Chapter _. SOFTWARE-AIDED DESIGN OF IDEALISED PROGRAMMABLE NUCLEIC ACID CIRCUITS

Authors names: Iuliia Zarubiieva, Vishwesh Kulkarni
Affiliation: School of Engineering, University of Warwick, UK
Abstract: The idea to use nucleic acid as a substrate for design of programmable biomolecular circuits was firstly introduced almost 4 decades ago, however, up till now, the field of DNA computing holds many challenges and uncertainties to be discovered. This chapter describes the historical evolution of DNA programming along with its most noticeable breakthroughs till the current days, describes the basics of such important theoretical concepts as DNA strand displacement and Abstract Chemical Reaction Networks, and finally, familiarizes the reader with various platforms for *in-silico* synthesis and simulation of genetic circuits.
Keywords: DNA programming, Abstract Chemical Reaction Networks (ACRN), DNA strand displacement (DSD).

1. Introduction

Synthetic biology is an interdisciplinary field laying on the intersection of biology and engineering. It applies the principles from both disciplines in order to design, programme and control the behavior of biological systems. The origins of synthetic biology can be traced back to 1960s when F. Jacob and J. Monod published their work on “Cellular regulation by molecular networks” [1]. However, the modern era of this discipline started in early 2000s with the publication of two notable works in *Nature* journal: one on synthetic oscillatory network of transcriptional regulators by M. Elowitz and S. Leibler [2], and another on genetic toggle switch by T. Gardner *et al.* [3]. In the following years, however, the complexity of programmed circuits increased gradually, so new solutions were required to keep up with the progress.

One of such solutions was the use of nucleic acids to perform computation in biological circuits [4]. Deoxyribonucleic acid (DNA) and ribonucleic acid (RNA) are one of the essential components for all known forms of life. They are responsible for storing genetic information and enabling the production of proteins. DNA can take form of a single strand as well as of multi-stranded complex, where the strands are connected to one another by hydrogen bonds. RNA can only be single-stranded, and has quite different functions compared to DNA. Interested readers are directed to learn more about the functionality of nucleic acids in other sources which are widely available.

In the proposed approach, the interaction rules between different nucleic acid strands are defined through well-understood mechanisms of the Watson-Crick base pairing. This way, the biomolecular interactions inside a cell can be precisely programmed simply by choosing the complementary sequences of nucleotides: C (cytosine) G (guanine), A (adenine) or T (thymine), which is replaced by U (uracil) in RNA. Recent advances in synthetic biology approaches paired with the high and predictable DNA programmability, made DNA and RNA a highly promising base for the implementation of biochemical circuits.

There is a number of approaches for the design of nucleic acid circuits. One example is the programmed RNA self-assembly for catalysis of hydrogen biosynthesis [5], another - DNA structures operating on basic logic were used to release the molecular payload only in case the two antigens, each of them indicating a specific type of cancer, were present simultaneously on the target cell surface [6]. But with the recent advances in synthetic biology, more complex dynamics was required for the regulation of biochemical processes in biomolecular circuits [7]. A few strategies that can implement such dynamics have recently been proposed, based on principles of DNA strand displacement [8], DNA- [9] and RNA enzymes [10]. They were used for various applications, namely, implementing feedback control mechanisms [11], predator-prey dynamics [12], transcriptional oscillators [13], and control of the *in vitro* protein production [14]. Each of the various methodologies for implementing nucleic acid circuits can have their own benefits and drawbacks. That is why it

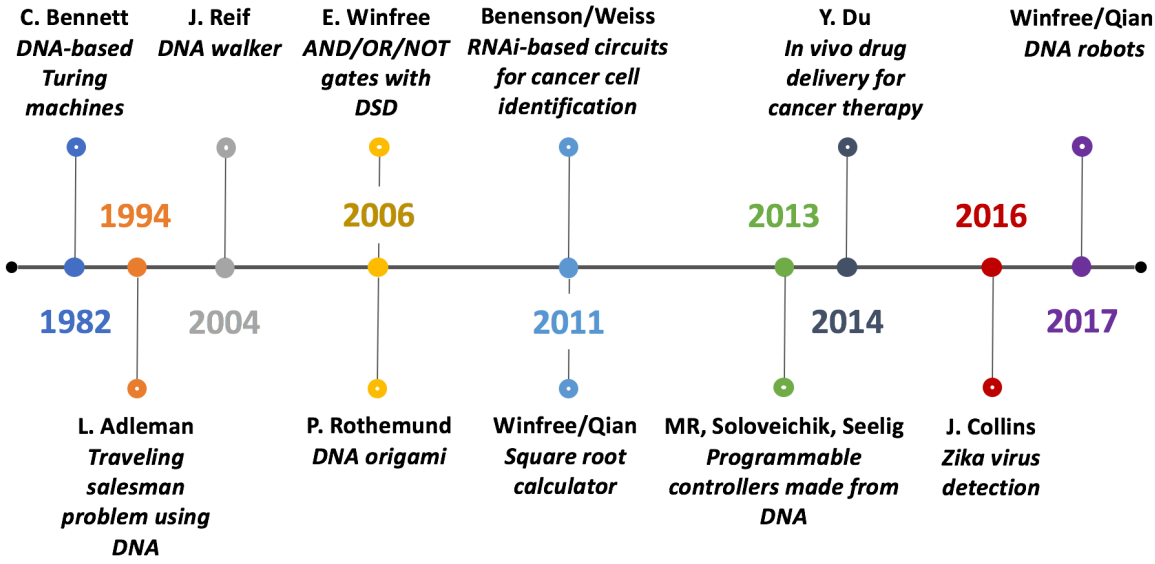


Fig.1. Some of the most notable results in the history of DNA programming. The authors specified are either main authors or group leaders.

is highly useful to apply them to the same system design in order to compare their performance and help identify design strategies that combine the strengths of these different approaches.

A coherent theoretical approach of implementing such circuits by the means of idealised DNA strand displacement reactions has recently been illustrated in [15-18]. In the ideal world, a re-use of well-characterised modules, which is called modularity, simplifies the design of circuits. However, it requires modifications for such a scale up of biomolecular circuits since the building blocks that were characterized separately may change their behaviour once connected to other components. In the context of biochemical networks, Hartwell [19], Lauffenburger [20] and Weiss *et.al.* [21, 22] made the first reliable presentation of this concept.

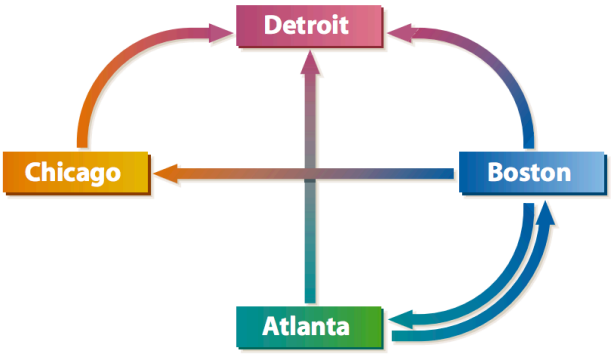
2. Evolution of DNA programming

This section gives an overview of some of the most noticeable results in DNA programming.

DNA-based Turing machine. The history of DNA computing started in 1982, when Charles Bennett designed Turing machine using DNA. In this work, he proposed a conceptual design of Brownian Turing machine and the transition rules for changing the state. However, this approach involved hypothetical enzymes needed for performing the transition. In 1996, Paul Wilhelm and Karl Rothemund extended this work and presented restriction enzymes and ligases that were commercially available [23].

Travelling salesman problem using DNA. Next, in 1994, Leonard Adleman solved a travelling salesman problem using DNA. Travelling salesman is a classic problem in computer science where a salesman must travel between N cities and visit each of them only once. The order in which he visits them is irrelevant, but each connection between 2 cities has its weight, or cost, which determines the “difficulty level” of the trip. The salesman wants to keep the cost and the travel distance of the overall trip as low as possible.

