan-out (A+B⇌C, A+D⇌E). (c) Dissociation (A⇌B+C). (d) Dimerization (A+A⇌Adimer). (e) Monomerization (Adimer⇌A+A). Green, blue, and red blocks denote a protein block, an input variable, and an output variable, respectively.



(a)



(b)



(c)

Figure 3-6: Configuration examples for various reactions and network topologies (continued). (a) Michaelis-Menten reaction (E+S⇌ES→E+P). (b) Fan-in with degradation (∅→A, ∅→B, A⇌C, B⇌C, C→∅). (c) Loop (∅→A, A→B→A, B→∅). Green, blue, and red blocks denote a protein block, an input variable, and an output variable, respectively.

3.3.1 Cascade (Figure 3-5(a))

Cascade is a type of chemical process where a product of a reaction serves as a reactant of the following reaction. In our framework, it can be modeled as follows:

1. Insert the current block's Ctot to Atot (or Btot) of the next reaction block. In the current block, Afree=Atot–Ctot; Bfree=Btot–Ctot.

2. In case a reverse reaction or a degradation reaction exists in the current block, bring Afree (or Bfree) from the next block to Cfree of the current block. Repeat step 1 and 2 until the end of the cascade chain.

3. For the last block of the cascade chain, its Ctot is not used anywhere, such that Ctot and Cfree are identical as configured in the rightmost block of Figure 3-5(a).

Note that in the example cascade network in Figure 3-5(a) which models ∅→A, A→B→C, and C→∅, the leftmost block implements a zeroth-order reaction (∅→A). Thus, Atot should be a constant parameter, Btot/KDfw is set as 1, and both A_FB_EN and B_FB_EN should be switched off. The two blocks on the right each implement a first-order reaction (A→B and B→C). Thus, Atot is a variable, Btot/KDfw is 1, and only A_FB_EN is switched on.

3.3.2 Degradation (Figure 3-5(a))

If a block has a nonzero degradation rate, it decreases Ctot. Then, since Afree=Atot–Ctot, Afree effectively increases, although it should not. In reality, degradation of Ctot decreases the pool of its generating Atot variable, thus keeping Afree=Atot–Ctot constant. This means that degradation fluxes in a given variable should be back propagated to all total variables upstream of it that it is a part of, as in Figure 3-5(a). Therefore, the algorithm for network composition simply becomes:

1. In each stage, receive the degradation rate from its subsequent stage at the Cdeg port. In case of fan-out, there may be multiple rates coming.

2. Add the degradation rate of the current stage, whose rate constant is set by ratC and kdeg in Figure 3-3. The resulting "lumped" rate is at the rv_up output.

3. Send rv_up to the previous stage. For second-order reactions such as A+B→C and C→∅, the degradation rate of [C] should be propagated to both of the previous stages (i.e., the two blocks responsible for [A] and [B], respectively). To serve this purpose, two copies of rv_up are generated in each block.

Figure 3-7 shows the chip simulation results of a simple cascade reaction network, A→B→C→D→E. From Figure 3-7(a), It can be seen that all [A] become [E] over time, so that the final value of [E] is matching with the initial value of [A]. Figure 3-7(b) is the result when a degradation reaction E→∅ is added to the network. The value of [E] decreases as expected, while the remaining free variables remain the same.

3.3.3 Fan-out (Figure 3-5(b))

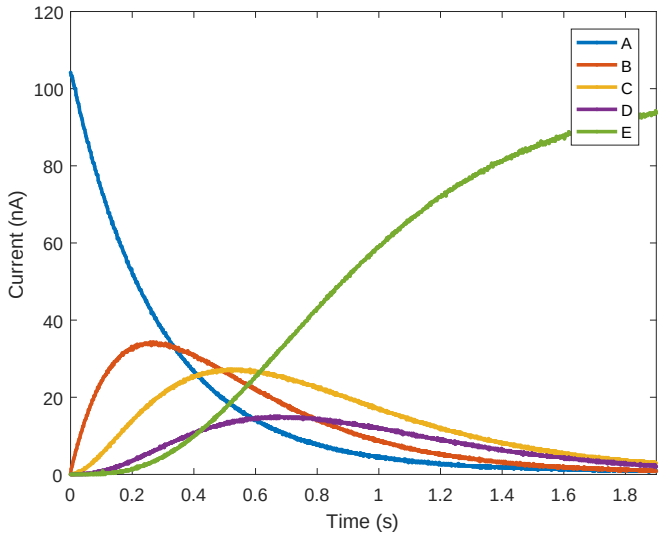
Fan-out is a network topology where one species serves as a reactant of multiple reactions. Assuming that [A] is such a species, the following is the configuration strategy for fan-out:

1. Designate one block as the main block where [Atot] comes in.

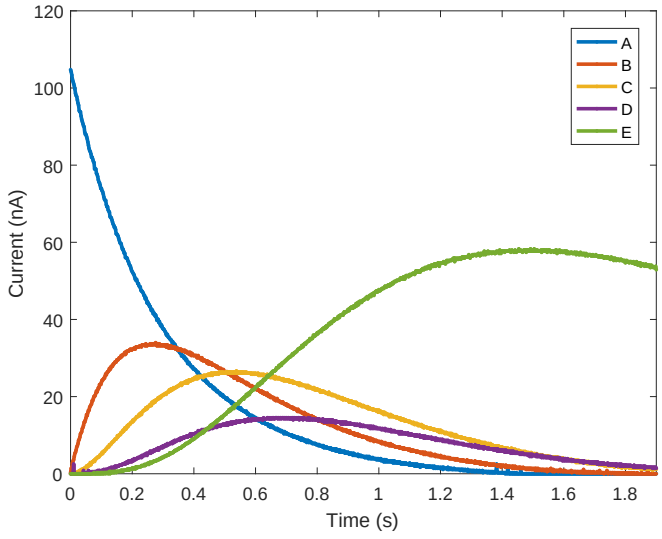
2. Take the Ctot outputs of all the blocks where [A] is used as a reactant ([C] and [E] in Figure 3-5(b)) and subtract them from [Atot] of the main block to compute [Afree]. Note that Ctot of the main block is subtracted inside the block by enabling the A_FB_EN switch, and thus not seen in Figure 3-5(b).

3. Send the resulting [Afree] to Atot (or Btot) of another block that uses [A] as a reactant. For this non-main block, disable the A_FB_EN (or B_FB_EN) switch, since the incoming variable [Afree] is already the desired "free" amount.

4. If there is a third block using [A] as a reactant, send the Afree output of the second block to the Atot (or Btot) port of the third block. Repeat this for all such blocks.



(a)



(b)

Figure 3-7: Chip simulation results for cascade with degradation. (a) A→B→C→D→E and (b) E→∅ added.

This method preserves scalability since it does not require additional subtractors or variable copiers (i.e., current mirrors) proportional the number of fan-out blocks. Note also that Dfree/KDrv should be set as 1 for both protein blocks in Figure 3-5(b) because the reverse reactions are first-order in this particular example.

3.3.4 Dissociation / Replacement (Figure 3-5(c))

This is how to configure when reactions have two products (e.g., dissociation and replacement reactions), assuming that [A] is a reactant and [B] and [C] are the two products:

1. If [B] and [C] do not degrade or have the same degradation rate constants, they will have identical values in terms of "total" variables. In this case, compute A⇌B with a block and use one copy of Ctot as [Btot] and another copy as [Ctot].

2. If [B] and [C] have different degradation rate constants, they need to be considered separately using two blocks. One block computes A⇌B+C to obtain [Btot] and sends its forward (rate_fw) and reverse (rate_rv) rates to the Cprod and Cdeg inputs of the other block, respectively, to obtain [Ctot]. This [Ctot] is sent to the Dfree input of the former block to produce the reverse reaction rate, i.e., kr·Cfree·Dfree/KDrv in Figure 3-3.

It should be pointed out that in terms of dealing with the "total" variable, [Atot] keeps track of only the [B] side and the [C] side is on its own, forming a separate branch. That is, [Afree]=[Atot]–[Btot] ([Ctot] is irrelevant) and only the degradation of [Btot] is propagated to [Atot].

3.3.5 Dimerization (Figure 3-5(d))

Dimerization is a special type of synthesis reaction which calls for special treatment. The key is to have the same species for the two reactants and consider the fact that the

production of one dimer consume two monomers (i.e., [Afree]=[Atot]–2[Adimertot] in Figure 3-5(d)).

1. First, [Atot] goes into the Atot port, and Ctot is subtracted from [Atot] via the routing bus. Besides, another subtraction occurs inside the block by enabling the A_FB_EN switch. Thus, Ctot (=[Adimer]) is subtracted twice from [Atot] to yield [Afree].

2. Take the Afree output and connect it to the Btot input. Disable B_FB_EN since math is done for [Afree] in step 1.

3. When [Adimer] degrades, [Atot] should degrade twice as fast. Hence, send both of the two rv_up outputs to the Cdeg input of the block which produces [Atot].

3.3.6 Monomerization (Figure 3-5(e))

Monomerization is the reverse process of dimerization, where one reactant produces two products of a kind. Thus, half the amount of the product should be subtracted from the reactant (i.e., [Adimerfree]=[Adimertot]–[A]/2 in Figure 3-5(e)) and Cfree and Dfree should receive the same variable. The trick to realize this is as below:

1. Add two of the Ctot outputs and treat it as [Atot]. Then, one Ctot corresponds to half of [Atot], which is subtracted from [Adimertot] inside the block to create the desired value of [Adimerfree].

2. Connect [Afree] (equal to [Atot] in Figure 3-5(e) since it is used nowhere else) to Cfree. In addition, link the Cfree_cp output to the Dfree input, so that both Cfree and Dfree take [Afree] as inputs.

3. When a degradation rate comes from the next stage, it becomes double due to the addition of the two Ctot outputs in step 1. To offset this effect, set kdeg of this stage and all upstream stages as half the value of that of the next stage. Then, half of the degradation rate of [Atot] propagates to [Adimertot] and all upstream "total" variables, as desired.

3.3.7 Michaelis-Menten Reaction (Figure 3-6(a))

Michaelis-Menten kinetics is one of the most well-known models of enzymatic dynamics. It forms a loop topology in terms of the enzyme, which may require complex connections. However, we describe a simpler way of implementing a Michaelis-Menten reaction network:

1. Use a block to model E+S⇌ES, a typical reversible synthesis reaction.

2. Use another block to model ES→P. Just like a cascade configuration, connect Ctot of the first block to Atot of the second block, and Afree of the second block to Cfree of the first block.

3. Send Ctot of the second block to add to the Atot input of the first block.

4. When [Ptot] degrades, propagate it to [EStot] and [Stot] but not to [Etot], by making connections between rv_up ports and Cdeg ports accordingly.

If [Ptot] does not degrade, the amount of [Ptot] produced is equal to that of [E] released with [P]. Thus, adding that [Ptot] to Atot of the first block would yield the right value of [Efree]. Furthermore, even when [Ptot] degrades, the configuration above leads to mathematically correct results, and here is why: The expressions derived from the network in Figure 3-6(a) are given by

$$[Efree] = [Etot] - [EStot] + [Ptot] \tag{3.8}$$

$$[Sfree] = [Stot] - [EStot] \tag{3.9}$$

$$[ESfree] = [EStot] - [Ptot]. \tag{3.10}$$

Since the degradation of [Ptot] propagates to [EStot] and [Stot], the amount of [ESfree] and [Sfree] are preserved as desired. [Efree], on the other hand, is also preserved because the change in –[EStot] and +[Ptot] cancel out and [Etot] does not change with the degradation of [Ptot]. Therefore, all the results come out right.

Figure 3-8 shows the chip simulation result for a typical Michaelis-Menten reaction configured as in Figure 3-6(a). It can be observed that by the action of the enzymes,

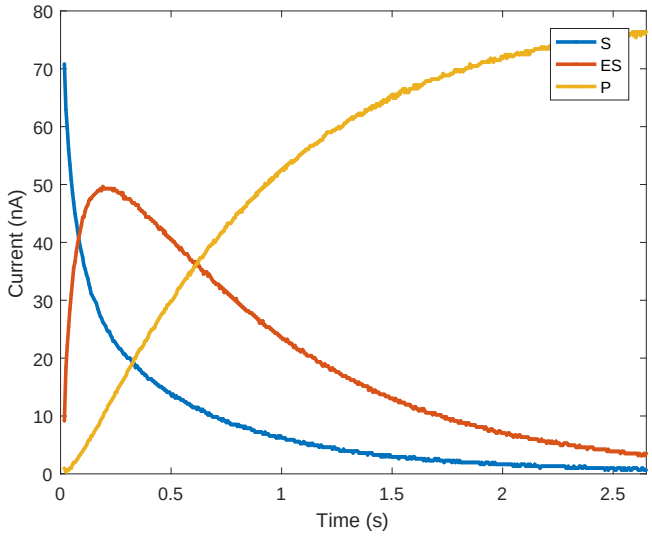


Figure 3-8: Chip simulation result for a Michaelis-Menten reaction (E+S⇌ES→E+P).

all [S] are gradually converted to [P] over time and [ES] is formed when the enzymes are at work.

3.3.8 Fan-in (Figure 3-6(b))

Fan-in is the case when a species is produced by two or more different reactions, which necessitates the most sophisticated setup among all topologies. The basic principle is to add or subtract whatever amount that comes from or goes to other reaction branches, respectively, in the current and all upstream stages, just like the amount that degrades is subtracted in the current and all upstream stages. If this idea is not implemented properly, it may lead to physically impossible situations where Ctot tries to go above Atot or Btot or below zero. The algorithm for fan-in is given by:

1. Among all the blocks that produce the same species (e.g., [C] in the example of Figure 3-6(b)), designate one block as the main block (in Figure 3-6(b), the top right block).

2. Gather all rates in the main block. That is, send all the forward (rate_fw) and reverse rates (rate_rv) of the other (non-main) blocks to the Cprod and Cdeg

ports of the main block, respectively. This creates the "total" forward (fw_tot) and reverse (rv_tot) rates, which are used to compute Ctot in the main block. Note that this rv_tot includes the degradation rate.

3. Send fw_tot and rv_tot to Cprod and Cdeg of the non-main blocks, respectively. When multiple non-main block exists, one block receives the two rates and sends its own fw_tot and rv_tot to the next block.

4. In each of the non-main block, subtract its own forward and reverse rates from the "total" forward and reverse rates, respectively. This is done by flipping the four FF_EN switches shown in Figure 3-3, and the results are produced at the fw_up and rv_up ports.

5. Propagate fw_up and rv_up of each block to its upstream stages.

6. Connect Ctot of the main block to Ctot_in of the non-main blocks, so that Afree and Bfree of these blocks can be calculated. For the non-main blocks, the Ctot switch shown in Figure 3-3 should be toggled up for this purpose.

7. The main block receives Cfree from the subsequent stage, or from its own Ctot when no subsequent stage exists, as in Figure 3-6(b). The copy of Cfree (i.e., Cfree_cp) is sent to the Cfree port of the non-main blocks, so that they can compute the reverse rates.

Note that the algorithm described above is the most general method which works for all fan-in cases. If a network consists of irreversible reactions or reactions where the reactant is not consumed (e.g., transcription or translation), its configuration becomes much simpler.

3.3.9 Loop (Figure 3-6(c))

Loop indicates the case in which the last reaction of a pathway regenerates the reactant of the first reaction. A well-known example in biology is the citric acid cycle, where oxaloacetate serves as both the first reactant and the final product. When

the final reaction of a loop is irreversible, its configuration is relatively simple, as illustrated in Figure 3-6(c) and described below:

1. Form a cascade configuration, from the first to last reaction.

2. Add the final product (Ctot of the right block in Figure 3-6(c)) to the Atot input of the first reaction block in the loop, similar to the trick used for the Michaelis-Menten reaction.

3. To prevent this Ctot from increasing without bound, set an arbitrary degradation rate for it.

4. Propagate this degradation rate to all upstream blocks except the one which creates [Atot] (the left block in Figure 3-6(c)). Or, propagate to all upstream blocks and offset this in [Atot] by sending the rate also to Cprod of the block creating [Atot]. In this case, set kr of the block equal to kdeg and adjust the forward and reverse reaction rates using KDfw and KDrv.

On the other hand, when the final reaction is reversible, it can be implemented by the combination of fan-out and fan-in configurations. For example, the network given by A⇌B⇌C⇌D⇌A can be considered as having a fan-out from [A] to [B] and [D] and a fan-in from [B] and [D] to [C].

3.4 Programmability of the Protein Chip

When designing the protein chip, we have paid particular attention to add sufficient flexibility (e.g., diverse digitally programmable parameters with wide range; several useful input and output ports and programmable on-chip interconnection among them) such that the dynamics of most gene-protein networks can be modeled using a combination of protein blocks. The following are the summary of the programmable components of a protein block:

1. **6 parameters**: (KDfw), (KDrv, kr), and (kdeg, ratC) are used to control forward, reverse, and degradation rates, respectively. These five parameters are

set by the currents generated by DACs and programmable over a dynamic range of 100 dB. The Hill coefficient, n, can be tuned between 1 and 4 by programming the effective size of an above-threshold transistor.

2. **3 types of switches**: The A_FB_EN and B_FB_EN switches are used to select the order of reaction. For a second-order reaction, they are both switched on, so that the two feedback loops described in Section 3.2 are at work; for a first-order reaction, only A_FB_EN is on and Btot is set to be constant (or vice versa); and for a zeroth-order reaction, both are off, and both Atot and Btot are constants. They are also switched off when reactants such as DNA or mRNA are recycled after use by polymerases or ribosomes and thus not effectively consumed when products are made. In the latter cases, reactants may be depleted from the cytoplasm and reside in the nucleus but they are not consumed. Other cases where these switches are manipulated are depicted in Section 3.3 (e.g., fan-out). Next, the four FF_EN switches are used for fan-in configurations (see Section 3.3.8). Lastly, the Ctot switch is responsible for 1) fan-in configurations, 2) setting Ctot's initial condition, and 3) stabilizing the integrator during startup and when not in use. To implement the latter two functions, the output of each integrator is connected to the reverse rate input (i.e., $I_{Ctot} \rightarrow I_{rv}$ in Figure 3-2) before simulation starts.

3. **7 input and 11 output ports**: Atot, Btot, Ctot, Afree, Bfree, and Cfree are widely used for most reactions; Dfree is used to model a second-order reverse reaction for dissociation/replacement and monomerization configurations; rate_fw and rate_rv are used for dissociation/replacement and fan-in configurations; rv_up and Cdeg are used to transmit degradation rates; and, rv_tot, fw_tot, fw_up, Ctot_in, Cprod, Cfree_cp, and Dfree_cp are used for fan-in and loop configurations. All of these input and output ports can be connected to any other ports via a routing bus composed of 100 parallel wires. Since variables are represented by electric currents, merging of multiple signals at a node results in addition or subtraction, depending on the direction of each current.

Finally, as for the output variables that may be sent to multiple ports in certain configurations, multiple copies of currents are produced to handle every possible case. Table 3.1 shows the number of copies for each output variable.

Table 3.1: The Number of Copies for Output Variables

Output Variable(s)	**Number of Copies**
Ctot	5 (3 positive, 2 negative)
Afree, Bfree	2
rate_fw, rate_rv	1
fw_up, rv_up	2
fw_tot, rv_tot, Cfree_cp, Dfree_cp	1

3.5 Simulation Examples of Biological Processes

This section demonstrates the functionality and usability of the protein chip by providing two examples of simulating the dynamics of well-known biological processes, the p53 pathway [100] and the glycolysis pathway [94], based on their published models.

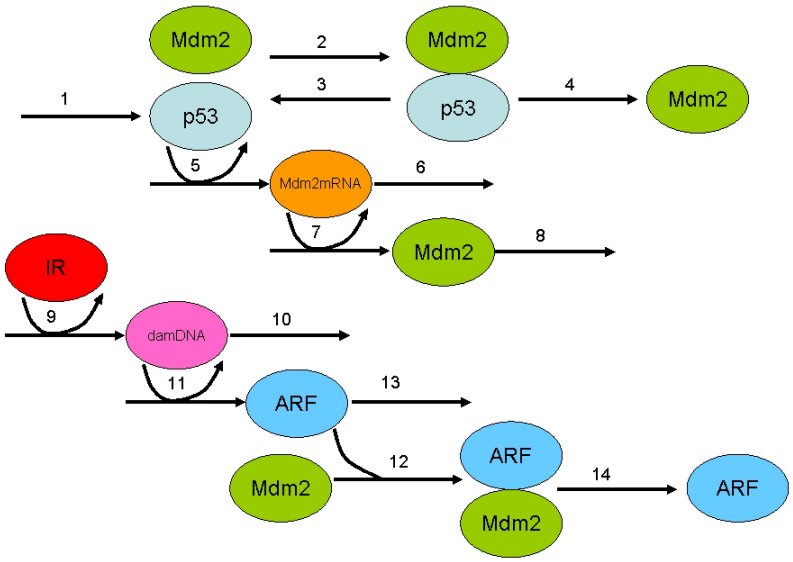
The mathematical models of the two pathways are obtained from BioModels Database [59], which offers curated computational models of a variety of biological processes in SBML format [56]. To port these software models to the protein chip, they have to go through a certain procedure: First, the models are analyzed to find out how rate laws are represented, the order of reactions, the range of parameters, and species concentrations.

