The DNA-Based Algorithms of Implementing Arithmetical Operations of Complex Vectors on a Biological Computer

Weng-Long Chang*, Athanasios V. Vasilakos, and Michael Shan-HuiHo

***Abstract*—Here we show that arithmetical operations of complex vectors can be implemented by means of the proposed DNA-based algorithms.**

***Index Terms*—DNA-based algorithms, molecular computing.**

I. INTRODUCTION

FROM [1], [2], MANY biological algorithms of solving different problems were introduced. From [3], molecular algorithms of implementing biomolecular databases were proposed. From [4], quantum algorithms of implementing biomolecular solutions of the vertex cover problem were proposed. Our major contributions in this journal paper are as follows.

- We show that addition of complex vectors and closure axioms of addition of complex vector can be implemented by means of biological operations and DNA strands.

II. THE FORMAL MODEL OF COMPUTATION

DNA (deoxyribonucleic acid) in [1], [2] includes polymer chains which are commonly regarded as DNA strands. Each strand may be made of a sequence of nucleotides, or bases, attached to a sugar-phosphate "backbone." The four DNA nucleotides are adenine, guanine, cytosine, and thymine, commonly abbreviated to A, G, C, and T respectively. The following biomolecular operations will be applied to develop molecular algorithms of implementing arithmetical operations of complex vectors. Their implementation can be found in [1].

Definition 2-1: Given set $X = \{x_n x_{n-1} \ldots x_2 x_1 | \forall x_d \in \{0,1\}$ for $1 \leq d \leq n\}$ and a bit x_j, the biomolecular operation "Append-Head" appends x_j onto the head of every element in set X. The formal representation is written as $\text{Append} - \text{Head}(X, x_j) = \{x_j x_n x_{n-1} \ldots x_2 x_1 | \forall x_d \in \{0,1\}$ for $1 \leq d \leq n$ and $x_j \in \{0,1\}\}$.

Manuscript received May 13, 2015; revised July 31, 2015; accepted September 29, 2015. Date of publication October 26, 2015; date of current version January 07, 2016. This work was supported by National Science Foundation of Republic of China under Grants 103-2622-E-151-013-CC3. *Asterisk indicates corresponding author.*

*W.-L. Chang is with the Department of Computer Science and Information Engineering, National Kaohsiung University of Applied Sciences, Kaohsiung 807, Taiwan (e-mail: changwl@cc.kuas.edu.tw).

A. V. Vasilakos is with the Department of Computer Science, National Technical University of Athens, Greece (e-mail: vasilako@ath.forthnet.gr).

M. Ho is with Computer Center and Institute of Electrical Engineering, National Taipei University, Taiwan (e-mail: MHoInCerritos@yahoo.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNB.2015.2492568

Definition 2-2: Given set $X = \{x_n x_{n-1} \ldots x_2 x_1 | \forall x_d \in \{0,1\}$ for $1 \leq d \leq n\}$ and a bit x_j, the biomolecular operation, "Append-Tail," appends x_j onto the end of every element in set X. The formal representation is written as $\text{Append} - \text{Tail}(X, x_j) = \{x_n x_{n-1} \ldots x_2 x_1 x_j | \forall x_d \in \{0,1\}$ for $1 \leq d \leq n$ and $x_j \in \{0,1\}\}$.

Definition 2-3: Given set $X = \{x_n x_{n-1} \ldots x_2 x_1 | \forall x_d \in \{0,1\}$ for $1 \leq d \leq n\}$, the biomolecular operation "$\text{Discard}(X)$" sets X to be an empty set and can be represented as "$X = \varnothing$."

Definition 2-4: Given set $X = \{x_n x_{n-1} \ldots x_2 x_1 | \forall x_d \in \{0,1\}$ for $1 \leq d \leq n\}$, the biomolecular operation "$\text{Amplify}(X, \{X_i\})$" creates a number of identical copies X_i of set X, and then "$\text{Discard}(X)$."

Definition 2-5: Given set $X = \{x_n x_{n-1} \ldots x_2 x_1 | \forall x_d \in \{0,1\}$ for $1 \leq d \leq n\}$ and a bit, x_j, if the value of x_j is equal to one, then the biomolecular *extract* operation creates two new sets, $+(X, {x_j}^1) = \{x_n x_{n-1} \ldots {x_j}^1 \ldots x_2 x_1 | \forall x_d \in \{0,1\}$ for $1 \leq d \neq j \leq n\}$ and $-(X, {x_j}^1) = \{x_n x_{n-1} \ldots {x_j}^0 \ldots x_2 x_1 | \forall x_d \in \{0,1\}$ for $1 \leq d \neq j \leq n\}$. Otherwise, it produces another two new sets, $+(X, {x_j}^0) = \{x_n x_{n-1} \ldots {x_j}^0 \ldots x_2 x_1 | \forall x_d \in \{0,1\}$ for $1 \leq d \neq j \leq n\}$ and $-(X, {x_j}^0) = \{x_n x_{n-1} \ldots {x_j}^1 \ldots x_2 x_1 | \forall x_d \in \{0,1\}$ for $1 \leq d \neq j \leq n\}$.

Definition 2-6: Given m sets $X_1 \ldots X_m$, the biomolecular *merge* operation, $\cup(X_1, \ldots, X_m) = X_1 \cup \ldots \cup X_m$.

Definition 2-7: Given set $X = \{x_n x_{n-1} \ldots x_2 x_1 | \forall x_d \in \{0,1\}$ for $1 \leq d \leq n\}$, the biomolecular operation "$\text{Detect}(X)$" returns *true* if $X \neq \varnothing$. Otherwise, it returns *false*.

Definition 2-8: Given set $X = \{x_n x_{n-1} \ldots x_2 x_1 | \forall x_d \in \{0,1\}$ for $1 \leq d \leq n\}$, the biomolecular operation "$\text{Read}(X)$" describes any element in X. Even if X contains many different elements, the biomolecular operation can give an explicit description of exactly one of them.

III. THE DNA-BASED ALGORITHMS OF IMPLEMENTING ARITHMETICAL OPERATIONS OF COMPLEX VECTORS

The following subsections are applied to show how arithmetical operations of complex vectors are implemented by the proposed DNA-based algorithms that are made of biological operations and DNA sequences.

A. The DNA-Based Algorithms of Implementing Addition of Complex Vectors and Closure Axioms of Addition

It is assumed that a nonempty set V is equal to $\{[a_{1,1} + b_{1,1}i \ \ldots \ a_{1,p} + b_{1,p}i]_{1 \times p} | \forall a_{1,k}$ and $b_{1,k}$ are real numbers for $1 \leq k \leq p$, and $i = \sqrt{-1}\}$. It is supposed that the values of $a_{1,k}$ and $b_{1,k}$ in V can be subsequently represented as two *signed* binary number, $u_{1,k,l+1} u_{1,k,l} \ldots u_{1,k,1}$ and $v_{1,k,l+1} v_{1,k,l} \ldots v_{1,k,1}$ for $1 \leq k \leq p$. ${u_{1,k,l+1}}^0$ and ${v_{1,k,l+1}}^0$ represent positive sign and

${u_{1,k,l+1}}^1$ and ${v_{1,k,l+1}}^1$ represent negative sign. It is assumed for $1 \le k \le p$ that $u_{1,k,l} \ldots u_{1,k,s+1}$ and $v_{1,k,l} \ldots v_{1,k,s+1}$ are their integer, and $u_{1,k,s} \ldots u_{1,k,1}$ and $v_{1,k,s} \ldots v_{1,k,1}$ are their fraction. From [1], [2], for every bit $u_{1,k,j}$ and $v_{1,k,j}$ $1 \le k \le p$ and $1 \le j \le l+1$, two *distinct* DNA sequences are designed to encode them. It is assumed that ${u_{1,k,j}}^1$ and ${v_{1,k,j}}^1$ denote their value to be 1, ${u_{1,k,j}}^0$ and ${v_{1,k,j}}^0$ defines their value to be 0, and $u_{1,k,j}$ and $v_{1,k,j}$ defines their value to be 0 or 1.

It is supposed for $1 \le k \le p$ that two binary numbers $y_{k,l+1}, y_{k,l}, \ldots y_{k,1}$ and $i_{k,l+1}, i_{k,l}, \ldots i_{k,1}$ represent, respectively, the sum of the real part and the imaginary part to the kth element in α and β that are any element in V. ${y_{k,l+1}}^0$ and ${i_{k,l+1}}^0$ represent positive sign, and ${y_{k,l+1}}^1$ and ${i_{k,l+1}}^1$ represent negative sign. It is assumed that ${y_{k,j}}^1$ and ${i_{k,j}}^1$ denotes their value to be 1, and ${y_{k,j}}^0$ and ${i_{k,j}}^0$ defines their value to be 0. The following DNA-based algorithm, **Algorithm 3-1**, is used to implement addition of complex vectors and closure axioms of addition. In **Algorithm 3-1**, the initial state of each tube is set to an empty tube.

