DIGITALCOMMONS
@WAYNESTATE

Wayne State University

Wayne State University Dissertations

January 2018

Design Of Dna Strand Displacement Based Circuits

Aby Konampurath George
Wayne State University, fs6159@wayne.edu

Follow this and additional works at: https://digitalcommons.wayne.edu/oa_dissertations

Part of the Nanoscience and Nanotechnology Commons

Recommended Citation

Konampurath George, Aby, "Design Of Dna Strand Displacement Based Circuits" (2018). *Wayne State University Dissertations*. 1937.
https://digitalcommons.wayne.edu/oa_dissertations/1937

This Open Access Dissertation is brought to you for free and open access by DigitalCommons@WayneState. It has been accepted for inclusion in Wayne State University Dissertations by an authorized administrator of DigitalCommons@WayneState.

DESIGN OF DNA STRAND DISPLACEMENT BASED CIRCUITS

by

ABY KONAMPURATH GEORGE

DISSERTATION

Submitted to the Graduate School
of Wayne State University,
Detroit, Michigan
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

2018

MAJOR: ELECTRICAL ENGINEERING

Approved By:

Harpreet Singh Date

Feng Lin

Guangzhao Mao

Le Yi Wang

© COPYRIGHT BY

ABY KONAMPURATH GEORGE

2018

All Rights Reserved

DEDICATION

This thesis is dedicated to my parents, my wife, and my teachers who made me who I am today..

ACKNOWLEDGMENTS

I believe that Dr. Singh, my thesis advisor, molded my thought process as a researcher immensely. He also gave me tremendous support to go into a research area which is very new to me. Without his encouragements I may not be able to achieve whatever I have now as a researcher. Dr. Singh supported me by providing almost everything I needed for my research and the scientific growth. He always encouraged me by giving challenging problems and constructive feedback. I will be always grateful to Dr. Singh for his nurturing and guidance, not only for my research, but also for making me an even better person.

I am extremely grateful to my thesis committee members: Dr. Le Yi Wang, Dr. Feng Lin, and Dr. Guangzhao Mao. They were always willing to give valuable suggestions, and feedback for making my research fruitful. Particularly their suggestion during my prospectus helped me in designing my thesis work in a more professional way. I would also like to thank Dr. Mumtaz Usmen, Dr. Abhilash Pandya, and Dr. Syed Mahmud for encouraging me throughout my research. I would like to thank our collaborator Dr. Athanasios V. Vasilakos for his valuable suggestions and guidance in creating a visionary paper on the emerging DNA circuits.

I would also like to express my gratitude to Dr. Haipeng Liu and his Ph.D. student Ms. Jingchao Xi for helping me in implementing the circuits in the wet lab. As a novice to wet lab experiments, without their help I may not be able to see how the simulation experiments I performed worked in the real world. I would like to thank all my teachers present and past, without their support I may not have achieved this much in life. I would like to thank my mentors from IIT Roorkee, Dr. P Sumathi, my supervisor during my Master's thesis, Professor M K Vasantha, who showed me who is a good teacher and many other

good teachers. As a person coming from a remote village in India, I would like to thank my primary teachers who inspired me to achieve great things in life. I especially thank Shyla Teacher who was my class teacher in my primary class. Her love and care are always with me.

I would also like to thank my lab-mates with whom I spent most of my life at Wayne State University. The discussions we had on the experimental parts on the implementation of different works help me a lot in my research. In particular, I would like to thank Amjad Almatrood, Ishak O K, Ishan Jindal, Mohammad Ashraf Ali, and Chang Fu. Wayne State University is truly a great place to do scientific work. I would also like to thank my dear friends on our Friday poker nights, especially Dr. Sandeep and Deepesh Gopalakrishnan for their valuable suggestions throughout my research.

In conclusion, I would like to thank my family for everything. I would like to thank my father K V George, mother Achamma George, brother Vibin George, and grandparents for being with me in completing all my dreams, aspirations, and wishes. Finally, a big thank to my wife Vincy Mary Vavachan, who stood as a pillar of support with love, care, and support throughout the journey.

TABLE OF CONTENTS

DEDICATION **iii**

ACKNOWLEDGMENTS **iv**

LIST OF FIGURES **x**

LIST OF TABLES **xv**

CHAPTER 1: Introduction **1**

1.1 Motivation and Context 2

1.2 Preliminaries 5

1.2.1 Structure of DNA 5

1.2.2 DNA Strand Displacement Operation 8

1.3 Summary of Contributions 9

CHAPTER 2: Literature Review: DNA Circuits **11**

2.1 Introduction 11

2.2 Enzyme-free DNA Digital Design 11

2.2.1 Seesaw Gate Motif Based Circuits 12

2.2.2 Localized DNA Hairpins Based Circuits 17

2.2.3 DNA-based CRN on a Surface [1] 21

2.2.4 Molecular Spider System Based Circuits [2] 25

2.2.5 Comparison of Different Methods 29

2.3 DNA Analog Design 32

2.3.1 Amplification Circuits 33

2.3.2 Arithmetic Circuits 33

2.3.3 Control Circuits 34

2.3.4 Timer Circuits 35
2.3.5 Challenges for Analog Circuits 35
2.4 Conclusion 36
CHAPTER 3: Spatially Localized Majority Gates **37**
3.1 Introduction 37
3.2 Spatially Localized DNA Majority Gate 38
3.2.1 Methodology 38
3.2.2 Simulation Results and Discussion 43
3.3 Spatially Localized Five Input Majority Gate 47
3.3.1 Input Strand 47
3.3.2 Input Translator Hairpin Strand 47
3.3.3 Fuel Strand 48
3.3.4 Threshold Strand 49
3.3.5 Output Translator Strand 49
3.4 Simulation Results 51
3.4.1 Implementation of Five Input Majority Gate in Visual DSD 51
3.4.2 Circuit Design Using Majority Gates 55
3.4.3 Case Study: Full Adder Design 57
3.4.4 Design of an n-Input Majority Gate 59
3.5 Conclusion 61
CHAPTER 4: DNA Strand Displacement Based Inverter Logic Gate **62**
4.1 Introduction 62
4.2 DNA Inverter Gate Design 63
4.3 Simulation Results and Discussion 66

4.4 Conclusion 71

CHAPTER 5: DNA Strand Displacement Based Fuzzy Inference Engine 73

5.1 Introduction 73

5.2 Minimum and Maximum Functions Using DNA 75

5.2.1 Basic Concepts 75

5.2.2 Minimum Gate 75

5.2.3 Fan-out Gate 78

5.2.4 Maximum Gate 80

5.3 Fuzzy Logic 83

5.3.1 Modeling of Fuzzy Expert System 83

5.3.2 DNA Implementation of Mamdani Fuzzy Inference 85

5.4 Simulation Results and Discussion 87

5.4.1 Minimum Gate 87

5.4.2 Fan-out Gate 89

5.4.3 Maximum Gate 92

5.4.4 Mamdani Fuzzy Inference System 94

5.4.5 Novelty of the Work 96

5.5 Wet Lab Implementation of Minimum Gate 97

5.5.1 Sequence Design 97

5.5.2 Circuit Preparation 99

5.5.3 Kinetic Experiments 101

5.6 Conclusion 103

CHAPTER 6: Challenges and Future Research 104

6.1 Introduction 104

6.2 Challenges 104
6.3 Future Research 107
6.4 Conclusion 109
CHAPTER 7: Summary and Discussion **110**
APPENDIX: Visual DSD Codes **113**
PUBLICATIONS **125**
BIBLIOGRAPHY **129**
ABSTRACT **143**
AUTOBIOGRAPHICAL STATEMENT **145**

LIST OF FIGURES

Figure 1.1 The graph showing the advances in productivity of DNA strands [3] . 5

Figure 1.2 Example of a DNA Molecule 6

Figure 1.3 A simple model showing the structure of double-stranded DNA [4] . . 7

Figure 1.4 Basic structures formed by the annealing reaction of single-stranded(a, b, d, e) molecules or partially double stranded (c) molecules 8

  Figure (a) Structure A 8

  Figure (b) Structure B 8

  Figure (c) Structure C 8

  Figure (d) Structure D 8

  Figure (e) Structure E 8

Figure 1.5 Example of a DSD operation 9

Figure 2.1 Stoichiometric triggering and catalytic cycle of a DNA gate motif [5] . 13

Figure 2.2 Example of threshold operation in a DNA gate motif [5] 14

Figure 2.3 Abstract diagram of a DNA seesaw gate motif acting as an amplifying gate 14

Figure 2.4 Abstract diagram of a DNA seesaw gate motif acting as an integrating gate 15

Figure 2.5 Cascaded operation of integrating gate and amplifying gates as AND gate and OR gate [6] 16

Figure 2.6 DSD operation in a hairpin structure 17

Figure 2.7 OR operation with two hairpin structures 18

Figure 2.8 AND operation with a single hairpin structure 19

Figure 2.9 Hairpin opening operation when the input bind to the hairpin 20

Figure 2.10 Hairpin opening operations for spatially localized 2 input AND gate with both inputs were present [7] 21

Figure 2.11 DNA implementation of unimolecular ($A \to B$) reaction on a surface [1] 22

Figure 2.12 DNA implementation of bimolecular ($A + B \to C + D$) reaction on a surface [1] 23

Figure 2.13 CRN based logic gates with corresponding molecular reactions [1] . . 24

Figure (a) NOT Gate 24

Figure (b) AND Gate 24

Figure (c) OR Gate 24

Figure 2.14 The spatial path followed by a molecular spider in normal sites when $X = 0$ and $X = 1$ [2] 26

Figure 2.15 DNA walker based logic gates with corresponding molecular reactions [2] 27

Figure (a) AND Gate 27

Figure (b) OR Gate 27

Figure (c) NOT Gate 27

Figure 2.16 A possible implementation of molecular spider [2] 28

Figure (a) Spider Legs 28

Figure (b) Spider with arm 0/1 28

Figure (c) Non-alterable Sites 28

Figure (d) Exit mechanism 28

Figure 3.1 An example of DSD operation in a DNA hairpin structure 40

Figure 3.2 DNA hairpin strand operation for localized majority gate with only one input present 41

Figure 3.3 DNA hairpin strand operation for localized majority gate with two inputs present 42

Figure 3.4 Abstract diagram of a three input majority gate with labels showing interaction between hairpins 43

Figure 3.5 Time courses for different input combinations in Visual DSD 44

Figure 3.6 The final concentrations obtained for a three input majority gate for different input combinations 45

Figure 3.7 The spatial arrangement of DNA strands for a five input majority gate 48

Figure 3.8 Abstract diagram of a five input majority gate 52

Figure 3.9 The main reactions associated with a five input majority gate for three inputs (A, B, and E) 54

Figure 3.10 Circuit diagram of a full adder using majority gates 57

Figure 3.11 Dual rail implementation of full adder using majority gates 58

Figure 3.12 Abstract diagram showing strands of full adder circuit using majority gates 59

Figure 4.1 Block diagram of the proposed DNA Inverter 64

Figure 4.2 The DNA reactions associated with the DNA Inverter 65

Figure 4.3 Simulation results of the proposed DNA Inverter 66

Figure 4.4 Full adder circuit using majority gates and inverter 67

Figure 4.5 Full adder abstract diagram using seesaw gates and inverter gate with gate enable 68

Figure 4.6 Full adder simulation results for different input combinations 72

Figure 5.1 DNA design of minimum gate 76

Figure 5.2 DNA reaction diagrams in the minimum gate 77

Figure 5.3 DNA design of fan-out gate 79

Figure 5.4 DNA reaction diagrams in the fan-out gate 80

Figure 5.5 DNA design of fan-out gate with a fan-out of 3 81

Figure 5.6 Block diagram of the proposed maximum gate 81

Figure 5.7 Architecture of fuzzy expert system 83

Figure 5.8 Simulation results of minimum gate in Visual DSD. Response of minimum gate for $[i_2]_0 = 0.6$ and $[i_1]_0 = 0.4$ for the range $r_{min} = 1$ 88

Figure 5.9 Simulation results of minimum gate in Visual DSD. Response of minimum gate for $[i_2]_0 = 0.6$ and $[i_1]_0 = 0.4$ for the range $r_{min} = 2$ 88

Figure 5.10 A heat map showing the variation of settling time (t_s) with changes in the inputs $[i_1]_0$ and $[i_2]_0$ for the range $(0, 1)$ 89

Figure 5.11 A heat map showing the variation of settling time (t_s) with changes in the inputs $[i_1]_0$ and $[i_2]_0$ for the range $(0, 2)$ 90

Figure 5.12 Simulation results of 2 output fan-out gate in Visual DSD. Response of fan-out gate for $[i_1]_0 = 0.5$ for the range $r_f = 1$ 90

Figure 5.13 Simulation results of 2 output fan-out gate in Visual DSD. Response of fan-out gate for $[i_1]_0 = 0.5$ for the range $r_f = 2$ 91

Figure 5.14 A graph showing the variation of settling time (t_s) with changes in the inputs $[i_1]_0$ for the range $(0, 1)$ 91

Figure 5.15 A graph showing the variation of settling time (t_s) with changes in the inputs $[i_1]_0$ for the range $(0, 2)$ 92

Figure 5.16 Simulation results of the maximum gate in Visual DSD. Response of maximum gate for $[i_1]_0 = 0.4$ and $[i_2]_0 = 0.6$ for $r = 1$ 93

Figure 5.17 Heat-map showing the variation of settling time (t_s) with changes in the inputs $[i_1]_0$ and $[i_2]_0$ for the range $(0, 1)$. 93

Figure 5.18 Variation of $\mu_{low}(f)$ with changes in inputs x and y 95

Figure 5.19 Variation of $\mu_{high}(f)$ with changes in inputs x and y 95

Figure 5.20 Variation in the settling time (t_s) of $\mu_{low}(f)$ with changes in inputs x and y 96

Figure 5.21 Variation in the settling time (t_s) of $\mu_{high}(f)$ with changes in inputs x and y. 96

Figure 5.22 Different DNA strands generated from NUPACK 99

Figure (a) I1 99

Figure (b) I2 99

Figure (c) Reporter 99

Figure (d) M1 99

Figure (e) M2 99

Figure 5.23 Experimental results for the minimum gate with different set of input concentrations 103

LIST OF TABLES

Table 2.1 Comparison of different scalable DNA digital design circuits 29

Table 3.1 Truth table of a three input majority gate 39

Table 3.2 Different functions possible from five input majority gate 56

Table 4.1 Abstract diagram for the 16 basic Boolean expressions with two variables 69

Table 5.1 The domains and corresponding base pair sequences for minimum gate obtained from NUPACK software 98

Table 5.2 The details of different strands used in the design of minimum gate . . 100

Table 5.3 The volume of different strands in the solution for different set of input concentrations 102

CHAPTER 1: INTRODUCTION

The conventional electronic circuits are made up of silicon transistors. For the past five decades, the miniaturization of transistors used in integrated circuits were closely following Moore's Law [8]. Nevertheless, in Moore's recent interview given to the IEEE spectrum magazine on the 50^{th} anniversary of Moore's law, he foresees the death of his law in the next decade [9]. According to him, the heat management issues are major while reducing the size of silicon-based transistors and increasing the density further. The cooling methods used are also a major source of e-waste. The photo-lithographic techniques used to manufacture electronic circuits have many limitations such as cost-effectiveness for further miniaturization. Researchers have been searching for an alternative for silicon-based circuits. Even though some replacements are available in the literature for silicon for constructing transistor, they are designed to work at liquid helium temperatures [10,11]. Currently, researchers are searching for computing devices which can perform at scales such as within a cell, where traditional computing devices cannot be used. These issues show the importance of a smaller, faster, powerful, cleaner, and application-specific technology. The research on the development of nano-scale devices and structures altogether is called nanotechnology research. There were many groundbreaking types of research in the field of nanotechnology in the recent years. In my opinion, the greatest achievement of nanotechnology to this date was the development of CRISPR/cas9 [12] technology, which can specifically edit the gene and thus cure many genetic disorders.

Quantum and molecular computing are considered the popular candidates for designing circuits on the nanoscale [13]. In these techniques, instead of transistors, researchers use different mechanisms as the basic building blocks. Quantum cellular automata (QCA) uses quantum dot cells as their basic unit [14]. QCA circuits normally use a majority gate and an inverter to build the circuits [14, 15]. On the other hand, molecular computing deals with computing with molecules at the molecular scale. For medical applications, computing devices made up of materials compatible with biological organisms are preferred. In this thesis, we are developing circuit design approaches using bio-compatible materials with a focus to the biological applications of these circuits.

1.1: Motivation and Context

Biological computing is a discipline that deals with two fundamental questions: How to design automated computing/decision-making circuits using bio-materials and how to incorporate these circuits inside the cell? Someone may wonder, why we need biological/-molecular circuits when there are silicon-based electronic circuits? The traditional electronic circuits are not suitable for directly interacting with biological signals, such as nucleic acids or other analytes. The circuits or devices that can sense, transmit, or make decisions in a biological system must be very small size, lightweight, programmable, and fundamentally bio-compatible. Different types of molecular Boolean logic gates are available in the literature [16–19]. But in these logic gates, the inputs and outputs are of different types, therefore, the scaling up of the circuits are not possible. This problem can be solved by using DNA strands as input and output. DNA is the most suitable material for making biological devices and circuits.

The research on the development of DNA computing circuits is becoming more popular among biologists and engineers with the development of different DNA circuits and devices. Here, the word “circuit” stands for any practical setup which can execute an algorithm. Similar to voltage in conventional silicon-based electronic circuits, the concentration of the

DNA strand is considered as the signal in the DNA circuits. The most interesting feature of DNA strands which makes them suitable for building circuits is the predictability of their double helix structure and the Watson-Crick binding thermodynamics [20]. Another advantage of using DNA computing techniques is the availability of well established experimental procedures of biotechnology and biochemistry. The DNA circuits can act as a programmable bridge between the biological inputs and outputs. There are translators available in the literature, which can convert many biological signals into the required single-stranded DNA signals [21]. Another important aspect of DNA circuits is its modularity. The inputs and outputs are of the same kind, and thus, the output of one gate can be fed as input to another gate. Many complex functions can be physically realized using DNA circuits. The advantage of using DNA strands is that they can store information encoded in their sequence. Also, the structure and physical properties of DNA are well studied and highly predictable compared to other molecules such as proteins. The DNA circuits available in the literature can be broadly classified into enzyme-free circuits and enzymatic circuits. In the enzyme-free set-up, the circuit works completely by itself. On the other hand, enzymatic circuits are assisted by protein enzymes.

The applications of programmable devices and circuits which is made up of biological components are far-reaching. The applications could be *in vitro* (test-tube experiments) or *in vivo* (within the living organism). The applications range from disease diagnostics *in vitro* to targeted drug delivery systems or SMART drugs *in vivo.* The DNA circuits can be used with different DNA structures and machines in order to perform some specific task. For instance, consider a targeted drug delivery system which detects the cancerous cells and delivers the drug to the specified cell. Here, the device has to make some kind of decisions based on the inputs. The decision-making circuitry could be simply a digital system or a more complex analog decision-making system [22]. Another example is the imaging of a particular biomarker. The strength of the signal may be weak. In such a situation, we need an amplifier to increase the visibility of the imaging [23].

The computational powers of DNA were first explored by Adleman, in his famous seven-city Hamiltonian path problem [24]. Ever since researchers are working on DNA, and investigating the computational powers of DNA to build nanostructures and circuits to solve many complex problems [25–28]. The studies available for thermodynamics of DNA hybridization reactions [29] are helping the researchers to predict the structure and interactions of DNA molecules. Many researchers predict DNA computing as a possible replacement of current silicon-based technology [13] for implantable medical applications, because of its computational powers, small size, lightweight, and compatibility with bio-signals. The cost of DNA strand preparation and purification are also decreasing exponentially with advances in the DNA nanotechnology [3]. From the graph given by Carlson (as shown in Fig. 1.1), it can be seen that the productivity has increased by 5 orders of magnitude since. If the trend continues for another few decades, it could be expected that more sophisticated DNA devices and circuits could be produced at an affordable price. DNA as nano-materials can be used for non-biological applications also [30–32].

One remarkable work, towards enzyme-free DNA computing, was done by Erik Winfree [33], in which the self-assembly of the DNA strands was used for computation and problem-solving. A DNA seesaw gate motif based AND and OR gates were developed in [5]. The scaling up of the DNA gate motif circuits are also possible by a dual rail AND-OR logic [6]. The operation of such circuits is based on DNA strand displacement (DSD) technique [34]. Arithmetic circuits [35], feedback controllers [36], and nanorobots for the transport of molecular payloads [22] are also available in the literature, which employs the DSD technique

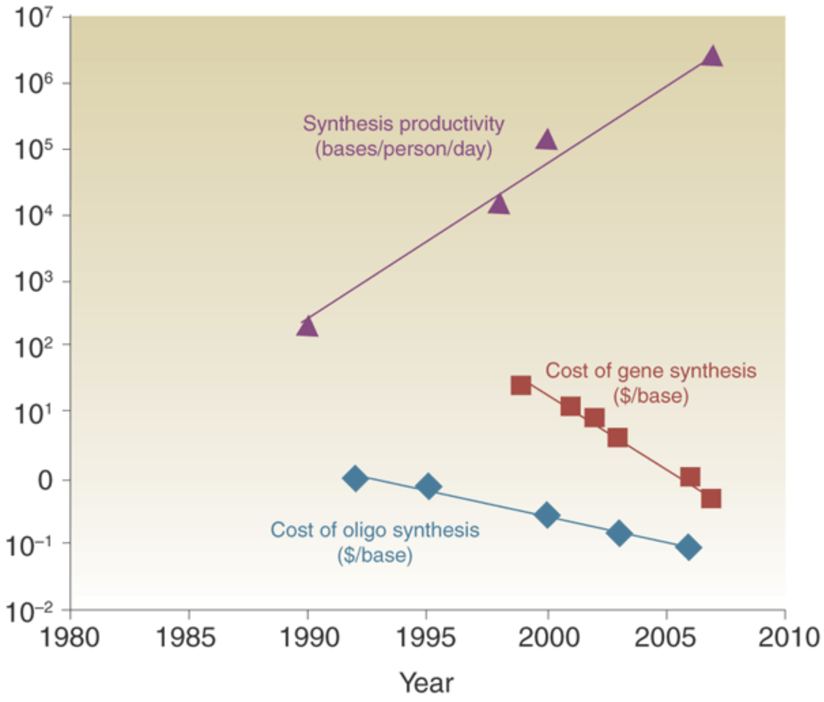


Figure 1.1: The graph showing the advances in productivity of DNA strands [3]

for their operation. A review of devices working on the principle of DSD can be found in [37]. The studies on the conductivity of DNA and DNA polymers are also available in the literature [38–40]. There are some basic digital as well as analog enzyme-free DNA circuits available in the literature which use DSD for their operation. However, more efficient circuit design approaches are still needed to develop DNA circuits with minimum leakage. In this research, we are proposing novel logic operations and analog gates with a view to reducing leakage reactions in the circuits.

1.2: Preliminaries

1.2.1: Structure of DNA

The DNA strand is made up of many individual units called nucleotides. A nucleotide has the following components, (1) a sugar molecule having five carbon atoms, (2) a phosphate group, and (3) one of the four different nitrogenous bases called adenine(A), thymine (T),

5' – G C C T T G A T G T - 3'
3' – C G G A A C T A C A - 5'

Figure 1.2: Example of a DNA Molecule

cytosine (C) and guanine (G). The nucleotides in DNA are called deoxyribonucleotides. The different carbon atoms used in the sugar molecule are numbered from 1' to 5'. The 1' carbon attaches to the base, 5' carbon to the phosphate group, and the 3' carbon to a hydroxyl (OH) group. Nucleotides differ from one another based on the nitrogenous bases they use. Hence, we can refer a nucleotide as A, T, C or G. The nucleotides can be joined together by covalent bonds, to form a sequence. The base pairs (A, T) and (G, C) are complementary to each other, such that a hydrogen bonding is possible only between these pairs [13]. The length of such a base pair is approximately $0.34nm$. The nucleotide sequences have a directionality, and this directionality is determined by the carbon atoms used for the covalent bond while forming the sequence. The directionality can be 3' to 5' or 5' to 3'. Two single-stranded sequences with opposite directionality and complement to each other, are hydrogen-bonded together to form a double-stranded sequence. This complementarity is called Watson-Crick complementarity [20]. A DNA molecule with a double-stranded sequence is shown in Fig. 1.2. It could be noted that the upper strand is aligned with 5'-3' direction and the lower strand with 3'-5', from left to right, and the base pairs are complementary to each other. A simple model showing the helical structure and dimensions of a double-stranded DNA is given in Fig. 1.3. The ribbons represent the two single-stranded sequences, and the sticks between these ribbons indicate the base pairs. The measurements are given in nanometers.

A DNA molecule can exist in different structures. Some of these structures are shown in Fig. 1.4. These structures are formed by the annealing reactions of different single-stranded

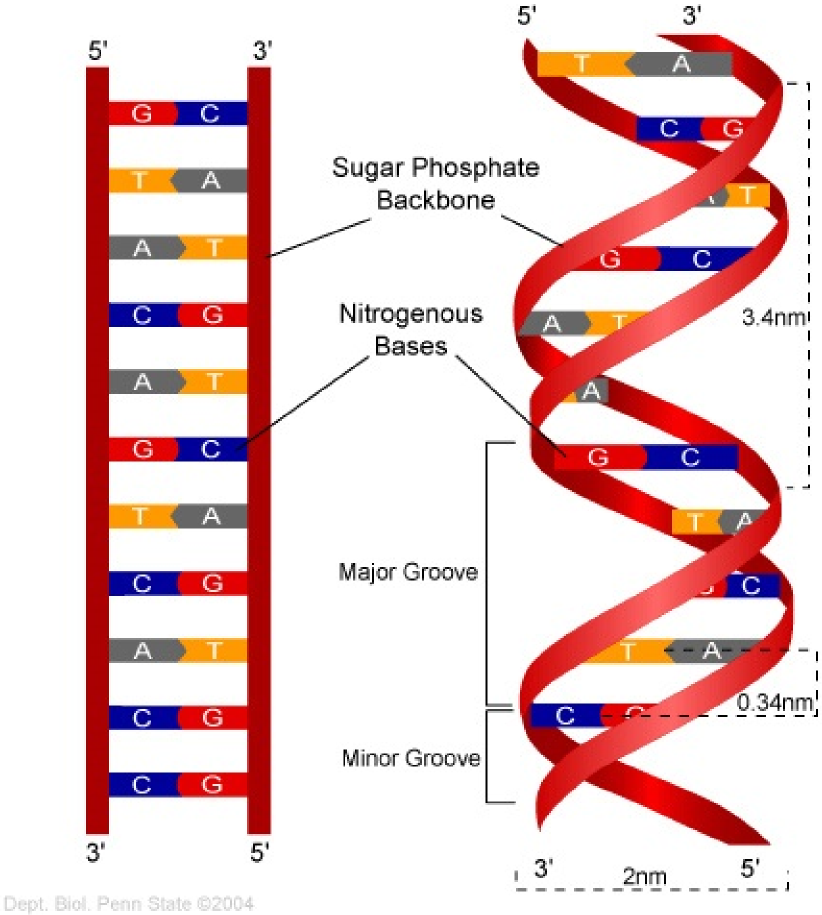
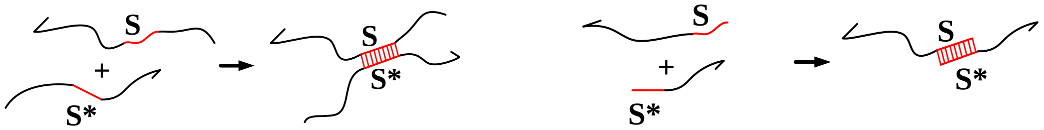


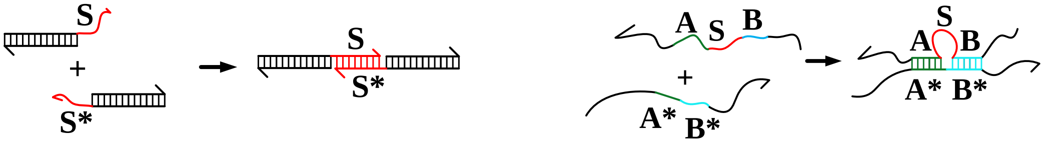
Figure 1.3: A simple model showing the structure of double-stranded DNA [4]

molecules (Fig. 1.4a, 1.4b, 1.4d, 1.4e) or partially double-stranded molecules (Fig. 1.4c). Each curve represents a DNA strand, and the half arrow at the end of each curve represents the direction of the strand (*i.e.*, the 3' of the strand). A particular sequence is represented by a letter (A, B, C or S). The complements of these sequences are represented by A^*, B^*, C^*, and S^*. A Watson-Crick pairing is formed between these complement pairs, and the small lines between two strands in the figure represent the hydrogen bond, which connects these strands. In some representations, the double-stranded molecules are denoted without these small lines. The helical conformation of the DNA double-strand can vary based on different



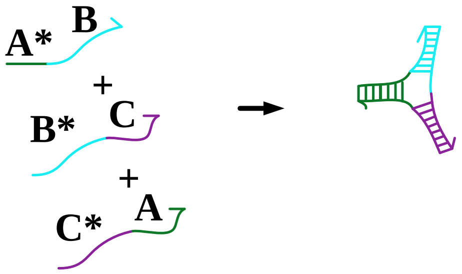
(a) Structure A

(b) Structure B



(c) Structure C

(d) Structure D



(e) Structure E

Figure 1.4: Basic structures formed by the annealing reaction of single-stranded (a, b, d, e) molecules or partially double stranded (c) molecules

factors such as hydration condition, pH, and ionic concentrations of the environment. [41].

1.2.2: DNA Strand Displacement Operation

Strand displacement is a technique in which a pre-hybridized strand is displaced from an incomplete double-strand [42–44]. Unlike the strand displacement mentioned in molecular biology, this DSD is an enzyme-free operation. A simple DSD reaction is shown in Fig. 1.5. Here, 2^* is a short domain, which is exposed for reacting with its complement. This domain is called a toehold. When two strands reacted together, the toehold domain (2) of the single-stranded molecule (B), binds to its complement (2^*) in the partially double-stranded

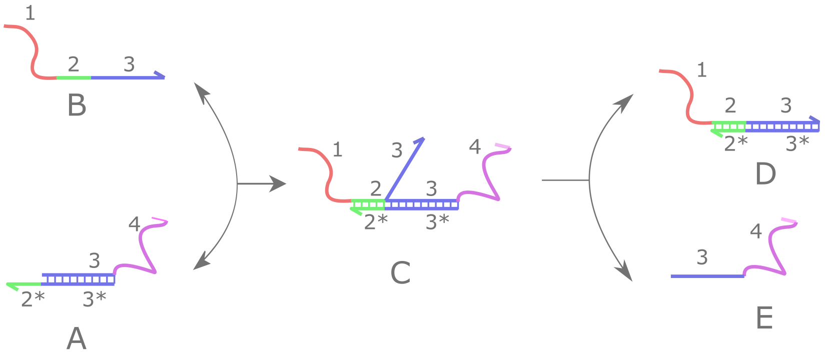


Figure 1.5: Example of a DSD operation

molecule A. Once the toehold binds to its complement, the branch migration takes place, and starts to displace the pre-hybridized strand E, from the incomplete double strand. The branch migration continues, and completely displaces the pre-hybridized strand [34]. The DSD operation in this particular example will result in another double-stranded molecule D, and a single-stranded molecule (E). From this example, we can say that a DSD operation has three steps: Toehold binding, Branch migration, and Release of the strand.

The DSD technique can be used for designing nano-scale circuits [1, 2, 7, 45], nano-structures [46–49], and solving different mathematical problems [25–28]. The kinetics of the DSD operation can be controlled by changing the length and sequence of the toehold domain [50,51]. The kinetics of different strand displacement reactions were predicted in [51].

1.3: Summary of Contributions

The scientific contributions given in this thesis are summarized below:

- Chapter 2 gives a detailed review of existing DNA circuits. The enzyme-free DNA circuits available in the literature can be broadly classified into digital circuits and analog circuits, based on the input and output signals they are using. In the digital circuits, the signals are *true* or *false*, in other words, 0 or 1. On the contrary, the analog circuits can take any value within a specified range and produce output within that range. In this chapter, we discuss the existing enzyme-free digital and analog circuits, and will give a comparative study based on the existing techniques. Chapter 2 is based on our published manuscript given in ref. [52].

- Chapter 3 introduces our work on the development of spatially localized DNA majority logic gates. The majority logic gate is considered as a suitable candidate for designing digital circuits in many promising nanotechnologies. The introduction of a majority logic gate can give more flexibility to the designer while designing large complex circuits. The first section of the chapter gives the design of a three input spatially localized majority logic gate, while the second part gives a generalized procedure for the design of n-input majority logic gate in a spatially localized DNA arrangement. These works are based on our published manuscripts given in ref. [53, 54].
- The design of a DSD based logic inverter gate is presented in Chapter 4. The introduction of an inverter logic gate will pave the way for designing mono-rail circuits, instead of the existing dual-rail circuits. This chapter is based on our published work given in ref. [55].
- Chapter 5 describes the implementation of a DSD based Fuzzy Inference Engine. Here, we discuss different analog logic gates such as the minimum gate, maximum gate, and fan-out gate. A Mamdani fuzzy inference engine was developed using these analog gates to make decisions based on a pre-defined rule set. This chapter is based on our published manuscript given in ref. [56].
- There are different challenges in the development of fully automated decision making systems using DNA strands. These challenges and the future research directions are discussed in Chapter 6.

CHAPTER 2: LITERATURE REVIEW: DNA CIRCUITS

2.1: Introduction

DNA is considered as a perfect candidate for designing implantable structures and dynamic devices, because of its biocompatibility, small size, and programmability. There are different types of nucleic acid circuits which can be used in these structures and devices. Recent advancements in functional DNA nanotechnology will pave the way for designing autonomous device architectures which can be implemented inside biological organisms. Even though the advancements in this field are highly encouraging, there are still many challenges which are to be addressed. The use of functional DNA nanotechnology for medical applications is going to play a vital role in improving the future medical devices and methods.

The rest of the chapter is organized as follows. Section 2.2 explains the different enzyme-free DNA digital design techniques available in the literature. Different enzyme-free analog circuits are explained in section 2.3. Section 2.4 provides a summary of the DNA strand displacement based digital and analog circuits.

2.2: Enzyme-free DNA Digital Design

Different researchers have proposed molecular Boolean logic gates using different properties of molecules [16–19]. But the input and output properties used for these logic gates are of different types, and hence, the scaling up of the circuits are not possible in these types of molecular circuits. This problem can be solved by using DNA strands as input and output. DNA based logic gates that work on the basis of DSD operations were proposed by Qian and coworkers [5, 6]. Even though scaling up is possible with this technique, each gate uses a unique DNA strand, to avoid spurious reactions. Hence, these types of circuits need a large number of unique strands. The speed of operation is also slow since all the strands are diffused together. To overcome these issues, localized design of DNA circuits was proposed [1, 2, 7, 45]. The localized design could be implemented in a DNA substrate [57] or in an Origami box [46]. In this section, we give a review of different techniques that could be used for scalable enzyme-free DNA Boolean circuit design. These circuits use DNA hairpins attached to a DNA Origami box or DNA substrate [7, 45], localized 3-way initiated 4-way strand displacement chemical reactions [1], or molecular spider systems [2] for their operation. Although we are focusing on enzyme-free DNA digital design techniques, there is another category of nucleic acid based computation in cell-free systems, which uses ribozymes and deoxyribozymes for their operations [58, 59]. This logic is called enzyme-based computation systems. The scaling up property in enzyme-based systems are limited because of the mutual interference between the basic building blocks (oligonucleotides) [60].

Basic logic operations such as AND, OR, or NOT alone will not be enough to solve complex problems. In such cases, the number of logic gates required will be very large. Hence, for any design to be suitable for complex digital circuits, the most important attribute is the scalability of the design. In this section, we discuss various methods with a scalable architecture. Even though there are many molecular logic gates available in the literature, they are not suitable for digital design because of the non-uniformity in inputs and outputs. But some molecular circuits use DNA strands as their inputs and outputs, which make them suitable for scalable design. These circuits are broadly classified as: (1) seesaw gate motif based circuits; (2) localized DNA hairpins based circuits; (3) chemical reaction network (CRN) on surface based circuits; and (4) molecular spider based circuits.

2.2.1: Seesaw Gate Motif Based Circuits

The DNA circuit design using seesaw gate motifs were first introduced by Lulu Qian and Erik Winfree in [5, 6]. The seesaw gate motif is a combination of an integrating gate

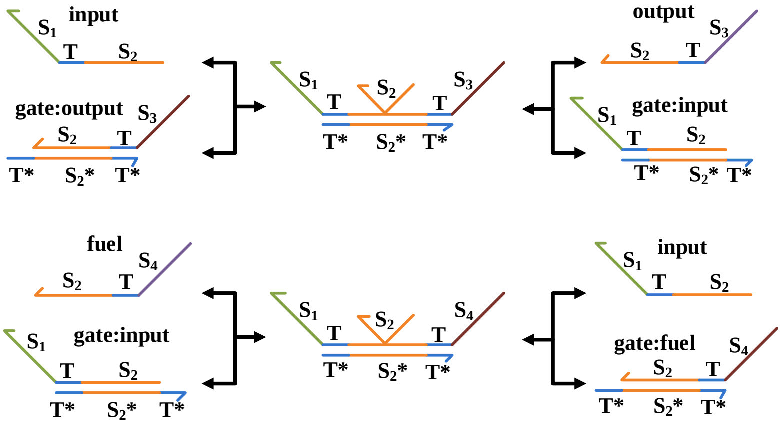


Figure 2.1: Stoichiometric triggering and catalytic cycle of a DNA gate motif [5]

and an amplifying gate. A seesaw gate motif is designed by using four basic elements: inputs, fuel, threshold, and output. The basic idea of a seesaw gate is that, when the input signal concentration exceeds the threshold concentration, the input converts the fuel to the output. The signal strands (input, output, and fuel) have a similar structure. They all are single-stranded molecules, with recognition domains at left and right, and a central toehold domain. Another important component in this design is a gate complex. The left and right recognition domains of the signal strands determine to which gate complex it binds. It should be noted that the toehold domain used here are all same, and their length is small compared to the recognition domains.

