PNAS RESEARCH ARTICLE | COMPUTER SCIENCES
BIOPHYSICS AND COMPUTATIONAL BIOLOGY OPEN ACCESS

Parallel molecular computation on digital data stored in DNA

Boya Wang[a,1,2,3], Siyuan Stella Wang[b,c,1,4], Cameron Chalk[a,5], Andrew D. Ellington[b,c], and David Soloveichik[a,2]

Edited by Darko Stefanovic, The University of New Mexico, Albuquerque, NM; received October 14, 2022; accepted July 10, 2023, by Editorial Board Member James J. Collins

DNA is an incredibly dense storage medium for digital data. However, computing on the stored information is expensive and slow, requiring rounds of sequencing, in silico computation, and DNA synthesis. Prior work on accessing and modifying data using DNA hybridization or enzymatic reactions had limited computation capabilities. Inspired by the computational power of "DNA strand displacement," we augment DNA storage with "in-memory" molecular computation using strand displacement reactions to algorithmically modify data in a parallel manner. We show programs for binary counting and Turing universal cellular automaton Rule 110, the latter of which is, in principle, capable of implementing any computer algorithm. Information is stored in the nicks of DNA, and a secondary sequence-level encoding allows high-throughput sequencing-based readout. We conducted multiple rounds of computation on 4-bit data registers, as well as random access of data (selective access and erasure). We demonstrate that large strand displacement cascades with 244 distinct strand exchanges (sequential and in parallel) can use naturally occurring DNA sequence from M13 bacteriophage without stringent sequence design, which has the potential to improve the scale of computation and decrease cost. Our work merges DNA storage and DNA computing, setting the foundation of entirely molecular algorithms for parallel manipulation of digital information preserved in DNA.

DNA storage | DNA computation | molecular programming | strand displacement

DNA is an incredibly dense (up to 455 exabytes per gram, 6 orders of magnitude denser than magnetic or optical media) and stable (readable over millennia) digital storage medium (1–3). Storage and retrieval of up to gigabytes of digital information in the form of text, images, and movies have been successfully demonstrated. DNA's essential biological role ensures that the technology for manipulating DNA will never succumb to obsolescence. However, performing computation on the stored data typically involves sequencing the DNA, electronically computing the desired transformation, and synthesizing new DNA. This expensive and slow loop limits the applicability of DNA storage to rarely accessed data (cold storage).

In contrast to traditional (passive) DNA storage, schemes for dynamic DNA storage allow access and modification of data without sequencing and/or synthesis. Upon binding to molecular probes, files can be accessed selectively (4) and modified through PCR amplification (5). Introducing or inhibiting binding of molecular probes with existing data barcodes can rename or delete files (6). Information encoded in the hybridization pattern of DNA can be written and erased (7) and can even undergo basic logic operations such as AND and OR (8) using strand displacement. By encoding information in the nicks of naturally occurring DNA [a.k.a. native DNA (9)], data can be modified through ligation or cleavage (10). With image similarities encoded in the binding affinities of DNA query probes and data, similarity searches on databases can be performed through DNA hybridization (11, 12). Although these advances allow information to be directly accessed and edited within the storage medium, they nevertheless demonstrate limited or no capacity for computation in DNA.

Conveniently, beyond its role as a storage medium, DNA has proved to be a programmable medium for computation, primarily via "strand displacement" reactions. With the understanding of the kinetics and thermodynamics of DNA strand displacement (13–15), a variety of rationally designed molecular computing devices have been engineered. These include molecular implementations of logic circuits (16–18), neural networks (19, 20), consensus algorithms (21), dynamical systems including oscillators (22), and cargo-sorting robots (23). Given the achievements of strand displacement systems and their inherent molecular parallelism, DNA computation schemes appear well suited to directly carry out computation on big data stored in DNA.

Significance

The exponential accumulation of digital data is expected to outstrip magnetic and optical storage media. DNA's significantly higher information density combined with the availability of technology for manipulating and reading DNA owning to its essential biological role makes DNA an attractive alternative. Often, general information processing on the data stored in DNA requires an inefficient change of domain from the chemical to the electronic and back. Here, we develop dynamic DNA storage capable of general programmable (Turing universal) computation entirely "in chemistry." Based on DNA strand displacement reactions, algorithms are executed on all data entries in parallel. We demonstrate high-throughput readout via next-generation sequencing, identifier-based access to specific data entries, and the use of naturally occurring sequences for greater scalability.

[1]B.W. and S.S.W. contributed equally to this work.

[2]To whom correspondence may be addressed. Email: boyawang.chem@gmail.com or david.soloveichik@utexas.edu.

[3]Present address: Bioengineering, California Institute of Technology, Pasadena, CA 91125.

[4]Present address: Wyss Institute for Biologically Inspired Engineering, Harvard University, Boston, MA 02115.

[5]Present address: Computation and Neural Systems, California Institute of Technology, Pasadena, CA 91125.

This article contains supporting information online at https://www.pnas.org/lookup/suppl/doi:10.1073/pnas.2217330120/-/DCSupplemental.

Published September 5, 2023.

