Heliyon 5 (2019) e02901

Contents lists available at ScienceDirect

Heliyon

journal homepage: www.cell.com/heliyon

Research article

A new DNA-based model for finite field arithmetic

Iván Jirón [a,*], Susana Soto [a], Sabrina Marín [b], Mauricio Acosta [b], Ismael Soto [c]

[a] Departamento de Matemáticas, Universidad Católica del Norte, Antofagasta, Chile
[b] Centro de biotecnología "Profesor Alberto Ruiz", Universidad Católica del Norte, Antofagasta, Chile
[c] Departamento de Ingeniería Eléctrica, Universidad de Santiago de Chile, Santiago, Chile

ARTICLE INFO

Keywords:
Bioinformatics
Applied mathematics
Molecular computing technologies
Galois fields
Finite fields
DNA computing
Gel electrophoresis
Polymerase chain reaction (PCR)

ABSTRACT

A Galois field $GF(p^n)$ with $p \geq 2$ a prime number and $n \geq 1$ is a mathematical structure widely used in Cryptography and Error Correcting Codes Theory. In this paper, we propose a novel DNA-based model for arithmetic over $GF(p^n)$. Our model has three main advantages over other previously described models. First, it has a flexible implementation in the laboratory that allows the realization arithmetic calculations in parallel for $p \geq 2$, while the tile assembly and the sticker models are limited to $p = 2$. Second, the proposed model is less prone to error, because it is grounded on conventional Polymerase Chain Reaction (PCR) amplification and gel electrophoresis techniques. Hence, the problems associated to models such as tile-assembly and stickers, that arise when using more complex molecular techniques, such as hybridization and denaturation, are avoided. Third, it is simple to implement and requires 50 ng/μL per DNA double fragment used to develop the calculations, since the only feature of interest is the size of the DNA double strand fragments. The efficiency of our model has execution times of order $O(1)$ and $O(n)$, for the addition and multiplication over $GF(p^n)$, respectively. Furthermore, this paper provides one of the few experimental evidences of arithmetic calculations for molecular computing and validates the technical applicability of the proposed model to perform arithmetic operations over $GF(p^n)$.

1. Introduction

The fast-paced technological development keeps pushing computer science to new boundaries. The field of DNA computing was born to address hard computational problems. The strategy of most algorithms developed within this novel area of study, is brute force, relying on the huge capacity for parallel processing of DNA computing. The interest in designing a molecular computer is not limited to difficult search problems. If a computer should be able to carry out addition and multiplication, a wider range of problems could be addressed. However, most of the work done in the field of DNA computing is theoretical. Researchers laxly count on the supposed feasibility of the biomolecular techniques proposed in the works of Adleman (Adleman, 1994, 1996; Roweis at al., 1998), Lipton (Lipton, 1995, Winfree (Winfree et al., 1998); LaBean et al., 1999; Rothemund et al., 2004) and Rothemund (Rothemund, 2006). Most algorithms that have appeared in the literature are based on a reduced number of DNA computing models and introduced with no experimental work to back up their actual implementation.

We propose a new DNA based model specifically designed to do arithmetic over Galois fields, which was successfully implemented in the laboratory. Galois fields, $GF(p^n)$, are mathematical structures widely used in Cryptography and in Error Correcting Codes Theory. In Cryptography, the key exchange scheme of Diffie-Hellman is implemented on elliptic and hyperelliptic curves defined on Galois fields (Menezes et al., 1996; Koblitz, 1998; Cohen et al., 2006). On the other hand, in Error Correcting Codes Theory, algebraic geometric codes use algebraic curves, such as Hermitian curves, defined on Galois fields (Sklar, 2001; Guajardo, 2004; Carrasco and Johnston, 2008). Our model has two main properties. First, the molecular techniques employed, Polymerase Chain Reaction (PCR) and electrophoresis, are standard techniques, widely used, easily implemented and not expensive, with only a few designed components needed to carry out the experiments. Secondly, our model allows calculations over $GF(p^n)$, for p prime number, $p \geq 2$, and an integer $n > 0$. In contrast, all works on DNA molecular computation over finite fields found in the literature, are restricted to $GF(2)$ and $GF(2^n)$.

This paper is organized as follows. Section 2 introduces the DNA computing model. Section 3 presents mathematical basic concepts about Galois fields. Section 4 presents the proposed DNA-based model for arithmetic over Galois fields. Section 5 describes the physical molecular implementation for the proposed DNA-based model. Section 6

* Corresponding author.
E-mail address: ijiron@ucn.cl (I. Jirón).

https://doi.org/10.1016/j.heliyon.2019.e02901
Received 2 January 2019; Received in revised form 15 March 2019; Accepted 18 November 2019
2405-8440/© 2019 Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

summarizes the obtained experimental results for $GF(3^5)$ as case study. Section 7 presents a simulation of the proposed DNA-based model using Field Programmable Gate Array (FPGA) technology. Section 8 contains the discussion of the experimental results, the analysis of the advantages and the description of a possible DNA-based computer that implements arithmetic over $GF(p^n)$ using the proposed model. Section 9 presents conclusions and future works.

2. DNA computing models

The tendency in computer technology is to produce devices with greater memory and speed than the previous generation but much smaller. The idea of building a tiny computer is not new. In the late 1950s, Richard Feynman suggested the possibility of having sub-microscopic computers in his famous talk "There's Plenty of Room at the Bottom". However, only about two decades ago Leonard Adleman made a breakthrough when he used the tools of molecular biology to address an NP-complete problem (Adleman, 1994). He succeeded in solving a case of the Hamiltonian path, by manipulating DNA. This event marked the birth of the field known as DNA computing (Kari, 1997).

The speed of any computing device, bio-molecular or not, depends on how many parallel processes it has and how many steps, per each process, it can realize per unit of time. Electronic computers can calculate millions of instructions per second, a task a biological system cannot emulate. However, a DNA computer has a huge advantage in parallel processing and memory (Goldman et al., 2013) and this compensates for the much slower execution time for one instruction (Lipton, 1995; Guarnieri et al., 1996). The immense capacity for parallelization of DNA computing appeared to be the key to outperform electronic computers. The advent of the new discipline at first augured the end of silicon-based computers, however, scientists in the field soon acknowledged there were some obstacles in the way of realizing a competitive DNA-based computer (Gibbons et al., 1996; Regalado, 2000).

The models developed for DNA computing can be classified in two types: those which require human intervention during the process of calculation and those that can be programmed to function autonomously. Early research, following the works of Adleman (Adleman, 1994) and Lipton (Lipton, 1995), provided a variety of non-autonomous models, known as filtering models, for solving complex computational problems. Filtering models use large DNA combinatorial libraries as search spaces for algorithms of parallel filtering (Ignatova et al., 2008). Most of these works were theoretical (Adleman, 1996; Gibbons et al., 1996; Reif, 1995; Rozenberg and Spaink, 2003), however, a few specific problems were actually solved in the laboratory: a 3-SAT problem with 3 (Liu et al., 2000), 6 (Braich et al., 2000) and 20 (Braich et al., 2002) variables, and a variation of the SAT problem, known as the knight problem (Faulhammer et al., 1999).

To solve a wider range of problems a computer should be able to carry out addition and multiplication. However, carrying out binary operations poses other challenges. Guarnieri and colleagues (Guarnieri et al., 1996) presented a general algorithm to perform addition of two nonnegative binary numbers. In the same year, Roweis and colleagues introduced the sticker model, a complete and universal system (Roweis et al., 1998), which has been considered to do arithmetic over finite fields (Chang et al., 2005; Guo and Zhang, 2009; Li et al., 2013a). The sticker system is also a filtering model, which uses two types of single stranded DNA molecules, named memory strand and sticker strand. A memory strand and a number of sticker strands, hybridize to form a partial duplex (memory complex), which represents a bit string of zeros and ones. The main issues with this model are the limited length of a memory complex - it might fragment if it is longer than 15,000 bases – and time consuming operations, which are prone to error - stickers may bind to the wrong sites, or unbind when they are not supposed to. In 1998, Erik Winfree (Winfree, 1998) provided a remarkable new approach in the emerging field, when he proposed that DNA self-assembly could be used to do computation in an autonomous manner. Winfree explored algorithmic self-assembly, which is the result of the combination of Wang's tiling theory (Wang, 1961) and DNA nanotechnology, introduced by Seeman (Seeman, 1982). Winfree showed that DNA computation is Turing-universal and proposed that DNA self-assembly can be used to compute functions or assemble shapes (Winfree, 1996; Winfree et al., 1998; Rothemund et al., 2004). The introduced model by Winfree and colleagues, known as tile assembly model (TAM), has been considered to implement arithmetic over a finite field (Barua and Das, 2003; Li et al., 2013b, 2016; Li, and Xiao, 2014). TAM is based on the self-assembly of double-crossover DNA molecules (known as tiles) into a rectangular lattice, a pseudo-crystalline growth that occurs in the presence of an infinite supply of a finite number of tile types (Rothemund and Winfree, 2000; Jonoska et al., 2011). Tiles glue together or not depending on the binding domains on their sides. To carry out a computation one must start with an arrangement of tiles, called seed configuration, and a set of unattached tiles of different types. The calculation proceeds by annealing, ligation and melting, which occur in a controlled manner. A final configuration containing the result is obtained (Brun, 2007). The disadvantages of this model are the high error rate, the big number of components that a single calculation requires, and the fact that the seed configuration cannot be recycled (Brun, 2008; Brun and Medvidovic, 2007).

Despite of the progress achieved in the field of DNA computing, big drawbacks such as time consuming operations with a high error rate, the output following statistical laws, and the amount of DNA molecules growing exponentially with problem size, are still unresolved in all the mentioned models (Kari et al., 2012). Recently, Woods and colleagues have presented a reprogrammable model of self-assembly (Woods et al., 2019). On the other hand, Currin and colleagues presented a non-deterministic Turing universal model which offered to overcome the problems that previous models posed (Currin et al., 2017). However, the drawbacks associated to the complexities of the experiments are still an issue.

3. Basic concepts about Galois fields

In this section, basic concepts about Galois fields are presented (Guajardo, 2004; Hungerford, 2012; Koblitz, 1998; Menezes et al., 1996; Sklar, 2001).

A Galois field is a finite set $GF(p) = \{0, 1, 2, \cdots, p-1\}$ with addition, $+$, and multiplication, *, module p , defined in Tables 1(a) and (b), respectively. Here, $p > 1$ is a prime number.

Next, we briefly explain the method for constructing an extension field $GF(p^n)$, with $n \in \mathbb{Z}$ and $n > 1$, using $GF(p)$ as the underlying field.

First, an irreducible polynomial $Q(x)$ of degree $n \in \mathbb{Z}$, $n > 1$ over $GF(p)$ is selected,

Table 1. Definitions for addition and multiplication in $GF(p)$.

