Journal of Innovations in Computer Science and Engineering
Shahid Beheshti Unversity

June 2025, Volume 3, Issue 1

An Efficient Encoding Mechanism for Digital Microfluidic-based DNA Data Storage Systems

M. Pourasadollah, **CODE ORCID**:
Faculty of Computer Science and Engineering, Shahid Beheshti University
A.Jahanian, ✉, **CODE ORCID**:
Faculty of Computer Science and Engineering, Shahid Beheshti University
M. Taajobian, **CODE ORCID**:
Department of Computer Engineering, Mahdishahr Branch, Islamic Azad University

ABSTRACT

Digital data production speeds up dramatically every day, and the need to store extensive data rises consequently. Many storage media are used today, but only a few are capable of the required features in terms of high density, data duration, storage lifetime, and reliability. DNA-based digital data storage systems are an emerging and high-capacity medium, capable of storing vast amounts of data in a very small volume with an exceptionally long lifespan. However, building dense and straightforward DNA storage comes with some challenges. Using manual methods to transfer and mix liquids containing DNA molecules makes this storage more error-prone. Moreover, the encoding method of digital data into DNA molecules is very crucial. This paper proposes a novel method for automatic and controllable DNA strand manipulation using a customized Digital Microfluidic Biochip (DMFB) corresponding to the CAD algorithm. Then, we offer a new approach to encoding the digital data into DNA molecules that helps error detection and correction in DNA storage while it brings efficient encoding density. Experimental results show that we can write 1.6 bits on each nucleotide with a 200-base strand which is a good density compared with other methods. In contrast, the encoding method is much simpler and easier to implement.

Keywords: DNA storage, Microfluidic, Data encoding.

1. Introduction

The intense growth rate of digital data production and lack of enough space in current storage media has forced the industry to look for newer storage technologies. A quick look at the current data-store procedures shows that the continuation of using current storage media will be faced with severe problems soon [1]. DNA-based digital data storage is getting more and more attention due to its wonderful features among the new technologies for data storage [2] and [3]. High data density and very long durability and availability are some of the attractive features of these molecules. Storing more than 109 GB of data only on one cubic millimeter of space is a very critical property for the mentioned demand [4], [5] and [6]. Moreover, recovering data from DNA molecules for thousands of years also reminds us of a durable storage media with no competitors or even close numbers. However, using DNA as a medium for storage faces some challenges [7]. The main challenges of current DNA storage are as follows:

- Low read/write bandwidth due to the chemical reactions.
- Low controllability of the required liquid operating media.
- Considerable fault rate in DNA molecules.
- Considerable cost of DNA synthesis for storing/retrieving data.

Due to the liquid nature of DNA molecules, they need to be kept in a solution. Therefore, working with these molecules requires using tools like pipettes to transfer a specific volume of solution, which means a user must do DNA related operations manually. This process is very time-consuming, error-prone, and costly, with a considerable waste of expensive experimental materials.

Digital Microfluidic Biochip (DMFB) is a promising technology to improve controllability and reduce the consuming materials in liquid assays. We proposed a new DMFB architecture specialized for DNA storage systems and the corresponding CAD algorithm that helps us make an automatic and parallel digital DNA storage system. We used the DNA storage structure of [4] as the base model and customized it on a DMFB to speed up read/write operations in DNA storage with a high degree of automation and parallelism.

The significant level of potential faults in DNA synthesis and reactions is another challenge for DNA storage systems [8]. Moreover, we introduced a lightweight encoding method to improve the fault tolerance of these systems. We demonstrate that digital data encoding can be performed efficiently at the nucleotide level, independently of encodings at higher levels of abstraction.

Submit Date: 2025-06-15
Revise Date: 2026-05-18
Accept Date: 2026-05-26
✉ Corresponding author

This paper is organized as follows. Section 2 describes the basic concepts of digital DNA storage systems, and Section 3 reviews the previous work on DNA storage systems. The proposed DMFB architecture/algorithm is illustrated in detail in Section 4. In Section 5, simulation results are presented and discussed, and finally, Section 6 concludes the paper.

2. Basic Concepts

2.1. Digital Microfluidic Biochips

Microfluidic Biochips are large bio laboratories in the size of an electronic chip that removes the human agents from the bio-experiments. This technology improves the controllability of the assays significantly and also hugely reduces the volume of the costly chemical materials and test samples. The first generation of these chips is based on the continuous flow of liquid inside some microchannels that are etched inside the chip. Thus, a tiny sample gets sucked into the chip and directed to some enzymes through microchannels to reach the final results. This second-generation, called Digital Microfluidic Biochip (DMFB), is built based on the movements of discrete droplets of liquid by electro-wetting that can be of tiny sizes compared with the continuous flow biochips [8].

The attractive features of DMFBs make them a perfect choice for many industrial, commercial, and military applications. Nowadays, they are used as remote healthcare for patients, robotic and artificial intelligent systems. In the military industries, they can play an essential role in detecting biological threats on the battlefield. One of the hottest applications of these chips is in DNA computers and DNA storage systems. DMFBs generally work by moving discrete droplets on a surface using electricity. Droplets are moved on a two-dimensional array of electrodes under an electrical field. The droplets are moved on the chip's surface by applying a specific voltage to each electrode based on a physical process called electro-wetting. A DMFB has four primary operations; MOVE, MIX, MERGE, and SPLIT. The MERGE combines two different droplets to generate a new droplet, and the MIX operation mixes the materials inside a droplet, while SPLIT does the opposite and separates a droplet into two new droplets. Finally, MOVE transports the droplets on the DMFB surface from the source to the target electrode. All these operations get done by applying a voltage to different electrodes and the electrodes responsible for doing a specific operation like MIX, MERGE, and SPLIT are called modules.

