©2010 International Journal of Computer Applications (0975 - 8887)
Volume 1 – No. 22

An Efficient Algorithm for Constructing DNA Boolean Circuit

Michael Arock
Dept. of Computer Applications, National Institute of Technology, Trichy, Tamilnadu, 620 015, India

R.Ponalagusamy
Department of Mathematics,, National Institute of Technology, Trichy, Tamilnadu, 620 015, India

B.S.E.Zoraida
Dept. of Computer Applications, National Institute of Technology, Trichy, Tamilnadu, 620 015, India

ABSTRACT

Computation using biological Deoxyribonucleic acid (DNA) strand is increasingly found to be a viable proposition. A unique generalized efficient algorithm for forming any Boolean circuit strand is proposed in this paper. The implementation of the Boolean circuit using the proposed algorithm is also presented. This Boolean circuit requires only one kind of bio-operation at each level. Further, this paper adopts a uniform representation for logical 0 and 1 for all Boolean circuit. Simulation to validate the efficient algorithm is also presented in this work.

Categories and Subject Descriptors

1.1 **[Computer Science / Information technology] : BioComputing –** *DNA Computing*

General Terms

Algorithms, Design

Keywords

DNA computing, Boolean circuit, Truth table, Traffic light signaling, Graph

1.INTRODUCTION

Ever since Adleman has published a paper on molecular computation for solving Hamiltonian path problem (HPP), attempts are being made to utilize DNA manipulations for solving computationally difficult problems [1]. Several models of computation using bio-molecular methods have also been proposed. Some of the models for DNA computation are Turing machine [12], Sticker model [13], Splicing systems [4], Surface-based computing [7,14,16] and Boolean circuits [10,11].

The inherent parallelism in DNA was utilized before by many researchers in constructing Boolean operators. Amos *et a*l described the simulation of a bounded fan-in Boolean circuit with NAND gate, which takes time proportional to the depth of the circuit for computation. Ogihara *et al.*, proposed a bounded fan-in Boolean circuit functioning in O(1)-time complexity[10]. Ogihara *et al*., proposed the building of DNA-based Boolean circuits for a semi-unbounded fan-in Boolean circuit [11]. Subsequently, Erk, developed an abstract DNA model for simulating Boolean circuits by finite splicing systems [4]. The main drawback of this model is that the rules need to be altered with the complexity of the Boolean circuit. Mulawka *et al*., proposed another simulation of the NAND gate using the Fok I enzymes of nuclease class II [9]. Ahrabian *et al*. proposed a different construction of NAND gate and the same authors presented a DNA algorithm for solving an unbounded fan-in Boolean circuit in O(1)-time complexity [2,3]. Liu *et al.* presented a theoretical model of the NAND gate through the induced hairpin formation [8]. Jianzhong *et al.* suggested reusable logic gates for AND and OR functions using molecular beacon [6]. Zoraida *et al.* has proposed a generalized design methodology for realizing any Boolean circuit using the truth table of the Boolean circuit [17,18]. Here the scanning (top to bottom, bottom to top) the truth table of the corresponding Boolean circuit has to be altered accordingly to get the efficient Boolean circuit strand with shortest length.
In this context, this paper has attempted to construct a graph, using the truth table. The longest path of the graph is utilized to construct Boolean circuit strand having shortest length.
The rest of the paper is organized as follows: Section 2 describes about molecular beacon, blocker and graph representation of truth table. Section 3 gives the proposed design algorithm for generating the strands for Boolean circuit with and time complexity analysis. Also, it deals with a real time example that needs Boolean circuitry and applies the algorithm. Section 4 presents the implementation of Boolean circuit. Section 5 explains the simulation of the Boolean circuit using the proposed method. Conclusion is given in section 6.

PRELIMINARIES

1.1Molecular Beacon

Molecular beacons (MB) are single-stranded oligonucleotide hybridization probes that possess a stem and a loop structure. The loop has a complementary probe sequence of a target sequence. The stem is formed by annealing with the complementary sequence present in either side of the probe sequence. A fluorophore and a quencher are linked to the two ends of the stem. The two moieties are kept in close proximity to each other by the stem, enabling fluorescence of the fluorophore to be quenched through energy transfer and at this point the MB is “dark”. When the MB encounters its target DNA molecule, it

undergoes a spontaneous conformational reorganization that forces the stem apart so that the fluorophore and quencher moves away. So there is a transition from "dark" to fluorescence "bright" in MB. Molecular recognition specificity is one of the major advantages of MB. They are highly target-specific to the extent that they ignore target sequences that differ even by a single nucleotide [5,15]. The hybridization of the MB with the target is shown in Figure 1.
MB can be one among the choices for the inputs to the proposed Boolean circuit.

