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. The feasibility of the DNA algorithm has been successfully tested on a small circuit by actual biochemical experiments.

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

*Department of Computer Science, University of Rochester, Rochester, NY 14627. email: ogihara@cs.rochester.edu. Supported in part by the National Science Foundation Grants CCR-9701911 and CCR-9725021.

†Department of Biology, University of Rochester, Rochester, NY 14627. email: ray@ar.biology.rochester.edu. Supported in part by the National Science Foundation Grants MCB-9630402 and IBN-9728239.

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 (the PAM model, in short). The PAM model, in addition to the standard Recombinant DNA operations, assumes an operation called PA-Match, which is a generalized form of ligation of long DNA strands. Reif showed that the PAM model can simulate the Concurrent-Read, Exclusive-Write PRAM model (the 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 with $O(T+S)$ PA-Match steps and $O(S \log S)$ other steps, and with space polynomial in S. Here S is $\log P + \log M$ and the PA-Match steps are performed on $O(S)$ length strands.

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

The question on simulation efficiency was addressed in the past by Reif [Rei95]. First, combined with the fact that 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]), the results of Reif [Rei95] indicate that the PAM model can simulate Boolean circuits of size m and depth d in time $O(d \log m + \log^2 m)$ and space polynomial in m. However, this indirect analysis may be unsatisfactory, because simpler and more accurate methods are possible if we directly attack circuit evaluation problem. In particular, concerns have been expressed regarding accuracy about the use of sequence-specific separation as employed in Reif's method. There has been much discussion on the feasibility of the "extract" operation [Adl96, 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. Second, Boneh *et al.* [BDLS96] presented a method for evaluating a Boolean circuit with many inputs. In this method, evaluation goes gate by gate, and each DNA strand is the concatenation of the input bits and the values of the Boolean gates that have been evaluated so far. In order to evaluate a gate g, one uses sequence-specific separation with respect to input bits of g and splits the strands input two groups depending the value of the gate, then appends the pattern to represent the output of g. This approach by Boneh *et al.* is in some sense orthogonal to the approach we are studying in this present paper. Namely, although their method allows concurrent evaluation of the circuit at different inputs, it requires time proportional to the circuit size. Third, Beaver [Bea96] proposes a method for simulating with DNA polynomial space bounded computation in polynomial time using site-directed DNA mutagenesis. Although this strong simulation results provide a circuit evaluation method with the same efficiency as the one derived from Reif's result, we are skeptical about feasibility of the mutagenesis operation. Finally, Winfree [Win96] proposes a method for simulating cellular automata using annealing and ligation applied to crossed-strand DNA junctions (Holliday junctions). As in the method by Beaver, this method can provide an efficient evaluation of boolean circuits, but we have doubts about its feasibility when applied to circuits of useful size.

Here we propose a method for simulating the semi-unbounded fan-in Boolean circuit model by DNA computers without the extract operation

but still assuming DNA pattern matching of logarithmically long strands as is done in the PAM model by Reif. This method enables us to evaluate a boolean circuit at a single input in time proportional to the depth of the circuit. In our 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 allows 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. Since the depth of semi-unbounded fan-in circuits is in general $\Omega(\log m)$, the efficiency of our algorithm matches the bound in the afore-mentioned analysis from Reif's result. 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 reported here.

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 $i, 1 \leq i \leq m$, 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 $i, 1 \leq i \leq m$, 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 $i \neq j, 1 \leq i, j \leq m$, $\sigma[i]$ 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 $i_1, \ldots, i_a$ be the indices of the gates at level $k-1$ and $j_1, \ldots, j_b$ 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 $\mathit{size}(C_n) = O(n^a)$ for some fixed constant a and $\mathit{depth}(C_n) = O(\log^k n)$. 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, the context-free languages [Ven91]. Also, many combinatorial and mathematical problems are known to belong

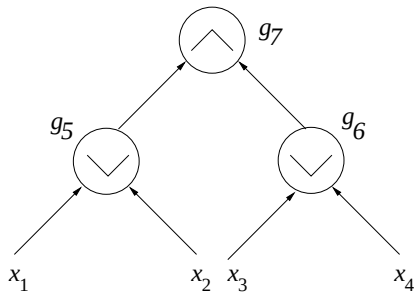

Figure 1: The Depth-2 And-Or Circuit

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 attempted to experimentally simulate 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

Tube Number	I_1	I_2	I_3	I_4	
1	–	–	–	–	monomer size standard
2	1	1	1	1	
3	0	1	0	1	
4	1	0	1	0	
5	0	0	1	1	
6	–	–	–	–	monomer + dimer size standards