The sequence of applying a voltage to specific electrodes that produce primary operations and causes droplets on a chip to pass through some specific path and reach specific spots on the chip is called DMFB architecture. Thus a chip architecture depends on the actual assay mapped into the chip [9]. Figure 1 shows the design flow of an assay on a DMFB. As shown in this figure, the assay gets converted to a sequence graph that determines when and what operation must be done on droplets until the final result. After that, scheduling the assay steps according to the sequencing graph is considered, and resources on the chip for doing primary operations get allocated. Then it is time to determine the placement of each module and, finally, find the electrodes' activation sequence. This way, the assay completely gets mapped into the DMFB.

The output of this flow is a string of electrode activation sequences that must be applied to the DMFB to do the bioassay automatically.

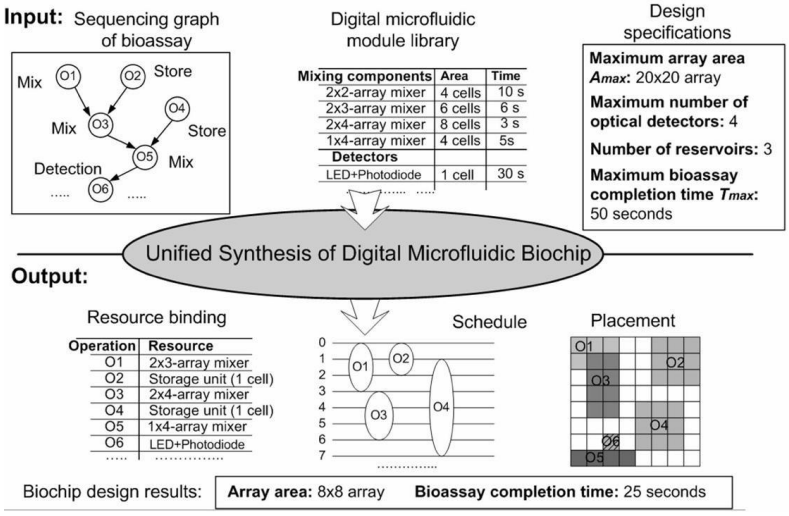

Figure 1: Stages of the DMFB design flow [9]

2.2. DNA-based Digital Data Storage

For Billions of years, DNA molecules have been working in nature to encode the protein generation of living cells and transfer the genetic data between generations of creatures by storing the genetic information of the life on these molecules, similar to digital data storage. DNA molecules include two helix ribbons that are the backbone of these molecules. On each ribbon, many capsules are sitting next to each other while their other head is stuck to another capsule (making it a pair) on the other ribbon. The capsules are called nucleotides, and there are only four types of these nucleotides on a DNA molecule; Adenine (A), Cytosine (C), Guanine (G), and Thymine (T).

Any DNA molecule has many nucleotides, which may differ in count and sequence. The single ribbon in a DNA molecule is called a strand. Generally, the sequence of nucleotides on a strand is shown by the first letter of their name, like "ACCTG" or "GTCATGGCAATG". The nucleotides sequence is just like a sequence of binary data like "00010111010101", but the only difference is that the binary string consists of two values (0 and 1) while the DNA strand consists of four A, C, G, and T [4]. On the other hand, DNA molecules have an exciting feature that nucleotides of one type on one strand only stick to a specific type on the pairing strand; that is, nucleotides of type A only stick to T (and vice versa), and C only sticks to G (and

vice versa). This feature is called reverse complementariness, and with that, if we have only a single strand, we can make its complementary strand with complementary nucleotides [4].
We need to encode the binary data and convert it to a sequence of nucleotides to store binary data on a DNA strand. Then using a device called a synthesizer, this sequence is converted to actual DNA molecules. Then data are retrieved from DNA molecules using another device called a sequencer.
The details of synthesizing and sequencing operations are out of the scope of this paper, but we can see how DNA storages generally work. Usually, a key-value method is used in digital DNA storage to random-access store and retrieve data. This method converts a file with digital contents to a key-value pair. The key part is unique for each file, and the value is the data within that file. The synthesizer converts this key and value to primer and payload, where the primer is a unique sequence of nucleotides for each file and the payload is a sequence of nucleotides representing the value of the data. In current technologies, synthesizing long strands (e.g., more than 200 nucleotides) faces a significant error rate. Therefore, payload data should be split into many strands. The synthesizer puts the primer part of a file to the beginning and ending head of each strand belonging to that specific file together with an index so it would be easy to retrieve all strands belonging to one specific file in the correct order.
To retrieve the data from DNA strands, the sequencer reads strands with similar primers, sorts them according to each strand's index, and then converts it back to binary data after recovering the primary payload DNA strand. The processof reading and writing data from/to DNA strands is shown in Figure 2.

