Chapter 3

Study on the Development of a Low Cost Generalized Model with Reusable Readout Technique

3.1 An Overview

DNA computing models for logic gate simulation was initially visualized as the first step towards molecular computer but with time it took a small deviation from the initial dream of replacing traditional computers, instead such unconventional computing opens a new spectrum of applications such as evolved hybrid computers obtained by integrating molecular computing features to silicon technology but most importantly DNA computing finds its demanding role in nano-technology such as smart drug delivery, nano robots etc. Despite of several shortcomings DNA computing scores more to silicon technology because of its molecular scale miniaturization and bio-compatibility. Many algorithms and models have been developed in past 20 years; each of the prior models came up with features to obtain

advantages such as parallelism, generalizability, robustness, uniformity, reusability, flexibility, automation and cost effectiveness, but none of the proposed model integrated all of these features into a single model.

In this chapter, a theoretical model is proposed to address these shortcomings and to partially or fully incorporate desirable features such as reusability, generalizability, robustness, uniformity into a single model.

During the design of DNA-based logic gate certain fundamental elements are necessary to be taken care of, such as:
(a) input signal, either physical (in the form of light signal, pressure signal, etc) or chemical signals (in the form of ions, proteins or nucleic acids, etc). In this work, input signal is in the form of nucleic acid sequences;
(b) an oligonucleotide sequence which acts as logic gate on interaction with the input signals;
(c) an output signal which is obtained by transducing and monitoring the changes occurs in the state of gate strand.

3.2 Algorithm to Design Operator / Gate Strand

A Boolean variable may be in either ON state (= 1) or in OFF state (= 0). For 'n' numbers of input variables, maximum possible output states can be represented in 2^n combinations. If I_1 and I_2 are two Boolean variables, for simplicity of understanding unique notations are assigned to each variable depending on their values. $I_{1,0}$ and $I_{2,0}$ are notations assigned to $I_1 = 0$ and $I_2 = 0$ respectively, similarly $I_{1,1}$ and $I_{2,1}$ represents I_1=1 and I_2=1 respectively. Complements of the variables are denoted by $\bar{I}_{1,0}$, $\bar{I}_{1,1}$, $\bar{I}_{2,0}$ and $\bar{I}_{2,1}$. Each of the variable notation, i.e. $I_{1,0}$, $I_{2,0}$, $I_{1,1}$ and $I_{2,1}$ has been allotted unique single stranded 3'$\rightarrow$ 5'oligonucleotide sequence of length 10 bases and its complementary variables $\bar{I}_{1,0}$, $\bar{I}_{1,1}$, $\bar{I}_{2,0}$ and $\bar{I}_{2,1}$ are assigned

with complements of pre-assigned sequences with 5'→ 3'orientation as shown in **Table 3.1**. **Table 3.2** shows a general form of a truth table with I_1 and I_2 as input variables and Z_i as output. Value of Z_i varies with types of logic gates needed to be simulated. The number of output column is equal to the number of output lines of the logic gate e.g. in half adder there are two output lines, i.e. sum and carry so the corresponding truth table has two output columns.

Table 3.1: Pre-assigned assigned oligo sequences

Variable Notation	**3'→ 5'**	**Complementary Variable Notation**	**5'→ 3'**
$I_{1,1}$	ACTGCGCTAT	$\bar{I}_{1,1}$	ATAGCGCAGT
$I_{1,0}$	GCTATTTGCA	$\bar{I}_{1,0}$	TGCAAATAGC
$I_{2,1}$	CTGGATCATT	$\bar{I}_{2,1}$	AATGATCCAG
$I_{2,0}$	AGACCTATTA	$\bar{I}_{2,0}$	TAATAGGTCT

Table 3.2: Truth table with two inputs and one output

$\mathbf{I_1}$	$\mathbf{I_2}$	$\mathbf{Z}_i$
0	0	Z_1
0	1	Z_2
1	0	Z_3
1	1	Z_4

In the proposed algorithm, all the rows of the corresponding truth table must be scanned one after another and accordingly the assigned variable notations are inserted into an array by following the design strategy shown in **Equation 3.2.1**. The insertion process depends on the output value, i.e. when $Z_i = 1$; $I_{1,value}$ and $I_{2,value}$ are stored as an element of the array, but when $Z_i = 0$, complement of

the variable are inserted i.e. $\bar{\mathtt{I}}_{1,value}$, $\bar{\mathtt{I}}_{2,value}$ are inserted into the array in linear fashion one after another. Scanning and insertion of input variable into the array is continued until the end of the table is reached.

$$\texttt{Strand\_array[ ]} = \begin{cases} \mathtt{I}_{j,value}, & \text{if } \mathrm{Z}_i = 1; \\ \bar{\mathtt{I}}_{j,value}, & \text{if } \mathrm{Z}_i = 0. \end{cases} \tag{3.2.1}$$

where 'j' is the counter to count the number of input variables associated with the truth table. 'i' is the counter to count the number of rows in truth table. "strand_array" is the name of the array where variable notations are inserted as elements.

After successful execution of the algorithm, the elements in the array must be replaced by the corresponding DNA strands allotted earlier (as shown in **Table 3.1**). The desired gate strand obtains at the end with orientation 3'→ 5'is shown in **Table 3.3**. The above procedure can be expressed in terms of an algorithm named "Operator_design". The pseudocode of the proposed algorithm is depicted in Algorithm 1. This algorithm can be implemented to simulate any logic functionality, on the basis of its truth table.

Parameters to be provided as inputs to the algorithm are:
(a) truth table of the desired logic gate;
(b) '*input_max*' contains value of maximum number of variables associated with the truth table;
(c) counter 'i' used for counting the rows to be scanned in the truth table, i.e. i = 1, 2, $\cdots\cdots$, 2^{input_max}.

