Current Trends
in Science

PLATINUM JUBILEE SPECIAL

Transforming bases to bytes: Molecular computing with DNA

SURUCHI SHARMA, DHIRAJ BHATIA and YAMUNA KRISHNAN*

National Centre for Biological Sciences, Tata Institute of Fundamental Research, Bellary Road, Bangalore 560 065, India
** e-mail: yamuna.ncbs@gmail.com*

Despite the popular image of silicon-based computers for computation, an embryonic field of molecular computation is emerging, where molecules in solution perform computational operations. DNA, which is known to store biological information, is being used as a substrate for molecular computation. In this review, we describe why DNA is an ideal material for computing and follow the developments outlining the applicability of DNA to solve a few hard computational problems HPP, SAT and maximum clique problems. We also describe the use of DNA to build molecular logic gates that produce their output in binary logic and their further use for more complex computations such as half and full adders. We present an outlook highlighting the major opportunities and challenges for the field and, in our opinion, its likely future trajectory.

Computation refers to any type of information processing. In very simple terms, computation involves input of some information, processing of that input information according to a set of instructions, resulting in an output. Computers with silicon chips look very different from a biological cell. Yet the cell is also a computer – a cell and indeed groups of cells can take a decision based on a set of inputs. Every action of a cell to upregulate or downregulate a protein in response to a molecular trigger may be viewed as a computation. The specific molecular triggers could be considered inputs and the resultant molecular change, i.e., upregulation or downregulation can be considered the output. Thus the constituents required for computing are universal: the first is a method or place to store and read information and the second is a few defined operations to act on that information, and process it. In a living cell, DNA is the storehouse of all cellular information. A set of protein players perform a variety of operations on that DNA in accordance with a set of molecular triggers, to process the DNA in such a way that a specific information is released to the cell. Hence DNA is a natural choice for *in vitro* molecular computing, given that it can both store information and be a substrate on which a variety of operations may be performed. A simple chemical reaction may also be viewed as a computational operation, where the reactant is the input, the product is the output and the reaction itself is the operation. Thus, molecules can also be used for computation [1] and DNA is one such molecule. The main advantage of molecular computation is that the speed of information processing in the former is many-fold higher than that seen in silicon based computers. This is because, millions of molecules in solution can work simultaneously, i.e., in parallel to process the information into outputs [2].

DNA stores information in the form of its sequence of nucleotides A, T, G and C. Information strings here result in the association of different DNA sequences through Watson–Crick base pairing between complementary DNA strands. These DNA strands can then self assemble into architecture that can either act as molecular switches or into rigid architecture that can further self assemble in the presence of molecular

Keywords. Molecular computation; DNA computation; DNA binary logic.

cues. These scaffolds thus act as the basis of DNA computation. Various operations can be performed on these scaffolds, such as,

- changing the assembly conformation in the presence of various triggers like protons [3], small molecules [4] or other DNA strands [5],
- further self assembly of combinations of rigid architecture by base pairing into different superstructures [6],
- biochemical transformations of DNA fragments by specific proteins like restriction endonucleases [7].

There are various methods by which the output of these operations can be read. They include fluorescence to sense both biochemical transformations as well as molecular switching [3], microscopy methods like AFM to visualize the output superstructures, and analytical methods [6] like gel electrophoresis combined with biochemical methods using proteins (polymerases, ligases, nucleases) to read out the output sequence in case of sequence transformations [7].

1. DNA-based Turing machines

A Turing machine is a powerful computational and mathematical concept which is a basic symbol-manipulating device that can be adapted to simulate the logic of any computer algorithm. In its simplest form, it works by reading the symbols written on a string, performing a pre-designated operation at every symbol and producing an output also in the form of a different string. DNA also carries strings of information in the form of its nucleotide sequence. For example, a polymerase can process this information one base (symbol) at a time, according to an operation instruction that specifies: for A insert a T, for G insert a C, etc., which results in a new string. It is easy to see that transcription and translation machinery are nothing but natural cellular Turing machines. Thus Shapiro and colleagues described the operation of a much simpler biomolecular Turing machine constructed from DNA [8]. Their design consisted of a long DNA duplex, shorter DNA strands which are the inputs, and two enzymes which process in a predesignated manner, the resulting information, resulting in a new DNA sequence which is the output [9]. Such a finite molecular automaton could be designed to analyse the levels of single stranded RNA molecules in solution and in response, release a given ssDNA. This implies that such DNA-based Turing machines could find potential use in not just diagnosis but also for simultaneous anti-sense therapy to regulate key genes [8,10].

