Continuous computation in engineered gene circuits

Angel Goñi-Moreno[a,*], Martyn Amos[a]

[a]School of Computing, Mathematics and Digital Technology, Manchester Metropolitan University, Manchester M1 5GD, United Kingdom.

Abstract

In this paper we consider the problem of *representation* and *measurement* in genetic circuits, and investigate how they can affect the reliability of engineered systems. We propose a design scheme, based on the notion of *continuous computation*, which addresses these issues. We illustrate the methodology by showing how a concept from computer architecture (namely, branch prediction) may be implemented *in vivo*, using a distributed approach. Simulation results confirm the in-principle feasibility of our method, and offer valuable insights into its future laboratory validation.

Keywords:

1. Introduction

The engineering of logic functions into living cells is of growing interest (Hunziker et al., 2010; Zhan et al., 2010; Silva-Rocha and de Lorenzo, 2011; Ayukawa et al., 2010; Pearson et al., 2011), and forms the basis of the emerging field of *synthetic biology* (Weiss et al., 1998; Benner and Sismour, 2005; Purnick and Weiss, 2009; Serrano, 2007). However, while "traditional" electronics-based logic is an established area, with associated standards and methodologies, genetic logic is still in its infancy. As a result, many basic assumptions vary from one paper to another. The main issue we address in this paper concerns *timing*, which is a significant issue in engineered genetic circuit (Franco et al., 2011). There are several design considerations when dealing with timing, but the two main factors on which we focus here are *representation* and *measurement*.

*Corresponding author.
Email address: a.moreno@mmu.ac.uk (Angel Goñi-Moreno)

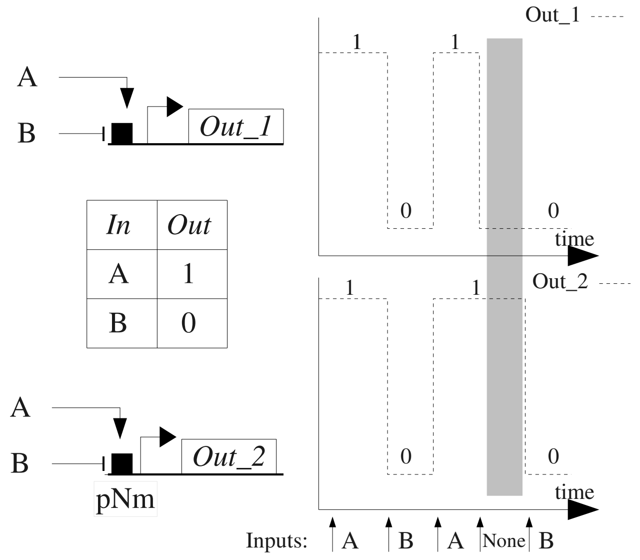
The first issue arises due to the fact that electronic logic has a limited number of ways in which to represent logical values, whereas in genetic logic there are *several* ways in which to specify the two *abstract* values, "0" and "1". In a direct parallel with electronic logic, the value 1 could correspond to a high *concentration* of some signal molecule, while the value 0 corresponds to a low concentration. Alternatively, we can consider the *presence* of the molecule to represent one value, and its absence the other. Finally, we might consider two *different molecules*, A and B, each one representing one of the two values needed to implement Boolean logic. The critical point is that different molecules give rise to different behaviours, and this fact must be considered when they are selected as signal codifiers.

The second open question concerns the temporal behaviour of components. Electronic circuits may be "clocked", such that each operation requires unit time, but no such mechanism exists in engineered biological systems. This is especially problematic when we consider the problem of large circuits with many sub-components. It is generally the case that only static and steady-state measurements are considered when a biological circuit is engineered, but differences in reaction times, molecular degradation rates and so on can make a circuit's behaviour over time much more unpredictable. Therefore, a significant challenge in synthetic biology is the problem of obtaining *reliable* and *continuous* computation.