Adleman presented a map, where each of 7 cities and each of 14 connecting flights between them were assigned a DNA sequence. Each city name has its complement, and each connecting flight consists of half of each corresponding city name – e.g., if Atlanta is ACTTGCAG, Boston is TCGGACTG, then the flight Atlanta-Boston becomes GCAGTCGG. Because of the Watson-Crick base pairing, the complementary sequences will bind together. Now every flight could pair up into a double-stranded complex with two cities, and every city could be connected by two flights. This way, DNA complex will grow in length, with each DNA



CITY	DNA NAME	COMPLEMENT
ATLANTA	ACTTGCAG	TGAACGTC
BOSTON	TCGGACTG	AGCCTGAC
CHICAGO	GGCTATGT	CCGATACA
DETROIT	CCGAGCAA	GGCTCGTT

FLIGHT	DNA FLIGHT NUMBER
ATLANTA - BOSTON	GCAGTCGG
ATLANTA - DETROIT	GCAGCCGA
BOSTON - CHICAGO	ACTGGGCT
BOSTON - DETROIT	ACTGCCGA
BOSTON - ATLANTA	ACTGACTT
CHICAGO - DETROIT	ATGTCCGA

Fig.2. Adapted from [24]. A map of cities for travelling salesman problem represented with DNA sequences.

flight number connected to complementary DNA city names. It took Adleman seven consecutive days to successfully complete the DNA computation [24].

DNA walker. The next significant discovery was a creation of a DNA walker – a nanodevice that is able to "walk" autonomously along a DNA "track". First presented by John Reif in 2003, he and his group were able to demonstrate it experimentally the following year. The walker operates based on the phenomena of DNA strand displacement, which we will focus on more closely in section 4.

The process is as follows: one leg of the DNA walker is *strand A*, which is bound to complementary *strand B*, through normal base pairing. *Strand A* contains an additional unbound sequence on one of its ends, named a toehold. Next, the A-B complex encounters *strand C*. This strand is complementary to *strand A*, including the toehold part. Once the toehold of *strand C* binds with the toehold of *strand A*, it begins to displace each nucleotide of *strand B*, base by base, until *strand B* has been completely replaced by *strand C*. *Strand B* then dissociates from *strand A* and the process can begin again [25].

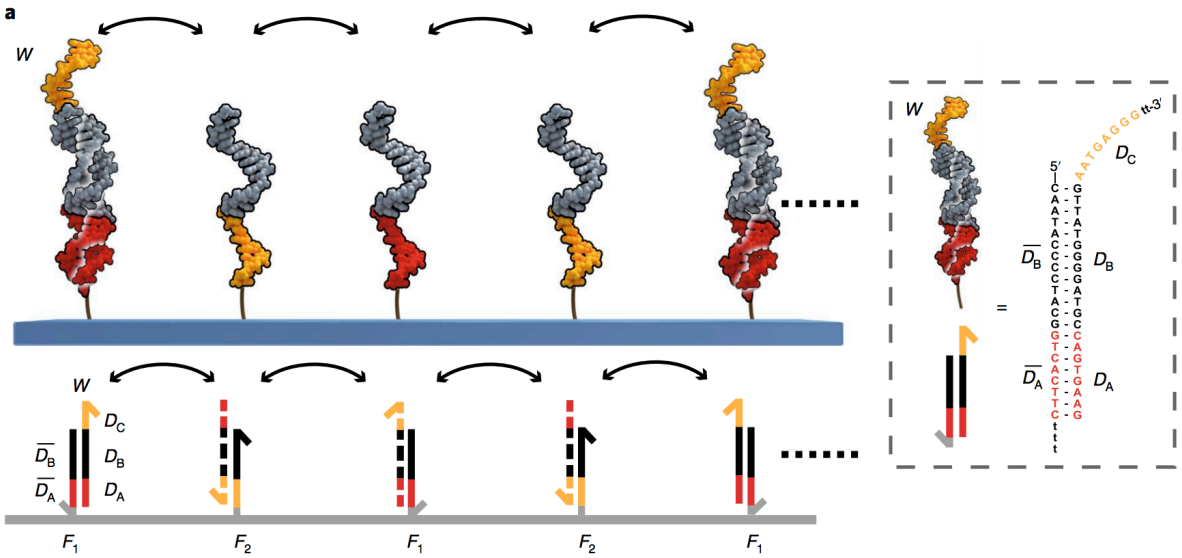


Fig.3. Adapted from [26]. Schematic showing the intended mechanism of DNA walker movement.

In 2018, the team led by Nils Walter presented a DNA walker able to "walk" as fast as 300 nm/min. The applications of DNA walkers include nanomedicine, nanorobotics, biosensing and much more [26].

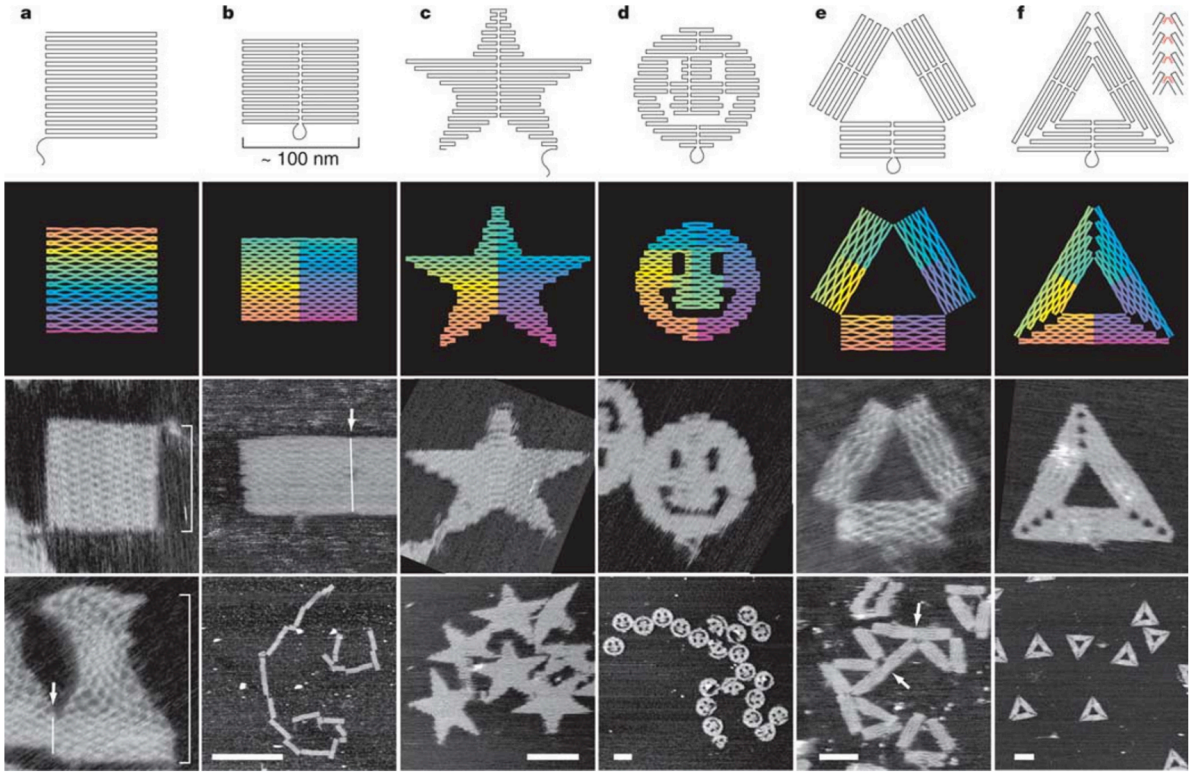


Fig. 4. Adapted from [27]. Examples of DNA origami: a) square; b) rectangle; c) star; d) disk with three holes; e) triangle with rectangular domains; f) sharp triangle with trapezoidal domains and bridges between them. All images and panels without scale bars are the same size, 165 nm × 165 nm. Scale bars for lower AFM images: b) 1 mm; c-f) 100 nm.

DNA origami. A real breakthrough in nanoengineering happened in 2006 when Paul Rothemund presented his work on DNA origami. The concept includes folding of a long single DNA strand supported by multiple small strands, acting like staples. The staple strands hold the main strand in various places, helping to form the required 2- or 3-dimensional shape. Some of the examples of DNA origami are a smiley face, a star, a triangle (2-dimentional), and various cubic and tube-like shapes (3-dimentional). Results can be directly observed via several imaging techniques [27].

