Microfluidic informatics:

An alternative model of computation for the solution of intractable problems

by

Constantin Siourbas

School of Computer Science

McGill University

Montreal, Quebec, Canada

October, 2000

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements of the degree of Master of Computer Science

0-612-70505-6

Abstract

As computer chips become faster and more compact, computer systems are quickly approaching their physical limits. As a result, numerous research initiatives involving fluidic, molecular, and quantum computing are in full gear. This thesis discusses the aforementioned alternate computing methods by presenting the advantages those methods enjoy over their digital counterparts. The current state of the industry and the roadblocks encountered in physically implementing such alternate computing devices are also presented. Finally, in order to display the power of such concepts, fluidic systems that may be capable of cracking certain cryptographic algorithms as well as solving complex graphical problems such as the travelling salesman problem in a fraction of the time it would take digital computers, are presented.

Résumé

A fur et à mesure que les puces électroniques plus rapides et plus compactes sont dévelopées. les limites physiques des systèmes informatiques seront bientôt atteintes. Pour franchir ces limites. nombreuses initiatives de recherche. dont fluide. moléculaire. et quanta. sont en cours. Les avantages que possèdent ces méthodes de calcul informatiques par rapport aux méthodes de calcul traditionnelles seront présentés dans cette thèse. Le statut de l'industrie informatique actuel et les difficultés associées à la mise en application de ces nouvelles initiatives seront aussi exposés. Deux examples seront abordés pour démontrer l'efficacité de ces initiatives: les systèmes fluides capables de déchiffrer des algorithms cryptographiques: et les systèmes fluides capables de résoudre des problèmes graphiques complexes. tel que *the travelling salesman problem* (problème du voyageur de commerce).

Acknowledgements

I would like to take this opportunity to thank George M. Whitesides' chemistry research group at Harvard University, and especially Janelle Anderson for introducing me to and demonstrating the microscopic fluidic devices developed by their lab, and also for subsequently asking me whether such a technology could be used for computing purposes. Special thanks to my supervisor at McGill University, Sue Whitesides, who had suggested that I should meet with them in the first place, and also for all her input and feedback; it was greatly appreciated. More thanks to my reviewer, Helen, who as an English teacher had countless great ideas on how my thesis should be organized and structured; without her I am convinced that my thesis would have been one long run-on sentence. I would also like to thank her for helping me translate my abstract; the translation would not have been as eloquent without her guidance. But above all, I would like to thank Susana for being patient with me (most of the time) and allowing me to work on my thesis when deadlines loomed.

Contents

1. Introduction **4**

1.1 STATEMENT OF ORIGINALITY 6

1.2 OVERVIEW 7

2. Alternate Computing Methods **9**

2.1 FLUIDIC COMPUTING 11

2.2 QUANTUM COMPUTING 13

2.3 MOLECULAR COMPUTING 16

2.4 FUTURE POSSIBILITIES 18

3. Fluidics **21**

3.1 FLUIDIC LOGIC CIRCUITS 22

3.1.1 OR/NOR *23*

3.1.2 AND/NAND *24*

3.1.3 EXCLUSIVE-OR *26*

3.1.4 Half Adder *27*

3.1.5 Equivalence *27*

3.2 FLUIDIC ARITHMETIC DEVICES 28

3.2.1 Addition *28*

3.2.2 Subtraction *31*

3.2.3 Comparator *32*

3.2.4 Multiplication *33*

3.2.5 Factoring *35*

4. Cryptography **38**

4.1 Conventional Algorithms 39

4.1.1 Data Encryption Standard (DES) *40*

4.1.2 Digital Signature Algorithm (DSA) *45*

4.1.3 Rivest, Shamir, and Adleman (RSA) *47*

4.2 Quantum Cryptography 49

5. Travelling Salesman Problem **51**

5.1 Problem Definition 51

5.2 Optimal Algorithms 52

5.2.1 Brute Force Method *53*

5.2.2 Dynamic Programming *53*

5.3 Approximation Algorithms 54

5.3.1 Guaranteeing a Length Twice the Optimal *54*

5.3.2 Guaranteeing a Length 1.5 Times the Optimal *56*

5.3.3 Karp's Fixed Dissection Algorithm *56*

5.4 Initial Fluidic Models 58

5.5 Future Directions 62

6. Conclusion **64**

7. References **66**

Appendix: DES Permutation Values **69**

Figures

Figure 1: Comparison of computational resources available to computers and brains 5