The rest of the paper is organised as follows. In Section 2 we illustrate some representation and measurement issues of concern, by describing the behaviour of typical genetic logic devices. This motivates the work presented in Section 3, where we describe a scheme (using *distributed* and *continuous* computation) to address these problems. We conclude, in Section 4, with a discussion of our results and their broader implications for synthetic biology and biological computing.

2. Representation and measurement issues in gene circuits

The first issue we consider is that of *representation.* In Figure 1 we show two different implementations of a genetic switch, which takes two inputs, A and B, and returns "on" (or 1) if A is true, and "false" (or 0) if B is present (if *both* A and B are present then the switch defaults to 0). The logical values in this system are represented by the *presence* or *absence* of different molecules; the key difference between the two *implementations* lies in the fact that in the top version, the promoter (corresponding to *Out_1*) is *inducible*, and in the bottom version the promoter controlling *Out_2* is *constitutively expressed* (always “on”). Effectively, in the top implementation, the default state of the switch is “false”, and this is only temporarily forced to be “true” by the incoming signal from A. However, in the bottom implementation, the default state is “true”, and this is temporarily “pulled low” (“false”) by the incoming signal from B. To the right of both circuits we give a graph showing the qualitative behaviour of both circuits over time. We notice that the two behaviour profiles are identical until time x, at which point we observe a discrepancy, with the top circuit giving an output of 0, and the bottom circuit an output of 1 (due to their differing implementations). Given that we cannot reasonably expect gates (within the *genetic* model) to be *always* receiving signals, their default state is therefore of great significance when we consider their behaviour over time. In fact, although the implementations are superficially similar, their behaviours are actually significantly different.



In	*Out*
A	1
B	0

Figure 1: (Left) Two implementations of a simple genetic switch. pNm: neomycin promoter (constitutive). (Right) Qualitative behaviour over time. The shaded region corresponds to the different behaviour when no input is introduced.

We now consider issues of *measurement*. In Figure 2 we show a different circuit, formed by two inverters arranged in series. We also show both *static* and *dynamic* observations of the same case (input 0 $\rightarrow$ output 0). Under a strictly static regime, where a “snapshot” is taken of the circuit’s state, it behaves correctly as an *imply* gate. However, a *continuous* observer would see incorrect behaviour at certain points, due to the fact that the biological processes involved take time. We therefore observe an erroneous "true" output for a certain period *after* introducing the corresponding input. This highlights the importance of considering transitions *between* states in genetic circuits, and of not simply relying on steady-state observations.

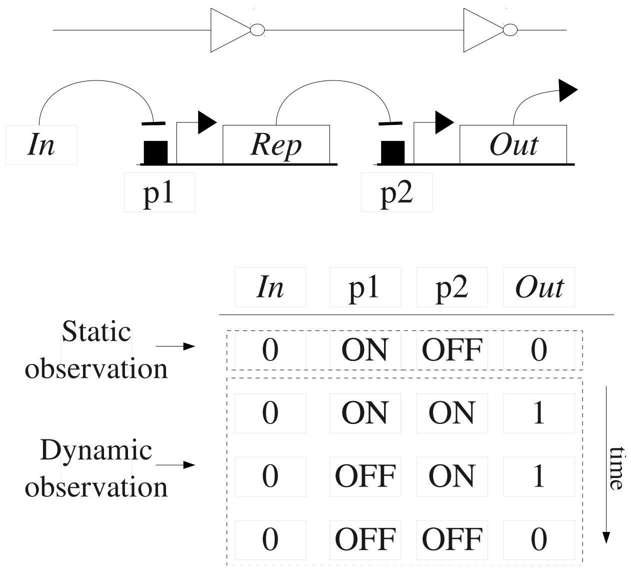


Figure 2: (Up) Implementation of a 2-inverter circuit in series. (Down) Difference in output reading if we consider time (dynamic) or not (static).

3. Continuous computation in synthetic biology

In this Section we present one possible route towards effective continuous computation in engineered gene circuits. We use a model of *distributed* computation, whereby the functionality of a system is partitioned into different components across a number of cell strains. This model of synthetic biology is attracting increasing interest (Regot et al., 2010; Tamsir et al., 2010) with the realisation that *compartmentalization* can help to avoid problems of noise and interference in genetic circuits.