(a)					
+	0	1	2	⋯	$p-1$
0	0	1	2	⋯	$p-1$
1	1	2	3	⋯	0
2	2	3	4	⋯	1
⋮	⋮	⋮	⋮	⋯	⋮
$p-1$	$p-1$	0	1	⋯	$p-2$
(b)					
*	0	1	2	⋯	$p-1$
0	0	0	0	⋯	0
1	0	1	2	⋯	$p-1$
2	0	2	4	⋯	$p-2$
⋮	⋮	⋮	⋮	⋯	⋮
$p-1$	0	$p-1$	$p-2$	⋯	1

$$Q(x) = x^n + q_{n-1}x^{n-1} + \ldots + q_1 x + q_0 \quad (1)$$

where $q_i \in GF(p)$ for $i = 0, 1, \cdots, n-1$. The polynomial $Q(x)$ is called a primitive polynomial. Let α a root of $Q(x)$, that is $Q(\alpha) = 0$, then

$$\alpha^n = q'_{n-1}\alpha^{n-1} + \ldots + q'_1\alpha + q'_0 \quad (2)$$

where q'_i is the additive inverse of q_i according to Table 1a.

Next, α^{n+1} is constructed recursively as,

$$\alpha^{n+1} = \alpha * \alpha^n = \alpha^*\left[q'_{n-1}\alpha^{n-1} + \ldots + q'_1\alpha + q'_0\right]$$

$$\alpha^{n+1} = q'_{n-1}\alpha^n + \ldots + q'_1\alpha^2 + q'_0\alpha$$

and, the element α^n is replaced using Eq. (2),

$$\alpha^{n+1} = q'_{n-1}\left[q'_{n-1}\alpha^{n-1} + \ldots + q'_1\alpha + q'_0\right] + \ldots + q'_1\alpha^2 + q'_0\alpha$$

Then,

$$\alpha^{n+1} = a_{n-1}\alpha^{n-1} + \ldots + a_1\alpha + a_0$$

where $a_{n-1} = q'_{n-1}{}^*q'_{n-1}$, …, $a_1 = q'_{n-1}{}^*q'_1 + q'_0$, and $a_0 = q'_{n-1}{}^*\, q'_0$.

Thus, the nonzero elements of $GF(p^n)$ are generated as linear combinations of $\{1, \alpha, \alpha^2, \cdots, \alpha^{n-1}\}$ in the following manner,

$$\alpha^k = a_{n-1}\alpha^{n-1} + \ldots + a_1\alpha + a_0 \quad (3)$$

with $k \geq 0$, $a_i \in GF(p)$, $i = 0, 1, \cdots, n-1$. We should note that $\alpha^0 = 1$, and the null element $0 \in GF(p^n)$ does not have a representation as power of α. Hence, the field $GF(p^n)$ has p^n elements, which are stored in a look-up table according to the powers of each element.

Next, we explain how addition and multiplication of the elements of the field $GF(p^n)$ are carried out.

Let α^k, $\alpha^l \in GF(p^n)$, where

$$\alpha^k = a_{n-1}\alpha^{n-1} + \ldots + a_1\alpha + a_0, \quad \alpha^l = b_{n-1}\alpha^{n-1} + \ldots + b_1\alpha + b_0$$

Their addition is calculated as follows

$$\alpha^k + \alpha^l = [a_{n-1} + b_{n-1}]\alpha^{n-1} + \ldots + [a_1 + b_1]\alpha + [a_0 + b_0] \quad (4)$$

where $a_i + b_i$ is calculated in $GF(p)$ using the Table 1a, for $i = 0, 1, \cdots, n-1$. There is not carry or borrow, because $a_i + b_i$ are independent of each other.

On the other hand, the multiplication, $\alpha^k{}^*\alpha^l$, is calculated using Algorithm 1 (Guajardo, 2004).

In the seventh step of the algorithm, we must set $a_{-1} = 0$, when $j = 0$.

In the following sections, we will refer to steps 2 to 9 as the external cycle and to the two internal **for** cycles, that is, the first cycle from steps 3 to 5, and the second cycle from 6 to 8, as cycles IF-A and IF-B, respectively. These can be executed in parallel, since these are independent of each other.

Algorithm 1. Multiplication for α^k, $\alpha^l \in GF(p^n)$.

Input:

$\alpha^k = \sum_{i=1}^{n-1} a_i\alpha^i$, $\alpha^l = \sum_{i=1}^{n-1} b_i\alpha^i$, $Q(\alpha) = \alpha^n + \sum_{i=1}^{n-1} q_i\alpha^i$

where $a_i, b_i, q_i \in GF(p)$.

Output:

$C = \alpha^k{}^*\alpha^l = \sum_{i=1}^{n-1} c_i\alpha^i$

where $c_i \in GF(p)$.

1. $C \leftarrow 0$
2. **for** $i = 0$ to $n-1$ **do**
3. **for** $j = n-1$ to 0 **do**

(continued on next column)

(continued)

4. $c_j \leftarrow c_j + b_i{}^*a_j$
5. **end_for**
6. **for** $j = n-1$ to 0 **do**
7. $a_j \leftarrow a_{j-1} - q_j{}^*a_{n-1}$
8. **end_for**
9. **end_for**
10. **Return** C

Example 1. For $p = 3$ the field $GF(3) = \{0, 1, 2\}$, the addition and multiplication are defined in Tables 2(a) and (b), respectively.

The extension field $GF(3^5)$ is constructed using the primitive polynomial $Q(x) = x^5 + 2x + 1$. If α a root of $Q(x)$, then

$$Q(\alpha) = \alpha^5 + 2\alpha + 1 = 0 \Leftrightarrow \alpha^5 = \alpha + 2$$

Now, we construct the non-null elements of $GF(3^5)$ recursively as follows,

$$\alpha^6 = \alpha\alpha^5 = \alpha[\alpha + 2] = \alpha^2 + 2\alpha = 0\cdot\alpha^4 + 0\cdot\alpha^3 + 1\cdot\alpha^2 + 2\cdot\alpha + 0\cdot 1$$

$$\alpha^7 = \alpha\alpha^6 = \alpha\left[\alpha^2 + 2\alpha\right] = \alpha^3 + 2\alpha^2 = 0\cdot\alpha^4 + 1\cdot\alpha^3 + 2\cdot\alpha^2 + 0\cdot\alpha + 0\cdot 1$$

Equivalently, we can represent the elements of $GF(3^5)$ as arrays of elements in $GF(3)$.

$$\alpha^6 = (0\ 0\ 1\ 2\ 0), \quad \alpha^7 = (0\ 1\ 2\ 0\ 0)$$

Next we build a look-up table that contains all the elements of $GF(3^5)$. In particular, this field has $3^5 = 243$ elements and Table 3 shows some of its elements.

We use Algorithm 1 to calculate the multiplication $\alpha^6{}^*\alpha^{20} = \alpha^4 + 2\alpha^3 + 2\alpha + 2 = \alpha^{26}$ in $GF(3^5)$, where $\alpha^6 = \alpha^2 + 2\alpha$ and $\alpha^{20} = \alpha^4 + 2\alpha^3 + 2\alpha + 1$. Initially, the array C is initialized with

$$c_4 = 0, c_3 = 0, c_2 = 0, c_1 = 0, c_0 = 0$$

Then, the input values for α^6 and α^{20} are

$$a_4 = 0, \quad a_3 = 0, a_2 = 1, a_1 = 2, a_0 = 0$$

$$b_4 = 1, b_3 = 2, b_2 = 0, b_1 = 2, b_0 = 1$$

The coefficients of the primitive polynomial $Q(x) = x^5 + 2x + 1$ are

$$q_4 = 0, q_3 = 0, q_2 = 0, q_1 = 2, q_0 = 1$$

In Tables 4, 5, 6, 7, and 8, we detail the iterations of Algorithm 1 to calculate $\alpha^6{}^*\alpha^{20}$.

Finally,

Table 2. Definitions for addition and multiplication in $GF(3)$.

(a)

+	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

(b)

*	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

Table 3. Look-up table with some non-null elements of $GF(3^5)$.

α^i	α^4	α^3	α^2	α	1
$i=0$	0	0	0	0	1
$i=1$	0	0	0	1	0
$i=2$	0	0	1	0	0
$i=3$	0	1	0	0	0
$i=4$	1	0	0	0	0
$i=5$	0	0	0	1	2
$i=6$	0	0	1	2	0
$i=7$	0	1	2	0	0
$i=8$	1	2	0	0	0
$i=9$	2	0	0	1	2
⋮	⋮	⋮	⋮	⋮	⋮
$i=20$	1	2	0	2	1
⋮	⋮	⋮	⋮	⋮	⋮
$i=26$	1	2	0	2	2
⋮	⋮	⋮	⋮	⋮	⋮
$i=35$	1	0	1	2	2
⋮	⋮	⋮	⋮	⋮	⋮

Table 4. Iteration $i=0$ for the external cycle.

$a_4=0,\ a_3=0,\ a_2=1,\ a_1=2,\ a_0=0$		
$b_4=1,\ b_3=2,\ b_2=0,\ b_1=2,\ b_0=1$		
$q_4=0,\ q_3=0,\ q_2=0,\ q_1=2,\ q_0=1$		
$c_4=0,\ c_3=0,\ c_2=0,\ c_1=0,\ c_0=0$		
$i=0$		
Cycle IF-A	$j=4$	$c_4 \leftarrow c_4 + b_0{*}a_4 = 0+(1*0)=0$
	$j=3$	$c_3 \leftarrow c_3 + b_0{*}a_3 = 0+(1*0)=0$
	$j=2$	$c_2 \leftarrow c_2 + b_0{*}a_2 = 0+(1*1)=1$
	$j=1$	$c_1 \leftarrow c_1 + b_0{*}a_1 = 0+(1*2)=2$
	$j=0$	$c_0 \leftarrow c_0 + b_0{*}a_0 = 0+(1*0)=0$
Cycle IF-B	$j=4$	$a_4 \leftarrow a_3 - q_4{*}a_4 = 0-(0*0)=0$
	$j=3$	$a_3 \leftarrow a_2 - q_3{*}a_4 = 1-(0*0)=1$
	$j=2$	$a_2 \leftarrow a_1 - q_2{*}a_4 = 2-(0*0)=2$
	$j=1$	$a_1 \leftarrow a_0 - q_1{*}a_4 = 0-(2*0)=0$
	$j=0$	$a_0 \leftarrow a_{-1} - q_0{*}a_4 = 0-(1*0)=0$

Table 5. Iteration $i=1$ for the external cycle.