Algorithm 3-1: Implement addition of complex vectors and closure axioms of addition.

(1) **Init**(T_{10}).
(2) **InitialValue**(T_0).
(3) **For** $k = p$ **down to** 1
  (3a) **ImaginaryParallelComparator**$(T_0, T_{0,0}, T_{1,1}, {T_{0,1}}^{>=}, {T_{0,1}}^{<}, {T_{1,0}}^{>=}, {T_{1,0}}^{<}, k, k)$.
  (3b) **ImaginaryBinaryParallelAdder**$(T_{0,0}, T_{1,1}, k)$.
  (3c) **ImaginaryBinaryParallelSubtractorGE**$({T_{0,1}}^{>=}, {T_{1,0}}^{>=}, k)$.
  (3d) **ImaginaryBinaryParallelSubtractorLT**$({T_{0,1}}^{<}, {T_{1,0}}^{<}, k)$.
  (3e) $T_0 = \cup(T_{0,0}, T_{1,1}, {T_{0,1}}^{>=}, {T_{0,1}}^{<}, {T_{1,0}}^{>=}, {T_{1,0}}^{<})$.
  (3f) **RealParallelComparator**$(T_0, T_{0,0}, T_{1,1}, {T_{0,1}}^{>=}, {T_{0,1}}^{<}, {T_{1,0}}^{>=}, {T_{1,0}}^{<}, k, k)$.
  (3g) **RealBinaryParallelAdder**$(T_{0,0}, T_{1,1}, k)$.
  (3h) **RealBinaryParallelSubtractorGE**$({T_{0,1}}^{>=}, {T_{1,0}}^{>=}, k)$.
  (3i) **RealBinaryParallelSubtractorLT**$({T_{0,1}}^{<}, {T_{1,0}}^{<}, k)$.
  (3j) $T_0 = \cup(T_{0,0}, T_{1,1}, {T_{0,1}}^{>=}, {T_{0,1}}^{<}, {T_{1,0}}^{>=}, {T_{1,0}}^{<})$.
EndFor
(4) **For** $k = 1$ to p
  (5) **For** $j = 1$ to $l+1$
    (5a) $T_{20} = +(T_0, {y_{k,j}}^1)$ and $T_{50} = -(T_0, {y_{k,j}}^1)$.
    (5b) $T_{21} = +(T_{20}, {i_{k,j}}^1)$ and $T_{22} = -(T_{20}, {i_{k,j}}^1)$.
    (5c) $T_{51} = +(T_{50}, {i_{k,j}}^1)$ and $T_{52} = -(T_{50}, {i_{k,j}}^1)$.
    (5d) $T_{30} = +(T_{10}, {u_{1,k,j}}^1)$ and $T_{60} = -(T_{10}, {u_{1,k,j}}^1)$.
    (5e) $T_{31} = +(T_{30}, {v_{1,k,j}}^1)$ and $T_{32} = -(T_{30}, {v_{1,k,j}}^1)$.
    (5f) $T_{61} = +(T_{60}, {v_{1,k,j}}^1)$ and $T_{62} = -(T_{60}, {v_{1,k,j}}^1)$.
    (5g) **If** (Detect(T_{21}) = "yes") **then**
      (5h) Discard$(T_{32}, T_{61}, T_{62}, T_{22}, T_{51}, T_{52})$.
      (5i) $T_0 = \cup(T_0, T_{21})$ and $T_{10} = \cup(T_{10}, T_{31})$.
    (5j) **ElseIf** (Detect(T_{22}) = "yes") **then**
      (5k) Discard$(T_{31}, T_{61}, T_{62}, T_{21}, T_{51}, T_{52})$.
      (5l) $T_0 = \cup(T_0, T_{22})$ and $T_{10} = \cup(T_{10}, T_{32})$.
    (5m) **ElseIf** (Detect(T_{51}) = "yes") **then**
      (5n) Discard$(T_{31}, T_{32}, T_{62}, T_{21}, T_{22}, T_{52})$.
      (5o) $T_0 = \cup(T_0, T_{51})$ and $T_{10} = \cup(T_{10}, T_{61})$.
    (5p) **ElseIf** (Detect(T_{52}) = "yes") **then**
      (5q) Discard$(T_{31}, T_{32}, T_{61}, T_{21}, T_{22}, T_{51})$.
      (5r) $T_0 = \cup(T_0, T_{52})$ and $T_{10} = \cup(T_{10}, T_{62})$.
    EndIf
  EndFor
EndFor
(6) **If** (Detect(T_{10}) = "yes") **then**
  (6a) Read(T_{10}).
EndIf
EndAlgorithm

Theorem 3-1: **Algorithm 3-1** can be used to implement addition of complex vectors and closure axioms of addition.

Proof: After the first execution of Step (1) and Step (2) is implemented, tube T_{10} contains DNA sequences encoding all of the elements in V and tube T_0 contains DNA sequences encoding the real part and the imaginary part of α and β. Next, after the kth execution of Step (3a) is implemented, two operands in $T_{0,0}$ are positive real values, two operands in $T_{1,1}$ are negative real values, in ${T_{0,1}}^{>=}$ the absolute value of the first positive operand is greater than or equal to the absolute value of the second negative operand, in ${T_{0,1}}^{<}$ the absolute value of the first positive operand is less than the absolute value of the second negative operand, in ${T_{1,0}}^{>=}$ the absolute value of the first negative operand is greater than or equal to the absolute value of the second negative operand and in ${T_{1,0}}^{<}$ the absolute value of the first negative operand is less than the absolute value of the second positive operand.

Next, after the kth execution of Step (3b) through Step (3d) is implemented, the required computation for their kth elements in the imaginary part of α and β in $T_{0,0}$, $T_{1,1}$, ${T_{0,1}}^{>=}$, ${T_{1,0}}^{>=}$, ${T_{0,1}}^{<}$, and ${T_{1,0}}^{<}$ is completed. Next, the kth execution of Step (3e) pours their contents into T_0. Next, similar computation to the real part of α and β is completed by means of implementing the kth execution of Step (3f) through Step (3j). Next, each execution of Step (5a) through Step (5f) yields different tubes with different DNA strands. Next, each execution of Step (5g) through Step (5r) removes illegal DNA strands and reserves legal DNA strands. Finally, DNA sequences in T_{10} give the answer of satisfying closure axioms. Next, the execution of Step (6) and Step (6a) completes the process of reading the answer(s). Therefore, it is inferred that **Algorithm 3-1** can be applied to implement addition of complex vectors and closure axioms of addition. ■

B. Constructing Molecular Solutions to Domain and Range of Closure Axioms of Addition

The following DNA-based algorithm, **Init**(T_{10}), is used to construct molecular solutions of a nonempty set V denoted in Section III-A. The notations used in **Init**(T_{10}) are denoted in Section III-A. The first parameter T_{10} is an empty tube.

Procedure **Init**(T_{10})
(1) **For** $k = 1$ to p
  (1a) Append − tail$(T_{11}, {u_{1,k,l+1}}^1)$.
  (1b) Append − tail$(T_{12}, {u_{1,k,l+1}}^0)$.
  (1c) $Y_k = \cup(T_{11}, T_{12})$.
  (2) **For** $j = l$ **down to** 1
    (2a) Amplify(Y_k, T_{11}, T_{12}).
    (2b) Append − tail$(T_{11}, {u_{1,k,j}}^1)$.
    (2c) Append − tail$(T_{12}, {u_{1,k,j}}^0)$.
    (2d) $Y_k = \cup(T_{11}, T_{12})$.
  EndFor
  (3) **For** $j = l+1$ **down to** 1
    (3a) Amplify(Y_k, T_{11}, T_{12}).
    (3b) Append − tail$(T_{11}, {v_{1,k,j}}^1)$.
    (3c) Append − tail$(T_{12}, {v_{1,k,j}}^0)$.
    (3d) $Y_k = \cup(T_{11}, T_{12})$.
  EndFor
EndFor

(4) **For** $k = 1$ to p
(4a) $T_{10} = \cup(T_{10}, Y_k)$.
EndFor
EndProcedure

Lemma 3-1: The DNA-based algorithm, $\textbf{Init}(T_{10})$, can be applied to construct molecular solutions to domain and range of closure axioms of addition.