Algorithm 1 : Algorithm for inserting elements into the array

```
Operator_design(truth table, input_max)
{
index ← 0;
for (i = 1; i ≤ 2^input_max; i++) do
  {
  if (Z_i = = 1) then
    {
    for (j = 1; j ≤ input_max; j++) do
      {
      if (I_j = = 0) then
        {
        gate_array[++index] ← I_j,0;
      else
        gate_array[++index] ← I_j,1;
        }
      end if
      }
    end for
    }
  else
    {
    for (j = 1; j ≤ input_max; j++) do
      {
      if (I_j = = 0) then
        {
        gate_array[++index] ← Ī_j,0;
      else
        gate_array[++index] ← Ī_j,1;
        }
      end if
      }
    end for
    }
  end if
  }
end for
}
```

strand_array obtained for some two input logic gate after applying the proposed algorithm is shown in **Table 3.3**.

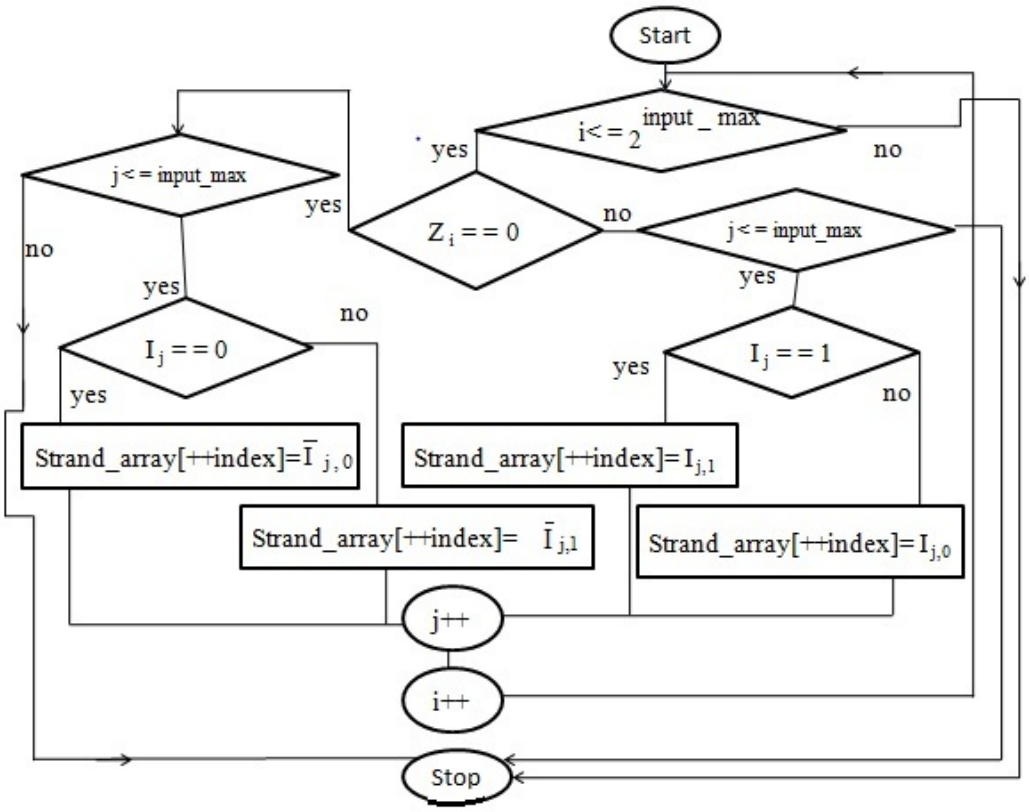

Figure 3.1: Flowchart representation of the proposed algorithm.

Table 3.3: "strand_array" of few logic gates

Gate	**Derived DNA strand (3'- 5')**
OR	$\bar{I}_{1,0}$-$\bar{I}_{2,0}$-$I_{1,0}$-$I_{2,1}$-$I_{1,1}$-$I_{2,0}$-$I_{1,1}$-$I_{2,1}$
AND	$\bar{I}_{1,0}$-$\bar{I}_{2,0}$-$\bar{I}_{1,0}$-$\bar{I}_{2,1}$-$\bar{I}_{1,1}$-$\bar{I}_{2,0}$-$I_{1,1}$-$I_{2,1}$
NOT	$I_{1,0}$-$\bar{I}_{1,1}$
NAND	$I_{1,0}$-$I_{2,0}$-$I_{1,0}$-$I_{2,1}$-$I_{1,1}$-$I_{2,0}$-$\bar{I}_{1,1}$-$\bar{I}_{2,1}$
XOR	$\bar{I}_{1,0}$-$\bar{I}_{2,0}$-$I_{1,0}$-$I_{2,1}$-$I_{1,1}$-$I_{2,0}$-$\bar{I}_{1,1}$-$\bar{I}_{2,1}$
XNOR	$I_{1,0}$-$I_{2,0}$-$I_{1,0}$-$\bar{I}_{2,1}$-$\bar{I}_{1,1}$-$\bar{I}_{2,0}$-$I_{1,1}$-$I_{2,1}$
Half-Adder	SUM: $\bar{I}_{1,0}$-$\bar{I}_{2,0}$-$I_{1,0}$-$I_{2,1}$-$I_{1,1}$-$I_{2,0}$-$I_{1,1}$-$I_{2,1}$ CARRY: $\bar{I}_{1,0}$-$\bar{I}_{2,0}$-$\bar{I}_{1,0}$-$\bar{I}_{2,1}$-$\bar{I}_{1,1}$-$\bar{I}_{2,0}$-$I_{1,1}$-$I_{2,1}$

Efficiency of the algorithm lies in its ability to mimic any Boolean function. The gate strands retain the entire information of the truth-table row-by-row therefore the corresponding truth-table can be retrieved by simply scanning the given gate sequence, hence it ensures the new feature of reverse engineering. Also the algorithm does not use any blocker sequence which saves from the extra overhead of designing the blocker sequence.

3.2.1 Input Design

For the successful execution of the algorithm, special care must be taken while providing the inputs. Digital inputs must be encoded in the form of DNA sequence without any distortion of information. Its design principle is relatively easy compared to "Operator_design" algorithm. The Input sequence is the concatenation of the complements of the input variables. The entire process in the form of a procedure named input_sequence() is shown below (Algorithm 2) **Table 3.4** shows the designed DNA inputs of any gate having I_1 and I_2 as input variables.