Figure 2: OR/NOR wall reattachment fluidic circuit 23

Figure 3: AND passive fluidic circuit 25

Figure 4: EXCLUSIVE-OR passive fluidic circuit 26

Figure 5: Half Adder passive fluidic circuit 27

Figure 6: EQUIVALENCE active circuit 28

Figure 7: Three stage full adder passive fluidic device 29

Figure 8: Three stage full adder using simultaneous carry 30

Figure 9: Six stage subtractor fluidic device 31

Figure 10: Passive three by two bit multiplier 33

Figure 11: Wall reattachment fluidic memory component 34

Figure 12: Wall reattachment three bit multiplier 35

Figure 13: DES expansion permutation 41

Figure 14: DES S-Box substitution 42

Figure 15: TSP heuristic using a minimal spanning tree 55

Figure 16: TSP heuristic using Karp's fixed dissection algorithm 57

Figure 17: Fluidic structure ensuring that nodes are not visited twice 59

Figure 18: Fluidic Euclidean travelling salesman problem (TSP) system 61

1. Introduction

As the 21st century approaches, limitations to traditional silicon chip computers are becoming an impending reality rather than a distant prospect. Those limitations will ultimately arise from the fundamental physical bounds of miniaturization, since the width of transistors and electrical wiring cannot be made slimmer than the width of an atom (Gershenfeld and Chuang, 1998). However, miniaturization has another, more restricting threshold, one that imposes a limit on the amount of heat that can be produced through energy dissipation. If a device becomes any hotter than that theoretical upper boundary, cooling the device would consume as much energy as it would dissipate (Li and Vit'Anyi, 1996), thus rendering the device unusable since there would be nothing preventing it from overheating.

Because incredibly fast and powerful computers are being demanded by the biotechnology sector to aide in mapping the human genome and discovering the origin and weaknesses of many diseases, and also because current machines are inadequate in implementing a cognizant artificial intelligence machine or establishing a secure communication channel, there are tremendous economical and societal interests involved in the ultimate realization of alternate computing methods. Also, since interconnecting machines has become a necessity with the introduction of the Internet, selecting an alternate computing method that has similar properties to human brains should be of paramount importance in order to avoid the latency issues associated with wiring conventional computers in a network.

In order to determine which properties of human brains and digital computers are desirable in an alternate computing method. Figure 1 presents data compiled by Russel and Norvig in 1995 that compares the properties of a digital computer to a human brain. In the five years since Russel and Norvig's work, the storage capacity of digital computers has expanded drastically (current RAM capacity is roughly 10^{10} bits and current disk space is between 10^{11} and 10^{12} bits) and will continue to do so in the years to come. Even with such dramatic improvements with regard to storage, the difference in storage capacity between digital computers and human brains is negligible when compared to their difference in speed. Although computer chips require tens of nanoseconds to execute instructions (what is known as cycle time or switching speed) and human brains require milliseconds, human brains take full advantage of their parallelism (neuron updates per second) resulting in a lopsided victory for the human brain. As Russel and Norvig (1995) stated, "even though a computer is a million times faster in raw switching speed, the brain ends up being a billion times faster at what it does" (p. 566). The reason for such a performance difference is that all neurons and synapses are active simultaneously in a human brain, whereas most digital computers have only a single processor or a few processors active simultaneously (Haykin, 1999).

	Digital Computer	Human Brain
Computational units	1 CPU, 10^5 gates	10^{11} neurons
Storage units	10^9 bits RAM, 10^{10} bits disk	10^{11} neurons, 10^{14} synapses
Cycle time	10^{-8} sec	10^{-3} sec
Bandwidth	10^9 bits/sec	10^{14} bits/sec
Neuron updates/sec	10^5	10^{14}

Figure 1: Comparison of computational resources available to computers and brains

Thus. the ideal computer would be one that combines the parallelism of the human brain with the switching speed of the computer. It is in this direction that current research is heading. Certain promising computing techniques addressing the issue of amalgamating the best of the human brain and of digital computers will be discussed in greater detail in an upcoming section of this thesis. Fluidic, molecular. and quantum computing will be addressed although emphasis will be placed on possible uses for fluidic devices as described by Duffy. McDonald. Schueller. and Whitesides (1998). since fluidics is a technology available today.