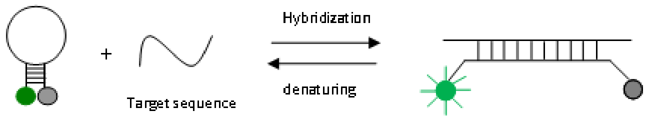

Figure 1. Hybridization of the MB with the target

2.2 Blocker

The hybridization of input strand to the Boolean circuit strand is necessary for implementing logical Boolean circuits. However, hybridization should be prevented in certain locations of the Boolean Circuit strand. Hence, a DNA sequence is used as a blocker in the Boolean Circuit. In this paper, the assigned blocker sequence is "GGGGGG". Input strands should be different from the blocker sequence. The blocker is denoted by a circle and an "X" through it in the subsequent figures, and by an asterisk (*) in the text..

2.3 Graph representation

The proposed efficient algorithm represents the truth table as a graph. It considers only those input bit strings whose output is 1. All such strings are represented as nodes. Those strings are henceforth known as 1-output bit strings.

2.3.1 Overlapping and Edge Formation

Two 1-output strings are said to overlap when the suffix of *n*-1 bits of first string matches exactly with the prefix of *n*-1 bits of the second. In the proposed method, a directed edge from one node to another is introduced only when they overlap.

2.PROPOSED DESIGN OF BOOLEAN CIRCUIT USING DNA

Each Boolean variable *I* can take two values *I*=0 and *I*=1 which is denoted as I^0 and I^1. Each Boolean circuit has *n* inputs $I_1, I_2, \ldots, I_n$. We take only one strand to represent $I_1^0\ I_2^0, \ldots, I_n^0$ and another strand to represent both $I_1^1, I_2^1, \ldots\ I_n^1$. The respective complements of the chosen strand are employed to represent $\overline{I_1^0}, \overline{I_2^0}, \ldots, \overline{I_n^0}$ and $\overline{I_1^1}, \overline{I_2^1} \ldots, \overline{I_n^1}$ .Thus we require only four strands for each Boolean circuit. A colored pattern representation instead of representing actual strand for $\overline{I_1^0}, \overline{I_2^0}, \ldots, \overline{I_n^0}$ and $\overline{I_1^1}, \overline{I_2^1} \ldots, \overline{I_n^1}$ is followed in this paper for visualizing the implementation. Figure 2 shows the patterns and logical inputs and their complements.

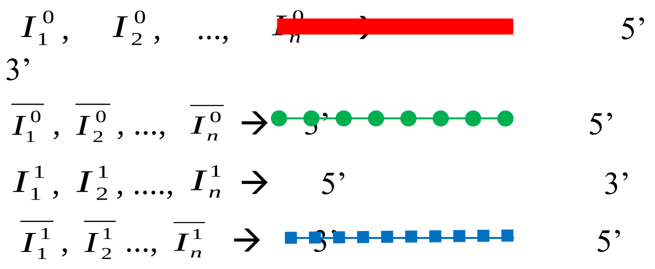
$I_1^0, I_2^0, \ldots, I_n^0 \rightarrow$
5'
3'
$\overline{I_1^0}, \overline{I_2^0}, \ldots, \overline{I_n^0} \rightarrow$
3'
5'
$I_1^1, I_2^1, \ldots, I_n^1 \rightarrow$
5'
3'
$\overline{I_1^1}, \overline{I_2^1} \ldots, \overline{I_n^1} \rightarrow$
3'
5'

Figure 2. Colored pattern representation of the Strand

A general form of truth table for *n*-input Boolean circuit is shown in Table 1. $I_1, I_2, \ldots, I_n$ are the inputs and R_1 to R_{2^n} are the outputs.