Algorithm 2 : Input Sequence

input_sequence()
{
for (j = 1; j ≤ input_max; j++) **do**
{
Ligate(5'- $\bar{\mathtt{I}}_{j,value}$ - 3');
}
end for
}

Table 3.4: DNA Input Strands equivalent to digital inputs

Input	Derived Input	DNA Input Strand
0,0	$\bar{I}_{1,0}$-$\bar{I}_{2,0}$	5'- CGATAAACGTTCTGGATAAT -3'
0,1	$\bar{I}_{1,0}$-$\bar{I}_{2,1}$	5'- CGATAAACGTGACCTAGTAA -3'
1,0	$\bar{I}_{1,1}$-$\bar{I}_{2,0}$	5'- TGACGCGATATCTGGATAAT -3'
1,1	$\bar{I}_{1,1}$-$\bar{I}_{2,1}$	5'- TGACGCGATAGACCTAGTAA -3'

3.3 Theoretical Simulation of Some Logic Gates

3.3.1 NAND Gate Realization

NAND gate evaluates as '1' if at least one of the inputs are "false" otherwise its output evaluates '0' i.e. NAND is the NOT of AND. **Table 3.5** shows the truth table of NAND gate. In this work, the electronic property of NAND gate is attempted to realize at the molecular level. The first step is to encode the NAND gate as a strand of DNA. **Table 3.3** shows the encoded array of NAND gate as $I_{1,0}$-$I_{2,0}$-$I_{1,0}$-$I_{2,1}$-$I_{1,1}$-$I_{2,0}$-$\bar{I}_{1,1}$-$\bar{I}_{2,1}$. On replacing the variable notations by the pre-assigned DNA sequences shown in **Table 3.1** the final DNA sequence is obtained as: 3'- GCTATTTGCA AGACCTATTA GCTATTTGCA CTGGATCATT ACT-GCGCTAT AGACCTATTA ATAGCGCAGT AATGATCCAG - 5'. The encoded inputs strands are poured to the test tube which already contain gate strands and are allowed to hybridize. **Figure 3.2** shows the four cases for simulation of NAND gate on the basis of different inputs. In every case, separate input is provided and depending on the success or failure of hybridization the output is read i.e. the success of hybridization is read as '1' otherwise as '0'. In Case 1, Case 2, Case 3 the output is '1' and in Case 4 the output is '0'. The simulation result is in agreement

with the digital NAND gate. Wet lab experimental validation for NAND gate is demonstrated in **Section 3.4**.

Table 3.5: Truth table of NAND gate

$\mathbf{I}_1$	$\mathbf{I}_2$	$\mathbf{Z}_i$
0	0	1
0	1	1
1	0	1
1	1	0

Figure 3.2: Theoretical simulation of DNA-NAND gate.

3.3.2 Half-adder Realization

In a half-adder two single bits adds to produce two bits as output i.e. Sum and Carry bit. **Table 3.6** represents the truth table of a half-adder. In digital electronics half-adder can be realized using one XOR gate and one AND gate as shown in

Figure 3.3. Similarly in case of molecular realization of half-adder, Sum strand obtained after using the proposed algorithm is same as DNA-XOR gate and Carry strand is similar to DNA-AND gate. By executing biomolecular operations on the Sum and Carry strands, half adder can be realized.

Table 3.6: Truth table of half adder

$\mathbf{I_1}$	$\mathbf{I_2}$	**SUM**	**CARRY**
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

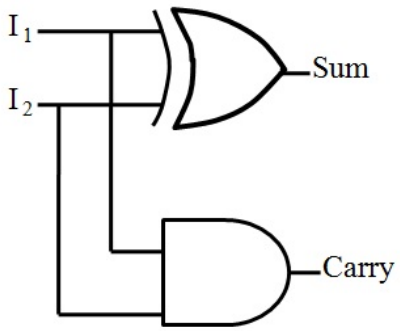

Figure 3.3: Logic diagrams of half adder.

Sum strand: $I_{1,0}$-$I_{2,0}$-$I_{1,0}$-$I_{2,1}$-$I_{1,1}$-$I_{2,0}$-$\bar{I}_{1,1}$-$\bar{I}_{2,1}$

After replacing the variable notations by assigned DNA sequences from **Table 3.1**, the final DNA sum sequence is obtained as follows:

3'- TGCAAATAGC TAATAGGTCTG CTATTTGCAC TGGATCATTA CT-GCGCTATA GAC CTATTA ATAGCGCAGT AATGATCCAG - 5'

Figure 3.4 shows the theoretical simulation of Sum sequence. When the inputs are provided in the form of DNA sequences equivalent to (0,0), (0,1), (1,0) and

(1,1) as shown in **Table 3.4**, the success or failure of DNA hybridization reaction is used as a measure to detect output. All the four cases are shown as Case 1: (0,0), Case 2: (0,1), Case 3: (1,0) and Case 4: (1,1). On exposing the gate strands to the inputs, successful hybridization reaction occurs in Case 2 and Case 3 whereas in Case 1 and Case 4 the input and Sum sequence failed to hybridize.

Figure 3.4: Theoretical simulation of Sum sequence.

As the success of hybridization is read as '1', Case 2 and Case 3 evaluated as '1' and Case 1 and Case 4 evaluated as '0'. Similar technique is followed to simulate Carry strand as well. Once the intermediate variable notations are obtained from the array named gate_arrary[ ], the notations are replaced by the pre-assigned DNA sequence from **Table 3.1**.

