DNA-based parallel computation of simple arithmetic

Hubert Hug
Universitäts-Kinderklinik Ulm
Prittwitzstrasse 43
D-89075 Ulm, Germany

Rainer Schuler
Abt. Theoretische Informatik
Universität Ulm
D-89069 Ulm, Germany

Abstract

We propose a model for representing and manipulating binary numbers on a DNA chip which allows parallel execution of simple arithmetic. As an example we describe how addition of large binary numbers can be done by using a DNA chip. The number of steps is independent of the size (bits) of the numbers. However, the time for some biochemical reactions is still large, and increases with the size of the sequences to be assembled.

1 Introduction

Biological macromolecules can be used for storing information in a new kind of computers and biochemical reactions, like nucleic acid hybridization and enzyme reactions, can be used to solve algorithmical problems [Adl94]. Since a vast number of biochemical reactions can take place at many molecules simultaneously, parallel computations involving millions of operations seem possible. Methods for solving several well known NP-complete problems have been proposed, and have been performed on small examples in many cases [Adl94, Adl98, LWF+00] [SGK+00, OKLL97, FCLL00]. Moreover, based on certain biochemical reactions, formal models capturing the power of DNA-computing have been developed [Lip95].

Many arithmetical operations, like addition and multiplication of binary numbers can be performed in parallel on classical hardware. In this paper we consider the implementation of simple arithmetic operations in parallel, using addressable DNA-chips.

The addition of (small) binary numbers with DNA-molecules has been done by Guarnieri *et al.* [GFB96]. However the procedure is sequential and does not use the power of DNA-computing for parallelization. Other approaches are based on selecting the correct result from a set of all possible values [Ata00, Fri00, QL98] [GPZ97, OR96, Rei97].

We propose a way of representing binary numbers on DNA-chips such that operations, which allow to initialize and manipulate numbers, can be performed with standard methods from gene technology. The methods used here have already been performed in practice by

others. In particular, the use of a restriction enzyme to cut single stranded DNA which builds hairpin loops has been used in [SGK+00].

We show that it is possible to read, write and manipulate numbers. For example, we describe how to

- store a number on the chip,
- read the number on the chip,
- add a (second) number to the number on the chip,

Each of the above operations can be performed in parallel (for each bit) with a constant number of steps for each bit of the numbers.

2 The model

We use addressable DNA-chips for storing and manipulating large binary numbers. A standard tool used in DNA computing is hybridization of reverse complementary sequences. To minimize errors we assume that different sequences are chosen in such a way, that cross hybridization (i.e. the hybridization of sequences which are not reverse complementary) is minimal [FLT+97]. We use a restriction enzyme R_A to cut a double stranded DNA sequence. The restriction enzyme and the recognition site, denoted by A, is also chosen such that cross hybridization is minimal. Recall that a restriction site is a palindromic sequence of typically 4 or 6 bp length.

2.1 Representing numbers

A number t is represented by a binary string $t_{n-1} \cdots t_1 t_0$, where $t_i \in \{0, 1\}$, such that $t = \sum_{i=0}^{n-1} t_i \cdot 2^i$. The number is stored bitwise, each bit is represented by specific DNA-sequences which are attached at a particular position (spot) on the chip via its 3' end (e.g. from ThermoInteractiva). The sequences for the i-th bit t_i consist of the sequences p_i, A, and t_i (see Figure 1). The sequence p_i indicates the position i and is specific for each i. Sequence A is the recognition site of a restriction enzyme R_A and will be cut in the presence of R_A if paired with its complementary sequence. The sequence t_i is either a sequence indicating value 0 or a sequence indicating value 1 (depending on whether the i-th bit t_i is 0 or 1). We note here that for all bits the same sequences are used to indicate the values 0 and 1. In order to minimize cross hybridization, sequence selection can be done as described by Frutos et al. [FLT+97].