1.1 Statement of Originality

Due to the immense growth and popularity of transistors and electronics. fluidics has been forgotten for the last thirty years: the latest published works are from the 1960's. Therefore. unbeknownst of such research from the past. I had designed and thus rediscovered certain fluidic circuits. notably the OR. AND. XOR. half-adder. and adder components. although the figures presented in this paper are from Foster and Parker's textbook (1970). In addition. although continuing investigation into the literature on fluidics brought forth NOR. NAND. equivalence. comparator. and subtraction devices to my attention. the literature failed to provide any further fluidic mathematical circuits. As such. the multiplication and factoring fluidic components presented in this thesis are my ideas. as is all mention of the fluidic model that could potentially solve the travelling salesman problem.

1.2 Overview

To begin. this thesis is intended to be mainly a survey of alternative computing methods. As a result. it necessarily involves references to fields of science other than computer science. and thus a variety of sources have been consulted. including books. articles in the general press. as well as technical articles. and that many descriptions are at a general level.

Secondly. the focus of this thesis is mainly on fluidics because I have interacted with the chemistry research laboratory at Harvard University headed by George Whitesides. where one of the lab's ongoing projects is the development of techniques for rapid prototyping of small (micrometer scale) objects. including fluidic channels. Hence. I have been able to discuss with Janelle Anderson. one of the lab members. what fluidic designs seem within the realm of possibility from the viewpoint of a practitioner in the field of computer science. Unfortunately. carrying out the ideas presented in this thesis is beyond the time scale of a Master's thesis. since it would involve continued collaboration with an outside field at another university.

As a result. the main goal of this thesis is to assemble a range of future computing possibilities. and will cover three major alternate computing methods being investigated today (fluidic. quantum. and molecular computing) before focussing on the advantages of fluidic devices. such as the parallelism they offer unlike their digital counterparts. In order to evaluate such advantages. an inherently inefficient (seemingly) computation performed on digital computers. factoring. will be modelled using a fluidic device. In addition. since factoring plays an important role in cryptography. the third and fourth sections of the paper will discuss in detail the major cryptographic algorithms and techniques discovered to date, and the method in which a fluidic system can potentially factor an integer of arbitrary size in relatively little time. Finally, in order to provide an example of a fluidic device being used towards solving an actual computer science problem, the fifth section will provide background and the current optimal solutions for the travelling salesman problem using digital computers, followed by a fluidic model that could potentially solve that classical problem.

2. Alternate Computing Methods

As conventional computers become increasingly more powerful. they begin to depend more on their physical properties rather than on their logical organization. in contrast to the situation at the onset of the electronic revolution. One such physical dependency is in the laws of physics. from which latency issues arise as the interconnect wires in multiprocessor computers become more disarrayed. Another vital dependency is in the laws of thermodynamics. from which limits regarding maximal energy dissipation are imposed.

The alternate computational methods discussed in this section hold promise for eliminating the problems of latency and wiring associated with parallel computers and the rapidly approaching ultimate limits to computing power imposed by the fundamental limits of thermodynamics. These computational methods also open the possibility for unlimited parallelism and. as an added benefit. have been shown to be very efficient for some standard problems like factoring. a problem for which all known classical methods produce exponential time algorithms.

The first imposed limit is that of the interconnect wires and the associated latency issues. also known as the "spaghetti problem." Parallel computers that allow processors to randomly access a large shared memory. or rapidly access a member of a large number of other processors. will necessarily have large latency. For example. if *n* processing elements each of unit volume are used. then the tightest they can be packed is in a sphere of volume *n*. Assuming that the units have no unorthodox shapes. such as being spherical themselves. no unit in the enveloping sphere can be closer to all other units than a distance of radius $R=(3n/4\pi)^{1/3}$. Because signal transmission is bounded by the speed of light, the lower time bound on any unit using n processing elements is at least order $n^{1/3}$. Thus, through the representation of the processing elements and their interconnecting wires as a graph, Vit'Anyi (1988) derived a general theorem giving the lower bounds for the average total edge length that is optimal within a constant multiplicative factor of an upper bound. The facts that longer wires will need larger drivers and have a larger diameter, and that the larger volume will again cause the average interconnect length to increase, may make embedding in Euclidean 3-space altogether impossible with finite bounded length interconnects.

