American Journal of Applied Sciences 5 (12): 1758-1763, 2008
ISSN 1546-9239
© 2008 Science Publications

Simulation of Parallel Logical Operations with Biomolecular Computing

[1]Mahnaz Kadkhoda and [2]Ali A. Pouyan
[1]Department of Computer Engineering, University of Birjand, Birjand, Iran
[2]School of IT and Computer Engineering, Shahrood University of Technology, Shahrood, Iran

Abstract: Biomolecular computing is the computational method that uses the potential of DNA as a parallel computing device. DNA computing can be used to solve NP-complete problems. An appropriate application of DNA computation is large-scale evaluation of parallel computation models such as Boolean Circuits. In this study, we present a molecular-based algorithm for evaluation of Nand-based Boolean Circuits. The contribution of this paper is that the proposed algorithm has been implemented using only three molecular operations and the number of passes in each level is decreased to less than half of previously addressed in the literature. Thus, the proposed algorithm is much easier to implement in the laboratory.

Key words: DNA computing, simulation, Boolean circuit, parallel computation, modeling, complexity

INTRODUCTION

Molecular computing was emerged in 1994 by original paper of Adleman[8]. 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.

Since Adleman's pioneering experiment, several authors attempted to present efficient DNA algorithms to solve hard problems[18] and simulating conventional computing models such as Turing machines[15], Finite state automata[19], splicing system[3].

Lipton[16] presented on early proposal for Boolean Circuit evaluation as a solution to SAT (Boolean formula satisfiability). Lipton and Adleman used exhaustive search to implement their algorithm. In exhaustive search, all the possible solutions are encoded by strands. Then the solution can be obtained from the exponentially sized initial set, by applying DNA operations. This approach is possible because of the inherent criteria of DNA strands as computing devices[12].

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[6] 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. For example, Hartmanis shows that a mass of DNA greater than that of the earth would be required to solve a 200-city instance of the Hamiltonian path problem. In both Adleman's APP and Lipton's SAT algorithms; the total volume of DNA present in a test tube at any time of the computation grows exponentially as a function of input size. As a result, their algorithms can handle instance size of up to 70 which are within the reach of silicon-based computers. Therefore it is urgent to find applications where DNA computers outperform silicon-based computers.

In spite of all efforts has been done to propose algorithms with low rate of volume[1,9], It is realized that NP-Complete problems may not be best suited for DNA computing. But other classes of problems are identified in which DNA based computation has real advantages. The best subject could be the evaluation of parallel computing models and Boolean circuits appears to be one of such problems.

Boolean circuits embody the notion of massively parallel signal processing and are frequently encountered in many parallel algorithms. Many

Corresponding Author: Mahnaz Kadkhoda, Department of Computer Engineering, University of Birjand, Birjand, Iran

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[10]. The implementation of Boolean circuits would allow importing to the world of molecules the vast progress that has been made on information processing in electronic computers. A successful implementation of Boolean Circuits would lead to the construction of ordinary computers in bio-molecular, particularly the construction of parallel computers.

There are numbers of issues to be considered when simulating Boolean circuits. The first one is the choice of a computational basis. The standards 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.

Ogihara and Ray[10] suggested a DNA algorithm for implementing AND-OR basis Boolean Circuits that runs in time proportional to the size of the circuit. Their proposed algorithm works without exhaustive search. Amos et al.[13] 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 and Nowzari[4] proposed another algorithm for NAND circuits. They claimed that their algorithm is easier and the number of operation used is less than before. But they used error-prone techniques such as PCR (Polymerase Chain Reaction).

Since then, all simulation models of Boolean Circuits has been constructed by OR and AND gates[12,11,5].

Since it is well-known that the NAND functions provide a complete basis by itself and any Boolean functions can be implemented only by NAND gates[17,14], we restrict our model to the simulation of NAND Boolean circuits.

The contribution of this research is that the proposed approach is much easier for implementing in the laboratory, because the number of DNA operation used is much less than other models reported 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 previously reported ones. Therefore, the proposed algorithm is much faster in comparison with algorithms before proposed. Also, in this simulation, Amplify operation is not used because it is one of the most error-prone operations.

