Chapter 4

Induced Hairpin Based Labeled DNA Model for Evaluating Logic Gate and Boolean Circuit

4.1 An Overview

Fluorescence labeling techniques for detection of hybridization are effective because of their sensitivity, quick response time and easy handling. In several prior models such as Zoraida et al. [10] all the input strands i.e. four sets of DNA input sequences were labeled with fluorescence tags so that they acts as Molecular Beacons (MB) which increase the entire setup cost. Also in models like W. Liu et al. [8,9] generalizability feature was not included, instead demonstrated specifically for NAND and XOR gate only. In this chapter an algorithmic approach is introduced which exploits the induced hairpin formation property of a single stranded DNA with G.G mismatch in presence of chemicals like Naphthyridine Dimer (ND). During this method the gate strands itself act as sensor and works effectively even

in case of partial hybridization with the input strands. Moreover, the set up cost is reduced, as the fluorescence labeling is done only in one set of DNA i.e. in gate strand, while the labeling of the input strands are not necessary.

In this chapter an improved reusable cost effective DNA computing algorithm to evaluate any logic gate and Boolean circuit has been proposed. This model harnesses the hairpin formation property of DNA in presence of Naphthyridine Dimer, combined with a new gate design algorithm. The advantage of this model lies in its capacity to evaluate Boolean circuit consisting of different logic gates. Also, it ensures a high degree of parallelism, as several gates in the circuit can be simulated by designing only a few DNA gate strands. The number of gate strands required to simulate the circuit does not depend on the number of electronic gates in the circuit but depends on the number of final outputs of the circuit to be simulated. In case of Boolean circuit design process the gate strands are designed using the final truth table of the entire circuit instead of considering each of the individual logic gate separately.

Working Principle of the Model:

DNA is well known for its high specificity in its complementary base pairing i.e. A to T and G to C. Watson-Crick base pairing ensures ideal double helical structure of DNA. However mismatch occurs when non complementary bases are aligned in same base pair position of the duplex, resulting in compromised structure and stability. The distortion of G.G mismatch is very sever compare to other mismatches. Considerable conformational heterogeneity can be observed at G.G mismatched nucleotide and their nearest neighbors. The non-complementary bases are weakly hydrogen bonded and are only partially stacked into the helix resulting in local alteration in groove dimension. The chemical structure of G.G mismatch is shown in **Figure 4.1**.

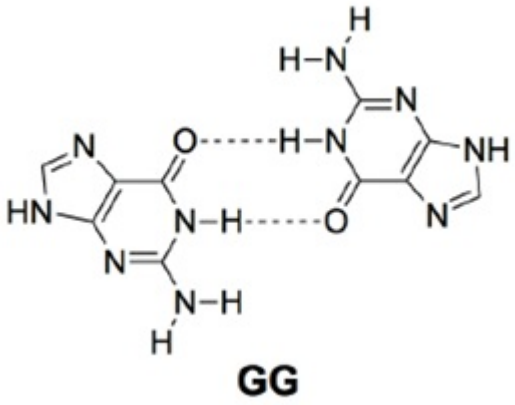

Figure 4.1: Chemical structure of mismatched G.G base pair.

The working principle of the model relies on simple hybridization property of two complementary strands of DNA and the action of Naphthyridine Dimer (dimeric form of 2-Amino-1,8-Naphthyridine) on G.G mismatched nucleotides. DNA molecule has a specific binding tendency to its complementary nucleotides such as: A always binds to T and G to C. But in presence of Naphthyridine Dimer, G.G pair starts acting as artificial complementary nucleotides as ND selectively binds to G-G with high affinity. **Figure 4.2** shows the chemical structure of ND.

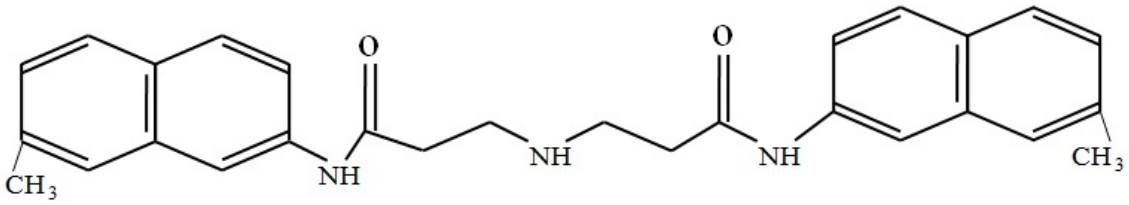

Figure 4.2: Structure of Naphthyridine Dimer (ND).

The dimer of Naphthyridine form six hydrogen bonds with the two mismatch Guanine molecules. The imino and amino protons of Guanine molecules establish four hydrogen bonds with the nitrogen atoms of Naphthyridine ring and the other two of the hydrogen bonds are formed between the amide NH atoms of ND and carboxyl oxygen atom of Guanine (depicted in **Figure 4.3**).

