# DNA Computer

DNA Computer can store billions of times more information then your PC hard drive and solve complex problems in a less time.We know that computer chip manufacturers are racing to make the next microprocessor that will more faster. Microprocessors made of silicon will eventually reach their limits of speed and miniaturization. Chips makers need a new material to produce faster computing speeds.

To understand DNA computing lets first examine how the conventional computer process information. A conventional computer performs mathematical operations by using electrical impulses to manipulate zeroes and ones on silicon chips. A DNA computer is based on the fact the information is “encoded” within deoxyribonucleic acid (DNA) as as patterns of molecules known as nucleotides. By manipulating the how the nucleotides combine with each other the DNA computer can be made to process data. The branch of computers dealing with DNA computers is called DNA Computing.

The concept of DNA computing was born in 1993, when Professor Leonard Adleman, a mathematician specializing in computer science and cryptography accidentally stumbled upon the similarities between conventional computers and DNA while reading a book by James Watson. A little more than a year after this, In 1994, Leonard M. Adleman, a professor at the University of Southern California, created a storm of excitement in the computing world when he announced that he had solved a famous computation problem. This computer solved the traveling salesman problem also known as the “Hamiltonian path" problem,which is explained later. DNA was shown to have massively parallel processing capabilities that might allow a DNA based computer to solve hard computational problems in a reasonable amount of time.

There was nothing remarkable about the problem itself, which dealt with finding the shortest route through a series of points. Nor was there anything special about how long it took Adleman to solve it — seven days — substantially greater than the few minutes it would take an average person to find a solution. What was exciting about Adleman’s achievement was that he had solved the problem using nothing but deoxyribonucleic acid (DNA) and molecular chemistry.

# 2. Some Informations About DNA

“Deoxyribonucleic acid”. The molecules inside cells that carry genetic information and pass it from one generation to the next. See mitosis, chromosomes.

We have heard the term DNA a million times. You know that DNA is something inside cells .We know that each and every one looks different and this is because of they are having different DNA.

Have you ever wondered how the DNA in ONE egg cell and ONE sperm cell can produce a whole human being different from any other? How does DNA direct a cell's activities? Why do mutations in DNA cause such trouble (or have a positive effect)? How does a cell in your kidney "know" that it's a kidney cell as opposed to a brain

cell or a skin cell or a cell in your eye? How can all the information needed to regulate the cell's activities be stuffed into a tiny nucleus?
A basic tenet is that all organisms on this planet, however complex they may beperceived to be,are made of the same type of genetic blueprint.The mode by which that blue print is coded is the factor that decides our physical makeup-from color of our eyes to what ever we are human.
To begin to find the answers to all these questions, you need to learn about the biological molecules called nucleic acids.
An organism (be it bacteria, rosebush, ant or human) has some form of nucleic acid Which is the chemical carrier of its genetic information. There are two types of nucleic acids, deoxyribonucleic acid (DNA) and ribonucleic acid (RNA) which code for all the information that determines the nature of the organism's cells. As a matter of fact, DNA codes for all the instructions needed for the cell to perform different functions. Did you know that human DNA contains enough information to produce about 100,000 proteins?
Genes are made up of DNA ,which is shaped like a twisted ladder with rungs made up of molecules called nucleotide bases linked together in specific pairs.The arrangement of these bases along the DNA provides the cell with instructions on making proteins. DNA is tightly coiled into rod-shaped structures called chromosomes, which are stored in the nucleus of the cell. There are 22 pairs of chromosomes in each body cell plus two sex chromosomes.

## 2.1) Structure of DNA

This structure has two helical chains each coiled round the same axis (see diagram). We have made the usual chemical assumptions, namely, that each chain consists of phosphate diester groups joining ß-D-deoxyribofuranose residues with 3',5' linkages. The two chains (but not their bases) are related by a dyad perpendicular to the fibre axis. Both chains follow right- handed helices, but owing to the dyad the sequences of the atoms in the two chains run in opposite directions.
There is a residue on each every 3.4 A. in the z-direction. We have assumed an angle of 36° between adjacent residues in the same chain, so that the structure repeats after 10 residues on each chain, that is, after 34 A. The distance of a phosphorus atom from the fibre axis is 10 A. As the phosphates are on the outside, cations have easy access.
The structure is an open one, and its water content is rather high. At lower water contents we would expect the bases to tilt so that the structure could become more compact.
The novel feature of the structure is the manner in which the two chains are held together by the purine and pyrimidine bases. The planes of the bases are perpendicular to the fibre axis. The are joined together in pairs, a single base from the other chain, so that the two lie side by side with identical z-co-ordinates. One of the pair must be a purine and the other a pyrimidine for bonding to occur.
The hydrogen bonds are made as follows : purine position 1 to pyrimidine position 1 ; purine position 6 to pyrimidine position 6.
If it is assumed that the bases only occur in the structure in the most plausible tautomeric forms (that is, with the keto rather than the enol configurations) it is found that only specific pairs of bases can bond together. These pairs are : adenine (purine) with thymine (pyrimidine), and guanine (purine) with cytosine (pyrimidine).

In other words, if an adenine forms one member of a pair, on either chain, then on these assumptions the other member must be thymine ; similarly for guanine and cytosine. The sequence of bases on a single chain does not appear to be restricted in any way. However, if only specific pairs of bases can be formed, it follows that if the sequence of bases on one chain is given, then the sequence on the other chain is automatically determined.

It has been found experimentally (3,4) that the ratio of the amounts of adenine to thymine, and the ration of guanine to cytosine, are always bery close to unity for deoxyribose nucleic acid.

It is probably impossible to build this structure with a ribose sugar in place of the deoxyribose, as the extra oxygen atom would make too close a van der Waals contact. The previously published X-ray data (5,6) on deoxyribose nucleic acid are insufficient for a rigorous test of our structure. So far as we can tell, it is roughly compatible with the experimental data, but it must be regarded as unproved until it has been checked against more exact results. Some of these are given in the following communications. We were not aware of the details of the results presented there when we devised our structure, which rests mainly though not entirely on published experimental data and stereochemical arguments.

It has not escaped our notice that the specific pairing we have postulated immediately suggests a possible copying mechanism for the genetic material.

## 2.2) Arrangement of Nucleotieds in DNA

### One strands

Strands of DNA are long polymers of millions of linked nucleotides. These nucleotides consist of one of four nitrogen bases, a five carbon sugar and a phosphate group. The nucleotides that make up these polymers are named alter,the nitrogen bases that comprise it, namely, Adenine (A), Cytosine (C), Guanine (G), and Thymine (T). These nucleotides only combine in such a way that C always pairs with G, and T always pairs with A. These two strands of a DNA molecule are anti-parallel in that each strand runs in a opposite direction. Here below figure shows two strands of DNA and the bonding principles of the four types of nucleotides.

The linkage of the sugar-phosphate "backbone" of a single DNA strand is such that there is a directionality. That is, the phosphate on the 5' carbon of deoxyribose is linked to the 3' carbon of the next deoxyribose. This lends a directionality to a DNA strand which is said to have a 5' to 3' direction. The two strands of a DNA double helix are arranged in opposite directions and are said to be anti-parallel in that one strand is 5' - 3' and the complementary strand is 3' - 5'.

### Double Helix

The particular order of the bases arranged along the suger-phosphate backbone is called the DNA sequnce and the combinations of the four nucleotides in the estimated millions long polymer strands results in a billions of combinations within a single DNA double helix. These massive amounts of combinations allow for the multitude of differences between every living thing on the plane-form the large scale (for example, mammals as opposed to plants)to the small scale (differences in human hair colour). Here the above fig. Shows the double helix shape of the DNA.

## 3. Operations on DNA

While a number of equivalent formalizations exist, we follow the descriptions. Note that the types of operations available are result of the capability of molecular biology rather than the wishes of algorithms designers. Also note that this algorithms are performed in constant time on testtubes which, for the sake of this discussion, may be of arbitrary size this operations are:

1) MERGE

This is the simple operations of combining the contents of two test tubes in a third tube.