In previous work (Goñi Moreno and Amos, 2011), we showed, in principle, how a simple "client-server" architecture could be implemented. This scheme used two engineered bacterial strains to build a simple oscillator, with a central "server" strain responding to signals from two "client" strains (one per oscillator state). We use a similar scheme in this paper, whereby "processing" is performed in one strain, and "reporting" in another (we call the former "gate" cells, and the latter "bulb" cells, since they "light up" report a result).

We present a cell-based solution to the problem of *branch prediction*. This is a technique commonly used in computer architecture in order to speed up the flow of instructions through a processor chip's pipeline (McFarling and Hennesey, 1986). A branch is a fork in the execution of a program, whereby the arm of the branch to be taken depends on the evaluation of a Boolean expression (e.g., "If A is true, do X, otherwise do Y"). Branch prediction, in its simplest form, makes an assumption about which branch is to be taken, and only corrects the situation if it turns out that the wrong branch was taken. This is particularly useful, as we will see, when one branch has a much higher probability of being taken than the other.

In what follows, we give a design for NAND-based branch prediction. This Boolean function is of particular interest because it provides a *complete basis* (that is, any arbitrary Boolean function may be translated into a circuit of only NAND gates) (Savage, 1998, p. 392).

3.1. Branch prediction using NAND

In our design for NAND branch prediction, we use a two-strain bacterial population. The two different engineered cells are (1) a two-input NAND logic gate, in which the branch predictor is engineered (Figure 3(a)(top)), and the *bulb* cell, which reads the output of the gate cell and gives a signal figure (Figure 3(a)(bottom)). In this system, the inputs are sensed by the NAND strain and the final output is the expression product of the bulb cell (in this case, green fluorescent protein).

We first consider the nature of the NAND function. As it is a negated AND function, it produces "false" if and only if *both* of its inputs are "true" (or 1), and "true" in all other cases. We see from the truth table in Figure 3(c) that the output, O, of NAND is 1 75% of the time, and 0 only 25% of the time. It may therefore be possible to speed up computations by *assuming* an output value of 1, and then "flipping" this to zero (thus correcting the incorrect prediction) if both inputs turn out to be equal to 1. This is precisely the scheme that we adopt here. The basic flowchart is shown in Figure 3(b); we take two inputs and begin to process them, the first stage being to assume a predicted output value of 1. We then check to see if the prediction is correct, and "turn off" (or repress) the 1 signal if it turns out to be incorrect (thus giving an output of 0).

The structure of the NAND gate cell is shown in Figure 3(a). We use abstract labels for genes (i.e., *G1*, *G2*), except for those directly involved in the *quorum sensing* system (but see (Goñi Moreno et al., 2011) for a specific NAND circuit design). The output of this cell is the quorum sensing molecule AHL, which is produced by the *luxI* gene. The important thing to note about this gene is that it is governed by a *constitutive* promoter, and is therefore "always on" (that is, the default output value from the gate cell is 1). The product of *G1 represses* the *luxI* gene, whereby the product of *G2* inhibits the product of *G1*. Importantly, we choose the representation of inputs such that the molecule representing *Input_2 turns off G2* (we recall the discussion of representation in the previous Section).