The seesaw gate operation consists of three basic operations. The signal strand (input) with exposed toehold domain binds to the gate:output complex, and displaces another signal strand (output) by DSD operation. It could be noted that the output strand produced can again take part in a DSD operation, with the gate:input complex, to produce the input. This process is called stoichiometric triggering, and this is reversible. In a Catalytic cycle, the fuel binds to the gate:input complex produced by stoichiometric triggering, and displaces more

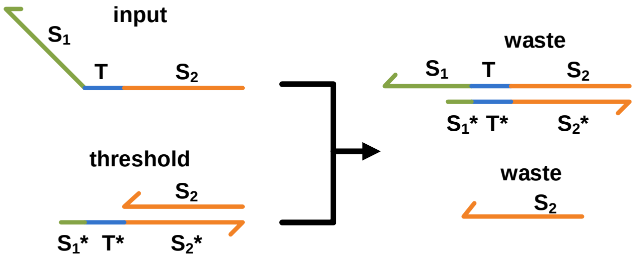


Figure 2.2: Example of threshold operation in a DNA gate motif [5]

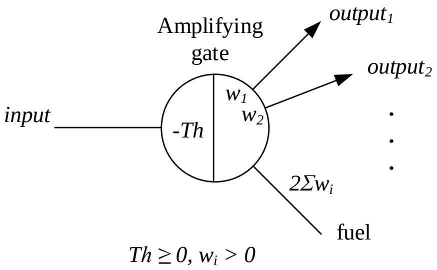


Figure 2.3: Abstract diagram of a DNA seesaw gate motif acting as an amplifying gate

output. The operation of stoichiometric triggering and catalytic cycle are shown in Fig. 2.1. Another operation associated with the seesaw gate is thresholding. In this operation, the input strand binds to the threshold complex to produce a waste strand. The threshold operation is shown in Fig. 2.2. The input strand can combine with either gate:output complex or threshold complex. The selection of operation depends on the concentration of input and threshold complexes. The stoichiometric triggering will occur only when the input concentration is greater than the threshold complexes.

The seesaw gate motif can act either as an amplifying gate or as an integrating gate. An amplifying gate could be used to drive a single output or many outputs. When the amplifying gate drives more than one output, the right side signal recognition domain of the gate:output complexes will have many strands, each strand corresponds to a particular

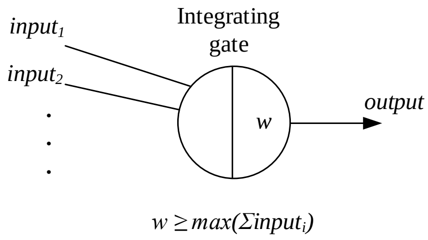


Figure 2.4: Abstract diagram of a DNA seesaw gate motif acting as an integrating gate [5]

output. An abstract diagram of an amplifying gate is shown in Fig. 2.3. It could be noted that the concentration of the fuel strand should be equal to twice that of the summation of all required output concentrations. The individual outputs are given by:

$$output_i = \begin{cases} w_i & \text{if } input > Th \\ 0 & \text{if } input \leq Th \end{cases} \tag{2.1}$$

The abstract diagram of a seesaw gate motif, working as an integrating gate is shown in Fig. 2.4. The integrating gate sums up all the inputs and gives the output. This is achieved by using different left side signal recognition domains and a single output recognition domain. An integrating gate does not require a thresholding operation. The output obtained from an integrating gate is given by:

$$output = input_1 + input_2 + ... \tag{2.2}$$

Different functions can be realized by the cascaded operation of the integrating and amplifying gates. In this section, we are limiting our discussion to logical circuits. The AND gate as well as the OR gate can be designed by selecting the appropriate threshold values. Fig. 2.5 represents the cascaded operation of integrating gate and amplifying gate that results in either an AND gate or an OR gate. X_1 represents the concentration of the input

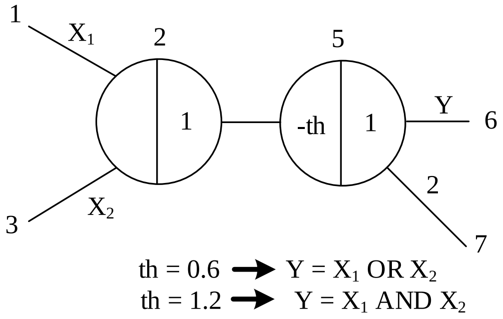


Figure 2.5: Cascaded operation of integrating gate and amplifying gates as AND gate and OR gate [6]

strand S_1TS_2, and X_2 represents the concentration of the input strand S_3TS_2. The first gate used here is an integrating gate, and hence, the output concentration will be a summation of the input concentrations X_1+X_2. This integrating gate is cascaded with an amplifying gate. Here, we are considering the concentration of $0.9\times$ as logical high and $0.1\times$ as logical low (where concentration $1\times = 100nM$ at 20^0C). Now if the threshold concentration is $0.6\times$, then this circuit will act as an OR gate. If the threshold concentration is $1.2\times$, the circuit will be an AND gate [6].

While designing a complex circuit, AND gates in addition to OR gates are not enough. We cannot make an inverter logic gate using seesaw gate motifs. Because of this, a dual rail AND-OR implementation is required for larger circuits. In a dual rail logic, each signal is represented by a pair of signals. One signal represents the logical zero and other represents the logical one. For example, if we want to represent an input $A = 1$, we need two signals with $(A_1 = 1, A_0 = 0)$.

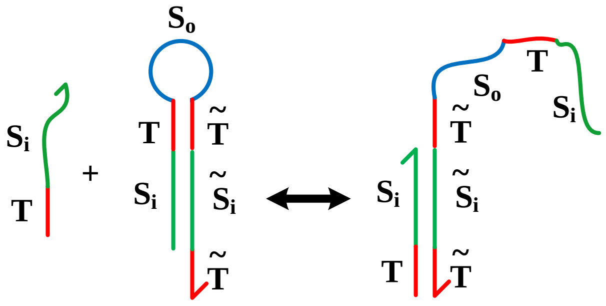


Figure 2.6: DSD operation in a hairpin structure

2.2.2: Localized DNA Hairpins Based Circuits

There are mainly two methods available in the literature [7, 45] which use tethered DNA hairpins to design basic logic gates. Chandran *et al.* [45] propose a method in which, the speed of operation is increased by increasing the concentration at the localized sites. The other method [7], uses a fuel to improve the speed of operation. The structure of the hairpins is also different in these two methods.

Localized Hybridization Circuits by Chandran *et al.* [45]

In this method, cascaded DSD operations on DNA hairpin structures are used to create the AND as well as the OR operations. A propagation gate and fan-out gate are also designed in the same manner. All the gates used in this method use a common toehold domain ($\tilde{T}$) and different specificity domains ($\tilde{S}_i$). The specificity domains are unique to each gate. The basic DSD operation associated with a hairpin structure is shown in Fig. 2.6. When an input domain TS_i reacts with a hairpin structure, the toehold domain T binds to its complement $\tilde{T}$ in the hairpin, initiating a branch migration, which opens the hairpin. This operation is similar to a propagation gate, in which the input domain S_i is converted to an output domain S_o.

Similarly, the OR operation can be performed by using two hairpin structures as shown

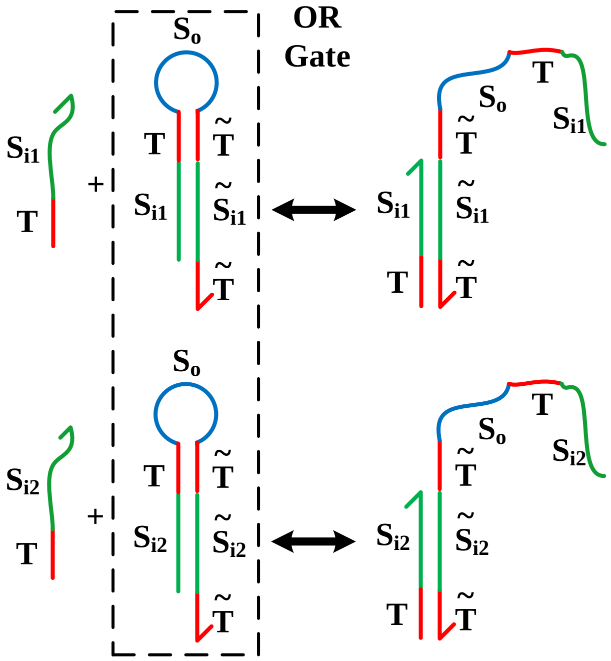


Figure 2.7: OR operation with two hairpin structures

in Fig. 2.7. Here, the inputs are strands TS_{i1} and TS_{i2}. If both inputs are not present, there will not be any DSD operation, and hence no output domain will be present. If any one of the input domains or both the input domains (S_{i1}, S_{i2}) are present, then the input strand will bind to the corresponding DNA hairpin structure, and the branch migration will take place. This results in the opening of any one or both of the hairpin/s, and will result in the presence of the output domain S_o.

The AND operation can be performed by using a single hairpin structure as shown in Fig. 2.8. Here also, S_{i1} and S_{i2} represent the input recognition domains, while S_o represents the output domain. If both inputs are not present, then no reaction will take place, and hence, no output will be produced. If TS_{i1} is present, then toehold domain T binds to its complement, and branch migration of S_{i1} will take place. This will remove the $S_{i1}T$ from the hairpin. If the second input is not present, the hairpin will not open, and hence, no

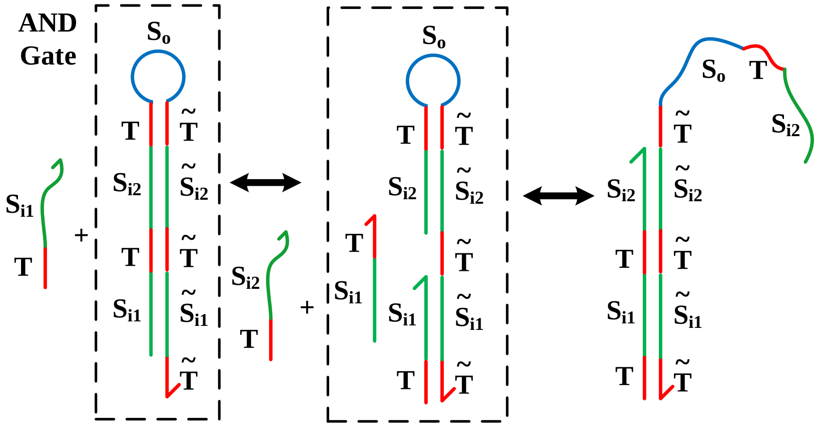


Figure 2.8: AND operation with a single hairpin structure

output will be produced. If the second input TS_{i2} is also present, the toehold mediated DSD operation will take place, which opens the hairpin. Hence, an output recognition domain S_o will be produced, only when both the inputs are present.

The NOT operation is impossible in this method. The scaling up of the circuit is possible by using a dual rail AND-OR logic. The dual rail AND-OR logic is implemented by arranging the DNA AND as well as OR gates in a DNA substrate. These gates are cascaded with the help of propagation gates. A modular design approach discussed in [61] is followed to avoid the spurious reactions.

Spatially Localized Architecture by R A Muscat et. al. [7]

Another spatial architecture for designing logic gates using DNA hairpin structures was discussed in [7]. The architecture developed in [45] uses the diffusion of DNA strands for their operation, and hence they lack in spatial isolation. In the design proposed by Muscat *et al.*, a hairpin structure attached to a DNA substrate (DNA Origami box) is the basic building block of any circuit. The structure of such a hairpin $H(A, Y)$ is shown in Fig. 2.9. We follow the same variable names that are used in the literature. In Fig. 2.9, an

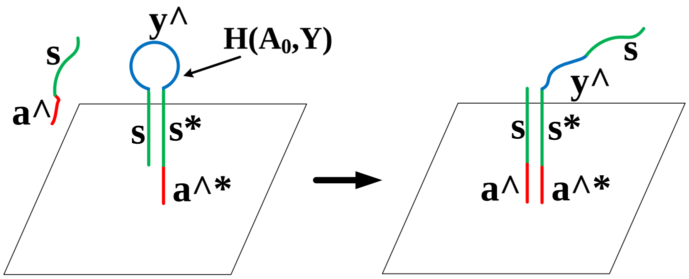


Figure 2.9: Hairpin opening operation when the input bind to the hairpin

input strand $a\hat{}s$ binds to the hairpin with an exposed $a\hat{}*$, and undergoes a DSD operation to displace $y\hat{}s$ by opening the hairpin. This operation could be considered as an input translation, in which the hairpin structure attached to an origami box, binds a diffusible input. For designing different logical operations, various hairpins such as input translators $(H(A,Y), H(B,Y))$, Fuel $(F(Y,X))$, output translator $(H(X,Z))$, and threshold $(H(X,-))$ are used. The spatial arrangement of these DNA hairpins determines the logical operation associated with that design.

An AND gate is designed by arranging the hairpin structures as shown in Fig. 2.10. In this example, we give two inputs. If both the inputs are present, input A will open the hairpin $H(A,Y)$, and input B will open hairpin $H(B,Y)$. These opened hairpins will have an exposed $y\hat{}s$. This exposed $y\hat{}s$ will bind to a freely floating fuel hairpin $F(Y,X)$ to open $x\hat{}s$. These domains can either bind to the threshold or to the output translator. The spatial arrangement of the hairpin structures is in such a way, that the distance between the input translator and the threshold is less compared to the distance between the input translator and output translator. Hence, the $x\hat{}s$ corresponds to the first input, binds to the threshold, and produces a blank domain $blank\hat{}s$. The $x\hat{}s$ corresponds to the second input, binds to the output translator $H(X,Z)$ to produce an output. Thus, an output is produced

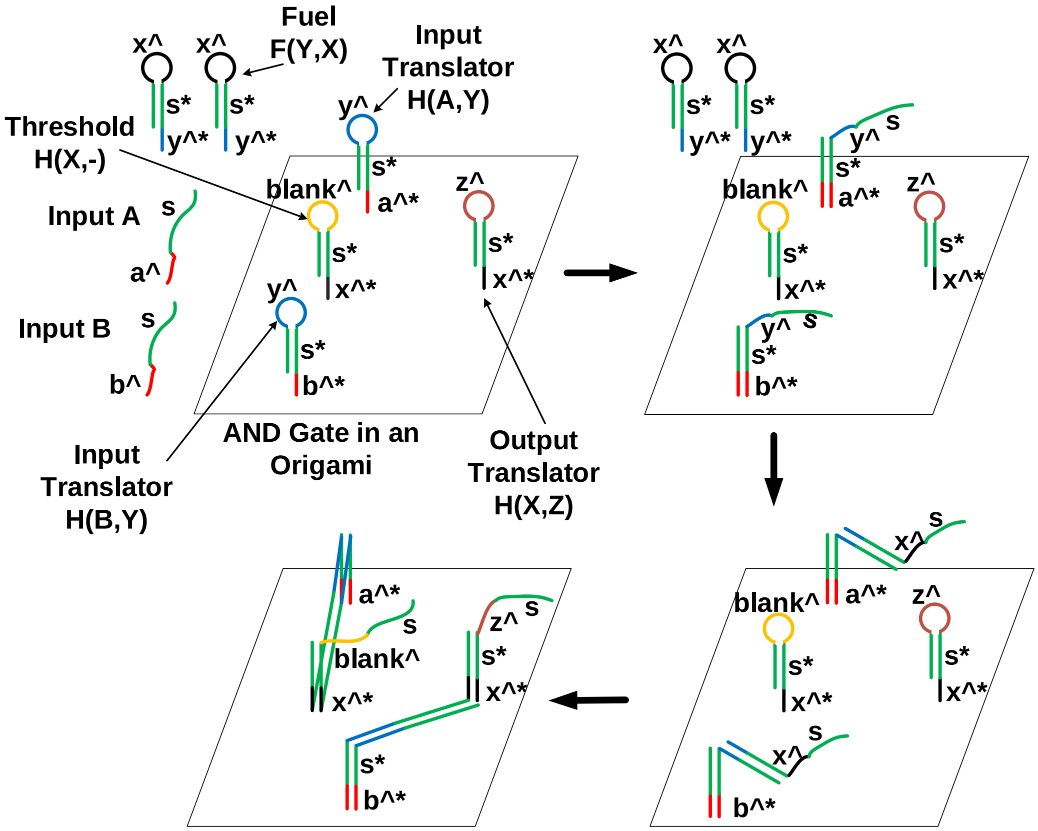


Figure 2.10: Hairpin opening operations for spatially localized 2 input AND gate with both inputs were present [7]

only when both the inputs are present. A similar structure without the threshold hairpin will act as an OR gate. In such an operation, if any, or both, of the inputs are present, the input translator hairpins will produce the corresponding $x\hat{}s$, and it can directly bind to the output translator $H(X, Z)$ to produce the output.

2.2.3: DNA-based CRN on a Surface [1]

Lulu Qian and Erik Winfree proposed a chemical reaction network (CRN) based DNA circuit on a surface. This method uses unimolecular ($A \rightarrow B$) and bimolecular reactions ($A+B \rightarrow C+D$) on a surface, based on 3-way initiated 4-way strand displacement operation. $A + B \rightarrow C + D$ means that if A and B are neighbors, then A gets converted to C, and B

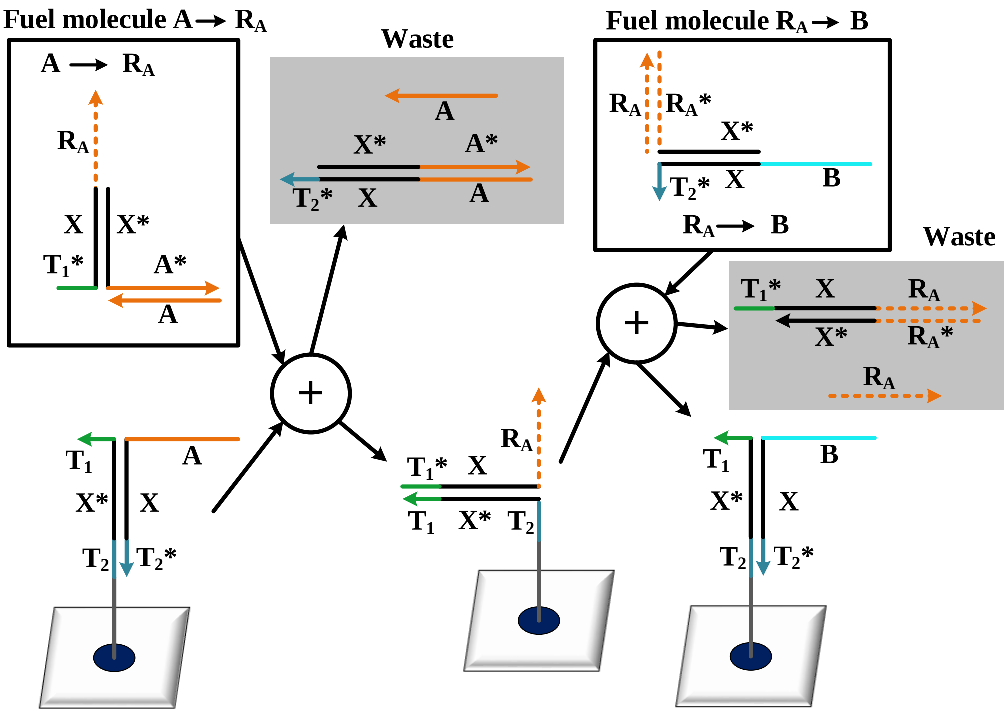


Figure 2.11: DNA implementation of unimolecular ($A \to B$) reaction on a surface [1]

gets converted to D. The DNA strands used in these operations are attached to an Origami surface, and hence the reaction is called surface CRN. A unimolecular reaction ($A \to B$), can be explained with the help of Fig. 2.11. Here, $A \to R_A$ and $R_A \to B$ are the free floating fuel molecules. The fuel first binds to the input A, through the exposed toehold domains T_1 and T_1^*. A three-way initiated four-way strand displacement will produce waste molecules, in addition changes the input A to R_A. The signal R_A thus produced will again undergo another three-way initiated four-way strand displacement operation with the fuel molecule $R_A \to B$, to produce signal B and waste molecules.

Similar to $A \to B$ chemical reactions, $A + B \to C + D$ can also be designed with the help of DNA strands. Such a design is shown in Fig. 2.12. A and B are located at the neighboring sites. The fuel molecule $A + B \to R_{AB}$ will first react with input signals A

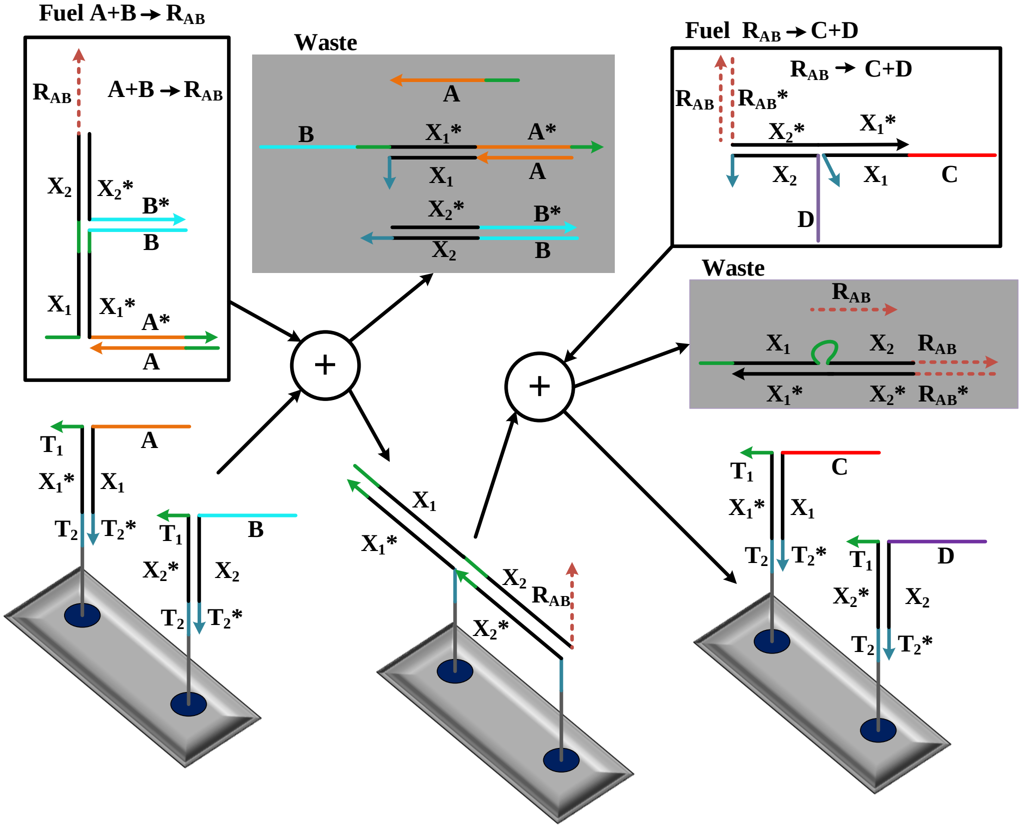
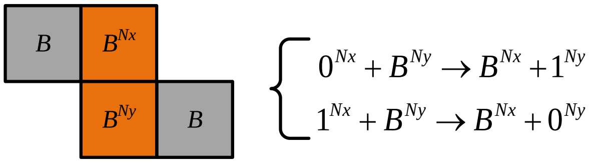


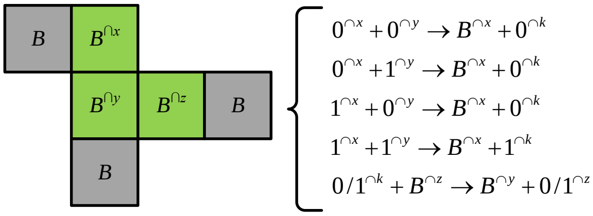
Figure 2.12: DNA implementation of bimolecular ($A + B \to C + D$) reaction on a surface [1]

and B, through a three-way initiated four-way strand displacement reaction, attach the R_{AB} domain to the surface, displaces the inputs A and B from the surface, and produces some waste molecules. The intermediate structure, thus formed will again react with the fuel molecule $R_{AB} \to C + D$, through another three-way initiated four-way strand displacement, by displacing R_{AB} from the surface with signals C and D, and creates some waste molecules. Thus the chemical reactions replace A and B with C and D respectively on the surface.

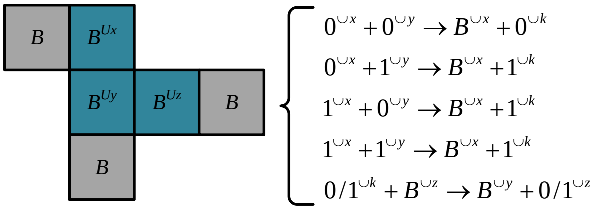
Different logic operations could be performed by using the two different chemical reactions ($A \to B$ and $A + B \to C + D$), and geometrically arranging the surface signals. A simple wire can be designed by two bimolecular reactions $0/1 + B \to B + 0/1$, which means that $0/1$ input signal moves to a blank site, by changing the previously occupied site to a blank


B
B^{Nx}
B^{Ny}
B
$0^{Nx} + B^{Ny} \rightarrow B^{Nx} + 1^{Ny}$
$1^{Nx} + B^{Ny} \rightarrow B^{Nx} + 0^{Ny}$

(a) NOT Gate


B
$B^{\cap x}$
$B^{\cap y}$
$B^{\cap z}$
B
B
$0^{\cap x} + 0^{\cap y} \rightarrow B^{\cap x} + 0^{\cap k}$
$0^{\cap x} + 1^{\cap y} \rightarrow B^{\cap x} + 0^{\cap k}$
$1^{\cap x} + 0^{\cap y} \rightarrow B^{\cap x} + 0^{\cap k}$
$1^{\cap x} + 1^{\cap y} \rightarrow B^{\cap x} + 1^{\cap k}$
$0/1^{\cap k} + B^{\cap z} \rightarrow B^{\cap y} + 0/1^{\cap z}$

(b) AND Gate


B
B^{Ux}
B^{Uy}
B^{Uz}
B
B
$0^{\cup x} + 0^{\cup y} \rightarrow B^{\cup x} + 0^{\cup k}$
$0^{\cup x} + 1^{\cup y} \rightarrow B^{\cup x} + 1^{\cup k}$
$1^{\cup x} + 0^{\cup y} \rightarrow B^{\cup x} + 1^{\cup k}$
$1^{\cup x} + 1^{\cup y} \rightarrow B^{\cup x} + 1^{\cup k}$
$0/1^{\cup k} + B^{\cup z} \rightarrow B^{\cup y} + 0/1^{\cup z}$

(c) OR Gate

Figure 2.13: CRN based logic gates with corresponding molecular reactions [1]

site. The surface arrangement for different logic gates and their reactions are shown in Fig. 2.13. A NOT operation consists of two bimolecular reactions. The input site and output site are connected to wires with blank signals. Initially the neighboring sites, those participating in the NOT operation are assigned with blank signals B^{Nx} and B^{Ny}. The site x is an input site, while y is an output site. Based on the input signal 0^{Nx} or 1^{Nx} reaching the x site, the NOT gate changes y site to 1^{Ny} or 0^{Ny} respectively, and resets the input site x to blank (B^{Nx}). Similarly, the AND gate as well as the OR gate are designed by 6 bimolecular reactions as given in the figure. Here, x and y are the input sites, and z is the output site. The bimolecular reactions are defined for all the possible input combinations (00, 01, 10, and

11). The scaling up of the circuit is possible by interconnecting the basic logic gates such as AND, OR, and NOT with wires. The CRN on surface architectures for fan-out wires and cross-wiring are also given in [1].

2.2.4: Molecular Spider System Based Circuits [2]

Molecular walkers are a special type of molecular machines, designed to move on a pre-programmed track, with a directionality via localized reactions [62–64]. In [65], Dannenberg *et al.* explore the potential of DNA walkers for implementing Boolean logic. But this method is limited to sequential design since the circuit is designed in the form of a binary decision diagram (BDD). Dandan Mo *et al.* [2] utilizes the molecular spider systems with spatial localization to implement different logic circuits. The molecular walker is also called as 'molecular spider' because of its shape. The localized design of molecular spider used in [2] for implementing logic gates has two legs, and one arm with a value of 0 or 1.

A molecular spider is having a body and three limbs (two legs and one arm), and it moves through a track. There are some pre-defined laws for the movement of the molecular spider. Two limbs are always attached to neighboring sites, which are not occupied by other spiders. The sites can be classified as normal and functional sites. The normal site $S_{norm} = \{S_0, S_1, S_l\}$. Where, S_l is the site which binds to a spider leg, and S_0 and S_1 are the sites near to a junction, and they direct the spider based on the spider value. The path followed by the spider for spider value $X = 1$ and $X = 0$ are shown in Fig. 2.14. The attachment sites in the figure are the sites which are attached to the two legs of the spider.

The logical operation of a walker circuit is based on the arrangements of the functional sites and normal sites. A functional site has three possible states. An "on" state which allows the binding of the spider, an "off" state which will not allow binding of the spider, and a "trapped" state which will not allow the spider to move from there. A functional

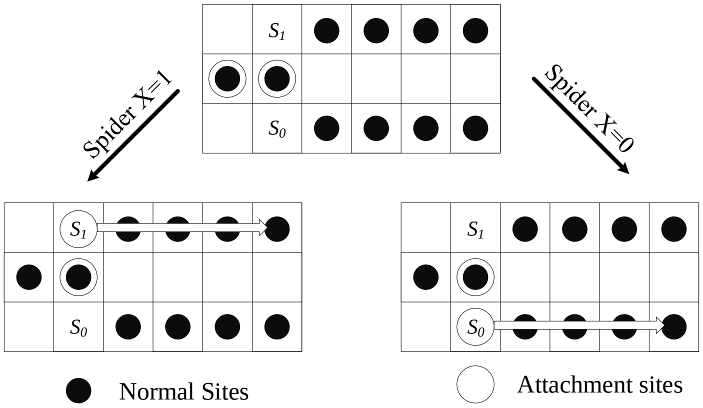
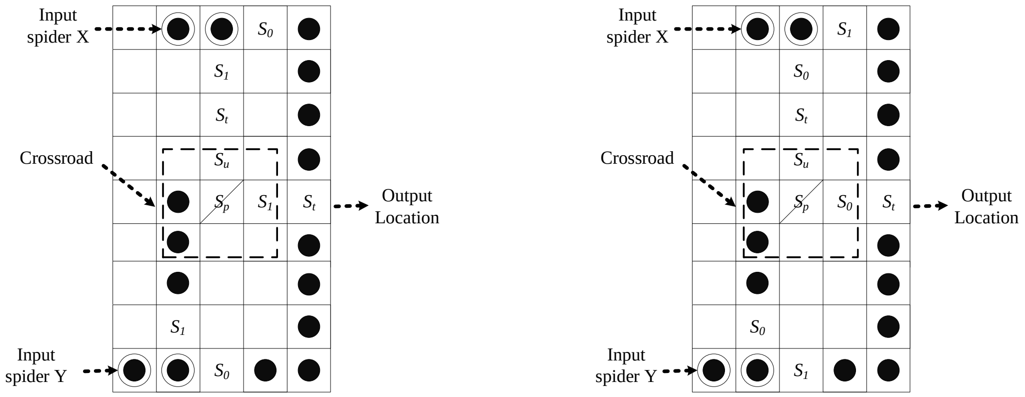


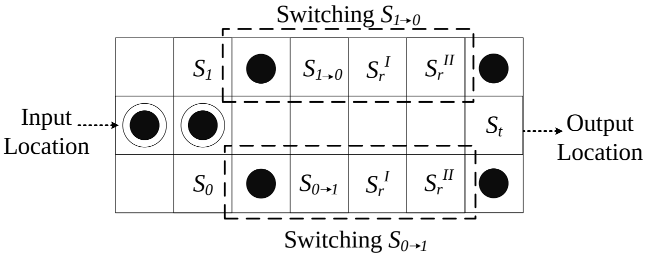
Figure 2.14: The spatial path followed by a molecular spider in normal sites when $X = 0$ and $X = 1$ [2]

site can also send signals to the neighboring sites. These signals may be a “turning on/off” signal which sets the state of the neighboring site as on/off or “switching to 1/0” signal which switches the spider value to 0/1. The design of AND gate and OR gate using the DNA walkers are shown in Fig. 2.15. In this figure, S_t represents a site which can trap a spider binds to it, S_p represents a site which is initially off and blocks the path of the spider until it gets a “turning on” signal, and S_u is a site which will trap the spider binds to it and send a “turning on” signal to the site S_p. The coordinated operation of S_p and S_u in the crossroad guarantee that when the spiders are reaching the junction from both directions, it traps one spider while passing the other. For the AND operation, the output location will receive a spider with value zero, when either spiders X or Y or both are having a value zero. When both the spiders are having a value 1, then the crossroad will come into the picture and it will pass a spider with value 1 to the output location. For the OR operation, a spider with value 1 will reach the output location when either spider X or Y or both are having a value 1. When both are zero, both spiders will reach the crossroad, and will pass only one spider with value zero to the output.



(a) AND Gate

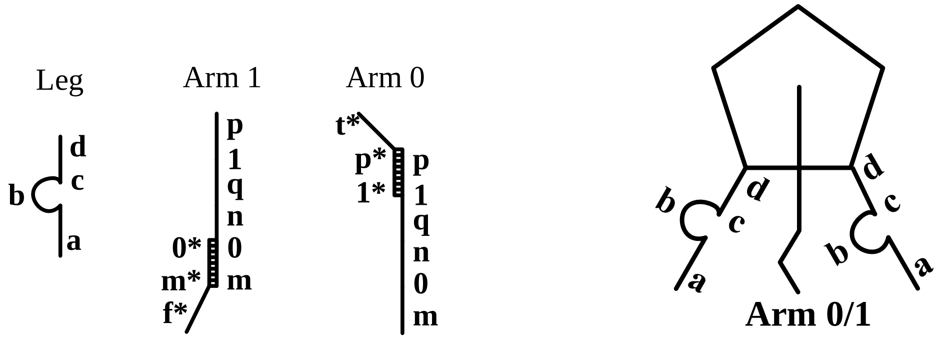
(b) OR Gate



(c) NOT Gate

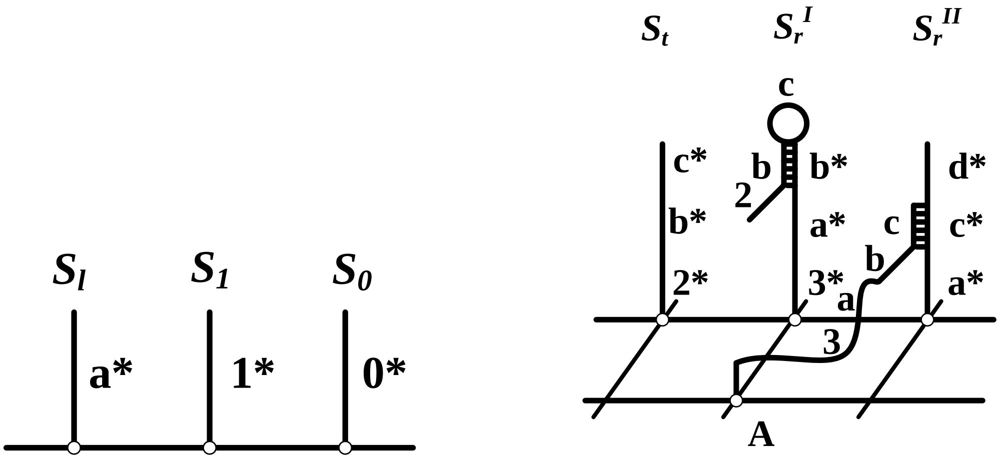
Figure 2.15: DNA walker based logic gates with corresponding molecular reactions [2]

A NOT gate consists of functional sites which perform switching mechanisms. There are two switching mechanisms ($SW_{0\to 1}$ and $SW_{1\to 0}$) associated with a NOT gate. While a spider pass through a switching mechanism, the spider value is switched, and the sites before the switching mechanism are cut off. A switching mechanism consists of three functional sites. The first site is $S_{1\to 0}$ or $S_{0\to 1}$. Consider $S_{1\to 0}$, this trap a spider with value 1, sent a "switch to 0" signal to the next site S_r^I. On the other hand, $S_{0\to 1}$ trap a spider with value 0 and sent a "switch to 1" signal to the next site. The second site is S_r^I which receives a "switch to 0" or "switch to 1" signal from the site $S_{1\to 0}$ or $S_{0\to 1}$ respectively, and send back a "turning off" signal. From S_r^I, the spider can move only to the third site S_r^{II}, since the sites $S_{1\to 0}$ or



(a) Spider Legs

(b) Spider with arm 0/1



(c) Non-alterable Sites

(d) Exit mechanism

Figure 2.16: A possible implementation of molecular spider [2]

$S_{0\rightarrow 1}$ are turned off. Now S_r^{II} will send back a "turning off" signal to S_r^{I}. Now, the spider in the site S_r^{II} cannot move back to S_r^{I} since it is turned off, and hence move towards the output location. Thus, if the input is having a value 1, then the output location contains a spider with value 0, and vice versa.

The scaling up of the circuit is possible by implementing the logic function in the form of a Boolean tree model. Each node in this Boolean tree represents a logic gate (AND, OR or NOT). Each AND and OR gate nodes are connected to two inputs, while a NOT gate

Table 2.1: Comparison of different scalable DNA digital design circuits

Method	**Reusability of Circuit**	**Reusability of Gates**	**Localized Design**	**Fan-out**	**Cross wiring**	**Scaling up Method**	**Speed of operation**	**Simulation**	**Logic value**
Cascades of seesaw gate motifs	No	No	No	Yes	N/A	Dual Rail AND-OR	Low	Available	Concentration of domain
Localized hybridization circuits	No	Yes	Yes	Yes	No	Dual Rail AND-OR	High	Available	Concentration of domain
Spatially localized hairpin circuits	No	Yes	Yes	Yes	Yes	Dual Rail AND-OR	High	Available	Concentration of domain
CRN on surface based circuits	Partially Re-usable	Yes	Yes	Yes	Yes	AND-OR -NOT	High	Not available	Concentration of domain
Molecular spider based circuits	Partially Re-usable	Yes	Yes	No	No	AND-OR -NOT	High	Not available	Arm Value

node is connected to a single input. This Boolean tree implementation will not allow fan-out gate or cross-wiring. There are different DNA implementations possible for the normal and functional sites. A typical implementation of molecular spider, its corresponding non-alterable sites, and possible exit mechanism are shown in Fig. 2.16. Showing all the possible implementations are beyond the scope of this chapter, these implementations can be found in [2].