Initially created as a form of art, in the coming years DNA origami was developed into various applications, including drug delivery, nanoelectronic circuitry, enzyme immobilization, etc. As an example, a group from the Harvard University Wyss Institute reported the vessels for drug delivery that were capable of self-assembly and self-destruction. Such vessels consist of open DNA-origami tube that has a “door” which can be open or closed. The tube containing the drug is closed by a DNA aptamer which is designed to recognize certain proteins associated with a disease. Once the container reaches cell recognized as infected, the aptamer breaks apart and the tube releases the drug. The researchers used a disease model for leukemia and lymphoma. The results of the research were highly promising, however, one month after the announcement, the publication was retracted.

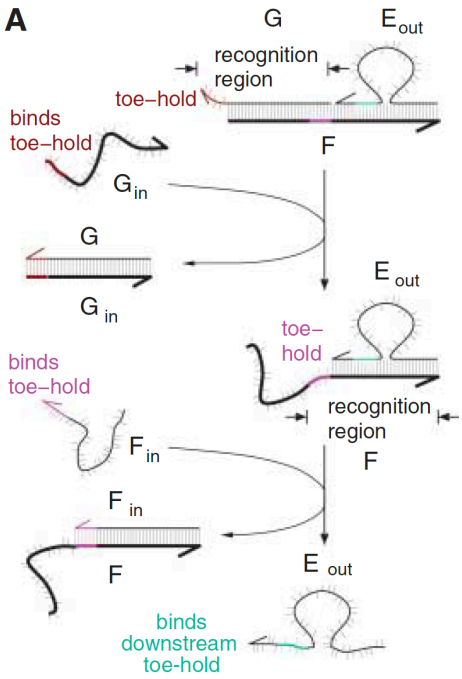


Fig.5. Adapted from [28]. Two-input AND gate. F_{in} and G_{in} are the inputs and E_{out} is the output.

AND/OR/NOT gates with DSD. Same year, Eric Winfree's team published the results on building basic logic elements such as AND/OR/NOT gates. The researchers stated that the specified set is enough for computation of any Boolean logic function. The gates treat single DNA strands as inputs and outputs, and the approach works entirely on the DNA strand displacement principle.

An AND gate produces an output signal only when two input signals are both present. The gate design consists of two single strands and a double-stranded complex. In the first reaction, a first oligonucleotide input (G_{in}) reacts with a gate through toehold-mediated strand displacement. This creates a waste and an intermediate product with an open toehold which then participates in the next reaction together with the second input F_{in}. One strand of the original gate complex is fluorescent and the other one is fluorescence-quenching. The second input separates these, eventually causing the fluorescence that can be recorded as the circuit's readout [28].

The designed gates were a subject to errors, specifically, a gate could produce not enough output, or it could "leak" by releasing the output strand without any trigger. Both types of errors could be corrected using signal restoration.

Square root calculator. Together with Lulu Qian, Eric Winfree has published another prominent work, presenting a circuit that computed a square root of four-bit binary numbers for inputs from 0000 to 1111. The circuit involved 74 initial DNA species, excluding inputs, and in total contains 130 different strands, each of them being 15 to 33 nucleotides long. The design uses dual-rail logic and consists of a combination of AND and OR logic gates. The results are read off in binary from four different fluorescent outputs [29].

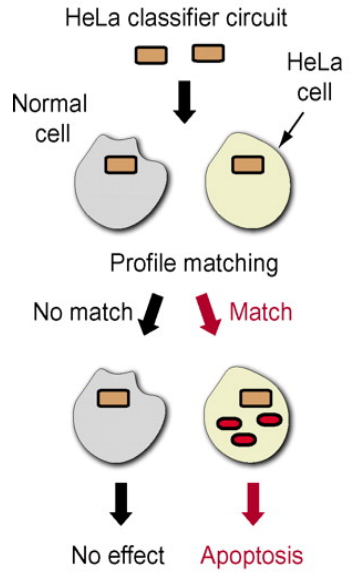


Fig.6. Adapted from [30]. Schematic representation of a HeLa-specific classifier circuit operation. Gray circle, healthy cells; light green, HeLa cells.

RNAi-Based Circuits for Cancer Cell Identification. In 2011, a group led by Yaakov Benenson and Ron Weiss presented intriguing results for anticancer therapy. They designed a device that detects expression levels of a customizable set of microRNAs and selectively initiates apoptosis for the specified cell type without affecting other cells. They demonstrated successful results for HeLa cells, however, also observed certain degree of false-positive cell detection and cell death as well as false-negative cell survival. Despite the observed error rate, this work was an important step to demonstrate how synthetic biological networks can trigger pre-programmed biological behaviour upon detection of specific conditions. The work also defined that in order to use the designed circuit in cancer therapeutics, the challenge of efficient *in vivo* DNA delivery should be overcome. The research opened the possibility of *in vitro* drug screening and monitoring of the developmental process [30].

Programmable controllers made from DNA. After a success of programming language Visual DSD that was

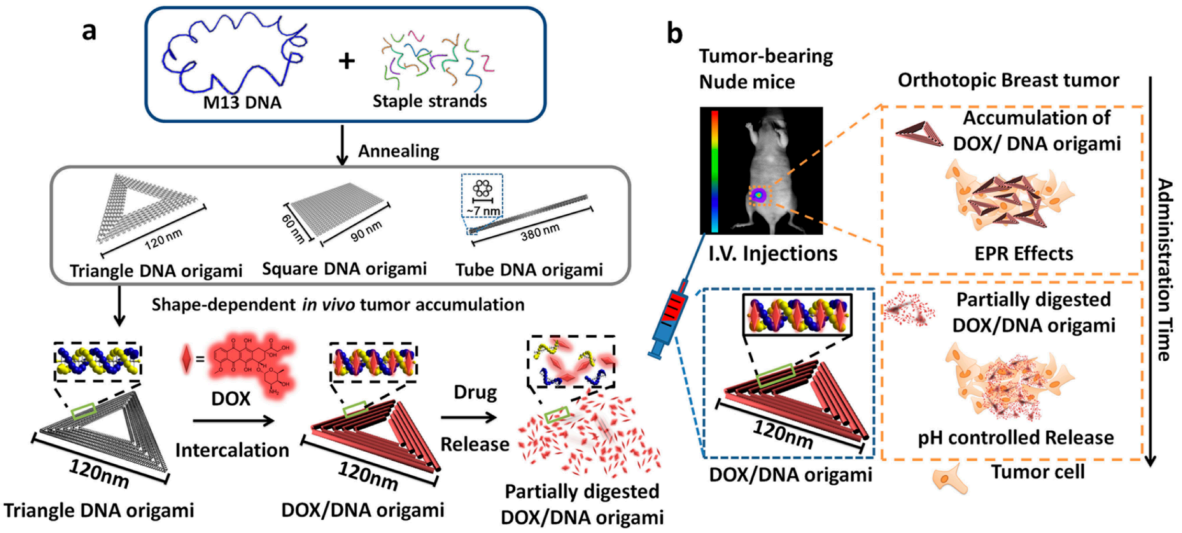


Fig. 7. Adapted from [35]. Schematic design of DNA-carrier complex. (a) Formation of triangular, square, and tube origami. The triangle-shaped DNA origami showed the best results for doxorubicin administration. (b) Tail-injected DOX/DNA origami complexes accumulated in the breast tumor of nude mice because of EPR effects.

developed by Microsoft Research, they teamed up with David Soloveichik and Georg Seelig to design programmable DNA controllers that could work with DNA sensors and motors. This work illustrated the diversity of computing using DNA strand displacement. Unlike Boolean logic circuits which were a more popular choice among researchers, analogue devices that were presented in this work allow complex signal processing of various biological and chemical inputs. Components of the controller can be obtained from biologically synthesized DNA, which is more beneficial compared to its chemically synthesized counterpart due to lower synthesis error. The algorithm was tested on three fundamental reaction types: noncatalytic (A + B → C), catalytic (A + B → C + B) and autocatalytic (A + B → C + 2B) and identified correct stoichiometry and kinetics [11].

