ELSEVIER

Future Generation Computer Systems 17 (2001) 361–378

FGCS
Future Generation Computer Systems

www.elsevier.nl/locate/future

DNA computing: implementation of data flow logical operations

Piotr Wąsiewicz [a,*], Artur Malinowski [b], Robert Nowak [b], Jan J. Mulawka [a], Piotr Borsuk [c], Piotr Węgleński [c], Andrzej Płucienniczak [d]

[a] *Institute of Electronic Systems, Warsaw University of Technology, Nowowiejska 15/19, 00-665 Warsaw, Poland*
[b] *Institute of Computer Science, Warsaw University of Technology, Nowowiejska 15/19, 00-665 Warsaw, Poland*
[c] *Institute of Genetics, University of Warsaw, Pawińskiego 5A, 02-106 Warsaw, Poland*
[d] *Institute of Biotechnology and Antibiotics, Starościńska 5, 02-516 Warsaw, Poland*

Abstract

Self-assembly of DNA is considered a fundamental operation in realization of molecular logic circuits. We propose a new approach to implementation of data flow logical operations based on manipulating DNA strands. In our method the logic gates, input, and output signals are represented by DNA molecules. Each logical operation is carried out as soon as the operands are ready. This technique employs standard operations of genetic engineering including radioactive labeling as well as digestion by the second class restriction nuclease and polymerase chain reaction (PCR). To check practical utility of the method a series of genetic engineering experiments have been performed. The obtained information confirms interesting properties of the DNA-based molecular data flow logic gates. Some experimental results demonstrating implementation of a single logic NAND gate and only in one vessel calculation of a tree-like Boolean function with the help of the PCR are provided. These techniques may be utilized in massively parallel computers and on DNA chips. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: DNA computing; Molecular logic gates; Data flow computers; Boolean functions; One vessel PCR solution; Class 2 restriction nuclease; FokI; DNA chips

1. Introduction

Digital logic circuits create the base of computer architecture [1,2,28]. In constructing computer hardware different types of logic gates, e.g., OR, AND, XOR, NAND are utilized. In conventional von Neumann machines, a program counter is used to perform in sequence the execution of instructions. These machines are based on a control-flow mechanism by which the order of program execution is explicitly stated in the user program.

In a data flow machine the flow of the program is not determined sequentially, but rather each operation is carried out as soon as the operands are ready. This data-driven mechanism allows the execution of any instruction depending on data availability. This permits a high degree of parallelism at the instruction level and can be used in recent superscalar microprocessor architectures [31]. However, in the field of parallel computing, data flow techniques are still in the research stage and industry has not yet adopted these techniques [3]. One of the approaches suitable for massively parallel operations seems to be DNA-based molecular computing [4], of course with the help of DNA chips. These high density oligonu-

* Corresponding author.
E-mail addresses: pwas@ise.pw.edu.pl (P. Wąsiewicz), amalinow@ii.pw.edu.pl (A. Malinowski), rnowak2@ii.pw.edu.pl (R. Nowak), jml@ise.pw.edu.pl (J.J. Mulawka), wegle@ibb.waw.pl (P. Węgleński), apl@iba.waw.pl (A. Płucienniczak).

cleotide DNA arrays were developed as tools for sequencing by hybridization (SBH) [22,24]. Now it is possible to design and synthesize in situ on the support using light-directed solid phase combinatorial chemistry [23,25] square inch high-density oligonucleotides arrays for monitoring the expression levels of nearly all (about 6500) yeast genes equaling no more than 14 MB of information [17].

The primary objective of this contribution is to show striking adequacy of molecular computing for data flow techniques. Logical operations considered here are based on self-assembly of DNA strands [6–8,18]. By self-assembly operation we understand putting together fragments of DNA during the process of hybridization [5]. We present a new approach to implementation of logical operations suitable for data flow architectures and report first experiments with the second class enzyme and calculation of the Boolean function with PCR in one vessel.

2. Computing properties of DNA molecules

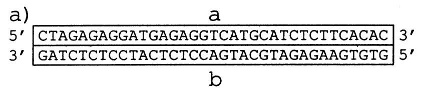
A number of papers have appeared on general concept of molecular computers [9–12]. In such an approach computations are performed on molecular level and different chemical compounds can be utilized [32]. Deoxyribonucleic acid (DNA) is usually used in molecular computing because this compound serves as a carrier of information in living matter and is easy to manipulate in genetic engineering laboratory.

The DNA molecule is composed of definite sequence of nucleotides. There are four different nucleotides in such molecule depending on purine and pyrimidine bases present in the given nucleotides, which we denote by the letter: A, T, C, G (adenine, thymine, cytosine, guanine). Thus, the DNA molecule can be considered as a polynucleotide or a strand or a string of letters. Since the strands may be very long, they can carry an immense number of bits. Therefore, particular molecules may serve as information carriers. Another interesting property of DNA is that particular bases may connect with each other together forming pairs: A with T and C with G [33]. This process known as hybridization or annealing causes self-assembly of DNA fragments. It occurs when two DNA strands are composed of matching (complementary) nucleotide sequences. Due to favored intermolecular interactions particular molecules can recognize each other. As a result a kind of key–lock decoding of information is possible. The self-assembly property may be utilized to implement the memory of an associate type [27,29]. According to Baum [13] this memory may be of greater capacity than that of human brain. The other important feature of self-assembling is that particular molecules may be considered as processing units performing some computing [30,31]. The following techniques of the genetic engineering will be utilized in our approach: synthesis of DNA strands, labeling, hybridization, analysis by DNA electrophoresis, digestion of double strands by restriction enzymes, synthesis of DNA strands by polymerase chain reaction (PCR) [14,15].

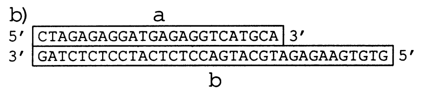
Examples given below show our DNA notation created in order to simplify descriptions of our experiments. One letter marks one strand as is seen in Fig. 1a. In our systems the upper strand usually presents data and lower (underlined) — logic devices, rules and so on. Therefore, they cannot be exchanged. Symbols $\lceil$ and $\rfloor$ denote position of single strands in space. A corner of such symbol represents 5' end, while an open edge represents 3' end. An underlined one is always in the same position (3' on the left, 5' on the right) even in other double-stranded oligos.

In Fig. 1b a sign $+$ at the right side of b describes a sticky end of b shorter than the nearest complementary strand a. In Fig. 1c a sign $-$ at the right side of b describes a sticky end of b longer than the nearest complementary strand a. In Fig.1d the same signs $-$ at the same sides of complementary strands a and b mean that these strands form a double strand with blunt ends. As is seen signs can be omitted or exchange by the pair of $+$. If complementary strands have the same letter, e.g., a, then in order to distinguish them, a sign tilde ($\tilde{}$) is added to the one.

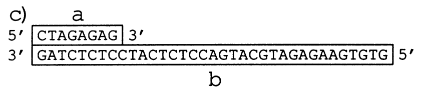
In Figs. 2–6 examples of notation in different cases are explained. Note that the sign $+$ may be additionally applied to mark a symbolic disjunction between two ligated strands and the sign $-$ to lengthen a strand &, of course only in the equations, not in real experiments. PCR may be used to lengthen an oligo by a length of a sticky end of a primer. A DNA oligo has its length. In our experiments its length will be written in an upper right index of an oligo letter, e.g., a^{345} and ligated DNA strands with letters in brackets (and)


a)
a
5′ CTAGAGAGGATGAGAGGTCATGCATCTCTTCACAC 3′
3′ GATCTCTCCTACTCTCCAGTACGTAGAGAAGTGTG 5′
b

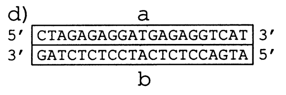
$$=\left[\begin{array}{c} a \\ \underline{b} \end{array}\right]$$


b)
a
5′ CTAGAGAGGATGAGAGGTCATGCA 3′
3′ GATCTCTCCTACTCTCCAGTACGTAGAGAAGTGTG 5′
b

$$=\left[\begin{array}{cc} a_1 & \\ \underline{b} & \underline{+} \end{array}\right]$$



$$=\left[\begin{array}{cc} a_2 & \\ \underline{b} & \underline{-} \end{array}\right]$$



$$=\left[\begin{array}{c} a_3 \\ \underline{b_1} \end{array}\right]=\left[\begin{array}{ccc} - & a_3 & - \\ \underline{-} & \underline{b_1} & \underline{-} \end{array}\right]=\left[\begin{array}{c} a_3 \\ \underline{\tilde{a_3}} \end{array}\right]$$

Fig. 1. Analytical representation of DNA strands.

$$\left[\begin{array}{ccccccccc} a & + & b & + & c & + & d & + & e \\ \underline{-} & \underline{-} & \underline{-} & \underline{-} & \underline{\&} & \underline{-} & \underline{-} & \underline{-} & \underline{-} \end{array}\right]$$

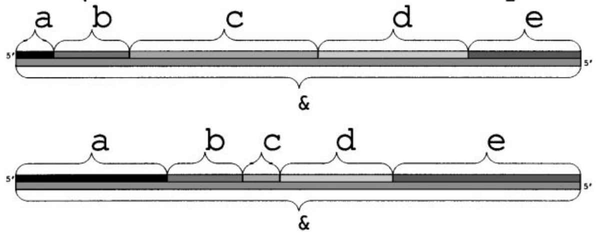


Fig. 2. The exemplary double strands described in the above equation.

$$\left[\begin{array}{cccccccccc} & a & + & b & + & c & + & d & + & e \\ \underline{-} & \underline{-} & \underline{-} & \underline{-} & \underline{-} & \underline{\&} & \underline{-} & \underline{-} & \underline{-} & \underline{+} \end{array}\right]$$