This information serves as a foundation for determining the time and magnitude mappings. The time mapping determines the simulation speed of the chip and is set by considering the range of parameters available in protein blocks. For example, when a simple degradation reaction A$\rightarrow$0 with the rate of 1 s^{-1} is mapped into a protein block where C=50 pF, I=10 nA, the rate of the block is $I/(C\phi_t)\approx$8,000 s^{-1}, corresponding to 1:8000 mapping between chip time and biological time. The fastest rate constant in the biological network is mapped into the largest available current

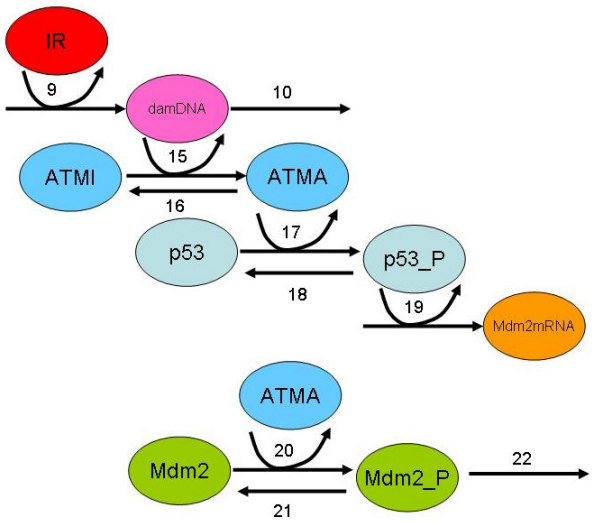
so that the simulation runs at its maximum speed. In addition, when a network contains feedback loops, high gain along with time delay arising from parasitic poles might cause instability and in turn unwanted oscillations in the protein blocks. In this case, the time mapping is also determined by the stability condition in those blocks. That is, parameters are selected such that a dominant pole is created at a frequency low enough to render the effect of parasitic poles negligible, so that the loop is stable. On the other hand, the magnitude mapping is carried out such that the range of the level of variables in the chip reliably covers that in the target biological network. Based on these time and magnitude mappings, biological parameters are converted to chip parameters (which are outlined in Section 3.4).

Next, the models are analyzed with respect to the structure of the network and the interaction among its state variables. This informs how many protein blocks are needed, which reactions each protein block models, and how interconnections among the blocks are made. When making interconnections, the methods described in Section 3.3 are utilized depending on the types of reactions. More details on how to configure protein blocks can be found in the two specific examples in this section.

After determining all the chip parameters and connections, a user can input these settings in a software program running in MATLAB. A bitstream is then generated to convey them to an FPGA. For the testing purposes, an Opal Kelly XEM6310 module featuring the Xilinx Spartan-6 FPGA is mounted on a printed circuit board (PCB) along with our protein chips. The FPGA uses the information in the bitstream to generate programming bits to orchestrate the operation of the chips and reads variables from the chips as needed, which are sent to MATLAB to graphically show the simulation results in real time. To verify the chip operation, we compared them with the software simulation results of COPASI [54] (by importing the SBML files downloaded from BioModels Database) and MATLAB (by constructing the target network using the ideal models of the protein block in Simulink, as in Figure 3-3).



(a)



(b)

Figure 3-9: (a) The ARF model and (b) the ATM model (only showing the reactions changed from the ARF model) of the p53 signaling pathway [100]. Reprinted figures originally published in [C. J. Proctor and D. A. Gray, "Explaining oscillations and variability in the p53-Mdm2 system," *BMC Syst. Biol.*, vol. 2, p. 75, 2008], available from: http://www.biomedcentral.com/1752-0509/2/75. Copyright 2008 Proctor and Gray; licensee BioMed Central Ltd. This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/2.0).

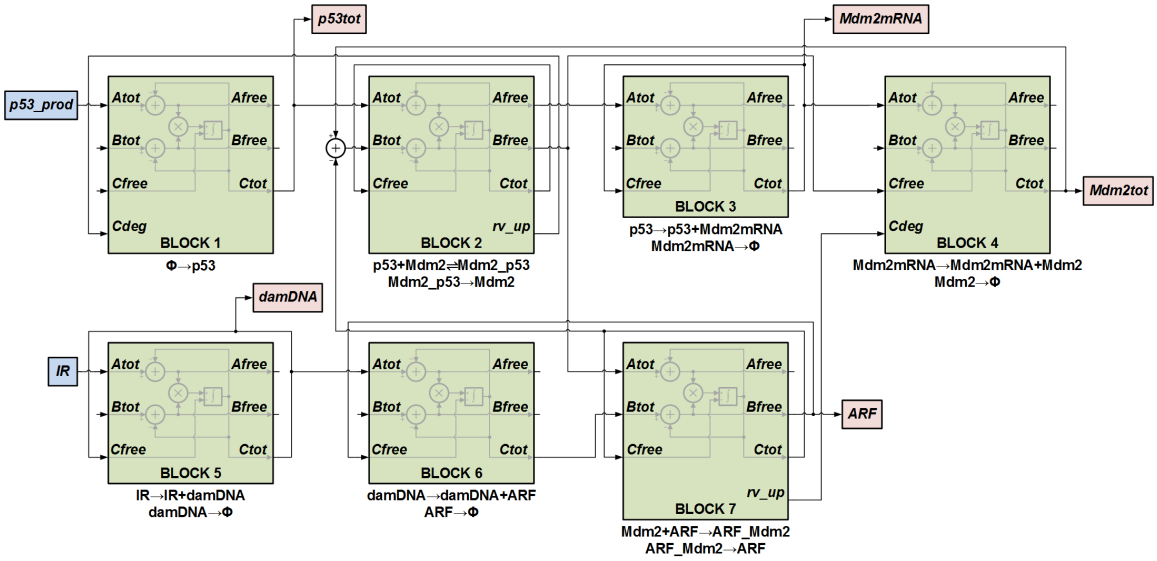


Figure 3-10: Protein block configuration for the ARF model.

Table 3.2: Chip Parameters for p53 Simulation

Reac #	Reaction	Biological Rate Const	Chip Rate Const	Blk #	Atot (nA)	Btot (nA)	KDfw (nA)	Dfree (nA)	KDrv (nA)	ratC	kr (nA)	kdeg (nA)	A_FB _EN	B_FB _EN
1	Ø→p53	7.8E-2 mol s^{-1}	7.8E4 nA s^{-1}	1	94.6	10	10	0	N/A	1	197	197	0	0
2	p53+Mdm2→Mdm2_p53	1.2E-3 $mol^{-1}s^{-1}$	1.2E3 $nA^{-1}s^{-1}$											
3	Mdm2_p53→p53+Mdm2	1.2E-5 s^{-1}	1.2E1 s^{-1}	2	p53tot	Mdm2tot -ARF_Mdm2	0.724	10	10	1	200	0	1	1
4	Mdm2_p53→Mdm2	8.3E-4 s^{-1}	8.3E2 s^{-1}											
5	p53→p53+Mdm2mRNA	1.0E-4 s^{-1}	1.0E2 s^{-1}	3	p53free	10	10	0	N/A	1	23.9	23.9	0	0
6	Mdm2mRNA→Ø	1.0E-4 s^{-1}	1.0E2 s^{-1}											
7	Mdm2mRNA→Mdm2mRNA+Mdm2	5.0E-4 s^{-1}	5.0E2 s^{-1}	4	Mdm2mRNA	10	10	0	N/A	0.433	118	239	0	0
8	Mdm2→Ø	4.3E-4 s^{-1}	4.3E2 s^{-1}											
9	IR→IR+damDNA	8.0E-2 s^{-1}	8.0E4 s^{-1}	5	25 (IR pulse)	400	10	0	N/A	1	478	4.78	0	0
10	damDNA→Ø	2.0E-5 s^{-1}	2.0E1 s^{-1}											
11	damDNA→damDNA+ARF	3.3E-5 s^{-1}	3.3E1 s^{-1}	6	damDNA	10	10	0	N/A	1	7.89	23.9	0	0
13	ARF→Ø	1.0E-4 s^{-1}	1.0E2 s^{-1}											
12	Mdm2+ARF→ARF_Mdm2	1.0E-2 $mol^{-1}s^{-1}$	1.0E4 $nA^{-1}s^{-1}$	7	Mdm2free	ARFtot	0.5	0	N/A	1	1200	239	0	1
14	ARF_Mdm2→ARF	1.0E-3 s^{-1}	1.0E3 s^{-1}											
15	damDNA+ATMI→damDNA+ATMA	1.0E-4 $mol^{-1}s^{-1}$	3.3E0 $nA^{-1}s^{-1}$	8	600	damDNA	15	10	10	0	12	0	1	0
16	ATMA→ATMI	5.0E-4 s^{-1}	5.0E1 s^{-1}											
17	ATMA+p53→ATMA+p53_P	5.0E-4 $mol^{-1}s^{-1}$	1.7E1 $nA^{-1}s^{-1}$	9	p53free	ATMA	3000	10	10	0	12000	0	0	0
18	p53_P→p53	5.0E-1 s^{-1}	5.0E4 s^{-1}											
19	p53_P→p53_P+Mdm2mRNA	1.0E-4 s^{-1}	1.0E1 s^{-1}	3	p53free +p53_P	10	10	0	N/A	1	2.39	2.39	0	0
20	ATMA+Mdm2→ATMA+Mdm2_P	2.0 mol^{-1} s^{-1}	6.7E2 nA^{-1} s^{-1}											
21	Mdm2_P→Mdm2	5.0E-1 s^{-1}	5.0E2 s^{-1}	10	Mdm2free	ATMA	0.75	10	10	1	120	9.56	0	0
22	Mdm2_P→Ø	4.0E-4 s^{-1}	4.0E1 s^{-1}											
23	Ø→p53mRNA	1.0E-3 mol s^{-1}	3.0E2 nA s^{-1}	11	30	10	10	0	N/A	1	2.39	2.39	0	0
24	p53mRNA→Ø	1.0E-4 s^{-1}	1.0E1 s^{-1}											

The reactions 1-14 are for the ARF model, and the reactions 15-24 are added for the ATM model, replacing the reactions 11-14 in the ARF model. KDrv of N/A refers to "no reverse reaction", in which case KDrv can be set as any value. All Hill coefficients are 1. The unit "mol" in biological rate constants denotes the number of molecules. IR indicates the dose of irradiation and damDNA indicates the amount of damaged DNA.

3.5.1 p53 Signaling Pathway

The p53 signaling pathway shown in Figure 3-9 is a network of great significance involved in DNA repair, apoptosis, and suppressing cancer [100]. The authors of [100] mathematically account for the stabilization (i.e., activation) of p53 due to DNA damage, using two stochastic models of the network, the ARF and ATM model. The models particularly focus on explaining the sustained oscillatory behavior observed in cells as long as the damage remains, caused by a negative feedback loop with delay arising from intermediate steps (e.g., transcription and translation).

Table 3.2 summarizes the reactions in both models, their reactions rate constants, and chip rate constants set via time and magnitude mappings. Note that the reaction numbers in the leftmost column match with those in Figure 3-9. Furthermore, as an example to illustrate block-to-block connections, the configuration for the ARF model is presented in Figure 3-10. The reactions that each block is responsible for and the parameters of the block selected to simulate those reactions are also listed in Table 3.2.

Basically, for the ARF model, 7 protein blocks are used to model 14 reactions with 8 state variables. The blocks 1–4 form a negative feedback loop (p53 production $\rightarrow$ Mdm2mRNA $\rightarrow$ Mdm2 $\rightarrow$ Mdm2_p53 $\rightarrow$ p53 degradation) and the blocks 5–7 comprise a cascade reaction (IR event $\rightarrow$ damDNA $\rightarrow$ ARF $\rightarrow$ ARF_Mdm2 $\rightarrow$ Mdm2 degradation) which triggers an oscillation in the loop. As for the ATM model, 9 blocks (blocks 1–5 and 8–11) are used to model 20 reactions with 11 state variables. An oscillation is triggered by the phosphorylation of Mdm2, which is in turn caused by the irradiation (IR) event.

Time and Magnitude Mappings

For the ARF model, the dynamics of the reactions 2–4 in determines the time mapping, since it requires both a high gain and a fast time constant, posing a stability threat to the protein block 2. We found that up to the time mapping of 10^6:1 between biological time and chip time, the simulation ran without any unwanted ringing in

the waveforms. Thus, 30 hours of the system behavior can be simulated in approximately 0.1 s. Given that the maximum magnitude for all the species in this model is approximately 350 molecules, 1 molecule is mapped into 1 nA of current in the chip.

It is a bit more challenging to maintain stability for the ATM model, since it contains very fast time constants—those for the dephosphorylation of p53 and Mdm2 that are more than 500 times faster than the fastest time constant of the ARF model. Besides, due to the high rate constant for Mdm2 phosphorylation, the loop gain of the block 10 (i.e., ATMA/KDfw) goes as high as 800. This high gain lowers the level of Mdm2free, which in turn creates low-frequency parasitic poles. The combined effect of the high gain, fast time constant, and low-frequency parasitic poles leads to an undesirable oscillation in the block. Hence, we first reduced the rate constants for Mdm2 phosphorylation and dephosphorylation (reactions 20 and 21) by 100 times. This adjustment hardly changes the simulation result because the rapid equilibrium approximation requires that the rate constant of the reaction 21 is much greater than that of the reaction 22, and the former are still 12.5 times greater than the latter after adjustment. In addition, to decrease the impact of parasitic poles, the magnitude and time mappings are altered such that 1 molecule corresponds to approximately 3 nA of current, and 10^5 s of biological time to approximately 1 s of electronic time.

Connections among Protein Blocks

The block-to-block connections for the ARF model are straightforward and self-explanatory in Figure 3-10. However, there are a few points that require special attention: As explained in Section 3.4, the A_FB_EN and B_FB_EN switches should be set according to the order of reaction—for example, zeroth for the p53 synthesis, first for transcription, translation, and degradation reactions, and second for the p53/Mdm2 binding. However, except for the reactions 2 and 12, reactants are not consumed by reactions, in which case the corresponding A_FB_EN and B_FB_EN switches are disabled. Next, note that the rates of Mdm2-dependent p53 degradation (reaction 4) and ARF-dependent Mdm2 degradation (reaction 14) should be propagated to the blocks that produce p53tot (block 1) and Mdm2tot (block 4),

respectively. Note also that Mdm2 reacts with both p53 and ARF, so the fan-out configuration described in Section 3.3.3 is applied for the blocks 2 and 7. As for the ATM model, the fan-out configuration is needed for p53 and Mdm2, and the fan-in configuration for Mdm2mRNA.

Analysis of the Chip Results

The 8 plots in Figures 3-11 and 3-12 are the deterministic and stochastic simulation results of software and the chip, for the ARF and the ATM model, when a short irradiation event occurs at time=0. It can be seen that the chip simulations produce the waveforms that are highly correlated to those of the software simulations, taking into account the time and magnitude scale factors described previously. To perform stochastic simulations, p53, Mdm2, Mdm2mRNA, ARF, and p53mRNA are selected as stochastic variables, where the noise generators add noise.

Note that the oscillatory behavior of the ATM model is exhibited only when noise is added, due to the averaging effect present in the deterministic model [100]. That is, it is a kind of biological phenomena which can only be studied by taking stochastic modeling approaches. For such phenomena, deterministic simulations are not sufficient to verify that the model adequately reflects the essence of the biological process enough to explain the data acquired from biological experiments [137]. Our chip can potentially provide huge speedup in such cases, especially because of the fact that running stochastic simulations, regardless of the number of reactions or number of molecules, essentially does not increase the chip simulation time. The stochastic simulation for the ARF and ATM model took 0.1 s and 2.5 s, respectively, in COPASI on a 3.4 GHz computer consuming nearly tens of Watts of power in its microprocessor, and 0.1 s and 1 s, respectively, in our mm-size chip consuming 30 mW of power. Thus, we are already seeing some speedup over software even for this relatively simple network. Much higher advantages can arise when the scale of the network is increased as we outline in the next chapter.

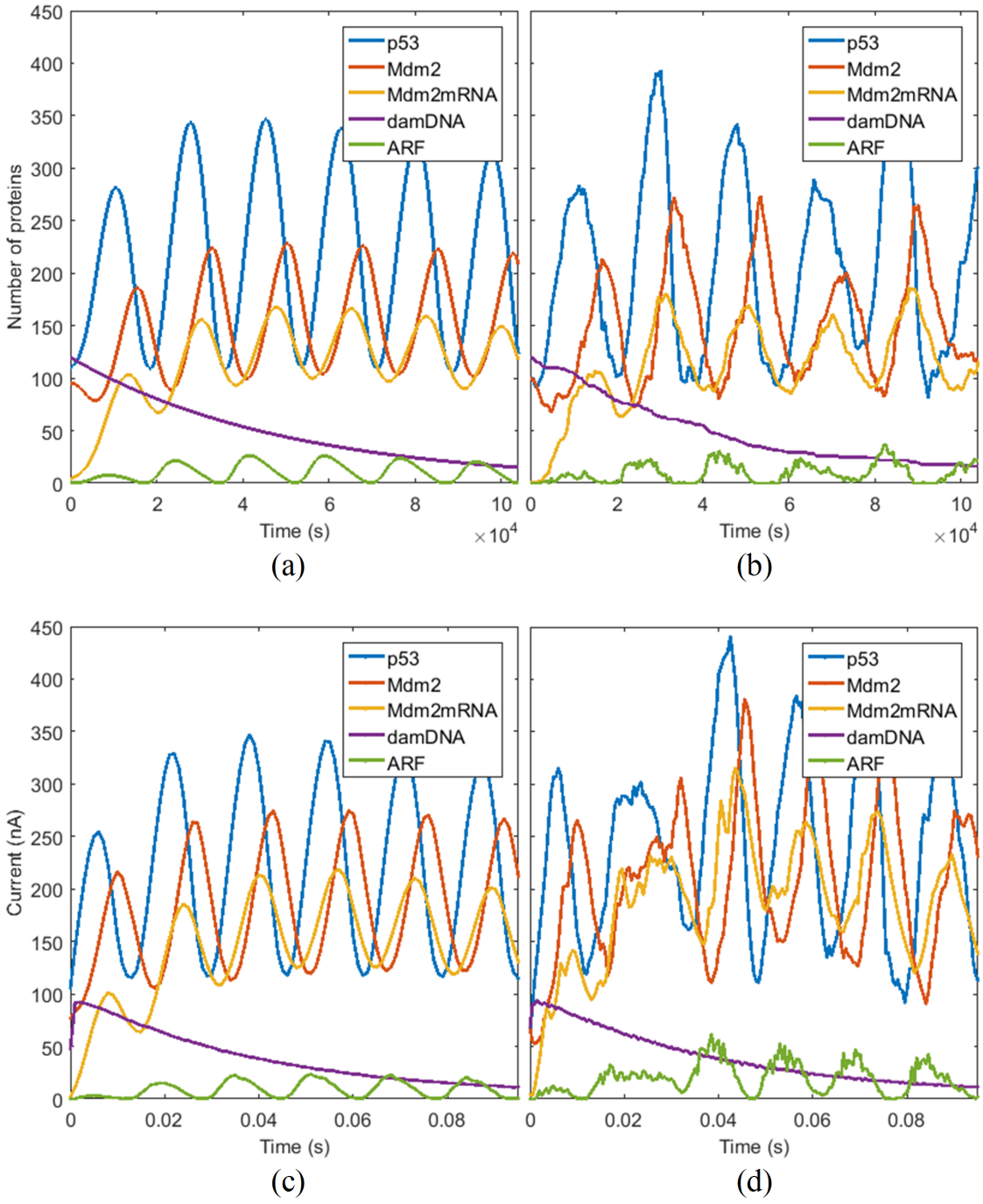


Figure 3-11: (a) Deterministic and (b) stochastic simulation results of software for the ARF model. (c) Deterministic and (d) stochastic simulation results of the chip for the ARF model. To be consistent with the original published plots (Fig. 12 and 13 in [100]), the total levels (free + bound amounts) of p53 and Mdm2 are plotted.

