Minimal Reversible Circuit Synthesis on a DNA Computer manuscript No.
(will be inserted by the editor)

Minimal Reversible Circuit Synthesis on a DNA Computer

Mayukh Sarkar · Prasun Ghosal · Saraju P. Mohanty

Received: date / Accepted: date

Abstract DNA computing has attracted the attention of many researchers in recent years to solve NP-complete problems by over-performing conventional computers with its inherent massively parallelism nature. On the other hand, proper synthesis of reversible circuit is a well researched important problem in terms of computing for present and future days with extremely low power consumption (ideally zero) and inherent reversible nature of reversible logic. Minimum synthesis of a reversible truth table means finding the reversible circuit made up of reversible gates satisfying given truth table with minimum cost. But none of the approaches running on conventional computers can perform minimum synthesis till date without using brute-force DFS tree search. In other words, DFS tree-search approach can not be reasonably implemented for larger circuits as searching a DFS tree is extremely costly on a conventional computer. In this paper, first, a procedure to search a DFS tree in constant time has been proposed to run on a DNA computer. Second, another procedure has also been proposed to apply a reversible gate on a reversible truth table in linear time that can be used to generate a DFS tree library. Finally, the generated library can then be searched in constant time to get minimum reversible circuit given a reversible truth table. An analytical feasibility study has been done and a novel methodology has been developed that can extend enormous scope of future research in this area. To the best of authors' knowl-

M. Sarkar
Indian Institute of Engineering Science and Technology, Shibpur, India

P. Ghosal *(Corresponding Author)*
Indian Institute of Engineering Science and Technology, Shibpur, India
Tel.: +91-33-26684561 x260
Fax: +91-33-26682916
E-mail: p_ghosal@it.iiests.ac.in

S. P. Mohanty
University of North Texas, USA

edge this is a pioneering approach to bridge Reversible Computing and DNA Computing.

Keywords Minimal Reversible Circuit Synthesis · DNA Computing · Natural Computing

1 Introduction

1.1 DNA Computing: Preliminaries

The DNA (Deoxyribonucleic acid) found in the living cells is composed of four bases, viz. Adenine(A), Guanine(G), Thiamine(T), and Cytosine(C). The order of these bases is unique in each individual and determines the unique characteristics of that particular individual. Each base is attached to its neighbour base in the sequence via phosphate bonding. The base, sugar, and the phosphate are together called a nucleotide. Two DNA sequences bond with each other via hydrogen bonding between each Watson-Crick complementary base pairs (A with T, and C with G), forming DNA double helix. Each DNA strand has two ends : 5'-end and 3'-end that determine the polarity of the DNA strand. During the formation of DNA double strand two complementary single strands bond with each other in anti-parallel fashion. Several molecular biological operations can be performed on DNA, discussed in Section 4.

1.2 Power of DNA Computing

The problems are solved on a DNA computer by encoding the problems using A, T, G and C, synthesizing corresponding DNA strands and performing several operations on those strands by methods available in a molecular biology laboratory. The main power of DNA computing over conventional ones are as follows.

1. Massively parallel operation. A single test tube of DNA can contain trillions of DNA strands and all strands respond to the biological operations in parallel.
2. The information density of DNA is huge over silicon. Estimated storage capacity of 2.2 petabytes per gram of DNA has been reported in [11].

1.3 Recent Trends and Challenges

These powers of DNA computer has attracted several researchers over the years. Several problems have been solved on DNA computer with reasonable time complexity, which are otherwise hard on conventional computer, as specified in section 3. In 2002, researchers from Weizmann Institute of Science, Israel, have developed a programmable molecular computing machine composed of enzymes and DNA molecules. A year later, the team advanced one

step further, as in the new device, the single DNA molecule that provides the computer with the input data also provides all the necessary fuel [2]. In 2004, Benenson et. al. [5] described an autonomous bio-molecular computer that logically analyses the levels of messenger RNA species, and in response produces a molecule capable of affecting levels of gene expression. This computer, in theory, would be capable of diagnosing cancer, and producing anti-cancer drug. In 2013, Goldman et. al. [11] encoded computer files totaling 739 kB of hard disk storage and with an estimated Shannon information of 5.2×10^6 bits into a DNA code, synthesized, sequenced and then reconstructed the original data with 100% accuracy. In the same year, bio-engineers at Stanford University created the first biological transistors, named *transcriptor*, using DNA and RNA [3]. On 26^{th} October this year (2014), Israeli scientists, in collaboration with researchers from around the world, have developed DNA strands capable of carrying electrical charges for DNA-based electrical circuits [16] [1]. They have reported reproducible charge transport in guanine-quadruplex(G4) DNA molecules adsorbed on a mica substrate,and have measured currents of tens of picoamperes to more than 100pA in the G4-DNA over distances ranging from tens of nanometres to more than 100nm.

All these latest advancements in the field of DNA computing proves the necessity of more algorithms to get successfully implemented on DNA computer, such that they can be executed in reasonable amount of time.

Another major contender for the replacement of conventional computing is quantum computing, which requires synthesis of reversible logic circuits, as quantum gates are, by default, reversible in nature. DFS searching is one of the very few techniques that can perform exact synthesis of reversible logic circuits [10]. Exact minimum synthesis of reversible circuit is not possible to perform on a conventional computer within reasonable amount of time, as these methods are computationally very expensive and applicable only on very small functions [26] [15].

The proposed work applies the power of DNA computer to perform this important operation. First, a procedure to apply a reversible gate on a truth table in linear time has been proposed. Then, a DFS searching algorithm in constant time has been given. Using the first procedure, a library can be generated in the form of DFS tree searching input that can be stored in only two test tubes. From that library, the minimum circuit can be found in constant time.

1.4 Organization of the paper

Overall organization of rest of the paper is as follows. Section 2 summarizes about the novel contributions of the present paper. Recent related research works and thereby the motivation are discussed in section 3. Section 4 describes different Molecular Biological operations and their mathematical modeling. Representations of reversible truth tables and gates in terms of those fundamental operations are presented in section 5 followed by different types

of reversible logic operations in section 6. Procedure to search a DFS tree in constant time has been proposed in section 7. Section 8 describes the proposed procedure of creating the reversible circuit library. An analytical note regarding the experimental feasibility of the molecular biological operations described throughout has been presented in section 9. Finally section 10 concludes the paper.

