DNA Based Evolvable Instruction Set Architecture & Arithmetic Unit

(Extended Abstract)

Abstract. The paper proposes a novel DNA based concept towards the natural evolution of instruction set architecture and arithmetic unit. Silicon based conventional systems have insignificant storage and huge hardware size when compared with DNA based systems. The silicon systems can never have natural evolution. Current intelligent systems based on Artificial Intelligence concepts falsify the basic law of evolution. Huge silicon systems are designed and fabricated and treating them as idiots, a learning process is evolved to train them. It is just like allowing a baby to grow into an adult and then trying to teach them ”ABCD”. On the other hand DNAAP (DNA based Arithmetic Processing unit), naturally evolves with time into a larger intelligent system as and when it learns.

1 Introduction

A strand of DeoxyriboNucleic Acid (DNA) is encoded with four bases, represented by the letters A(Adenine), T(Thymine), C(Cytosine), and G(Guanine). The bases (also known as nucleotides) are spaced every 0.35 nanometers along the DNA molecule, giving DNA an remarkable data density of nearly 18 Mbits per inch [1]. Another important property of DNA is its double stranded nature. The bases A and T, and C and G, can bind together, forming base pairs. This complementary nature makes DNA a unique data structure for computation and can be exploited in many ways.In this paper we model a test tube as a set of DNA molecules (i.e. finite strings over the alphabet set A, C, G, T). The following operations can be performed using the test tubes[2]:

- Separate: Given a tube TU and a string of symbols S over A, C, G, T,two tubes with(TU,S) and without(TU,S) can be formed, where with(T,S) is all of the molecules of DNA in TU which contain the consecutive subsequence S and without(T,S) is all of the molecules of DNA in TU which do not contain the consecutive subsequence S.
- Merge: Given tubes T1, T2, we can form U(T1,T2) where: U(T1,T2) = T1 U T2 Note that U denotes the multiset union operation.
- Extract/Detect: Given a tube T, if T contains at least one DNA molecule respond positively, if it contains none respond negatively.
- Amplify: use PCR (polymerase chain reaction) to make copies of DNA strands
- Cut: Cut DNA strands with restriction enzymes.

After extensive analysis of basic DNA operations and its characteristics it is interesting to find that they are inherently amenable for natural evolution of hardware systems (example presented being arithmetic unit) and the concerned instruction set.

This paper presents an evolvable arithmetic unit and instruction set. The DNA operations involved in this process of natural evolution is being simulated and the outcome is along the authors expectations and simulation results will be presented in the final submission.The concepts are presented here.

When NP problems are being executed, new heuristics may be evolved to arrive at near optimal solutions in the smallest time frame in real time applications. Under such environments the self evolving DNA computer will be able to generate its own new instruction and its corresponding hardware matching the heuristics evolved. Evolution takes place over a time period which has its own scaling in the context of DNA Computing.

2 Instruction Evolution

The natural evolution of Instruction set architecture enables the DNAAP to solve NP problems. The Instruction Gene (IGENE) for the proposed evolvable Instruction set architecture along with the algorithm is presented in this section.

2.1 IGENE Format & Evolution

Current instruction formats include the mnemonic, source and destination addresses. These are stored in form of binary digits. We propose a novel instruction format for DNAAP, the IGENE (Instruction Gene) that includes

- DNA operation: The DNA operation to be performed.
- Source molecule: Data Availability for the operation.
- Sink molecule: Specific DNA address for transfer.
- Sequencing count: No. of times the instruction is executed.(Incremented after every execution)
- Threshold value: Threshold count exceeding which instruction must be evolved.
- Seed: The codon part of the new instruction is specified here.

The figure 1 depicts the conventional instruction format and the IGENE format. The instruction format is split in to two parts

- CTRL of the IGENE
- Codon of the IGENE

The motivation for the name stems from the nucleus of our origin, (i.e) the genes that store our genetic code. The proteins evolve from the Codons (Base triplets in the mRNA) and in a similar manner the Seed gives the genetic sequence for the instruction to be evolved. All the instructions are specified as sequences of the basic nucleotides (A,T,C,G).The figure 2 depicts the same.