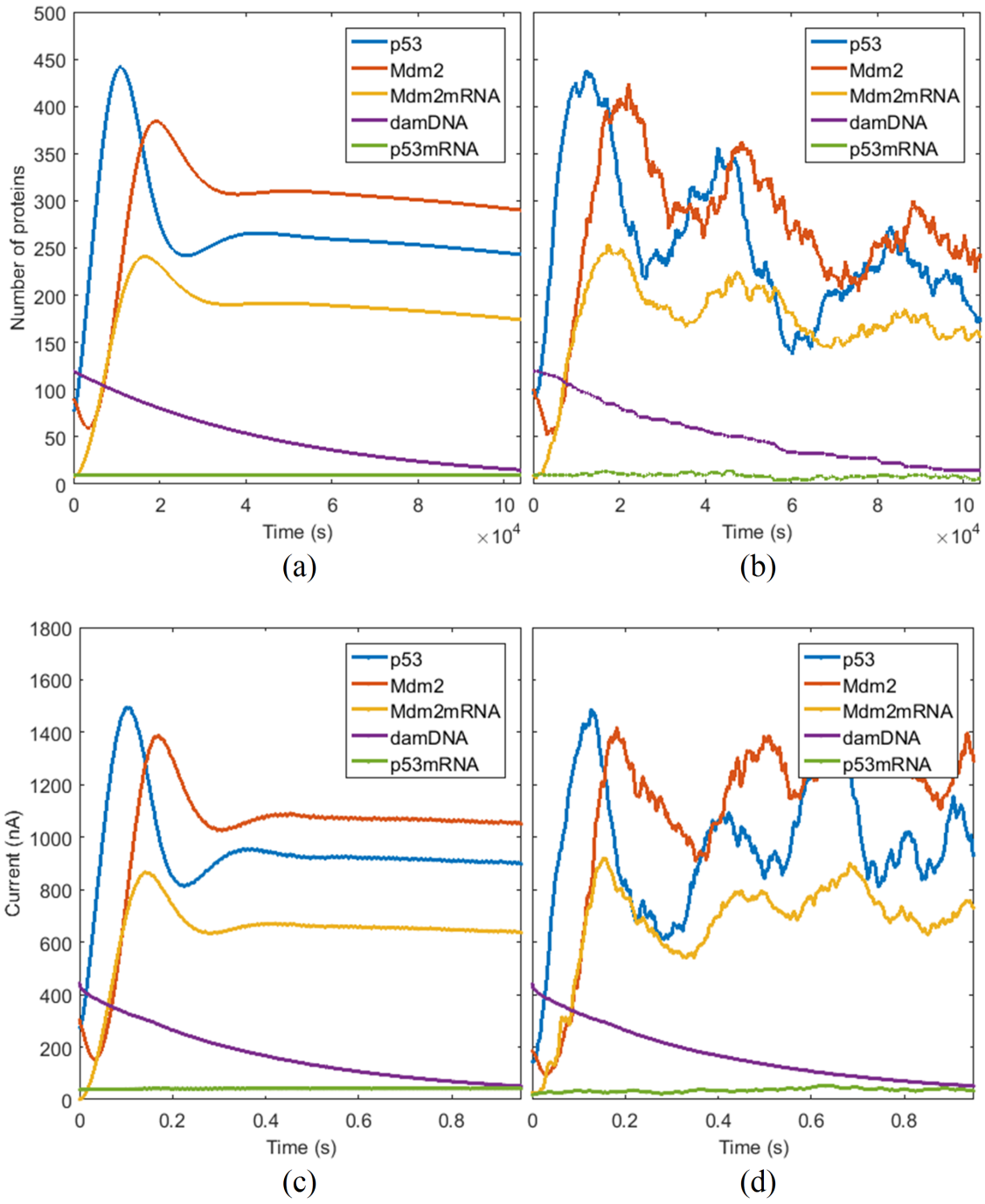


Figure 3-12: (a) Deterministic and (b) stochastic simulation results of software for the ATM model. (c) Deterministic and (d) stochastic simulation results of the chip for the ATM model. To be consistent with the original published plots (Fig. 12 and 13 in [100]), the total levels (free + bound amounts) of p53 and Mdm2 are plotted.

3.5.2 Glycolysis Pathway

Glycolysis is one of the most important metabolic pathways which converts glucose into pyruvate. Our target model presented in [94] consists of 10 reactions, most of which are enzyme-catalyzed reactions. The oscillatory behavior of glycolysis has been studied by researchers, since quantitative analysis of the behavior can give insight into characterizing the kinetics of its pathway. Our goal was to reproduce the oscillation of this relatively complex biological system on the chip.

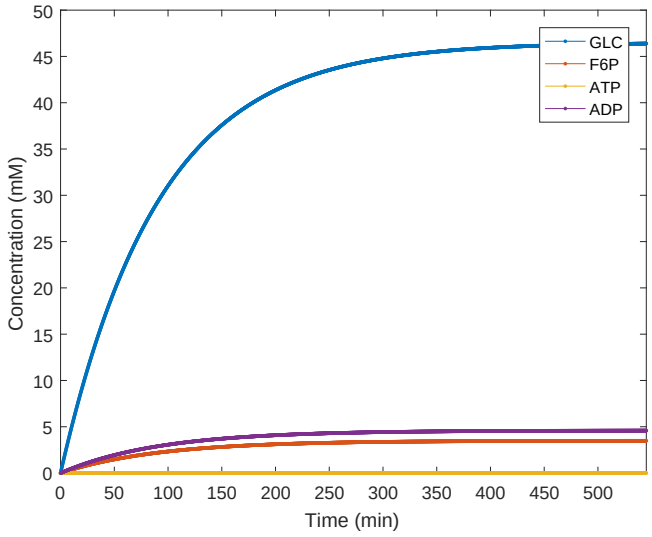
Decomposing the Reactions

Table 3.3: Chip Parameters for Glycolysis Simulation

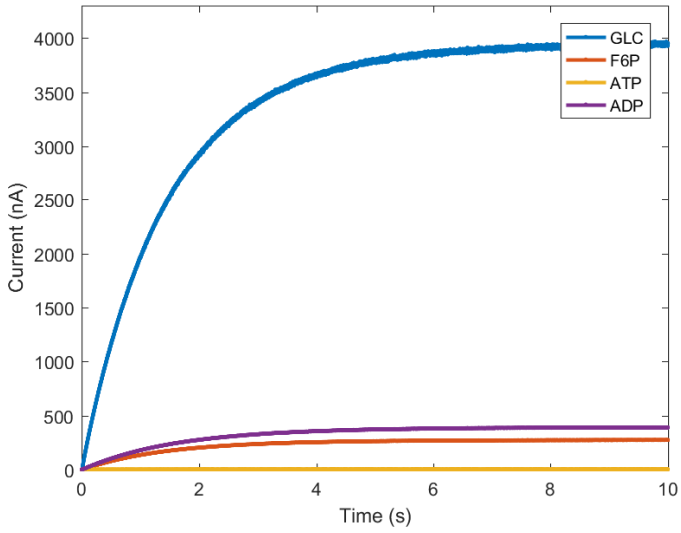
Reaction	Block #	Decomposed Reactions	Atot (nA)	Btot (nA)	KDfw (nA)	Dfree (nA)	KDrv (nA)	ratC	kr (nA)	kdeg (nA)	n	A_FB _EN	B_FB _EN
GLC+ATP→F6P+ADP (E: HK, A: GLC, B: ATP, P: F6P, Q: ADP)	1	∅→GLC→∅	55	10	10	0	N/A	1	10	0.11	1	0	0
	2	∅→ATP→∅	3.85	10	10	0	N/A	1	10	0.11	1	0	0
	3	∅→ADP→∅	1.21	10	10	0	N/A	1	10	0.11	1	0	0
	4	E+A⇌EA	50	A	10	10	10	0	100	0	1	1	0
	5	EA+B⇌EAB	EA	B	6.3	10	10	0	100	0	1	1	0
	6	EAB→E+P+Q, P→∅	EAB	10	10	0	N/A	1	10	0.11	1	0	0
F6P+ATP→FBP+ADP (E: PFK, E': PFK_T, A: F6P, B: ATP, C: AMP, P: FBP, Q: ADP)	7	E+2A⇌EA^2	150	A	2.83	20	10	0	100	0	2	1	0
	8	EA^2+2B⇌EA^2B^2	EA^2	B	1	10	10	0	100	0	1	1	0
	9	EA^2B^2→E+2P+2Q, P→∅	EA^2B^2	10	10	0	N/A	1	10	0.11	1	0	0
	10	E+2B⇌E'+2C	E	B	7.67	C	1	0	100	0	2	0	0
FBP⇌2GAP	11	FBP⇌2GAP, GAP→∅	FBP	10	10	GAP	2	1	10	0.11	1	1	0
GAP+NAD→DPG+NADH (E: GAPDH, A: GAP, B: NAD, P: DPG, Q: NADH)	12	∅→NAD→∅	4.4	10	10	0	N/A	1	10	0.11	1	0	0
	13	∅→NADH→∅	0.264	10	10	0	N/A	1	10	0.11	1	0	0
	14	E+A⇌EA	1000	A	100	10	10	0	100	0	1	1	0
	15	EA+B⇌EAB	EA	B	100	10	10	0	100	0	1	1	0
	16	EAB→E+P+Q, P→∅	EAB	10	10	0	N/A	1	10	0.11	1	0	0
DPG+ADP⇌PEP+ATP	17	DPG+ADP⇌PEP+ATP, PEP→∅	DPG	ADP	100	ATP	200	1	10	0.11	1	1	0
PEP+ADP→PYR+ATP (E: PK, A: PEP, B: ADP, P: PYR, Q: ATP)	18	E+A⇌EA	1000	A	20	10	10	0	100	0	1	1	0
	19	EA+B⇌EAB	EA	B	30	10	10	0	100	0	1	1	0
	20	EAB→E+P+Q, P→∅	EAB	10	10	0	N/A	1	10	0.11	1	0	0
PYR→ACA (E: PDC, A: PYR, P: ACA)	21	E+A⇌EA	200	A	30	10	10	0	100	0	1	1	0
	22	EA→E+P, P→∅	EA	10	10	0	N/A	1	10	0.11	1	0	0
ACA+NADH⇌EtOH+NAD	23	ACA+NADH⇌EtOH+NAD, EtOH→∅	ACA	NADH	1	NAD	7000	1	0.1	0.11	1	1	1
F6P→P	24	F6P→P, P→∅	F6P	10	200	0	N/A	1	10	0.11	1	0	0
2ADP⇌AMP+ATP	25	2ADP⇌AMP+ATP, AMP→∅	ADP	ADP	10	ATP	10	1	10	0.11	1	1	0

KDrv of N/A refers to "no reverse reaction", in which case KDrv can be set as any value.

It should be noted that the rate laws of enzyme-catalyzed reactions in biochemical networks are often not expressed as mass action kinetics. However, it is in fact possible to decompose most such rate equations into rates based on mass action kinetics. For example, the first reaction of the glycolysis pathway, GLC+ATP→F6P+ADP, is known to be an irreversible ordered bi-bi mechanism, with the rate expression of V_1[ATP][GLC]/((K_1+[GLC])(K_2+[ATP])) [94]. Such mechanism can be decomposed



(a)



(b)

Figure 3-13: Simulation results from (a) software and (b) the chip for the 12 decomposed unidirectional reactions to model the first enzymatic reaction in the glycolysis pathway, GLC+ATP→F6P+ADP.

into these reactions: E+A⇌EA, EA+B⇌EAB, and EAB→E+P+Q (refer to Table 3.3 to see notation and selected chip parameters). Figure 3-13 reveals that the chip simulation of these reactions, i.e., 12 unidirectional reactions for block 1–6, produces the dynamics that are in good agreement with software simulation (refer to the mappings in the following section).

Another reaction in the pathway, F6P+ATP→FBP+ADP, involves allosteric inhibition and activation by ATP and AMP, respectively, and can be decomposed into the 8 unidirectional reactions shown in Table 3.3 [94, 132]. Their kinetic rate constants can be derived from the original "lumped" rate expression to yield the same dynamics.

For some rate equations that appear frequently, it may be possible to design a few dedicated circuits to create those equations. The block to set a Hill coefficient (parameter "n" in Figure 3-3) or the analogic DAC in the gene chip (see Section 2.2.4) are the examples of such circuits. As for the functions more suitable for digital computation, such as delay, the FPGA will readily be able to process them. Finally, in many cases where abstraction is allowed, it may be sufficient to approximate the input-output characteristics of a given function using the Hill coefficient and the dissociation constant (KDfw in Figure 3-3) of the protein block.

Protein Block Configuration

To carry out a time mapping, the fastest dynamics in the system needs to be clarified. For the reactions where rapid equilibrium approximation or quasi-steady-state approximation is applied to produce the desired rate law (e.g., the reactions for block 4 and 5 in Table 3.3), we expect their dynamics to be faster than others. As such, the values of their kr were set to be 10 times larger than that of the block which create the final rate at which the products are produced (e.g., block 6). Then, the stability of those blocks with high kr dictates that a rate of 1 min^{-1} in biology is mapped into approximately 54.5 s^{-1} in the chip, corresponding to 3270x faster simulation than biological time. In addition, since the maximum concentration of all species is about 37 mM in software simulation, 1 mM of concentration was mapped into approximately

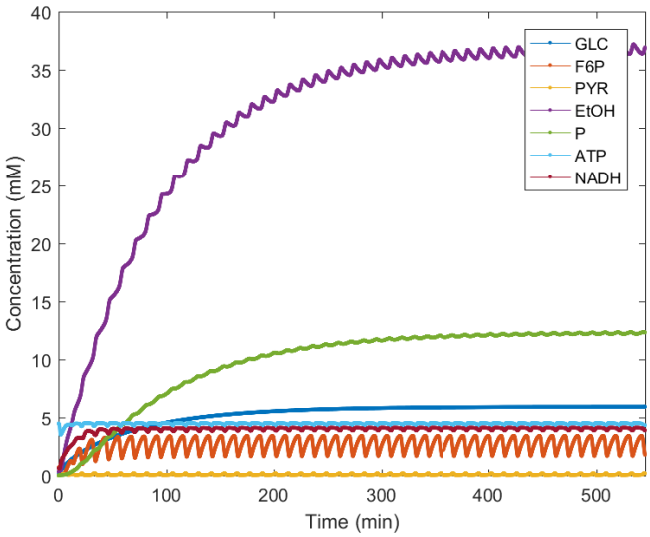
100 nA of current in the chip.

Based on these mappings, the rate constants for the decomposed reactions can be used to determine the chip parameters. Table 3.3 shows the 10 reactions in the glycolysis pathway decomposed into 54 unidirectional reactions with 30 state variables and the parameters selected for each protein block. Note that one additional block (block 26) is required to compute the reverse rate of block 10, since a Hill circuit is needed to set the rate. One protein chip contains 20 protein blocks, so it is necessary to use two chips to simulate the glycolysis pathway. Block 1–10, 17–20, and 24–26 are located in the first chip and block 11–16 and 21–23 are in the second chip.

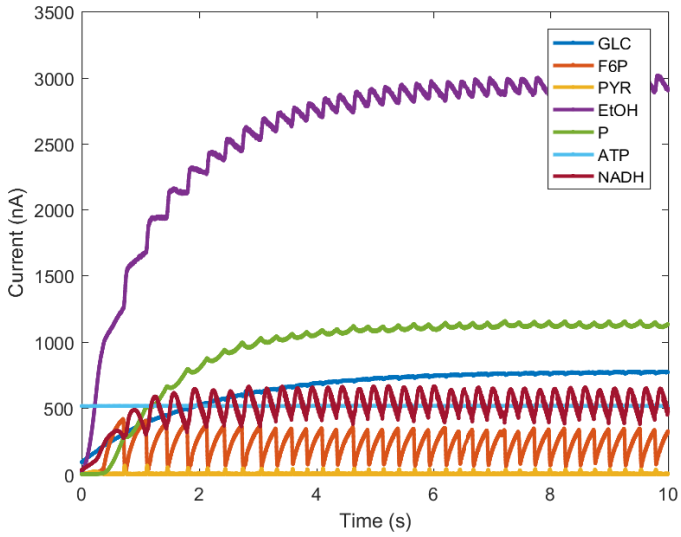
Next, we set the connections between the input and output ports of the blocks, using 48 wires and 29 wires of the routing bus of the first and the second chip, respectively. Interestingly, the glycolysis pathway contains many different types of reactions and topologies illustrated in Section 3.3—irreversible and reversible reactions, Michaelis-Menten reactions (all enzymatic reactions), cascade (GLC$\rightarrow$F6P$\rightarrow$FBP$\rightarrow$...), fan-out (e.g., ATP used in multiple reactions), fan-in (e.g., ADP produced in multiple reactions), and loop (e.g., NAD$\rightarrow$DPG$\rightarrow$...$\rightarrow$ACA$\rightleftharpoons$NAD). For each case, the algorithms described in Section 3.3 were employed to route signals. Furthermore, 2ADP$\rightleftharpoons$AMP+ATP and FBP$\rightleftharpoons$2GAP were implemented in a way similar to the configuration for dimerization and monomerization, respectively, and the reactions that take the form of A+B$\rightarrow$C+D were implemented using the configuration method for replacement reactions.

Analysis of the Chip Results

For the chip simulation of the glycolysis pathway, the primary sources of error were current mismatches (due to random variations in transistor sizes) and the limited resolution of the DACs (4-bit in all ranges). In fact, we found that the effect of these non-idealities on the glycolysis simulation was high, because it is a long chain of reactions with the aforementioned complexities. Software simulations of the model also showed that its dynamics is sensitive to such non-idealities, e.g., changes in the initial levels of substrate concentrations. Thus, to alleviate the difficulties of



(a)



(b)

Figure 3-14: Simulation results from (a) software and (b) the chip for the mathematical model of the glycolysis pathway [94].

this simulation, we incorporated two simplifications: First, to mitigate the effect of the error caused by the DACs, chip-to-chip connections were made by directly sending analog current variables. Second, since the ATP concentration remains nearly constant, we set it as a constant value, which decreases the complexity of the network.

Figures 3-14(a) and 3-14(b) show the results of the simulations run by software and the chip, respectively. In can be seen that the chip simulation is successful in capturing the small oscillations in the network. In periodic steady state, when the time scale factor is taken into account, the period of oscillation in the chip simulation is 17.6% higher than that in the software simulation. The oscillation amplitude of NADH shows the biggest discrepancy, because it is particularly sensitive to the mismatches in the variables sent between the blocks in the feedback loop around NADH. The delay due to parasitic capacitances in the system also has an effect of increasing both the amplitude and period of oscillation.

The lessons learned from these simulations are threefold: First, with the capability of the protein chip, the dynamics of the biological networks that are small-scale or less sensitive to mismatches (e.g., genetic networks, where connections are simpler because reactants are not consumed in transcription and translation) can be effectively modeled and simulated. Second, even for a relatively complex network such as a glycolysis pathway, the chip is able to reproduce the essential behavior of its dynamics. Third, to precisely simulate subtle dynamics of sophisticated large-scale networks, efforts should be made to reduce simulation errors. It can be achieved by employing various error correction techniques, many of which are outlined in textbooks such as [116] for systems such as ours.

3.6 Specifications of the Protein Chip

Table 3.4 outlines the performance characteristics of the protein chip. The dynamic range of 100 dB (five orders of magnitude) is wide enough to represent reaction rates and molecular concentrations of many biochemical reaction networks in living cells [3, 92]. A protein block currently occupies 0.04 mm^2 in the chip. Note that

Table 3.4: Performance Characteristics of the Protein Chip

Parameter	**Value**
Technology	AMS 0.35 μm BiCMOS
Supply voltage	3.3 V
Maximum number of reactions	60
Maximum number of state variables	60
Number of protein blocks	20
Number of noise generators	4
Area occupied by one protein block	0.04 mm^2
Dynamic range of variables (DAC output)	100 dB (100 pA–10 μA)
Number of DACs per protein block	8
Hill coefficient	1–4
Typical kr, kdeg range	0.4–40,000 s^{-1}
Number of wires in the routing bus	100
Programming clock frequency	5 MHz
Programming time	70 μs
ADC sampling clock frequency	5–40 MHz
ADC readout time	2.5 μs–2 ms
ADC input range	100 pA–10 μA
Number of ADCs per chip	24
Signal-to-noise ratio	4–35 dB
Power consumption	$\sim$30 mW
Chip size	4.3 mm $\times$ 4.0 mm

this area will shrink with the use of modern process technologies that offer smaller transistor sizes and better matching properties [98].

3.7 Conclusion

We have presented a 0.35 μm BiCMOS cytomorphic chip capable of simulating the mass-action kinetics of up to 60 chemical reactions by configuring universal analog computational units, i.e., the protein blocks. Several features of the protein block, including various digitally programmable parameters with wide dynamic range, carefully designed input and output ports, and connection mechanisms of those ports depending on reactions types and network topologies, provide the chip the ability to model and simulate arbitrary biochemical reaction networks in cells.

The detailed simulation examples of two published computational models, the p53 and the glycolysis pathways, suggest that the chip effectively captures the dynamics of such networks and enables reduction in simulation time even in a relatively small-scale network with 11 state variables. As one of the possible ways of simulating the rate laws not expressed as mass action kinetics, we demonstrated how the reactions can be decomposed to create the desired dynamics.

Furthermore, since it is indeed time-consuming to manually set the parameters and connections of large-scale networks, we are also working toward developing a compiler which automates this task. It will eventually take a mathematical model from a standard database such as BioModels Database, analyze and process it, and yield chip programming bits in an optimized fashion [1].

With these concerted efforts, including the strategies to scale and to reduce the effect of mismatches, in the near future, we expect to achieve the computing power capable of executing the deterministic and stochastic simulation of biological networks with more than thousands of genes and proteins, providing orders-of-magnitude speedup over conventional simulation methods.

Chapter 4

Toward Large-Scale Simulation of Biological Networks

This chapter is devoted to describing the complete cytomorphic system which can carry out large-scale simulations and provide performance benefits, built with an array of gene chips and protein chips presented in previous chapters. First, Section 4.1 explains the scalable and parallel architecture of the system, which is essentially an ensemble of cytomorphic chips, FPGAs, and software. In Section 4.2, for demonstrating purposes, we show a proof-of-concept implementation of the system (i.e., the cytomorphic board) and its test results. Section 4.3 compares the speed performance of the board with COPASI [54], a widely used software application to simulate biochemical reaction networks. Finally, Section 4.4 discusses the factors that promote or constrain the speedup of the system, how to overcome those constraints, several performance trade-offs present in our architecture, and the advantages of our design over alternative approaches.

4.1 The Architecture of the Cytomorphic System

Figure 4-1 shows the high-level block diagram of the cytomorphic system, which illustrates the general flow of interactions between components. Basically, the system is divided into computer (software) and cytomorphic-board (hardware) domains.