The next vital limit is that of miniaturization. The ultimate limits of miniaturization of computing devices are governed by an unavoidable heat increase through energy dissipation. This limit has already been reached by current high-density electronic chips; thus the necessity of reducing the energy dissipation of computation limits future advances in computing power. Since battery technology improves by only twenty percent every ten years, the development of low-power computing will similarly govern advances in mobile communication and computing. Over the last fifty years, the energy dissipation per logic operation has been reduced from 10^{-3} joules in 1945 to 10^{-13} joules in 1988 (Li and Vit'Anyi, 1996). Extrapolations of current trends show that the energy dissipation per binary logic operation needs to be reduced below the thermal noise level (3×10^{-21} joules at room temperature) to implement high powered machines. Even at that level, a future laptop containing 10^{18} gates/cm^3 operating at 1 GHz would dissipate 3 MW/sec. Cooling down the laptop to almost absolute zero costs at least as much energy as it saves for computing. An alternative to cooling is to develop reversible logic that can perform computations with (almost) no energy being dissipated. An operation is considered to be logically reversible if its inputs can always be deduced from its outputs. In the 1960's, R. Landauer concluded that only logically irreversible operations must dissipate energy and that all computations can be performed in a logically reversible manner at the cost of eventually filling up computer memory with unwanted garbage information, since deleting information is an irreversible process. Hence, reversible computers with bounded memories require irreversible bit operations. Thus, it is believed that the number of irreversible bits a computation requires will determine the ultimate lower limit of heat dissipation required for that computation (Li and Vit'Anyi, 1996). The alternate computing methods that follow resolve the aforementioned limitations.

2.1 Fluidic Computing

The first alternate computing method to be considered by this thesis is fluidic or micro-fluidic systems. These systems are usually developed for specific purposes such as genetic analysis, clinical diagnosis, drug screening, and environmental monitoring. Although fluidic systems do not address the problems associated with miniaturization or interconnect wires (Duffy, McDonald, Schueller, Whitesides, 1998), their ability to solve problems in parallel make them an ideal starting point for the development of new algorithms based on such properties.

Fluidic systems are composed of micro-fluidic channels with a width between 1 and 100 micrometers that can be joined using a number of micro-fluidic components such as valves, pumps, mixers, filters, and interconnects. The main purpose of micro-fluidic channels is to allow liquid or gas samples to be spewed into them and travel along the systems channels. For instance. when a maze is constructed out of the micro-fluidic channels with one entry point and one exit for the liquid. the fluidic system will display the correct path through the maze on its first attempt because of the vacuum the micro-fluidic system imposes. Similarly. blood cells injected into one end of a 'Y'-shaped fluidic system have their red and white blood cells split by separating them at the 'Y' junction using a chemical compound that attracts one cell type to one fork and repels the other cell type. forcing it towards the other fork (Duffy et al.. 1998). Such a process takes a few minutes to complete. whereas it can take several hours for conventional computers to perform the same process through the use of pattern recognition and clustering techniques.

Most micro-fluidic systems are capillary electrophoresis (CE) systems that have been fabricated in glass or oxidized silicon and then enclosed by sealing the substances containing the channels using fusion bonding (Kenis. Ismagilov. Whitesides. 1999). This process could take days or weeks to perform. Although such systems promote reusability by introducing a reversible contact between multiple sections of the system that allow the sections to be dismantled. cleaned. and reused in later systems. the length of time it would take to create such systems would be unacceptable when performing calculations that are expected to compete against conventional computers (Duffy et al.. 1998).

However. recent progress in the fabrication of micro-fluidic systems makes it possible to build these systems in less than a day using an elastomeric material. polydimethylsiloxane or PDMS. This material is then oxidized in an oxygen plasma in order to create a tight and irreversible seal. Because of the nature of the material. water can be used as the liquid of choice to pump through the system. However. even though the systems are irreversibly sealed and thus cannot be taken apart and cleaned, they can be cleansed by flushing the systems between uses using salt water. On the other hand, because fabricating such systems is fast and inexpensive, treating them as single-use disposables is another viable option (Duffy et al., 1998).

