COMPUTATIONAL AND STRUCTURAL BIOTECHNOLOGY JOURNAL

Volume No: 7, Issue: 8, April 2013, e201304003, http://dx.doi.org/10.5936/csbj.201304003

CSBJ

The biological microprocessor, or how to build a computer with biological parts

Gerd HG Moe-Behrens [a,*]

Abstract: Systemics, a revolutionary paradigm shift in scientific thinking, with applications in systems biology, and synthetic biology, have led to the idea of using silicon computers and their engineering principles as a blueprint for the engineering of a similar machine made from biological parts. Here we describe these building blocks and how they can be assembled to a general purpose computer system, a biological microprocessor. Such a system consists of biological parts building an input / output device, an arithmetic logic unit, a control unit, memory, and wires (busses) to interconnect these components. A biocomputer can be used to monitor and control a biological system.

Review Article

Introduction

Nature and computers are words that used to mean unrelated things. However, this view changed, starting in the 1940s, when a revolutionary scientific paradigm, systemics based on platonic idealistic philosophy, gained popularity [1] [2] [3].

The roots of philosophical idealism based systemics goes back to Plato. A centerpiece of Plato's (428/7 to 348/7 BC) work is his theory of forms, also called theory of ideas [2]. Forms are archetypes, blueprints, the essences of the various phenomena of the same thing. The superior world consists, due to Plato, of mathematical objects, terms and non-materialistic abstract ideas. Moreover, Plato introduced in his dialogue Philebus a concept called System [4]. A system is according to Plato a model for thinking about how complex structures are developed. Another idealistic philosopher, Kant, introduced, in 1790, in his Critique of Judgment the concept of self-organizing [5]. Idealistic concepts based systemics have become important in contemporary science in order to understand complexity and big data problems. Between the 1950s and 60s three groundbreaking works were published: 1948, Norbert Wiener publishes "Cybernetics or Control and communication in the animal and machine" [1]. In 1955 William Ross Ashby's "Introduction to cybernetics" came out [6]. 1968, Ludwig Bertalanffy published "General System theory: Foundations, Development, Applications" [7]. Bertalanaffy defined the concept of systems. Cybernetics explains complex systems that exist of a large number of interacting and interrelated parts. Wiener and Ashby pioneered the use of mathematics to study systems. This systems theory was further developed in the following years. Important contributions to the field are by Heinz Foerster, whose work focused on cybernetics, the exploration of regulatory systems, and who founded in 1958 the Biological Computer Lab (BCL) at the Department of Electrical Engineering at the University of Illinois [8]. The work of the BCL was focused on the similarities in cybernetic systems and electronics and especially biology inspired computing [9].

[a]*Leukippos Institute, Berlin, Germany*

* Corresponding author.
E-mail address: leukipposinstitute@googlemail.com

Other important contributions to systemics are by the Nobel-prize winning work of Ilya Prigogine on self-organization and his systems theory concepts in thermodynamics [10]. Furthermore: Mitchell Feigenbaums work on Chaos theory [11]. Contemporary application finds systems theory in bioscience in fields such as systems biology, and its practical application synthetic biology [12]. The term systems biology was created by Bertalanffy in 1928 [13]. Systems biology focuses on complex interactions in biological systems by applying a holistic perspective [12].

Altogether, this kind of thinking has led to the identification of ideas behind data processing in nature, but also in machines, such as silicon computers.

Natural Computing

This idea based thinking led to three distinct, but inter-related approaches, termed natural computing: computing inspired by nature, computer models of nature, and computing with natural materials [14] (Figure 1).

Data processing in nature

Focusing on information flow can help us to understand better how cells and organisms work [15]. Data processing can be found in nature all down to the atomic and molecular level. Examples are DNA information storage, and the histone code [16]. Moreover, cells have the potential to compute, both intra cellular (e.g. transcription networks) and during cell to cell communication [17]. Higher order cell systems such as the immune and the endocrine system, the homeostasis system, and the nerve system can be described as computational systems. The most powerful biological computer we know is the human brain [18].

