

DNA Simulation of Nand Boolean Circuits *

H. Ahrabian

*Department of Mathematics and Computer Science,
Faculty of Science, University of Tehran,
Tehran, Iran.
E-mail: ahrabian@ut.ac.ir*

and

A. Nowzari-Dalini

*Department of Mathematics and Computer Science,
Faculty of Science, University of Tehran,
Tehran, Iran.
E-mail: nowzari@ut.ac.ir*

Abstract

In this paper we describe a simulation of Boolean circuits using bio-molecular techniques. Boolean circuits embody the notion of massively parallel signal processing and are frequently encountered in many parallel algorithms. This model operates on the gates and the inputs by standard molecular techniques of sequence-specific annealing, ligation, melting, cleavage, separation and detection by size. It is shown that the simulation is run in the time proportional to the depth of the circuit.

Keywords: DNA computing, Parallel processing, Boolean circuits.

*This research is supported by University of Tehran.

1 Introduction

In 1961 Feynman proposed that classical computer operations can be simulated by molecular operations [Feynman, 1961], and few decades later Adelman implemented his idea in the laboratory [Adleman, 1994]. He used standard tools of molecular biology to solve an instance of the Hamiltonian path problem in a test tube, just by handling DNA strands. After that a major goal of subsequent research is how to use DNA manipulations to solve hard problems [Braich et al., 2002, Chang and Guo, 2003, Chang et al., 2004, Lipton, 1995], and several authors have described various models of computation using bio-molecular methods such as Turing machine [Beaver, 1995, Rothmund, 1996], finite state automata [Gao et al., 1999], and Boolean circuits [Amos and Dunne, 1997, Ogihara and Ray, 1998]. Boolean circuits are an important model of parallel computation [Dunne, 1998]. An n -input *bounded fan-in* Boolean circuit may be viewed as a directed, acyclic graph, S , with three types of nodes: n *input* nodes with in-degree zero, g *gate* nodes with maximum in-degree two, and m *output* nodes with out-degree zero. Each input node is associated with a unique Boolean variable x_i from the input set $\mathbf{x} = (x_1, x_2, \dots, x_n)$ of Boolean function. Each gate node g_i is associated with some Boolean function $f_i \in \Omega$. We refer to Ω as the circuit *basis*. A complete basis is a set of functions that are able to express all possible Boolean functions.

Ogihara and Ray described the first DNA-based simulation of NAND Boolean circuits [Ogihara and Ray, 1998, Ogihara and Ray, 1999]. As we know NAND Boolean circuits contain only NAND gates. They proved that the runtime slow-down 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. Amos and Dunne also presented another simulation of NAND Boolean circuits [Amos and Dunne, 1997]. Their simulation runs in time proportional to the depth of the circuits and their simulation is much easier to implement than previous one. Since then, all the other simulations of Boolean circuits are constructed by OR and AND gates. This might reduce the completeness of the circuits with respect to propositional logic.

In this paper another DNA simulation of NAND Boolean circuits is presented. Since it is well-known that the NAND gates provide a complete basis for Boolean operators and any Boolean functions can be implemented only by NAND gates, therefore, we restrict our model to the stimulation of such gates. In fact, using these gates offer the most suitable basis for Boolean circuits simulation with in DNA computation. The simulation of Boolean circuits with NAND gates provides a unique method for evaluation of the circuits. This model is similar to Amos and Dunne simulation for Boolean circuits with

the complexity of proportional to the depth of the circuit [Amos and Dunne, 1997]. Our model is much easier to implement in the laboratory, and the number of DNA operations used in the library is much less.

2 The Circuit Simulation

Our simulation is preceded for a Boolean circuit with the following specification.

An n -input, m -output Boolean circuit is modeled as a directed cyclic graph, $S(V, E)$, in which the set of vertices V is formed from three disjoint sets: I_n , the *inputs* of the circuit which there are exactly n ; and G , the *internal gates*; and O_m , the *outputs* which there are exactly m . Each input vertex of I_n has in-degree 0 and are associated with a single Boolean variable x_i from a given Boolean function. Each internal gate and output gate have in-degree 2 and are associated with the Boolean operation NAND. The internal gates will also have out-degree 1. The m distinguished output gates O_m are conventionally regarded as having out-degree equal to 0.