According to Figure 2-A, a sample file named "foo.txt" for a write process in DNA storage gets converted to a key-value combination. Then the key part gets encoded into a primer, and the value part gets encoded into fragments to keep the number of nucleotides of one strand lower than the limit on strand length. Finally, the synthesizer converts the encoded nucleotides to actual DNA strands with the same primer but different payloads.

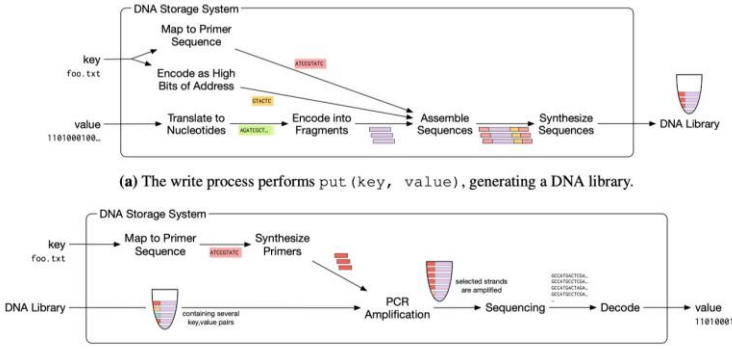

Figure 2: Internal operations of a digital DNA storage system [4]

To simplify recovering original binary data from these strands, an address part is also included inside each strand that helps sort out different fragments to recover original data. The output DNA strands will be stored in a pool that may also contain strands from other files. Figure 2-b shows a read process in which the synthesizer produces the required primer based on the given key (from the "foo.txt" file). Then using PCR, the contents of the pool that contains similar primers to the "foo.txt" file will get amplified and then sent to the sequencer to retrieve the original "foo.txt" file data [4].

2.3. DNA Encoding

There are many encoding methods to convert binary data to DNA nucleotides. Here, we address some parameters that are used to compare different encoding methods to see better improvements made using new methods [10]. The main goals of encoding in digital DNA storage systems are as follows:

- **Better Fault Tolerance:** generating more copies of the same DNA strand is called physical redundancy. When we have more than one copy of a strand, there are more backups in case of an incident or fault that can help recover original data. Having a proper physical redundancy helps recover data in DNA storage when it gets damaged for any reason. It also makes DNA storage more fault-tolerant. Moreover, some nucleotides can be added to a strand responsible for adding fault tolerance in logical redundancy despite physical redundancy. Depending on the encoding method, these extra nucleotides will help recover original data in case of any fault. It is good to keep the number of added nucleotides as low as possible so the ratio of data bits to present nucleotides in a strand (also called logical density) would be high. Also, while using logical redundancy increases the total number of strands the synthesizer needs to generate, the number of final strands is much lower than physical redundancy.
- **Strand length:** Due to some technical limitations that synthesizers have, it is critical to keep the number of nucleotides on each strand below 200 counts. Thus the way we encode data is very important to make a proper logical redundancy while not over-passing strand length limitations.

Encoding methods have been widely used to increase fault tolerance and reduce DNA synthesis costs.

3. Previous Work

DNA-based storage was introduced to store digital information more than ten years ago, and many contributions have been proposed to push this technology forward. In 2016, Seelig et al. [4] proposed a novel and practical method that could store different digital data files in DNA. He showed that it is possible to convert any binary file into a DNA-based

key-value pair, so each file has its unique key and related DNA strands. Therefore, it could be possible to differentiate between DNA strands from multiple files. They stored 42 KB of binary data in DNA successfully. In 2017, Seelig et al. [11] raised their storage capacity to 200 MB. They used their key-value system to store DNA strands in different pools called DNA libraries. In this way, they organized their storage to locate different files in a unified DNA library that simplified access to different strands and made random access possible.

The liquid nature of DNA strands needs the use of devices that can work with liquid and get automated. Micro-array devices that enable automatic liquid transport operations have been available for nearly a decade, and digital microfluidic biochips (DMFB) are the better choice. They are an excellent platform to control and automate the movement of liquid droplets on a glass or chip surface.

In 2001 Chakrabarty et al. [12] used the electro-wetting feature of liquids to move the droplets between different electrodes on two-dimensional surface electrodes. This method provided the ability to perform biological tests on sample droplets by scheduling voltage applications on electrodes. In 2006, authors of [4] described synthesizing and reconfiguring electrodes to shape different paths for droplet movements. They have shown how to schedule basic operations on a chip surface to implement a complete chemical or biological test. In 2016 a new architecture was proposed [13] to overcome the problems of the previous DMFB generations. They use a micro-electrode-dot-array instead of the old two-dimensional array to move the droplets in almost any direction and with different volumes. They have added a sensor underneath each micro-electrode to help detect the droplet's position. Although this method made some improvements, it increased the costs so much. Douglas et al. [14] in 2018 mentioned for the first time that DMFBs could be a good option for manipulating the droplets containing DNA molecules, but they did not propose a practical method.

In 2019, Newman et al. [15] used a DMFB to move droplets to a dried spot on its surface and rotate the droplets. Therefore, they could solve DNA molecules from that spot into the droplet. This approach is attractive, but DNA molecules are generally available in a solution in an actual application, and the proposed method in [15] did not help implement the DNA-based storage.