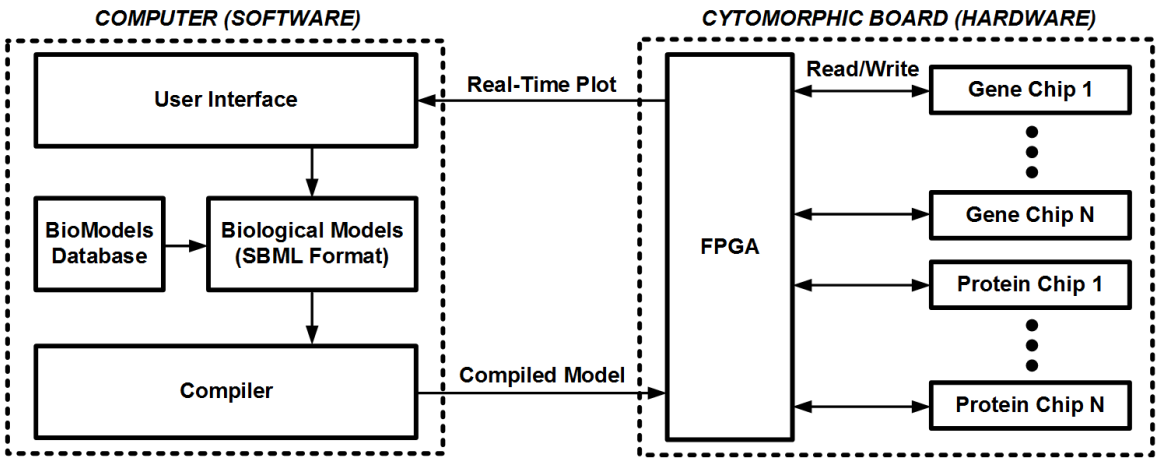


Figure 4-1: High-level block diagram of the cytomorphic system.

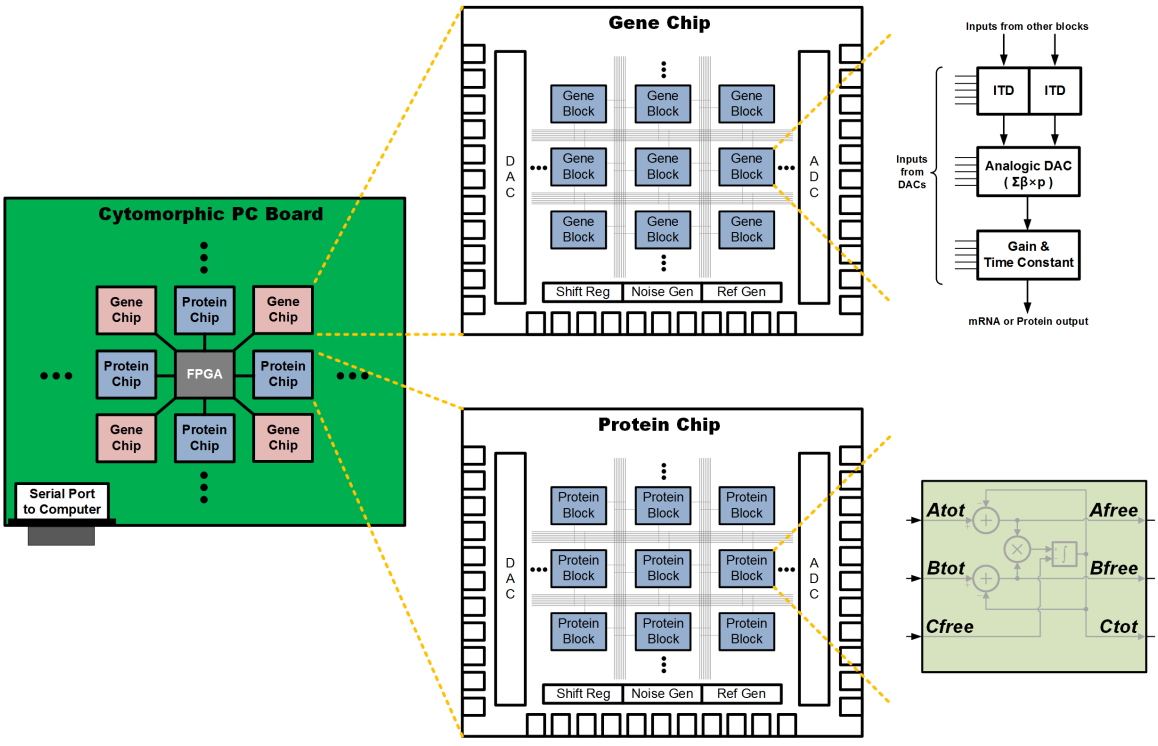


Figure 4-2: Overall architecture of the cytomorphic PC board and the cytomorphic chips.

A computer talks with an FPGA, which in turn talks with cytomorphic chips. A compiler analyzes given biological models represented in standard formats such as SBML [56] to generate optimized configurations for the analog component instances in the cytomorphic chips [1][1]. A bitstream is then generated to convey the interpreted model to the FPGA. The FPGA stores the information in memory elements and uses it to produce programming bits to set the configurations (i.e., parameters, connectivity, initial conditions, etc.) of the chips. Not only that, if needed, the FPGA carries out various functions that are suitable in the digital domain, including establishing connections among the chips through periodic read, decode, encode, and write operations, performing high-speed digital signal processing such as scaling, time delay, and error correction, and transferring data to a computer to create real-time plots for users or to analyze the results with software tools (e.g., sensitivity analysis).

By encapsulating and streamlining what happens behind the scene—the transition and communication between different layers and domains (software and hardware; MATLAB, SBML, compiler language, FPGA language, and chip-specific data representation; analog and digital; and continuous and discrete)—the system can be considered as a black box which rapidly gives out outputs when inputs are fed in. Without knowing hardware details or underlying principles, a user can interact with a user interface to load and simulate biological models and benefit from accelerated simulation.

Figure 4-2 is a visual representation of the architecture of the cytomorphic board. It emphasizes the array of cytomorphic chips mounted on the board, communicating with each other via an FPGA, and the arrays of gene blocks and protein blocks in the gene chips and the protein chips, respectively, interconnected with each other via on-chip routing channels. The serial port on the board serves as an interface between the FPGA and the computer. The details of the gene chip and the protein chip can be found in Chapters 2 and 3, respectivly, and the actual implementation of the board

[1]The development of this compiler is being led by the Program Analysis and Compilation Group at MIT. See [1] for more information. Since it is not yet fully compatible with our system, the chip testing results presented in this thesis are obtained via manually interpreting biological models and configuring chips.

can be seen in Section 4.2.

On this board, by mounting a number of chips, parallel simulation of large quantities of gene-protein interactions can be performed. This is one of the main ideas to achieve speedup, as in multi-core computation techniques or dedicated hardware accelerators such as graphics processing units (GPUs). Note that there exist various computing devices in the board working in different domains: When molecular basis functions (e.g., binding reactions and Hill functions) are directly mapped into analog building blocks in the chips, their computation takes place in the continuous time and continuous signal domain; on the other hand, the FPGA operates in the discrete time and discrete signal domain, when it performs read, write, and additional data processing; ADCs and DACs in the chip allow to cross between these continuous and discrete domains. These analog and digital devices synergistically interact with each other to attain efficiency, flexibility, and robustness.

Most importantly, all of the above operations run "simultaneously", regardless of the number of reaction mechanisms to simulate or the number of variables to read and write, and those operations altogether handle all the computationally intensive portion of the simulation. Otherwise, if a computer (which typically talks with hardware through a serial cable and executes commands in a serial fashion) undertook any intensive data processing during the running time, it would undermine the idea of full parallelization, inevitably becoming a bottleneck of simulation. Thus, the computer in our system only takes part in tasks that are irrelevant to main computation, such as taking user commands, analyzing models, and data plotting. Ensuring parallelism in every aspect of the design while avoiding serial data processing or communication is the key to harnessing the potential of the cytomorphic chips to the full extent and further increasing the scale and speed of the system.

4.2 Implementation of the Cytomorphic Board

The scalable architecture of our cytomorphic system enables massively parallel computation, and it can be realized by constructing a massively parallel array of cyto-

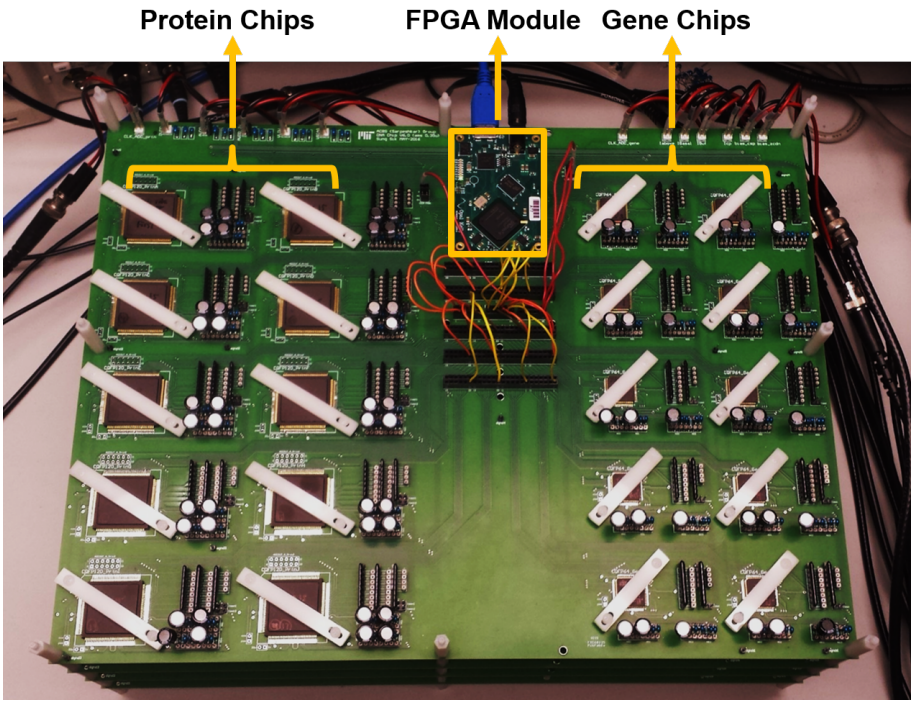


Figure 4-3: A prototype of the cytomorphic board to run large-scale simulations.

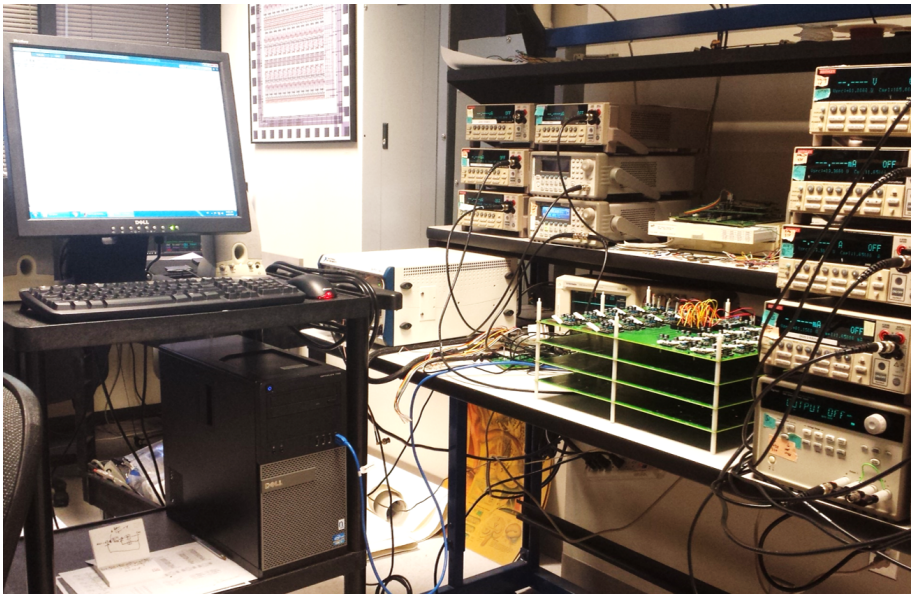


Figure 4-4: Experimental setup for testing the cytomorphic board.

morphic chips and FPGAs on a printed circuit board (PCB) substrate. Figure 4-3 shows a prototype of such a board. It has the dimensions of 15.9 × 12.0 inches and consists of 10 gene chips, 10 protein chips, and an FPGA module. Figure 4-4 shows the setup for our chip measurements.

Each gene chip, specialized in modeling the dynamics of inducer-transcription-factor-DNA binding, transcription, and translation, can simulate up to 80 reactions. Each protein chip, on the other hand, can simulate up to 60 reactions of general biochemical reaction networks. Both chips possess 4 noise generators. Hence, this board provides a computing power to simulate up to 1,400 reactions in total, with 80 stochastic state variables. Note that as shown in Figure 4-4, the boards can be stacked on top of each other. Although we only tested with one board to prove the concept, we have enough parts to build 4 boards and stack them. This will create a peak computing power equivalent to 5,600 reactions, with 320 stochastic state variables.

The FPGA module mounted in the middle of the board is Opal Kelly's XEM6310 FPGA integration modeule[2]. It features the Xilinx Spartan-6 FPGA and 124 I/O pins to interface with the chips. Using this module instead of a separate FPGA chip eliminates the time-consuming tasks to configure the FPGA and set up FPGA-to-chip and FPGA-to-computer interfaces.

Each cytomorphic chip currently requires 8 digital signals for initial programming of the chip and variable updates. Among them, 4 signals are clock and enable signals which can be shared among all the chips. Thus, with 124 I/O pins, the FPGA module is able to simultaneously write to all the 20 chips on the board, while reading out 40 variables from the chips. Note that we can take advantage of remarkable technical improvements of modern FPGAs. For example, Xilinx's Virtex UltraScale FPGAs[3] offer >5.5 millions of logic cells, >88 Mb of memory, and >1,400 I/O pins, providing substantial capacity to orchestrate the operation of a large number of chips in a parallel fashion.

With the completed prototype of the board, the functionality of individual cy-

[2] https://www.opalkelly.com/products/xem6310/

[3] http://www.xilinx.com/support/documentation/selection-guides/ultrascale-fpga-product-selection-guide.pdf

tomorphic chips was first tested, using various biological network models simulated previously in Sections 2.4 and 3.5. After verifying the functionality, we proceeded to run simulations in all of the chips at once, for the same networks. By doing so, we verified that all the chips were reliably producing simulation results at the same time and that the FPGA was successful in controlling 20 chips simultaneously. For example, Figuress 4-5 and 4-6 show the waveforms for the stochastic simulations of 10 repressilator networks and 10 p53 networks, respectively, performed both in the board and in software. The variables representing the levels of LacI molecules and p53 molecules are plotted over time for the former and the latter network, respectively.

Chip-to-chip variations exist in the system due to various sources of errors such as mismatches in ADC characteristics or in reference current sources for DACs. They can be compensated for by applying mapping tables or scale factors (stored in the FPGA) for the inputs and outputs of the chips, based on the measured characteristics of individual chips, as is now routinely used in analog-to-digital converters for non-linearity, gain, and offset error correction. Note also that the simulations to produce results for Figures 4-5 and 4-6 are performed in parallel in the cytomorphic board and in series in software. The performance gain we achieved through implementing this board is discussed in the following section.

4.3 Speed Comparison with Software

From the stochastic simulations of 10 repressilator and p53 networks described in the previous section, a simple but important attribute of our system can be seen—that simulation time is irrelevant to network scale. This is because all computations are performed in parallel, including the read and write operations of the FPGA. Note that in drawing this relationship, we assumed that the number of simulated networks represents the network scale. It is reasonable to make this assumption because for the Gillespie algorithm [39, 40], as far as simulation time is concerned, what matters is the number of reactions and the total reaction rate. Accordingly, 10 copies of a network can be viewed as computationally analogous to a bigger network with

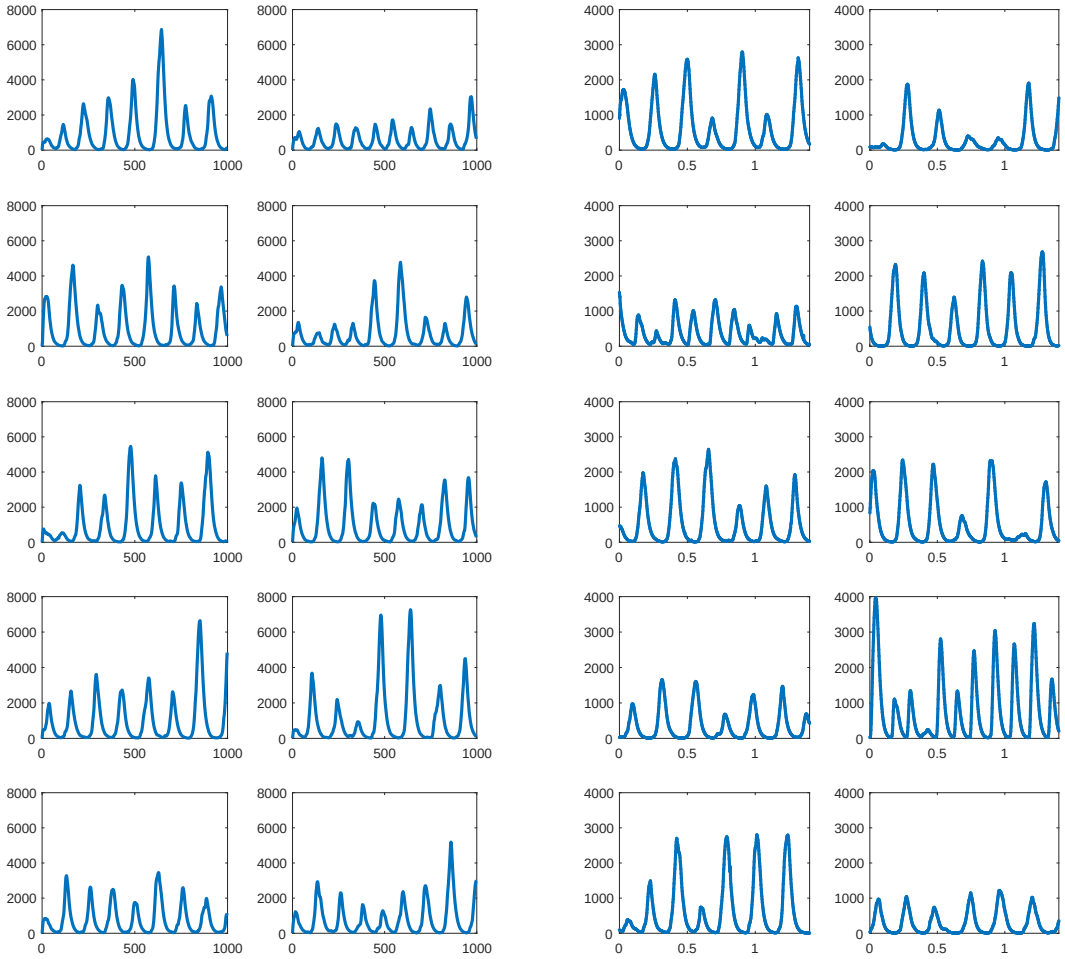


Figure 4-5: The waveforms for the levels of LacI molecules, obtained from stochastic simulations of the repressilator network (see Section 2.4.1), using (a) software (10 runs in series) and (b) the cytomorphic board (1 run in parallel). Units – (a) x axis: minute, y axis: number of molecules, (b) x axis: second, y axis: ADC output (1 ADC output $\approx$ 2.5 molecule. Variations may exist among chips.)