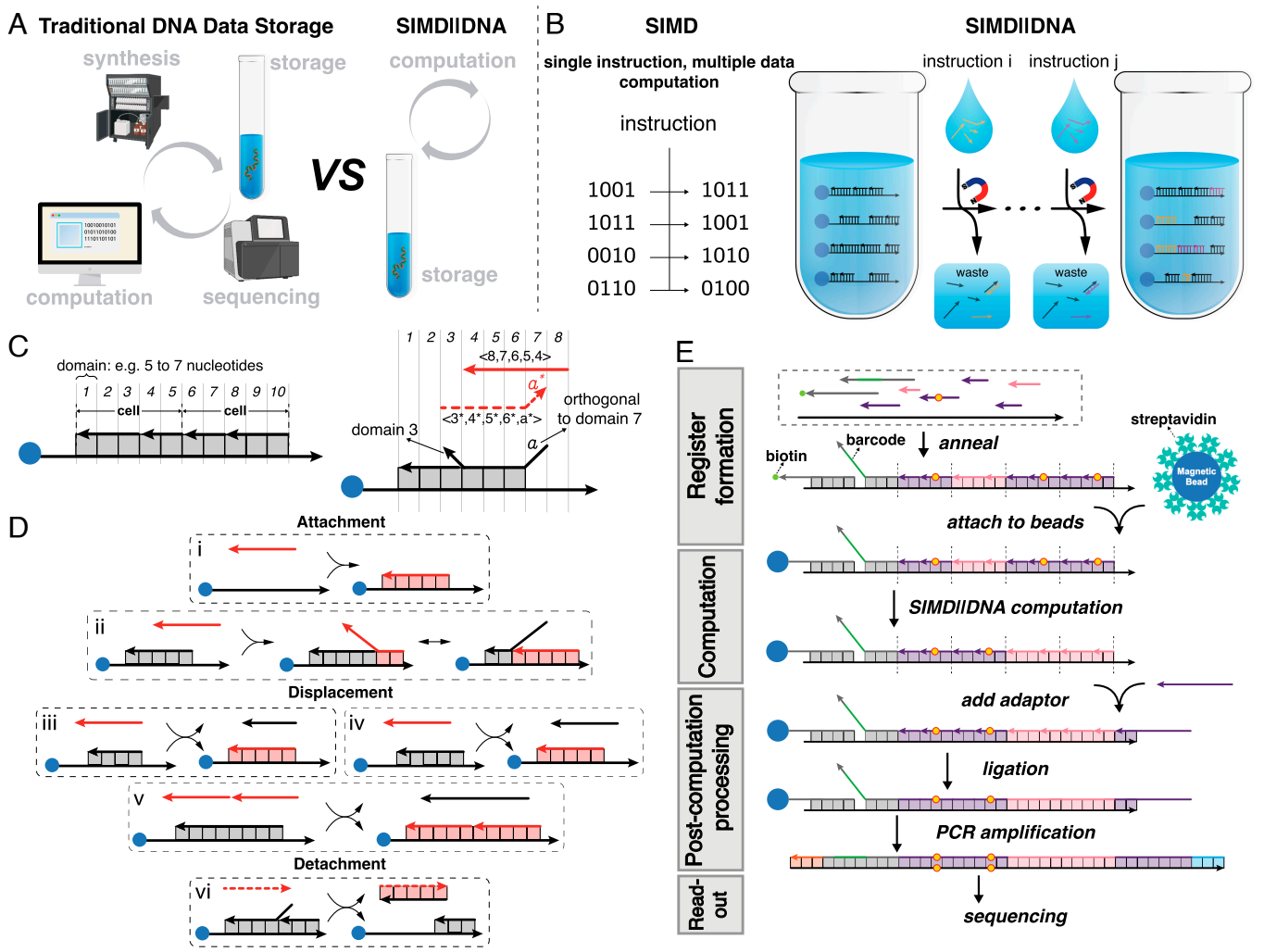

Fig. 1. Overview of SIMD||DNA. (*A*) Instead of outsourcing the computation process to an electronic computer with required sequencing and synthesis steps, SIMD||DNA allows in-memory computation performed by DNA itself. (*B*) Analogous to SIMD computation in classical computing, in SIMD||DNA, a set of instruction strands is added to the solution to simultaneously react with all the registers (multistranded complexes with information encoded in the pattern of nicks and exposed single-stranded regions). Magnetic beads (blue) labeling of registers allows separation of unreacted instruction strands and displaced reaction products. (*C*) Notation: Vertically aligned Top and Bottom domains are complementary unless specifically labeled (e.g., a and a^*, with "*" indicating complementarity). We assume that domains bind only their complements. Dashed strands share the same sequence as their vertically aligned Bottom strands. (*D*) Instruction strands induce three types of events: attachment, displacement, and detachment. The toehold exchange reaction in (*iv*) is considered irreversible since instruction strands are at high concentration. (*E*) Experimental workflow: Registers are assembled through annealing and attached to magnetic beads, followed by computation. The postcomputation process of ligation and PCR amplification prepares for sequencing readout. Yellow dots indicate Top-Bottom strand mismatches, allowing data readout after ligation erases the nick information (secondary sequence–based encoding).

Here, we propose a paradigm called SIMD||DNA* (Single Instruction Multiple Data† computation with DNA) which integrates DNA storage with in-memory computation by strand displacement (Fig. 1*A*). A preliminary version of the theoretical results in this work appeared as a conference paper (25). Inspired by methods of storing information in the positions of nicks in double-stranded DNA (9, 10), SIMD||DNA encodes information in a multistranded DNA complex with a unique pattern of nicks and exposed single-stranded regions (a register). Although storage density is somewhat reduced (approximately a factor of 30; see *Discussion*), encoding information in nicks still achieves orders of magnitude higher density than magnetic and optical technologies. To manipulate information, an instruction (a set of strands) is applied in parallel to all registers which all share the same sequence space (Fig. 1*B*). The strand composition of a register changes when the applied instruction strands trigger strand displacement reactions within that register. Nonreacted instruction strands and reaction waste products are washed away via magnetic bead separation to prepare for the next instruction. Each instruction can change every bit on every register, yielding a high level of parallelism. Our experiments routinely manipulated 10^{10} registers in parallel; DNA storage recovery studies suggest that, in principle, 10^8 distinct register values can be stored in one such sample (*SI Appendix*, section S20).

To enable data multiplexing (i.e., the pooling, separation, and identification of multiple datasets), registers can be labeled with unique identifiers, or "barcodes" (e.g., in putative medical record applications, the barcodes could identify specific patients).

*"||" indicates "parallel computation" and the DNA double helix.

† Single instruction, multiple data (SIMD) is one of the four classifications in Flynn's taxonomy (24). The taxonomy captures computer architecture designs and their parallelism. The four classifications are the four choices of combining single instruction (SI) or multiple instruction (MI) with single data (SD) or multiple data (MD). SI versus MI captures the number of processors/instructions modifying the data at a given time. SD versus MD captures the number of data registers being modified at a given time, each of which can store different information. Our scheme falls under SIMD, since many registers, each with different data, are affected by the same instruction.

Specific information may be retrieved or erased through the addition of a query strand that contains the complement of the barcode sequence and may therefore be used to displace the register from the magnetic bead without disturbing the remaining registers. Unlike typical strand displacement computation systems, SIMD||DNA leverages the bandwidth of next-generation sequencing (NGS) to read out mixed pools of information. *SI Appendix*, section S19 discusses abstract data manipulation operations that SIMD||DNA achieves.

Reliance on custom-synthesized oligonucleotide strands is an important bottleneck to the adoption of both strand displacement and DNA storage systems, limiting their scale and increasing cost. With a few notable exceptions [such as for diagnostic applications (26)], sophisticated sequence design incompatible with natural sequences is typically used (27–29). In the context of strand displacement systems, naturally occurring, nonoptimized DNA sequences may in general lead to undesired binding, trigger spurious displacement, or prevent displacement from completing. Surprisingly, we show that SIMD||DNA algorithms may be realized on the M13 phagemid while eschewing careful sequence design and the high costs and low yield of long oligonucleotide phosphoramidite synthesis. M13, a mainstay of DNA origami (30), provides a large storage space for parallel SIMD computation on multiple registers. To date, we constructed the largest strand displacement system using naturally occurring DNA sequences: using SIMD||DNA, we implemented in total 244 distinct strand displacement reactions.

SIMD||DNA

In SIMD||DNA, molecular algorithms manipulate digital information stored in DNA strand nicks through DNA strand displacement reactions (Fig. 1). Each data register contains a long Bottom strand and multiple bound Top strands. As registers share the same sequence space, the Bottom strand is the same between different registers, while the information is encoded in the location of the nicks between Top strands. *Domains* represent consecutive nucleotides that act as a functional unit. Strand displacement is initiated by hybridization at a single-stranded *toehold* domain followed by displacement of the incumbent strand. The domain lengths are chosen so that ideally 1) each domain can initiate strand displacement (i.e., can act as a toehold), 2) strands bound by a single domain readily dissociate, and 3) strands bound by two or more domains cannot dissociate. The Bottom strand is partitioned into sets of consecutive domains called *cells*, each representing a bit of information (Fig. 1*C*).

Each *instruction* of a program corresponds to the addition of a set of DNA strands at high concentration to registers bound to magnetic beads. Instruction strands are allowed to react for a few minutes before washing to prevent waste products from interacting with registers. Physical separation of instruction strands, waste, and registers can be achieved through applying a magnetic field.

The instruction strands cause three types of events (Fig. 1*D*): *Attachment* events attach new top strands to the register, preserving all the previously bound strands. Attachment can include partial displacement of a preexisting Top strand on the register (Fig. 1, *D*, *ii*). In *displacement* events, an instruction strand displaces and replaces a previously bound Top strand. This event can occur only if a toehold is available for the displacement. Note that since the displaced strands are in relatively low concentration, we assume they do not subsequently react with the registers. We assume toehold exchange (13) is irreversible in our model (Fig. 1, *D*, *iv*). Two instruction strands can also cooperatively (31) displace strands on the register (Fig. 1, *D*, *v*). In a *detachment* event, an instruction strand displaces a complementary Top strand from a register without introducing a new Top strand. This event can only occur if there is a toehold on the Top strand to initiate the displacement.

To enable the parallel reading of a pool of registers holding different data through NGS, we rely on a secondary sequence–level data encoding consisting of single-base mismatches between certain Top strands relative to the Bottom strand. This encoding allows us to continue to distinguish 0's and 1's after ligation and PCR amplification—steps that erase information stored in nicks. To ensure fast kinetics of displacement events in the presence of mismatches, any mismatch is on the displaced strand rather than the displacing strand (32); see *SI Appendix*, section S3.4.

