CONCENTRATION-BASED POLYNOMIAL CALCULATIONS ON NICKED DNA

Tonglin Chen, Marc Riedel

University of Minnesota, Twin-Cities
Department of Electrical and Computer Engineering
chen5202@umn.edu, mriedel@umn.edu

ABSTRACT

In this paper, we introduce a novel scheme for computing polynomial functions on a substrate of nicked DNA. We first discuss a fractional encoding of data, based on the concentration of nicked double DNA strands. Then we show how to perform multiplication on this representation. Next we describe the read-out process, effected by releasing single strands. We show how to perform simple mathematical operations such as addition and subtraction, as well as how to scale constant values using probabilistic switches. We also describe two complex operations: calculating a vector dot product and computing a general polynomial function. We conclude by discussing potential applications of our scheme, practical challenges, and future research directions.

Index Terms**—** DNA Computing, Stochastic Computing, DNA Strand Displacement,

1. INTRODUCTION

This paper demonstrates novel schemes for implementing operations such as multiplication, dot product, and polynomial functions on data stored in DNA. Computation is effected via the mechanism of of *toehold-mediated DNA strand displacement* [1], [2]. Recent research has shown how data can be encoded via *nicks* on DNA using gene-editing enzymes like CRISPR-Cas9 and PfAgo [3]. *Probabilistic switching* of concentration values has been demonstrated by the DNA computing community [4]. In previous work, we demonstrated how a concept from computer engineering called *stochastic logic* can be adapted to DNA computing [5]. In this paper, we bring these disparate threads together: we demonstrate how to perform stochastic computation on *fractionally-encoded* data stored on nicked DNA.

2. ENCODING VALUES

The conventional approach to storing data in DNA is to use a single species of strand to represent a value. It is either encoded as a binary value, where the presence of the specific strand represents a 1 and its absence a 0 [6]; or as a non-integer value, encoded according to its concentration, called a *direct representation* [7]. In recent research, we have shown how a *fractional representation* can be used [5]. The idea is to use the concentration of two species of strand X_0, X_1 to represent a value x with

$$x = \frac{X_1}{X_0 + X_1}$$

where $x \in [0, 1]$. This encoding is related to the concept of *stochastic logic* in which computation is performed on randomized bit streams, with values represented by the fraction of 1's versus 0's in the stream [8], [9], [10].

In this work, we store values according to nicking sites on double DNA strands. For a given site, we will have some strands nicked there, others not. Let the overall concentration of the double strand equal C_0, and the concentration of strands nicked at the site equal C_1. The ratio of the concentration of strands nicked versus the overall concentration is

$$x = \frac{C_1}{C_0}$$

So this ratio is the relative concentration of the nicked strand at this site. We use it to represent a variable $x \in [0, 1]$.

Setting this ratio can be achieved by two possible methods. One is that we nick a site using a gene-editing guide that is not fully complementary to the nicking site. The degree of complementarity would control the rate of nicking and so set the relative concentration of strands that are nicked. A simpler method is to split the initial solution containing the strand into two samples; nick all the strands in one sample; and then mix together the two samples with the desired ratio x.

3. MULTIPLICATION

The core component of our design is the multiplication operation. It requires that the presence of nicks at a given sites be statistically independent to the presence of nicks at other sites on the same strand.

This work was funded by DARPA Grant #W911NF-18-2-0032. We thank David Soloveichik, Olgica Milenkovic and Boya Wang for helpful discussions.

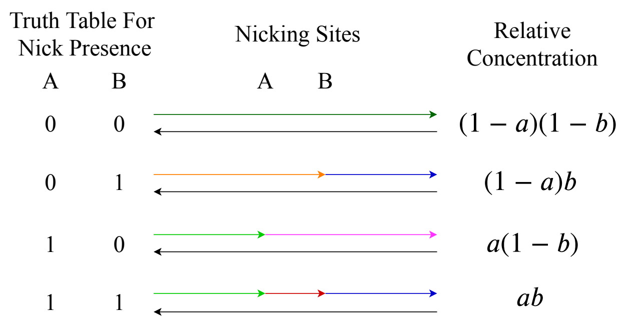

Fig. 1. Multiplying two values, a and b. The result is the relative concentration of the red strand.