2.2 Basic operations

Let us first recall how to read a sequences from a chip. To do this we use reverse complementary oligonucleotides for 0 and 1 which are marked with different fluorescent dyes, say 0 with red (e.g. Cy3) and 1 with green (e.g. Cy5) [IER+99]. These sequences are put

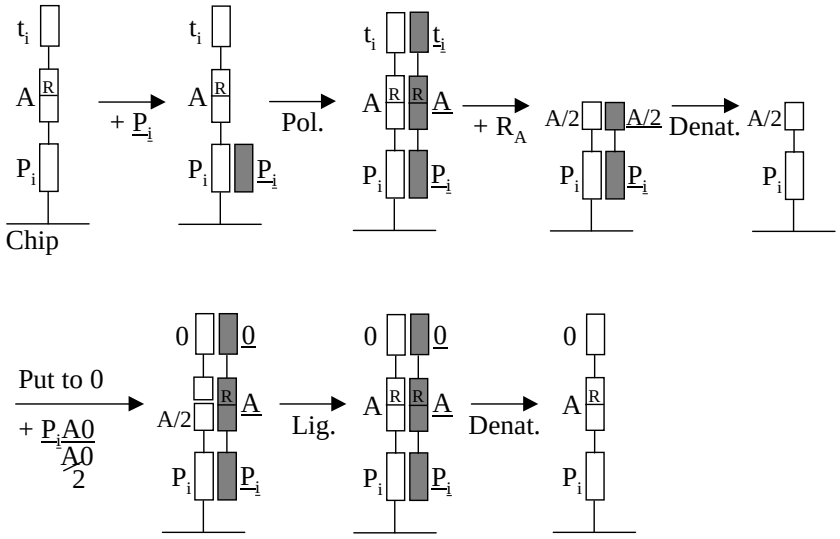

Figure 1: Representing numbers with DNA on an addressable chip surface. p_i carries the positional information, A contains the recognition site for a restriction enzyme R_A, and t_i codes either 1 or 0. Reverse complementary sequences are underlined. Abbr.: Pol., DNA polymerase; Denat., denaturation; Lig., ligase.

on the chip, where the reverse complementary sequences bind to the 0/1 sequences on the chip. Additional sequences are washed away, and the chip is read using a chip reader. All positions (bits) equal to 0 will be marked red and all positions (bits) equal to 1 will be marked green.

We consider the addition of two numbers a and b using a small set of basic operations. In a first step, number a is stored on the chip. In a second step, number b is added bitwise to t. (That is bitwise addition modulo 2 of a_i and b_i is computed on the chip). In a third step the carry bits c are added to the number on the chip. We will further show that the carry bit can be computed in parallel using the method of Adleman [Adl94].

Let I_a denote the positions such that $a_i = 1$. Similarly let I_b and I_c denote the positions such that $b_i = 1$ and $c_i = 1$ respectively. The the addition of a and b can be done as follows.

For all $i \in I_a$ do select i and set $t_i = 1$
For all $i \notin I_a$ do select i and set $t_i = 0$
For all $i \in I_b$ do select i set $t_i = t_i \oplus 1$
For all $i \in I_c$ do select i set $t_i = t_i \oplus 1$

The first two steps store a number a on the chip. The third and fourth step perform bitwise addition modulo 2.

2.3 Writing numbers on the chip

Writing numbers on a chip is done in two parts. In the first part all bits which are equal to 0 are set, and in the second part all bits which are equal to 1 are set. To do this we prepare

- for all bits i which will be set to 0, reverse complementary sequences $\underline{p_i}$, $\underline{p_i A 0}$ and sequences $0A/2$, and
- for all bits k which will be set to 1, reverse complementary sequences $\underline{p_k}$, $\underline{p_k A 1}$ and sequences $1A/2$,

where $A/2$ is the sequence A cut by the restriction enzyme R_A. To set bits to 0, two steps are performed. In a first step the values of the respective positions are deleted, i.e. all positions $i \notin I_a$ are selected. In a second step all deleted positions are extended to 0, i.e. all selected positions are set to 0.