2) ANNEAL

This is the process by which complementary strands of DNA are paired to form the famous double-helix structure of Watson and crick. Annealing is achieved by cooling a DNA solution, which encourages pairing. Adleman uses this in step 1 to generate all legal paths through the graph.

3) MELT

Melting is inverse operation of annealing. By heating the contents of a tube, double-stranded DNA sequences are denatured, or separated into its two single-stranded parts.

4) SEPERATION BY LENGTH

The contents of test tube can be separated by increasing length. This is achieved by hel electrophoresis, whereby longer strands travel more slowly through the gel. This operation was used by Adleman in step 3 of his solution to HP.

5) SEPERATION BY SEQUENCE

This operation allows one to remove from solution all the DNA strands that contain a desired sequence. This is performed by generating the strands whose complement is the desired sequence. This newly generated strands is attached to magnetic substance which is used to extract the sequences after annealing. This operation crux of Adleman's step 4.

6) COPYING/AMPLIFICATION

Copies are made of DNA strands in a test tube. The strands to be copied must have known sequences at both the beginning and end in order for this operation to be performed.

7) APPEND

This process makes a DNA strand longer by adding a character or strand to the end of each sequence.

8) DETECT

It is also possible to analyze test tube inorder to determine whether or not it contains atleast one strand of DNA.
This operation, for example, is the last in Adleman's algorithm where we attempt to find a DNA sequence that has survived the previous steps.

# 4. Aldeman's Hamilton path problem

## The Hamiltonian Path problem

In 1994, Leonard M. Adleman solved an unremarkable computational problem with a remarkable technique. It was a problem that a person could solve it in a few moments or an average desktop machine could solve in the blink of an eye. It took Adleman, however, seven days to find a solution. Why then was this work exceptional? Because he solved the problem with DNA. It was a landmark demonstration of computing on the molecular level.
The type of problem that Adleman solved is a famous one. It's formally known as a directed Hamiltonian Path (HP) problem, but is more popularly recognized as a variant of the so-called "traveling salesman problem." In Adleman's version of the traveling salesman problem, or "TSP" for short, a hypothetical salesman tries to find a route through a set of cities so that he visits each city only once. As the number of cities increases, the problem becomes more difficult until its solution is beyond analytical analysis altogether, at which point it requires brute force search methods. TSPs with a large number of cities quickly become computationally expensive, making them impractical to solve on even the latest super-computer. Adleman's demonstration only involves seven cities, making it in some sense a trivial problem that can easily be solved by inspection. Nevertheless, his work is significant for a number of reasons.
It illustrates the possibilities of using DNA to solve a class of problems that is difficult or impossible to solve using traditional computing methods.
It's an example of computation at a molecular level, potentially a size limit that may never be reached by the semiconductor industry. It demonstrates unique aspects of DNA as a data structure. It demonstrates that computing with DNA can work in a massively parallel fashion.

# 5. The Adleman's experiment

There is no better way to understand how something works than by going through an example step by step. So let's solve our own directed Hamiltonian Path problem, using the DNA methods demonstrated by Adleman. The concepts are the same but the example has been simplified to make it easier to follow and present.
Suppose that I live in Boston, and need to visit four cities: Atlanta, San Diego , St.Louis, and NY, with NY being my final destination. The airline I'm taking has a

specific set of connecting flights that restrict which routes I can take (i.e. there is a flight from Boston. to San Diego, but no flight from St.Louis to San Diego). What should my itinerary be if I want to visit each city only once?

*Figure 1. A sample traveling salesman problem involving the shortest path connecting all cities. Arrows indicate the direction that someone can travel. For example, a voyager can leave Atlanta and arrive in St. Louis, and vice versa*

It should take you only a moment to see that there is only one route. Starting from Boston you need to fly to San Diego , Atlanta, St.Louis and then to N.Y. Any other choice of cities will force you to miss a destination, visit a city twice, or not make it to N.Y. For this example you obviously don't need the help of a computer to find a solution. For six, seven, or even eight cities, the problem is still manageable. However, as the number of cities increases, the problem quickly gets out of hand. Assuming a random distribution of connecting routes, the number of itineraries you need to check increases exponentially.

Pretty soon you will run out of pen and paper listing all the possible routes, and it becomes a problem for a computer.....or perhaps DNA. The method Adleman used to solve this problem is basically the shotgun approach mentioned previously. He first generated all the possible itineraries and then selected the correct itinerary. This is the advantage of DNA. It's small and there are combinatorial techniques that can quickly generate many different data strings. Since the enzymes work on many DNA molecules at once, the selection process is massively parallel.

Specifically, the method based on Adleman's experiment would be as follows:

1) Generate all possible routes.
2) Select itineraries that start with the proper city and end with the final city.
3) Select itineraries with the correct number of cities.
4) Select itineraries that contain each city only once.

All of the above steps can be accomplished with standard molecular biology techniques.

**Part I: Generate all possible routes**

Strategy : Encode city names in short DNA sequences. Encode itineraries by connecting the city sequences for which routes exist.

DNA can simply be treated as a string of data. For example, each city can be represented by a "word" of six bases:

| | |
|---|---|
| Boston | GCTACG |
| San Diego | CTAGTA |
| Atlanta | TCGTAC |
| St.Louis | CTACGG |
| New York | ATGCCG |

The entire itinerary can be encoded by simply stringing together these DNA sequences that represent specific cities. For example, the route from Boston -> San Diego -> Atlanta -> St.Louis -> New York would simply be

GCTACGCTAGTATCGTACCTACGGATGCCG, or equivalently it could be represented in double stranded form with its complement sequence.

So how do we generate this? Synthesizing short single stranded DNA is now a routine process, so encoding the city names is straightforward. The molecules can be made by a machine called a DNA synthesizer or even custom ordered from a third party. Itineraries can then be produced from the city encodings by linking them together in proper order. To accomplish this you can take advantage of the fact that DNA hybridizes with its complimentary sequence.

For example, you can encode the routes between cities by encoding the compliment of the second half (last three letters) of the departure city and the first half (first three letters) of the arrival city. For example the route between St.Louis (CTACGG) and NY (ATGCCG) can be made by taking the second half of the coding for St.Louis (CGG) and the first half of the coding for NY (ATG). This gives CGGATG. By taking the complement of this you get, GCCTAC, which not only uniquely represents the route from St.Louis to NY, but will connect the DNA representing St.Louis and NY by hybridizing itself to the second half of the code representing St.Louis (...CGG) and the first half of the code representing NY (ATG...). For example:

Random itineraries can be made by mixing city encodings with the route encodings. Finally, the DNA strands can be connected together by an enzyme called ligase. What we are left with are strands of DNA representing itineraries with a random number of cities and random set of routes. For example:

We can be confident that we have all possible combinations including the correct one by using an excess of DNA encodings, say 10^13 copies of each city and each route between cities. Remember DNA is a highly compact data format, so numbers are on our side.

**Part II: Select itineraries that start and end with the correct cities**

Strategy: Selectively copy and amplify only the section of the DNA that starts with LA and ends with NY by using the Polymerase Chain Reaction.

After Part I, we now have a test tube full of various lengths of DNA that encode possible routes between cities. What we want are routes that start with Boston and end with NY. To accomplish this we can use a technique called Polymerase Chain Reaction (PCR), which allows you to produce many copies of a specific sequence of DNA. PCR is an iterative process that cycles through a series of copying events using an enzyme called polymerase. Polymerase will copy a section of single stranded DNA starting at the position of a primer, a short piece of DNA complimentary to one end of a section of the DNA that you're interested in.

By selecting primers that flank the section of DNA you want to amplify, the polymerase preferentially amplifies the DNA between these primers, doubling the amount of DNA containing this sequence. After many iterations of PCR, the DNA you're working on is amplified exponentially. So to selectively amplify the itineraries that start and stop with our cities of interest, we use primers that are complimentary to Boston and NY. What we end up with after PCR is a test tube full of double stranded DNA of various lengths, encoding itineraries that start with Boston and end with NY.