$a_4=0,\ a_3=1,\ a_2=2,\ a_1=0,\ a_0=0$		
$b_4=1,\ b_3=2,\ b_2=0,\ b_1=2,\ b_0=1$		
$q_4=0,\ q_3=0,\ q_2=0,\ q_1=2,\ q_0=1$		
$c_4=0,\ c_3=0,\ c_2=1,\ c_1=2,\ c_0=0$		
$i=1$		
Cycle IF-A	$j=4$	$c_4 \leftarrow c_4 + b_1{*}a_4 = 0+(2*0)=0$
	$j=3$	$c_3 \leftarrow c_3 + b_1{*}a_3 = 0+(2*1)=2$
	$j=2$	$c_2 \leftarrow c_2 + b_1{*}a_2 = 1+(2*2)=2$
	$j=1$	$c_1 \leftarrow c_1 + b_1{*}a_1 = 2+(2*0)=2$
	$j=0$	$c_0 \leftarrow c_0 + b_1{*}a_0 = 0+(2*0)=0$
Cycle IF-B	$j=4$	$a_4 \leftarrow a_3 - q_4{*}a_4 = 1-(0*0)=1$
	$j=3$	$a_3 \leftarrow a_2 - q_3{*}a_4 = 2-(0*0)=2$
	$j=2$	$a_2 \leftarrow a_1 - q_2{*}a_4 = 0-(0*0)=0$
	$j=1$	$a_1 \leftarrow a_0 - q_1{*}a_4 = 0-(2*0)=0$
	$j=0$	$a_0 \leftarrow a_{-1} - q_0{*}a_4 = 0-(1*0)=0$

Table 6. Iteration $i=2$ for the external cycle.

$a_4=1,\ a_3=2,\ a_2=0,\ a_1=0,\ a_0=0$		
$b_4=1,\ b_3=2,\ b_2=0,\ b_1=2,\ b_0=1$		
$q_4=0,\ q_3=0,\ q_2=0,\ q_1=2,\ q_0=1$		
$c_4=0,\ c_3=2,\ c_2=2,\ c_1=2,\ c_0=0$		
$i=2$		
Cycle IF-A	$j=4$	$c_4 \leftarrow c_4 + b_2{*}a_4 = 0+(0*1)=0$
	$j=3$	$c_3 \leftarrow c_3 + b_2{*}a_3 = 2+(0*2)=2$
	$j=2$	$c_2 \leftarrow c_2 + b_2{*}a_2 = 2+(0*0)=2$
	$j=1$	$c_1 \leftarrow c_1 + b_2{*}a_1 = 2+(0*0)=2$
	$j=0$	$c_0 \leftarrow c_0 + b_2{*}a_0 = 0+(0*0)=0$
Cycle IF-B	$j=4$	$a_4 \leftarrow a_3 - q_4{*}a_4 = 2-(0*1)=2$
	$j=3$	$a_3 \leftarrow a_2 - q_3{*}a_4 = 0-(0*1)=0$
	$j=2$	$a_2 \leftarrow a_1 - q_2{*}a_4 = 0-(0*1)=0$
	$j=1$	$a_1 \leftarrow a_0 - q_1{*}a_4 = 0-(2*1)=1$
	$j=0$	$a_0 \leftarrow a_{-1} - q_0{*}a_4 = 0-(1*1)=2$

Table 7. Iteration $i=3$ for the external cycle.

$a_4=2,\ a_3=0,\ a_2=0,\ a_1=1,\ a_0=2$		
$b_4=1,\ b_3=2,\ b_2=0,\ b_1=2,\ b_0=1$		
$q_4=0,\ q_3=0,\ q_2=0,\ q_1=2,\ q_0=1$		
$c_4=0,\ c_3=2,\ c_2=2,\ c_1=2,\ c_0=0$		
$i=3$		
Cycle IF-A	$j=4$	$c_4 \leftarrow c_4 + b_3{*}a_4 = 0+(2*2)=1$
	$j=3$	$c_3 \leftarrow c_3 + b_3{*}a_3 = 2+(2*0)=2$
	$j=2$	$c_2 \leftarrow c_2 + b_3{*}a_2 = 2+(2*0)=2$
	$j=1$	$c_1 \leftarrow c_1 + b_3{*}a_1 = 2+(2*1)=1$
	$j=0$	$c_0 \leftarrow c_0 + b_3{*}a_0 = 0+(2*2)=1$
Cycle IF-B	$j=4$	$a_4 \leftarrow a_3 - q_4{*}a_4 = 0-(0*2)=0$
	$j=3$	$a_3 \leftarrow a_2 - q_3{*}a_4 = 0-(0*2)=0$
	$j=2$	$a_2 \leftarrow a_1 - q_2{*}a_4 = 1-(0*2)=1$
	$j=1$	$a_1 \leftarrow a_0 - q_1{*}a_4 = 2-(2*2)=1$
	$j=0$	$a_0 \leftarrow a_{-1} - q_0{*}a_4 = 0-(1*2)=1$

Table 8. Iteration $i=4$ for the external cycle.

$a_4=0,\ a_3=0,\ a_2=1,\ a_1=1,\ a_0=1$		
$b_4=1,\ b_3=2,\ b_2=0,\ b_1=2,\ b_0=1$		
$q_4=0,\ q_3=0,\ q_2=0,\ q_1=2,\ q_0=1$		
$c_4=1,\ c_3=2,\ c_2=2,\ c_1=1,\ c_0=1$		
$i=4$		
Cycle IF-A	$j=4$	$c_4 \leftarrow c_4 + b_4{*}a_4 = 1+(1*0)=1$
	$j=3$	$c_3 \leftarrow c_3 + b_4{*}a_3 = 2+(1*0)=2$
	$j=2$	$c_2 \leftarrow c_2 + b_4{*}a_2 = 2+(1*1)=0$
	$j=1$	$c_1 \leftarrow c_1 + b_4{*}a_1 = 1+(1*1)=2$
	$j=0$	$c_0 \leftarrow c_0 + b_4{*}a_0 = 1+(1*1)=2$
Cycle IF-B	$j=4$	$a_4 \leftarrow a_3 - q_4{*}a_4 = 0-(0*0)=0$
	$j=3$	$a_3 \leftarrow a_2 - q_3{*}a_4 = 1-(0*0)=1$
	$j=2$	$a_2 \leftarrow a_1 - q_2{*}a_4 = 1-(0*0)=1$
	$j=1$	$a_1 \leftarrow a_0 - q_1{*}a_4 = 1-(2*0)=1$
	$j=0$	$a_0 \leftarrow a_{-1} - q_0{*}a_4 = 0-(1*0)=0$

$$\alpha^6 * \alpha^{20} = \alpha^{26} = \alpha^4 + 2\alpha^3 + 2\alpha + 2 \equiv 12022 \Leftrightarrow c_4 = 1,\ c_3 = 2,\ c_2 = 0,\ c_1 = 2,\ c_0 = 2$$

4. Proposed DNA-based model for arithmetic over $GF(p)$ and $GF(p^n)$

We have developed a simple DNA-based model to perform addition and multiplication over the fields $GF(p)$ and $GF(p^n)$, $n>1$. It is based on the differential migration of dsDNA fragments of different sizes in a gel electrophoresis, which is a standard technique for the separation of double-stranded DNA (dsDNA) fragments of different sizes that are previously obtained by PCR. Here the size of a dsDNA fragment corresponds to the number of base pairs $[bp]$ that are contained in the fragment.

Each element $r \in GF(p)$ is represented by a dsDNA fragment whose size is unique to the element r. Therefore, only p dsDNA fragments are necessary to represent all the elements of $GF(p)$. Table 9 shows this representation using dsDNA of different sizes, where the smallest size is S_0 and the largest size is S_{p-1}.

The gel electrophoresis is used to visualize the DNA molecular representation of a nonzero element $\alpha^k \in GF(p^n)$ representing the coefficients of the polynomial expression, which is given for Eq. (3). The dsDNA fragments for each coefficient $a_i \in GF(p)$, $i=0,1,\ldots,n-1$ are loaded into different slots of the agarose gel matrix. The slots and their respective columns are numbered as $n-1,\ n-2,\ \cdots,\ 2,\ 1,\ 0$ according to the order of powers $\alpha^{n-1},\ldots,\ \alpha^2,\ \alpha,\ 1$ from left to right. Then, an electric field is applied to make the molecules migrate through the gel and be separated by sizes. Figure 1 shows the dsDNA fragments representation of $\alpha^k = 2\alpha^{n-1} + (p-1)\alpha^{n-2} + \alpha^2 + (p-1)\alpha + (p-1)$. For this purpose, chains with size S_2 were loaded in the slot $n-1$, chains with size S_{p-1} were loaded in the slot $n-2$, and from slot $n-3$ to slot 3, chains with size S_0 were loaded. Finally, chains with size S_1 were loaded in the slot 2, and chains with size S_{p-1} were loaded in the slots 1and 0. Thus, our model defines a unique DNA-based representation for each element of $GF(p^n)$.

Table 9. DNA representation for elements $r \in GF(p)$.

$r \in GF(p)$	0	1	2	…	$p-1$
Size of DNA fragment $[bp]$	S_0	S_1	S_2	…	S_{p-1}

Example 2. For $GF(3)=\{0,1,2\}$ and the extension field $GF(3^5)$, the dsDNA fragments S_0, $S_1$and S_2 are required to represent the elements of $GF(3)$. Figure 2 shows the DNA-based representation of $\alpha^5 = \alpha + 2$ and $\alpha^{20} = \alpha^4 + 2\alpha^3 + 2\alpha + 1 \in GF(3^5)$. Such representations are obtained at the end of the gel electrophoresis process.

To calculate addition and multiplication in $GF(p^n)$, first it is necessary to establish a key configuration to interpret addition and multiplication in $GF(p)$, according to Tables 1(a) and (b), respectively. In the configuration key, the band pattern in any column (depicting dsDNA fragments on a gel matrix) represents the addition or multiplication of two elements of the field $GF(p)$. This is illustrated in the following example.

Example 3. The DNA-based implementation of addition in $GF(3)$, requires dsDNA fragments of 3 different sizes. For example, to carry the addition $1+2$, dsDNA fragments S_1 and S_2 are loaded into a slot in the agarose gel matrix. Then, the electrophoresis is executed, and the resulting band pattern is interpreted according to the key configuration for addition over $GF(3)$, shown in Figure 3. In this figure, a pattern as the one shown in column (v), will be interpreted as 0, the result of $1+2$ or $2+1$. In a similar way, for Figure 4, to calculate $1*0$, the dsDNA fragments S_1 and S_0 are loaded into the gel matrix and the electrophoresis is run. The resulting configuration, identical to the one shown in column (ii) of the key configuration for multiplication, will be read as 0, the result of $1*0$ or $0*1$.

