Simulating Boolean Circuits on a DNA Computer

Mitsunori Ogihara* Animesh Ray†

Abstract

We demonstrate that DNA computers can simulate Boolean circuits with a small overhead. Boolean circuits embody the notion of massively parallel signal processing and are frequently encountered in many parallel algorithms. Many important problems such as sorting, integer arithmetic, and matrix multiplication are known to be computable by small size Boolean circuits much faster than by ordinary sequential digital computers. This paper shows that DNA chemistry allows one to simulate large semi-unbounded fan-in Boolean circuits with a logarithmic slowdown in computation time. Also, for the class NC^1, the slowdown can be reduced to a constant. In this algorithm we have encoded the inputs, the Boolean AND gates, and the OR gates to DNA oligonucleotide sequences. We operate on the gates and the inputs by standard molecular techniques of sequence-specific annealing, ligation, separation by size, amplification, sequence-specific cleavage, and detection by size. Additional steps of amplification are not necessary for NC^1 circuits. Preliminary biochemical experiments on a small test circuit have produced encouraging results. Further confirmatory experiments are in progress.

1 Introduction

Adleman [Adl94], subsequently Lipton [Lip95] showed the potential of Recombinant DNA-based combinatorial chemistry as a tool for solving computationally difficult search problems. The massive parallelism of liquid phase DNA chemistry, coupled with the encoding of information in DNA strands, raises the hope for solving "intractable" problems. These novel approaches to computation also raise the question whether the DNA computers as a devise for simulating existing massively parallel computation models can go beyond the limit of digital computers.

Among many massively parallel computation models, typical are the Parallel Random Access Machines (PRAM) and the Boolean circuits. In the PRAM model, the computation is carried out by ordinary serial processors that have as storage a shared, global memory. The processors execute individual programs. All processors can read from or write to the global memory "in parallel" (at the same time), and depending on the outcome of the simultaneous read and the simultaneous write, various PRAM models are defined. The complexity of a PRAM algorithm is measured by the number of processors involved and the running time. Recently, Reif [Rei95] formulated an abstract parallel DNA computation model, the Parallel Associative Memory model, and showed that his parallel model can simulate the Concurrent-Read, Exclusive-Write PRAM model (CREW PRAM model), with a small time overhead. More precisely, a CREW PRAM algorithm running in time T on P processors with the total memory size M can be simulated by a Parallel Associative Memory algorithm in time $O(T+\log^2 S)$ and space polynomial in S where $S = PM$. Implementing the abstract model on DNA computers with Recombinant DNA techniques results in $O(\log S)$ slowdown. Thus, in this case, the time complexity becomes $O(T \log S + \log^3 S)$.

On the other hand, in the Boolean circuit model, the computation is carried out by a network of signal processors (called gates) computing simple Boolean functions, the AND and the OR. These gates have no memory and process their incoming signals only once during the computation. The complexity of a Boolean circuit is measured by the size (the number of gates) and the depth (the length of the longest directed path). Depending on the input capacity of the AND and the OR, three Boolean circuit models are defined. They are (i) the unbounded fan-in circuits (the input capacity is unlimited for both the AND gate and the OR gate), (ii) the semi-unbounded fan-in circuits (the input capacity is two for the AND and unlimited for the OR), and (iii) the bounded fan-in circuits (the input capacity is two for both the AND gate and the OR gate).

The present communication studies the DNA computer simulation of the Boolean circuits. Our attention is on the following three issues: (1) the biochemical operations that are assumed; (2) the efficiency of the simulations, i.e., the cost functions (the time and the space) of the DNA simulation algorithms expressed as the depth and the size of

*Department of Computer Science, University of Rochester, Rochester, NY 14627. email: ogihara@cs.rochester.edu.

†Department of Biology, University of Rochester, Rochester, NY 14627. Supported in part by National Science Foundation Grant MCB-9630402. email: ray@ar.biology.rochester.edu.

the circuits to be simulated; (3) the maximum size of the circuits that the DNA algorithms can simulate.