Carry strand: $\bar{I}_{1,0}$-$\bar{I}_{2,0}$-$\bar{I}_{1,0}$-$\bar{I}_{2,1}$-$\bar{I}_{1,1}$-$\bar{I}_{2,0}$-$I_{1,1}$-$I_{2,1}$

The final DNA carry sequence obtained by replacing the elements of array by the pre-assigned DNA strands is as shown below:

3'- TGCAAATAGC TAATAGGTCT TGCAAATAGC AATGATCCAG ATAGCGCAGT

TAATAGGTCT ACTGCGCTAT CTGGATCATT - 5'

Figure 3.5 shows the theoretical simulation of Carry strand on providing inputs in the form of DNA sequences equivalent to (0,0), (0,1), (1,0) and (1,1) (from **Table 3.4**). All this four cases are demonstrated as Case 1: (0,0), Case 2: (0,1), Case 3: (1,0) and Case 4: (1,1).

Figure 3.5: Theoretical simulation of Carry sequence.

Only in Case 4 the hybridization reaction is successful which is read as '1' whereas in all other cases i.e. Case 1, Case 2 and Case 3, the inputs failed to hybridize, which evaluates as output '0'.

3.4 Experimental Verification of the Algorithm

To check the proposed algorithm experimentally a simple NAND gate is tested in laboratory.

3.4.1 Materials and Chemicals / Reagents

Deoxyribose nucleic acid (DNA): four sets of encoded strands of DNA purchased from MolBiogen. SM193 ladder has been purchased from ThermoFisher Scientific. Other chemicals like TRIS, EDTA, HCl, NaCl, Bromophenol Blue, Agarose powder, and TAE buffer are obtained from local vendor Sigma-Aldrich chemicals.

3.4.2 Preparation of Solutions and Experiment:

Samples of lyophilized DNA is dissolved using TRIS-EDTA buffer (pH 8.0). The hybridization buffer (pH 8.0) has been prepared using NaCl, TRIS, EDTA and HCl. 1.46 gm of NaCl is mixed with 6 gm TRIS, 1mM of EDTA and 2.1 ml of HCl. The gate strand of DNA are mixed with the four input strands in different test tubes and incubated for 12 Hours at 42^oC so that they can hybridize. To check for success of hybridization the incubated DNA samples are run through gel. 5% agarose gel has been prepared using 2 gm agarose powder and 40 ml TAE buffer. Ethidium Bromide is added to help in florescence visualization of bands after performing gel electrophoresis. The four samples are run through the gel for 45 minutes and the bands can be visualized.

Figure 3.6 represents the four cases i.e. Case 1: (0,0), Case 2: (0,1), Case 3: (1,0) and Case 4: (1,1) of NAND gate. Case 1, Case 2, Case 3 shows success of hybridization whereas Case 4 depicts failure of hybridization which is in agreement with digital NAND gate. 3'- GCTATTTGCA AGACCTATTA GCTATTTGCA CTGGATCATT ACTGCGCTAT AGACCTATTA ATAGCGCAGT AATGATCCAG -5'strand acts as template strand and 5'-CGATAAACGTTCTGGATAAT-3', 5'-CGATAAACGTGACCTAGTAA-3',5'-TGACGCGATATCTGGATAAT-3'and 5'-TGACGCGATAGACCTAGTAA-3'act as probe strands.

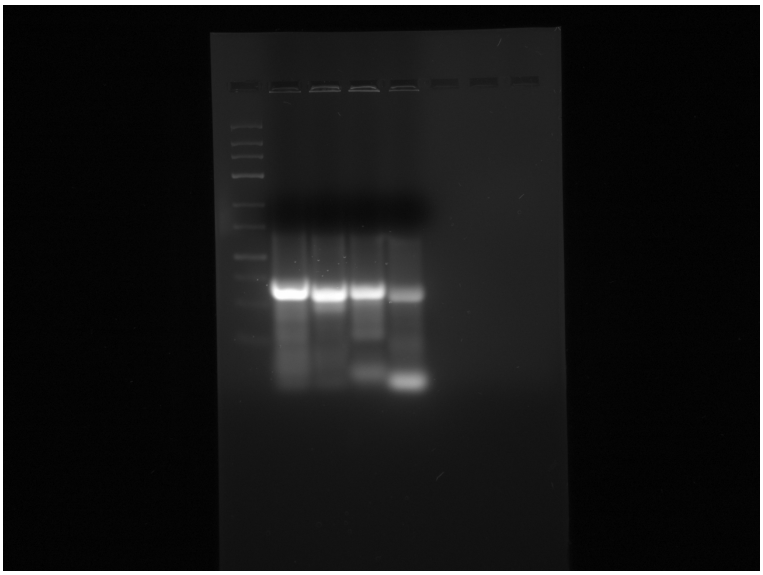

Figure 3.6: Experimental simulation of DNA NAND gate: Case 1, Case 2, Case 3 and Case 4 are visible as four bands from left hand side respectively.

3.5 Cantilever Deflection Mechanism used as Read-Out Technique

MB is one of the most common and efficient readout mechanism employed in several prior models however in this chapter the use of micro cantilever deflection as a label-free read-out technique has been proposed to use for the first time in the field of simulation of DNA logic gates due to their inherent ability to generate high sensitive and quantitative measurements with low cost, portability, real time and label-free detection. In the following sections the proposed model is demonstrated with the use of MBs and cantilever as read-out technique.

Micro cantilever-based sensors emerges as new label-free detection technique since Thundat et al. [87] proposed the possibility to use cantilever as nano sensors alternative to conventional biochemical sensors because of its high sensitivity, fast response and parallelization. Typically, microcantilever are fabricated on silicon wafer or silicon-on-insulator (SOI) having dimension ranging from ten to hundred

micrometers in length, 20 microns wide and 1 micron thick. SOI wafer is made of three layers; a top layer of single-crystal silicon or silicon nitrite, middle layer consists of silicon oxide layers and the bottom layer consists of single-crystal silicon. The upper surface is chemically modified that has a high affinity towards the receptor molecules and the lower surface is made inert so that all the reaction is restricted to one surface. The mechanical deformation is transduced to measurable quantitative signal. The designed single stranded DNA gate strands (ssDNA) are covalently immobilized on the upper surface of the cantilever beam. When input strands are poured on the beam, hybridization occur resulting in change in surface stress between the upper and lower surface and hence bending the cantilever.