2. NP-complete problems computed with DNA

Computational problems are classified into those that are easy, hard and uncomputable [11]. Hard problems, such as NP complete problems, are relatively inaccessible for a Turing machine as the computation time required increases exponentially with the input size. However, this is addressable using DNA as, due to the massive parallelism inherent in molecular computation, the computation time increases only linearly with input size [12–14]. DNA has thus been used to solve NP-complete problems like the Hamiltonian path problem (HPP) [15–17], satisfyability problem (SAT) [18–22] and maximum clique problems [12]. There is also theoretical work to establish algorithms for DNA-based computation [23–25]. The use of DNA to solve NP-complete problems broadly involve two steps

- DNA library construction, i.e., making a collection of DNA sequences that represent all the possibilities for the given problem and
- a procedure to select the output DNA sequences that are the answer to the problem.

Thus, in a pioneering experiment, Adleman [15] established proof-of-principle that DNA could be used for computation by solving the traveling salesman problem which is a Hamiltonian path problem (HPP). In this problem, given a set of directed edges joining a set of vertices, what is the connectivity pattern between the vertices starting and ending at two distinct vertices such that the path traverses every vertex exactly once (see figure 1a). This problem was solved by making a library where:

- each vertex was represented by a DNA sequence 20 bases long
- the edge connecting any two vertices A and B was a DNA sequence 20 bases long, where half of this sequence was complementary to 3′ end of vertex A and the other half complementary to the 5′ end of vertex B (see figure 1b).

Thus the set of all possible edges was constructed. The library was made by mixing all the edge sequences with sequences complementary to those of all the vertices. The latter brings together two edge sequences at either terminus of each vertex-complementary sequence (say $\underline{B}$). The sequences resulting from the ligation of this mixture represents the set of all connectivity patterns to this problem. The patterns that fit the desired connectivity could be obtained by:

- PCR using sequences corresponding to any two given vertices,

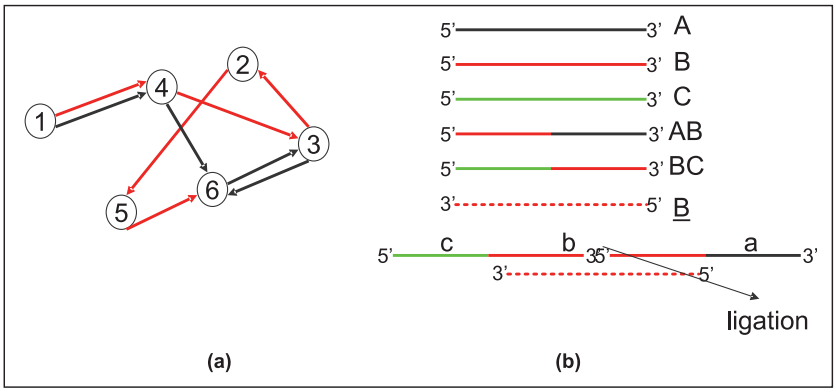

Figure 1. **(a)** Two kinds of paths starting from 1 and ending at 6. The path in red is a Hamiltonian Path while that in black is a Non Hamiltonian Path. **(b)** Schematic showing the design to solve a HPP problem. A, B and C are the sequences corresponding to the points A, B and C. AB or BC are the sequences corresponding to the edges connecting A and B or B and C, where complementary sequences are indicated by the same colour. B is the DNA sequence complementary to point B.

- gel purification to fix the number of vertices, and
- sequential selection against DNA coated magnetic beads corresponding to each of the six vertices to ensure that the solutions indeed had all six vertices,
- The solutions could then be read by simple DNA sequencing.