Results

Binary Counting Program. The binary counting SIMD||DNA program treats each register as a binary integer and increments all the registers by one. Abstractly, a binary increment operation flips the right-most (least significant) zero and all bits to its right. At the high level, our SIMD algorithm does the following (Fig. 2*A*): Starting from the rightmost domain (designed to initiate displacement), the program erases all 1's in between the rightmost cell and the rightmost value-0 cell (Instructions 1 and 2), and changes those cells to 0 at Instructions 4 and 5. The rightmost value-0 cell is first marked (Instruction 3) and then changed to value 1 (Instructions 6 and 7). (See *SI Appendix*, section S1 for the proofs of correctness of our programs.)

We show that strand displacement computation can proceed directly on naturally occurring M13mp18 plasmid (M13) sequence space with minimal sequence optimization (Fig. 2). To choose the parts of the M13 plasmid to use, we first eliminated areas with 4 consecutive C's or G's and hairpin "[A]" (33) and then selected 9 random locations as candidates at which we encoded registers (Fig. 2*B*). As a scalable technique to alleviate spurious interactions in naturally occurring DNA while keeping desired binding intact, we increased reaction temperature and the strength of domain binding (increasing their length). We partitioned the sequence from the chosen locations into domains of different lengths according to the binding energy: M13.1, M13.2, and M13.3 registers were designed with weak (short) domains, M13.4, M13.5, M13.6 registers with medium domains, and M13.7, M13.8, M13.9 registers with strong (long) domains (details in *SI Appendix*, section S3.2). We tested initial values 0010 and 0011 with different reaction temperatures (*SI Appendix*, section S6) on these 9 registers and then picked 5 of them for further experiments.

We first performed SISD (single-instruction, single-data) computation on M13.8 registers for each of the 16 4-bit initial values, where each test tube contained only one initial value. After NGS sequencing, the reads were organized according to the barcode sequences associated with their encoded initial values, and the percentage of reads representing the correct value was calculated (*Materials and Methods* in *SI Appendix*, section S5.2). More than 99% of M13.8 registers that were assembled, processed, and sequenced contained the expected initial value (*SI Appendix*, Fig. S4*A*). After a round of binary counting computation, registers corresponding to all 16 initial values show the correct output as the dominant output (Fig. 2*C*), with the minimum correct percent at 68%. We observed similar results for M13.3 registers (*SI Appendix*, Fig. S4*B*).

To demonstrate SIMD computation, we executed the binary counting program on a pool of M13.8 registers with all 16 initial

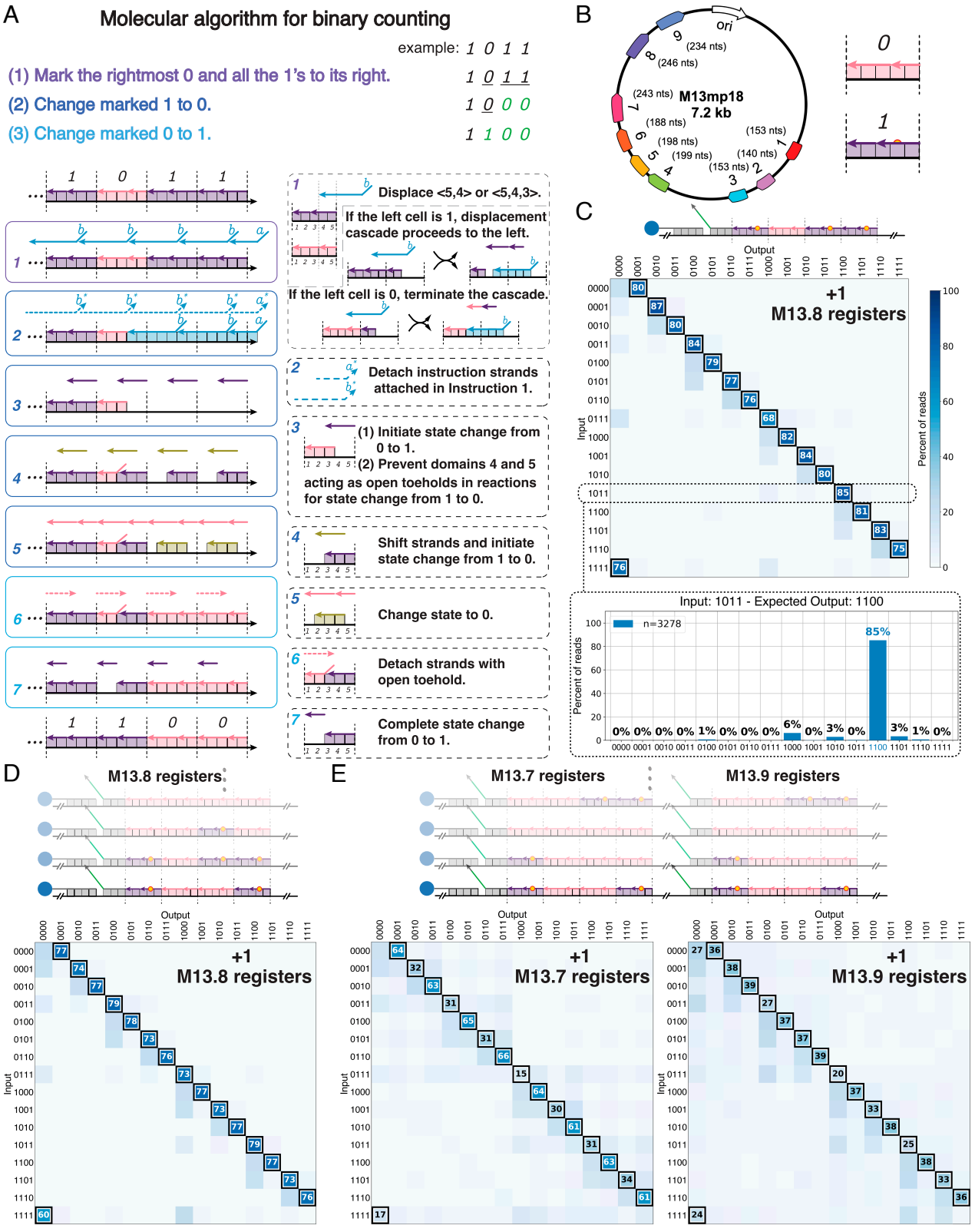

Fig. 2. Binary counting program on naturally occurring (M13) sequences. (*A*) Molecular program implementing binary counting. The example register (Top) encoding the initial value updates to the new value (Bottom) after 7 instructions. Strands are labeled by colors: value 1 (purple), value 0 (pink), intermediates (others). Solid boxes show the instruction strands and the configuration of the register before reacting. Dashed boxes explain the logical meaning of the instructions. (*B*) Locations of registers on M13mp18 and the encoding of 0 and 1. In the text, M13.*i* refers to registers at location choice *i*. The yellow circle on one of the Top strands encoding the value 1 indicates a single-base mismatch with respect to the Bottom strand. (*C*) NGS results of SISD binary counting (40 °C) on M13.8 registers. For each initial value, the distribution of the output values is represented in the heat-map matrix. The lower bar plot shows an example of the distribution of output values in one row (input 1011) of the heat map. (*D*) NGS results of SIMD binary counting (40 °C) on M13.8 registers. (*E*) NGS results of SIMD binary counting (50 °C) on M13.7 and M13.9 registers in parallel. In the heat maps, black borders indicate the correct output value; the percent value is shown if the output value is at least 25% of all reads for a given sample.