There already exists a general answer to the second question. Boolean circuits of size m and depth d can be simulated by CREW PRAM algorithms with $O(m \log m)$ processors in time $O(d \log m)$ (see [SV84]). Thus, it follows from the result by Reif, that DNA computers can simulate Boolean circuits of size m and depth d in time $O(d \log^2 m)$ and space polynomial in m. However, we should not be satisfied with the answer for the following two reasons. First, direct simulations may result in a smaller overhead. Second, more importantly, concerns have been expressed about the use of extract for separation, as employed in Reif's method. There has been much discussion on the feasibility of extract [Adl95, Rei95, BL95, KKW96] because its error rate, even with the current best Recombinant DNA technique, is as large as 10^{-6} (see, [ABL+94]) which is high enough to fail the whole computation. We thus study the problem of simulating Boolean circuits by DNA computers without using extract.

This paper presents a DNA algorithm for simulating semi-unbounded fan-in circuits. In the DNA algorithm, the separated strands are always much longer than the others, e.g., 40-base strands are separated from 20-base strands. This property suggests us to conduct separation by gel electrophoresis, not by extract. Gel electrophoresis is a well-established biochemical method for ordering DNA strands according to the length. Adleman [Adl94] employed this operation for detection and Lipton's 3SAT algorithm [Lip95] uses it for separating legitimate truth assignments from those that are not. In addition to the separation by gel electrophoresis, our algorithm assumes appending (by ligation), cleavage (by restriction enzymes), detection, and amplification as the necessary biochemical operations. We show that, under the assumption that the above biochemical operations are error-less, a semi-unbounded fan-in circuit of depth d and size m can be simulated by a DNA computer in time $O(d \log \mathcal{F})$ and space $O(m\mathcal{F})$, where $\mathcal{F}$ denotes the maximum fan-out (the number of the outgoing edges from a gate) of the circuit. This is an improvement of the method we obtain from Reif's result. The time complexity has been reduced from $O(d \log^2 m)$ to $O(d \log m)$ and the space complexity has become $O(m^2)$. Our analysis may suggest that the fan-out will play a crucial role when measuring the efficiency of circuit simulations on DNA computers.

The result gives us an added bonus: a real-time simulation of the class NC^1 [Pip79]. NC^1 is the class of problems solved by bounded fan-in circuits of $O(\log n)$ depth and polynomial-size. Many fundamental computational problems from the integer arithmetic to sorting are known to belong to this class [BCH86, AKS83]. We show that an NC^1 circuit of depth d can be simulated on a DNA computer in $3d$ steps assuming only appending, cleavage, detection, and separation by size.

We have performed an actual biochemical experiment wherein we have attempted to compute the output of a circuit with four Boolean inputs, two OR gates, and one AND gate. Preliminary results of this experiment are encouraging but not yet unambiguous. Confirmatory experiments are in progress.

2 Simulating Semi-unbounded Fan-in Circuits

2.1 Semi-unbounded Fan-in Circuits

A semi-unbounded fan-in Boolean circuit of n inputs is a directed acyclic graph with labeled nodes. There are exactly $2n$ nodes with indegree 0. These nodes are called input gates and are labeled $x_1, \ldots, x_n, \overline{x_1}, \ldots, \overline{x_n}$. Other nodes are labeled by one of $\wedge$ and $\vee$. The nodes with label $\wedge$ compute the AND of at most two Boolean values while those with $\vee$ compute the OR of an arbitrary number of Boolean values. Nodes with outdegree 0 are called output gates. On an input $x = x_1 \cdots x_n \in \{0,1\}^n$, the gates of the circuits evaluate to 0 or 1 according to the following rules:

- If gate g is an input gate with label x_i, g evaluates to 1 if $x_i = 1$ and 0 otherwise.
- If gate g is an input gate with label $\overline{x_i}$, g evaluates to 1 if $x_i = 0$ and 1 otherwise.
- If gate g is labeled $\vee$ with incoming edges from gates $h_1, \ldots, h_m$, g evaluates to 1 if h_i evaluates to 1 for some $i, 1 \le i \le m$, and 0 otherwise.
- If gate g is labeled $\wedge$ with incoming edges from gates h_1 and h_2, g evaluates to 1 if both h_1 and h_2 evaluate to 1 and 0 otherwise.