Using these key configurations, we can carry out addition and multiplication of any two elements of $GF(p^n)$ by gel electrophoresis. Addition is calculated by adding the coefficients of corresponding powers of each operand, as explained in formula (4) of Section 3. Each one of these

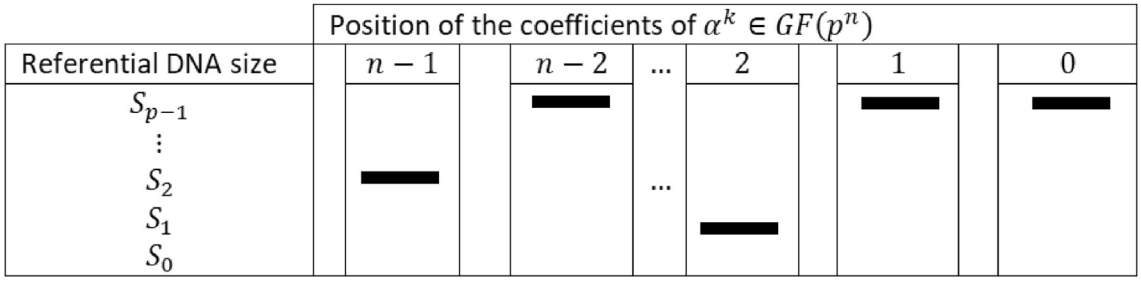

Figure 1. dsDNA fragments representation of $\alpha^k = 2\alpha^{n-1} + (p-1)\alpha^{n-2} + \alpha^2 + (p-1)\alpha + (p-1)$ performed by agarose gel electrophoresis.

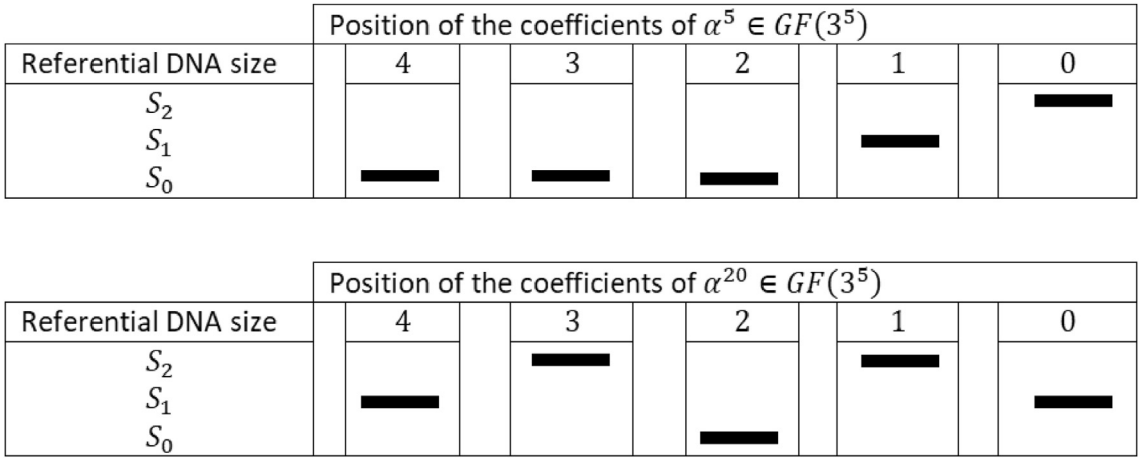

Figure 2. DNA-based representation for $\alpha^5,\ \alpha^{20} \in GF(3^5)$.

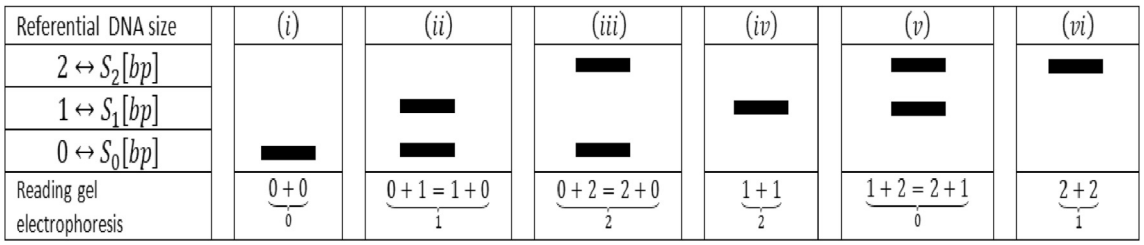

Figure 3. Key configuration for addition over $GF(3)$, used to interpret band patterns in gel electrophoresis.

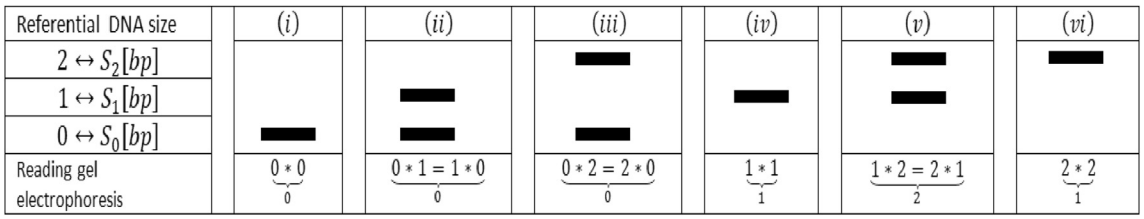

Figure 4. Key configuration for multiplication over $GF(3)$, used to interpret band patterns in gel electrophoresis.

additions is independent of the others, since there is not carry or borrow. This is best explained by the following example of addition over $GF(3^5)$.

Example 4. The addition of $\alpha^5 = \alpha + 2$ and $\alpha^{20} = \alpha^4 + 2\alpha^3 + 2\alpha + 1 \in GF(3^5)$ is calculated as

$$\alpha^5 + \alpha^{20} = (0+1)\alpha^4 + (0+2)\alpha^3 + (0+0)\alpha^2 + (1+2)\alpha + (2+1)1 = \alpha^4 + 2\alpha^3$$

Next, we use the Table 3, which is the look-up table of $GF(3^5)$, previously constructed, to find the representation of $\alpha^5 + \alpha^{20}$ as a power of a root α of primitive polynomial $Q(x) = x^5 + 2x + 1$. Then, the linear combination $\alpha^4 + 2\alpha^3$ or equivalently the array $(1\ 2\ 0\ 0\ 0)$ corresponds to α^8.

For the implementation of addition by gel electrophoresis, dsDNA fragments representing corresponding coefficients of α^5 and α^{20}, are loaded into five slots in the agarose matrix. The slots (columns) are numbered 4, 3, 2, 1, 0 from left to right. The dsDNA fragments S_0 and S_1 representing 0 and 1 are loaded in slot 4, S_0 and S_2 representing 0 and 2 are loaded into slot 3. This procedure is repeated for the rest of the coefficients. Next, the electrophoresis is run. Figure 5 shows the resulting band pattern from calculating $\alpha^5 + \alpha^{20}$. The configurations in columns 4 and 3, match the configurations in columns (*ii*) and (*iii*) of Figure 3, respectively. Thus the band patterns in these columns are read as 1 and 2. The columns 4, 3, 2, 1, 0 are independent of each other, because there is not carry or borrow, therefore they are interpreted separately. The complete band pattern is interpreted as the array $(1\ 2\ 0\ 0\ 0)$, which corresponds to element $\alpha^8 \in GF(3^5)$, according to the Table 3.

Below, we explain the DNA-based implementation for the multiplication of two elements of $GF(p^n)$.

Example 5. We use Algorithm 1 to calculate the multiplication $\alpha^{6*}\alpha^{20} = \alpha^4 + 2\alpha^3 + 2\alpha + 2 = \alpha^{26}$ in $GF(3^5)$, where $\alpha^6 = \alpha^2 + 2\alpha$ and $\alpha^{20} = \alpha^4 + 2\alpha^3 + 2\alpha + 1$. Initially, the agarose gel matrix is empty. Then, the array C is initialized according to the step 1 of Algorithm 1, with

$$c_4 = 0 \Leftrightarrow S_0, c_3 = 0 \Leftrightarrow S_0, c_2 = 0 \Leftrightarrow S_0, c_1 = 0 \Leftrightarrow S_0, c_0 = 0 \Leftrightarrow S_0$$

Next, dsDNA fragments representing corresponding coefficients of both elements are loaded into five slots of the agarose gel matrix. Then, the electrophoresis is run according to Algorithm 1, where the input values and their representation as dsDNA fragments for α^6 and α^{20} are

$$a_4 = 0 \Leftrightarrow S_0, a_3 = 0 \Leftrightarrow S_0, a_2 = 1 \Leftrightarrow S_1, a_1 = 2 \Leftrightarrow S_2, a_0 = 0 \Leftrightarrow S_0$$

and

$$b_4 = 1 \Leftrightarrow S_1, b_3 = 2 \Leftrightarrow S_2, b_2 = 0 \Leftrightarrow S_0, b_1 = 2 \Leftrightarrow S_2, b_0 = 1 \Leftrightarrow S_1,$$

respectively. The coefficients of the primitive polynomial $Q(x) = x^5 + 2x + 1$ and their representation as dsDNA fragments are

$$q_4 = 0 \Leftrightarrow S_0, q_3 = 0 \Leftrightarrow S_0, q_2 = 0 \Leftrightarrow S_0, q_1 = 2 \Leftrightarrow S_2, q_0 = 1 \Leftrightarrow S_1$$

Tables 4 and 8 show the iterations of the algorithm for cycles IF-A and IF-B for $i = 0$ and $i = 4$, respectively. Then, the array $C \Leftrightarrow (1\ 2\ 0\ 2\ 2)$ is searched in the rows of Table 3 for $GF(3^5)$, and it is determined that $\alpha^{26} = \alpha^{6*}\alpha^{20}$.

For Table 4, the theoretical scheme of gel electrophoresis for cycles IF-A (steps 3 to 5) and IF-B (steps 6 to 8) in Algorithm 1 is shown in Figure 6. We use three different sizes $\{S_0, S_1, S_2\}$ for the dsDNA fragments, in order to implement the addition and the multiplication in steps 4 and 7. Furthermore, as explained in Section 3, the cycles IF-A and IF-B

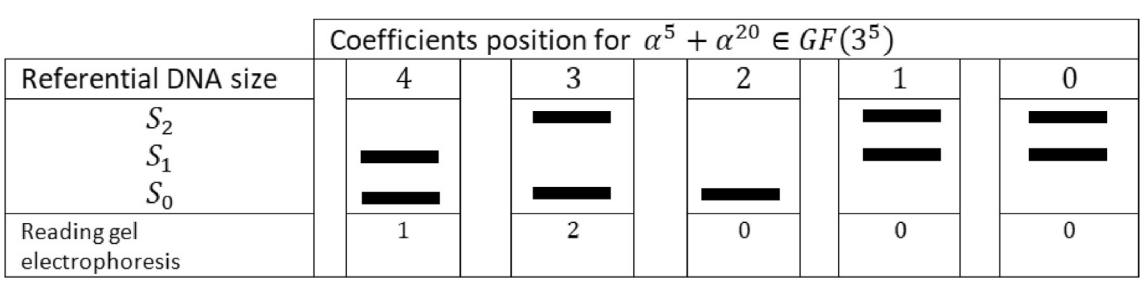

Figure 5. Gel electrophoresis implementation of $\alpha^5 + \alpha^{20} = \alpha^8$ in $GF(3^5)$.