Table 1.Truth table for n-input Boolean circuit

1	I_1	I_2	**....**	I_n	R_1
2	I_1	I_2	....	I_n	R_2
⋮	⋮	⋮	⋮	⋮	⋮
2^n-1	I_1	I_2	....	I_n	R_{2^n-1}
2^n	I_1	I_2	....	I_n	R_{2^n}

Our aim is to form a single DNA strand capable of recognizing 1-output bit strings and rejecting 0-output bit strings. An algorithm is proposed that forms the single DNA strand. The algorithm considers only the rows having their output R_i = 1 (i = 1 to 2^n). These rows are represented as nodes of a graph. The nodes of the graph have the binary input bits ($I_1, I_2 \ldots I_n$) as labels of the nodes. The edges are formed for these constructed nodes if each node overlaps with other nodes in their binary input bits (as overlapping defined in subsection 2.3.1). From the graph obtained, the longest path is found so that as many as possible number of nodes is connected in the path. If the first longest path found doesn't involve all the nodes, for the remaining nodes which are not included in the longest path, next longest path from the remaining nodes is formed. This process is continued until all the nodes are in a path or other. After the paths are identified, consider the first longest path and find its starting node. The binary input bits of the starting node ($I_1, I_2 \ldots I_n$) are stored in an array. From the next node of the path onwards, only the I_nth bits are stored and a binary string is formed. The same is done for other paths also and their corresponding binary strings are appended to the first one separating the strings with a '*' between every pair of the binary strings. Then, we replace each value in the array with the corresponding complementary strand which is assigned earlier. The symbol '*' is replaced by the blocker sequence. The desired Boolean circuit strand is obtained. The design is given in the form of an algorithm below and the same is illustrated with the case studies:

Algorithm:

First, the truth table is represented as a graph, G = (V,E) (as stated in section 2.3)

1. Apply Depth-first Search (DFS) to find the longest path possible in the graph.

2. Check if it involves all the nodes. If so, form a binary string with the binary values (labels) of the nodes. For the first node alone, the entire binary value is considered and for the successive nodes in the path, the nth bit is appended to the so-far formed binary string. Once all nodes are processed, go to step 7.

3. Otherwise, form a binary string as stated in step 2 for the longest path found and make a subgraph with the left–out nodes.

4. Repeat steps 1 through 3 on the subgraph.

5. Repeat steps 1 through 4 until no node is left.

6. Combine the binary strings of all subgraphs by introducing '*' between every pair of the strings into the final single string.

7. Replace the 0's and 1's of the final string with the respective complementary strands and the ('*') with blockers, so that it can attract input strands.

Note:

1. The edges that make cycles in the graph are duly identified and ignored, as DFS employs back-edge determining technique.
2. The proposed algorithm produces a set of longest paths corresponding to subgraphs: $\bigcup_{i=1}^{k} V_i = V$ and $\bigcap_{i=1}^{k} V_i = \phi, k \geq 1$ where V_i and V are vertex sets of 'k'subgraphs and of the graph respectively. When k =1, a single longest path involving all nodes of the graph is formed.
3. The blockers serve to avoid binary strings corresponding to 0-output bit strings to be formed anywhere in the final string. Hence, the final DNA strand simulates the gate or the circuit precisely.

The strand formed by the algorithm is of minimum length due to the exploitation of overlapping feature.

3.1. Time Complexity Analysis

Theorem: The algorithm needs $\Theta(r^2)$ time, where r is the row-size of the truth-table.

Proof:

The algorithm has two stages: graph representation and production of the final strand. Graph representation takes $O(r^2)$ time, as every 1-output row is to be compared with every other 1-output row for making edges between any two nodes, in the worst case.

The next stage involves DFS and DFS requires $O(|V| + |E|)$ time for a graph, G=(V,E). In our strategy, $|V|$ is $O(r)$ and $|E|$ is $O(r^2)$. Though there can be more than one call for DFS for different subgraphs(when single longest path involving all nodes is not available), the amortized time complexity is O(|V|+|E|) only, as $\bigcup_{i=1}^{k} V_i = V$ and $\bigcap_{i=1}^{k} V_i = \phi$, where V_i 's are vertex sets of subgraphs.

So, combining the time complexities of two stages, the total time complexity is $O(r^2)$, in worst case. The lower bound for the minimum-length strand forming problem is $\Omega(r^2)$, as every row is to be compared with every other row of the truth table.