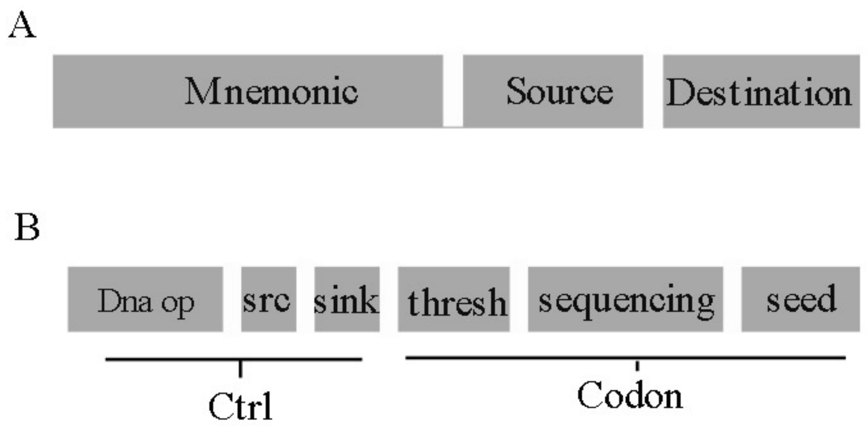

Figure 1: Conventional Instruction format(A) vs IGENE(B)

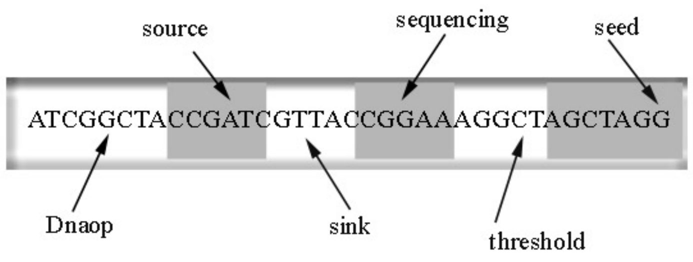

Figure 2: Genetic code of an Instruction of DNAAP

Table 1: Splitting an IGENE ; Method name: cut()

```
Input: Existing instruction genetic sequence.

Output:
 1. CTRL of the Igene
 2. Codon of the Igene

Algorithm:
1.Start.
2.Analyse the input and register the genetic
  sequence.
3.Identify the part of the instruction that must not be
  altered(i.e) the codon part.
4.Employ the cutting operation of DNA(Cut DNA with restriction
  enzymes)
5.Two strands are formed from a single instruction strand.
6.Terminate.
```

The instruction evolution process is explained using the algorithms in Tables 1, 2, 3. The evolution of ”MUL” instruction from ”ADD” instruction is discussed here as a case study.

The execution of the instruction increments the sequencing count number and as it reaches the threshold mark the new instruction evolution process begins. Once the sequencing count value equals the threshold, the seed for the new instruction is sown and results in the birth of the new instruction. The ”MUL” instruction is evolved from ”ADD” instruction as specified in the algorithm given in Table 3.

3 HGENE Format & Evolution

Molecular modules (Processing elements) in DNAAP are defined in terms of DNA strands. Each module has its own Hardware Gene (HGene) giving its genetic sequence. The authors define the genetic sequence for the adder in terms of nucleotides. The DNA implementation of adder is based on the horizontal chain reaction[3]. The HGENE specifies

Table 2: Formation of new IGENE; Method name: form()

```
Input:
1.Genetic code of the ctrl part of the new born
  instruction.
2.codon part of the Igene.

Output:Evolved Instruction Genetic sequence

Algorithm:
1.start
2.Merge the two input genetic sequences using
  Merging operation of DNA
3.This results in  the synthesis of a desired instruction
  strand.
4.stop
```

Table 3: Algorithm For ”MUL” Instruction Evolution

```
Start:
if repetitive addition of a number takes place
    for every successful addition
        Increment the "sequencing" count existing in the add instruction by one.
    end for
    if sequencing count = threshold count
        activate cut()
        activate form()
    end if
end if
```

Table 4: Algorithm for evolving molecular modules for 4-bit multiplication

```
Input: Threshold value and sequencing count for evolution of "mul"
instruction
Output: Evolved molecular modules.

  start sub replicate
    if( sequencing count > 1/4 (threshold value))
            self replicate a single adder molecular module
    end if
    if( sequencing count > 1/2 (threshold value))
            self replicate one more adder molecular module
    end if
    if( sequencing count > 3/4 (threshold value))
            self replicate one more adder molecular module
    end if
  end sub replicate
```

- Type of molecular module : Adder/Mul/...
- No. of operands : 2/3/...
- Replication bit: Initiates the self replication of the molecular module.

