Processing DNA Tokens in Parallel Computing

Robert Nowak, Piotr Wąsiewicz,
Jan J. Mulawka
The Institute of Electronic Systems

Andrzej Płucienniczak
Institute of Biotechnology and Antibiotics

Abstract

In this paper a new technique of sending data between molecular processors is presented. The molecular processor is a processing data unit. Its computation results have to be sent to other units in the form of addressed messages - tokens. Necessary experiments were performed. All operations were implemented in DNA. DNA processors and tokens were specially designed DNA strings. Results of experiments prove our assumptions.

Keywords: **DNA computing, molecular processor, DNA tokens, molecular** *modus ponens*, **parallel data tokens processing**

1 Introduction

FOR over 50 years traditional electronic computers operating principles have remained almost the same like those prepared by Turing and von Neumann [1, 2]. Modern computers are very quick and work unfailingly, but do not compute massively in parallel connecting up to about 9000 processors. Other their faults are connected with their large dimensions about 10000 nm^3 per 1 bit and their great energy consumption, though smaller 1000 times than 50 years ago. Researchers and computer experts are convinced that some alternative technologies will appear [3, 4]. Quantum computing and molecular computing are potential candidates for such technologies, and are under extensive development [5].

To perform computing DNA molecules may be used. In DNA computing [6] information is stored in DNA molecules. An initial set of molecules usually contains *all* correct and wrong solutions e.g one vessel contains about 10^{20} DNA strings. DNA computing may be considered as a set of processing steps on DNA molecules for solving a specific problem according to a precisely defined procedure. A solution - extraction of correct strings - is reached by the exclusive use of genetic engineering operations on DNA such as hybridization, denaturation, ligation, PCR, etc.

DNA molecules are linear polymers built with chemical bonds from four building blocks - nucleotides denoted by symbols A, C, G and T as is depicted in Fig. 1. Since DNA polymers are composed of four nucleotides, they represent chains of symbols over 4-letter alphabet. Therefore DNA computing is adequate for processing symbols and logical structures [7–13], but also for general implementation of DNA computer [14–16] and especially for solving NP-complete problems [17–20] requiring large solution space scanning.

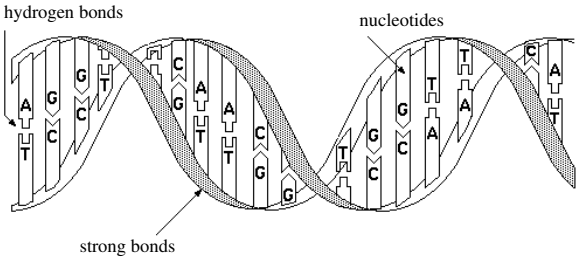

Fig. 1. DNA structure

Recently high-density oligonucleotide arrays so called DNA chips were developed as tools for sequencing by hybridization *(SBH)* [21]. It is possible to design and synthesize in situ on the support using light-directed solid phase combinatorial chemistry [22] the square inch high-density oligonucleotides arrays for monitoring the expression levels of nearly all (about 6500) yeast genes equalling about 14 Mb of information [23]. The progress in this area allows DNA integrated circuits to be basic devices for extremely miniaturized DNA computers [24].

Special microreactors are used in automatic data-flow systems [25, 26]. Microreactor consists of microchannel and reaction chamber systems intended for performing chemical reactions on nanoscale. Microreactor layers consists of microchannels, micropumps, microvalves made in silicon wafers. Microcapillaries join the reactor providing input and output chemical compounds.

In this paper we described and implemented a new methodology of processing data tokens sent or exchanged between molecular processors in parallel. This new technique is based on molecular implementation of the *modus ponens* rule represented by the molecular processor. From the input tokens the quite new output tokens are generated and can be used in further parallel computation. After per-

R. Nowak, P. Wąsiewicz and J.J. Mulawka are from the Institute of Electronic Systems, Warsaw University of Technology, Nowowiejska 15/19, 00-665 Warsaw, Poland, pwas@ieee.org , {pwas,jml}@ise.pw.edu.pl ; A. Płucienniczak is from Institute of Biotechnology and Antibiotics, Starościńska 5, 02-516 Warsaw, Poland, apl@iba.waw.pl

forming successfully done experiments it seems that this methodology paves a way towards powerful and massively parallel molecular computers.

2 Fundamental Operations on DNA

DNA molecules are directional polymers, due to the details of the biochemical structure and synthesis. Their beginning is denoted as 5' and end as 3'. Due to specific stereochemical interactions between A:T and C:G nucleotides DNA molecules can form antiparallel duplexes, provided that their sequence is complementary - allowing to form A:T and C:G pairs. Therefore in double stranded DNA the information is stored in both strands, in standard and complementary sequence.

Single or double DNA fragments are often called oligonucleotides or oligos, primers, strings and strands. In DNA computing a DNA *string* is represented by a sequence of four basic nucleotides and is usually described by letters A, T, G, C. It may exist as a separate DNA fragment or within a longer one e.g. a string a may be denoted by a sequence: $5'AGTC3'$ or may exist within a longer string $z = 5'AGAAGTCCTA3'$. A formal language may be created from DNA strings. The set of all single DNA strings over the alphabet $\Lambda = \{A, T, G, C\}$ is called the basic language of DNA computing and denoted by Λ^*. Up to now several DNA computing notation standards was worked out e.g. DNA-Pascal [27], splicing [28–31] and other [32–35]. Here we introduce our symbolic representation, which is useful for molecular binary operations.