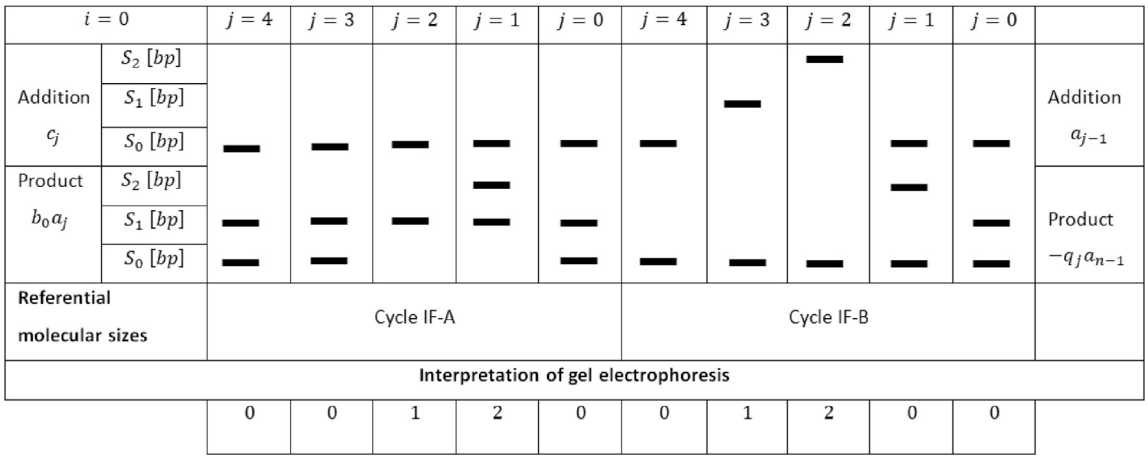

Figure 6. Interpretation of gel electrophoresis: cycles IF-A and IF-B for $i = 0$ and $j = 4, 3, 2, 1, 0$.

are independent of each other. This allows executing them in parallel, using gel electrophoresis. Figure 8 in Section 6 shows this condition of parallelism empirically in the lab.

For internal cycles IF-A and IF-B, the per column configuration in the lower half of the gel matrix ($b_0 a_j$ or $-q_j a_{n-1}$), is interpreted according to the key configuration for multiplication shown in Figure 4. The obtained result of multiplication ($b_0 a_j$ or $-q_j a_{n-1}$) is added to the operand (c_j or a_{j-1}), on the upper half of the gel matrix, which is in the same column. The addition is calculated according to Table 2a. The first iteration ($i = 0$) is shown in Figure 6. At the last iteration ($i = 4$), the final configuration is interpreted as the array (1 2 0 2 2) using the look-up table described in Table 3 for $GF(3^5)$, concluding that α^{26} is the result of $\alpha^6 * \alpha^{20}$.

Therefore, our DNA-based model has a flexible implementation in the laboratory, because we only need to change:

- The number p, which defines the amount of dsDNA fragments with different sizes that are previously obtained by PCR.
- The number n, which defines the amount of slots in the gel matrix to execute an electrophoresis.

It allows us to calculate additions and multiplications in different fields $GF(p)$ and $GF(p^n)$ with $n > 1$. For example, if we want to change from the field $GF(2^{163})$ to the field $GF(5^{80})$, we would only have to change from $p = 2$ to $p = 5$, and from $n = 163$ to $n = 80$. Obviously for each field we must build the respective tables and DNA-based representations for the addition and multiplication.

Finally, we analyze the efficiency of our model to calculate the addition and the multiplication in a field $GF(p^n)$. For this, we assume that all electrophoresis are executed in a constant time for the addition and the multiplication. Thus,

- The addition has an execution time of the order $O(1)$, since the addition is calculated in only one electrophoresis, as shown in Examples 3 and 4, Figures 3 and 5.
- The multiplication has an execution time of the order $O(n)$, since the internal cycles (Cycle IF-A and Cycle IF-B) of the Algorithm 1 are calculated in parallel, and in only one electrophoresis for each iteration of the external cycle (steps 2 to 9), as shown in Example 5 and Figure 6.

5. Physical implementation of the proposed DNA-based model

Before performing the agarose gel electrophoresis experiments, a dsDNA template is required from which to generate the different dsDNA fragments of known size by the PCR technique. For that purpose, the bacterial strain Sulfobacillus sp. CBAR-13, whose genomic DNA sequence was already known, was grown by microbial culture in the laboratory. Detailed information about the culture methodology is described below. Then, the genomic DNA was purified.

5.1. Cells growth and DNA template preparation

Bacterial strain CBAR-13 of Sulfobacillus sp. was grown in a shaking incubator at 59 °C in Single Strength medium [0.2 g/liter (NH4)2SO4, 0.4 g/liter MgSO4·7H2O, and 0.1 g/liter K2HPO4 (initial pH 1.7)] with 50 mM ferrous sulfate (membrane filtered) and 0.02% yeast extract. At the mid-exponential-growth phase, the bacterial cells were harvested, and the total genomic DNA was extracted with High Pure PCR Template Preparation Kit (Roche Product No. 11796828001) following the protocol prescribed by the manufacturer and then used for Polymerase chain reaction (PCR). The DNA fragments were obtained by PCR using a set of DNA primers designed specifically and the genomic DNA of CBAR-13 as a PCR template. The details are explained below.

5.2. Primers design

Primers were designed using Primer-BLAST software (Ye et al., 2012). The genomic sequence of *S.* sp. CBAR-13 (access numbers; NZ_LGRO01000001 and NZ_LGRO01000002) was used as template for primers design. Table 10 shows all primers designed, synthesized and used in this study.

5.3. PCR protocol for generation of dsDNA fragments

PCR amplification was carried out in a 50 μl reaction volume containing 50–100 ng of template DNA, primers (1 μM each Fw and Rv), dNTPs (10 μM each), MgCl2 (2 mM), 5X Green GoTaq® Flexi Buffer (1X final concentration) and 1.25 U GoTaq® DNA Polymerase (Promega catalog M7801).

Table 10. The three pair of PCR primers used in this study and the expected size for each PCR product. Fw and Rv are forward and reverse primers, respectively.

Primer	Sequence $5' \rightarrow 3'$	Product length (bp)
1-Fw	GACAGACCTGCTCGCTTCTT	639
1-Rv	TGGTAAACGCGGGCAACTTA	
4-Fw	TACTCCATCCGCCAGTCAGA	110
4-Rv	GTTGACGTGCTGTGACAACC	
5-Fw	GTTGTCACAGCACGTCAACC	77
5-Rv	AAGTACAAGAGCGCCAACGA	

The conditions for the PCR reactions were: 98 °C for 3 min, followed by 30 cycles of denaturation at 95 °C for 30 s, annealing at 60 °C for 45 s, extension at 72 °C for 30 s, and a final extension at 72 °C for 5 min.

5.4. Agarose gel electrophoresis

The products of the PCR reactions were separated as follows, 4 μl of each reaction were revealed by agarose gel electrophoresis at 90 V for 1.5 h on 1 or 3% agarose in Tris-acetate-EDTA buffer (40mM Tris, 20mM acetic acid, and 1mM EDTA) and 3 μl GelRed® 10000X (Biotium catalog 41002). The agarose gels were visualized by a transilluminator and then documented and confirmed. The same electrophoretic procedure was applied to perform the arithmetic calculations with the obtained DNA fragments.

6. Experimental results

The dsDNA fragments of specific size were generated for the experimental development of the proposed model. Figure 7 shows the size and quality of the generated fragments S_0, S_1, S_2, that represent the elements of the field $GF(3)$, verified by electrophoresis of the PCR products.

To test the validity of the proposed model, we performed the calculation described in Section 4 using the fragments generated previously.

Figure 8 shows the true implementation by gel electrophoresis of the multiplication described in Figure 6 of Section 4, which considers the iterations $i = 0$, where $j = 4, 3, 2, 1, 0$, for internal cycles IF-A and IF-B of Algorithm 1.

Once the results of c_j and a_j for $i = 0$ (cycle IF-A and IF-B) are interpreted and obtained, then c_j and a_j are replaced for calculation of $i = 1$, and so on. The calculation of the multiplication of α^6 and α^{20} in $GF(3^5)$ ends when all iterations $(i = 0, 1, 2, 3, 4)$ have been completed.

7. FPGA simulation of the proposed model

In this section, we present the simulation of our proposed model using Field Programmable Gate Array (FPGA) technology. The simulation consists in the design and testing of arithmetic circuits optimized for $GF(3^5)$, achieving shorter times than it would take sequential computers. For this we use the FPGA ZYNQ7000 of Xilinx, which is incorporated into a SoM TE0729-02 of Trenz electronic GmbH.

For the case study of the field $GF(3^5)$, the operations (base 3) were designed as a virtual layer on the components of the architecture of FPGA ZYNQ7000 (base 2). Thus, virtual minimum logical units of 3 states are considered to establish a homologation between the virtual layer and the physical layer. Figure 9 shows the logical mapping using a FPGA for the addition and multiplication of $GF(3)$, according to Tables 2a and b with $p = 3$. These operations are used to develop addition and multiplication of elements $A, B \in GF(3)$, as explained in Sections 3 and 4.

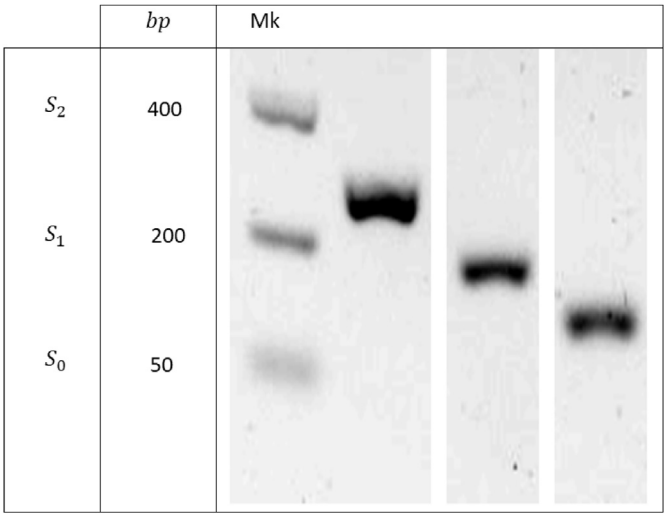

Figure 7. Practical implementation for Table 8 by DNA gel electrophoresis. Mk = Molecular weight marker.

On the other hand, Figure 10 shows the simulation using a FPGA, for $\alpha^6 * \alpha^{20}$, which was described in Example 1. The multiplication $\alpha^6 * \alpha^{20}$ is done in 5 iterations. These iterations appear in red color in the row corresponding to the result (R). It should be noted that in each operation performed with the coefficients of α^6 and α^{20}, the logical mapping described in Figure 9 is used. Although $\alpha^6 * \alpha^{20}$ is performed in one clock, there is an additional computational cost of converting non-binary coefficients in $GF(3)$ to their respective binary representation in order to operate with FPGA.