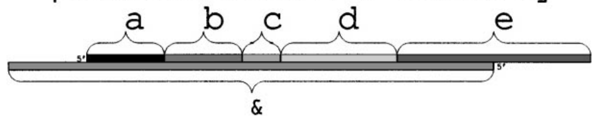


Fig. 3. The exemplary double strands described in the above equation.

$$\left[\begin{array}{ccc} & a & + \\ \underline{+} & \underline{-} & \underline{-} \end{array}\left[\begin{array}{ccccc} b & + & c & + & d \\ \underline{-} & \underline{-} & \underline{\&} & \underline{-} & \underline{-} \end{array}\right]^{\$}\begin{array}{cc} + & e \\ \underline{-} & \underline{-} \end{array}\right]$$

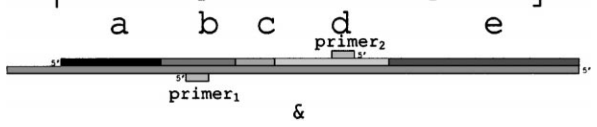


Fig. 4. The exemplary double strands described in the above equation.

or | and |, e.g., $(bcd)^{123} = |bcd|^{123}$, $|ab|^{67} = (ab)^{67}$. In Figs. 4–6 PCR process description is presented. 5’ ends of both primers are like [and] brackets and an unknown (or not so important to be written) number of PCR cycles is defined by a symbol \$. In Fig. 5 both 5’ ends of primers terminates exactly above 3’ ends of strands a and d. Thus, brackets [and] are just between strand letters. Now the number of amplification cycles is equal to 35. After PCR reaction in a vessel there are millions of amplified oligos and very small amounts of others. Therefore, a sign $\simeq$ is also utilized in Fig. 5.

$$\left[\begin{array}{cc} & a \\ \underline{+} & \underline{-} \end{array}\left[\begin{array}{ccccc} b & + & c & + & d \\ \underline{-} & \underline{-} & \underline{\&} & \underline{-} & \underline{-} \end{array}\right]^{\$}\begin{array}{c} e \\ \underline{-} \end{array}\right]\simeq\left[\begin{array}{ccccc} b & + & c & + & d \\ \underline{-} & \underline{-} & \underline{\&} & \underline{-} & \underline{-} \end{array}\right]$$

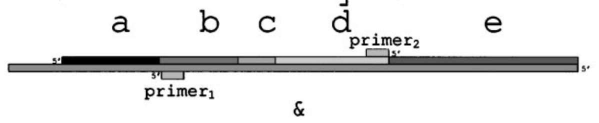


Fig. 5. The exemplary double strands described in the above equation.

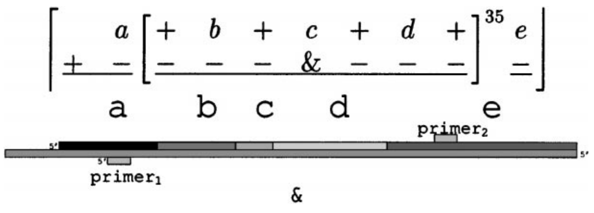


Fig. 6. The exemplary double strands described in the above equation.

3. Data flow parallel processing units

In a data flow machine [16], data availability rather than a program counter is used to drive the execution of instructions. It means that every instruction that has all of its operands available can be executed in parallel. Such an evaluation of instructions permits the instructions to be unordered in a data-driven program [3]. Instead of being stored in a shared memory, data are directly hold inside instructions. Computational results are passed directly between instructions via data tokens. In a data flow processor the stored program is represented as a directed graph. Such the graph may be reduced by evaluation of branches or subgraphs. Different parts of a graph or subgraph can be reduced or evaluated in parallel upon demand. Nodes labeled with particular operations calculate a result whenever the inputs are valid, and pass that result on to other nodes as soon as it is valid which is known as firing a node. To illustrate this operation some process of computation is described in Fig. 7. As is seen a node representing a performed process has two incoming branches, which represent inputs. The flow of data is shown in these pictures by introducing data tokens denoted by black dots. In Fig. 7a there is a moment when data on a single input has appeared and the processor is waiting for a second data token while in Fig. 7b

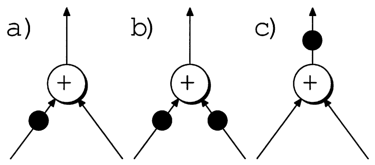


Fig. 7. Successive phases of firing a node: (a) waiting for a second data token, (b) ready to execute an operation, and (c) after firing a new token is generated.

the processor is ready to execute its function and finally in Fig. 7c there is a situation after firing the node. The data flow graphs usually are more complicated and there are different types of operations applied.

Generally, molecular logic gates may be utilized in constructing computer architecture based on data flow graphs. Such graphs consisting of DNA strands can exchange tokens as DNA single strings representing gate inputs and outputs. Some sectors of strands can transport values of variables or special constants. In such a solution a number of gates–nodes (processing units) and a number of data and control tokens may be theoretically unlimitable. In Section 4 we demonstrate how to realize data flow logical processing units using DNA molecules.

4. A new concept of data flow logic gates

In classical computer any logic gate is an electronic circuit that has one or more inputs and one output. In such a circuit the electrical condition of the output at any time is dependent on those of the inputs at that time [1,2,28]. An alternative approach to implement logic operations may be performed on molecular level by self-assembly of DNA strands. In such methods DNA molecules as is depicted in Fig. 8, respectively, may represent the logic gates, input and output signals.

Assume that strands representing gates consist of three sectors belonging to the following groups: $\underline{\tilde{X}} = \{\underline{\tilde{x}}_1, \underline{\tilde{x}}_2, \underline{\tilde{x}}_3, \dots, \underline{\tilde{x}}_N\}$, $\underline{\tilde{Z}} = \{\underline{\tilde{z}}_1, \underline{\tilde{z}}_2, \underline{\tilde{z}}_3, \dots, \underline{\tilde{z}}_N\}$, $\underline{\tilde{Y}} = \{\underline{\tilde{y}}_1, \underline{\tilde{y}}_2, \underline{\tilde{y}}_3, \dots, \underline{\tilde{y}}_N\}$. Representing inputs strands are composed of two sectors from groups: $X = x_1, x_2, x_3, \dots, x_N$, $Y = y_1, y_2, y_3, \dots, y_N$. Representing outputs strands consist of three sectors belonging to groups: $X = x_1, x_2, x_3, \dots, x_N$, $Z = z_1, z_2, z_3, \dots, z_N$, $Y = y_1, y_2, y_3, \dots, y_N$. The composition of DNA strands from Fig. 8 are depicted in Fig. 9. To enable hybridization process, input and output signal sectors from the groups X, Z, Y are complementary to adequate sectors in logic gates strands: $\underline{\tilde{X}}$, $\underline{\tilde{Z}}$, $\underline{\tilde{Y}}$. This means that two DNA strands can be connected together with the two sequences in the following way: x_i with $\underline{\tilde{x}}_i$ or z_i with $\underline{\tilde{z}}_i$ or y_i with $\underline{\tilde{y}}_i$. There must be one sequence: x_i or z_i or y_i in the first string and another complementary sequence: $\underline{\tilde{x}}_i$ or $\underline{\tilde{z}}_i$ or $\underline{\tilde{y}}_i$ in the second string.

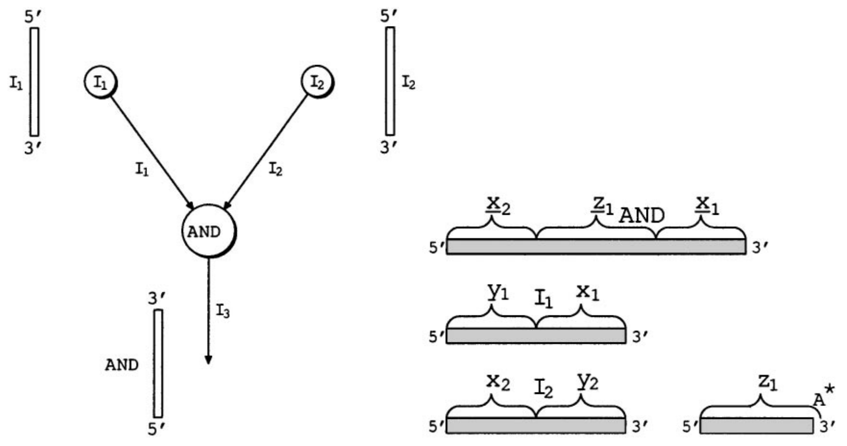


Fig. 8. Data flow molecular logic gate AND: a scheme of the gate, oligonucleotides representing the AND gate, input signals I_1, I_2 and a part of an output I_3.

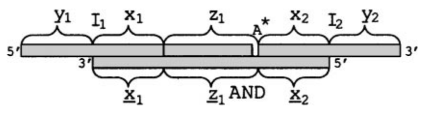


Fig. 9. Data flow molecular logic gate AND execution: hybridized oligos; A* — regions adding labeled radioactive adenine.

The mentioned two sequences hybridize each other forming a double-strand fragment.