2.2.5: Comparison of Different Methods

The different scalable DNA digital design techniques differ in their design, re-usability of gates and circuits, the speed of operation, scaling up methods etc. A summary of these differences is given in Table. 2.1.

Speed of Operation

The circuits discussed in this section can be broadly classified as localized and non-localized circuits. In non-localized circuits, the DNA strands are freely floating in the solution and they are free to react with any other strand with an exposed toehold domain. In the case of localized circuits, the DNA strands are attached to a DNA substrate. In localized design, the strands attached to the substrate can react only with its neighboring strands and the free floating fuels, on the other hand, in the non-localized designs, a strand has to search the strand, it has to react within the solution. The diffusion in non-localized design is a stochastic process, and speed of reaction depends on the mobility and the concentrations of the molecules involved. Hence, the speed of operation of a large, complex circuit will be less for non-localized designs compared to the localized designs.

Re-usability of Logic Gates

In localized circuits, the gates are re-usable within the circuit, without causing any spurious reactions. But for non-localized circuits, the strands used in each gate should be unique to avoid the spurious reactions. In localized circuits, the strands that are attached to the DNA substrate can react only with its neighboring strands or free floating fuels. Hence, the chances of having spurious reactions are not present.

Re-usability of Logic Circuits

If the design changes its strands during one operation cannot be used again for the same operation. In such cases, the strands no longer have the same functionality after reacting with the inputs and fuels. The cascade of seesaw gate motifs [6], localized hybridization circuits using DNA hairpins in [45] and spatially localized circuits using DNA hairpins in [7] are not reusable. But CRN on surface based circuits in [1] and molecular spider based circuits in [2] are partially re-usable. In CRN on surface based circuits, the gates are updated by using the excess fuel molecules. For the molecular spider based design, most of the sites are non-alterable and this will make them partially re-usable.

Scaling up Methods

The NOT operation is not possible in the cascade of seesaw gate motifs, localized hybridization circuits using DNA hairpins, and spatially localized circuits using DNA hairpins. Hence, these circuits use a dual-rail AND-OR logic for scaling up of the circuits. The CRN on surface based circuits and molecular spider based circuits use the NOT gate and hence, the dual rail logic is not required in these designs. The fan-out gates are possible in all the designs, except the molecular spider based design. The cross-wiring is not defined for molecular spider based circuits and localized hybridization circuits using hairpins given in [45]. In molecular spider based circuits, the circuit is converted to a binary tree model in which each node represents a gate. Since the fan-out is not available, the inputs may have to be used many times in the circuit.

Boolean Logic Used

The circuits explained in this section also differ in the Boolean logic used. All the circuits except the molecular spider based circuit, use the presence or absence of a particular domain as logic high or low signal. The presence of a domain is determined by finding the concentration of that particular domain using reporter strands. The molecular spider based circuit uses the spider arm value as the Boolean value.

Simulation in Visual DSD

A programming language for representing DNA sequences and DNA reactions was developed in [66, 67]. A simulation software, called “Visual DSD”, could be used to simulate the DNA reactions defined by a programming language [68]. In this software, the simulations are performed as either deterministic or stochastic process. The hairpin structures can be implemented in the Visual DSD. But localization is still a problem. The cascade of seesaw gate motifs can be easily implemented in Visual DSD. In localized hybridization circuit design using DNA hairpins [45], the authors use different toehold domains to ensure localization of the DNA strands while doing the simulation. Lakin *et al.* modified the visual DSD software to include tethered DNA circuits [69] and implemented the AND gate discussed in [7]. Even though Visual DSD can perform the basic operations, implementing strands that have a secondary structure as in the case with CRN on surface based circuits is still a challenge. The localized design for a more complex design is also a major challenge.

2.3: DNA Analog Design

Analog circuits using DSD operation include amplifier circuits, arithmetic circuits, timer circuits, control circuits, etc. The digital circuits use only two discrete values, logic 0 and 1. The physical input is compared with a threshold value, and if the input is greater than the threshold then it is treated as logic high, otherwise, it is treated as a logic low. Digital circuits are less affected by the offset errors. On the other hand, the analog circuits can use the physical inputs without any conversions. There will be a specific range of operation for an analog circuit in which the circuit will work with a particular accuracy and precision. The advantage of analog circuits is that it requires less number of gates compared to its digital equivalent. Hence, analog circuits are preferred to digital circuits, in resource-limited environments such as living cells [70]. Furthermore, some analog computations are less affected by DNA hybridization errors [71]. The applications of analog circuits range from imaging to complex decision making systems. In this section, we are discussing different enzyme-free DNA analog circuits that are available in the literature.

2.3.1: Amplification Circuits

The amplifier circuits are useful in imaging applications and other nano-machines in order to interface with sensors and actuators. They are also useful in case of imaging of strands with small concentration. The amplification circuits can be used as a DNA fuel for free running machines [72, 73]. The first toehold mediated DSD based amplifier was developed by Turberfield and Mitchell [72]. Seelig and co-workers later modified this work to give more than 10 fold gain. Another amplifying circuit, called entropy-driven catalysis (EDC) circuit was developed by Zhang *et. al.* [74]. This amplifier is simple, fast, modular, composable, and robust compared to the previous designs. Another type of amplification circuit based on the hybridization chain reaction was developed by Dirks and Pierce [75]. A DNA hairpin based system using the cross-catalytic circuit for exponential amplification is given by Yin and collaborators [76]. Catalyzed hairpin assembly mechanism is used to produce 20 to 50 fold amplification was proposed by Li and his team [77]. Zhang and Seelig developed a fixed gain amplifier [78], which is a combination of catalytic amplifier and a threshold element. Even though different amplification techniques are available in the literature, the experimental implementations were corrupted by circuit leakage. Chen and co-workers proposed a new technique with minimum leakage and maximum amplification (up to 600,000 fold amplification with two stage four layer cascade) [79]. A detailed review of nucleic acid based amplification reactions for diagnostic applications is given by Jung and Ellington [80].

2.3.2: Arithmetic Circuits

Even though there are many arithmetic circuits using DNA digital logic gates available in the literature [35,81–83], the analog arithmetic circuits are very limited. Song and collaborators recently developed a set of analog gates such as addition, subtraction, and multiplication gates using DNA strands [84]. Instead of Boolean signals, these gates take analog signals as input and produce an analog output. The concentration of the DNA strands is considered as the signal in such circuits. They also computed a polynomial function using these gates. But the operations of these analog gates are limited to positive numbers. Zou *et. al.* modified the existing gates to build addition, subtraction, multiplication and division operations [85]. This design uses a dual rail logic and hence, they can work with positive as well as negative numbers.

2.3.3: Control Circuits

Another major area of DSD based circuits comes under the category of control circuits. Abstract CRNs are widely used for implementing such enzyme-free DNA control circuits. The circuits written using CRN can be implemented using DSD operations [86]. An enzyme-free DNA implementation of the proportional integral (PI) controller using abstract CRN [87] is also available in the literature. Such controllers take a time-varying DNA strand concentration as an input signal and produce an output signal. The four basic operations associated with these linear I/O systems are signal splitting, integration, gain, and summation. A design flow, from transfer function to CRNs, based on the linear I/O system developed by Oishi and collaborators [87] is proposed by Chiu and co-workers [88]. Yordanov and collaborators [89] implemented the PI controller using only catalysis and annihilation reactions. A different DNA implementation of integration feedback circuit is proposed by Birat *et. al.* [90]. A quasi-sliding mode (QSM) feedback controller using unimolecular and bimolecular enzyme-free entropy driven DNA reactions [36] is also available in the literature.

2.3.4: Timer Circuits

A DSD based timer circuit was developed by Fern and co-workers [91]. With the introduction of the timer circuit, it is possible to produce a controllable release of strands which could act as triggers in different circuits. These timer circuits are used in designing logic NOT gate [55].

2.3.5: Challenges for Analog Circuits

Even though different analog circuits are available in the literature, most of the designs are not practically implemented in a wet lab. The major source of error in the cascaded amplifier circuits is the unwanted background amplification. Even though there is no trigger, the output signal produced by the upstream amplifier could be fed back to the downstream amplifier as a signal. The design of DNA strands is very critical, and improper design may lead to leakage reactions. Another source of leakage reaction is the triggering of the DNA circuit by unwanted DNA strands. The leakage reactions are not limited to amplifier circuits, but common to all DSD based circuits. For the control circuits, the major factor in the design is the reaction rate constants. In a practical situation, there are many uncertainties and these reaction rate constants may vary. Control systems with more adaptive properties are needed to overcome this issue. Another major issue is the number of unique DNA strands that are required for the designs. There is a need of alternative CRNs which require a fewer number of DNA strands for implementation. Even though DNA circuits for non-linear controllers such as the sliding mode controller is available in the literature, there is scope for many potential non-linear controllers. The controllers available in the literature are conceptual and their practical implementation is still a challenge. Another issue with the enzyme-free implementation of the controller circuits is that strands used in the circuit may get consumed after some time, and we have to continuously supply these strands for longer computations.

2.4: Conclusion

DNA circuits are considered as a possible candidate for replacing the traditional silicon transistor based technology in implantable medical devices, because of its computational powers, small size, light weight and compatibility with bio-signals. In this chapter, we gave a brief review concentrating on the recent developments in DNA digital design and analog design techniques. For the digital design techniques we considered scalable designs, which could be used to design large, complex digital circuits. We gave a comparison of different techniques with the advantages and disadvantages of each technique based on speed of operation, re-usability of the circuits and strands, availability of simulation software *etc.* For analog circuits, we limited our discussions to amplifying circuits, control circuits, arithmetic circuits and timer circuits. The methods discussed in this chapter were based on enzyme-free operations. The basic principle behind all these DNA circuits is the toehold mediated DSD operation. The DNA circuits will have a pivotal role in designing decision-making system in future therapeutic applications, bio-sensing applications, bio-signal processors, and controllers, driving circuits for molecular actuators, *etc.*

CHAPTER 3: SPATIALLY LOCALIZED MAJORITY GATES

3.1: Introduction

Molecular circuits are considered as a popular candidate for replacing silicon-based technologies for designing implantable medical circuits. With the recent advancements in DNA nanotechnology, medical devices and circuits could be built inside living organisms in a near future. Different molecular Boolean logical circuits are available in the literature, but their input and output types are different [16–19]. The output of one gate cannot be used as the input of another logic gate, and hence, such circuits are not suitable for designing a large, complex design. A DNA circuit using strands, as their inputs and output can overcome this obstacle. The power of DNA, as a computational device was first explored by Adleman, for solving the seven city Hamiltonian path problem [24]. Ever since this famous experiment, many researchers were working on DNA, to solve different problems [26–28].

Lulu Qian and Erik Winfree introduced a DNA strand displacement (DSD) reaction based circuit called seesaw circuit in [5]. In [6], the scaling up of the circuit, using dual-rail AND-OR logic is discussed. Based on this dual-rail AND-OR logic, an arithmetic cell and control cell were designed [35]. The speed of operation of such circuits is limited because the entire reaction is taking place by diffusing all the molecules together. Unique strands are required for signal, and gate complexes, to avoid spurious reactions in a seesaw circuit. In general the seesaw based circuits have the following disadvantages (1) need a large number of unique strands, (2) possibility of spurious reactions, and (3) limited speed of operation. A localized design was introduced to solve these issues [1,7,45,92]. All the enzyme-free scalable DNA digital design techniques available in the literature can be found in [52].

In a localized DNA circuit, the interactions of strands are limited to their neighbors. This helps the designer to reuse the strands without causing spurious reactions. The localized circuits are developed by attaching the DNA strands to a fully addressable lattice or a DNA Origami substrate [45]. The localized architecture speeds up computation in a DNA circuit, by removing much of the speed bottleneck due to diffusion. Even though different designs are available in the literature for a three input majority gate [93, 94], none of them are using a localized architecture. In this work, we are introducing a spatially localized architecture of DNA strands for implementing a three input majority logic operation.

In this work, we are introducing a novel three-input majority gate with spatially localized architecture. This architecture is based on the designs used in [7], for designing AND and OR gates. We are extending this three input majority gate design into a five input majority gate. The addition of a five input majority gate into the set of DNA logic gates family will give more flexibility for the designer while developing the complex digital computing circuits. We also discuss the full adder circuit design using majority gates. A general condition for obtaining n-input majority gate is also discussed.

3.2: Spatially Localized DNA Majority Gate

3.2.1: Methodology

A majority gate gives a logical high output when the majority of the inputs are at logical high level. For a three input majority gate, the truth table is shown in Table 3.1. A, B and C are the inputs and Z is the output. The Boolean expression for the majority gate is derived from the truth table:

$$\begin{aligned} Z &= Maj(A, B, C) \\ &= AB + AC + BC \qquad (3.1) \end{aligned}$$

Table 3.1: Truth table of a three input majority gate

A	B	C	Z
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

It can be observed from the Boolean expression that the majority gate will act as a Boolean AND gate if one of the input is set to zero.

$$Z = Maj(A, B, 0) = AB \tag{3.2}$$

Similarly, by keeping one input at logical high, the majority gate will work as a Boolean OR gate.

$$Z = Maj(A, B, 1) = A + B \tag{3.3}$$

The majority logic is implemented by using DNA hairpin structures attached to a DNA substrate or an Origami box. The origami is a technique developed by Rutherford [46], in which a long single strand is self assembled to form certain shapes using short staple strands. Each staple strand represent a particular address. Origami box is a structure used to hold different Origami pieces. The basic operation associated with this design is the DSD reaction. A simple DSD operation associated with a DNA hairpin is shown in Fig. 3.1.

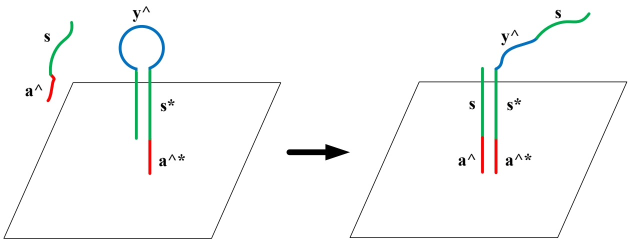


Figure 3.1: An example of DSD operation in a DNA hairpin structure

Here, the domain $a\hat{}s$, is a freely floating domain with a toehold domain $a\hat{}$. The toehold can bind to its complement $a\hat{}*$, which is a part of the hairpin structure attached to the origami substrate. In such a situation, a DSD operation will take place and will result in the opening of the hairpin. The opened hairpin $y\hat{}s$, is not a freely floating domain, consequently, its further reactions are limited to the neighboring strands and other freely floating strands with exposed $y\hat{}*$. Please note that the ˆ symbol is used denote a toehold domain.

In a localized architecture, the majority logic is achieved by arranging the anchored hairpins (input translator, threshold, and output translator) in a way such that the propensity of reaction between an input translator and a threshold hairpin is greater than the propensity of reaction between an input translator and the output translator. The propensity of reaction between any two tethered strands is directly related to the distance between those strands. The propensity of reaction will be more when the strands are anchored in closer proximity to each other. A possible architecture of such a majority gate is shown in Fig. 3.2.

If there is no input (all inputs are low), there is no reaction and hence, no output is produced (output is low). The majority operation with only one input is present (one input is high) is shown in Fig. 3.2. The ˆ symbol is used to identify a toehold domain and * to identify the complement of a particular domain. The input $a\hat{}s$ binds to the input translator

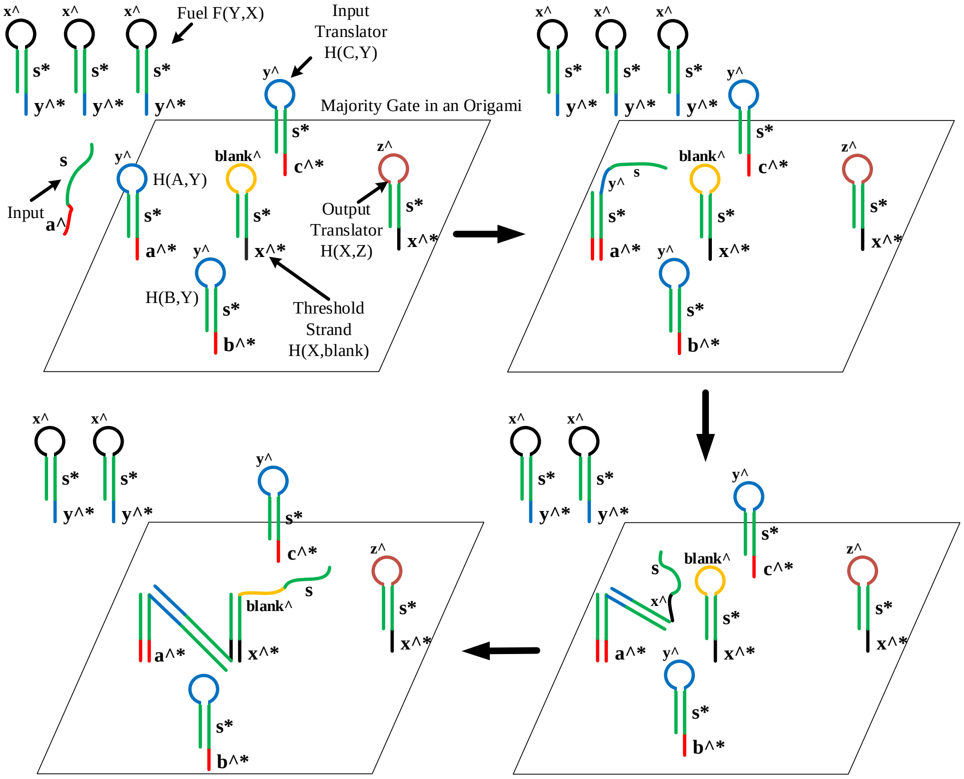


Figure 3.2: DNA hairpin strand operation for localized majority gate with only one input present

$H(A,Y)$ with an exposed $a\hat{}^*$, and displaces the domain $y\hat{}s$. The domain $y\hat{}s$ will bind to the free floating fuel strand $F(Y,X)$, to displace a domain $x\hat{}s$. The domain $x\hat{}s$ can bind to either the threshold $H(X,blank)$ or the output translator $H(X,Z)$ since both of them are having an exposed $x\hat{}^*$. However, the strands are arranged in such a manner, that the proneness of reaction between the input translator and the threshold is greater than the proneness of reaction between input translator and output translator. Hence, the domain $x\hat{}s$ binds to $H(X,blank)$, to displace $blank\hat{}s$, and thus, no output is produced (output is low).

Now, consider the majority operation with two inputs present (2 inputs are high) as

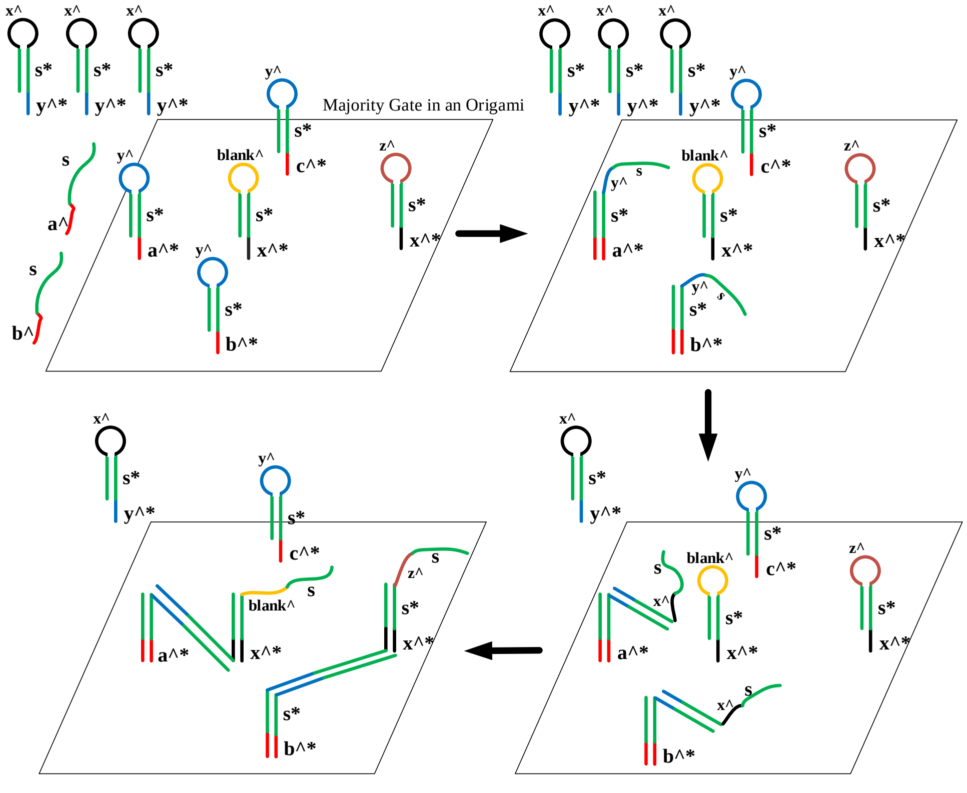


Figure 3.3: DNA hairpin strand operation for localized majority gate with two inputs present

shown in Fig. 3.3. The inputs A and B open the corresponding input translator hairpins to displace $y\hat{\ }s$. These domains bind to the fuel strands $F(Y, X)$, to open $x\hat{\ }s$ in both input translators. As the propensity of reaction between input translator and threshold is greater, the first $x\hat{\ }s$ domain binds to the threshold, to displace $blank\hat{\ }s$. The second input translator, with an open $x\hat{\ }s$ domain, can no longer react with the threshold. Hence, they will bind to the output translator, and displaces $z\hat{\ }s$, which is the output domain (output is high). A similar operation can be observed, when all the three inputs are present (all the inputs are high). The presence of first two inputs is enough to open the output domain $z\hat{\ }s$.

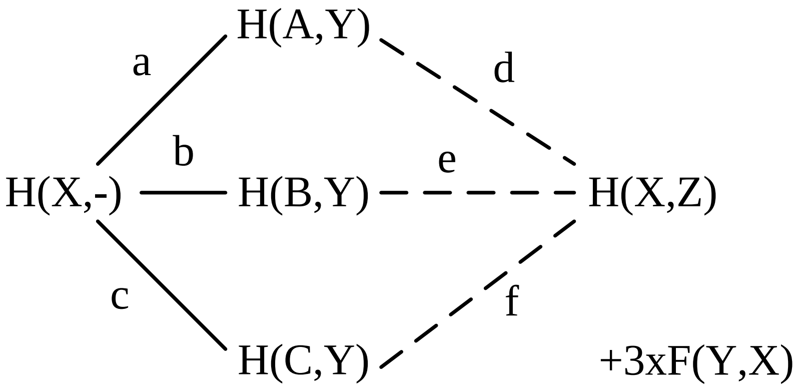


Figure 3.4: Abstract diagram of a three input majority gate with labels showing interaction between hairpins

3.2.2: Simulation Results and Discussion

The proposed majority gate using DNA hairpins is written in a programming language [66], implemented in the Visual DSD software [68], and simulated its operation for different input combinations. An abstract diagram of the proposed three input majority gate is shown in Fig. 3.4. The solid lines represent the possible reaction between the input translator strands and the threshold strand; while the dotted lines represent the possible reaction between the input translator strands and the output translator strand. The letters given above these lines are the tag names, indicating the relative spatial arrangements of the strands.

Each strand is associated with a set of tag names. A reaction is possible only between two neighboring strands, which shares a common tag within an Origami substrate. It could be noted that each input translator strand shares a common tag with the threshold strand and the output translator strand. For the simulation, the local concentration of the tags, connecting the input translator strands and the threshold strands ($lc(a) = lc(b) = lc(c)$) are set to $10 \times 10^5 nM$. Similarly, the local concentration of the tags connecting the input trans-

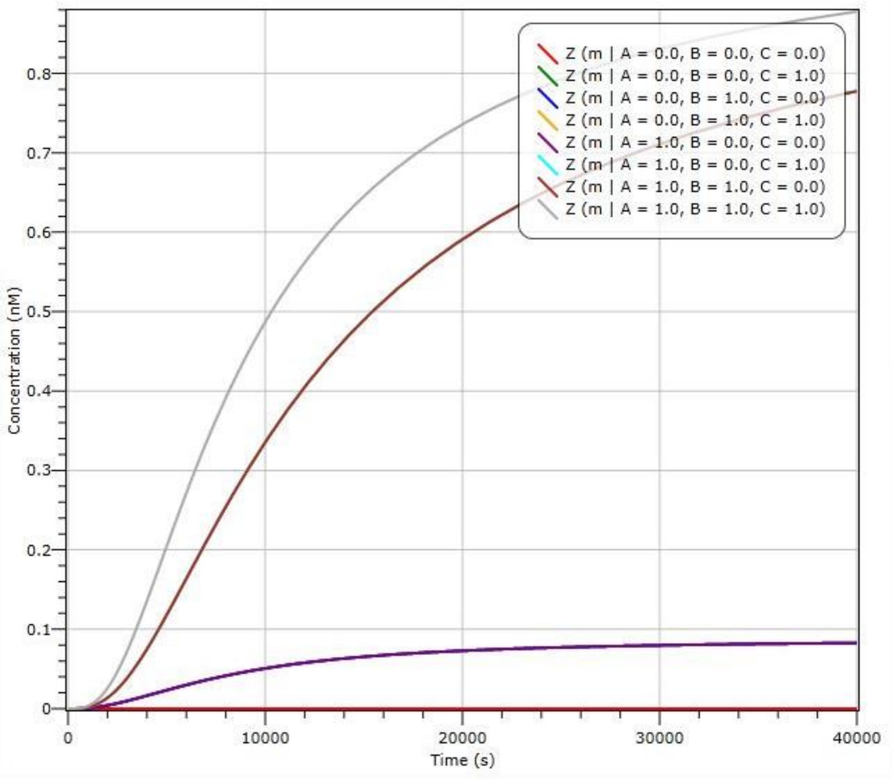


Figure 3.5: Time courses for different input combinations in Visual DSD

lator strands and the output translator strand ($lc(d) = lc(e) = lc(f)$) are set to $1 \times 10^5 nM$. A high local concentration indicates that the strands sharing those tags are physically more close to each other. The propensities of reactions can be calculated by the formula given in [69]:

$$p \triangleq k \times max(lc(a_1), lc(a_2), ...lc(a_n)) \tag{3.4}$$

where k is the rate constant (the default DSD toehold bind rate constant is assumed to be $3 \times 10^{-4} nM^{-1} s^{-1}$), and $a_1, a_2....a_n$ are the common location tags, associated with two reacting strands, for a particular reaction, in an Origami substrate.

For a three input majority gate, the propensity of reaction between each input translator strand and the threshold strand is 300, and that of the input translator strand and

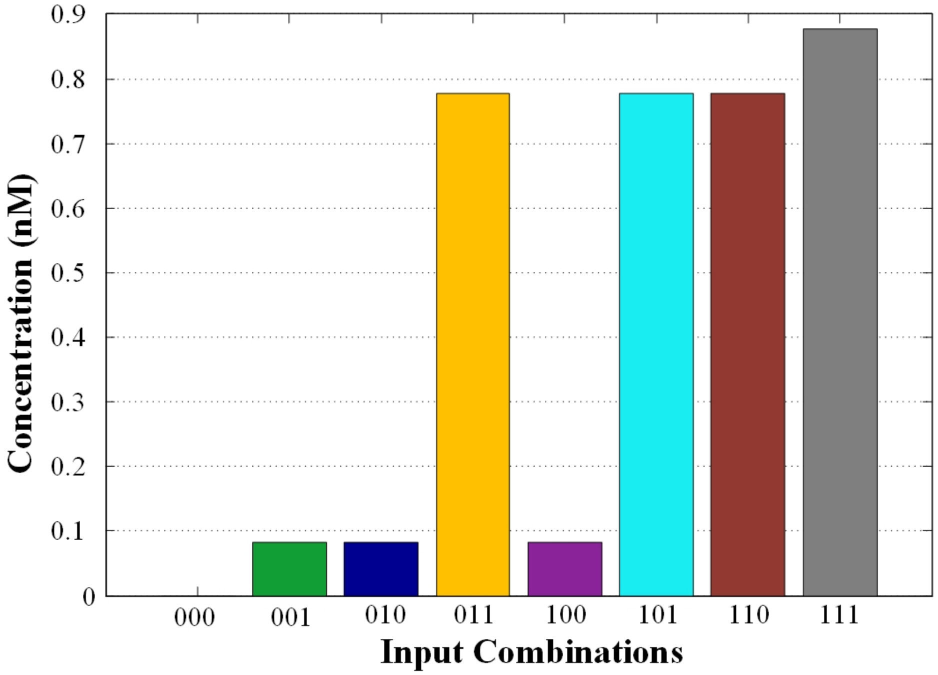


Figure 3.6: The final concentrations obtained for a three input majority gate for different input combinations

output translator strand is 30. Therefore, the input translator strand is physically closer to the threshold. Hence, the probability of a reaction between the input translator strand and threshold strand is greater, compared to the probability of a reaction between the input translator strand and the output translator strand. The visual DSD simulation traces obtained for the three input majority gate with different input combinations are shown in Fig. 3.5. In this graph, different output waveforms are merged together and hence it is difficult to differentiate between them. For more clarity, the final output concentration is given for different input combinations in Fig. 3.6. It is evident that the output is high only when two or more inputs are high. Logical AND and OR gates are also implemented from the three input majority gate by selecting one input as either zero or one. This is also simulated in Visual DSD.

The design of existing DNA three input majority gates can be found in [93,94]. The first design [93] uses different types of input strands and have three AND and one OR operations. The input and outputs are not uniform and hence the scaling up of the circuit is difficult. Also, the number of strands used in the design is more compared to the proposed spatially localized design. In the second design [94], a central circular DNA strand structure is used to make a three input majority gate operation. This design is prone to the scaling up issues such as the requirement of a large number of unique strands to avoid spurious reactions, and limited speed of operation since all the molecules are diffused together. These issues could be solved by using the new localized majority gate proposed in this chapter.

The DSD based circuits cannot produce a NOT operation, hence a dual-rail logic is required. The designs available in the literature for scaling up of circuits concentrate on dual-rail AND-OR circuits [6]. In a dual-rail circuit, one signal is represented by the signal itself and its complement. For example, the signal A can be represented in dual-rail logic as $A_1 = A$ and $A_0 = \overline{A}$. Majority gate is a good candidate for designing dual-rail circuits. The majority gate is having the following property:

$$\overline{M(A,B,C)} = M(\overline{A},\overline{B},\overline{C}) \tag{3.5}$$

In dual-rail logic, (A,B,C) can be represented by (A_1,B_1,C_1), and $(\overline{A},\overline{B},\overline{C})$ by (A_0,B_0,C_0) then:

$$\begin{aligned} M(A,B,C) &= M(A_1,B_1,C_1) \\ \overline{M(A,B,C)} &= M(A_0,B_0,C_0) \end{aligned} \tag{3.6}$$

With the development of localized majority gate, it is possible in the future to design either dual-rail majority logic or a dual-rail AND-OR-majority logic for complex circuits. Thus the availability of majority gate will introduce more flexibility while designing complex circuits.

3.3: Spatially Localized Five Input Majority Gate

The spatial arrangement of hairpin structures can be used to implement different logical operations. The design of AND gate and OR gate using the DNA hairpins can be found in [7]. A majority gate gives a logical high output when the majority of its inputs are logical high. The expression for a three input majority gate is given in eq. 3.1. Similarly, the expression for a five input majority gate is given by:

$$\begin{aligned} M(A,B,C,D,E) = ABC + ABD + ABE + ACD \\ +ACE + ADE + BCD + BCE \\ +BDE + CDE \end{aligned} \tag{3.7}$$

The spatial arrangement of DNA hairpins in a DNA origami substrate for five input majority logic is shown in Fig. 3.7. There are different strands used to design the majority logic operation. These strands can be attached to the origami structure using the staple strands as shown in Fig. 3.7. The functions of these strands are discussed below.

3.3.1: Input Strand

An input strand is a free floating single strand with a domain of the form $a\hat{}s$. We can call such a strand as input A. This strand will bind to its complement ($a\hat{}^*$) which is exposed in an input translator hairpin. Similarly, four other strands are also present in this design. They are $b\hat{}s$, $c\hat{}s$, $d\hat{}s$, and $e\hat{}s$ which correspond to the inputs B, C, D and E, respectively.

3.3.2: Input Translator Hairpin Strand

The input translator is a DNA hairpin with one end anchored to an Origami substrate. The tail end (toehold) of the input translator is exposed and can bind to its complement. For example, an input translator hairpin $H(A,Y)$ is having a domain $a\hat{}^*$ which is exposed to react with its complement $a\hat{}$. Thus, when the input translator hairpin $H(A,Y)$ reacts

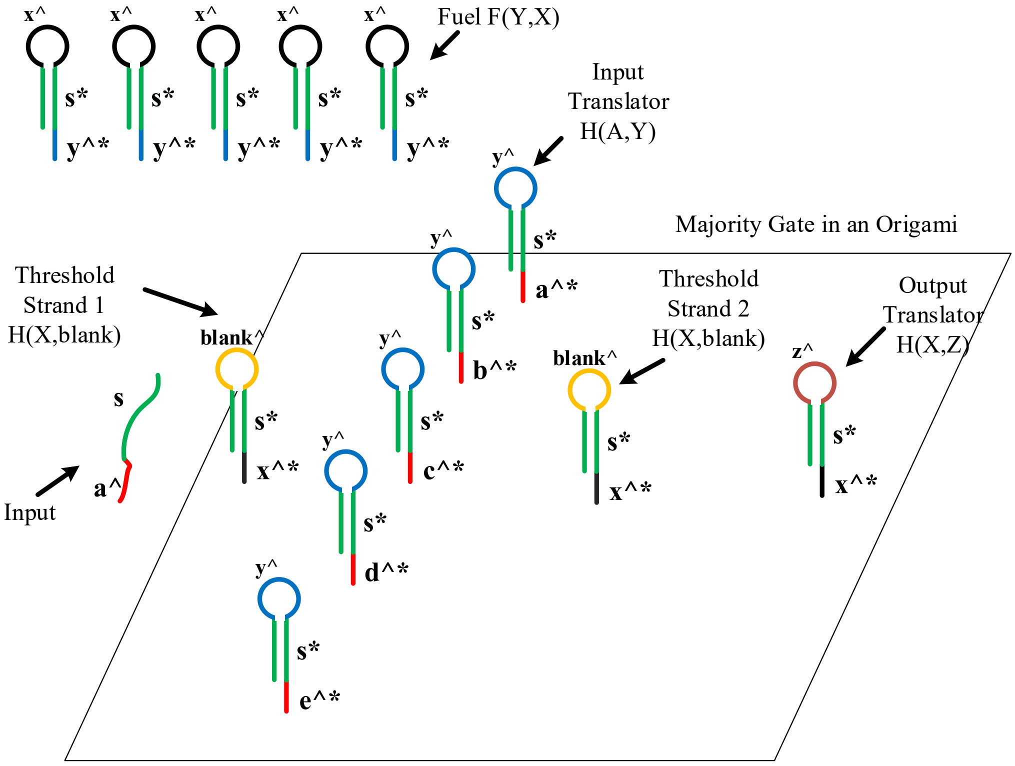


Figure 3.7: The spatial arrangement of DNA strands for a five input majority gate

with an input $a\hat{}s$, it displaces $y\hat{}s$ by a strand displacement process. Other input translator hairpins used in the majority gate design are $H(B,Y)$, $H(C,Y)$, $H(D,Y)$, and $H(E,Y)$ which will bind to inputs $b\hat{}s$, $c\hat{}s$, $d\hat{}s$, and $e\hat{}s$, respectively.

3.3.3: Fuel Strand

Fuel is a free floating hairpin strand and it is used as a connector between two anchored hairpins in the neighborhood. Here, the fuel strand has the form $F(Y,X)$; *i.e.,* the fuel strand will recognize a domain of the form $y\hat{}s$ and releases a domain $x\hat{}s$. The domain $x\hat{}s$ can then bind to a hairpin with an exposed $x\hat{}*$, thus connecting the two anchored hairpins. Five fuel strands, each corresponds to an input translator strand are used for a single five input majority gate.

3.3.4: Threshold Strand

Threshold is a DNA hairpin which is anchored to the Origami substrate. The architecture of a threshold DNA hairpin is similar to an input translator. A threshold strand is represented as $H(X, blank)$. This means that the threshold will bind to a domain $x\hat{\ }s$, and displaces a blank domain $blank\hat{\ }s$ which will not take part in any further reactions. There are two threshold strands used for a five input majority logic operation.

3.3.5: Output Translator Strand

Output translator strand architecture is similar to an input translator strand and threshold strand. The output translator is represented as $H(X, Z)$. *i.e.,* the output translator will recognize a domain $x\hat{\ }s$ and displaces $z\hat{\ }s$ which is the output domain.

Let the distance between the input translator strands and threshold strands are:

$$\begin{aligned} D_{it} = \{&d_{at_1}, d_{bt_1}, d_{ct_1}, d_{dt_1}, d_{et_1}, \\ &d_{at_2}, d_{bt_2}, d_{ct_2}, d_{dt_2}, d_{et_2}\} \end{aligned} \tag{3.8}$$

where D_{it} is a set, which contains the distance (d) between each input translator strand and the threshold strands. The subscripts represent the input translator strand (a, b, c, d, and e) and the threshold number (t_1 and t_2). Similarly, the distance between the input translator strands and output translator strand (D_{io}) is given by:

$$D_{io} = \{d_{ao}, d_{bo}, d_{co}, d_{do}, d_{eo}\} \tag{3.9}$$

For the spatial arrangement of input translators, threshold strands and output translator to act as a five input majority logic, the following condition should be satisfied.

$$\{\forall x \in D_{it} \land \forall y \in D_{io} : x < y\} \tag{3.10}$$

In other words, the distance between the input translator strands and threshold strands must be less than the distance between the input translator strands and the output translator strand.