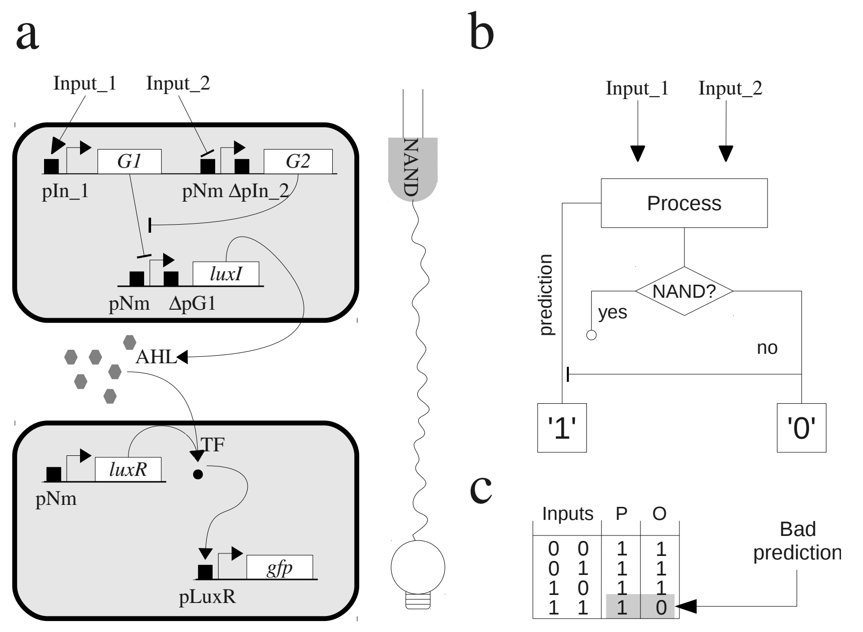
So, if both *Input_1* and *Input_2* are 1, then *G1* is on and *G2* is off (and cannot therefore repress the product of *G1*). Since *G1* is on, it then represses the *luxI* gene, and no AHL is produced (corresponding to an output value of 0). This corresponds to the "correction" of the bad prediction. In the situation where *Input_1* is 1 and *Input_2* is 0, *G1* is on, but so is *G2* (as it is also governed by a constitutive promoter), so it represses the product of *G1*, and *luxI* is expressed. In the other two situations, where *Input_1* is 0, clearly *luxI* is free to be expressed, since there is no *G1* repression possible. The presence (or absence) of AHL is then detected by the bulb cell, which produces the appropriate response.

A key aspect of the model is the resistance to the noise produced by unwanted expression of *luxI*. As a certain AHL signal concentration threshold must be reached before the output is effective, bad predictions will not negatively affect the behaviour of the system (that is, incorrect predictions are corrected before they become significant).

We now present the results of simulation studies of our NAND-based branch prediction, in which we represent the system as a set of Ordinary Differential Equations (ODEs).

Equation 1 represents the rate of change of *G1* over time. This expression product is controlled by the amount of *Input_1* which induces the promoter *pIn_1*. *G1* degradation is given by the degradation rate δ_{G1}.

$$\frac{dG1}{dt} = \alpha_{G1} \cdot \frac{[Input\_1]^{h_1}}{K_{d_1} + [Input\_1]^{h_1}} - \delta_{G1} \cdot [G1] \tag{1}$$



Inputs		P	O
0	0	1	1
0	1	1	1
1	0	1	1
1	1	1	0

Figure 3: **a.** Circuit proposed. Design of NAND cell and bulb cell. *pIn_1* is the Input_1 inducible promoter, Δ*pIn_2* is the truncated Input_2 inducible promoter, Δ*G1* is the truncated *G1* promoter, *pNm* is the neomycin constitutive promoter, *AHL* represents quorum signals, *TF* is a transcription factor, and *gfp* is green fluorescent protein **b.** Functioning of the prediction. **c.** Truth table of the NAND logic gate. *P*, predicted value (default 1): *O*, real output

The formation of the expression product of *G2* is described by equation 2. The upstream promoter complex is repressed by *Input_2* and the final product is degraded at some rate. In the same way, equation 3 denotes the rate of change of *luxI* which is controlled by the element S^*, which is the remaining (active) amount of *G1* after being inhibited by *G2* (direct subtraction at every integration step, allowing *G1* to become 0.0 if the concentration of *G2* is greater).

$$\frac{dG2}{dt} = \alpha_{G2} \cdot \frac{1}{1 + (\frac{[Input\_2]}{\beta_{G2}})^{h_2}} - \delta_{G2} \cdot [G2] \tag{2}$$

$$\frac{dluxI}{dt} = \alpha_{luxI} \cdot \frac{1}{1 + (\frac{[S^*]}{\beta_{luxI}})^{h_2}} - \delta_{luxI} \cdot [luxI] \tag{3}$$