The main advantages of the PDMS fabrication procedure is that it is fast, inexpensive, robust, and very flexible in the designs it can accommodate because the micro-fluidic channels can be designed directly by the user with a computer aided design program (Duffy et al., 1998). Also, with the added possibility of fabricating three-dimensional PDMS micro-fluidic systems using this method (Anderson, Chiu, Jackman, Cherniavskaya, McDonald, Wu, Whitesides S., Whitesides G., 1999), virtually any structure could be realized by such fluidic systems.

2.2 Quantum Computing

Quantum computing is a new computational technology that promises to eliminate problems of latency and wiring associated with parallel computers along with the impending ultimate limits to computing power imposed by the fundamental limits of thermodynamics. Such computing possibilities raise the prospect of unlimited parallelism. They have also been shown to be efficient for some standard problems like factoring, where all known classical methods use exponential time algorithms. Another area where quantum computing has proved to be advantageous is in search algorithms for unstructured databases, where quantum computing algorithms have been shown to be quadratically faster than classical algorithms can ever be (Bennett, Brassard, and Ekert, 1992).

The advantage of quantum computing arises from the method by which it encodes a bit, the fundamental unit of information. The state of a bit in a conventional digital computer is either off (specified by the number 0) or on (specified by the number 1). Thus, an *n*-bit binary word in a typical computer is described by a string of *n* zeros and ones. On the other hand, a quantum bit (called a qubit) is represented by an atom in one of two different states, which can also be interpreted as 0 or 1. Therefore, two qubits, like two conventional bits, can attain four different well-defined states (00, 01, 10, and 11). However, qubits can exist simultaneously as 0 and 1, a feat not possible for conventional bits, with the probability for each state given by a numerical coefficient. As a result, describing an n qubit quantum computer in a conventional computer requires 2^n coefficients (one for each string representation). In order to place that statement into context, 50 qubits would require approximately 10^{15} numbers to describe all the probabilities for all the possible states of a quantum machine, a value that exceeds the capacity of the largest hard drives in production today (Gershenfeld and Chuang, 1998).

Therefore, a quantum computer shows immense power and promise as it not only supports multiple states at once but can also act on all its possible states simultaneously. Thus, with the ability to perform many operations in parallel using a single processing unit, quantum computers could be the ideal candidate to replace conventional computers in executing exponential algorithms such as factoring and solving other classical computer science problems such as the travelling salesman problem.

Photons are the chosen vehicle when implementing a quantum computer. The polarization of photons is indeterminate until detected; however. once the polarization is detected for one photon. the states of the other photons become immediately fixed regardless of the distance between the individual photons, thus losing their coherence. As a result. quantum systems develop an interconnection that wires the photons together once someone measures a single photon. However. one cannot measure all of the photon's properties (such as position. polarization and velocity) at once: therefore. once one quantity is measured. the possibility of measuring the other quantities is immediately eliminated (Schneier. 1996). Even though de-coherence enabled Anton Zweilinger (2000) and his research associates at the University of Innsbruck in Austria to execute quantum teleportation in 1997 (quantum teleportation can eventually be used to transfer quantum information between quantum processors thus eliminating the latency issues present in digital networks since such a teleportation mechanism is faster than the speed of light). de-coherence makes it impossible to do any form of quantum calculation. Thus. before any useful quantum calculations. such as Peter W. Shor's algorithm (1994) to find the prime factors of an integer (much faster than on conventional computers). can be accomplished. a method of maintaining coherence to prevent the photons from losing their ability to be in multiple states simultaneously is of paramount importance. If physically realizable. such a computing method would also allow the breaking of the most commonly used cryptosystems.

2.3 Molecular Computing

The idea behind molecular computing is to use molecules instead of silicon to store and manipulate information. The advantages of molecular devices are that their size can be reduced by a factor of 1000 from that of the smallest microelectronic device components and that they emit about 50 picowatts at room temperature whereas silicon-based chips emit almost 100 watts (Reed & Tour. 2000). Therefore. the number of molecular devices that can be placed on a chip is roughly 100 000 times more that what can be done using silicon microtransistors.