Thus the sectors x_1 and x_2 of input I_1 and I_2 strands pair with matching sections ($\underline{\tilde{x}}_1$ and $\underline{\tilde{x}}_2$) in the AND strand. Output signals are composed of input strands and other DNA fragments. These fragments ($\underline{\tilde{z}}_i$) are reserved to accomplish special tasks. For example, they can transport additional information about logical gates, nodes in a computer program, etc. The complete molecule after annealing is shown in Fig. 9. In Eq. (1) there is a symbolic description of the whole process.

$$\left\lceil \begin{matrix} I_1 \\ \ \end{matrix} \right\rfloor, \left\lceil \begin{matrix} I_2 \\ \ \end{matrix} \right\rfloor, \left\lceil \begin{matrix} z \\ \ \end{matrix} \right\rfloor, \left\lceil \begin{matrix} \ \\ \underline{-\quad \& \quad -} \end{matrix} \right\rfloor = \left\lceil \begin{matrix} \ \\ \underline{\tilde{I}_1 \quad \tilde{z} \quad \tilde{I}_2} \end{matrix} \right\rfloor \qquad (1)$$

$$\left\lceil \begin{matrix} I_1 & z & I_2 \\ \underline{+} & \underline{\&} & \underline{+} \end{matrix} \right\rfloor \qquad (2)$$

Note that the upper strand is labeled. Labeling of this strand can be achieved by introducing radioactive, biotinylated or fluorescent nucleotide. In our experiments (see Section 6) in order to detect oligos we have employed the technique of filling the gap (A*) made of several free tymine nucleotides in the area between z_1 and I_2 by addition to a 3' end of z_1 labeled radioactive adenine ^{32}PdATP using the Klenow polymerase. All strands are ligated with T4 DNA ligase and then the output strands are disconnected from gate strands as is shown in Fig. 10 and then analyzed using standard method of the DNA electrophoresis in polyacrylamide gel. Single stranded radioactive oligonucleotides representing output of the logic gate are detected by autoradiography. Fragments of proper length could be isolated from mentioned gel and used as signal and data tokens in the next layer of molecular gates. Both AND and OR gates have similar construction, but when OR gates have two or more inputs, then two or more different molecules each with one input

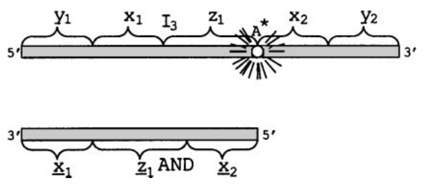


Fig. 10. Data flow molecular logic gate AND detection: disconnecting of the AND gate and an output I_3.

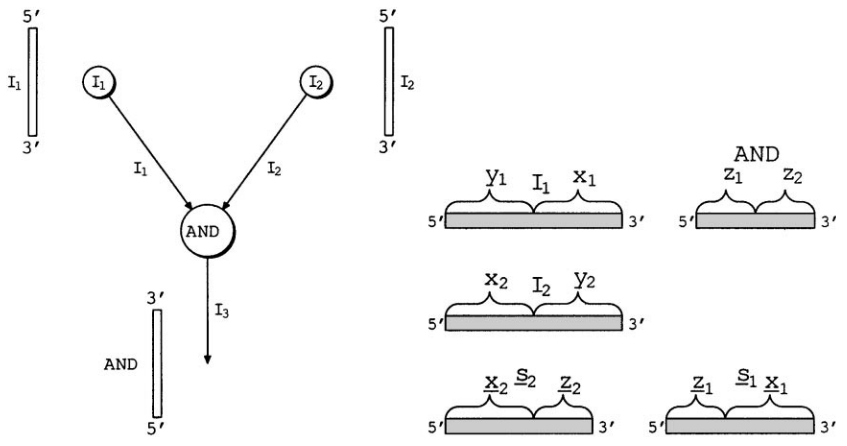


Fig. 11. Ogihara's molecular logic gate AND.

are created.

$$\begin{bmatrix} I_1 \\ \end{bmatrix}, \begin{bmatrix} I_2 \\ \end{bmatrix}, \begin{bmatrix} \& \\ \end{bmatrix}, \begin{bmatrix} \\ \underline{s_1} \end{bmatrix}$$

$$= \begin{bmatrix} \\ \underline{\tilde{I}_1 \quad + \quad \tilde{\&}} \end{bmatrix}, \begin{bmatrix} \\ \underline{s_2} \end{bmatrix} = \begin{bmatrix} \\ \underline{\tilde{\&} \quad + \quad \tilde{I}_2} \end{bmatrix} \tag{3}$$

$$\begin{bmatrix} I_1 \quad + \quad \& \quad + \quad I_2 \\ \underline{+ \quad s_1 \quad + \quad s_2 \quad +} \end{bmatrix} \tag{4}$$

Let us compare now our implementations with that of Ogihara's proposals [6,7]. Ogihara's logic gate strands from Fig. 11 and Eqs. (3) and (4) do not anneal to the input strands and are bound to these input strands by additional synthesized on-line splints as depicted in Fig. 12, strands complementary in first half to an input strand and in second half to a gate strand. Output strands shown in Fig. 13 consist of ligated input and gate strands. They are detected in a gel electrophoresis. In the case of an AND gate an output strand is composed of two input strands and a gate strand supported in the middle. In the case of an OR gate an output strand is made from one of two input strands

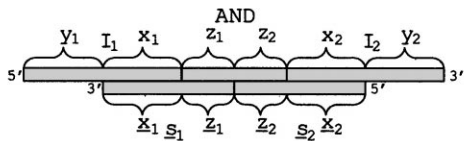


Fig. 12. Ogihara's molecular logic gate AND: execution.

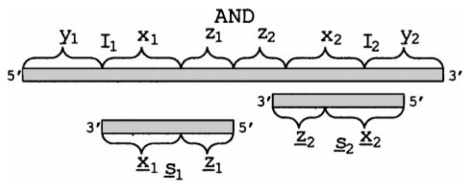


Fig. 13. Ogihara's molecular logic gate AND: detection.

and a gate strand. Thus, logic gate strands are not separated. Therefore, it is not possible to use those gates with DNA chips without problems.

5. DNA-based implementation of simple logic network

A Boolean function is composed of binary variables, operation symbols, brackets and a symbol of equivalence relation. The Boolean function describes usually more complicated logic networks composed of logic gates. Some number of connected logic gates with sets of inputs and outputs is called a combinational network. However, in a sequential network there are not only logic gates, but also flip-flops with memory [1,28].

We have considered to implement simple combinational network by operations on DNA strands. An example of a simple combinational network and its implementation in DNA as well as its perfor-

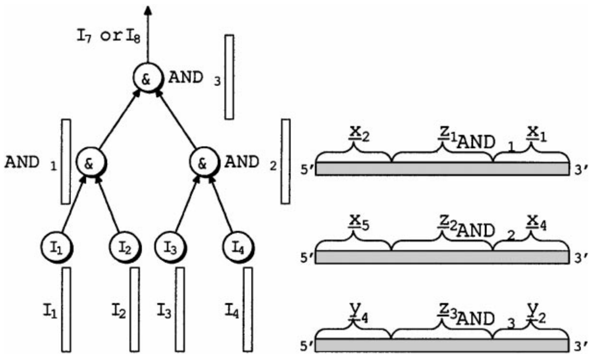


Fig. 14. Data flow molecular logic gate AND combinational network: a scheme of the logic system, oligonucleotides representing logic gates.

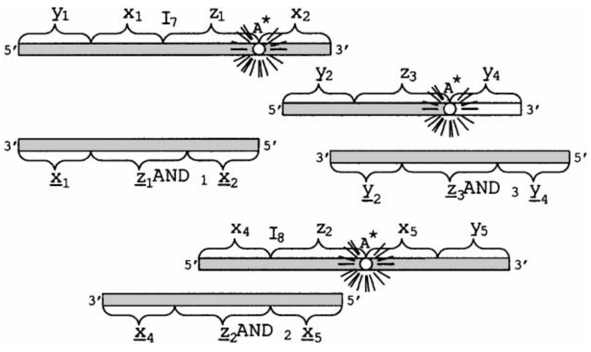


Fig. 17. Data flow molecular logic gate AND system detection: digestion by restriction enzyme of the restriction sites E_1, E_2, disconnecting of the AND gates and outputs I_7 and I_8.

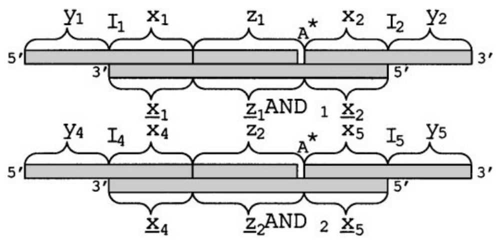


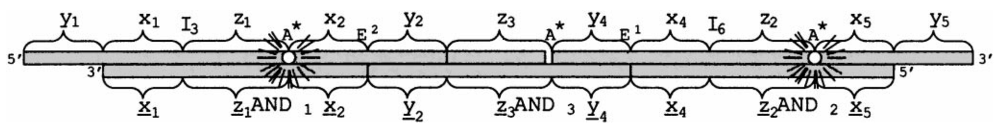
Fig. 15. Data flow molecular logic gate AND system first level execution: hybridized oligonucleotides of gates AND_1, AND_2 with oligos of inputs signals and parts of output signals; A^* regions of adding labeled radioactive adenine.

mance are shown in Figs. 14–17. The logic gates are DNA oligonucleotides. Two gates AND_1, AND_2 are self-assembling molecules. These two molecules anneal to DNA strand of the AND_3 gate. The regions named A^* are for adding labeled radioactive adenine by the Klenow polymerase to a 3’ end of oligonucleotides as described in Section 4. It should be noted that $E_{1,2}$ restriction sites are ready for digestion if there is a double DNA strand there. After hybridization and ligation a process of digestion a double strand molecule by restriction enzymes is performed. The enzymes cut the double strand in areas E_1 and E_2 yielding three fragments from which output strands could be potentially rescued after strand separation. It is necessary to remove enzymes and redundant strings before further computation. The redundant strings are removed in a process of electrophoresis.

Neighboring logic gates layers could be connected with each other to form the combinational network. A number of layers may be theoretically unlimitable if the oligonucleotides are recycled during computing after the execution of some number of layers. For construction of our molecular gate layers one can envisage use of solid support. These gates can be for example attached to DNA chips. If the AND_3 gate strand was attached to the DNA chip and the AND_1 and AND_2 gate strands to the magnetic beads with biotin, then output strings could be easily separated.