Computing inspired by nature

General systems theory is an important fundament for computer science [1]. Interesting work has be done, as discussed above, by the Biological Computer Laboratory led by Heinz Foerster [8] [9].

In practical terms, nature inspired to programming paradigms such as cellular automata, artificial neural networks, evolutionary algorithms, evolutionary biology, genetic programming, swarm

intelligence, artificial immune systems, membrane computing and amorphous computing [14] [19]. The common aim of all these concepts is solving complex problems.

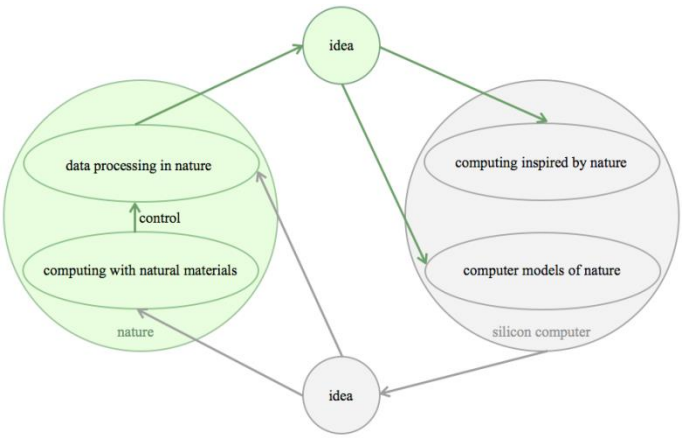

Figure 1. Natural computing: A platonic idea is an archetype, a blueprint, the essence of various phenomena of the same thing. Systemics and systems biology are such ideas, describing data processing systems in nature in terms of mathematics and formal logic. Systemic ideas have been used as a blueprint for silicon computing. Ideas derived from the observation of nature have also inspired computer models of nature. Engineering ideas behind silicon computer (such as standardized parts, switches, logic gates, input /output device, arithmetic logic unit, control unit, memory, and busses) have been used by synthetic biologists to build computers with biological parts, with the ultimate goal to control data processing in nature.

Computer models of nature

The aim of the simulation and emulation of nature in computers is to test biological theories, and provide models that can be used to facilitate biological discovery. Moreover, these models can potentially be used for computer aided design of artificial biological systems.

Systems biology provides theoretical tools to model complex interactions in biological systems [12]. Design principles of biological circuits have been translated into mathematical models. These design models find their practical application in synthetic biology in general, and cellular computer especially. The different areas of natural computing clearly influence each other.

A breakthrough in the modeling and synthesis of natural patterns and structures was the recognition that nature is fractal [14]. A fractal is a group of shapes that describes irregular and fragmented patterns in nature, different from Euclidean geometric forms [20].

Other mathematical systems, as cellular automata, are both inspired by nature and can be used to modulate nature *in silico*, as some biological processes occur, or can be simulated, by them such as shell growth and patterns, neurons and fibroblast interaction [21] [22].

Another computational model of nature is the Lindenmayer-system (or L-system), which is used to model the growth process of plant development [23]. A major step towards the creation of artificial life was recently achieved by Karr et al [24]. This group reports a whole-cell computational model of the life cycle of the human pathogen Mycoplasma genitalium that includes all of its molecular components and their interactions. This model provides new insight into the *in vivo* rates of protein-DNA association and an inverse relationship between the durations of DNA replication initiation and replication. Moreover, model predictions led to experiments which identified previously undetected kinetic parameters and biological functions.

Computing with natural materials

Engineering ideas behind silicon computers can be applied to engineering with natural materials in order to gain control over biological systems. This concept started to emerge in the 1960s when Sugita published ground breaking theoretical work where he performed a functional analysis of chemical systems *in vivo* using a logical circuit equivalent [25] [26]. He discussed the idea of a molecular automaton, the molecular biological interpretation of the self-reproducing automata theory, and the chemico-physical interpretation of information in biological systems [27] [28]. Sugita made analogies between an enzymatic cascade and logic, values and concentrations, and interactions and circuit wires.