Table 1: The Test Inputs

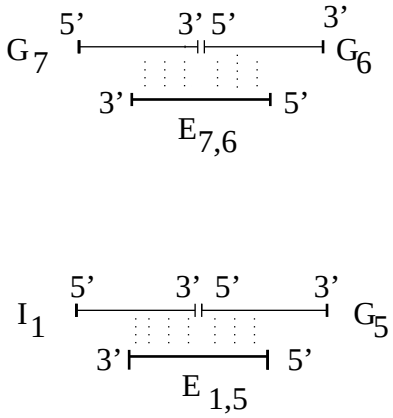

Figure 2: The Match of the Oligonucleotides

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,

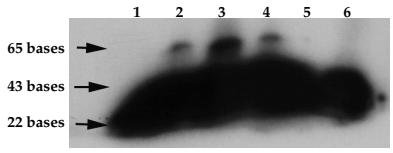

Figure 3: Test Results

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 *Rsa*I. 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.

We have begun to test the simulation by actual experiments (for methods, see Section 5). The preliminary experiments have used four different input combinations three of which should generate the output 1 and the other should produce the output 0 (Table 1). Thus, test tube 2 contained inputs I_1, I_2, I_3, and I_4; test tube 3 contained only I_2 and I_4; test tube 4 contained only I_1 and I_3; and test tube 5 contained only I_3 and I_4. In test tubes 1 and 6, we added DNA size markers for monomer and monomer+dimer, respectively. 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 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 *Rsa*I 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 (see Figure 3). Our results indicate that we detected the presence of the 65 base trimer in all three tubes that should

generate 1 (tubes 2–4), and not in tube 5, which should generate output 0. The dimer bands, expected in all four outputs, were present in all tubes.

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 show 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 *Rsa*I. 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]. So, we can handle at least one trillion wires by encoding the gates as 40 base oligonucleotide sequences. On the other hand, one trillion wires are perhaps beyond the reach of digital computers. Then one might think that evaluation of such large circuits could be a "killer application" of DNA-based computation. However, one should note that synthesis of one trillion linker sequences of DNA has to be done sequentially unless the evaluated circuit has a highly regular gate-connection pattern so that much of gate-sequence synthesis process can be done concurrently. Therefore, one trillion wires are beyond the reach of DNA computer as well. However, we note that a technique for parallel synthesis of many DNA sequences currently exists [CYH+96].

Nevertheless, we still argue that our DNA-based evaluation method will be useful, in particular, when we need to evaluate one circuit on many distinct input sets. Suppose that we need to evaluate a Boolean circuit C of S wires, depth D, with I input gates on T different sets of input bits. If we are to conduct evaluation one input after another, then the total amount of time required is

$$O(T(S + I)) + O(TD),$$

where the first term is for preparation and the second for evaluation. Noting that the same set of linkers and gate strands are used for all the input sets, we can spare time by keeping the originals synthesized for the first input set and by using copies of the originals for the other input sets. Since copying process can be done level-wise, this modification reduces the total time to

$$O(T(D + I) + S) + O(TD),$$

thereby reducing the amortized cost to $O(D + S/T)$. For example, with $D = 10^4$, $I = 10^4$, $S = 10^8$, and $T = 10^4$, the total preparation time is

only an order of 10^8 and the amortized running is an order of 10^4. On the other hand, the amount of time required for sequential digital processing is $TS = 10^{12}$.

In the same situation, we can even run the evaluation of all input sets concurrently in one test tube. To do this we employ a slightly different encoding scheme. We insert in the middle of each gate sequence a pattern that specifies a number $J, 1 \leq J \leq I$. This middle pattern is used to specify which of the input set the strand will be used for. We also make the linker strands specify their input names J by attaching the strand that is complementary antiparallel to the first half (the second half) of J at their $5'$-ends ($3'$-ends). If all the input names are assigned short common patterns at both ends, then we can generate the necessary strands for concurrent evaluation by controlling the volume of the strands and then by chemically connecting the components. Then the preparation cost is $O(T + S)$, and the total running time becomes $O(T + S + D)$. Again, the amount of time required for sequential digital processing is still $TS = 10^{12}$.