**Part III: Select itineraries that contain the correct number of cities**

Strategy: Sort the DNA by length and select the DNA whose length corresponds to 5 cities.

Our test tube is now filled with DNA encoded itineraries that start with Boston and end with NY, where the number of cities in between Boston and NY varies. We now want to select those itineraries that are five cities long. To accomplish this we can use a technique called Gel Electrophoresis, which is a common procedure used to resolve the size of DNA. The basic principle behind Gel Electrophoresis is to force DNA through a gel matrix by using an electric field. DNA is a negatively charged molecule under most conditions, so if placed in an electric field it will be attracted to the positive potential.

However since the charge density of DNA is constant (charge per length) long pieces of DNA move as fast as short pieces when suspended in a fluid. This is why you use a gel matrix. The gel is made up of a polymer that forms a meshwork of linked strands. The DNA now is forced to thread its way through the tiny spaces between these strands, which slows down the DNA at different rates depending on its length. What we typically end up with after running a gel is a series of DNA bands, with each band corresponding to a certain length. We can then simply cut out the band of interest to isolate DNA of a specific length. Since we known that each city is encoded with 6 base pairs of DNA, knowing the length of the itinerary gives number of cities. In this case we would isolate the DNA that was 30 base pairs long (5 cities times 6 base pairs).

**Part IV: Select itineraries that have a complete set of cities**

Strategy: Successively filter the DNA molecules by city, one city at a time. Since the DNA we start with contains five cities, we will be left with strands that encode each city once.

DNA containing a specific sequence can be purified from a sample of mixed DNA by a technique called affinity purification. This is accomplished by attaching the compliment of the sequence in question to a substrate like a magnetic bead. The beads are then mixed with the DNA. DNA, which contains the sequence you're after then hybridizes with the complement sequence on the beads. These beads can then be retrieved and the DNA isolated.

So we now affinity purify fives times, using a different city complement for each run. For example, for the first run we use Boston.'-beads (where the ' indicates compliment strand) to fish out DNA sequences which contain the encoding for Boston (which should be all the DNA because of step 3), the next run we use Atlanta '-beads, and then San Diego '-beads, St.Louis '-beads, and finally NY'-beads.

The order isn’t important. If an itinerary is missing a city, then it will not be "fished out" during one of the runs and will be removed from the candidate pool. What we are left with are the are itineraries that start in Boston, visit each city once, and end in NY. This is exactly what we are looking for. If the answer exists we would retrieve it at this step.

## Reading out the answer

One possible way to find the result would be to simply sequence the DNA strands. However, since we already have the sequence of the city encodings we can use an alternate method called graduated PCR. Here we do a series of PCR amplifications using the primer corresponding to Boston, with a different primer for each city in succession. By measuring the various lengths of DNA for each PCR product we can piece together the final sequence of cities in our itinerary. For example, we know that the DNA itinerary starts with Boston and is 30 base pairs long, so if the PCR product for the LA and Atlanta primers was 24 base pairs long, you know Atlanta is the fourth city in the itinerary (24 divided by 6). Finally, if we were careful in our DNA manipulations the only DNA left in our test tube should be DNA itinerary encoding Boston, San Diego, St.Louis, Atlanta, and NY. So if the succession of primers used is Boston & San Diego, Boston & St.Louis, Boston & Atlanta, and Boston & NY, then we would get PCR products with lengths 12, 18, 24, and 30 base pairs.

### Caveats

Adleman's experiment solved a seven city problem, but there are two major shortcomings preventing a large scaling up of his computation. The complexity of the traveling salesman problem simply doesn't disappear when applying a different method of solution - it still increases exponentially. For Adleman's method, what scales exponentially is not the computing time, but rather the amount of DNA. Unfortunately this places some hard restrictions on the number of cities that can be solved; after the Adleman article was published, more than a few people have pointed out that using his method to solve a 200 city HP problem would take an amount of DNA that weighed more than the earth. Another factor that places limits on his method is the error rate for each operation. Since these operations are not deterministic but stochastically driven (we are doing chemistry here), each step contains statistical errors, limiting the number of iterations you can do successively before the probability of producing an error becomes greater than producing the correct result. For example an error rate of 1% is fine for 10 iterations, giving less than 10% error, but after 100 iterations this error grows to 63%.

## Conclusions

So will DNA ever be used to solve a traveling salesman problem with a higher number of cities than can be done with traditional computers? Well, considering that the record is a whopping 13,509 cities, it certainly will not be done with the procedure described above. It took this group only three months, using three Digital AlphaServer 4100s (a total of 12 processors) and a cluster of 32 Pentium-II PCs. The solution was possible not because of brute force computing power, but because they used some very efficient branching rules. This first demonstration of DNA computing used a rather unsophisticated algorithm, but as the formalism of DNA computing becomes refined, new algorithms perhaps will one day allow DNA to overtake conventional computation and set a new record.

On the side of the "hardware" (or should I say "wetware"), improvements in biotechnology are happening at a rate similar to the advances made in the semiconductor industry. For instance, look at sequencing; what once took a graduate

student 5 years to do for a Ph.D thesis takes Celera just one day. With the amount of government funded research dollars flowing into genetic-related R&D and with the large potential payoffs from the lucrative pharmaceutical and medical-related markets, this isn't surprising. Just look at the number of advances in DNA-related technology that happened in the last five years. Today we have not one but several companies making "DNA chips," where DNA strands are attached to a silicon substrate in large arrays (for example Affymetrix's genechip). Production technology of MEMS is advancing rapidly, allowing for novel integrated small scale DNA processing devices. The Human Genome Project is producing rapid innovations in sequencing technology. The future of DNA manipulation is speed, automation, and miniaturization.

And of course we are talking about DNA here, the genetic code of life itself. It certainly has been the molecule of this century and most likely the next one. Considering all the attention that DNA has garnered, it isn't too hard to imagine that one day we might have the tools and talent to produce a small integrated desktop machine that uses DNA, or a DNA-like biopolymer, as a computing substrate along with set of designer enzymes. Perhaps it won't be used to play Quake IV or surf the web -- things that traditional computers are good at -- but it certainly might be used in the study of logic, encryption, genetic programming and algorithms, automata, language systems, and lots of other interesting things that haven't even been invented yet.

# 6. How DNA Computers Will Work

## 6.1) A Fledgling Technology

DNA computers can't be found at your local electronics store yet. The technology is still in development, and didn't even exist as a concept a decade ago. In 1994, Leonard Adleman introduced the idea of using DNA to solve complex mathematical problems. Adleman, a computer scientist at the University of Southern California, came to the conclusion that DNA had computational potential after reading the book "Molecular Biology of the Gene," written by James Watson, who co-discovered the structure of DNA in 1953. In fact, DNA is very similar to a computer hard drive in how it stores permanent information about your genes.

Adleman is often called the inventor of DNA computers. His article in a 1994 issue of the journal Science outlined how to use DNA to solve a well-known mathematical problem, called the directed Hamilton Path problem, also known as the "traveling salesman" problem. The goal of the problem is to find the shortest route between a number of cities, going through each city only once. As you add more cities to the problem, the problem becomes more difficult. Adleman chose to find the shortest route between seven cities.

You could probably draw this problem out on paper and come to a solution faster than Adleman did using his DNA test-tube computer. Here are the steps taken in the Adleman DNA computer experiment:

Strands of DNA represent the seven cities. In genes, genetic coding is represented by the letters A, T, C and G. Some sequence of these four letters represented each city and possible flight path.

These molecules are then mixed in a test tube, with some of these DNA strands sticking together. A chain of these strands represents a possible answer.
Within a few seconds, all of the possible combinations of DNA strands, which represent answers, are created in the test tube.
Adleman eliminates the wrong molecules through chemical reactions, which leaves behind only the flight paths that connect all seven cities.
The success of the Adleman DNA computer proves that DNA can be used to calculate complex mathematical problems. However, this early DNA computer is far from challenging silicon-based computers in terms of speed. The Adleman DNA computer created a group of possible answers very quickly, but it took days for Adleman to narrow down the possibilities. Another drawback of his DNA computer is that it requires human assistance. The goal of the DNA computing field is to create a device that can work independent of human involvement.