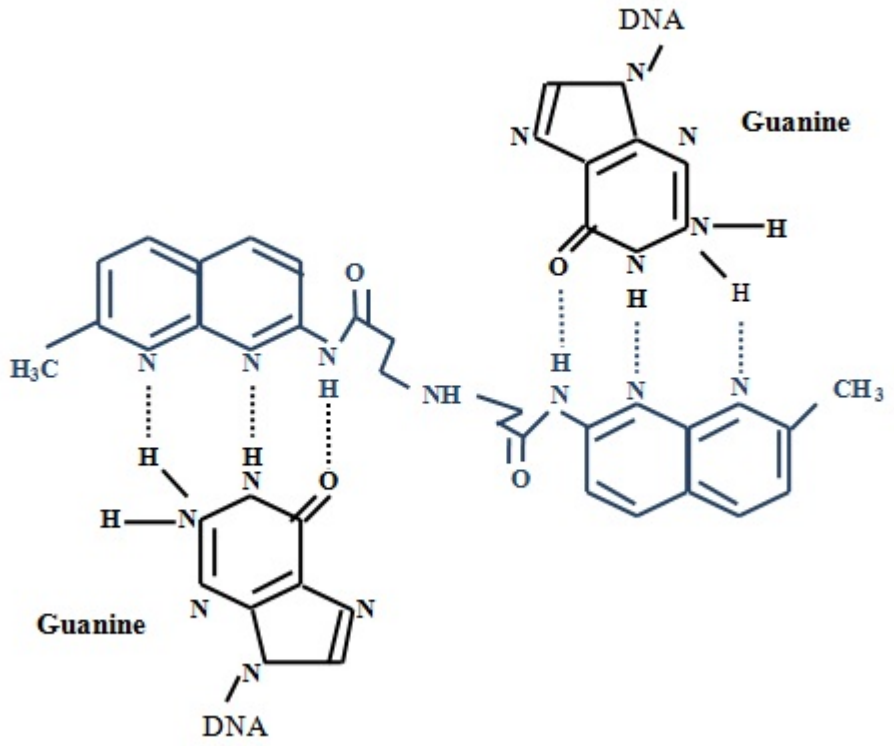

Figure 4.3: Hydrogen bonds between two G-G molecule induced by Naphthyridine Dimer.

Smith et al. [62] reported in 2002 that in the presence of Naphthyridine Dimer, free energy of hybridization is enhanced which leads to the increase in its melting temperature, as a result G.G behaves as artificial complementary pair that could be used to chemically induce the formation of hairpin structure. This artificial G.G pairing is easy to handle as the Napthyridine Dimer can be removed by simply washing off from the surface. As soon as the dimer is removed the hydrogen bonds between Guanine and dimer is broken and G.G complementarity is no longer retained. Again on adding dimer, the G.G bonds can be easily restored. Using this chemical property, hairpin structure can be induced in a strand having subsequences with G.G mismatched bases in its both ends. In normal circumstance the self complementary segment with G.G mismatch in the strand does not establish stable bonds and hence prefer to remain linear whereas on adding Naphthyridine

Dimer, the G.G acts as complementary pairs which establishes stable hydrogen bonds to obtain hairpin like structure. On adding the target complementary sequence to the loop sequence, the hairpin opens up even in presence of ND as the length of loop sequence is longer than the stem, therefore the rigidity and the length of loop hybrid is strong enough to preclude the stem hybrid. Due to the potentiality of inducting hairpin structure in gate strand, it is used as sensors for detection of molecular output by attaching a fluorophore at 3'end and a quencher at 5'end of stem sequence. A random spacer sequence is also ligated to the 5'end of gate strand to protect the gate strand from information loss while immobilizing onto the solid surface. Whenever the gate strand remains in its hairpin structure, the fluorophore and the quencher exists in close proximity and hence the energy of fluorophore is quenched which results in "dark" signal. The gate strand transforms from "dark" to fluorescence "bright" on finding the target complementary sequence to the loop sequence as the hairpin opens up which results in separation of fluorophore and quencher (as shown in **Figure 4.4**).

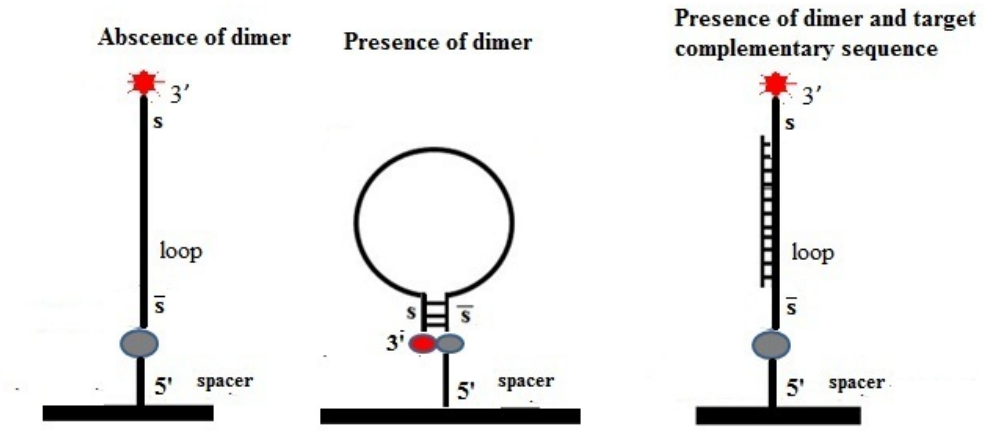

Figure 4.4: The immobilized gate strand is shown without dimer, with dimer and with target complementary sequence.

The functionality of gate simulation is based on the success or failure of hybridization reaction which results in emission of fluorescence signal. During the gate simulation the output is read as '1' if the loop opens up (fluorescence emission)

otherwise as '0' (no emission). After every cycle of operation, the gate strands can be reused by washing off the dimer and retaining its prior linear form.

4.2 Gate Design Strategy

The simulation process mainly involves four phases; in the first phase, the designing and immobilization of logic gates is done, in the second phase the input strands are encoded and in the third phase the inputs are poured into the test tube along with gate strands so that they can hybridize and finally in fourth phase the output is read-out in terms of success or failure of hairpin formation. After each cycle, the immobilized gate strands can be reused by washing off the dimer and retaining the linear structure of the gate strands.

4.2.1 PHASE I: Encoding of Logic gate

The gate strand consists of two parts: stem part with G.G mismatch and loop part. While designing the loop part of the gate strand certain design strategy is followed which is represented in the form of an algorithm (Algorithm 3) named Loop_design( ). In the proposed algorithm the truth table of the desired logic function is fetched as input parameter. The rows of the corresponding truth table are scanned one after another and only those rows are considered having output value as '1' while the rows with output value as '0' are ignored. The scanning is continued until the end of the table is reached. For all rows where $Z_i = 1$; $I_{j,value}$ (where j = 1 to 2 in case of two input truth table) are ligated to each other in linear fashion. '*value*' represents the associated boolean value of the input variable for that particular row i.e. $value = 1$ or $value = 0$. **Table 4.1** shows a general form of a truth table with I_1 and I_2 as input variables and Z_i as output.