Along with the computation media of DNA operations, DNA encoding is a critical factor in improving DNA-based storage. In 2012, Church et al. [16] proposed an innovative method for encoding binary data into DNA strands. They used "A" or "C" nucleotide for 0 and "G" or "T" for 1. This way, they always had two nucleotides for each binary digit to make an efficient way for multiple strands of arbitrary binary data to improve the fault-tolerance.

In 2013, Goldman et al. [17] showed that it is possible to convert initial binary data into a string of digits in base three and then convert them into a string of DNA nucleotides. Afterward, DNA strands could get generated with an overlapping pattern, as shown in Figure 3.

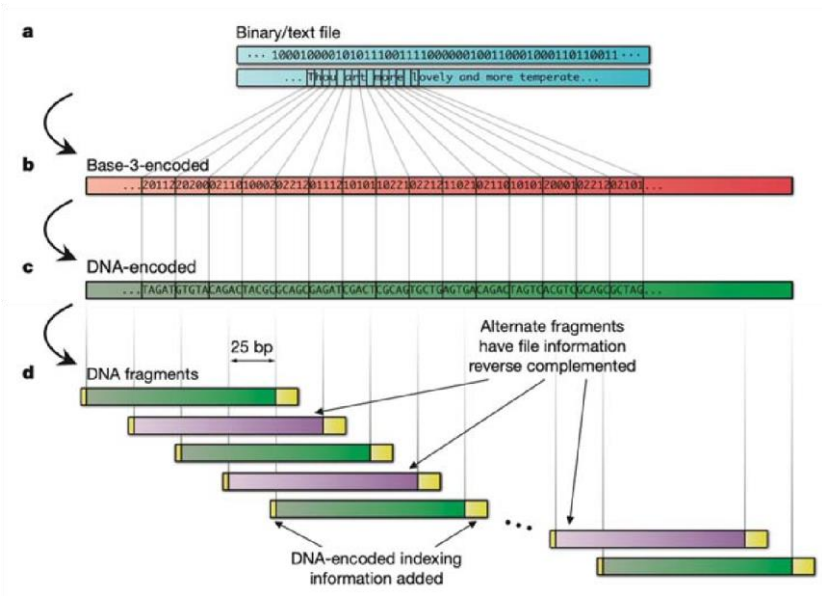

Figure 3: Encoding the digital information into DNA nucleotides using overlapping strands [17]

In this method, the original DNA strand (that is converted from the binary string) is divided into fragments with equal length so that any fragment has 75% nucleotides in common with its previous and next fragment. Moreover, indexing information is added to each fragment so it could be possible to sort them. It is worth noting that this method increases the number of total strands (system cost) in case of finding a faulty or damaged strand.

Another method proposed in 2015 by Grass et al. [18] could encode and protect DNA strands by encapsulating them. Their approach for encoding data first converts every two bytes of primary binary data into three elements of a Galois field of size 47 (Figure 4-A). Then they sorted this data into a block with M rows and N columns as shown in Figure 4-B.

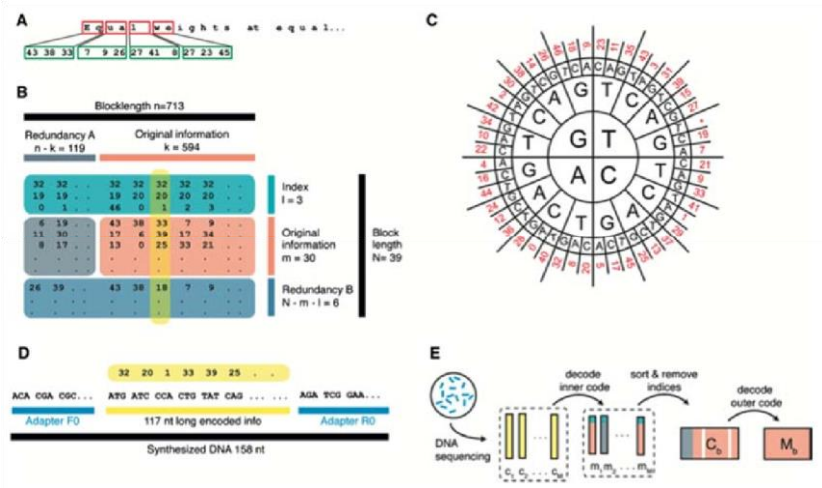

Figure 4: Encoding text into DNA strands using Galois field [18]

In the next step, the authors of [18] used the Reed-Solomon code to add logical redundancy to each element row and the generated elements near each row as inner coding. Then, they added another logical redundancy for each column using RS code and added the generated elements underneath each column (Figure 4-B) as outer coding. Finally, an index is added for every column, and every element gets replaced with a corresponding nucleotide after generating the final block picked up from the Galois wheel (Figure 4-D). The Galois wheel is a mapping table for converting the 47 elements into a three-nucleotide string designed to prevent building homo-polymers (Figure 4-C). Then, the synthesizer converts every column to an actual DNA strand. After sequencing the related strands and generating the primary data block, reversing process results in the primary binary data in order to convert DNA strands back to binary data (Figure 4-E).

In 2016, Blawat et al. [19] offered a new encoding method based on converting each byte of binary data to 5 nucleotides. To do this, they have built two tables, as shown in Figure 5. They used Table 1 for the first six bits of higher value, and Table 2 is used for the 6th and 7th bits (from left). In this way, bits 0 and 1 are mapped to the First nucleotide, 2 and 3 to the second, 4 and 5 to the fourth, and 6 and 7 to the third and fifth.