6. Initial results of experiments

To confirm oligonucleotide self-assembling as a technique useful in data flow logical operations two experiments were performed. Ten oligonucleotides were designed for self-assembling. The lower strands



y1 x1 I3 z1 A* x2 E2 y2 z3 A* y4 E1 x4 I6 z2 A* x5 y5
x1 z1AND 1 x2 y2 z3AND 3 y4 x4 z2AND 2 x5
5′ 3′

Fig. 16. Data flow molecular logic gate AND system second level execution: hybridized an oligonucleotide of the AND_3 gate with hybridized earlier oligos of gates AND_1 and AND_2; $E_{1,2}$ restriction sites ready for digestion if there is a double DNA strand there.

are composed of longest oligonucleotides named AND(30 bp), AND_2(36 bp), AND_3(42 bp) and the upper strand of three other: I_1(16 bp), SR(12 bp), SR_2(20 bp), SR_3(24 bp) and I_2(16 bp).

In the first experiment I_1 and/or I_2 were hybridized to AND in presence of SR oligonucleotide. Typically 5–10 pM of each oligonucleotide were used and hybridization was performed in T4 DNA bacteriophage ligase buffer for 5–30 min in 40°C. Under these conditions formation of the SR/AND hybrid was preferred. In the next step 5 μCi of ^{32}PdATP (3000 Ci mM^{-1}, Amersham-USB), 100 pM dGTP and 10 μl of Klenow polymerase (Amersham-USB) were added, and samples were incubated for 30 min at 37°C. Afterwards the temperature was lowered to 25°C (to hybridize I_1 and/or I_2 to AND) and samples were incubated for 30 min. 5 μl of T4 DNA ligase (Promega) were added. Ligations were performed at 25°C for 2 h, and at 14°C for 2–24 h. Ligation reactions were terminated by addition of the equal volume of the stop mixture (formamide with 0.05% bromophenol blue and 0.05% xylene cyanol). Samples were denatured by 5 min incubation in 80 °C and loaded onto polyacrylamide (8 or 20%) denaturing (8 M urea) gel. Electrophoresis was performed under standard conditions [14]. Radioactive single stranded DNA were detected in electrophoretograms by autoradiography (Figs. 18 and 19). Typically only proper single stranded DNA fragments were detected as depicted in Fig. 18. In the lane A the oligonucleotide SR was hybridized to oligonucleotide AND and radioactive nucleotides (A*) was added to 3' end of oligonucleotide SR by Klenow polymerase; in the lane B, self-assembly of oligonucleotides I_1 and SR with oligonucleotide AND; in the lane C, self-assembly of oligonucleotides SR and I_2 with oligonucleotide AND. Oligonucleotide self-assembling was not disturbed by addition of the random oligonucleotide (100 pM per reaction) (in Fig. 19 A_1, B_1 and C_1).

In the second experiment I_1 and/or I_2 were hybridized to AND in presence of SR oligonucleotide and to AND_2 in presence of SR_2 oligonucleotide. Further steps were the same as during the first experiment, but radioactive ^{32} PdATP (3000 Ci/mM, Amersham-USB) labeling adenine was exchange for $^{32}\gamma$ ATP (7000 Ci/mM, New England BioLabs) added to 3' end of SR, SR_2, SR_3 by the same Klenow polymerase. Oligos, T4 DNA ligase (Promega) and ligase buffer was incubated for 3 h in 14°C, then for 12 h in 4°C. Results are depicted in Fig. 20. Experiments with AND_3 have not been so successful. Therefore our logic system is incomplete, but there are enough results to confirm our expectations.

7. Construction of logic NAND gate using genetic methods

In theory of logic design [1] a simple NAND (¬&) gate with two inputs I_1, I_2 and one output is described by the truth table as shown in Table 1.

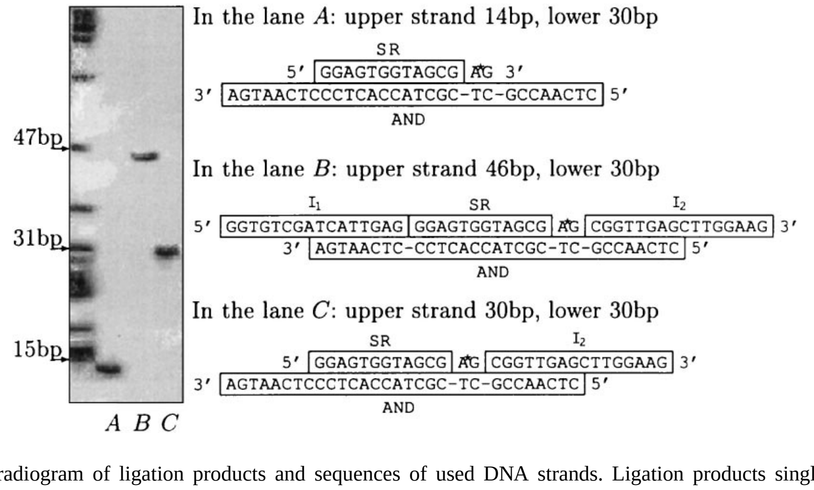

47bp
31bp
15bp
A B C
In the lane A: upper strand 14bp, lower 30bp
SR
5' GGAGTGGTAGCG A*G 3'
3' AGTAACTCCCTCACCATCGC-TC-GCCAACTC 5'
AND
In the lane B: upper strand 46bp, lower 30bp
I1 SR I2
5' GGTGTCGATCATTGAG GGAGTGGTAGCG A*G CGGTTGAGCTTGGAAG 3'
3' AGTAACTC-CCTCACCATCGC-TC-GCCAACTC 5'
AND
In the lane C: upper strand 30bp, lower 30bp
SR I2
5' GGAGTGGTAGCG A*G CGGTTGAGCTTGGAAG 3'
3' AGTAACTCCCTCACCATCGC-TC-GCCAACTC 5'
AND

Fig. 18. The first autoradiogram of ligation products and sequences of used DNA strands. Ligation products single stranded DNA were analyzed by electophoresis in denaturing 8% polyacrylamide gel.

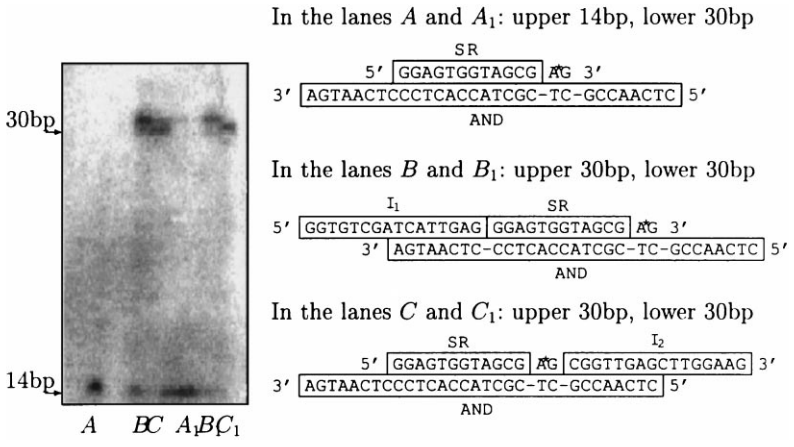
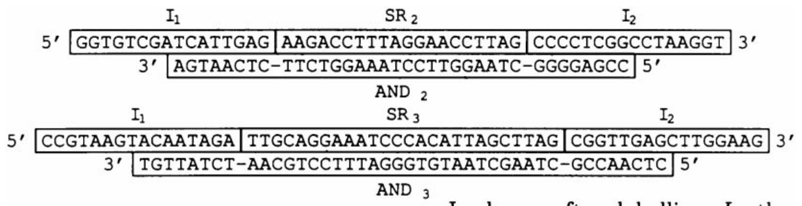
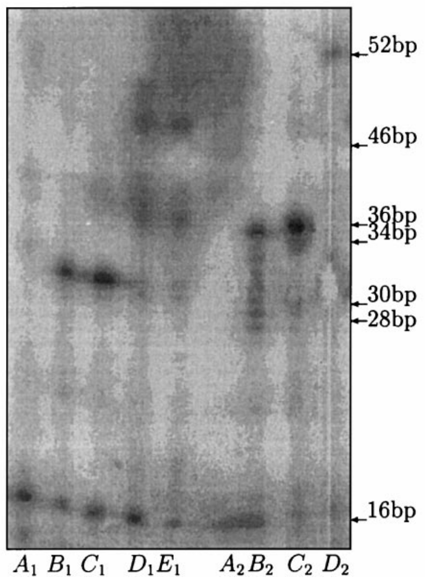

In the lanes A and A_1: upper 14bp, lower 30bp
SR
5′ GGAGTGGTAGCG AG 3′
3′ AGTAACTCCCTCACCATCGC-TC-GCCAACTC 5′
AND
In the lanes B and B_1: upper 30bp, lower 30bp
I1
SR
5′ GGTGTCGATCATTGAG GGAGTGGTAGCG AG 3′
3′ AGTAACTC-CCTCACCATCGC-TC-GCCAACTC 5′
AND
In the lanes C and C_1: upper 30bp, lower 30bp
SR
I2
5′ GGAGTGGTAGCG AG CGGTTGAGCTTGGAAG 3′
3′ AGTAACTCCCTCACCATCGC-TC-GCCAACTC 5′
AND
30bp
14bp
A
B C
A1 B C1

Fig. 19. The second autoradiogram of ligation products and sequences of used DNA strands. Ligation products single stranded DNA were analyzed by electrophoresis in denaturing 20% polyacrylamide gel.