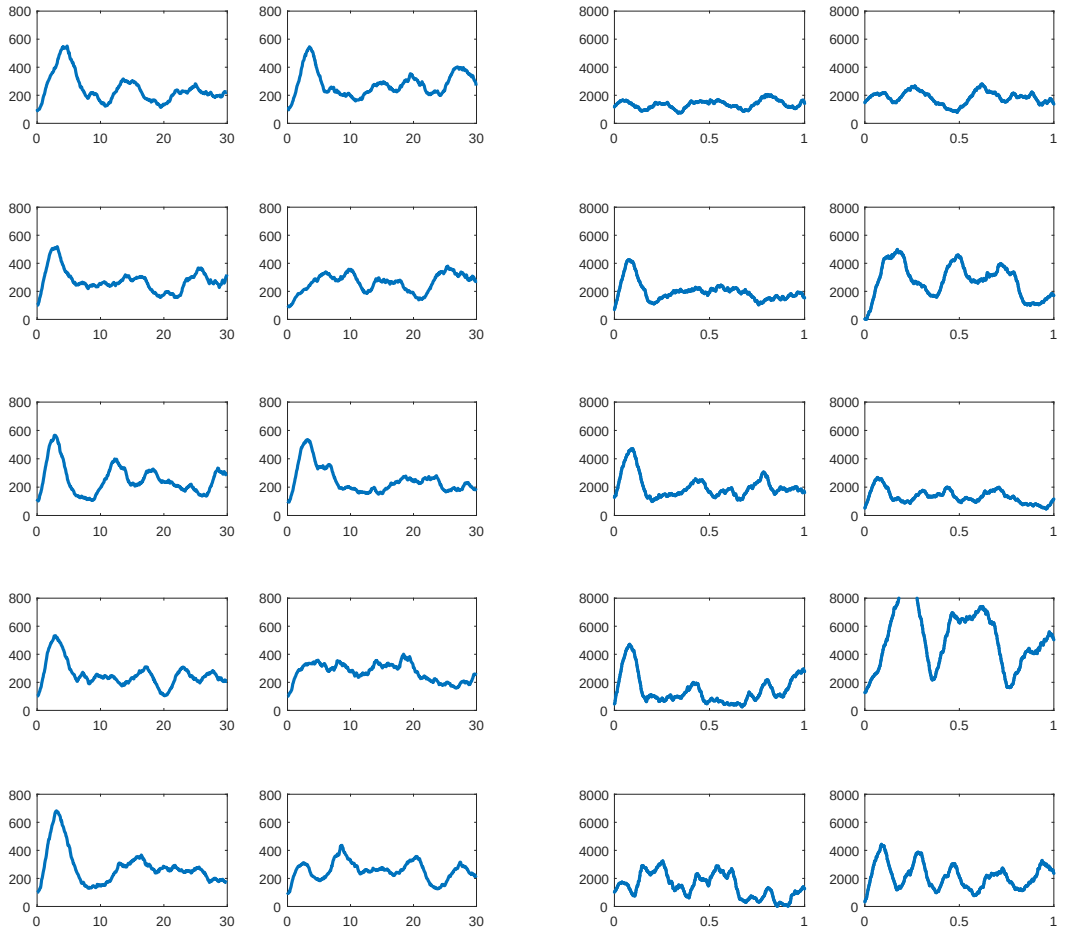


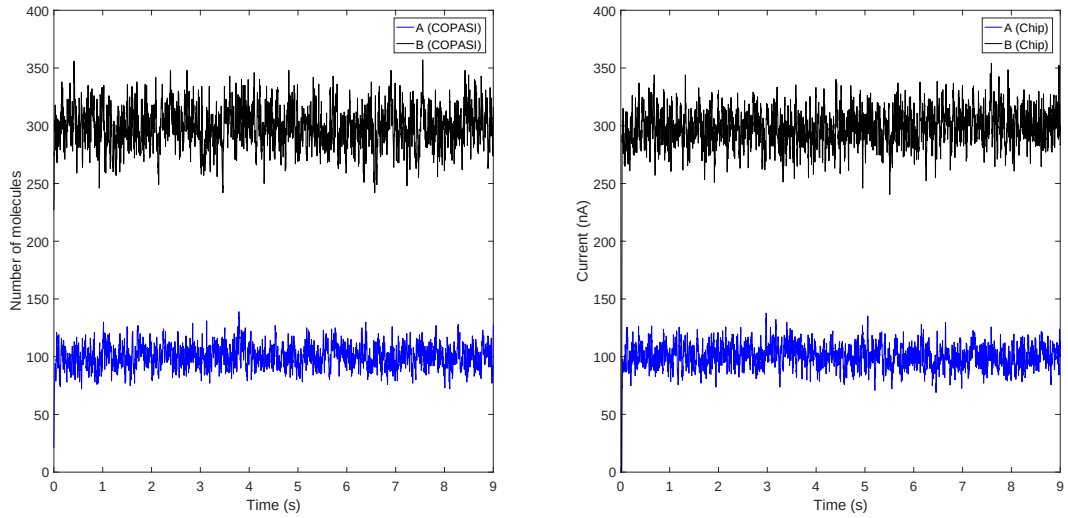
Figure 4-6: The waveforms for the levels of p53 molecules, obtained from stochastic simulations of the p53 network (see Section 3.5.1), using (a) software (10 runs in series) and (b) the cytomorphic board (1 run in parallel). Units – (a) x axis: hour, y axis: number of molecules, (b) x axis: second, y axis: ADC output (1 ADC output $\approx$ 0.1 molecule. Variations may exist among chips.)

10 times more reaction mechanisms with similar reaction rates. In case of the p53 pathway, a simulation of 10 copies took 30 s in COPASI (on a 3.4 GHz computer) and 1 s in our system, corresponding to a 30x speedup.

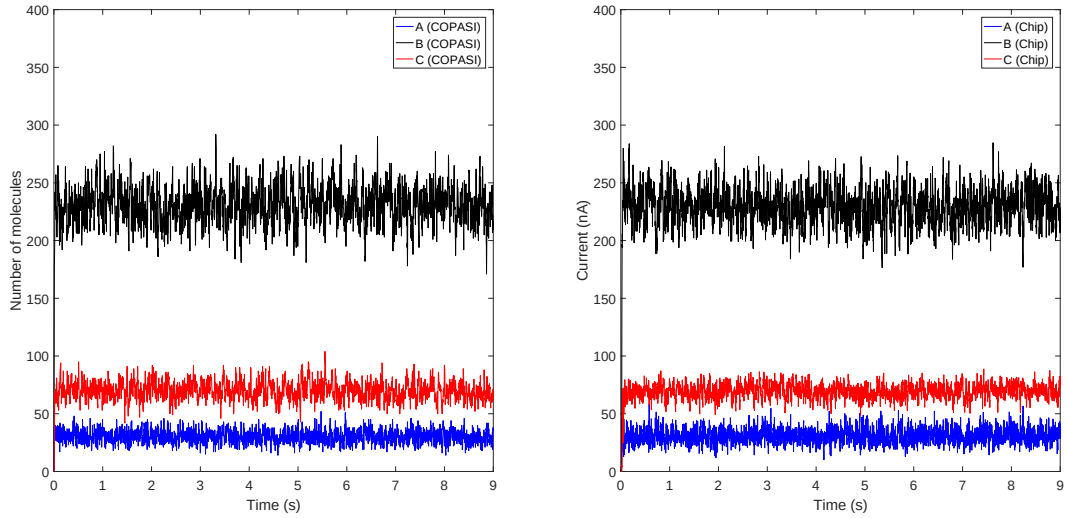
Next, we were motivated to examine the maximum speed gain the board can provide. In a nutshell, the board gives greater advantages for long stochastic simulations of networks containing many fast reactions. This is essentially the same as the condition for the simulation of stiff systems, where fast and slow timescales coexist. Simulation runs slowly because the time step size goes small to capture fast dynamics, while the number of time steps becomes large to account for slow dynamics [42, 137]. In biochemical reaction networks, fast dynamics is produced by the existence of large reaction rate constants and/or high molecule copy numbers.

For instance, Figure 4-7 shows the stochastic simulation results of COPASI and the chip, for a simple 7-reaction network which consists of $\varnothing \xrightarrow{k_1} \mathrm{A} \xrightarrow{k_2} \varnothing$, $\varnothing \xrightarrow{k_3} \mathrm{B} \xrightarrow{k_4} \varnothing$, and $\mathrm{A}+\mathrm{B} \underset{k_6}{\overset{k_5}{\rightleftharpoons}} \mathrm{C} \xrightarrow{k_7} \varnothing$ [63]. 1 molecule is mapped into 1 nA of current and the time mapping is 1:1. It can be seen that the data from COPASI and the chip show good quantitative agreement to each other. To investigate the relationship between reaction rate and simulation time, this network was simulated for various parameters.

Figure 4-8 presents the change in measured simulation time, as the number of molecules increases and time constants are fixed [63]. That is, the degradation/reverse rate constants (k_2, k_4, k_6, and k_7) are fixed and the synthesis/forward rate constants (k_1, k_3, and k_5) are varied such that the desired number of molecules are created for all three species. For software simulation, the former rate constants are set as $k_2 = k_4 = k_6 = 3.2 \times 10^3$ s^{-1} and $k_7 = 0$ and the time to simulate the time span of 1 second is measured. For chip simulation, three different time mappings are applied to map the software parameters to the chip: The top, middle, and bottom lines in Figure 4-8 represent the measured simulation time when the rate constant of 3.2×10^3 s^{-1} in the software model is mapped into 3.2×10^3, 3.2×10^4, and 3.2×10^5 s^{-1}, respectively, which yield the simulation time of 1, 0.1, and 0.01 second, respectively. Next, Figure 4-9 shows the change in measured simulation time, as the number of replicated reactions increases (1:1 time mapping, 1 molecule=1 nA, molecule count=3000 for all species)



(a)



(b)

Figure 4-7: COPASI (left) and chip (right) data for a simple 7-reaction network ($\varnothing \xrightarrow{k_1} \mathrm{A} \xrightarrow{k_2} \varnothing$, $\varnothing \xrightarrow{k_3} \mathrm{B} \xrightarrow{k_4} \varnothing$, and $\mathrm{A}+\mathrm{B} \underset{k_6}{\overset{k_5}{\rightleftharpoons}} \mathrm{C} \xrightarrow{k_7} \varnothing$), for two different parameter sets [63]. Parameters (for both software and the chip): (a) $k_1 = 1.2 \times 10^4$, $k_2 = k_4 = 120$, $k_3 = 3.6 \times 10^4$, and $k_5 = k_6 = k_7 = 0$. (b) $k_1 = 1.2 \times 10^4$, $k_2 = k_4 = k_7 = 120$, $k_3 = 3.6 \times 10^4$, $k_5 = 1.2$, and $k_6 = 0$.

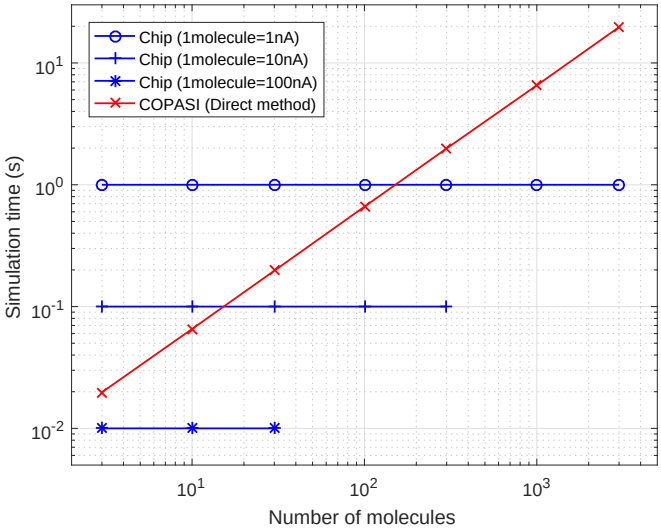


Figure 4-8: The relationship between simulation time and the number of molecules, for stochastic simulation performed using COPASI and the chip [63]. Chip simulation is done with three different time and magnitude mappings.

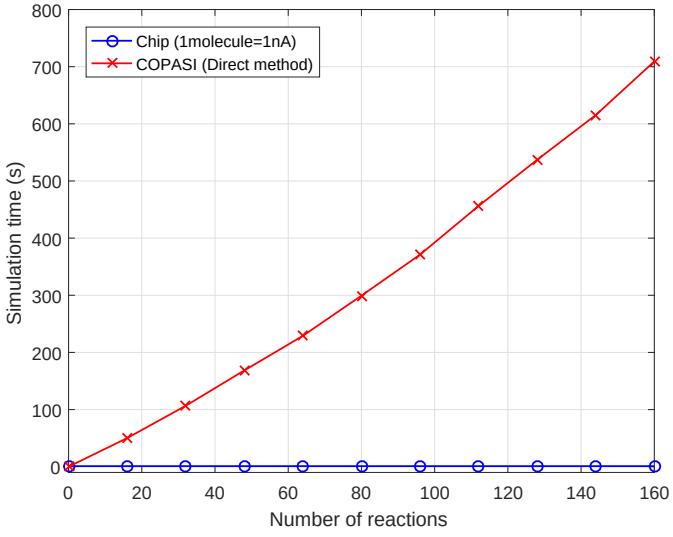


Figure 4-9: The relationship between simulation time and the number of reactions, for stochastic simulation performed using COPASI and the chip [63].

[63]. Essentially, the two figures illustrate the fact that as the number of molecules or the number of reactions scales, simulation time increases nearly proportionally in software but remains unchanged in our platform. In other words, higher total reaction rates require longer simulation time only for software. The peak speedup is achieved when the rates are highest.

What determines this peak speedup is the dynamic range of the mean random clock frequency. In the current implementation of the noise generator circuit, it has to be configured according to the reaction rate constant. That is, owing to its architecture, faster reaction rate constants (high I_{kr}/C) lead to higher levels of noise. Thus, as we map a given rate constant of a model to a faster rate constant in a cytomorphic chip, the noise level has to be reduced accordingly via switching certain digital selection bits, such that the mean frequency of the random clock for a given molecule number is increased. However, this programmable frequency range has an upper limit. For example, for a magnitude mapping of 1 molecule = 1 nA, the highest reaction rate constant for the chip is kr = 3.2×10^3 s^{-1}. This corresponds to 1:1 time mapping between software and chip parameters, so in this case, the time the chip takes to simulate the time span of 1 second simply becomes 1 second.

On the other hand, for the given time and magnitude mappings, when the molecule number increases, the mean frequency of the random clock also increases automatically to adjust the level of noise, by the operation of a molecule-number sensing circuit and a frequency-locked loop [63]. At around 3,000 molecules (= 3,000 nA), however, the frequency reaches its maximum value. Beyond this point, the noise circuit can no longer produce appropriate levels of signal-to-noise ratio (SNR). In sum, when 1 molecule is mapped into 1 nA, the fastest chip simulation time is 1 second (with the time mapping of 1:1), and the peak speedup (∼20x for the 7-reaction network, as shown in Figure 4-8) arises when the molecule number is around 3,000.

Another important point to note is that when the molecule number is low, there is a chance to simulate faster by manipulating the magnitude mapping. For example, as shown in Figure 4-8, software simulation runs faster than chip simulation when

the number of molecules is 100[4]. In this region, there may be less motivation to use the chip. However, if we change the time mapping such that 1 second of the model is mapped into 0.1 second of the chip, the chip simulation can be run in 0.1 second. Then, why not use this time mapping from the beginning? The reason is as follows: With the mappings described in the previous paragraph, there is no more room to adjust the time mapping; the noise circuit is already configured to produce the highest frequencies possible, so this new time mapping results in excessive molecular fluctuations. A trick to alleviate this constraint is to adjust the magnitude mapping as well. If we map 1 molecule to 10 nA instead of 1 nA, the SNR for a given molecule number effectively increases, thereby compensating for the amount decreased by adjusting the time mapping. Consequently, it allows to use the new time mapping which leads to faster chip simulation.

However, it should be noted that this comes at the expense of a reduced dynamic range of molecule levels. Since the mean random clock frequency reaches it maximum value at 3,000 nA, the new magnitude mapping dictates that the maximum molecule number is now 300. Thus, in this particular case, the dynamic range of species concentrations has been reduced by a factor to 10. Therefore, this scheme can be used when it is assured that the molecule levels stay within a certain boundary. If so, adaptive changing of mappings enables similar degrees of speedup over software in a wide range of molecular copy numbers, as depicted in Figure 4-8. Furthermore, it is also possible to apply different magnitude mappings for different species, by leveraging scaling functions of FPGAs and on-chip multipliers. This technique will help when high and low molecule levels coexist in a system.

Finally, the maximum performance of the cytomorphic board is exhibited when the full capacity of all noise generators is utilized. The aforementioned 7-reaction system uses 3 noise generators. As shown in Figure 4-9, as we simulate more replicated reactions with the same reaction rate, the board provides higher speed advantage over software—up to 700x over COPASI and 30,000x over MATLAB, when all 80

[4]Note that in light of the typical number of molecules in cells [92], this is an unlikely condition, especially when simulating large-scale networks.

noise generators (4 per chip $\times$ 20 chips) are used to simulate 160 reactions with stochasticity.

Note that as the number of reactions increases, software simulation time increases faster than proportionally. This is due to the nature of the Gillespie algorithm[5]: In every iteration of the algorithm, there are tasks whose cost is relevant to the number of reactions (e.g., calculating the propensity function of each reaction) and irrelevant to the number of reactions (e.g., two random number generations). Among these tasks, random number generation is the most expensive and dominates the total computation time [81]. In addition, when the number of reactions (with the same rate) or molecular copy numbers increases, it leads to a proportional increase in the total reaction rate and in turn the number of time steps. When the number of reactions is small, this results in a nearly proportional increase in simulation time. However, when it becomes large, the cost of the tasks that is dependent on the number of reactions becomes increasingly expensive. Thus, both the number of time steps and the work load in each time step increase, so simulation time increases faster than proportionally. In other words, simulation time is approximately the function of $xy(ax + b)$ (a$\ll$b), where x is the number of reactions, y is the number of molecules, a is the cost of tasks that are dependent on the number of reactions, and b is the cost of tasks that do not vary with the number of reactions. When x is small, this expression behaves like bxy (proportional to the number of reactions), and when x is big, it behaves like ax^2y (proportional to the square of the number of reactions). This reemphasizes that our system will provide greater advantages as the scale grows.

Note also that COPASI is a state-of-the-art software application optimized for simulation of biochemical reaction networks. As a reference, we simulated the same 7-reaction system in MATLAB, a commonly used mathematical solver, using the same Gillespie algorithm[6]. It took approximately 45 times longer than in COPASI, elucidating that MATLAB may be a less desirable environment to run stochastic

[5]Specifically, the direct method is considered here.

[6]To run this simulation, we used an open-source MATLAB function available in this link (Copyright (c) 2012, Nezar Abdennur): `http://www.mathworks.com/matlabcentral/fileexchange/34707-gillespie-stochastic-simulation-algorithm`

simulation of biological networks.

Prudent readers might wonder why the p53 network simulation could only achieve a 30x speedup over COPASI. This is because it is limited by stability, rather than noise generation. The next section describes this issue and how to overcome it to achieve faster simulation.

4.4 A Discussion of Simulation Speed

4.4.1 Analog vs. Digital

In the previous section, we showed how to utilize our cytomorphic computer to accelerate simulation. However, in fact, digital approaches using multi-core technologies, GPUs, FPGAs, or custom digital integrated circuits might also create computing capabilities that surpass that of software. Such efforts are introduced in Sections 1.2.1 and 1.2.2. On the other hand, we also introduced in Sections 1.2.3 and 1.3 analog approaches to develop computing systems and the merits they can provide. Indeed, whether analog or digital solutions can yield better results is dependent on applications, and the best would be an optimized combination of the two worlds [109, 114]. Thus, we would like to highlight the most crucial reasons why the strategy to implement analog-circuit-based computing cores can thrive in our application:

1. Most biological systems inevitably span multiple scales in terms of timescale, network size, or molecular population size. The system we designed is largely immune to this multiscale nature (see Section 4.3), whereas digital tools suffer when simulating such systems with stochasticity. This is because they necessitate a large number of computationally expensive random number generations, owing to a large number of time steps and a large computational load in each time step [81].

   Despite the possibilities to ease this computational challenge by algorithmic improvements (e.g., the next reaction method by Gibson & Bruck [36]) and parallel processing (e.g., the use of GPUs [70, 127]), some multiscale barriers are trickier

to overcome. For example, the first reaction method contains several types of tasks which can take advantage of parallel computing, especially random number generation (one per reaction per time step). However, because of this, when the number of reactions becomes large, the total computational cost for random number generation grows prohibitively high. Thus, the direct method is often more preferable. However, although the number of random number generation per time step is fixed as 2 for the direct method, an increase in the number of reactions has a direct impact on the number of time steps (especially if the number of "fast" reactions increases). Consequently, the number of random number generation increases. In addition, the both methods slow down nearly proportionally as the molecular population increases. Thus, digital acceleration techniques may bring about substantial benefits when executing multiple runs of small-scale simulations, but not as much for multiscale simulations.

Let's reexamine this in a more intuitive manner. If the total reaction rate[7] increases, it means that reaction events occur more frequently. This requires small time intervals to be created by the Gillespie algorithm, since it separately accounts for each reaction event. That is, it randomly generates one event, updates the molecule count of corresponding species, and then move on to the next iteration. In other words, to distinguish between two reaction events occurring almost at the same time, the algorithm has to reduce the size of the discrete time step. As a result, the number of time steps and in turn simulation time increase. On the other hand, analog computation is characterized by asynchronous, continuous-time, and continuous-signal processing and direct mapping of the state variables of dynamical system models to physical entities (e.g., voltages and currents) of analog circuits. Electrons go around the system to compute and change state variables at every moment of time, by moving toward the same direction (addition) or opposite directions (subtraction) or accumulating on a capacitor (integration), etc. As such, if two reaction events occur almost at the same time, it is inherently accounted for in the continuous-

[7]The sum of the rates of all reactions in the system.

time domain, without any special treatment to distinguish between the two. Updating opeations also happen naturally and immediately, via explicit wire connections and Kirchhoff's current law which enable straightforward addition and subtraction functions. This fundamental difference between analog and digital computation suggests that analog computation makes it possible to implement genuine parallel processing of biochemical reactions and thus may be a better fit for our application.

2. Contrary to the analog solution, most digital methods to tackle the above issues necessarily demand accuracy to be compromised for speed. For example, the explicit tau-leaping method allows multiple reaction events to occur during a time interval, thereby performing better when molecule numbers are relatively high [41,103]. However, it comes at the price of a loss of accuracy and yet cannot completely remove the molecule-number dependency. Other approximate algorithms including the Langevin approaches, the implicit tau-leaping algorithm, and the slow-scale stochastic simulation algorithm also sacrifice accuracy to gain speed [42]. Unfortunately, there is little unifying theory as to the required accuracy of stochastic simulation algorithms for biological studies [137]. Yet, we view it as an opportunity where our system can be leveraged to shed light on such uncertainties.

3. Studies have shown theoretically and experimentally that at the levels of precision which biological systems commonly use for computation, analog computation can be much more energy efficient than digital computation [109, 110], both in electronics and in cells. Grounded on the cytomorphic mapping, our computing units use the rich basis functions that already exist in analog transistor citcuits for addition, multiplication, integration, noise generation, etc. This eliminates the cost for reinventing these functions with logic gates. Although we did not aim at optimizing power consumption in this thesis project, our chips consume tens of mW of power and still perform faster simulation than a microprocessor, which typically consumes tens of Watts. Furthermore, by

achieving higher speedup by using the techniques outlined in Section 4.4.3, our system will be able to offer considerably higher power efficiency. It will become an appealing trait in this era of putting more emphasis on producing higher performance "per power".

4.4.2 Overcoming Speed-Limiting Factors

We mentioned in the previous section that our analog-circuit-based computation is much less susceptible to the challenges arising from multiscale stochastic models. Which factors then limit the speed of our system? In this section, we discuss three factors that govern simulation time and several trade-offs associated with the system.