Hence, the worst case time complexity of our algorithm is $\Theta(r^2)$.

3.2. Case 1 Development of Boolean circuit for NAND gate

Consider the truth table of the NAND gate given in Table 2.

Table 2.Truth table of the NAND gate

Row No.	I_1	I_2	R
1	0	0	1
2	0	1	1
3	1	0	1
4	1	1	0

In constructing a graph for the above truth table shown in Table 2, only three nodes are considered as shown in Figure 3, Since the rows 1,2,3 has the output value R=1.

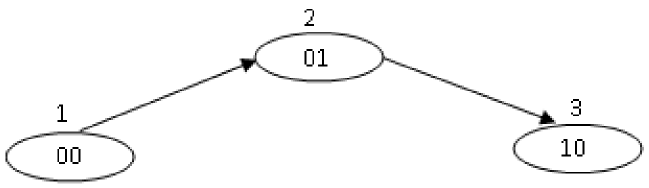

Figure 3. Graph representation for the NAND truth table

In order to form the edge, check for the overlap for the node 1 and node 2. Since there is an overlap an edge between node 1 and node 2 is added and also, there is an overlap between the node 2 and node 3 to form the edge between node 2 and node 3. Here, the start node is node 1 and in the array the binary input bits (00) is added. Node 2 is the next node in the path so that 1(I_2) is added in the array. Next is node 3 having 0 as I_2 value is appended to the array. As a result the array has (0,0,1,0). Replacing the values with the corresponding complementary strand, the NAND gate strand is obtained as shown in Figure 4.

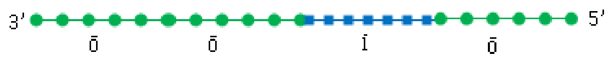

Figure 4. Gate strand for NAND gate

The same process is carried out for all other gates using their respective truth table and their gate strands can be obtained.

3.2 Case 2: Development of Boolean circuit for Traffic Light Signaling

Consider the problem to construct a Boolean circuit for controlling the traffic light signaling. Two roads, one is the main highway and the other is the secondary access road intersects each other as shown in Figure 5.

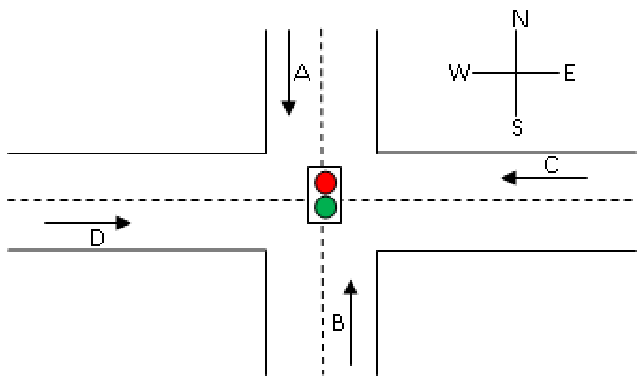

Figure 5. Traffic Light Signaling

Vehicle-detection sensors are placed along lanes *C* and *D* (main road) and lanes *A* and *B* (access road). These sensor outputs are low (0) when no vehicle is present and High (1) when a vehicle is present. The intersection traffic light is to be controlled according to the following logic:

a) The east-west (*E-W*) traffic light will be green whenever both lanes *C* and *D* are occupied.

b) The *E-W* light will be green whenever either *C* or *D* is occupied but lanes *A* and *B* are not both occupied.

c) The north-south (*N-S*) light will be green whenever both lanes *A* and *B* are occupied but *C* and *D* are not both occupied.

d) The *N-S* light will also be green when either *A* or *B* is occupied while *C* and *D* are both vacant.

e) The *E-W* light will be green when no vehicle is present

For the above problem the truth table is constructed having four inputs (*A*,*B*,*C*,*D*) and two outputs *E-W* and *N-S* as shown in table.