Equation 4 shows the formation of the active transcription factor (TF) which results from the binding, and later dimerization, of *luxR* and *AHL* (those AHL molecules inside the bulb cell) . The product of the molecules is dimerized by the parameter ρ_{TF}, as well as the final value is decreased by the degradation of the TF.

$$\frac{dTF}{dt} = \rho_{TF} \cdot [AHL]^2 \cdot [LuxR]^2 - \delta_{TF} \cdot [TF] \tag{4}$$

Output production (GFP) is controlled by an inducible promoter as describe in equation 5. As with all molecules, GFP is affected by a degradation rate.

$$\frac{dGFP}{dt} = \alpha_{GFP} \cdot \frac{[TF]^{h_1}}{K_{d_2} + [TF]^{h_1}} - \delta_{GFP} \cdot [GFP] \tag{5}$$

As in (Goñi Moreno and Amos, 2011), we determine all parameter values from (Gardner et al., 2000; Basu et al., 2004, 2005; Brenner et al., 2007; Balagaddé et al., 2008; Pai and You, 2009; T. and K., 2006). Based on that information, we consider a parameter value plausible if it is included in the following intervals: synthesis rate $\alpha \in [0 \ldots 1]\mu$M min^{-1}, dimerization coefficient $\rho \in [0.4 \ldots 0.6]\mu$M^{-3} min^{-1}, Hill coefficients $h_1 = 1$ and $h_2 = 2$, dissociation constants $K \in [0 \ldots 1]\mu$M, protein decay $\delta \in [0.01 \ldots 0.1]$min^{-1} and repression coefficients $\beta \in [0.03 \ldots 0.08]\mu$M. The expression product of *luxR* is constant in the cell (constitutively expressed) at 0.5μM.

Initial studies of of the system aimed to identify those parameters whose value can change drastically the behaviour of the model. In this case, we conclude that the relation $\alpha_{G1} \ggg \alpha_{G2}$ and/or $K_{d_1} \lll 1$ must always be kept inside the valid intervals in order to ensure a similar concentration of both *G1* and *G2*. Otherwise, the circuit will not work as expected. The other parameter values may change within the intervals without significantly affecting the qualitative behaviour of the system.

3.1.1. NAND branch prediction results

Using the equations above (parameters set at: $\alpha_{G1} = 1.0$, $\alpha_{G2} = 0.7$, $K_{d_1} = 0.01$), we investigate the behaviour of the NAND-based branch prediction system. All the simulations shown in this paper were implemented using Python and the ODEPACK package (Hindmarsh, 1983), which solves a system of ordinary differential equations (ODEs) by integration steps.

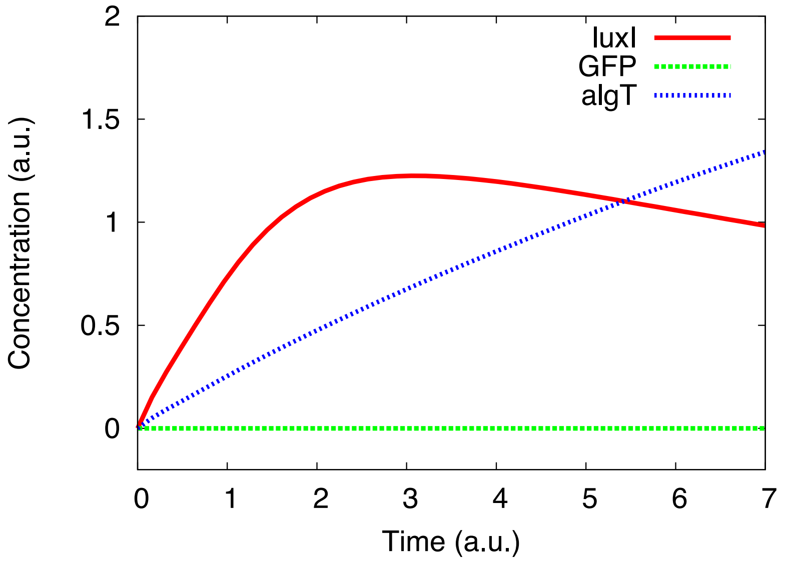


Figure 4: Simulation of NAND-based branch prediction. Bad prediction (deterministic results) (a.u. arbitrary units).

