

A New Approach for Modelling Circuits Containing NAND Gates Using Biomolecular Computing

Mahnaz Kadkhoda

Department of Computer Engineering, University of Birjand, Birjand, Iran
mkadkhoda@birjand.ac.ir

Ali A. Pouyan, Member IEEE

Faculty of IT and Computer Engineering, Shahrood University of Technology, Shahrood, Iran
apouyan@ieee.org

Abstract

In the past few years, a lot of work has been done on simulating Boolean Circuits with biomolecular computation. In this paper, we present a new DNA-based evaluation algorithm for a Boolean circuit consists of NAND gates. This algorithm employs standard bio-molecular techniques. The contribution of this research is that the proposed model has been implemented using only three molecular operations. Furthermore, the number of passes in each level is decreased to less than half of existing models. Also the proposed implementation avoids the use of error-prone techniques such as PCR. These advantages have led to a faster, easier and more efficient algorithm. Time complexity of this algorithm is proportional to the depth of circuit.

1. Introduction

Molecular computing has been introduced in 1994 by Adleman [1]. He proposed an algorithm for solving Hamiltonian path problem using molecular operations. Adleman's experiment ushered in a new computational paradigm for several reasons. First, it showed that it is indeed possible to orchestrate individual molecules to perform computational tasks. Second, it showed the enormous potential of DNA molecules for solving problems beyond the reach of conventional computers that have been or may be developed in the future based on solid-state electronics.

There are two standards (criteria) to measure the efficiency of molecular algorithms: time complexity that is proportional to the number of molecular

operation on test tubes and space or volume complexity that is the maximum number of strings in all test tubes at any time.

Since Adleman's pioneering experiment, several authors attempted to present efficient DNA algorithms to solve hard problems [5] and simulating conventional computing models such as Turing machines [16], finite state automata [7] and splicing systems [15].

Another important approach in molecular computing is the implementation of Boolean circuits, because it would allow importing to the world of molecules the vast progress, which has been made on information processing in digital electronic computers. A successful implementation of Boolean Circuits would lead to the construction of ordinary computers in bio-molecular domain, particularly, computers capable to implementing massive parallelism. Furthermore, 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 electronic computers [14].

There are numbers of issues to be considered when simulating Boolean circuits. The first one is the choice of a computational basis. The standard basis consists of the AND, OR and negation. Another basis is the NAND gate. The second issue is feasibility of the methods and the third issue is the speed of the simulation.

Lipton [12] presented an early proposal for Boolean Circuit evaluation as a solution to SAT (satisfiability of Boolean formula). He used exhaustive search to implement his algorithm. In exhaustive search, all possible solutions are encoded by strands.

Then, the solution is found from this exponentially sized initial set by applying DNA operations. This approach is possible because of the inherent criteria of DNA strands as computing devices [13]. These criteria lie, on the one hand, in the potential of massive parallelism, which results in a greater number of computations per second in the sense that billions (or trillions) of DNA strands can be processed concurrently. On the other hand, it is because of large memory size that DNA molecules can provide for the entire computation processes.

Nevertheless, Hartmanis [8] shows that, although laboratory computations should work for the small problem sizes, the experiments do not realistically work for even modest problem size because of the vast amount of DNA molecules required for initialization. Exhaustive search needs high volume and much time and increases possibility of errors when implementing in laboratory. Ogihara and Ray [14] suggested a DNA algorithm for implementing AND-OR basis Boolean Circuits that run in time proportional to the size of the circuit. Proposed algorithm works without exhaustive search. Amos [3] described the first DNA-based simulation of NAND Boolean circuits and improved the implementation to have run time that is proportional to the depth of the circuit. Ahrabian [2] proposed another algorithm for NAND circuits. They claimed that their algorithm is easier and the number of operation used is less than the earlier versions. But they used error-prone techniques such as PCR (Polymerase Chain Reaction).

Since then, all other simulations of Boolean circuits are constructed by OR and AND gates [10]. Since it is well-known that the NAND functions provides a complete basis by itself [9] and any Boolean functions can be implemented only by NAND gates [6], we restrict our model to the simulation of NAND Boolean circuits.

The novel of our approach is that it is much easier to implement in the laboratory because the number of DNA operation used is much less than the previous versions in the literature. We use only three operations: Annealing, Ligation and Denaturing gelelectropherese. Furthermore the number of passes in each level is decreased to three passes that is less than half of previous ones. Therefore, it is much faster in comparison with other proposed algorithms. Furthermore, in the proposed simulation, “amplify” operation is not used because it is one of the error-prone operations.

The rest of this paper is organized as follows: section 2 presents the basic operation of DNA computing. In section 3, we give a brief introduction to circuits. Section 4 is devoted to describing the

proposed simulation method. We also show that how the model may be implemented in the laboratory. Section 5, presents an analysis of the model. Finally, we conclude the paper in section 6.

