BioSystems 97 (2009) 146–153

Contents lists available at ScienceDirect

BioSystems

journal homepage: www.elsevier.com/locate/biosystems

A novel generalized design methodology and realization of Boolean operations using DNA

B.S.E. Zoraida[a,*], Michael Arock[a], B.S.M. Ronald[b], R. Ponalagusamy[c]

[a] Department of Computer Applications, National Institute of Technology, Trichy, Tamilnadu, 620 015, India
[b] Vaccine Research Centre, Bacterial Vaccine, TANUVAS, Chennai, Tamilnadu, 600 051, India
[c] Department of Mathematics, National Institute of Technology, Trichy, Tamilnadu, 620 015, India

ARTICLE INFO

Article history:
Received 5 November 2008
Received in revised form 27 May 2009
Accepted 27 May 2009

Keywords:
DNA computing
Molecular beacon
Adder
Subtractor
Logic gates
Boolean circuit

ABSTRACT

The biological deoxyribonucleic acid (DNA) strand has been increasingly seen as a promising computing unit. A new algorithm is formulated in this paper to design any DNA Boolean operator with molecular beacons (MBs) as its input. Boolean operators realized using the proposed design methodology is presented. The developed operators adopt a uniform representation for logical 0 and 1 for any Boolean operator. The Boolean operators designed in this work employ only a hybridization operation at each stage. Further, this paper for the first time brings out the realization of a binary adder and subtractor using molecular beacons. Simulation results of the DNA-based binary adder and subtractor are given to validate the design.

1. Introduction

Ever since Adleman (1994) 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. Several models of computation using DNA have been proposed earlier and they are Turing machine (Rothemund, 1996), Sticker model (Roweis et al., 1998), Splicing systems (Erk, 1999), Surface-based computing (Wang et al., 1999; Liu et al., 2000; Su and Smith, 2004) and Boolean circuits (Ogihara and Ray, 1998, 1999).

The inherent parallelism in DNA was utilized before by many researchers in constructing Boolean operators. Amos and Dunne (1997) 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 and Ray (1998), proposed a bounded fan-in Boolean circuit functioning in O(1)-time complexity. Ogihara and Ray (1999) also proposed the building of DNA-based Boolean circuits for a semi-unbounded fan-in Boolean circuit. Subsequently, Erk (1999), developed an abstract DNA model for simulating Boolean circuits by finite splicing systems. The main drawback of this model is that the rules need to be altered with the complexity of the Boolean circuit. Mulawka et al. (1999) proposed another simulation of the NAND gate using the Fok I enzymes of nuclease class II. Ahrabian and Nowzari-Dalini (2004) and Ahrabian et al. (2005) 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. Liu et al. (2005) presented a theoretical model of the NAND gate through the induced hairpin formation. Jianzhong et al. (2006) suggested reusable logic gates for AND and OR functions using MB.

The major demerit of the previous models is that they employ many different types of bio-operations like annealing, ligation, polymerase chain reaction, gel electrophoresis and cleavage. Amos (2005) in his book has stated that it is difficult to ensure 100% ligation and hence strands that should have been restricted escape into the next stage and thereby resulting in errors in computation. Further, the above papers were proposed for simulating few gates only and they do not maintain the uniformity in representing logical 0 and 1, either. The previous authors had also tried simulating stratified Boolean circuits which required the gates to be of the same type at each stage (i.e. either AND or OR gate). Furthermore, in earlier implementations, after obtaining the outputs, the strands employed cannot be reused and they have to be necessarily reconstructed for subsequent use. It should also be noted that a unique generalized algorithm for designing any combinational logic operators has not been formulated in the above articles. In this context, the authors of this work (Zoraida et al., 2008) had earlier proposed a generalized algorithm for constructing logic gates. The proposed realization of a logic operator is attempted with only a

* Corresponding author. Tel.: +91 9443647737; fax: +91 431 2500657.
E-mail addresses: b.s.e.zoraida@gmail.com (B.S.E. Zoraida), michael@nitt.edu (M. Arock), romasa68@yahoo.com (B.S.M. Ronald), rpalagu@nitt.edu (R. Ponalagusamy).

0303-2647/$ – see front matter © 2009 Elsevier Ireland Ltd. All rights reserved.
doi:10.1016/j.biosystems.2009.05.010

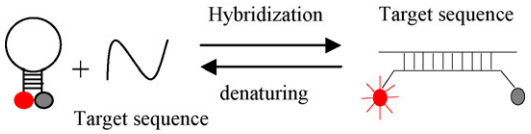

Fig. 1. Hybridization of the MB with the target.

single bio-operation, namely hybridization at each stage. However, the algorithm does not address the realization of combinational logic operators and this paper proposes a generalized algorithm for constructing both logic operators and combinational logic operators such as adders and subtractors using DNA with MB as inputs. The proposed model has uniform representation for logical 0 and 1 throughout. MBs are employed, so that the bio-operations are reliable. MBs also have extremely high selectivity and ability to identify single base-pair mismatch (Fang et al., 1999). The DNA sequence which acts as gate strand is made immobilized onto a surface. In the proposed method, reusability is achieved by a denaturing operation followed by a wash, which leaves back the strands on the surface for subsequent operations.