Three years after Adleman's experiment, researchers at the University of Rochester developed logic gates made of DNA. Logic gates are a vital part of how your computer carries out functions that you command it to do. These gates convert binary code moving through the computer into a series of signals that the computer uses to perform operations. Currently, logic gates interpret input signals from silicon transistors, and convert those signals into an output signal that allows the computer to perform complex functions.
The Rochester team's DNA logic gates are the first step toward creating a computer that has a structure similar to that of an electronic PC. Instead of using electrical signals to perform logical operations, these DNA logic gates rely on DNA code. They detect fragments of genetic material as input, splice together these fragments and form a single output. For instance, a genetic gate called the "And gate" links two DNA inputs by chemically binding them so they're locked in an end-to-end structure, similar to the way two Legos might be fastened by a third Lego between them. The researchers believe that these logic gates might be combined with DNA microchips to create a breakthrough in DNA computing.
DNA computer components -- logic gates and biochips -- will take years to develop into a practical, workable DNA computer. If such a computer is ever built, scientists say that it will be more compact, accurate and efficient than conventional computers. In the next section, we'll look at how DNA computers could surpass their silicon-based predecessors, and what tasks these computers would perform.

# 7. COMPARISON OF DNA AND CONVENTIONAL ELECTRONIC COMPUTERS

As we have discussed the concepts and characteristics of DNA Computer, we can now compare the DNA Computers with Conventional Electronic Computers.

## 7.1) Similarities

**1) Transformation of Data**
Both DNA computers and electronic computers use Boolean logic (AND, OR, NAND, NOR) to transform data. The logical command "AND" is performed by

separating DNA strands according to their sequences, and the command "OR" is done by pouring together DNA solutions containing specific sequences. For example, the logical statement "X or Y" is true if X is true or if Y is true. To simulate that, the scientists would pour the DNA strands corresponding to "X" together with those corresponding to "Y."[2][3]. Following is an example of how a Bio Chemical Inverter works.

Bio-chemical Inverter: The characteristics of natural gene regulation systems can be exploited to design in vivo logic circuits (Weiss et al., 1999).

How a biochemical inverter achieves the two states in digital inversion using genetic regulatory elements? Here, the concentration of a particular messenger RNA (mRNA) molecule represents a logic signal. In the first case, the input mRNA is absent and the cell transcribes the gene for the output mRNA using RNA polymerase (RNAp) molecules. In the second case, the input mRNA is present and the cell translates the input mRNA into the input protein using ribosomes.

A digital inverter that consists of a gene encoding the instructions for protein B and containing a region (P) to which protein A binds. When A is absent (left)—a situation representing the input bit 0—the gene is active and B is formed—corresponding to an output bit 1. When A is produced (right)—making the input bit 1—it binds to P and blocks the action of the gene—preventing B from being formed and making the output bit 0.

The input protein then binds specifically to the gene at the promoter site (labeled \P") and prevents the cell from synthesizing the output mRNA.

Now more complete picture that explains the role of transcription and translation cellular processes in inversion is explained here.

Biochemical inversion uses the transcription and translation cellular processes. Ribosomal RNA translates the input mRNA into an amino acid chain, which then folds into a three-dimensional protein structure. When the protein binds an operator of the gene's promoter, it prevents transcription of the gene by RNA polymerase (RNAp). In the absence of the repressor protein, RNAp transcribes the gene into the output mRNA.

It depicts a functional model of the inverter derived from its biochemical reaction phases. The first phase in inversion is the translation stage, denoted as L. The input signal to this stage, and thus the inverter, corresponds to the concentration level of the input mRNA, φA. Ribosomal RNA (rRNA) translates the input mRNA into the input repressor protein, ψA , where L represents the steady state mapping between the mRNA and protein concentrations. The relationship between the input mRNA and repressor protein is initially linear, with increases in φA corresponding to increases in ψA, until an asymptotic boundary is reached. The properties of this boundary are determined by characteristics of the cell such as amino acid synthesis capabilities, the efficiency of the ribosome-binding site, and mRNA stability. Since cells degrade mRNA as well as protein molecules, constant synthesis of the input mRNA is needed to maintain a steady level of the input repressor protein. In the second phase, input protein monomers combine to form polymers that bind the operator, and subsequently repress the transcription of the output gene.

The Functional composition of the inversion stages: the translation stage maps input mRNA levels (ψA) to input protein levels (φA), the cooperative binding stage maps input protein levels to bound operator levels (ρA), and the transcription stage maps bound operator levels to output mRNA levels (ψZ). The degradation of the mRNA and protein molecules is represented with the electrical ground symbol. The degradation of mRNA is part of the translation stage, while the degradation of

proteins is part of the cooperative binding stage. The graphs illustrate the steady-state relationships for each of these stages and the overall inversion function that results from combining these stages.

This cooperative binding, which ensures that only dimerized proteins can bind the DNA, decreases the digital noise. Let us define the concentration of operator that is bound to the repressor, or the strength of the repression, as ρA. In addition, denote the cooperative binding stage that occurs between ψA and ρA as C. In steady state, the relationship between ψA and ρA is sigmoidal. At low levels of ψA, the strength of repression does not increase significantly for increases in ρA because these concentrations are too low for appreciable dimerization. At higher concentrations of ψA, however, considerable dimerization occurs, resulting in a nonlinear increase in repression activity. For values of ψA approaching saturation, the operator is mostly bound, and repressor activity is close to maximal. At this point, increasing the concentration of ψA does not increase repression, and instead causes the ψA/ρA curve to move toward an asymptotic boundary. In this way, the cooperative binding stage performs signal restoration in which the analog output signal better represents the appropriate digital meaning than the corresponding analog input signal. Because each stage of the computation reduces the noise in the system through signal restoration, multiple inverters can be combined into more complex circuits, while still maintaining or even increasing the overall reliability of the system.

In the final stage of the inverter, the transcription stage, RNA polymerase (RNAp) transcribes the regulated gene and the input signal is inverted. Let us define Z to be the output signal of the inverter and ψZ to be its corresponding mRNA concentration. The transcription stage, with input ρA and output φZ, has a steady state relationship in which increases in ρA correspond to monotonic decreases in φZ. During periods of minimal repression, transcription progresses at rapid rates resulting in maximal concentrations of φZ. However, for high levels of repression, the transcriptional activity declines and the level of φZ drops.

Overall, the three stages combine to form a system that behaves as an inverter, negating the input mRNA signal, φA, to yield the output mRNA signal, φZ. Furthermore, with efficient signal restoration during the cooperative binding stage of inversion, complex but reliable digital logic circuits are attainable.

**2) Manipulation of Data**

Electronic computers and DNA computers both store information in strings, which are manipulated to do processes. Vast quantities of information can be stored in a test tube. The information could be encoded into DNA sequences and the DNA could be stored. To retrieve data, it would only be necessary to search for a small part of it - a key word, for example – by adding a DNA strand designed so that its sequence sticks to the key word wherever it appears on the DNA [3].

**3) Computation Ability**

All computers manipulate data by addition and subtraction. A DNA computer should be able to solve a satisfiability problem with 70 variables and 1,000 AND-OR connections. To solve it, assign various DNA sequences to represent 0's and 1's at the various positions of a 70 digit binary number. Vast numbers of these sequences would be mixed together, generating longer molecules corresponding to every possible 70-digit sequence [2][3].

## 7.2) Differences

**1) Size**

Conventional computers are about 1 square foot for the desktop and another square foot for the monitor. One new proposal is for a memory bank containing more than a pound of DNA molecules suspended in about 1,000 quarts of fluid, in a bank about a yard square. Such a bank would be more capacious than all the memories of all the computers ever made.

The first ever-electronic computer (Eniac) took up a large room whereas the first DNA computer (Adleman) was 100 micro liters. Adleman dubbed his DNA computer the

TT-100, for test tube filled with 100 micro liters, or about one-fiftieth of a teaspoon of fluid, which is all it took for the reactions to occur.