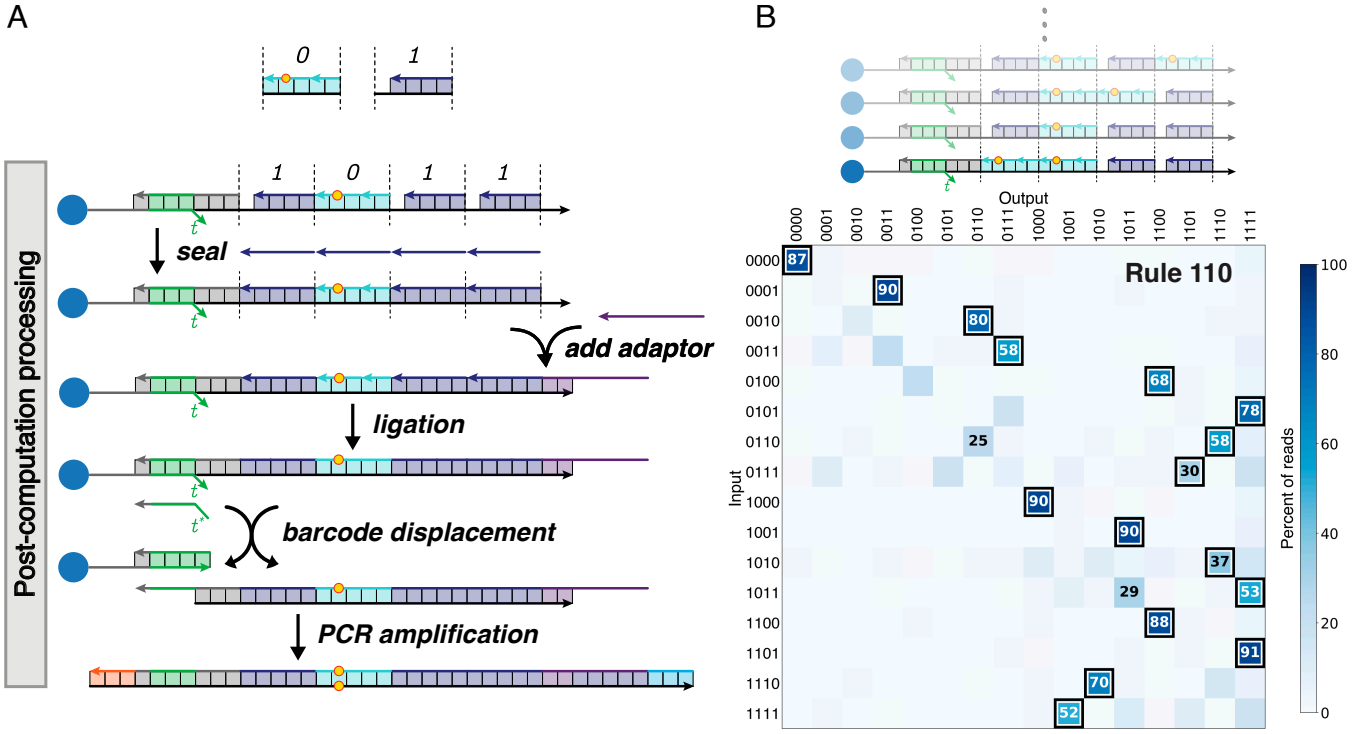

Fig. 3. SIMD cellular automaton Rule 110 computation. (*A*) Postcomputation process for the Rule 110 program with chemically synthesized DNA. After computation, "seal" strands are added to fill in the gap for cells representing bit 1 for the following ligation step. Strands complementary to the barcode sequences are added to selectively displace the register with certain initial values. (*B*) NGS results of SIMD Rule 110 computation (25 °C) on 16 registers with unique initial values. The Leftmost and Rightmost cells of the input only have one neighbor; thus, they undergo a modified Rule 110 update which proceeds as if the missing neighbor were 0 (see *SI Appendix*, section S10 for the implementation of this boundary condition). The registers were composed of chemically synthesized DNA. The correct output value is indicated with a black border; values that appear in at least 25% of all reads for a given sample are printed.

values in the same test tube (*SI Appendix*, Fig. S5*A*), obtaining a similar correct fraction of computation as for SISD (Fig. 2*D*). We also achieved similar SIMD computation results on M13.7 and M13.9 registers (*SI Appendix*, Fig. S5 *B* and *C*) at a higher temperature.

The 7.2-kb-long M13mp18 plasmid is, in principle, capable of accommodating SIMD||DNA computation on several hundreds of bits. As a proof of scaling, we investigated the ability to store (*SI Appendix*, Fig. S6) and compute (Fig. 2*E*) simultaneously on M13.7 and M13.9 registers assembled on the same M13 molecules. As shown in Fig. 2*E*, most registers produced the highest readcount for the correct output, albeit with more errors for some inputs. Note that different programs could be simultaneously executed on registers of orthogonal sequence, thereby potentially generalizing SIMD||DNA to MIMD (multiple instruction, multiple data).

Cellular Automaton Rule 110 Program. Expanding the diversity of functions computable by SIMD||DNA, we implemented a Turing universal program based on cellular automaton Rule 110 (the full program is shown in *SI Appendix*, section S1.2). The Turing universality of Rule 110 (34) argues that, in principle, SIMD||DNA is capable of performing any computation that can be performed on any computer.

An abstract elementary cellular automaton (35), one of the simplest models of computation, consists of a line of cells, each in one of two states 0 or 1. At each time step, updates to a cell depend on the states of its Left and Right neighbors. A simple two-rule characterization of Rule 110 is as follows: 0 updates to 1 if and only if the state to its right is a 1, and 1 updates to 0 if and only if both neighbors are 1. The SIMD||DNA program for implementing a time-step is shown in *SI Appendix*, Fig. S1.

To implement the Rule 110 program, we used artificially designed sequences (sequence design explained in *SI Appendix*, section S3.1) as well as M13. The encoding of bit value 1 for the Rule 110 program contains an exposed (toehold) region; thus, to enable ligation and sequencing, "seal" strands were added at the completion of computation to fill in the gaps on the patterns of the Top strands (Fig. 3*A*). We performed SIMD computation for all 16 initial values taking the bits to the Left and Right of the four register bits as 0 (the implementation of this boundary condition is explained in *SI Appendix*, section S10). The sequencing readout shows that the correct values are the dominant output: Fig. 3*B* shows the data for registers composed of artificially designed sequences (the control without computation is shown in *SI Appendix*, Fig. S8*A*). We achieved similar results using the M13 sequence as seen in *SI Appendix*, Fig. S8*B*.

Random Access (Selective Retrieval and Selective Erasure). Random access is the selective reading, modification, or erasure of data within a database and importantly allows access to a subset of information without expending resources (such as NGS) to read the entire repository each time. Other DNA storage schemes typically use PCR to selectively amplify data (4, 5) or selectively pull out information by tuning the binding affinity between sequences (11, 36). However, designing sequences or multiplexed orthogonal PCR probes with high specificity can be challenging. In contrast, strand displacement achieves specificity through kinetically and energetically favorable reactions that displace a preexisting strand (37).

To accommodate random access in SIMD||DNA, registers holding different data are prepared with specific barcode sequences that can serve as a point of access. After computation on all registers together, query strands complementary to the