Select positions i: To delete the values at positions i we use reverse complementary sequences $\underline{p_i}$ to bind to the corresponding sequences p_i on the chip. Using DNA polymerase and free nucleotides the complementary sequence is extended to yield the double stranded DNA sequence $p_i A t_i$, where t_i is 0 or 1 depending on the value at position i. Using the restriction enzyme R_A, the sequence is cut at position $p_i A$ (see Figure 1).

Set $t_i = 0$ **for all selected positions** i: To set the bit at the selected positions to 0, we hybridize oligonucleotides consisting of the reverse complementary sequences $\underline{p_i A 0}$ and sequences $0A/2$. Sequence $0A/2$ is able to bind to its reverse complement, thus building a sequence $\underline{p_i A 0}$ which is double-stranded in the $0A/2$ region. (We assume that $0A/2$ is in excess such that all $\underline{p_i A 0}$ are double-stranded in the $0A/2$ region). The sequences are put on the chip and the single stranded part of the $\underline{p_i A 0}$ sequence which is complementary to $p_i A/2$ will bind to the sequence $p_i A/2$ of a selected position. Using DNA ligase the free neighboring ends in the resulting double stranded DNA sequence can be closed (see Figure 1).

To set bits to 1 is done similarly by using the reverse complementary sequences $\underline{p_k}$, $\underline{p_k A 1}$ and sequences $1A/2$.

2.4 Bitwise addition modulo 2

As second operation we consider addition modulo 2. Assume that a number t is stored on the surface of the chip and a number b will be added bitwise modulo 2. That is, for all i such that the i-th bit of b is equal to 1, the values of t_i have to be flipped. To achieve this we use the fact that reverse complementary sequences in a single stranded DNA molecule form hairpin loops.

The bit flipping is done in two parts. In the first part the bits t_i equal to 0 are set to 1, in the second part the bits t_i equal to 1 are set to 0 (see Figure 2). To do this we prepare

- for all bits k such that $b_k = 0$, reverse complementary sequences $\underline{p_i}$, and

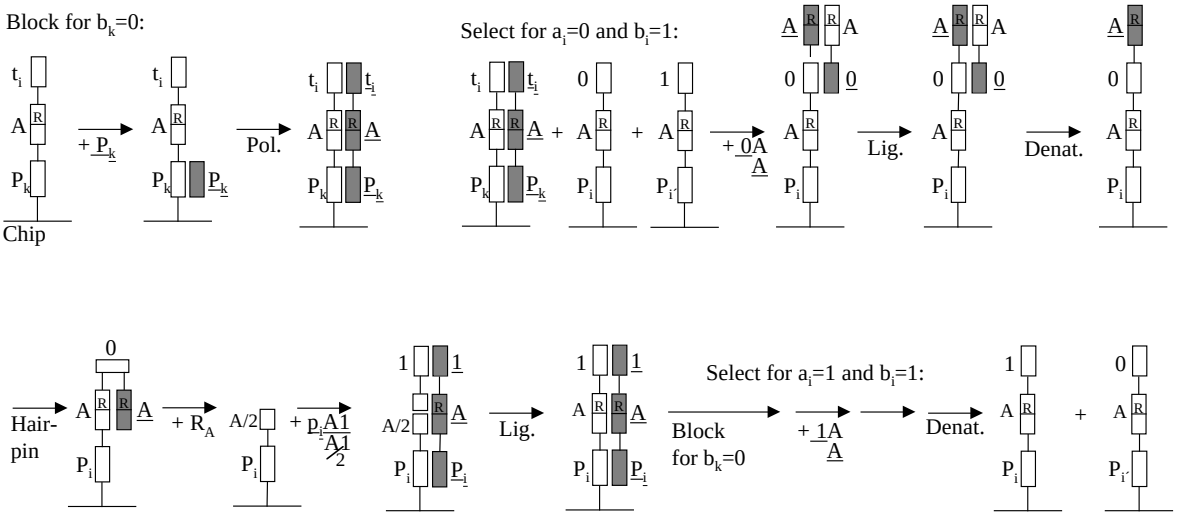

Figure 2: Adding modulo 2 on a chip surface. Abbr. are as for Figure 1.