The consensus algorithm that is used for the controller compares two populations of input signals, and the signal that has a lower concentration is fully removed and substituted by the higher value signal. This approach agrees with the one used in computing across networked systems and provides a proof that CRN algorithms can be directly translated into the DNA controllers. The downside of the design is the computation speed. Andrew Phillips from Microsoft Research notes that the algorithm is still quite slow. “This algorithm takes 15 hours. You wouldn’t have a phone that runs on these algorithms. Their advantage is that they can potentially run inside cells.”

***In vivo* drug delivery for cancer therapy.** In 2014, a group of Chinese researchers presented the results where self-assembled DNA origami nanostructures were used as vehicles for delivering anticancer drug (doxorubicin) into cancer tumors. They showed that triangle-shaped origami structures fitted particularly well for tumor accumulation. The vehicles were prepared through the self-assembly using large number of staple strands, after what the doxorubicin was placed inside of the structure. The result analysis showed that the doxorubicin-containing DNA origami demonstrated outstanding antitumor efficiency without noticeable toxicity in nude mice with breast tumors. The approach proved to be prominently efficient for *in vivo* therapeutics and demonstrated the potential of DNA origami nanostructures for the successful drug delivery [35].

DNA robots. In 2017, the collaboration of groups of Eric Winfree and Lulu Qian gave another outstanding result which is a DNA robot capable to autonomously sort cargo. Most of the robots developed earlier could perform only one simple operation - walking in a direction defined by user. So, the researchers were devoted to create a DNA nanomachine that could perform complex nanomechanical tasks. They aimed to tackle two major difficulties that are met when designing DNA robots – namely, modularity and algorithm simplicity. The robot moves along a DNA origami surface and collects two types of cargo, followed by the delivery of this cargo to a specified final location. The robot does not require any energy supply for its operation. It is also possible for several DNA robots to work simultaneously on the same surface and to perform the task collectively [32].

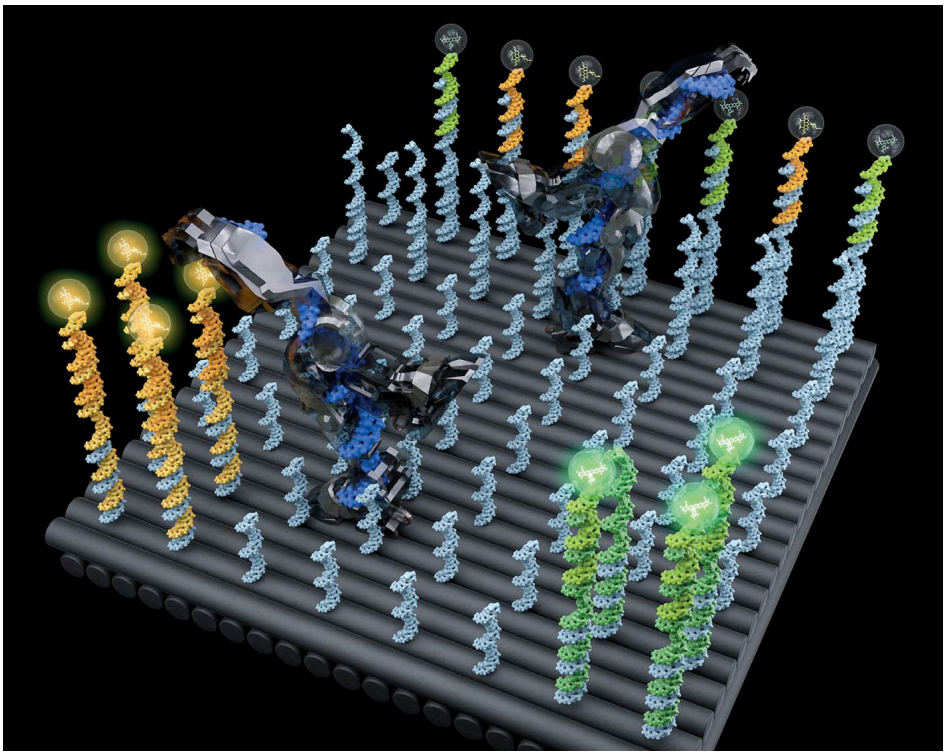


Fig.9. Adapted from [32]. Conceptual illustration of two DNA robots that collectively perform cargo-sorting task on a DNA origami surface.

The designed robots were reported to perform in average 300 steps needed for fulfilling the task. The design could be further improved to include the possibility of sorting multiple types of cargo and running multiple tasks separately, where for each task a specific number of robots is allocated depending on the task difficulty level. The developed methodology could also be used for various purposes other than sorting cargo. The authors envision the possibility applying the principles of macroscopic robots to design biomolecular robots that will be able to work in microscopic environment.

3. DNA Strand Displacement (DSD)

Various studies have proven that it is possible to describe any abstract chemical reaction network (CRN) with appropriately designed DNA strand displacement reactions. Biomolecular computation, i.e., computation performed by biomolecules, has a great potential for use in supercomputing, healthcare, smart materials, nanotechnology, and so on. Nucleic acids are particularly suited to serve as such biomolecules since these form stable structures that can be inserted into cells, and, furthermore, since the interactions between these species can be controlled to a good extent by modifying their nucleotide sequences: this essentially exploits the well-known Watson-Crick base-pairing. Current section will give an overview of DNA strand displacement process.

DNA strand displacement (DSD) is a competitive hybridization reaction. Fig. 10 gives an overview of DSD process. It is conceptually subdivided into 3 stages: toehold binding, branch migration and strand dissociation. The strand displacement is mediated by short domains called *toehold*, and long domains participate in *branch migration*. Although branch migration process was familiar to researchers since 1970s, toehold-mediated strand displacement (TMSD) has been properly introduced only in 2000 by using TMSD for designing DNA tweezers [33].

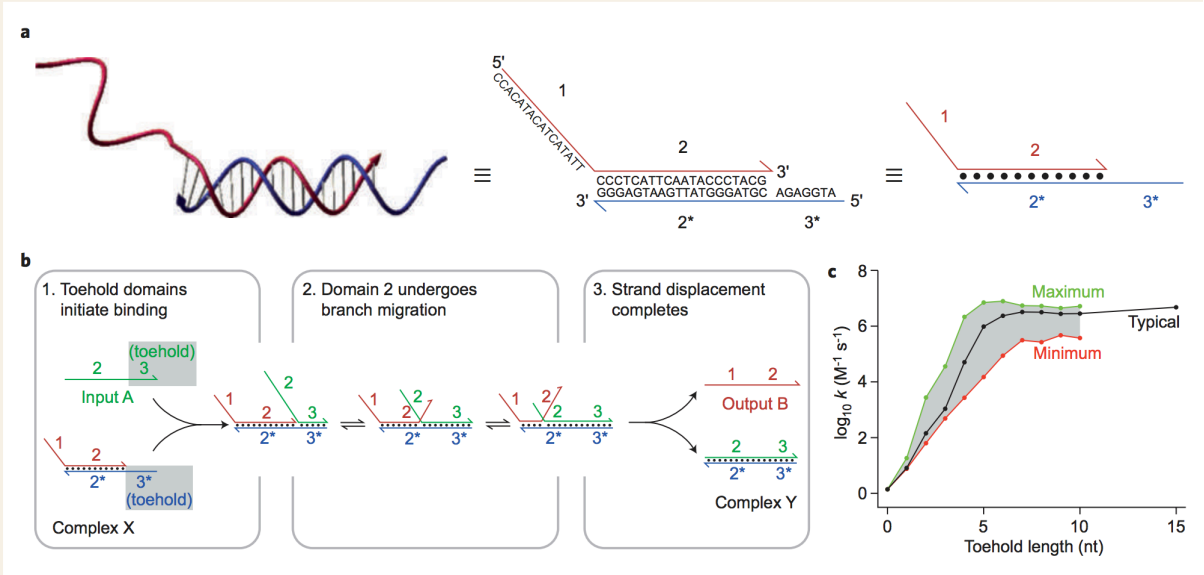


Fig.10. Adapted from [30]. Schematic illustration of DNA strand displacement process. a) illustrates double-stranded DNA molecule; b) demonstrates an example of DSD reaction; c) shows that the kinetics of strand displacement can be accurately modelled and predicted from the length and sequence of the toehold domain.