Suppose, only one input strand ($a\hat{}s$) is present. The input strand will bind to the input translator with exposed $a\hat{}*$, *i.e.,* the input translator strand $H(A, Y)$. This is a strand displacement operation, and it will open the hairpin structure and displaces $y\hat{}s$. This operation is similar to that shown in Fig. 3.1. Further operations of $y\hat{}s$ are limited to its neighboring strands and freely floating fuel strands. The freely floating fuel strands $F(Y, X)$ is having an exposed $y\hat{}*$. Hence, the strand will bind to the fuel strand through a strand displacement operation and open the fuel hairpin by displacing $x\hat{}s$. It could be noted that the strand is still attached to the Origami and further reactions are limited. Now, the $x\hat{}s$ domain can either bind to the $x\hat{}*$ of any one of the threshold strands $H(X, blank)$ or with the output translator strand $H(X, Z)$. But the spatial arrangement is following the condition given in eq. 3.10. The probability of a strand displacement reaction is more, when the strands are close to each other [7]. Consequently, the opened $x\hat{}*$ will binds to one of the threshold strands and displaces the $blank\hat{}s$. No further reaction is possible, and output is not produced.

When the second signal (say, $b\hat{}s$) arrives, it will bind to its corresponding input translator strand ($H(B, Y)$) by opening $y\hat{}s$. This domain will bind to the fuel and open $x\hat{}s$. Since one threshold strand is already occupied by the first input, the $x\hat{}s$ domain will bind to the second threshold strand which is more close to the strand by opening $blank\hat{}s$. Hence, the output is not produced when two inputs are present. If the third input ($c\hat{}s$) is also present, then it will bind to the corresponding input translator strand ($H(C, Y)$) and opens $y\hat{}s$. This $y\hat{}s$ domain will bind to the freely floating fuel strand $H(Y, X)$ and opens $x\hat{}s$.

Since both the threshold strands are already bound to its complements, no threshold strand is available for further reaction. Consequently, the $x\hat{}s$ will bind to the output translator strand to displace $z\hat{}s$. This is the output domain. The presence of the output domain can be determined by using a reporter strand [6]. When more than three inputs are present, they will also opens the $y\hat{}s$ domains from the corresponding input translator strands. But there will not be any further reactions since there are no exposed toeholds in the threshold strands and output translator strands. Thus, the five input majority logic gate architecture produces an output, if there are three or more inputs are present.

The DSD based systems are always affected by some kind of leakage. This may be due to the problems in synthesis such as improper selection of strands, and improper arrangement of strands. Another major issue while using the hairpin strands is that there is a possibility of unavoidable openings of helices. While practically implementing the circuit, there is also chances of strand displacement to happen, even if the input is not present. Another source of leakage is the possibility of input stuck at some place without producing output.

3.4: Simulation Results

3.4.1: Implementation of Five Input Majority Gate in Visual DSD

A language for representing DSD reactions was first developed by A. Phillips and L. Cardelli in [66]. M. R. Lakin *et. al.* developed modified the programming language with higher levels of molecular details [67]. A graphical interface called Visual DSD was developed to implement the DSD programming language [68]. The Visual DSD tool uses stochastic and deterministic models to simulate the DSD program. The concept of localization and origami was later added to the Visual DSD tool [69].

The abstract diagram of the proposed five input majority gate is given in Fig. 3.8.

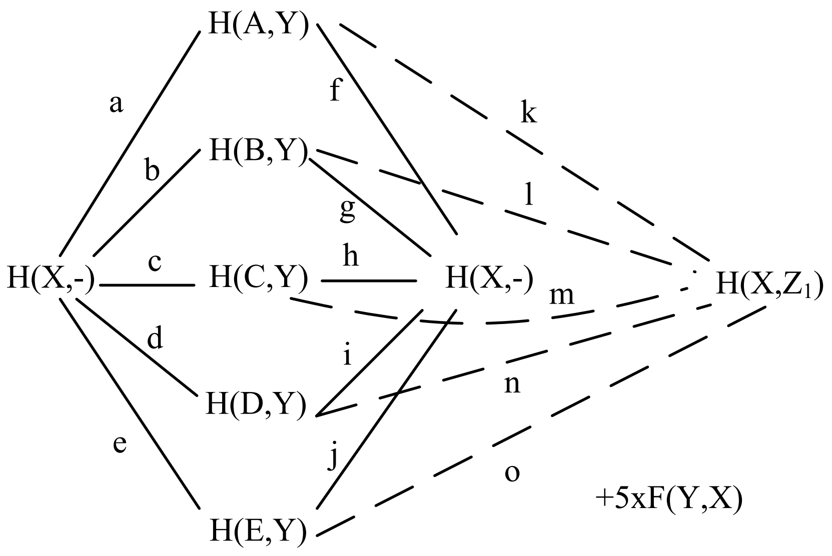


Figure 3.8: Abstract diagram of a five input majority gate

Here, $H(A,Y), H(B,Y), H(C,Y), H(D,Y)$, and $H(E,Y)$ are the input translator strands, $H(X,-)$ is the threshold strand, $H(X,Z_1)$ is the output translator strand, and $F(Y,X)$ is the fuel strand. The lines connecting two strands indicates that there is a possibility of reaction between those two strands. $(a, b, ..., m, n, o)$ are the location tags. While converting this abstract diagram into a programming language, we use the location tags to indicate the possible reaction between two strands. A reaction is possible only between two neighboring strands sharing a common location tag. It could be noted that in our design, each input translator strand shares a location tag with both threshold strands and output translator strands. A set of neighboring strands attached to an Origami are represented by using a keyword "$tether(a_1, a_2.....a_n)$", where $a_1, a_2....a_n$ are the location tags connecting any two strands. For example, the input translator strand $H(A,Y)$ can be represented as "$\{tether(a,f,k)\ a\hat{}^*\}[s]\{y\hat{}\ >$". This means that the input translator strand $H(A,Y)$ can react with either any of the thresholds $H(X,-)$, or with the output translator strand $H(X,Z_1)$.

We know that the input translator strands are physically more close to the threshold strands than to the output translator strands. To implement this spatial localization, the local concentration of the location tags are defined. If the local concentration of a location tag is high, then the strands corresponding to that location tag will be physically more close. We choose the local concentrations:

$$C_{it} = \{lc(a), lc(b), ..., lc(j)\} \tag{3.11}$$

and

$$C_{io} = \{lc(k), lc(l), ..., lc(o)\} \tag{3.12}$$

Where C_{it} is set of local concentrations between input translator strands and threshold strands. Similarly, C_{io} is the set of local concentrations between the input translator strands and output translator strand. *lc(location tag)* represents the local concentration of a particular location tag. For designing a five input majority gate, the concentrations of the location tags are selected in such a way that:

$$\{\forall x \in C_{it} \land \forall y \in C_{io} : x > y\} \tag{3.13}$$

The propensities of reaction between two strands can be calculated by the formula given in [69]:

$$p \triangleq k \times max(lc(a_1), lc(a_2), ...lc(a_n)) \tag{3.14}$$

where k is the DSD toehold bind rate constant and $a_1, a_2...a_n$ are the location tags associated with two reacting strands in an Origami, for a particular reaction. The propensities of reaction calculated will be greater for the reaction between input translator strand and threshold strand compared to that of input translator strand and output translator strand.

The abstract diagram shown in Fig. 3.8 is implemented in Visual DSD tool. We used a “just in time” (JIT) compiler which runs a stochastic simulation for simulating the design.

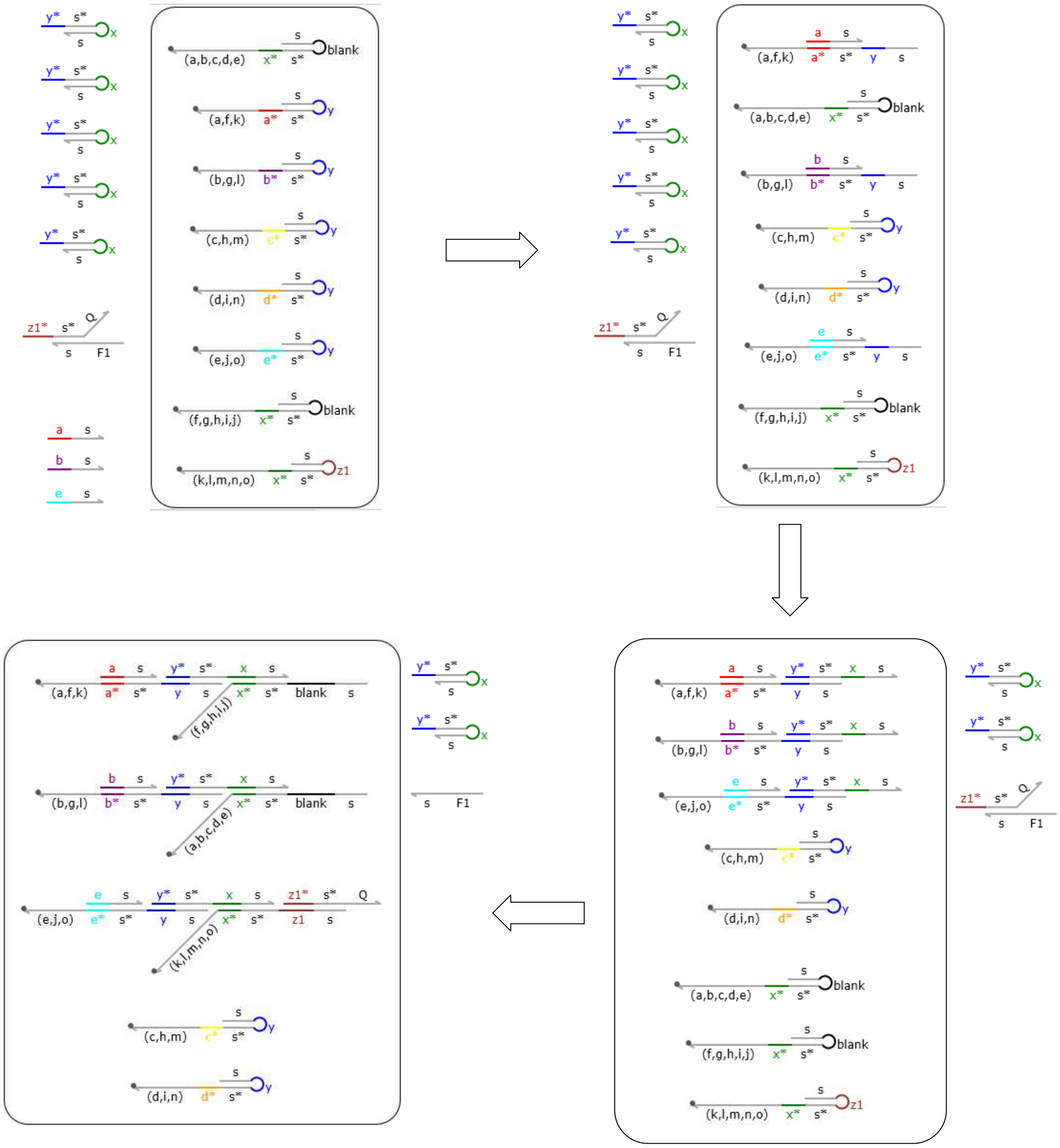


Figure 3.9: The main reactions associated with a five input majority gate for three inputs $(A, B,$ and $E)$

The reactions taking place for three inputs $(A, B,$ and $E)$, is shown in Fig. 3.9. It could be noted that an output domain z_1s is produced, and it reacts with the reporter strand to displace F_1s.

The NOT operation is not possible with this method. Consequently, these logic gates cannot perform as a universal logic gate. To solve this issue, a dual rail logic is proposed [6]. Instead of using AND-OR dual rail logic, an AND-OR-Majority logic could be used for scaling up the circuit. Different functions can be realized using a single five input majority gate [95]. Hence, the addition of a five input majority gate into the DNA digital design will provide greater flexibility for the designer.

3.4.2: Circuit Design Using Majority Gates

Different functions can be realized from a single five input majority gate. Some of these functions are given in Table. 3.2. Many other functions can also be implemented by cascading two majority gates. Even though circuits can be realized using only AND and OR gates, a simple architecture is possible using majority gates in most of the cases. For example, the function $F = W(X + Y + Z) + X(Y + Z) + YZ$ requires 5 two input OR gates, and 3 two input AND gates. The same function can be implemented using only one five input majority gate as given in Table. 3.2. This reduces the complexity of the circuit and increases the speed of operation.

The functions given in Table. 3.2 are not using any complement functions. If any complement functions are present, then the design requires a dual rail majority logic. In a dual rail operation, each input is represented by two bits. One bit is the original signal and the other bit is the complement of the original signal. For example, a single bit input A, can be represented by two bits, $A_0 = \overline{A}$ and $A_1 = A$. A majority gate is having the following property:

$$\overline{M(A, B, C)} = M(\overline{A}, \overline{B}, \overline{C}) \tag{3.15}$$

and for five input majority gate:

$$\overline{M(A, B, C, D, E)} = M(\overline{A}, \overline{B}, \overline{C}, \overline{D}, \overline{E}) \tag{3.16}$$

Table 3.2: Different functions possible from five input majority gate

A	B	C	D	E	**Function**
0	0	0	X	Y	0
1	1	1	X	Y	1
0	0	1	X	Y	XY
0	1	1	X	Y	$X+Y$
0	0	X	Y	Z	XYZ
0	1	X	Y	Z	$XY+YZ+XZ$
1	1	X	Y	Z	$X+Y+Z$
0	W	X	Y	Z	$WX(Y+Z)+YZ(W+X)$
1	W	X	Y	Z	$W(X+Y+Z)+X(Y+Z)+YZ$

For a dual rail operation, eq. (3.15) can be written as:

$$\overline{M(A,B,C)} = M(A_0,B_0,C_0) \tag{3.17}$$

Similarly, eq. (3.16) can be written as:

$$\overline{M(A,B,C,D,E)} = M(A_0,B_0,C_0,D_0,E_0) \tag{3.18}$$

Any circuits can be designed with the help of only majority gates. This is not a good idea always. But there are certain functions, which can be realized using majority gates with less complexity.

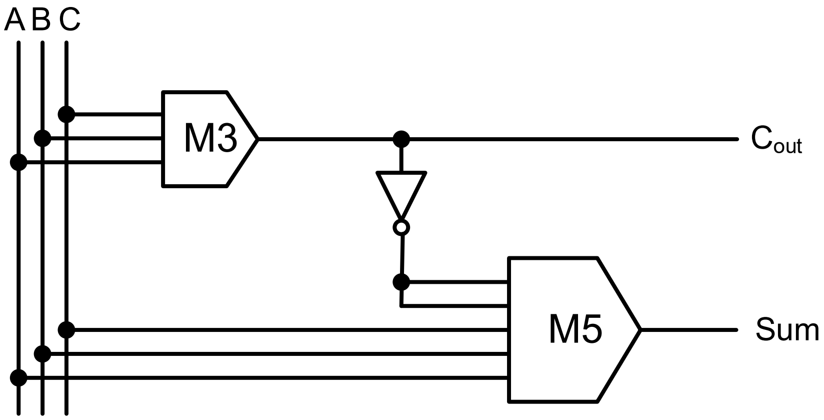


Figure 3.10: Circuit diagram of a full adder using majority gates

3.4.3: Case Study: Full Adder Design

A full adder is a circuit which is used to add three bits at a time. There are two outputs for a full adder; one is Sum and other is C_{out}. If A, B, and C are the inputs, then the outputs are given by:

$$Sum = A \oplus B \oplus C \tag{3.19}$$

$$C_{out} = AB + AC + BC \tag{3.20}$$

The same function can be represented using majority gate as follows:

$$C_{out} = M3(A, B, C) \tag{3.21}$$

$$Sum = M5(\overline{C_{out}}, \overline{C_{out}}, A, B, C) \tag{3.22}$$

Here, $M3$ and $M5$ are three input and five input majority gate respectively. The circuit diagram of a full adder using majority gates is shown in Fig. 3.10. In order to implement the circuit using spatially localized DNA hairpins, the circuit should be converted to a dual

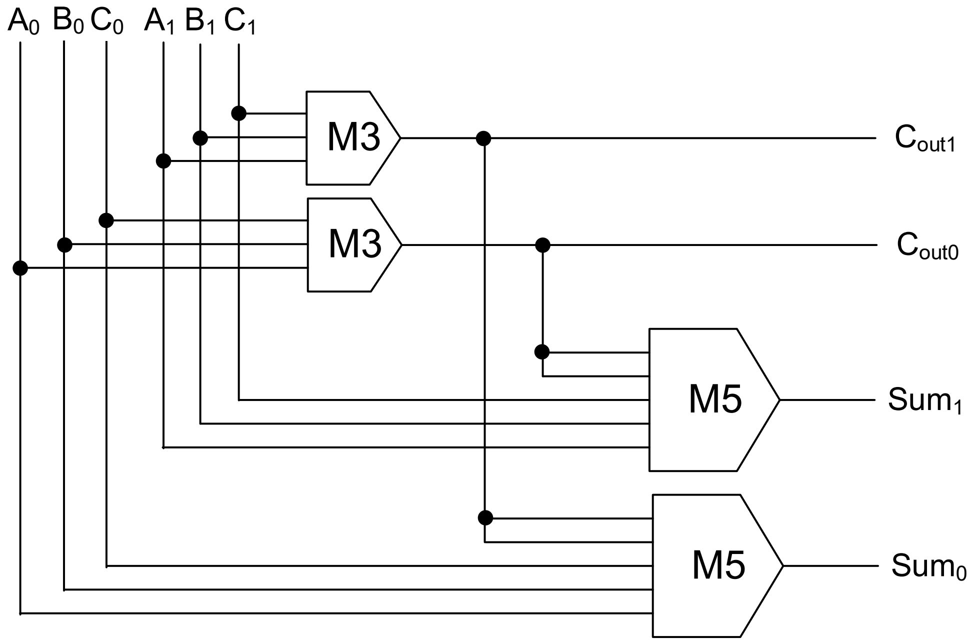


Figure 3.11: Dual rail implementation of full adder using majority gates

rail logic. The function for carry out given in eq. (3.21) can be re-written as:

$$\begin{aligned} C_{out1} &= M3(A_1, B_1, C_1) \\ C_{out0} &= M3(A_0, B_0, C_0) \end{aligned} \tag{3.23}$$

Similarly, the function for sum can be written as:

$$\begin{aligned} Sum_1 &= M5(C_{out0}, C_{out0}, A_1, B_1, C_1) \\ Sum_0 &= M5(C_{out1}, C_{out1}, A_0, B_0, C_0) \end{aligned} \tag{3.24}$$

The dual rail implementation of full adder circuit using majority gates is shown in Fig. 3.11. Now, the circuit can be implemented using spatially localized DNA hairpins. While implementing the circuit, we have to use 2 three input majority gates, 2 five input majority gates, and fan-out gates. The full adder diagram given in [7] uses 9 AND gates, 8 fan-out gates and one OR gates. It also requires some wire crossing. The abstract diagram of the full adder circuit using majority gate is shown in Fig. 3.12. In this circuit, we require 2 three

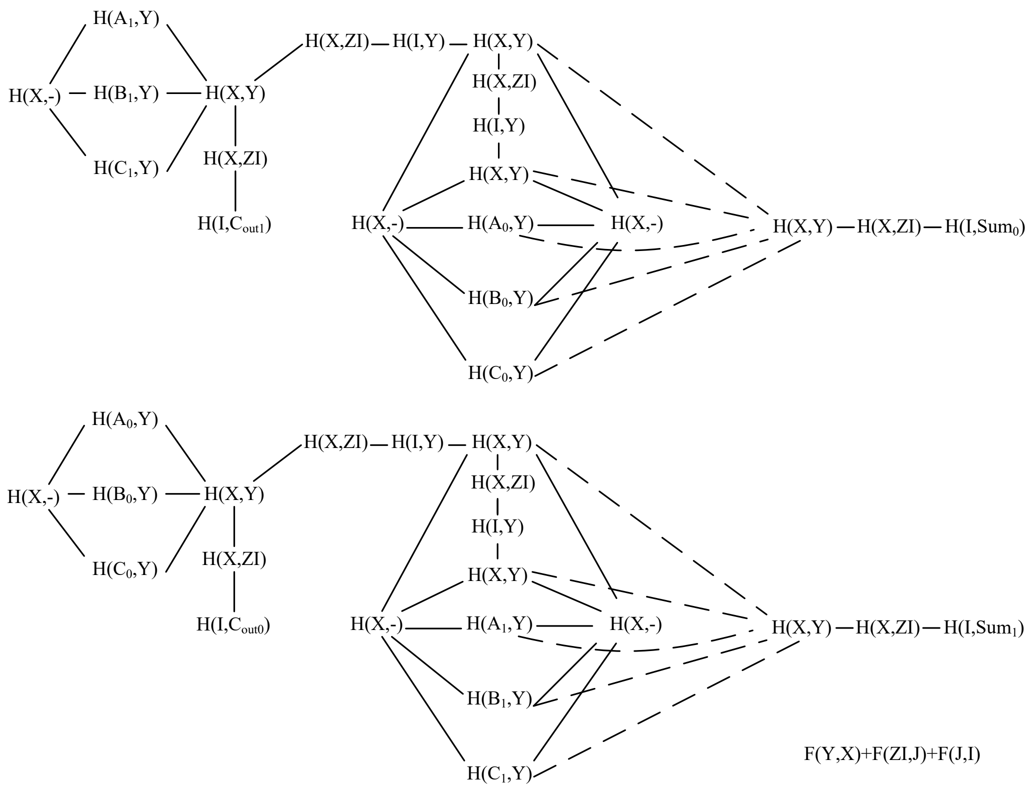


Figure 3.12: Abstract diagram showing strands of full adder circuit using majority gates

input majority gate, 4 fan-out gates, and 2 five input majority gate. This design does not require any additional wire crossing also.

Even though different logical circuits can be designed by using spatially localized DNA hairpins, the experimental validation is still an issue. The complexity increases while simulating large circuits with spatially localized architecture, since the accurate domain lengths and strand spacings are critical to obtain the desired functionalities. The experimental design may also produce some spurious reactions, which may be due to the errors in DNA synthesis, errors in parameter design, etc.

3.4.4: Design of an n-Input Majority Gate

The method used in this chapter can be extended to design any majority gate. For an n-input majority gate, n should be an odd integer greater than one. It requires n input translator strands for its operation. The number of threshold strands (k) required is given by:

$$k = \frac{n-1}{2} \tag{3.25}$$

We can define the set of distances between the input translator strands and threshold strands is given by:

$$\begin{aligned} D_{it} = \{&d_{i_1t_1}, d_{i_2t_1}, ..., d_{i_nt_1}, \\ &d_{i_1t_2}, d_{i_2t_2}, ..., d_{i_nt_2}, \\ &.............................. \\ &d_{i_1t_k}, d_{i_2t_k}, ..., d_{i_nt_k}\} \end{aligned} \tag{3.26}$$

where, $i_1, i_2.....i_n$ represents the input translator strands, $t_1, t_2, ...t_k$ represents the threshold strands, and d is the distance. Similarly, the set of distances between input translator strands and output translator strands can be defined by:

$$D_{io} = \{d_{i_1o}, d_{i_2o}, ..., d_{i_no}\} \tag{3.27}$$

Here, the subscript o is used to indicate the output translator strand. Now, the condition for the spatial arrangement of n-input translator strands, k-threshold strands, and one output translator strands to functions as an n-input majority gate is given by:

$$\{\forall x \in D_{it} \land \forall y \in D_{io} : x < y\} \tag{3.28}$$

Even though the condition for designing an n-input majority gate is simple, the practical implementation is very difficult. The parameters such as domain length and strand spacing are critical while doing simulation and experimental validation. The complexity of the circuit also increases with the increase in the number of inputs.

The design presented in this chapter is a simulation model. The experimental validation is still a challenge. However, Muscat *et al.* were able to implement the translation operation with a three step transmission line. There are many system parameters such as sequence design, physical positioning of the strands, the length of the domains, etc., which will play an important role while experimentally implementing the proposed circuit [7].

3.5: Conclusion

DNA is considered as a possible candidate for designing future implantable medical devices. A novel five input and five input majority logic gates using spatially localized DNA hairpins is proposed in this chapter. The spatial localization helps the designer to reuse the same strands while scaling up the circuit. The architecture of the proposed logic gates is explained by giving different input combinations. The majority circuits are simulated in Visual DSD software. The conversion of the abstract diagram into a programming language is also explained with necessary equations and diagrams. The flexibility that can be provided by the addition of a new five input majority logic gate is also discussed. A full adder circuit is developed using majority gates and also gave a general condition for designing an n-input majority gate using spatially localized DNA hairpin strands. We also discussed the issues for simulation and experimental validation of large complex circuits.

CHAPTER 4: DNA STRAND DISPLACEMENT BASED INVERTER LOGIC GATE

4.1: Introduction

The advancements in nanotechnology are paving the way for building the bio-compatible molecular devices in-vivo or in-vitro. Deoxyribonucleic acid (DNA) is considered as a suitable candidate for building such devices because of the small size, weight, bio-compatibility, and programmability. The computational power of DNA was first explored by Adleman in [24]. Different circuits were developed using the DNA strands since then. All these circuits are working on the basis of the toehold mediated DNA strand displacement (DSD) operation. Similar to silicon transistor in a conventional digital computing device, the DNA strands can be used as the basic building blocks of a DNA computing device.

Enzyme-free DNA logic gates such as AND, OR, and Majority gates are already available in the literature [5–7, 45, 53, 93, 94]. Among these logic gate designs, all of them are not suitable for the design of large complex circuits. A brief review of all the scalable digital DNA designs is given in [52]. Digital circuits made up of DNA strands can be used in nano-machines and devices such as DNA nano-robots [22, 96, 97]. Presently, most of these logic gate circuits lack the logic inverter or NOT operation. Hence, a dual-rail AND-OR or AND-OR-Majority logic is considered for the scaling up of the digital circuits in such DNA circuits. In this research letter, we are introducing a new logic inverter gate or NOT gate design using enzyme-free DSD operations. The use of DNA logic gate inverters in a circuit will reduce the number of unique strands required for that circuit into approximately half. A modular design of DNA subtraction operation is used for the design of the logic inverter gate.

4.2: DNA Inverter Gate Design

The basic operation associated with any enzyme-free DNA circuit is the toehold mediated strand displacement operation. The DNA inverter can be developed from the subtraction gate proposed by Song *et al.* in [84], by treating the second input as a constant. The subtraction operation given in [84] is not having modularity property. Here, the second input itself is acting as the output of the subtraction gate. If a circuit is connected to this subtraction gate directly, then the input may get consumed before the inversion operation, and it may lead to inaccurate results. To make the subtraction operation modular, a switching circuit called gate enable is used. The gate enable can be considered as an electrical relay switch, which passes the signal when a control signal is present. The control signal is generated by the delay circuit [91]. The delay is programmable and hence, this control signal can be used to control the inversion operations in different levels. Such a subtraction gate is shown in Fig. 4.1. The DNA strands in the figure are drawn using Visual DSD software [68].

The standard concentration of the inverter gate is considered as r_{min}. The input (I_{inv}) to the subtraction gate is a single-stranded DNA (ssDNA). The concentration of the DNA strand is considered as the signal. If the concentration is in the range $(0, 0.2 \times r_{inv})$, then the signal can be considered as a logic *low*. Similarly, if the input concentration is in the range $(0.8 \times r_{inv}, r_{inv})$, then the signal can be considered as a logic *high*. The DNA strands used in the logic inverter gate design (Ds and Gs) are similar to that given in [84] for the subtraction gate. The initial concentration of Ds and Gs is set at r_{inv}. The second input in this subtraction operation I_S is kept constant at logic high (r_{inv}) such that $I_{inv} \leq I_S$. The DNA reactions associated with the subtraction operation are reaction 3, 4, and 5 given in Fig. 4.2.

The output of the subtracting gate is the equilibrium concentration of the second input

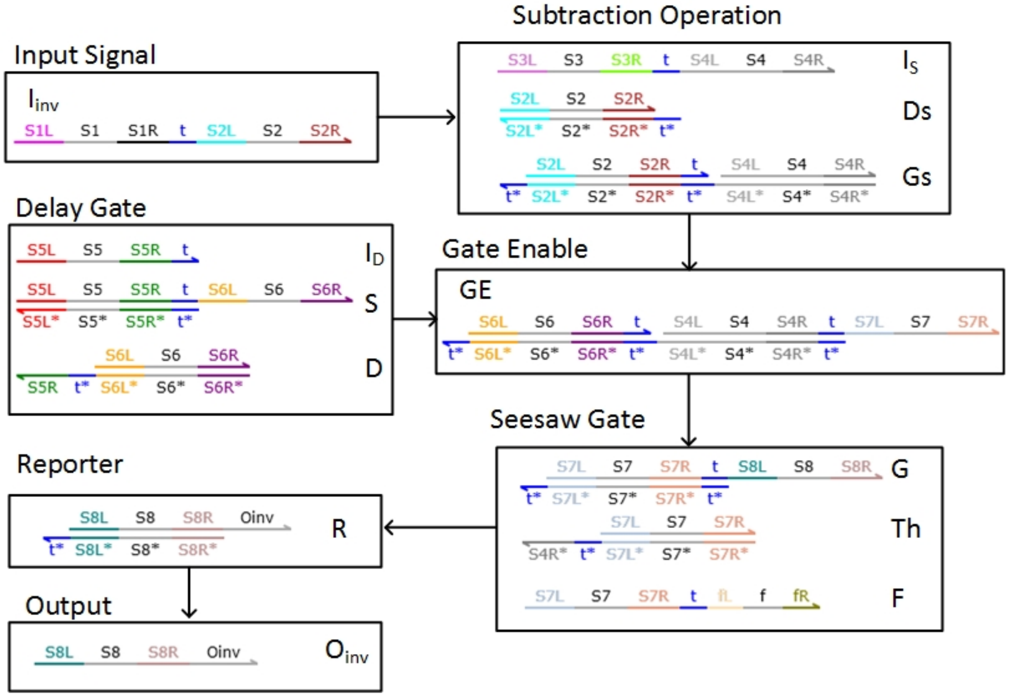


Figure 4.1: Block diagram of the proposed DNA Inverter

I_S. Hence, this subtracting gate is not modular. A delayed signal with a gate enable switch will make the subtraction gate into modular. The delayed signal is produced from a delay gate which uses an initiator (I_D), a source (S), and a delay (D) strand [91]. The DNA reactions associated with the delay gate are reaction 1 and 2 in Fig. 4.2. Reaction 1 is having a very small rate constant. The $sp13$ strand produced from the reaction 1 bind to the D strand to produce two waste strands ($sp23$ and $sp24$). This reaction (reaction 2) is a faster reaction, and the D strand fully gets consumed by the $sp13$ strand. When the concentration of the D strand becomes zero, the $sp13$ signal can react with the GE strand and switch the gate enable to allow further reactions. The concentration of the D strand determines the delay of the signal $sp13$ to reach the GE strand. The delay is chosen in such a way that the subtraction operation is completed and the output of the subtraction operation is at equilibrium. The concentration of the initiator strand (I_D) and source strand (S) are very

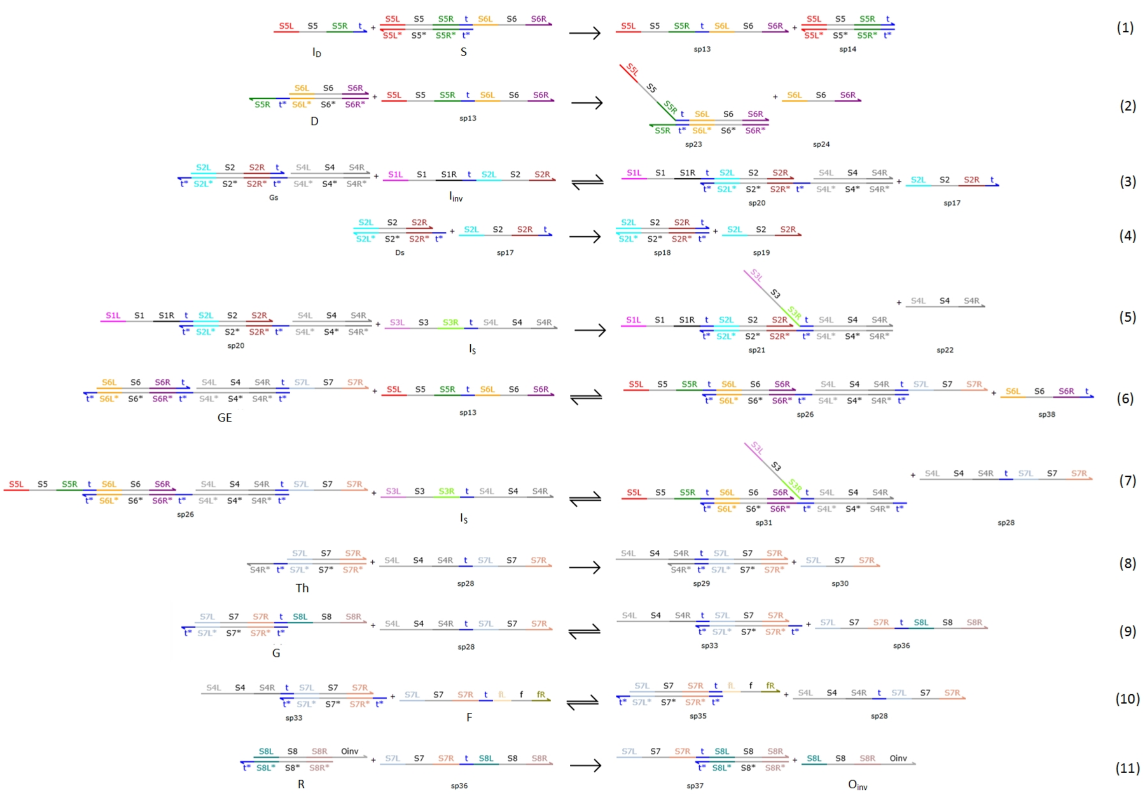


Figure 4.2: The DNA reactions associated with the DNA Inverter

high (approximately 1000 × r_{inv}).

The $sp13$ strand reacts with the GE strand to produce $sp26$ which can further react with the output of the subtraction gate. The GE strand prevents the subtraction gate output from reacting with the next level DNA strands until the subtraction operation is completed. Reaction 6 and 7 in Fig. 4.2 represents the gate enable operation. The initial concentration of GE is also set at r_{inv}. The next level in the inverter design is a seesaw gate motif [5, 6], which consists of a threshold strand (Th), a gate strand (G) and a fuel strand (F). The detailed description of the seesaw gate motif operation can be found in [5, 6]. The concentration of G and F are set to r_{inv}, and that of threshold gate is set to 0.5 × r_{inv}. The seesaw gate operation consists of reaction 8, 9 and 10. The output from the seesaw gate is the strand $sp36$. This strand is further reacted with the reporter strand (R) to produce the

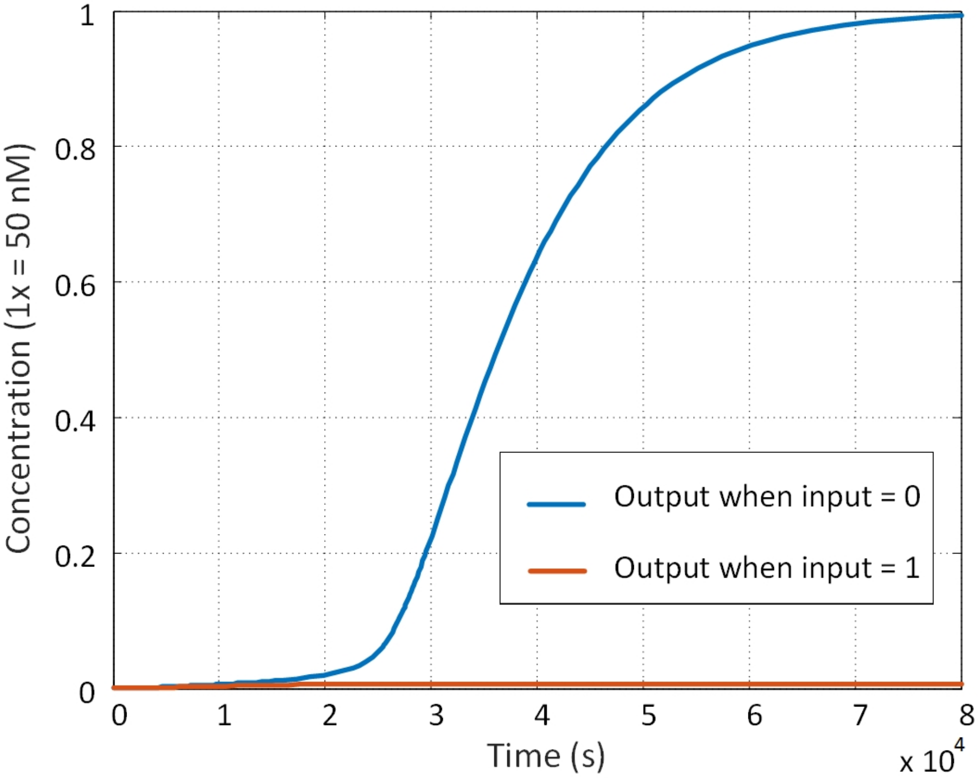


Figure 4.3: Simulation results of the proposed DNA Inverter

output signal O_{inv}. This reaction is given in Fig. 4.2 as reaction 11.