Hansen et al. [88, 89] experimentally showed that the number and position of mismatch pairs affect the deflection of the cantilever and hence could be used to detect mismatches. In Experiment conducted by Stachowiak et al. [90] it is indicated that the effects of the immobilization density, chain length and hybridization efficiencies are coupled, and the cantilever deflection may primarily depend on factors like length and sequence, salt concentration, temperature, grafting density, hybridization density, moisture concentration and time. Fritz et al. [91] hypothesized that the cantilever deflection is the result of electrostatic repulsion arising due to the negative charges on the DNA strands and relaxation of steric hindrance. Yue Zhao et al. [92] proposed a model for predicting hybridization induced bending of micro-cantilever based on the minimization of the energy functional that accounts for cantilever bending energy and DNA interchain interactions. N.H. Zhang et al. [93] concluded from experiments and theories related to interaction of single stranded DNA immobilized on micro-cantilevers that at high grafting density cantilever deflection is actually controlled by hydration force and not by electrostatic and conformational entropy.

The theoretical simulation of NAND gate with the cantilever deflection is shown in **Figure 3.7** to **Figure 3.10**. The success of hybridization can be easily detected

by the bending and hence read as output '1' otherwise as '0'. In Case 1, Case 2, Case 3 the output is "true" whereas in Case 4 the output is "false".

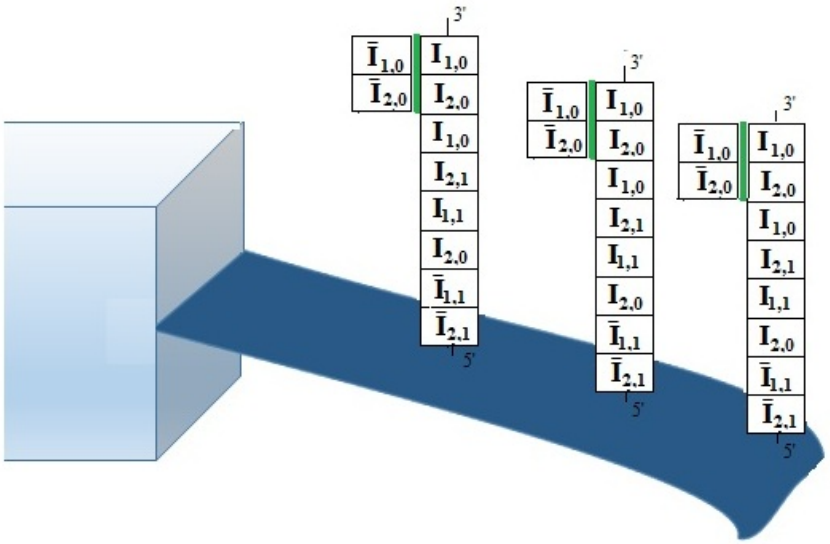

Figure 3.7: Case 1: Input (0, 0).

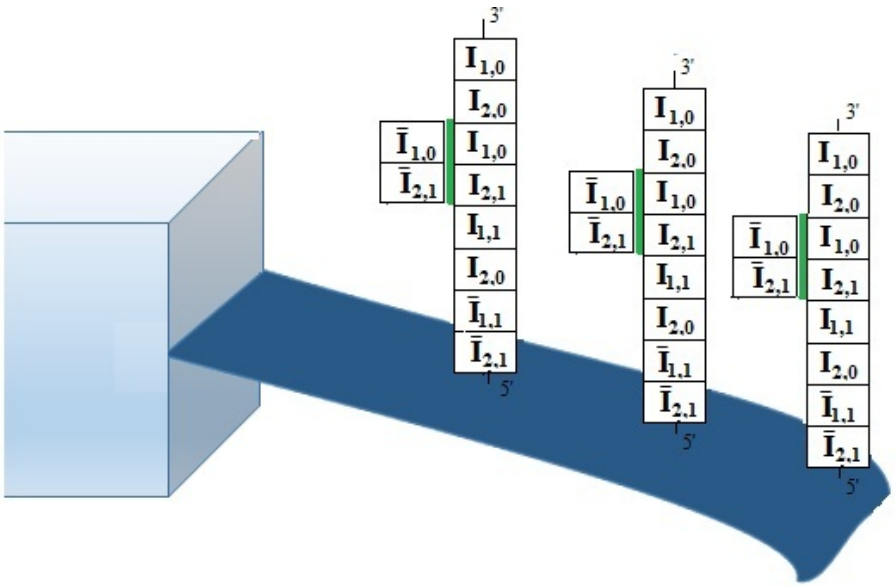

Figure 3.8: Case 2: Input (0, 1).

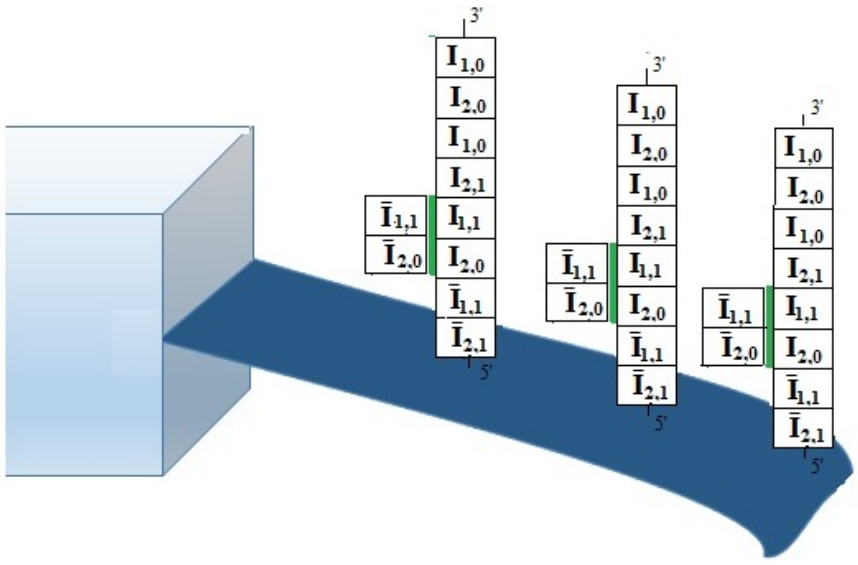

Figure 3.9: Case 3: Input (1, 0).