- for all bits i such that $b_i = 1$, sequences $\underline{0}A$, $\underline{1}A$ and $\underline{A}$. Note that $\underline{0}A$ and $\underline{1}A$ consist of reverse complementary sequences of 0 or 1 followed by sequence A.

In general, reverse complementary sequences are underlined. To set t_i from 0 to 1 is done as follows.

1. All positions k such that b_k is equal to 0 are blocked from further reactions.
2. All remaining positions i such that t_i is equal to 0 are selected. In this reaction the blocking of the positions k such that b_k is equal to 0 is released. Selected positions are set to 1 and blocked from further reactions.
3. All positions k such that b_k is equal to 0 are blocked from further reactions. All remaining positions i such that t_i is equal to 1 are selected.
4. Selected positions are set to 0.

We describe the procedure in detail. In the first step, reverse complementary oligonucleotides $\underline{p_k}$, for all k such that $b_k = 0$, are added to the chip and will bind to the sequences $p_k A t_k$. The 3' end of $\underline{p_k}$ will be elongated by adding DNA polymerase and free nucleotides (see Figure 2).

For the next step, partially reverse complementary oligonucleotides $\underline{0}A$ and $\underline{A}$ are hybridized and added to the chip. Now oligonucleotides $\underline{0}A$ which are double stranded on the A part, will hybridize to 0 at all positions $p_i A 0$ which are not blocked by step 1. The DNA ligase is used to attach $\underline{A}$ to 0. Then $A\underline{0}$ is removed by denaturation. Note that this also removes the blocking of positions p_k performed in the first step above.

Now $\underline{A}$ can hybridize intramolecularly with sequence A to form a hairpin loop. This double stranded part of the DNA sequence can be cut using restriction enzyme R_A. Now

all positions p_i such that $b_i = 1$ and $t_i = 0$ are selected and can be set to 1 as described in paragraph "Set $t_i = 0$ for all selected positions i" above.

3 Computing the carry bits

Our construction follows the method of Adleman to build all possible paths in a graph. We prepare nodes and edges as follows. Each node specifies whether position i generates, propagates, or deletes a carry. An edge connects a node $i + 1$ to node i, if position i propagates a carry.

To build the nodes and edges we prepare

- for all bits i such that $a_i = 1$, reverse complementary sequences $\underline{p_{i+1}A1_i}$,
  for all bits i such that $a_i = 0$, reverse complementary sequences $\underline{p_{i+1}A0_i}$.
  The 5' ends of the oligonucleotides are biotinylated.

- for all bits i such that $b_i = 1$, sequences $0_i q_i$ and $1_i 1$ (i.e. sequences 3'-$0_i q_i$-5' and 3'-1_i1-5'),
  for all bits i such that $b_i = 0$, sequences $0_i 0$ and $1_i q_i$ (i.e. sequences 3'-0_i0-5' and 3'-$1_i q_i$-5').

- for all i, sequences $q_i p_i$ (i.e. sequences $3' - q_i p_i - 5'$).

- reverse complimentary sequences $\underline{p_0 A0}$. The 5' end is biotinylated.

First we compute the sequences for the nodes, indicating whether a carry is generated, propagated or deleted. For all i, we select the respective oligonucleotides for a_i and b_i as defined above. The oligonucleotides are mixed under conditions that complementary sequences hybridize. Double strands are extended by DNA polymerase, then the double strands are denatured. Single strands containing a biotin are coupled with paramagnetic streptavidine molecules (e.g. from Promega). They can now be separated by using a magnet from the rest of the DNA molecules in the solution [Adl94]. Observe, that for all positions i,

if $a_i = b_i = 1$	then the nodes are sequences $\underline{p_{i+1}A1_i1}$
if $a_i = b_i = 0$	then the nodes are sequences $\underline{p_{i+1}A0_i0}$
if $a_i = 1$ and $b_i = 0$	then the nodes are sequences $\underline{p_{i+1}A1_i q_i}$
if $a_i = 0$ and $b_i = 1$	then the nodes are sequences $\underline{p_{i+1}A0_i q_i}$

