Manuscript version: Author’s Accepted Manuscript
The version presented in WRAP is the author’s accepted manuscript and may differ from the published version or Version of Record.

Persistent WRAP URL:
http://wrap.warwick.ac.uk/127001

How to cite:
Please refer to published version for the most recent bibliographic citation information. If a published version is known of, the repository item page linked to above, will contain details on accessing it.

Copyright and reuse:
The Warwick Research Archive Portal (WRAP) makes this work by researchers of the University of Warwick available open access under the following conditions.

Copyright © and all moral rights to the version of the paper presented here belong to the individual author(s) and/or other copyright owners. To the extent reasonable and practicable the material made available in WRAP has been checked for eligibility before being made available.

Copies of full items can be used for personal research or study, educational, or not-for-profit purposes without prior permission or charge. Provided that the authors, title and full bibliographic details are credited, a hyperlink and/or URL is given for the original metadata page and the content is not changed in any way.

Publisher’s statement:
Please refer to the repository item page, publisher’s statement section, for further information.

For more information, please contact the WRAP Team at: wrap@warwick.ac.uk.
A method for compiling arbitrary transfer functions to molecular circuits

Iuliiia Zarubiieva  
University of Warwick  
Coventry CV4 7AL, UK  
i.zarubiieva@warwick.ac.uk

Andrew Phillips  
Microsoft Research  
Cambridge CB1 2FB, UK  
andrew.phillips@microsoft.com

Francesca Mantellino  
University of Warwick  
Coventry CV4 7AL, UK  
francesca.mantellino@warwick.ac.uk

Vishwesh Kulkarni  
University of Warwick  
Coventry CV4 7AL, UK  
v.kulkarni@warwick.ac.uk

1 INTRODUCTION

As an engineering substrate, DNA is well suited for the construction of biochemical circuits and systems because it is simple enough that its interactions can be rationally designed using the Watson-Crick base pairing rules while ensuring that the design space is remarkably rich [1]. DNA Strand Displacement (DSD) is an implementation strategy based on the hybridization of DNA strands with partial or full complementarity, resulting in the displacement of one or more pre-hybridized strands [1], [2], [3]. Visual DSD is a graphical user interface (GUI) aided software platform that enables an in silico design and simulation of DSD-based circuits [4], [5]. If a given system is modelled as either a set of DSD reactions or as an Abstract Chemical Reaction Network (ACRN) then it can be simulated by running the corresponding text file in, respectively, the “DSD” or the “CRN” mode of Visual DSD. Furthermore, it supports the notion of a degree of complementarity which makes it possible to model interactions between domains that are not exactly complementary but instead contain one or more mismatched bases [4].

One of the difficulties in designing DSD systems is that the end result is quite sensitive to the values of parameters – typically, these include the reaction rates, concentrations, the degree of complementarity, etc.

An interesting reference point is due to the semiconductor-based electronic circuits, the development of which started in the 1940’s. The integrated circuit design and optimization used to be a highly time-intensive manual process until it got accelerated by the development of the hardware description languages (HDLs) that enabled a user to design an electronic system through textual commands that are transformed into a physical implementation of the circuit in silicon. Recently, Cell10 has applied this approach to genetic circuits to transform an HDL design into a linear DNA sequence that can be constructed and run in living cells [6]. However, Cell10 does not facilitate the design of computational nucleic acid devices. In [3], a general purpose CRN-to-DSD compiler Nuske11 has been developed and its benefit has been illustrated through interesting applications [7]. Unlike Visual DSD which uses a bottom-up approach, Nuske11 uses a top-down approach and it is argued in [7] that “in a bottom-up approach it is not obvious whether a particular DSD implementation or its components can be generalized to implement different algorithms or whether conceptually new modules are required”. In this manuscript, we show how such generalizations can be obtained.

2 OUR COMPILERS: RATIONALE AND OVERVIEW

Any linear time-invariant (LTI) system S can be represented by a so-called transfer function (TF), which is a frequency-domain representation. This, in turn, can be expressed as a connection of a finite number of integrators, scalar gains, and summation blocks, as shown in Figure 1(C and D). Each of these component blocks can be realised by a set of DSD reactions. Each of these component blocks can also be realised through a well-known set of DSD reactions. Also, the particle swarming optimisation (PSO), summarised in [8], can be adopted to optimise the reaction parameters. So, we have synthesized two MATLAB-based compilers (1) TF-to-CRN: its output is a text file of an ACRN representation of S which should be run in the “CRN” mode of Visual DSD and (2) TF-to-DSD: its output is a text file of a DSD representation of S which should be run in the “DSD” mode of Visual DSD. Our compilers support 2-domain, 3-domain, and 4-domain representations: in general, the representation is called $k$-domain if each chemical species is implemented as a single strand of DNA comprising $k$ distinct domains. The operation of our TF-to-DSD compiler can be summarised as follows; TF-to-CRN can be explained similarly:

1. **Step 1**: In the GUI of TF-to-DSD, the user inputs TF of the LTI system that they want to design.
2. **Step 2**: In the GUI of TF-to-DSD, the user inputs the number of domains of the desired DSD architecture.
Figure 1: Our TF-to-DSD compiler in action. A: This Visual DSD screenshot illustrates the DSD code to implement a low-pass filter (LPF), given in C, generated by our TF-to-DSD compiler. B: The output of this LPF when excited with a staircase input—the red color plot represents the input and the green color plot represents the output; in this case, ideally, both plots should overlap, which can be ensured after some parameter tuning. D: In the generated code, all necessary blocks such as “Scaled Summation” (see line # 73) can be created and modified on the fly. To obtain the output B and the DSD design for the TF given in C, the user need only enter 5 numbers in the GUI E and click on a DSD architecture choice in the GUI C.

(3) **Step 3:** TF-to-DSD computes a minimal state-space model M for this TF; M comprises a finite number of integrators, scalar gains, and summation blocks.

(4) **Step 4:** For each block, building on [2], TF-to-DSD creates a text file containing (1) the DNA strand compositions and (2) a minimal set of DSD reactions.

(5) **Step 5:** Using PSO, TF-to-DSD optimises all DSD parameters such as the concentrations, chemical reaction rates, the degree of complementarity.

The resulting text file can now be run in the “DSD” tab of Visual DSD for the simulation and analysis purposes, and to start the process for the wet-lab implementations.

3 RESULTS AND DISCUSSION

As illustrated in Figure 1, our compiler gives satisfactory results. Also, we have shown how the bottom-up architecture of Visual DSD can be adequately generalised to implement not only different algorithms but also to synthesise new conceptual modules on the fly for LTI systems. Our approach can be generalised for dynamic nonlinear systems as well. Besides speeding up the in silico design of DSD-based circuits, our compilers also increase the outreach of DNA computation to users who might only be comfortable with mathematical models and MATLAB/Python.

Acknowledgments

This research is supported, in parts, by the EPSRC INDUSTRIAL CASE AWARD (CASE Voucher 16000070), Microsoft Research, the EPSRC/BBSRC grant BB/M017982/1 to the Warwick Integrative Synthetic Biology Centre, and the Erasmus+ grant to the University of Warwick (Agreement # 2018-1-UK01-KA107-047454).

REFERENCES