A Boolean circuit contains k levels ($0 \cdots k - 1$). The input gates are appeared in first level (level zero) , and the output gates appear in the last level (level $k - 1$), all the intermediate gates are presented in between the first and the last level (levels $1 \cdots k - 2$). The inputs of each intermediate and output gates are supported by the outputs of the gates in the previous level. An assignment of Boolean variables from $\langle 0, 1 \rangle^n$ to the input I_n ultimately induces Boolean values at the output gates O_m .

An n -input , m -output Boolean circuit C , is said to compute an n -input , m -output Boolean function, $f(I_n) : \langle 0, 1 \rangle^n \rightarrow \langle 0, 1 \rangle^m$, on other words, for any input we have

$$f^{(i)}(I_n) : \langle 0, 1 \rangle^n \rightarrow \{0, 1\} : 1 \leq i \leq m \text{ if } \forall \alpha \in \langle 0, 1 \rangle^n \text{ and } \forall 1 \leq i \leq m O_i(\alpha) = f^{(i)}(\alpha).$$

For complexity measures of such Boolean circuits, there are two criteria ,the size and the depth. The size of a circuit C , denoted by $size(C)$, is the number of gates in C and the depth of C , denoted by $depth(C)$ is the number of gates in the longest directed path connecting an input gate to an output gate.

A DNA computation proceeds in two phases: Encoding of the problem by generating a solution space and applying molecular operations. For Boolean circuit evaluation we also need these two phases, which both are discussed in this section.

For any Boolean circuit modeled as a directed graph $S(V, E)$, the encoding scheme is illustrated as follows: A DNA strand with length l is assigned to any gate j in the level i denoted by g_j^i . In addition for each intermediate gate g_j^i with two inputs from gates g_p^{i-1} and g_q^{i-1} in $(i - 1)$ th level, one strand with length $3l$ is also assigned and is called

link-strand. This strand contains of three sections: The complement strand corresponding to the left input in the level $i - 1$, the complement strand corresponding to the gate g_j^i , and the complement strand corresponding to the right input in the level $i - 1$. Generally, if g_p^{i-1} and g_q^{i-1} be the input gates of g_j^i , and x , y , and z be corresponding strands to gates g_p^{i-1} , g_q^{i-1} , and g_j^i , respectively, and \bar{x} , \bar{y} , and \bar{z} be complement strands of x , y , and z , then the link-strand of gate g_j^i is $\bar{x} \bar{z} \bar{y}$.

For example, consider the strands $\mathbf{z} = 5' - GGAAGAGTCCC - 3'$, $\mathbf{x} = 5' - GGGTAGAA GCCC - 3'$, and $\mathbf{y} = 5' - GGGTCTAGCCCC - 3'$ for the gates g_j^i and g_p^{i-1} and g_q^{i-1} , respectively. Therefore, the link-strand for g_j^i is

$$3' - CCCATCTTCGGGCCCTTCTCAGGGCCAGATCGGGG - 5'.$$

If we pour \mathbf{z} and link-strand into a test tube and with appropriate conditions for annealing then the following strand is constructed:

$$3' - CCCATCTTCGGGCCCTTCTCAGGGCCAGATCGGGG - 5' .$$

Every strand corresponding to a gate starts and ends with a specific pattern as a restriction site, such that these sites are cut by the restriction enzymes. In previous example all strands start with GGG and end with CCC which are restriction site of *SmaI* enzyme. Now, if we add strand $\mathbf{x} = 5' - GGGTAGAAGCCC - 3'$ in the test tube the above strand is transformed to the following strand:

$$\begin{array}{l} 5' - GGGTAGAAGCCC GGAAGAGTCCC - 3' \\ 3' - CCCATCTTCGGGCCCTTCTCAGGGCCAGATCGGGG - 5' . \end{array}$$

With adding the restriction enzymes *SmaI*, the above strand is cut into two strands

$$\begin{array}{l} 5' - GGGTAGAAGCCC - 3' \\ 3' - CCCATCTTCGGG - 5' , \end{array}$$

and

$$\begin{array}{l} 5' - GGAAGAGTCCC - 3' \\ 3' - CCTTCTCAGGGCCAGATCGGGG - 5' . \end{array}$$