Another instance of solving an NP-complete problem using DNA is by Qi Ouyang *et al* [12] where they addressed the maximum clique problem. If there is a set of vertices (point) and edges connecting them, a clique is a subset of points such that each point in that subset is connected to all other points in that subset (see figure 2a). The maximum clique problem is to find the clique which has maximum number of points. First, all possible subsets of an 'n' point set are represented as an n-digit number in binary, where at a given position, 1 specifies that that point is present in the subset and 0 specifies the opposite, i.e., a subset with 8 points represented by the binary number 10011000 indicates that it has points 8, 5 and 4. Such a binary number is represented by a DNA duplex in which a 20 bp duplex specifies the point identity and this is followed by a 10 base pair sequence if the value at the position is 0 or a unique 6 bp restriction site if the value is 1. Thus a library which represents the collection of all possible points was constructed. The condition of a clique in general specifies that in a subset, the simultaneous presence of any two points without a connecting edge precludes it from being a clique. The above sequence design ensures that points x and y without a connecting edge cannot have simultaneous occurrence of restriction sites for enzymes Rx and Ry in any sequence corresponding to a clique. Thus such sequences must be eliminated from the library in order to arrive at the solution. This is achieved by using distinct restriction enzymes R1-RN corresponding to each of the unique 6 bp restriction sites representing the points 1 to n. The operation or processing step consists of the following:

(i) splitting the contents into two fractions,
(ii) treating each split fraction with a distinct restriction enzyme corresponding to a given pair of unconnected points and
(iii) mixing the two fractions and
(iv) repeating steps (i) to (iii) for all pairs of unconnected points (see figure 2b).

The end result is a set of fragments of different sizes corresponding to the set of all cliques. To read the output and find the clique which has maximum number of points or maximum number of 1 s, the sequences were size separated by gel electrophoresis; the smallest DNA sequences were isolated and sequenced to reveal the points present in the maximum clique.

A SAT problem is an NP-complete problem and is a more general method of solving NP-complete problems. Lipton [23] *et al* showed that the traveling salesman problem could be solved as a SAT problem instead of a HPP problem, thus enabling the solution of larger sizes of such problems by DNA computing. Following this, several instances of SAT problems were solved using DNA [18,19,21]. However in all of these solutions, since the logic was not encoded in the DNA, one had to eliminate those DNA sequences that were not solutions in order to access those that were part of the solution. Importantly, in work done by Kensaku [20], the sequences were designed such that all non-solution sequences were differentiated from solution sequences by hairpin-formation of the former. One of the issues with DNA computation is that, as the size of the problem increases, the quantity of DNA needed will also increase exponentially and

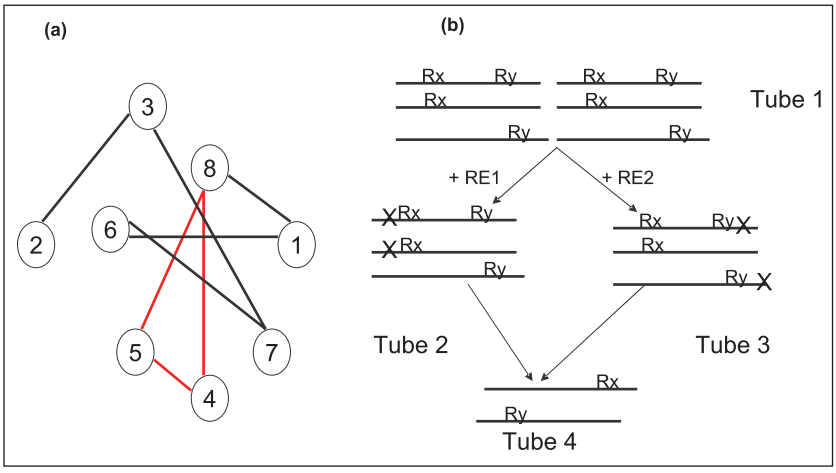

Figure 2. **(a)** Shown is a collection of points and edges connecting them. A subset (black) represented by 00000111 indicates that it contains points 1, 2 and 3 but is not a clique as 1 and 2 do not have a connecting edge. A subset (red) represented by 10011000 indicates that it contains points 8, 5 and 4 and is a clique. **(b)** Schematic showing a repetitive split – digest – mix sequence will separate all those sequences having both Rx and Ry yet retaining those sequences containing only one Rx or Ry at the end.

this is due to detection limits in order to read the solution. However, this has been addressed by applying single molecule detection of hybridization to solve a SAT problem [22].