I1
SR2
I2
5′ GGTGTCGATCATTGAG AAGACCTTTAGGAACCTTAG CCCCTCGGCCTAAGGT 3′
3′ AGTAACTC-TTCTGGAAATCCTTGGAATC-GGGGAGCC 5′
AND2
I1
SR3
I2
5′ CCGTAAGTACAATAGA TTGCAGGAAATCCCACATTAGCTTAG CGGTTGAGCTTGGAAG 3′
3′ TGTTATCT-AACGTCCTTTAGGGTGTAATCGAATC-GCCAACTC 5′
AND3



In lanes after labelling I_1 the following strands are seen: A_1: I_1^*
B_1: $I_1^* + SR/AND_1$; ligation is not 100% effective; 28 bp correct band;
C_1: $I_1^* + SR^+ + AG/AND_1$; added normal adenine and guanine; 30 bp correct band
D_1: $I_1^* + SR^+ + AG/AND_1 + I_2$; 46 bp correct band
E_1: repeated D_1
A_2: I_1^*
B_2: $I_1^* + SR_2/AND_2$; ligation is not 100% effective; 34 bp correct band;
C_2: $I_1^* + SR_2^+ + AG/AND_2$; added normal adenine and guanine; 36 bp correct band
D_2: $I_1^* + SR_2^+ + AG/AND_2 + I_2$; 52 bp correct band

Fig. 20. The sequences of two next logic gates AND_2, AND_3 and the third autoradiogram of ligation products made from AND_1, AND_2. Ligation product single stranded DNA were analyzed by electrophoresis in denaturing 10% polyacrylamide gel.

Table 1
Logic gate ¬& inputs and outputs

I_1	I_2	$I_1\neg\& I_2$
0	0	1
0	1	1
1	0	1
1	1	0

In further described experiments all combinations of inputs are implemented. In molecular computing logic gates inputs and outputs are represented by DNA molecules, strands. To enable hybridization process, input signals are complementary to adequate sectors in logic gates strands. Two strands, which are complementary, anneal to each other. This means stick or bind together forming a double-stranded fragment. Implementation of the ¬& gate is provided in Fig. 21. As follows from this figure, it consists of three strands. A first input strand is complementary to a sector of the ¬& gate. A second input strand is complementary to the another ¬& gate strand sector. If an input signal is equal to 0, an adequate input strand is absent. If both input signals are equal to 1 this means they are present in a reaction, so an output of an ¬& gate is equal to 0. Thus, a final molecule after annealing is illustrated in Fig. 21. Equations of creation of the logic gate ¬& are the following:

$$\left\lceil I_1 \right\rfloor, \left\lceil I_2 \right\rfloor, \left\lceil \underline{\;-\quad \neg\& \quad -\;} \right\rfloor = \left\lceil \underline{\tilde{I}_1 \quad + \quad \tilde{I}_2} \right\rfloor$$

$$\left\lceil \begin{array}{ccc} {}^{\#^{1,\bar{1}}}I_1 & + & {}^{1\#}I_2 \\ & {}^{\#_{\bar{1}}} & \\ \underline{-} & \underline{\neg\&} & \underline{-} \end{array} \right\rfloor$$

When both input strands are present then appropriate strands can be cut, digested by the second class restriction enzyme. An area closed in a rectangle is just the restriction site of the second class enzyme, at the ends of arrows — the digestion site of the second class enzyme. The symbol # has a very important way in describing enzyme restriction and digestion sites. The first class enzymes have both sites in the same place and two cuts only for double-stranded oligos, e.g., for an enzyme with no 1 are two cuts: 1 from top and $\tilde{1}$ from bottom ${}^{\#_{1,\bar{1}}}a$. Thus, their cuts are marked by lower indices of the symbol #. The second class enzymes have different restriction and digestion sites. Therefore, they cut sometimes far from their restriction site, where they connect to and their cuts at the restriction site are marked by upper indices of # in order to identify enzymes, but at the digestion site — by lower indices of # as in the case of the first class enzymes.

Annealing is the opposite process of melting taking place during heating, where a test tube is cooled, permitting complementary strands to hybridize. If

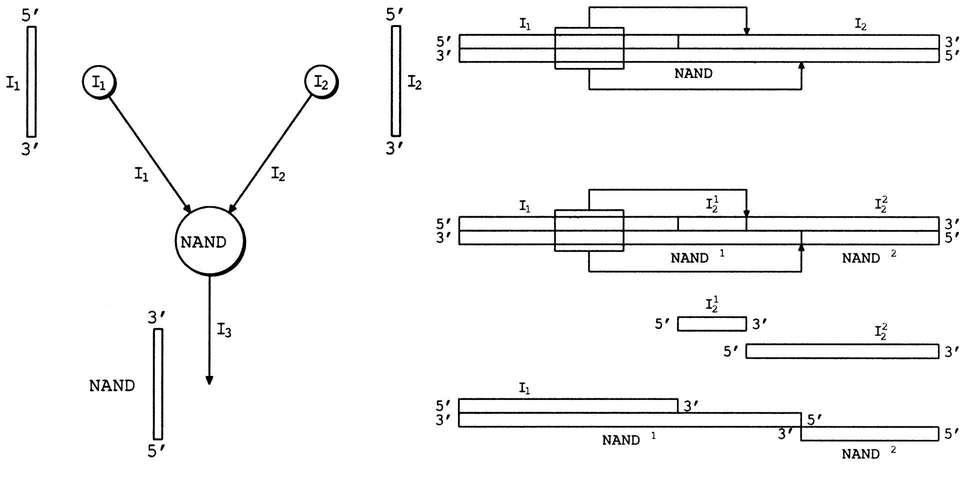


Fig. 21. Implementation of a logic ¬& gate: its scheme, its DNA strands representing the gate and input signals: I_1, I_2, its hybridized and digested by the second class enzyme strands; if both input signals are present, the gate and the second input are destroyed.

double-stranded DNA has no links between nucleotides then a unified strand may be created by DNA ligase, but in this experiment ligase is not utilized. After hybridization a digestion process by a second class restriction enzyme is performed. The enzyme destroys ¬& gate and input I_2 strand as is shown in Fig. 21 only on that condition that two input and one gate strands are present in the reaction so a result of operation equals logic zero. Resulting from molecular computation logic ones are the whole ¬& strands, and are extracted in the process of electrophoresis. In future it will be possible to protect ¬& gate strands against the digestion and synthesize on a DNA chip. Thus logic ones will be in this case when the second input strand is cut.

In another implementation of NAND gate we apply a concept of a fluorogenic probe. A new idea in the area of DNA sequence detection emerged in 1991 [26]. It employs the fluorogenic probe, which consists of a strand with both a reporter (R) and a quencher dye (Q) attached and separated. A fluorogenic probe anneals specifically if and only if the target sequence is present. Subsequently, not hybridized parts of the probe are cleaved by special nucleases. Cleavage of the probe destroys free single-stranded parts and causes contacting of the quencher dye with the reporter dye. In this case it generates an increase in the fluorescence intensity of the reporter dye. It is possible to implement this methodology in the field of molecular gates. In our case the fluorogenic probe will be connected with the second strand and the second class enzyme will destroy this strand causing an increase in the fluorescence. Of course, an image of a DNA chip with such ¬& gates can be obtained with a specially designed scanning confocal fluorescence microscope.

It is a well-known fact that double-stranded DNA may be dissolved into single strands (or denatured) by heating to a temperature determined by the composition of the strand. Heating breaks the hydrogen bonds between complementary strands. By the way since G–C pair is joined by three hydrogen bonds, the temperature required to break it is slightly higher than that for an A–T pair, joined by only two hydrogen bonds. This factor must be considered during designing sequences to represent computational elements [5,15]. In described below experiments sequences of strands are balanced. This means about 50% for G–C pairs and 50% for A–T pairs.

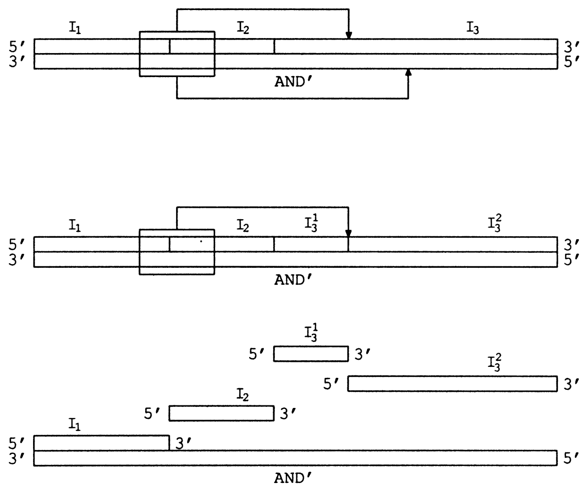


Fig. 22. The gate AND with overflow after adding two input logic ones.

After digestion remained double-stranded DNA is melted to single strands, because input signals are attached to the gate strands with some sectors (Fig. 21). The output or gate strands may be then separated and extracted. For example the NAND gate strands may have a 5’ end biotinylated and attached to streptavidin-coated microplates, polystyrene, or magnetic beads or synthesized on a DNA chip. The not bonded, cut input strands then may be washed and the NAND gates used in next molecular computation. Of course, the ¬& strand have to be synthesized in a special way in order not to be digested by the restriction enzyme. In Fig. 22 the adding two inputs AND gate with possibility of sending overflow (adding two ones generates else another one during ordinary binary computation) is presented.

8. Further laboratory verifications

To check our concept experimentally a simple logic NAND gate (denoted by ¬&) was designed and tested in the laboratory of genetic engineering. The FokI enzyme was selected as the second class restriction nuclease.

