# DNA Reservoir Computing: A Novel Molecular Computing Approach

Alireza Goudarzi[1], Matthew R. Lakin[1], and Darko Stefanovic[1,2]

[1] Department of Computer Science
University of New Mexico
[2] Center for Biomedical Engineering
University of New Mexico
alirezag@cs.unm.edu

**Abstract.** We propose a novel molecular computing approach based on reservoir computing. In reservoir computing, a dynamical core, called a *reservoir*, is perturbed with an external input signal while a *readout layer* maps the reservoir dynamics to a target output. Computation takes place as a transformation from the input space to a high-dimensional spatiotemporal feature space created by the transient dynamics of the reservoir. The readout layer then combines these features to produce the target output. We show that coupled deoxyribozyme oscillators can act as the reservoir. We show that despite using only three coupled oscillators, a molecular reservoir computer could achieve 90% accuracy on a benchmark temporal problem.

## 1 Introduction

A reservoir computer is a device that uses transient dynamics of a system in a critical regime—a regime in which perturbations to the system's trajectory in its phase space neither spread nor die out—to transform an input signal into a desired output [1]. We propose a novel technique for molecular computing based on the dynamics of molecular reactions in a microfluidic setting. The dynamical core of the system that contains the molecular reaction is called a reservoir. We design a simple *in-silico* reservoir computer using a network of deoxyribozyme oscillators [2], and use it to solve temporal tasks. The advantage of this method is that it does not require any specific structure for the reservoir implementation except for rich dynamics. This makes the method an attractive approach to be used with emerging computing architectures [3].

We choose deoxyribozyme oscillators due to the simplicity of the corresponding mathematical model and the rich dynamics that it produces. In principle, the design is generalizable to any set of reactions that show rich dynamics. We reduce the oscillator model in [2] to a form more amenable to mathematical analysis. Using the reduced model, we show that the oscillator dynamics can be easily tuned to our needs. The model describes the oscillatory dynamics of three product and three substrate species in a network of three coupled oscillators. We introduce the input to the oscillator network by fluctuating the supply of substrate molecules and we train a *readout layer* to map the oscillator dynamics onto a target output. For a complete physical reservoir computing design, two main problems should be addressed: (1) physical implementation of the

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/ 978-3-319-01928-4_15 reservoir and (2) physical implementation of the readout layer. In this paper, we focus on a chemical design for the reservoir and assume that the oscillator dynamics can be read using fluorescent probes and processed using software. We aim to design a complete chemical implementation of the reservoir and the readout layer in a future work (cf. Section 5). A similar path was taken by Smerieri et al. [4] to achieve an all-analog reservoir computing design using an optical reservoir introduced by Paquot et al. [5].

We use the molecular reservoir computer to solve two temporal tasks of different levels of difficulty. For both tasks, the readout layer must compute a function of past inputs to the reservoir. For Task A, the output is a function of two immediate past inputs, and for Task B, the output is a function of two past inputs, one $\tau$ seconds ago and the other $\frac{3}{2}\tau$ seconds ago. We implement two varieties of reservoir computer, one in which the readout layer only reads the dynamics of product concentrations and another in which both product and substrate concentrations are read. We show that the product-only version achieves about 70% accuracy on Task A and about 80% accuracy on Task B, whereas the product-and-substrate version achieves about 80% accuracy on Task A and 90% accuracy on Task B. The higher performance on Task B is due to the longer time delay, which gives the reservoir enough time to process the input. Compared with other reservoir computer implementations, the molecular reservoir computer performance is surprisingly good despite the reservoir being made of only three coupled oscillators.

## 2 Reservoir Computing

As reservoir computing (RC) is a relatively new paradigm, we try to convey the sense of how it computes and explain why it is suitable for molecular computing. RC achieves computation using the dynamics of an excitable medium, the reservoir [6]. We perturb the intrinsic dynamics of the reservoir using a time-varying input and then read and translate the traces of the perturbation on the system's trajectory onto a target output.

RC was developed independently by Maass et al. [7] as a model of information processing in cortical microcircuits, and by Jaeger [8] as an alternative approach to time-series analysis using Recurrent Neural Networks (RNN). In the RNN architecture, the nodes are fully interconnected and learning is achieved by updating all the connection weights [8, 9]. However, this process is computationally very intensive. Unlike the regular structure in RNN, the reservoir in RC is built using sparsely interconnected nodes, initialized with fixed random weights. There are input and output layers which feed the network with inputs and obtain the output, respectively. To get the desired output, we have to compute only the weights on the connections from the reservoir to the output layer using examples of input-output sequence.

Figure 1 shows a sample RC architecture with sparse connectivity between the input and the reservoir, and between the nodes inside the reservoir. The output node is connected to all the reservoir nodes. The input weight matrix is an $I \times N$ matrix $\mathbf{W}^{in} = [w^{in}_{i,j}]$, where $I$ is the number of input nodes, $N$ is the number of nodes in the reservoir, and $w^{in}_{j,i}$ is the weight of the connection from input node $i$ to reservoir node $j$. The connection weights inside the reservoir are represented by an $N \times N$ matrix $\mathbf{W}^{res} = [w^{res}_{j,k}]$, where