3. Binary logic with DNA

When DNA is used to compute solutions to hard NP complete problems, the output is read using a sequencing paradigm. However, silicon-based computation is achieved using binary logic, where the input and output are in a binary format. Therefore there was also interest to explore DNA's ability to compute within the same paradigm of binary logic, where the output is also in a binary format. As a testament to DNA's universal applicability to computation, this was achieved in two ways (i) construction of logic gates using DNA and (ii) logical self-assembly of DNA into specific super-architecture.

(i) Logic gates using DNA: Small computational units which control the processing of information according to a set of operations are called logic gates. A logic gate senses one or more inputs and based on the processing information that they contain, produce an output. A biochemical reaction on DNA can also function as a logic gate, where the reactants are inputs, products are the output and the actual reaction is the operation. Here DNA assemblies contain fluorescent tags that are either activated or deactivated (the output) based on a trigger (the input) [26] because the underlying DNA scaffold undergoes a molecular transformation in response to the input logic. Breaker and Joyce found that certain DNA sequences, called DNAzymes, in the presence of divalent metal ions were able to cleave other DNA/RNA sequences [27]. A DNAzyme consists of two components – a catalytic core and an internal single stranded loop. When a single stranded DNA substrate binds to the internal loop of the DNAzyme, it is cleaved at a specific location by the DNAzyme in the presence of Mg^{2+}. Following this, Stojanovic and colleagues designed methods to use such DNAzymes as 'molecular logic gates' [28]. Since single stranded overhangs in a DNAzyme can bind to complementary strands and activate the DNAzyme, different sequences of ssDNA are considered as inputs, activation of the DNAzyme and cleavage of the target DNA is the operation and the properties of the cleaved products are the output. The most commonly used property of output measurement is fluorescence detection. Figure 3 shows the operation of such an AND gate based on a DNAzyme. This is a simple logic gate in which the output is 1 or YES only if both inputs are present. In the absence of at least one input, the output is 0 or NO. In a major advance, Stojanovic *et al* combined different logic gates to create a platform for multiple operations autonomously [29]. This set-up called a molecular automaton, performs all the downstream actions once the operation is triggered by Mg^{2+}. Since this automaton was made by a combination of YES and ANDANDNOT gates using an array of DNAzymes, it was called MAYA (Molecular Array of Yes and ANDANDNOT gates). Given that it performed all the operations depending on the input DNA strands added by an operator, it was also described as a tic-tac-toe device similar to the traditional tic-tac-toe which is played in 9 well structures. Stojanovic and colleagues have

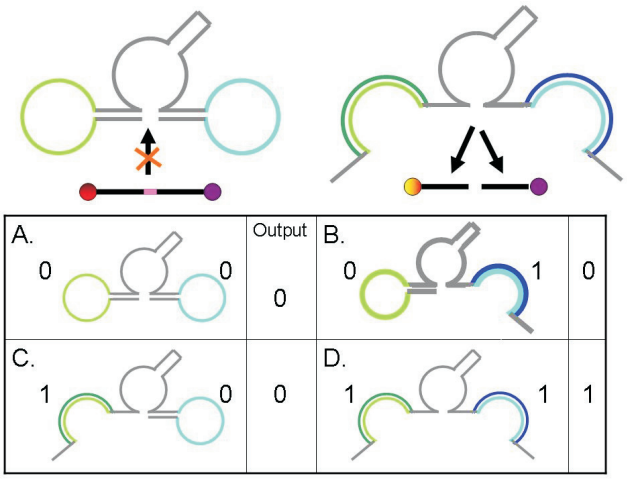

Figure 3. A representation of DNAzyme based 'AND gate'. AND gate has two inputs and the output is 1 only if both the inputs are 1. The closed stems deactivate the enzyme activity and it becomes active only if both the input strands bind to loops and open the stems of the enzyme. This happens only in 'D' shown below in truth table. (Adapted from J Macdonald *et al Sc. Am.* 2008, 85–91).