The emerging field of synthetic biology has contributed with novel engineering concepts for biological systems [29] [30]. The development of standardized biological parts has been a major task in synthetic biology, which led among other things to the open MIT Registry of Standard Biological Parts, and the BIOFAB DNA tool kit [30] [31] [32]. Another engineering principle, abstraction hierarchy, deals with the question of how standardized parts build a complex system. Systems (systemics) are another important engineering paradigm dealing with complexity [9] [33]. A system is a set of interacting or independent components forming an integrated whole. Common characteristics of a system are: components, behaviors and interconnectivity. Systems have a structure defined by components. Systems behavior involves input, processing and output of data. Behavior can be described with terms such as self-organizing, dynamic, static, chaotic, strange attractor, adaptive. Systems have interconnectivity. This means that the parts of the system have functional as well as structural relationships between each other. This kind of thinking represents a move form molecular to modular biology [34]. The challenge is to define the hierarchical abstraction for such a modular system for biocomputers, and finally actually build such a system.

A breakthrough paper was published in 1994 by Leonard Adleman [35]. For the first time a biocomputer, based on DNA, was built. This system was able to solve a complex, combinatorial mathematical problem, the directed Hamiltonian path problem. This problem is in principle similar to the following: Imagine you wish to visit 7 cities connected by a set of roads. How can you do this by stopping in each city only once? The solution of this problem, a directed graph was encoded in molecules of DNA. Standard protocols and enzymes were used to perform the "operations" of the computation. Other papers using DNA computing for solving mathematical problems followed [36]. Adelman's paper basically kick started the field of biological computers (reviewed in [17] [18] [37] [38] [39]).

Biological parts as system components for biocomputers

A system consists of defined components. In order to build a biocomputer system, we need to identify these components and standardize them. Although important work is done in synthetic biology in respect to part standardization in general, for biocomputer parts this work is so far rudimentary. Thus, we will try in the following to identify and classify them. Such standardized biological parts suitable for computing can be found all along the line of the central dogma of biology: DNA, RNA, protein, and cells (Table 1 to 4).

Table 1. DNA based parts and their application in biocomputing. Representative references are provided.

Part	Circuit	Switch	I/O	Arithmetic Logic	Control Unit	Memory	Buss
nucleotides (order)						41 - 45	
DNA (recombination)				46, 47		42, 45	
DNA (hybridization)				95, 133, 143, 146, 147, 148			
DNA (self-assembly, tiling)				50	18		
gene regulatory circuit / network	67 - 72	116		73, 112, 123 - 130		116	
transcription factors				75	74		
combinatorial promoters				77			
aptamers				131, 132	157		
deoxyribozyme (DNAzymes)				80, 127, 128, 134			
I-switch		117					
transcriptor				135			

DNA

The natural function of DNA is to store hereditary information and regulate the expression of this information [40]. Following the Adelman paper a wide range of DNA properties suitable for computing were explored. DNA may serve either as a principal structural component, or as a mediator that arranges tethered ligands or particles [40].

Structural properties of the DNA as the order of nucleotides, recombinational behaviors, self-assembly due to Watson-Crick base paring and storage of free energy have been used for different aspects of computational systems (see Table 1).

Nucleotide sequence: The order of nucleotides within a DNA molecule can be used to store information [41] [42] [43] [44] [45].

DNA recombination: Recombinational DNA behavior, allowed by specified classes of enzymatic activities, has been described in terms of the formal language theory, a branch of theoretical computer science [46]. The associated language consists of strings of symbols that represent the primary structures of the DNA molecules that may potentially arise from the original set of DNA molecules under the given enzymatic activities. Moreover, DNA recombination has been used to [47] solve a mathematical problem: sorting a stack of distinct objects (genetic elements) into proper order and orientation (site-specific DNA recombination) using the minimum number of manipulations [47].