THE PRIMITIVE OPERATIONS IN DNA COMPUTATIONS

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′->3′ and 3′->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 into 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[12].

NAND-BASED BOOLEAN CIRCUIT

In this study the simulation model has been proposed based on a Boolean circuit with the following specification[4]: An n-input, m-input 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: In, the inputs of the circuit which there are exactly n; G, the internal gates and Om, the outputs which there are exactly m. Each input vertex of In has in-degree 0 and are associated with a single Boolean variable xi from a given Boolean function. Each internal gate and output gate has in-degree 2 and is associated with the Boolean operation NAND. The internal gates will also have out-degree 1. The m distinguished output gates Om are conventionally regarded as having out-degree = 0.

A Boolean circuit contains k levels (0…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...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 <0, 1> to the input In ultimately induces Boolean values at the output gates Om.

The n-input, m-output Boolean circuit C is said to compute an n-input, m-output Boolean function f, f (In): <0,1>n → <0,1>m, on other words, for any input we have :

f (i) (In): <0,1>n → <0,1> : 1≤ i ≤m if α ∈<0,1>n and 1 ≤ i ≤m Oi (α) =f (i) (α).

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.

IMPLEMENTING IN LABORATORY

Molecular computation consists of two phases[2]: 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:

- Initial Pass
- Level implementation

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.

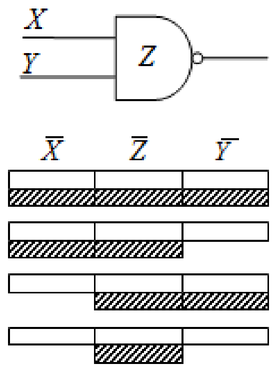
Initial pass: For initialization, we consider a tube Ti for each level i, 0≤i≤ depth (C). In this method, in spite of previous ones, we encode the inputs with value 0. Therefore we consider a tube T0, consisting unique strands of length l, each of which corresponds to only those having value 0. Then, for each level 1≤k< depth(C), we create a tube Tk containing unique strands corresponds to each gate in that level. We denote the jth gate at level k by gjk . If gate gik takes its input from gates g^{m}_{k-1}, g^{n}_{k-1} and x, y and z be corresponding strands to gates g^{m}_{k-1}, g^{n}_{k-1} and gik, respectively. Suppose that $\overline{x},\overline{y},\overline{z}$ be the complement string of x, y, z. Then, For each gate gik we consider two strings: a string of length l and a linked-string of length 3l that is in the form of $\overline{x}\,\overline{z}\,\overline{y}$.

Level implementation: In this path, for each level k, 1≤ k ≤ depth(C), we pour tube T_{k-1} into tube T_k. After decreasing the temperature, the strands are annealed. This process is showed for one gate in Fig. 1. 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≤ k ≤ depth(C):

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

Eventually, after repeating the above stages for all levels, if Tdepth(C); the tube in the last level, does not contain any strand with length l, it can be considered that the final output for the circuits is one; otherwise is zero.



X	Y	X NAND Y
0	0	1
0	1	1
1	0	1
1	1	0

Fig. 1: Simulation of NAND gate

COMPUTER BASED SIMULATION

We illustrate with a small circuit how the simulation works. Consider the circuit in Fig. 2. x1, x2, x3 and x4 are inputs and p, q and r are gates. We consider the following strands for input variables:

μ_1: 5′ -GATTACGAAC- 3′

μ_2: 5′ -CTACCCTGCT- 3′

μ_3: 5′ -TGCATCTTGG- 3′

μ_4: 5′ -GCCTACGTCA- 3′

The gates p, q and r are also represented by

μ_p: 5′ -ATCGGCTAAG- 3′

μ_q: 5′ -CTGTCGAATG- 3′

μ_r: 5′ -TTAGCGGTAC- 3′

We evaluate the circuit with inputs: x1 = 1, x2 = 0, x3 = 1 and x4 = 1. Therefore the tube T0 contains μ2 and T1 contains μp, μq and linked-strands of them. Now, the tube T0 is poured into T1. After hybridization and ligation, T1 contains:

w1:
$$\overbrace{\text{ATCGGCTAAG}}^{\mu_p}\overbrace{\text{CTACCCTGCT}}^{\mu_2}$$
$$\underbrace{\text{CTAATGCTTG}}_{\bar{\mu}_1}\underbrace{\text{TAGCCGATTC}}_{\bar{\mu}_p}\underbrace{\text{GATGGGACGA}}_{\bar{\mu}_2}$$

w2:
$$\overbrace{\text{CTGTCGAATG}}^{\mu_q}$$
$$\underbrace{\text{CGTAGAACCG}}_{\bar{\mu}_3}\underbrace{\text{ACAGCTTACCG}}_{\bar{\mu}_q}\underbrace{\text{GATGCAGT}}_{\bar{\mu}_4}$$