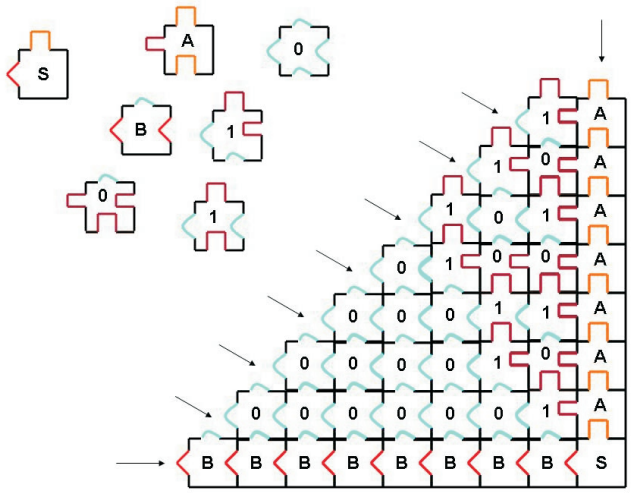

Figure 4. A set of 7 tiles can be arranged in a 2D array based on the complementarity and this forms a binary counter. S represents the seed tile. (Adapted from E Winfree, NAE's *The Bridge*, 2003, 33, 31–38).

also used other combinations of DNAzyme-based logic gates to create a half adder [30], full adder and logic gates with more than two inputs [31]. One could envisage combining these smart units to create molecular cascades or 'circuits' in future.

(ii) Logical assembly using DNA: The second approach to demonstrate DNA's capability of binary logic uses as its input, combinations of different DNA assemblies that can further self-assemble according to the input logic into different kinds of super-architecture that may be visualized using high-resolution microscopy [6]. Using base-pair recognition of complementary sequences, it is possible to form discrete DNA assemblies of pre-defined shapes or motifs that are programmed with single stranded overhangs. Combinations of these discrete DNA motifs can again self assemble via these overhangs into larger superstructures in 2D [32–34]. Such programmed assembly using DNA is called algorithmic self assembly. The theoretical principles of building large 2D architecture using discrete DNA motifs was outlined by Ned Seeman in 1982 [35]. Using DNA sequences that assembled into 3 way or 4 way junction motifs with programmable overhangs, these motifs were shown to be capable of further assembly into 2D lattices [32–34]. However, in a paradigm shift, this principle was exploited by Eric Winfree to demonstrate logical assembly of combinations of input motifs where the process of algorithmic self assembly resulted in a well-defined output architecture. This approach uses the assembly of DNA 'tiles' where a DNA-tile is a DNA architecture with a defined geometry possessing programmable single stranded overhangs. Winfree used previously characterized DNA-tiles [36] and programmed them for further assembly. Differently programmed tiles are considered as various inputs, and accordingly assemble into a specific super-architecture which can be read by high resolution AFM. The positional information of the input tiles on the super-architecture, according to binary logic encodes the output. However this is limited by high error-rates in tile assembly and a major challenge will be to design programmed overhangs that minimize self-assembly error rates. Figure 4 shows the assembly of a simple 2D superstructure from different types of DNA tiles. Rothemund and Winfree used four different kinds of DNA tiles to assemble 2D structures which resembled Sierpinski triangles [37]. Similarly, triple crossover junctions were used to create a 1D assembly that was used to perform cumulative XOR operation on binary units [38].

4. Outlook

With DNA computation we need not be limited to binary logic. It is possible to construct assemblies that have multiple inputs and multiple operational states. Thus it is possible to perform higher order logical computation. This is unlike silicon based computing that is restricted to binary logic resulting in the flow of current or no current. One of the advantages of computation based on higher order logic is that more information can be stored in the same memory space. Despite all this, it is unlikely that DNA can ever replace silicon-based computers. However, it definitely has a few niche

applications in biology. One of these is detection and diagnostics in various biological samples, reminiscent of small molecule binding of RNA aptamers [39]. Along these lines, Winfree and colleagues recently demonstrated that DNA based logic gates [40] could be used to detect microRNAs *in vitro*. One of the major future challenges is the coupling of different DNA logic gates to produce molecular automata and their use *in vitro* as well as *in vivo*. Thus the emergence of a parallel field of RNA logic gates is not surprising. Smolke and colleagues have designed RNA devices which can process information inside a cell [41]. They showed the coupling of RNA devices to ribozyme activity containing actuators in the UTRs of a few genes and based on ribozyme cleavage they could regulate gene expression at the RNA level. Si-RNA based automata are also being developed where the level of si-RNA may be considered the input and the alteration of gene expression, its output [42]. Taken together, DNA logic gates either on their own, or in tandem with RNA partners could find myriad applications not only in sensing and diagnosis, but also as *in vivo* and intracellular tools given their logical control of gene expression.