4.3: Simulation Results and Discussion

The proposed DSD based circuits are implemented in visual DSD software using the programming language developed by Phillips and Cardelli [66]. The standard concentration r_{inv} of the design is selected as $50nM$ ($1\times = 50nM$). $0.2\times$ is considered as a logic *low* and $0.8\times$ is considered as logic *high*. The toehold domain (t) is having a toehold dissociation rate constant of $26s^{-1}$ and a toehold binding rate constant of $5 \times 10^{-5} nM^{-1}s^{-1}$. The reaction rate constant of the slow reaction in the delay gate (reaction 1 in Fig. 4.2) is $3.6nM/hour$. The initial concentration of the initiator strand (I_D) and the source strand (S) are set at $1000 \times r_{inv}$ to produce a delay of approximately 1×10^4 seconds for a $30nM$ concentration of the delay

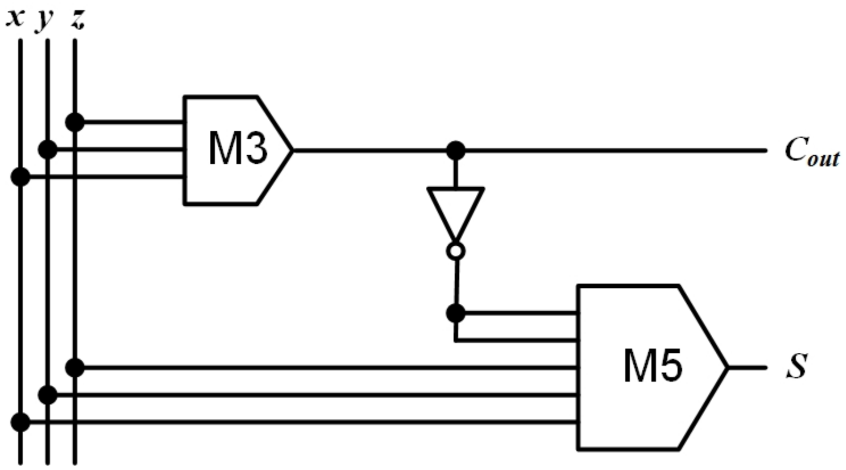


Figure 4.4: Full adder circuit using majority gates and inverter

strand (D). The delay time linearly increases with increase in the initial concentration of the D strand [91]. A MATLAB file is generated from the visual DSD software, and the simulations are performed in MATLAB. The concentration of the inverter output strand (O_{inv}) is observed for input (I_{inv}) logic low ($0.2\times$) and logic high ($0.8\times$) conditions and it is shown in Fig. 4.3.

In order to check the modular property of the proposed inverter gate, we use the logic inverter gate in the design of a full adder. The full adder takes three inputs, x, y, and z and produces the sum (S) and carry (C_{out}) outputs:

$$S = x \oplus y \oplus z \tag{4.1}$$

$$C_{out} = xy + yz + xz \tag{4.2}$$

These functions can be implemented using a three input majority gate (M3), a five input majority gate (M5), and an inverter [98] as shown in Fig. 4.4. The abstract diagram of the full adder seesaw circuit is shown in Fig. 4.5. Here, the three input majority gate is designed by selecting the threshold as $1.4\times$ and five input majority gate by selecting threshold as $2.8\times$. The control signal with a switch in the diagram indicates the gate enable operation.

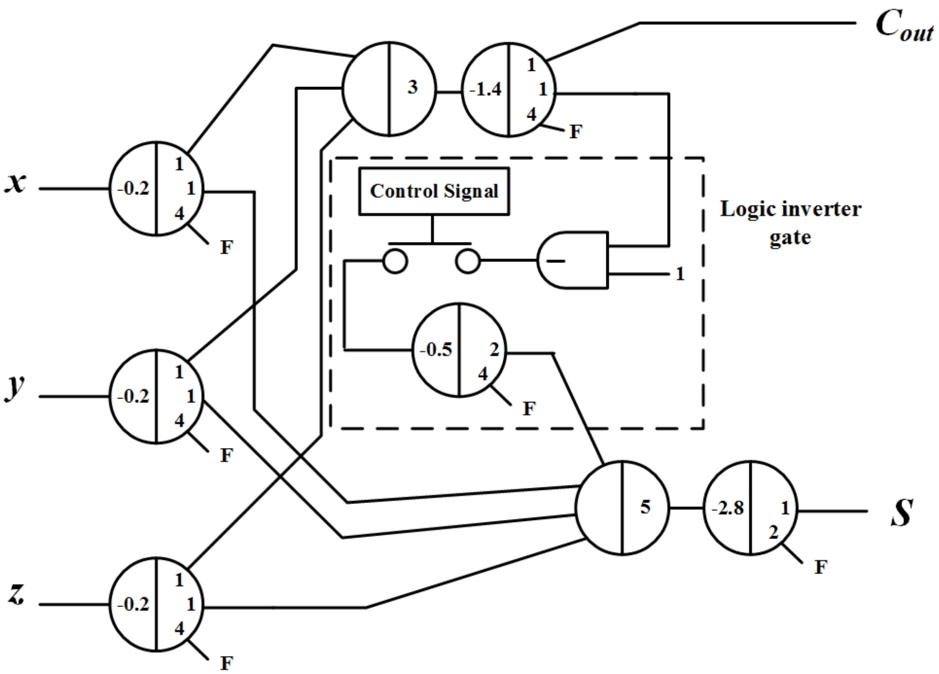


Figure 4.5: Full adder abstract diagram using seesaw gates and inverter gate with gate enable

Subtraction operation is represented by a subtraction gate symbol as given in [84].

The simulation of full adder circuit using majority gates and inverter gate for different input combinations is shown in Fig. 4.6. It can be noted that the C_{out} response is faster compared to the S response. This is due to that fact that, the S output is a function of $\overline{C_{out}}$. We can observe a faster response for S, when $x = 1, y = 1$ and $z = 1$, since the $\overline{C_{out}}$ has no effect in this particular case, and hence, no extra delay in the output.

The modular design approach proposed in this letter is not limited to digital designs. It can also be used with the analog circuits [84] for the modular designs of the subtraction gate. The proposed method can reduce the number of unique strands required for the seesaw DNA digital design into approximately half by removing the dual rail design.

Table 4.1: Abstract diagram for the 16 basic Boolean expressions with two variables

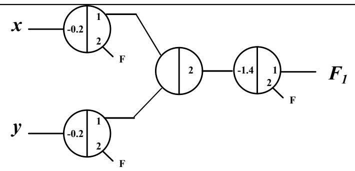
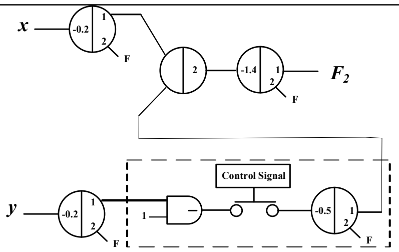
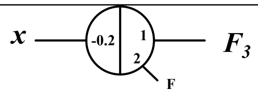
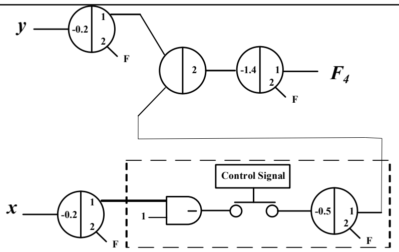
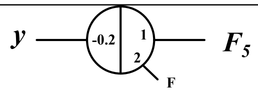
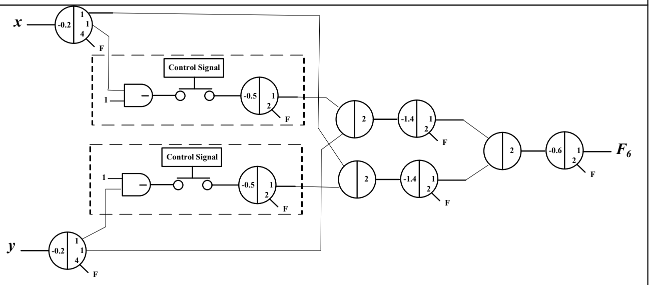
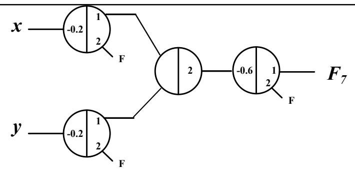
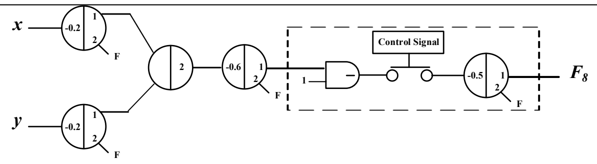
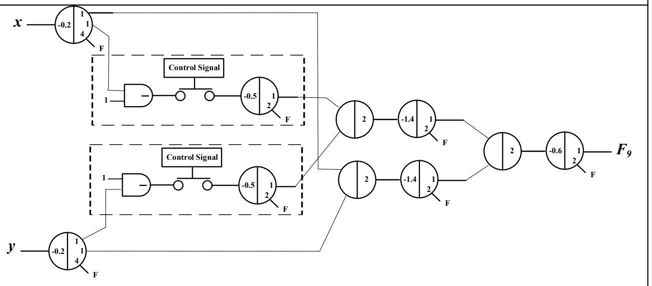
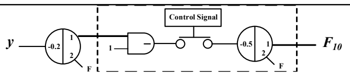
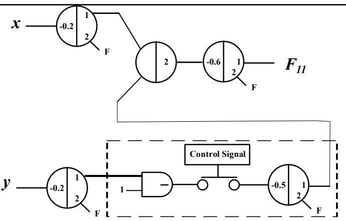
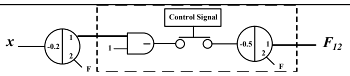
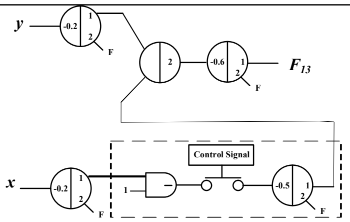
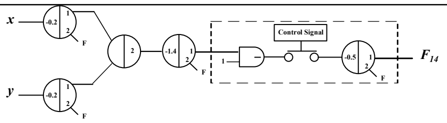
Boolean Function	**Name**	**Abstract diagram**
$F_0 = 0$	Null	
$F_1 = xy$	AND	 x -0.2 1 2 F y -0.2 1 2 F 2 -1.4 1 2 F F_1
$F_2 = xy'$	Inhibition	 x -0.2 1 2 F 2 -1.4 1 2 F F_2 Control Signal y -0.2 1 2 F 1 -0.5 1 2 F
$F_3 = x$	Transfer	
$F_4 = x'y$	Inhibition	 y -0.2 1 2 F 2 -1.4 1 2 F F_4 Control Signal x -0.2 1 2 F 1 -0.5 1 2 F
$F_5 = y$	Transfer	
$F_6 = xy' + x'y$	Exclusive-OR	 x -0.2 1 1 4 F Control Signal 1 -0.5 1 2 F Control Signal 1 -0.5 1 2 F y -0.2 1 1 4 F 2 -1.4 1 2 F 2 -1.4 1 2 F 2 -0.6 1 2 F F_6
$F_7 = x + y$	OR	 x -0.2 1 2 F y -0.2 1 2 F 2 -0.6 1 2 F F_7

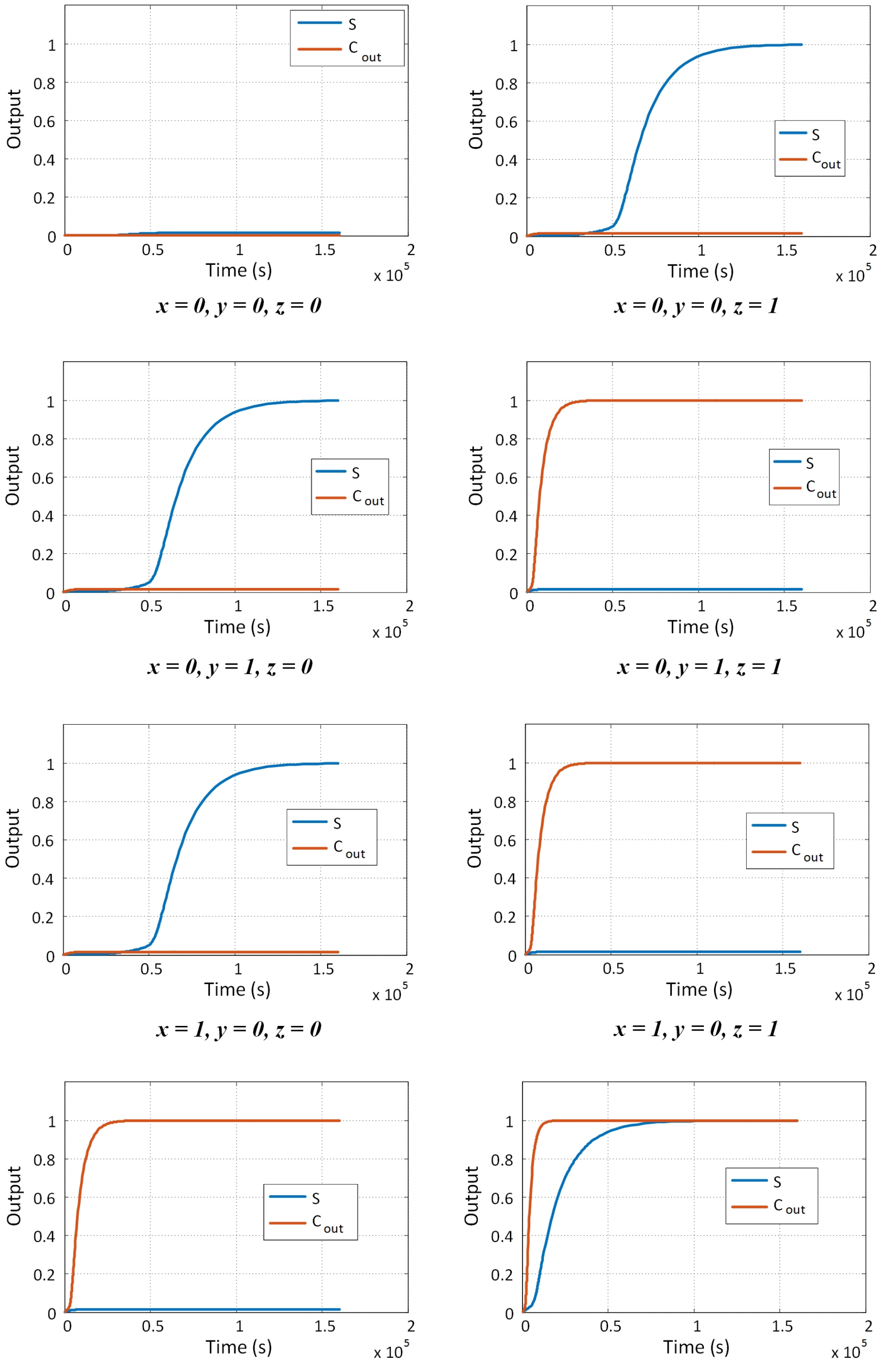
Table 4.1 continued from previous page

$F_8 = (x+y)'$	NOR	
$F_9 = xy + x'y'$	Equivalence	
$F_{10} = y'$	Complement	
$F_{11} = x + y'$	Implication	
$F_{12} = x'$	Complement	
$F_{13} = x' + y$	Implication	
$F_{14} = (xy)'$	NAND	
$F_{15} = 1$	Identity	

The abstract diagram for the 16 basic Boolean expression with two variables are shown in Table. 4.1. Here, we are limiting the designs to two variables. This can be expanded to functions with any number of input variables. The abstract diagrams shown in the Table. 4.1 are not the optimum circuits, but a direct representation of the function using AND, OR, and NOT gates. For larger circuits further simplification is possible by designing the circuits with threshold logic gates, and inverters. Further optimization in the number of strands can be achieved by optimizing the number of NOT gates in the circuit. The design of such a synthesis tool is problem for future research. The reduction in the number of inputs in the circuit is another major advantage of circuits with NOT logic gate. For instance, if we are designing a Boolean function with two variables in dual rail logic, we need four inputs. The conversion of the inputs and output into the dual rail mode is another major burden for the circuit since the inputs are in a mono-rail mode in the practical scenario. Often additional circuitry is required for this conversion.

4.4: Conclusion

A logic inverter gate using DSD operation is proposed in this chapter. A gate enable switch which operates on a control signal from a delay circuit is employed to make the circuit modular. Approximately, 50% reduction in number of DNA strands required for the DNA circuit design can be achieved by using the proposed approach. The logic inverter gate is modular and is capable of using anywhere in the circuit. The modularity property of the proposed logic inverter gate is tested by designing a full adder circuit. The proposed gate-enable operation proposed in this work can be used in analog circuits to give them the modular property.



$x = 0, y = 0, z = 0$

$x = 0, y = 0, z = 1$

$x = 0, y = 1, z = 0$

$x = 0, y = 1, z = 1$

$x = 1, y = 0, z = 0$

$x = 1, y = 0, z = 1$

$x = 1, y = 1, z = 0$

$x = 1, y = 1, z = 1$

Figure 4.6: Full adder simulation results for different input combinations

CHAPTER 5: DNA STRAND DISPLACEMENT BASED FUZZY INFERENCE ENGINE

5.1: Introduction

Conventional computing devices are made up of electronic circuits. Electronic circuits have many notable merits such as high-speed operation, a higher degree of automation, higher precision, higher complexity, etc. The bio-medical devices currently available in the market are also using some kind of electronic circuitry for their operation. There are some other situations where the electronic circuits cannot be used. In such situations, computational systems with biological mechanisms are preferred. Consider a targeted drug delivery system, which searches for biomarkers of cancer and delivers the drug to the particular cell affected by cancer [99]. Here, a bio-compatible device which is purely made up of biological components is required to perform the task. Deoxyribonucleic Acid (DNA) is considered as a suitable candidate for designing nanostructures and circuits for future medical applications. Translators can be used to convert the bio-markers to nucleic acid strands [21]. The structure, size, programmability and bio-compatibility of DNA are the main characteristics which make them a suitable candidate for such designs. The computing power of DNA was first explored by Adleman in his famous seven city Hamiltonian path problem [24]. The Watson-Crick base pairing which exists in nucleic acids, make their operations more predictable, and hence easily programmable to carry out different operations [20]. A survey of recent developments in the field of nucleic acid based devices can be found in [100]. The nucleic acid based computations available in the literature uses either an enzyme based or an enzyme-free platform [60]. In an enzyme-based platform, the operation is driven by enzyme-biocatalyzed reactions [58, 101]. On the other hand, the enzyme-free systems use the base pairing principle for hybridization proposed by Watson and Crick [20] for their operation. Studies are available in the literature which investigates the thermodynamics of DNA motifs [29].

Most of the enzyme-free systems work on the basis of a toehold mediated DNA strand displacement (DSD) operation. These circuits can be broadly classified into digital and analog circuits. Digital circuits which will perform basic logic operations such as AND, OR, Majority, and other functions using these basic logic operations are available in the literature [1, 2, 6, 7, 45, 52, 53, 93]. The analog circuits require fewer resources and are more efficient compared to their digital counterparts in a biological perspective [70]. A fixed gain amplifier and linear classifier circuit using DNA strands were developed by Zhang and Seelig in reference [78]. Analog arithmetic circuits [84] and control circuits [36, 87, 89, 102] which uses DSD operation is also available in the literature. With the recent developments in the implantable medical devices, targeted drug delivery systems, bio-nanorobots etc. [22, 96, 103, 104], the development of control and decision-making systems in the bio-molecular level using bio-compatible materials becomes a necessity. Currently, the decision-making process is primarily carried out using digital logic circuits and some chemical controllers. Analog DNA circuits can provide a more precise real-time control, compared to purely digital control.

The analog computational models could be a complex mathematical model or a rule-based model. The fuzzy logic approach is a popular technique for designing decision-making systems based on human expert knowledge. The human expert knowledge is implemented in a fuzzy system as a set of rules defined in the linguistic form. The biological data is generally noisy and imprecise, hence, fuzzy logic is more suitable for bio-medical applications [105–107]. For example, the exact protein interaction data may be difficult to define mathematically, but it could be easily defined by a set of rules in linguistic form with the help of a human expert. Another example could be a “DNA Doctor” [108], where the system will make some decision on the amount of drug to be delivered based on the concentration of different mRNAs. The fuzzy based models can be either used to predict the output or to control the output.

In this research, we are proposing an analog fan-out gate, minimum gate, and maximum gate using enzyme-free DSD operations. We further used these analog gates to implement the basic fuzzy operations such as fuzzy intersection, and fuzzy union. These fuzzy operators are further used to develop a Mamdani fuzzy inference engine which produces a fuzzy output, from a set of fuzzy inputs, based on a set of predefined rules. As far as our knowledge, no fuzzy inference circuit is available in the literature which uses enzyme-free DSD operations.

5.2: Minimum and Maximum Functions Using DNA

5.2.1: Basic Concepts

There are some basic concepts associated with all the analog circuits such as input range, and valid output range. Input range can be defined as the range in which the inputs of a gate will give results within the required precision. Similarly, the valid output range is the range of the output of the analog gate within which it is considered correct. The performance of the analog gate can be expressed in terms of the time in which the output reaches the valid output range and stays within this range. The DNA strands used in the figures in this chapter are taken from the Visual DSD software [68] simulation. The syntax for representing DNA strands are based on the programming language rules for DSD operated devices [66,67]. In our designs, the concentration of the single stranded DNA (ssDNA) is considered as the signal.

5.2.2: Minimum Gate

A minimum gate computes the minimum between two input signals. Consider the DNA design of the minimum gate given in Fig. 5.1. Here, i_1 and i_2 represents the input strands,

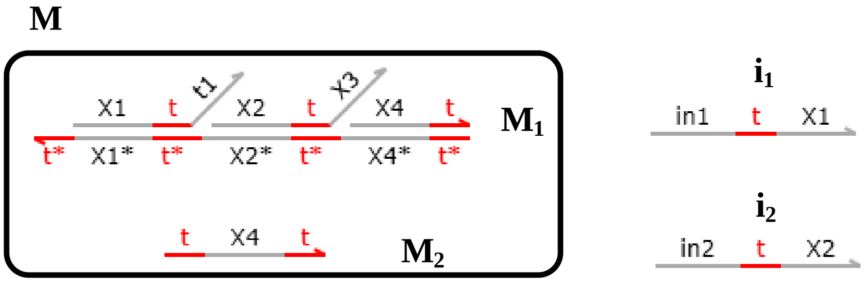


Figure 5.1: DNA design of minimum gate

M represents the minimum gate in which there are two DNA strands M_1 and M_2. The t domains are the toehold and Xi domains are the branch migration domains. The initial concentrations of the input species $[i_1]_0$ and $[i_2]_0$ are the two input signals to the minimum gate.

Consider that the input range of the minimum gate is $(0, r_{min})$. Hence, $[i_1]_0, [i_2]_0 \in (0, r_{min})$. The initial concentration of the chemical species M is set to r_{min}. *i.e.,* $[M_1]_0 = [M_2]_0 = r_{min}$. The output of the minimum gate is O_{min} which is the strand $< X2\ t^{\wedge}\ X3 >$ ($5'$ to $3'$ direction) in the figure. The $^{\wedge}$ symbol is used to differentiate the toehold domain from the branch migration domains [67]. The concentration of O_{min} at equilibrium will give the minimum of $[i_1]_0$ and $[i_2]_0$. Hence,

$$[O_{min}]_\infty = min([i_1]_0, [i_2]_0) \tag{5.1}$$

where, $[O_{min}]_\infty$ and $min()$ represents the concentration of the output strand at equilibrium and minimum function, respectively.

The DNA reaction diagram of the minimum gate is shown in Fig. 5.2. Each reaction in this diagram is given with a number. The DNA strands inside the bold boxes are the reactants initially used (which are available in Fig. 5.1). The DNA strands at the end of the

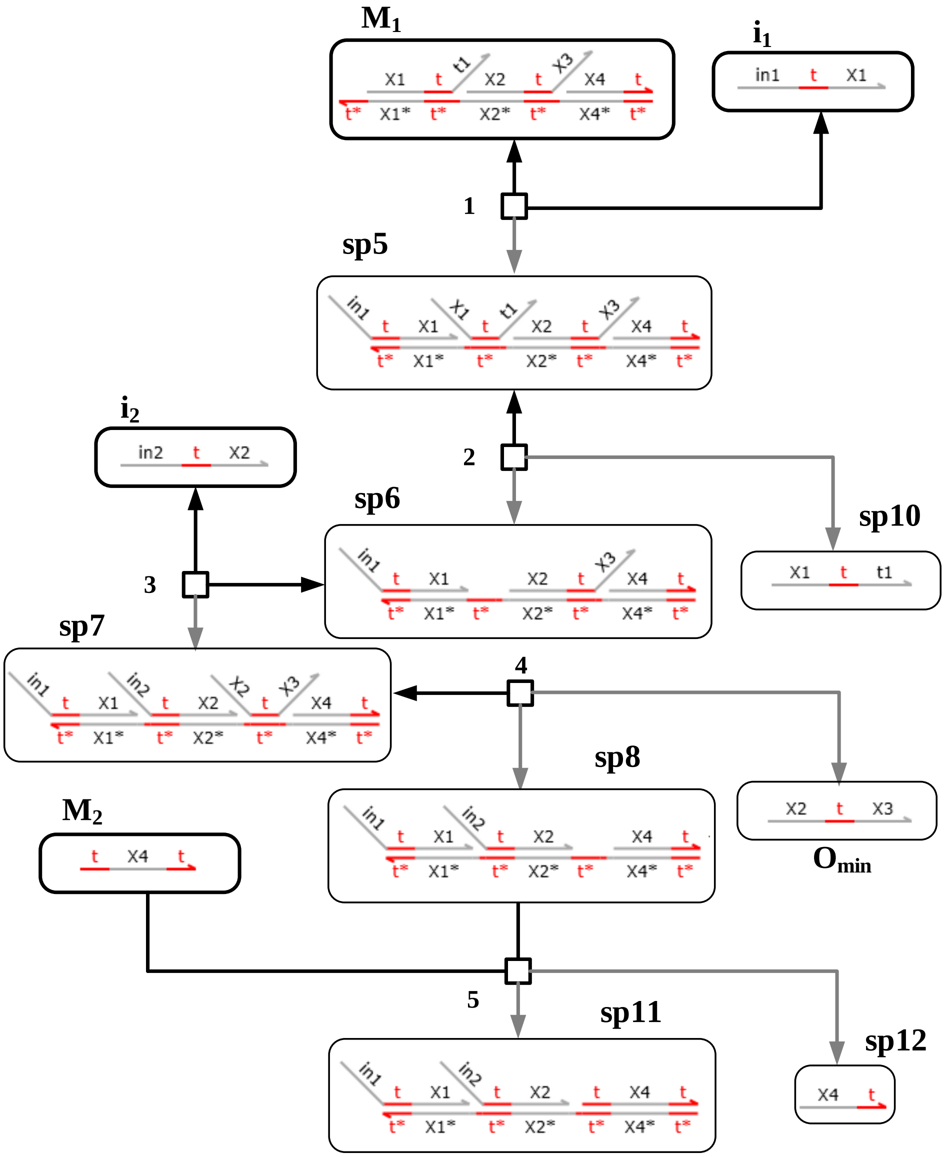


Figure 5.2: DNA reaction diagrams in the minimum gate

gray line with arrows indicate the products of forward reaction while those at the end of the black arrow indicate the products of the backward reaction. The DNA strands/complexes with edges without arrow represents the reactants.

Ideally, i_1 will bind to M_1 to produce $sp6$ and $sp10$ as shown in reaction 1 and 2 in Fig. 5.2. If i_2 is not present, then the whole i_1 will be consumed to produce $sp6$ and $sp10$. The equilibrium concentration of $sp6$ will be equal to $[i_1]_0$ in this case. If i_2 is not present then there will not be any further reactions and no O_{min} is produced.

Now consider $[i_1]_0 \geq [i_2]_0$. In this case also, the reaction 1 and 2 will occur and produce $sp6$ and $sp10$. The input i_2 will be consumed by the $sp6$ to produce $sp8$ and O_{min}. The $sp8$ will further react with $M2$ to produce $sp11$ and $sp12$ by preventing the backward reaction. Therefore, the concentration of O_{min} at equilibrium will be equal to the initial concentration of i_2. *i.e.,* $[O_{min}]_\infty = min([i_1]_0, [i_2]_0) = [i_2]_0$. If $[i_1]_0 < [i_2]_0$, the $sp6$ concentration will not exceed $[i_1]_0$, hence, the whole of the i_2 will not react with $sp6$. The equilibrium concentration of O_{min} in this case will be equal to $[i_1]_0$. *i.e.,* $[O_{min}]_\infty = min([i_1]_0, [i_2]_0) = [i_1]_0$.

Minimum Gate as Subtraction Gate

It is interesting to note that the minimum gate can act as a subtraction gate when i_2 is considered as the output. Consider that $[i_1]_0 < [i_2]_0$, in this case, the whole i_1 will react with M_1 to produce $sp6$ whose equilibrium concentration will be equal to $[i_1]_0$. The i_2 will bind to $sp6$ such that there $[sp6]_\infty$ is zero. Therefore, the equilibrium concentration of i_2 will be $[i_2]_\infty = [i_2]_0 - [i_1]_0$. The subtraction operation is valid only when $[i_1]_0 \leq [i_2]_0$. If $[i_1]_0 > [i_2]_0$, then i_2 will fully react with $sp6$ and $[i_2]_\infty$ will be zero in that case.

5.2.3: Fan-out Gate

A fan-out gate produces multiple output signals from a single input. The DNA implementation of a fan-out gate which gives two outputs is shown in Fig. 5.3. The input strand (i_1) and output strands (O_{f1} and O_{f2}) are ssDNAs. Here, F is the fan-out gate which contains the DNA strands F_1 and F_2. The initial concentration of input strand $[i_1]_0$ is considered as the input signal for the fan-out gate. The input range of the fan-out gate is $(0, r_f)$. The initial concentration of the fan-out gate is set to r_f. Thus, $[F_1]_0 = [F_2]_0 = r_f$. The outputs

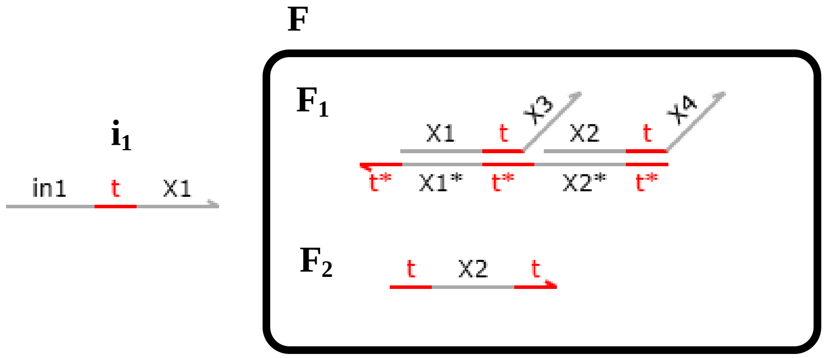


Figure 5.3: DNA design of fan-out gate

of the fan-out gate are the concentration of O_{f1} and O_{f2}. Under equilibrium,

$$[O_{f1}]_\infty = [O_{f2}]_\infty = [i_1]_0 \tag{5.2}$$

The DNA reaction diagram of the fan-out gate is shown in Fig. 5.4. Each reaction in the figure is assigned a number. The initially available DNA strands F_1, F_2, and i_1 are given in bold boxes. The input DNA strand i_1 first reacts with F_1 to produce $sp7$ and an output strand O_{f1}. Now, $sp7$ will further react with F_2 to produce $sp8$ and the second output O_{f2}. The $sp8$ thus produced does not have any exposed toehold and consequently, no further reactions will occur. Since the initial concentration of F_1 and F_2 are greater than the initial concentration of i_1, the whole of the i_1 will be consumed to produce O_{f1} and O_{f2}. Here, the fan-out gate is producing two outputs and hence the gate is said to have a fan-out of two. The same method can be extended to produce a fan-out of 3 by using F_1, F_2 and F_3 as shown in Fig. 5.5. The outputs will be $< X1\ t^\wedge\ X4 >, < X2\ t^\wedge\ X5 >$, and $< X3\ t^\wedge\ X6 >$. In a similar way, this method can be extended to produce a fan-out of n by using n DNA strands $(F_1, F_2, ...F_n)$ in the fan-out gate, where, F_1 is given by $\{t^{\wedge *}\}[X_1\ t^\wedge] < X_{n+1} >: [X_2\ t^\wedge] < X_{n+2} >: ....[X_n\ t^\wedge] < X_{2n} >$, for $2 \leq i < n$, F_i is given by

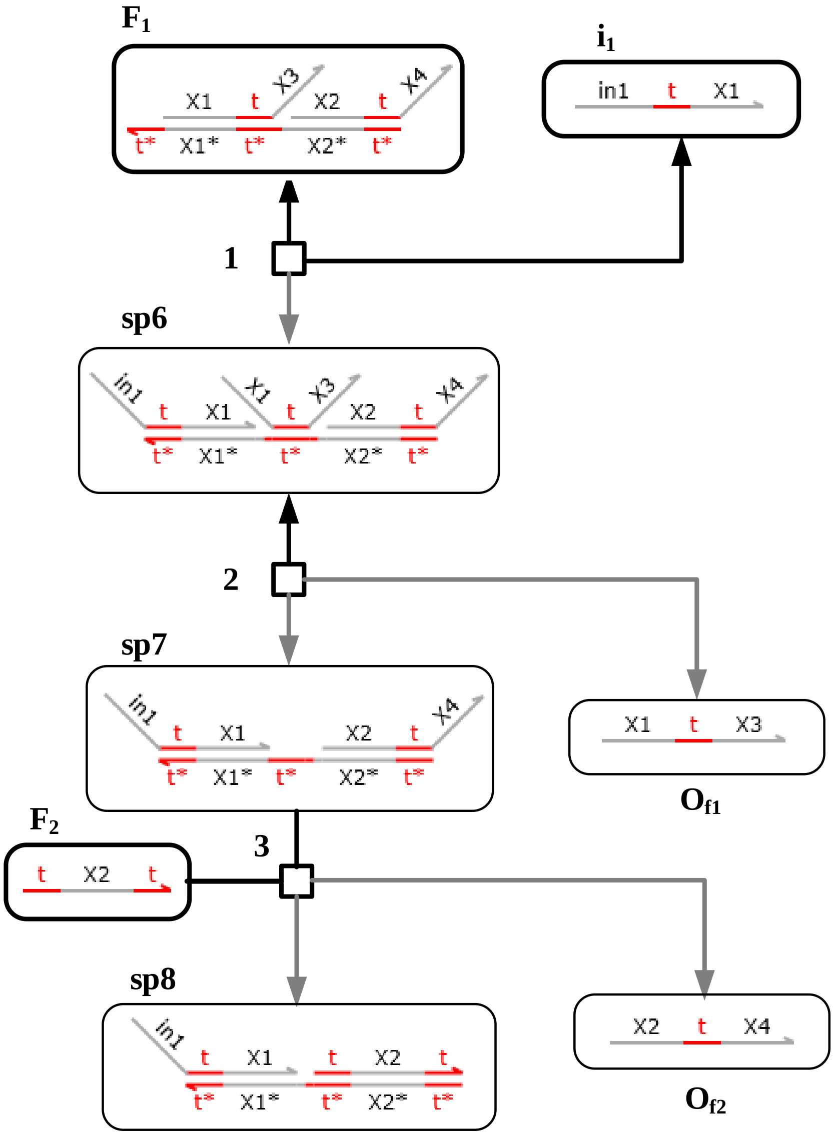


Figure 5.4: DNA reaction diagrams in the fan-out gate

$< t^{\wedge}\ X_i >$ and F_n is of the form $< t^{\wedge}\ X_n\ t^{\wedge} >$.

5.2.4: Maximum Gate

The maximum gate will provide the maximum of the inputs applied to the gate as output. The maximum of two inputs A and B can be given by:

$$max(A, B) = (A + B) - min(A, B) \tag{5.3}$$

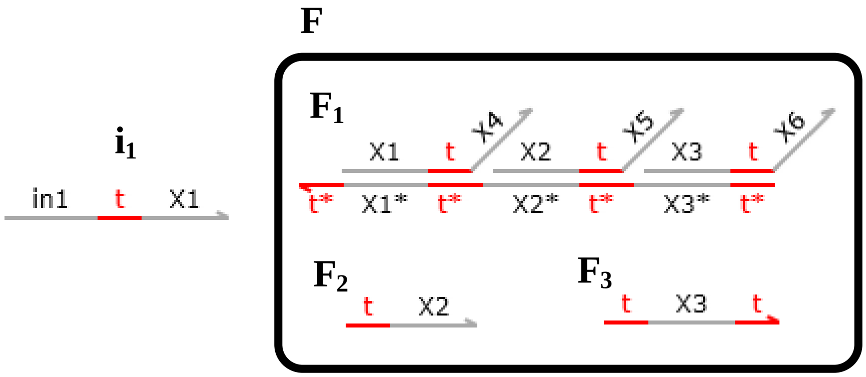


Figure 5.5: DNA design of fan-out gate with a fan-out of 3

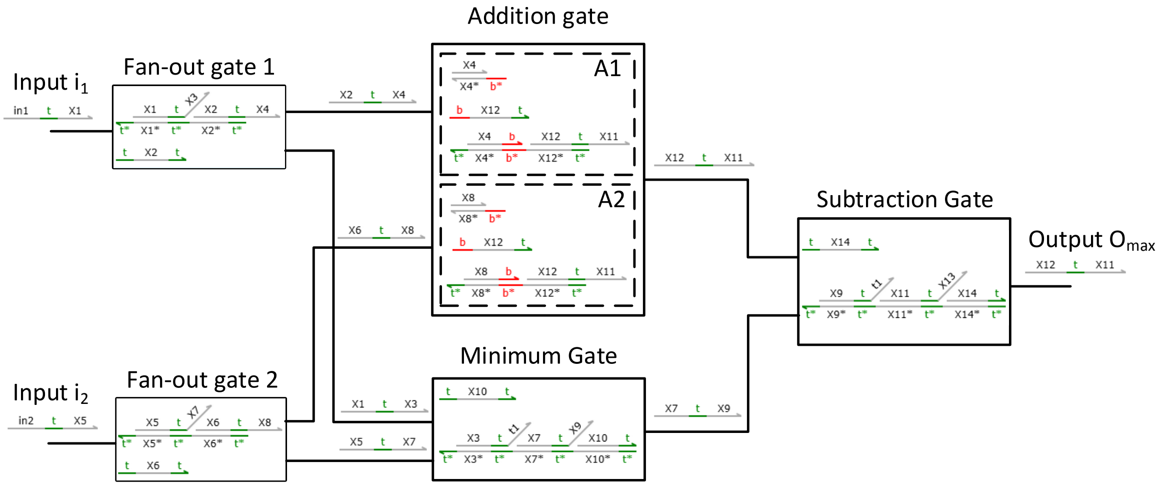


Figure 5.6: Block diagram of the proposed maximum gate

The maximum gate based on DSD operation consists of two fan-out gates, one minimum gate, one adder gate, and one subtraction gate. The block diagram of the proposed maximum gate is shown in Fig. 5.6. The minimum gate and the fan-out gates proposed in this chapter are having a modular design, *i.e,* the output of one gate can be used as an input to another gate. However, the subtraction gate does not have the modular property. For this reason, the maximum gate should be made in such a way that the subtraction operation should be the last operation in the circuit. The initial concentration of the fan-out gates and the minimum gate is set at $r_f = r_{min} = r$, and the initial concentration of adder r_a should be greater than or equal to $2 \times r$. Similarly, the initial concentration of the subtraction gate in the circuit should also be greater than or equal to $2 \times r$.