3.1. Multiplying two values

Multiplying two fractionally encoded values requires two neighboring nicking sites. Suppose we have two such sites A and B on a double strand. For each site, there are two possibilities: nicked or not nicked. Let a and b be the relative concentrations of strands nicked at sites A and B, respectively. We assume the nicking is done independently. Figure 1 shows four different possible scenarios on a given strand. The truth table on the left shows whether a specific location is nicked or not, and the concentration on the right shows the relative concentration for each scenario. Here we use the single strand between A and B in the last scenario, shown in red, to represent the result of the multiplication. The concentration of this strand specifies the result since it only appears when both sites A and B are nicked. According to our assumption, the presence of nicks at A and B are statistically independent. So the probability that both sites are nicked is $a \times b$. In Section 4, we discuss how to release the single strand between the two sites. Its relative concentration translates to the value of the multiplication operation, $a \times b$.

3.2. Multiplying three or more values

We can extend this scheme to multiply three or more values at once. We again use the concentration of one species of single-stranded DNA to represent the result. Suppose we have a vector of fractional values $x_1, x_2, ..., x_n$. We operate on N neighboring nicking positions $A_1, A_2, ..., A_n$, ordered sequentially, on a double-stranded DNA complex. To compute the product, positions A_1 and A_n should be nicked with probability x_i and positions A_2 to A_{n-1} should be nicked with probability $1 - x_i$ (so *not* nicked with probability x_i). Therefore the probability that the strand is nicked at sites A_1 and A_n but *not* in between will be $\prod_{i=1}^{n} x_i$. Again, we assume that the nicking at each site is statistically independent. Below, we discuss how to release the single strand between the two end sites. Its relative concentration translates to the value of the multiplication operation, $\prod_{i=1}^{n} x_i$.

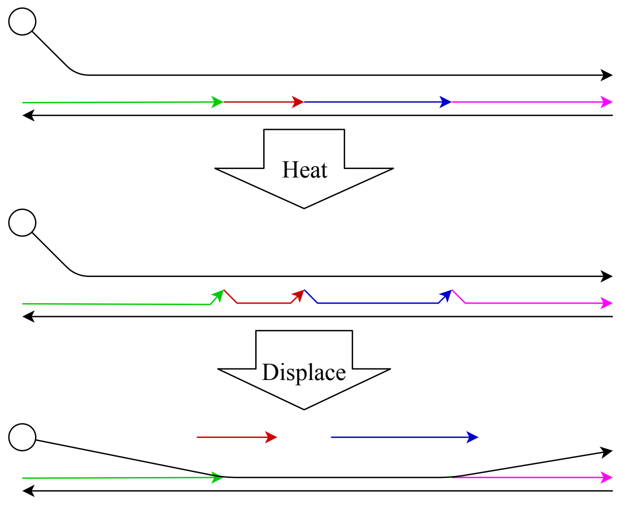

Fig. 2. Using a *probe* strand to read out the result. When a complementary single strand of DNA, called a *probe*, shown as the top black strand, is added, it displaces and releases the single strands from regions in between nicks.

4. READING OUT

To process the result of multiplication, we need to detach single strands from the base double strand. The diagram shown in Figure 2 illustrates this process. When a complementary single strand called a *probe* is supplied, it will displace and release single strands located between nicks, provided that the distance between nicks is small [11]. In Section 6, we discuss how these single strands can then participate in further strand-displacement operations.

5. APPLICATION: DOT PRODUCT

Suppose we have two vectors of fractional values: x_1, x_2, x_3, x_4 and y_1, y_2, y_3, y_4, where $x_i, y_i \in [0, 1]$, and we want to calculate the dot product of the two vectors, $\sum_{i=1}^{4} x_i y_i$. Figure 3 illustrates the process. First we encode the values on DNA in the following order: $x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4$. Each pair represents the calculation of one term $x_i * y_i$ using the sequence of operations discussed in Section 3. The resulting strands are A_1, A_2, A_3, A_4, respectively. Using probes, these strands are released, separately but in parallel, for subsequent reactions.