According to Fig. 21 particular oligonucleotides were chosen as shown in Fig. 23. Synthetic oligonucleotides INF_1 $(I_1, 16\,\mathrm{bp})$, INF_2 $(I_2, 19\,\mathrm{bp})$ and

```
            Before cutting - in the lane A−
               I1                      I2
5' |CTAGAGAGGATGAGAG|GTCATGCATCTCTTCACAC| 3'
3' |GATCTCTCCTACTCTC-CAGTACGTAGAGAAGTGTG| 5'
                      NAND
            After digestion - in the lane A+
               I1            I2^1          I2^2
5' |CTAGAGAGGATGAGAG|GTCAT|GCAT-CTCTTCACAC| 3'
5' |GATCTCTCCTACTCTC-CAGTA-CGTA|GAGAAGTGTG| 3'
                NAN D^1            NAN D^2
               In the lanes B+,B−
                                  I2
                   |GTCATGCATCTCTTCACAC| 3'
3' |GATCTCTCCTACTCTCCAGTACGTAGAGAAGTGTG| 5'
                      NAND
               In the lanes C+,C−
               I1
5' |CTAGAGAGGATGAGAG|
3' |GATCTCTCCTACTCTCCAGTACGTAGAGAAGTGTG| 5'
                      NAND
```

Fig. 23. Sequences of experiment oligonucleotides together with their appropriate lanes in the autoradiogram: in the lanes A with both inputs equal to 1; in the lanes B with the first input equal to 0; in the lanes C with the second input equal to 0.

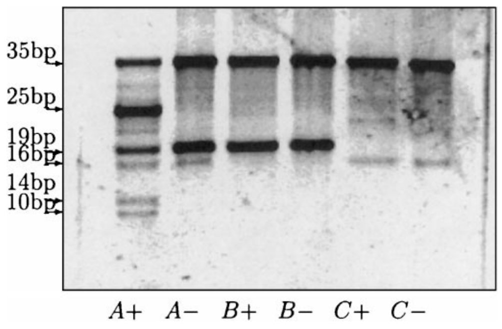


Fig. 24. The first electrophoretic pattern of synthetic oligonucleotides I_1, I_2 and ¬& hybridized and digested with the FokI restriction nuclease.

INF_5 (¬&, 35 bp) were mixed in equimolar amounts, 60 pM each, in 20 μl solution (10 mM Tris–HCl pH 7.5, 10 mM $MgCl_2$, 50 mM NaCl, 1 mM dithioerythritol). The oligonucleotides were placed in 50°C for 1 min and after that transferred to 37°C. Half of the sample was digested with 17 units FokI (Amersham) restriction nuclease for 1 h at 37°C. After digestion 5 μl formamide was added to the digested and control samples, denatured at 80°C during 1 min and applied on 15% polyacrylamide gel containing 7 M urea and TBE buffer. The gel was run until bromophenol blue passed 15 cm. After that the gel was soaked in SYBR Green II RNA gel strain (Molecular Probes) and photographed. The strands were cut by the second class enzyme in such a way as described in Fig. 23. The photographs of electrophoretic gel are shown in Fig. 24.

In the provided experiment the NAND gate was constructed from the strand INF_5 complementary to two input strands INF_1 and INF_2. The first input strand INF_1 created with the strand INF_5 a restriction site for the second class enzyme FokI: GGATG $N_{9/13}$ (the eight position in the strand INF_1 from the 5' end). The second input strand INF_2 bonded to the strand INF_5 enabled cutting by the enzyme. Thus, it demonstrates that our biochemical reaction is equivalent to the logic ¬& gate operation under a condition that if both input signals are equal to logic one then a respective output of the logic NAND gate is equal to logic zero. In order to check whether lack of either one or both input strands stopped digestion of the INF_5 by the second class enzyme, two successive experiments were performed.

In the experiment as shown in Fig. 24 all combinations of inputs (except that with two absent inputs) were put in reactions. In the lane A both input strands were present and in the lanes B reactions — the first input strand was absent, during the C reactions — the second input strand was absent. In the mentioned figure it is placed a photograph of a gel used in the process of electrophoresis. Lanes with the symbol — were made of DNA taken from vessels before digestion by the second class enzyme FokI: GGATG $N_{9/13}$, but after hybridization, and lanes with the symbol + after hybridization and digestion. Results show that the FokI enzyme may be useful for implementation of the logic gate. The all states from the truth table placed in Table 1 can be implemented. We have not decreased the amount of strands in order to optimize the reaction conditions. Thus gate and second input strands did remain after digestion.

9. A new concept of calculating Boolean functions

After the first ideas and experiments [5–8,19–21] we show a new method of calculating a Boolean (logic) function interpreted as a graph or a tree which is executed with the help of self-assembly DNA molecules. The virtue of this method is to use auto-

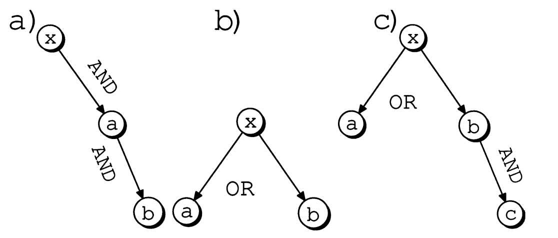


Fig. 25. The exemplary trees (a) $x = f(a, b) = a \wedge b$, (b) $x = f(a, b) = a \vee b$ and (c) $x = f(a, b, c) = a \vee (b \wedge c)$ with the predefined root x.

mated DNA amplification process (PCR) in only one vessel and check the experiment's result only with DNA length comparison process (electrophoresis) during only few hours. Specially prepared DNA fragments can be obtained from data flow machines or are specially prepared. The exemplary Boolean function is defined below as

$$f : \{a, b, \dots\} \Rightarrow x, \quad \text{where } a, b, \dots, x$$

can be TRUE or FALSE.

Thus, each Boolean function can be formed only with its arguments and three operations: negation (NOT $\neg$), conjunction (AND$\wedge$) and disjunction (OR$\vee$). Each arithmetic or logical expression (including the Boolean one or a rule with two premises and one conclusion from a knowledge base) may be presented as a tree. The construction of the tree starts with creating nodes, which are related to function arguments plus one node — root of the tree — named x. Edges reflect the layout of operations in the expression, where each disjunction operation separates engaged arguments (nodes) and each conjunction operation lengthens participating ones. There are three examples depicted in Fig. 25. The above model of the logic function's graph does not use the negation operation (NOT) by now. The $\neg x$ and x are treated as separated arguments and negative operations have to be transformed to normal operations with negative arguments. Thus, the method cannot calculate all Boolean functions and requires further research. The function's result is equal to TRUE when at least one leaf of the tree can be reached from the root through the successive nodes set only to TRUE. The new method of calculating logic functions relies on the above concept.

To explain this idea consider the implemented in our laboratory logic function with four arguments as follows:

$$x = f(a, b, c, d) = a \wedge (b \vee (c \wedge d)).$$

The relating tree is depicted in Fig. 27a. If the predefined node x is connected by the operation AND only with one node (a), it can be bypassed and replaced with the next one (a). The same mechanism could be applied to the exemplary tree presented in Fig. 25a — the node x would be removed. After creating the function tree, it is necessary to design DNA sequences with DNA molecules. The node oligos have to be unique and meet the criteria of being not similar. Unwanted hybridization is not allowed here. Both halves of an edge oligo ought to be complementary with appropriate oligo halves of both nodes, which are connected with this edge. The design is similar to that one of Adleman [9] and ensures, that edges will anneal only with appropriate nodes as is seen in Fig. 26. These neighboring DNA strands are *single* and their presentation is not standard. The DNA fragment a is assumed as a sequence to be amplified, and the rest of fragments $(b, c, d, xp, a \gg b, a \gg c, a \gg d)$ — as primers. Primers partially (in their halves) have identical sequences of nucleotides. An additional DNA fragment (*xp*) will serve as a primer for PCR method at the second end of (*xa*). The *Taq* DNA polymerase adds nucleotides to the 3' end of primers annealed to single DNA strands.

In Fig. 26 there are depicted nine double strands, if we connect neighboring, completely complementary oligos. However, a number of internal reactions on double-stranded oligos may be equal to five. For one-time amplification reaction they may be written as

$$\left\lceil \begin{array}{l} xp \\ \underline{-\quad a_1} \end{array} \right\rfloor = \left\lceil \left[\begin{array}{l} xp \\ \underline{-\quad a_1} \end{array} \right]^{\$} \right\rfloor = \left\lceil \begin{array}{l} \tilde{a}_1 \\ \underline{a_1} \end{array} \right\rfloor$$

$$\left\lceil \begin{array}{l} -\quad \tilde{a}_1 \\ \underline{\quad a >> c \quad +} \end{array} \right\rfloor = \left\lceil \left[\begin{array}{l} -\quad \tilde{a}_1 \\ \underline{\quad a >> c \quad +} \end{array} \right]^{\$} \right\rfloor = \left\lceil \begin{array}{l} \tilde{a}_2 \\ \underline{a_2} \end{array} \right\rfloor$$

$$\left\lceil \begin{array}{l} -\quad \tilde{a}_2 \\ \underline{\qquad c \quad +} \end{array} \right\rfloor = \left\lceil \left[\begin{array}{l} -\quad \tilde{a}_2 \\ \underline{\qquad c \quad +} \end{array} \right]^{\$} \right\rfloor = \left\lceil \begin{array}{l} \tilde{a}_3 \\ \underline{a_3} \end{array} \right\rfloor$$

$$\left\lceil \begin{array}{l} -\quad \tilde{a}_3 \\ \underline{\quad c >> d \quad +} \end{array} \right\rfloor = \left\lceil \left[\begin{array}{l} -\quad \tilde{a}_3 \\ \underline{\quad c >> d \quad +} \end{array} \right]^{\$} \right\rfloor = \left\lceil \begin{array}{l} \tilde{a}_4 \\ \underline{a_4} \end{array} \right\rfloor$$

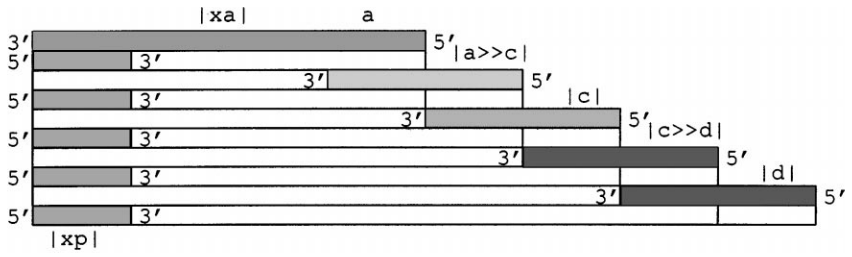


Fig. 26. The way of creating the longest DNA strand with one vessel PCR reaction (strands above are *single*).

$$\left\lceil \begin{array}{cc} - & \tilde{a}_4 \\ \underline{} & \underline{d \quad +} \end{array} \right\rfloor = \left\lceil \left[\begin{array}{cc} - & \tilde{a}_4 \\ \underline{} & \underline{d \quad +} \end{array} \right]^{\$} \right\rfloor = \left\lceil \begin{array}{c} \tilde{a}_5 \\ \underline{a_5} \end{array} \right\rfloor,$$

where $a_1 < a_2 < a_3 < a_4 < a_5$.