2. DNA Strand

DNA is a linear polymer of four repeating units (bases) A, G, C and T that may occur in any order. Two linear chains can associate with each other to form partial or complete duplex only when two conditions are fulfilled: first the two linear chains must have complementary base sequences, where A is complementary to T and C to G. second, strands must have opposite chemical polarities ($5' \rightarrow 3'$ and $3' \rightarrow 5'$). Thus, for making a duplex structure the two strands are antiparallel and complementary. When the two strands of DNA form a partial duplex, and at least one of the two strands has a recessed 3'-end, then that end can grow to extend the duplex structure by laying down new portion of complementary antiparallel chain using the longer chain as a template. This is primer extension. Extension stops when it reaches the end of the template, under the condition that there is sufficient supply of monomeric precursors of A, T, G and C in the solution. This is also the basis of self-propagation of DNA. The process of making a duplex molecule from two complementary antiparallel single strands of DNA is annealing (or Hybridization, or Renaturation). The reverse process in which a duplex is converted to two single strands is Denaturation. Denaturation is conveniently accomplished by heating while cooling under appropriate conditions causes annealing. The melting temperature of a duplex of a particular sequence is the temperature in which fifty percent of the DNA molecules in a given mixture denature. This temperature is a function of DNA length and base sequence. When two DNA strands of the same polarity are annealed to a template strand such that the two shorter strands are adjacent to each other with no gap, then it is possible to connect the two shorter strands (Ligation) to produce one longer strand. The reverse of this process, in which a duplex DNA is converted to at least two shorter duplex molecules is “Restriction”. All the above processes are accomplished by enzymes that have either evolved naturally or can be designed tailor-made [13].

3. Boolean Circuits

Our proposed simulation method is for a Boolean circuit with the following specifications [2]:

An n-input, m-input Boolean circuit is modelled 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 has 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 ($0 \dots k-1$). The input gates are appeared in the 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 $1 \dots k-2$). The input s 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 .

The 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:

$$f^{(i)}(I_n): \langle 0,1 \rangle^n \rightarrow \langle 0,1 \rangle : 1 \leq i \leq m \text{ if}$$

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

There are two criteria for Boolean circuits that are considered for standard complexity measures: the size of the circuit and the depth of the circuit. The size of the circuit C , denoted by $size(C)$, is the number of gates in C and the depth of C , denoted by $depth(C)$ is the length of the longest directed path in it.

4. Simulation Model

Molecular computation consists of two phases [4]: generating volumes and computation step. In the first stage, the needed strand is generated in tubes and in the next pass, DNA operations is applied on tubes to get result.

Therefore our simulation consists of two phases too:

- Initialization
- Level simulation

In this simulation, we try to present an easier and faster algorithm. In this method, we focus on inputs with value zero and encode them. Output of gates with value zero in each level is passed to next level as inputs. In the following we describe each phase in

more detail. A graphical representation is given in Figure 1.

4.1. Initialization

For initialization, we consider a tube T_i for each level i , $0 \leq i \leq \text{depth}(C)$. In this method, in spite of previous ones, we encode the inputs with value 0. Therefore, we consider a tube T_0 , consisting unique strands of length l , each of which corresponds to only that have the value 0. then, for each level $1 \leq k < \text{depth}(C)$, we create a tube T_k containing unique strands corresponds to each gate in that level. We denote the j th gate at level k by g_k^j . If gate g_k^j takes its input from gates g_{k-1}^m, g_{k-1}^n and x, y , and z be corresponding strands to gates g_{k-1}^m, g_{k-1}^n and g_k^j , respectively. Suppose that $\bar{x}, \bar{y}, \bar{z}$ be the complement string of x, y, z then: For each gate g_k^j we consider two strings: a string of length l and a linked-string of length $3l$ that is in the form of $\bar{x} \bar{z} \bar{y}$.

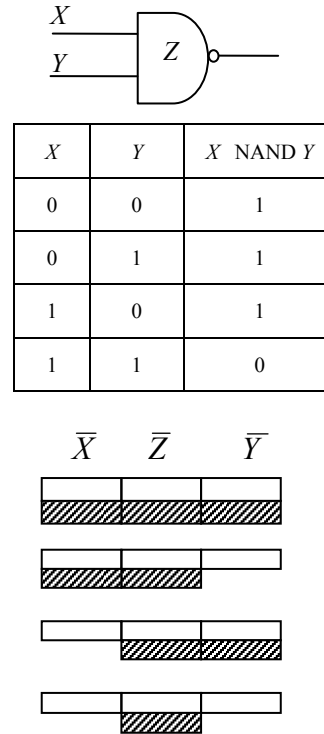


Figure 1. Simulation of NAND gate

4.2. level Simulation

In this path, for each level k , $1 \leq k \leq \text{depth}(c)$, we pour tube T_{k-1} into tube T_k . After decreasing the temperature, the strands are annealed. Then we prepare the condition for melting. After that, we separate all strands of length l representing gates with output 0. This subset forms the input to tube T_{k+1} .

