Toward integration of *in vivo* molecular computing devices: successes and challenges

Sikander Hayat[1] and Thomas Hinze[2]

[1]Lehrstuhl für Computational Biology, Universität des Saarlandes, Gebäude C7.1, Saarbreucken, Saarland 66123, Germany
[2]Bio Systems Analysis Groups, Jena Centre for Bioinformatics and Friedrich-Schiller-Universität Jena, Ernst-Abbe-Platz 1–4, D-07743 Jena, Germany

(Received 17 July 2008; published online 13 August 2008)

The computing power unleashed by biomolecule based massively parallel computational units has been the focus of many interdisciplinary studies that couple state of the art ideas from mathematical logic, theoretical computer science, bioengineering, and nanotechnology to fulfill some computational task. The output can influence, for instance, release of a drug at a specific target, gene expression, cell population, or be a purely mathematical entity. Analysis of the results of several studies has led to the emergence of a general set of rules concerning the implementation and optimization of *in vivo* computational units. Taking two recent studies on *in vivo* computing as examples, we discuss the impact of mathematical modeling and simulation in the field of synthetic biology and on *in vivo* computing. The impact of the emergence of gene regulatory networks and the potential of proteins acting as "circuit wires" on the problem of interconnecting molecular computing device subunits is also highlighted. [DOI: 10.2976/1.2968443]

CORRESPONDENCE
Sikander Hayat: s.hayat@bioinformatik.uni-saarland.de

Should *in vivo* computing devices be envisioned as a replacement for the current state of the art silicon based computers? Since the inception of the first DNA based computing device by Leonard Adleman (Adleman, 1994) in 1994, many scientific investigations have been carried out, and it seems that when it comes to *in vivo* computing, "problem-specific" molecular computing devices (MCDs) take precedence over all purpose computing devices. Based on the environment in which the computation takes place, MCDs can be broadly classified into *in vitro* computers (mainly based on DNA, RNA, proteins, hybrid structures, or artificial chemistries) and *in vivo* computing devices. As described by Adleman, the MCDs that belong to the first category make use of *in vitro* replication of the DNA subunits while computational units that aim at harnessing the whole protein translational machinery of a living cell and employ gene regulation by proteins comprise the second category (Bogunia-Kubik and Sugisaka, 2002).

Studies in the realm of MCDs have successfully demonstrated individual subunits that can compute both basic and moderately complex mathematical problems; however, the realization of the truly massively parallel MCD can only be possible when these individual subunits can be efficiently circuited together (Sprinzak and Elowitz, 2005; Simpson, 2004). Making proteins act as the information carrying "wire" in a circuit, recent studies (Benenson *et al.*, 2004; Yaakov *et al.*, 2001; Hinze *et al.*, 2008) have brought forth the notion of implementing MCDs as a massively parallel and fully autonomous problem-specific automaton. For example, the autonomous system as described by Yaakov and coworkers (Yaakov *et al.*, 2001) uses ATP, restriction nuclease, and ligase as the "hardware." Double stranded DNA molecules act as the input, and the automaton processes the input molecule via a cascade of restriction, hybridization, and ligation cycles, producing a detectable output molecule that encodes the automaton's final state and thus the computa-

tional result. The computing performance for an output resulting from five transitions was reported to be on the order of 10^9 transitions per second. Related work (Benenson *et al.*, 2004) described a modular, robust, and flexible MCD capable of logical analysis of mRNA disease indicators *in vitro* and controlled administration of biologically active ssDNA molecules. The MCD was reported to operate at concentrations close to 10^{12} molecules per microliter. These and other studies in literature exemplify the emerging use of intelligent diagnostic computing devices in drug delivery, genetic engineering, and biochemical sensing (Rinaudo *et al.*, 2007; Bogunia-Kubik and Sugisaka, 2002; McDaniel and Weiss, 2005).

COMPUTING BOOLEAN AND ARITHMETIC FUNCTIONS

In contrast to the *in vitro* computing devices discussed above, *in vivo* MCDs make use of the naturally occurring translational regulation mechanism of the host organism. Since *in vivo* systems have to go through an additional step of protein translation, they are suggested to be implicitly slower than DNA based *in vitro* MCDs. However, the use of error correction mechanisms naturally implemented in the evolutionarily optimized transcription regulation machinery of living cells makes the overall computation more robust and hence justifies the trade-off with speed (Baker *et al.*, 2006). A recent theoretical study conducted by Cory and Perkins (Cory and Perkins, 2008) has laid the focus on the use of a transcriptional regulatory mechanism to solve basic arithmetic operations. The study shows how different parametrizations of a simple chemical kinetic model of transcription regulation can give rise to these different operations. The accuracy of such theoretical arithmetic calculations based on the transcription regulatory mechanism is dependent on the kinetic parameters as well as the transcription factor concentrations.