References and recommended reading

Papers of particular interest, published within the period of review, have been highlighted as:

- • of special interest
- •• of outstanding interest

[1] • De Silva A P and Uchiyama S 2007 Molecular logic and computing; *Nat. Nanotech.* **2** 399–410.
A nice review describing various aspects of molecular computing.

[2] Nowak R, Wasiewicz P, Plucienniczak A and Mulawka J J 2001 Processing DNA tokens in parallel computing; *Proc. Int. Parallel and Distributed Processing Symp.* 1307–1314.

[3] • Modi S, Swetha M G, Goswami D, Gupta G D, Mayor S and Krishnan Y 2009 A DNA nanomachine that measures spatial and temporal pH changes inside living cells; *Nat. Nanotech.* **4** 325–330.
The first demonstration of a functional DNA-based AND gate inside living cells.

[4] Mao C, Sun W, Shen Z and Seeman N C 1999 A nanomechanical device based on the B-Z transition of DNA; *Nature* **397** 144–146.

[5] Venkataraman S, Dirks R M, Rothenmund P W K, Winfree E and Pierce N 2007 An autonomous polymerization motor powered by DNA hybridization; *Nat. Nanotech.* **2** 490–494.

[6] Winfree E 2003 DNA computing by self assembly; *NAE's the bridge* **33** 31–38.

[7] Chen J and Seeman N C 1991 Synthesis from DNA of a molecule with the connectivity of a cube; *Nature* **350** 631–633.

[8] Shapiro E and Benenson Y 2006 Bringing DNA computers to life; *Sc. Am.* 45–51.

[9] Benenson Y, Adar R, Elizur T P, Livneh Z and Shapiro E 2003 DNA molecule provides a computing machine with both data and fuel; *Proc. Nat. Acad. Sci. USA* **100** 2191–2196.

[10] Benenson Y, Gill B, Dor U B, Adar R and Shapiro E 2004 An autonomous molecular computer for logical control of gene expression; *Nature* **429** 423–429.

[11] Casti J 1997 Computing the Uncomputable; *New Sci.* **2082** 34.

[12] Ouyang Q, Kaplan P D, Liu S and Libchaber A 1997 DNA solution of the maximal clique problem; *Science* **278** 446–449.

[13] • Adleman L M 1998 Computing with DNA; *Sc. Am.* 54–61.
A review written for the informed lay-person and ideal for beginners.

[14] Lipton R J 1995 DNA solution of hard computational problems; *Science* **268** 542–545.

[15] •• Adleman L M 1994 Molecular computation of solution to combinatorial problems; *Science* **266** 1021–1024.
The first proof of principle that DNA could be used to perform computations *in vitro*.

[16] Lee C M, Kim S W, Kim S M and Sohn U 1999 DNA computing the Hamiltonian path problem; *Mol. Cell.* **9** 464–469.

[17] Perez I M M, Zhang G, Ignatova Z and Zimmermann K H 2005 Biomolecular autonomous solution of the Hamiltonian path problem via hairpin formation; *Int. J. Bioinform. Res. Appl.* **1** 389–398.

[18] Liu Q, Wang L, Frutos A G, Condon A E, Corn R M and Smith L M 2000 DNA computing on surfaces; *Nature* **403** 175–179.

[19] Faulhammer D, Cukras A R, Lipton R J and Landweber L F 2000 Molecular computation: RNA solutions to chess problems; *Proc. Nat. Acad. Sci. USA* **96** 1385–1389.

[20] •• Sakamoto K, Gouzu H, Komiya K, Kiga D, Yokoyama S, Yokomori T and Hagiya M 2000 Molecular computation by DNA hairpin formation; *Science* **288** 1223–1226.
In this work logic was encoded in the sequence.