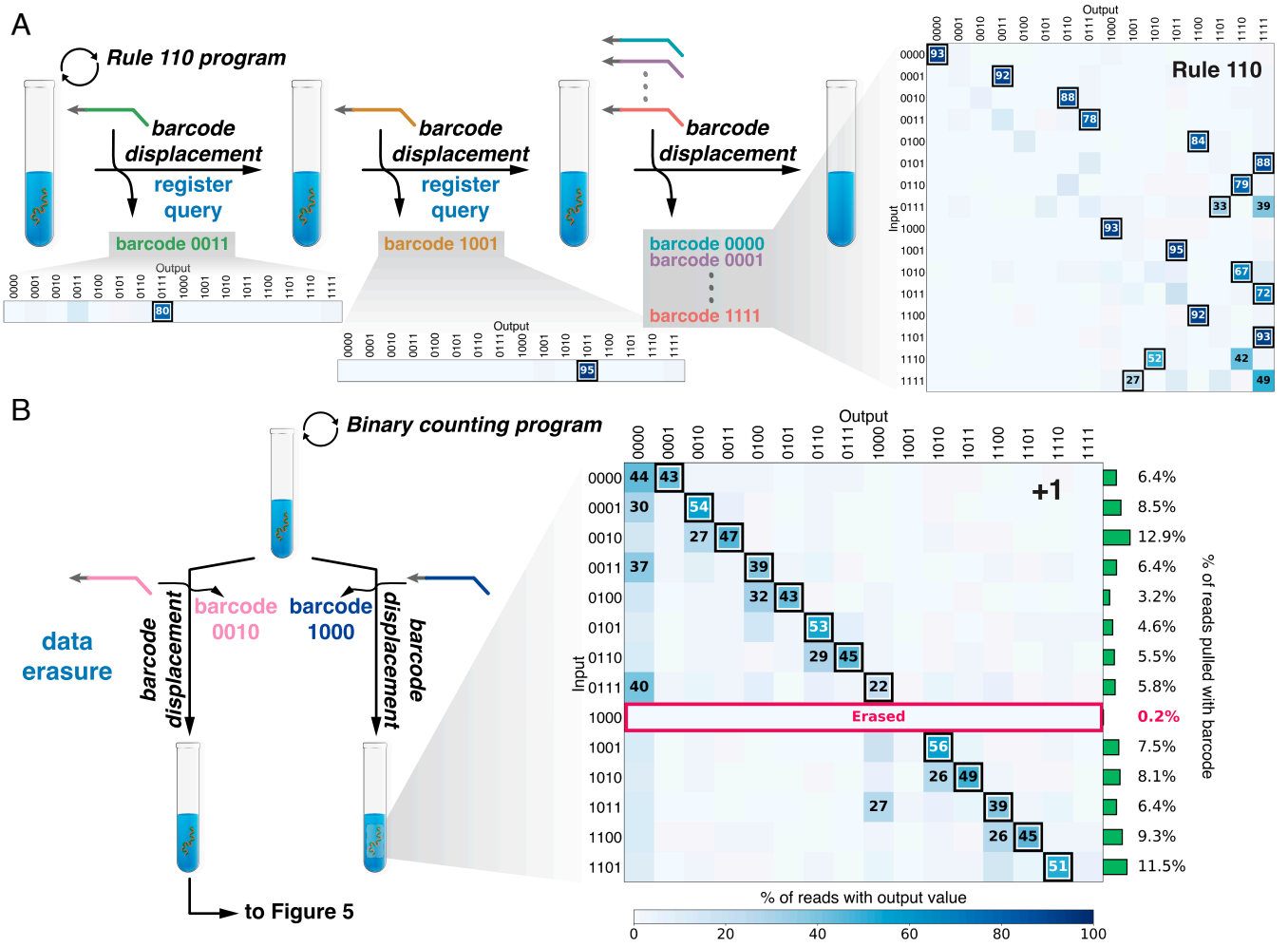

Fig. 4. NGS results for random access: Series of selective retrieval operations following Rule 110 computation and selective erasure after the binary counting computation. (*A*) Registers with initial value 0011 were accessed first (*Left*), 1001 second (*Middle*), and all remaining values last (*Right*) by adding the query strands corresponding to their barcodes. The query strand concentration was lower than the estimated register concentration; thus, displacement achieved partial extraction of registers. Registers reacted with instruction strands at 25 °C. (*B*) The remaining registers after selective erasure are plotted. The query strand concentration was higher than the estimated register concentration; thus, displacement achieved full erasure of registers. The green bars indicate the fraction of the readout registers with the corresponding barcode. Registers reacted with Instruction 1 strands at 40 °C and all other instruction strands at 25 °C. For both (*A*) and (*B*), the registers were composed of chemically synthesized DNA.

barcode sequences can be added to elute registers with the matching barcodes (Fig. 3*A*). Thus, specific registers can be accessed separately for readout from the register mix.

We experimentally demonstrated SIMD computation combined with random access for both the Rule 110 and the binary counting programs (Fig. 4). To test random access, we assigned registers barcodes corresponding to their initial inputs. After Rule 110 computation on a mix of registers with all 16 inputs, we added a query strand with the barcode corresponding to input 0011 and processed the displaced registers (ligation, PCR amplification, sequencing). Next, we queried the remaining registers with the barcode corresponding to initial value 1001 in the same way. Finally, the remaining registers were queried with all 16 different barcodes. The sequencing results confirmed that, for the first and second queries, the desired registers accounted for 90% and 67% of the registers displaced from the mix (*SI Appendix*, Fig. S9). In random access combined with the binary counting computation (Fig. 4*B*), all queries showed high specificity for the chosen barcode (at least 77%, *SI Appendix*, Fig. S10).

Another feature of our random access scheme is the targeted deletion of selected data, which may be crucial for the secure management of sensitive information. In our running example of medical records, with the patients' ID encoded in the barcodes, the information about a particular set of patients can be specifically removed. This contrasts with prior methods of erasure in which the erased information remains intact but is marked as erased (e.g., ref. 6). We demonstrate selective erasure by adding excess query strands and reading out the values from the remaining mix. We observed that reads corresponding to the displaced registers were almost completely removed (Fig. 4*B* and also *SI Appendix*, Fig. S11).

Multiple Rounds of Computation. The promise of dynamic DNA storage in removing slow and expensive read-write cycles requires that the computation cycle can be applied multiple times. Toward this goal, we performed three rounds of Rule 110 computation on an input mix containing 5 initial values (Fig. 5*A*). In the first round of computation, for all inputs, at least 83% of the readout registers updated correctly. In the second and third rounds of computation, the correct percent decreased more for some register values (e.g., second column) than others. For the binary counting program, we took the input mix that underwent