There are two complexity measures for Boolean circuits, the size and the depth. The size of a circuit C, denoted by *size*(C), is the number of gates in it and the depth of C, denoted by *depth*(C), is the length of the longest directed path in it.

2.2 The Simulation

Let C be a circuit of depth d and size m and $x = x_1 \cdots x_n$ an input to C. Let $g_1, \cdots, g_m$ be the gates of C and $\mathcal{F}$ the maximum outdegree of the gates in C. For simplicity, we assume that C has only one output gate. Prior to the actual run of the simulation, we fix for each $i, 1 \le i \le m$, a pattern $\sigma[i]$ of DNA. The presence of $\sigma[i]$ will indicate that g_i evaluates to 1. These patterns will be designed so that they satisfy the following conditions:

- All of these patterns consist of a fixed number $\mathcal{L}$ of DNA molecules.
- For every $i \neq j, 1 \le i, j \le m$, $\sigma[i]$ and $\sigma[j]$ as well as $\overline{\sigma[i]}$ and $\sigma[j]$ agree at less than one fourth positions.

Also, we select a restriction enzyme $\mathcal{E}$ together with its cleavage pattern $\alpha \downarrow \beta$, and demand the following be satisfied:

- For all i, $\sigma[i]$ starts with α and ends with β but has neither of these patterns in the middle.

One may use results from the theory of error-correcting codes [vL91] to discover the patterns to satisfy all these conditions. The determination may be computationally intensive, but once the patterns are fixed, they can be used for all other length n inputs. We introduce one more parameter $\mathcal{P}$, which is an upper bound on the population size of the DNA synthesis.

Our simulation proceeds from level 0 toward level d. We assume for every k, that the following conditions hold after processing the gates at level k:

1. For every gate g_i at level k, the test tube contains $\sigma[i]$ if and only if g_i evaluates to 1, and the number of copies of $\sigma[i]$ present is at most $\mathcal{P}$.

2. For every gate g_i at level $k-1$, at most $\mathcal{F} \cdot \mathcal{P}$ copies of $\sigma[i]$ are present.

3. For every gate g_i at level either greater than k or less than $k-1$, no copies of $\sigma[i]$ are present.

4. All the strands contained in the test tubes are of length $\mathcal{L}$.

Now we describe how the gates are simulated. In the description below, it should be understood that by "pouring $\sigma[i]$" we mean pouring a population of the strand. We begin with the description of the simulation of the gates at level 0, the input gates. We create a test tube so that the conditions (1) through (4) are all satisfied. For each gate g_i at level 0, we do the following:

- If g_i computes the positive form of some x_j, then we will pour in the test tube $\sigma[i]$ if and only $x_j = 1$.

- If g_i computes the negative form of some x_j, then we will pour in the test tube $\sigma[i]$ if and only $x_j = 0$.

This requires only one step.

Next consider the gates at level $k > 0$. The simulation of the OR gates are different from that of the AND gates. Let $i_1, \dots, i_a$ be the indices of the gates at level $k-1$ and $j_1, \dots, j_b$ those of the gates at level k. We first describe the OR case.

In the first step, we amplify the existing $\sigma[i_r]$ to the amplitude of at least $\mathcal{F} \cdot \mathcal{P}$. One amplification step doubles the number of copies of any strand present in the test tube. Thus, we have only to run $(\log \mathcal{F} + \log \mathcal{P})$ amplification steps to achieve the desired amplitude, where the logarithm is base 2.