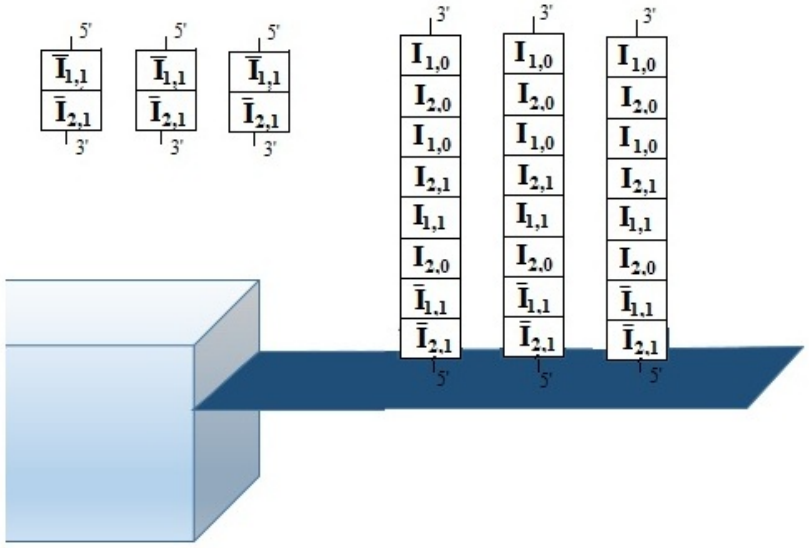

Figure 3.10: Case 4: Input (1, 1).

3.6 Universality of DNA-NAND gate

In digital electronics NAND and NOR gates are universal gate as any other Boolean functions can be derived from either of them alone. All the properties held by dig-

ital NAND or NOR gate should also be valid for DNA-NAND or NOR gate therefore in **section 3.6.1** and **section 3.6.2** the theoretic validation of universality of DNA-NAND gate is demonstrated.

3.6.1 Realization of AND gate with DNA-NAND gate:

AND gate can be derived from NOT of NAND gate. With the help of digital NAND gate, AND gate can be derived as shown in **Figure 3.11**. There are two NAND gates integrated in such a fashion that the output of first gate acts as input to the second gate. The property of AND gate can be reproduced at molecular level with the help of DNA-NAND gate as shown in **Figure 3.12** to **Figure 3.16**. During the theoretical simulation process, initially the input and the gate strands of gate 1 are allowed to undergo hybridization and the output obtained acts as inputs to the reaction with gate strands of gate 2. Depending on four input combinations, the four cases are theoretically shown as Case 1, Case 2, Case 3 and Case 4. However there is substantial human intervention between the gates. The output of each gate is read and the input to the next gate is provided in a non-automated manner.

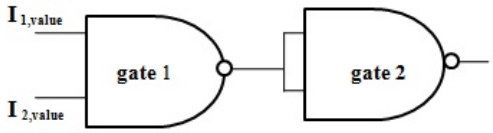

Figure 3.11: AND gate from NAND.

Case 1: When inputs are (0, 0) i.e. $I_1 = 0$ and $I_2 = 0$, the equivalent DNA input sequence obtained is 5'- CGATAAACGTTCTGGATAAT-3'(shown in **Table 3.4**).

Simulation at gate 1:

The input strand 5'- CGATAAACGTTCTGGATAAT-3'and the gate strand 3'- GCTATTTGCA AGACCTATTAGCTATTTGCACTGGATCATTACTGCGCTATA-GACCTATTAATAGCGCAGTAATGATCCAG -5'are partially complementary to

each other hence they successfully hybridizes as shown in **Figure 3.12**. The output of gate 1 is read as "True" or '1'.

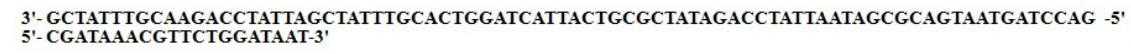

Figure 3.12: Reaction at Gate 1.

Simulation at Gate 2:

Since the output of gate 1 is '1' therefore the input to gate 2 is also '1'. The DNA equivalent of (1, 1) is 5'-TGACGCGATAGACCTAGTAA-3'(From **Table 3.4**). In presence of input and gate strand the reaction at gate 2 is demonstrated in **Figure 3.13**.

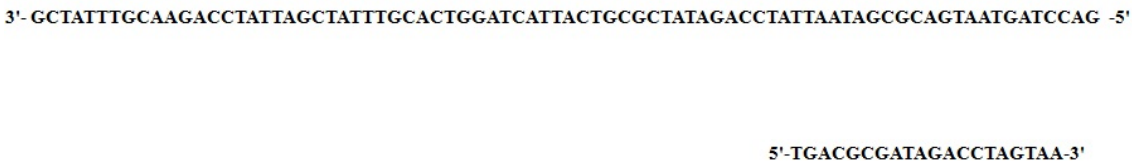

Figure 3.13: Reaction at Gate 2.

In gate 2, no segment of gate strand and input strand are complementary hence no hybridization occurs. Therefore, it can be stated that for (0,0) input, final output is '0' which is in agreement with digital AND gate.

Similarly, theoretical validation is shown for all other digital input cases i.e. Case 2: (0,1), Case 3: (1,0) and Case 4: (1,1). **Table 3.7** shows the result of molecular simulation process and of digital AND gate.

Case 2: The input (0,1) is provided as a DNA strand of 20 nucleotides length i.e. 5'-CGATAAACGTGACCTAGTAA-3'. During gate 1 simulation, the input finds its counterpart in the gate strand and hence hybridizes providing the output as '1'. Similarly the gate 2 simulation is also examined. Here in this case the output of gate 1 acts as input to gate 2. Therefore input (1,1) i.e. 5'-s-TAATAGGTCTATAGCGCAGT-s̄-3'is provided to gate 2. The simulation of gate

1 and gate 2 are shown in **Figure 3.14**.