2 Contributions of the paper

Synthesis of reversible logic circuits has drawn the attention of many researchers due to its promising aspects in future lossless computing [8] [4] [31]. Several approaches till date have been proposed to synthesize a reversible truth table on a conventional silicon computer [18] [12] [25] [23] [30]. In *Transformation-based method* [18], at each step a gate is added to the circuit to bring the truth table closer to the identity permutation. In *Search-based method* [12], PPRM representation of truth table is used to search a tree to find good circuit. In *Cycle-based method* [25] [23], a permutation is decomposed into disjoint cycles and each cycle is synthesized individually. In *BDD-based method* [30], a binary decision diagram is used to improve sharing between controls of reversible gates. Along with these promising approaches several others have been proposed. But none of these algorithms can provide exact minimum circuits except exhaustive DFS search. But this is inappropriate to apply on a conventional computer for large circuits as searching a DFS tree exhaustively induces larger time complexity issues on a conventional silicon computer. From this perspective proposed approach of synthesizing the reversible circuit using a DNA computer has several advantages as follows.

1. In this paper, an algorithm has been proposed to search a DFS tree in constant time on a DNA computer and hence synthesizing a reversible truth table that will result in proven minimal circuit in constant time. Though the preparation for such search requires large time, but once prepared, it can be kept in only two test tubes for long time with suitable preservatives.
2. A reversible gate on an $n \times n$ truth table cannot be applied in polynomial time on a conventional computer as each of the 2^n rows of the truth table needs to be processed individually by the gate. In this work, an algorithm has also been proposed to run on a DNA computer that can apply a gate on a complete truth table in linear time.
3. Synthesizing a reversible truth table means finding the reversible circuit made up of reversible gates satisfying given truth table. The target has always been among the researchers to get the minimum cost circuit for the reversible truth table given. Though there are few such exact synthesis techniques but they are computationally too expensive to apply on a large truth table as stated above. Though several approaches have been proposed to simulate a reversible gate using DNA, however, to the best of authors' knowledge, no work has still been reported to perform synthesis of a reversible truth table using DNA computing principles. This work may

be considered to be the pioneering one to bridge the reversible computing and DNA computing. Simulating some reversible gates has been the only application of DNA computing techniques for reversible synthesis till now, but an important application like finding minimum reversible synthesis using DNA computer has still been untouched.

3 Related Works

In 1994, Adleman [17] has shown a new way of solving NP-complete problems using DNA by solving Travelling Salesman Problem. Since then, many scientists all over the world have chosen this powerful tool to solve various NP-complete problems. In the very next year (1995), Lipton [21] developed an algorithm to solve the SAT problem on a DNA computer in linear time. In the same year(1995), Boneh et.al. [6] broke DES, a famous encryption method, using DNA. In 1997, Ouyang et. al. [19] solved maximal clique problem given a six vertex graph. A huge achievement was made by Adleman and others in 2002, when they solved 20-variable 3-SAT problem by performing an exhaustive search over 1 million possibilities [22]. Along with these experiments, several other NP-complete problems e.g. Graph Coloring [33], Bin Packing [7] etc. have been solved using DNA computing.

As an application of DNA computing in reversible logic, several implementations of reversible gates have been performed using DNA. Wood et.al. [13] has constructed Fredkin gate using DNA, whose outputs can be used as inputs in other Fredkin gates. Thapliyal et.al. [28] has performed reversible design of complex sequential circuits, such as, reversible latches, flip-flops, registers etc. using the Fredkin gate, hence opening the pathway to design complex reversible sequential circuits using DNA. Song et. al. [27], has simulated Fredkin gate using two DNA models, namely, Sticking System and Enzyme System. Sarker et. al. [24] has implemented Toffoli gate using DNA and has proposed the design of conventional logic gates, such as AND, OR, EX-OR and NOT gates using the DNA-based Toffoli gate.

4 Molecular Biological Operations and Symbolic Representation

Each molecular biological operation on DNA strand filled test tube is considered to be run in $O(1)$ time. In this section, the possible operations on DNA strands have been stated in symbolic forms as functions for the ease of representation of algorithms. Laboratory procedures to perform these operations have been described in section 9.

Single DNA strand G will be written as G and double strand of G and $\overline{G}$ is represented as $\left(\frac{G}{\overline{G}}\right)$, where $\overline{G}$ is the Watson-Crick complement of G. Possible operations on DNA strands and corresponding notations used throughout the paper are summarized as follows.

1. $Merge(T1, T2)$ - Pouring contents of test tube T2 in the test tube T1. T1 contains $T1 \cup T2$.
2. $Copy(T1, T2)$ - Copying contents of test tube T1 in an empty test tube T2.
3. $Anneal(T1)$ - All possible complementary strands S_1S_2 and $\overline{S_1S_2}$ in T1 gets joined together to form double strands $\left(\frac{S_1S_2}{S_1S_2}\right)$. This is performed by cooling up the solution.
4. $Denature(T1)$ - Denatures all double strands $\left(\frac{S_1S_2}{S_1S_2}\right)$ present in T1 in single strands S_1S_2 and $\overline{S_1S_2}$. This is performed by heating up the solution.
5. $Cut(T1, R_1R_2)$ - All double stranded DNA in T1 containing a string $\left(\frac{S_1R_1R_2S_2}{S_1R_1R_2S_2}\right)$ are cut into two parts $\left(\frac{S_1R_1}{S_1R_1}\right)$ and $\left(\frac{R_2S_2}{R_2S_2}\right)$. As not all DNA string can act as restriction site, so some fixed strings R_1R_2, C_1C_2 or K_1K_2 can only be used as restriction site.
6. $Append(T1, strand)$ - This will append the strand *strand* at the end of each strand in T1.
7. $Separate\_DS(T1, ds, ss, T2)$ - T1 is a test tube containing a solution of double-strand and single-strand DNAs. This operation separates the double-strands from T1 and keeps it into T2, while T1 is left with all single strands.
8. $Separate\_Length(T1, l, T2)$ - This operation separates the strands of length l from T1 and put them in T2.
9. $Separate\_String(T1, G, T2)$ - This operation separates all strands containing G as substring and put them in T2.