Digits $5'$, $3'$ denoting *orientation* of a DNA string can be replaced with symbols $|\ >$. *The length* of the string a is denoted by: $|a|$, and its value is equal to a number of symbols forming the string a e.g. $|AGTC|$ has a length of four basic symbols: nucleotides. Using exemplary strings we can write:

$$a = |a >= 5'AGTC3'$$
$$b = |b >= |TCAGTCTAG >$$
$$z = |z >= |AGAAGTCCTA > \Leftrightarrow$$
$$\Leftrightarrow z = 5'AGA * a * CTA3'$$
$$s =< s| = 3'GATGACTGA5'$$
$$|a| = 4, |z| = 10$$

It should be noticed that a null string denoted by a symbol ε is a set with zero basic symbols. Thus $|\varepsilon| = 0$. In DNA computing the null string represents logical zero. *A right part* of the string a is described by a symbol $''$ in the upper, right index of the letter a: a'', and *a left part* of the string by a symbol $'$ in the upper, right index of the letter a: a'. If a number of string parts is greater than three, then in the upper, right letter index an ordinal number is placed e.g. the string a is divided into four parts: $a', a^{ii}, a^{iii}, a^{iv}$.

A string *complementary* to a is described by the same letter, but with an added symbol tilde ($\tilde{}$) this means $\tilde{a}$. Two complementary strings a and $\tilde{a}$ create after hybridization a double stranded string $\hat{a}$ made of complementary pairs $A = T, T = A, C \equiv G, G \equiv C$. Note that a string with an orientation $5' \rightarrow 3'$ is always an upper string or a single string, and a single string with an orientation $3' \rightarrow 5'$ should be underlined.

$$\hat{a} = \begin{matrix} |a > \\ < \tilde{a}| \end{matrix} = \left\lceil \begin{matrix} a \\ \underline{\tilde{a}} \end{matrix} \right\rfloor$$

To the described below DNA chip strings $\underline{a}, \underline{\tilde{b}}, \underline{\tilde{a}}$ are attached. They are underlined to mark their 3'5' orientation. Other strings $\tilde{a}, b, a$ are annealed to them and can be extracted together with the array. This linear representation of square arrays is quite conventional.

$$\begin{pmatrix} \tilde{a} & b & a \\ \underline{a} & \underline{\tilde{b}} & \underline{\tilde{a}} \end{pmatrix}$$

Operations on DNA oligos [36, 37] may be described in the following way:

1. *Hybridization or Renaturation* means connecting of single complementary DNA strings and forming double stranded molecules. This operation is caused by cooling down the test tube reaction solution and denoted by symbols $\mathrm{heat} \downarrow$.
2. *Denaturation* means disconnecting single complementary strings from double stranded DNA molecules and is caused by heating the test tube reaction solution. Usually this operation is connected with the operation of *mixing* the solution. It is denoted by $\mathrm{heat} \uparrow$.
3. *Cutting* of a double DNA string into two parts is performed in DNA computing with the help of enzymes. This means that a given string d may be digested by the enzyme in the presence of a hybridized complementary to d (at least in the neighbourhood of a place to cut) string denoted by a letter c. The enzyme with an ordinal number equal to 5 cuts the string d together with the string c what is described below.

$$\left\lceil \begin{matrix} - & {}_5d & - \\ \underline{\quad} & {}_5c & \end{matrix} \right\rfloor = \left\lceil \begin{matrix} - & d' & d'' & - \\ \underline{\quad} & c' & c'' & \end{matrix} \right\rfloor \Rightarrow$$
$$\Rightarrow \left\lceil \begin{matrix} - & d' & + & - & d'' & - \\ \underline{\quad} & c' & - & + & c'' & \end{matrix} \right\rfloor \Rightarrow$$
$$\Rightarrow \left\lceil \begin{matrix} - & d' & \\ \underline{\quad} & c' & + \end{matrix} \right\rfloor + \left\lceil \begin{matrix} + & d'' & - \\ & \underline{c''} & \end{matrix} \right\rfloor$$

A sign $+$ at the side of a DNA string describes a sticky end of it shorter than the nearest complementary oligo. A sign $-$ at the right side of the DNA string describes a sticky end of it longer than the nearest complementary string. The same signs at both ends of complementary strings mean that these strings form a double stranded oligo with blunt ends. Note that the sign $+$ may be additionally applied to mark a symbolic disjunction between two hybridized primers, and the sign $*$ to denote concatenation of

strings (after hybridization and ligation), and the sign $-$ to lengthen a string (of course, only in the equations). These rules are obligatory only within brackets $\lceil$ and $\rfloor$ or (and).