In the second step, we execute appending. For each $s, 1 \leq s \leq b$, we pour $\sigma[j_s]$ into the test tube. Also, for each pair (i_r, j_s) such that there is an edge from g_{i_r} to g_{j_s}, we pour a "linker" for binding $\sigma[i_r]$ after $\sigma[j_s]$. Then we allow ligation. By condition (1), for every $s, 1 \leq s \leq b$, g_{j_s} evaluates to 1 if and only if some $\sigma[i_r]$ such that g_{i_r} is an input to g_{j_s} is present. This implies that g_{j_s} evaluates to 1 if and only if there exists some r such that $\sigma[i_r]$ and the linker between $\sigma[i_r]$ and $\sigma[j_s]$ are both present when the ligation takes place. We have already amplified any existing g_{i_r} to the amplitude of $\mathcal{F} \cdot \mathcal{P}$ in the first step. The output of g_{i_r} is plugged into at most $\mathcal{F}$ distinct g_{j_s}. We have poured into at most $\mathcal{P}$ copies of g_{j_s}. Thus, for each g_{j_s} that evaluates to 1, regardless of the combinations of the strands that are when ligation takes place, at least one copy of $\sigma[j_s]$ is appended to some $\sigma[i_r]$, thereby yielding a length $2\mathcal{L}$ strand. On the other hand, for every g_{j_s} that evaluates to 0, there exist no such strands to which the linkers can bound $\sigma[j_s]$. Thus, no strands of length $2\mathcal{L}$ ending with $\sigma[j_s]$ are created. Furthermore, no strands of length greater than $2\mathcal{L}$ are created.

In the third step, we separate length $2\mathcal{L}$ strands from length $\mathcal{L}$ strands. We use denaturing polyacrylamide gel electrophoresis [SFM89] for that purpose. The strands of length $2\mathcal{L}$ correspond to the gates that evaluate to 1.

In the fourth step, we cleave the length $2\mathcal{L}$ strands at $\alpha \downarrow \beta$ by restriction enzyme $\mathcal{E}$. This step produces all the strands $\sigma[j_s]$ such that g_{j_s} evaluates to 1.

We estimate the number of strands that are present when the fourth step has been finished. As to a gate g_{j_s} at level k, we have poured at most $\mathcal{P}$ copies of $\sigma[j_s]$, so at most $\mathcal{P}$ copies of $\sigma[j_s]$ should be present. As to a gate g_{i_r} at level $k-1$, the copies of $\sigma[i_r]$ are linked to at most $\mathcal{F}$ different $\sigma[j_s]$ and there are at most $\mathcal{P}$ copies of such a $\sigma[j_s]$, so at most $\mathcal{F} \cdot \mathcal{P}$ copies of g_{i_j} should be present. As to the strands for the gates at level below $k-1$ level, even if they may have existed prior to the processing of the gates at level k, no linkers have been added to bind them to other strands, so none of them will remain after the third step. Finally, as to the strands for the gates at higher levels, we have not poured them yet. Hence, all the loop invariant conditions are met.

Next we consider the AND case. The first step is identical to that of the OR case. We amplify the strands from the previous level to the amplitude of $\mathcal{F} \cdot \mathcal{P}$. In the second step, we execute appending. For each AND gate g_{j_s} at this level, we pour into the test tube $\sigma[j_s]$. Also, for each triple (j_s, i_q, i_r) such that $q < r$ and that g_{j_s} takes as an input both g_{i_q} and g_{i_r}, we pour the "linker" for appending $\sigma[j_s]$ after $\sigma[i_q]$ and the one for appending $\sigma[i_r]$ after $\sigma[j_s]$. Then we allow ligation. By an argument similar to that of the ligation step for the OR case, we observe that a length $3\mathcal{L}$ strand with $\sigma[j_s]$ in the middle is created if and only if g_{j_s} outputs 1 and that the other strands are of length either $2\mathcal{L}$ or $\mathcal{L}$. In the third step, we use electrophoresis to separate the strands of length $3\mathcal{L}$ from those of length at most $2\mathcal{L}$. In the fourth step, we cleave at $\alpha \downarrow \beta$ using the restriction enzyme $\mathcal{E}$. Then, from $\sigma[i_q]\sigma[j_s]\sigma[i_r]$, $\sigma[j_s]$ is produced. By following the discussion similar to the OR case, we observe that the loop invariant conditions are all met.