5 Representation of reversible truth table and gates

Let us assume that we have $n \times n$ truth table. Let us take a language containing strings as follows.

$\Sigma = \{\#, 1, 0, A_1, A_2, ..., A_{2^n}, G_1, G_2, ...G_n, T, R_1, R_2,$
$C_1, C_2, K_1, K_2\}$

Each of the elements of the language is represented by unique DNA string such that no DNA string is prefix of another. Among those R_1R_2, C_1C_2 and K_1K_2 should be designed as a possible restriction site.

Sorting the rows of the truth table in increasing order of the input the b^{th} bit of the r^{th} row is can be represented by a DNA strand $A_rR_1R_2\overline{G_b}C_1C_2\overline{V}$, where V is the value of the bit. All the DNA strands are then put into a test tube. As an example, let us take the following truth table 1 of **3_17**.

This can be represented by the test tube containing 24 DNA strands as follows.

$T_{in} = \{A_1R_1R_2\overline{G_1}C_1C_2\overline{1}, A_1R_1R_2\overline{G_2}C_1C_2\overline{1}, A_1R_1R_2\overline{G_3}C_1C_2\overline{1},$
$A_2R_1R_2\overline{G_1}C_1C_2\overline{0}, A_2R_1R_2\overline{G_2}C_1C_2\overline{0}, A_2R_1R_2\overline{G_3}C_1C_2\overline{0}, \cdots,$
$A_8R_1R_2\overline{G_1}C_1C_2\overline{1}, A_8R_1R_2\overline{G_2}C_1C_2\overline{0}, A_8R_1R_2\overline{G_3}C_1C_2\overline{1}\}.$

Table 1: Truth Table Of **3_17**

a b c	f_1 f_2 f_3
0 0 0	1 1 1
0 0 1	0 0 0
0 1 0	0 0 1
0 1 1	0 1 1
1 0 0	1 0 0
1 0 1	0 1 0
1 1 0	1 1 0
1 1 1	1 0 1

i^{th} bit of a Toffoli gate is represented as the DNA strand $\overline{R_1R_2}G_i\overline{C_1C_2}\alpha_i$, where

$$\alpha_i = \begin{cases} 1, & \text{if } i^{th} \text{ bit is a control bit} \\ T, & \text{if } i^{th} \text{ bit is the target bit} \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

For example, the Toffoli gate *T(b; a,c)* may be represented by the test tube as follows.

$T_{gate} = \{\overline{R_1R_2}G_1\overline{C_1C_2}1, \overline{R_1R_2}G_2\overline{C_1C_2}T, \overline{R_1R_2}G_3\overline{C_1C_2}1\}$

Similarly, the Toffoli gate *T(a; c)* may be represented by the test tube as follows.

$T_{gate} = \{\overline{R_1R_2}G_1\overline{C_1C_2}T, \overline{R_1R_2}G_2\overline{C_1C_2}0, \overline{R_1R_2}G_3\overline{C_1C_2}1\}$

(a) T(b;a,c) (b) T(a;c)

Fig. 1: Circuit notations of two reversible gates $T(b; a, c)$ and $T(a; c)$

6 Reversible Logic Operations

Following sequence of operations is applied by the gate in test tube T_{gate} on the input reversible truth table in test tube T_{in} and gives the output truth table in the test tube T_{out}. An example of applying *T(b; a, c)* on *3_17* can also be

given. To perform this operation, three test tubes T_{rc}, T_{cc} and T_{cseq} containing $\{\overline{R_1R_2}\}$, $\{\overline{C_1C_2}\}$ and $\{\overline{A_1R_1}, \overline{A_2R_1}, \cdots, \overline{A_8R_1}\}$ respectively are required.

(1) Copy(T_{in}, T_{out}) and Copy(T_{gate}, T_{backup})

Now T_{out} contains the same DNA solution as T_{in} and a new test tube T_{backup} is created by copying the contents of T_{gate}.

(2) Merge(T_{in}, T_{gate}) and Anneal(T_{in})

After these operations, the gate control bits $\overline{R_1R_2}G_1\overline{C_1C_2}1$ and $\overline{R_1R_2}G_3\overline{C_1C_2}1$ anneal with the complementary strands of the truth table and form double strands. So T_{in} now contains the double strands as follows. Rest of the solution contain single strands.

$\left(\begin{matrix}A_1R_1R_2\overline{G_1}C_1C_2\overline{1}\\ \overline{R_1R_2}G_1\overline{C_1C_2}1\end{matrix}\right)$, $\left(\begin{matrix}A_1R_1R_2\overline{G_3}C_1C_2\overline{1}\\ \overline{R_1R_2}G_3\overline{C_1C_2}1\end{matrix}\right)$, $\left(\begin{matrix}A_3R_1R_2\overline{G_3}C_1C_2\overline{1}\\ \overline{R_1R_2}G_3\overline{C_1C_2}1\end{matrix}\right)$, $\left(\begin{matrix}A_4R_1R_2\overline{G_3}C_1C_2\overline{1}\\ \overline{R_1R_2}G_3\overline{C_1C_2}1\end{matrix}\right)$, $\left(\begin{matrix}A_5R_1R_2\overline{G_1}C_1C_2\overline{1}\\ \overline{R_1R_2}G_1\overline{C_1C_2}1\end{matrix}\right)$, $\left(\begin{matrix}A_7R_1R_2\overline{G_1}C_1C_2\overline{1}\\ \overline{R_1R_2}G_1\overline{C_1C_2}1\end{matrix}\right)$, $\left(\begin{matrix}A_8R_1R_2\overline{G_1}C_1C_2\overline{1}\\ \overline{R_1R_2}G_1\overline{C_1C_2}1\end{matrix}\right)$, $\left(\begin{matrix}A_8R_1R_2\overline{G_3}C_1C_2\overline{1}\\ \overline{R_1R_2}G_3\overline{C_1C_2}1\end{matrix}\right)$

(3) Separate_DS(T_{in}, ds, ss, T_{temp1}), Denature(T_{temp1}) and Separate_String(T_{temp1}, $\overline{1}$, T_{temp2})

All the above double strands are separated into T_{temp1}, denatured and the strands coming from the truth table are kept into T_{temp2}. So, now T_{temp2} contains the followings.