4. *Concatenation* of two strings is a string formed by placing the second string after the first string without any gap. In DNA computing joining of two strings is done during hybridization and ligation. They form together a longer single string. In order to concatenate two oligos a and b the complementary to them in the place of joint, hybridized *third one* is needed. Usually at least eight complementary pairs without a gap are necessary (four pairs for each joining string). The third string c is a concatenation of the oligo complementary to the first string right part a'' and the oligo complementary to the second string left part b'.

$$c = \widetilde{a''} * \widetilde{b'} =< TCAGAGTC|$$

Thus concatenation of two strings a and b in the presence of the complementary, hybridized to them third one c is denoted by:

$$ab = a * b \stackrel{*}{=} a + b \stackrel{c}{=} \{a, b\} \text{ or}$$
$$a + b \stackrel{*}{\Rightarrow} a * b$$
$$ab = |AGTCTCAGTCTAG >$$
$$|ab| = 13$$

The symbol $*$ means in this case the concatenation operation. The null string is the neutral element for concatenation this means $\varepsilon * a = a * \varepsilon = a$.

5. *Amplification (PCR)* encreases a number of double DNA strings chosen by specially designed primers two times in each cycle. The ends of these primers (square brackets) denote ends of amplified oligos. A number of PCR cycles is given in the upper, right corner of the right square bracket. If the number is unknown it is replaced by a sign \$. After tens of amplification cycles in the test tube there are millions of chosen DNA fragments copies, which are in the majority.

$$\text{heat} \downarrow;\ \hat{e} \approx \widehat{a[e]^{\$}b};\ \text{heat} \uparrow;$$

Given above amplification of double string can be described in another way as an algorithm:

$$\left\lceil \begin{array}{cccccccc} \tilde{a} & * & \tilde{e} & - & - & * & \tilde{b} & \\ [& & & & p_2]^{\$} & & & \\ & & [p_1 & & & &]^{\$} & \\ \underline{a} & * & - & - & e & * & \underline{b} & \end{array} \right\rfloor \Rightarrow$$

$$\Rightarrow \left\lceil \begin{array}{cccccc} & \tilde{e} & - & - & * & \tilde{b} \\ & [& & p_2]^{\$} & & \\ & [p_1 & &]^{\$} & & \\ \underline{a} & * & - & - & \underline{e} & \end{array} \right\rfloor \Rightarrow \left\lceil \begin{array}{c} \tilde{e} \\ \underline{e} \end{array} \right\rfloor;,$$

where three amplification cycles are presented, and additional primers p_1, p_2 are short oligos complementary to small parts of the given double string.

In the cycle of amplification single strings may be lengthened from its 3' end up to their complementary 5' end e.g.:

$$\text{heat} \downarrow; \left\lceil \begin{array}{cc} [w &]^{\$} \\ \underline{-} & \underline{v} \end{array} \right\rfloor \Rightarrow \left\lceil \begin{array}{c} \tilde{v} \\ \underline{v} \end{array} \right\rfloor; \text{heat} \uparrow;$$

$$\text{heat} \downarrow; \left\lceil \begin{array}{cc} - & x \\ \underline{[} & \underline{y]^{\$}} \end{array} \right\rfloor \Rightarrow \left\lceil \begin{array}{c} \tilde{x} \\ \underline{x} \end{array} \right\rfloor; \text{heat} \uparrow;$$

$$\text{heat} \downarrow; \left\lceil \begin{array}{c} \tilde{v} \\ \underline{v} \end{array} \right\rfloor \Leftarrow \left\lceil \left\lceil \begin{array}{ccc} + & w & \\ & \underline{z} & \underline{+} \end{array} \right\rfloor^{\$} \right\rfloor; \text{heat} \uparrow;$$

Every amplification described above is done in one cycle between cooling (heat $\downarrow$) and heating (heat $\uparrow$).

6. *Mixing* of DNA fragments enables their uniform distribution. It improves search for good hybridizations in the space of all possible ones.

7. *Extracting* of DNA fragments with specific sequences from other DNA strings can be performed in several ways e.g. with magnetic beads or with DNA chips.

3 DNA Tokens Addressing

The biochemical system of many molecules called molecular processors can process information. In this paper the mechanism of sending data between processors will be described. Data sending is realized with use of tokens which have addresses determining the target processor. In such system there is no need for shared memory implementations. Tokens transfer enables synchronization that is important in not only distributed, but also in massively parallel computation. A token is made from a DNA string which structure is depicted in Fig. 2.

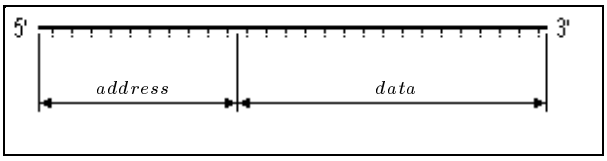

Fig. 2. Token structure

The token molecule is a single-stranded DNA string with two given sectors. At the 5' end there is an address area with fixed length. After the address sector there is the data area with arbitrary length.

Many molecular processors can be put into a single tube. Therefore there is a need for the addressing mechanism enabling the first processors to receive messages from the second ones. The method bases on selective complementary strings hybridization. Every processor has always the unique address sector DNA sequence defined especially for this processor.