The process starts with a double-stranded DNA complex. One of the strands in the complex is called *original strand*, and another – *protector strand*. Original strand has an unbound toehold domain, which is complimentary to the toehold of the third strand – an *invading strand*, which is single-stranded DNA with nucleotide sequence complementary to the original strand. The toehold regions of original and invading strands bind together with hydrogen bonds and create a DNA complex consisting of 3 strands. The toehold binding step effects the reaction rate, which can be tuned by varying the length and sequence composition of the toehold region. After the binding of original and invading strands, the invading strand starts replacing the protector strand out of the complex through branch migration process. Eventually, the protector strand is being released from the complex and can further participate in other reactions, for which it can become an invading strand. The forward process is more energetically favored, and proceeds with reaction rate up to 6 orders of magnitude faster than the reverse reaction. This is a fundamental mechanism for genetic material exchange.

4. Chemical Reaction Networks (CRN)

To be able to discuss DNA strand displacement in more details, we need to first mention the theory that gave a basis for describing system performance with DSD. It has been shown that chemical reaction networks can be used to describe the dynamic behaviour of biological systems. Living systems rely on complex networks of chemical reactions to control the concentration of molecules in space and time. Dynamical properties of such networks were widely studied by chemists and physicists after the introduction of the law of mass action, which states that a chemical reaction rate is directly proportional to the product of the activities or concentrations of the reactants.

A chemical reaction network comprises a set of reactants, a set of products, and a set of reactions. For example, let us take the following set of chemical reactions:

$$A + B \rightarrow C + D$$
$$D \rightarrow E + F$$
$$F \rightarrow \emptyset$$

We can describe any abstract chemical reaction network (CRN) with linear ordinary differential equations (ODEs). The main focus of mathematical modelling of chemical reaction networks is on the change over time of the concentrations of the various chemical species present in the solution. Following the above set of reactions, let's assume a is the concentration of species A, b - concentration of species B, and so on. As all of these concentrations will change over time, they can be written as $a(t), b(t)$, etc.

Then we can combine them into a vector:

$$x(t) = \begin{pmatrix} a(t) \\ b(t) \\ \vdots \end{pmatrix}$$

and their evolution with time can be written as:

$$\dot{x} \equiv \frac{dx}{dt} = \begin{pmatrix} \frac{da}{dt} \\ \frac{db}{dt} \\ \vdots \end{pmatrix}$$

By solving the designed ODEs, it is possible to find the species concentration at the particular time point.

Example. Here we provide an example of multi-step Michaelis-Menten kinetics model and how it is described with corresponding ODE model:

An enzyme E interacts with substrate S, and their interaction produces an output product P:

$$E + S \underset{k^-}{\overset{k^+}{\rightleftharpoons}} E \cdot S \xrightarrow{k_{cat}} E + S + P$$

We are interested in the dynamics of the output P:

$$\frac{dp}{dt} = k_{cat} e \cdot s$$

Now we need to express the complex $E \cdot S$. Assume it reaches steady state quickly, then:

$$\frac{de \cdot s}{dt} = k^+ es - k^- e \cdot s - k_{cat} e \cdot s = 0$$

Equilibrium concentration of complex $E \cdot S$ at steady state is:

$$(e \cdot s)_{ss} = \frac{k^+}{k^- + k_{cat}} e \cdot s = \frac{1}{KM} e \cdot s$$

where KM is known as Michaelis constant.

If we further assume that the amount of substrate S is much larger compared to E, then $e^{tot} = e + e \cdot s$. After substituting this into equation of $E \cdot S$ at the steady state, we obtain:

$$e^{tot} = e + \frac{e \cdot s}{KM} \Rightarrow e = \frac{e^{tot}}{1 + \frac{s}{KM}}$$

Then we can obtain the production rate for product P as:

$$\frac{dp}{dt} = k_{cat} e^{tot} \frac{s}{KM + s}$$

This can be rewritten into more familiar form of:

$$\frac{dp}{dt} = \frac{v_{max} s}{KM + s}$$

At very high concentrations of substrate S, we obtain the maximum production rate, which is equal to v_{max}. When s is equal to KM, we are at the half-max production rate $v_{max}/2$.

4.1. Representing signals as chemical species

To ensure consistency, the notation used in [35] and [19] is used throughout in this paper. For example, a bidirectional (reversible) is represented as

$$X_1 + X_2 \underset{\delta_2}{\overset{\delta_1}{\rightleftharpoons}} X_3 + X_4$$

where X_i are chemical species with X_1 and X_2 being the reactants and X_3 and X_4 being the products. Here, δ_1 and δ_2 denote the forward and backward reaction rates, respectively. Unlike a unimolecular reaction that has only one reactant, a multimolecular reaction has two or more reactants. Degradation of a chemical species X at rate K (or conversion of X into an inert form at a rate K) is denoted by $X \xrightarrow{K} \emptyset$.

Whereas signals in systems theory can take both positive and negative values, biomolecular concentrations (with Molar (M) as unit) can only take non-negative values. Thus, following the same approach suggested in [35] and [19], we represent a signal, x as the difference in concentration of two chemical species, x^+ and x^-. Here, x^+ and x^- are respectively the positive and negative components of x such that $x = x^+ - x^-$. The consequence of adopting this scheme is that there is no unique representation for a particular signal. As an example, $x = 20$M can be represented by both $x^+ = 50$M and $x^- = 30$M or equivalently, $x^+ = 20$M and $x^- = 0$M. In practice, x^+ and x^- can be realised as single strand DNA molecules, as illustrated in [19] where these complementary positive and negative components would annihilate each other at reaction rate η (i.e. $x^+ + x^- \xrightarrow{\eta} \emptyset$). A key advantage of using this scheme is that it allows the realisation of the "subtraction" operation, as discussed further below.

Oishi and Klavins were first to show how to use idealised abstract chemical reactions to describe theoretic operators of any linear system, namely, integration, summation and gain blocks [35]. It was shown that it only requires the following three types of chemical reactions: catalysis, annihilation and degradation. In [19], this number is further reduced to only two.

Throughout the rest of the chapter, equations with superscript $\pm$ and $\mp$ are used as shorthand notations that represent the "+" and "–" individual reactions – for example, $x_i^{\pm} \xrightarrow{K} x_i^{\pm} + x_o^{\pm}$ are implied as the set of two reactions: $x_i^+ \xrightarrow{K} x_i^+ + x_o^+$ and $x_i^- \xrightarrow{K} x_i^- + x_o^-$. Likewise, the notation $x_i^{\pm} \xrightarrow{K} x_i^{\pm} + x_o^{\mp}$ is used to represent the set of two reactions: $x_i^+ \xrightarrow{K} x_i^+ + x_o^-$ and $x_i^- \xrightarrow{K} x_i^- + x_o^+$. To be in line with [35], we will represent such a set of reactions as $x_i^{\pm} \xrightarrow{K} x_i^{\pm} + x_o^{\pm}$ and $x_i^{\pm} \xrightarrow{K} x_i^{\pm} + x_o^{\mp}$.

As noted in [35], one limitation of representing signals as the difference of concentrations is that the requirement of having the same reaction rate, K, for both positive and negative components may not be easy to implement experimentally. However, as shown in [35], this requirement can be relaxed if the annihilation rate, η in the annihilation reaction, $x_o^+ + x_o^- \xrightarrow{\eta} \emptyset$ is chosen to be sufficiently large. Hence, we assume this condition of $\eta \gg K$ throughout the rest of this chapter.

4.2. Representing basic building blocks as a set of ACRNs

Below subsections illustrate how to represent basic operators such as scalar gain, integrator, summation/subtraction (linear), as well as multiplication and power of N block (nonlinear) in terms of three basic reaction types.