The Molecular algorithm for Evaluation of Boolean circuit C (MEBC) proceeds as follows for each level $1 \leq k \leq \text{depth}(c)$:

1. Pour the contents of tube T_{k-1} into tube T_k . By decreasing temperature, the strands are annealed.
2. Add ligase enzyme to T_k in order to seal any "nicks".
3. Denature the strands and run T_k through a gel, retaining only those strands of length l . Retrieve the product and place it in an (empty) tube T_k . This tube forms the input to T_{k+1} . Now, we can proceed the simulation of level $k+1$.

Eventually, after repeating the above stages for all levels, if $T_{\text{depth}(c)}$ (the tube in the last level) is not contained any strand with length l , it can be deduced that the final output for the circuits is one; otherwise it is zero.

5. Algorithm Analysis

First, we analyze our algorithm in terms of feasibility of its molecular operations. In this model, we use standard molecular operations, that have been previously used, plus other operations by Ogihara and Ray [14] and Amos and E. Dunne [3].

With regard to the above discussion, the time and volume complexity of this algorithm is given in the following theorem:

Theorem: Algorithm MEBC for NAND-based Boolean circuits of size S , depth D is performed in $O(D)$ with volume complexity $O(S)$.

Proof: In this algorithm we use only three standard molecular operations in each level: Annealing, Ligation, Denaturing gelelectrophoresis.

Therefore, for evaluating a NAND Boolean circuit of size S , depth D , 3D computation steps is needed. Then, the time complexity of this algorithm is $O(D)$. On the other hand, the number of strands used in this simulation are bounded by $O(S)$.

Based on Gilbert-Varshamov theorem [11], One can show that there is a set of 1.6×10^{12} distinct 40 base oligonucleotide sequences such that

For any two sequences A and B , A disagrees with B and its complement at 10 positions.

No sequences contain the pattern that is cleaved by the Restriction enzyme.

Thus, 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 [13].

6. Conclusions

In this research, we described an abstract model for the simulation of NAND-based Boolean circuits using DNA. The novel of our method is that we only use three standard bio-molecular operations. In addition, there are three passes in each level. In contrast to the previous simulation methods which have five or seven passes, our proposed algorithm is more efficient. Furthermore, the proposed implementation of our model avoids the use of error-prone techniques, such as PCR. Thus, it reduces the degree of physical manipulation of tubes of DNA considerably. This minimizes potential problems such as strand shear and material loss due to strands sticking to the surface of tubes.

In spite of previous algorithms, We implement this algorithm without using restriction operation, whereas others use this operation twice in each level.

Consequently, the proposed simulation method is much faster and easier to implement in laboratory.

In this research we assume that the fan-out of each gate and the number of output gates is one. The future trend of our research is focused on the DNA-based algorithms for evaluating circuits with fan-out greater than one.

7. References

- [1] L.Adleman, "Molecular computation of solutions to combinatorial problems", *Science* 266, 1994 , pp. 1021-1024.
- [2] H.Ahrabian, Nowzari-Dalini, , "DNA simulation of NAND Boolean circuits", *Advanced modelling and optimization'06*, 2004, pp.33-39 .
- [3] M.Amos, P.Dunne, "DNA simulation of Boolean circuits". *Technical report CTAG-97009*. University of Liverpool,1997.
- [4] E.Bach, A.Glaser and C.Tanguay, "DNA models and algorithms for NP-complete problems". In proceedings of the 11th annual conference on structure in complexity theory, 1996, pp.290-299.
- [5] W.Chang, M.Guo, "Solving the set cover problem and the problem of exact cover by 3-sat in the Adleman-Lipton model". *Biosystems'72*, 2003, pp.263-275.
- [6] P.Dunne, "The complexity of Boolean Networks", *Academic Press*, 1998.
- [7] Y.Gao, M.Garzon, R.Murphy, J.Rose, R.Deaton, D.Franceschetti and S.Stevens, "DNA implementation of non-determinism". *DNA based computers III*, American Mathematical Society, Providence'48, 1999.
- [8] J.Hartmanis, "On the weight of computation". *Bulletin of the European Association for Theoretical Computer Science'55*, 1995, pp.136-138.
- [9] T.Head, "Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behaviours", *Bulletin of Mathematical Biology'49*(6), 1987, pp.737-752.
- [10] M. Kadkhoda, A.Pouyan, "A linear order complexity algorithm for evaluating bounded fan-in circuits using molecular computing", *WSEAS Transaction on Computers'5*, 2006, pp.2793-2798.
- [11] J.V.Lint, "Introduction to coding theory", *Springer Verlag*,1991.
- [12] Lipton, R., 1995. DNA solutions of hard computational problems. *Science'268*, pp.542-545.
- [13] M.Ogihara, A.Ray, "Executing parallel logic with DNA". In Proceedings IEEE congress on evolutionary computation, 1999, pp.972-979.
- [14] M.Ogihara, A.Ray, "Simulating Boolean circuits on DNA computers" . *Algorithmica'2*, 1999,pp.239-250.
- [15] G.Paun, G.Rozenberg and A.Salomaa, "DNA computing: new computing paradigms". *Springer Verlag*,1998.
- [16] P.Rothemund, "A DNA and restriction enzyme implementation of Turing machines". *DNA based computers*, American Mathematical Society, Providence'27, 1996, 75-119.