An exemplary processor is set to do one simple instruction: after receiving the addressed to it token A it sends a token B to another processor. Without a token A it sends nothing. Processor operation can be described in the following way: if in the solution the token A appears, then

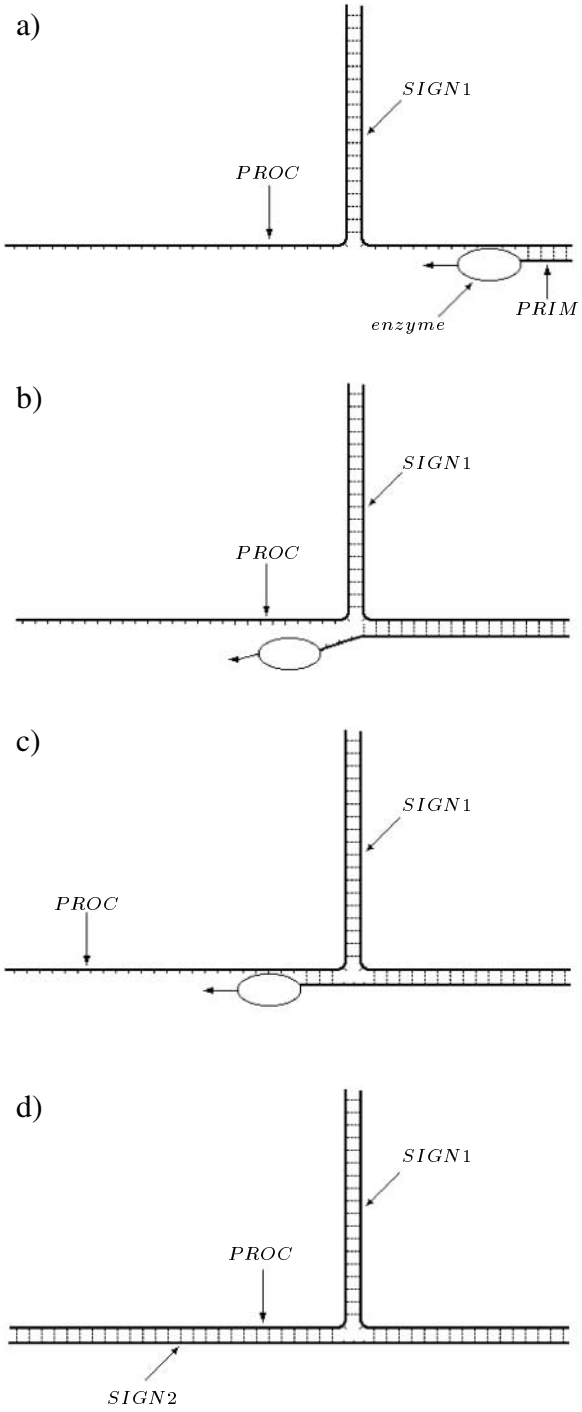

Fig. 3. The amplification operation of passing

a token B addressed to the appropriate processor should be generated, otherwise no output token ought to be generated. Fortunately, such a processor can be built from one DNA string and with the *Taq* DNA polymerase enzyme.

In experiments the following DNA strings were utilised: PROC - the processor DNA string, SIGN1 - the input token DNA string, PRIM - an additional primer. As an output token the DNA string named SIGN2 is generated. The primary DNA string named PROC consists of three sectors: the complementary first one to the address sector of token A, the special second one only with nucleotides T and the complementary third one to the address sector of token B. The mentioned earlier enzyme extends the 3' end of the complementary to the primary DNA string primer and creates the complementary DNA string SIGN2. The used here unusual attribute of this enzyme allows the complementary string extending only with nucleotides A usually beyond the 5' end of the primary DNA string, but in this case after reaching the DNA double-stranded joint. The generated nucleotide A sector will be complementary to the filled with nucleotides T fragment of the PROC string placed between the address sectors. The passing operation is illustrated in Fig. 3.

During first steps the SIGN1 molecule hybridizes to the PROC molecule, which address sector is complementary to it as is seen in Fig. 3a. The PRIM primer also attaches to the SIGN1 molecule to enable its complementary DNA string creating. During amplification with *Taq* polymerase the PRIM string is extended and changed into the first part of SIGN2 string. The enzyme can not break hydrogen bonds, so when it reaches the beginning of the double-stranded address sector of SIGN1, it starts to extend the SIGN2 with nucleotides A as is depicted in Fig. 3b. After some time the SIGN2 molecule attaches to the filled with T fragment of the PROC string (Fig. 3c). *Taq* polymerase continues extending of the SIGN2 string adding to it the complementary to the PROC molecule DNA fragment as is seen in Fig. 3d. If in the solution the token A does not appear, then the PRIM molecule will not attach itself to this token, so no output token B will be generated.

4 Experimental Results

The experiment was implemented in genetic engineering laboratory and consists of the following steps:

- passing reaction - generating a molecule with sequences complementary to both input DNA strings as is shown in Fig. 3,
- amplification reaction,
- detection with process of electrophoresis - initial estimation of experiment results,
- process of sequencing of generated molecule.