**2) Representation of Data**

DNA computers use Base4 to represent data, whereas electronic computers use Base2 in the form of 1's and 0's. The nitrogen bases of DNA are part of the basic building blocks of life. Using this four letter alphabet, DNA stores information that is manipulated by living organisms in almost exactly the same way computers work their way through strings of 1's and 0's.

**3) Parallelism**

Electronic computers typically handle operations in a sequential manner. Of course, there are multi-processor computers, and modern CPUs incorporate some parallel processing, but in general, in the basic Von Neumann architecture computer [4], instructions are handled sequentially. A von Neumann machine, which is what all modern CPUs are, basically repeats the same "fetch and execute cycle" over and over again; it fetches an instruction and the appropriate data from main memory, and it executes the instruction. It does this many, many times in a row, really, really fast. The great Richard Feynman [5], in his Lectures on Computation, summed up von Neumann computers by saying, "the inside of a computer is as dumb as hell, but it goes like mad!" DNA computers, however, are non-von Neuman, stochastic machines that approach computation in a different way from ordinary computers for the purpose of solving a different class of problems. Typically, increasing performance of silicon computing means faster clock cycles (and larger data paths), where the emphasis is on the speed of the CPU and not on the size of the memory.

For example, will doubling the clock speed or doubling your RAM give you better performance? For DNA computing, though, the power comes from the memory capacity and parallel processing. If forced to behave sequentially, DNA loses its appeal. For example, let's look at the read and write rate of DNA. In bacteria, DNA can be replicated at a rate of about 500 base pairs a second. Biologically this is quite fast (10 times faster than human cells) and considering the low error rates, an impressive achievement. But this is only 1000 bits/sec, which is a snail's pace when compared to the data throughput of an average hard drive. But look what happens if you allow many copies of the replication enzymes to work on DNA in parallel. First

of all, the replication enzymes can start on the second replicated strand of DNA even before they're finished copying the first one. So already the data rate jumps to 2000 bits/sec. But look what happens after each replication is finished - the number of DNA strands increases exponentially (2n after n iterations). With each additional strand, the data rate increases by 1000 bits/sec. So after 10 iterations, the DNA is being replicated at a rate of about 1Mbit/sec; after 30 iterations it increases to 1000 Gbits/sec. This is beyond the sustained data rates of the fastest hard drives.

**4) Material**

Obviously, the material used in DNA Computers is different than in Conventional Electronic Computers. Generally, people take a variety of enzymes such as restriction nuclease and ligase as the hardware of DNA Computers, encoded double-stranded or single-stranded DNA molecules as software and data are stored in the sequences of base pairs. As for conventional electronic computers, electronic devices compose hardware. Software and data are stored in the organized structure of electronic devices represented by the electrical signals.

The other difference between DNA Computers and conventional electronic computers in material is the reusability. The materials used in DNA Computer are not reusable. Whereas an electronic computer can operate indefinitely with electricity as its only input, a DNA computer would require periodic refueling and cleaning. On the other side, until now, the molecular components used are still generally specialized. In the current research of DNA Computing, very different sets of oligonucleotides are used to solve different problems.

**5) Methods of Calculation:**

By synthesizing particular sequences of DNA, DNA computers carry out calculations. Conventional computers represent information physically expressed in terms of the flow of electrons through logical circuits. Builders of DNA computers represent information in terms of the chemical units of DNA. Calculating with an ordinary computer is done with a program that instructs electrons to travel on particular paths; with a DNA computer, calculation requires synthesizing particular sequences of DNA and letting them react in a test tube [3]. As it is, the basic manipulations used for DNA Computation include Anneal, Melt, Ligate, Polymerase Extension, Cut, Destroy, Merge, Separate by Length which can also be combined to high level manipulations such as Amplify, Separate by Subsequence, Append, Mark, Unmark. And the most famous example of a higher-level manipulation is the polymerase chain reaction (PCR).

# 8. Advantages of DNA Computers

**1) Parallelism**

"The speed of any computer, biological or not, is determined by two factors: (i) how many parallel processes it has; (ii) how many steps each one can perform per unit time. The exciting point about biology is that the first of these factors can be very large: recall that a small amount of water contains about 1022 molecules. Thus, biological computations could potentially have vastly more parallelism than conventional ones."[6]

In November of 1994, Leonard Adleman published a dramatic reminder that computation is independent of any particular substrate. By using strands of DNA annealing to each other, he was able to compute a solution to an instance of the Hamiltonian path problem (HPP) (Figure 4). While working in formal language theory and artificial selection of RNA had presaged the concept of using DNA to do computation, these precursors had largely gone unnoticed in mainstream computer science. Adleman's work sparked intense excitement and marked the birth of a new field, DNA computation [7].

## The Hamiltonian Path problem

The goal is to find a path from the start city to the end city going through every city only once.

The Hamiltonian Path problem is shown in Figure 3. To solve this problem Adleman used a non-deterministic algorithm (brute force method) to solve this problem. The main thinking of using DNA other than electronic computer to solve this problem is the parallelism of DNA operations. In fact, the real interesting thing on the DNA solution for the Hamiltonian path problems is that most input data grow just linearly with the growth of the number of edges.

That means it is almost impossible to solve this kind of problems (NP or NP-Compete) using a normal computer when the complexity of the problem grows because they must try each option one at a time. While, as for DNA based computers, just the quantity of DNA's should grow exponentially but this is not a problem because the quantity of DNA's for all known problems is small enough. (In reasonable concentrations, a liter of DNA solution can store up to 1022 bits of information [8]) They can try all of the options at the same time, determining possible solutions while weeding out wrong answers.

Let's now look a little bit more deeply into the biochemical operation. In the cell, DNA is modified biochemically by a variety of enzymes, which are tiny protein machines that read and process DNA according to nature's design. There is a wide variety and number of these "operational" proteins, which manipulate DNA on the molecular level. For example, there are enzymes that cut DNA and enzymes that paste it back together. Other enzymes function as copiers, and others as repair units. Molecular biology, Biochemistry, and Biotechnology have developed techniques that allow us to perform many of these cellular functions in the test tube.

It's this cellular machinery, along with some synthetic chemistry, that makes up the palette of operations available for computation. Just like a CPU has a basic suite of operations like addition, bit-shifting, logical operators (AND, OR, NOT NOR), etc. that allow it to perform even the most complex calculations, DNA has cutting, copying, pasting, repairing, and many others. And note that in the test tube, enzymes do not function sequentially, working on one DNA molecules at a time. Rather, many copies of the enzyme can work on many DNA molecules simultaneously. So this is the power of DNA computing that it can work in a massively parallel fashion.

**2) Gigantic memory capacity**

Just as we have discussed, the other implicit characteristic of DNA Computer is its gigantic memory capacity. Storing information in molecules of DNA allows for an information density of approximately 1 bit per cubic nanometer. The bases (also known as nucleotides) of DNA molecules, which represent the minimize unit of

information in DNA Computers, are spaced every 0.34 nanometers along the DNA molecule (Figure 4), giving DNA a remarkable data density of nearly 18 Megabits per inch. In two dimensions, if you assume one base per square nanometer, the data density is over one million Gigabits per square inch. Compare this to the data density of a typical high performance hard driver, which is about 7 gigabits per square inch -- a factor of over 100,000 smaller [8]. Researchers from Pacific Northwest National Laboratory are tapping forces of nature to store information more permanently. The researchers used artificial DNA sequences to encode portions of the text of the children's song it's a Small World, added the sequences to bacteria DNA, allowed the bacteria to multiply, and then extracted the message part of a DNA strand and retrieved the encoded information. Because DNA is passed down through generations of living organisms, information stored this way should survive for as long as the line of organisms survives, said Pak Wong, a chief scientist at the Pacific Northwest National Laboratory.

Storing information is DNA's natural function, said Wong. "We [are] taking advantage of a time-tested, natural, nanoscale data storage technology perfected over the last 3 billion years." The encoding method could be used to store any digital information, he said. "Text, pictures, music -- anything you can send or receive over the Web could be saved in this form."