2. Preliminaries

2.1. Molecular Beacon

Molecular beacons 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. This is known as fluorescence resonance energy transfer (FRET). 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. Since 4-(dimethylaminoazo) benzene-4-carboxylic acid (DABCYL) can serve as a universal quencher for many fluorophores, a MB is generally synthesized using DABCYL-controlled pore glass (CPG) as the starting material. Different fluorescent dye molecules can be covalently linked to the 5′-end to report fluorescence at different wavelengths (Tyagi and Kramer, 1996). The hybridization of the MB with the target is shown in Fig. 1.

2.2. Blocker

The hybridization of a MB to the target strand is necessary for implementing combinational binary operators. However, in certain location of the target sequence, hybridization should be prevented and hence, a DNA sequence is needed as a blocker. In this paper, the assigned blocker sequence is "GGGGGG". The sequence assigned for each input strand should be different from the blocker sequence. The blocker is denoted by a circle and an "X" through it in the subsequent figures.

3. Proposed Algorithm for Realization of Boolean Operators

It is well known that each binary Boolean variable I can take two values $I = 0$ and $I = 1$. In this work they are denoted as I^0 and I^1. A

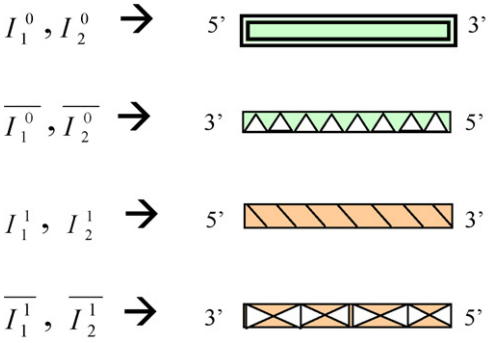

Fig. 2. Colored pattern representation of the strand.

two-input logic gate has their inputs denoted as I_1 and I_2. A unique DNA strand is taken to represent I_1^0 and I_2^0 (logic 0) and another unique strand is chosen to represent I_1^1 and I_2^1 (logic 1). The respective complements of the chosen strands are represented as $\overline{I_1^0}$, $\overline{I_2^0}$, $\overline{I_1^1}$ and $\overline{I_2^1}$. Thus, only four DNA strands are required for a two-input gate. A multiple pattern representation instead of representing the actual strands for $I_1^0, I_2^0, I_1^1, I_2^1, \overline{I_1^0}, \overline{I_2^0}, \overline{I_1^1}$ and $\overline{I_2^1}$ is followed in this paper for visualizing the realization. Fig. 2 shows the patterns and logical inputs and their complements.

A general form of truth table of a two-input gate is shown in Table 1, where I_1 and I_2 are the inputs and R_1 to R_4 are the outputs of the gate.

In the proposed design algorithm, the rows having their output $R_i = 1$ ($i = 1$–4) are considered initially. Among these rows, the first row having $R_i = 1$ is taken and its I_1 and I_2 values are stored in an array. Subsequently, the next row with $R_i = 1$ is considered and its I_1 value is compared with the last value stored in the array. If I_1 value is the same as the last value in the array, I_2 of the row presently considered is added to the array. Otherwise, "*" followed by both the values I_1 and I_2 of the present row is added to the array. The above process is repeated for the remaining rows having their R_i value as 1. After scanning all the rows with $R_i = 1$ in the truth table, the elements in the array are replaced by their corresponding complementary strands which were assigned at the start of the process. The symbol "*" is replaced by the blocker sequence. Consequently, the desired strand which performs the operation of the required Boolean circuit is obtained.

The above process is described as a unique procedure "*Strand_construct*" for designing the required strands. "*Strand_construct*" is the procedure with the following input parameters: (a) truth table of the desired Boolean circuit with the Boolean inputs I_i, $i = 1, 2, \ldots, n$ and R_i, $i = 1$ to 2^n as the Boolean outputs, (b) the number of inputs to the Boolean operator – n, which is assigned to the variable *in_no*, (c) *start* and *finish* variables help to scan the truth table in either of the direction, i.e. top to bottom or bottom to top, by assigning values 1 or 2^n, respectively, (d) variable *step_val* is assigned with -1, if *start* takes 2^n (when scanned from bottom to top), otherwise *step_val* is given a value 1.

Table 1
Truth table for two-input gates.

I_1	I_2	R_i
0	0	R_1
0	1	R_2
1	0	R_3
1	1	R_4