TABLE I
Oligonucleotides sequences

Name	[bp]	Sequence of nucleotides
SIGN1	38	5' - CGC CCT ACC CAT CCC TTT TTT GAC ACA CAT TCC ACA GC 3'
PROC	62	5' - TCG AGC AGA CAT GAT AAG ATA AAT GGA GAA CAT CAC ATC AGG TTT TTG GGA TGG GTA GGG CG - 3'
PRIM1	32	5' - AAA AGC TTG GTA CCG CTG TGG AAT GTG TGT CA - 3'
PRIM2	27	5' - GGG GAA TTC TCG AGC AGA GAT AAG ATA - 3'

TABLE II
DNA strings concentration

Name	Concentration	liquid	amount
SIGN1	3 pM/μl		5 pM
PROC	3 pM/μl		5 pM
PRIM1	12 pM/μl	4 μl	48 pM
dNTP	3 nM/μl	3 μl	9 nM
buffer	10$\times$	5 μl	
Taq		1 μl	
H_2O		the rest	
SUM		50 μl	

Oligonucleotides sequences was introduced in Tab. I. In the passing reaction substances from Tab. II were used. The reaction proceeds according to plan in Tab. III. All necessary compounds together with the buffer (20 mM Tris-HCl, 50 mM KCl, 2.0 mM $MgCl_2$, 0.01% gelatine, pH 8.6.), primers PRIM1, PRIM2 were utilized in PCR reaction, but only SIGN2 molecules were amplified exponentially. Electrophoretogram in Fig. 4 shows bands produced by resolving PCR products in acrylamid gel. Standard marker was added to lanes A, E, I, L. In lanes B, C, D were put 1 min reaction products (fourth step in Tab. III) and in lanes F, G, H - 2 min reaction products, in lanes J, K - 4 min reaction products. As is seen, results do not depend on reaction time. In lanes B, F were put 27 cycle products, in lanes C, G, J - 31 cycle products, in lanes D, H, K - 35 cycle products. The final reaction product SIGN2 has length about 100 bp.

In order to increase reaction efficiency and in the consequence generating correct DNA structures, temperature was lowered up to 47^{o}C (third step in Tab. III). Standard marker was put in lanes A, D, I. Lanes B, C, E, F, G, H contains reaction products. After 30 PCR cycles distinct

TABLE III
Passing operation steps

No	[^{o}C]	time	description
1	95 ^{o}C	2 min	denaturation
2	48 ^{o}C	20 min	initial hybridization
3	48 ^{o}C	-	adding *Taq* polymerase
4	48 ^{o}C	2 min	passing reaction (1,2 or 4 min)
5	0 ^{o}C	-	reaction stopping

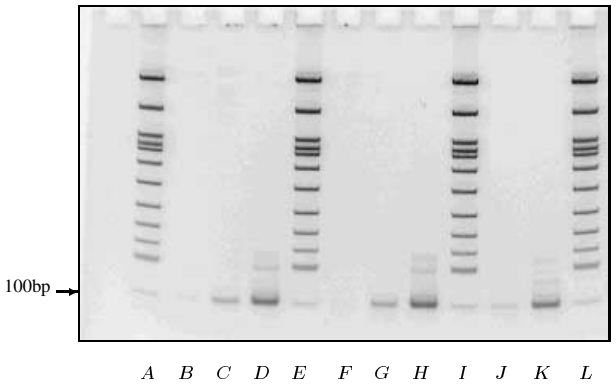

Fig. 4. Electrophoretogram of passing reaction products

bands were obtained as is depicted in Fig. 5. Some bands has about 100 bp, other ones - about 70 bp, some ones - 30 bp. The shortest ones are primers. The longest ones were put in a sequencing machine.

Result is described in Tab. IV and proves our assumptions that polymerase built the SIGN2 DNA string based on two input molecules PROC and SIGN1 as is seen in Fig. 6. The SIGN2 beginning is complementary to the SIGN1 molecule. Desired passing follows 18 SIGN2 nucleotides counted from its 5' end. Further part of the SIGN2 molecule is complementary to the PROC string. This jump executed during amplification is denoted by

TABLE IV
SIGN2 sequence

Name	[bp]	Sequence of nucleotides
SIGN2	104	5' - AA AAG CTT GGT ACC GCT GTG GAA TGT GTG TCA CGC CCT ACC CAT CCC AAA AAC CTG ATG TGA TGT TCT CCA TTT ATC TTA TCA TGT CTG CTC GAG AAT TCC CCC - 3'

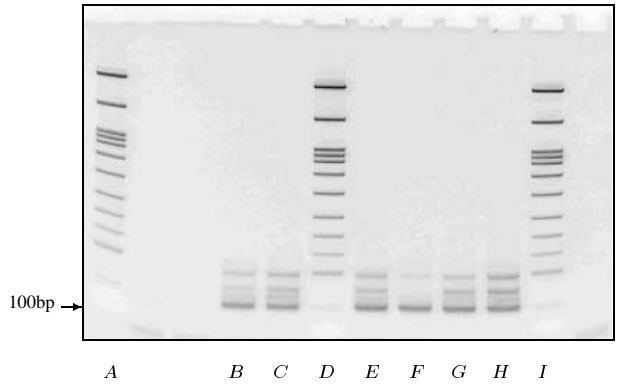

Fig. 5. Electrophoretogram of lowered temperature reaction products

the dashed line. It is not really probable that the reaction proceeds in the same way. It is suspected that more sophisticated DNA structures are created before amplification [38–40]. After repeating the experiment several times under the same conditions it was sure that the resulting products SIGN2 had very similar sequences and it was no change in the reaction course and the product structure. It is very probable that DNA strands with more proper sequences [41] can produce more correct output DNA string (without the address sector of PROC). Any way polymerase ability of passing (by jumping over double-stranded joints) was proved here.