Next, using DNA strand displacement operations [1], [2], we translate each of the released strands to a common single strand, S:

$$A_i \rightarrow S$$

(The details are omitted here due to space constraints.) The accumulated concentration s will be the the sum of the concentrations of released strands. (As always, s as the sum is a

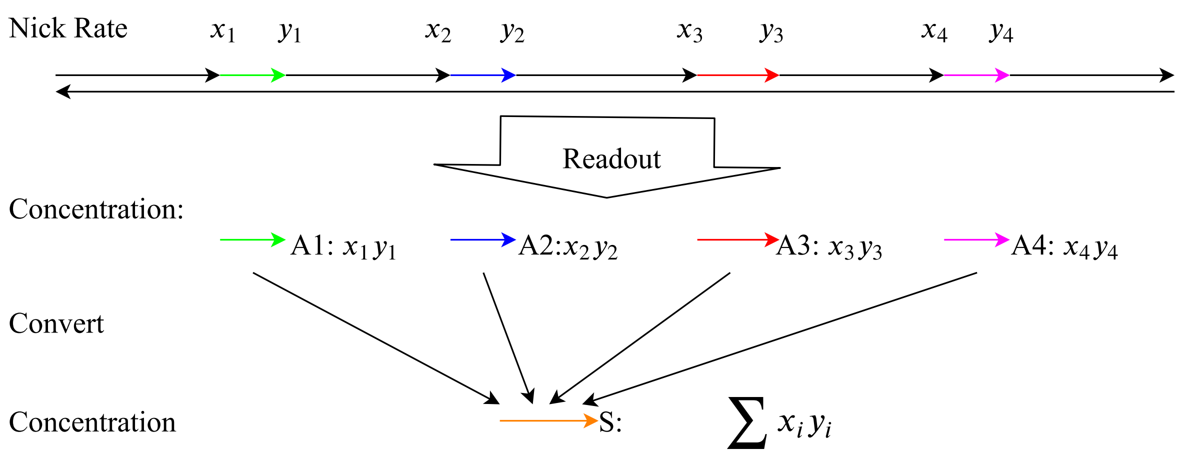

Fig. 3. Calculating a Dot Product

relative concentration. Therefore, it cannot exceed the maximum value, corresponding to $s = 1$, since the meaning is undefined.) This is analogous to a "hard-wired" sum in computer engineering.

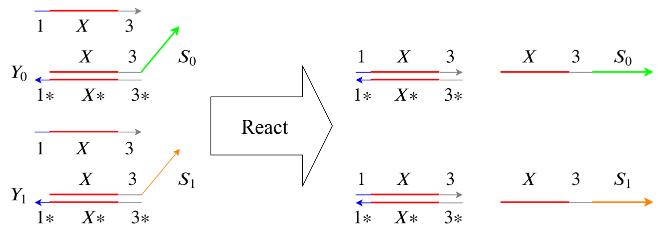

Fig. 4. Probabilitic Switch. The concentration of the input X is scaled by a factor $a/(a+b)$ to S_0 given input strands Y_0 and Y_1 with concentrations set to the ratio $a : b$.

6. SCALING

To further extend the current scheme, we use a scaling operation. This can be implemented with the *probabilistic switch* scheme proposed by Cherry et al. [4]. It consists of a competitive pair of strand displacement operations, $X \rightarrow S_0$ and $X \rightarrow S_1$. These operate on double-stranded DNA complexes Y_0 and Y_1 which are set with a ratio of concentrations $a : b$. The pair of reactions will scale the input concentration of X by a factor $a/(a+b)$ to produce the output strand S_0, and by a factor $b/(a+b)$ to produce the output strand S_1. This process is illustrated in Figure 4. Again, due to space constraints, further details are omitted.

To scale the concentration of a strand X by a constant factor $c \in [0,1]$, we set the ratio of the the concentrations of Y_0 and Y_1 to be $c : (1-c)$. The output strand is S_0. (Strand S_1 is be considered waste in this scenario). We annotate this operation as

$$X \xrightarrow{c} S_0$$

where $c \in [0,1]$ as the scaling constant.