[21] • Braich R S, Chelyapov N, Johnson C, Rothemund P W K and Adleman L 2002 Solution of a 20-Variable 3-SAT problem on a DNA computer; *Science* **296** 499–502.
The specialty of this work is that the size of the problem that has been computed is much larger than any other reports.

[22] Schmidt K A, Henkel C V, Rozenberg G and Spaink H P 2004 DNA computing using single molecule hybridization detection; *Nucleic Acids Res.* **32** 4962–4968.

[23] • Lipton R J 1994 *Speeding up computations by molecular biology* (unpublished).
Clarifies how fragmenting a problem into a simpler form also simplifies computation.

[24] Chang W L, Ho M and Guo M 2004 Fast parallel molecular algorithms for DNA-based computation: Factoring integers; *Proc. Fourth IEEE Symp. on Bioinformatics and Bioeng. (BIBE'04)*, 125–133.

[25] • Roweis S, Winfree E, Burgoyne R, Chelyapov N V, Goodman M F, Rothemund P W K and Adleman L 1998 A sticker based model for DNA computation; *J. Comput. Biol.* **5** 615–629.
Gives a nice insight into DNA based computation.

[26] • Macdonald J, Stefanovic D and Stojanovic M N 2008 DNA computers for work and play; *Sci. Am.* 85–91.
An excellent review outlining the applications of DNAzymes as logic gates.
[27] Breaker R R and Joyce G F 1994 A DNA enzyme that cleaves RNA; *Chem. Biol.* **1** 223–229.
[28] Stojanovic M N, Mitchell T E and Stefanovic D 2002 Deoxy-ribozyme based logic gate; *J. Am. Chem. Soc.* **124** 3555–3561.
[29] Stojanovic M N and Stefanovic D 2003 A deoxyribozyme – based molecular automaton; *Nat. Biotech.* **21** 1069–1074.
[30] Stojanovic M N and Stefanovic D 2003 Deoxyribozyme based half adder; *J. Am. Chem. Soc.* **125** 6673–6676.
[31] Lederman H, MacDonald J, Stefanovic D and Stojanovic M N 2006 Deoxyribozyme-based three input logic gates and construction of a molecular full adder; *J. Am. Chem. Soc.* **45** 1194–1199.
[32] Mao C, Sun W and Seeman N C 1999 Designed two dimensional DNA Holliday junction arrays visualized by atomic force microscopy; *J. Am. Chem. Soc.* **121** 5437–5443.
[33] Lui Y, Ke Y and Yan H 2005 Self assembly of finite size DNA nanoarrays; *J. Am. Chem. Soc.* **127** 17140–17141.
[34] Ding B, Sha R and Seeman N C 2004 Pseudohexagonal 2D DNA crystals from double crossover cohesion; *J. Am. Chem. Soc.* **126** 10230–10231.
[35] Seeman N C 1982 Nucleic-acid junctions and lattices; *J. Theor. Biol.* **99** 237–242.
[36] Sa-Ardyen P, Vologodskii A V and Seeman N C 2003 The flexibility of DNA double cross over molecules; *Biophys. J.* **84** 3829–3827.
[37] Rothenmund P W K, Papadakis N and Winfree E 2004 Algorithmic self-assembly of DNA seirpinski triangles; *PLoS Biol.* **2** 2041–2054.
[38] Mao C, LaBean T H, Reif J H and Seeman N C 2000 Logical computation using algorithmic self assembly of DNA triple-crossover molecules; *Nature* **407** 493–496.
[39] Barrick J E and Breaker R R 2007 The power of riboswitches; *Sci. Am.* **296** 50–57.
[40] Seelig G, Soloveichik D, Zhang D Y and Winfree E 2006 Enzyme-free nucleic acid logic circuits; *Science* **314** 1585–1588.
[41] • Win M N and Smolke C D 2008 Higher-order cellular information processing with synthetic RNA devices; *Science* **322** 456–460.
Challenging problem successfully demonstrated. First demonstration that RNA logic is functional inside living cells to control gene expression.
[42] Rinaudo K, Bleris L, Maddamsetti R, Subramanian S, Weiss R and Benenson Y 2007 A universal RNAi-based logic evaluator that operates in mammalian cells; *Nat. Biotech.* **25** 795–801.