4.2.1. Scalar gain

Let $x_o = Kx_i$ where x_i is the input, x_o is the output and K is the gain. This operation is implemented using the following set of abstract chemical reactions:

$$x_i^{\pm} \xrightarrow{\gamma K} x_i^{\pm} + x_o^{\pm},$$
$$x_o^{\pm} \xrightarrow{\gamma} \emptyset,$$

$$x_o^+ + x_o^- \xrightarrow{\eta} \emptyset,$$

where γK, γ and η are the kinetic rates associated with catalysis, degradation and annihilation respectively.

Following generalised mass-action kinetics, it shows that the gain block can be represented with the following set of ODEs:

$$\frac{dx_o^+}{dt} = \gamma(Kx_i^+ - x_o^+) - \eta x_o^+ x_o^-$$
$$\frac{dx_o^-}{dt} = \gamma(Kx_i^- - x_o^-) - \eta x_o^+ x_o^-$$
$$\frac{dx_0}{dt} = \frac{dx_o^+}{dt} - \frac{dx_o^-}{dt} = \gamma(Kx_i - x_0)$$

According to the final value theorem, the steady state value of x_o for constant input x_i is given by $\lim_{t\to\infty} x_o(t) = Kx_i(t)$.

4.2.2. Summation

Consider the summation operation $x_o = x_i + x_d$, where x_i and x_d are the inputs and x_o is the output. The summation is implemented as follows:

$$x_i^\pm \xrightarrow{\gamma} x_i^\pm + x_o^\pm,$$
$$x_d^\pm \xrightarrow{\gamma} x_d^\pm + x_o^\pm,$$
$$x_o^\pm \xrightarrow{\gamma} \emptyset,$$
$$x_o^+ + x_o^- \xrightarrow{\eta} \emptyset.$$

Using the following set of abstract chemical reactions, the subtraction $x_o = x_i - x_d$ is implemented:

$$x_i^\pm \xrightarrow{\gamma} x_i^\pm + x_o^\pm,$$
$$x_d^\mp \xrightarrow{\gamma} x_d^\mp + x_o^\pm,$$
$$x_o^\pm \xrightarrow{\gamma} \emptyset,$$
$$x_o^+ + x_o^- \xrightarrow{\eta} \emptyset.$$

Scaled summation $x_o = K(x_i + x_d)$, and scaled subtraction $x_o = K(x_i - x_d)$, can be implemented by choosing the catalysis rates in the construct of Summation block to be γK.

4.2.3. Integration

Consider the integrator $x_o = \int x_i\, dt$ where x_i is the input, x_o is the output. Using the following set of abstract chemical reactions:

$$x_i^\pm \xrightarrow{\gamma} x_i^\pm + x_o^\pm,$$
$$x_o^+ + x_o^- \xrightarrow{\eta} \emptyset,$$

such an integrator is implemented.

The ODE representing the operation is $\frac{dx_o}{dt} = Kx_i$.

4.2.4. Multiplication

Consider the multiplication operation $x_o = x_i \times x_d$, where x_i and x_d are the inputs and x_o is the output. The multiplication is implemented as follows:

$$x_i^\pm + x_d^\pm \xrightarrow{\gamma} x_o^\pm,$$
$$x_o^\pm \xrightarrow{\gamma} \emptyset,$$
$$x_o^+ + x_o^- \xrightarrow{\eta} \emptyset.$$

The ODEs representing the operation are as follows:

$$\frac{dx_o^+}{dt} = \gamma(x_i^+ \times x_d^+ - x_o^+) - \eta x_o^+ x_{o,}^-$$
$$\frac{dx_o^-}{dt} = \gamma(x_i^- \times x_d^- - x_o^-) - \eta x_o^+ x_{o,}^-$$
$$\frac{dx_o}{dt} = \frac{dx_o^+}{dt} - \frac{dx_o^-}{dt} = \gamma(x_i \times x_d - x_o).$$

4.2.5. Power of N

Consider the operation $x_o = x_i^N$ where x_i is the input, x_o is the output and N is the constant specifying power. Similar to multiplication operation, it is implemented using the following set of abstract chemical reactions:

$$N \times x_i^{\pm} \xrightarrow{\gamma} x_o^{\pm},$$
$$x_o^{\pm} \xrightarrow{\gamma} \emptyset,$$
$$x_o^+ + x_o^- \xrightarrow{\eta} \emptyset.$$

5. From CRN to DSD – Transition rules

The transition between chemical reaction networks and its physical representation by DNA molecules was greatly facilitated in 2010 by David Soloveichik, Georg Seelig and Eric Winfree. In their paper, they explored the ways to use DNA strand displacement reactions as a primitive in order to approximate the dynamic behaviour of arbitrary chemical reactions systems [31]. The authors saw three reasons for the need of such approximation: using only CRNs was limiting the design and functionality of synthetic circuits as it could not fully capture its possibilities and limitations; there was more than extensive literature on CRNs, as well as widely used approach for approximation of CRNs with a set of suitably chosen ODEs; and finally, the fundamental model of chemical reaction system could act as a useful programming model for biomolecular system design.

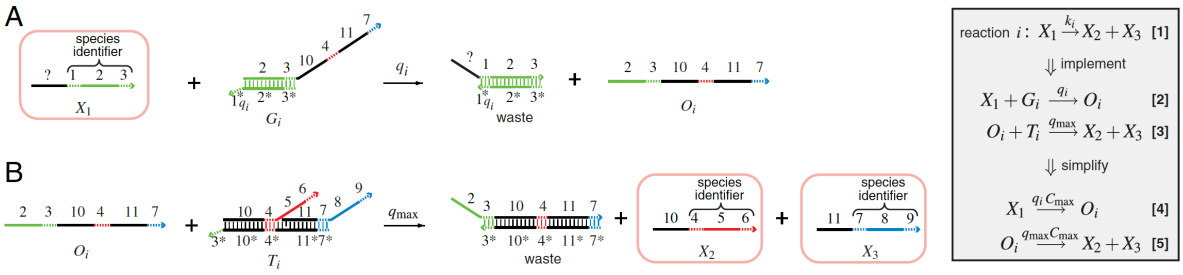


Fig. 11. Adapted from [31]. DNA implementation of unimolecular reaction $X_1 \rightarrow X_2 + X_3$. The overall DSD process consists of two reactions (A and B) where product of reaction A serves as reactant of reaction B.

In the proposed approach, each chemical species is represented as a single-stranded DNA molecule which reacts with double-stranded auxiliary species, that are present in much higher concentration than the signal species. At each step, a signal species can only react with an auxiliary species, eliminating the direct interaction. This way the auxiliary species that are present in significantly larger concentrations can mediate the correct reaction and ensure that all the required signal species participate in it.

Fig. 11 and 12 provide an illustration for a unimolecular ($X_1 \rightarrow X_2 + X_3$) and bimolecular reaction ($X_1 + X_2 \rightarrow X_3$), respectively. Signal species (X_1, X_2, X_3) are represented as 4-domains single strands, starting domain being a history domain, and the following three being a species identifier. History domain depends on the previous reaction that produced the species, and the species identifier consists of one recognition domain surrounded by two toehold domains. Two molecules with different history domains but same species identifier will

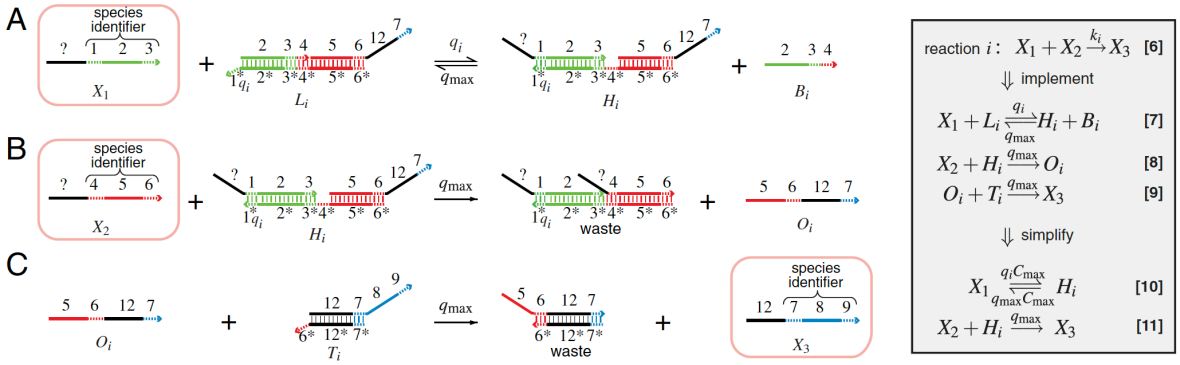


Fig. 12. Adapted from [31]. DNA implementation of bimolecular reaction $X_1 + X_2 \rightarrow X_3$. The overall DSD process consists of three reactions (A, B and C) where product of reaction A serves as reactant of reaction B, and product of B serves as reactant of reaction C

be recognised as the same species. The signal strand is considered active when it is fully unbound, and inactive otherwise.