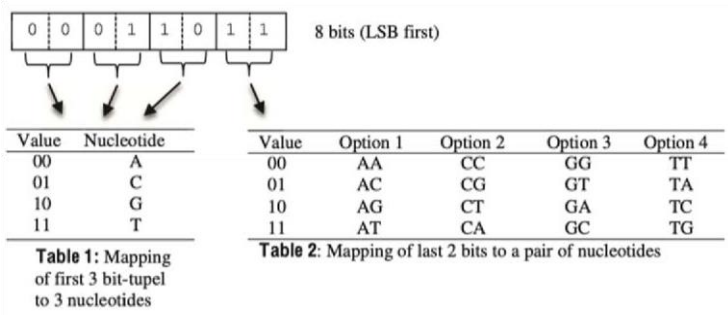

Figure 5: Mapping tables used for converting binary data into nucleotides [19]

It is worth noting that using the four options in Table 2 prevents some common errors in DNA strand synthesizing. Therefore, there are two rules for choosing the best option from Table 2. Firstly, in a byte conversion to 5 nucleotides, the first three nucleotides (from left) should not be similar, and secondly, the last two nucleotides should not be similar. Using this method, they finally showed that there would be at least 2 DNA strands option for a byte of the binary value. Thus, they have called these clusters A and B to have two clusters to choose the final DNA strands for each byte. They showed that there might be another option available called cluster C, but it is not always available. The ability to choose strands from either cluster A or B (and sometimes C) gives enough freedom to generate better strands and efficiently manage the process. They have also added a parity bit to every strand so that in case of any fault, they can discard it and choose alternatives.

4. The Proposed DMFB Architecture

As described before, two main processes in a DNA-based storage system are reading and writing. In a WRITE process, synthesized DNA strands are moved into the DNA pools, and in the READ process, the liquid from the synthesizer should get mixed with the contents of one or more pools and then sent to sequencers. These processes are done manually using tools in a laboratory, like a pipette in a conventional scenario.

This paper shows that the READ/WRITE process in DNA-based storage can be done using a DMFB efficiently. We propose a novel DMFB architecture that controls the scheduling process to activate signals for different electrodes in a DMFB. We suppose that the synthesizer and sequencer modules are out of our DMFB. Therefore, we need to facilitate the moving of DNA-consisted droplets between the storage pools and these devices with maximum parallelism and controllability.

Figure 6 shows the overall view of the proposed architecture. In this architecture, droplets enter from the bottom part of the chip and exit from the top in the WRITE process or exit from the right in the reading process [20]. As shown in Figure 6, pools are placed at the top of DBMF to minimize the droplet's path, and outputs to the sequencers are at the right part of the chip. Moreover, mixer modules that are activated in a READ process are diagonally arranged so there would be no conflict between droplets to perform parallel readings. The green arrow shows the travel path for droplets in the WRITE process, while the blue arrows show how the reading process is done in two steps. In the WRITE process, the mixer module in the path gets deactivated, as shown in the gray in the picture.

Considering R as the number of rows of the chip and C as the number of columns, and N as the number of droplets, we define S, E, CM, CW, and CR as chip area, the minimum number of electrodes, number of cycles for MIX process, cycles for the WRITE process, and cycles for the READ process respectively. It can be quickly approved that for N number of droplets, there should be at least R=2N rows and C=2N columns to provide enough space for droplets to move across the chip without sticking together. Therefore, the minimum value of chip area (S) can be calculated as Equation 1.

$$S = R \times C = 4 \times N^2 \quad (1)$$

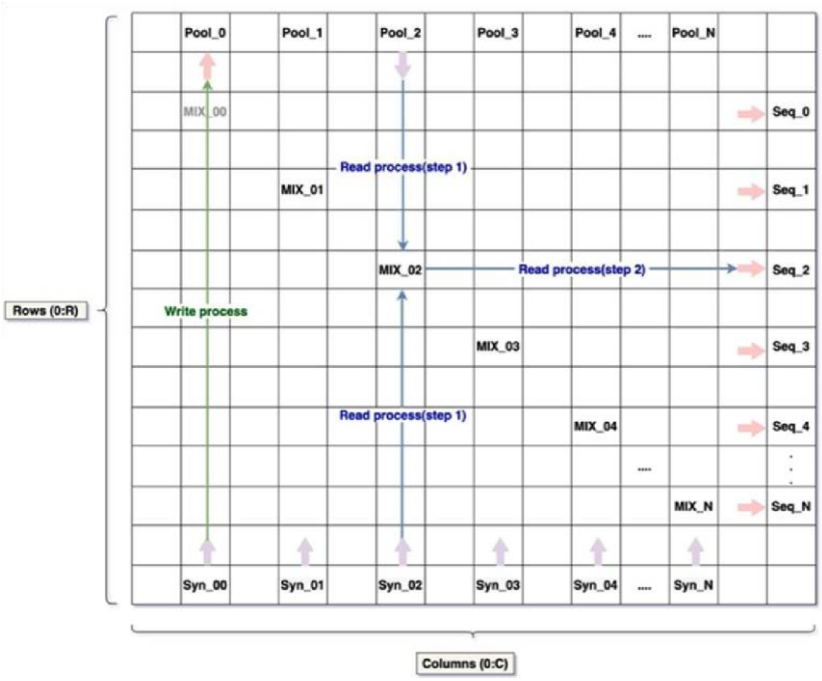

Figure 6: Overview of the proposed architecture

To calculate the number of required electrodes for a complete READ/WRITE process using N droplets, the number of electrodes (E) can be calculated as:

$$E = N \times R + N \times \frac{(C-1+1)}{2} = N \times (R + \frac{C}{2}) \quad (2)$$