8. Discussion

This paper is the first that introduces a new DNA model in the area of molecular computing, designed to perform arithmetic over Galois fields, based on the differential migration of dsDNA fragments of different sizes. The proposed model presents several advantages over other models that have been widely studied and previously published, such as the Tile Assembling model (TAM) (Winfree et al., 1998) and the Sticker model (Roweis et al., 1998). All the arithmetic calculations covered by the TAM have been performed only in a theoretical way (e. g. Brun, 2007; Li et al., 2013a, 2013b; Li and Xiao, 2014; Li and Xiao, 2016; Li et al., 2016; Li, 2018; Li and Zhang, 2018). Li et al. (Li et al., 2013b) designed a tile assembly system that, in theory, could compute a square over $GF(2^n)$, based on the condition that all DNA operations are perfect. But it is widely known that this is not the case but quite the opposite. In the same way, Jonoska et al. (Jonoska et al., 2011) assumed, in their flexible-TAM study, that the assembly process happens in ideal conditions. In the more recent studies (Li et al., 2016; Li, 2018) the authors only reference the article (Rothemund, 2012) to justify the technical feasibility of their methodology for DNA computation of modular-multiplication and modular-square over $GF(2^n)$. However (Rothemund, 2012), only uses DNA self-assembly for fabrication of nanostructures (DNA origami) at laboratory scale. There is still not empirical evidence of the application of this molecular technique in the calculation of the mentioned problem. This may be due to the recognized complexity associated to the implementation of the DNA self-assembly technique in the laboratory (Rothemund, 2006; Jonoska et al., 2011). In (Woods et al., 2019) the authors present a reprogrammable DNA self-assembly system based on tiles, which can copy, sort, recognize palindromes, find multiples of 3, and other functions that are detailed in that article. However, this reprogrammable DNA self-assembly system is limited to the binary case, since the system uses iterated Boolean circuits. This hinders its application to develop calculations over a $GF(p^n)$ with $p > 2$, $n \geq 1$.

With respect to the molecular process associated to the TAM, Rothemund (Rothemund, 2006) and Jonoska (Jonoska et al., 2011) described and showed some typical experimental deviations that can occur during the practical work in the laboratory. First, the known difficulty in determining the stoichiometry for complex test tubes with different types of molecules, could result in annealing or thermodynamical problems and, in consequence, in hybridization mismatches and low performance of the reaction. Second, the low proportion of well-formed structures resulting from self-assembling (only 53%), evidenced by Rothemund (Rothemund, 2006), could predict an important percentage of error in the tile assembly process and, consequently, in the molecular calculations. Third, the presence of large dislocations at unbridged seams, where two halves of one assembled structure get completely separated, are also common. It is highly probable that all these technical issues hinder an accurate and successful computation by the TAM. In contrast, our model is based on conventional PCR reactions and agarose gel electrophoresis,

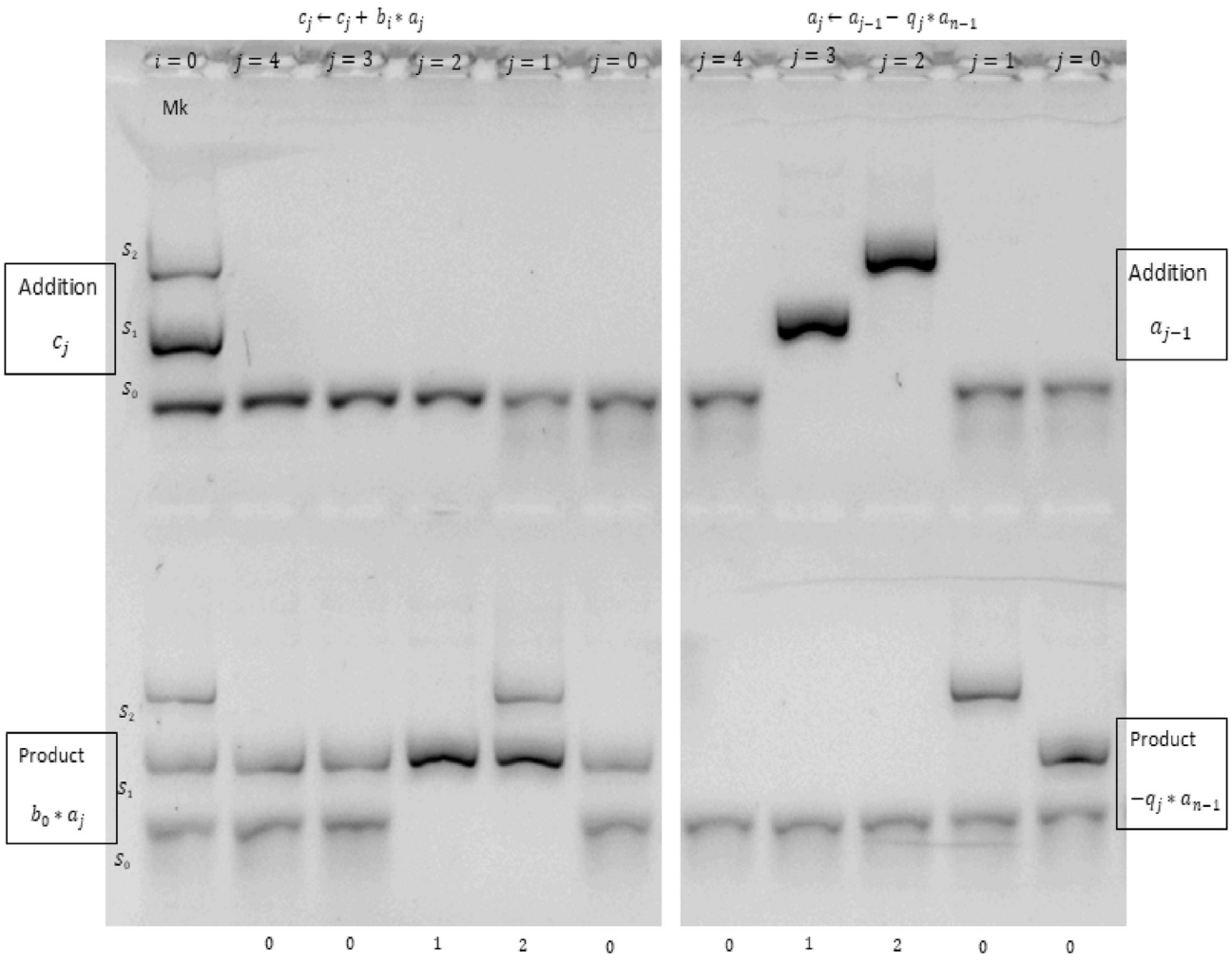

Figure 8. Practical implementation for Figure 7 by DNA gel electrophoresis.

both highly stable, reproducible and relatively low-cost molecular techniques. Another consideration for the technical application of the TAM on Galois fields calculations is that all technical work performed for one calculation (for example a multiplication of two elements in $GF(2^n)$) cannot be recycled for a different calculation, which would have to be completely performed from the start (Rothemund et al., 2004; Rothemund, 2012). Furthermore, it has been reported that time required for a calculation by TAM increases proportionally with the increase of n in a $GF(2^n)$, just because the complexity of the DNA assembly increase with n (Brun, 2008). If it is considered that the time required for the design of the sequences necessary for structure formation is one week in addition to one week needed for sequences synthesis and 2 h for mixing and annealing reactions (Rothemund, 2006), then TAM is a time and money expensive technique for algorithmic calculation. Conversely, most of the technical work of our model could be reused for different arithmetic calculations transforming it into a very attractive model in terms of costs and time. In an eventual new arithmetic calculation, only the electrophoresis must be repeated, while the different DNA fragments could be reused or, at most, re-amplified by PCR (90 min) with the previously designed primers. The application of the TAM model to arithmetical problems has been studied for more than ten years. However, there are still no records of its successful implementation in the laboratory to do arithmetic over $GF(p^n)$ with $p \geq 2,\ n \geq 1$. This supports our hypothesis that the TAM works well at a theoretical level but there is high uncertainty about the feasibility of its practical implementation and application in the short or medium term.

On the other hand, the only methodological complexity of our DNA-based model is that the number of different dsDNA fragments for the input depends on parameter p, and the number of slots in the gel matrix depends on parameter n for any $GF(p^n)$. For example, for the addition and multiplication over $GF(3^5)$ we only need dsDNA fragments of three different sizes and a gel matrix with capacity for five slots. For a much large field, such as $GF(2^{163})$, we only require dsDNA fragments of two different sizes and a gel matrix with capacity for 163 slots, to perform addition and multiplication over that field. This technical component of our model, also makes it simpler than the sticker model. The sticker model uses the hybridization of complementary DNA fragments to represent bit strings and do binary arithmetic, in theory (Zimmermann, 2002). However, the implementation of such calculations requires a careful adjustment of the hybridization conditions to ensure reproducibility and the correct assembly of all fragments. Consequently, any modification in the sequence of stickers or an increase in the number of stickers required, will need a resetting of the hybridization conditions. In (Li et al., 2013a), the authors present a stickers-based algorithm for parallel reduction in a field $GF(2^n)$, and they use the field $GF(2^{163})$ as a theoretical example. Then, to represent all the elements of $GF(2^{163})$ it is required to design and handle 163 different stickers to represent a bit 1 in the different positions of the memory strand. Moreover, it would not be possible to effectively manage the melting and annealing temperatures to bind and unbind selectively up to 163 different stickers. This fact makes the biological operations of merge, separate, set and clear difficult

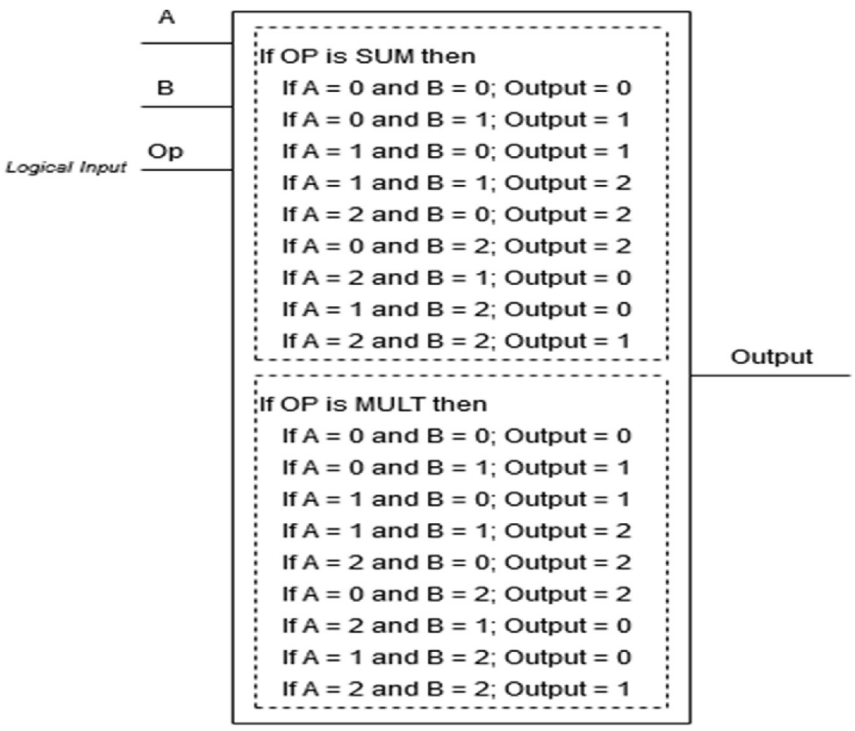

Figure 9. The logical mapping for the addition and multiplication of $A, B \in GF(3)$ using a FPGA.