At the end of the computation, namely when processing the level d (i.e., the output) gate, instead of the last two steps we execute gel electrophoresis to find the output of the circuit. If the output gate is an OR gate, the output is 1 if and only if length $2\mathcal{L}$ strands exist and if it is an AND gate, the output is 1 if and only if length $3\mathcal{L}$ strands exist.

The complexity analysis

Here we analyze the complexity of the simulation described above. Ligation and gel electrophoresis are executed d times each. Cleavage is executed $d-1$ times. amplification is executed $(\log \mathcal{F} + \log \mathcal{P})d$ times and we need one step for setting up the strands for the gates at level 0. Thus, the total number of steps is $(3 + \log \mathcal{F} + \log \mathcal{P})d$. On the other hand, the maximum number of DNA strands that remain in the test tube after processing a single level is bounded by $m\mathcal{F}\mathcal{P}$.

Thus, we have proven the following theorem.

Theorem 1 *A semi-unbounded fan-in circuit C of size m, depth d, the maximum fan-out $\mathcal{F}$ can be simulated by a DNA computer in $d \log \mathcal{F} + O(d)$ steps with space complexity $O(m\mathcal{F})$.*

For a natural number k, SAC^k denotes the collection of problems that are solvable by a family $\{C_n\}_{n \geq 1}$ of semi-unbounded fan-in circuits such that $size(C_n) = O(n^a)$ for some fixed constant a and $depth(C_n) = O(\log^k n)$. SAC without the superscript denotes the union of all $\mathrm{SAC}^k, k \geq 0$. It is well-known that the class SAC^1 coincides with the problems that are logarithmic space transformable to CFL,

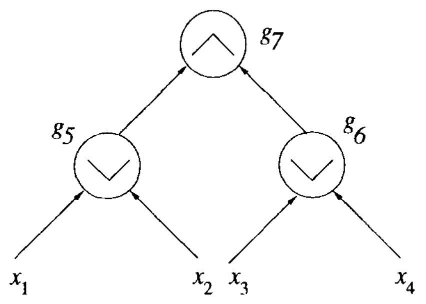


Figure 1: The Depth-2 And-Or Circuit

Number	I_1	I_2	I_3	I_4
1	1	1	1	1
2	0	1	0	1
3	0	0	1	1

Table 1: The Test Inputs

the context-free languages [Ven91]. Also, many combinatorial and mathematical problems are known to belong to SAC. Since the maximum fan-out $\mathcal{F}$ is bounded by the size m, and m is a polynomial in n for SAC circuits, we have the following corollary.

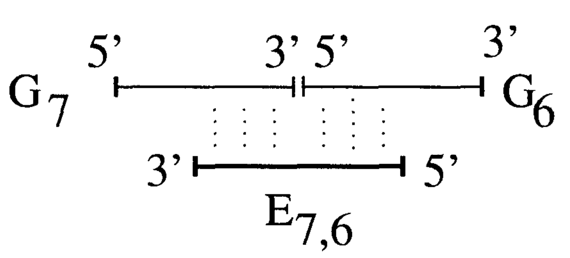
Corollary 2 *For every $k \geq 0$, SAC^k can be simulated by a DNA computer in time $O(\log^{k+1} n)$ and space polynomial in n.*

The above analysis gives us an added bonus. The class NC [Pip79] is the counterpart of SAC in the bounded fan-in circuit model. There is a vast literature on NC^1 classes and numerous problems from the integer arithmetic to sorting are proven to belong to this class. NC^1 circuits can be converted to a tree, by allowing many copies of input gates to exist. For a circuit in the form of a tree, the maximum fan-out $\mathcal{F}$ is 1, and thus, the simulation of such a circuit does not require the amplification step. This allows us to get rid of the $(\log \mathcal{F} + \log \mathcal{P})$ factor in the running time and the $\mathcal{F}$ factor in the space complexity.