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

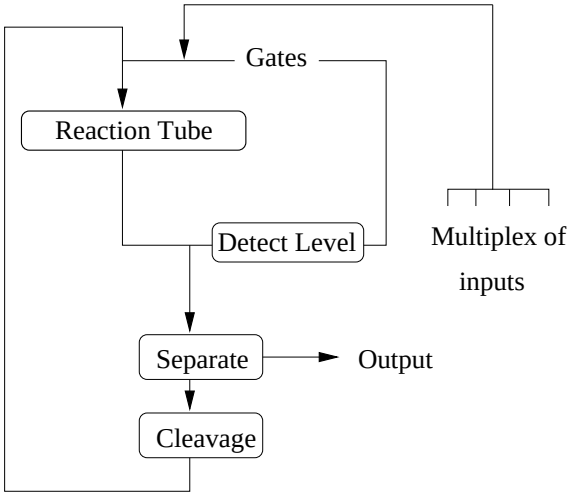

Figure 4: A Diagram for Automated Experiments

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.

5 Methods

Radioactive labeling: 2μg of the oligonucleotide was mixed with 20μCi of $[\gamma^{32}\mathrm{P}]$ATP (DuPont) in 0.25x concentrated DNA ligase buffer (NewEngland Biolabs) and one unit of T_4 polynucleotide kinase (NewEngland Biolabs) was added in a total volume of 19 μl. After 15 minutes at 37°C, 4 μl of 5x concentrated DNA ligase buffer was added and incubation was continued for 30 more minutes. The labeled DNA was stored frozen at -20°C.

Phosphorylation of the $5'$ end: 2μg of the oligonucleotide was mixed with 1 unit of T_4 polynucleotide kinase in 1x concentrated DNA ligase buffer in a total volume of 20μl, and incubated at 37°C for 45 minutes. The phosphorylated DNA was stored at -20°C until used.

Ligation: 1 nanomole of each oligonucleotide was mixed together in a total volume of 6μl; 2μl of 5x concentrated DNA ligase buffer was added and the reaction was started by 2μl of T_4 DNA ligase (NewEngland Biolabs). Ligation was carried out at 23°C for 1 hour.

Electrophoresis, detection, recovery and treatment of DNA: The DNA bands were separated by electrophoresis in 15% urea-polyacrylamide gels at 50 Volts and approximately 10 mA current in Tris-acetate-EDTA buffer [SFM89]. At the end of electrophoresis, an X-ray film was placed on the wet gel in Saran wrap and exposed at the room temperature for 2.5 hours in the dark. The X-ray film was developed, placed under the gel to ascertain the position of a 43 mer band from a radioactive marker of the same size electrophoresed in a parallel lane, and acrylamide pieces corresponding to this size of DNA were cut out with a scalpel from the appropriate lanes. The gel pieces were macerated, soaked in 150 μl of 0.3 M sodium acetate and agitated at 37°C for 2 hours to elute the DNA. The gel pieces were discarded, an additional 100 μl of 0.3 M sodium acetate was added, extracted successively with equal volumed of phenol-chloroform mixture (1:1), and chloroform; finally the DNA was precipitated with 2.5 volume of ethanol at -20°C. The dried pellet was resuspended in 8μl of water, and digested for 2 hours at 37°C with the enzyme *Rsa*I (NewEngland Biolabs) in the appropriate buffer as recommended by the manufacturer. Following digestion, the DNA was boiled for 2 minutes to inactivate the enzyme.

Acknowledgments

The authors thank John Reif and anonymous referees for valuable comments.

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*, 266:1021–1024, 1994.

[Adl96] L. Adleman. On constructing a molecular computer. In R. Lipton and E. Baum, editors, *Proceedings of 1st DIMACS Workshop on DNA Based Computers*, pages 1–21. The American Mathematical Society, 1996.

[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.

[BDLS96] D. Boneh, C. Dunworth, R. Lipton, and J. Sgall. On the computational power of DNA. *Discrete Applied Mathematics*, 71:79–94, 1996.

[Bea96] D. Beaver. A universal molecular computer. In R. Lipton and E. Baum, editors, *Proceedings of 1st DIMACS Workshop on DNA Based Computers*, pages 29–36. The American Mathematical Society, 1996.

[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.

[CYH+96] M. Chee, R. Yang, E. Hubbell, A. Berno, X. Huang, D. Stern, J. Winkler, D. Lockhart, M. Morris, and S. Fodor. Accessing genetic information with high density DNA arrays. *Science*, 274:610–614, 1996.

[KE94] S. C. Kowalczykowski and A. K. Eggleston. Homologous pairing and DNA strand-exchange proteins. *Ann. Rev. Biochem.*, 63:991–1043, 1994.

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

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

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

[Rei95] J. Reif. Parallel molecular computation. In *Proceedings of 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.

[Win96] E. Winfree. On the computational power of DNA annealing and ligation. In R. Lipton and E. Baum, editors, *Proceedings of 1st DIMACS Workshop on DNA Based Computers*, pages 199–221. The American Mathematical Society, 1996.

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