Synchronization Algorithms from High-Rate LDPC Codes for DNA Data Storage

Belaïd Hamoum[‡], Aref Ezzeddine[†], Elsa Dupraz[†]
[‡] Lab-STICC, CNRS UMR 6285, Université Bretagne-Sud, Lorient
[†] IMT Atlantique, Lab-STICC, UMR CNRS 6285, F-29238, France,

***Abstract*—Data storage on synthetic DNA is a novel technique that offers improved durability and density compared to conventional storage supports. Current challenges in the practical implementation of DNA data storage include the high cost of DNA synthesis and the significant amount of errors introduced during DNA sequencing. These errors are not only substitutions but also insertions and deletions, which compromise the sequence synchronization and cannot be corrected using conventional error correction methods. To overcome these issues, this paper proposes an error correction scheme that prioritizes the reduction of DNA synthesis costs through the use of high rate codes. The proposed solution utilizes a consensus algorithm that leverages the inherent redundancy in the output data, followed by a proposed synchronization method for correcting residual insertions and deletions after the consensus. Numerical results show the effectiveness of the proposed method at correcting a large amount of errors, while maintaining a high coding rate.**

I. Introduction

Data storage on synthesis DNA molecules is an emerging technology that offers substantial enhancements in durability and density compared to conventional data storage supports [1]. DNA data storage relies on two crucial steps: synthesis, which converts digital data into DNA molecules, and sequencing, which reads the DNA molecules to retrieve the original digital data. Although DNA sequencing costs have significantly decreased over time, DNA synthesis remains expensive, representing the current main bottleneck in the development of this technology [2].

Another concern is that modern sequencing devices, such as the widely used Oxford Nanopore Technology (ONT) sequencer, still introduce a large amount of errors in the read sequences [1], with an error-rate of about 10% [3]. These errors include not just substitutions, but also insertions and deletions, which conventional error correction codes (ECC) designed for wireless communications are unable to handle [4]. As a result, it is essential to develop new ECC specifically for DNA data storage. A crucial aspect to consider is that the sequencer produces a large number of reads of the same input sequence, each with distinct error patterns. Leveraging this inherent redundancy is key for correcting a large number of errors.

Varshamov-Tenengolts (VT) codes [5] were first proposed to correct a single deletion or insertion, and later extended to correct bursts of errors [6], [7], or a few deletions or insertions [8]–[10]. However, these approaches cannot correct substitutions, and they are insufficient for DNA data storage regarding the high error-rate of the sequencer. Alternatively, low to medium rate low density parity check (LDPC) codes with markers [4] or without markers [11] can correct a fair amount of errors of the three types, from a single sequence. Further, the concatenated code construction of [12] combines an inner convolutional Code (CC) with an outer LDPC code, and it can be applied to multiple reads. This construction can correct a large amount of errors (around 10%) from a small number of reads (less than 5), but it relies on coding rates lower than $1/2$, which is costly in terms of synthesis.

In this paper, we present an alternative ECC solution that operates at higher rates and leverages the inherent redundancy across read sequences to correct a substantial number of errors. We first utilize a consensus algorithm from the field of bioinformatics [13], which constructs a candidate sequence using multiple reads. Unfortunately, we observe that a small number of errors across all three types still persist in the candidate sequence. To address this, we combine the approach with a high-rate LDPC code, and propose a synchronization algorithm to correct the residual insertion and deletions errors. The proposed algorithm builds upon the one proposed in [3], which corrected only a single deletion. It attempts to insert or delete bits at specific positions in the coded sequence, retaining positions that enable to satisfy the most parity check equations. The proposed algorithm is evaluated in terms of complexity and theoretical evaluation of the bit error probability after synchronization. It is followed by a standard LDPC decoders targeting remaining substitution errors. Simulation results demonstrate the effectiveness of the proposed method in correcting a large number of errors while maintaining a high coding rate $3/4$, showcasing its potential as a viable solution in the field of DNA data storage.

The paper outline is as follows. Section II presents the channel model for DNA data storage. Section III describes the considered coding scheme and the consensus algorithm. Section IV introduces the synchronization algorithm. Section V shows simulation results.

II. Channel Model

This section presents the channel model we consider to represent the DNA data storage process, encompassing synthesis and sequencing. We consider a sequence $\mathbf{x}$ of length

This work was supported by the Labex Cominlabs with funding from the French National Research Agency (ANR-10-LABX-07-01), and by the PEPR MolecularXiv.