Corollary 3 *Let $k > 0$ and $\{C_n\}_{n \geq 1}$ be an NC^k circuit family. For every n, C_n can be simulated by a DNA computer in time $3 \cdot depth(C_n)$ and space $\mathcal{P} \cdot size(C_n)$, where only appending, separation by size, detection, and cleavage are assumed.*

3 The Experiment

We have experimentally simulated a small instance of a bounded fan-in Boolean circuit given in Figure 1. The four input variables $x_i, i = 1, \ldots, 4$, were encoded as four 21-mer oligonucleotides of unique sequences, each of whose $3'$ terminus ends in the sequence $5'$-GT-$3'$. These are designated I_1, I_2, I_3, and I_4, respectively. The two OR gates were encoded as two 22-mer oligonucleotides each of whose $5'$ and $3'$ ends are $5'$-AC$\cdots$GT-$3'$. In addition, a phosphate group was attached to each of these two oligonucleotides at their $5'$ end [SFM89]. These are identified as G_5 and G_6, respectively. The AND gate (designated, G_7) was chosen as a 21-mer oligonucleotide of unique sequence whose $5'$ end was attached to a phosphate group with a radioactive phosphorus atom. In addition, we synthesized six 20-mer oligonucleotides corresponding to the six edges ($E_{1,5}, E_{2,5}, E_{3,6}, E_{4,6}, E_{5,7}$, and $E_{7,6}$). These edges are directional. For example, the $5'$ half of edge $E_{1,5}$ is complementary to 10 nucleotides in the $5'$ half of G_5 with the reversed polarity, and the $3'$ half of $E_{1,5}$ is complementary to the $3'$ half of I_1 with the reversed polarity. This is shown in Figure 2. Thus, the edge oligonucleotide $E_{1,5}$ can hybridize to both G_1 and G_5, but to no other sequence, such that a DNA joining enzyme (ligase) [SFM89] can covalently join the $5'$ phosphate of G_5 to the $3'$ hydroxyl group of I_1. This will make a 43 base long dimeric oligonucleotide $5'$-I_1-G_5-$3'$. Similarly, $E_{2,5}$, $E_{3,6}$, $E_{4,6}$, and $E_{5,7}$ make the dimeric oligonucleotides $5'$-I_2-G_5-$3'$, $5'$-I_3-G_6-$3'$, $5'$-I_4-G_6-$3'$, and $5'$-I_5-G_7-$3'$, respectively. Each such dimer will contain in the middle of the molecule a 4-base sequence $5'$-GTAC-$3'$. This is the target sequence of a single-stranded DNA endonuclease *RsaI*. Finally, the sequence and the polarity of the edge oligonucleotide $E_{7,6}$ was so chosen that it could make possible the synthesis of the molecule $5'$-G_7-G_6-$3'$. If oligonucleotides $G_5, G_6, G_7, E_{5,7}$, and $E_{7,6}$ are all present in a test tube, a ligase reaction should produce the 65 nucleotide long trimer $5'$-G_5-G_7-G_6-$3'$ which should be radioactive. If either G_5 or G_6 is absent in the tube, the 65 nucleotide long trimer cannot be formed.



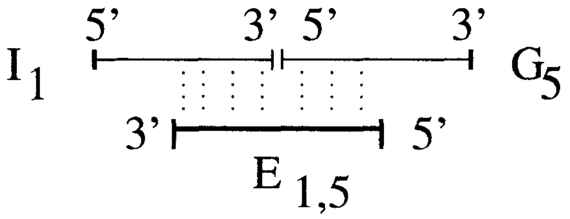


Figure 2: The Match of the Oligonucleotides

We have begun to test the simulation by actual experiments. The preliminary experiments have used three different input combinations two of which should generate the output 1 and the other should produce the output 0 (Table 1). Thus, test 1 contained in the tube inputs I_1, I_2, I_3, and I_4; test 2 contained only I_2 and I_4; and test 3 contained only I_3 and I_4. To implement the OR gate, we added $G_5, G_6, E_{1,5}, E_{2,5}, E_{3,6}$, and $E_{4,6}$, allowed annealing and ligation, and separated the products by size on a denaturing polyacrylamide gel by electrophoresis [SFM89]. Positions of