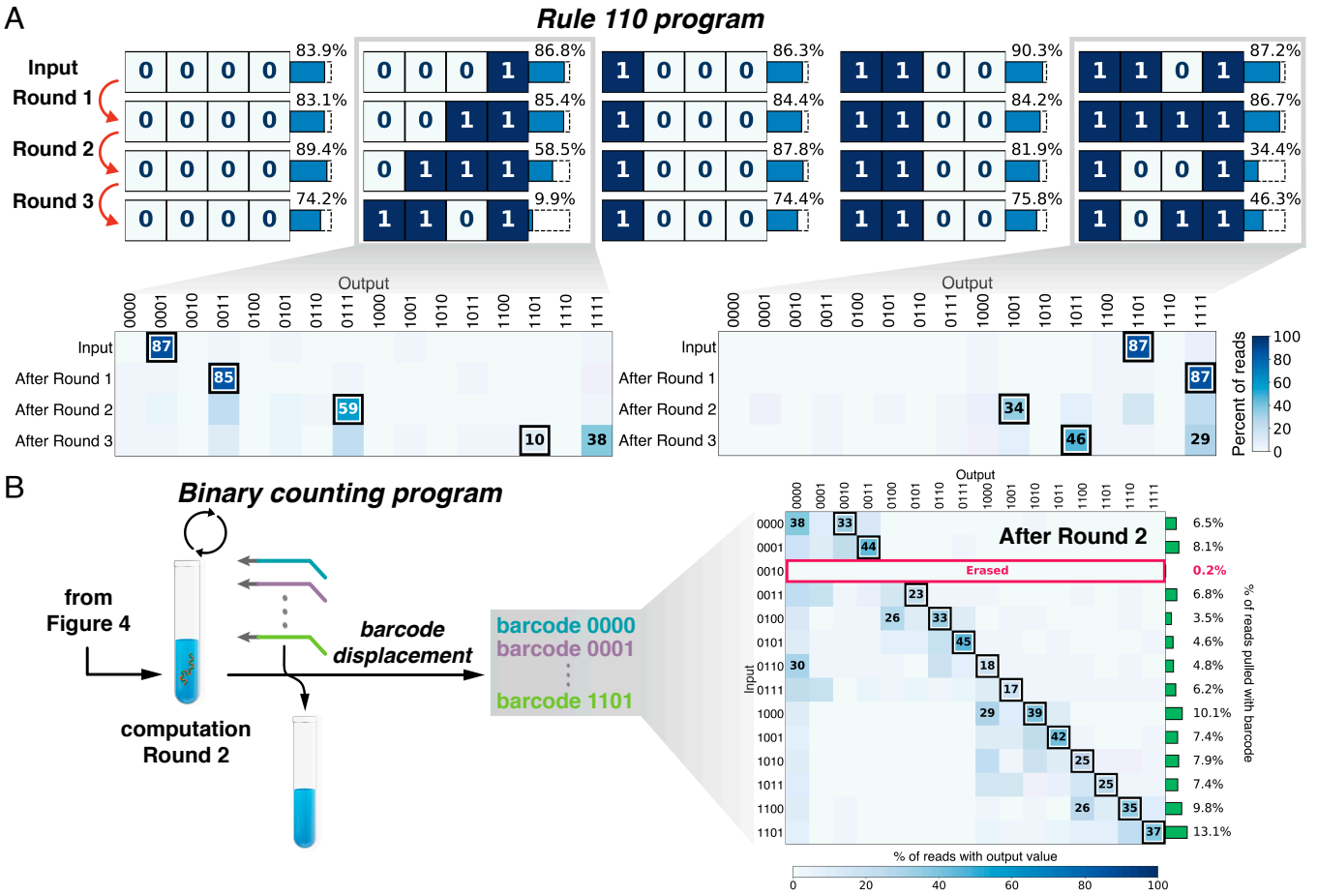

Fig. 5. Multiple rounds of SIMD computation. (*A*) Three rounds of Rule 110 computation were performed on an input mix with 5 distinct initial values (columns). *Lower* panels show the distribution of output values for initial values 0001 and 1101. Registers reacted with instruction strands at 25 °C. The reaction time for every instruction was 2 min except for Instruction 2, which was 10 min. (*B*) The mix of inputs that underwent selective erasure from Fig. 4*B* went through another round of binary counting computation. Registers reacted with Instruction 1 strands at 40 °C, and all other instruction strands at 25 °C. For (*A*) and (*B*), the registers were composed of chemically synthesized DNA.

one round of computation and selective erasure (from Fig. 4*B*) and performed another round of computation (Fig. 5*B*). The correct value was present in at least 17% of all readout registers. Similar results were achieved for a register mix where a different barcode was erased (*SI Appendix*, Fig. S12). These experiments demonstrate the feasibility of not only multiple computational rounds but the compatibility of selective retrieval, erasure, and computation in the same register pool.

Discussion

SIMD||DNA bridges the fields of DNA computing and DNA storage by allowing computation directly on the storage medium, potentially allowing massively parallel processing on large-scale databases. Since the first experiment using DNA to solve instances of NP-complete problems (38) in the 1990s, there have been more than 20 y of study of parallel DNA computing machines. Many models rely on enzymes to introduce covalent modification on DNA (39–42), and they are incompatible with current DNA storage paradigms. Other enzyme-free models that are compatible with information storage have limits and implementation challenges. For example, the sticker model (43) encodes information in the pattern of exposed domains (similar to SIMD||DNA). However, these domains require well-tuned binding affinities to allow a melting procedure which selectively dissociates some strands but not others—which could account for the lack of experimental implementation. In contrast, instead of computation through controlled hybridization and melting, strand displacement is one of the most versatile mechanisms to modify chemically encoded information (16–23). In a sense, we realize an extension of the sticker model envisioned 20 y ago (44): "Recent research suggests that DNA 'strand invasion' might provide a means for the specific removal of stickers from library strands. This could give rise to library strands that act as very powerful read-write memories." Other theoretical work showed that strand displacement can, in principle, perform Turing universal computation on information-storing polymers (45). However, this method cannot be parallelized and assumes that there is exactly one copy of certain species requiring a single-molecule implementation.

While the SIMD||DNA architecture was inspired by elements of the above theoretical proposals, it achieves experimental feasibility. We implemented two algorithms (binary counting and Rule 110) and demonstrated random access, erasure, and multiple rounds of computation. Binary counting is a fundamental function in computer programming; in our running example of medical records storing patient characteristics, disease history, and treatment plans, this operation could be used to track events. Rule 110 demonstrates that, in principle, our paradigm is capable of executing arbitrary algorithms since any algorithm can be

"compiled" to Rule 110 (34). Although arbitrary algorithms could be executed via Rule 110, the simulation is indirect and memory inefficient (*SI Appendix*, section S2.7), justifying the development of algorithms specifically for our architecture. After the binary counting and Rule 110 programs shown in the conference version of this work (25), additional algorithms were developed for bit shifting and parallel bubble sorting (46) (fundamental algorithms in computer science). Avenues other than Rule 110 for executing arbitrary algorithms were also explored (47).

Algorithms can be recast to SIMD||DNA in many distinct ways; this diversity motivates the investigation of input encoding, running time, space usage, parallelism, and randomization (*SI Appendix*, section S2). All SIMD||DNA programs work in parallel over different registers, and some have an additional level of parallelism. For example, in our Rule 110 program, every cell within a register updates concurrently. This contrasts with binary counting where Instruction 1 requires a cascade of strand displacement reactions across multiple cells. Parallelism across registers may lead to improved scalability of SIMD||DNA programs by decreasing the duration of instruction steps as well as the overall number of instructions. Note that our two programs have different data encodings (differing nick patterns encoding 0 and 1). Encodings that are compatible across multiple programs or programs for translating between encodings would enable applications that require composing different programs (*SI Appendix*, section S2.8).

Crucially for scalability, SIMD||DNA "software" can be run on different underlying 'hardware' choices. We constructed registers and achieved strand displacement computation for both SIMD||DNA programs using either chemically synthesized or naturally occurring M13 DNA. Indeed, we show that large strand displacement systems—244 distinct displacement steps (*SI Appendix*, Table S3)—can operate on naturally occurring sequences without much sequence optimization—comparable to the most complex strand displacement systems using rationally designed sequences (20). Due to the key role that the M13 plasmid plays in structural DNA nanotechnology (30, 33), recent work has developed low-cost biological production of short single-stranded DNA with M13 subsequences (48). Thus, it may now be possible to biologically produce both registers and instruction strands at scale, something that would finally allow dynamic information storage with low cost.

Importantly, we demonstrated high-throughput readout of the dynamically manipulated information through NGS sequencing. In almost all cases, the correct outputs had the highest fraction of all sequencing reads. In those few instances where there was only a substantial, but not majority, count for the right answer, it is likely that further refinement will yield higher accuracy. The current design is not robust to undesired displacement reactions that initiate from fraying at nicks (leak), which may account for some of the errors. Reducing the duration of undesired reactions such as limiting concentrations and time may result in lower completion levels (as observed in *SI Appendix*, section S16). The recently published design principles for leak reduction (49, 50) or "software level" error correction codes/schemes may be incorporated to address the problem of error. Note that sequence-dependent correlation of errors hampers simple error correction codes that rely on error independence; the correlation could potentially be reduced by an additional level of redundancy offered by two or more register sequences. More hypotheses about attendant errors and potential for improvement are discussed in *SI Appendix*, section S17. (See *SI Appendix*, section S15 for additional forms of product loss not captured by read counts, including loss of registers due to washing and incomplete ligation. We believe that NGS substitution errors did not substantially contribute to overall error, as confirmed by the readout of registers without computation, $\approx 99\%$, *SI Appendix*, Fig. S4.)

Data storage in the nicking patterns does trade data density and stability for computability. Our current encodings in SIMD||DNA store data at a density of approximately 0.03 bit per nucleotide and 18 Pbytes per gram of DNA assuming 100 duplicate copies of each distinct register value (1) (*SI Appendix*, section S20), a decrease from traditional storage schemes that encode information in the DNA sequence itself (theoretical maximum data density of 2 bits per nucleotide). In principle, data density can be increased by using different encoding schemes, such as allowing overhangs on the Top strands to encode information. Additionally, compared to information stored in the DNA sequence, which could be stable for thousands of years (1), information stored in the pattern of nicks may be more prone to change since the patterns of nicks are more readily disrupted (e.g., via undesired 4-way branch migration between different registers, or due to high temperature). Indeed, the temperature instability of nicked strands could enable easy and complete erasure of information through heating (51). For applications requiring longevity, however, in addition to using specialized materials for the protection of DNA (52, 53), it is possible to covalently link the strands in a register with a highly efficient reversible photo-cross-linking reaction (54).