N containing symbols from a quaternary alphabet $\mathcal{A}$. The sequence $\mathbf{x}$ is initially converted into DNA molecules built from nucleotides A,C,G,T, through a synthesis operation [2]. To read the molecules, we consider the widely used ONT sequencer [14], which outputs a large number J of digital sequences $\mathbf{y}^{(j)}$. Each sequence $\mathbf{y}^{(j)}$ has a length $N^{(j)}$ and consists of symbols from the alphabet $\mathcal{A}$.

The ONT sequencer reads k symbols of $\mathbf{x}$ at a time, referred to as k-mers, where $k = 6$. Specifically, the k-mer at time t involves symbols from $\mathbf{x}$ at positions $[\![t-5, t]\!]$, while the k-mer at time $t+1$ involves symbols at positions $[\![t-4, t+1]\!]$. Given that the ONT sequencer introduces three types of errors: insertions (I), deletions (D), and substitutions (S), we define a set of channel events $\mathcal{E} = \{\mathrm{I}, \mathrm{D}, \mathrm{S}, \mathrm{M}\}$, where a match (M) means no error. In addition, we consider for each output $\mathbf{y}^{(j)}$ an underlying sequence of events $\mathbf{e}^{(j)}$ consisting of N symbols $e_t^{(j)} \in \mathcal{E}$. Most prior works assume i.i.d. channel models for the sequences $\mathbf{e}^{(j)}$ [4], [11], [12]. Here, instead, we adopt the more realistic channel model with memory presented in [3] which provides a set of probabilities $\mathbb{P}(e_t|e_{t-1}, k\mathrm{mer}_t)$, where $e_t, e_{t-1} \in \mathcal{E}$, and $k\mathrm{mer}_t$ is the current k-mer.

The coding solution we propose next is not dependent on this specific channel model. However, we use this model in our simulations because it accurately represents the DNA data storage process, thereby yielding more relevant results from a practical perspective.

III. Coding scheme

This section describes the code construction as well as the consensus algorithm which we consider in our approach.

A. Code construction

We consider a binary LDPC code [15] represented by a sparse binary parity check matrix H of dimension $m \times n$, and by a generator matrix G of dimension $k \times n$, where $k = n - m$. Assuming that H is full rank, the code rate is given by $R = k/n$. The LDPC code can also be represented as a bipartite graph between m check nodes (CN) and n variable nodes (VN), with an edge between a VN and a CN if there is a 1 at the corresponding position in H. The average VN and CN degrees are denoted $\tilde{d}_v$ and $\tilde{d}_c$, respectively.

In our scheme, a binary input sequence $\mathbf{u}$ of length k is first encoded into a binary sequence $\mathbf{v} = G^T\mathbf{u}$ of length n. The binary sequence $\mathbf{v}$ is then transformed into a quaternary sequence $\mathbf{x}$ of length $N = n/2$. The sequence $\mathbf{x}$ is passed through the DNA storage channel, which outputs the J sequences $\mathbf{y}^{(j)}$, see Section II. Next, the decoder applies a consensus algorithm [13] onto m of the output sequences $\mathbf{y}^{(j)}$, as we now describe.

B. Consensus Algorithm

The consensus algorithm presented in [13] picks $m \leq J$ sequences $\mathbf{y}^{(j)}$ as inputs and initially constructs a consensus graph. The graph's nodes represent subsequences of length K, which appear more than T times at the same positions in the m input sequences. An edge is created between two nodes if the predecessor's L-length suffix matches the successor's L-length prefix. Moreover, the edge weight is set to $K-\ell$, where ℓ represents the number of common symbols between the two subsequences. The choice of K is crucial, as smaller values can lead to loops in the graph, while larger values can result in disconnected components. Consequently, the consensus algorithm starts with a large K value and gradually decreases it until a single-component consensus graph is established.

Once the consensus graph is constructed, a Viterbi-like algorithm is applied to identify a path between the starting and ending primers. Unlike the usual Viterbi algorithm, multiple paths with different lengths are retained for each node. The algorithm ultimately outputs several sequences of varying lengths, with the one closest to N being considered as the candidate sequence $\mathbf{w}$ for the subsequent synchronization step.