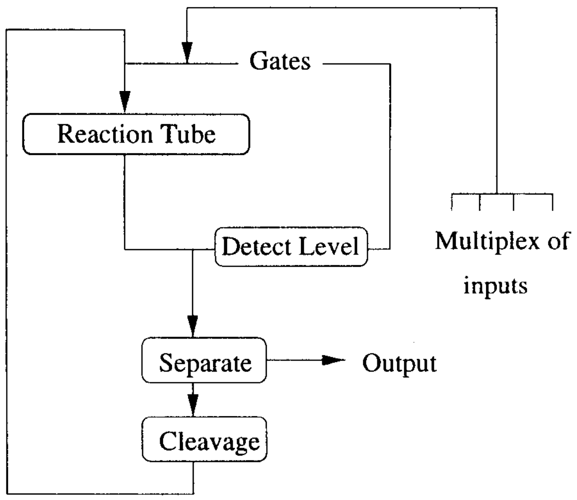


Figure 3: A Diagram for Automated Experiments

the gel corresponding to the dimer size band were cut out and DNA, if any, was extracted. These solutions were then treated with the enzyme *RsaI* to cleave any dimer DNA to the monomer length. To each tube were then added the oligonucleotides $G_7, E_{5,7}$, and $E_{7,6}$. They were allowed to anneal and ligate. The final products were again separated according to size by electrophoresis on a denaturing polyacrylamide gel and visualized by autoradiography on an X-ray film. Our initial results indicate that we could detect the presence of the 65 base trimer in one of the two tubes that should generate 1 (tube 2). The trimer could not be detected in the tube that should generate 0 (tube 3). However, the presence of the trimer in tube 1 with the expected output of 1 could not be unambiguously detected. The dimer bands, expected in all three outputs, were present in all tubes. This allows us to narrow down the cause of the discrepancy to pipetting error in tube 1. We are now repeating these experiments with an improved version of the protocol.

Is molecular circuit analysis feasible for a large network on which digital computes are inefficient? By a proof similar to that of the Gilbert-Varshamov Theorem (see [vL91]), one can prove that there is a set of 1.6×10^{12} distinct 40 base oligonucleotide sequences such that (i) for any two sequences A and B, A disagrees with B and its complement at 10 positions and (ii) no sequence contains the pattern that is cleaved by the restriction enzyme *RsaI*. So, we can handle at least 1 trillion gates by encoding the gates as 40 base oligonucleotide sequences. The number 1 trillion is far beyond the reach of digital computers. Artifactual annealing between short stretches of homology in otherwise noncomplementary oligonucleotides maybe avoided by enzymatic annealing catalyzed by the DNA strand transfer protein RecA [KE94].

The runtime of digital computation for simulating bounded fan-in circuits is proportional to the size while in our DNA algorithm the runtime is a linear function of the depth. Thus, for certain very large circuits, it is possible that DNA computers may outperform digital computers in computation time, too. The step time in DNA computation can be significantly reduced by automated devices: the separation of oligonucleotides at each level can be accomplished by robotically controlled electrophoresis in microcapillaries and high performance liquid chromatography [WV93] programmed to achieve an optimum resolution between 20 and 70 nucleotides. The final output can be detected by fluorescence emission by wave guidance techniques [SHH+95]. Efficiency of the cleavage reaction can be similarly monitored by fluorescence measurements and the step levels extended by feedback until a preset limit of cleavage is attained. When a non-tree circuit is simulated, an amplification step needs to be incorporated. This can be conveniently made by Ligase Chain Reaction (LCR) [AM93] in which the new junctions can be amplified by annealing and ligation of two half edge molecules on the complementary dimer. This also can be performed automatically on silicon microchips [CSM+96]. Since LCR does not contain a DNA synthesis step during amplification, it is inherently less error-prone than amplification by PCR.