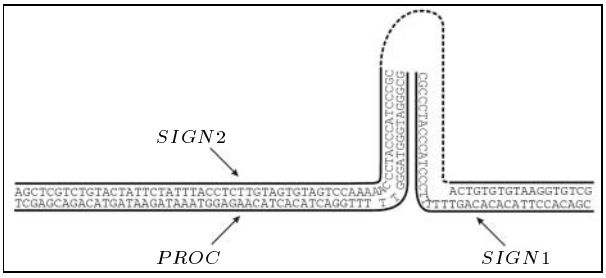

Fig. 6. Input and result sequences

5 DNA System Description

Implemented in the experiment DNA token system is denoted before amplification by

$$\left[\begin{array}{cc} S_1' & -^{(1)} \\ \underline{P_A} & \underline{-_{(2)}} \end{array}\right]$$
$$\left[\begin{array}{ccccc} P_D & * & \overset{(2)(1)}{+\ +} & * & S_1\text{''} \\ \underline{\lceil} & & & & \underline{p\rceil^{\$}} \end{array}\right],$$

and after amplification by

$$\left[\begin{array}{cc} S_1' & -^{(1)} \\ \underline{P_A} & \underline{-_{(2)}} \end{array}\right]$$
$$\left[\begin{array}{ccccc} P_D & * & \overset{(2)(1)}{+\ +} & * & S_1\text{''} \\ \underline{S_2\text{''}} & \underline{-} & \underline{*} & \underline{-} & \underline{S_2'} \end{array}\right],$$

where PROC$=\ P_D + P_A$ and P_A is the address sector; SIGN1$= S_1' + S_1$'' and S_1' is the address sector; SIGN2$= S_2' + S_2$''.

Given above mechanism can be utilised in many specific DNA structures describing processors and tokens. For example the following command: "get data D from the A processor and send it to the B processor" is denoted after execution by

$$\left[\begin{array}{cc} A & -^{(1)} \\ \underline{\tilde{A}} & \underline{-_{(2)}} \end{array}\right]$$
$$\left[\begin{array}{ccccc} \tilde{D} & * & \overset{(2)(1)}{+\ +} & * & B \\ \underline{D} & \underline{-} & \underline{*} & \underline{-} & \underline{\tilde{B}} \end{array}\right].$$

Another exemplary command: "send data D_1 to the B processor, get there data D_2 and send together with D_1 to the A processor" is described after execution by

$$\left[\begin{array}{cc} \tilde{B} & -^{(1)} \\ \underline{B} & \underline{-_{(2)}} \end{array}\right]$$
$$\left[\begin{array}{ccccccc} A & * & D_2 & * & \overset{(2)(1)}{+\ +} & * & D_1 \\ \underline{\tilde{A}} & \underline{*} & \underline{\tilde{D}_2} & \underline{-} & \underline{*} & \underline{-} & \underline{\tilde{D}_1} \end{array}\right].$$

It is worth mentioning that it depends on a chosen primer whether D_2 or $D_2 * D_1$ will be amplified together with the part D_3 of the A processor as is seen in the given below descriptions:

$$\left[\begin{array}{cc} A & -^{(1)} \\ \underline{\tilde{A}} & \underline{-_{(2)}} \end{array}\right]$$
$$\left[\begin{array}{ccccccc} D_3 & * & \overset{(2)(1)}{+\ +} & * & D_2 & * & D_1 \\ \underline{\lceil} & & & & \underline{+} & \underline{p_1\rceil^{\$}} & \end{array}\right],$$
$$\left[\begin{array}{cc} A & -^{(1)} \\ \underline{\tilde{A}} & \underline{-_{(2)}} \end{array}\right]$$
$$\left[\begin{array}{ccccccc} D_3 & * & \overset{(2)(1)}{+\ +} & * & D_2 & * & D_1 \quad - \\ \underline{\lceil} & & & & & & \underline{p_2\rceil^{\$}} \end{array}\right].$$

It is very important for every data sector D to have at the 3' end two special sequences: the first one for the reading token amplification primer, the second one filled only with T nucleotides for passing polymerase reaction DNA string.

It is no doubt that well-known *modus ponens* inference rule [42, 43] denoted by

$$\frac{\alpha, \alpha \Rightarrow \beta}{\beta}$$

can be implemented with use of passing mechanism, where first α is the name of $\tilde{\beta}' * \tilde{\alpha}$ similar to the PROC processor, $\alpha \Rightarrow \beta$ is just the name of $\alpha * \tilde{\beta}$" the input token name like SIGN1 and β means the output token similar to SIGN2:

$$\left[\begin{array}{cc} \alpha & -^{(1)} \\ \underline{\tilde{\alpha}} & \underline{-_{(2)}} \end{array}\right]$$

$$\left[\begin{array}{ccccc} \tilde{\beta}' & * & \overset{(2)(1)}{+\ +} & * & \tilde{\beta}" \\ - & - & \beta & - & - \end{array}\right].$$

Thus, DNA passing token systems can also implement inference systems [8, 45, 46] on molecular level, which are simplified versions of future much more larger expert systems with capacity greater than that one of human brain [14].

6 Conclusions

In this paper the mechanism of sending data between two molecular processors with use of DNA tokens was described and implemented during experiments in the genetic engineering laboratory. This method allows sending data to more than one processor in the so called multicast sending (to many, but not all processors) and broadcast sending (to all processors) known in distributed net computing using e.g. IP (Internet Protocol). During solution cooling the probability of attaching tokens to the processors with different address sequence of nucleotides increases. With appriopriate design of these sequences only strictly known processor groups can receive tokens of the same type.