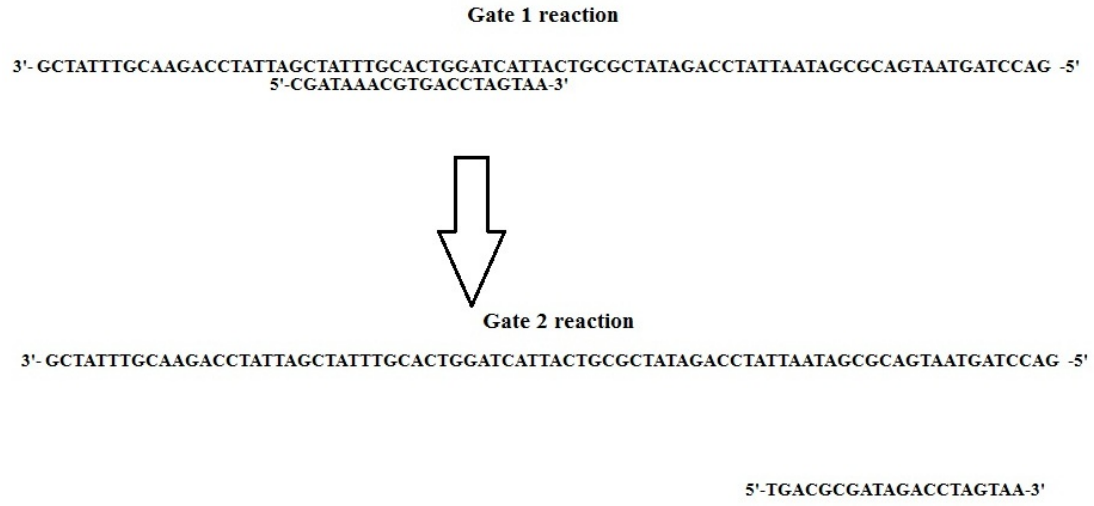

Figure 3.14: DNA-NAND gate as AND gate for input (0,1).

Case 3: The input for third case is (1, 0) i.e. 5'-TGACGCGATATCTGGATAAT-3'. The simulation process is illustrated in **Figure 3.15**.

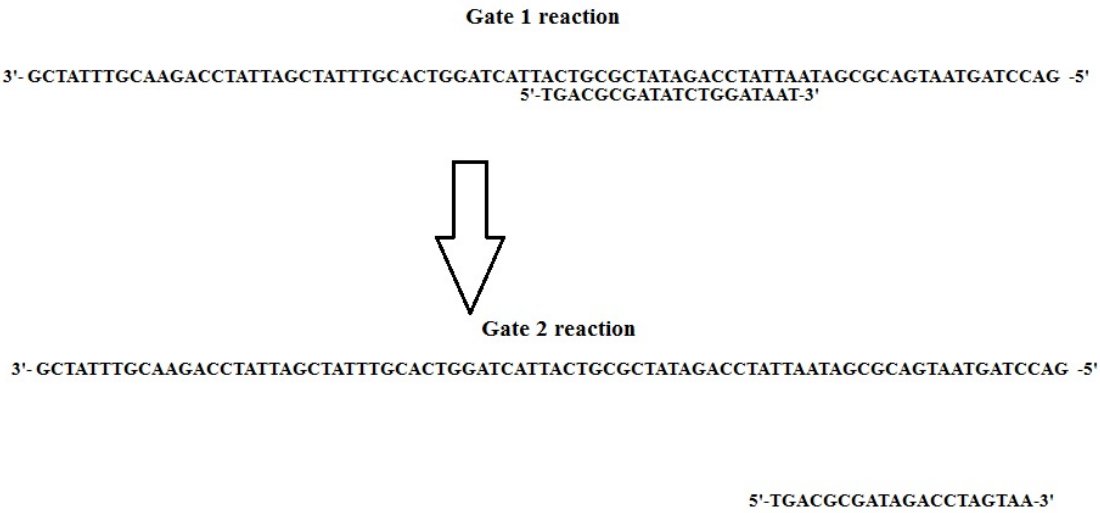

Figure 3.15: DNA-NAND gate as AND gate for input (1,0).

Case 4: The input for Case 4 is (1,1) i.e. 5'- TGACGCGATAGACCTAGTAA-3'. **Figure 3.16** demostrates the simulation for input (1,1).

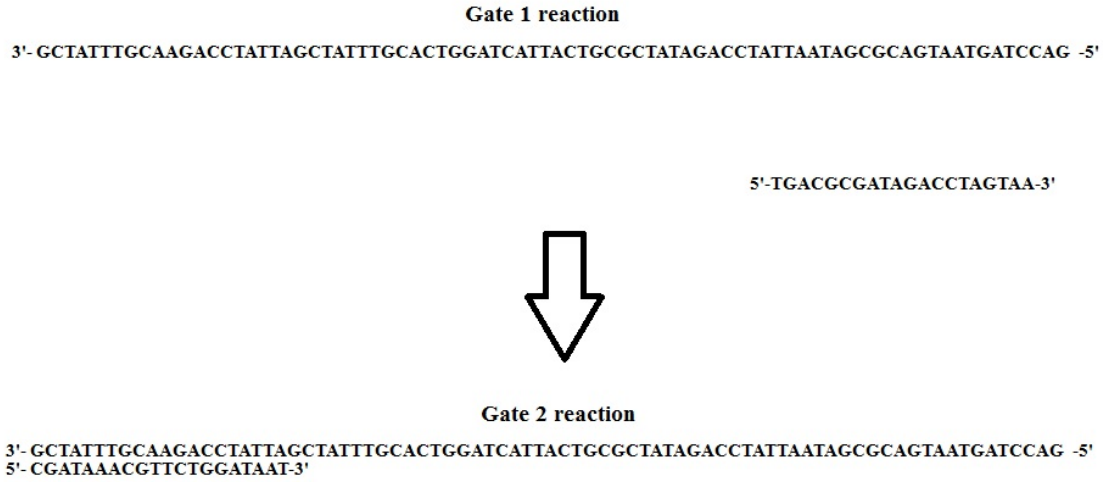

Figure 3.16: DNA-NAND gate as AND gate for input (1,1).