**3) Low Power Dissipation**

“The potential of DNA-based computation lies in the fact that DNA has a gigantic memory capacity and also in the fact that the biochemical operations dissipate so little energy,” says University of Rochester computer scientist Mitsunori Ogihara [10]. DNA computers can perform 2 x 1019 ligation operations per joule. This is amazing, considering that the second law of thermodynamics dictates a theoretical maximum of 34 x 1019 (irreversible) operations per joule (at 300K). Existing supercomputers aren’t very energy-efficient, executing a maximum of 109 operations per joule [11]. Just think about the energy could be very valuable in future. So, this character of DNA computers can be very important.

4) Suitable For Ambinatorial Problems:-

From the first day that DNA Computation is developed, Scientists used it to solve combinatorial problems. In 1994, Leonard Adleman used DNA to solve one of Hamiltonian Path problem -Traveling Salesman problem. After that Lipton used DNA Computer to break Data Encryption Standard (DES) [12]. And then much of the work on DNA computing has continued to focus on solving NP-complete and other hard computational problems. In fact, experiments have proved that DNA Computers are suitable for solving complex combinatorial problems, even until now, it costs still several days to solve the problems like Hamiltonian Path problems. But the key point is that Adleman's original and subsequent works demonstrated the ability of DNA Computers to obtain tractable solutions to NP-complete and other hard computational problems, while these are unimaginable using conventional computers.

**5) Clean, Cheap And Available**

Besides above characteristics, clean, cheap and available are easily found from performance of DNA Computer. It is clean because people do not use any harmful material to produce it and also no pollution generates. It is cheap and available because you can easily find DNA from nature while it’s not necessary to exploit

mines and that all the work you should do is to extract or refine the parts that you need from organism.
DNA processors are cheaper and cleaner than today's silicon-based microprocessors. DNA resources are also more readily available than traditional microprocessor's. The field is highly multidisciplinary, attracting a host of extremely bright computer scientists, molecular biologists, geneticists, mathematicians, physicists, and others. Because of DNA computer's massive parallel processing powers (about 10E20 computations a second), computations that would take years to be done on a conventioal computer could be computed in minutes. Certain operations in DNA computing are also over a billion times more energy efficient than conventional computers. DNA stores information at a density of about one bit per cubed nm—about a trillion times as efficiently as videotape. In addition to its potential applications, such as DNA computation, nanofabrication, storage devices, sensing, and healthcare, biocomputation also has implications for basic scientific research.

## 8.1) Key benefits

Today, the new Pentium 4 has a massive 42 million electronic switches. According to recent stastics, one cubic-centimeter of DNA material can store a upto 10E21 bits of information, whereas the current computer have a maximum memory capacity of 10E14. As estimated, a single DNA computer could contain more data compared to all the existing computer memories combined. Adleman's experiment was carried out at 1.2x 10E18 operations per second. This is approximately 1,200,000 times faster than any existing super computing device.

The following are the benefits of DNA computer:

**1) PREDICTABILITY**

After a year in lab, Adleman realized that strands of DNA behave much like mathematical equations. DNA's chemical bases-adenine, thymine,cystosine, and guanine—hook up in a predictable manner:adenine always links with thymine and cytosine with guanine. Because of regularity of pattern Adleman hypothesized that he could use molecules to process data the same way PCs use microprocessors.

**2) DNA DIRECTIONS**

Over a period of time, Adleman performed a series of biochemical reactions to eliminate the wrong answers—strands encoding routes that either started or ended in the wrong city, those that visited a city more than once, and so on. When all the wrong answers had been destroyed, Adleman was able to look under the microscope and find only strands that carried the right answers.
Adleman's experiment used just seven cities, a problem that isn't hard to solve on modern computers. But Adleman's biological computations showed that DNA has the potential to solve more complex problem than even the most advance electronic computer can. The fastest supercomputer wouldn't be able to solve a problem if more

than about 50 cities, Adleman says. He believes that a testtube full of DNA would solve the problem with as many as 200 cities.
First, DNA is incredibly enery-efficinet.Take ligase, a molecule whose job is to stick strands of DNA together. With just one joule of energy – the amount of energy a human expends to lift one kilogram one meter, ligase molecules can perform 20x10 the 18th operations, Adleman says. Thats a million times 20 operations. Such efficiency push computing to new levels since electronics are limited by the amount of power—and the heat it gives off—needed to run increasingly sophisticated operations.

**3) INCREDIBLY CHEAP**

DNA is also a wonderful way to store information. One gram of genetic material, which would occupy about one cubic centimeter, can hold as much information as 1 trillion CDs, according to Adleman. It's also incredibly cheap: Commercial labs sell a molecule of SNA for about one-thousand trillionth of a cent. The cost is about $30. for a DNA sequence big enough to compute on. Intel sells its latest P4 chiop for more than $500. "DNA has been storing the blueprint of life for several billion years," says Adleman. "its powers are untapped legacy for the 21st century.

**4) HALF-HOUR TEST**

The first practical applications are also emerging. Last January Olympus optical company and the university of TOKYO claim to have jointly develop a fast way identifying genes associated with diseases. Researchers developed a process that synthesizes 10,000 different DNA strands that are known to bond with genes related to specific diseases such as cancer. The strand are numbered and mixed with fluid containing genes extracted from the patient. The fluid is than tested to determine which genes are functioning in the patient's cells by reading the number of DNA strands that appear after a series of biochemical reactions. Researchers can complete a single test in about 3 hours, about one-half to one-third time taken by conventional biological methods.

**5) AMAZING TOOL TEST**

Thus, as the complexity of problems increases, the manual labour require for DNA computing would outweight the benefits of super fast computation. "Here we have the most amazing tool chest we have ever seen. We know it's great because it was used to build you and me, "enthuses Adleman. "Right now, though, we are very clumsy with it."
DNA computers have the potential to take computing to new levels, picking up where Moore's law leaves off. There are several advantages of using DNA instead of silicon:

1) As long as there are cellular organisms, there will always be a supply of DNA.
2) The large supply of DNA makes a cheap resource.
3) Unlike the toxic materials use to make traditional microprocessors, DNA biochips can be made cleanly.
4) DNA computers are many times smaller than today's computer.

# 9. DRAWBACKS

**1) Occasionally Slow**

The speed of each process in DNA Computing is still an open issue until now. In 1994, Adleman's experiment took still a long time to perform. The entire experiment took Adleman 7 days of lab work [13]. Adleman asserts that the time required for an entire computation should grow linearly with the size of the graph. This is true as long as the creation of each edge does not require a separate process.

Practical experiments proved that when people using DNA to solve more complex problems such like SAT problems, more and more laborious separation and detection steps are required, which will only increase as the scale increases. But these problems may be overcomed by using autonomous methods for DNA computation, which execute multiple steps of computation without outside intervention. Actually, autonomous DNA computations were first experimentally demonstrated by Hagiya et al. [14] using techniques similar to the primer extension steps of PCR and by Reif, Seeman et al. [15] using the self-assembly of DNA nanostructures [16]. Recently, Shapiro et al. reported the use of restriction enzymes and ligase [2] on the Nature (Figure 5). They demonstrated a realization of a programmable finite automaton comprising DNA and DNA-manipulating enzymes that solves computational problems autonomously. In their implementation, 1012 automata sharing the same software run independently and in parallel on inputs (which could, in principle, be distinct) in 120 micro liters solution at room temperature at a combined rate of 109 transitions per second with a transition fidelity greater than 99.8%. Thus, the laborious processes can be reduced largely. We can forecast that this problem can be settled very well in not long time.

**2) Hydrolysis**

The DNA molecules can fracture. Over the six months you're computing, your DNA system is gradually turning to water. DNA molecules can break – meaning a DNA molecule, which was part of your computer, is fracture by time. DNA can deteriorate. As time goes by, your DNA computer may start to dissolve. DNA can get damaged as it waits around in solutions and the manipulations of DNA are prone to error. Some interesting studies have been done on the reusability of genetic material in more experiments, a result is that it is not an easy task recovering DNA and utilizing it again.