The authors also came up with a list of system requirement that are essential for correct operation. Firstly, for all the double-stranded complexes, only toehold domains can be unbound and capable to hybridize with incoming single strands. Secondly, the toeholds should be short enough so that the toehold binding is reversible and not permanent. And lastly, only the desired target should have the correct combination of domains, which would prevent undesired reactions.

In such a way, any noncatalytic, catalytic, autocatalytic, annihilation or degradation reactions could be represented with DNA. 2-domain and 3-domain representations have also probed themselves useful in a number of other studies.

6. *In-silico* and *in-vitro* synthesis and simulation of genetic circuits

6.1. Cello

In the last several years, many studies have focused on design analysis of biomolecular circuits. Probably, one of the most noticeable recent researches is a programming language presented by MIT in 2016, that allows users to design DNA circuits that give new functions to living cells. Using this language, a user can create any function of interest, such as detecting and responding to certain environmental conditions, and the software will design the DNA sequences that can achieve it. “Cello” (a computational language) allows automatically generating circuits for highly specified physical systems and operating conditions. It constructs the desired genetic Boolean logic circuits by choosing from a library of simpler repressor-based NOT/NOR logic gates and connecting them to one other.

Cello makes use of Verilog, which is a commonly used hardware description language for programming computer chips. A user specifies the desired circuit function in Verilog code that captures the desired computational operation. Further, the functional details are transformed into a ready DNA sequence in 3 steps:

1) Verilog text code is converted into a truth table and further into a circuit diagram;
2) Given the circuit diagram, specific regulators are assigned to each gate in the diagram. Since gates based on different regulators may have different response functions, not all repressors can be paired together. The challenge here is that they may interfere with each other once placed in the complex environment of a living cell. To resolve this issue and help identify correct connections, Cello uses simulated annealing principle;
3) From a circuit diagram and regulators assignment, Cello can now create a DNA sequence using combinatorial design.

Cello language is easy to use and doesn't require much of a prior knowledge of programming which makes it easily accessible. This development allowed to save huge amount of time for building complex circuits. Having a ready DNA sequence that you can test, greatly reduces the time for obtaining successful results. During the experimentation, 40 out of 60 designed circuits with different functions operated correctly from the first time.

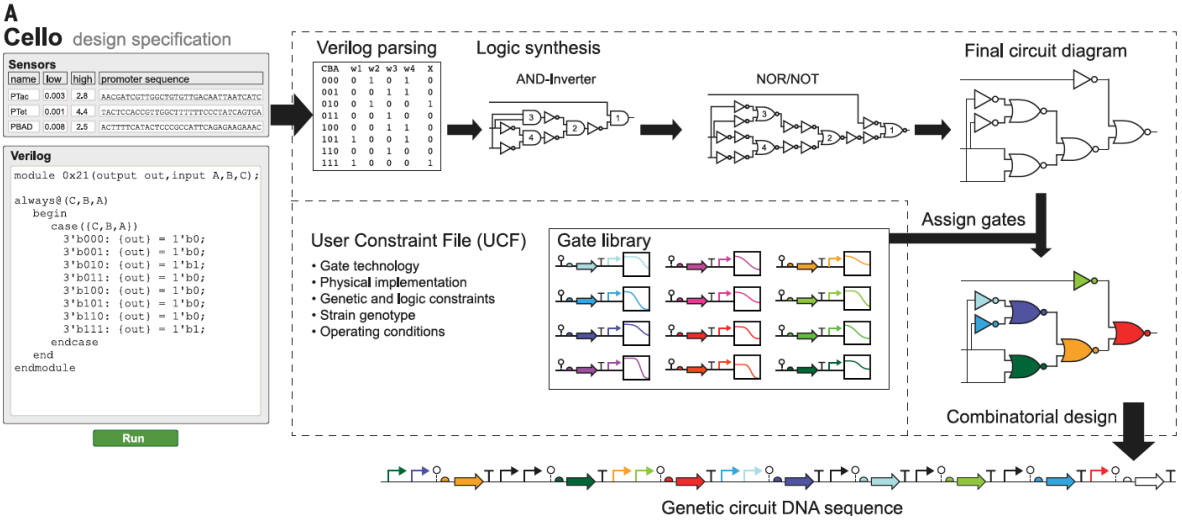


Fig. 13. Adapted from [32]. Overview of Cello computation process. First step is user input, then Verilog code conversion to circuit diagram, followed by assigning of specific regulators to each gate, and, finally, creation of DNA sequence of genetic circuit.

Future applications for this kind of programming include designing bacterial cells that can release a cancer drug when they detect a tumour, or creating yeast cells that can pause their own fermentation process if too many toxic byproducts build up.

6.2. Station B

Another promising platform for programming biology announced in 2019 is Station B by Microsoft Research. The platform is a result of over 20 years of cooperation between Microsoft and their technology, academic and commercial partners which allowed to develop new methods and technology for programming biology.

The goal of the platform is to improve each step of the Design-Build-Test-Learn workflow in the process of programming biological systems. On the first stage, the biological programming languages will be collected into a hierarchy of biological abstractions, each with their associated methods of analysis. Next, high-level programs will be translated to DNA code, accounting for the biological experiments that are intended. Further, biological experiments will be performed using lab robots, and lastly, the received experimental data will be used as a training data by a wide range of learning methods. As new experiments are performed, the knowledge base will be updated via automated learning.

Station B is a promising tool that uses computational models to describe interactions within a cell that are too complex for humans to parse using notebooks and spreadsheets alone. It will allow scientist to make use of underlying knowledge that has been collected for decades and perform their work more efficiently, reliably and cost-effectively.

6.3. Visual DSD

Today, a wide range of tools is available to model the molecular conformations of nucleic acid strands, that are useful in the design of DNA-based biomolecular circuits. The software Visual DSD [33-35] is a good choice for modelling a broad class of strand displacement systems, including all that can be expressed in terms of chemical reactions.

Visual DSD is a programming language for designing and simulating computational devices made of DNA. Visual DSD uses DNA strand displacement as the main computational mechanism, which allows devices to be designed solely in terms of nucleic acids. It accounts for system complexity and the potential unwanted interference between molecules in the system. After inputting a collection of DNA strands of interest, the software will show how these DNA strands interact with each other. It can also be used to predict their behaviour over time. Visual DSD can be used to detect and fix bugs *in-silico* before the circuit is attempted to be built *in-vitro* or *in-vivo*.