In order to evaluate the Boolean circuit network with the given input gates, first a tube T^0 is created which contains strands of length l each of which corresponds to only these input gates with value 1. A test tube is created for each level k ($1 \leq k \leq d = \text{depth}(C)$) which contains the strands corresponding to g_j^k for $j = 1$ to n_k (n_k is the number of gates in level k). During the laboratory operations, the contents of T^i which corresponds to level i is added to the test tube of the $i + 1$ th level. Finally, a complete double-stranded DNA sequence is created and this means that the output gates with value zero are found in the level $i + 1$. The DNA algorithm for evaluating of Boolean circuits C proceeds as follows for level $1 \leq k \leq d$:

1. Pour the contents of T^{k-1} into T^k . The strands are annealed at the appropriate position by the decreasing the temperature in the tube.
2. Add ligase enzyme to T^k in order to occur ligation between the double strands.
3. All the complete double-stranded DNA sequences which show the zero value of outputs gates are eliminated from tube T^k by running on gelelectropherese.
4. The incomplete DNA strands are cut by enzyme *SmaI* from the restriction site of this enzyme.
5. These strands are melted and the non-complement parts are kept and the other strands are ignored. This step can be performed by amplification of non-complement section of these strands by PCR.

Eventually, after repeating the above operations for all the levels, if T^d (the tube in last level) is not contained any complete double-stranded DNA strand, it can be induced that the final output for the circuits is one; otherwise is zero.

From [Amos et al., 1999, Deaton et al., 1999, Deaton et al., 1998] errors in the hybridization and ligation of the library strands are errors in the DNA computation. Therefore, the DNA sequences corresponding to our encoding can be generated by a genetic algorithm which minimizes the potential errors in sequences for reliable molecular operations [Deaton and Rose, 2000, Shin et al., 2002]. For this reason, a genetic algorithm is designed and employed for generating the DNA sequences used in this paper.

3 Computer Based Simulation

We illustrate with a small experience circuit how the simulation is designed. Consider the circuit in Figure 1. The $x_1, x_2, x_3,$ and x_4 are input gates and $p, q,$ and r are NAND gates, and also m is output gate. The four input variables at the first level are represented as:

$$\begin{aligned}\delta_1 &= 5' - GGGGATTAACCC - 3', \\ \delta_2 &= 5' - GGGAAATGTCCC - 3', \\ \delta_3 &= 5' - GGGCAGCAGCCC - 3', \\ \delta_4 &= 5' - GGGTTTAGACCC - 3'.$$