Table 4.1: General form of Truth-table with two inputs

$\mathbf{I}_1$	$\mathbf{I}_2$	$\mathbf{Z}_i$
0	0	Z_1
0	1	Z_2
1	0	Z_3
1	1	Z_4

Algorithm 3 : Loop Part of the gate strand

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

'*input_no*' represents the number of input variables in the given truth table. For a truth table with 'n' number of input variables the maximum numbers of rows possible are 2^n. The final output of the algorithm is obtained in the form of an array named "gate_array". **Table 4.2** represents some of the loop sequences obtained

from the above design strategy. Each of the variable notations i.e. $I_{j,value}$ (j = 1 to input_no) is pre-assigned with a unique nucleotide sequence. Once the entire gate sequence is designed, the variable notation are replaced by the pre-assigned DNA sequences.

Table 4.2: Loop Sequence of Gate Strand

	Loop Sequence of Gate strand
OR	$\{I_{1,0}$-$I_{2,1}$-$I_{1,1}$-$I_{2,0}$-$I_{1,1}$-$I_{2,1}\}$
AND	$\{I_{1,1}$-$I_{2,1}\}$
NAND	$\{I_{1,0}$-$I_{2,0}$-$I_{1,0}$-$I_{2,1}$-$I_{1,1}$-$I_{2,0}\}$
XOR	$\{I_{1,0}$-$I_{2,1}$-$I_{1,1}$-$I_{2,0}\}$
XNOR	$\{I_{1,0}$-$I_{2,0}$-$I_{1,1}$-$I_{2,1}\}$
Half Adder	Sum: $\{I_{1,0}$-$I_{2,1}$-$I_{1,1}$-$I_{2,0}\}$ Carry: $\{I_{1,1}$-$I_{2,1}\}$

To obtain the complete gate strand, the above designed loop sequence should be ligated to the stem sequence represented by s and $\bar{s}$ at 3'and 5'ends respectively. The s and $\bar{s}$ are complementary sequence with G.G mismatching as shown below:
s = 5'-AAAGGGTTTGGGT-3'
$\bar{s}$ = 3'-TTTGGGAAAGGGA-5'

The final gate strand obtained after ligating the stem part to the loop part is as shown in **Table 4.3**.

Table 4.3: Final Gate Strands

	Final Gate Strands (3'→ 5')
OR	s-$I_{1,0}$-$I_{2,1}$-$I_{1,1}$-$I_{2,0}$-$I_{1,1}$-$I_{2,1}$-$\bar{s}$
AND	s-$I_{1,1}$-$I_{2,1}$-$\bar{s}$
NAND	s-$I_{1,0}$-$I_{2,0}$-$I_{1,0}$-$I_{2,1}$-$I_{1,1}$-$I_{2,0}$-$\bar{s}$
XOR	s-$I_{1,0}$-$I_{2,1}$-$I_{1,1}$-$I_{2,0}$-$\bar{s}$
XNOR	s-$I_{1,0}$-$I_{2,0}$-$I_{1,1}$-$I_{2,1}$-$\bar{s}$
Half Adder	Sum Strand: 3'-s-$I_{1,0}$-$I_{2,1}$-$I_{1,1}$-$I_{2,0}$-$\bar{s}$- 5' Carry Strand: 3'-s-$I_{1,1}$-$I_{2,1}$-$\bar{s}$ - 5'

4.2.2 PHASE II: Input Design Strategy:

While simulating molecular logic gate, the sole attempt is to maintain the input-output pattern in agreement with digital logic. In this model, input strand design strategy is rather simple compared to loop design strategy. For a two input Boolean gate there are four input cases, i.e. (0,0), (0,1), (1,0) and (1,1). The input strand is obtained by ligating the complements of variables according to its values, i.e. $\bar{I}_{1,value}$ and $\bar{I}_{2,value}$ which are ligated to form input sequence for a logic gate with two variables (value = 1 or value = 0). The input design process is represented by a procedure (Algorithm 4) named Input_sequence(). **Table 4.4** represents DNA input strands for a two variable truth table.

Algorithm 4 : Input Design Process

```
Input_sequence()
{
for (j = 1; j ≤ input_no; j++) do
  {
  Ligate(Ī_{j,value});
  }
end for
}
```

Table 4.4: Input strands for a two variable gate.

Input	**5'- $\bar{I}_{1,value}$ - $\bar{I}_{2,value}$ - 3'**
0,0	$\bar{I}_{1,0}$ - $\bar{I}_{2,0}$
0,1	$\bar{I}_{1,0}$ - $\bar{I}_{2,1}$
1,0	$\bar{I}_{1,1}$ - $\bar{I}_{2,1}$
1,1	$\bar{I}_{1,1}$ - $\bar{I}_{2,1}$

4.2.3 PHASE III: Theoretical Simulation of Logic Function

NAND gate Realization

NAND gate evaluates as '1' if at least one of the inputs is "false" otherwise as '0'. In this work, it is attempted to realize the electronic property of NAND gate at the molecular level. The first step is to encode the NAND gate as a strand of DNA and immobilize it on a surface. The encoded NAND gate sequence is 3'- s - $I_{1,0}$ - $I_{2,0}$ - $I_{1,0}$ - $I_{2,1}$ - $I_{1,1}$ - $I_{2,0}$ - $\bar{s}$ - 5'. Once the gate strand is immobilized and Naphthyridine Dimer is added, the encoded input strands are provided to the immobilized strands (as shown in **Table 4.4**) and are allowed to hybridize. The gate strands which successfully undergo hybridization with the input strands retain its linear form as it can overcome the self-hybridization tendency of the s

and $\bar{s}$ sequence of the gate strand whereas the unhybridized gate strands continue to remain in hairpin structure in presence of Naphthyridine Dimer. **Figure 4.5** shows the four cases of simulation of NAND gate. In every case, separate inputs are provided and depending on the success or failure of hybridization the output is read. The appearance of linear structure results in "bright" signal and is read as '1' otherwise as '0'. The simulation result is in agreement with the digital NAND gate as in cases 1, case 2 and case 3 the output obtained is '1' and in case 4 the output is '0'. The simulation result of DNA-NAND gate is shown in **Table 4.5**.