Another way of sending tokens in broadcast mode requires utilising of artificially synthesized nucleotides N complementary to all typical nucleotides A,T,G,C. In this case first tokens will hybridize only to arbitrary (during design) chosen processors, second ones to groups of processors and third ones to all processors. For example if there are processors with adresses 1) AATT, 2) AAGG, 3) GCAA, 4) TATA, then the token with the address TTAA attaches to the processor number 1, the token with the address TTNN - to the processor number 1 and processor number 2, at last the token with the address NNNN - to all processors. Thus, implementation of address sequences with nucleotides N allows for different sending ways at the same temperature. This is the great advantage. Unfortunately, not all enzymes can digest such new type sequences. Therefore the only problem is connected with proper enzyme selection.

It is expected that the described molecular processor with changing states [44] during computation will be invented in the near future and will consist of more than one DNA molecule. It is obvious that the real advantage of such systems appears during massively parallel computation. Million molecular processors could be attached with their 3' ends to a typical DNA chip or magnetic beads placed in a microreactor. Thus, it would be the first prototype of future powerful molecular computers.

Acknowledgments

This work was supported by the Polish State Committee for Scientific Research, through the KBN grant number 8T11F00816. This paper was processed with LaTeX 2_ε.

References

[1] N.N. Biswas, *Logic Design Theory*, Prentice-Hall International Editions, USA (1993).

[2] F.J. Hill and G.P. Peterson, *Switching Theory and Logical Design*, Wiley, New York (1974).

[3] J.R. Heath, P.J. Kuekes, G.S. Snider, R.S. Williams, A Defect-Tolerant Computer Architecture: Opportunities for Nanotechnology, *Science* **280** (1999) 1716-3721.

[4] C.P. Collier, E.W. Wong, M. Belohradsky, F.M. Raymo, J.F. Stoddart, P.J. Kuekues, R.S. Williams, J.R. Heath, Electronically Configurable Molecular-Based Logic Gates, *Science* **285** (1999) 391-393.

[5] M.S. Malone, Beyond semiconductors, *The Microprocessor: A Biography*, Springer-Verlag (1995).

[6] *The Bibliography of Molecular Computation and Splicing Systems*, at http://liinwww.ira.uka.de/bibliography/Misc/dna.html

[7] P. Wąsiewicz, P. Borsuk, J.J. Mulawka, P. Węgleński, Implementation of Data Flow Logical Operations via Self-Assembly of DNA, *Lect. Not. Comp. Sci.* **1586** (1999) 174–382.

[8] P. Wąsiewicz, T. Janczak, J.J. Mulawka, A. Płucienniczak, The Inference Based on Molecular Computing, *Cybernetics and Systems: An International Journal*, Taylor & Francis, vol. **31/3** (2000) 283-315.

[9] P. Wąsiewicz, A. Malinowski, R. Nowak, J.J Mulawka, P. Borsuk, P. Węgleński, A. Płucienniczak, DNA Computing: Implementation of Data Flow Logical Operations, *Future Generation Computer Systems Elsevier Journal*, **17/4** (2001) 361–378.

[10] M. Amos and P.E. Dunne, *DNA Simulation of Boolean Circuits*, Technical Report CTAG-37009, Department of Computer Science, University of Liverpool UK (1997).

[11] T. Leete, J. Klein, H. Rubin, Bit Operations Using a DNA Template, *Proc. of the Third DIMACS Workshop on DNA-based Computers*, Philadephia, USA (1997) 159–366.

[12] M. Ogihara and A. Ray, *Simulating Boolean Circuits On a DNA Computer*, TR 631, University of Rochester, Computer Science Department (1996).

[13] M. Ogihara and A. Ray, *The Minimum DNA Computation Model and Its Computational Power*, TR 672, University of Rochester, Singapore (1998).

[14] E.B. Baum, Building an associative memory vastly larger than the brain, *Science* **268** (1995) 583–585.

[15] P. Wasiewicz et al., Adding Numbers with DNA, *Proc. 2000 IEEE International Conference on Systems, Man & Cybernetics - SMC2000*, Nashville, USA (2000) 265-270.

[16] P. Wąsiewicz, J.J. Mulawka, Molecular Genetic Programming, *Journal of Soft Computing*, Springer, in print (2001).

[17] L.M Adleman, Molecular computation of solutions to combinatorial problems, *Science* **266** (1994) 1021–3024.

[18] R.J. Lipton, *DNA* Solution of Hard Computational Problems, *Science* **268** (1995) 542–545.

[19] L.M. Adleman, P.W.K. Rothemund, S. Roweis, E. Winfree, On applying molecular computation to the data encryption standard, *Journal of Computational Biology*, **6(1)** (1999) 53-63.

[20] M. Amos, *DNA Computation*, Ph.D. Thesis, Department of Computer Science, University of Warwick UK (1997).

[21] D. Shalon, S.J. Smith, P.O. Brown, A DNA Microarray System for

Analyzing Complex DNA Samples Using Two-color Fluorescent Probe Hybridization, *Genome Research* **6** (1996) 639-645.

[22] S.P.A. Fodor, et al., Light-Directed, Spatially Addressable Parallel Chemical Synthesis, *Science* **251** (1991) 767–773.