The NAND gates $p, q,$ and r are also represented by

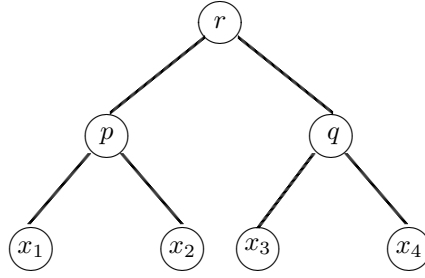


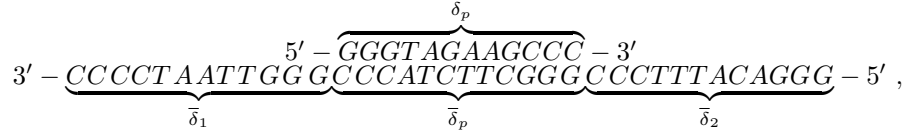
Figure 1: Example

$$\delta_p = 5' - GGGTAGAAGCCC - 3',$$

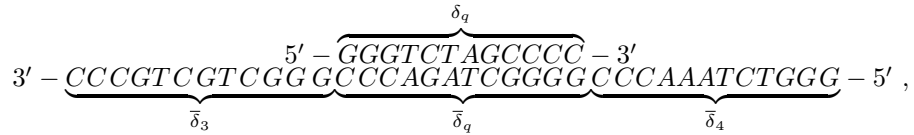
$$\delta_q = 5' - GGGTCTAGCCCC - 3',$$

$$\delta_r = 5' - GGGAAGAGTCCC - 3'.$$

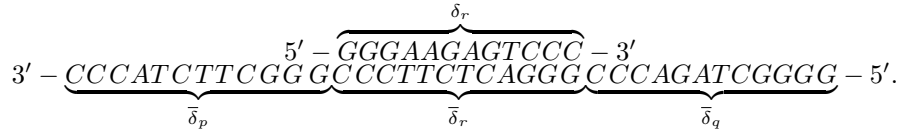
Now, we want to evaluate the circuits for inputs: $x_1 = 1$, $x_2 = 1$, $x_3 = 1$, and $x_4 = 0$. Therefore, the test tube T^0 is contained the strands δ_1 , δ_2 , and δ_3 , and T^1 contains corresponding strands to p and q and link-strands of them. Therefore, after hybridization and adding ligase enzyme, T^1 contains:



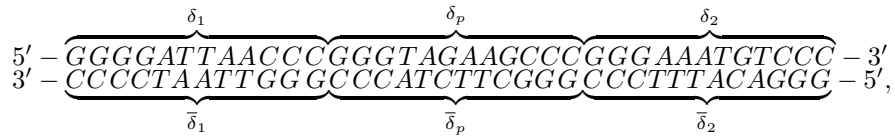
and



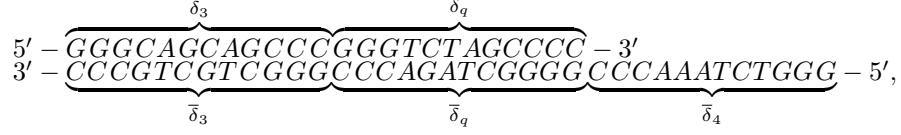
and T^2 contains corresponding strands to r :



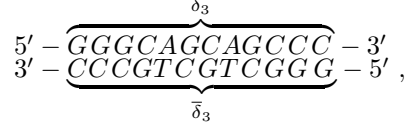
Now, the T^0 is poured into T^1 and after hybridization and ligation, T^1 contains:



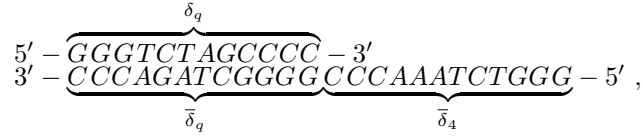
and



which the complete double-stranded DNA sequence is ignored and with adding restriction enzyme *SmaI* to T^1 , we have

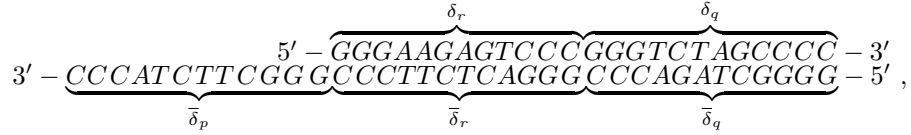


and



and with melting operation and ignoring the complement parts, T^1 contains only corresponding to q , *i.e.* δ_q .

Now, T^1 is poured into T^2 and after hybridization and adding ligase enzyme, T^2 contains:



and because we do not have any complete double DNA strands, therefore the result is evaluated as one.

4 Complexity analysis

Consider the depth and size of the Boolean circuit C be d and s , respectively. In DNA evaluation of Boolean circuit for each level of the circuit C three operations takes place: Annealing, melting and cleavage. Thus the total number of steps is $3d$, and circuit evaluation is performed in $O(d)$. On the other hand, the maximum number of DNA strands that is remain in the test tube after processing a single level is bound by $O(n)$, where n is the number of gates in the corresponding level, and the total number of strands in our operations are bounded by $O(s)$.

5 Conclusion

A DNA simulation of NAND Boolean circuit is presented. The model employs standard molecular techniques of sequences annealing, ligation, melting, cleavage, separation and detection by size. Our molecular algorithm is run in time proportional to the depth of circuit with space complexity is bounded by size of the circuit.