The inputs i_1 and i_2 are given to the fan-out gate 1 and 2, respectively. The fan-out gates used in this design are having a fan-out of two. The fan-out gate 1 outputs $< X2\ t^\wedge\ X4 >$ and $< X1\ t^\wedge\ X3 >$ are going as input to the A1 block of addition gate and minimum gate, respectively. Similarly, the second fan-out gate produces two outputs $< X6\ t^\wedge\ X8 >$ and $< X5\ t^\wedge\ X7 >$, and they are given as input to the A2 block of addition gate and the minimum gate, respectively. The addition gate [84] consists of two blocks A1 and A2. The first input $< X2\ t^\wedge\ X4 >$ reacts with the A1 block to produce $< X12\ t^\wedge\ X11 >$ with a concentration equivalent to the concentration of the input $< X2\ t^\wedge\ X4 >$ *i.e.,* concentration of i_1. Similarly, the second input $< X6\ t^\wedge\ X8 >$ react with the A2 block of the addition gate to produce $< X12\ t^\wedge\ X11 >$ with a concentration equivalent to $< X6\ t^\wedge\ X8 >$ *i.e.,* concentration of i_2. Now, the concentration of $< X12\ t^\wedge\ X11 >$ produced will be equal to the sum of concentrations of the inputs i_1 and i_2. The minimum gate will react with $< X1\ t^\wedge\ X3 >$ (having concentration equal to $[i_1]$) and $< X5\ t^\wedge\ X7 >$ (having concentration equal to $[i_2]$) to produce an output strand $< X7\ t^\wedge\ X9 >$ whose concentration will be equal to the minimum of two inputs $[i_1]$ and $[i_2]$. Now, the subtraction gate takes $< X7\ t^\wedge\ X9 >$ as the first input and $< X12\ t^\wedge\ X11 >$ as the second input. The second input is considered as the output of a subtraction gate. Hence, the equilibrium concentration of the output strand O_{max} is given

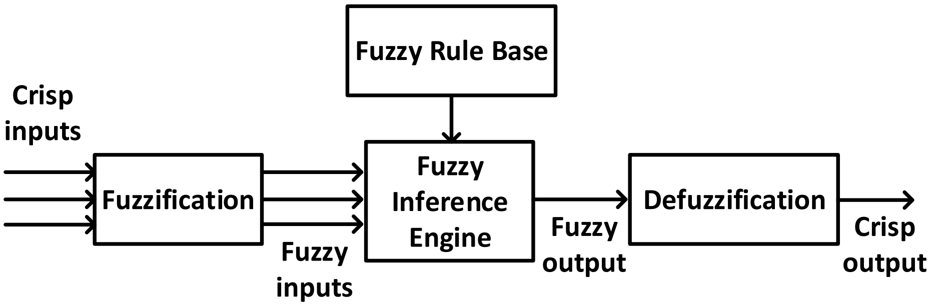


Figure 5.7: Architecture of fuzzy expert system

by:

$$\begin{aligned} [O_{max}]_\infty &= ([i_1]_0 + [i_2]_0) - min([i_1]_0, [i_2]_0) \\ &= max([i_1]_0, [i_2]_0) \end{aligned} \tag{5.4}$$

Some of the properties of the minimum and maximum functions can be found in [109]. The minimum and maximum functions are very strong mathematical operators for many non-linear systems. One of such systems which uses the minimum and maximum operators is the fuzzy inference engine.

5.3: Fuzzy Logic

Fuzzy set theory was first proposed by Zadeh in 1965 [110]. Fuzzy logic is a heuristic approach in which there is a non-linear mapping of input attributes to output, which is possible by defining a set of rules. As compared to the Boolean logic, a smooth transition from true to false is possible in fuzzy logic.

5.3.1: Modeling of Fuzzy Expert System

The fuzzy expert system is based on *if-then* rules and fuzzy reasoning. The block diagram of a fuzzy expert system is given in Fig. 5.7. It consists of a fuzzification block, an inference engine which acts on the fuzzy rule base, and a defuzzification block. The fuzzification block converts the crisp input into a fuzzy input (membership and degree of membership). Let X be the universe of discourse, then a fuzzy set A, on X is defined by:

$$A = \{(x, \mu_A(x)) \mid x \in X\} \tag{5.5}$$

where, $\mu_A(x)$ is the membership function (MF) for the fuzzy set A *i.e.,* the degree to which the input x has the property A. The MF maps each element of X to a membership value between 0 and 1. The shape of the MF can be triangular, trapezoidal, Gaussian, etc. [111]. Linguistic variables can be used to represent these properties. For example, if the height of a patient is considered as the input, then the linguistic variables could be very short, short, medium, tall, and very tall. If we say that *'the patient's height is short'*, this could be true to some degree between 0% to 100%, where 0% indicates non-membership and 100% indicates full membership. For example, *'The grade of membership of the patient's height being short is 0.8'* indicates that the probability that the patient would be considered as *short* is 0.8.

The fuzzy input generated from the fuzzifier is then sent to an inference engine. The inference is created using the fuzzy logic operators. Similar to the Boolean logic, a fuzzy AND and OR logic operations can be used to define the rules [112, 113]. The Zadeh fuzzy logic AND operation (fuzzy intersection operation) is given by:

$$\mu_{A \cap B}(x) = min(\mu_A(x), \mu_B(x)) \tag{5.6}$$

where $min()$ is the minimum operator, and $\mu_A(x)$ and $\mu_B(x)$ are the degrees of memberships corresponding to fuzzy sets A and B for an input x. Similarly, the Zadeh fuzzy logic OR operation (fuzzy union operation) is defined as:

$$\mu_{A \cup B}(x) = max(\mu_A(x), \mu_B(x)) \tag{5.7}$$

where $max()$ is the maximum operator.

A fuzzy logic system can be considered as an expert system which operates on a set of rules [114]. Rules are some conditional statements which link the inputs and output. The rule base uses *if-then* conditions. The fuzzy inference system may be a Mamdani fuzzy inference system which uses the minimum and maximum operators [115, 116], or a Takagi-Sugeno fuzzy inference system which uses a weighted linear combination of crisp inputs [117, 118]. The Mamdani fuzzy inference engine takes fuzzy inputs and produces fuzzy output based on the pre-defined rules. On the other hand, Takagi-Sugeno fuzzy inference system takes fuzzy inputs and produces crisp outputs. The Mamdani model requires a defuzzification block to convert the membership degrees of fuzzy output to a real value. There are different defuzzification methods available in the literature [119]. In this chapter, we are designing a DSD based Mamdani fuzzy inference using the minimum and maximum operators.

5.3.2: DNA Implementation of Mamdani Fuzzy Inference

The Mamdani fuzzy inference engine takes fuzzy inputs and produces a fuzzy output based on a pre-defined set of rules. For the DNA implementation of the Mamdani fuzzy inference, we assume that the fuzzy inputs are already available. The different DNA strands can be used as the different memberships and the concentration of these DNA strands can be considered as the degree of membership. The DNA circuit can be designed based on the rule set. The minimum gate discussed in this chapter have a modular design, hence they can be connected in cascades to implement the rules.

Consider a system which has two inputs (x and y) and one output (f). For simplicity, we assume that the inputs and output are having two membership functions (*low* and *high*). The following rules are considered for the DNA fuzzy inference engine:

Rule 1: If x is *low* and y is *low*, then f is *low*

Rule 2: If x is *low* and y is *high*, then f is *high*

Rule 3: If x is *high* and y is *low*, then f is *high*

Rule 4: If x is *high* and y is *high*, then f is *low*

From these rules, it can be seen that the output f is *low* for Rule 1 and Rule 4. Hence, these two conditions can be combined with OR logic. Similarly, the Rule 2 and 4 can be combined using OR logic to define the *high* condition of the output. Therefore, we can define the MFs for output as:

$$\begin{aligned} \mu_{low}(f) &= max(min(\mu_{low}(x), \mu_{low}(y)), \\ &\qquad min(\mu_{high}(x), \mu_{high}(y))) \qquad (5.8) \\ \mu_{high}(f) &= max(min(\mu_{low}(x), \mu_{high}(y)), \\ &\qquad min(\mu_{high}(x), \mu_{low}(y))) \qquad (5.9) \end{aligned}$$

where, $\mu_{low}(x)$, $\mu_{low}(y)$, and $\mu_{low}(f)$ corresponds to the degree of membership for the fuzzy set *low* for input x, y, and output f, respectively. Similarly, $\mu_{high}(x)$, $\mu_{high}(y)$, and $\mu_{high}(f)$ correspond to the degree of membership for the fuzzy set *high* for input x, y, and output f, respectively. The expressions given in eq. (5.8) and eq. (5.9) are using only the minimum and maximum operations and correspondingly, it can be implemented using DNA minimum and maximum gates. This fuzzy inference could be considered as a system which checks the similarity between two inputs concentration. The system will produce a high output when the inputs are not equal and produce a low output when the inputs are similar. These conditions are very similar to well known exclusive-OR logic gate, which has got a variety of applications. It could be noted that our research is focused only on the fuzzy inference block. The output of the fuzzy inference engine is not a crisp output. A defuzzification block is required to convert the fuzzy output to a crisp output. In future, with DNA circuits which can perform the complex arithmetic operations such a division, averaging etc. it could be possible to implement fuzzification and defuzzification blocks. The DSD based fuzzy inference engine discussed here can be extended to use any number of fuzzy inferences with the help of minimum, fan-out and maximum gates. Even though, the Takagi-Sugeno fuzzy inference system uses the fuzzy operators discussed in this chapter, the output of the inference system is a polynomial function. Hence, the DSD based fuzzy inference engine discussed here will not fit to the Takagi-Sugeno fuzzy system.

5.4: Simulation Results and Discussion

All the circuits proposed in this chapter are designed and tested in Visual DSD [68]. We use a programming language described in [66] to write the program in the Visual DSD. The Visual DSD also generated the MATLAB code and we used this code in MATLAB to speed up the simulation. We choose 10nM as the unit for concentration in the simulation. We used a toehold binding rate of $2 \times 10^{-3} nM^{-1} s^{-1}$ and unbinding rate of $10 s^{-1}$ for all the designs [6]. The settling time (t_s) of the analog gate is the time required for the response to reach the valid output range. Here, the valid output range is set with an error bank of 5%.

5.4.1: Minimum Gate

The minimum gate is simulated in different ranges such as $(0, 1)$, and $(0, 2)$. The corresponding r_{min} for the design is 1 and 2, respectively. As we discussed in section 5.2.2, the minimum gate can give both the minimum and subtraction outputs. The response of the system for $[i_2]_0 = 0.6$ and $[i_1]_0 = 0.4$ for the r_{min} equal to 1 and 2 are shown in Fig. 5.8 and 5.9, respectively. The minimum output $[O_{min}]_\infty$ is obtained at 0.4 and the subtracted

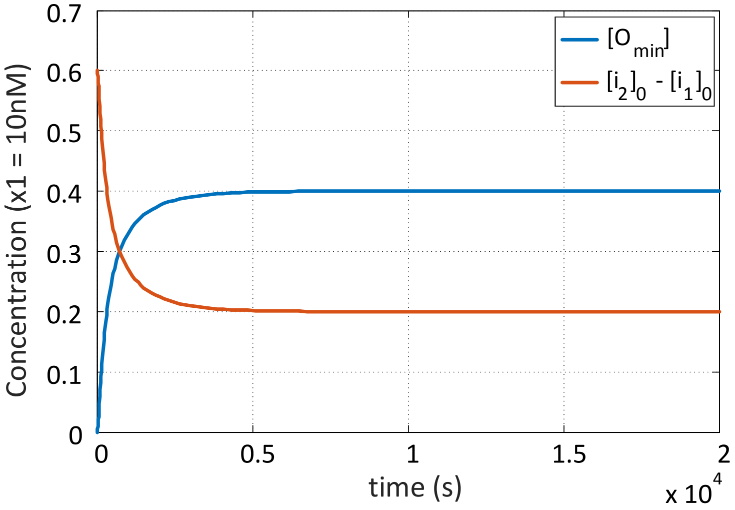


Figure 5.8: Simulation results of minimum gate in Visual DSD. Response of minimum gate for $[i_2]_0 = 0.6$ and $[i_1]_0 = 0.4$ for the range $r_{min} = 1$

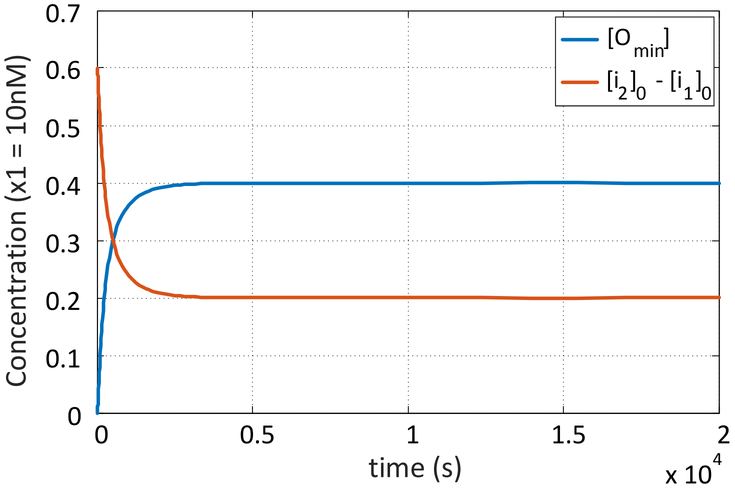


Figure 5.9: Simulation results of minimum gate in Visual DSD. Response of minimum gate for $[i_2]_0 = 0.6$ and $[i_1]_0 = 0.4$ for the range $r_{min} = 2$

output $[i_2]_\infty$ is obtained as 0.2. It was found that the minimum gate with $r_{min} = 2$ is giving a faster response than the minimum gate with $r_{min} = 1$. The subtraction operation will give a result only when $[i_2]_0 \geq [i_1]_0$. In all other cases, it gives a zero output.

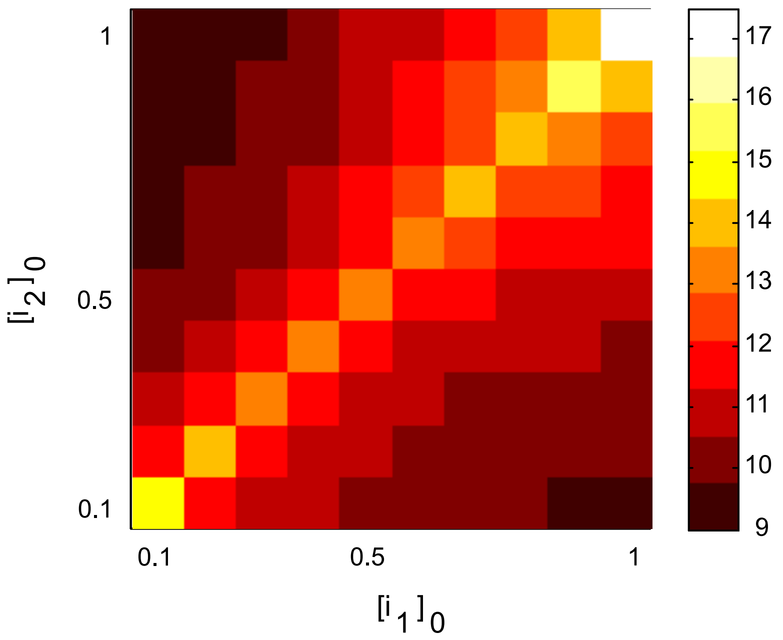


Figure 5.10: A heat map showing the variation of settling time (t_s) with changes in the inputs $[i_1]_0$ and $[i_2]_0$ for the range $(0, 1)$

We simulated the minimum gate for the input values in the range $(0, 1)$ with increments of 0.1 and in the range $(0, 2)$ with increments of 0.2. The variation in settling time with the input combinations in the range $(0, 1)$ with $r_{min} = 1$, and $(0, 2)$ with $r_{min} = 2$ is shown in Fig. 5.10 and 5.11, respectively. The settling time is taken in log_2 scale while plotting to clearly show the variation in t_s with changes in $[i_1]_0$ and $[i_2]_0$. It can be seen from this figure that, the settling time is more when the inputs initial concentrations are close to each other. Similarly, when the difference between input initial concentrations increases, the settling time decreases. Another observation from the figures is that when the range increases, t_s decreases.

5.4.2: Fan-out Gate

The 2 output fan-out gate is simulated for the ranges $r_f = 1$ and $r_f = 2$. The response obtained for the input $[i_1]_0 = 0.5$ for $r_f = 1$ and $r_f = 2$ are shown in Fig. 5.12 and 5.13,

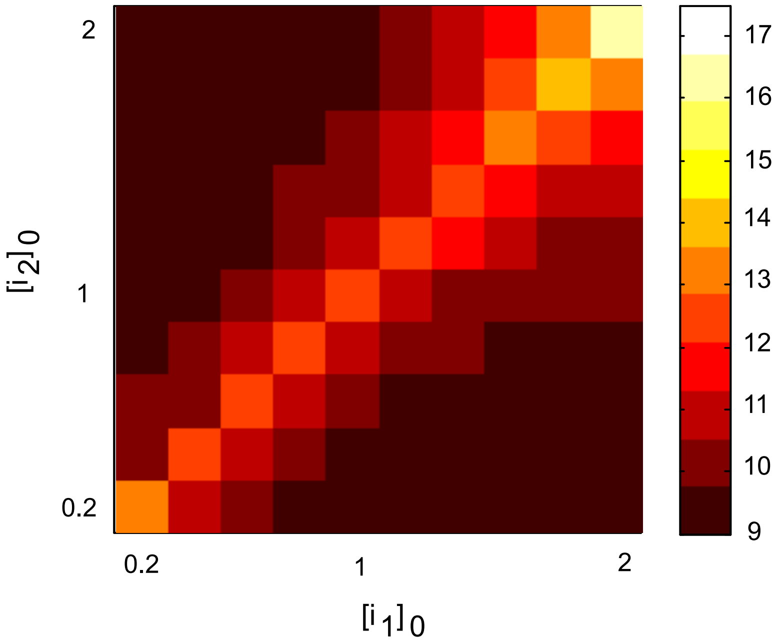


Figure 5.11: A heat map showing the variation of settling time (t_s) with changes in the inputs $[i_1]_0$ and $[i_2]_0$ for the range $(0, 2)$.

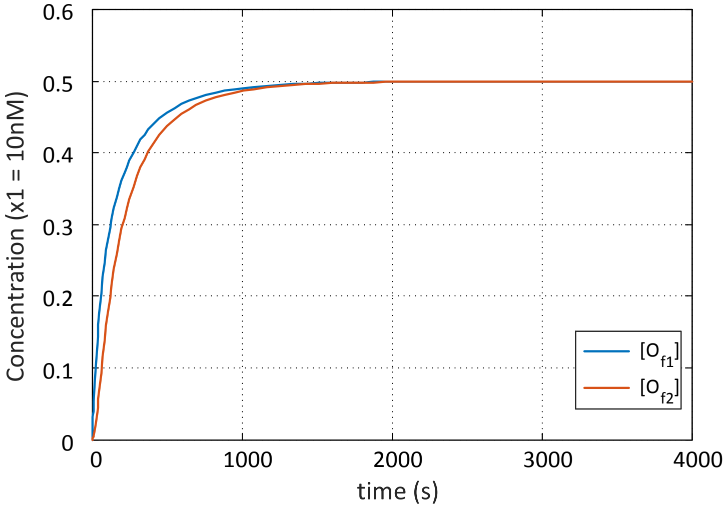


Figure 5.12: Simulation results of 2 output fan-out gate in Visual DSD. Response of fan-out gate for $[i_1]_0 = 0.5$ for the range $r_f = 1$

respectively. There is a small delay in the response of the outputs $[O_{f1}]$ and $[O_{f2}]$. It is clear from the figure that $[O_{f2}]$ lags behind $[O_{f1}]$. When the range increases, this lag decreases.

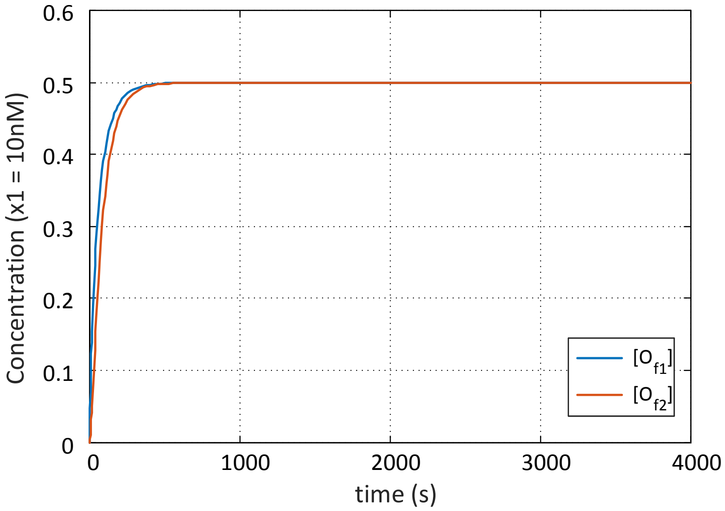


Figure 5.13: Simulation results of 2 output fan-out gate in Visual DSD. Response of fan-out gate for $[i_1]_0 = 0.5$ for the range $r_f = 2$

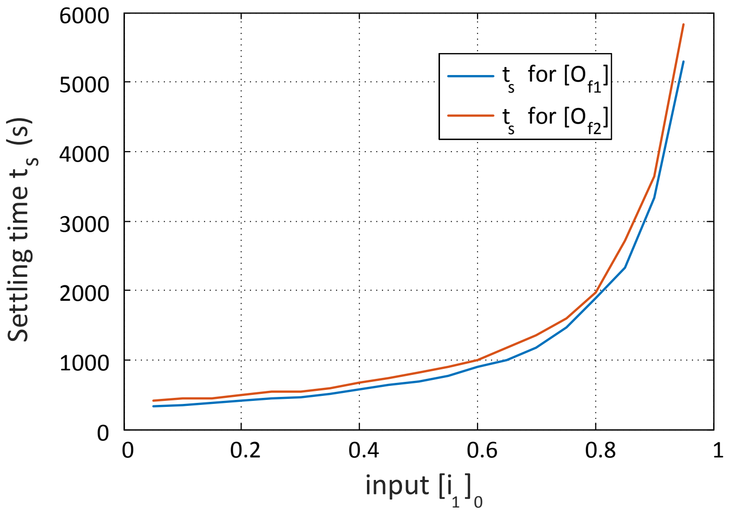


Figure 5.14: A graph showing the variation of settling time (t_s) with changes in the inputs $[i_1]_0$ for the range $(0, 1)$

The fan-out gate gives a faster response for higher ranges.

The variations in settling time of $[O_{f1}]$ and $[O_{f2}]$ with changes in $[i_1]_0$ for the range $(0, 1)$ and $(0, 2)$ are shown in Fig. 5.14 and 5.15, respectively. The input values in the range $(0, 1)$

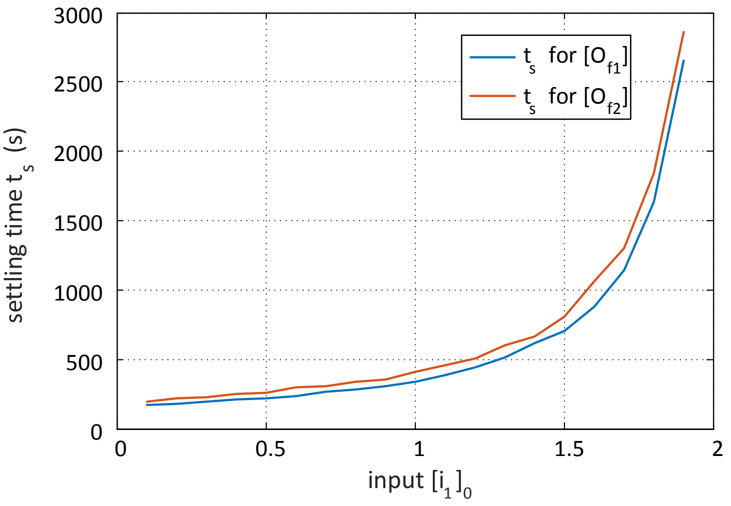


Figure 5.15: A graph showing the variation of settling time (t_s) with changes in the inputs $[i_1]_0$ for the range $(0, 2)$.

are taken with increments of 0.1 and in the range $(0, 2)$ with increments of 0.2. The settling time increases with an increase in the input $[i_1]_0$. This graph is exponential in nature. From Fig. 5.14 and 5.15, it can be observed that the settling time decreases with increase in range. This is because when the concentration of the reactants increases, the reactions run more quickly.

5.4.3: Maximum Gate

The maximum gate consists of two fan-out gates, two minimum gates, and an addition gate. The input range is selected as $r = 1$. The simulation result of the maximum gate $[O_{max}]$ for $[i_1]_0 = 0.4$ and $[i_2]_0 = 0.6$ is shown in Fig. 5.16. The variation in the settling time (t_s) of the maximum gate with changes in input initial concentrations is shown in Fig. 5.17. Here also, the settling time is taken in log_2 scale to clearly show the variations. It can be seen that the settling time is maximum when the inputs are equal. This high settling time is due to the effect of minimum gates, which also gives the maximum settling time when

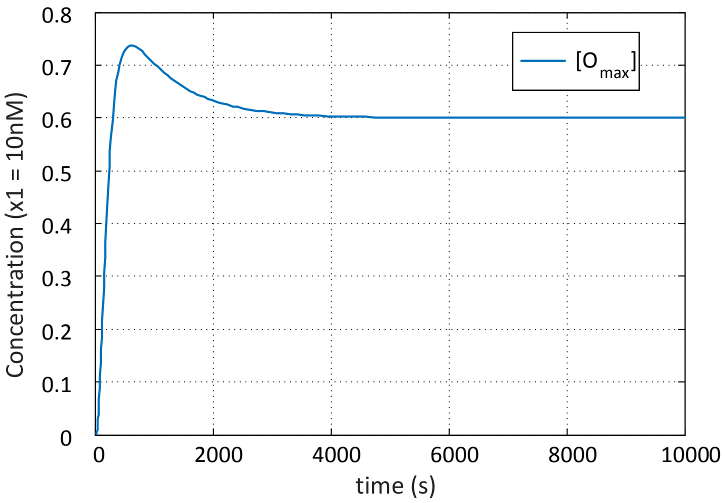


Figure 5.16: Simulation results of the maximum gate in Visual DSD. Response of maximum gate for $[i_1]_0 = 0.4$ and $[i_2]_0 = 0.6$ for $r = 1$

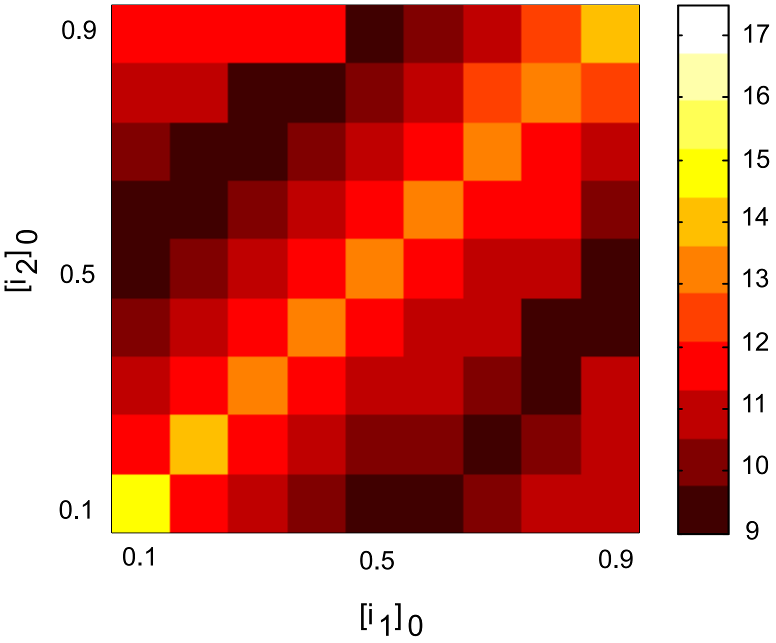


Figure 5.17: Heat-map showing the variation of settling time (t_s) with changes in the inputs $[i_1]_0$ and $[i_2]_0$ for the range $(0, 1)$.

the inputs are equal. When one of the inputs approach the range r, there is an increase in settling time. This comes from the fan-out gates, which shows a sudden increase in settling time while the input approaches range r.

5.4.4: Mamdani Fuzzy Inference System

The inference engine with the rules defined in section 5.3.2 is implemented in Visual DSD software using the minimum and maximum gates. We consider that the fuzzy inputs and outputs are having a triangular membership function. For the x input in the range $(0, 1)$, the membership function $\mu_{low}(x)$ and $\mu_{high}(x)$ can be defined as:

$$\mu_{low}(x) = 1 - x \tag{5.10}$$

$$\mu_{high}(x) = x \tag{5.11}$$

Similarly, for the y input also, the $\mu_{low}(y)$ and $\mu_{high}(y)$ membership functions are assigned with values $(y - 1)$ and y, respectively. There is no restriction that we should use the triangular membership function for the operation of the Mamdani fuzzy inference system. It can be any membership function such as triangular, Gaussian, trapezoidal, etc. The generation of fuzzy input from the crisp input is assigned to the fuzzifier. In our simulation, we are manually giving these fuzzy inputs ($\mu_{low}(x), \mu_{high}(x), \mu_{low}(y)$, and $\mu_{high}(y)$). We simulated our design for x and y by taking values in the range $(0, 1)$ with increments of 0.1. The variations in fuzzy outputs $\mu_{low}(f)$ and $\mu_{high}(f)$ for the inputs x and y (as given in eq. 5.8 and 5.9) obtained from the simulation is shown in Fig. 5.18 and 5.19, respectively.

The variation in the settling time (t_s) of outputs $\mu_{low}(f)$ and $\mu_{high}(f)$ is also analyzed. The $log_2(t_s)$ is calculated for $\mu_{low}(f)$ and $\mu_{high}(f)$ as shown in Fig. 5.20 and 5.21, respectively. It could be noted that the settling time is maximum when $\mu_{low}(x) = \mu_{low}(y)$ or $\mu_{high}(x) = \mu_{high}(y)$. Similarly, a sudden increase in settling time can be observed when $\mu_{low}(x) = \mu_{high}(y)$ or $\mu_{low}(y) = \mu_{high}(x)$. The variation is also symmetrical with respect to the diagonal, *i.e,* $x = y$. These sudden changes in settling time are due to the properties of the minimum,

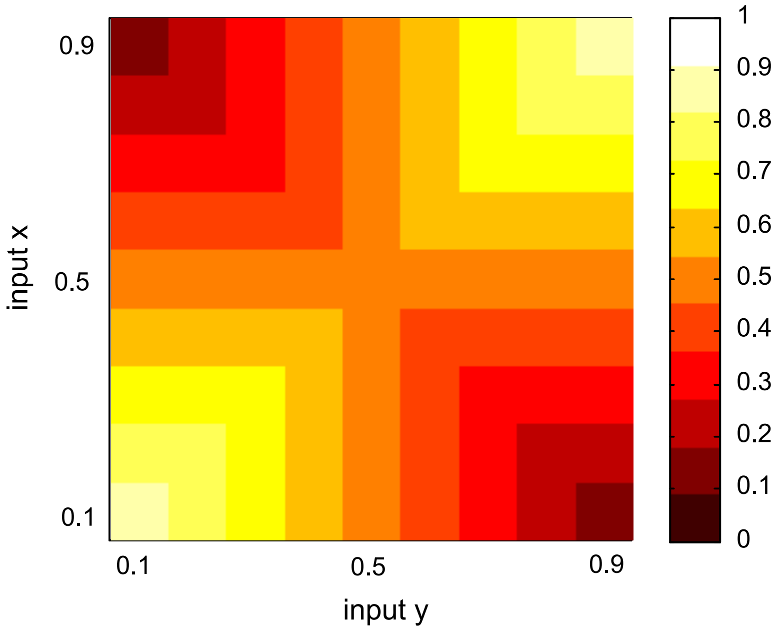


Figure 5.18: Variation of $\mu_{low}(f)$ with changes in inputs x and y

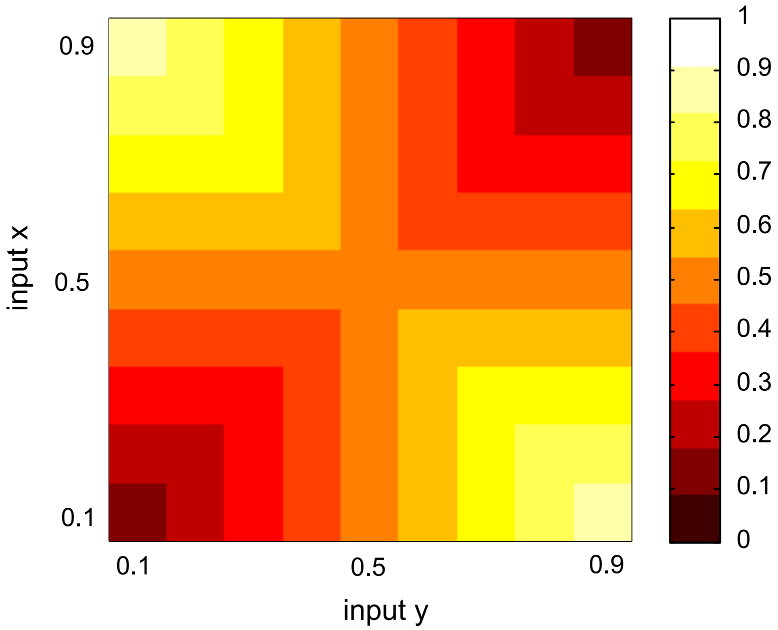


Figure 5.19: Variation of $\mu_{high}(f)$ with changes in inputs x and y

maximum gates used in the DNA fuzzy inference engine.

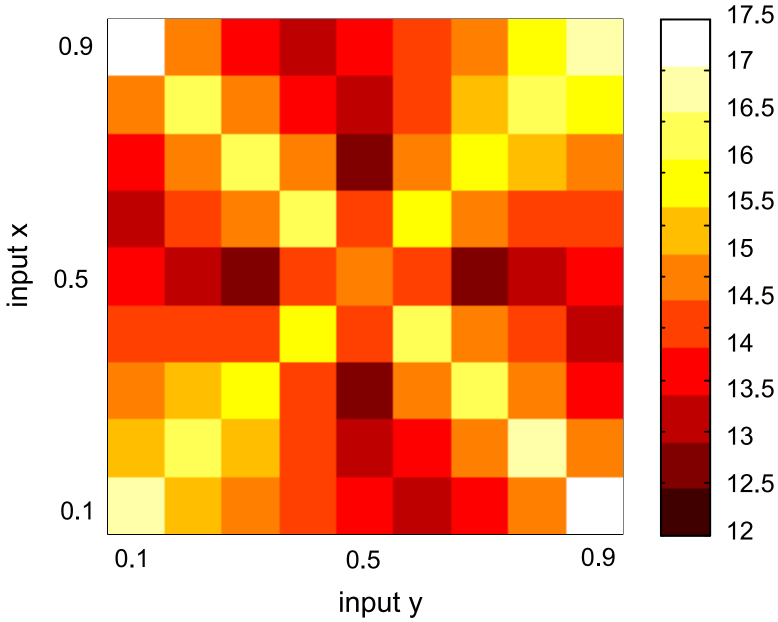


Figure 5.20: Variation in the settling time (t_s) of $\mu_{low}(f)$ with changes in inputs x and y

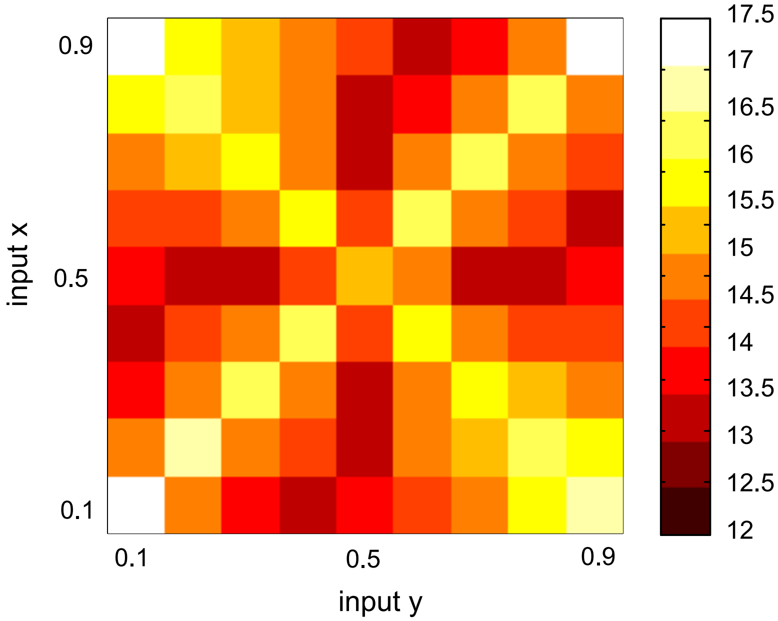


Figure 5.21: Variation in the settling time (t_s) of $\mu_{high}(f)$ with changes in inputs x and y.

5.4.5: Novelty of the Work

As far as the authors' knowledge, there is no DSD based designs available in the literature for the enzyme-free DNA implementation of minimum, maximum, or analog fan-out circuits. The minimum and maximum gate circuits can be used as a building block for many applications such as fuzzy inference engine, neural network, non-linear signal processing, function fitting, etc. [120]. The fan-out gates can be used in any analog circuit, to produce the copies of the signal. We hope that the introduction of minimum, maximum, and fan-out gates will pave the way for developing many other complex DNA circuits. Here, we are limiting our discussion to the fuzzy inference engine using the minimum and maximum gates. Even though fuzzy logic circuits implemented at the molecular level are available in the literature [17, 121], such circuits have a fixed function associated with it, in other words, they are not programmable. These molecular circuits also lack the modular property. The inputs and outputs of such circuits are of different types, and hence, they cannot be used as part of a large complex circuit. The inference engine developed here is programmable and the rules for the inference engine can be changed. The DNA based decision making systems currently available in the literature are fully digital [6, 7, 45]. The introduction of fuzzy inference based decision-making systems will give more control over systems with uncertainty such as biological systems.