**3) Information Untransmittable**

The model of the DNA computer is concerned as a highly parallel computer, with each DNA molecule acting as a separate process. In a standard multiprocessor connection-buses transmit information from one processor to the next. But the problem of transmitting information from one molecule to another in a DNA computer has not yet to be solved. Current DNA algorithms compute successfully without passing any information, but this limits their flexibility.

**4) Reliability Problems**

Errors in DNA Computers happen due to many factors. In 1995, Kaplan et al. [17] set out to replicate Adleman's original experiment, and failed. Or to be more accurate, they state, "At this time, we have carried out every step of Adleman's experiment, but we have not got an unambiguous final result." There are a variety of errors that can

come along with experimentation. Typical errors are annealing errors, errors in PCR, errors during affinity separation (Purification).

# 10. APPLICATIONS OF DNA COMPUTER

## 10.1) Massively Parallel Processing

The primary advantage offered by most proposed models of DNA based computation is the ability to handle millions of operations in parallel. The massively parallel processing capabilities of DNA computers may give them the potential to find tractable solutions to otherwise intractable problems, as well as potentially speeding up large, but otherwise solvable, polynomial time problems requiring relatively few operations. The use of DNA to perform massive searching and related algorithms will be referred to as "classic" DNA computation for the purposes of this discussion.
Proposed "classical" models of DNA computers derive their potential advantage over conventional computers from their ability to:

- Perform millions of operations simultaneously;
- Generate a complete set of potential solutions;
- Conduct large parallel searches; and
- Efficiently handle massive amounts of working memory.

These models also have some of the following drawbacks :

- Each stage of parallel operations requires time measured in hours or days, with extensive human or mechanical intervention between steps;
- Generating solution sets, even for some relatively simple problems, may require impractically large amounts of memory; and
- Many empirical uncertainties, including those involving: actual error rates, the generation of optimal encoding techniques, and the ability to perform necessary bio-operations conveniently in vitro or in vivo.

With these qualities in mind, the comparison between conventional computing and "classic" DNA computation comes down to one of depth versus breadth. A working DNA based computer might hold an advantage over conventional computers when applied to decomposable problems, those problems that are able to be divided into separate, non-sequential tasks, because they can hold so much data in memory and conduct so many operations at once. However, due to the length of time required to conduct the biochemical operations, non-decomposable problems, those requiring many sequential operations, are likely to remain much more efficient on a conventional computer. [2]
Within the paradigm of "classic" DNA computation there exists many different models, each with different advantages and degrees of applicability to classes of problems. An example of one model differing from Adleman's original search algorithm is the surface-based DNA algorithm used to solve the minimal set cover problem in [8]. This technique proposes to decrease the potential for error from lost

DNA strands by affixing them to a surface of glass or silicon. The efficiency of their algorithm is dependent on the method of encoding quantity by strand length so the authors have speculated that such a model might be inappropriate for application to problems where the validity of a solution is not proportional to its size. A surface-based approach to DNA computation was also considered in [13], which suggests the products of bit operations could be identified using optical readers scanning for the relative surface position of hybrid double strands consisting of a previously unknown bitstring and a value being held by that bit. The ability to read output in this manner may drastically increase the feasibility of implementing a DNA computer outside the conventional laboratory setting. It might also encourage the development of DNA/Electronic computer hybrids, with different problem types being handled by different components, and the electronic portion conveying the output to the user. Another model [10] makes use of 3-Dimensional DNA structures to reduce errors and necessary steps. This method of using the structral properties of DNA may also lead to more efficient storage mechanisms.

Classical DNA computing techniques have already been theoretically applied to a real life problem: breaking the Data Encryption Standard (DES). Although this problem has already been solved using conventional techniques in a much shorter time than proposed by the DNA methods, the DNA models are much more flexible, potent, and sender of cost effective.

DES is a method of encrypting 64-bit messages with a 56-bit key, used extensively in the United States. Electronic keys are normally a string of data used to code and/or decode sensitive messages. By finding the appropriate key to a set of encrypted messages, one can either read encoded messages or pose as the such messages. Using a special purpose electronic computer and differential cryptanalysis, it has been shown that the key to DES can be found in several days. However, to do so would require 2^43 examples of corresponding encrypted and unencrypted messages (known as plain-text/cipher-text pairs) and would slow down by a factor of 256 if the strength of the encrypting key was increased to 64-bits. In [6] it is proposed that DES could be broken using a DNA based computer and a search algorithm similar to Adleman's original technique. This procedure would be expected to take 4 months, but would only need a single plain-text/cipher-text pair or an example of cipher text with several plain text candidates to be successful. The feasibility of applying DNA computation to this problem was also addressed in [3] using a more refined algorithm (the sticker model approach) which enabled the researchers to suggest that they could solve the problem using less than a gram of DNA, an amount that could presumably be handled by a desk top sized machine. Both models would likely be more cost and energy effective than the expensive electronic processors required by conventional means, but are entirely theoretical. The first model ignores error rates incurred though laboratory techniques and the inherent properties of the DNA being used. The second model requires an error rate approaching .0001, with higher rates substantially affecting the volume of DNA required. Despite these assumptions, these models show that existing methods of DNA computation could be used to solve a real life problem in a way that is both practical and superior to methods used by conventional computers. In [3] it is also demonstrated that such benefits can be obtained despite error rates that would be unacceptable in an electronic computer and that may be unavoidable in a molecular one.

## 10.2) Storage and Associative Memory

DNA might also be used to mirror, and even improve upon, the associative capabilities of the human brain. In [4] Baum proposed a method for making a large content addressable memory using DNA. A truly content addressable memory occurs when a data entry can be directly retrieved from storage by entering an input that most closely resembles it over other entries in memory. This input may be very incomplete, with a number of wildcards, and in an associative memory might even contain bits that do not actually occur within the closest match. This contrasts with a conventional computer memory, where the specific address of a word must be known to retrieve it. Rather, the use of this technique would replicate what is thought by many to be a key factor in human intelligence.

Baum's models for a DNA associative memory are quite simple, and build from the techniques used in other areas of DNA computation. Storing a word could be done by assigning a specific DNA subsequence to each component value pair and building a fixed length word from these subsequences. To then retrieve the word closest to the input, one would introduce marked complementary subsequences into the storage medium and chose the molecule that has the most matches to this input. This technique could be further refined to more closely approximate the brain by appending words to only store attributes that an object has, rather than wasting space using '0's to represent attributes that an object does not have.

Baum has further speculated that a memory could be constructed where only portions of the data are content-addressable and associative, with other information on an object compactly stored in addresses relative to the associative portion of the entry. To save on operating costs and reduce error frequency, this portion of the memory could be kept in double-stranded form.

Considering the brain's limit of about 10^15 synapses and Feynman's low-end estimate that the brain can distinguish about 10^6 concepts, such a DNA based associative memory could have certain advantages over the brain. Without accounting for redundant molecules, Baum estimates that a large bath tub of DNA, about 50g in 1000L, could hold over 10^20 words. In attempting to come up with practical uses for this memory scheme one will have to weigh the massive size and ability to retrieve in an associative manner against the slow retrieval times necessitated by current biomolecular engineering techniques. And, although Baum's simplistic approach has accounted for some error rates, his initial paper remains quite speculative.

## 10.3) DNA2DNA Applications

Another area of DNA computation exists where conventional computers clearly have no current capacity to compete. This is the concept of DNA2DNA computations as suggested in [12] and identified as a potential killer app. DNA2DNA computations involve the use of DNA computers to perform operations on unknown pieces of DNA without having to sequence them first. This is achieved by re-coding and amplifying unknown strands into a redundant form so that they can be operated on according to techniques similar to those used in the sticker model of DNA computation. Many of the errors inherent in other models of DNA computing can hopefully be ignored in DNA2DNA computing because there will be such a high number of original strands available for operations.

The potential applications of re-coding natural DNA into a computable form are many and include:

- DNA sequencing;
- DNA fingerprinting;
- DNA mutation detection or population screening; and
- Other fundamental operations on DNA.