References

- [Adleman, 1994] Adleman, L., (1994) Molecular computation of solutions to combinatorial problems. *Science*, vol.266, pp.1021–1029.
- [Amos and Dunne, 1997] Amos, M. and Dunne, P., (1997) DNA simulation of Boolean circuits. Technical Report CTAG-97009, Department of Computer Science, University of Liverpool.
- [Amos et al., 1999] Amos, M., Gibbons, A., and Hodgson, D., (1999) Error-resistant implementation of DNA computations. In: *DNA Based Computers II* (Landweber, L. and Baum, E., Eds.), American Mathematical Society, Providence, vol.44, pp.151–162.
- [Beaver, 1995] Beaver, D., (1995) Computing with DNA. *Journal of Computational Biology*, vol.2, pp.1–13.
- [Braich et al., 2002] Braich, R., Chelyapov, N., Johnson, C., Rothmund, P., and Adleman, L., (2002) Solution of a 20–variable 3–sat problem on a DNA computer. *Science*, vol.296, pp.499–502.
- [Chang and Guo, 2003] Chang, W. and Guo, M., (2003) Solving the set cover problem and the problem of exact cover by 3-sets in the adleman-lipton model. *BioSystems*, vol.72, pp.263–275.
- [Chang et al., 2004] Chang, W., Ho, M., and Guo, M., (2004) Molecular solutions for the subset-sum problem on DNA-based supercomputing. *BioSystems*, vol.73, pp.117–130.
- [Deaton et al., 1998] Deaton, R., Garzon, M., Murphy, R., Rose, J., Franceschetti, D., and Stevens Jr, S., (1998) Reliability and efficiency of a DNA based computation. *Phys. Rev. Lett.*, vol.80, pp.417–420.
- [Deaton et al., 1999] Deaton, R., Murphy, R., Garzon, M., Franceschetti, D., and Stevens Jr, S., (1999) Good encodings for DNA-based solutions to combinatorial problems. In:

- DNA Based Computers II (Landweber, L. and Baum, E., Eds.), American Mathematical Society, Providence, vol.44, pp.247–258.
- [Deaton and Rose, 2000] Deaton, R. and Rose, J., (2000) Simulations of statistical mechanical estimates of hybridization error. In: Preliminary Proceedings of the Sixth International Meeting on DNA Based Computers (Condon, A. and Rozenberg, G., Eds.), Leiden University, Leiden Center for Natural Computing. pp.251–259.
- [Dunne, 1998] Dunne, P., (1998) Complexity of Boolean Networks. Academic Press, London.
- [Feynman, 1961] Feynman, R., (1961) There is plenty of room at the bottom. In: Miniaturization (Gilbert, D., Ed.), Reinhold, pp.282–296.
- [Gao et al., 1999] Gao, Y., Garzon, M., Murphy, R., Rose, J., Deaton, R., Franceschetti, D., and Stevens Jr, S., (1999) DNA implementation of nondeterminism. In: DNA Based Computers III (Rubin, H. and Wood, D., Eds.), American Mathematical Society, Providence, vol.48, pp.137–148.
- [Lipton, 1995] Lipton, R., (1995) DNA solution of hard computational problem. Science, vol.268, pp.542–545.
- [Ogihara and Ray, 1998] Ogihara, M. and Ray, A., (1998) DNA-based self-propagating algorithm for solving bounded-fan-in Boolean circuits. In: Third Conference on Genetic Programming, Morgan Kaufman Publishers, San Francisco, pp.725–730.
- [Ogihara and Ray, 1999] Ogihara, M. and Ray, A., (1999) Simulating Boolean circuits on DNA computers. Algorithmica, vol.2, pp.239–250.
- [Rothmund, 1996] Rothmund, P., (1996) A DNA and restriction enzyme implementation of turing machines. In: DNA base computers (Rubin, H. and Wood, D., Eds.), American Mathematical Society, Providence, vol.27, pp.75–119.
- [Shin et al., 2002] Shin, S., Kim, D., Lee, I., and Zhang, B., (2002) Evolutionary sequence generation for reliable DNA computing. In: Proceedings of the 2002 Congress on Evolutionary Computation (Fogel, D., El-Sharkawi, M., Yao, X., Greenwood, G., Iba, H., Marrow, P., and Shackleton, M., Eds.), IEEE Press, New Jersey, pp.79–84.