7. APPLICATION: EVALUATING A POLYNOMIAL FUNCTION

We present a scheme to to calculate polynomials of the form of $\sum(-1)^k C x^i$, where $k \in \{0,1\}$, $C, x \in [0,1]$, using scaling with the scheme discussed in Section 6. Figure 5 shows an example. We evaluate

$$f(x) = 1 - x + \frac{1}{2!}x^2 - \frac{1}{3!}x^3,$$

the first four term of the Taylor series expansion of $f(x) = e^{-x}$.

First we encode the following fractional values on the strand: $1, x, x, x, x, 1-x, x$. The first two terms generate strand A with concentration x; the third and fourth terms generate strand B with concentration x^2; and the last three terms generate strand C with concentration x^3.

After reading out strands A, B and C, we perform scaling by the magnitude of the constant of the corresponding term, using the operation discussed in Section 6. If the term is positive, we translate the concentration into a common strand P; else if it is negative, we translate its concentration into a common term N. The exact scaling factors for the target polynomial above are:

$$A \xrightarrow{100\%} N$$

$$B \xrightarrow{50\%} P$$

$$C \xrightarrow{16.67\%} N$$

We also prepare strand P with relative concentration 100% to represent the constant 1 in the function.

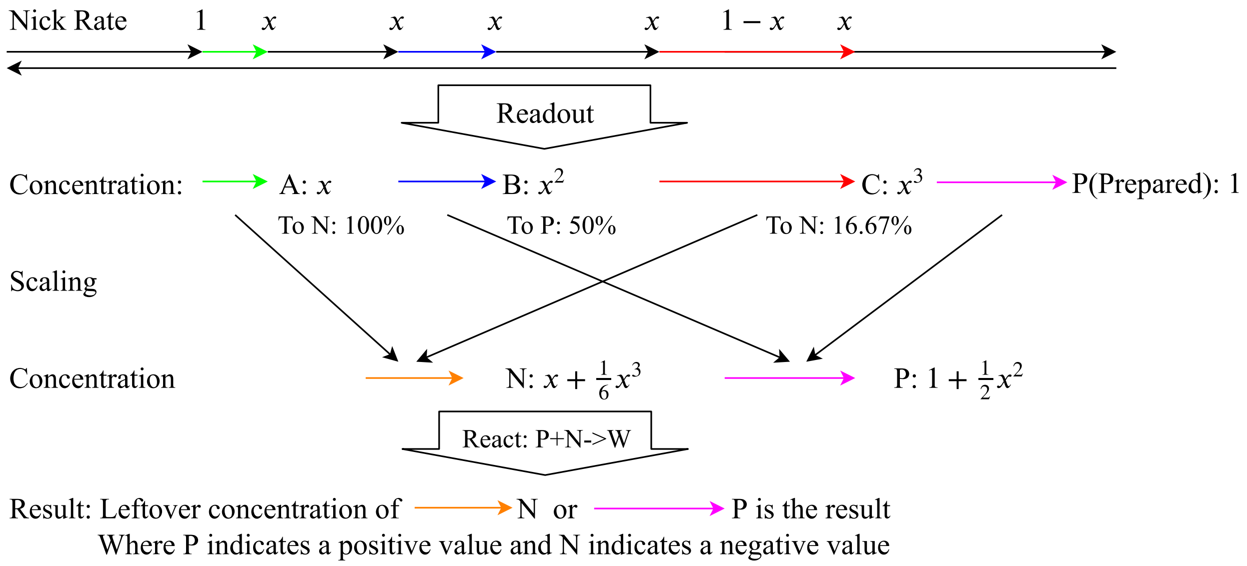

Fig. 5. Evaluation of a Polynomial Function: the Taylor series expansion of e^{-x}.

In the final step, we effect the following transformation, vi strand displacement:

$$P + N \rightarrow W$$

Here W represents a waste product. The purpose of the reaction is to perform the "minus" operation. The result of evaluating the polynomial is the leftover of either strand P or N. Which species survives represents the sign of the result; its concentration represents the numeric part of the result.