Several follow-up approaches could be used to further scale up SIMD||DNA. Recently developed Nanopore sequencing methods could potentially read information encoded in nicks and single-stranded gaps in double-stranded DNA directly in a high-throughput manner without PCR amplification (55). To increase both the yield and scale of computation, registers can be affixed to the surface of a microfluidic chip to achieve autonomous control of instruction strand addition and elution. Further, as discussed in *SI Appendix*, section S2.1, our programs in principle only require a small set of orthogonal domains (7 for Rule 110 and 8 for binary counting) independent of the length of the register. Reusing domains in this manner would also allow each instruction to be implemented with up to two distinct strands, potentially reducing chemical synthesis cost significantly.

SIMD||DNA ultimately adds a "wet" CPU to the opportunities inherent in DNA encoding for "cold" data storage. Such dynamic DNA storage could revolutionize the DNA storage architecture for applications that involve multiple computation cycles and parallel computation, especially since SIMD||DNA circumvents repeated sequencing and synthesis of oligonucleotides. The demonstrations that distinct strand displacement cascades can be read out in parallel, that long strand displacement cascades may not require extensive sequence design and can operate with naturally occurring sequences, and that barcoded registers can be separately processed is important when we begin to consider DNA data storage beyond purely archival purposes—for example, for the storage and updating of medical records for specific patients. In particular, more advanced querying could be used to select a population of patients satisfying certain criteria (such as receiving a combination of diagnostics or treatment plans), and such algorithmic queries could be practically realized via SIMD||DNA programs that control the accessibility of the toehold for the query strand as the output of the computation (*SI Appendix*, Fig. S17).

Materials and Methods

The theoretical proofs and open questions for the SIMD||DNA model are in *SI Appendix*. All DNA sequences are listed in *SI Appendix*. DNA oligonucleotides were purchased from Integrated DNA Technologies. The registers were annealed separately and then attached to magnetic beads. Computation was performed by adding instruction strands to a test tube containing registers with different initial values. After computation, the Top strands in each register were ligated, PCR amplified, and then sequenced. Detailed experimental *Materials and Methods* and *SI Appendix*, Figs. S1–S17 and *SI Appendix*, Tables S1–S3 are included in *SI Appendix*.

Data, Materials, and Software Availability. Raw sequencing data from Illumina analysis of computation products are available on NCBI SRA under SRP455481 (56) and SRP455485 (57). All study data are included in the article and/or *SI Appendix*. Sequence design code and NGS analysis code are available on GitHub (sequence design, https://github.com/boyawang-github/SIMDDNA (58); NGS data analysis, https://github.com/SiyuanSWang/simddna) (59).

ACKNOWLEDGMENTS. We thank Marc Riedel, Olgica Milenkovic, and Tonglin Chen for invaluable discussions on the theoretical model. We also thank Marc Riedel for suggesting the analogy to Single-Instruction, Multiple-Data Computers. We thank the Genomic Sequencing and Analysis Facility at UT-Austin for providing sequencing services. We thank Cheulhee Jung and Bingling Li for sharing experience of working with magnetic beads, and David Doty for discussion and suggestion of using ViennaRNA. B.W., C.C., and D.S. were supported by NSF grants CCF-1652824, CCF-2200290, DARPA grant W911NF-18-2-0032, and the Alfred P. Sloan Foundation. A.D.E. and S.S.W. were supported by the Welch Foundation grant F-1654, and S.S.W. received additional support from NSF grant DGE-1610403.

Author affiliations: [a]Electrical and Computer Engineering, University of Texas at Austin, Austin, TX 78712; [b]Department of Molecular Biosciences, University of Texas at Austin, Austin, TX 78712; and [c]Department of Chemistry and Biochemistry, University of Texas at Austin, Austin, TX 78712

Author contributions: B.W., C.C., and D.S. devised the project; B.W. and C.C. designed the programs and proved their correctness; B.W. performed initial fluorescence experiments, experiments for register preparation, computation, post-computation ligation and displacement; S.S.W. performed qPCR assays, post-computational sequencing library preparation, NGS and Sanger analysis; B.W. and S.S.W. prepared the figures; and D.S. and A.D.E. provided guidance throughout the project.

The authors declare no competing interest.

This article is a PNAS Direct Submission. D.S. is a guest editor invited by the Editorial Board.