Dynamic Range

The most straightforward factor that may affect simulation speed is the dynamic range of variables. Let's first consider the range of the parameters that represent reaction rate constants. Since rate constants are proportional to I_{kr}/C and I_{kr} is produced by a digital-to-analog converter (DAC), their range is determined by the range of electric current which the DACs can create (see Section 2.2.7). The DACs are able to produce at least five orders of magnitude of current levels (100 pA–10 μA), so if C is fixed, the dynamic range of reaction rate constants is 100 dB. This is sufficient to model and simulate most gene-protein networks [3, 92], with time mappings that enable fast simulation. Besides, to add more flexibility to the system, the current implementation of the cytomorphic chips allows the placement of external capacitors to increase the value of C (originally, 10–50 pF). Unless the leakage current of external capacitors goes too high, this essentially eliminates the lower bound of rate constants. However, since it takes up the pins of the chips, when the system is scaled, this feature should only be permitted for a portion of building blocks.

Next, the dynamic range of the variables representing molecular concentration is also determined by the output current range of DACs, as well as the operating range of the current-mode circuits in the system. Thus, it also spans five orders of magnitude,

which again is wide enough to model the majority of gene-protein networks [3, 92]. Furthermore, if there exist species with molecule levels that go beyond the range, they can separately be mapped into smaller levels of current within the range. In this case, the associated K_D (dissociation constant) levels should also be adjusted with the same scale factor. If these species react with other species with a different magnitude mapping, the amount of products should be properly scaled before being subtracted from the amount of reactants, using current-mode multipliers in the chip. In fact, if the molecule copy number of these species is so high that the number of products is relatively negligible, these scaling and subtraction operations may be omitted. The method described above is similar to automatic gain control, a widely used technique in analog circuit design, which allows the system to operate for wider ranges of molecular concentrations.

With the wide ranges of variables and additional flexibilities, it seems that the dynamic range rarely confines simulation speed and there is little motivation to improve it. Nevertheless, there exists another important motivation. That is, the dynamic range influences the other two speed-limiting factors, so wider ranges may lead to faster simulation speed. This will be explained in the following sections.

How big is the room for improving the dynamic range? First of all, since we use a BiCMOS process technology which allows us to use bipolar transistors that exhibit nearly ideal exponential current-voltage characteristics over a very wide range of current, our current-mode cicuits yield accurate results even with higher current levels. In addition, for a bipolar transistor, a 10-fold increase in the collector current results in less than a 0.1 V increase in its base-emitter voltage. Thus, as long as the biasing circuits are slightly modified such that all bipolar transistors operate in the forward-active mode, the current-mode circuits will operate faithfully with higher levels of current. On the other hand, the current mirrors in the building blocks are implemented with MOS transistors. At higher (above-threshold) current levels, as their drain current increases, their gate-source voltage increases rapidly and the transistors run out of the voltage headroom[8]. Simulation results show that without

[8]This effect of course may be mitigated by designing current mirrors with bipolar transistors.

much change in design, the current mirrors operate reliably up to 100 μA of current. In fact, current levels higher than 100 μA may be impractical in terms of power consumption of the chip. Hence, this level seems to be a reasonable target for future chips. Modifying other components of the system (e.g., DACs and ADCs) for the maximum operating current of 100 μA will also be a straightforward task.

Noise Generation

In Section 4.3, it is described that at a magnitude mapping of 1 molecule = 1 nA, the highest reaction rate constant for the chip is kr = 3.2×10^3 s^{-1}, and this is determined by the programmable range of the random clock frequency. Currently, its upper bound is limited because of a 700 fF capacitor placed at the output of the thermal noise amplifier to low-pass filter the amplified noise. This low-pass filtering is done to produce appropriate levels of noise for sufficiently small rate constants. If we remove the capacitor, the capacitance seen at the output will reduce from 700 fF to around 5.4 fF, thereby increasing the cutoff frequency of the filter by 130x. Meanwhile, the lower bound of the frequency can be preserved by making the value of this capacitance digitally programmable. Taking into account the fact that we can use better process technologies (which offer smaller parasitic capacitances and faster speed), when estimated conservatively, we will be able to achieve at least a 100x higher random clock frequency. This means that reaction rate constants can be increased as high as kr = 3.2×10^5 s^{-1}.

It is also explained in Section 4.3 as to how to leverage an extra dynamic range of variables to decrease simulation time, via manipulating both time and magnitude mappings. If we earn a 10x wider dynamic range with the modifications described in the previous section, and alter the mappings as necessary, the rate constants much higher than kr = 3.2×10^5 s^{-1} will be available. This will enable faster chip simulation when molecule levels are low.

Stability

For the protein block introduced in Section 3.2, let's consider a case where it is configured to model a simple reversible binding reaction (A+B⇌C), i.e., Dfree/KDrv=1, ratC=kdeg=0, n=1, and Cfree=Ctot. Let's also assume that the concentration of species B is much higher than A (i.e., [Atot]≪[Btot] and [Bfree]≈[Btot]). In the negative feedback loop formed around A, there exist two major poles affecting its stability—one "dominant" pole created at the capacitor node of the integrator and the other "parasitic" pole created by a current mirror which mirrors Afree before it enters a multiplier. To ensure good stability of the loop with a reasonable phase margin, the frequency of the parasitic pole should be higher than the crossover frequency (where the magnitude of the loop gain becomes 1). Hence, the criteria for stability is given by

$$\left(\frac{I_{Btot}}{I_{KDfw}}\right)\left(\frac{I_{kr}}{C}\right) < \frac{I_{Afree}}{C_{par}} \tag{4.1}$$

where C is the value of the integrator capacitor and C_{par} is the parasitic capacitance seen at the gate node of the current mirror.

The above expression indicates that the stability of the loop is influenced by the combined effect of multiple parameters and variables and more importantly, that stability is directly related to simulation speed. That is, the upper bound of the term I_{kr}/C, proportional to the reaction rate constant[9], is set by the above criteria. For example, if I_{Btot}/I_{KDfw} is 100, I_{Afree} is 1 nA, and C_{par} is 1 pF, the upper bound of I_{kr}/C becomes 10 (which corresponds to k_r≈400 s^{-1}). Thus, if the three conditions, 1) high I_{Btot}/I_{KDfw}, 2) high I_{kr}/C, and 3) low I_{Afree}, occur at the same time, a stability threat is posed. In this case, to meet the criteria, the time mapping ought to be adjusted such that we simulate with lower values of I_{kr}/C. It should also be noted that this criteria is meaningful only when the feedback loop exists. If reactants in a reaction are not consumed (as in transcription or translation), no feedback loop exists, and stability does not become an issue.

Let's now examine the implications of each term. First, I_{Btot}/I_{KDfw} is a term

[9]The reaction rate constant, k_r, is equal to $\frac{I_{kr}}{C\phi_t}$. Thus, $\frac{I_{kr}}{C} = 1$ yields approximately $k_r = 40$ s^{-1}.

which sets the percentage of species A that is used up to produce the product C, since $\frac{I_{Ctot}}{I_{Atot}} = \frac{I_{Btot}/I_{KDfw}}{1+I_{Btot}/I_{KDfw}}$ at steady state. For example, when I_{Btot}/I_{KDfw} is 100, 99% of A is consumed to produce C. In most biological systems, the typical range of I_{Btot}/I_{KDfw} is 0.1–10 and it rarely goes higher than 100. The higher I_{Btot}/I_{KDfw} goes, the smaller I_{kr}/C should be to fulfill the stability criteria. It therefore limits how fast chip simulation can run.

Second, as far as I_{Afree} is concerned, what matters is the minimum level of I_{Afree}, which unfortunately happens when I_{Btot}/I_{KDfw} is highest in most cases. Obviously, the miminum level of I_{Afree} is affected by the magnitude mapping. For example, if 1 molecule is mapped into 1 nA and the molecule number changes between 1–1,000 molecules, $\min(I_{Afree})$ is 1 nA. To achieve a higher speedup, it is desirable that this $\min(I_{Afree})$ is higher, which can be done by changing the magnitude mapping such that 1 molecule is mapped into higher levels of current. Thus, a higher speedup can be achieved by simply consuming more power (i.e., speed-power trade-off). This also implies that efforts to increase the dynamic range (as described in the 'Dynamic Range' section above) will make additional room for boosting simulation speed. Lastly, a clever use of automatic gain control (e.g., applying separate magnitude mappings for low-copy-number species) will allow for faster simulation for the given dynamic range of the chip.

Third, the parasitic capacitance, C_{par}, has to be small to obtain a higher speedup. C_{par} is proportional to the transistor size of the current mirror, which is in turn determined by the design requirement regarding accuracy. Ideally, the two transistors used in the current mirror to copy current levels should have the same electrical properties. However, random mismatches exist between them due to process variations (e.g., statistical variation in the scattering of dopant atoms). A way to reduce these mismatches is to increase the size of the transistors. Thus, the desired accuracy determines the transistor size, the transistor size sets the parasitic capacitance size, and finally, the parasitic capacitance size affects simulation speed via the stability criteria in (4.1). This reveals a speed-precision and an area-precision trade-off existing in our design.

In summary, the goal to perform faster simulation can be accomplished by sacrificing power or precision. However, in fact, there are more possible ways to tackle the speed limit caused by stability. Here we list a few:

1. The value of C_{par} can be reduced by two different methods. First, if we switch from a 0.35 μm to a 0.13 μm process technology, the matching constant for the threshold voltage (A_{VT}) of an MOS transistor will improve approximately by 3x [98]. This means that the required transistor area (i.e., W×L) to yield the same matching characteristics will be reduced by 9x. Therefore, the value of C_{par} will also decrease by 9x. Second, the effective C_{par} can be reduced by inserting a buffer transistor between the drain and gate of the input transistor of the current mirror (similar to M_3 in Figure 2-19). Then, on behalf of I_{Afree}, the buffer will charge the gate node. Assuming that the current through the buffer transistor is sufficiently high, the frequency of the parasitic pole increases by a factor of C_{par}/C_D, where C_D is the capacitance seen at the drain node. For our transistor sizing, C_{par} is ∼1 pF and C_D is ∼2.4 fF. This corresponds to more than a 400x improvement. It should be emphasized that to fully take advantage of this technique, buffers should be placed in every node which might create a low-frequency parasitic pole (especially, potential high-impedance nodes created by small current). Considering the combined effect of those parasitic poles, we believe that the effective value of C_{par} can be reduced by at least 100x.

2. The lower bound of I_{Afree} can be limited by artificially adding a small current I_{basal}. In normal conditions where I_{basal} is negligible compared to I_{Afree}, it causes little impact on the calculation result of the block. However, when I_{Btot}/I_{KDfw} surges, the level of I_{Afree} decreases until it is limited by I_{basal}. Then, what appears at the output (I_{Ctot}) at steady state is $I_{basal}I_{Btot}/I_{KDfw}$ instead of $I_{Afree}I_{Btot}/I_{KDfw}$ (which is smaller). In cases where this causes an intolerable amount of error, the same level of current may be subtracted from the output node to cancel it out. In this case, extra care is needed to minimize the mismatch between the two currents. Furthermore, the scale factor for the magnitude

mapping affects the size of the error. That is, if higher levels of current are used to represent molecule numbers, the effect of I_{basal} on the output will be smaller. Thus, having a wide dynamic range of operating current is helpful in implementing this strategy as well.

3. A limiting function can be implemented to prevent excessively high values of I_{Btot}/I_{KDfw}. Such values that appear in mathematical models may sometimes be unphysically high in real biological systems, in which case the limiting function can be useful. Most molecules in biochemical reactions are rarely above $10K_D$, likely for reasons of energy consumption in the cell [110], which may naturally limit parameters to physical rather than artificial mathematical modeling ranges. Besides, even if they are physical values consistent with experimental results, the function does not significantly change simulation results in most cases. For example, when I_{Btot}/I_{KDfw} increases from 100 to 1000, the value of $\frac{I_{Ctot}}{I_{Atot}} = \frac{I_{Btot}/I_{KDfw}}{1+I_{Btot}/I_{KDfw}}$ changes from 0.99 to 0.999. This is less than a 1% difference, likely to be irrelevant in most biological situations. The limiting function can be implemented digitally in FPGAs or as a special circuitry in cytomorphic chips.

4. Under certain conditions, fast dynamics can be neglected. For example, let's consider a simple 3-reaction network, $\mathrm{A} \underset{k_2}{\overset{k_1}{\rightleftharpoons}} \mathrm{B} \xrightarrow{k_3} \varnothing$. If the transformation between A and B occurs on a much faster timescale than the degradation of B (i.e., $k_1 + k_2 \gg k_3$), the transformation reactions rapidly reach chemical equilibrium. In this case, we can assume that it reaches equilibrium "instantaneously" and the equilibrium is maintained at all times. Then, the concentration of B becomes

$$[B] = \frac{k_1}{k_2}[A] \tag{4.2}$$

and the degradation rate of B is given by

$$\frac{d[B]}{dt} = k_3[B] = \frac{k_1 k_3}{k_2}[A]. \tag{4.3}$$

This is called the rapid equilibrium assumption. Once we use it to remove fast timescales of a model, we can change the time mapping to run faster chip simulation.

5. Alternatively, fast dynamics can be separately mapped into slower timescales of the chip. That is, in the above example, as long as the condition of $k_1 + k_2 \gg k_3$ is satisfied, the rapid equilibrium assumption is valid in terms of the accuracy of the resulting dynamics. Thus, in cases where k_1 and k_2 are significantly greater than k_3, they can be reduced by the same ratio, as long as the error it casuses is within a tolerable range. Just like the previous method, such adjustments made in the fast dynamics can lead to faster simulation.

Therefore, we have at least seven different strategies to alleviate the speed constraint posed by stability. A compiler may be able to analyze given computational models of biological networks to choose which techniques to use for which reactions. Taken together, it seems feasible to improve our circuit such that $\max(I_{Btot}/I_{KDfw})$ is 100, $\min(I_{Afree})$ is 10 nA, and C_{par} is 10 fF. Then, the upper bound of I_{kr}/C becomes 10,000, corresponding to $k_r \approx 4 \times 10^5$ s^{-1}.

4.4.3 Expected Performance After Improvements

We have shown in the previous section that automatic gain control techniques and commonly used circuit improvements in our current-mode circuits, DACs, noise generators, and biasing circuits can bring about at least 100x faster simulation speed. To create supercomputing power, such efforts should be accompanied by strategies to integrate larger quantaties of computational units in a chip. The most straightforward way to accomplish it is to switch to a process technology with a smaller feature size. Using a commercially available 0.13 μm BiCMOS process offered by IHP or GlobalFoundries, the area occupied by a unit computational block will be reduced approximately by 4–9x, while the accuracy of computation is preserved [98]. Another straightforward method is to build bigger size chips. Since the die size of a protein chip is 4.3 mm $\times$ 4.0 mm (17.2 mm^2), a 25 mm $\times$ 25 mm (625 mm^2) chip will give

us 36x more area. We can also employ other area-saving techniques including 1) using a minimum number of shared DACs and dynamic current mirrors to periodically update parameters and variables, 2) reducing the number of parameters by sharing several global parameters, and 3) adding simplified blocks that are less flexible but smaller in size. Although it is challenging to provide an accurate estimation at this point, it seems possible to integrate more than 100x greater number of computational units in a chip.

In light of the 700x peak speedup over COPASI demonstrated in Section 4.3, a 100x improvement by addressing the speed-limiting factors in Section 4.4.2, and a 100x increase in the system scale as described in the previous paragraph, a simple calculation of the peak speedup factor for the improved system yields $700 \times 100 \times 100 = 7 \times 10^6$. With this speedup, simulation of a reaction network which takes a year in COPASI will take 4.5 seconds in the board. Note that this calculation excludes the fact that software simulation time increases faster than proportionally as the number of reactions increases.

It is worth noting that there are ongoing efforts by researchers to utilize huge quantities of chips to realize massively parallel simulation. One such effort is the SpiNNaker Project which aims at building a neuromorphic system with more than a million computational cores (18 cores per chip) [33, 91]. Scaling our system to such a scale would entail great challenges regarding power, area, data communication, and so forth, but it would undoubtedly amplify the capabilities of the system. The design of on-chip molecular data packet routers and other circuitry that enable direct chip-to-chip communication will help solve potential communication bottlenecks [58, 87, 93].

4.5 Conclusion

The value of our hybrid analog-digital cytomorphic computer will culminate in running very-large-scale stochastic simulations. To elucidate the feasibility of increasing simulation scale, we implemented a proof-of-concept board which can control and simulate with a large number of chips in a parallel fashion. The board is stuffed with

10 gene chips and 10 protein chips, along with an FPGA, and has a computing power capable of modeling up to 1,400 biochemical reactions. It was demonstrated that the board enables up to a 700x speedup over COPASI, an efficient biochemical system simulator, and 30,000x over MATLAB.

More importantly, is was shown that the simulation speed and accuracy of the system is immune to the network scale or molecular population size. This suggests that our system may be well-suited for fast stochastic simulation of multiscale models. In addition, the three factors that limit the speedup of our system—dynamic range, noise generation, and stability—were examined in detail. Finally, based on the lessons learned through implementing this demonstration, we proposed several well-known and established circuit techniques to improve simulation speed and scale, which is expected to create more than a million speedup.

Chapter 5

Conclusions

5.1 Summary

This thesis project initiated after realizing the critical needs for a high-speed computation tool in systems and synthetic biology and how pervasive software methods are ill-suited for the simulation of large-scale biochemical reaction networks that include parallel, nonlinear, non-modular, stochastic, and feedback dynamical effects. We saw that it is mainly because conventional methods have been performing computation to study networks that compute with a totally different mechanism. Therefore, our goal was to create a cell-inspired electronic system which can address this issue. Here are the summary of the contributions we made to this area:

1. Leveraging the mathematical similarities between chemistry and electronics and the rich analog basis functions inherent in transistor circuits, we designed seven efficient building blocks using BiCMOS log-domain current-mode circuits. They can quantitatively represent fundamental bio-molecular functions in cells, which include complex behaviors such as biochemical binding with non-modular or loading effects, cooperative binding, probabilistic gene transcription, stochasticity, etc. In doing so, they do not make on/off digital approximations but fully capture analog effects. The use of bipolar junction transistors enables these circuits to function over more than five orders of magnitude of molecular

concentration. The seven building-block circuits serve as universal and versatile computational units throughout the system.

2. By composing the building-block circuits, hybrid analog-digital integrated-circuit cytomorphic gene and protein chips are constructed in a 0.35 μm BiCMOS process. Both chips feature digitally programmable network connectivity and wide-dynamic-range reaction-rate constants and initial conditions, which are flexible enough to model mass-action kinetics of arbitrary zeroth, first, and second-order reactions in gene-protein networks. A scheme to use "total" and "free" variables and corresponding building-block configuration mechanisms are described, which allows for transient and steady-state solutions of differential equations in the presence of transistor mismatch. The use of on-chip ADCs and DACs enables the chips to interact digitally via an FPGA and thus provides robustness and scalability to the system. The deterministic and stochastic simulations of published models of biological networks such as a repressilator, a p53 signaling pathway, and a glycolysis pathway produce chip data that show good quantitative agreement with those obtained from conventional software simulations. The noise generators incorporated in the system enable stochastic simulations by amplifying analog thermal noise that is consistent with Gillespie simulations.

3. A proof-of-concept system which integrates and orchestrates hardware (i.e., cytomorphic chips and FPGAs), software, analog, and digital computations is implemented. An array of cytomorphic chips are mounted on a PCB, along with an FPGA, to create a computing power capable of simulating networks involving up to 1,400 biochemical reactions. Experimental results show that with the capacity of our current implementation, the system can achieve a peak speedup of 700x over COPASI and 30,000x over MATLAB. Through this demonstration, it is shown that the system has not only a scalable architecture amenable to massively parallel computation but also a fundamental structural advantage over classic purely digital solutions. That is, an increase in network

scale or molecular population size does not require the simulation speed or accuracy to be sacrificed. This finding suggests that the system may be able to provide tremendous performance advantages when further scaled, particularly for simulation of multiscale networks.

4. Finally, strategies to upgrade the speed and scale of the system are presented. For this, the three speed-limiting factors of the system—dynamic range, noise generation, and stability—are scrutinized and practical solutions are proposed for each of them. It is shown that those solutions, combined with circuit techniques to miniaturize our system, will enable further orders of magnitude of speedup, estimated to be more than a million fold.

Table 5.1: Comparison of Various Approaches for Biological Simulations

Property	**General-Purpose Processors (e.g., CPU)**	**Special-Purpose Processors (e.g., GPU)**	**Reconfigurable Hardware (e.g., FPGA)**	**Custom Digital ICs**	**Custom Analog ICs**	**Our Hybrid System**
Speed	+	++	++	++	+++	+++
Power Efficiency	+	++	++	++	+++	+++
Flexibility	+++	+++	+++	++	+	++
Scalability	++	++	++	++	+	+++
Dynamic Range	+++	+++	+++	+++	+	++
Precision	+++	+++	+++	+++	+	++
Ability to Parallelize	+	++	++	++	+++	+++
Ease of Development	+++	++	++	+	+	+

Table 5.1 shows the comparison of important characteristics of various approaches for solving the computational challenge that this thesis deals with. Although somewhat generalized, it gives an overview of pros and cons of each method. Overall, the solution this thesis proposes can offer high speed and power efficiency and reasonable flexibility, dynamic range, and precision, at the expense of development time and cost. The idea of direct mapping in the continuous domain allows our system to achieve parallelism naturally and thus maintain high speed even for large-scale simulations.