$\{A_1R_1R_2\overline{G_1}C_1C_2\overline{1}, A_1R_1R_2\overline{G_3}C_1C_2\overline{1}, A_3R_1R_2\overline{G_3}C_1C_2\overline{1}, A_4R_1R_2\overline{G_3}C_1C_2\overline{1}, A_5R_1R_2\overline{G_1}C_1C_2\overline{1}, A_7R_1R_2\overline{G_1}C_1C_2\overline{1}, A_8R_1R_2\overline{G_1}C_1C_2\overline{1}, A_8R_1R_2\overline{G_3}C_1C_2\overline{1}\}$.

All those bits of the truth table that resembles with one of the gate control bits and whose value is 1.

(4) *For each bit i do*

(4a) Separate_String(T_{temp2}, $\overline{G_i}$, T_{Gi})

This will create 3 different test tubes,

$T_{G1} = \{A_1R_1R_2\overline{G_1}C_1C_2\overline{1}, A_5R_1R_2\overline{G_1}C_1C_2\overline{1}, A_7R_1R_2\overline{G_1}C_1C_2\overline{1}, A_8R_1R_2\overline{G_1}C_1C_2\overline{1}\}$,

T_{G2} is empty,

$T_{G3} = \{A_1R_1R_2\overline{G_3}C_1C_2\overline{1}, A_3R_1R_2\overline{G_3}C_1C_2\overline{1}, A_4R_1R_2\overline{G_3}C_1C_2\overline{1}, A_8R_1R_2\overline{G_3}C_1C_2\overline{1}\}$

(4b) Merge(T_{Gi}, T_{rc}), Anneal(T_{Gi}), Cut(T_{Gi}, R_1R_2) and Denature(T_{Gi})

This will cut each strand in each tube between R_1 and R_2. That is, now the test tube contents are like,

$T_{G1} = \{A_1R_1, A_5R_1, A_7R_1, A_8R_1, R_2\overline{G_1}C_1C_2\overline{1}\}$

T_{G2} is empty,

$T_{G1} = \{A_1R_1, A_3R_1, A_4R_1, A_8R_1, R_2\overline{G_3}C_1C_2\overline{1}\}$

(4c) Separate_String(T_{Gi}, R_1, T_{Gi_temp})

Create 3 test tubes,

$T_{G1\_temp} = \{A_1R_1, A_5R_1, A_7R_1, A_8R_1\}$

T_{G2_temp} is empty,
$T_{G3\_temp} = \{A_1R_1, A_3R_1, A_4R_1, A_8R_1\}$

(5) ***For each bit i do***
(5a) ***If*** T_{Gi_temp} ***is not empty***

(5b) Merge(T_{Gi_temp}**,** T_{cseq}**), Anneal(**T_{Gi_temp}**), Separate_DS(**T_{Gi_temp}**, ds, ss,** T_{Gi_ds}**), Denature(**T_{Gi_ds}**), Empty(**T_{cseq}**) and Separate_String(**T_{Gi_ds}**,** $\overline{R_1}$**,** T_{cseq}**)**

During processing T_{G1_temp}, after merging T_{cseq} and annealing, T_{G1_temp} contains the double strands

$$\left(\frac{A_1R_1}{A_1\overline{R_1}}\right), \left(\frac{A_5R_1}{A_5\overline{R_1}}\right), \left(\frac{A_7R_1}{A_7\overline{R_1}}\right) \text{ and } \left(\frac{A_8R_1}{A_8\overline{R_1}}\right)$$

Separating these double strands in T_{G1_ds}, denaturing them and separating the strings containing $\overline{R_1}$ remakes the test tube T_{cseq} as,

$T_{cseq} = \{\overline{A_1R_1}, \overline{A_5R_1}, \overline{A_7R_1}, \overline{A_8R_1}\}$

After performing the similar operation on T_{G3_temp} using new T_{cseq} outputs new T_{cseq} as,

$T_{cseq} = \{\overline{A_1R_1}, \overline{A_8R_1}\}$

The target of this step was to get the complements of all DNA strands in the intersection of all T_{Gi_temp}s. After this step, T_{cseq} contains DNA strands $\overline{A_iR_1}$, where the input truth table has all the control bits set at i^{th} row, and hence the target bit needs to be modified.

(6) Separate_String(T_{backup}**,** T**,** T_{target}**)**

After this operation, T_{target} contains a single DNA strand from the gate that corresponds to the target bit.

$T_{target} = \{\overline{R_1R_2}G_2\overline{C_1C_2}T, \}$

(7) ***For each bit i do***
(7a) Separate_String(T_{target}**,** G_i**,** T_{Gi}**)**

After this operation, only one of T_{Gi} is non-empty and contains the only strand of T_{target}. In this case, T_{G2} is non-empty.

(7b) ***If*** T_{Gi} ***is non-empty***
(7c) Append(T_{cseq}**,** $\overline{R_2}G_i\overline{C_1C_2}$**) and** ***break***

In this case, after this operation,

$T_{cseq} = \{\overline{A_1R_1R_2}G_2\overline{C_1C_2}, \overline{A_8R_1R_2}G_2\overline{C_1C_2}\}$

(8) Merge(T_{out}**,**T_{cseq}**), Anneal(**T_{out}**), Separate_DS(**T_{out}**, ds, ss,** T_{change_ds}**)**

After merging and annealing, T_{out} contains the double strands as follows.

$$\left(\frac{A_1R_1R_2\overline{G_2}C_1C_2\overline{1}}{\overline{A_1R_1R_2}G_2\overline{C_1C_2}}\right) \text{ and } \left(\frac{A_8R_1R_2\overline{G_2}C_1C_2\overline{0}}{\overline{A_8R_1R_2}G_2\overline{C_1C_2}}\right)$$

The double strands are being separated in T_{change_ds} and T_{out} contains the remaining single strands.