IV. Synchronization Algorithm

We now present our proposed synchronization algorithm, which is applied to the binary version $\mathbf{z}$ of the candidate sequence $\mathbf{w}$ output by the consensus algorithm. We first describe the process for correcting a single deletion, then $t > 1$ deletions, and finally t insertions. Note that any channel error event (insertion, deletion, or substitution) will impact two bits of $\mathbf{z}$.

A. Correcting 1 deletion

First, we assume that $t = 1$ deletion remains in the consensus sequence $\mathbf{w}$, so that $\mathbf{z}$ is of length $N - 2$. The sequence $\mathbf{z}$ is initially divided into $\lfloor N/L_s \rfloor$ blocks of length L_s, where L_s is a key parameter of the algorithm. Then, the algorithm tries to insert two random bits at the start of each block, one after the other. By adding two random bits at the beginning of the b-th block, a sequence $\mathbf{z}^{(b)}$ of length N is produced, for which we can evaluate the LDPC code parity check equations by computing $H\mathbf{z}^{(b)}$. The resulting score c_b is the number of parity check equations satisfied by the sequence $\mathbf{z}^{(b)}$. The sequence $\mathbf{z}^{(b)}$ with the highest score is selected and passed through a standard BP decoder to generate an estimated sequence $\widehat{\mathbf{x}}$. The underlying logic of this algorithm is to convert deletions into substitutions, which are subsequently corrected by the LDPC decoder.

B. Correcting t deletions

Now, when $t > 1$ deletions need to be corrected, we consider two strategies: *greedy* and *exhaustive*. The greedy approach consists of correcting one deletion after another. In this method, $2t - 2$ random bits are first inserted at the beginning of the sequence $\mathbf{z}$, to achieve a length of $N - 2$. The algorithm described in Section IV-A (without the LDPC decoder) is then applied to correct one deletion. Then, two of the random bits are removed from the beginning of the sequence, and the next deletion is corrected from the algorithm of Section IV-A. This process is repeated until all t deletions have been corrected. The final sequence is passed through the LDPC decoder. It is easy to show that the complexity of the greedy approach is in $\mathrm{O}\Big(t \times \Big\lfloor \frac{N}{L_s} \Big\rfloor \times m \times \tilde{d}_c\Big)$, where $\Big\lfloor \frac{N}{L_s} \Big\rfloor$

is the number of positions that are tested in each round, and $m \times \tilde{d}_c$ is the cost of evaluating the m parity check equations, where $\tilde{d}_c \ll n$.

Alternatively, the exhaustive approach consists of correcting the t deletions simultaneously. For a given combination (without replacement) of t blocks $(b_1, \cdots, b_t)$ of length L_s, it inserts two random bits at the beginning of each block and evaluates the code parity check equations. The sequence $\mathbf{z}^{(b_1,\cdots,b_t)}$ with the highest score is passed through the LDPC decoder. The complexity of the exhaustive approach is higher, in $\mathrm{O}\Big(\binom{\lfloor N/L_s \rfloor}{t} \times m \times \tilde{d}_c\Big)$, where $\binom{\lfloor N/L_s \rfloor}{t} \gg t \times \frac{N}{L_s}$ (with equality for $t = 1$) is the number of positions that are evaluated in each round. Although the exhaustive strategy is more complex, our simulation results will show that it is far more effective at correcting t deletions. Note that the previous complexity analyses do not account for the LDPC decoder, which is the same in both the greedy approach and the exhaustive approach.

C. Correcting deletions and substitutions

The previous synchronization algorithms which corrected t deletions can also be applied without any change in case the consensus sequence also contains substitutions. These substitutions will be corrected by the LDPC decoder, together with the ones introduced by the synchronization method. In this case, assuming that the synchronization process correctly retrieved the t blocks in which the deletions occured, the bit-error probability after synchronization can be expressed as follows.