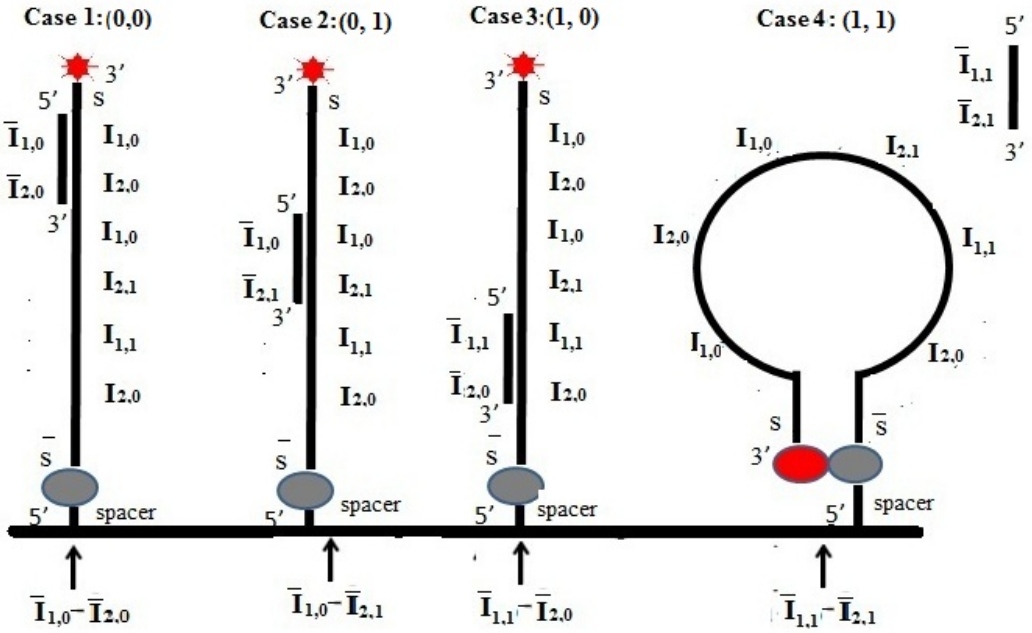

Figure 4.5: Simulation of a DNA-NAND gate. Case 1: $I_1 = 0$ and $I_2 = 0$. Case 2: $I_1 = 0$ and $I_2 = 1$. Case 3: $I_1 = 1$ and $I_2 = 0$. Case 4: $I_1 = 1$ and $I_2 = 1$.

Table 4.5: Simulation Result of DNA-NAND gate

$\mathbf{I_1}$	$\mathbf{I_2}$	$\mathbf{I_1}$ **NAND** $\mathbf{I_2}$	**Simulation Result**
0	0	1	Linear
0	1	1	Linear
1	0	1	Linear
1	1	0	Hairpin

NOR gate Realization

Simulation of NOR gate follows a similar procedure as NAND gate. In NOR gate output evaluates to '0' if at least one input is "true" otherwise as '0' (shown in **Table 4.6**). After applying the proposed algorithm, the gate realization at the molecular level is shown below in **Figure 4.6**. Gate sequence of NOR gate is 3'-s–$I_{1,0}$-$I_{2,0}$–$\bar{s}$-5'. Digital inputs (0,0), (0,1), (1,0) and (1,1) are provided in terms of DNA sequences as shown in **Table 4.4**. The inputs and the immobilized gate sequences are allowed to hybridize and the final output is read out in the form of success or failure of hybridization. Case 1 gives output '1' whereas Case 2, 3 and 4 gives output '0' as shown in **Figure 4.6**.

Table 4.6: Simulation result of DNA-NOR gate.

$\mathbf{I_1}$	$\mathbf{I_2}$	$\mathbf{I_1}$ **NOR** $\mathbf{I_2}$	**Simulation Result**
0	0	1	Linear
0	1	0	Hairpin
1	0	0	Hairpin
1	1	0	Hairpin

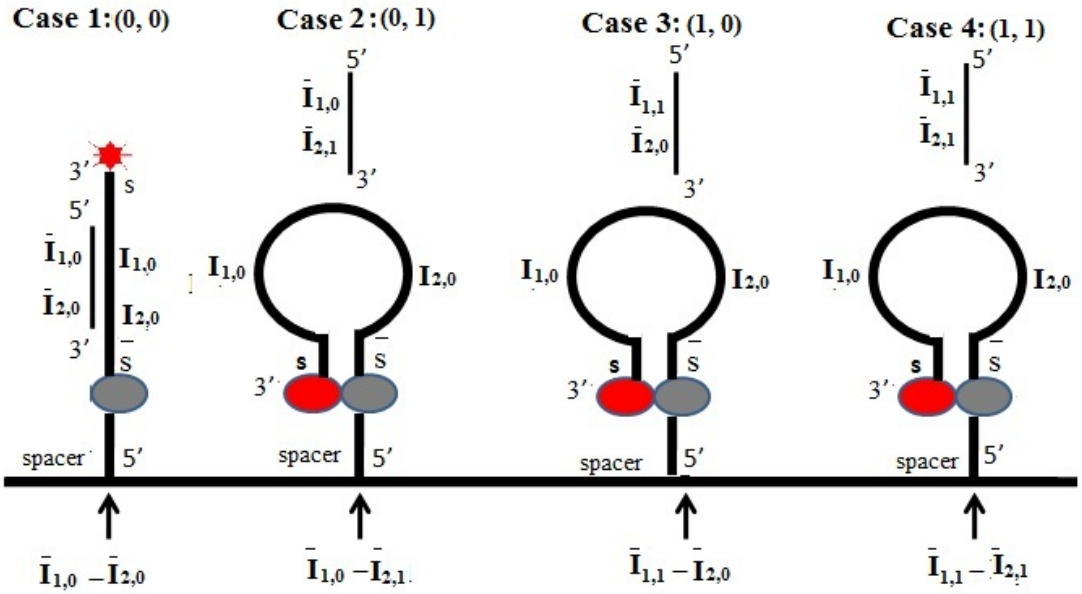

Figure 4.6: Simulation of a DNA-NOR gate. Case 1: $I_1 = 0$ and $I_2 = 0$. Case 2: $I_1 = 0$ and $I_2 = 1$. Case 3: $I_1 = 1$ and $I_2 = 0$. Case 4: $I_1 = 1$ and $I_2 = 1$.