By replacing the R and C variables in the above equation, we have:

$$E = N.(2 \times N + 2 \times fracN2) = 3 \times N^2 \quad (3)$$

Therefore, the number of electrodes needed to build this DMFB is in the order of N2, which is an accepted order. Considering BR and BC as blank rows between sequencers and blank columns between synthesizers, the CW and CR will be as Equation 4.

$$C_W = N \times (B_R + 1) \quad (4)$$

Furthermore, CR can be computed as:

$$C_R = N.(B_R + 1) + N \times (B_C + 1) + C_M = N.(B_R + B_C + 2) + C_M \quad (5)$$

It is now proved that the CW and CR have a direct linear relationship with the number of droplets.

The above formulas demonstrate that parallel READ and WRITE processes can be performed efficiently, and the system's complexity is increased linearly when the size of the system grows. It is worth noting that the proposed architecture, with its dynamic size, can be adapted to the size of any parallel process needed to perform DNA storage. The cost overhead will be in a linear relationship to the size of the chip. Here CAD algorithm of droplet routing is described to show how this architecture works. When a WRITE process is supposed to happen, droplets containing DNA molecules get placed at the bottom row of the DMFB. The droplets are in the same column as the corresponding pools at the top of the DMFB. As in a WRITE process, it is only needed to put the droplets into appropriate pools (considering droplets are initially placed in their appropriate place in the same column of the corresponding pool); the control signals will activate the electrodes on the upper row (and same column) of the droplets. Therefore, droplets move one row up. The same process gets repeated until all droplets parallelly reach the pools at the top of the DMFB. In a READ process, droplets from the bottom row (containing required primers) move toward the pools on top. At the

same time, sample droplets from pools at the top move toward the bottom row. As stated before, in every column, there is a Mixer module. Whenever a droplet reaches the Mixer module in the same column, it waits for another droplet to arrive. When both droplets reach the mixer module, it starts to mix these two droplets (for specified time cycles), and then the mixed droplet moves in the right direction toward the appropriate sequencer output. This way, the WRITE, and READ processes are done using our architecture on a DMFB.

5. The Proposed DNA Encoding

As mentioned before, the fault rate of DMFBs is considerable, and this challenge must get noticed in a practical scenario. Many fault tolerance improvement methods in DNA-based computation systems are generally based on encoding the binary data before converting them to DNA strands. Usually, the binary data gets compressed, and then some bits are added to them to increase the fault tolerance of the data in these methods, making it more straightforward to synthesize and recover the data with fewer errors.

The proposed method in this paper differs from this aspect as it is done independently of the type of compression or encoding used for binary data. It can
be added to any data before the synthesis process. In other words, this method works in both binary-encoded data and DNA-encoded data scope.

The base of this method is mapping the logical XOR operations from binary-coded data to nucleotide-coded data. We use a logical operator called nt-XOR, the same logical operations on nucleotide operands. The nt-XOR here is entirely similar to [16] which uses two bits for any nucleotide as Table 1.

Table 1: A possible binary to DNA encoding method

Nucleotide	A	C	G	T
Binary equivalent	00	01	10	11

The truth table for the equivalent binary values of the above table will be as Table 2.

Table 2: The truth table of two-bits XOR in nucleotide-encoded data

	A	**C**	**G**	**T**
A	00	01	10	11
C	01	00	11	10
G	10	11	00	01
T	11	10	01	00

Thus, the truth table for an nt-XOR on nucleotide values will be as Table 3. The important point about the new nt-XOR is that it is directly used for a parity check

Table 3: The truth table of nt-XOR operation on nucleotide values

nt XOR	**A**	**C**	**G**	**T**
A	A	C	G	T
C	C	A	G	T
G	G	T	A	C
T	T	G	C	A

operation for nucleotides. It can detect any change (e.g., fault or error) in the original strands when this parity is violated. However, the parity operations can be done more advanced, like using a parity block that detects the errors and can correct some too. In a parity block, the string of binary data is organized as a two-dimensional array in which a separate parity bit is generated for each row and column. As a result, any change in the value of an element of this array will violate the parity bits in the corresponding row and column, as shown in Table 4. The changed values are highlighted.

Table 4: A parity block on binary values

Row parity = 0	0	1	0	1
Row parity = 1	0	1	1	1
Row parity = 0	1	1	1	1
Row parity = 1	1	0	1	1
	Column parity = 0	**Column parity = 1**	**Column parity = 1**	**Column parity = 0**

Using the above concept, we will implement the parity block on a string of nucleotides. Consider having a string of 64 nucleotides organized as a two-dimensional array as shown in Table 5. In this table, 64 nucleotides are shown that are divided into eight strings with a length of 8 and put each string in each row.