The entire simulation results are summarized in the form of a table shown as **Table 3.7**.

Table 3.7: Digital and simulated output (AND)

I_1	I_2	**Simulated gate 2 output**	**Digital I_1 AND I_2**
0	0	Not Hybridized	0
0	1	Not Hybridized	0
1	0	Not Hybridized	0
1	1	Hybridized	1

3.6.2 Realization of OR gate with DNA-NAND gate:

In digital electronics NAND gate can be used to obtain the functionality of OR gates (shown in **Figure 3.17**). A simulation process is demonstrated at molecular level to depict the derivability of OR gate functionality with the help of DNA-NAND gate (shown in **Figure 3.18** to **Figure 3.21**), simulated on the basis of

different inputs cases are shown in Case 1, Case 2, Case 3 and Case 4 respectively. The output from gate 1 and gate 2 act as input to gate 3. **Table 3.8** shows the simulation and digital output of OR gate.

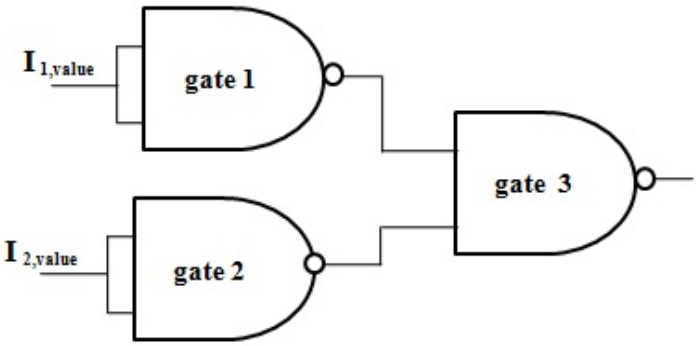

Figure 3.17: OR gate obtained from NAND gate.

Case 1: Simulation for input case (0, 0) is demonstrated in **Figure 3.18**

Figure 3.18: DNA-NAND gate as OR gate for input (0,0).

Case 2: Simulation for input case (0, 1) is demonstrated in **Figure 3.19**

Figure 3.19: DNA-NAND gate as OR gate for input (0,1).

Case 3: Simulation for input case (1, 0) is demonstrated in **Figure 3.20**

Figure 3.20: DNA-NAND gate as OR gate for input (1,0).

Case 4: Simulation for input case (1, 1) is demonstrated in **Figure 3.21**

Figure 3.21: DNA-NAND gate as OR gate for input (1,1).

Table 3.8: Digital and simulation model output for OR gate

I_1	I_2	**Simulated Output (gate 3 output)**	**Digital Output (I_1 OR I_2)**
0	0	Not Hybridized	0
0	1	Hybridized	1
1	0	Hybridized	1
1	1	Hybridized	1

3.6.3 NAND gate as NOT gate

Whenever the two inputs of NAND gate are joined together to a same input value, then it acts as NOT gate (shown in **Figure 3.22**). In molecular level this theoretical validation is shown in **Figure 3.23** and **Figure 3.24**). **Table 3.9** shows the simulation output of DNA-NOT gate. There are two input cases are shown as Case 1 and Case 2.

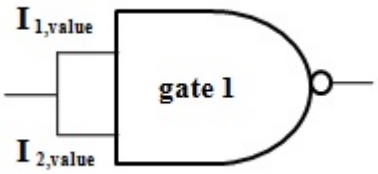

Figure 3.22: NAND gate as NOT gate.

Case 1:

In Case 1, both the inputs are '0' or "false" therefore the DNA equivalent input strand is 5'- CGATAAACGTTCTGGATAAT-3'(**Table 3.4**). Since the gate strand and the input strand are complementary, they successfully hybridizes and results in fluorescence signal emission.

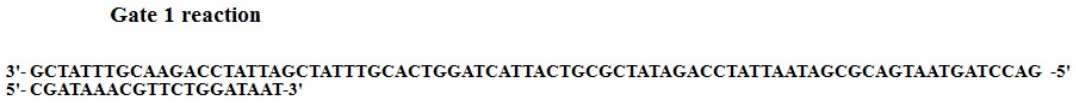

Figure 3.23: DNA-NAND gate as NOT gate for input 0.

Case 2:

Similarly, Case 2 shows the simulation process for input '1'. Since the gate strand and the input strand fails to hybridize the output is read as '0'.

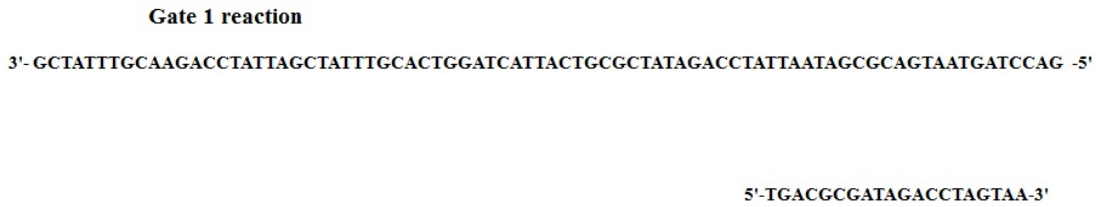

Figure 3.24: DNA-NAND gate as NOT gate for input 1.

Table 3.9: Digital and simulation model output of NOT gate

INPUT	**Gate 2 Output**	**NOT**
0	Hybridized	1
1	Not Hybridized	0

3.7 Summary

An algorithm has been proposed and verified for simulation of any logic function. The simulation results of the molecular level logic gates are demonstrated theoretically. Further the functional completeness of DNA-NAND gate is verified and tested by realizing AND, OR and NOT gates from DNA-NAND gate. The model is easy to implement in laboratory as it employs minimum biochemical reactions primarily based on DNA hybridization property. This logic gate model can be used for many biomedical applications.