Proof: Please refer to the proof of **Theorem 3-1**. ■

C. Constructing Molecular Solutions to Any Two Elements With p-Tuples of Complex Numbers

It is supposed that $\alpha = [\alpha a_{1,1} + \alpha b_{1,1}i \ \ldots \ \alpha a_{1,p} + \alpha b_{1,p}i]_{1\times p}$ and $\beta = [\beta a_{1,1} + \beta b_{1,1}i \ \ldots \ \beta a_{1,p} + \beta b_{1,p}i]_{1\times p}$, where $i = \sqrt{-1}$ and $\alpha a_{1,k}$, $\alpha b_{1,k}$, $\beta a_{1,k}$ and $\beta b_{1,k}$ are real numbers for $1 \le k \le p$. It is supposed that the value of $\alpha a_{1,k}$ in α and $\beta a_{1,k}$ in β can be subsequently represented as two *signed* binary numbers, $\alpha r_{k,l+1}\alpha r_{k,l}\ldots\alpha r_{k,1}$ and $\beta r_{k,l+1}\beta r_{k,l}\ldots\beta r_{k,1}$ for $1 \le k \le p$. $\alpha r_{k,l+1}{}^0$ and $\beta r_{k,l+1}{}^0$ represent positive sign, and $\alpha r_{k,l+1}{}^1$ and $\beta r_{k,l+1}{}^1$ represent negative sign. It is assumed for $1 \le k \le p$ that $\alpha r_{k,l}\ldots\alpha r_{k,s+1}$ and $\beta r_{k,l}\ldots\beta r_{k,s+1}$ are their integer and $\alpha r_{k,s}\ldots\alpha r_{k,1}$ and $\beta r_{k,s}\ldots\beta r_{k,1}$ are their fraction. It is also supposed that the value of $\alpha b_{1,k}$ in α and $\beta b_{1,k}$ in β can be also represented as two *signed* binary numbers, $\alpha i_{k,l+1}\alpha i_{k,l}\ldots\alpha i_{k,1}$ and $\beta i_{k,l+1}\beta i_{k,l}\ldots\beta i_{k,1}$ for $1 \le k \le p$. $\alpha i_{k,l+1}{}^0$ and $\beta i_{k,l+1}{}^0$ represent positive sign, and $\alpha i_{k,l+1}{}^1$ and $\beta i_{k,l+1}{}^1$ represent negative sign. It is assumed for $1 \le k \le p$ that $\alpha i_{k,l}\ldots\alpha i_{k,s+1}$ and $\beta i_{k,l}\ldots\beta i_{k,s+1}$ are their integer, and $\alpha i_{k,s}\ldots\alpha i_{k,1}$ and $\beta i_{k,s}\ldots\beta i_{k,1}$ are their fraction. To $1 \le k \le p$ and $1 \le j \le l+1$, it is assumed that $\alpha r_{k,j}{}^1$, $\alpha i_{k,j}{}^1$, $\beta r_{k,j}{}^1$, $\beta i_{k,j}{}^1$, $\alpha r_{k,j}{}^0$, $\alpha i_{k,j}{}^0$, $\beta r_{k,j}{}^0$ and $\beta i_{k,j}{}^0$ denote their values to be one and zero. The following DNA-based algorithm, $\textbf{InitialValue}(T_0)$, is used to construct molecular solutions of α and β. The first parameter T_0 is an empty tube.

Procedure InitialValue(T_0)
(1) **For** $k = p$ **to** 1
(2) **For** $j = 1$ **to** $l+1$
(2a) $\text{Append} - \text{head}(T_0, \alpha i_{k,j})$.
EndFor
(3) **For** $j = 1$ **to** $l+1$
(3a) $\text{Append} - \text{head}(T_0, \alpha r_{k,j})$.
EndFor
EndFor
(4) **For** $k = p$ **to** 1
(5) **For** $j = 1$ **to** $l+1$
(5a) $\text{Append} - \text{head}(T_0, \beta i_{k,j})$.
EndFor
(6) **For** $j = 1$ **to** $l+1$
(6a) $\text{Append} - \text{head}(T_0, \beta r_{k,j})$.
EndFor
EndFor
EndProcedure

Lemma 3-2: The DNA-based algorithm, $\textbf{InitialValue}(T_0)$, can be applied to construct molecular solutions of α and β.

Proof: Please refer to the proof of **Theorem 3-1**. ■

D. Constructing Parallel Comparators of Complex Numbers

The following first DNA-based algorithm, $\textbf{ImaginaryParallelComparator}(T_0, T_{0,0}, T_{1,1}, T_{0,1}{}^{>=}, T_{0,1}{}^{<}, T_{1,0}{}^{>=}, T_{1,0}{}^{<}, k, k)$, is proposed to complete the function of a parallel comparator of $(l+1)$ bits for the *imaginary* part of complex numbers, and the following *second* DNA-based algorithm, $\textbf{RealParallelComparator}(T_0, T_{0,0}, T_{1,1}, T_{0,1}{}^{>=}, T_{0,1}{}^{<}, T_{1,0}{}^{>=}, T_{1,0}{}^{<}, k, k)$, is also presented to complete the function of a parallel comparator of $(l+1)$ bits for the *real* part of complex numbers. When the two DNA-based algorithms are called by **Algorithm 3-1**, molecular solutions of α and β are in the *first* parameter, T_0, the *second* parameter through the *seventh* parameter are all empty tubes, and the values of the *eighth* parameter and the *ninth* parameter are the value of the index variable of the *first single* loop in **Algorithm 3-1**. Notations used in them are denoted in Section III-C.

Procedure
$\textbf{ImaginaryParallelComparator}(T_0, T_{0,0}, T_{1,1}, T_{0,1}{}^{>=}, T_{0,1}{}^{<}, T_{1,0}{}^{>=}, T_{1,0}{}^{<}, k, k)$
(1) $\textbf{ParallelComparator}(T_0, T_{0,0}, T_{1,1}, T_{0,1}{}^{>=}, T_{0,1}{}^{<}, T_{1,0}{}^{>=}, T_{1,0}{}^{<}, k, k, \alpha i, \beta i)$.
EndProcedure

Lemma 3-3: The DNA-based algorithm, $\textbf{ImaginaryParallelComparator}(T_0, T_{0,0}, T_{1,1}, T_{0,1}{}^{>=}, T_{0,1}{}^{<}, T_{1,0}{}^{>=}, T_{1,0}{}^{<}, k, k)$, can be applied to complete the function of a parallel comparator of $(l+1)$ bits for the *imaginary* part of complex numbers.

Proof: Please refer to the proof of **Theorem 3-1**. ■

Procedure
$\textbf{RealParallelComparator}(T_0, T_{0,0}, T_{1,1}, T_{0,1}{}^{>=}, T_{0,1}{}^{<}, T_{1,0}{}^{>=}, T_{1,0}{}^{<}, k, k)$
(1) $\textbf{ParallelComparator}(T_0, T_{0,0}, T_{1,1}, T_{0,1}{}^{>=}, T_{0,1}{}^{<}, T_{1,0}{}^{>=}, T_{1,0}{}^{<}, k, k, \alpha r, \beta r)$.
EndProcedure

Lemma 3-4: The DNA-based algorithm, $\textbf{RealParallelComparator}(T_0, T_{0,0}, T_{1,1}, T_{0,1}{}^{>=}, T_{0,1}{}^{<}, T_{1,0}{}^{>=}, T_{1,0}{}^{<}, k, k)$, can be employed to complete the function of a parallel comparator of $(l+1)$ bits for the *real* part of complex numbers.

Proof: Please refer to the proof of **Theorem 3-1**. ■

E. Constructing a Parallel Comparator of $(l+1)$ Bits for the Real Part and the Imaginary Part of Complex Numbers

The following DNA-based algorithm, $\textbf{ParallelComparator}(T_0, T_{0,0}, T_{1,1}, T_{0,1}{}^{>=}, T_{0,1}{}^{<}, T_{1,0}{}^{>=}, T_{1,0}{}^{<}, k, k, \alpha\theta, \beta\theta)$ is proposed to complete the function of a parallel comparator of $(l+1)$ bits for the real part and the imaginary part of complex numbers. The content from the first parameter to the ninth parameter is the same as that of nine arguments in two callers. If it is called for dealing with the imaginary part, then the *tenth* and *eleventh* parameters, $\alpha\theta$ and $\beta\theta$, are replaced by αi and βi. Otherwise, they are replaced by αr and βr.