Now we add the sequences corresponding to the edges, i.e. sequences $q_i p_i$. (Note that some edges are connecting nodes that are not present). Furthermore sequences $\underline{p_0 A0}$ are added since at position 0 no carry has to be considered. Complementary sequences hybridize. Double strands are extended by DNA polymerase and then denatured. Single strands containing a biotin are separated as described above.

We observe that for every position i, the resulting DNA molecules contain sequences starting with $\underline{p_{i+1}}$ at the 5' end, and ending with a 0 or a 1 at the 3' end. Moreover, if the

carry bit $c_{i+1} = 1$, then the sequences end with a 1. If the carry bit $c_{i+1} = 0$, then the sequences end with a 0.

We prepare a dextran matrix, to which the oligonucleotides 0 are attached (e.g. from ThermoHybaid). This selects two sets of DNA, one set containing sequences with p_i for all positions i such that the carry bit is 1, and one set containing sequences with p_k for all positions k such that the carry bit is equal to 0.

We add to each set the oligonucleotide A, hybridize and cut with restriction enzyme R_A. The double strands are denatured and separated by using a magnet. The second set contains $\underline{p_k A/2}$ or $\underline{q_k p_k A/2}$ for all k such that $b_k = 0$, and can be used to block positions k as described in subsection "Bitwise addition modulo 2".

4 Discussion

Standard methods of molecular biology are used to carry out all described calculations. Since all involved methods are feasible [ABK+01], the whole procedure seems also feasible. Single reactions might take a long time (e.g. ligation) and are error prone. Since we are using a constant number of steps for each position i, the overall error will be small enough to read out the result on the chip. Except cutting with restriction enzymes (e.g. BamH1, HindIII, EcoR1 etc.) the reactions are reversible. DNA double strands are more stable at lower temperature and higher salt concentration and the conditions for hybridization and denaturation have to be optimized. For the ligation reaction we can use T4 DNA ligase, for extending the 3' ends of p_i possible candidates are the Klenow fragment of DNA polymerase I, the large fragment of Bst-DNA polymerase or T4 DNA polymerase [ABK+01].

A crucial step involved is the computation of the carry bits. Here the length of the sequences depends on the number of bits used, slowing down the hybridization kinetics. A possible approach to keep sequences to an expected constant length could be to involve splicing mechanisms. (Splicing removes middle parts of the sequence, e.g. removes the $q_i p_i 0_i A$ part from $p_{i+1} A 1_i q_i p_i A 0_i 0$). For example, if we use RNA instead of DNA [FCLL00], we could exploit self splicing reactions. Currently there are limitations in exploiting splicing mechanisms: (1) RNA is less stable than DNA. (2) Splicing mechanisms are complex but generally irreversible. They depend on consensus sequences. Self splicing introns, which could be used, are RNA sequences with catalytic activity, so called ribozymes. Splicing of pre-mRNAs of higher eucaryotes is mediated by spliceosomes which are complexes of proteins and small RNAs [Sha94] and are therefore not applicable.

By removing biotin-streptavidine labeled oligonucleotides via a magnet [Adl94], or selecting oligonucleotides $\underline{0}$ losses may occur. These losses should not fall below a certain threshold level. The exact amounts have to be determined experimentally.

Currently we are investigating further biochemical reactions to simplify the procedure and to optimize the involved algorithm. The proposed model will be optimized to use as little as possible manual steps. An interesting question is, whether the calculation and addition of the carry bits is possible on the chip surface.

References

[ABK+01] F.M. Ausubel, R. Brent, R.E. Kingston, D.D. Moore, J.G. Seidman, J.A. Smith, and K. Struhl, editors. *Current Protocols in Molecular Biology.* Wiley and Sons, 2001.