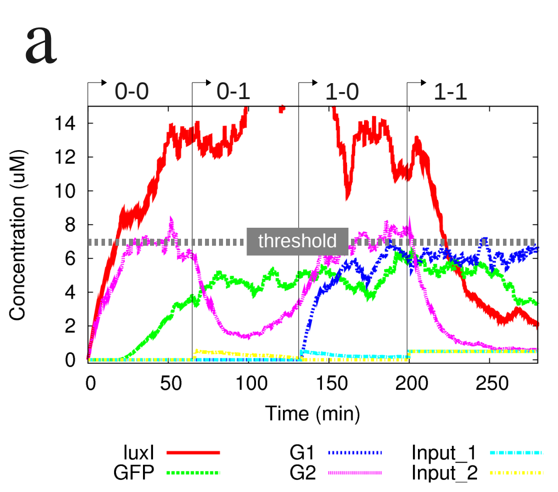
Figure 4 shows the result of simulating the case when *both* inputs are 1, so the system erroneously predicts a positive output. As previously stated, the amount of AHL molecules (directly proportional to the expression level of *luxI*) is not sufficient to reach the threshold (fixed at 7μM) required to induce GFP production in the bulb cell. After a specific internal *processing time*, (at around 1.5 a.u.), *G1* represses the production of the output. In all others case (not shown), the output molecule is constantly expressed until the threshold is reached (when AHL becomes 0.5μM inside the bulb cell).

A transition through different input values is shown in the stochastic simulation results in Figure 5. For this set of simulations we added Gaussian noise ($mean =$ previous value; $standard\ deviation =$ noise $\cdot$ previous value) at every iteration of the integration method (for every equation in the system) in order to obtain a cumulative noise effect for more realistic values (the noise value of 4%-8% was obtained from (Paulsson, 2004)). With this method, we still measure concentration (instead of number of molecules) as the resulting equations are contiuous (instead of discrete). The transition between the four input combinations (0-0, 0-1, 1-0 and 1-1) is clearly marked (Figure 5a), as is the threshold for GFP production (7μM), corresponding to an output of 1. After 15 minutes, the *luxI* expression level is sufficient to trigger a 1 signal, which correctly remains in place for the first three input combinations. However, when we reach the bad prediction (1-1), there is a natural delay before the production of *luxI* ceases. However, only in one case (out of four) do we actually need to process the inputs in order to return the correct output. After the gene *luxI* is no longer expressed, the concentration of AHL molecules is controlled purely by degradation. Due to the fact that those molecules are still present in the system, the *effective off* cannot be applied to the reporter *gfp* until a short time later. Figure 5b depicts the expression of GFP over time in 30 parallel stochastic simulations, plus 1 deterministic guide. Here we can clearly see the interval of time where the output is considered erroneous.

In order to read the figures correctly it is important to notice that the inputs, as well as the rest of values, are affected by a degradation rate. That is why, for example, *G2* starts being expressed again within the interval *0-1* (time = 100). That is due to the fact that *Input_2* is no longer present in the system. By doing this, we simulate an scenario in which the input concentrations are introduced just at the beginning of *their* turn in order to highlihgt the continuous computation (correct prediction of output). However, the last case (*1-1*) is maintained over time to check the *bad* prediction interval.