Until recently. the best method for storing the molecules was in a test tube in which the molecules floated freely; a test tube the size of one's finger joint could contain billions and billions of molecular strands. In the late 1990's. researchers developed a thin. gold-coated. one-inch square of glass on which to store data (Reed & Tour. 2000). Because this glass keeps out impurities. it is less prone to error than its test tube counterpart. Other advantages of using a solid surface to store the molecular strands exist: a surface the size of a penny could hold up to 10 terabytes (10 000 GB) of data. and a solid surface brings molecular computers a step closer to reality when designing them for an individual's desktop (Liu. Wang. Frutos. Condon. Corn. & Smith. 2000). Regardless of the method used to store the molecules. the set of molecular strands is manufactured by combining the nucleotides with one another. Using that process. long molecular chains that can be used as wires as well as molecular devices that behave like electronic diodes were constructed in 1997 although ways of linking the devices together is still out of reach (Reed & Tour. 2000).

DNA, or deoxyribonucleic acid, has also been used in molecular computing. The reason for using DNA is that it has the ability to process all possible computation paths simultaneously, thus working in parallel. Also, whereas silicon represents information as a series of electrical pulses, DNA represents information as a pattern of molecules on a strand of DNA, where each strand signifies one possible computational path (Kiernan, 1997).

Thus, in order to solve a problem, all conceivable computational paths are generated and stored as desired. Since the strands are tailored to a specific problem, a new set would have to be created for each new problem. After the strands have been created, the DNA computer subjects them to a series of chemical reactions, thereby leaving only the correct computational paths in the medium. One method of eliminating the unwanted computational paths is through the use of enzymes whose function depends on the information found in only one or in multiple areas of a single DNA strand, thus allowing DNA computers to perform sophisticated logical and mathematical computations. The only caveat in using enzymes to perform the calculations is that the molecules are destroyed, rendering the DNA computer useless after a single use (Kiernan, 1997).

DNA computers were initially proposed to solve seemingly intractable computational problems where, for all known algorithms, the running time grows exponentially with the problem size (NP-complete problems). Over the last few years, numerous advances were accomplished in this field. Leonard Adleman, of the University of Southern California, successfully found the largest maximum clique in a 6 node and 11 edge graph using DNA computers in one day (Kiernan, 1997). Although it is a calculation that would have taken conventional computers an instant to calculate. increasing the number of nodes to 50 or more would probably result in DNA computers being more efficient. Another interesting discovery is that it is possible for a fully automated DNA computer to crack the Data Encryption Standard (DES) in less than two hours by taking advantage of the DNA computer's ability to test all encryption keys simultaneously (Adleman. Rothemund. Roweis. & Winfree. 1997). Knowing that security can be compromised so easily when using alternate computing methods. it is remarkable how much trust and faith is placed on digital computers in storing and transmitting one's personal and financial information over the Internet.

Although molecular computing is still in its infancy and there are many problems to overcome such as linking molecular devices together and implementing a molecular device that behaves like an electronic transistor. many researchers believe that this is the only method that would yield circuitry dense and complex enough to give rise to computers with true cognitive abilities (Reed & Tour. 2000).

2.4 Future Possibilities

For each of these alternate computing methods. there remain certain stumbling blocks to overcome before the full potential of such methods can be achieved. For instance. the main setback in implementing a quantum computer is the difficulty in maintaining coherence among the photons in order to prevent them from losing their ability to exist in multiple states simultaneously. To overcome such limitations. Gershenfeld and Chuang (1998) discovered that nuclear magnetic resonance (NMR) techniques could manipulate quantum information in an ordinary liquid that, when composed of certain molecules, actually addresses the problem of de-coherence to a certain degree. In other words, nuclei placed within a fixed magnetic field can be induced to change direction by a magnetic field that oscillates at radio frequencies. The radio waves need only be turned on for a few millionths of a second in order to rotate the nuclei enough for them to be reset to spin in both states simultaneously once again. Such NMR machines could also sequence their radio waves using specific frequency pulses to produce logical gates such as exclusive-OR gates, or to simply search through n items in a database in roughly $n^{1/2}$ tries, when it would take a conventional computer $n/2$ tries (assuming, of course that the items are not sorted and that there is no prior knowledge of the items).