HALF-ADDER Realization

Half-adder adds two single-bit binary digits to produce two bits as output; one bit for the Sum and other bit for the Carry. The **Table 4.7** represents the truth table of half-adder. Applying the above proposed strategy, two DNA gate strands are obtained; one for sum and another for carry as shown below.

Sum Strand: 3'-s-$I_{1,0}$-$I_{2,1}$-$I_{1,1}$-$I_{2,0}$-$\bar{s}$-5'

Carry Strand: 3'-s-$I_{1,1}$-$I_{2,1}$-$\bar{s}$-5'

Table 4.7: 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

Both strands are immobilized on a gold surface at its 5'end and inputs are provided for each of the four digital input combinations as shown below:

Case 1 for Sum strand: Input are provided in the form of DNA sequences equivalent to (0,0). As shown in **Figure 4.7**, the gate strand continues to remain in hairpin structure hence the output is read as '0' which is in agreement to digital half-adder.

Case 2 for Sum strand: Simulation of Case 2 is shown in **Figure 4.7**. On providing input $\bar{\mathtt{I}}_{1,0}$-$\bar{\mathtt{I}}_{2,1}$ which is equivalent to digital input (0,1). The loop finds its complementary counterpart and hybridizes to obtain linear structure even in presence of dimer (output = 1).

Case 3 and Case 4 for Sum strand: Figure 4.7 shows the Case 3 and Case 4 simulation on providing (1,0) and (1,1) as input. The output of Case 3 is '1' whereas Case 4 has output '0'.

Similarly, the simulations for Carry strand for all the four input combinations are shown in **Figure 4.8**.

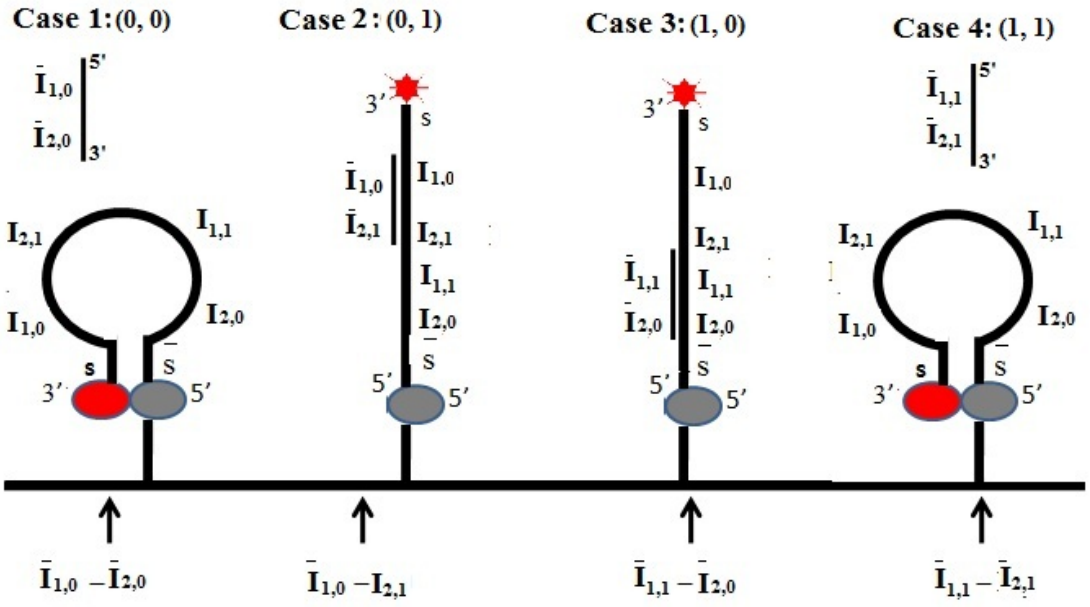

Figure 4.7: Simulation of Sum strand for Half Adder

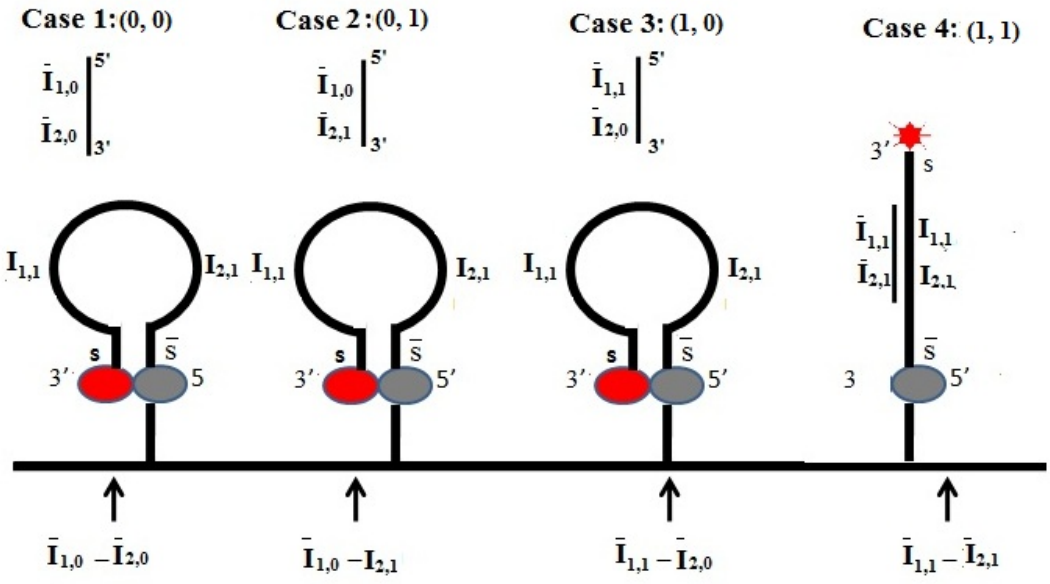

Figure 4.8: Simulation of Carry Strand for Half Adder

Full Adder Realization

Full adder adds three single bit inputs and produce two bits as output i.e. one for Sum bit and other for Carry (C_{OUT}). Two input bits are represented as I_1 and I_2

whereas the third input bit is carry bit represented as C_{IN} in general. Truth table of Full adder is shown **Table 4.8** below:

Table 4.8: Truth Table of Full Adder.