From today's perspective, two implementation approaches for MCDs using gene regulation mechanisms have emerged: discrete and continuous devices. They reflect the encoding of processed information. In discrete devices, the domain of possible species concentration is divided into several layers separated by intermediate forbidden ranges. Each layer represents a digit, see Fig. 1, upper part. Within this encoding scheme, binary MCDs have been explored. The discretized interpretation of species concentrations might offer some preferences concerning MCD implementation. Since a range of concentration levels encodes the same digit, the discrete approach benefits from a certain fault tolerance. This leads to robust systems according to electronic gates employing electric current instead of species concentration. Slight variations of signal quality (caused by intrinsic stochasticity of chemical reactions in practice) can be compensated this way. Furthermore, an opportunity for signal refresh becomes feasible. By this procedure, the signal levels (e.g., species concentrations) are adjusted within each layer

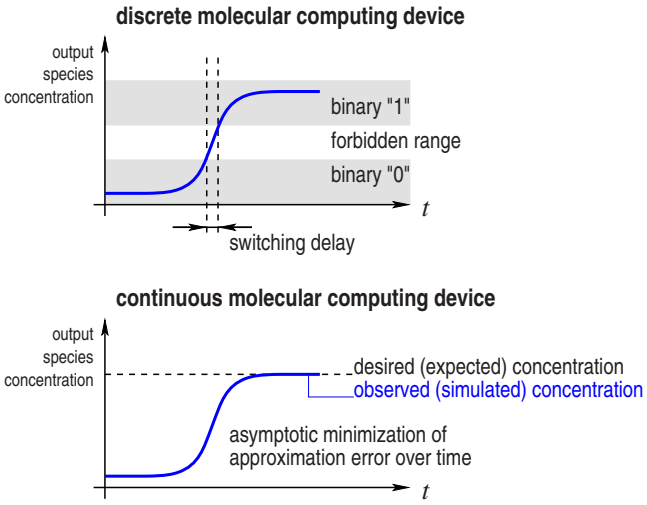

Figure 1. Comparison of discrete and continuous molecular computing devices regarding interpretation of species concentration courses as output data of a computation. While discrete devices perform a predefined sequence of switches typically controlled by a clock, a computation in a continuous device appears as an asymptotic minimization of approximation error over time.

in order to maximize their distance to the forbidden range. Signal refresh can be seen as an essential prerequisite for construction of complex circuits applying feedback loops or consecutive processing units. Typically, those circuits are equipped with a clock whose oscillating binary signals strictly determine the time period for Boolean switching. The clock assures that a computational step is finished before the next one starts (synchronization). Standard devices for Boolean computations were successfully implemented *in vivo* using gene regulation mechanisms of *E. coli*. While Gardner *et al.* introduced a genetic toggle switch (bit storage unit acting as a flip-flop) (Gardner *et al.*, 2000), the pioneering work of Guet *et al.* (Guet *et al.*, 2002) resulted in a NAND gate composed of inhibitors. Each Boolean function can be expressed by interconnected NAND gates. A gene regulatory network (repressilator) generating oscillating clock signals was provided by Elowitz and Leibler (Elowitz and Leibler, 2000). Recent studies investigate capabilities of coupling logic gates, toggle switches, and clock generators toward genetic circuits, a computer architecture based on gene regulation. This promising biochemical realization of von-Neumann's concept will probably lead to reliable as well as programmable discrete MCDs, but it lacks some properties to outperform electronic computers. For instance, an *in vivo* binary switching consumes several hours, and the clock rate of the repressilator is rather slow (approximately 2 h for one clock cycle). Moreover, the potential of massively data parallel computation known from *in vitro* molecular computation has not been effectively utilized *in vivo* up to now.