Procedure $\textbf{ParallelComparator}(T_0, T_{0,0}, T_{1,1}, T_{0,1}{}^{>=}, T_{0,1}{}^{<}, T_{1,0}{}^{>=}, T_{1,0}{}^{<}, k, k, \alpha\theta, \beta\theta)$
(1) $T_7{}^{ON} = +(T_0, \alpha\theta_{k,l+1}{}^1)$ and $T_7{}^{OFF} = -(T_0, \alpha\theta_{k,l+1}{}^1)$.
(2) $T_{1,1} = +(T_7{}^{ON}, \beta\theta_{k,l+1}{}^1)$ and $T_{1,0} = -(T_7{}^{ON}, \beta\theta_{k,l+1}{}^1)$.
(3) $T_{0,1} = +(T_7{}^{OFF}, \beta\theta_{k,l+1}{}^1)$ and $T_{0,0} = -(T_7{}^{OFF}, \beta\theta_{k,l+1}{}^1)$.
(4) **For** $j = l$ **to** 1

(4a) $\mathbf{OneBitComparator}(T_{0,1}, T_{1,0}, {T_{0,1}}^{>=}, {T_{0,1}}^{<}, {T_{1,0}}^{>=}, {T_{1,0}}^{<}, k, j, \alpha\theta, \beta\theta)$.
(4b) **If** ((Detect$(T_{0,1})$ = "no") and (Detect$(T_{1,0})$ = "no")) **then**
(4c) Terminate the execution of the loop.
EndIf
EndFor
(5) ${T_{0,1}}^{>=} = \cup({T_{0,1}}^{>=}, T_{0,1})$.
(6) ${T_{1,0}}^{>=} = \cup({T_{1,0}}^{>=}, T_{1,0})$.
EndProcedure

Lemma 3-5: The DNA-based algorithm, $\mathbf{ParallelComparator}(T_0, T_{0,0}, T_{1,1}, {T_{0,1}}^{>=}, {T_{0,1}}^{<}, {T_{1,0}}^{>=}, {T_{1,0}}^{<}, k, k, \alpha\theta, \beta\theta)$, can be applied to complete the function of a parallel comparator of $(l+1)$ bits for the real part and the imaginary part of molecular solutions of α and β.

Proof: Please refer to the proof of **Theorem 3-1**. ■

F. Constructing a Parallel Comparator of One Bit for the Real Part and the Imaginary Part of Complex Numbers

The following DNA-based algorithm, $\mathbf{OneBitComparator}(T_{0,1}, T_{1,0}, {T_{0,1}}^{>=}, {T_{0,1}}^{<}, {T_{1,0}}^{>=}, {T_{1,0}}^{<}, k, j, \alpha\theta, \beta\theta)$, is presented to finish the function of a parallel comparator of one bit. The first parameter $T_{0,1}$ includes DNA sequences that have $\alpha\theta_{k,l+1} = 0$ and $\beta\theta_{k,l+1} = 1$, the second parameter $T_{1,0}$ contains DNA sequences that have $\alpha\theta_{k,l+1} = 1$ and $\beta\theta_{k,l+1} = 0$, the *third* parameter through the *sixth* parameter are all empty tubes, the value of the *seventh* parameter is the value of the index variable of the *first single* loop in **Algorithm 3-1**, and the value of the *eighth* parameter is the value of the index variable of the *only single* loop in the caller. If it is called for dealing with the imaginary part, then the *ninth* and *tenth* parameters, $\alpha\theta$ and $\beta\theta$, are replaced by αi and βi. Otherwise, they are replaced by αr and βr.

Procedure $\mathbf{OneBitComparator}(T_{0,1}, T_{1,0}, {T_{0,1}}^{>=}, {T_{0,1}}^{<}, {T_{1,0}}^{>=}, {T_{1,0}}^{<}, k, j, \alpha\theta, \beta\theta)$
(1) ${T_1}^{ON} = +(T_{0,1}, \alpha\theta_{k,j}{}^1)$ and ${T_1}^{OFF} = -(T_{0,1}, \alpha\theta_{k,j}{}^1)$.
(2) ${T_2}^{ON} = +({T_1}^{ON}, \beta\theta_{k,j}{}^1)$ and ${T_2}^{OFF} = -({T_1}^{ON}, \beta\theta_{k,j}{}^1)$.
(3) ${T_3}^{ON} = +({T_1}^{OFF}, \beta\theta_{k,j}{}^1)$ and ${T_3}^{OFF} = -({T_1}^{OFF}, \beta\theta_{k,j}{}^1)$.
(4) ${T_4}^{ON} = +(T_{1,0}, \alpha\theta_{k,j}{}^1)$ and ${T_4}^{OFF} = -(T_{1,0}, \alpha\theta_{k,j}{}^1)$.
(5) ${T_5}^{ON} = +({T_4}^{ON}, \beta\theta_{k,j}{}^1)$ and ${T_5}^{OFF} = -({T_4}^{ON}, \beta\theta_{k,j}{}^1)$.
(6) ${T_6}^{ON} = +({T_4}^{OFF}, \beta\theta_{k,j}{}^1)$ and ${T_6}^{OFF} = -({T_4}^{OFF}, \beta\theta_{k,j}{}^1)$.
(7) $T_{0,1} = \cup(T_{0,1}, {T_2}^{ON}, {T_3}^{OFF})$
(8) $T_{1,0} = \cup(T_{1,0}, {T_5}^{ON}, {T_6}^{OFF})$.
(9) ${T_{0,1}}^{>=} = \cup({T_{0,1}}^{>=}, {T_2}^{OFF})$.
(10) ${T_{0,1}}^{<} = \cup({T_{0,1}}^{<}, {T_3}^{ON})$.
(11) ${T_{1,0}}^{>=} = \cup({T_{1,0}}^{>=}, {T_5}^{OFF})$.
(12) ${T_{1,0}}^{<} = \cup({T_{1,0}}^{<}, {T_6}^{ON})$.
EndProcedure

Lemma 3-6: The DNA-based algorithm, $\mathbf{OneBitComparator}(T_{0,1}, T_{1,0}, {T_{0,1}}^{>=}, {T_{0,1}}^{<}, {T_{1,0}}^{>=}, {T_{1,0}}^{<}, k, j, \alpha\theta, \beta\theta)$, can be employed to complete the function of a parallel comparator of one bit.

Proof: Please refer to the proof of **Theorem 3-1**. ■

G. Constructing Parallel Adders of Complex Numbers

For the *imaginary* part and real part of complex numbers, $\mathbf{ImaginaryBinaryParallelAdder}(T_{0,0}, T_{1,1}, k)$ and $\mathbf{RealBinaryParallelAdder}(T_{0,0}, T_{1,1}, k)$ are offered to complete the function of a parallel adder of $(l+1)$ bits. DNA sequences in $T_{0,0}$ encode two positive operands, DNA sequences in $T_{1,1}$ encode two negative operands, and the value of k is the value of the index variable of the first single loop in **Algorithm 3-1**. The notations used in the following two DNA-based algorithms are denoted in the previous subsections.

Procedure $\mathbf{ImaginaryBinaryParallelAdder}(T_{0,0}, T_{1,1}, k)$
(1) $\mathbf{BinaryParallelAdder}(T_{0,0}, T_{1,1}, k, \alpha i_{k,j}, \beta i_{k,j}, i_{k,j})$.
EndProcedure

Lemma 3-7: The DNA-based algorithm, $\mathbf{ImaginaryBinaryParallelAdder}(T_{0,0}, T_{1,1}, k)$, can be used to complete the function of a parallel adder of $(l+1)$ bits for the *imaginary* part of complex numbers.

Proof: Please refer to the proof of **Theorem 3-1**. ■

Procedure $\mathbf{RealBinaryParallelAdder}(T_{0,0}, T_{1,1}, k)$
(1) $\mathbf{BinaryParallelAdder}(T_{0,0}, T_{1,1}, k, \alpha r_{k,j}, \beta r_{k,j}, y_{k,j})$.
EndProcedure

Lemma 3-8: The DNA-based algorithm, $\mathbf{RealBinaryParallelAdder}(T_{0,0}, T_{1,1}, k)$, can be applied to complete the function of a parallel adder of $(l+1)$ bits for the *real* part of complex numbers.

Proof: Please refer to the proof of **Theorem 3-1**. ■

H. Constructing a Parallel Adder of $(l+1)$ Bits for the Real Part and the Imaginary Part of Complex Numbers

The following DNA-based algorithm, $\mathbf{BinaryParallelAdder}(T_{0,0}, T_{1,1}, k, \alpha\theta, \beta\theta, sum)$ is offered to complete the function of a parallel adder of $(l+1)$ bits for the real part and the imaginary part of complex numbers. The front three parameters are the same as the two callers. If it is called by $\mathbf{ImaginaryBinaryParallelAdder}(T_{0,0}, T_{1,1}, k)$, then the *fourth*, *fifth* and *sixth* parameters, $\alpha\theta$, $\beta\theta$ and sum, are subsequently replaced by $\alpha i_{k,j}$, $\beta i_{k,j}$ and $i_{k,j}$. Otherwise, they are subsequently replaced by $\alpha r_{k,j}$, $\beta r_{k,j}$ and $y_{k,j}$. An adder of one bit can be applied to figure out the sum and the carry of two input bits and a previous carry. An adder for two operands with $(l+1)$ bits can be completed by means of the adder of one bit.