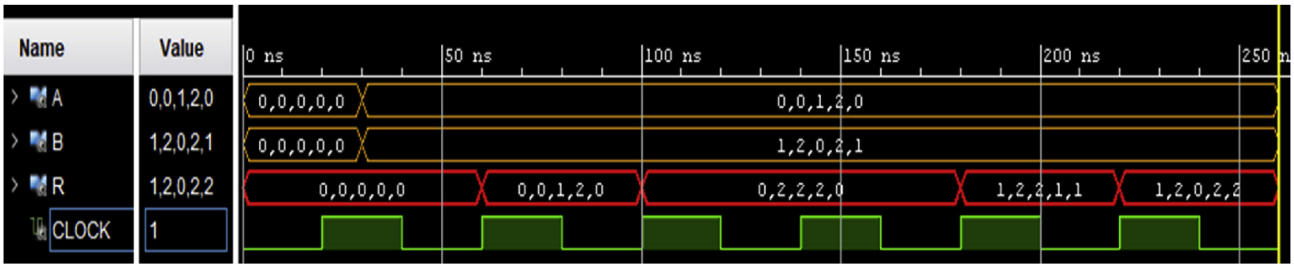

Figure 10. Simulation using a FPGA for $\alpha^6 * \alpha^{20}$.

to implement in the laboratory for addition and multiplication over $GF(2^{163})$.

Parallel computing is being widely studied and has been implemented using different approaches in recent years and it is expected that the threshold of ExaFLOP (1018 floating-point operations per second) will be reached by the year 2020. Thus, parallel computers could replace the current computers, which mostly have a sequential architecture (Li, 2018; Wright, 2019). However, silicon and molecular computers continue to use binary logic, and this force translating non-binary operations into binary atomic operations using Boolean algebra (Zhang et al., 2019; Eshra et al., 2019). In particular, to calculate addition and multiplication in a non-binary field $GF(p^n)$, $p > 2$, there is an additional computational cost associated with using atomic operations in Boolean algebra for the implementation of these operations. In this context, our model can also operate in parallel and it has a flexible implementation in the laboratory that allows avoiding the translation to Boolean operations to implement addition and multiplication over a non-binary field, which gives it a big advantage over current silicon and molecular systems. Even in the simulation with FPGA ZYNQ7000 for the case study of $GF(3^5)$, in which we programmed the logical mapping using the circuits to simulate the addition and the multiplication over any Galois field, it was necessary to translate the no-binary operations performed by our model into binary ones.

Therefore, our model has a flexible implementation in the laboratory which allows arithmetic operations over binary $GF(2^n)$ and non-binary $GF(p^n)$, $p > 2$ fields without conversion cost, it is easy to implement, economic, less prone to error, more tolerant to changes without altering the result.

On the other hand, we analyze the efficiency of our model to calculate the addition and the multiplication in a field $GF(p^n)$ with $p \geq 2$, $n \geq 1$. For this, we assume that each electrophoresis is executed in a constant time for the addition and the multiplication. Thus, we obtain that the addition has an execution time of the order $O(1)$, while the multiplication has an execution time of the order $O(n)$.

We use the multiplication as the worst case to compare the efficiency of our model with the efficiency of TAM-based models. Since the multiplication is more expensive than the addition in computational terms (memory, processor and time).

The authors of (Li, 2018; Li and Xiao, 2016; Li et al., 2016) state that the execution time for the multiplication over a field $GF(2^n)$ is $O(n)$ using the TAM system, which is equal to the execution time of our model. But in the TAM model, 7746 different tiles are needed to do the calculations, and as explained above this leads to great complexity of the implementation in the laboratory of the TAM model. Further, we say again TAM model can only be used in the binary case, that is, for $GF(2^n)$, $n \geq 1$.

Finally, with a plausible engineering intervention, our model can be fully automated. This is discussed with more detail below.

Figure 11 shows a system diagram of a possible DNA-based computer for our model that performs addition and multiplication over $GF(p^n)$ for $p \geq 2$ and $n \geq 1$ in an autonomous way.

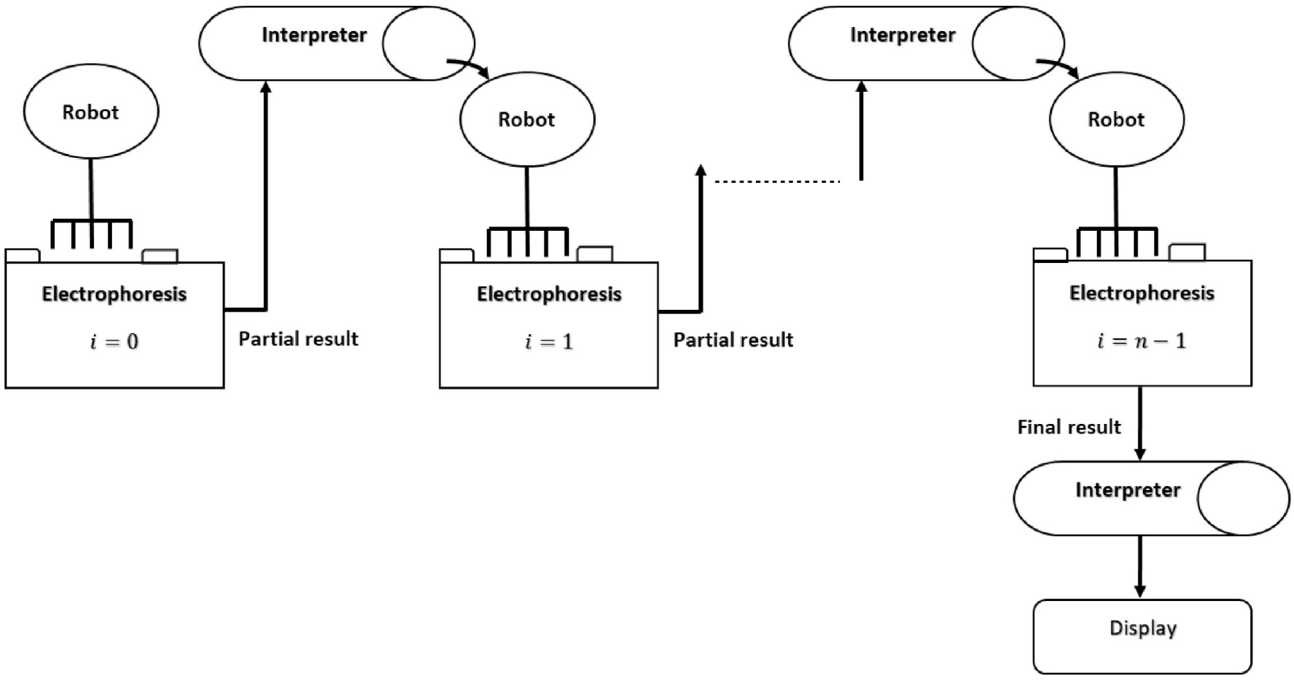

Figure 11. System diagram for a DNA-base molecular computer for proposed model.

In the system there are Robots, Interpreters, Electrophoresis boxes and a Display. A Robot makes the loading of the previously generated dsDNA fragments in the respective slots in the gel matrix for the electrophoresis. Thus, our model avoids any human error in pipetting and loading process. An Interpreter contains the look-up table of all the elements of the field $GF(p^n)$, and the key configurations for addition and multiplication. This device makes the interpretation of the images obtained by electrophoresis, and gives instructions to the next Robot to load the dsDNA fragments of the next iteration or Electrophoresis box. The Electrophoresis boxes represent the DNA electrophoresis processes for each addition or multiplication calculations, according to previously explained and showed in Sections 4, 5 and 6. At the end of the system, there is a Display, which is a device that receives information from the last Interpreter and shows the final result of the completed arithmetic calculation.

Many of the mentioned possible improvements for the proposed model are currently in use. The simplicity of the technique makes feasible its adaptation to devices currently available in the market, like automated and miniaturized systems. Microfabricated capillary array electrophoresis is a microfluidic device system that allows the separation of molecules, in this case DNA fragments of different size, and could be used for DNA sequencing (Paegel et al., 2002). The DNA sequencing technologies are relevant for our model, because they resume most of the advances towards increasing the analysis throughput and fragment resolution, and decreasing consumption of reactants and samples. For comparison, a slab gel requires 0.5–1.0 μL of DNA sample and 6–8 h of analysis time, the DNA separation in a microfluidic device could require 0.0001 – 0.0005 μL of DNA sample and 0.1 – 0.5 h of analysis time (Sinville and Soper, 2007). The handling of samples and the distribution of liquid solutions is a field where multiple solutions have been developed through robotics, e.g. QIAsymphony SP/AS instruments of Qiagen. Those devices can handle the mixture and the loading of samples into the analytic instrument, so the entire wet procedure is free of human intervention. This offers the advantages of a uniform loading of samples, time saving and avoidance of the error prone process of handling and loading larger number of samples by hand. Another area where improvements can be incorporated is in reading and interpreting of the results through image analysis by simple software, for example (Intarapanich et al., 2015; Abeykoon et al., 2015). This software can be of great help, as it can quickly and accurately read and translate the image results in a user-friendly format.

9. Conclusions and future work

This work is the first that introduces a novel DNA model to implement arithmetic operations over a field $GF(p^n)$, with $p \geq 2$ and $n \geq 1$, which is based on the differential migration of dsDNA fragments of different sizes. It has three major advantages over TAM and sticker models. First, because of its flexible implementation in the laboratory, it allows performing arithmetic operations over binary and non-binary Galois fields without the translation to Boolean operations, while finite field arithmetic, using the TAM model or the sticker model, is limited to $p = 2$. The second asset of our model is that it is less prone to error than other systems. It is based on conventional PCR amplification and electrophoresis, highly stable, reproducible and low-cost molecular techniques. Hence, the problems associated to other models that arise when using more complex molecular techniques, such as hybridization and denaturation, are avoided. The third advantage is that it is simple to implement and, when fully developed, it will use 50 ng/μL per DNA fragment used to develop the calculations. There is no need for designing complex DNA structures, since the only feature of interest is the size of the used dsDNA fragments. Also, to do arithmetic over $GF(p^n)$ only fragments of p different sizes and a gel matrix with capacity for n slots are necessary. This contrasts with TAM and sticker models, where the design of the DNA strands is of major importance and the concentration of reactants increases greatly with the size of the problem.