In a sense, this work is an endeavor to bridge the huge performance gap between human-engineered systems and biological systems [110, 116]: To achieve high performance, our system mimics the wise computational paradigm of cells—the use of efficient analog computing units to carry out massively parallel computation. On the other hand, our system is built to better understand or synthesize complex biological networks and in turn gain more wisdom from them, by enhancing the ability to simulate, analyze, and design such networks. We hope that this positive feedback can give a fresh insight into computational principles that are generally applicable to the field of biology and electronics.

5.2 Future Work

5.2.1 Hardware Enhancements

The cytomorphic chips have evolved over years through five iterations of chip fabrication, as shown in Figure 5-1. The first two chips were fabricated in a UMC 0.18 μm CMOS process and the next three chips were redesigned and fabricated in an AMS 0.35 μm BiCMOS process. Yet, there are still lots of opportunities to improve our hardware:

1. Several circuit techniques can be used to tackle speed-limiting factors and enhance simulation speed. They include increasing the dynamic range of variables, increasing the range of random clock frequency, reducing the effective parasitic capacitance, etc. Refer to Section 4.4.2 for detailed explanations.

2. Integrating a greater number of computational units in a chip is one of the keys to creating higher computing power for large-scale simulations, as described in Section 4.4.3. It can be accomplished by implementing the ideas of shared DACs (using dynamic current mirrors), global parameters, small-sized static blocks, etc. We can also take advantage of better transistor matching, smaller transistor size, and more number of metal layers that modern process technologies can offer.

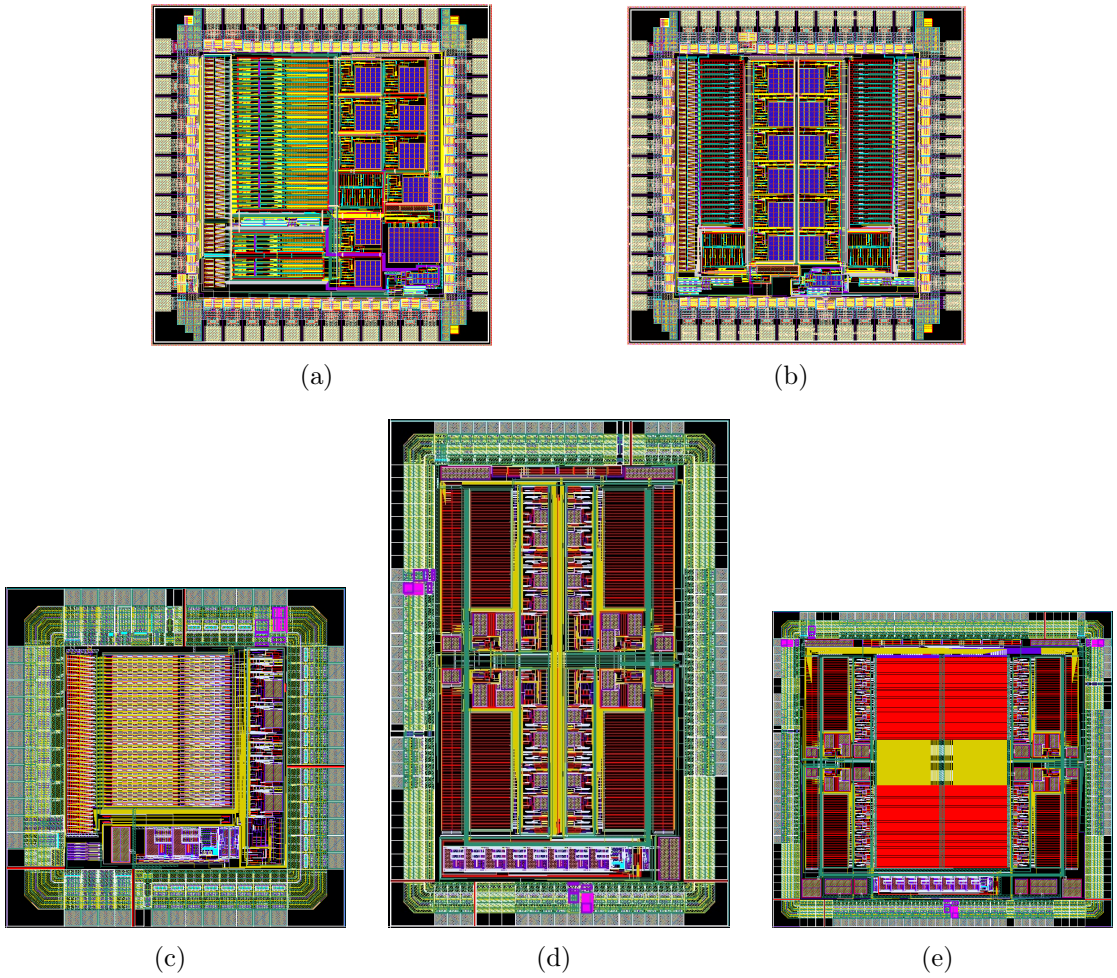


Figure 5-1: Evolution of the cytomorphic chips, fabricated in (a)(b) a CMOS process and (c)-(e) a BiCMOS process.

3. Eventually, it will be required to create an improved platform where a huge number of cytomorphic chips and FPGAs can systematically be operated in parallel (see Section 4.4.3). To prevent a potential communication bottleneck, an on-chip router which enables molecular data packets to be sent via direct chip-to-chip connections may be developed. To save chip area, communication channels may be divided into local and global channels, dedicated to interactions among neighboring building-block circuits and distant circuits, respectively.

4. To increase computation accuracy, various error reduction techniques such as fully differential circuitry, common-mode feedback, autozeroing (to cancel offsets in current-mode circuits), and lookup tables (to compensate for the variability in ADCs and DACs) can be employed.

5. The structure of the noise generator in the system can be improved to produce proper levels of noise for lower levels of SNR. This will open up new opportunities for very-low-copy-number stochastic simulations.

5.2.2 Software Enhancements

A computer is a suitable platform for converting software models of biological networks to the chip representation and running rich software algorithms for network analysis and parameter searching. These operations can be carried out in the following ways:

1. Without a compiling tool, it is in fact cumbersome and time-consuming to set up a chip simulation by manually interpreting software models and configuring the chip. A typical user of our simulation tool would not want to handle such tasks. The development of a compiler will drastically reduce this burden, enabling greater ease of use, especially for large-scale simulations.

   Thus, the primary role of the compiler is to inspect biological networks represented in standard formats such as Systems Biology Markup Language (SBML) [56] to generate configurations for the building-block instances of our system.

Based on the given specifications of the chip, it first creates a netlist describing the optimal set of building blocks to use, connectivity among them, and the parameters for each block. In doing so, the compiler carries out signal magnitude mapping such that the dynamic range of variables in the chip reliably covers that in the target biological network. Additionally, the compiler determines the time mapping such that the chip performs simulation at a maximum speed. Finally, the compiler converts the netlist to a bitstream to be sent to the FPGA. Therefore, thanks to this compiler, a user will ultimately be able to write a model in SBML or download one from BioModels Database [59] and immediately run it on the chip.

A preliminary method to do compilation has already been explored at the Program Analysis and Compilation Group at MIT. The details of the idea and implementation of the compiler can be found in [1]. Its compatibility with our system and actual simulation with hardware are yet to be tested.

2. Although more and more pathways and kinetic parameters are being discovered and becoming publicly available [59, 62, 65, 88], a ton of them still remain unknown [14, 53, 105]. Thus, to further enhance the utility of our high-throughput computing device, it is important to explore ways to use it in conjunction with existing searching and learning algorithms running on a digital computer. It may also be needed to develop an optimized method that makes best use of our unique hybrid analog-digital environment. This effort may be synergistically combined with other useful approaches to finding parameters, such as improving experimental techniques for measuring parameters [48, 76, 126] and predicting parameters without experimental analysis [7, 43, 44, 72, 124].

5.2.3 Future Applications

The system we have built has potential to be used for many applications:

1. Various large-scale gene-protein networks can be simulated for exploring network topologies, discovering parameters, analyzing complex cell functions, pre-

dicting and preventing diseases, and optimizing drug dosage. We are currently looking at several pathways related to matabolism, cancer, cell cycle, and chemotaxis. Ways to incorporate reaction-diffusion equations may also be explored, to account for space-dependent effects.

2. Our stochastic simulator can be utilized to study rare stochastic effects that may lead to macroscopic consequences. They include drug resistance (due to random mutations), ageing (due to the stochastic failures of DNA damage repair), phenotypic heterogeneity (due to noisy gene expression), chemotaxix (due to the random switching between tumble and swim phases), and the onset of Alzheimer's disease (due to the random formation of plaques and tangles) [69, 101, 125, 137].

3. Our system can be used to facilitate the design of synthetic circuits in living cells. The synthetic biology community has been finding the "design rules" for engineering synthetic circuits [9, 10, 102], appreciating the potential of analog approaches [17, 110], and recognizing the importance of understanding and exploiting the stochasticity in cells [102, 128, 141]. In the near future, it would be far easier to design and assemble synthetic circuit "modules" to construct complex large-scale systems in an automated fashion. These systems may be "uploaded" into target cells to regulate their functions, by interacting with existing networks such as cell-cycle, cancer, and metabolic pathways [61, 107]. All of these trends will increase the demand for a powerful tool for design, analysis, and simulation of biological networks. Our cytomorphic system may have an opportunity to play an important role in this area, as CAD tools have played in the semiconductor industry.

4. Owing to the ability of cytomorphic chips to model various forms of linear and nonlinear differential equations (e.g., the one in equation (1.5)), the system may be used as a general-purpose stochastic differential equation solver. Thus, the application of the system may be further extended to other fields, including physics, engineering, finance, and economics.

It seems clear that this work has an interdisciplinary nature as well as plenty of room for improvements. Hence, further innovations can be best achieved by the collaborative efforts of experts in diverse fields including circuit design, synthetic and systems biology, computer science, and numerical simulation.

Bibliography

[1] Sara Achour, Rahul Sarpeshkar, and Martin C. Rinard. Configuration Synthesis for Programmable Analog Devices with Arco. In *Proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2016, pages 177–193, New York, NY, USA, 2016. ACM.

[2] G. K. Ackers, A. D. Johnson, and M. A. Shea. Quantitative model for gene regulation by lambda phage repressor. *Proceedings of the National Academy of Sciences*, pages 1129–1133, 1982.

[3] U. Alon. *An Introduction to Systems Biology: Design Principles of Biological Circuits*. Chapman and Hall, Boca Raton, Florida, 2007.

[4] R. Ananthanarayanan, S.K. Esser, H.D. Simon, and D.S. Modha. The cat is out of the bag: cortical simulations with 10^9 neurons, 10^{13} synapses. In *High Performance Computing Networking, Storage and Analysis, Proceedings of the Conference on*, pages 1–12, Nov 2009.

[5] A. G. Andreou, K. A. Boahen, P. O. Pouliquen, A. Pavasovic, R. E. Jenkins, and K. Strohbehn. Current-mode subthreshold MOS circuits for analog VLSI neural systems. *IEEE Transactions on Neural Networks*, 2(2):205–213, March 1991.

[6] M. W. Baker and R. Sarpeshkar. Low-Power Single-Loop and Dual-Loop AGCs for Bionic Ears. *IEEE Journal of Solid-State Circuits*, 41(9):1983–1996, September 2006.

[7] Michael A. Beer and Saeed Tavazoie. Predicting Gene Expression from Sequence. *Cell*, 117(2):185–198, April 2004.

[8] B.V. Benjamin, Peiran Gao, E. McQuinn, S. Choudhary, AR. Chandrasekaran, J.-M. Bussat, R. Alvarez-Icaza, J.V. Arthur, P.A Merolla, and K. Boahen. Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations. *Proceedings of the IEEE*, 102(5):699–716, May 2014.

[9] Jennifer A. N. Brophy and Christopher A. Voigt. Principles of genetic circuit design. *Nature Methods*, 11(5):508–520, May 2014.

[10] Jarred M. Callura, Charles R. Cantor, and James J. Collins. Genetic switchboard for synthetic biology applications. *Proceedings of the National Academy of Sciences*, 109(15):5850–5855, April 2012.

[11] D. Ewen Cameron, Caleb J. Bashor, and James J. Collins. A brief history of synthetic biology. *Nature Reviews Microbiology*, 12(5):381–390, May 2014.

[12] Stefano Cardinale and Adam Paul Arkin. Contextualizing context for synthetic biology–identifying causes of failure of synthetic biological systems. *Biotechnology Journal*, 7(7):856–866, July 2012.

[13] Han-Yu Chuang, Matan Hofree, and Trey Ideker. A decade of systems biology. *Annual Review of Cell and Developmental Biology*, 26:721–744, 2010.

[14] John Cole, Michael J. Hallock, Piyush Labhsetwar, Joseph R. Peterson, John E. Stone, and Zaida Luthey-Schulten. Stochastic Simulations of Cellular Processes: From Single Cells to Colonies. In *Computational Systems Biology*, pages 277–293. Elsevier, 2014.

[15] G. E R Cowan, R.C. Melville, and Y. Tsividis. A vlsi analog computer/digital computer accelerator. *Solid-State Circuits, IEEE Journal of*, 41(1):42–53, Jan 2006.

[16] R. Danial, S. S. Woo, L. Turicchia, and R. Sarpeshkar. Analog transistor models of bacterial genetic circuits. In *Proceedings of the 2011 IEEE Biological Circuits and Systems (BioCAS) Conference*, pages 333–336, San Diego, CA, November 2011.

[17] Ramiz Daniel, Jacob R. Rubens, Rahul Sarpeshkar, and Timothy K. Lu. Synthetic analog computation in living cells. *Nature*, 497(7451):619–623, May 2013.

[18] Hidde de Jong. Modeling and simulation of genetic regulatory systems: a literature review. *Journal of Computational Biology: A Journal of Computational Molecular Cell Biology*, 9(1):67–103, 2002.

[19] Domitilla Del Vecchio, Alexander J Ninfa, and Eduardo D Sontag. Modular cell biology: retroactivity and insulation. *Molecular Systems Biology*, 4(1), 2008.

[20] T. Delbruck and A Van Schaik. Bias current generators with wide dynamic range. In *Circuits and Systems, 2004. ISCAS '04. Proceedings of the 2004 International Symposium on*, volume 1, pages I–337–I–340 Vol.1, May 2004.

[21] V. Douence, A. Laflaquière, G. Le Masson, T. Bal, Thierry Bal, and Gwendal Le Masson. Analog electronic system for simulating biological neurons. In *in Proceedings of the International Work-Conference on Artificial and Natural Neural Networks, IWANN*, pages 188–197, 1999.

[22] J.E. Duarte, J. Velasco-Medina, and P.A Moreno. Hardware simulation of yeast glycolytic oscillations. In *Bioinformatics and Biomedicine Workshops (BIBMW), 2011 IEEE International Conference on*, pages 396–399, Nov 2011.

[23] D. Dupeyron, S. Le Masson, Y. Deval, G. Le Masson, and J.-P. Dom. A BiCMOS implementation of the Hodgkin-Huxley formalism. In *, Proceedings of Fifth International Conference on Microelectronics for Neural Networks, 1996*, pages 311–316, February 1996.

[24] R. Timothy Edwards and Gert Cauwenberghs. Synthesis of Log-Domain Filters from First-Order Building Blocks. *Analog Integrated Circuits and Signal Processing*, 22(2-3):177–186, March 2000.

[25] Tom Ellis, Xiao Wang, and James J. Collins. Diversity-based, model-guided construction of synthetic gene networks with predicted functions. *Nature Biotechnology*, 27(5):465–471, May 2009.

[26] Michael B. Elowitz and Stanislas Leibler. A synthetic oscillatory network of transcriptional regulators. *Nature*, 403(6767):335–338, January 2000.

[27] Michael B. Elowitz, Arnold J. Levine, Eric D. Siggia, and Peter S. Swain. Stochastic gene expression in a single cell. *Science*, 297(5584):1183–1186, 2002.

[28] Drew Endy and Roger Brent. Modelling cellular behaviour. *Nature*, 409(6818):391–395, January 2001.

[29] A. D. Fokker. Die mittlere Energie rotierender elektrischer Dipole im Strahlungsfeld. *Annalen der Physik*, 348(5):810–820, January 1914.

[30] Claire M. Fraser, Jeannine D. Gocayne, Owen White, Mark D. Adams, Rebecca A. Clayton, Robert D. Fleischmann, Carol J. Bult, Anthony R. Kerlavage, Granger Sutton, Jenny M. Kelley, Janice L. Fritchman, Janice F. Weidman, Keith V. Small, Mina Sandusky, Joyce Fuhrmann, David Nguyen, Teresa R. Utterback, Deborah M. Saudek, Cheryl A. Phillips, Joseph M. Merrick, Jean-Francois Tomb, Brian A. Dougherty, Kenneth F. Bott, Ping-Chuan Hu, Thomas S. Lucier, Scott N. Peterson, Hamilton O. Smith, Clyde A. Hutchison, and J. Craig Venter. The minimal gene complement of mycoplasma genitalium. *Science*, 270(5235):397–404, 1995.

[31] D. R. Frey. State-space synthesis and analysis of log-domain filters. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 45(9):1205–1211, September 1998.

[32] A. Funahashi, Y. Matsuoka, A. Jouraku, M. Morohashi, N. Kikuchi, and H. Kitano. CellDesigner 3.5: A Versatile Modeling Tool for Biochemical Networks. *Proceedings of the IEEE*, 96(8):1254–1265, August 2008.

[33] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana. The SpiNNaker Project. *Proceedings of the IEEE*, 102(5):652–665, May 2014.

[34] Socorro Gama-Castro, Heladia Salgado, Alberto Santos-Zavaleta, Daniela Ledezma-Tejeida, Luis Muñiz-Rascado, Jair Santiago García-Sotelo, Kevin Alquicira-Hernández, Irma Martínez-Flores, Lucia Pannier, Jaime Abraham Castro-Mondragón, Alejandra Medina-Rivera, Hilda Solano-Lira, César Bonavides-Martínez, Ernesto Pérez-Rueda, Shirley Alquicira-Hernández, Liliana Porrón-Sotelo, Alejandra López-Fuentes, Anastasia Hernández-Koutoucheva, Víctor Del Moral-Chávez, Fabio Rinaldi, and Julio Collado-Vides. RegulonDB version 9.0: high-level integration of gene regulation, coexpression, motif clustering and beyond. *Nucleic Acids Research*, 44(Database issue):D133–D143, January 2016.

[35] Timothy S. Gardner, Charles R. Cantor, and James J. Collins. Construction of a genetic toggle switch in escherichia coli. *Nature*, 403(6767):339–342, January 2000.

[36] Michael A. Gibson and Jehoshua Bruck. Efficient exact stochastic simulation of chemical systems with many species and many channels. *The Journal of Physical Chemistry A*, 104(9):1876–1889, 2000.

[37] B. Gilbert. Translinear circuits: a proposed classification. *Electronics Letters*, 11(1):14–16, January 1975.

[38] Barrie Gilbert. Translinear circuits: An historical overview. *Analog Integrated Circuits and Signal Processing*, 9(2):95–118, March 1996.

[39] Daniel T Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, 22(4):403–434, December 1976.

[40] Daniel T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81(25):2340–2361, 1977.

[41] Daniel T. Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. *The Journal of Chemical Physics*, 115(4):1716–1733, 2001.

[42] Daniel T. Gillespie. Stochastic simulation of chemical kinetics. *Annual Review of Physical Chemistry*, 58:35–55, 2007.

[43] Abel González Pérez, Vladimir Espinosa Angarica, Julio Collado-Vides, and Ana Tereza Ribeiro Vasconcelos. From sequence to dynamics: the effects of transcription factor and polymerase concentration changes on activated and repressed promoters. *BMC Molecular Biology*, 10:92, 2009.

[44] Joshua A Granek and Neil D Clarke. Explicit equilibrium modeling of transcription-factor binding and gene regulation. *Genome Biology*, 6(10):R87, 2005.

[45] Nicholas J. Guido, Xiao Wang, David Adalsteinsson, David McMillen, Jeff Hasty, Charles R. Cantor, Timothy C. Elston, and J. J. Collins. A bottom-up approach to gene regulation. *Nature*, 439(7078):856–860, February 2006.

[46] Richard H. R. Hahnloser, Rahul Sarpeshkar, Misha A. Mahowald, Rodney J. Douglas, and H. Sebastian Seung. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405(6789):947–951, June 2000.

[47] Michael J. Hallock, John E. Stone, Elijah Roberts, Corey Fry, and Zaida Luthey-Schulten. Simulation of reaction diffusion processes over biologically relevant size and time scales using multi-GPU workstations. *Parallel Computing*, 40(5-6):86–99, May 2014.

[48] Petter Hammar, Mats Walldén, David Fange, Fredrik Persson, Özden Baltekin, Gustaf Ullman, Prune Leroy, and Johan Elf. Direct measurement of transcription factor dissociation excludes a simple operator occupancy model for gene regulation. *Nature Genetics*, 46(4):405–408, April 2014.

[49] Jeff Hasty, David McMillen, and J. J. Collins. Engineered gene circuits. *Nature*, 420(6912):224–230, November 2002.

[50] Edward H. Hellen, Syamal K. Dana, Jürgen Kurths, Elizabeth Kehler, and Sudeshna Sinha. Noise-aided logic in an electronic analog of synthetic genetic networks. *PLoS ONE*, 8(10):e76032, 10 2013.

[51] Edward H. Hellen, Evgenii Volkov, Jurgen Kurths, and Syamal Kumar Dana. An electronic analog of synthetic genetic networks. *PLoS ONE*, 6(8):e23286, 08 2011.