$\mathbf{I_1}$	$\mathbf{I_2}$	$\mathbf{C_{IN}(I_3)}$	$\mathbf{Carry(C_{OUT})}$	$\mathbf{Sum(S_i)}$
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

A Full-adder is electronically implemented using five logic gates: two EX-OR gates, two AND gates and one OR gate (shown in **Figure 4.9(a)**). In this model parallelism feature is fulfilled as only two sets of DNA gate sequences i.e. one set for sum sequence and one set for carry sequence is sufficient to simulate a full adder instead of simulating all the five logic gates separately. (shown in **Figure 4.9(b)**).

The technique of designing the sum and carry strand is same as half-adder but with a small modification i.e. instead of two variables, three variables are involved where the third variable is C_{IN}. Whenever the input strand finds its complementary sequence in Sum and Carry strand, it undergo hybridization and hence retains its linear conformation whereas in case of unsuccessful hybridization the sum or carry strand folds to induce hairpin like structure in presence of ND.

Sum strand: 3'-s-$I_{1,0}$-$I_{2,0}$-$I_{3,1}$-$I_{1,0}$-$I_{2,1}$-$I_{3,0}$-$I_{1,1}$-$I_{2,0}$-$I_{3,0}$-$I_{1,1}$-$I_{2,0}$- $I_{3,0}$-$I_{1,1}$-$I_{2,1}$-$I_{3,1}$-$\bar{s}$-5'

Carry strand: 3'-s-$I_{1,0}$-$I_{2,1}$-$I_{3,1}$-$I_{1,1}$-$I_{2,0}$-$I_{3,1}$-$I_{1,1}$-$I_{2,1}$-$I_{3,0}$-$I_{1,1}$-$I_{2,1}$-$I_{3,1}$-$\bar{s}$-5'

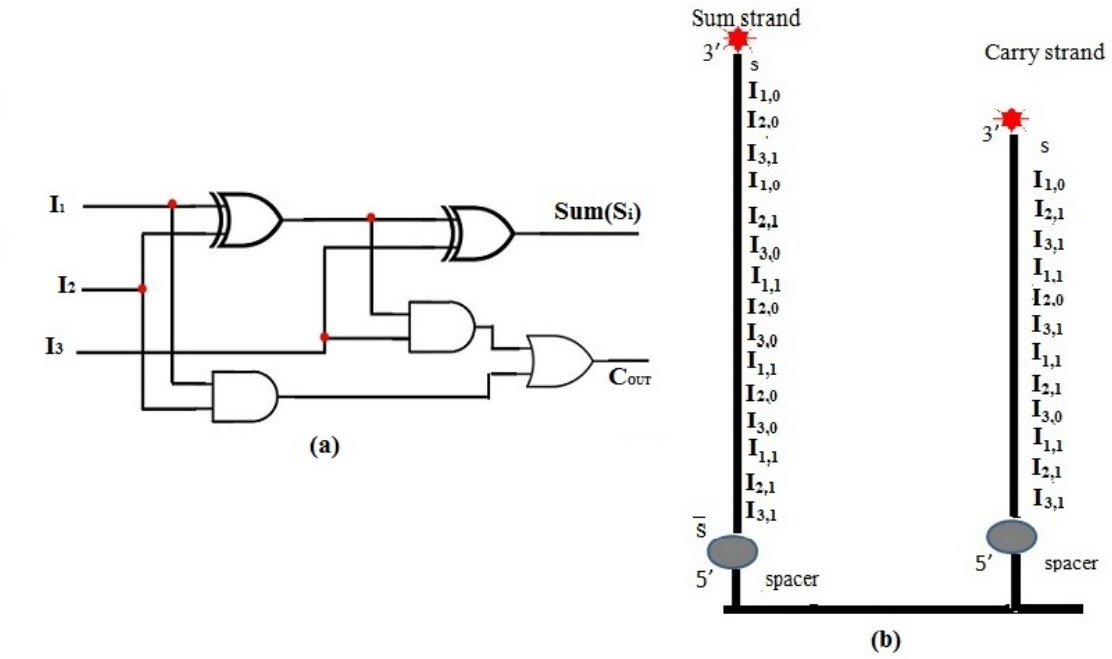

Figure 4.9: (a) Digital Full-Adder circuit (b) DNA Sum and Carry strands for Full-Adder

The simulation result of Full-Adder is shown in **Table 4.9**.

Table 4.9: Simulation result of DNA Full-Adder.

$\mathbf{I_1}$	$\mathbf{I_2}$	$\mathbf{C_{IN}(I_3)}$	$\mathbf{Carry(C_{OUT})}$	$\mathbf{Sum(S_i)}$
0	0	0	Hairpin	Hairpin
0	0	1	Hairpin	Linear
0	1	0	Hairpin	Linear
0	1	1	Linear	Hairpin
1	0	0	Hairpin	Linear
1	0	1	Linear	Hairpin
1	1	0	Linear	Hairpin
1	1	1	Linear	Linear

Boolean Circuit Simulation

Apart from simulating logic gates, the proposed algorithm is employed to simulate Boolean circuits as well. An instance of a Boolean circuit with five logic gates is shown in **Figure 4.10**. I_1, I_2 and I_3 are the three input variables and Z_i is the output variable. **Table 4.10** shows the truth table for the corresponding circuit.

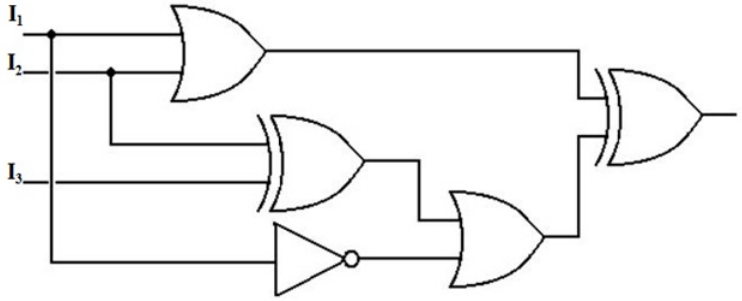

Figure 4.10: An instance of Boolean circuit with five logic gates

Table 4.10: Truth table of the Boolean circuit.