Furthermore, the flexible implementation in the laboratory of our model allows us to perform arithmetic calculations in parallel over $GF(p^n)$, $p \geq 2$, $n \geq 1$, and also without the cost of translating non-binary operations into binary atomic operations using Boolean algebra. Then, it is easy to implement, economic, less prone to error, more tolerant to changes without altering the result.

On the other hand, the efficiency of our model has execution times of order $O(1)$ and $O(n)$, for the addition and multiplication over a field $GF(p^n)$ with $p \geq 2$, $n \geq 1$, respectively. For this, we assume that each

electrophoresis is executed in a constant time for the addition and the multiplication.

This paper provides one of the few experimental evidences of arithmetic calculations for molecular computing and validates the technical applicability of the proposed model to perform arithmetic operations over $GF(p^n)$ with $p \geq 2,\ n \geq 1$.

Finally, our future work will be focused on making faster the interpretation of the DNA patterns produced in each electrophoresis, and with this achieve a cheaper and faster implementation of the addition and multiplication on a field $GF(p^n)$ with $p \geq 2,\ n \geq 1$ in the laboratory.

Declarations

Author contribution statement

Ivan Jiron, Susana Soto, Sabrina Marin, Mauricio Acosta, Ismael Soto: Conceived and designed the experiments; Performed the experiments; Analyzed and interpreted the data; Contributed reagents, materials, analysis tools or data; Wrote the paper.

Funding statement

This work was supported by VRIDT/UCN N° 171/17, VRIDT/UCN N° 084/2018 and Project FONDEF IT17M10012.

Competing interest statement

The authors declare no conflict of interest.

Additional information

Supplementary content related to this article has been published online at https://doi.org/10.1016/j.heliyon.2019.e02901.

References

Abeykoon, A.M.K.W.K., et al., 2015. An automated system for analyzing agarose and PolyacrylamideGel images. Ceylon J. Sci. (Bio. Sci.) 44 (1), 45–54.
Adleman, L.M., 1994. Molecular computation of solutions to combinatorial problems. Science 266, 1021–1024.
Adleman, L.M., 1996. On constructing a molecular computer. DNA Based Comp. 27. DIMACS - Series in Discrete Mathematics and Theoretical Computer Science.
Barua, R., Das, S., 2003. Finite field arithmetic using self-assembly of DNA tilings. IEEE the Congress on Evolutionary Computation. CEC '03.
Braich, R.S., et al., 2000. Solution of a satisfiability problem on a gel-based DNA computer. In: Condon, A., Rozenberg, G. (Eds.), Lecture Notes in Computer Science, 2054. Springer-Verlag, New York, p. 27—42.
Braich, R.S., et al., 2002. Solution of a 20-variable 3-SAT problem on a DNA computer. Sciencexpress.
Brun, Y., 2007. Arithmetic computation in the tile assembly model: addition and multiplication. Theor. Comput. Sci. 378, 17–31.
Brun, Y., Medvidovic, N., 2007. Discreetly distributing computation via self-assembly. In: Technical Report USC-CSSE-2007-714. Center for Software Engineering, University of Southern California.
Brun, Y., 2008. Non-deterministic polynomial time factoring in the tile assembly model". Theor. Comput. Sci. 395, 3–23.
Carrasco, R.A., Johnston, M., 2008. Non-Binary Error Control Coding for Wireless Communication and Data Storage. John Wiley & Sons, Ltd.
Chang, W.-L., et al., 2005. Fast parallel molecular algorithms for DNA-based computation: factoring integers. IEEE Trans. NanoBioscience 4 (No.2).
Cohen, H., et al., 2006. Handbook of Elliptic and Hyperelliptic Curve Cryptography. Taylor & Francis Group, LLC.
Currin, A., et al., 2017. Computing exponentially faster: implementing a non-deterministic universal Turing machine using DNA. J. R. Soc. Interface 14.
Eshra, A., et al., 2019. Renewable DNA hairpin-based logic circuits. IEEE Trans. Nanotechnol. 18, 252–259.
Faulhammer, D., et al., 1999. Molecular computation: RNA solutions to chess problems. Proc. Natl. Acad. Sci. 97, 1385–1389.
Gibbons, A., et al., 1996. Models of DNA computation", LNCS 1113, 21th. Int. Sym. Math. Foundations Comp. Sci. 18–36.
Goldman, N., et al., 2013. Towards practical, high-capacity, low-maintenance information storage in synthesized DNA. Nature 494, 77–80.
Guajardo, J., 2004. Arithmetic architectures for finite fields GF(p̂m) with cryptographic applications. Ruhr-Universität, Bochum.
Guarnieri, F., et al., 1996. Making DNA add. Science 273, 220–223.
Guo, P., Zhang, H., 2009. DNA implementation of arithmetic operations. In: Fifth International Conference on Natural Computation.
Hungerford, T.W., 2012. Abstract Algebra: an Introduction, 3 ed. CENGAGE Learning Custom Publishing.
Ignatova, Z., et al., 2008. DNA Computing Models. Springer.
Intarapanich, A., et al., 2015. Automatic DNA diagnosis for 1D gel electrophoresis images using bio-image processing technique. BMC Genom. 16 (Suppl 12), S15. http://www.biomedcentral.com/1471-2164/16/S12/S15.
Jonoska, N., et al., 2011. On stoichiometry for the assembly of flexible tile DNA Complexes. Nat. Comput. 10, 1121–1141.
Kari, L., 1997. DNA computing: the arrival of biological mathematics. Math. Intell. 19 (No. 2), 9–22.
Kari, L., et al., 2012. DNA computing – Foundations and implications. In: Back, T., Kok, J.N. (Eds.), Handbook of Natural Computing. Springer, pp. 1073–1127.
Koblitz, N., 1998. "Algebraic Aspect of Cryptography". Algorithms and Computation in Mathematics, 3. Springer, Berlin.
LaBean, T.H., et al., 1999. Experimental progress in computation by self-assembly of DNA tilings. In: 5th International Meeting on DNA Based Computers (DNA5). MIT, Cambridge, M.A.
Li, Y., et al., 2013a. A DNA sticker algorithm for parallel reduction over finite field GF(2n). Int. J. Grid Distributed Comput. 6 (No. 5), 17–28.
Li, Y., et al., 2013b. Square over finite field GF(2n) using self-assembly of DNA tiles. Int. J. Hosp. Inf. Technol. 6 (No. 4).
Li, Y., Xiao, L., 2014. Arithmetic computation using self-assembly of DNA tiles: integer power over finite field GF(2n). Int. Conf. Bioinfo. Biomed. 471–475.
Li, Y., et al., 2016. A molecular computation model to compute inversion over finite field GF(2n). In: 22nd International Conference on Parallel and Distributed Systems, pp. 1151–1156.
Li, Y., Xiao, L., 2016. Molecular computation based on tile assembly model: modular-multiplication and modular-square over finite field GF(2n). In: IEEE 14th International Conference on Smart City, pp. 1007–1014.
Li, Y., 2018. A revised DNA computing model of inversion and division over finite field GF(2^n). In: IEEE Intl. Conf. On Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications, pp. 477–484.
Li, Y., Zhang, Z., 2018. Parallel computing: review and perspective. In: IEEE 5th International Conference on Information Science and Control Engineering 2018, pp. 365–369.
Lipton, R.J., 1995. DNA solution of hard computational problems. Science 268, 542–545.
Liu, Q., et al., 2000. DNA computing on surfaces. Nature 403, 175.
Menezes, A., van Oorschot, P., Vanstone, S., 1996. Handbook of Applied Cryptography. CRC Press.
Paegel, B.M., Emrich, C.A., et al., 2002. High throughput DNA sequencing with a microfabricated 96-lane capillary array electrophoresis bioprocessor. Proc. Natl. Acad. Sci. U.S.A. 99 (2), 574–579.
Regalado, A., 2000. DNA computing. MIT Technol. Rev. 103 (3), 80.
Reif, J.H., 1995. Parallel Molecular Computation. SPAA'95 Santa Barbara CA USA@ 1995 ACM 0-89791-717-0/95/07.
Rothemund, P.W.K., Winfree, E., 2000. The program size complexity of self assembled squares. In: STOC '00 Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, pp. 459–468.
Rothemund, P., Papadakis, N., Winfree, E., 2004. Algorithmic self-assembly of DNA Sierpinski triangles. PLoS Biol. 2 (12), 2041–2053 e424.
Rothemund, P.W.K., 2006. Folding DNA to create nanoscale shapes and patterns. Nature 440, 297–302.
Rothemund, P.W.K., 2012. Beyond Watson and Crick: programming DNA self-assembly for nanofabrication. In: IEEE 7th International Conference on Nano/Micro Engineered and Molecular Systems (NEMS).
Roweis, S., et al., 1998. A sticker-based model for DNA computation. J. Comput. Biol., Winter 5 (4), 615–629.
Rozenberg, G., Spaink, H., 2003. DNA computing by blocking. Theor. Comput. Sci. 292, 653–665.
Seeman, N.C., 1982. Nucleic acid junctions and lattices. J. Theor. Biol. 99, 237–247.
Sinville, R., Soper, S.A., 2007. High resolution DNA separations using microchip electrophoresis. J. Sep. Sci. 30 (11), 1714–1728.
Sklar, B., 2001. Digital Communications. Fundamentals and Applications, second ed. Prentice-Hall, Inc.
Wang, H., 1961. "Proving theorems by pattern recognition, II. Bell Syst. Tech. J. 40, 1–41.
Winfree, E., June 1998. Algorithmic self-assembly of DNA. In: Ph.D. Thesis. Caltech, Pasadena, CA.
Winfree, E., 1996. On the computational power of DNA annealing and ligation. DNA Based Comp. 199–221.
Winfree, E., Liu, F.R., Wenzler, L.A., Seeman, N.C., 1998. Design and self-assembly of two-dimensional DNA crystals. Nature 394, 539–544.
Wright, A.S., 2019. Performance modeling, benchmarking and simulation of high performance computing systems. Future Gener. Comput. Syst. 92, 900–902.
Woods, D., et al., 2019. Diverse and robust molecular algorithms using reprogrammable DNA self-assembly. Nature 567, 366–372.
Ye, J., et al., 2012. Primer-BLAST: A tool to design target-specific primers for polymerase chain reaction". BMC Bioinf. 13, 134.
Zhang, C., et al., 2019. DNA computing for combinational logic. Sci. China Inf. Sci. 62, 61301.
Zimmermann, K.H., 2002. Efficient DNA sticker algorithms for NP-complete graph problems. Comput. Phys. Commun. 114, 297–309.