[23] L. Wodicka, et al., Genome-wide expression monitoring in saccharomyces cerevisiae, *Nature Biotechnology* **15** (1997) 1359–3367.

[24] Q. Liu, A. Frutos, L. Wang, A.J. Thiel, S.D. Gillmor, T. Strother, A.E. Condon, R.M. Corn, M.G. Lagally, L.M. Smith, Progress Toward Demonstration of a Surface Based DNA Computation: a One Word Approach to Solve a Model Satisfiability Problem, *Proc. of the Fourth DIMACS Workshop on DNA-based Computers*, Pennsylvania, USA (1998) 15–25.

[25] J.S. McCaskill, T. Maeke, U. Gemm, L. Schulte, and U. Tangen, NGEN: A massively parallel reconfigurable computer for biological simulation: towards a self-organizing computer, *Lect. Not. Comp. Sci.* **1259** (1997) 260–276.

[26] J.S. McCaskill, *Open Flow Microreactors*, Technical report, IMB Company, Germany (1997).

[27] D. Rooß, K.W. Wagner, *On the power of DNA-computers*, Technical reports, University of Wurzburg (1995, 1996).

[28] T. Head, Formal language theory and DNA: an analysis of the generative capacity of recombinant behaviors, *Bulletin of Mathematical Biology* **49** (1987) 737–759.

[29] E. Csuhaj-Varjú, L. Kari, G. Pun, Test tube distributed systems based on splicing, *Computers and AI* **15(2–3)** (1996) 211–232.

[30] T. Head, G. Paun, D. Pixton, Language theory and molecular genetics. Generative mechanisms suggested by DNA recombination, chapter 7 in vol. 2 of G. Rozenberg, A. Salomaa, eds., *Handbook of Formal Languages* 3 volumes, Springer-Verlag, Heidelberg (1997).

[31] G. Paun, G. Rozenberg, A. Salomaa, *DNA Computing - New Computing Paradigms*, Springer-Verlag Berlin Heidelberg (1998).

[32] Z. Li, Algebraic properties of DNA operations, *Proc. of the Fourth DIMACS Workshop on DNA-based Computers*, Pennsylvania, USA (1998) 57–70.

[33] V. Gupta, S. Parthasarathy and M. Zaki, Arithmetic and Logic Operations with DNA, *Proc. of the Third DIMACS Workshop on DNA-based computers*, Philadelphia (1997) 212–220.

[34] V. Mihalache, Prolog approach to dna computing, *Proc. of the IEEE International Conference on Evolutionary Computation (ICEC97), Special Session on DNA-based Computation* (1997).

[35] S. Roweis, E. Winfree, R. Burgoyne, N. Chelyapov, M. Goodman, P. Rothemund, L. Adleman, A Sticker-Based Model for DNA Computation, *Journal of Computational Biology*, **5(4)** (1998) 615-629.

[36] F. Ausubel et al., K. Struhl, *Short Protocols in Molecular Biology. A Compendium of Methods from Current Protocols in Molecular Biology*, Third Edition, Wiley & Sons (1995).

[37] J. Sambrook, E.F. Fritsch, T. Maniatis, *Molecular Cloning. A Laboratory Manual.*, Second Edition, Cold Spring Harbor Laboratory Press (1989).

[38] K. Langohr, *Sources of Error in DNA Computation*, University of Western Ontario (1997).

[39] S. Roweis, E. Winfree, On the reduction of errors in DNA computation, *Journal of Computational Biology*, **6(1)** (1999) 65-75.

[40] R. Lipton, D. Boneh, C. Dunworth, *Making DNA Computers Error Resistant*, Princeton University (1996).

[41] J.J. Mulawka, P. Wąsiewicz, K. Piętak, Virus-enhanced genetic algorithms inspired by DNA computing, *Lect. Not. Art. Int. - Subseries LNCS* **1609** (1999) 527–537.

[42] Pederson K., *Expert System Programming*, Wiley, New York (1989).

[43] Mulawka J.J., *Expert Systems (in Polish)*, WNT, Warsaw (1996).

[44] K. Sakamoto, D. Kiga, K. Komiya, H. Gouzu, S. Yokoyama, S. Ikeda, H. Sugiyama, M. Hagiya, State Transitions by Molecules, *Proc. of the Fourth DIMACS Workshop on DNA-based Computers*, Pennsylvania, USA (1998) 87–300.

[45] J.J. Mulawka, P. Borsuk, P. Węgleński, Implementation of the Inference Engine Based on Molecular Computing Technique, *Proc. IEEE Int. Conf. Evolutionary Computation (ICEC'98)*, Anchorage (1998) 493–498.

[46] P. Wąsiewicz, T. Janczak, J.J. Mulawka, A. Płucienniczak, The Inference Via DNA Computing, *Proc. Congress on Evolutionary Computation (CEC'99)*, **2**, Washington, USA (1999) 988-393.

Author Biographies

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

Piotr Wąsiewicz graduated from the Warsaw University of Technology in automatics in 1996 and obtained in 2000 a Ph.D. degree 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

Prof. Jan J. Mulawka got the 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

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 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 an author of many patents and papers. Recently he is the Vice-director of the Institute of Biotechnology and Antibiotics in Warsaw.