[Adl94] L.M. Adleman. Molecular computation of solutions to combinatorial problems. *Science*, 266:1021–1024, November 11, 1994.

[Adl98] L.M. Adleman. Computing with DNA. *Scientific American*, 279(2):54–61, August 1998.

[Ata00] A. Atanasiu. Arithmetic with membranes. In *Workshop on Multiset Processing, Curtea de Arges, Romania, August 2000*, pages 1–17. C. S. Calude and M. J. Dinneen and Gh. Păun, 2000.

[FCLL00] D. Faulhammer, A. R. Cukras, R. J. Lipton, and L. F. Landweber. Molecular computation: RNA solutions to chess problems. *Proc. Natl. Acad. Sci. USA 97*, pages 1385–1389, 2000.

[FLT+97] A. G. Frutos, Q. Liu, A. J. Thiel, A. M. Sanner, A. E. Condon., L. M. Smith, and R. M. Corn. Demonstration of a word design strategy for DNA computing on surfaces. *Nucleic Acids Res*, 25(23):4748–4757, 1997.

[Fri00] P. Frisco. Parallel arithmetic with splicing. *Romanian Journal of Information Science and Technology (ROMJIST)*, 2000. to appear.

[GFB96] F. Guarnieri, M. Fliss, and C. Bancroft. Making DNA add. *Science*, 273(5272):220–223, July 12 1996.

[GPZ97] V. Gupta, S. Parthasarathy, and M.J. Zaki. Arithmetic and logic operations with DNA. In *Proceedings of the 3rd DIMACS Workshop on DNA Based Computers, held at the University of Pennsylvania, June 23 - 25, 1997* [RW99], pages 212–220.

[IER+99] V.R. Iyer, M.B. Eisen, D.T. Ross, G. Schuler, T. Moore, J.C. Lee, J.M. Trent, L.M. Staudt, J. Hudson, M.S. Boguski, D. Lashkari, D. Shalon, D. Botstein, and P.O. Brown. The transcriptional program in the response of human fibroblasts to serum. *Science*, 283:83–87, 1999.

[Lip95] R. J. Lipton. DNA solution of hard computational problems. *Science*, 268:542–545, April 28, 1995.

[LWF+00] Q. Liu, L. Wang, A. G. Frutos, A. E. Condon, R. M. Corn, and L. M. Smith. DNA computing on surfaces. *Nature*, 403:175–179, 2000.

[OKLL97] Q. Ouyang, P. D. Kaplan, S. Liu, and A. Libchaber. DNA solution of the maximal clique problem. *Science*, 278:446–449, 1997.

[OR96] M. Ogihara and A. Ray. Simulating boolean circuits on a DNA computer. Technical Report TR 631, University of Rochester, Computer Science Department, August 1996.

[QL98] Z. F. Qiu and M. Lu. Arithmetic and logic operations for DNA computers. *Second IASTED International Conference on Parallel and Distributed Computing and Networks*, pages 481–486, December 1998.

[Rei97] J. H. Reif. Local parallel biomolecular computing. In *Proceedings of the 3rd DIMACS Workshop on DNA Based Computers, held at the University of Pennsylvania, June 23 – 25, 1997* [RW99], pages 243–264.

[RW99] H. Rubin and D. Wood, editors. *Proceedings of the 3rd DIMACS Workshop on DNA Based Computers, held at the University of Pennsylvania, June 23 – 25, 1997*, volume 48 of *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science.*, Providence, RI, 1999. American Mathematical Society.

[SGK+00] K. Sakamoto, H. Gouzu, K. Komiya, D. Kiga, S. Yokoyama, T. Yokomori, and M. Hagiya. Molecular computation by DNA hairpin formation. *Science*, 288:1223–1226, 2000.

[Sha94] P.A. Sharp. Nobel lecture: Split genes and RNA splicing. *Cell*, 77:805–815, 1994.