Continuous MCDs as discussed in Cory and Perkins (Cory and Perkins, 2008) might be a clue to overcome insuf-

ficiencies of discrete devices. In electronic engineering, analogous computers exploit the continuous principle of operation for computing purposes. Historically, they became established for some niche applications such as functional differentiation or integration. Magnasco (Magnasco, 1997) and Deckart and Sauro (Deckart and Sauro, 2004) showed that the dynamical behavior of chemical reaction networks based on a kinetic model can perform computations in a continuous manner. Here, initial species concentrations represent the input data, and dedicated species concentrations reached within the steady state form the output. Particularly, the computation of arithmetic functions seems to be effectively feasible this way. The non-mass-conserving reaction $A+B \xrightarrow{k_1} A+B+C$ along with the decay $C \xrightarrow{k_2} \emptyset$ is a simple example for a reaction network able to multiply two non-negative numbers: The initial concentration values of species A and B {denoted as $[A](0)$ and $[B](0)$} directly define both multipliers. In mass-action kinetics (Sienko *et al.*, 2003), the concentration course of output species C over time is specified by the ordinary differential equation $d[C]/dt = k_1 \cdot [A] \cdot [B] - k_2 \cdot [C]$ while catalyst concentrations $[A]$ and $[B]$ remain constant. In the steady state, the concentration $[C]$ does not change anymore, such that $d[C]/dt = 0$. Thus, $[C] = [A] \cdot [B]$ holds for parametrization of kinetic reaction rates with $k_1 = k_2 = 1$. Assuming a more complex reaction network incorporating several motifs, predetermined parts of the network can be deactivated by setting corresponding reaction rates $k_i = 0$ and activated $(k_i > 0)$, respectively. This way, a reaction network becomes capable of performing multiple arithmetic operations. Cory and Perkins seized this idea to propose analogous computation of arithmetic functions by a continuous MCD. The selection of activated reactions controls the executed operation whose result (output species concentration) evolves asymptotically up to the steady state (see Fig. 1, lower part). Therefore, an arithmetic computation appears as a temporal minimization of the approximation error of the output species. To this end, continuous MCDs benefit from the compactness of the underlying reaction networks in combination with the advantages of assisting self-repair mechanisms available *in vivo*. The contribution by Cory and Perkins marks an important step in understanding how arithmetic computations can be done effectively and reliably by a continuous genetic MCD from a modeling and simulation point of view. A thorough understanding of the influence of kinetics in terms of synthetic biology can be used to design a gene regulatory network that can faithfully capture the desired network behavior (Hayat *et al.*, 2006).

TACKLING COMBINATORIAL SEARCH

In a recently conducted experimental study, Haynes and coworkers (Haynes *et al.*, 2008) used site-specific DNA recombination to solve a burnt-pancake problem (BPP) *in vivo*. The advantage that an *in vivo* system offers in such a case, where an exponential expansion in search space is known to occur, is that DNA replication and bacterial cell growth are inexpensive, feasible, and scalable. Haynes *et al.* draw parallels between the actual BPP and their *in vivo* computational units by suggesting the pancakes be represented by flippable DNA segments. A flip move is mediated by Hin DNA recombinase that regulates gene expression by switching the orientation of the promoter and the coding region while the two palindromic 26bp *hix* sequences flanking the invertible DNA segment act as recognition sites for cleavage and strand exchange. Cells with sorted DNA segments represent the final state. As a proof of concept, Haynes *et al.* designed a two-pancake BPP comprised of a *lac* promoter (pLac) and a tetracycline resistance coding region, each flanked by *hixC* sites. The reaction proceeds by cotransforming cells with a BPP plasmid containing a *hixC*-flanked RBS-tetA(C)rev coding region, a *hixC*-flanked pLac promoter [permutation $(-2,1)$], and a HinLVA expression plasmid. As described, the system reaches equilibrium state after more than 11 h, with most cells in the starting pancake arrangement while about one-third of the cells were detected to have undergone simultaneous inversion of both DNA segments. However, it was also found that due to general leakiness of the pLac promoter, HinLVA-mediated inversion did not seem to require induction of the pLac promoter on the HinLVA plasmid. Furthermore, cells with a strong pLac repressor ($lacI^Q$) were found to be tetracycline resistant even without pLac activation. Cells were still found to be tetracycline resistant even when the pLac promoter was removed from the plasmid construct, leading to the conclusion that pLac is not necessary for expression in the pBR322-derived cloning vector. Tests with a pLac sensitive vector showed that the pLac promoter demonstrates both forward and backward transcription activity, hence rendering phenotypic characterization of the final state ambiguous. In addition, the experimental setup does not allow for quick reversal of states. However, by revealing the unexpected behavior of the pLac promoter, these results illustrate the interdependence of synthetic biology and *in vivo* computing.