Proposition 1. *When the synchronization algorithm correctly retrieved the t blocks in which the t deletions occured, and the probability of bit substitution in the consensus sequence is p_{sub}, the bit error probability $\mathbb{P}_e$ after synchronization (and before applying the LDPC decoder) is*

$$\mathbb{P}_e = p_{sub} + \frac{t}{n}\left(1 + \frac{L_s}{2}\right)\left(\frac{1}{2} - p_{sub}\right). \tag{1}$$

The proof is omitted due to the lack of space. In the previous Proposition, the number of deletions is expressed with a fixed number t, while we consider a probability of bit substitution p_{sub}. This is because t is a parameter of the algorithm, which is assumed to be known and can be inferred from the consensus sequence length. On the opposite, we do not have any insights on the number of substitutions in the consensus sequence, and therefore represent its average number by p_{sub} (obtained *e.g.* from numerical simulations).

At the end, Proposition 1 is useful to evaluate the effect of key parameters, especially t and L_s onto the synchronization performance. In our simulations, we will also use the error probability $\mathbb{P}_e$ in (1) to initialize the LDPC decoder. Note that this error probability does not take into account the case where the algorithm outputs incorrect blocks for the t deletions, although the probability of this event is low when L_s and t are small. This will be discussed into details in the simulation section.

D. Correcting insertions and substitutions

The algorithms of Sections IV-A and IV-B can be adapted to correct 1 insertion and t insertions, respectively. In this case, the synchronization process will try to remove one pair of bits in the beginning of each block. For a combination of insertions and substitutions, the bit-error probability after synchronization can be expressed as follows.

Proposition 2. *When the synchronization algorithm correctly retrieved the t blocks in which the t insertions occured, and the probability of bit substitution in the consensus sequence is p_{sub}, the bit error probability $\mathbb{P}_e$ after synchronization (and before applying the LDPC decoder) is*

$$\mathbb{P}_e = p_{sub} + \frac{t}{n}\left(\frac{1}{2}\left(\frac{L_s}{2} - 1\right) - p_{sub}\left(\frac{L_s}{2} + 1\right)\right) \tag{2}$$

The proof is omitted due to the lack of space. Finally, both the complexity analysis and the expressions of $\mathbb{P}_e$ show that the parameter L_s is key in our algorithm, as it addresses a tradeoff between complexity and performance.

E. Synchronization after consensus

We now describe how the previous algorithms are applied to the consensus sequence $\mathbf{z}$. The actual number of deletions and insertions in the consensus sequence is unknown, but we assume that only deletions and substitutions occured, or only insertions and substitutions occured, and we use the length of the consensus sequence to infer the number of insertions or deletions. Therefore, we apply the following rules, depending on the length N_c of the consensus sequence $\mathbf{w}$:

- If $N_c = N$, we directly apply the LDPC decoder onto $\mathbf{z}$, and the LDPC decoder is initialized with $\mathbb{P}_e = p_{\text{sub}}$
- If $N_c < N$, we apply the synchronization algorithm to correct t deletions, where $t = \frac{N}{2} - N_c$, followed by the LDPC decoder initialized with $\mathbb{P}_e$ in (1).
- If $N_c > N$, we apply the synchronization algorithm to correct t insertions, where $t = N_c - \frac{N}{2}$, followed by the LDPC decoder initialized with $\mathbb{P}_e$ in (2).

For simplicity, we do not consider the case where both insertions and deletions remain in the consensus algorithm, leaving this case for future works. We now evaluate the performance of this approach from numerical simulations.

V. Simulation Results

Throughout this section, we consider short sequences of length $N = 256$, which is a typical length given the synthesis constraints [16]. We use a regular LDPC code of rate $R = 3/4$, with fixed degrees $d_v = 3$ and $d_c = 12$ [15]. We first evaluate the ability of the synchronization algorithm proposed in Section IV-B to correct t deletions. Figure 1 shows the Bit Error Rate (BER) after the synchronization algorithm with respect to the number t of deletions. We consider both the greedy approach with $L_s = 2$, and the exhaustive approach with $L_s = 2, 4, 16$. We observe that the exhaustive approach is far more efficient than the greedy approach, although it is more complex. We also observe a clear gain at considering

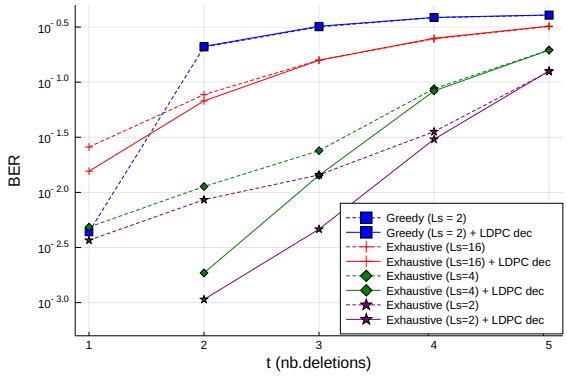

Fig. 1. BER after synchronization, for t deletions, for various values of L_s, for the greedy algorithm and the exhaustive algorithm. BER before LDPC decoding (dashed lines) and after LDPC decoding (plain lines) are provided.