We calculated the nucleotide-based parity for each row and column by applying the nt-XOR on all rows and columns. Therefore, the corresponding nucleotide-based parity for each row and column will be changed with a change

Table 5: A parity block on a string of 64 nucleotides

	Parity Column	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	A	T	G	C	G	A	T	C	A
Byte 1	C	A	G	C	T	G	C	A	G
Byte 2	T	C	T	G	T	T	G	C	A
Byte 3	A	G	C	T	A	A	C	C	A
Byte 4	A	G	T	A	C	C	A	T	G
Byte 5	T	C	G	A	C	G	A	C	G
Byte 6	A	T	T	C	C	G	G	A	A
Byte 7	G	T	A	G	A	C	G	T	T
Parity Row		T	A	G	T	C	C	A	C

on any element of the table which detects the data corruption. In this scheme, data corruption will be detectable and corrected by saving the primary calculated values for each row, column, and parity row using Equation 6.

$$CorrectedV\ alue = CV \oplus NP \oplus OP \quad (6)$$

where $\oplus$ is nt-XOR as described in Table 3.

Where CV is the changed value after data corruption, NP is the newly calculated parity, and OP is the old calculated parity. It can be proved that this method can detect and correct up to two errors.

We build our block using eight rows and 20 columns to implement the above parity block for a DNA strand (w) due to the limitation of a strand length (about 200 nucleotides, as already said [10]). Therefore, it consists of 160 nucleotides for

data and 28 nucleotides for parity (e.g., eight parity nucleotides for eight rows and 20 nucleotides for 20 columns). We can have another nucleotide parity for checking the 28 parity nucleotides. Thus, the total number of nucleotides for the payload part of the strand will be 189, as shown in Table 6.

Table 6: The DNA strand structure of the proposed encoding

Primer(5nt)	Parity 1nt	Parity (28 nt)	Data (160 nt)	Primer(5nt)

Table 6 shows that every strand in our DNA data storage can have this

structure. The synthesizer can produce the parity nucleotides according to the nt-XOR formula. Therefore, we can have a parity block for any binary data.

The above method is entirely independent of the primary encoding of the binary data and is done at the nucleotide level.

5. Simulation Results

The encoding method proposed in this paper improves the fault-tolerant in the cost of logical redundancy due to the extra nucleotides. We calculated its logical density to compare it with other methods. Logical density is the number of bits that can be described with each nucleotide. We assumed that the payload length of the final strand would be 189 nucleotides (e.g., 160 nucleotides for data and 29 nucleotides for parity). Moreover, we have taken that each nucleotide is equal to 2 bits. Therefore, the logical density of the proposed method can be calculated as follows:

$$\frac{(160 \times 2)bits}{189nt} \simeq 1.7bit/nt \quad (7)$$

The calculated value of 1.7 bit/nt shows a great density compared to other encoding methods. Table 7, compares this paper's achievements in comparison with some other methods:

As shown in Table 7, the proposed encoding has a higher logical density than other methods. This improvement decreases the cost and complexity of the storage system.

6. Conclusion

DNA-based data storage is a new technology to overcome the challenges of current storage systems. A major limitation of this technology is working with the

Table 7: The truth table of nt-XOR operation on nucleotide values

Study	Strand length (nucleotides)	Logical density (bits/nt)	Logical density (Payload only)
Church et al. [16]	115	0.60	0.83
Goldman et al. [17]	117	0.19	0.29
Grass et al. [18]	158	0.86	1.16
Bornholt et al. [4]	117	0.57	0.85
Erlich and Zeilinsky [21]	152	1.18	1.55
Blawat et al. [19]	230	0.89	1.08
Organick et al. [22]	150-200	0.81	1.10
This Paper	199	1.60	1.69 ≃ 1.7

liquid environment of DNA operations. This paper's proposed digital microfluidic architecture removes the human manual operations with DNA-consisted solutions and provides a parallel and controllable platform for this storage. Aside from a platform for DNA-based storage, this paper proposed a novel method to encode binary data to nucleotides to improve the fault tolerance of the DNA-based system.

In the proposed encoding, adding extra information to the original data to make it fault-tolerant is done at the nucleotide level instead of the binary level. The synthesizers can learn how to calculate the parity nucleotides and help with error detection and correction. The proposed nucleotide-based encoding is independent of the binary data encoding that enables both techniques to be feasible concurrently. Our analyses show that the overhead of this method is also lower than other methods.

It is worth noting that all the resources related to this research are available by sending a message to jahanian@sbu.ac.ir.

7. Conflict of interest

On behalf of all authors, the corresponding author states that the authors have no conflicts to disclose.

References

[1] Y. Dong, F. Sun, Z. Ping, Q. Quyang and L. Qian, DNA storage: research landscape and future prospects, In National Science Review, Vol.7, No.6, June 2020.

[2] Joao Henrique Diniz, Brandao Gervasio, Henrique da Costa Oliveira, Andre Guilherme da Costa Martins, Joao Bosco Pesquero, Bruno Marinaro Verona, Natalia Neto Pereira Cerize, How close are we to storing data in DNA?, Trends in Biotechnology, DOI: 10.1016/j.tibtech.2023.08.001, 2024.

[3] Meng Yu, Xiaohui Tang, Zhenhua Li, Weidong Wang, Shaopeng Wang d, Min Li d, Qiuliyang Yu, Sijia Xie, Xiaolei Zuoand Chang Chen, High-throughput DNA synthesis for data storage, In Chem. Soc. Rev., 2024, 53, 4463-4489.