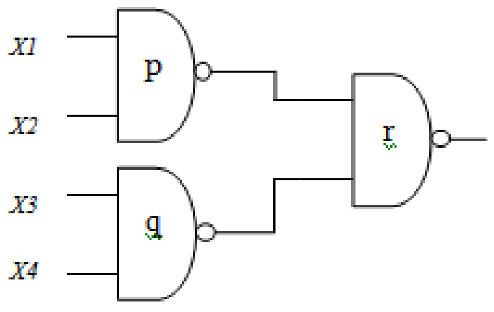


Fig. 2: Simulation of NAND gate

The temperature is increased, then the strands are melted. We have:

$$\overbrace{\text{ATCGGCTAAG}}^{\mu_p}\overbrace{\text{CTACCCTGCT}}^{\mu_2}\ ,\quad \overbrace{\text{CTGTCGAATG}}^{\mu_q}$$

and

$$\underbrace{\text{CTAATGCTTG}}_{\bar{\mu}_1}\underbrace{\text{TAGCCGATTC}}_{\bar{\mu}_p}\underbrace{\text{GATGGGACGA}}_{\bar{\mu}_2}$$

$$\underbrace{\text{ACGTAGAACC}}_{\bar{\mu}_3}\underbrace{\text{GACAGCTTACCG}}_{\bar{\mu}_q}\underbrace{\text{GATGCAGT}}_{\bar{\mu}_4}$$

We eliminate all the strands with length more than l. Therefore, the tube T_1 containing the strand μq forms the input to next level. Now, T_1 is poured into T_2. The strands are annealed. After Ligation, T_2 contains:

w3:
$$\overbrace{\text{TTAGCGGTAC}}^{\mu_r}\overbrace{\text{CTGTCGAATG}}^{\mu_q}$$
$$\underbrace{\text{TAGCCGATT}}_{\bar{\mu}_p}\underbrace{\text{CAATCGCCAT}}_{\bar{\mu}_r}\underbrace{\text{GGACAGCTTAC}}_{\bar{\mu}_q}$$

After melting the strands would be as follow:

$$\overbrace{\text{TTAGCGGTAC}}^{\mu_r}\overbrace{\text{CTGTCGAATG}}^{\mu_q}$$

$$\underbrace{\text{TAGCCGATTC}}_{\bar{\mu}_p}\underbrace{\text{AATCGCCATG}}_{\bar{\mu}_r}\underbrace{\text{GACAGCTTAC}}_{\bar{\mu}_q}$$

We remove all the strands with length more than l. finally, after doing above stages since T_2 is not contained any strands with length l, we can induce that the final output of circuit is one.

ALGORITHM ANALYSIS

At first, we analyze the proposed algorithm in terms of feasibility of its molecular operations. In this model, we use standard molecular operations that are used by Ogihara and Ray[10] and Amos and E. Dunne[13].

There are two standards (criterion) to measure the efficient of molecular algorithms: time complexity, which is proportional to the number of molecular operations on test tubes and space or volume complexity, which is the maximum number of strings in all test tubes at any time. The time and volume complexity for the proposed algorithm are given by the following theorem:

Theorem: Algorithm MEBC for NAND_based Boolean circuits of size S and 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, Legation and Denaturing gelelectropherese. Therefore, for evaluating a NAND Boolean circuit of size S and 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[7], One can show that there is a set of 1.6×1012 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 beyond the reach of digital computers[12].

CONCLUSION

In this research, we described an abstract model for the simulation of NAND-based Boolean circuits using DNA. This model is implemented in linear order of time and volume complexity and can be considered as a “killer” application in DNA computing.