Procedure $\mathbf{BinaryParallelAdder}(T_{0,0}, T_{1,1}, k, \alpha\theta, \beta\theta, sum)$
(0) **If** (Detect$(T_{0,0})$ == "yes") **then**
(1) Append − head$(T_{0,0}, {z_{k,0}}^0)$.
(2) $T_{20} = \cup(T_{0,0}, T_{20})$.
(3) **For** $j = 1$ to l
(3a) $\mathbf{ParallelOneBitAdder}(T_{20}, k, j, \alpha\theta, \beta\theta, sum)$.
EndFor
(4) $T_{0,0} = \cup(T_{0,0}, T_{20})$.
(5) Append − head$(T_{0,0}, {sum_{k,l+1}}^0)$.
EndIf
(5a) **If** (Detect$(T_{1,1})$ == "yes") **then**
(6) Append − head$(T_{1,1}, {z_{k,0}}^0)$.
(7) $T_{20} = \cup(T_{1,1}, T_{20})$.
(8) **For** $j = 1$ to l
(8a) $\mathbf{ParallelOneBitAdder}(T_{20}, k, j, \alpha\theta, \beta\theta, sum)$.
EndFor
(9) $T_{1,1} = \cup(T_{1,1}, T_{20})$.

(10) Append $-$ head$(T_{1,1}, sum_{k,l+1}{}^1)$.
EndIf
EndProcedure

Lemma 3-9: The DNA-based algorithm, $\mathbf{BinaryParallelAdder}(T_{0,0}, T_{1,1}, k, \alpha\theta, \beta\theta, sum)$, can be applied to complete the function of a parallel adder of $(l+1)$ bits for the real part and the imaginary part of complex numbers.

Proof: Please refer to the proof of **Theorem 3-1**. ■

I. Constructing a Parallel Adder of One Bit for the Real Part and the Imaginary Part of Complex Numbers

The following DNA-based algorithm, $\mathbf{ParallelOneBitAdder}(T_{20}, k, j, \alpha\theta, \beta\theta, sum)$, is proposed to complete the function of a parallel adder of one bits for the real part and the imaginary part of complex numbers. If it is called by Step (3a) in $\mathbf{BinaryParallelAdder}(T_{0,0}, T_{1,1}, k, \alpha\theta, \beta\theta, sum)$, then the *first* parameter, T_{20}, contains DNA sequences encoding positive operands. If it is called by Step (8a) in $\mathbf{BinaryParallelAdder}(T_{0,0}, T_{1,1}, k, \alpha\theta, \beta\theta, sum)$, then the *first* parameter, T_{20}, contains DNA sequences encoding negative operands. The value of the *second* parameter, k, is the value of the index variable of the *first single* loop in **Algorithm 3-1**, and the value of the *third* parameter, j, is the value of the index variable of the single loop in Step (3) or Step (8) in the caller. The last three parameters are the same as the last three arguments in the caller.

In an adder of one bit, it is assumed for $1 \leq k \leq p$ and $1 \leq j \leq l+1$ that $\alpha\theta_{k,j}$ represents the first input, $sum_{k,j}$ represents the first output, $\beta\theta_{k,j}$ represents the second input, $z_{k,j-1}$ represents the third input, and $z_{k,j}$ represents the second output. It is assumed that for $1 \leq k \leq p$ and $1 \leq j \leq l+1$, $z_{k,j-1}{}^1$, $z_{k,j}{}^1$, $\alpha\theta_{k,j}{}^1$, $\beta\theta_{k,j}{}^1$ and $sum_{k,j}{}^1$ are used to represent their values to be one, and $z_{k,j-1}{}^0$, $z_{k,j}{}^0$, $\alpha\theta_{k,j}{}^0$, $\beta\theta_{k,j}{}^0$ and $sum_{k,j}{}^0$ are used to represent their values to be zero.

Procedure ParallelOneBitAdder$(T_{20}, k, j, \alpha\theta, \beta\theta, sum)$
(1) $T_1 = +(T_{20}, \alpha\theta_{k,j}{}^1)$ and $T_2 = -(T_{20}, \alpha\theta_{k,j}{}^1)$.
(2) $T_3 = +(T_1, \beta\theta_{k,j}{}^1)$ and $T_4 = -(T_1, \beta\theta_{k,j}{}^1)$.
(3) $T_5 = +(T_2, \beta\theta_{k,j}{}^1)$ and $T_6 = -(T_2, \beta\theta_{k,j}{}^1)$.
(4) $T_7 = +(T_3, z_{k,j-1}{}^1)$ and $T_8 = -(T_3, z_{k,j-1}{}^1)$.
(5) $T_9 = +(T_4, z_{k,j-1}{}^1)$ and $T_{10} = -(T_4, z_{k,j-1}{}^1)$.
(6) $T_{11} = +(T_5, z_{k,j-1}{}^1)$ and $T_{12} = -(T_5, z_{k,j-1}{}^1)$.
(7) $T_{13} = +(T_6, z_{k,j-1}{}^1)$ and $T_{14} = -(T_6, z_{k,j-1}{}^1)$.
(8) Append $-$ head$(T_7, sum_{k,j}{}^1)$ and Append $-$ head$(T_7, z_{k,j}{}^1)$.
(9) Append $-$ head$(T_8, sum_{k,j}{}^0)$ and Append $-$ head$(T_8, z_{k,j}{}^1)$.
(10) Append $-$ head$(T_9, sum_{k,j}{}^0)$ and Append $-$ head$(T_9, z_{k,j}{}^1)$.
(11) Append $-$ head$(T_{10}, sum_{k,j}{}^1)$ and Append $-$ head$(T_{10}, z_{k,j}{}^0)$.
(12) Append $-$ head$(T_{11}, sum_{k,j}{}^0)$ and Append $-$ head$(T_{11}, z_{k,j}{}^1)$.
(13) Append $-$ head$(T_{12}, sum_{k,j}{}^1)$ and Append $-$ head$(T_{12}, z_{k,j}{}^0)$.
(14) Append $-$ head$(T_{13}, sum_{k,j}{}^1)$ and Append $-$ head$(T_{13}, z_{k,j}{}^0)$.
(15) Append $-$ head$(T_{14}, sum_{k,j}{}^0)$ and Append $-$ head$(T_{14}, z_{k,j}{}^0)$.
(16) $T_{20} = \cup(T_7, T_8, T_9, T_{10}, T_{11}, T_{12}, T_{13}, T_{14})$.
EndProcedure

Lemma 3-10: The DNA-based algorithm, $\mathbf{ParallelOneBitAdder}(T_{20}, k, j, \alpha\theta, \beta\theta, sum)$, can be applied to complete the function of a parallel adder of one bit.

Proof: Please refer to the proof of **Theorem 3-1**. ■

J. Constructing Parallel Subtractors for the Absolute Value of the First Operand Greater Than or Equal to the Absolute Value of the Second Operand in Complex Numbers

For the absolute value of the first operand greater than or equal to the absolute value of the second operand in the imaginary and real parts of complex numbers, $\mathbf{ImaginaryBinaryParallelSubtractorGE}(T_{0,1}{}^{>=}, T_{1,0}{}^{>=}, k)$ and $\mathbf{RealBinaryParallelSubtractorGE}(T_{0,1}{}^{>=}, T_{1,0}{}^{>=}, k)$ are proposed to complete the function of a parallel subtractor of $(l+1)$ bits. DNA sequences in $T_{0,1}{}^{>=}$ encode the first positive operand and the second negative operand in which the absolute value of the first positive operand is greater than or equal to the absolute value of the second negative operand. DNA sequences in $T_{1,0}{}^{>=}$ encode the first negative operand and the second positive operand in which the absolute value of the first negative operand is greater than or equal to the absolute value of the second positive operand. The value of the third parameter is the value of the index variable of the first single loop in **Algorithm 3-1**. The notations used in the following two DNA-based algorithms are denoted in the previous subsections.

Procedure
$\mathbf{ImaginaryBinaryParallelSubtractorGE}(T_{0,1}{}^{>=}, T_{1,0}{}^{>=}, k)$
(1) $\mathbf{BinaryParallelSubtractorGE}(T_{0,1}{}^{>=}, T_{1,0}{}^{>=}, k, \alpha i_{k,j}, \beta i_{k,j}, i_{k,j})$.
EndProcedure

Lemma 3-11: The DNA-based algorithm, $\mathbf{ImaginaryBinaryParallelSubtractorGE}(T_{0,1}{}^{>=}, T_{1,0}{}^{>=}, k)$, can be employed to complete the function of a parallel subtractor of $(l+1)$ bits for the absolute value of the first operand greater than or equal to the absolute value of the second operand in the *imaginary* part of complex numbers.