5.5: Wet Lab Implementation of Minimum Gate

5.5.1: Sequence Design

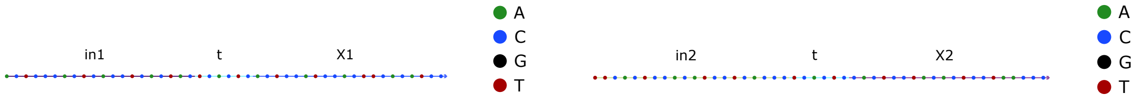
The sequence used for the minimum gate was first generated from the Visual DSD code, based on the visual DSD program. The toeholds are chosen such that the number of nucleotides in the toehold is very less compared to the recognition domains.The Visual DSD code generated a length of $6nt$ for the toeholds and $20nt$ for the recognition domains. The software generated base pairs were then optimized in the NUPCK software [122]. The strand details of each domain used in the minimum gate are shown in Table 5.1. It can be seen

Table 5.1: The domains and corresponding base pair sequences for minimum gate obtained from NUPACK software

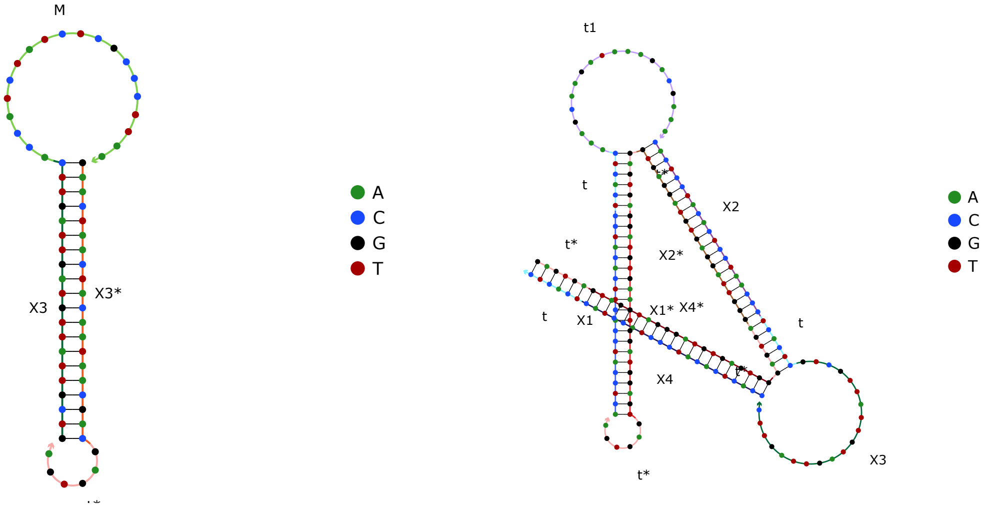
Domain Name	**Domain Sequence**
t	TCACTC
t1	AAAGCAAGATAAAGACGAAA
in1	ACTCCCACTCACCTCACTAC
in2	TTCACTCCACATCCTCACCC
X1	ACTCCATCCACTTCACATCC
X2	CACTCCTACACTCCTAACCC
X3	GTCGTTATTGTAGTTAGTTC
X4	CCACTCACATCCCTACCTAC
M	ACCATCTATCTCGCCCTTAA

that these domains are optimized to be functionally independent of each other in order to minimize the leak reactions. It can be observed from the table that the domains are using either A, C, or T in most of the cases and the use of G is minimized to avoid the generation of the secondary structures [123–125]. The C content in the sequence is chosen between 30% to 70% to ensure the normal melting temperatures [126]. The NUPACK software took all these constraints into consideration while designing the strands. Different DNA strands used in the design are generated by cascading the domains given in Table 5.1. A common toehold domain (t) is used for all the strands.

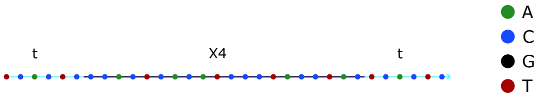
The final complexes for the minimum gate as given in Fig. 5.1 are generated using the NUPACK software and analyzed them. The different designs generated in the NUPACK for



(a) I1 (b) I2



(c) Reporter (d) M1



(e) M2

Figure 5.22: Different DNA strands generated from NUPACK

various complexes are shown in Fig. 5.22.

5.5.2: Circuit Preparation

After analyzing the strands designed from the Visual DSD and NUPACK, the strands are purchased from Integrated DNA Technologies (IDT). The details of the different strands of the design are shown in Table 5.2. It could be noted that the fluorophore and quencher are connected to the leg8 and leg9 respectively. The fluorophore used here is 6-FAM which can be

Table 5.2: The details of different strands used in the design of minimum gate

Strand	**Doamins**	**Sequence**
leg1	In1 t X1	ACTCCCACTCACCTCACTACTCACTCACTCCATCCACTTCACATCC
leg2	in2 t X2	TTCACTCCACATCCTCACCCTCACTCCACTCCTACACTCCTAACCC
leg3	X1 t t1	ACTCCATCCACTTCACATCCTCACTCAAAGCAAGATAAAGACGAAA
leg4	X2 t X3	CACTCCTACACTCCTAACCCTCACTCGTCGTTATTGTAGTTAGTTC
leg5	X4 t	CCACTCACATCCCTACCTACTCACTC
leg6	t* X4* t* X2* t* X1* t*	GAGTGAGTAGGTAGGGATGTGAGTGGGAGTGAGGGTTAGGAGTGTAGGAGTGGAGTGAGGATGTGAAGTGGATGGAGTGAGTGA
leg7	t X4 t	TCACTCCCACTCACATCCCTACCTACTCACTC
leg8	X3-F	GTCGTTATTGTAGTTAGTTC-/36-FAM/
leg9	Q-X3* t*	/5IABkFQ/-GAACTAACTACAATAACGACGAGTGA

connected to the 3' end of the strand and the quencher used is the Iowa Black FQ which can be connected to the 5' end of the strand. We modified the reporter strand used in the simulation by adding fluorophore and quencher to the reporter strand for the experimental setup. All the strands except the reporter strands are ordered from IDT, purified by polyacrylamide gel electrophoresis (PAGE). On the other hand, the reporter strand with a fluorophore and quencher were ordered with high-performance liquid chromatography (HPLC) purification. All the strands are ordered as 250 nM DNA oligos.

The strands are shipped to us as a chemical powder after purification from IDT. At first, we dissolve the DNA strands in 500 mL DM water. The solution is placed in a Vertex to fully dissolve the strands in the DM water. The concentrations of each of the strands were calculated by measuring the absorbance at 260 nm. The instrument we used for measuring the absorbance was Multiskan GO Microplate Spectrophotometer. The absorption is measured by taking 2 μL of sample and diluted it in 98 μL of DM water. The extinction coefficient (in L/(mol. cm) given in the specification sheet of the DNA strands (provided by the IDT) were used for calculating the concentrations of the DNA strands. The formula we used for the calculation of the concentration (c) of the DNA strands in μM is given by:

$$c = 100 \times OD260/e \times 10^6 \tag{5.12}$$

were, $OD260$ is the absorbance measured at 260 nm, and e is the extinction coefficient. Please note that the Cuvette we are using for the experiment is having 1cm pathlength and hence we can use the extinction coefficient directly in the equation. We verified the results by comparing it with the equivalent concentration of the strands given in the specification sheet for 100 μM provided by IDT.

The DNA strands are then annealed at 100 μM in Tris-acetate-EDTA buffer containing 12.5 mM Mg^{2+} ($1 \times TAE/Mg^{2+}$). The $1 \times TAE/Mg^{2+}$ solution is prepared by mixing 0.5 mL of TAE 10× solution, Magnesium Acetate Tetrahydrate, and 4.5 mL of DI water. The TAE 10× solution and Magnesium Acetate Tetrahydrate were purchased from Fisher Scientific. The annealing was performed by heating the strands to 95^0 C and slowly cooling it down to atmospheric temperature. The complex strands such as M1 and Reporter were annealed by mixing the domains used in them together. For example, for the M1 complex, we mixed the strands leg3, leg4, leg5, and leg6 together.

5.5.3: Kinetic Experiments

The kinetic experiments for the minimum gate were performed with a spectrofluorometer (FluoroMax-4 from Horiba Scientific). For the 6-FAM fluorophore, the excitation and emission wavelengths were set at 495 nm and 515 nm, respectively. The data points were recorded in the experiment and the cuvette is thoroughly cleaned with TAE buffer in between experiments. The raw data from the experiment were normalized to the relative concentra-

Table 5.3: The volume of different strands in the solution for different set of input concentrations

in1 (nM)	**in2** (nM)	**M1+M2+Reporter**(μL)	**Buffer** (μL)	**in2** (μL)	**in1** (μL)
20	20	180	0	60	60
0	20	180	60	0	60
5	15	180	60	15	45
10	10	180	60	30	30
15	5	180	60	45	15
20	0	180	60	60	0
5	10	180	75	15	30
10	5	180	75	30	15
15	5	180	60	45	15
5	15	180	60	15	45

tion of the output strands. The amount of different strands volumes that must be used in the experiment for the different set of input concentrations are given in Table 5.3. In the experiment, the maximum output concentration ($20nM$) is taken as 1. The simulation results for the different set of input concentrations are shown in Fig. 5.23. Even though the wet lab experimental results are encouraging, it can be seen that there are some errors in the experimental results while comparing it with the simulation result. This is due to the presence of leakage reactions in the experiments. This error is high when the inputs are high. The leakage reactions are due to the design imperfections of the DNA strands. We use the best available software (NUPACK) for the design of the strands. More competitive software is required to get optimum strands which can minimize the leakage reactions in the wet lab

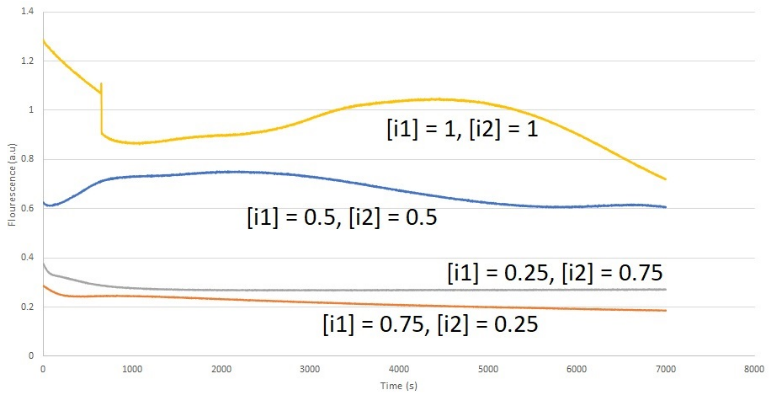


Figure 5.23: Experimental results for the minimum gate with different set of input concentrations

experiment.

5.6: Conclusion

In this research, we developed the analog minimum and maximum gates using DSD operations. We also implemented the analog fan-out gate. Using these DNA gates, a fuzzy inference engine which is capable of making decisions based on a set of pre-defined rules is designed. We analyzed the performance of the analog DNA gates and the fuzzy inference engine for different input ranges. All the proposed DNA implementations were tested in Visual DSD and analyzed by importing the code into MATLAB. To the best of our knowledge, the fuzzy inference engine using DSD operation proposed in this chapter is the first of its kind. These systems can be used for the design of decision-making systems for the bio-nanorobots, smart drugs, and engineered viruses for the treatment of genetic diseases in future. The minimum and maximum gates proposed in this research can also be used for the design of more complex non-linear mathematical functions.

CHAPTER 6: CHALLENGES AND FUTURE RESEARCH

6.1: Introduction

The circuits using DNA provide many advantages such as small size, programmability, and bio-compatibility, but its wide applications will be based on their interface with the biology. In this research, we developed different DNA strand displacement (DSD) based circuits such as majority logic gate, inverter logic gate, and a fuzzy inference engine. Even though these circuits and all the other DNA circuits available in the literature are valuable towards the design of autonomous devices and structures for future medical applications, there are many challenges that need to be addressed. In this section, we are discussing those challenges and the future research directions. The challenges discussed in this chapter are not limited to the designs proposed in this thesis, but also to most of the DNA based circuits available in the literature. We are giving the future directions in the field of DNA circuit design such as developing a full fuzzy system, the design of analog to digital and digital to analog converters, and synthesis tools.

6.2: Challenges

Even though there are different techniques available for the enzyme-free DNA implementation of digital and analog circuits, they are not user-friendly to people from other disciplines. A language for representing the DSD reactions was first developed by Lakin *et. al.* [127]. There are very powerful simulation tools available such as seesaw compiler which can take a Boolean expression and generate the equivalent DSD code [6, 128]. Visual DSD is a software for analyzing the circuits made up of DSD operations [68] and is compatible with the language developed for DNA hybridization reactions [127]. This software can be used to simulate analog and digital circuits. Initially, this software was not able to handle localized designs. In 2014, the software was updated to support the localized circuits [69]. Recently, Peterson and co-workers developed the calculus for the modeling of DNA circuits with secondary structures such as CRN on surface-based circuits [129]. A complete synthesis flow is still not available for designing the digital circuits using DNA strands. Even though software like Visual DSD is available, it cannot take the common circuit representations such as schematic diagrams or hardware description language codes. Further research is needed in developing a design flow, which can take a hardware description language code and generate the DNA strands which are required for the circuit.

The verification of the DNA circuits is one of the major challenges. The conventional verification techniques will not work with the DNA circuits. This is because the faults and errors in conventional electronic circuits are entirely different from those in a DNA circuit. For example, the major source of error in conventional electronic circuits could be physical faults such as device failure or systematic errors [130, 131]. On the contrary, the errors in DNA circuits are primarily due to the spurious reactions. The spurious reactions will affect the concentration of the output strand and it will lead to inaccurate results. Another issue is the improper design of the strands which may cause some unwanted reactions and the signal may be stuck at some intermediate stage. In the localized circuits, the reaction is always associated with the reaction probability of two neighboring strands and this inherently has some errors.

Even though different analog circuits are available in the literature, most of the designs are not practically implemented in a wet lab. Background noise is a major issue in the analog circuits compared to its digital counterpart. For example, even though there is no trigger, the output signal can be produced because of the background noise. But in case of digital designs, we always consider a threshold to determine OFF and ON states. Thus the presence of noises will not make severe issues in the digital circuits as compared to the analog circuits. The design of DNA strands is very critical and improper design may lead to leakage reactions. Another source of leakage reaction is the triggering of the DNA circuit by unwanted DNA strands. The leakage reactions are not limited to analog circuits, but are common to all DSD based circuits. Another major issue is the number of unique DNA strands that are required for the designs. If we are not using unique strands for each gate, there is a chance of spurious reactions. One efficient method to counter this issue is the use of localized circuits, but the localized circuits are available only for the digital circuits. Most of the circuits which are practically implemented are use once type. Once the circuit reacts to the inputs, it cannot be used again for further reactions. Reusable designs of DNA circuits are a topic for future research.

The majority logic gate proposed in this research is a valuable addition to the family of the spatially localized digital circuits, but the practical implementation of localized designs is very difficult to achieve. The design requires a precise spatial arrangement of DNA hairpins within the Origami, which is a difficult task to achieve in the wet lab. The fuzzy inference engine proposed in this research cannot be used for medical applications in its current form. The inputs to the proposed fuzzy inference engine should be a fuzzy value, therefore we need a fuzzifier. Similarly, the output of the fuzzy inference engine is also a fuzzy value. To convert this fuzzy value to a crisp output we need a defuzzifier. In order to develop fuzzifiers and defuzzifiers, we need modular DNA circuits which can perform exponential operations, division operation, and many different complex mathematical operations. We hope that the future developments in DNA analog computing will produce modular architectures that can be used for the design of fuzzifier and defuzzifier functions. If the DNA circuits for fuzzifier and defuzzifier are available, it could be possible to integrate these circuits with the proposed fuzzy inference engine to design a complete fuzzy system. Such a fuzzy system will have DNA strand concentrations as inputs and outputs. We have done the implementation of minimum logic gate in the wet lab. However, the development of more complicated DNA circuits in the wet lab is highly challenging, time consuming, would require major funding.

6.3: Future Research

In this research work, we have developed an approach by which any Boolean function can be realized using DNA strands. This has opened a way for several challenging problems which can be taken up such as a generalized design of threshold gate, neural network, and fuzzy systems. We have performed the realization of well-known circuits such as full adder using manual method and also limited to a few number of variables. Sophisticated software programs are required to design generalized synthesis software, which could be applicable to any Boolean functions. Even though there are different DSD based circuits available in the literature, there is a lack of synthesizing software which can produce the DNA strands from the schematic design or from a code given in any hardware description language (HDL). The development of such a system, which can produce the DNA strand details of the schematic circuit or the HDL code is a fruitful problem for the future research. In our current research, we developed a DSD based fuzzy inference engine. The development of a full decision-making system, which consists of fuzzifier, fuzzy inference engine, and de-fuzzifier is a future research problem. The fuzzifier will convert the input signal to a fuzzy input, which is having a set of memberships and corresponding membership degrees. A particular DNA strand could be considered as a membership function and the concentration of the DNA strand as the degree of the membership function. The fuzzifier can be connected to the fuzzy inference engine and the output of the fuzzy inference engine can be connected to the defuzzifier. The fuzzy inference engine produces a fuzzy output based on a predefined set of rules. The defuzzifier converts the fuzzy output to a crisp output. For defuzzifier, the DNA circuit can take different DNA strands with different concentrations, and produce a single output strand. The fuzzifier and defuzzifier require complex mathematical operations. The implementation of such strong mathematical operations using DNA is a problem for future research.

Some of the bio-markers and bio-signals are analog and some are digital. The analog computing circuits use a much less strands compared to their digital counterpart, but they are highly affected by leak reactions. Hence, we have to select the computation based on the application. We can make a bridge between the analog and digital world of the DSD based circuits by developing an analog to digital converter and a digital to analog converter. The design of such analog to digital, and digital to analog converters are problems for future research. Another important component of any DNA devices is a signal processing unit. There are some control circuits, amplification circuits, arithmetic circuits, and timer circuits, which are available in the literature. Most of these circuits are based on chemical reaction networks, but proper DNA implementation is not available. Further research is required on the DNA implementation of different signal processing elements.

All the research discussed previously in this article was developed and tested in a cell-free setting. The transition from the test-tube to cell introduces a lot of challenges and opportunities to researchers. There are significant differences between a cellular environment and a cell-free wet-lab experimental set-up. In a cellular environment, there may be different elements which will adversely affect the dynamics of the circuits which are not considered in the cell-free environment. The structured nature of the cell and different proteins are some of the examples of such elements.

A detailed review of the challenges and future opportunities while integrating synthetic

DNA circuits into the cellular environment is given by Chen *et. al.* [132]. The major challenges while integrating synthetic DNA circuits into the cell can be summarized as the delivery of the circuit into the cell, sensing of bio-signals in the cellular environment, the stability of the nucleic acid circuits inside the cell, circuit functioning in the crowded cells, and prevention of immune activation. We hope that in a near future it will be possible to overcome these challenges and researchers could be able to produce biological devices that will bring revolutionary changes in the field of DNA nanotechnology and medicine.

6.4: Conclusion

Even though the future research of DNA functional nanotechnology revolves around the development of DNA circuits and their applications in the medical field, such circuits have to face many challenges to fulfill this dream. The development of more user friendly, strong, and efficient synthesis tools, verification methodologies specific to the DNA circuits, more accurate software for the design of DNA strand domains, and development of circuits that can perform more complex mathematical operations are some of the challenges that needs to be addressed by the engineering community. The circuits with complex structures such as spatially localized circuits are difficult to implement in the wet lab. Re-usability of the strands in the design, continuous use of the circuit, and the transition from test tube-based circuits to cell-based circuits are some of the other challenges identified. Based on the research presented in this thesis, we also gave the directions for future research.

CHAPTER 7: SUMMARY AND DISCUSSION

Circuits that can perform a pre-programmed function are an integral part of any autonomous device. These circuits that can directly interact with the biological signals are useful in case of biological devices. DNA is considered as a suitable candidate for designing such biological circuits because of the bio-compatibility, small size, light weight, programmability, well-known thermodynamics and structure formations, and exponentially decreasing cost of the synthesis. In this research, we investigated the possibility of using enzyme-free DNA strand displacement (DSD) operations for designing various logical operations. The DNA circuits can be modeled either as a digital circuit, or as an analog circuit.

The contributions presented in this research are summarized as follows:

- A comprehensive study of all the existing digital, as well as analog DNA circuits, is done.
- A spatially localized DSD based majority logic gate is presented.
- An inverter logic gate which could be used with the existing seesaw logic gates is developed.
- A fuzzy inference engine which is useful for the design of analog decision-making systems using DNA strands is presented.

The comprehensive study of the existing DSD based circuits gives the state of the art of the research in this direction and the problems of the existing techniques. The most popular methods available in the literature were discussed and a comparison of the existing techniques was also presented. In the second part of our research, a spatially localized architecture for majority logic gate is developed. The majority logic operations can reduce the size of the circuits considerably since many operations can be implemented with the majority logic gate at the expense of very few numbers of extra strands. The three input majority logic gate and five input majority logic gates were developed. We also gave the procedure for designing an n - input majority logic gate. The introduction of majority logic gate will provide more flexibility to the designer while designing large complex circuits.

In the third part of our research, an inverter logic gate is developed. This logic gate uses a subtraction gate and a gate enable circuit. The gate enable technique used in this design provides modularity property to the NOT gate. This NOT gate design can be used with the existing seesaw-based logic gates. Currently, the seesaw logic operations are using a dual-rail approach because of the unavailability of the NOT operation. With the introduction of an inverter logic gate, it could be possible to use mono-rail designs and thus reduce the number of unique strands required for the seesaw circuit design into approximately half.

Even though there are well-studied circuits and methods available for developing digital circuits, the number of unique strands required for computing more complex operations are very high. Such operations can be performed with less number of strands if we use analog circuits. For instance, the circuit for calculating the minimum between two numbers (each number represented by at least 3 bits) will result in a very complex circuit in the digital world. On the other hand, the same circuit can be designed with only two strands in an analog circuit as given in chapter 5. In this research, we developed different analog circuits such as the minimum gate, maximum gate, and fan-out gate using DSD operation. The concentration of the strands is used to represent the analog signal. The minimum logic gate will produce an output strand concentration which will be the minimum between the two input strand concentrations. The fan-out gate is used for producing multiple copies of the input strand concentrations. Similarly, the maximum gate will produce the maximum between two input concentrations in the output. The minimum and maximum operations are very powerful non-linear operations and have many applications. In view of developing powerful decision making systems using DNA strands, we gave an algorithm for implementing Mamdani fuzzy inference engine using the proposed minimum, maximum, and fan-out gates. The fuzzy inference rules were written using the minimum and maximum operators. The fuzzy inference engine could be useful for a future complete fuzzy system for designing sophisticated control and decision-making applications.

All the circuits were tested in the Visual DSD software. The minimum gate proposed in this thesis is further analyzed using NUPACK and tested in wet-lab. Even though the DNA circuits proposed in this thesis will be a strong addition to the family of logic gates and analog gates using DNA strands, there are still many challenges that needed to be addressed. These challenges are discussed and the future research directions are also presented in chapter 6. One of the major challenges in this field is the transformation of these circuits from test tube to the cell. By considering the pace at which the research is going on in this field, it could be expected to have DNA decision making circuits even inside human body used for the treatment of many critical genetic disorders and other life-threatening diseases. We strongly believe that the logic gates and the fuzzy inference engine we developed during this research will be a valuable addition to the DNA circuits family and the proposed gates could be used for making such circuits for medical applications in future.

APPENDIX: VISUAL DSD CODES

Spatially localized majority logic gate

Three input majority logic gate

```
(* Spatially localized three input Majority gate design
Author: Aby K George *)
directive sample 700000.0 1000
directive polymers
directive simulation deterministicstiff
directive localconcentrations [ (a, 1000000); (b, 1000000); (c, 1000000);
(d, 100000); (e, 100000); (f, 100000)]
directive plot Z()

dom a = { colour = "red" }
dom x  = { colour = "green" }
dom y  = { colour = "blue" }
dom b = { colour = "purple" }
dom c = { colour = "yellow" }
dom blank = { colour = "black" }

def inputA()   = <a^ s>
def inputB()   = <b^ s>
def inputC()   = <c^ s>
def fuel()     = <y^*>[s*]{x^>
def probe()    = <z^*>[s*]<Q>{F}
def Z() = {s F}
def Origami()  =  [[ {tether(a,d) a^*}[s]{y^>
                   | {tether(b,e) b^*}[s]{y^>
     | {tether(c,f) c^*}[s]{y^>
```

```
                  | {tether(a,b,c) x^*}[s]{blank^>
                  | {tether(d,e,f) x^*}[s]{z^>
      ]]

( 1 * inputA()
| 1 * inputB()
| 0 * inputC()
| 3 * fuel()
| Origami()
| probe()
| 0 * Z()
)
```

Five input majority logic gate

```
(* Spatially localized five input Majority gate design
Author: Aby K George *)
directive sample 40000.0 1000
directive polymers
directive simulation jit
directive localconcentrations [ (a, 1000000); (b, 1000000); (c, 1000000);
(d, 1000000); (e, 1000000); (f, 1000000); (g, 1000000); (h, 1000000);
(i, 1000000); (j, 1000000); (k, 100000); (l, 100000); (m, 100000);
(n, 100000); (o, 100000)]
directive plot Z1()

dom a = { colour = "red" }
dom x  = { colour = "green" }
dom y  = { colour = "blue" }
dom b = { colour = "purple" }
dom c = { colour = ”yellow” }
dom blank = { colour = ”black” }


def inputA()   = <a^ s>
def inputB()   = <b^ s>
def inputC()   = <c^ s>
def inputD()   = <d^ s>
def inputE()   = <e^ s>
def fuel()     = <y^*>[s*]{x^>
def probe1()     = <z1^*>[s*]<Q>{F1}
def Z1() = {s F1}
def Origami()  =  [[(* First Majority Operation*)
     {tether(a,f,k) a^*}[s]{y^>
                | {tether(b,g,l) b^*}[s]{y^>
      | {tether(c,h,m) c^*}[s]{y^>
          | {tether(d,i,n) d^*}[s]{y^>
                | {tether(e,j,o) e^*}[s]{y^>
                | {tether(a,b,c,d,e) x^*}[s]{blank^>
      | {tether(f,g,h,i,j) x^*}[s]{blank^>
                | {tether(k,l,m,n,o) x^*}[s]{z1^>
      ]]


( 1 * inputA()
| 0 * inputB()
| 1 * inputC()
| 0 * inputD()
| 1 * inputE()
| 5 * fuel()
| Origami()
| probe1()
| 0 * Z1()
)
```

Inverter logic gate

```
(* Inverter Circuit
Author: Aby K George*)

directive sample 80000.0 100
directive simulation deterministic
directive plot <_ _ _ Oinv>

(* Rates at 25 C from the Qian and Winfree 2011 (page 48 SI) *)
(* Long toeholds bind fast and unbind slowly while short toeholds bind slow
   and unbind faster*)
directive toeholds 2.0E-3 1.3 (* Long toehold binding rate 2x10^-3 /nM/s,
   unbinding rates 1.3/s*)
directive leak 1.0E-8 (* Leak rate 10^-8 /nM/s *)
directive tau 1.0 (* Branch migration rate 1/s (from Zhang and Winfree 2009)*)
def shorttoeholdunbind = 26.0 (* Short toehold dissociation rate constant 26/s
    *)
def shorttoeholdbind = 5.0E-5 (* Short toehold binding rate constant 5x10^-5/
   nM/s *)

(* Short toehold *)
dom t={seq = TCT; bind=shorttoeholdbind; unbind=shorttoeholdunbind}

dom fL = {seq=CATT}
dom f = {seq=TTTTTTT}
dom fR = {seq=TTCA}

(* Set concentration with 1x = 50 nM *)
def N = 5

(* a seesaw signal *)
def signal(N,(iL,i,iR),(jL,j,jR)) = (N* <iL^ i iR^ t^ jL^ j jR^> )

(* a seesaw gate with signal bound by the left side *)
def gateL(N,(iL,i,iR),(jL,j,jR)) = (N* {t^*}:[iL^ i iR^ t^]<jL^ j jR^> )

(* Enable gate for making Inverter modular *)
def gate_enable(N,(dL,d,dR),(iL,i,iR),(jL,j,jR)) =
( N*{t^*}[dL^ d dR^ t^]:[iL^ i iR^ t^]<jL^ j jR^> )

(* a seesaw threshold with signal coming in from the left side *)
def thresholdL(N,(iL,i,iR),(jL,j,jR)) = ( N* {iR^* t^*}[jL^ j jR^] )

(* a seesaw reporter *)
def reporter(N,(iL,i,iR),Fluor) = (N* {t^*}:[iL^ i iR^]<Fluor> )

(* Inverter *)
def Gs(N,(iL,i,iR),(jL,j,jR)) = (N*{t^*}[jL^ j jR^ t^]:[iL^ i iR^])
def Ds(N,(jL,j,jR)) = (N*[jL^ j jR^]{t^*})
def Inverter(N,(iL,i,iR),(jL,j,jR)) =(Gs(N,(iL,i,iR),(jL,j,jR)) | Ds(N,(jL,j,
    jR)))
```

```
(* Delay *)
def delay_gate(N,(iL,i,iR),(jL,j,jR)) = (1000.0*50.0*<iL^ i iR^ t^> |
1000.0*50.0*[iL^ i iR^ t^]<jL^ j jR^> |
rxn <iL^ i iR^ t^> + [iL^ i iR^ t^]<jL^ j jR^> -> {10.0E-13} <iL^ i iR^ t^ jL^
    j jR^> + [iL^ i iR^ t^] | (* k_0bp = 0.49 /M.s. *)
N*{iR^ t^*}[jL^ j jR^] (* x10 = 10000 s. delay *)
)

def X1 = (S1L,S1,S1R)
def X2 = (S2L,S2,S2R)
def X3 = (S3L,S3,S3R)
def X4 = (S4L,S4,S4R)
def X5 = (S5L,S5,S5R)
def X6 = (S6L,S6,S6R)
def X7 = (S7L,S7,S7R)
def X8 = (S8L,S8,S8R)

def F = (fL,f,fR)

(
signal(1*N,X1,X2) | (* x=1 OFF, x=8 ON *)
(* Inverter *)
signal(10*N,X3,X4) |
Inverter(10*N,X4,X2) |
delay_gate(30,X5,X6) |
gate_enable(10*N,X6,X4,X7) |
thresholdL(5*N,X4,X7) |
gateL(10*N,X7,X8) |
signal(20*N,X7,F) |
reporter(100*N,X8,Oinv)
)
```

Fuzzy inference engine

```
(* Inference Rule Set _ XOR
Author: Aby K George*)

directive sample 100000.0 100
directive simulation deterministic
directive plot <S47L^ S47 S47R^ t^ S44L^ S44 S44R^>; <S61L^ S61 S61R^ t^ S58L^
    S58 S58R^>

def X1 = (S1L,S1,S1R)
def X2 = (S2L,S2,S2R)
def X3 = (S3L,S3,S3R)
def X4 = (S4L,S4,S4R)
def X5 = (S5L,S5,S5R)
def X6 = (S6L,S6,S6R)
def X7 = (S7L,S7,S7R)
def X8 = (S8L,S8,S8R)
def X9 = (S9L,S9,S9R)
def X10 = (S10L,S10,S10R)
def X11 = (S11L,S11,S11R)
def X12 = (S12L,S12,S12R)
def X13 = (S13L,S13,S13R)
def X14 = (S14L,S14,S14R)
def X15 = (S15L,S15,S15R)
def X16 = (S16L,S16,S16R)
def X17 = (S17L,S17,S17R)

def X18 = (S18L,S18,S18R)

def X19 = (S19L,S19,S19R)

def X20 = (S20L,S20,S20R)

def X21 = (S21L,S21,S21R)

def X22 = (S22L,S22,S22R)

def X23 = (S23L,S23,S23R)

def X24 = (S24L,S24,S24R)

def X25 = (S25L,S25,S25R)

def X26 = (S26L,S26,S26R)

def X27 = (S27L,S27,S27R)

def X28 = (S28L,S28,S28R)

def X29 = (S29L,S29,S29R)

def X30 = (S30L,S30,S30R)

def X31 = (S31L,S31,S31R)

def X32 = (S32L,S32,S32R)

def X33 = (S33L,S33,S33R)

def X34 = (S34L,S34,S34R)

def X35 = (S35L,S35,S35R)

def X36 = (S36L,S36,S36R)

def X37 = (S37L,S37,S37R)

def X38 = (S38L,S38,S38R)

def X39 = (S39L,S39,S39R)

def X40 = (S40L,S40,S40R)

def X41 = (S41L,S41,S41R)

def X42 = (S42L,S42,S42R)

def X43 = (S43L,S43,S43R)

def X44 = (S44L,S44,S44R)
def X45 = (S45L,S45,S45R)
def X46 = (S46L,S46,S46R)
def X47 = (S47L,S47,S47R)
def X48 = (S48L,S48,S48R)
def X49 = (S49L,S49,S49R)
def X50 = (S50L,S50,S50R)
def X51 = (S51L,S51,S51R)
def X52 = (S52L,S52,S52R)
def X53 = (S53L,S53,S53R)
def X54 = (S54L,S54,S54R)
def X55 = (S55L,S55,S55R)
def X56 = (S56L,S56,S56R)
def X57 = (S57L,S57,S57R)
def X58 = (S58L,S58,S58R)
def X59 = (S59L,S59,S59R)
def X60 = (S60L,S60,S60R)
def X61 = (S61L,S61,S61R)

def minimum_gate((iL,i,iR),(jL,j,jR),(kL,k,kR),(lL,l,lR)) =
( 100.0* {t^*}[iL^ i iR^ t^]<t1>:[jL^ j jR^ t^] <kL^ k kR^>:[lL^ l lR^ t^]   |
  10.0*<t^ lL^ l lR^ t^> )

def reporter(N,(iL,i,iR),Fluor) = ( N * {t^*}:[iL^ i iR^]<Fluor> )

def input(j,(iL,i,iR)) = (<j t^ iL^ i iR^>)

def Fanout((iL,i,iR),(jL,j,jR),(kL,k,kR),(i1L,i1,i1R),(j1L,j1,j1R),(k1L,k1,k1R
    )) =
  ( <t^ jL^ j jR^ > | <t^ kL^ k kR^ t^> |
  {t^*}[iL^ i iR^ t^]<i1L^ i1 i1R^>: [jL^ j jR^ t^]<j1L^ j1 j1R^>: [kL^ k kR^
      t^]<k1L^ k1 k1R^>)

def gateL(N,(iL,i,iR),(jL,j,jR)) = (N* {t^*}:[iL^ i iR^ t^]<jL^ j jR^> )

def Gs(N,(iL,i,iR),(jL,j,jR)) = (N*{t^*}[jL^ j jR^ t^]:[iL^ i iR^])
def Ds(N,(jL,j,jR)) = (N*[jL^ j jR^]{t^*})
def Subtractor(N,(iL,i,iR),(jL,j,jR)) =(Gs(N,(iL,i,iR),(jL,j,jR)) | Ds(N,(jL,j
  ,jR)))

def gate_enable(N,(dL,d,dR),(iL,i,iR),(jL,j,jR)) =
(
N*{t^*}[dL^ d dR^ t^]:[iL^ i iR^ t^]<jL^ j jR^>
)

def delay_gate(N,(iL,i,iR),(jL,j,jR)) = (1000.0*50.0*<iL^ i iR^ t^> |
1000.0*50.0*[iL^ i iR^ t^]<jL^ j jR^> |
rxn <iL^ i iR^ t^> + [iL^ i iR^ t^]<jL^ j jR^> -> {10.0E-13} <iL^ i iR^ t^ jL^
    j jR^> + [iL^ i iR^ t^] | (* k_0bp = 0.49 /M.s. *)
N*{t^*}[jL^ j jR^] (* x10 = 10000 s. delay *)
)

(* Adder Gate *)
def Fa((iL,i,iR)) = <b^ iL^ i iR^ t^>
def Ga((iL,i,iR),(jL,j,jR),(kL,k,kR)) = {t^*}[iL^ i iR^ b^]:[kL^ k kR^ t^]<jL^
    j jR^>
def Da((iL,i,iR)) = [iL^ i iR^]{b^*}
def Adder((iL,i,iR),(i1L,i1,i1R),(jL,j,jR),(kL,k,kR)) =
   ( Fa((kL,k,kR)) | Ga((iL,i,iR),(jL,j,jR),(kL,k,kR)) | Da((iL,i,iR)) |
   Fa((kL,k,kR)) | Ga((i1L,i1,i1R),(jL,j,jR),(kL,k,kR)) | Da((i1L,i1,i1R)))

def x = 3.0 (*Range 0 to 10 *)
def y = 6.0 (*Range 0 to 10 *)

(
(10.0-x) * input(XL,X1) |
x * input(XH,X7) |
100.0 * Fanout(X1,X2,X3,X4,X5,X6) |  (* Fanout XL *)
100.0 * Fanout(X7,X8,X9,X10,X11,X12) | (* Fanout XH *)

(10.0-y) * input(YL,X14) |
y * input(YH,X20) |
100.0 * Fanout(X14,X15,X16,X17,X18,X19) |  (* Fanout YL *)
100.0 * Fanout(X20,X21,X22,X23,X24,X25) |  (* Fanout YH *)

100.0 * minimum_gate(X5,X17,X27,X26) | (* Min(XL, YL) *)
100.0 * minimum_gate(X11,X23,X29,X28) | (* Min(XH, YH) *)

100.0 * minimum_gate(X4,X24,X31,X30) | (* Min(XL, YH) *)
100.0 * minimum_gate(X10,X18,X33,X32) | (* Min(XH, YL) *)

(* Max(Min(XL,YL),Min(XH,YH)) *)
100.0 * Fanout(X27,X34,X35,X36,X37,X38) |
100.0 * Fanout(X29,X39,X40,X41,X42,X43) |
100.0 * minimum_gate(X37,X41,X46,X45) |
100.0 * Adder(X36,X42,X44,X47) |
Subtractor(100.0,X44,X46) |  (* The output is FL; <X36 t^ X44> *)


(* Max(Min(XL,YH),Min(XH,YL)) *)
100.0 * Fanout(X31,X48,X49,X50,X51,X52) |
100.0 * Fanout(X33,X53,X54,X55,X56,X57) |
100.0 * minimum_gate(X51,X55,X60,X59) |
100.0 * Adder(X50,X56,X58,X61) |
Subtractor(100.0,X58,X60)   (* The output is FH; <X50 t^ X58> *)
)
```

PUBLICATIONS

Journal Publications

1. George, A. K., and Singh, H., “DNA implementation of fuzzy inference engine: Towards DNA decision-making systems,” *IEEE Transactions on NanoBioscience*, 2017, DOI: 10.1109/TNB.2017.2760821
2. George, A. K., Singh, H., Dattathreya, M. S., and Meitzler, T., “A fuzzy simulation model for military vehicle mobility assessment,” *Advances in Fuzzy Systems, Hindawi*, 2017.
3. George, A. K., and Singh, H., “DNA strand displacement based logic inverter gate design,” *IET Micro and Nano letters*, vol. 12, no. 9, pp 611-614, 2017.
4. George, A. K., and Singh, H., “Three-input majority gate using spatially localized DNA hairpins,” *IET Micro and Nano Letters*, vol. 12, no. 3, pp 143-146, Mar. 2017, DOI: 10.1049/mnl.2016.0535.
5. George, A. K., and Singh, H., “Enzyme-free scalable DNA digital design techniques: A review,” *IEEE Transactions on NanoBioscience*, vol. 15, no. 8, pp 928-938, Dec. 2016, DOI: 10.1109/ TNB.2016.2623218.
6. George, A. K., and Sumathi, P., “Pre-filtered phase locking scheme for multi-component AM-FM signal decomposition,” *Circuits, Systems, and Signal Processing, Springer*, 2017, DOI: 10.1007/s00034-017-0576-8.
7. George, A. K., and Sumathi, P., “Mono-component AM-FM signal decomposition based on discrete second order generalized integrator - phase locked loop,” *Circuits, Systems, and Signal Processing, Springer*, vol. 36, no. 4, pp 1604-1620, 2017, DOI: 10.1007/s00034-016-0380-x

8. George, A. K., Ishak, O. K., Alazzawi, L., Singh, H., Dattathreya, M. S., and Meitzler, T., “Development of a Fuzzy Chip for Mobility Assessment,” *Advances in Fuzzy Systems, Hindawi*, Under review.
9. Vineet V., Sumathi, P., and George, A. K., “A frequency demodulator based on adaptive sampling frequency phase locking scheme for large deviation FM signals,” *Circuits, Systems, and Signal Processing, Springer*, Under review.