(9) Denature(T_{change_ds}) and Separate_String(T_{change_ds}, R_1R_2, T_{change})
Now $T_{change} = \{A_1R_1R_2\overline{G_2}C_1C_2\overline{1}, A_8R_1R_2\overline{G_2}C_1C_2\overline{0}\}$

(10) Separate_String(T_{change}, $\overline{1}$, T_{targ_1}) and Separate_String(T_{change}, $\overline{0}$, T_{targ_0})
Separate the contents of T_{change} in two test tubes, one for which 1 needs to be changed to 0, and for the other, 0 needs to be changed to 1.
$T_{targ\_1} = \{A_1R_1R_2\overline{G_2}C_1C_2\overline{1}\}$
$T_{targ\_0} = \{A_8R_1R_2\overline{G_2}C_1C_2\overline{0}\}$

(11) Merge(T_{targ_1}, T_{cc}), Anneal(T_{targ_1}), Cut(T_{targ_1}, C_1C_2), Denature(T_{targ_1}), Separate_String(T_{targ_1}, C_1, $T_{targ\_1\_temp}$)

(12) Merge(T_{targ_0}, T_{cc}), Anneal(T_{targ_0}), Cut(T_{targ_0}, C_1C_2), Denature(T_{targ_0}), Separate_String(T_{targ_0}, C_1, $T_{targ\_0\_temp}$)
In the above two operations, $\overline{C_1C_2}$ are annealed with the contents of T_{targ_0} and T_{targ_1}. The annealed strands may then be cut between C_1 and C_2. After denaturing the contents are as follows.
$T_{targ\_1} = \{A_1R_1R_2\overline{G_2}C_1, C_2\overline{1}, \overline{C_1}, \overline{C_2}\}$
$T_{targ\_0} = \{A_8R_1R_2\overline{G_2}C_1, C_2\overline{0}, \overline{C_1}, \overline{C_2}\}$
Separating the strings containing C_1 in $T_{targ\_1\_temp}$ and $T_{targ\_0\_temp}$ results in the followings respectively.
$T_{targ\_1\_temp} = \{A_1R_1R_2\overline{G_2}C_1\}$
$T_{targ\_0\_temp} = \{A_8R_1R_2\overline{G_2}C_1\}$

(13) Append($T_{targ\_1\_temp}, C_2\overline{0}$) and Append($T_{targ\_0\_temp}, C_2\overline{1}$)
After these operations,
$T_{targ\_1\_temp} = \{A_1R_1R_2\overline{G_2}C_1C_2\overline{0}\}$
$T_{targ\_0\_temp} = \{A_8R_1R_2\overline{G_2}C_1C_2\overline{1}\}$
Aim of the operations (10), (11), (12) and (13) was to change the bit value of the strands in T_{change}.

(14) Merge($T_{out}, T_{targ\_1\_temp}$) and Merge($T_{out}, T_{targ\_0\_temp}$)
Mix the changed bits in T_{out}. Now T_{out} contains the output truth table.
Clearly, the above procedure is applying a gate on a complete $n \times n$ truth table in O(n) laboratory experiments.

7 Searching a DFS tree in constant time

Let us assume the possible nodes of a DFS tree be V_1, V_2, $\cdots$, V_m. For a reversible circuit synthesis DFS tree may be considered as all possible reversible truth tables. For $n \times n$ reversible circuits, number of possible different nodes

are $2^n!$. The possible paths of the DFS tree can be taken as I_1, I_2, $\cdots$, I_k, where they are all possible CNT gates for a reversible synthesis. For $n \times n$ reversible circuits number of possible different paths are $n2^{n-1}$.

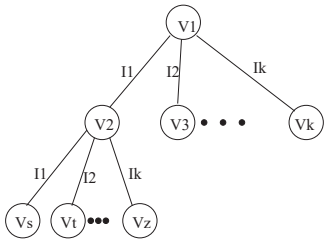

Fig. 2: Representation of an example DFS tree with root node V_1, where an edge with label I_j between nodes V_m and V_n represents that, V_n can be reached from V_m using input I_j. Searching a DFS tree specifies the search of a path, a sequence of inputs, to the desired destination node, say V_r from root V_1

Now let us consider the starting node as V_1 and we need to find the minimum path to reach V_r. Let us prepare two test tubes P and Q as follows.

$P = \{I_1, I_2, \cdots, I_k, V_iV_iI_j, \#V_1, V_r\# : 2 \leq i \leq m \text{ and } i \neq r, 1 \leq j \leq k, \textit{There is a path } I_j \textit{ from node } V_i\}$

$Q = \{\overline{\#}, \overline{V_{i1}I_jV_{i2}} : V_{i2} \textit{ is reachable from } V_{i1} \textit{ via path } I_j\}$

Now let us perform the following operations.

(1) Merge(P, Q) and Denature(P)

All possible DNA strands are grown up in the following way.

$$\begin{pmatrix} \#V_1 \\ \overline{\#} \end{pmatrix} \Rightarrow \begin{pmatrix} \#V_1 \\ \overline{\#V_1I_{p1}V_{q1}} \end{pmatrix} \Rightarrow \begin{pmatrix} \#V_1I_{p1} \\ \overline{\#V_1I_{p1}V_{q1}} \end{pmatrix} \Rightarrow \begin{pmatrix} \#V_1I_{p1}V_{q1}V_{q1}I_{p2} \\ \overline{\#V_1I_{p1}V_{q1}} \end{pmatrix} \Rightarrow$$

$$\begin{pmatrix} \#V_1I_{p1}V_{q1}V_{q1}I_{p2} \\ \overline{\#V_1I_{p1}V_{q1}V_{q1}I_{p2}V_{q2}} \end{pmatrix} \Rightarrow \begin{pmatrix} \#V_1I_{p1}V_{q1}V_{q1}I_{p2}V_{q2}\cdots V_{qm}V_{qm}I_{pm} \\ \overline{\#V_1I_{p1}V_{q1}V_{q1}I_{p2}V_{q2}\cdots V_{qm}I_{pm}V_r} \end{pmatrix}$$

$$\Rightarrow \begin{pmatrix} \#V_1I_{p1}V_{q1}V_{q1}I_{p2}V_{q2}\cdots V_{qm}V_{qm}I_{pm}V_r\# \\ \overline{\#V_1I_{p1}V_{q1}V_{q1}I_{p2}V_{q2}\cdots V_{qm}I_{pm}V_r} \end{pmatrix} \Rightarrow \begin{pmatrix} \#V_1I_{p1}V_{q1}V_{q1}I_{p2}V_{q2}\cdots V_{qm}V_{qm}I_{pm}V_r\# \\ \overline{\#V_1I_{p1}V_{q1}V_{q1}I_{p2}V_{q2}\cdots V_{qm}I_{pm}V_r\#} \end{pmatrix}$$

There may be many other possible DNA double strands. After denaturation all possible paths from V_1 to V_r are in the solution along with many other different DNA single strands. The paths from V_1 to V_r starts with $\#V_1$ and ends with $V_r\#$.