Evolution of molecular modules takes place due to two reasons:

- DNA computer encounters repeated arithmetic operation (say ,addition of a number)
- Reduce the overall computation time.

Evolution of multiplication molecular module is achieved by replicating the molecular modules of adder and subsequently linking them. The algorithm for the evolution of molecular module is given in Table 4.

The process of evolution of a molecular module is a continuous process. It occurs in stages, say at each stage a part of the molecular module is generated(Thresholds ratios in

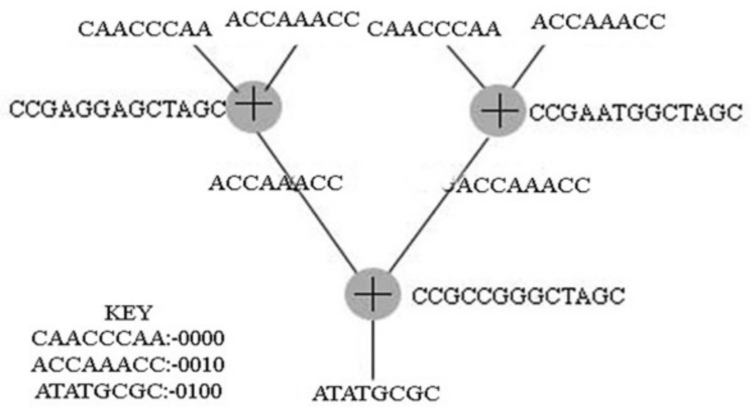

Figure 3: Example DNA sequence in evolving multiplication.

general could be any value. The values in the above algorithm are sample values). At the completion of the threshold time, the complete molecular module is evolved.

Each hardware gene(HGENE) has a replication bit. The replication bit is activated once the threshold and sequencing values are equal. The activated replication bit leads to the self replication of the molecular module.

Authors present a case study of the evolution of the multiplication molecular module. A 4-bit parallel array multiplication[9] DNA process is shown in the figure 3. The binary digits are converted to DNA sequence and the operation is performed. The result can be converted to binary digits and verified. The binary to DNA conversion is based on [10]. The partial product array is generated through DNA boolean operations [7]. Both the input and output of the 4-bit molecular module are DNA strands. The output DNA strand is made of the result of the operation and the target molecular module id. The molecular module evolution process employs two DNA operations namely,

- Self replication
- Merging or Combination

Table 5: Linking of replicated molecular modules.

```
Input : Discrete DNA molecular modules.
Output: Linked molecular modules.

Algorithm:
1.Start.
2.Analyse the pattern of arithmetic
  operations.
3.Check whether the number of replicated modules = number of bits.
4.Decide the structure of the molecular module.
5.Analyse the output DNA strand and get the target molecular
  module address.
6.Dispatch the output DNA strand as input to target molecular
  module.
7.Pass the control to the target to the target module.
8.Continue the processing.
9.Terminate
```

The adder molecular modules self replicate and are linked together using the merge operation. This linking is established by routing the output strands from adder modules to their respective target molecular modules. The algorithm in Table 5 describes the linking of the replicated modules.

References

[1] Will Ryu, DNA Computing: A Primer ,http://www.arstechnica.com/reviews/2q00/dna/dna-1.html

[2] Adam Geffen, Building A DNA Based Computer, http://transmet.org/ aj/words/dna$_comp.htm$

[3] Guarnieri, Frank, and Carter Bancroft. 1996. Use of a horizontal chain reaction for DNA-based addition. Proceedings of the Second Annual Meeting on DNA Based Computers. Providence, R.I.: American Mathematical Society .

[4] Adleman, Leonard M. 1994. Molecular computation of solutions to combinatorial problems. Science 266 (Nov. 11): 1021-1024.

[5] Guarnieri, Frank, Makiko Fliss, and Carter Bancroft. 1996. Making DNA add. Science 273 (July 12): 220-223.

[6] http://www.maa.org/mathland/

[7] Sandeep Junarkar , Making Boolean Logic Work on the Molecular Level http://www.rochester.edu/College/BIO/Ray$_Research/Boolean.html$

[8] Yali Friedman, DNA based computers http://dna2z.com/dnacpu/dna.html

[9] Kai Hwang ,Faye .A.Briggs, “Computer Architecture and Parallel Processing”, Tata Mc graw hill, computer science series, ISBN 0-07-031556-6.,1985.

[10] http://www.cs.iupui.edu/ pellison/n301/page2.html