These results were confirmed by the accompanying mathematical simulations in which flipping was modeled as a Markov chain such that each of the possible permutations was considered as a state. The probability of a plasmid to be in the sorted order after k flips was determined by dividing the number of paths from the initial to the desired final state by the total number of paths of length k from the start to any state. The results from the mathematical simulations showed that 25% of the total plasmids attained the equilibrium between an intermediate state and the final state after five flips. The overall slow speed (time scale of hours to days rather than minutes) and the lack of robustness could be the bottleneck of such an approach that involves a combinatorial search space of the possible states and involves extensive site-specific DNA recombination. For these reasons, it seems that like most other state of the art MCDs, a more compli-

cated network comprised of the basic subunits represented in the study would be difficult to achieve. However, the study provides experimental proof of employing DNA for computing *in vivo*. Furthermore, this approach gives a new perspective to MCDs, where each DNA segment itself can be considered as a state comprising a finite state automaton (FSA), and transition in states could be conceived as a flip in a DNA segment. The final state, defined by a phenotypic expression, is achieved when the automaton has flipped and oriented all the DNA segments in the desired order.

So far biomolecular computational units have been demonstrated *in vivo* and *in vitro* (Kobayashi *et al.*, 2004; Gardner *et al.*, 2000; Hayat *et al.*, 2006; Kari, 2001). Some of the key issues that still need to be handled are: noise propagation, interfacing output and input, and integrating basic subunits. Sustenance of a large population of cells, removal of intermediate and final products aiding a fast reversal of states are also problems that need to be looked into. Furthermore, the use of MCDs in drug targeting, where they can be programmed to sense the disease indicators and output the drug based on the computation module, makes it imperative to implement these biomolecular computational units in mammalian cells. Recently, Rinaudo *et al.* (Rinaudo *et al.*, 2007) successfully implemented an RNAi based approach to Boolean logic evaluation based on molecular computing, wherein they transfected human embryonic kidney cells with evaluator circuits. An evaluator circuit as described by them consisted of mRNA with fused siRNA targets at the noncoding sites. Based on the presence or absence of complimentary siRNA and the linkage of the mRNAs, various Boolean operations can be performed. Such and other attempts at *in vivo* computing in mammalian cells (Kramer *et al.*, 2004; An *et al.*, 2006) could benefit from developments in the understanding of genetic circuits (Sprinzak and Elowitz, 2005; Simpson, 2004; McDaniel and Weiss, 2005; Kramer *et al.*, 2004) and vice versa.

CONCLUDING REMARKS

In the last decade, independent *in vivo* biocomputational modules that can carry out basic arithmetic and logic operations have been successfully implemented, but the lack of standard protocols at fabricating these subunits is forcing their reimplementation and hence hampering research that needs to be focused on integrating the available MCD subunits. Generating libraries of compatible modules could be helpful in the fabrication of more complex MCDs (Baker *et al.*, 2006). With more than 1000 modules (in 2006) in the BioBrick library, work is already underway toward curating a universal library of biocomputational modules. Furthermore, with emerging rules (Baker *et al.*, 2006; Andrianantoandro *et al.*, 2006), a bottom up approach of implementing a problem-specific rather than an "all purpose" *in vivo* MCD at the forefront of the recent studies in the field of *in vivo* computing and with the recent developments in the use of mammalian cells as precise and robust computational units, the next crucial task of interconnecting (Simpson, 2004) and optimizing these subunits will be instrumental in applying MCDs as intelligent drug delivery systems and biochemical sensors.

ACKNOWLEDGMENTS

We thank Volkhard Helms from Saarland University and Peter Dittrich from Jena University for administrative support.

REFERENCES

Adleman, L (1994). "Molecular computation of solutions to combinational problems." *Science* **266**(5187), 1021–1023.

An, C, Trinh, V, and Yokobayashi, Y (2006). "Artificial control of gene expression in mammalian cells by modulating RNA interference through aptamer-small molecule interaction." *RNA* **12**(5), 710–716.

Andrianantoandro, E, Basu, S, Karig, D, and Weiss, R (2006). "Synthetic biology: new engineering rules for an emerging discipline." *Mol. Syst. Biol.* **2**(2006.0028), 10–1038.