1. G. M. Church, Y. Gao, S. Kosuri, Next-generation digital information storage in DNA. *Science* **337**, 1628 (2012).
2. L. Ceze, J. Nivala, K. Strauss, Molecular digital data storage using DNA. *Nat. Rev. Genet.* **20**, 456–466 (2019).
3. L. C. Meiser *et al*., Synthetic DNA applications in information technology. *Nat. Commun.* **13**, 1–13 (2022).
4. L. Organick *et al*., Random access in large-scale DNA data storage. *Nat. Biotechnol.* **36**, 242–248 (2018).
5. S. M. H. T. Yazdi *et al*., Random-access DNA-based storage system. *Sci. Rep.* **5**, 14138 (2015).
6. K. N. Lin, K. Volkel, J. M. Tuck, A. J. Keung, Dynamic and scalable DNA-based information storage. *Nat. Commun.* **11**, 1–12 (2020).
7. K. Chen, J. Zhu, F. Bošković, U. F. Keyser, Nanopore-based DNA hard drives for rewritable and secure data storage. *Nano Lett.* **20**, 3754–3760 (2020).
8. A. R. Chandrasekaran, O. Levchenko, D. S. Patel, M. MacIsaac, K. Halvorsen, Addressable configurations of DNA nanostructures for rewritable memory. *Nucleic Acids Res.* **45**, 11459–11465 (2017).
9. S. K. Tabatabaei *et al*., DNA punch cards for storing data on native DNA sequences via enzymatic nicking. *Nat. Commun.* **11** (2020).
10. C. Pan *et al*., Rewritable two-dimensional DNA-based data storage with machine learning reconstruction. *Nat. Commun.* **13**, 1–12 (2022).
11. C. Bee *et al*., Molecular-level similarity search brings computing to DNA data storage. *Nat. Commun.* **12**, 4764 (2021).
12. K. J. Tomek, K. Volkel, E. W. Indermaur, J. M. Tuck, A. J. Keung, Promiscuous molecules for smarter file operations in DNA-based data storage. *Nat. Commun.* **12**, 3518 (2021).
13. D. Y. Zhang, E. Winfree, Control of DNA strand displacement kinetics using toehold exchange. *J. Am. Chem. Soc.* **131**, 17303–17314 (2009).
14. J. SantaLucia, D. Hicks, The thermodynamics of DNA structural motifs. *Annu. Rev. Biophys. Biomol. Struct.* **33**, 415–440 (2004).
15. R. M. Dirks, J. S. Bois, J. M. Schaeffer, E. Winfree, N. A. Pierce, Thermodynamic analysis of interacting nucleic acid strands. *SIAM Rev.* **49**, 65–88 (2007).
16. G. Seelig, D. Soloveichik, D. Y. Zhang, E. Winfree, Enzyme-free nucleic acid logic circuits. *Science* **314**, 1585–1588 (2006).
17. L. Qian, E. Winfree, Scaling up digital circuit computation with DNA strand displacement cascades. *Science* **332**, 1196–1201 (2011).
18. G. Chatterjee, N. Dalchau, R. A. Muscat, A. Phillips, G. Seelig, A spatially localized architecture for fast and modular DNA computing. *Nat. Nanotechnol.* **12**, 920–927 (2017).
19. L. Qian, E. Winfree, J. Bruck, Neural network computation with DNA strand displacement cascades. *Nature* **475**, 368–372 (2011).
20. K. M. Cherry, L. Qian, Scaling up molecular pattern recognition with DNA-based winner-take-all neural networks. *Nature* **559**, 370–376 (2018).
21. Y. J. Chen *et al*., Programmable chemical controllers made from DNA. *Nat. Nanotechnol.* **8**, 755–762 (2013).
22. N. Srinivas, J. Parkin, G. Seelig, E. Winfree, D. Soloveichik, Enzyme-free nucleic acid dynamical systems. *Science* **358** (2017).
23. A. J. Thubagere *et al*., A cargo-sorting DNA robot. *Science* **357**, eaan6558 (2017).
24. M. J. Flynn, Some computer organizations and their effectiveness. *IEEE Trans. Comput.* **C–21**, 948–960 (1972).
25. B. Wang, C. Chalk, D. Soloveichik, "SIMD||DNA: Single instruction, multiple data computation with DNA strand displacement cascades" in *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, (LNCS, 2019), vol. 11648, pp. 219–235.
26. P. Craw, W. Balachandran, Isothermal nucleic acid amplification technologies for point-of-care diagnostics: A critical review. *Lab Chip* **12**, 2469–2486 (2012).
27. D. Y. Zhang, "Towards domain-based sequence design for DNA strand displacement reactions in DNA computing and molecular programming" in *DNA Computing and Molecular Programming*, Y. Sakakibara, Y. Mi, Eds. (Springer, Heidelberg, 2011), vol. 6518, pp. 162–175.
28. B. R. Wolfe, N. J. Porubsky, J. N. Zadeh, R. M. Dirks, N. A. Pierce, Constrained multistate sequence design for nucleic acid reaction pathway engineering. *J. Am. Chem. Soc.* **139**, 3134–3144 (2017).
29. C. G. Evans, E. Winfree, D. N. A. Sticky, "DNA sticky end design and assignment for robust algorithmic selfassembly" in *End Design and Assignment for Robust Algorithmic Self-assembly in DNA Computing and Molecular Programming*, D. Soloveichik, B. Yurke, Eds. (Springer International Publishing, Cham, 2013), vol. 8141, pp. 61–75.
30. S. Dey *et al*., DNA origami. *Nat. Rev. Methods Primers* **1**, 13 (2021).
31. D. Y. Zhang, Cooperative hybridization of oligonucleotides. *J. Am. Chem. Soc.* **133**, 1077–1086 (2011).
32. R. R. F. Machinek, T. E. Ouldridge, N. E. C. Haley, J. Bath, A. J. Turberfield, Programmable energy landscapes for kinetic control of DNA strand displacement. *Nat. Commun.* **5**, 5324 (2014).
33. P. W. Rothemund, Folding DNA to create nanoscale shapes and patterns. *Nature* **440**, 297–302 (2006).
34. M. Cook, Universality in elementary cellular automata. *Complex Syst.* **15**, 1–40 (2004).
35. S. Wolfram, Statistical mechanics of cellular automata. *Rev. Mod. Phys.* **55**, 601–644 (1983).
36. J. L. Banal *et al*., Random access DNA memory using Boolean search in an archival file storage system. *Nat. Mater.* **20**, 1272–1280 (2021).
37. D. Y. Zhang, S. X. Chen, P. Yin, Optimizing the specificity of nucleic acid hybridization. *Nat. Chem.* **4**, 208–214 (2012).
38. L. M. Adleman, Molecular computation of solutions to combinatorial problems. *Science* **266**, 1021–1024 (1994).
39. D. Beaver, "A universal molecular computer" in *DNA Based Computers, Proceedings of a DIMACS Workshop, Princeton, New Jersey, USA, April 4, 1995, DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, R. J. Lipton, E. B. Baum, Eds. (DIMACS/AMS, 1995), vol. 27, pp. 29–36.
40. D. Boneh, C. Dunworth, R. J. Lipton, J. Sgall, On the computational power of DNA. *Discrete Appl. Math.* **71**, 79–94 (1996).
41. R. Freund, DNA computing based on splicing: The existence of universal computers. *Theory Comput. Syst.* **32**, 69–112 (1999).
42. P. W. K. Rothemund, "A DNA and restriction enzyme implementation of Turing Machines" in *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* (1995), vol. 27, pp. 75–119.
43. S. Roweis *et al*., A sticker-based model for DNA computation. *J. Comput. Biol.* **5**, 615–629 (1998).
44. R. S. Braich, N. Chelyapov, C. Johnson, P. W. K. Rothemund, L. Adleman, Solution of a 20-variable 3-SAT problem on a DNA computer. *Science* **296**, 499–502 (2002).
45. L. Qian, D. Soloveichik, E. Winfree, "Efficient turing-universal computation with DNA polymers" in *Efficient Turing-Universal Computation with DNA Polymers in DNA Computing and Molecular Programming*, Y. Sakakibara, Y. Mi, Eds. (Springer, Heidelberg, 2011), vol. 6518, pp. 123–140.
46. T. Chen, A. Solanki, M. Riedel, "Parallel pairwise operations on data stored in DNA: Sorting, shifting, and searching" in *27th International Conference on DNA Computing and Molecular Programming (DNA 27)*, M. R. Lakin, P. Šulc, Eds. (Schloss Dagstuhl – Leibniz-Zentrum fur Informatik, Dagstuhl, Germany, 2021), vol. 205, pp. 11:1–11:21.
47. D. Doty, A. Ong, "Simulating 3-symbol Turing machines with SIMD||DNA" in *1st Symposium on Algorithmic Foundations of Dynamic Networks (SAND 2022), Leibniz International Proceedings in Informatics (LIPIcs)*, J. Aspnes, O. Michail, Eds. (Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2022), vol. 221, pp. 14:1–14:15.
48. F. Praetorius *et al*., Biotechnological mass production of DNA origami. *Nature* **552**, 84–87 (2017).

49. B. Wang, C. Thachuk, D. Soloveichik, Speed and Correctness Guarantees for Programmable Enthalpy-Neutral DNA Reactions. *ACS Synth. Biol.* **12**, 993–1006 (2023).
50. B. Wang, C. Thachuk, A. D. Ellington, E. Winfree, D. Soloveichik, Effective design principles for leakless strand displacement systems. *Proc. Natl. Acad. Sci. U.S.A.* **115**, E12182–E12191 (2018).
51. J. Kim, J. H. Bae, M. Baym, D. Y. Zhang, Metastable hybridization-based DNA information storage to allow rapid and permanent erasure. *Nat. Commun.* **11**, 5008 (2020).
52. R. N. Grass, R. Heckel, M. Puddu, D. Paunescu, W. J. Stark, Robust chemical preservation of digital information on DNA in silica with error-correcting codes. *Angew. Chem. Int. Ed.* **54**, 2552–2555 (2015).
53. N. V. Ivanova, M. L. Kuzmina, Protocols for dry DNA storage and shipment at room temperature. *Mol. Ecol. Res.* **13**, 890–898 (2013).
54. Y. Yoshimura, K. Fujimoto, Ultrafast reversible photo-cross-linking reaction: toward in situ DNA manipulation. *Org. Lett.* **10**, 3227–3230 (2008).
55. K. Liu *et al*., Detecting topological variations of DNA at single-molecule level. *Nat. Commun.* **10**, 3 (2019).
56. B. Wang, S. S. Wang, C. Chalk, A. D. Ellington, D. Soloveichik, Parallel molecular computation on digital data stored in DNA. NCBI Sequence Read Archive. https://trace.ncbi.nlm.nih.gov/Traces/?view=study&acc=SRP455481. Deposited 17 August 2023.
57. B. Wang, S. S. Wang, C. Chalk, A. D. Ellington, D. Soloveichik, Parallel molecular computation on digital data stored in DNA (synthetic sequences). NCBI Sequence Read Archive. https://trace.ncbi.nlm.nih.gov/Traces/?view=study&acc=SRP455485. Deposited 17 August 2023.
58. B. Wang, S. S. Wang, C. Chalk, A. D. Ellington, D. Soloveichik, Parallel molecular computation on digital data stored in DNA (Code for sequence design). https://github.com/boyawang-github/SIMDDNA. Deposited 16 August 2023.
59. B. Wang, S. S. Wang, C. Chalk, A. D. Ellington, D. Soloveichik, Parallel molecular computation on digital data stored in DNA (Code for NGS data analysis). https://github.com/SiyuanSWang/simddna. Deposited 16 August 2023.