Self-assembly: DNA can self-assemble through Watson-Crick base pairing to produce an arrangement of tiles (shapes) that covers the plane [48]. Computation by tiling is universal, because tiles and matching rules can be designed so that the tilings formed, correspond to a simulation of that device [49]. Thus, macroscopic self-assembly of different DNA-based tiles can be used to perform DNA-based computation. This was e.g. demonstrated by building a one-dimensional algorithmic self-assembly of DNA triple-crossover molecules that can be used to execute four steps of a logical XOR (if either input 1 or input 2 is true (1), so output true; if all input are false (0) or all input are true, so output false) operation on a string of binary bits [50]. Chemically, the value of a tile, 0 or 1, is denoted by the presence of a restriction site (eg Pvu II represents 0, false and EcoR V represents 1, true). Each molecular tile contains a reporter strand in order to extract the answer after self-assembly occurred. The answer produces a barcode display on an analytic gel. This system is static as self-assemble results into prescribed target structures. However it is also possible to engineer transient system dynamics such as in self-assembly pathways. It has been shown that it is possible to program diverse molecular self-assembly and disassembly pathways using a 'reaction graph' abstraction to specify complementarity relationships between modular domains in a versatile DNA hairpin motif [51]. Programming of functions such as a catalytic circuit, nucleated dendritic growth, and autonomous locomotion were achieved with this approach. Moreover, even something sophisticated such as barcodes have been engineered from self-assembled DNA [52].

Free energy stored in DNA: The hydrolysis of the DNA backbone and strand hybridization, are spontaneous because they are driven by the potential free energy stored in DNA itself. A molecular computer using these operations may, in principle, be fueled by its DNA input. Thus it is possible to use the potential energy of a DNA input molecule to drive molecular computation [40] [53] [54].

As mentioned, another way DNA may function in biocomputers is as a mediator that arranges tethered ligands or particles [40].

Transcriptional regulatory circuits: A cell senses its environment and calculates the amount of protein it needs for it various functions. This information processing is done by transcription networks. These networks, a major study object of systems biology, often contain recurring network topologies called 'motifs' [55]. Composition and engineering concepts for these circuits have been extensively studied [56] [57] [58] [59] [60] [61] [62] [63] [64] [65] [66]. Many interesting functions such as oscillators, frequency multipliers and frequency band-pass filter have been engineered [67] [68] [69] [70] [71] [72]. Transcriptional regulatory circuits can be seen as an analog to electronic circuits. Data input, data processing and data output is an abstraction found in both circuit types. Transcriptional circuits have chemicals as an input. Data processing happens as functional clusters of genes impact each other's expression through inducible transcription factors and cis-regulatory elements. The output is e.g. proteins. Diverse computational functions (see below) have been engineered through changes in circuit connectivity [73].

Transcription factors: Trigger-controlled transcription factors, which independently control gene expression, have been used as part of the processing unit in a programmable single-cell mammalian biocomputer [74]. Artificial Cys(2)-His(2) zinc finger transcription factors specifically bind different DNA sequences and thus provide components for designing of regulatory networks[75].

Table 2. RNA based parts and their application in biocomputing. Representative references are provided.

Part	Circuit	Switch	I/O	Arithmetic Logic	Control Unit	Memory	Buss
RNA library / ribonuclease				86			
aptamer			87, 88	88, 136			
ribozyme			88	88, 138, 139			
riboswitch / riboregulator		79, 92, 93, 119, 120, 137		137			
RNA (hybridization)				97, 98			
amber suppressor tRNA				140, 141			
orthogonal ribosomes				94			
miRNA			84	84, 95 - 97			
siRNA / shRNA		121	98	97, 98, 121			
CRISPR associated Cas9					99		

Combinatorial promoters: Promoters control the expression of genes in response to one or more transcription factors. Rules for programming gene expression with combinatorial promoters have been identified [76]. This opens the option to engineer a wide range of logic functions. Both Boolean and non-Boolean logic is possible as the concentration of regulators is not necessary binary. As an example, a combinatorial promoter has been engineered, which expresses an effector gene only when the combined activity of two internal input promoters is high [77].