Table 3. Truth table for the traffic light signaling

Row No.	A	B	C	D	E-W	N-S
1	0	0	0	0	1	0
2	0	0	0	1	1	0
3	0	0	1	0	1	0
4	0	0	1	1	1	0
5	0	1	0	0	0	1
6	0	1	0	1	1	0
7	0	1	1	0	1	0
8	0	1	1	1	1	0
9	1	0	0	0	0	1
10	1	0	0	1	1	0
11	1	0	1	0	1	0
12	1	0	1	1	1	0
13	1	1	0	0	0	1
14	1	1	0	1	0	1
15	1	1	1	0	0	1
16	1	1	1	1	1	0

Since there are two outputs (*E-W*, *N-S*), it is required to construct two graphs for obtaining two strands one for the (*E-W*) direction and the other for the (*N-S)* direction to implement the traffic light signaling. For constructing the (*E-W*) graph consider only the rows having logical output 1 for the *E-W* direction. In the above table 3, rows 1, 2, 3, 4, 6, 7, 8, 10, 11, 12, 16 have their *E-W* output as 1, and so their inputs (*A*,*B*,*C*,*D*) are considered as nodes shown in Figure 6. To construct their edges, it is to identify the overlapping between every pair of nodes. If there is overlap between every pair of nodes an edge be placed between nodes of each pair. For example node 2 has overlap with node 3 and node 4. So, node 2 has edges to node 3 and to node 4.The *E-W* graph is shown in Figure 6 for the truth table shown in Table 3.

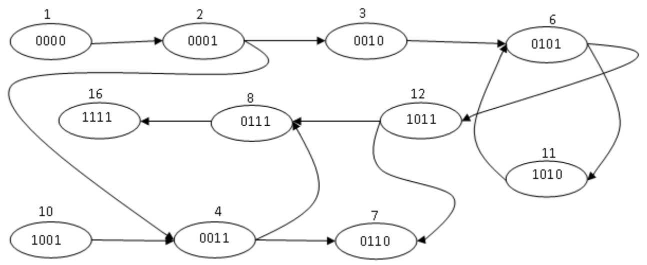

Figure 6. Graph representation for the *E-W* direction

In the graph shown in Figure 6 the longest path involves nodes 1→ 2→ 3→ 6→ 12→ 8→ 16. From the remaining nodes 10, 4, 7, 11, a path with nodes 10→4→7 can be constructed. The node 11 is left alone, as it has no overlap. Using the paths generated an efficient strand can be constructed for implementing *E-W* direction. In the first path the starting node is node 1. So that the inputs *A*,*B*,*C*,*D* of row 1 are added to an array followed by *D* input of the next node along the path. This process is carried out until the last node in the path is encountered. Now the array has

(0,0,0,0,1,0,1,1,1,1). The same process has to be carried out for the next path with a "*" symbol in between, so that array has (0,0,0,0,1,0,1,1,1,1, *, 1,0,0,1,1,0). The inputs (*A*,*B*,*C*.*D*) of node 11 is added to the array with a "*" symbol between which is the node left alone. The final array has the value (0,0,0,0,1,0,1,1,1,1,*,1,0,0,1,1,0,"*,1,0,1,0). Replacing the values with the corresponding complementary strand, the *E-W* strand is obtained as shown in Figure 7.

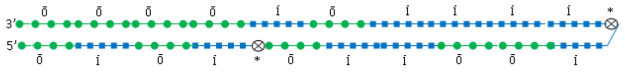

Figure 7. *E-W* strand

The same process is carried out for the *N-S* direction by considering the truth table shown in Table 3 with the inputs *A*,*B*,*C*,*D* and the *N-S* output. The graph obtained for the *N-S* direction is shown in Figure 8. The array has the value (1,1,1,0,0,0,"*",1,1,0,1,"*",0,1,0,0). Replacing the values with the corresponding complementary strand, the *N-S* strand is obtained as shown in Figure 9.