[4] J. Bornholt, R. Lopez, D. M. Carmean, L. Ceze, G. Seelig, and K. Strauss, "A DNA-based archival storage system," in Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems, 2016, pp. 637-649.

[5] Nirmala Natarajan and Gracia Nirmala Rani Duraisamy , Optimization techniques in digital microfluidic biochips: a survey of sample preparation algorithmic solutions and challenges, In Springer Microfluidics and Nanofluidics, V.29, No. 52, 2025.

[6] Ritwika Majumdar, Piyali Datta, Sagarika Chowdhury and Rajat Kumar Pal, Advancement with digital microfluidic biochips towards sustainability and secured outcome: a comprehensive survey on specific design metrics, Discover Electronics, Vol.1, No.14, 2024.

[7] Y. Cevallos et. al, A brief review on DNA storage, compression, and digitalization, In Nano Communication Networks, Vol.31, March 2022.

[8] K. Chakrabarty and F. Su, Digital microfluidic biochips: synthesis, testing, and reconfiguration techniques. CRC press, 2006.

[9] K. Chakrabarty, "Automated design of microfluidics-based biochips: connecting biochemistry to electronics CAD," in 2006 International Conference on Computer Design, 2006, pp. 93-100.

[10] L. Ceze, J. Nivala, and K. Strauss, "Molecular digital data storage using DNA," Nature Reviews Genetics, vol. 20, no. 8, pp. 456-466, 2019.

[11] J. Bornholt, R. Lopez, D. M. Carmean, L. Ceze, G. Seelig, and K. Strauss, "Toward a DNA-based archival storage system," Ieee Micro, vol. 37, no. 3, pp. 98-104, 2017.

[12] J. Ding, K. Chakrabarty, and R. B. Fair, "Scheduling of microfluidic operations for reconfigurable two-dimensional electro-wetting arrays," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 20, no. 12, pp. 1463-1468, 2001.

[13] Z. Li, K. Y.-T. Lai, P.-H. Yu, T.-Y. Ho, K. Chakrabarty, and C.-Y. Lee, "High-level synthesis for micro-electrode-dot-array digital microfluidic biochips," in Proceedings of the 53rd Annual Design Automation Conference, 2016, pp. 1-6.

[14] D. Carmean, L. Ceze, G. Seelig, K. Stewart, K. Strauss, and M. Willsey, "DNA data storage and hybrid molecular-electronic computing," Proceedings of the IEEE, vol. 107, no. 1, pp. 63-72, 2018.

[15] S. Newman et al., " High density DNA data storage library via dehydration with digital microfluidic retrieval," Nature communications, vol. 10, no. 1, pp. 1-6, 2019.

[16] G. M. Church, Y. Gao, and S. Kosuri, ”Next-generation digital information storage in DNA,” Science, vol. 337, no. 6102, pp. 1628-1628, 2012.
[17] N. Goldman et al., ”Towards practical, high-capacity, low-maintenance information storage in synthesized DNA,” nature, vol. 494, no. 7435, pp. 77-80, 2013.
[18] R. N. Grass, R. Heckel, M. Puddu, D. Paunescu, and W. J. Stark, ”Robust chemical preservation of digital information on DNA in silica with errorcorrecting codes,” Angewandte Chemie International Edition, vol. 54, no. 8, pp. 2552-2555, 2015.
[19] M. Blawat et al., ”Forward error correction for DNA data storage,” Proce dia Computer Science, vol. 80, pp. 1011-1022, 2016.
[20] M. Pourasadollah, M. Taajobian, and A. Jahanian, ”Flexible and Automatable Microfluidic-based Architecture and CAD Algorithm for Implementation of Large DNA Digital Storage,” in 2022 27th International Computer Conference, Computer Society of Iran (CSICC), Feb. 2022, pp. 1-7. doi: 10.1109/C SICC55295.2022.9780499.
[21] Y. Erlich and D. Zielinski, ”DNA Fountain enables a robust and efficient storage architecture,” Science, vol. 355, no. 6328, pp. 950-954, 2017.
[22] L. Organick et al., ”Random access in large-scale DNA data storage,” Nature biotechnology, vol. 36, no. 3, pp. 242-248, 2018.

Mostafa Pourasadollah received his B.S. degree in computer engineering from Babol Noshirbvani University of technology, Babol, Iran 2011 and the M.S. degrees in computer architecture from Shahid Beheshti University, Tehran, Iran in 2019. He is a PhD. Student at Shahid Beheshti University currently and his current research interests are hardware security and biochip design.

Ali Jahanian received the B.Sc. degree in Computer Engineering from University of Tehran, Tehran, Iran in 1996 and the M.Sc. and Ph.D. degrees in Computer Engineering at Amirkabir University of Technology, Tehran, Iran in 1998 and 2008, respectively. He joined Shahid Beheshti University, Tehran, Iran in 2008. His current research interests are focused on Hardware security and Biochip design.

Dr. Maryam Taajobian completed her B.Sc. in computer engineering at Tehran University and her Master degree in computer architecture at Amirkabir University of Technology. He completed his PhD. in the field of computer systems architecture at the Science and Research Unit of Azad University. He is currently a member of the faculty of Mahdishahr branch of Islamic Azad University and works in the field of digital microfluidic chips and computer networks.