Conference Publications

1. George, A. K., and Singh, H., “Design of Computing Circuits using Spatially Localized DNA Majority Logic Gates,” in *proceedings of 2017 IEEE International Conference on Rebooting Computing (ICRC)*, pp 314-320, Nov. 2017.
2. George, A. K., Almatrood, A., and Singh, H., “Design of arithmetic and control cells for a DNA binary processor,” in *proceedings of international conference on computational science and computational intelligence (CSCI 15), IEEE*, pp 7-12, Dec 2015.
3. George, A. K., and Sumathi, P., “Mono-component AM-FM signal decomposition using SRF-PLL,” in *proceedings of international conference on communication and signal processing (ICCSP 14), IEEE*, pp 792-796, April 2014.
4. Almatrood, A., George, A. K., and Singh, H., “On the development of multi-input multi-output nano digital circuits for molecular medicine,” in *proceedings of international conference on computational science and computational intelligence (CSCI 15), IEEE*, pp 873-878, Dec 2015.
5. Kumar, A., George, A. K., Singh, H., Singh, H. P., “Development of a fuzzy chip for predicting the confidence level of soldiers in the army vehicle,” *proceedings of second international conference on innovative trends in electronics engineering (ICITEE2-2016)*, Nov. 2016.

6. Sharma, A., Kamthan, S., George, A.K., Almatrood, A., Singh, H., and Pal, H., "On development of chip to control laser time for cell-selective arrhythmia ablation of heart," in *proceedings of international conference on innovative trends in electronics engineering (ICITEE-2016)*, Jan 2016.
7. Gowrisankar, S., Almatrood, A., George, A.K., Singh, H., and Singh, H.P., "On the development of arithmetic processors," in *proceedings of international conference on innovative trends in electronics engineering (ICITEE-2016)*, Jan 2016.
8. Twal, R., Singh, H., Almatrood, A., and George, A. K., "On the development of Boolean algebra approach to terminal-electromagnetic-compatibility of networks," in *proceedings of IEEE international symposium on electromagnetic compatibility (EMC 2016)*, pp 67-72, Jul 2016.

Posters Presented

1. George, A. K., and Singh, H., "Enzyme free DNA digital circuit design: A majority logic based approach," *Genomics@Wayne 2016 symposium*, Oct. 28, 2016, Detroit, MI, USA.
2. George, A. K., "A Novel dual rail AND-OR-Majority synthesizing technique for future implantable medical devices using DNA circuits," *IEEE Southeast Michigan section (SEM) humanitarian technology conference*, Jul. 09, 2016, Dearborn, MI, USA.
3. George, A. K., "On the development of digital nano circuits for predicting the severity of asthma," *IEEE Southeast Michigan section (SEM) fall conference*, Nov.17, 2015, U-M Dearborn, MI, USA.
4. George, A. K, Almatrood, A., Twal, R., Ali, O., and Singh, H., "Terminal vulnerability of Internet of things for big data applications in defense," *Big data & business analytics symposium*, March 23-24, 2016, Wayne State University, Detroit, MI, USA.

5. Almatrood, A., George, A. K, Twal, R., Ali, O., and Singh, H., “Reduction of big data by combination of factor analysis and clustering for business applications,” *Big data & business analytics symposium*, March 23-24, 2016, Wayne State University, Detroit, MI, USA.
6. Singh, H., and George, A. K.,“On the development of a functional simulation of bone marrow using fuzzy logic approach and implementation with nanotechnology,” *Tenth Cooley’s Anemia symposium*, Oct.18-22, 2015, Chicago, IL, USA.
7. Nam, D. H., Kamthan, S., George, A. K., Almatrood, A., and Singh, H., “Estimation of equity risk premium using preprocessed correlation based dimension reduction,” *Big data & business analytics symposium*, March 23-24, 2016, Wayne State University, Detroit, MI, USA.
8. Singh, H., Kamthan, S., George, A. K., and Xie, Y.,“An effective way to convert your requirements into integrated circuits (chip design),” *IEEE Southeast Michigan section (SEM) spring conference*, Apr. 28, 2015, U-M Dearborn, MI, USA.

BIBLIOGRAPHY

[1] L. Qian and E. Winfree, “Parallel and scalable computation and spatial dynamics with DNA-based chemical reaction networks on a surface,” in *DNA Computing and Molecular Programming.* Springer, 2014, pp. 114–131.

[2] D. Mo, M. R. Lakin, and D. Stefanovic, “Logic circuits based on molecular spider systems,” *Biosystems*, 2016.

[3] R. Carlson, “The changing economics of DNA synthesis,” *Nature biotechnology*, vol. 27, no. 12, pp. 1091–1094, 2009.

[4] “I’m loving my “genes”,” https://sites.google.com/site/imlovingmygenes/dna-structure, accessed: 2017-10-31.

[5] L. Qian and E. Winfree, “A simple DNA gate motif for synthesizing large-scale circuits,” *Journal of the Royal Society Interface*, p. rsif20100729, 2011.

[6] ——, “Scaling up digital circuit computation with DNA strand displacement cascades,” *Science*, vol. 332, no. 6034, pp. 1196–1201, 2011.

[7] R. A. Muscat, K. Strauss, L. Ceze, and G. Seelig, “DNA-based molecular architecture with spatially localized components,” in *ACM SIGARCH Computer Architecture News*, vol. 41, no. 3. ACM, 2013, pp. 177–188.

[8] G. E. Moore, “Cramming more components onto integrated circuits,” *Proceedings of the IEEE*, vol. 86, no. 1, pp. 82–85, Jan 1998.

[9] “The law that’s not a law,” *IEEE Spectrum*, vol. 52, no. 4, pp. 38–57, April 2015.

[10] M. Fuechsle, J. A. Miwa, S. Mahapatra, H. Ryu, S. Lee, O. Warschkow, L. C. Hollenberg, G. Klimeck, and M. Y. Simmons, “A single-atom transistor,” *Nature Nanotechnology*, vol. 7, no. 4, pp. 242–246, 2012.

[11] J. Martínez-Blanco, C. Nacci, S. C. Erwin, K. Kanisawa, E. Locane, M. Thomas, F. von Oppen, P. W. Brouwer, and S. Fölsch, “Gating a single-molecule transistor with individual atoms,” *Nature Physics*, vol. 11, no. 8, pp. 640–644, 2015.

[12] F. A. Ran, P. D. Hsu, J. Wright, V. Agarwala, D. A. Scott, and F. Zhang, “Genome engineering using the CRISPR-Cas9 system,” *Nature protocols*, vol. 8, no. 11, pp. 2281–2308, 2013.

[13] C. Calude and G. Paun, *Computing with cells and atoms: an introduction to quantum, DNA and membrane computing.* CRC Press, 2000.

[14] C. S. Lent, P. D. Tougaw, W. Porod, and G. H. Bernstein, “Quantum cellular automata,” *Nanotechnology*, vol. 4, no. 1, p. 49, 1993.

[15] P. D. Tougaw and C. S. Lent, “Logical devices implemented using quantum cellular automata,” *Journal of Applied physics*, vol. 75, no. 3, pp. 1818–1825, 1994.

[16] P. L. Gentili, “The fundamental fuzzy logic operators and some complex boolean logic circuits implemented by the chromogenism of a spirooxazine,” *Physical Chemistry Chemical Physics*, vol. 13, no. 45, pp. 20 335–20 344, 2011.

[17] P. Luigi Gentili, “Molecular processors: from qubits to fuzzy logic,” *ChemPhysChem*, vol. 12, no. 4, pp. 739–745, 2011.

[18] Q.-Q. Wu, X.-Y. Duan, and Q.-H. Song, “Resettable multiple-mode molecular arithmetic systems based on spectral properties of 2-quinolin-2-ylmethylene-malonic acids,” *The Journal of Physical Chemistry C*, vol. 115, no. 48, pp. 23 970–23 977, 2011.

[19] P. L. Gentili, V. Horvath, V. K. Vanag, and I. R. Epstein, “Belousov-zhabotinsky” chemical neuron” as a binary and fuzzy logic processor.” *IJUC*, vol. 8, no. 2, pp. 177–192, 2012.

[20] J. D. Watson, F. H. Crick *et al.*, “Molecular structure of nucleic acids,” *Nature*, vol. 171, no. 4356, pp. 737–738, 1953.

[21] J. M. Picuri, B. M. Frezza, and M. R. Ghadiri, “Universal translators for nucleic acid diagnosis,” *Journal of the American Chemical Society*, vol. 131, no. 26, pp. 9368–9377, 2009.

[22] S. M. Douglas, I. Bachelet, and G. M. Church, “A logic-gated nanorobot for targeted transport of molecular payloads,” *Science*, vol. 335, no. 6070, pp. 831–834, 2012.

[23] P. B. Allen, S. A. Arshad, B. Li, X. Chen, and A. D. Ellington, “DNA circuits as amplifiers for the detection of nucleic acids on a paperfluidic platform,” *Lab on a Chip*, vol. 12, no. 16, pp. 2951–2958, 2012.

[24] L. M. Adleman, “Molecular computation of solutions to combinatorial problems,” *Science*, vol. 266, no. 5187, pp. 1021–1024, 1994.

[25] R. J. Liptony, “Using DNA to solve np-complete problems,” 1995.

[26] L. Kari, G. Gloor, and S. Yu, “Using DNA to solve the bounded post correspondence problem,” *Theoretical Computer Science*, vol. 231, no. 2, pp. 193–203, 2000.

[27] L. M. Adleman, “Computing with DNA,” *Scientific american*, vol. 279, no. 8, pp. 34–41, 1998.

[28] Q. Ouyang, P. D. Kaplan, S. Liu, and A. Libchaber, “DNA solution of the maximal clique problem,” *Science*, vol. 278, no. 5337, pp. 446–449, 1997.

[29] J. SantaLucia Jr and D. Hicks, “The thermodynamics of DNA structural motifs,” *Annu. Rev. Biophys. Biomol. Struct.*, vol. 33, pp. 415–440, 2004.

[30] J. Chen and N. C. Seeman, “Synthesis from DNA of a molecule with the connectivity of a cube.” *Nature*, vol. 350, no. 6319, pp. 631–633, 1991.

[31] N. C. Seeman, “Nucleic acid junctions and lattices,” *Journal of theoretical biology*, vol. 99, no. 2, pp. 237–247, 1982.

[32] ——, “Nanomaterials based on DNA,” *Annual review of biochemistry*, vol. 79, p. 65, 2010.

[33] E. Winfree, “Algorithmic self-assembly of DNA,” Ph.D. dissertation, California Institute of Technology, 1998.

[34] B. Yurke, A. J. Turberfield, A. P. Mills, F. C. Simmel, and J. L. Neumann, “A DNA-fuelled molecular machine made of DNA,” *Nature*, vol. 406, no. 6796, pp. 605–608, 2000.

[35] A. K. George, A. Almatrood, and H. Singh, “Design of arithmetic and control cells for a DNA binary processor,” in *2015 International Conference on Computational Science and Computational Intelligence (CSCI)*, 2015, pp. 7–12.

[36] R. Sawlekar, F. Montefusco, V. V. Kulkarni, and D. G. Bates, “Implementing nonlinear feedback controllers using DNA strand displacement reactions,” *IEEE transactions on nanobioscience*, vol. 15, no. 5, pp. 443–454, 2016.

[37] D. Y. Zhang and G. Seelig, “Dynamic DNA nanotechnology using strand-displacement reactions,” *Nature chemistry*, vol. 3, no. 2, pp. 103–113, 2011.

[38] H.-W. Fink and C. Schönenberger, “Electrical conduction through DNA molecules,” *Nature*, vol. 398, no. 6726, pp. 407–410, 1999.

[39] B. Xu, P. Zhang, X. Li, and N. Tao, “Direct conductance measurement of single DNA molecules in aqueous solution,” *Nano letters*, vol. 4, no. 6, pp. 1105–1108, 2004.

[40] G. I. Livshits, A. Stern, D. Rotem, N. Borovok, G. Eidelshtein, A. Migliore, E. Penzo, S. J. Wind, R. Di Felice, S. S. Skourtis *et al.*, “Long-range charge transport in single

g-quadruplex DNA molecules," *Nature nanotechnology*, vol. 9, no. 12, pp. 1040–1046, 2014.

[41] H. Bui, H. Chandran, S. Garg, N. Gopalkrishnan, R. Mokhtar, T. Song, and J. H. Reif, "DNA computing, chapter in computing handbook, volume i: Computer science and software engineering, section 3: Architecture and organization, edited by teofilo f. gonzalez," 2013.

[42] M. S. Meselson and C. M. Radding, "A general model for genetic recombination," *Proceedings of the National Academy of Sciences*, vol. 72, no. 1, pp. 358–361, 1975.

[43] I. G. Panyutin and P. Hsieh, "The kinetics of spontaneous DNA branch migration," *Proceedings of the National Academy of Sciences*, vol. 91, no. 6, pp. 2021–2025, 1994.

[44] L. P. Reynaldo, A. V. Vologodskii, B. P. Neri, and V. I. Lyamichev, "The kinetics of oligonucleotide replacements," *Journal of molecular biology*, vol. 297, no. 2, pp. 511–520, 2000.

[45] H. Chandran, N. Gopalkrishnan, A. Phillips, and J. Reif, "Localized hybridization circuits," in *DNA Computing and Molecular Programming*. Springer, 2011, pp. 64–83.

[46] P. W. Rothemund, "Folding DNA to create nanoscale shapes and patterns," *Nature*, vol. 440, no. 7082, pp. 297–302, 2006.

[47] J. D. Le, Y. Pinto, N. C. Seeman, K. Musier-Forsyth, T. A. Taton, and R. A. Kiehl, "DNA-templated self-assembly of metallic nanocomponent arrays on a surface," *Nano Letters*, vol. 4, no. 12, pp. 2343–2347, 2004.

[48] C. M. Niemeyer, "Self-assembled nanostructures based on DNA: towards the development of nanobiotechnology," *Current opinion in chemical biology*, vol. 4, no. 6, pp. 609–618, 2000.

[49] Y. Ke, Y. Liu, J. Zhang, and H. Yan, “A study of DNA tube formation mechanisms using 4-, 8-, and 12-helix DNA nanostructures,” *Journal of the American Chemical Society*, vol. 128, no. 13, pp. 4414–4421, 2006.

[50] B. Yurke and A. P. Mills Jr, “Using DNA to power nanostructures,” *Genetic Programming and Evolvable Machines*, vol. 4, no. 2, pp. 111–122, 2003.

[51] D. Y. Zhang and E. Winfree, “Control of DNA strand displacement kinetics using toehold exchange,” *Journal of the American Chemical Society*, vol. 131, no. 47, pp. 17 303–17 314, 2009.

[52] A. K. George and H. Singh, “Enzyme-free scalable DNA digital design techniques: A review,” *IEEE Transactions on NanoBioscience*, vol. 15, no. 8, pp. 928–938, 2016.

[53] ——, “Three-input majority gate using spatially localised DNA hairpins,” *Micro & Nano Letters*, vol. 12, no. 3, pp. 143–146, 2017.

[54] ——, “Design of computing circuits using spatially localized DNA majority logic gates,” 2017.

[55] ——, “DNA strand displacement-based logic inverter gate design,” *Micro & Nano Letters*, vol. 12, no. 9, pp. 611–614, 2017.

[56] ——, “DNA implementation of fuzzy inference engine: Towards DNA decision-making systems,” *IEEE Transactions on NanoBioscience*, 2017.

[57] H. Yan, S. H. Park, G. Finkelstein, J. H. Reif, and T. H. LaBean, “DNA-templated self-assembly of protein arrays and highly conductive nanowires,” *Science*, vol. 301, no. 5641, pp. 1882–1884, 2003.

[58] M. N. Stojanovic and D. Stefanovic, “A deoxyribozyme-based molecular automaton,” *Nature biotechnology*, vol. 21, no. 9, pp. 1069–1074, 2003.

[59] I. Willner, B. Shlyahovsky, M. Zayats, and B. Willner, “DNAzymes for sensing, nanobiotechnology and logic gate applications,” *Chemical Society Reviews*, vol. 37, no. 6, pp. 1153–1165, 2008.

[60] T. Miyamoto, S. Razavi, R. DeRose, and T. Inoue, “Synthesizing biomolecule-based boolean logic gates,” *ACS synthetic biology*, vol. 2, no. 2, pp. 72–82, 2012.

[61] D. Y. Zhang, “Towards domain-based sequence design for DNA strand displacement reactions,” in *DNA Computing and Molecular Programming.* Springer, 2010, pp. 162–175.

[62] S. F. Wickham, J. Bath, Y. Katsuda, M. Endo, K. Hidaka, H. Sugiyama, and A. J. Turberfield, “A DNA-based molecular motor that can navigate a network of tracks,” *Nature nanotechnology*, vol. 7, no. 3, pp. 169–173, 2012.

[63] K. Lund, A. J. Manzo, N. Dabby, N. Michelotti, A. Johnson-Buck, J. Nangreave, S. Taylor, R. Pei, M. N. Stojanovic, N. G. Walter *et al.*, “Molecular robots guided by prescriptive landscapes,” *Nature*, vol. 465, no. 7295, pp. 206–210, 2010.

[64] J.-S. Shin and N. A. Pierce, “A synthetic DNA walker for molecular transport,” *Journal of the American Chemical Society*, vol. 126, no. 35, pp. 10 834–10 835, 2004.

[65] F. Dannenberg, M. Kwiatkowska, C. Thachuk, and A. J. Turberfield, “DNA walker circuits: computational potential, design, and verification,” *Natural Computing*, vol. 14, no. 2, pp. 195–211, 2015.

[66] A. Phillips and L. Cardelli, “A programming language for composable DNA circuits,” *Journal of the Royal Society Interface*, vol. 6, no. Suppl 4, pp. S419–S436, 2009.

[67] M. R. Lakin, S. Youssef, L. Cardelli, and A. Phillips, “Abstractions for DNA circuit design,” *Journal of The Royal Society Interface*, vol. 9, no. 68, pp. 470–486, 2012.

[68] M. R. Lakin, S. Youssef, F. Polo, S. Emmott, and A. Phillips, “Visual dsd: a design and

analysis tool for DNA strand displacement systems," *Bioinformatics*, vol. 27, no. 22, pp. 3211–3213, 2011.

[69] M. R. Lakin, R. Petersen, K. E. Gray, and A. Phillips, "Abstract modelling of tethered DNA circuits," in *DNA Computing and Molecular Programming.* Springer, 2014, pp. 132–147.

[70] R. Daniel, J. R. Rubens, R. Sarpeshkar, and T. K. Lu, "Synthetic analog computation in living cells," *Nature*, vol. 497, no. 7451, pp. 619–623, 2013.

[71] A. P. Mills, B. Yurke, and P. M. Platzman, "Article for analog vector algebra computation," *Biosystems*, vol. 52, no. 1, pp. 175–180, 1999.

[72] A. J. Turberfield, J. Mitchell, B. Yurke, A. P. Mills Jr, M. Blakey, and F. C. Simmel, "DNA fuel for free-running nanomachines," *Physical review letters*, vol. 90, no. 11, p. 118102, 2003.

[73] G. Seelig, B. Yurke, and E. Winfree, "Catalyzed relaxation of a metastable DNA fuel," *Journal of the American Chemical Society*, vol. 128, no. 37, pp. 12 211–12 220, 2006.

[74] D. Y. Zhang, A. J. Turberfield, B. Yurke, and E. Winfree, "Engineering entropy-driven reactions and networks catalyzed by DNA," *Science*, vol. 318, no. 5853, pp. 1121–1125, 2007.

[75] R. M. Dirks and N. A. Pierce, "Triggered amplification by hybridization chain reaction," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 101, no. 43, pp. 15 275–15 278, 2004.

[76] P. Yin, H. M. Choi, C. R. Calvert, and N. A. Pierce, "Programming biomolecular self-assembly pathways," *Nature*, vol. 451, no. 7176, pp. 318–322, 2008.

[77] B. Li, A. D. Ellington, and X. Chen, "Rational, modular adaptation of enzyme-free

DNA circuits to multiple detection methods," *Nucleic acids research*, vol. 39, no. 16, pp. e110–e110, 2011.

[78] D. Y. Zhang and G. Seelig, "DNA-based fixed gain amplifiers and linear classifier circuits," *DNA*, vol. 16, pp. 176–186, 2010.

[79] X. Chen, N. Briggs, J. R. McLain, and A. D. Ellington, "Stacking nonenzymatic circuits for high signal gain," *Proceedings of the National Academy of Sciences*, vol. 110, no. 14, pp. 5386–5391, 2013.

[80] C. Jung and A. D. Ellington, "Diagnostic applications of nucleic acid circuits," *Accounts of chemical research*, vol. 47, no. 6, pp. 1825–1835, 2014.

[81] X. Zhang, Y. Wang, Z. Chen, J. Xu, and G. Cui, "Arithmetic computation using self-assembly of DNA tiles: subtraction and division," *Progress in Natural Science*, vol. 19, no. 3, pp. 377–388, 2009.

[82] A. J. Genot, J. Bath, and A. J. Turberfield, "Combinatorial displacement of DNA strands: application to matrix multiplication and weighted sums," *Angewandte Chemie International Edition*, vol. 52, no. 4, pp. 1189–1192, 2013.

[83] W. Li, F. Zhang, H. Yan, and Y. Liu, "DNA based arithmetic function: a half adder based on DNA strand displacement," *Nanoscale*, vol. 8, no. 6, pp. 3775–3784, 2016.

[84] T. Song, S. Garg, R. Mokhtar, H. Bui, and J. Reif, "Analog computation by DNA strand displacement circuits," *ACS synthetic biology*, vol. 5, no. 8, pp. 898–912, 2016.

[85] C. Zou, X. Wei, Q. Zhang, C. Liu, C. Zhou, and Y. Liu, "Four-analog computation based on DNA strand displacement," *ACS Omega*, vol. 2, no. 8, pp. 4143–4160, 2017.

[86] D. Soloveichik, G. Seelig, and E. Winfree, "DNA as a universal substrate for chemical kinetics," *Proceedings of the National Academy of Sciences*, vol. 107, no. 12, pp. 5393–5398, 2010.

[87] K. Oishi and E. Klavins, “Biomolecular implementation of linear i/o systems,” *IET Systems Biology*, vol. 5, no. 4, pp. 252–260, 2011.

[88] T.-Y. Chiu, H.-J. K. Chiang, R.-Y. Huang, J.-H. R. Jiang, and F. Fages, “Synthesizing configurable biochemical implementation of linear systems from their transfer function specifications,” *PloS one*, vol. 10, no. 9, p. e0137442, 2015.

[89] B. Yordanov, J. Kim, R. L. Petersen, A. Shudy, V. V. Kulkarni, and A. Phillips, “Computational design of nucleic acid feedback control circuits,” *ACS synthetic biology*, vol. 3, no. 8, pp. 600–616, 2014.

[90] C. Briat, C. Zechner, and M. Khammash, “Design of a synthetic integral feedback circuit: dynamic analysis and DNA implementation,” *ACS synthetic biology*, vol. 5, no. 10, pp. 1108–1116, 2016.

[91] J. Fern, D. Scalise, A. Cangialosi, D. Howie, L. Potters, and R. Schulman, “DNA strand-displacement timer circuits,” *ACS synthetic biology*, vol. 6, no. 2, pp. 190–193, 2016.

[92] D. Mo, M. R. Lakin, and D. Stefanovic, “Logic circuits based on molecular spider systems,” *Biosystems*, pp. –, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0303264716300259

[93] J. Zhu, L. Zhang, S. Dong, and E. Wang, “Four-way junction-driven DNA strand displacement and its application in building majority logic circuit,” *ACS nano*, vol. 7, no. 11, pp. 10 211–10 217, 2013.

[94] W. Li, Y. Yang, H. Yan, and Y. Liu, “Three-input majority logic gate and multiple input logic circuit based on DNA strand displacement,” *Nano letters*, vol. 13, no. 6, pp. 2980–2988, 2013.

[95] R. Akeela and M. D. Wagh, "A five-input majority gate in quantum-dot cellular automata," in *NSTI Nanotech*, vol. 2, 2011, pp. 978–981.

[96] E. Torelli, M. Marini, S. Palmano, L. Piantanida, C. Polano, A. Scarpellini, M. Lazzarino, and G. Firrao, "A DNA origami nanorobot controlled by nucleic acid hybridization," *Small*, vol. 10, no. 14, pp. 2918–2926, 2014.

[97] J. Fu and H. Yan, "Controlled drug release by a nanorobot," *Nature biotechnology*, vol. 30, no. 5, pp. 407–408, 2012.

[98] K. Navi, S. Sayedsalehi, R. Farazkish, and M. R. Azghadi, "Five-input majority gate, a new device for quantum-dot cellular automata," *Journal of Computational and Theoretical Nanoscience*, vol. 7, no. 8, pp. 1546–1553, 2010.

[99] S. Venkataraman, R. M. Dirks, C. T. Ueda, and N. A. Pierce, "Selective cell death mediated by small conditional RNAs," *Proceedings of the National Academy of Sciences*, vol. 107, no. 39, pp. 16 777–16 782, 2010.

[100] Y. Krishnan and F. C. Simmel, "Nucleic acid based molecular devices," *Angewandte Chemie International Edition*, vol. 50, no. 14, pp. 3124–3156, 2011.

[101] R. Penchovsky and R. R. Breaker, "Computational design and experimental validation of oligonucleotide-sensing allosteric ribozymes," *Nature biotechnology*, vol. 23, no. 11, pp. 1424–1433, 2005.

[102] Y.-J. Chen, N. Dalchau, N. Srinivas, A. Phillips, L. Cardelli, D. Soloveichik, and G. Seelig, "Programmable chemical controllers made from DNA," *Nature nanotechnology*, vol. 8, no. 10, pp. 755–762, 2013.

[103] A. Dance, "Smart drugs: A dose of intelligence," *Nature*, vol. 531, no. 7592, pp. S2–S3, 2016.

[104] X. Yang, Y. Tang, S. D. Mason, J. Chen, and F. Li, "Enzyme-powered three-dimensional DNA nanomachine for DNA walking, payload release, and biosensing," *ACS nano*, vol. 10, no. 2, pp. 2324–2330, 2016.

[105] G. Rau, K. Becker, R. Kaufmann, and H.-J. Zimmermann, "Fuzzy logic and control: principal approach and potential applications in medicine," *Artificial Organs*, vol. 19, no. 1, pp. 105–112, 1995.

[106] S. Barro and R. Marín, *Fuzzy logic in medicine.* Physica, 2013, vol. 83.

[107] M. Mahfouf, M. F. Abbod, and D. A. Linkens, "A survey of fuzzy logic monitoring and control utilisation in medicine," *Artificial intelligence in medicine*, vol. 21, no. 1, pp. 27–42, 2001.

[108] Y. Benenson, B. Gil, U. Ben-Dor, R. Adar, and E. Shapiro, "An autonomous molecular computer for logical control of gene expression," *Nature*, vol. 429, no. 6990, pp. 423–429, 2004.

[109] A. Trybulec and C. Bylinski, "Some properties of real numbers operations: min, max, square, and square root," *Journal of Formalized Mathematics*, vol. 1, no. 198, p. 9, 1989.

[110] L. A. Zadeh, "Fuzzy sets," *Information and control*, vol. 8, no. 3, pp. 338–353, 1965.

[111] J.-S. Jang and C.-T. Sun, "Neuro-fuzzy modeling and control," *Proceedings of the IEEE*, vol. 83, no. 3, pp. 378–406, 1995.

[112] R. E. Bellman and L. A. Zadeh, "Decision-making in a fuzzy environment," *Management science*, vol. 17, no. 4, pp. B–141, 1970.

[113] M. M. Gupta and J. Qi, "Theory of t-norms and fuzzy inference methods," *Fuzzy sets and systems*, vol. 40, no. 3, pp. 431–450, 1991.

[114] A. Abraham, “Rule-based expert systems,” *Handbook of measuring system design*, 2005.

[115] E. H. Mamdani and S. Assilian, “An experiment in linguistic synthesis with a fuzzy logic controller,” *International journal of man-machine studies*, vol. 7, no. 1, pp. 1–13, 1975.

[116] E. H. Mamdani, “Application of fuzzy logic to approximate reasoning using linguistic synthesis,” in *Proceedings of the sixth international symposium on Multiple-valued logic*. IEEE Computer Society Press, 1976, pp. 196–202.

[117] T. Takagi and M. Sugeno, “Derivation of fuzzy control rules from human operator’s control actions,” in *Proceedings of the IFAC symposium on fuzzy information, knowledge representation and decision analysis*, vol. 6. sn, 1983, pp. 55–60.

[118] ——, “Fuzzy identification of systems and its applications to modeling and control,” *IEEE transactions on systems, man, and cybernetics*, no. 1, pp. 116–132, 1985.

[119] W. V. Leekwijck and E. E. Kerre, “Defuzzification: criteria and classification,” *Fuzzy Sets and Systems*, vol. 108, no. 2, pp. 159 – 178, 1999.

[120] R. Carvajal, J. Ramirez-Angulo, and J. Martinez-Heredia, “High-speed high-precision min/max circuits in cmos technology,” *Electronics Letters*, vol. 36, no. 8, pp. 697–699, 2000.

[121] P. L. Gentili, “Boolean and fuzzy logic implemented at the molecular level,” *Chemical physics*, vol. 336, no. 1, pp. 64–73, 2007.

[122] J. N. Zadeh, C. D. Steenberg, J. S. Bois, B. R. Wolfe, M. B. Pierce, A. R. Khan, R. M. Dirks, and N. A. Pierce, “NUPACK: analysis and design of nucleic acid systems,” *Journal of computational chemistry*, vol. 32, no. 1, pp. 170–173, 2011.

[123] K. U. Mir, “A restricted genetic alphabet for DNA computing,” *DNA Based Computers II*, pp. 243–246, 1999.

[124] R. S. Braich, N. Chelyapov, C. Johnson, P. W. Rothemund, and L. Adleman, “Solution of a 20-variable 3-SAT problem on a DNA computer,” *Science*, vol. 296, no. 5567, pp. 499–502, 2002.

[125] D. Faulhammer, A. R. Cukras, R. J. Lipton, and L. F. Landweber, “Molecular computation: RNA solutions to chess problems,” *Proceedings of the National Academy of Sciences*, vol. 97, no. 4, pp. 1385–1389, 2000.

[126] M.-Y. Kao, M. Sanghi, and R. Schweller, “Randomized fast design of short DNA words,” *ACM Transactions on Algorithms (TALG)*, vol. 5, no. 4, p. 43, 2009.

[127] M. R. Lakin, S. Youssef, L. Cardelli, and A. Phillips, “Abstractions for DNA circuit design,” *Journal of The Royal Society Interface*, vol. 9, pp. 470–486, 2012.

[128] A. J. Thubagere, C. Thachuk, J. Berleant, R. F. Johnson, D. A. Ardelean, K. M. Cherry, and L. Qian, “Compiler-aided systematic construction of large-scale DNA strand displacement circuits using unpurified components,” *Nature Communications*, vol. 8, 2017.

[129] R. L. Petersen, M. R. Lakin, and A. Phillips, “A strand graph semantics for DNA-based computation,” *Theoretical computer science*, vol. 632, pp. 43–73, 2016.

[130] J. Von Neumann, “Probabilistic logics and the synthesis of reliable organisms from unreliable components,” *Automata studies*, vol. 34, pp. 43–98, 1956.

[131] A. Avizienis, “Fault-tolerance: The survival attribute of digital systems,” *Proceedings of the IEEE*, vol. 66, no. 10, pp. 1109–1125, Oct 1978.

[132] Y.-J. Chen, B. Groves, R. A. Muscat, and G. Seelig, “DNA nanotechnology from the test tube to the cell,” *Nature nanotechnology*, vol. 10, no. 9, pp. 748–760, 2015.

ABSTRACT

DESIGN OF DNA STRAND DISPLACEMENT BASED CIRCUITS

by

ABY KONAMPURATH GEORGE

May 2018

Advisor: Dr. Harpreet Singh

Major: Electrical and Computer Engineering

Degree: Doctor of Philosophy

DNA is the basic building block of any living organism. DNA is considered a popular candidate for future biological devices and circuits for solving genetic disorders and several other medical problems. With this objective in mind, this research aims at developing novel approaches for the design of DNA based circuits. There are many recent developments in the medical field such as the development of biological nanorobots, SMART drugs, and CRISPR-Cas9 technologies. There is a strong need for circuits that can work with these technologies and devices. DNA is considered a suitable candidate for designing such circuits because of the programmability of the DNA strands, small size, lightweight, known thermodynamics, higher parallelism, and exponentially reducing the cost of synthesizing techniques. The DNA strand displacement operation is useful in developing circuits with DNA strands. The circuit can be either a digital circuit, in which the logic high and logic low states of the DNA strand concentrations are considered as the signal, or it can be an analog circuit in which the concentration of the DNA strands itself will act as the signal.

We developed novel approaches in this research for the design of digital, as well as analog circuits keeping in view of the number of DNA strands required for the circuit design.

Towards this goal in the digital domain, we developed spatially localized DNA majority logic gates and an inverter logic gate that can be used with the existing seesaw based logic gates. The majority logic gates proposed in this research can considerably reduce the number of strands required in the design. The introduction of the logic inverter operation can translate the dual rail circuit architecture into a monorail architecture for the seesaw based logic circuits. It can also reduce the number of unique strands required for the design into approximately half. The reduction in the number of unique strands will consequently reduce the leakage reactions, circuit complexity, and cost associated with the DNA circuits.

The real world biological inputs are analog in nature. If we can use those analog signals directly in the circuits, it can considerably reduce the resources required. Even though analog circuits are highly prone to noise, they are a perfect candidate for performing computations in the resource-limited environments, such as inside the cell. In the analog domain, we are developing a novel fuzzy inference engine using analog circuits such as the minimum gate, maximum gate, and fan-out gates. All the circuits discussed in this research were designed and tested in the Visual DSD software. The biological inputs are inherently fuzzy in nature, hence a fuzzy based system can play a vital role in future decision-making circuits. We hope that our research will be the first step towards realizing these larger goals. The ultimate aim of our research is to develop novel approaches for the design of circuits which can be used with the future biological devices to tackle many medical problems such as genetic disorders.

AUTOBIOGRAPHICAL STATEMENT

Education

PhD	Wayne State University, Detroit, MI, USA	4.0/4.0 (GPA), 2017
M.Tech.	IIT Roorkee, India	9.576/10.0 (GPA), 2014
B.Tech.	University of Calicut, Kerala, India	72.7%, 2007

Personal Details

Full Name	Aby Konampurath George
E-mail Address	aby.100@gmail.com
Date of Birth (mm/dd/yyyy)	01/21/1986
City of Birth	Tripunithura - Kerala
Country of Birth	India

Work Experience

Sep 2014 - Present	Graduate Teaching Assistant, Wayne State University, Detroit, USA
Aug 2013 - Jun 2014	Teaching Assistant, Indian Institute of Technology, Roorkee, India
Nov 2009 - May 2011	Service Engineer, Gulf Incon Controls Intl. LLC, Dubai, UAE
Feb 2009 - Nov 2009	Graduate Engineer, Keltron Controls, Kerala, India
Nov 2007 - Nov 2008	Graduate Apprentice, The FACT Ltd., Kerala, India

Research Interests

- DNA circuit design
- Non-linear control system
- Nano circuit design
- Functional nanotechnology