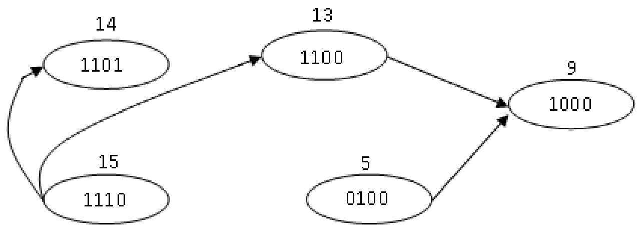

Figure 8. Graph representation for the E-W direction

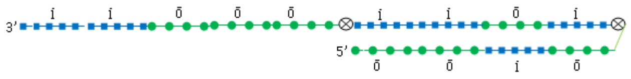

Figure 9. N-S strand

3.3 Inputs of the Boolean Circuit

Boolean circuit strand that has been obtained from executing the above algorithm will be fixed on the surface at the 3'end. The inputs for this Boolean circuit strand are the sequences representing the inputs of the Boolean circuit. The Boolean circuit for traffic light signaling has four inputs so that the input sequence has four strands which are the assigned input strands for the input variable as shown in Figure 10. In this paper molecular Beacon probe is used as the input. The four sequences I_1, I_2, I_3 and I_4 are placed in the loop of the Molecular Beacon (MB). Changes that occurred after introducing the MB represent the results of the Boolean circuit. In our simulation always the fluorescence represents 1 and the dark represents 0.

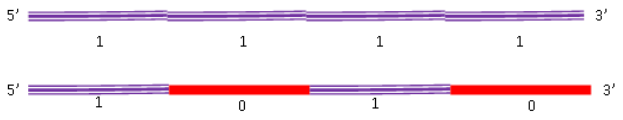

Figure 10. Few input strands for the Traffic light signaling Boolean circuit

When the number of inputs to the Boolean circuit is increased the appropriate probe can be chosen.

4. IMPLEMENTATION OF TRAFFIC LIGHT SIGNALING BOOLEAN CIRCUIT

To implement Boolean circuit, fixing the strands obtained from the algorithm for which the input is the truth table of the traffic light signaling. In this case, there are two strands one for the *N-S* direction and the other for the *E-W* direction are fixed on the surface so that they are immobilized. The input for the fixed strand is the two MB's having the same inputs (*A*,*B*,*C*,*D*) in the loop. The input MB is hybridized if it has desired target and becomes bright else it will be dark. The bright fluorescence light represents green light for the corresponding direction and the dark light represents red light for the particular direction.

5. SIMULATION OF TRAFFIC LIGHT SIGNALING BOOLEAN CIRCUIT

In the proposed method the two strands (*E-W*, *N-S*) as shown in Figure 7 and Figure 9 obtained after implementing the algorithm by giving the truth table as the input are fixed on the surface. Let the inputs for the Traffic light signaling Boolean circuit *A*, *B*, *C*, *D* be (1,1,1,1). Two MB's with the same input sequence on the loop is given as input to the two stands. The input MB given to *E*-W strand get hybridized after finding a target in the *E-W* strand indicating the green light signal for the *E-W* direction. The input MB to the *N-S* strand can not find its target sequence and it will be dark indicating a red signal for the N-S direction. The output of the Boolean circuit is depicted in Figure 10.

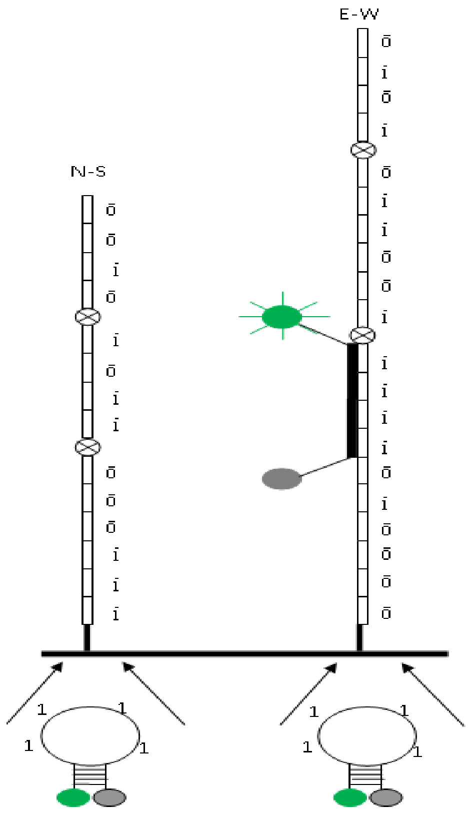

Figure 10. Simulation of Traffic light signaling Boolean circuit

6. CONCLUSION

A theoretical model using molecular beacons, for all the Boolean circuit with *n* inputs has been established in this paper. The application of the proposed algorithm to form any Boolean circuit has been illustrated. Compared to earlier models, the present work needs only one bio-operation to complete the computation. Case studies presented demonstrate amply that reusable and reliable logic gates can be developed with ease using the proposed algorithm. As MB's are highly target specific, the logic gates thus formed are reliable. Their specificity is to the extent that they distinguish target sequences by a single nucleotide. It is also to be noted that the Boolean circuits constructed can be reused for subsequent simulation. The logical 1 and logical 0 has been maintained the same for any logical gate in this paper.