(2) Separate_String(P, $\#V_1$, T) and Separate_String(T, $V_r\#$, T_{sol})

After these operations, T_{sol} contains all and only all possible paths from V_1 to V_r.

(3) Separate_Length(T_{sol}, $< minimum >$, T_{min})

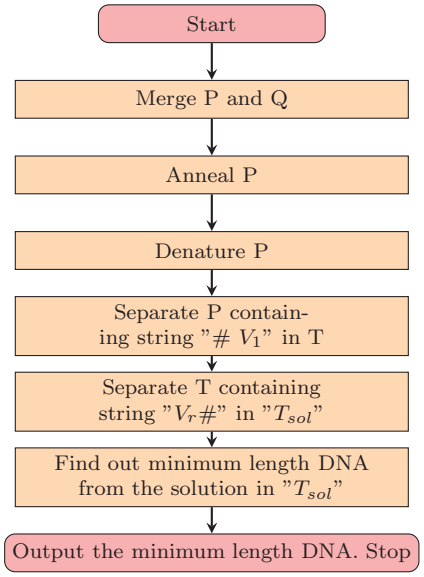

Fig. 3: Flowchart for searching DFS tree

Perform Gel Electrophoresis to separate the band of DNA strand that has the minimum length. T_{min} contains the desired minimum path from V_1 to V_r.

8 Creating Reversible Circuit Library

In the section 6 truth tables and gates are represented by combination of DNA strands in separate solutions in separate test tubes. To generate a library, the truth table can be represented by a single strand containing all DNA strands of section 6 joined by some restriction site other than R_1R_2 and C_1C_2, say K_1K_2. So, the DNA strand representing the truth table 1 of 3_17 in the example of the section 5 is as follows.

$A_1R_1R_2\overline{G_1}C_1C_2\overline{1}K_1K_2A_1R_1R_2\overline{G_2}C_1C_2\overline{1}K_1K_2A_1R_1R_2\overline{G_3}C_1C_2\overline{1}K_1K_2$
$A_2R_1R_2\overline{G_1}C_1C_2\overline{0}K_1K_2A_2R_1R_2\overline{G_2}C_1C_2\overline{0}K_1K_2A_2R_1R_2\overline{G_3}C_1C_2\overline{0}K_1K_2$
$\cdots A_8R_1R_2\overline{G_1}C_1C_2\overline{1}K_1K_2A_8R_1R_2\overline{G_2}C_1C_2\overline{0}K_1K_2A_8R_1R_2\overline{G_3}C_1C_2\overline{1}.$

Similarly, a gate may be represented by a single strand containing all DNA strands of section 6 representing the gate joined by K_1K_2. For example, the DNA strand representing *T(b; a, c)* may be

$\overline{R_1R_2}G_1\overline{C_1C_2}1K_1K_2\overline{R_1R_2}G_2\overline{C_1C_2}TK_1K_2\overline{R_1R_2}G_3\overline{C_1C_2}1$

This represents each truth table by some node V_i and each gate by some path I_k in the DFS tree. Cut the truth table and gate after merging and annealing with $\overline{K_1K_2}$. After cutting, two separate solutions of truth table and gate may be found like in section 6. Apply the linear operation to apply the gate on the truth table. Merge the output truth table by K_1K_2 and let the output truth table be V_j.

Generate all such relationships of V_i and I_k as in the section 7. This will be the reversible circuit library. The library is composed of two test tubes,

$P = \{I_1, I_2, \cdots, I_k, V_iV_iI_j, \#V_1 : 2 \leq i \leq m\ , 1 \leq j \leq k\}$

$Q = \{\overline{\#}, \overline{V_{i1}I_jV_{i2}} : \textit{Applying gate } I_j \textit{ on } V_{i1} \textit{ results in } V_{i2}\}$

Here V_1 is the node representing the identity truth table $(0, 1, 2, 3, \cdots)$.

This can be permanently kept as a database. This database can also be built using a simulation of DNA computer on a conventional computer and the output of such simulation can be fed to a DNA synthesizer to build the required DNA molecules comprising the database. This reduces the manual work only to feeding the required DNA sequences to the synthesizer. A simulator have been implemented in Java. But though such simulators reduce the manual work, as the implementations run on conventional machines, they cannot run with the desired time complexity. As for example, separating a DNA solution based on the presence of a particular string. Although the biological operation is O(1) the simulation requires to search the set following some good string searching algorithm that cannot be done in constant time on conventional computers. As a result, the algorithm proposed to apply gate on reversible truth table also runs in super-polynomial time on the simulator.

From the library, to get the circuit of some truth table T, denote V_r as the DNA strand representing T. Perform the following operations,

(1) Separate_String(P, V_rV_r, T_{temp}) and Discard(T_{temp})

(2) Merge(P, T_r) where T_r contains only the strand $V_r\#$.

These two operations convert P into the form as per the section 7, where V_r is the desired node to reach. Then perform the constant time DFS tree search to find the minimum path from V_1 to V_r. Decode the path to result the minimum circuit synthesizing the truth table T.

9 Experimental Feasibility

The molecular biological operations being used in the work are already proven feasible operations on DNA solutions. A theoretical computational model based on the RDNA model [20] have been assumed here with some additional operations, such as *Append* and *Separate_DS*. Fujiwara et.al. [9] uses the RDNA theoretical model to perform arithmetic and logic operations with DNA strands. Xiao et.al. [32] has used this mathematical model to perform DNA representation of elements in $\{0, 1\}^n$. Tsai et.al. [29] has used this model to construct parallel adder.

The operations comprising the theoretical model used in this work, can be performed practically in a biomolecular laboratory as follows.