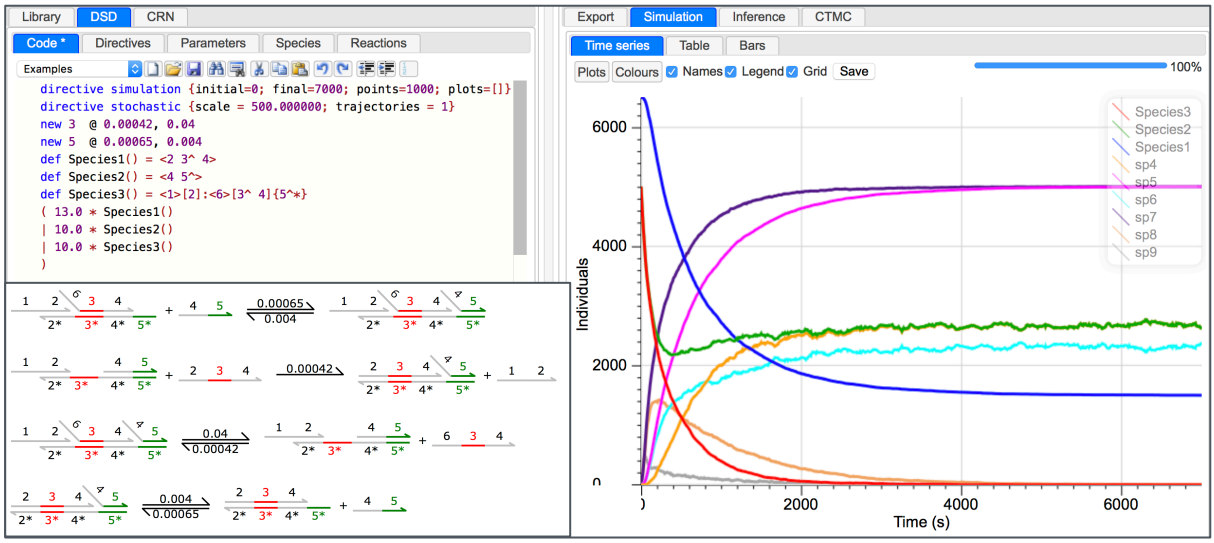


Fig. 14. Illustration of Visual DSD interface and functionality. Left top side contains molecular program coded in Visual DSD language, left down side contains the reactions generated by the software, and right side contains simulation results that show the change of species concentration over time.

Visual DSD compiles a collection of DNA molecules into a chemical reaction network (CRN). It also includes a stochastic simulator, which computes a possible trajectory of the system and plots the populations of species over time, together with a deterministic simulator, which forms and solves an Ordinary Differential Equation (ODE) representation of the dynamics of the system. The reachable state space of the system can also be constructed as a continuous time Markov chain (CTMC).

Conclusions and future remarks

For the last two decades, the field of synthetic biology experienced a substantial growth and has produces many notable achievements. Moreover, the speed of the technical progress increases gradually, as more and more discoveries are made. As synthetic circuits became more complex, there was a need for moving away from traditional design techniques, and the usage of nucleic acids for circuit design has successfully filled that niche. Programming with DNA has a number of advantages, including its high programmability due to Watson-Crick base pairing, structure stability which allows the structures to be inserted into cells, no requirement for any additional components, precise control of organisation and dynamics at the molecular level, and, finally, ability to interact organically with other biological entities. The range of applications for using DNA as a substrate for circuit design varies greatly, from healthcare to smart materials and to nanotechnology.

The field of DNA programming evolved greatly till present day. Started with using DNA for Turing machine in C. Bennet's theoretical work in 1982, it came a long way till the complex DNA robots, DNA origami, and programmable DNA controllers. However, many problems are still open for solving, including "leaky" reactions, fail in modularity, and methods for control of kinetic parameters. One of the possible solutions to overcome one or more of these problems

could be the use of xeno nucleic acids (XNA) as information carriers instead of DNA. XNA is a synthetic analogue of DNA that has a different sugar backbone structure. Currently there are at least six types of successfully synthesised XNAs. The discovery of XNA gave a start to the field of xenobiology which describes the use of novel biological systems that may not yet be familiar to the research community [41].

Another promising novel approach, currently very popular amongst synthetic biologists, is the use of CRISPR/Cas9 for targeted genome editing. It is an enzyme that allows to identify and cut desired DNA strands and is target-specific [42]. This technology gives high versatility for producing user-guided mutations within the DNA and alteration of the genome.

All this suggests that there will be less and less restrictions in technical and, possibly, cost aspects of the circuit synthesis in the near future, but the field is still behind on the full understanding of all the biological and technical aspects that are required to design flawless biomolecular systems.

Acknowledgements

This research is supported, in parts, by the EPSRC INDUSTRIAL CASE AWARD (CASE Voucher 16000070), Microsoft Research, and the EPSRC/BBSRC grant BB/M017982/1 to the Warwick Integrative Synthetic Biology Centre.

References:

[1] Monod, J. *et al.* **Cold Spring Harbor Sympos. on Quantit. Biol.** 26, 389-401,1961.
[2] Elowitz, M. B. *et al.* **Nature** 403 (6767), 335-338, 2000.
[3] Gardner T. S. *et al.* **Nature** 403 (6767), 339-342, 2000.
[4] Padirac, A. *et al.* **Curr. Opin. in Biotech**. 24, 575-580, 2013.
[5] Delebecque, C. J. *et al.* **Science** 333, 470-474, 2013.
[6] Douglas, S. M. *et al.* **Science** 335, 831-834, 2012.
[7] Daniel, R. *et al.* **Nature** 497, 619-623, 2013.
[8] Zhang, D. Y. *et al.* **Nat. Chem**. 3, 103-113, 2011.
[9] Montagne, K. *et al.* **Mol. Syst. Biol.** 7, 466, 2011.
[10] Kim, J. *et al.* **Mol. Syst. Biol.** 7, 465, 2011.
[11] Chen, Y.-J. *et al.* **Nat. Nanotech**. 8, 755-762, 2013.
[12] Fujii, T. *et al.* **ACS Nano** 7, 27-34, 2013.
[13] Weitz, M. *et al.* **Nat. Chem**. 6, 295-302, 2014.
[14] Franco, E. PhD Dissertation, CalTech, 2012.
[15] Yordanov B. *et al.* **ACS SynBio**, 3, pp. 600-616, 2014.
[16] Song T. *et al.* **ACS SynBio**, 5(8), pp 898-912, 2016.
[17] Zhou C. *et al.* **ACS Omega**, 2(8), pp. 4143-4160, 2017.
[18] Lakin M. *et al.* **ACS Synth. Biol**., 5(8), pp. 885-897, 2016.
[19] Hartwell L. *et al.* **Nature**, 402,47-52, 1999.
[20] Lauffenburger D. **Proc. Natl. Acad. Sci.** U.S.A., 97(10), 5031-5033, 2000.
[21] Purnick P. *et al.* **Nat. Rev. Mol. Cell. Biol**., 10(6),410-422, 2009.
[22] Andrianantoandro E. *et al.* **Mol. Syst. Biol**., 2(1), 2006.
[23] Bennett C. H. **Internat. Jour. of Theor. Phys.**, 21, no. 12, 1982.
[24] Adleman L. M. **Scient. Americ.**, 54-61, 1998
[25] Yin P. *et al.* **Angew. Chem. Int. Ed.**, 43, 4906-4911, 2004.
[26] Li J. *et al.* **Nature Nanotech.**, 13, 723-729, 2018.
[27] Rothemund P. W. K. **Nature**, 440, 297-302, 2006.
[28] Seelig G. *et al.* **Science**, 314, 5805, 1585-1588, 2006.
[29] Qian L. *et al.* **Science**, 332, No. 6034, 1196-1201, 2011.
[30] Xie Z. *et al.* **Science**, 02, 333, 6047, 1307-1311, 2011.
[31] Pardee K. *et al.* **Cell**, 165, 1255-1266, 2016.
[32] Thubagere J. *et al.* **Science**, 357, 1112, 2017.
[33] Yurke B. *et al.* **Nature**, 406(6796):605-8, 2000.
[34] Zhang D. Y. *et al.* **Nature Chem.**, vol. 3, 103-113, 2011.
[35] Oishi K., *et al.* **IET Syst. Biol.**, 5, iss. 4, 252-260, 2011.
[36] Soloveichik D. *et al.* **PNAS**, 107, no. 12, 5393-5398, 2010.
[37] Nielsen A. K. *et al.* **Science**, 112, iss. 6281, aac7341, 2016.

[38] Phillips A. *et al.* **Journ. of Royal Soc. Interf.**, 6 Suppl 4, 419-436, 2009.
[39] Lakin M. R., *et al.* **Bioinformatics**, 27(22), 3211-3213, 2011.
[40] Lakin M. R., *et al.* **Journ. of Royal Soc. Interf.**, 9(68), 470-486, 2012.
[41] Anosova I. *et al.* **Nucleic Acids Res.** 18, 44(3), 1007-1021, 2016.
[42] Jinek, M., *et al.* **Science**, 337, 816-821, 2012.