Baker, D, Church, G, Collins, J, Endy, D, Jacobson, J, Keasling, J, Mordich, P, Smolke, C, and Weiss, R (2006). "Engineering life: building a fab for biology." *Sci. Am.* **294**(6), 44–51.

Benenson, Y, Gil, B, Ben-Dor, U, Adar, R, and Shapiro, E (2004). "An autonomous molecular computer for logical control of gene expression." *Nature (London)* **429**(6990), 423–429.

Bogunia-Kubik, K, and Sugisaka, M (2002). "From molecular biology to nanotechnology and nanomedicine." *BioSystems* **65**(2–3), 123–138.

Cory, S, and Perkins, T (2008). "Implementing arithmetic and other analytic operations by transcriptional regulation." *PLOS Comput. Biol.* **4**(4).

Deckard, A, and Sauro, H (2004). "Preliminary studies on the in silico evolution of biochemical networks." *ChemBioChem* **5**(1), 1423–1431.

Elowitz, M, and Leibler, S (2000). "A synthetic oscillatory network of transcriptional regulators." *Nature (London)* **403**(6767), 335–338.

Gardner, T, Cantor, C, and Collins, J (2000). "Construction of a genetic toggle switch in *Escherichia coli*." *Nature (London)* **403**(6767), 339–342.

Guet, C, Elowitz, M, Hsing, W, and Leibler, S (2002). "Combinatorial synthesis of genetic networks." *Science* **296**(5217), 1466–1470.

Hayat, S, Ostermann, K, Brusch, L, Pompe, W, and Rödel, G (2006). "Towards *in vivo* computing: quantitative analysis of an artificial gene regulatory network behaving as a rs flip-flop and simulating the system in silico." *BIONETICS '06: Proceedings of the 1st International Conference on Bio Inspired Models of Network, Information and Computing Systems*, ACM, New York, 5.

Haynes, K, *et al.* (2008). "Engineering bacteria to solve the burnt pancake problem." *J. Biol. Eng.* **2**(1), 8.

Hinze, T, Lenser, T, Matsumaru, N, Dittrich, P, and Hayat, S (2008). "Biosignal-based computing by AHL induced synthetic gene regulatory networks—from an *in vivo* flip-flop implementation to programmable computing agents." In *BIOSIGNALS (1)*, 162–169; Institute for Systems and Technologies of Information, Control and Commmunication, Setubal.

Kari, L (2001). "DNA computing in vitro and in vivo." *FGCS, Future Gener. Comput. Syst.* **17**(7), 823–834.

Kobayashi, H, Kaern, M, Araki, M, Chung, K, Gardner, T, Cantor, C, and Collins, J (2004). "Programmable cells: interfacing natural and engineered gene networks." *Proc. Natl. Acad. Sci. U.S.A.* **101**(22), 8414–8419.

Kramer, B, Viretta, A, Daoud-El Baba, M, Aubel, D, Weber, W, and Fussenegger, M (2004). "An engineered epigenetic transgene switch in mammalian cells." *Nat. Biotechnol.* **22**, 867–870.

Magnasco, M. (1997). "Chemical kinetics is Turing universal." *Phys. Rev. Lett.* **78**, 1190–1193.

McDaniel, R, and Weiss, R (2005). "Advances in synthetic biology: on the path from prototypes to applications." *Curr. Opin. Biotechnol.* **16**(4), 476–483.

Rinaudo, K, Bleris, L, Maddamsetti, R, Subramanian, S, Weiss, R, and Benenson, Y (2007). "A universal RNAi-based logic evaluator that operates in mammalian cells." *Nat. Biotechnol.* **25**, 795–801.

Sienko, T, Adamatzky, A, Rambidi, N, and Conrad, M (2003). *Molecular Computing*, MIT Press, Cambridge, MA.

Simpson, M (2004). "Rewiring the cell: synthetic biology moves towards higher functional complexity." *Trends Biotechnol.* **22**(11), 555–557.

Sprinzak, D, and Elowitz, M (2005). "Reconstruction of genetic circuits." *Nature (London)* **438**(7067), 443–448.

Yaakov, B, Tamar, P, Rivka, A, Ehud, K, Zvi, L, and Ehud, S (2001). "Programmable and autonomous computing machine made of biomolecules." *Lett. Nature* **414**, 430–434.