Enzymatic machinery for DNA manipulation: Novel cleavage specificities have been designed by combining adapter oligodeoxynucleotide and enzyme moieties [78]. Moreover, functional higher-order nucleic acid complexes can be built from modular motifs such as aptamers (a DNA molecule that specifically binds a small molecule or biomolecule), aptazymes (a DNA molecule that is comprised of an aptamer domain fused to a catalytic domain) and deoxyribozymes (DNAzymes, a DNA molecule with catalytic properties) [79]. This kind of design results in highly programmable, smart complexes, which enable engineering beyond conventional genetic manipulation. In line with this, a DNA-based computational platform has been constructed that uses a library of deoxyribozymes, and their substrates, for the input-guided dynamic assembly of a universal set of logic gates and a half-adder/half-subtractor system [80].

Dynamic constructs formed by DNA: Furthermore, DNA can be used to engineer dynamic constructs such as molecular switches and oscillating molecular machines (see below).

RNA

Another promising approach for building biocomputers uses RNA molecules and RNA-based regulation [81]. RNA editing, the modification of RNA sequences, can be viewed as a computational process [37]. Moreover, RNA is involved in regulatory networks, which have been described as normal forms of logic function in the form of: input, logic gate and output [81] [82]. In many RNA based computational systems the inputs are often small RNA molecules or motifs, while the output is mRNA [81] [83] [84]. Different classes of regulatory RNA components for engineering such systems, have been identified e.g. RNA aptamer, ribozymes, riboswitches, orthogonal ribosomes, miRNA and siRNA (Table 2) [85].

Binary RNA library and ribonuclease (RNase) H digestion: The Adleman molecular computing approach has been expanded to RNA [86]. Using specific ribonuclease digestion to manipulate strands of a 10-bit binary RNA library, a molecular algorithm was developed and applied to solve a chess problem.

RNA aptamer: A RNA molecule that specifically binds a small molecule or biomolecule has been engineered to function as an input sensor in biological computing devices [87] [88].

Ribozymes: Catalytic RNA, ribozymes, can play an interesting role in biocomputing [89] [90]. In general, ribozyme activity (cleavage) in cis will repress translation, whereas activity (cleavage) in trans may repress or activate translation [85]. The hammerhead ribozyme is a small, naturally occurring ribozyme that site-specifically cleaves RNA [91]. This ribozyme can function as an actuator in a RNA computing device [88]. Input binding is translated to a change in the activity of the actuator, where a "ribozyme- active" state results in self-cleavage of the ribozyme [88]. The RNA device is coupled to the 3′ untranslated region of the target gene, where ribozyme self-cleavage inactivates the transcript and thereby lowers gene expression [88]. Different signal integration schemes act as various logic gates.

Riboswitches: Regulatory RNA elements can act by binding a small molecule, and thus switching gene expression on or off [92] [79] [93]. Ligand binding may repress or activate transcription or translation [85].

Orthogonal ribosomes: Multiple unnatural (orthogonal - O) ribosomes can be used combinatorially, in a single cell, to program Boolean logic functions [94]. O-ribosomes functioned as input, O-mRNAs as logic gate and fluorescence as output.

miRNA: Binding of microRNA (miRNA) represses translation [85]. This makes miRNA suitable to serve as sensory module to DNA-based digital logic circuits [95] [84] [96] [97].

siRNA: Small interfering RNAs (siRNA) are a class of short RNAs that stimulate post- transcriptional gene silencing through the RNA interference (RNAi) pathway in higher eukaryotes [85]. RNAi has been used to construct a synthetic gene network that implements general Boolean logic to make decisions based on endogenous molecular inputs [98] [97]. The state of an endogenous input was encoded by the presence or absence of 'mediator' small interfering RNAs (siRNAs).