[52] Anthony D. Hill, Jonathan R. Tomshine, Emma M. B. Weeding, Vassilios Sotiropoulos, and Yiannis N. Kaznessis. SynBioSS: the synthetic biology modeling suite. *Bioinformatics*, 24(21):2551–2553, November 2008.

[53] Christoph Hold and Sven Panke. Towards the engineering of in vitro systems. *Journal of the Royal Society Interface*, 6(Suppl 4):S507–S521, August 2009.

[54] Stefan Hoops, Sven Sahle, Ralph Gauges, Christine Lee, Jürgen Pahle, Natalia Simus, Mudita Singhal, Liang Xu, Pedro Mendes, and Ursula Kummer. COPASI—a COmplex PAthway SImulator. *Bioinformatics*, 22(24):3067–3074, December 2006.

[55] T.-c.D. Huang and C.A Zukowski. Reconfigurable digital/analog processor array for the simulation of gene regulatory networks. In *Circuits and Systems, 2006. MWSCAS '06. 49th IEEE International Midwest Symposium on*, volume 1, pages 552–556, Aug 2006.

[56] M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, and the rest of the SBML Forum, A. P. Arkin, B. J. Bornstein, D. Bray, A. Cornish-Bowden, A. A. Cuellar, S. Dronov, E. D. Gilles, M. Ginkel, V. Gor, I. I. Goryanin, W. J. Hedley, T. C. Hodgman, J.-H. Hofmeyr, P. J. Hunter, N. S. Juty, J. L. Kasberger, A. Kremling, U. Kummer, N. Le Novère, L. M. Loew, D. Lucio, P. Mendes, E. Minch, E. D. Mjolsness, Y. Nakayama, M. R. Nelson, P. F. Nielsen, T. Sakurada, J. C. Schaff, B. E. Shapiro, T. S. Shimizu, H. D. Spence, J. Stelling, K. Takahashi, M. Tomita, J. Wagner, and J. Wang. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531, March 2003.

[57] T. Ideker, T. Galitski, and L. Hood. A new approach to decoding life: systems biology. *Annual Review of Genomics and Human Genetics*, 2:343–372, 2001.

[58] N. Imam and R. Manohar. Address-Event Communication Using Token-Ring Mutual Exclusion. In *2011 17th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, pages 99–108, April 2011.

[59] N Juty, R Ali, M Glont, S Keating, N Rodriguez, Mj Swat, Sm Wimalaratne, H Hermjakob, N Le Novère, C Laibe, and V Chelliah. BioModels: Content, Features, Functionality, and Use. *CPT: Pharmacometrics & Systems Pharmacology*, 4(2):55–68, February 2015.

[60] Jonathan R. Karr, Jayodita C. Sanghvi, Derek N. Macklin, Miriam V. Gutschow, Jared M. Jacobs, Benjamin Bolival, Nacyra Assad-Garcia, John I. Glass, and Markus W. Covert. A whole-cell computational model predicts phenotype from genotype. *Cell*, 150(2):389–401, July 2012.

[61] Jay D. Keasling. Synthetic biology and the development of tools for metabolic engineering. *Metabolic Engineering*, 14(3):189–195, May 2012.

[62] Ingrid M. Keseler, Amanda Mackie, Martin Peralta-Gil, Alberto Santos-Zavaleta, Socorro Gama-Castro, César Bonavides-Martínez, Carol Fulcher, Araceli M. Huerta, Anamika Kothari, Markus Krummenacker, Mario Latendresse, Luis Muñiz-Rascado, Quang Ong, Suzanne Paley, Imke Schröder, Alexander G. Shearer, Pallavi Subhraveti, Mike Travers, Deepika Weerasinghe, Verena Weiss, Julio Collado-Vides, Robert P. Gunsalus, Ian Paulsen, and Peter D. Karp. EcoCyc: fusing model organism databases with systems biology. *Nucleic Acids Research*, 41(D1):D605–D612, January 2013.

[63] J. Kim*, S. S. Woo*, and R. Sarpeshkar. Fast and precise simulation of stochastic biochemical reactions with amplified thermal noise in transistor chips. to be published.

[64] Jaewook Kim, Tae-Kwang Jang, Young-Gyu Yoon, and SeongHwan Cho. Analysis and design of voltage-controlled oscillator based analog-to-digital converter.

Circuits and Systems I: Regular Papers, IEEE Transactions on, 57(1):18–30, Jan 2010.

[65] M. D. Shaji Kumar, K. Abdulla Bava, M. Michael Gromiha, Ponraj Prabakaran, Koji Kitajima, Hatsuho Uedaira, and Akinori Sarai. ProTherm and ProNIT: thermodynamic databases for proteins and protein-nucleic acid interactions. *Nucleic Acids Research*, 34(Database issue):D204–206, January 2006.

[66] Roberta Kwok. Five hard truths for synthetic biology. *Nature News*, 463(7279):288–290, January 2010.

[67] J. Lazzaro, S. Ryckebusch, M. A. Mahowald, and C. A. Mead. Advances in neural information processing systems 1. chapter Winner-take-all Networks of O(N) Complexity, pages 703–711. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1989.

[68] S. Le Masson, A Laflaquiere, T. Bal, and G. Le Masson. Analog circuits for modeling biological neural networks: design and applications. *IEEE Transactions on Biomedical Engineering*, 46(6):638–645, June 1999.

[69] Kim Lewis. Persister cells. *Annual Review of Microbiology*, 64(1):357–372, 2010.

[70] Hong Li and Linda Petzold. Efficient Parallelization of the Stochastic Simulation Algorithm for Chemically Reacting Systems On the Graphics Processing Unit. *Int. J. High Perform. Comput. Appl.*, 24(2):107–116, May 2010.

[71] Vladimir A Likic, Malcolm J. McConville, Trevor Lithgow, and Antony Bacic. Systems Biology: The Next Frontier for Bioinformatics. *Advances in Bioinformatics*, 2010:e268925, February 2011.

[72] Xiao Liu, David M. Noll, Jason D. Lieb, and Neil D. Clarke. DIP-chip: Rapid and accurate determination of DNA-binding specificity. *Genome Research*, 15(3):421–427, March 2005.

[73] H. Lodish et al. *Molecular Cell Biology*. W. H. Freeman and Company, New York, 6 edition, 2008.

[74] J.-B. Lugagne, D.A Oyarzun, and G.-B.V. Stan. Stochastic simulation of enzymatic reactions under transcriptional feedback regulation. In *Control Conference (ECC), 2013 European*, pages 3646–3651, July 2013.

[75] E.F. MacNichol. An analog computer to simulate systems of coupled bimolecular reactions. *Proceedings of the IRE*, 47(11):1816–1820, Nov 1959.

[76] Sebastian J. Maerkl and Stephen R. Quake. A Systems Approach to Measuring the Binding Energy Landscapes of Transcription Factors. *Science*, 315(5809):233–237, January 2007.

[77] Misha Mahowald and Rodney Douglas. A silicon neuron. *Nature*, 354(6354):515–518, December 1991.

[78] S. Mandal and R. Sarpeshkar. Circuit models of stochastic genetic networks. In *IEEE Symposium on Biological Circuits and Systems (BioCAS)*, pages 109–112, Beijing, China, November 2009.

[79] S. Mandal and R. Sarpeshkar. Log-domain circuit models of chemical reactions. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2697–2700, Taipei, Taiwan, May 2009.

[80] S. Mandal and R. Sarpeshkar. Electronic system for modeling chemical reactions and biochemical processes, October 9 2012. US Patent 8,285,523.

[81] Bo Marr and Jennifer Hasler. Compiling probabilistic, bio-inspired circuits on a field programmable analog array. *Frontiers in Neuroscience*, 8, May 2014.

[82] William Mather, Matthew Bennett, Jeff Hasty, and Lev Tsimring. Delay-induced degrade-and-fire oscillations in small genetic circuits. *Phys. Rev. Lett.*, 102:068105, Feb 2009.

[83] E.E. May and R.L. Schiek. Simulating metabolism in escherichia coli k-12 – a circuit-based approach. In *Biomedical Circuits and Systems Conference, 2006. BioCAS 2006. IEEE*, pages 85–88, Nov 2006.

[84] Robert McMillan. Darpa Has Seen the Future of Computing And Its Analog, August 2012.

[85] Donald A. McQuarrie. Stochastic approach to chemical kinetics. *Journal of Applied Probability*, 4(03):413–478, December 1967.

[86] Carver Mead. *Analog VLSI and Neural Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[87] Paul A. Merolla, John V. Arthur, Rodrigo Alvarez-Icaza, Andrew S. Cassidy, Jun Sawada, Filipp Akopyan, Bryan L. Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, Bernard Brezzo, Ivan Vo, Steven K. Esser, Rathinakumar Appuswamy, Brian Taba, Arnon Amir, Myron D. Flickner, William P. Risk, Rajit Manohar, and Dharmendra S. Modha. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014.

[88] Ron Milo, Paul Jorgensen, Uri Moran, Griffin Weber, and Michael Springer. BioNumbers—the database of key numbers in molecular and cell biology. *Nucleic Acids Research*, 38(Database issue):D750–D753, January 2010.

[89] B. A. Minch. Synthesis of multiple-input translinear element log-domain filters. In *Proceedings of the 1999 IEEE International Symposium on Circuits and Systems, 1999. ISCAS '99*, volume 2, pages 697–700 vol.2, July 1999.

[90] Dharmendra S. Modha, Rajagopal Ananthanarayanan, Steven K. Esser, Anthony Ndirango, Anthony J. Sherbondy, and Raghavendra Singh. Cognitive computing. *Commun. ACM*, 54(8):62–71, August 2011.

[91] Don Monroe. Neuromorphic Computing Gets Ready for the (Really) Big Time. *Commun. ACM*, 57(6):13–15, June 2014.

[92] Uri Moran, Rob Phillips, and Ron Milo. Snapshot: Key numbers in biology. *Cell*, 141(7):1262 – 1262.e1, 2010.

[93] Javier Navaridas, Mikel Luján, Jose Miguel-Alonso, Luis A. Plana, and Steve Furber. Understanding the Interconnection Network of SpiNNaker. In *Proceedings of the 23rd International Conference on Supercomputing*, ICS '09, pages 286–295, New York, NY, USA, 2009. ACM.

[94] K. Nielsen, P. G. Sørensen, F. Hynne, and H. G. Busse. Sustained oscillations in glycolysis: an experimental and theoretical study of chaotic and complex periodic behavior and of quenching of simple oscillations. *Biophysical Chemistry*, 72(1-2):49–62, May 1998.

[95] Ertugrul M. Ozbudak, Mukund Thattai, Iren Kurtser, Alan D. Grossman, and Alexander van Oudenaarden. Regulation of noise in the expression of a single gene. *Nature Genetics*, 31(1):69–73, May 2002.

[96] Konstantinos I. Papadimitriou, Guy-Bart V. Stan, and Emmanuel M. Drakakis. Systematic computation of nonlinear cellular and molecular dynamics with low-power cytomimetic circuits: A simulation study. *PLoS ONE*, 8(2):e53591, 02 2013.

[97] Hyungman Park and A Gerstlauer. Toward a fast stochastic simulation processor for biochemical reaction networks. In *Application-Specific Systems, Architectures and Processors (ASAP), 2013 IEEE 24th International Conference on*, pages 50–58, June 2013.

[98] M.J.M. Pelgrom, H.P. Tuinhout, and M. Vertregt. Transistor matching in analog cmos applications. In *Electron Devices Meeting, 1998. IEDM '98. Technical Digest., International*, pages 915–918, Dec 1998.

[99] M. Planck. Sitzungsber. *Preuss. Akad.Wiss. Phys.Math. Kl*, 325:3, 1917.

[100] Carole J. Proctor and Douglas A. Gray. Explaining oscillations and variability in the p53-Mdm2 system. *BMC Systems Biology*, 2:75, 2008.

[101] Carole J. Proctor, Ilse Sanet Pienaar, Joanna L. Elson, and Thomas B. L. Kirkwood. Aggregation, impaired degradation and immunization targeting of amyloid-beta dimers in Alzheimer's disease: a stochastic modelling approach. *Molecular Neurodegeneration*, 7:32, 2012.

[102] Priscilla E. M. Purnick and Ron Weiss. The second wave of synthetic biology: from modules to systems. *Nature Reviews Molecular Cell Biology*, 10(6):410–422, June 2009.

[103] Muruhan Rathinam, Linda R. Petzold, Yang Cao, and Daniel T. Gillespie. Stiffness in stochastic chemically reacting systems: The implicit tau-leaping method. *The Journal of Chemical Physics*, 119(24):12784–12794, December 2003.

[104] S.M. Rezaul Hasan. A novel mixed-signal integrated circuit model for dna-protein regulatory genetic circuits and genetic state machines. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 55(5):1185–1196, June 2008.

[105] Elijah Roberts, Andrew Magis, Julio O. Ortiz, Wolfgang Baumeister, and Zaida Luthey-Schulten. Noise contributions in an inducible genetic switch: a whole-cell simulation study. *PLoS computational biology*, 7(3):e1002010, March 2011.

[106] Elijah Roberts, John E. Stone, and Zaida Luthey-Schulten. Lattice microbes: high-performance stochastic simulation method for the reaction-diffusion master equation. *Journal of Computational Chemistry*, 34(3):245–255, January 2013.

[107] Warren C. Ruder, Ting Lu, and James J. Collins. Synthetic Biology Moving into the Clinic. *Science*, 333(6047):1248–1252, September 2011.

[108] Lukasz Salwinski and David Eisenberg. In silico simulation of biological network dynamics. *Nature Biotechnology*, 22(8):1017–1019, August 2004.

[109] R. Sarpeshkar. Analog versus digital: Extrapolating from electronics to neurobiology. *Neural Computation*, 10(7):1601–1638, October 1998.

[110] R. Sarpeshkar. Analog synthetic biology. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 372(2012), March 2014.

[111] R. Sarpeshkar. Guest Editorial—Special Issue on Synthetic Biology. *IEEE Transactions on Biomedical Circuits and Systems*, 9(4):449–452, August 2015.

[112] R. Sarpeshkar, M. W. Baker, C. D. Salthouse, J. J. Sit, L. Turicchia, and S. M. Zhak. An analog bionic ear processor with zero-crossing detection. In *ISSCC. 2005 IEEE International Digest of Technical Papers. Solid-State Circuits Conference, 2005.*, pages 78–79 Vol. 1, February 2005.

[113] R. Sarpeshkar, T. Delbruck, and C.A. Mead. White noise in mos transistors and resistors. *Circuits and Devices Magazine, IEEE*, 9(6):23–29, Nov 1993.

[114] R. Sarpeshkar and M. O'Halloran. Scalable hybrid computation with spikes. *Neural Computation*, 14:2003–2038, September 2002.

[115] R. Sarpeshkar, C. Salthouse, Ji-Jon Sit, M. W. Baker, S. M. Zhak, T. K. T. Lu, L. Turicchia, and S. Balster. An ultra-low-power programmable analog bionic ear processor. *IEEE Transactions on Biomedical Engineering*, 52(4):711–727, April 2005.

[116] Rahul Sarpeshkar. Cytomorphic electronics: cell-inspired electronics for systems and synthetic biology. In *Ultra Low Power Bioelectronics: Fundamentals, Biomedical Applications, and Bio-Inspired Systems*, chapter 24. Cambridge University Press, Cambridge, 2010.

[117] J. Schemmel, D. Bruderle, A Grubl, M. Hock, K. Meier, and S. Millner. A wafer-scale neuromorphic hardware system for large-scale neural modeling. In *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pages 1947–1950, May 2010.

[118] M. Schwehm. Fast stochastic simulation of metabolic networks. *Proceedings of German Conference on Bioinformatics (GCB 2001)*, pages 223–226, 2001.

[119] Markus Schwehm. Parallel stochastic simulation of whole-cell models. *Proceedings of the 2nd International Conference on Systems Biology (ICSB 2001)*, pages 333–341, 2001.

[120] E. Seevinck. Companding current-mode integrator: a new circuit principle for continuous-time monolithic filters. *Electronics Letters*, 26(24):2046–2047, November 1990.

[121] E. Seevinck, E. A. Vittoz, M. du Plessi, T. H. Joubert, and W. Beetge. CMOS translinear circuits for minimum supply voltage. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 47(12):1560–1564, December 2000.

[122] J. s Seo, B. Brezzo, Y. Liu, B. D. Parker, S. K. Esser, R. K. Montoye, B. Rajendran, J. A. Tierno, L. Chang, D. S. Modha, and D. J. Friedman. A 45nm CMOS neuromorphic chip with a scalable architecture for learning in networks of spiking neurons. In *2011 IEEE Custom Integrated Circuits Conference (CICC)*, pages 1–4, September 2011.

[123] David E. Shaw, Jack C. Chao, Michael P. Eastwood, Joseph Gagliardo, J. P. Grossman, C. Richard Ho, Douglas J. Lerardi, István Kolossváry, John L. Klepeis, Timothy Layman, Christine McLeavey, Martin M. Deneroff, Mark A. Moraes, Rolf Mueller, Edward C. Priest, Yibing Shan, Jochen Spengler, Michael Theobald, Brian Towles, Stanley C. Wang, Ron O. Dror, Jeffrey S. Kuskin, Richard H. Larson, John K. Salmon, Cliff Young, Brannon Batson, and Kevin J. Bowers. Anton, a special-purpose machine for molecular dynamics simulation. *Communications of the ACM*, 51(7):91, July 2008.

[124] Ryan K. Shultzaberger, Lindsey R. Roberts, Ilya G. Lyakhov, Igor A. Sidorov, Andrew G. Stephen, Robert J. Fisher, and Thomas D. Schneider. Correlation

between binding rate constants and individual information of E. coli Fis binding sites. *Nucleic Acids Research*, 35(16):5275–5283, August 2007.

[125] Sabrina L. Spencer, Suzanne Gaudet, John G. Albeck, John M. Burke, and Peter K. Sorger. Non-genetic origins of cell-to-cell variability in TRAIL-induced apoptosis. *Nature*, 459(7245):428–432, May 2009.

[126] Gary D. Stormo and Yue Zhao. Determining the specificity of proteinDNA interactions. *Nature Reviews Genetics*, 11(11):751–760, November 2010.

[127] Kei Sumiyoshi, Kazuki Hirata, Noriko Hiroi, and Akira Funahashi. Acceleration of discrete stochastic biochemical simulation using GPGPU. *Frontiers in Physiology*, 6, February 2015.

[128] Jeffrey J. Tabor, Travis S. Bayer, Zachary B. Simpson, Matthew Levy, and Andrew D. Ellington. Engineering stochasticity in gene expression. *Molecular bioSystems*, 4(7):754–761, July 2008.

[129] Ilias Tagkopoulos, Charles Zukowski, German Cavelier, and Dimitris Anastassiou. A custom fpga for the simulation of gene regulatory networks. In *Proceedings of the 13th ACM Great Lakes Symposium on VLSI*, GLSVLSI '03, pages 132–135, New York, NY, USA, 2003. ACM.

[130] M. Tavakoli and R. Sarpeshkar. A sinh resistor and its application to tanh linearization. *IEEE Journal of Solid-State Circuits*, 40(2):536–543, February 2005.

[131] J. J. Y. Teo, S. S. Woo, and R. Sarpeshkar. Synthetic Biology: A Unifying View and Review Using Analog Circuits. *IEEE Transactions on Biomedical Circuits and Systems*, 9(4):453–474, August 2015.

[132] Y. Termonia and J. Ross. Oscillations and control features in glycolysis: numerical analysis of a comprehensive model. *Proceedings of the National Academy of Sciences of the United States of America*, 78(5):2952–2956, May 1981.

[133] Chris Toumazou, F. J. Lidgey, and David Haigh. *Analogue IC Design: The Current-mode Approach*. IET, 1990.

[134] Y. Tsividis, N. Krishnapura, Y. Palaskas, and L. Toth. Internally varying analog circuits minimize power dissipation. *IEEE Circuits and Devices Magazine*, 19(1):63–72, January 2003.

[135] Y. P. Tsividis, V. Gopinathan, and L. Toth. Companding in signal processing. *Electronics Letters*, 26(17):1331–1332, August 1990.

[136] Andreas Wagner. Energy constraints on the evolution of gene expression. *Molecular Biology and Evolution*, 22(6):1365–1374, 2005.

[137] Darren J. Wilkinson. Stochastic modelling for quantitative description of heterogeneous biological systems. *Nature Reviews Genetics*, 10(2):122–133, February 2009.

[138] Theodore M. Wong, Robert Preissl, Pallab Datta, Myron Flickner, Raghavendra Singh, Steven K. Esser, Emmett McQuinn, Rathinakumar Appuswamy, William P. Risk, Horst D. Simon, and Dharmendra S. Modha. 10^{14}. *IBM Technical Paper*, November 2012.

[139] S. S. Woo, J. Kim, and R. Sarpeshkar. A Cytomorphic Chip for Quantitative Modeling of Fundamental Bio-Molecular Circuits. *IEEE Transactions on Biomedical Circuits and Systems*, 9(4):527–542, August 2015.

[140] Sung Sik Woo and R. Sarpeshkar. A spiking-neuron collective analog adder with scalable precision. In *Circuits and Systems (ISCAS), 2013 IEEE International Symposium on*, pages 1620–1623, May 2013.

[141] Mae L. Woods, Miriam Leon, Ruben Perez-Carrasco, and Chris P. Barnes. A Statistical Approach Reveals Designs for the Most Robust Stochastic Gene Oscillators. *ACS Synthetic Biology*, 5(6):459–470, June 2016.