4. Discussion

Although it is a rapidly-growing field, synthetic biology still lags behind traditional engineering disciplines in terms of standards and methodologies. In this paper, we first consider two important design considerations which are often overlooked when synthetic gene circuits are engineered. The representation scheme for signals within a circuit, and the measurement of these signals, are both important factors for biological computing, and we illustrate this with examples, using artificial circuits. We then consider the notion of *continuous computation*, and ask how it might be achieved in engineered cells. We present one possible scheme that uses a version of *branch prediction* (an idea borrowed from computer architecture), and give the results of simulation studies. Apart from demonstrating the in-principle feasibility of our design, the computational studies offer important insights into the key parameters for future experimental validation. Taking advantage of those designs (which have been applied successfully in computer engineering) in building new genetic circuits will became increasingly important for syn-



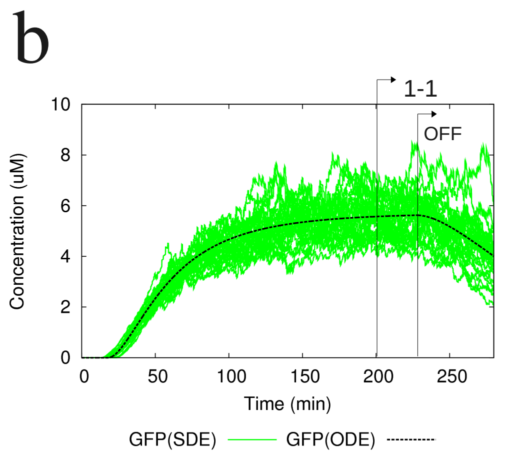


Figure 5: Simulation of NAND-based branch prediction. Stochastic simulation results. a: Computation over time of the circuit proposed while inputs are introduce (or not) to suit the four cases (Input_1 = 0, Input_2 = 0; Input_1 = 0, Input_2 = 1; Input_1 = 1, Input_2 = 0; Input_1 = 0, Input_2 = 1). b: Effective off signal raised after bad prediction. Results shown: GFP over time in 30 stochastic simulations (SDE) and 1 deterministic (ODE). Parameter set up: ($\alpha_{luxI} = \alpha_{GFP} = 0.7$; $\delta_{GFP} = 0.07$; $\delta_{TF} = 0.023$; $\delta_{luxI} = 0.05$; $\delta_{G1} = \delta_{G2} = 0.1$; $\beta_{G2} = 0.03$; $\beta_{luxI} = 0.04$; $K_{d_2} = 1.0$; $\rho_{TF} = 0.5$); logical 1 in inputs = 0.5μM

thetic biology. Future work will focus on coupling predictors to build *pipeline* structures which could be applied in ecological or medical domains.

5. Acknowledgments

This work was supported by the European Commission FP7 Future and Emerging Technologies Proactive initiative: Bio-chemistry-based Information Technology (CHEM-IT, ICT-2009.8.3), project reference 248919 (BACTOCOM).

Ayukawa, S., Kobayashi, A., Nakashima, Y., Takagi, H., Hamada, S., Uchiyama, M., Yugi, K., Murata, S., Sakakibara, Y., Hagiya, M., Yamamura, M., Kiga, D., 2010. Construction of a genetic AND gate under a new standard for assembly of genetic parts. BMC Genomics 11 (Suppl4 S16), doi:10.1186/1471-2164-11-S4-S16.

Balagaddé, F. K., Song, H., Ozaki, J., Collins, C. H., Barnet, M., Arnold, F. H., Quake, S. R., You, L., 2008. A synthetic Escherichia coli predator-prey ecosystem. Mol. Syst. Biol. 4, 187, doi:10.1038/msb.2008.24.

Basu, S., Gerchman, Y., Collins, C. H., Arnold, F. H., Weiss, R., 2005. A synthetic multicellular system for programmed pattern formation. Nature 434 (7037), 1130–1134, doi:10.1038/nature03461.

Basu, S., Mehreja, R., Thiberge, S., Chen, M., Weiss, R., 2004. Spatiotemporal control of gene expression with pulse-generating networks. Proceedings of the National Academy of Sciences of the United States of America 101 (17), 6355 – 6360.

Benner, S. A., Sismour, M., 2005. Synthetic biology. Nature Reviews Genetics 6, 533–543.

Brenner, K., Karig, D., Weiss, R., Arnold, F., 2007. Engineered bidirectional communication mediates a consensus in a microbial biofilm consortium. Proceedings of the National Academy of Sciences of the United States of America 104 (44), 17300–17304.