$\mathbf{I}_1$	$\mathbf{I}_2$	$\mathbf{I}_3$	$\mathbf{Z}_i$
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

This simulation example demonstrates the parallelism feature of this model. Instead of employing five different gate strands for each logic gate only one gate /

operator strand is sufficient to evaluate the entire circuit. Gate strand obtained for the circuit by applying the proposed modes is: 3'-$\mathtt{s}$-$I_{1,0}$-$I_{2,0}$-$I_{3,0}$-$I_{1,0}$-$I_{2,0}$-$I_{3,1}$-$I_{1,1}$-$I_{2,0}$-$I_{3,0}$-$I_{1,1}$-$I_{2,1}$-$I_{3,1}$-$\bar{\mathtt{s}}$-5'. **Figure 4.11** demonstrates the simulation result of Boolean circuit.

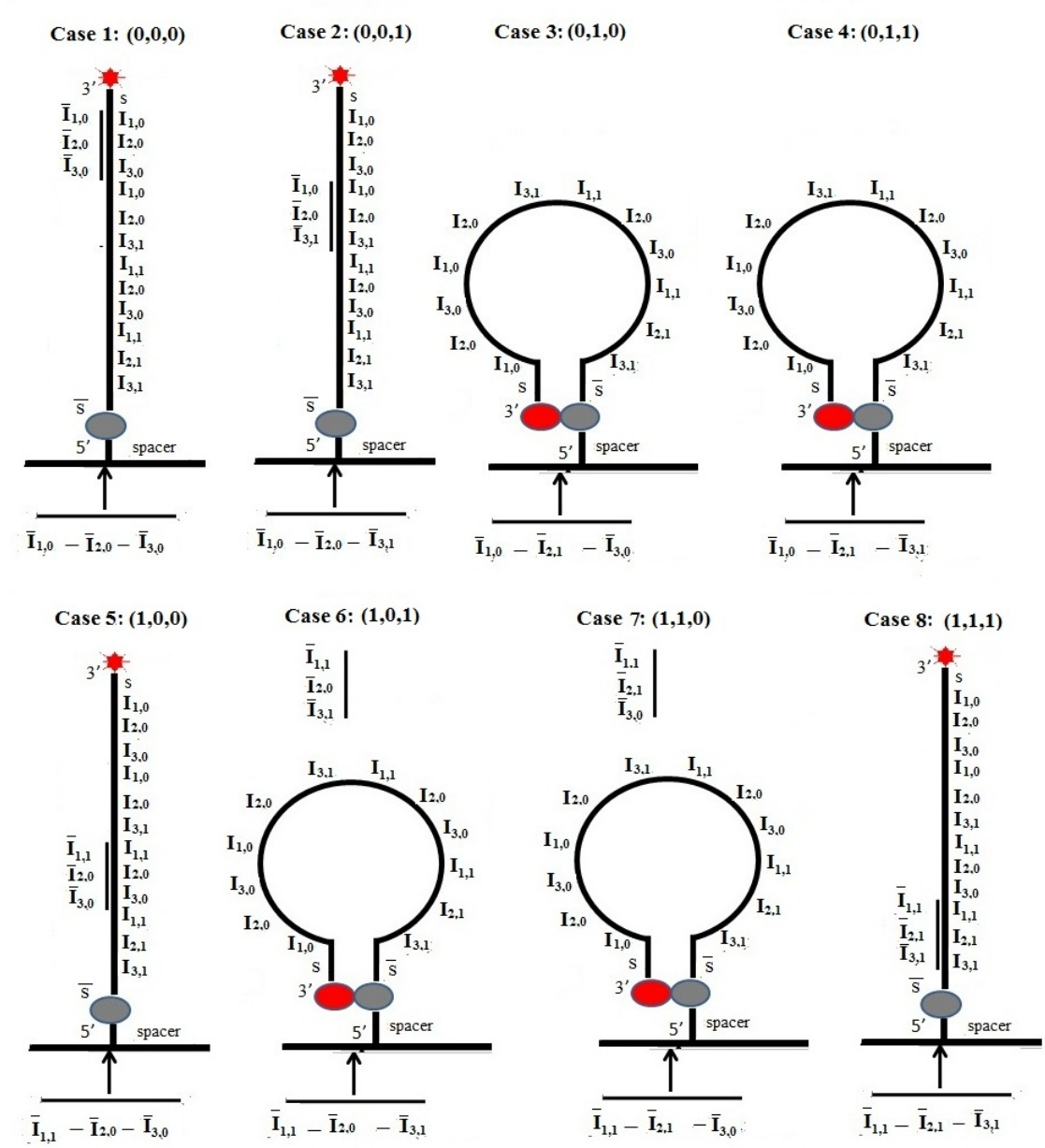

Figure 4.11: Simulation result of an instance of Boolean circuit

Realization of Four-Bit Carry Ripple Adder

Multiple Full adder circuits can be cascaded in parallel to add an N - bit number. For an N - bit parallel adder, there must be N number of Full adder circuits. A ripple carry adder is a logic circuit in which the carry-out of each Full adder is the carry in of the succeeding next most significant Full adder. It is called a ripple carry adder because each carry bit gets rippled into the next stage. Circuit diagram of a 4-bit ripple carry adder is shown in **Figure 4.12**.

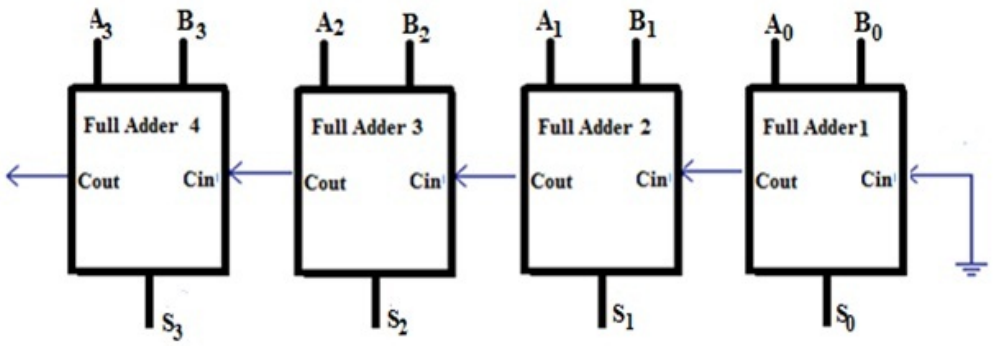

Figure 4.12: Four Bit Carry Ripple Adder.