These single strands are created during PCR cycles of lengthening primers annealed to DNA fragments. Thus, white parts of single strands from Fig. 26 are made by *Taq* DNA polymerase, but also with the help of lengthened oligos *xa* or a. The successive cycles of PCR process will create new DNA fragments, which will be the lengthened sequences of a. These new copies are the possible ways from the root *xa* to any point in the graph expression (a node or an edge). It is easy to see, that the lack of any primer (relative argument is set to FALSE) will not allow to build the final solution — the lowest strand *xd*, only possible solutions will be amplified.

Fig. 27b reveals the possibilities of getting TRUE as a result. In this case, at least one of two sequences must be realized: with a length of 376 or 418 (lengths of hybridized fragments are written in upper indices).

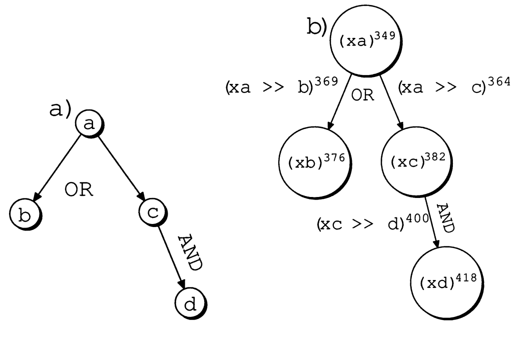


Fig. 27. The exemplary tree of $x = f(a, b, c, d) = a\wedge(b\vee(c\wedge d))$; (a) and (b) without the bypassed x (a instead of *xa*); in (b) additional information about lengths of DNA node and edge strands created during PCR cycle.

No other combination of primers will create similar solution. DNA fragments with the same number of nucleotides cannot be obtained. All possible lengths of created by *Taq* polymerase strands (from root to any node in the tree) are shown in Fig. 27b. It is clear that this coding is correct, because no other fragment combination gives the length of 376 or 418. With the DNA chips technology further obstacles will disappear. The first halves of the oligos d from their 5’ end may be synthesized on DNA chips, labeled with fluorescent materials and detected on the microscope photograph among many thousand other ones.

10. Implementation and verification of the method from Section 9

To verify the concept presented in previous point we performed the following experiments. It is the first step to collect all required DNA fragments as described in Fig. 28. The function’s arguments have the following values:

$$a = \text{TRUE},\quad b = \text{FALSE},\quad c = \text{TRUE},\quad d = \text{TRUE}.$$

In this situation, the solution is naturally TRUE, because going from the root of the tree (a) it is possible to reach one of its leaves (d) through the node c. The second solution (with the leaf-argument b set to FALSE) cannot be realized. The aim of the proposed method is to show, whether it is possible to go from the root of the tree to one of its leaves. In order to prefer longer sequences in PCR process, two parameters were tuned. The first method’s efficiency depends on regulation of primer’s concentration, e.g., the whole experiment may be divided into five parts. Table 2 presents all five parts and participating primers

```
b^22
5′ TCTCATGGGGGAATTCATATGG 3′

c^37
5′ AGACTCTGCACAGTTAGTAGTCTGATATTCATGAAAG 3′

d^36
5′ AGCTCAGCCTGACGTGCACCAGTACGAATGCCAGTG 3′

(a >> b)^40
5′ GGGGAATTCATATGGATTATAAAGATGATGATGACAAGGG 3′

(a >> c)^33
5′ AGTCTGATATTCATGAAAGATGATGATGACAAG 3′

(c >> d)^36
5′ CCAGTACGAATGCCAGTGAGACTCTGCACAGTTAGT 3′

xp^31
5′ AAAAGCTTAAGCAGTACCTCCAAGCTCGATC 3′

xa^349
5′ AAAGATGATGATGACAAG 3′
5′ GGATCCGATCAACATCTGCC 3′
5′ TGGGCGCTTCCTTACTCCTG 3′
5′ CCATTACCTCAGATGATAAA 3′
5′ TGTGTTTTCCCATTCATCTA 3′
5′ TAAAGGCAACCTGTATTTCG 3′
5′ TTGCACTCTGCATGACTCCA 3′
5′ CATACTACTGGTGTTCCGTA 3′
5′ ACTACCTATTATATGAAAAG 3′
5′ ATGGAGATACTGCAGAAGCA 3′
5′ CAGACTATGCCAGGTGTGCT 3′
5′ TTGCCCTTTATCTTTCGAGG 3′
5′ CAAAGAATAGACAGTTGCAT 3′
5′ AAAAGAAGGGAGTGTTTTTT 3′
5′ CAAAGTATTGGTGCCCGGTC 3′
5′ ACCCCAAATTATGACCAGAT 3′
5′ CGAGCTTGGAGGTACTGCTT 3′
5′ AAGCTTTT 3′
```

Fig. 28. Sequences of oligonucleotides used in the experiment.

Table 2
Subsets of oligos with their concentration

Part No.	Used primers (concentrations are put in brackets (pM))
1	$a \gg b$ (50)
2	$a \gg b$ (5); $a \gg c$ (50)
3	$a \gg b$ (5); $a \gg c$ (1); c (50)
4	$a \gg b$ (5); $a \gg c$ (1); c (1); $c \gg d$ (50)
5	$a \gg b$ (5); $a \gg c$ (1); c (1); $c \gg d$ (1); d (50)

with their concentrations. The last fifth subset was used during this experiment, because the presence of its primers satisfies assumptions described above and depends on the results of previous experiments with PCR [14]. As is shown, always the furthest primer from the tree root has larger concentration. This ensures that when the competitive hybridization process of PCR is taking place then the furthest primer is always preferred (the longest sequence will be amplified). It is the second method to increase the amount of nucleotides of type G or C in primers. DNA fragments, which mainly consist of G or C show better stability of annealing and their hybridization performs faster. In this experiment, the 18 bp part of primer $a \gg c$, which anneals to $\tilde{a}$ consists of 6 G or C. The 18 bp part of ci, which hybridizes to $\widetilde{c \gg d}$ consists of 6 too, but the second half of $c \gg d$: 8 and the half of d: 10. The implementation started with preparing the following reaction mixture ($7 \times 50\,\mu$l): $10 \times$ *Taq*DNA polymerase buffer (500 mM KCl, 25 mM MgCil$_2$, Tris–HCl pH 9.0) (35 μl), primer *xp* (7 μl), dNTP(14 μl, 2.5 mM), fragment a (2 μl), H_2O (up to 50 μl), *Taq* DNA polymerase (7 μl, 7 units). Five samples of successive task parts consisted of 46 μl of MIX, 1 μl of each participating primer shown in Table 2 and TE (10 mM Tris–HCl, 1 mM EDTA, pH 8.0) till 50 μl. It was the next step to design a program of sequences for PCR process presented in Table 3.

As is shown, this experiment does not only reveal the Boolean function's result (TRUE, because a leaf d was reached; look at Fig. 27b), but it also confirms

Table 3
A sequence of operations with their appropriate times

PCR's part	Phase's name	Temperature (°C)	Time (s)
First part (12 cycles)	Denaturation	94	60
	Hybridization	50	60
	Polymerization	72	30
Second part (18 cycles)	Denaturation	94	30
	Hybridization	60	30
	Polymerization	72	30

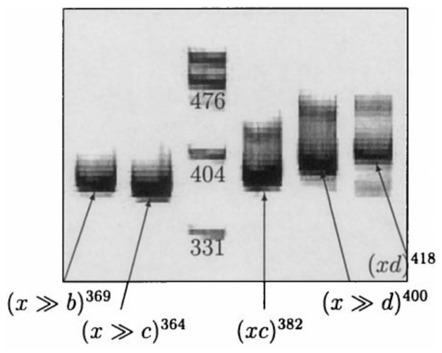


Fig. 29. DNA electrophoretogram of one-time PCR reaction.

all phases of lengthening (designed five parts of the task). It confirms that presented method works.

The second part of amplification was performed with increased hybridization temperature and less time for reaction. It forced more stable DNA fragments to participate more actively in the amplification process. The result proved that the parameters of the stepwise amplification leading to Boolean functions solution, were correctly chosen. It was important to take care of the amount of amplified sequences, because too many copies would make a photo illegible. The further a primer is from the root the later it participates in amplification. So, first samples (from Table 2) should be removed from PCR machine earlier. Samples 1 and 2 (reaching $a \gg b$ and $a \gg c$) were removed after the 15th cycle of PCR, sample 3 (reaching c) after the 20th cycle, sample 4 after the 25th cycle and the last subset at the end, i.e., after the 30th cycle.