Franco, E., Friedrichs, E., Kim, J., Jungmann, R., Murray, R., Winfree, E., Simmel, F., 2011. Timing molecular motion and production with a synthetic transcriptional clock. Proceedings of the National Academy of Sciences 108 (40), E784–E793.

Gardner, T. S., Cantor, C. R., Collins, J. J., 2000. Construction of a genetic toggle switch in Escherichia coli. Nature 403, 339–342, doi:10.1038/35002131.

Goñi Moreno, A., Amos, M., 2011. Model for a population-based microbial oscillator. BioSystems doi:10.1016/j.biosystems.2011.05.011.

Goñi Moreno, A., Redondo-Nieto, M., Arroyo, F., Castellanos, J., 2011. Biocircuit design through engineering bacterial logic gates. Natural Computing 10 (1), 119–127, doi:10.1007/s11047-010-9184-2.

Hindmarsh, A. C., 1983. A systematized collection of ODE solvers. IMACS Transactions on Scientific Computation 1, 55–64.

Hunziker, A., Tuboly, C., Horvath, P., Krishna, S., Semsey, S., 2010. Genetic flexibility of regulatory networks. Proc. Nat. Aca. Sci. 107 (29), 12998–13003.

McFarling, S., Hennesey, J., 1986. Reducing the cost of branches. ACM SIGARCH Computer Architecture News 14 (2), 396–403.

Pai, A., You, L., 2009. Optimal tuning of bacterial sensing potential. Molecular Systems Biology 5 (286), doi:10.1038/msb.2009.43.

Paulsson, J., 2004. Summing up the noise in gene networks. Nature 427 (6973), 415–418.

Pearson, B., Lau, K., Allen, A., Barron, J., Cool, R., Davis, K., DeLoache, W., Feeney, E., Gordon, A., Igo, J., Lewis, A., Muscalino, K., Parra, M., Penumetcha, P., Rinker, V., Roland, K., Zhu, X., Poet, J., Eckdahl, T., Heyer, L., Campbell, A., 2011. Bacterial hash function using DNA-based XOR logic reveals unexpected behavior of the LuxR promoter. Interdisciplinary Bio Central 3 (10), 1–8, doi:10.4051/ibc.2011.3.3.0010.

Purnick, P., Weiss, R., 2009. The second wave of synthetic biology: from modules to systems. Nature Reviews Molecular Cell Biology 10 (6), 410–422.

Regot, S., Macia, J., Conde, N., Furukawa, K., Kjellén, J., Peeters, T., Hohmann, S., De Nadal, E., Posas, F., Solé, R., 2010. Distributed biological computation with multicellular engineered networks. Nature 469 (7329), 207–211.

Savage, J. E., 1998. Models of computation: Exploring the power of computing. Addison-Wesley.

Serrano, L., 2007. Synthetic biology: promises and challenges. Molecular Systems Biology 3 (1).

Silva-Rocha, R., de Lorenzo, V., 2011. Implementing an ORNOT (ORN) logic gate with components of the SOS regulatory network of Escherichia coli. Mol. BioSyst. 7, 2389–2396, doi:10.1039/C1MB05094J.

T., T., K., B., 2006. Stochastic models for regulatory networks of the genetic toggle switch. Proceedings of the National Academy of Science 103 (22), E8372 – E8377.

Tamsir, A., Tabor, J., Voigt, C., 2010. Robust multicellular computing using genetically encoded NOR gates and chemical 'wires'. Nature 469 (7329), 212–215.

Weiss, R., Homsy, G., Nagpal, R., 1998. Programming biological cells. Proceedings of the 8th International Conference on Architectural Support for Programming Languages and Operating Systems, 147–149.

Zhan, J., Ding, B., Ma, X., Su, X., Zhao, Y., Liu, Z., Wu, J., Liu, H., 2010. Develop reusable and combinable designs for transcriptional logic gates. Molecular Systems Biology 6 (388), doi:10.1038/msb.2010.42.