Table 3. Protein based parts and their application in biocomputing. Representative references are provided.

Part	Circuit	Switch	I/O	Arithmetic Logic	Control Unit	Memory	Buss
enzymes			104 - 107	101, 104, 105, 106, 143			
transactivator / transrepressor				74, 108			
protein (chemically inducible dimerization)				109, 110			
DNA polymerase					152, 153		
restriction nuclease FokI, T4 DNA ligase					154		
recombinase / integrase / excisionase						42, 45	
T7 RNA polymerase				142			
zink finger transcription factor				75			

Table 4. Cell to cell communication based parts and their application in biocomputing. Representative references are provided.

Part	Circuit	Switch	I/O	Arithmetic Logic	Control Unit	Memory	Buss
quorum sensing							112
biological neural networks				113, 114			

Programmable DNA/RNA editing: Recently, a new kind of endonuclease has been discovered which can potentially play an interesting role in the design of biocomputers. This endonuclease is the CRISPR (Clustered regularly interspaced short palindromic repeats) associated Cas9 [99]. Cas9 forms a complex with dual-RNAs. The RNA sequence defines a site-specific DNA binding of this complex. This results dsDNA cleavage. This system has a potential for RNA-programmable genome editing. Another RNA-editing platform has been developed by using the bacterial CRISPR pathway [100]. This enables predictable programming of gene expression.

Proteins

Protein based logic systems have been generated *in vitro* [101]. Furthermore, recent studies have developed strategies for protein synthetic biology *in vivo* [102]. Proteins play both as input and output signals a crucial role in the information processing in the cell. Moreover, logic can also be implemented by the regulation of protein functions governing the production, destruction, localization, and activities of biochemical molecules [102] [103] (Table 3).

Enzymes: We have already discussed some roles of enzymes in biocomputing e.g. in DNA manipulation. Another interesting concept for engineering an *in vitro* protein-based logic system is based on input and output of enzymatic reactions [104] [105] [106] [107]. Different enzymes were used alone or coupled to construct different logic gates. The added substrates for the respective enzymes, act as the gate inputs, while products of the enzymatic reaction are the output signals that follow the operation of the gates.

Transactivator/transrepressor: Transcription control in mammalian cells can be enabled by logic gates [74]. It has been shown, that chimeric promoters containing operators specific for up to three different transactivators/transrepressor enable NOT and AND-type regulation profiles with three molecular intervention levels [108].

Chemically inducible dimerization (CID): In CID systems, a small molecule induces the dimerization of two different proteins, producing a ternary complex [101]. Such a system has been used to engineer a transcriptional logic device [109]. A major drawback of many engineered logic circuits is that they require minutes to hours to execute their logic functions due to the long processing time of the transcription and translation machinery [101]. Non genetic circuit devices based on CID might be able to overcome this obstacle. Such a rapid logic device has been built by Miyamoto et al [110]. Boolean logic gates were synthesized by using two chemical inputs. These gates produced output signals such as fluorescence and membrane ruffling on a timescale of seconds.

Cell to cell communication

Inter cellular signaling can be used to build logic into biological systems. An interesting aspect lies in compartmentalization of the circuit where all basic logic gates are implemented in independent single cells that can then be cultured together to perform complex logic functions [111]. Such systems are possible in a wide variety of settings. Examples are cell to cell communication in bacteria by quorum sensing and artificial neural networks (Table 4).

Quorum sensing: Quorum sensing is a system used by many species of bacteria to coordinate gene expression according to their population density. A simple genetic circuit has been combined with quorum sensing to produce more complex computations in space [112]. Biological neural networks: Biological neural networks are composed of circuits of biological neurons. This has not to be confused with the artificial neural networks we described above, which are programming constructs that mimic the properties of biological neurons. Biological neurons have been used to engineer logic gates [113] [114].