Proof: Please refer to the proof of **Theorem 3-1**. ■

Procedure
$\mathbf{RealBinaryParallelSubtractorGE}(T_{0,1}{}^{>=}, T_{1,0}{}^{>=}, k)$
(1) $\mathbf{BinaryParallelSubtractorGE}(T_{0,1}{}^{>=}, T_{1,0}{}^{>=}, k, \alpha r_{k,j}, \beta r_{k,j}, y_{k,j})$.
EndProcedure

Lemma 3-12: The DNA-based algorithm, $\mathbf{RealBinaryParallelSubtractorGE}(T_{0,1}{}^{>=}, T_{1,0}{}^{>=}, k)$, can be used to complete the function of a parallel subtractor of $(l+1)$ bits for the absolute value of the first operand greater than or equal to the absolute value of the second operand in the *real* part of complex numbers.

Proof: Please refer to the proof of **Theorem 3-1**. ■

K. Constructing Parallel Subtractors of $(l+1)$ Bits for the Absolute Value of the First Operand Greater Than or Equal to the Absolute Value of the Second Operand in the Real Part and the Imaginary Part of Complex Numbers

The following DNA-based algorithm, $\mathbf{BinaryParallelSubtractorGE}(T_{0,1}{}^{>=}, T_{1,0}{}^{>=}, k, \alpha\theta, \beta\theta, sum)$ is proposed to complete the function of a parallel *subtractor* of $(l+1)$ bits for that the absolute value of the first operand is greater than or is equal to the absolute value of the second operand in the real part and the imaginary part of complex numbers. The front three parameters are the same as the front three arguments of the two callers. If it is called by $\mathbf{ImaginaryBinaryParallelSubtractorGE}(T_{0,1}{}^{>=}, T_{1,0}{}^{>=}, k)$, then the *fourth*, *fifth* and *sixth* parameters, $\alpha\theta$, $\beta\theta$ and sum, are subsequently replaced by $\alpha i_{k,j}$, $\beta i_{k,j}$ and $i_{k,j}$. Otherwise, they are subsequently replaced by $\alpha r_{k,j}$, $\beta r_{k,j}$ and $y_{k,j}$. A subtractor of one bit can be used to compute the difference bit and the borrow bit for two input bits and a previous borrow. A subtractor for two operands with $(l+1)$ bits can be completed by means of the subtractor of one bit.

Procedure
$\mathbf{BinaryParallelSubtractorGE}(T_{0,1}{}^{>=}, T_{1,0}{}^{>=}, k, \alpha\theta, \beta\theta, sum)$
(1) **If** $(\text{Detect}(T_{0,1}{}^{>=}) ==$ "yes") **then**
(2) $\text{Append}-\text{head}(T_{0,1}{}^{>=}, z_{k,0}{}^{0})$.
(3) $T_{30} = \cup(T_{0,1}{}^{>=}, T_{30})$.
(4) **For** $j = 1$ **to** l
(4a) $\mathbf{ParallelOneBitSubtractorGE}(T_{30}, k, j, \alpha\theta, \beta\theta, sum)$.
EndFor
(5) $T_{0,1}{}^{>=} = \cup(T_{0,1}{}^{>=}, T_{30})$.
(6) $\text{Append}-\text{head}(T_{0,1}{}^{>=}, sum_{k,l+1}{}^{0})$.
EndIf
(7) **If** $(\text{Detect}(T_{1,0}{}^{>=}) ==$ "yes") **then**
(8) $\text{Append}-\text{head}(T_{1,0}{}^{>=}, z_{k,0}{}^{0})$.
(9) $T_{30} = \cup(T_{1,0}{}^{>=}, T_{30})$.
(10) **For** $j = 1$ **to** l
(10a) $\mathbf{ParallelOneBitSubtractorGE}(T_{30}, k, j, \alpha\theta, \beta\theta, sum)$.
EndFor
(11) $T_{1,0}{}^{>=} = \cup(T_{1,0}{}^{>=}, T_{30})$.
(12) $\text{Append}-\text{head}(T_{1,0}{}^{>=}, sum_{k,l+1}{}^{1})$.
EndIf
EndProcedure

Lemma 3-13: The DNA-based algorithm, $\mathbf{BinaryParallelSubtractorGE}(T_{0,1}{}^{>=}, T_{1,0}{}^{>=}, k, \alpha\theta, \beta\theta, sum)$, can be employed to complete the function of parallel subtractors of $(l+1)$ bits for that the absolute value of the first operand is greater than or equal to the absolute value of the second operand in the real part and the imaginary part of complex numbers.

Proof: Please refer to the proof of **Theorem 3-1**. ■

L. Constructing Parallel Subtractors of One Bits for the Absolute Value of the First Operand Greater Than or Equal to the Absolute Value of the Second Operand in the Real Part and the Imaginary Part of Complex Numbers

For the real part and the imaginary part of complex numbers, the following DNA-based algorithm, $\mathbf{ParallelOneBitSubtractorGE}(T_{30}, k, j, \alpha\theta, \beta\theta, sum)$, is offered to complete the function of a parallel subtractor of one bit. If it is called by Step (4a) in $\mathbf{BinaryParallelSubtractorGE}(T_{0,1}{}^{>=}, T_{1,0}{}^{>=}, k, \alpha\theta, \beta\theta, sum)$, then the *first* parameter, T_{30}, consists of DNA sequences encoding the first positive operand and the second negative operand in which the absolute value of the first positive operand is greater than or equal to the absolute value of the second negative operand. If it is called by Step (10a) in $\mathbf{BinaryParallelSubtractorGE}(T_{0,1}{}^{>=}, T_{1,0}{}^{>=}, k, \alpha\theta, \beta\theta, sum)$, then the *first* parameter, T_{30}, contains DNA sequences encoding the first negative operand and the second positive operand in which the absolute value of the first negative operand is greater than or equal to the absolute value of the second positive operand. The value of the *second* parameter, k, is the value of the index variable of the *first single* loop in **Algorithm 3-1**, and the value of the *third* parameter, j, is the value of the index variable of the single loop in Step (4) or Step (10) in the caller. The last three parameters are the same as the last three arguments of the caller. In a subtractor of one bit, it is supposed that for $1 \le k \le p$ and $1 \le j \le l+1$, $\alpha\theta_{k,j}$ represents the first input, $sum_{k,j}$ represents the first output, $\beta\theta_{k,j}$ represents the second input, $z_{k,j-1}$ represents the third input, and $z_{k,j}$ represents the second output. *Distinct* DNA sequences are designed to encode the value "0" or "1" for $z_{k,j-1}$, $z_{k,j}$, $\alpha\theta_{k,j}$, $\beta\theta_{k,j}$ and $sum_{k,j}$ for $1 \le k \le p$ and $1 \le j \le l+1$. It is assumed that for $1 \le k \le p$ and $1 \le j \le l+1$, $z_{k,j-1}{}^{1}$, $z_{k,j}{}^{1}$, $\alpha\theta_{k,j}{}^{1}$, $\beta\theta_{k,j}{}^{1}$ and $sum_{k,j}{}^{1}$ are applied to represent their values to be one, and $z_{k,j-1}{}^{0}$, $z_{k,j}{}^{0}$, $\alpha\theta_{k,j}{}^{0}$, $\beta\theta_{k,j}{}^{0}$ and $sum_{k,j}{}^{0}$ are used to represent their values to be zero.

Procedure
$\mathbf{ParallelOneBitSubtractorGE}(T_{30}, k, j, \alpha\theta, \beta\theta, sum)$

(1) $T_1 = +(T_{30}, \alpha\theta_{k,j}{}^{1})$ and $T_2 = -(T_{30}, \alpha\theta_{k,j}{}^{1})$.

(2) $T_3 = +(T_1, \beta\theta_{k,j}{}^{1})$ and $T_4 = -(T_1, \beta\theta_{k,j}{}^{1})$.

(3) $T_5 = +(T_2, \beta\theta_{k,j}{}^{1})$ and $T_6 = -(T_2, \beta\theta_{k,j}{}^{1})$.

(4) $T_7 = +(T_3, z_{k,j-1}{}^{1})$ and $T_8 = -(T_3, z_{k,j-1}{}^{1})$.

(5) $T_9 = +(T_4, z_{k,j-1}{}^{1})$ and $T_{10} = -(T_4, z_{k,j-1}{}^{1})$.