Merge : Two test tubes T_1 and T_2 containing DNA solutions can be merged by simply pouring the contents of T_2 in T_1. This leaves T_2 empty, so

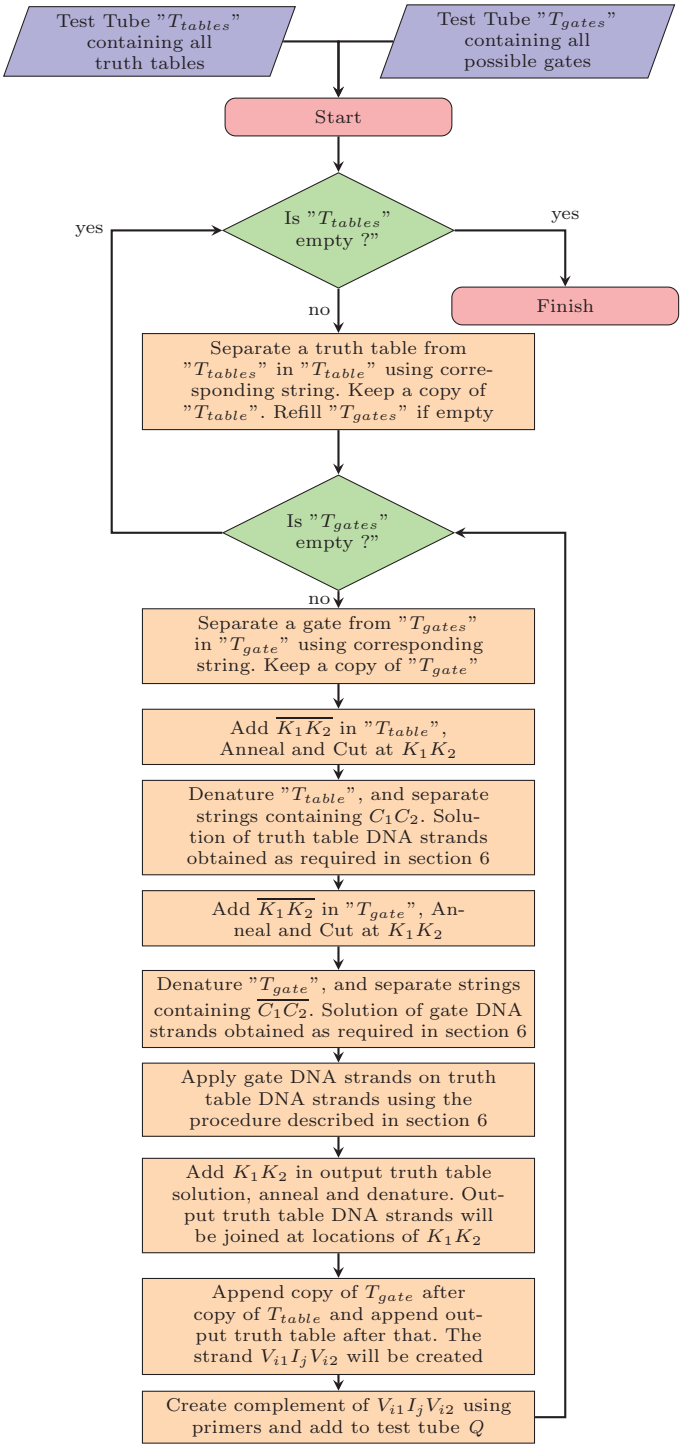

Fig. 4: Flowchart representing detailed step-by-step procedure for the creation of test tube Q

a backup of T_2 can be made by copying, before merging and after merge, the backup can be poured back in T_2.

Copy : The copy of DNA solution residing in a test tube T_2 can be made in a previously empty test tube T_1 using *Polymerase Chain Reaction*.

Anneal : If a test tube containing DNA strands is cooled down, in the course of the normal, random molecular motion, the complementary strands come closer to each other and stick tightly to form DNA double strands.

Denature : A tube of DNA strands dissolved into water, when heated up at a suitably high temperature, the hydrogen bonds between the strands are broken to separate the double strands into two single strands.

Cut (Cleavage) : DNA double strands can be cut at or near specific nucleotide sequence using *restriction enzyme* found in bacteria. Such nucleotide sequence is called *recognition site*. Each restriction enzyme can identify different recognition site in DNA, and hence not all nucleotide sequence can be used as recognition site. As for example, cutting using $EcoRI$, $BamHI$ etc. enzymes produce sticky ends, whereas, $SmaI$, $PvuII$ etc. produce blunt ends.

Append : Appending a DNA strand at the end of another DNA strand is a trivial operation, that can be performed by a specific enzyme, *DNA ligase*, that joins two DNA strands by catalysing the formation of phosphodiester bond.

Separate_DS : Double strands can be separated from the single strands in a DNA solution by the procedure proposed in [14]. Graphene oxide (GO) adsorbs single stranded DNA more quickly than the double-stranded ones. This property along with the centrifugation operation can be used to separate the double stranded and single stranded molecules.

Separate_Length : DNA strands in a solution can be separated by length using Gel Electrophoresis. DNA strands of different lengths move through the gel in different speeds, and hence can be separated by separating the band of the desired length DNA strands.

Separate_String : DNA strands in a solution can be separated, based on the presence of a particular string, by magnetic bead separation procedure, where a magnetic bead containing the Watson-Crick complement of of the string, is hold into the solution and DNA strands containing the string are attracted towards the bead.

10 Conclusion

On a conventional computer, it is not possible to apply a reversible gate on a reversible truth table in polynomial time. In this paper, a procedure has been proposed that can be used to do the same on a DNA computer in linear time. A procedure also has been proposed to search a DFS tree in constant time provided that the required relationships between the nodes and the paths are ready at hand. Based on the two approaches, the minimum synthesis of a reversible circuit is being performed. Though the generation of the library is a long manual process, but once designed, can be kept in only two test tubes as a database. This library can be used at any time to get the minimum reversible circuit of any truth table at constant time. To the best of authors' knowledge this is the pioneering approach of it's kind to bridge the gap between Reversible and DNA Computing and thus generates an avenue of future research directions for succesful application of DNA Computing to design future generations of Reversible Circuits in the post CMOS era.