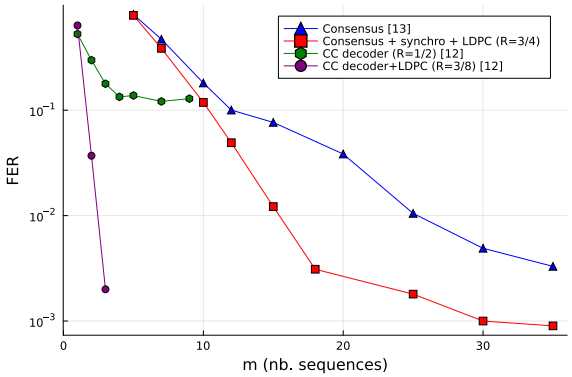

Fig. 3. FER of several solutions with varying rates, with respect to m (number of sequences used to decode).

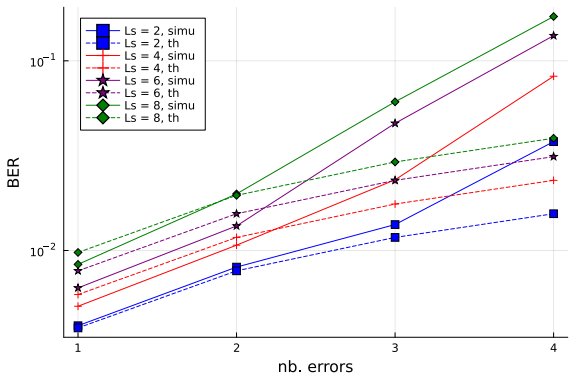

Fig. 2. Comparison between the BER measured from simulations after the synchronization algorithm, and the error probability provided in Proposition 1, for various values of L_s, for the exhaustive algorithm.

the smallest value $L_s = 2$, as well as the positive effect of the LDPC decoder. Note that the behavior of the algorithm for correcting insertions is very similar to the algorithm for correcting deletions.

Next, we aim to evaluate the accuracy of the error probability provided in Proposition 1. For $p_{\text{sub}} = 0$ and for the exhaustive approach, Figure 2 shows the BER with respect to t and L_s, both calculated from $\mathbb{P}_e$ in (1) and measured from Monte Carlo (MC) simulations. We observe that for the smallest considered values $L_s = 2, 4$, the error probability $\mathbb{P}_e$ predicts accurately the BER measured from MC simulations up to $t = 3$ deletions, which shows the ability of the synchronization algorithm to retrieve the correct positions of the t deletions when L_s is small. On the other hand, when L_s increases, a clear gap between $\mathbb{P}_e$ and the measured BER appears, showing the effect of incorrect identification of deletions positions.

To finish, we aim to compare the performance of the proposed approach (consensus + synchronization + LDPC decoding) against existing ones, by considering the realistic DNA storage channel model with memory proposed in [3]. We consider our approach summarized in Section IV-E used with the exhaustive algorithm and $L_s = 2$, and we evaluate three existing solutions which were also developed to correct the three types of error: (i) the consensus algorithm of [13] alone, without coding, (ii) the CC code and decoder proposed in [12], with coding rate $R = 1/2$, (iii) the concatenated construction of [12], built from a CC code of rate $1/2$ and a regular LDPC code of rate $3/4$, resulting in a rate $R = 3/8$. These solutions are compared in Figure 3 in terms of Frame Error Rate (FER) with respect to the number of sequences m which are used for the decoding. First, we observe a clear gain of the proposed approach compared to the consensus algorithm alone. Then, the CC code alone can be applied to a smaller number m of sequences, but it exhibits a high error floor around 10^{-1}, while our proposed solution allows to reach FER levels of about 10^{-3}. Finally, the concatenated code construction of [12] allows to achieve very low FER from a very small number of sequences $m = 3$. However, this construction has a code rate $R = 3/8$, which means that over the $N = 256$ stored bits, only $k = 96$ bits convey useful information. On the opposite, in our approach, the number of information bits is $k = 192$, which is much higher. We conclude that given that the sequencer outputs anyway a large number of sequences m, our proposed solution represents a better tradeoff in terms of synthesis cost.