8. DISCUSSION

In this paper, we proposed a novel scheme to perform mathematical operations with DNA. We demonstrated simple operations such as multiplication, as well as reasonably complex ones, such as evaluating polynomials functions. The read out process and the scaling process are highly parallel, since each term in the polynomial is encoded independently. We could potentially scale this method to evaluate a polynomial with a large number of terms efficiently, exploiting the inherent parallelism of DNA computing. We note that we have only validated these results through simulation. We are collaborating with the Soloveichik group at UT Austin to validate them experimentally.

There are a number of practical challenges. One of the concerns, ubiquitous with DNA strand displacement operations, is "leakage", that is to say errors in transforming concentrations. This occurs because we never have 100% of DNA strands participating in designated reactions. Based upon the actual experimental results, we might have to mitigate leakage with error correction methods or adopt so-called "leakless" designs [12].

A future direction of research is investigating how to reencode the result of a a computation back into the relative concentration of strands nicked at specific sites, so to end with the same data encoding that we started with. This would allow us to cascade computation, for instance using the result of a polynomial function as the input for another calculation directly. Cascading operations this way would allows us to explore interesting parallel algorithms, such as multi-layer convolutional neural networks.

9. REFERENCES

[1] Bernard Yurke, “A dna-fuelled molecular machine made of dna,” *Nature*, vol. 406, no. 6796: 605, 2000.

[2] David Soloveichik, Georg Seelig, and Erik Winfree, “Dna as a universal substrate for chemical kinetics,” *Proceedings of the National Academy of Sciences*, vol. 107, no. 12, pp. 5393–5398, 2010.

[3] S Kasra Tabatabaei, Boya Wang, Nagendra Bala Murali Athreya, Behnam Enghiad, Alvaro Gonzalo Hernandez, Jean-Pierre Leburton, David Soloveichik, Huimin Zhao, and Olgica Milenkovic, “Dna punch cards: Encoding data on native dna sequences via nicking,” *bioRxiv*, 2019.

[4] Kevin M. Cherry and Lulu Qian, “Scaling up molecular pattern recognition with dna-based winner-take-all neural networks,” *Nature*, vol. 559, no. 7714, pp. 370–376, Jul 2018.

[5] Sayed Ahmad Salehi, Xingyi Liu, Marc D. Riedel, and Keshab K. Parhi, “Computing mathematical functions using dna via fractional coding,” *Scientific Reports*, vol. 8, no. 1, pp. 8312, May 2018.

[6] Georg Seelig, David Soloveichik, David Yu Zhang, and Erik Winfree, “Enzyme-free nucleic acid logic circuits,” *Science*, vol. 314, no. 5805, pp. 1585–1588, 2006.

[7] Daniel Wilhelm, Jehoshua Bruck, and Lulu Qian, “Probabilistic switching circuits in dna,” *Proceedings of the National Academy of Sciences*, vol. 115, no. 5, pp. 903–908, Jan 2018.

[8] B. R. Gaines, *Stochastic Computing Systems*, pp. 37–172, Springer US, Boston, MA, 1969.

[9] W. Qian and Marc Riedel, “The synthesis of robust polynomial arithmetic with stochastic logic,” in *Design Automation Conference*, 2008, pp. 648–653.

[10] W. Qian, X. Li, Marc Riedel, K. Bazargan, and D. J. Lilja, “An architecture for fault-tolerant computation with stochastic logic,” *IEEE Transcations on Computers*, , no. 1, pp. 93–105, 2011.

[11] Gary S. Hayward, “Unique double-stranded fragments of bacteriophage t5 dna resulting from preferential shear-induced breakage at nicks,” *Proceedings of the National Academy of Sciences*, vol. 71, no. 5, pp. 2108–2112, 1974.

[12] Boya Wang, Chris Thachuk, Andrew D. Ellington, Erik Winfree, and David Soloveichik, “Effective design principles for leakless strand displacement systems,” *Proceedings of the National Academy of Sciences*, vol. 115, no. 52, pp. E12182–E12191, 2018.