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 previous simulation which have five or seven passes. Also, the proposed implementation of our model avoids using the error-prone techniques, such as PCR. Thus, it considerably reduces the degree of physical manipulation of tubes of DNA.

This minimizes potential problems such as strand shear and material loss due to strands sticking to the surface of the tubes. Despite of existing algorithms, We have implemented the proposed algorithm without using restriction operation.

Furthermore, our simulation method is much faster and easier to implement in laboratory. In this research, we consider that the fan-out of each gate and the number of output gates, is one. In the future work, we will present a DNA-based algorithm for evaluating circuits with fan-out greater than one.

But, we have not yet attempted to physically realize our model in laboratory. We acknowledge the substantial practical difficulties in implementing the model for even small circuits and we emphasize that much more work is needed to be done on establishing the error-resistance of basic operations.

REFERENCES

1. Baum, E. and D. Boneh, 1999. Running Dynamic Programming Algorithms on a DNA Computer. In: DNA Based Computers II, Landweber, L. and E. Baum (Eds.). Amer. Math. Soc. DIMACS Series, 44: 77-80.
2. Bach, E., A. Glaser, C. Tanguay, 1996. DNA models and algorithms for np-complete problems. In: Proceedings of the 11 Annual Conference on Structure in Complexity Theory, pp: 290-299.
3. Paun, G., G. Rozenberg and A. Salomaa, 1998. DNA Computing: New Computing Paradigms, Springer-Verlag, New York.
4. Ahrabian, H. and A. Nowzari-Dalini, 2004. DNA simulation of nand boolean circuits. Adv. Modeling Optimizat., 6: 33-39.
5. Ahrabian, H., M. Ganjtabesh and A. Nowzari-Dalini Abbas, 2005. DNA algorithms for an unbounded fan-in Boolean Circuit. Bio. Syst., 82: 52-60.
6. Hartmanis, J., 1995. On the weight of computation. Bull. Eur. Assoc. Theor. Comput. Sci., 55: 136-138.
7. Van Lint, J., 1991. Introduction to Coding Theory. Springer, Verlag, New York.
8. Adleman, L., 1994. Molecular computation of solutions to combinatorial problems. Science, 266: 1021-1024.
9. Adleman, L., P. Rothemund, S. Roweis and E. Winfree. On applying molecular computation to the Data.
10. Encryption Standard. In: DNA Based Computers II, Landweber, L. and E. Baum (Eds.). Amer. Math. Soc. DIMACS Series, 44: 31-44.
11. Ogihara, M. and A. Ray, 1999. Simulating Boolean circuit on DNA computers. Algorithmica, 2: 239-250.

12. Kadkhoda, M. and Ali A. Pouyan, 2006. A linear Order Complexity Algorithm for evaluating bounded fan-in circuits using molecular computing. WSEAS Transact. Comput., 5: 2793-2798.
13. Ogihara, M. and A. Ray, 1999. Executing parallel logical with DNA. In: Proceeding Cong. Evolutionary Computation, IEEE, pp: 972-979.
14. Amos, M. and P. Dunne, 1997. DNA Simulation of Boolean Circuits, Technical Report CTAG-97009, Department of Computer Science, University of Liverpool.
15. Paul E. Dune, 1988. The Complexity of Boolean Networks, Acodemic Press.
16. Rothemund, P., 1996. A DNA and restriction enzyme implementation of Turing machines. In: DNA Base Computers, American Mathematical Society, Providence, 27: 75-119.
17. Lipton, R., 1995. DNA solutions of hard computational problems. Science, 268: 542-545.
18. Thomas Head 1987. Formal language theory and DNA: An analysis of the generative capacity of specifif recombinant behaviors. Bull. Math. Biol., 49: 737-752.
19. Chang, W. and M. Guo, 2003. Solving the set cover problem and the problem of exact cover by 3-sat in the adleman-lipton model. Biosystems, 72: 263-275.
20. Gao, Y., M. Garzon, R. Murphy, J. Rose, R. Deaton, D. Franceschetti and S. Stevens, 1999. DNA implementation of nondeterminism, In: DNA Based Computers III, American Mathematical Society, Providence, 48: 137-148.