In the case of DNA mutation detection, the strand being operated on would already be partially known and therefore fewer steps would need to be taken to re-code the DNA into a redundant form applicable for computational form.
There are other models of DNA computation that suggest that DNA might be used to detect and evaluate other chemical and bio-chemical substances. In [16] it is suggested that nucleic acid structures, in addition to nucleic acid sequences, could play an important role in molecular computation. Various shapes of folded nucleic acids can be used to detect the presence of drugs, proteins, or other molecules. It is also suggested that selected or engineered ribozymes could be used as operators to effect re-write rules and to detect the presence of such non-nucleic acid molecules. Using these structures and operators to sense levels of substances, it would then be possible to compute an output readable using proposed biosensors that detect fluorescence or polarization. These biosensors could potentially allow communication between molecular sensory computers and conventional electronic computers.

# 11. PRESENT AND FUTURE SCENARIO

In 2000, Gardner and his colleagues James Collins and Charles Cantor, both also of Boston University, built a memory device in E. coli out of two inverters for which the output protein of one is the input protein of the other, and vice versa. In the same year, Michael Elowitz and Stanislas Leibler of Rockefeller University in New York City made an oscillator in which three inverters are connected in a loop so that the output protein of each inverter is the input protein of the next inverter. In one test of their system, a fluorescent protein became active whenever one of the proteins was in its low state. The result was a population of gently twinkling cells like flashing holiday lights, Elowitz says. "It was very beautiful," he says.
Weiss' team has just put the finishing touches on a five-gene circuit in E. coli that can detect a particular chemical in its surroundings and turn on a fluorescent protein when the chemical concentration falls within preselected bounds. Such circuits could eventually be used to detect environmental toxins, Weiss notes. Ultimately, he says, different cells could be programmed to respond to different concentrations of a toxic chemical and to fluoresce in different colors, so that the cell population would generate a color-coded topographical map of the toxin concentrations.
A. Bioware: All of the molecular computing methods mentioned above envision that the computation will be done in vitro. Although the molecules are of biological origin, they are extracted from the cell, and the reaction takes place in laboratory glassware. But why not turn the living cell itself into a computer, powered by its own metabolism? Several research collaborations have done work pointing toward this possibility. The following are the ideas of a group at MIT, who have examined the computational aspects of the problem in great detail. The MIT group consists of Thomas F. Knight, Jr., Harold Abelson and Gerald Jay Sussman, and several of their

present and former students, including Don Allen, Daniel Coore, Chris Hanson, George E. Homsy, Radhika Nagpal, Erik Rauch and Ron Weiss.The first major goal of the MIT group is to develop design rules and a parts catalogue for biological computers, like the comparable tools that facilitate design of electronic integrated circuits. An engineer planning the layout of a silicon chip does not have to define the geometry of each transistor individually; those details are specified in a library of functional units, so that the designer can think in terms of higher-level abstractions such as logic gates and registers. A similar design discipline will be needed before biocomputing can become practical.

The elements of the MIT biocomputing design library will be repressor proteins. The logic "family" might be named RRL, for repressor-repressor logic, in analogy with the long-established TTL, which stands for transistor-transistor logic. The basic not gate in RRL will be a gene encoding some repressor protein (call it Y), with transcription of the Y gene regulated in turn by a different repressor (call it X). Thus whenever X is present in the cell, it binds near the promoter site for Y and blocks the progress of RNA polymerase. When X is absent, transcription of Y proceeds normally. Because the Y protein is itself a repressor, it can serve as the input to some other logic gate, controlling the production of yet another repressor protein, say Z. In this way gates can be linked together in a chain or cascade.

Going beyond the not gate to other logical operations calls for just a little more complexity. Inserting binding sites for two repressor proteins (A and B) upstream of a gene for protein C creates a nand gate, which computes the logical function not and. With the dual repressor sites in place, the C gene is transcribed only if both A and are absent from the cell; if either one of them should rise above a threshold level, production of C stops. It is a well-known result in mathematical logic that with enough nand and not gates, you can generate any Boolean function you please. For example, the function (A or B) is equivalent to (not (A nand B)), while (A and B) is ((not A) nand (not B)). The not gate itself can be viewed as just a degenerate nand with only one input. Thus with no more resources than a bunch of nand gates, you can build any logical network.

*Figure 5 Design of a biochemical nand logic gate connected to a downstream inverter. The two-input nand gate consists of two separate inverters, each with a different input, but both with the same output protein. The nand gate output is always high unless both inputs are present. This output can then be connected to other downstream gates, such as an inverter.*

Pairs of nand gates can also be coupled together to form the computer memory element known as a flip-flop, or latch. Implementing this concept in RRL calls for two copies of the genes coding for two repressor proteins, M and N. One copy of the M gene is controlled by a different repressor, R, and likewise one copy of the N gene is regulated by repressor S. The tricky part comes in the control arrangements for the second pair of genes: Here the repressor of M is protein N, and symmetrically the repressor of N is M. In other words, each of these proteins inhibits the other's synthesis. Here's how the flip-flop works. Suppose initially that both R and S are present in the cell, shutting down both of the genes in the first pair; but protein M is being made at high levels by the M gene in the second pair. Through the cross coupling of the second pair, M suppresses the output of N, with the collateral result that M's own repressor site remains vacant, so that production of M can continue. But now imagine that the S protein momentarily falls below threshold. This event briefly lifts the repression of the N gene in the first pair. The resulting pulse of N protein represses the M gene in the second pair, lowering the concentration of protein M,

which allows a little more N to be manufactured by the second N gene, which further inhibits the second M gene, and so on. Thus a momentary change in S switches the system from steady production of M to steady production of N. Likewise a brief blip in R would switch it back again. (S and R stand for "set" and "reset.")

One conclusion to be drawn from this synopsis of a few RRL devices is that a computer based on genetic circuits will need a sizable repertory of different repressor proteins. Each logic gate inside a cell must have a distinct repressor assigned to it, or else the gates would interfere with one another. In this respect a biomolecular computer is very different from an electronic one, where all signals are carried by the same medium—an electric current. The reason for the difference is that electronic signals are steered by the pattern of conductors on the surface of the chip, so that they reach only their intended target. The biological computer is a wireless device, where signals are broadcast throughout the cell. The need to find a separate repressor for every signal complicates the designer's task, but there is also a compensating benefit. On electronic chips, communication pathways claim a major share of the real estate. In a biochemical computer, communication comes for free.

Are there enough repressor proteins available to create useful computational machinery? Note that interference between logic gates is not the only potential problem; the repressor molecules taking part in the computation must also be distinct from those involved in the normal metabolism of the cell. Otherwise, a physiological upset could lead to a wrong answer; or, conversely, a computation might well poison the cell in which it is running. A toxic instruction might actually be useful—any multitasking computer must occasionally "kill" a process—but unintended events of this kind would be a debugging nightmare. You can't just reboot a dead bacterium.

Nature faces the same problem: A multitude of metabolic pathways have to be kept under control without unwanted cross talk. As a result, cells have evolved thousands of distinct regulatory proteins. Moreover, the Biocomputing engineer will be able to mix and match among molecules and binding sites that may never occur together in the natural world. The aim of the RRL design rules is to identify a set of genes and proteins that can be encapsulated as black-box components, to be plugged in as needed without any thought about conflicts.

# 12. Conclusion

This is becoming one of the most exciting fields. I'll conclude this paper by just sharing a vision for the future in which a single drop of water holds a veritable army of living robots; in which people download software updates not for their computers, but for their bacteria; and in which specially programmed cells course through a person's arteries, monitoring blood sugar concentrations and keeping an eye out for cholesterol buildups.

These scenarios still belong to the realm of science fiction—but implanting computer programs into living creatures may not be far away. In the past few years, scientists have taken the first steps towards creating a host of cellular robots that are programmed to carry out tasks such as detecting and cleaning up environmental pollutants, tracking down cancer cells in a body, and manufacturing antibiotics or molecular-scale electronic components. These researchers have imported notions of electrical engineering—digital logic, memory, and oscillators—into the realm of biology.