For example, let us consider the addition of two four bit binary number i.e. A = 1010 and B = 1101 using DNA strands. The addition is done with the help of one set of Half-adder DNA strand and three set of Full-adder DNA strands. The LSB of both A and B i.e. A_0 and B_0 are added in first adder and the carry is fetched to the next adder and so on. The carry bit is ligated to the existing input strand and acts as third input variable and fetched to the next adder. The success or failure of hybridization of Sum sequence with the input strand gives the result of addition. The simulation of Four-Bit Carry Ripple adder is shown in **Figure 4.13**.

As the reaction of Sum sequence at Adder 1 results in linear structure therefore the output S_0 is read as 1 whereas the carry strand of Adder 1 result in hairpin

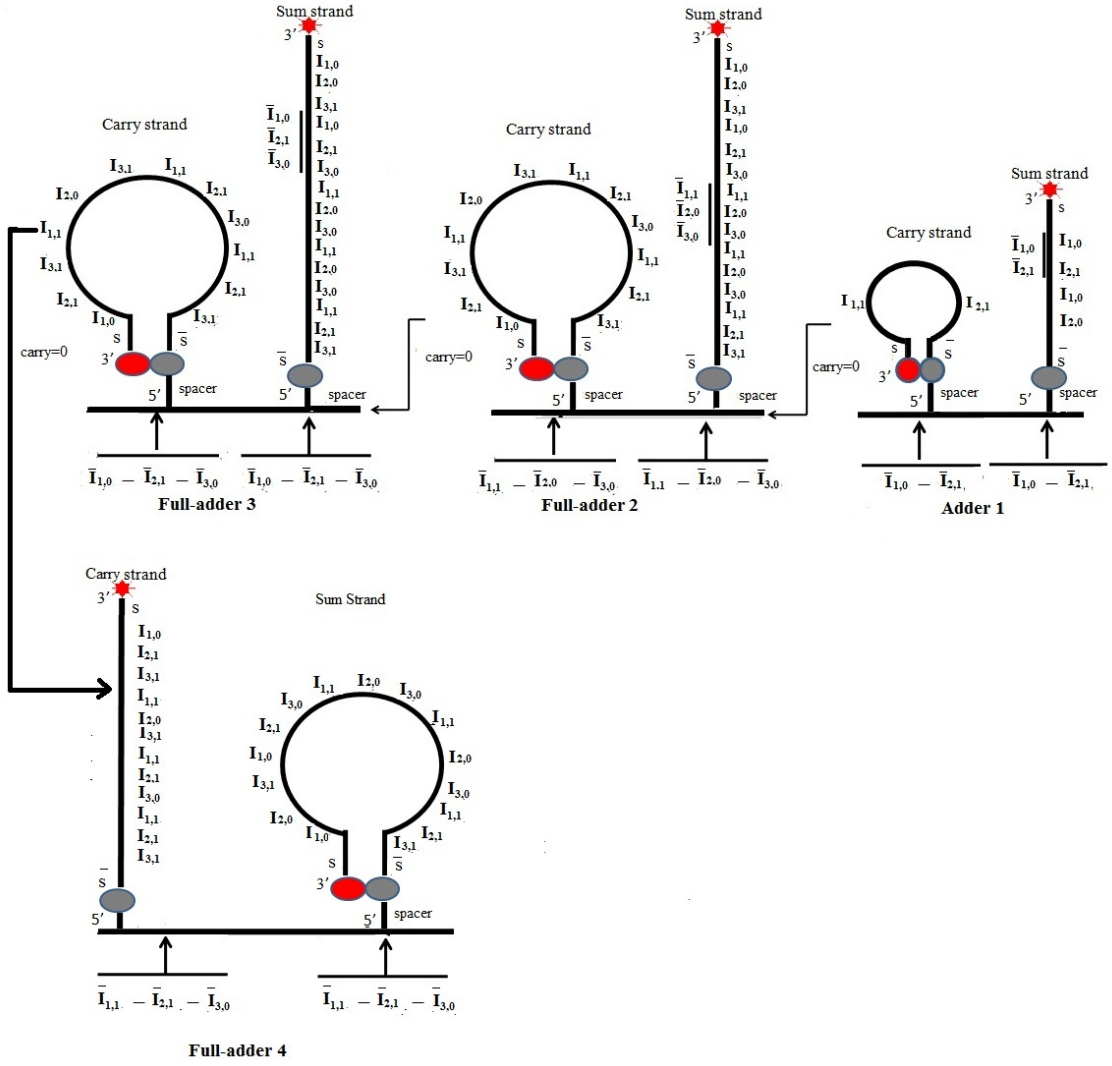

Figure 4.13: Simulation of Four Bit Carry Ripple Adder.

structure which is read as carry bit 0. The carry bit of Adder 1 is fetched as third input to Full-adder 2. Biochemical reaction of Sum sequence in Full-adder 2 gives S_1 as 1 similarly Sum strand in Full-adder 3 gives output as 1 but in adder 4 the reaction of Sum strand gives the output S_3 as 0. Therefore the result obtained after adding A and B is S_3 S_2 S_1 S_0 i.e. 0111 which is in agreement to digital Four-Bit Carry Ripple adder output.

4.3 Summary

A reusable, generalized, parallel model is presented in a cost effective way that incorporate an algorithm for designing the gate strands along with induced hairpin formation property of naphthyridine dimer. The model is applicable to simulate any kind of logic gate without making any change in the design strategy. Further the theoretical simulation results of DNA based NAND, NOR, Half-adder, Full-adder and Four-Bit Carry Ripple adder are demonstrated using the model. Due to limited use of error prone enzymatic reactions, this model is reliable and easy to implement. Parallelism feature is fulfilled as the operations of several logic gates could be implemented by DNA gate strands which equals to the number of output column in the corresponding truth table. The immobilized gate strands could be reused by washing off the dimers. Apart from simulating independent logic gates, this model could effectively be employed to simulate combinational circuits.