(6) $T_{11} = +(T_5, z_{k,j-1}{}^{1})$ and $T_{12} = -(T_5, z_{k,j-1}{}^{1})$.

(7) $T_{13} = +(T_6, z_{k,j-1}{}^{1})$ and $T_{14} = -(T_6, z_{k,j-1}{}^{1})$.

(8) $\text{Append}-\text{head}(T_7, sum_{k,j}{}^{1})$ and $\text{Append}-\text{head}(T_7, z_{k,j}{}^{1})$.

(9) $\text{Append}-\text{head}(T_8, sum_{k,j}{}^{0})$ and $\text{Append}-\text{head}(T_8, z_{k,j}{}^{0})$.

(10) $\text{Append}-\text{head}(T_9, sum_{k,j}{}^{0})$ and $\text{Append}-\text{head}(T_9, z_{k,j}{}^{0})$.

(11) $\text{Append}-\text{head}(T_{10}, sum_{k,j}{}^{1})$ and $\text{Append}-\text{head}(T_{10}, z_{k,j}{}^{0})$.

(12) $\text{Append}-\text{head}(T_{11}, sum_{k,j}{}^{0})$ and $\text{Append}-\text{head}(T_{11}, z_{k,j}{}^{1})$.

(13) $\text{Append}-\text{head}(T_{12}, sum_{k,j}{}^{1})$ and $\text{Append}-\text{head}(T_{12}, z_{k,j}{}^{1})$.

(14) $\text{Append}-\text{head}(T_{13}, sum_{k,j}{}^{1})$ and $\text{Append}-\text{head}(T_{13}, z_{k,j}{}^{1})$.

(15) $\text{Append}-\text{head}(T_{14}, sum_{k,j}{}^{0})$ and $\text{Append}-\text{head}(T_{14}, z_{k,j}{}^{0})$.

(16) $T_{30} = \cup(T_7, T_8, T_9, T_{10}, T_{11}, T_{12}, T_{13}, T14)$.

EndProcedure

Lemma 3-14: The DNA-based algorithm, $\mathbf{ParallelOneBitSubtractorGE}(T_{30}, k, j, \alpha\theta, \beta\theta, sum)$, can be applied to complete the function of a parallel subtractor with one bit for that the absolute value of the first operand is greater than or equal to the absolute value of the second operand.

Proof: Please refer to the proof of **Theorem 3-1**. ■

M. Constructing Parallel Subtractors for the Absolute Value of the First Operand Less Than the Absolute Value of the Second Operand in Complex Numbers

In the *imaginary* and real parts of complex numbers, **ImaginaryBinaryParallelSubtractorLT**$(T_{0,1}{}^{<}, T_{1,0}{}^{<}, k)$ and **RealBinaryParallelSubtractorLT**$(T_{0,1}{}^{<}, T_{1,0}{}^{<}, k)$ are proposed to complete the function of a parallel subtractor of $(l+1)$ bits for that the absolute value of the first operand is less than the absolute value of the second operand. When the two DNA-based algorithms are called by **Algorithm 3-1**, DNA sequences in $T_{0,1}{}^{<}$ encode the first positive operand and the second negative operand in which the absolute value of the first positive operand is less than the absolute value of the second negative operand. DNA sequences in $T_{1,0}{}^{<}$ encode the first negative operand and the second positive operand in which the absolute value of the first negative operand is less than the absolute value of the second positive operand. The value of the third parameter is the value of the index variable of the first single loop in **Algorithm 3-1**. The notations used in the following two DNA-based algorithms are denoted in the previous subsections.

Procedure ImaginaryBinaryParallelSubtractorLT$(T_{0,1}{}^{<}, T_{1,0}{}^{<}, k)$

(1) **BinaryParallelSubtractorLT**$(T_{0,1}{}^{<}, T_{1,0}{}^{<}, k, \alpha i_{k,j}, \beta i_{k,j}, i_{k,j})$.

EndProcedure

Lemma 3-15: The DNA-based algorithm, **ImaginaryBinaryParallelSubtractorLT**$(T_{0,1}{}^{<}, T_{1,0}{}^{<}, k)$, can be used to complete the function of a parallel subtractor of $(l+1)$ bits for that the absolute value of the first operand is less than the absolute value of the second operand in the *imaginary* part of complex numbers.

Proof: Please refer to the proof of **Theorem 3-1**. ■

Procedure RealBinaryParallelSubtractorLT$(T_{0,1}{}^{<}, T_{1,0}{}^{<}, k)$

(1) **BinaryParallelSubtractorLT**$(T_{0,1}{}^{<}, T_{1,0}{}^{<}, k, \alpha r_{k,j}, \beta r_{k,j}, y_{k,j})$.

EndProcedure

Lemma 3-16: The DNA-based algorithm, **RealBinaryParallelSubtractorLT**$(T_{0,1}{}^{<}, T_{1,0}{}^{<}, k)$, can be applied to complete the function of a parallel subtractor of $(l+1)$ bits for that the absolute value of the first operand is less than the absolute value of the second operand in the *real* part of complex numbers.

Proof: Please refer to the proof of **Theorem 3-1**. ■

N. Constructing Parallel Subtractors of $(l+1)$ *Bits for the Absolute Value of the First Operand Less Than the Absolute Value of the Second Operand in the Real Part and the Imaginary Part of Complex Numbers*

The following DNA-based algorithm, **BinaryParallelSubtractorLT**$(T_{0,1}{}^{<}, T_{1,0}{}^{<}, k, \alpha\theta, \beta\theta, sum)$ is presented to complete the function of a parallel *subtractor* of $(l+1)$ bits for that the absolute value of the first operand is less than the absolute value of the second operand in the real part and the imaginary part of complex numbers. DNA sequences in $T_{0,1}{}^{<}$ encode the first positive operand and the second negative operand in which the absolute value of the first positive operand is less than the absolute value of the second negative operand. DNA sequences in $T_{1,0}{}^{<}$ encode the first negative operand and the second positive operand in which the absolute value of the first negative operand is less than the absolute value of the second positive operand. The value of the third parameter is the value of the index variable of the first single loop in **Algorithm 3-1**. The *fourth*, *fifth* and *sixth* parameters, $\alpha\theta$, $\beta\theta$ and sum, are subsequently replaced by $\alpha i_{k,j}$, $\beta i_{k,j}$ and $i_{k,j}$ if it is called by **ImaginaryBinaryParallelSubtractorLT**$(T_{0,1}{}^{<}, T_{1,0}{}^{<}, k)$. Otherwise, they are subsequently replaced by $\alpha r_{k,j}$, $\beta r_{k,j}$ and $y_{k,j}$. A subtractor of one bit can be employed to figure out the difference bit and the borrow bit for two input bits and a previous borrow. A subtractor for two operands with $(l+1)$ bits can be completed by means of the subtractor of one bit.

Procedure BinaryParallelSubtractorLT $(T_{0,1}{}^{<}, T_{1,0}{}^{<}, k, \alpha\theta, \beta\theta, sum)$
(1) **If** (Detect$(T_{0,1}{}^{<})$ == "yes") **then**
(2) Append − head$(T_{0,1}{}^{<}, z_{k,0}{}^{0})$.
(3) $T_{40} = \cup(T_{0,1}{}^{<}, T_{40})$.
(4) **For** $j = 1$ **to** l
(4a) **ParallelOneBitSubtractorLT**$(T_{40}, k, j, \alpha\theta, \beta\theta, sum)$.
EndFor
(5) $T_{0,1}{}^{<} = \cup(T_{0,1}{}^{<}, T_{40})$.
(6) Append − head$(T_{0,1}{}^{<}, sum_{k,l+1}{}^{1})$.
EndIf
(7) **If** (Detect$(T_{1,0}{}^{<})$ == "yes") **then**
(8) Append − head$(T_{1,0}{}^{<}, z_{k,0}{}^{0})$.
(9) $T_{40} = \cup(T_{1,0}{}^{<}, T_{40})$.
(10) **For** $j = 1$ **to** l
(10a) **ParallelOneBitSubtractorLT**$(T_{40}, k, j, \alpha\theta, \beta\theta, sum)$.
EndFor
(11) $T_{1,0}{}^{<} = \cup(T_{1,0}{}^{<}, T_{40})$.
(12) Append − head$(T_{1,0}{}^{<}, sum_{k,l+1}{}^{0})$.
EndIf
EndProcedure

Lemma 3-17: The DNA-based algorithm, **BinaryParallelSubtractorLT**$(T_{0,1}{}^{<}, T_{1,0}{}^{<}, k, \alpha\theta, \beta\theta, sum)$, can be applied to complete the function of parallel subtractors of $(l+1)$ bits for that the absolute value of the first operand is less than the absolute value of the second operand.