7. REFERENCES

[1] Adleman, L., 1994. Molecular computation of solutions to combinatorial problems.Science 266, 1021–1029.

[2] Ahrabian, H., Ganjtabesh, M. and Nowzari-Dalini, A., 2005. DNA algorithm for an unbounded fan-in Boolean circuit. Biosystems 82, 52–60.

[3] Ahrabian, H. and Nowzari-Dalini, A., 2004. DNA simulation of NAND circuits. AMO Advanced Modeling and Optimization 6, 2.

[4] Erk, K., 1999. Simulating Boolean circuits by finite splicing. In: Proceedings of the Congress on Evolutionary Computation, vol. 2. IEEE Press, 1279–1285.

[5] Fang, X., Liu, X., Schuster, S. and Tan, W., 1999. Designing a novel molecular beacon for surface-immobilized DNA hybridization studies. Journal of the American chemical Society 121 (12), 2921–2922.

[6] Jianzhong, C., Zhixiang, Y., Wei, W., XiaoHong, S. and Linqiang, P., 2006. Towards reliable simulation of bounded fan-in Boolean circuits using molecular beacon. In: Proceedings of the World Congress on Intelligent Control and Automation, IEEE, Dalian, China, 3910–3914.

[7] Liu, Q.,Wang, L., Frutos, A.G., Condon, A.E., Corn, R.M., Smith, L.M., 2000. DNA computing on surfaces. Nature 403, 175–179.

[8] Liu, W., Shi, X., Zhang, S., Liu, X., Xu, J., 2005. A new DNA computing model for the NAND gate based on induced hairpin formation. Biosystems 77, 92–97.

[9] Mulawka, J.J.,Wasiewicz, P., Plucienniczak, A., 1999. Another logical molecular NAND gate system. In: Proceedings of the Seventh International Conference on microelectronics for Neural, Fuzzy and Bio-Inspired Systems, Granada, Spain, 340–346.

[10] Ogihara, M., Ray, A., 1998. DNA-based self-propagating algorithm for solving bounded-fan-in Boolean circuit. In: Proceedings of the Third Conference on Genetic Programming. Morgan Kaufman Publisher, San Francisco, . 725–730.

[11] Ogihara, M., Ray, A., 1999. Simulating Boolean circuits on a DNA computers. Algorithmica 25, 239–250.

[12] Rothemund, P., 1996. A DNA and restriction enzyme implementation of Turing machines. In: DIMACS Series, Proceedings of a DIMAC Workshop, AMS 27, pp. 75–119.

[13] Roweis, S., Winfree, E., Burgoyne, R., Chelyapov, N., Goodman, M., Rothemund, P., Adleman, L., 1998. A sticker based model for DNA computation. Journal of Computational Biology 5 (4), 615–629.

[14] Su, X., Smith, L.M., 2004.Demonstration of a universal surface DNA computer. Nucleic Acids Research 32 (10), 3115–3123.

[15] Tyagi, S., Kramer, F.R., 1996. Molecular beacon: probes that fluorescence upon hybridization. Nature Biotechnology 14, 303–308.

[16] Wang, L., Liu, Q., Frutos, A., Gillmor, S., Thiel, A., Strother, T., Condon, A., Corn, R., Lagally, M., Smith, L., 1999. Surface-based DNA computing operations: destroy and readout. Biosystems 52 (1), 189–191.

[17] Zoraida, B.S.E., Arock, M., Ronald, B.S.M. and Ponalagusamy, R., 2009. A novel generalized design methodology and realization of Boolean operations using DNA, Biosystems, vol.97, pp.146-153.

[18] Zoraida, B.S.E., Arock, M., Ronald, B.S.M. and Ponalagusamy, R., 2008. A novel generalized model for constructing reusable and reliable logic gates using DNA. In: Proceedings of the Fourth International Conference on Natural Computing, vol. 7. IEEE Press, China, pp. 353–357.