4 Conclusion

We have shown that the DNA computer is a potential tool for implementing standard computation models, in particular, for Boolean circuits. We have proven that the runtime slowdown is proportional to the logarithm of the maximum fan-out of the Boolean circuit and the space complexity is proportional to the product of the size and the maximum fan-out. The implication is that DNA computers are capable of simulating circuits with 10 billion gates, which may exceed the maximum number of parallel processing units in digital computing devices.

References

[ABL+94] B. Alberts, D. Bray, J. Lewis, M. Raff, K. Roberts, and J. Watson. *Molecular biology of the cell.* Garland Publishing Inc., New York, NY, 3rd edition, 1994.

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

[Adl95] L. Adleman. On constructing a molecular computer. Manuscript, 1995.

[AKS83] M. Ajtai, J. Komlos, and E. Szemerédi. Sorting in $c \log n$ parallel steps. *Combinatorica*, 3(1):1–19, 1983.

[AM93] R. D. Abramson and T. W. Myers. Nucleic acid amplification technologies. *Currerent Opinion in Biotechnology*, 4:41–47, 1993.

[BCH86] P. Beame, S. Cook, and H. Hoover. Log depth circuits for division and related problems. *SIAM Journal on Computing*, 15(4):994–1003, 1986.

[BL95] D. Boneh and R. Lipton. Making DNA computers error resistant. Research Report CS-TR-491-95, Department of Computer Science, Princeton University, May 1995.

[CSM+96] J. Chen, M. A. Shoffner, K. R. Mitchelson, L. J. Kricka, and P. Wilding. Analysis of ligase chain reaction products amplified in a silicon glass chip using capillary electrophoresis. *Journal of Chromatography, Series A*, 732:151–158, 1996.

[KE94] S. C. Kowalczykowski and A. K. Eggleston. Homologous pairing and DNA strand-exchange proteins. *Annual Review of Biochemistry*, 63:991–1043, 1994.

[KKW96] R. Karp, C. Kenyon, and O. Waarts. Error-resilient DNA computation. In *Proceedings of the 7th ACM-SIAM Symposium on Discrete Algorithms*, pages 458–467. ACM Press/SIAM, 1996.

[Lip95] R. Lipton. DNA solutions of hard computational problems. *Science*, 268(28):542–545, 1995.

[Pip79] N. Pippenger. On simultaneous resource bound. In *Proceedings of the 11th Symposium on Theory of Computing*, pages 307–311. ACM Press, 1979.

[Rei95] J. Reif. Parallel molecular computation. In *Proceedings of the 7th ACM Symposium on Parallel Algorithms and Architecture*, pages 213–223. ACM Press, 1995.

[SFM89] J. Sambrook, E. F. Fritsch, and T. Maniatis. *Molecular Cloning: A Laboratory Manual.* Cold Spring Harbor Press, NY, 2nd edition, 1989.

[SHH$^+$95] D. Simpson, J. Hoijer, W. Hsieh, C. Jou, J. Gordon, T. Theriault, R. Gamble, and J. Baldeschwieler. Real-time detection of DNA hybridization and melting on oligonucleotide arrays by using optical wave guides. *Proceedings of the National Academy of Science*, 92:6379–63830, 1995.

[SV84] L. Stockmeyer and U. Vishkin. Simulaiton of parallel random access machines by circuits. *SIAM Journal on Computing*, 13(2):409–422, 1984.

[Ven91] H. Venkateswaran. Properties that characterize LOGCFL. *Journal of Computer and System Sciences*, 43:380–404, 1991.

[vL91] J. van Lint. *Introduction to coding theory.* Springer-Verlag, 1991.

[WV93] W. J. Warren and G. Vella. Analysis of synthetic oligodeoxyribonucleotides by capillary gel electrophoresis and anion-exchange HPLC. *BioTechniques*, 14:598–606, 1993.