Proof: Please refer to the proof of **Theorem 3-1**. ■

O. Constructing Parallel Subtractors of One Bits for the Absolute Value of the First Operand Less Than the Absolute Value of the Second Operand in the Real Part and the Imaginary Part of Complex Numbers

The following DNA-based algorithm, **ParallelOneBitSubtractorLT**$(T_{40}, k, j, \alpha\theta, \beta\theta, sum)$, is presented to complete the function of a parallel subtractor of one bit for that the absolute value of the first operand is less than the absolute value of the second operand in the real part and the imaginary part of complex numbers. If it is called by Step (4a) in **BinaryParallelSubtractorLT**$(T_{0,1}{}^{<}, T_{1,0}{}^{<}, k, \alpha\theta, \beta\theta, sum)$, then the *first* parameter, T_{40}, includes DNA sequences encoding the first positive operand and the second negative operand in which the absolute value of the first positive operand is less than the absolute value of the second negative operand. If it is called by Step (10a) in **BinaryParallelSubtractorLT**$(T_{0,1}{}^{<}, T_{1,0}{}^{<}, k, \alpha\theta, \beta\theta, sum)$, then the *first* parameter, T_{40}, consists of DNA sequences encoding the first negative operand and the second positive operand in which the absolute value of the first negative operand is less than the absolute value of the second positive operand. The value of the *second* parameter, k, is the value of the index variable of the *first single* loop in **Algorithm 3-1**, and the value of the *third* parameter, j, is the value of the index variable of the single loop in Step (4) or Step (10) in $\mathbf{BinaryParallelSubtractorLT}(T_{0,1}{}^{<}, T_{1,0}{}^{<}, k, \alpha\theta, \beta\theta, sum)$. The last three parameters are the same as the last three arguments in the caller.

Because the absolute value of the first operand is less than the absolute value of the second operand, in a subtractor of one bit it is supposed for $1 \le k \le p$ and $1 \le j \le l+1$ that $\beta\theta_{k,j}$ represents the *first* input, $\alpha\theta_{k,j}$ represents the *second* input, $z_{k,j-1}$ represents the *third* input, $sum_{k,j}$ represents the *first* output, and $z_{k,j}$ represents the *second* output.

Procedure
$\mathbf{ParallelOneBitSubtractorLT}(T_{40}, k, j, \alpha\theta, \beta\theta, sum)$
(1) $T_1 = +(T_{40}, \beta\theta_{k,j}{}^1)$ and $T_2 = -(T_{40}, \beta\theta_{k,j}{}^1)$.
(2) $T_3 = +(T_1, \alpha\theta_{k,j}{}^1)$ and $T_4 = -(T_1, \alpha\theta_{k,j}{}^1)$.
(3) $T_5 = +(T_2, \alpha\theta_{k,j}{}^1)$ and $T_6 = -(T_2, \alpha\theta_{k,j}{}^1)$.
(4) $T_7 = +(T_3, z_{k,j-1}{}^1)$ and $T_8 = -(T_3, z_{k,j-1}{}^1)$.
(5) $T_9 = +(T_4, z_{k,j-1}{}^1)$ and $T_{10} = -(T_4, z_{k,j-1}{}^1)$.
(6) $T_{11} = +(T_5, z_{k,j-1}{}^1)$ and $T_{12} = -(T_5, z_{k,j-1}{}^1)$.
(7) $T_{13} = +(T_6, z_{k,j-1}{}^1)$ and $T_{14} = -(T_6, z_{k,j-1}{}^1)$.
(8) $\text{Append}-\text{head}(T_7, sum_{k,j}{}^1)$ and $\text{Append}-\text{head}(T_7, z_{k,j}{}^1)$.
(9) $\text{Append}-\text{head}(T_8, sum_{k,j}{}^0)$ and $\text{Append}-\text{head}(T_8, z_{k,j}{}^0)$.
(10) $\text{Append}-\text{head}(T_9, sum_{k,j}{}^0)$ and $\text{Append}-\text{head}(T_9, z_{k,j}{}^0)$.
(11) $\text{Append}-\text{head}(T_{10}, sum_{k,j}{}^1)$ and $\text{Append}-\text{head}(T_{10}, z_{k,j}{}^0)$.
(12) $\text{Append}-\text{head}(T_{11}, sum_{k,j}{}^0)$ and $\text{Append}-\text{head}(T_{11}, z_{k,j}{}^1)$.
(13) $\text{Append}-\text{head}(T_{12}, sum_{k,j}{}^1)$ and $\text{Append}-\text{head}(T_{12}, z_{k,j}{}^1)$.
(14) $\text{Append}-\text{head}(T_{13}, sum_{k,j}{}^1)$ and $\text{Append}-\text{head}(T_{13}, z_{k,j}{}^1)$.
(15) $\text{Append}-\text{head}(T_{14}, sum_{k,j}{}^0)$ and $\text{Append}-\text{head}(T_{14}, z_{k,j}{}^0)$.
(16) $T_{40} = \cup(T_7, T_8, T_9, T_{10}, T_{11}, T_{12}, T_{13}, T_{14})$.
EndProcedure

Lemma 3-18: The DNA-based algorithm, $\mathbf{ParallelOneBitSubtractorLT}(T_{40}, k, j, \alpha\theta, \beta\theta, sum)$, can be applied to complete the function of a parallel subtractor of one bit for that the absolute value of the first operand is less than the absolute value of the second operand.

Proof: Please refer to the proof of **Theorem 3-1**. ■

IV. Complexity Assessment

Theorem 4-1: For implementing addition and closure axioms of addition to complex vectors with p-tuples of complex numbers with that the values of each imaginary and real parts are encoded as signed binary numbers of $(l+1)$ bits and each bit is encoded by θ base pairs, its time complexity is $O(p \times l)$ biological operations, its volume complexity is $O(2^{2\times(l+1)} \times p)$ DNA strands, its tube complexity is O(1) tubes, and its longest DNA strand is $O(p \times l \times \theta)$ base pairs.

Proof: The DNA-based algorithm, $\mathbf{Init}(T_{10})$, in Step (1) of **Algorithm 3-1** is implemented by means of $O(p \times l)$ biological operations. Next, the DNA-based algorithm, $\mathbf{InitialValue}(T_0)$, in Step (2) of **Algorithm 3-1** is implemented by means of $O(p \times l)$ biological operations. The four DNA-based algorithms from Step (3a) through Step (3d) of **Algorithm 3-1** are implemented by means of $O(l)$ biological operations. Next, Step (3e) of **Algorithm 3-1** is implemented by means of O(1) biological operations. Similarly, the four DNA-based algorithms from Step (3f) through Step (3i) of **Algorithm 3-1** are implemented by means of $O(l)$ biological operations. Next, Step (3j) of **Algorithm 3-1** is implemented by means of O(1) biological operations. Each Step from Step (3a) through Step (3j) in **Algorithm 3-1** is implemented p times. Therefore, the total number of biological operations for implementing them is $O(p \times l)$. Next, the total number of biological operations for implementing Step (5a) through Step (5r) in **Algorithm 3-1** is $O(p \times l)$. Finally, Step (6) and Step (6a) of **Algorithm 3-1** are implemented by means of O(1) biological operations. Similarly proof can be used to show complexity of volume, tube and the longest DNA strand. Therefore, it is inferred that its time complexity is $O(p \times l)$ biological operations, its volume complexity is $O(2^{2\times(l+1)} \times p)$ DNA strands, its tube complexity is O(1) tubes, and its longest DNA strand is $O(p \times l \times \theta)$ base pairs. ■

V. Conclusions

With current biotechnology, the time for each operation is at least one second. Realistically, steps like gel electrophoresis take much longer, but for the sake of argument say each biological operation takes one second. From **Theorem 4-1**, if the values of p and l are equal to 10^{10}, then we need to take at least $10^{10} \times 10^{10}$ seconds which are about $10^{10} \times 317$ years.

References

[1] M. Amos, *Theoretical and Experimental DNA Computation*. New York: Springer, 2005.
[2] W.-L. Chang and A. V. Vasilakos, *Molecular Computing: Towards a Novel Computing Architecture for Complex Problem Solving*. New York: Springer, 2014.
[3] W.-L. Chang and A. V. Vasilakos, “Molecular algorithms of implementing bio-molecular databases on a biological computer,” *IEEE Trans. NanoBiosci.*, vol. 14, no. 1, pp. 104–111, Jan. 2015.
[4] W.-L. Chang, T.-T. Ren, and M. Feng, “Quantum algorithms and mathematical formulations of bio-molecular solutions of the vertex cover problem in the finite-dimensional Hilbert space,” *IEEE Trans. NanoBiosci.*, vol. 14, no. 1, pp. 121–128, Jan. 2015.