When the whole PCR was finished 15 μl of 0.05% bromophenol blue was added to each sample. Then 7.5 μl of each sample was put into the standard polyacrylamid gel and the electrophoresis was started (170 V, 12 W). After 1.5 h the gel was soaked for 10 min in ethidium bromide solution and photographed. The obtained result is depicted in Fig. 29. It is worth considering, that the whole demonstration with prepared oligos was done only during 4 h.

11. Conclusions

We have explored the possibility of implementation of data flow logical operations by DNA manipulations. We have demonstrated that self-assembly of DNA can be utilized to provide the flow of data. In our approaches standard genetic operations are employed among others digestion by the second class restriction nuclease FokI. An appropriate DNA reactions have been performed and their results confirmed our assumptions. We think that our methodology has several advantages when compared with that proposed by Ogihara and Amos [6–8] and can be further improved by the use of DNA chips in order to obtain their fluorescent images. Last described method in Sections 9 and 10 presents a new way of calculating logic functions with an usage of DNA computing. The major virtue of this method is to use one-time PCR in only one vessel and electrophoresis. It practically shows the possibility of quick implementation. Most significant fragments are taken into consideration through the lengthening process. Then the identification is performed with an ordinary electrophoresis. Unfortunately, this method does not include the negation operation. Implementing logic circuits on DNA chips allows building more sophisticated data flow devices. This would permit for easy separation of output and input strands and would provide an important step towards massively parallel molecular computer. However, further research is required to gain better understanding of this area.

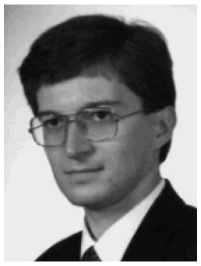
Acknowledgements

The work was supported by the Polish State Committee for Scientific Research, through the KBN grant no. 8T11F00816. The paper was processed with the LATEX 2ϵ macros package.

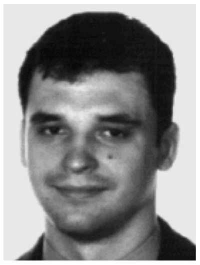
References

[1] N.N. Biswas, Logic Design Theory, Prentice-Hall, Englewood Cliffs, NJ, 1993.
[2] F.J. Hill, G.P. Peterson, Switching Theory and Logical Design, Wiley, New York, 1974.
[3] E. Landry, Y. Kishida, A survey of data flow architectures, http://www.sparc.spb.su/ans/project/master/, Internet mirror.
[4] J.J. Mulawka, P. Wąsiewicz, Molecular computing (in Polish), Informatyka 7/8 (1998) 36–39.
[5] J.J. Mulawka, P. Borsuk, P. Węgleński, Implementation of the inference engine based on molecular computing technique, in: Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC'98), Anchorage, 1998, pp. 493–498.

[6] M. Ogihara, A. Ray, Simulating Boolean circuits on a DNA computer, Computer Science Department, University of Rochester, Singapore, Technical Report TR 631, 1996.
[7] M. Ogihara, A. Ray, The minimum DNA computation model and its computational power, University of Rochester, Singapore, Technical Report TR 672, 1998.
[8] M. Amos, P. E. Dunne, DNA simulation of Boolean circuits, Department of Computer Science, University of Liverpool, UK, Technical Report CTAG-97009, 1997.
[9] L.M Adleman, Molecular computation of solutions to combinatorial problems, Science 266 (1994) 1021–1024.
[10] L.M. Adleman, On Constructing a Molecular Computer. In: E.B. Baum, R.J. Lipton (Eds.), DNA Based Computers, DIMACS 27, American Mathematical Society, 1996.
[11] S. Kurtz, S. Mahaney, J.S. Royer, J. Simon, Biological Computer, Internet.
[12] A Bibliography of Molecular Computation and Splicing Systems, http://liinwww.ira.uka.de/bibliography/misc/dna.html
[13] E.B. Baum, Building an associative memory vastly larger than the brain, Science 268 (1995) 583–585.
[14] J. Sambrook, E.F. Fritsch, T. Maniatis, Molecular Cloning, A Laboratory Manual, 2nd ed., Cold Spring Harbor Laboratory Press, New York, 1989.
[15] M. Amos, DNA computation, Department of Computer Science, University of Warwick, UK, Ph.D. Thesis, 1997.
[16] G.M. Papadopoulos, Implementation of a general purpose data flow mulitprocessor, MIT Laboratory for Computer Science, Cambridge, Ph.D. Thesis, 1988.
[17] L. Wodicka et al. Genome-wide expression monitoring in saccharomyces cerevisiae, Nature Biotechnol. 15 (1997) 1359–1367.
[18] R.J. Lipton, DNA solution of hard computational problems, Science 268 (1995) 542–545.
[19] P. Wąsiewicz, P. Borsuk, J.J. Mulawka, P. Węgleński, Implementation of data flow logical operations via self-assembly of DNA, Lecture Notes in Computer Science 1586 (1999) 174–182.
[20] J.J. Mulawka, P. Wąsiewicz, A. Płucienniczak, Another logical molecular NAND gate system, in: Proceedings of the Seventh International Conference on Microelectronics for Neural, Fuzzy, and Bio-Inspired Systems (MicroNeuro'99), 7–9 April 1999, Granada, Spain, pp. 340–345.
[21] T. Leete, J. Klein, H. Rubin, Bit operations using a DNA template, in: Proceedings of the Third DIMACS Workshop on DNA-based Computers, June 1997, Philadelphia, PA, USA, pp. 159–166.
[22] G.G. Lennon et al. Hybridization analyses of arrayed cDNA libraries, Trends Gen. 7 (1991) 314–317.
[23] S.P.A. Fodor et al., International Patent Application PCT/US91/09226, 1991.
[24] M. Schena et al. Quantitative monitoring of gene expression patterns with a complementary DNA microarray, Science 270 (1995) 467–470.
[25] S.P.A. Fodor et al. Light-directed, spatially addressable parallel chemical synthesis, Science 251 (1991) 767–773.
[26] P.M. Holland et al. Detection of specific polymerase chain reaction product by utilizing the 5'$\rightarrow$3' exonuclease activity of thermus aquaticus DNA polymerase, Proc. Natl. Acad. Sci. USA 88 (1991) 7276–7280.
[27] J.J. Mulawka, Expert Systems (in Polish), WNT, Warsaw, 1996.
[28] K. Pederson, Expert System Programming, Wiley, New York, 1989.
[29] J.J. Mulawka, T. Janczak, P. Borsuk, P. Węgleński, Reasoning via DNA based decision trees, in: Proceedings of the First International Conference on Rough Sets and Current Trends in Computing (RSCTC'98), Warsaw, 17–26 April 1998.
[30] R.C. Murphy et al., A new algorithm for DNA based computation, in: Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC'97), Indianapolis, IA, USA, 1997, pp. 207–212.
[31] M.S. Malone, Beyond semiconductors, The Microprocessor: A Biography, Springer, Berlin, 1995.
[32] J.J. Mulawka, M.J. Oćwieja, Molecular inference via unidirectional chemical reactions, in: Proceedings of the Second International Conference on Evolvable Systems (ICES'98), Lausanna, Switzerland, 1998, pp. 372–379.
[33] P. Węgleński, Molecular Genetics (in Polish), PWN, Warsaw, 1995.



Piotr Wąsiewicz graduated from the Warsaw University of Technology in automatics in 1996. Now, he is a Ph.D. student in the area of molecular DNA computing. He is a member of IEEE and participated in many international conferences. This is an address of his web site: http://www.ise.pw.edu.pl/~pwas.

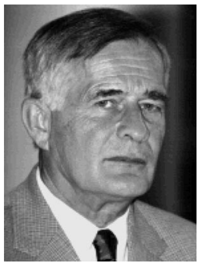


Artur Malinowski graduated in 1999 from the Faculty of Informatics at the Warsaw University of Technology. He wrote his Master Thesis in the area of DNA computing.

Robert Nowak graduated in 1999 from the Faculty of Informatics at the Warsaw University of Technology. He wrote his Master Thesis in the area of DNA computing.

Prof. Jan J. Mulawka got his M.Sc. degree with honors in 1971, doctor degree in 1976, and the title of Professor in 1995. Since 1973 he has been working at Warsaw University of Technology formerly as Research Fellow, obtaining the successive positions of Senior Assistant (1975), Assistant Professor (1976), Associate Professor (1987) and Professor (since 1996). During his employment at the Warsaw University of Technology he lectured many courses on Electronics and Artificial Intelligence. His recent research field is molecular DNA computing. Look at http://www.ise.pw.edu.pl/~mole.

Dr. Piotr Borsuk graduated in biology from University of Warsaw. For many years he has been employed at University of Warsaw in Faculty of Biology. His teaching, research and development activities have been in the area of genetics. He is a recipient of many awards for leading research activities and teaching excellence. Recently, he has been interested in molecular DNA computing.



Prof. Piotr Węgleǹski graduated in biology. For many years he has been the Director of Genetics Division in Faculty of Biology at University of Warsaw. He lectured a number of courses on Biology and Genetics. He holds the position of Professor in the Institute of Biochemistry and Biophysics of Polish Academy of Science. He is the author of many papers and books on Genetics. Recently (in 1999) he has been elected the Rector of University of Warsaw.

Prof. Andrzej Płucienniczak is considered as one of the best experimentators in genetic engineering. For many years he was the Director of Biochemistry Division at Medical University in Lodz. He was employed as a Professor in Center of Microbiology and Virology of Polish Academy of Science. He was the Head of Genetic Engineering Division of *PP. TERPOL* in Sieradz. He is the author of many patents and papers. Presently, he is the Vice-director of the Institute of Biotechnology and Antibiotics in Warsaw.