References

1. Israeli scientists achieve breakthrough in dna computing. http://www.haaretz.com/life/science-medicine/1.623755. Accessed: 2014-11-05
2. Computer made from DNA and enzymes (2003). URL http://news.nationalgeographic.com/news/2003/02/0224_030224_DNAcomputer.html
3. Stanford creates biological transistors, the final step towards computers inside living cells (2013). URL http://www.extremetech.com/extreme/152074-stanford-creates-biological-transistors-the-final-step-towards-computers-inside-living-cells
4. Al-Rabadi, A.N.: Reversible logic synthesis: from fundamentals to quantum computing. Springer Science & Business Media (2004)
5. Benenson, Y., Gil, B., Ben-Dor, U., Adar, R., Shapiro, E.: An autonomous molecular computer for logical control of gene expression. Nature **429**, 423–429 (2004)
6. Boneh, D., Dunworth, C., Lipton, R.J.: Breaking DES using a molecular computer. In: DIMACS workshop on DNA computing (1995)
7. C.A.A.Sanches, N.Y.Soma: A polynomial-time DNA computing solution for the bin-packing problem. Applied Mathematics and Computation **215**, 2055–2062 (2009)
8. De Vos, A.: Reversible computing: fundamentals, quantum computing, and applications. John Wiley & Sons (2011)
9. Fujiwara, A., Matsumoto, K., Chen, W.: Addressable procedures for logic and arithmetic operations with DNA strands. In: International Parallel and Distributed Processing Symposium, 2003 (2003)
10. Garey, M.R., Johnson, D.S.: Computers and intractability: a guide to np-completeness (1979)
11. Goldman, N., Bertone, P., Chen, S., Dessimoz, C., LeProust, E.M., Sipos, B., Birney, E.: Towards practical, high-capacity, low-maintenance information storage in synthesized DNA. Nature **494**, 77–80 (2013)
12. Gupta, P., Agrawal, A., Jha, N.: An algorithm for synthesis of reversible logic circuits. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **25**(11), 2317–2330 (2006)
13. Harlan Wood, D., Chen, J.: Fredkin gate circuits via recombination enzymes. In: Congress on Evolutionary Computation, 2004. CEC2004., vol. 2, pp. 1896–1900 Vol.2 (2004)

14. Huang, P.J.J., Liu, J.: Separation of short single- and double-stranded DNA based on their adsorption kinetics difference on graphene oxide. Nanomaterials **3**(2), 221–228 (2013)
15. Li, Z., Chen, H., Yang, G., Liu, W.: Efficient algorithms for optimal 4-bit reversible logic system synthesis. Journal of Applied Mathematics **2013** (2013)
16. Livshits, G.I., Stern, A., Rotem, D., Borovok, N., Eidelshtein, G., Migliore, A., Penzo, E., Wind, S.J., Di Felice, R., Skourtis, S.S., Cuevas, J.C., Gurevich, L., Kotlyar, A.B., Porath, D.: Long-range charge transport in single G-quadruplex DNA molecules. Nature Nanotechnology **advance online publication** (2014). URL `http://dx.doi.org/10.1038/nnano.2014.246`
17. L.M.Adleman: Molecular computation of solutions to combinatorial problems. Science **266**, 1021–1024 (1994)
18. Miller, D., Maslov, D., Dueck, G.: A transformation based algorithm for reversible logic synthesis. In: Proc. Design Automation Conference, pp. 318–323 (2003)
19. Q.Ouyang, P.D.Kaplan, S.Liu, A.Libchaber: DNA solution of the maximal clique problem. Science **278**, 446–449 (1997)
20. Reif, J.H.: Parallel biomolecular computation: Models and simulations. Algorithmica **25**, 21,322–3 (1995)
21. R.J.Lipton: DNA solution of hard computational problems. Science **268**, 542–545 (1995)
22. R.S.Braich, N.Chelyapov, C.Johnson, P.W.K.Rothemund, L.Adleman: Solution of a 20-variable 3-SAT problem on a DNA computer. Science **296**, 499–503 (2002)
23. Saeedi, M., Zamani, M.S., Sedighi, M., Sasanian, Z.: Reversible circuit synthesis using a cycle-based approach. J. Emerg. Technol. Comput. Syst. **6**(4), 13:1–13:26 (2010)
24. Sarker, A., Ahmed, T., Rashid, S., Anwar, S., Jaman, L., Tara, N., Alam, M., Babu, H.: Realization of reversible logic in dna computing. In: 2011 IEEE 11th International Conference on Bioinformatics and Bioengineering (BIBE), pp. 261–265 (2011)
25. Shende, V., Prasad, A., Markov, I., Hayes, J.: Synthesis of reversible logic circuits. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **22**(6), 710–722 (2003)
26. Shende, V.V., Prasad, A.K., Markov, I.L., Hayes, J.P.: Reversible logic circuit synthesis. In: Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design, pp. 353–360. ACM (2002)
27. Song, T., Wang, S., Wang, X.: The design of reversible gate and reversible sequential circuit based on DNA computing. In: 3rd International Conference on Intelligent System and Knowledge Engineering, 2008. ISKE 2008, vol. 1, pp. 114–118 (2008)
28. Thapliyal, H., Srinivas, M.B.: An extension to DNA based fredkin gate circuits: Design of reversible sequential circuits using fredkin gates. CoRR **abs/cs/0603092** (2006). URL `http://arxiv.org/abs/cs/0603092`
29. Tsai, S., Chang, W.L., Ho, S.H.: Constructing bio-molecular parallel adder with basic logic operations in the Adleman-Liption model. In: International Conference on Convergence Information Technology, 2007, pp. 925–930 (2007)
30. Wille, R., Drechsler, R.: BDD-based synthesis of reversible logic for large functions. In: Design Automation Conference, pp. 270–275 (2009)
31. Wille, R., Drechsler, R.: Towards a design flow for reversible logic. Springer Science & Business Media (2010)
32. Xiao, D., Li, W., Yu, J., Zhang, X., Zhang, Z., He, L.: Procedures for a dynamical system on $\{0,1\}^n$ with DNA molecules. Biosystems **84**(3), 207 – 216 (2006)
33. Y.Liu, J.Xu, L.Pan, S.Wang: DNA solution of a graph coloring problem. J Chem Inf Comput Sci. **42**, 524–528 (2002)