VI. Conclusion

In this paper, we proposed a complete solution to correct insertions, deletions, substitutions, introduced by the DNA data storage channel. Our solution used a consensus algorithm [13] followed by a proposed synchronization algorithm which corrects residual insertions or deletions. The proposed solution shows improved performance compared to the consensus alone, while requiring more input sequences than the concatenated code construction of [12]. However, our solution operates at much higher coding rates, which represents an interesting tradeoff in terms of synthesis costs. Future works will aim to improve the synchronization algorithm to correct combinations of insertions and deletions that could remain in the consensus sequence.

REFERENCES

[1] R. Heckel, G. Mikutis, and R. N. Grass, "A Characterization of the DNA data storage channel," *Scientific Reports*, vol. 9, no. 1, p. 9663, 2019.

[2] C. K. Lim, S. Nirantar, W. S. Yew, and C. L. Poh, "Novel modalities in dna data storage," *Trends in Biotechnology*, vol. 39, no. 10, pp. 990–1003, 2021.

[3] B. Hamoum, E. Dupraz, L. Conde-Canencia, and D. Lavenier, "Channel model with memory for DNA data storage with nanopore sequencing," in *11th International Symposium on Topics in Coding (ISTC)*, 2021, pp. 1–5.

[4] F. Wang, D. Fertonani, and T. M. Duman, "Symbol-level synchronization and LDPC code design for insertion/deletion channels," *IEEE transactions on communications*, vol. 59, no. 5, pp. 1287–1297, 2011.

[5] R. R. Varshamov and G. M. Tenengol'ts, "A code that corrects single unsymmetric errors," *Avtomatika Telemekhanika*, vol. 26, no. 2, pp. 288–292, 1965.

[6] V. Levenshtein, "Asymptotically optimum binary code with correction for losses of one or two adjacent bits," *Problemy Kibernetiki*, vol. 19, pp. 293–298, 1967.

[7] C. Schoeny, A. Wachter-Zeh, R. Gabrys, and E. Yaakobi, "Codes correcting a burst of deletions or insertions," *IEEE Transactions on Information Theory*, vol. 63, no. 4, pp. 1971–1985, 2017.

[8] V. I. Levenshtein *et al.*, "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet physics doklady*, vol. 10, no. 8. Soviet Union, 1966, pp. 707–710.

[9] A. Helberg and H. Ferreira, "On multiple insertion/deletion correcting codes," *IEEE Transactions on Information Theory*, vol. 48, no. 1, pp. 305–308, 2002.

[10] S. K. Hanna and S. El Rouayheb, "Guess & check codes for deletions, insertions, and synchronization," *IEEE Transactions on Information Theory*, vol. 65, no. 1, pp. 3–15, 2018.

[11] R. Shibata, G. Hosoya, and H. Yashima, "Design of irregular LDPC codes without markers for insertion/deletion channels," in *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.

[12] A. Lenz, I. Maarouf, L. Welter, A. Wachter-Zeh, E. Rosnes, and A. G. i Amat, "Concatenated codes for recovery from multiple reads of DNA sequences," in *IEEE Information Theory Workshop (ITW)*, 2021, pp. 1–5.

[13] D. Lavenier, "Constrained consensus sequence algorithm for DNA archiving," *CoRR*, vol. abs/2105.04993, 2021. [Online]. Available: https://arxiv.org/abs/2105.04993

[14] Y. Wang, Y. Zhao, A. Bollas, Y. Wang, and K. F. Au, "Nanopore sequencing technology, bioinformatics and applications," *Nature biotechnology*, vol. 39, no. 11, pp. 1348–1365, 2021.

[15] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Transactions on information theory*, vol. 47, no. 2, pp. 599–618, 2001.

[16] D. Lavenier, "DNA Storage: Synthesis and Sequencing Semiconductor Technologies," in *IEDM 2022 - 68th Annual IEEE International Electron Devices Meeting*. San Francisco, United States: IEEE, Dec. 2022, pp. 1–4. [Online]. Available: https://hal.science/hal-03902786