However, current limitations on such machines include a limit of 10 qubits because, at room temperature, the strength of the signal between the nuclei decreases rapidly as their numbers increase. Another restriction in NMR computation is that rotating nuclei in a fluid do eventually lose their coherence after a few minutes, thus allowing them to perform roughly one thousand operations before they induce errors and become unusable. Although researchers believe that quantum error checking is feasible by adding extra qubits, no one has yet to implement it in practice (Gershenfeld and Chuang, 1998).

As one can see, the future lies not in the fabrication of even smaller circuits but in nature itself. Ordinary molecules have immense computational power; the best way to leverage such power will be to integrate all technologies, such as classical silicon chips and any quantum, fluidic, or molecular computing mixture, in order to build a supercomputer.

3. Fluidics

Fluidics. which was revived in the late 1950's by Horton. Bowles. and Warren of the US Army's Harry Diamond Laboratory. is in essence a merging of two technologies: the 1930's idea of fluid power and electronics. The main purpose of fluidics is to process information through the use of a fluidic medium (Foster & Parker. 1970).

The main advantage of fluidic circuits is that. unlike their electronic counterparts. they can operate under severe environmental conditions. Under ordinary temperature ranges (between -50°C and 150°C) and under normal radiation conditions. electronic circuits provide a more reliable performance; however. under the extreme temperatures of -150°C and 250°C and under exposure to high radiation levels. when electronic devices may not even work at all. fluidic devices are more reliable. Another benefit of fluidic circuits is their irreproachable reliability when subjected to immense vibrations. In electronic systems. strong vibrations tend to weaken solder joints and other connections. yet a fluidic system can withstand 3500 million cycles of high speed vibration at 260 cycles per second without any detectable loss of performance (Angrist. 1964).

However. in the late 1960's. fluidic devices were never expected to compete with electronics. where speed is of paramount importance. because processing information using fluidic devices was inefficient at the time. Another problem was size. Because pneumatic tubes were used to construct fluidic devices in the 1960's. the devices were too large to be practical outside of the aeronautic. aerospace. and industrial industries. However. current technologies now allow us to miniaturize fluidic systems: fluidic channels can be as small as one micrometer (Duffy et al.. 1998). thereby competing with the compactness of electronic or mechanical components. Also, in contrast to the long manufacturing processes for fluidic devices in the past, current technologies enable fluidic systems to be constructed in less than a day. As a result, miniaturization will not only enable fluidic systems to have a more widespread use, but it will also improve efficiency. The liquids will no longer have to travel a total distance measured in meters: shorter paths measured in millimetres will provide quicker results.

3.1 Fluidic Logic Circuits

In order for fluidic devices to be used for computational purposes, they must be capable of implementing the logic operations used to perform common mathematical operations in conventional computers. There are different methods of implementing fluidic components: a passive element produces the desired result through the interaction between the control (or input) liquids only, whereas an active element uses a continuous power input from another liquid in order to deflect the control liquids into performing the desired action. Other properties of fluidic systems that can prove useful when designing circuits is the liquid's behaviour when subjected to pressure changes from either side of the fluidic channel and the behaviour liquids exhibit when they cling to a channel wall, known as "wall reattachment." By taking advantage of the wall reattachment behaviour fluidic systems exhibit, fluidic flip-flops and fluidic memory were successfully implemented in the 1960's (Foster & Parker, 1970). Further wall reattachment implementations can be found in Angrist (1964) and Foster & Parker (1970). Once logic circuits, which are the building blocks used by any automated system, are successfully implemented using fluidic devices, informational processing and mathematical computations suddenly become possible.

3.1.1 OR/NOR

The OR logic operation can be easily implemented using a "Y" shaped fluidic circuit in which two of the ends are inputs and the other is an output. Since liquid will flow out if either input channel has liquid entering it, such a "Y" structure would perform its purpose. However, using wall reattachment techniques, it is possible to combine the OR and NOR logic operations into a single fluidic circuit, as shown in Figure 2.

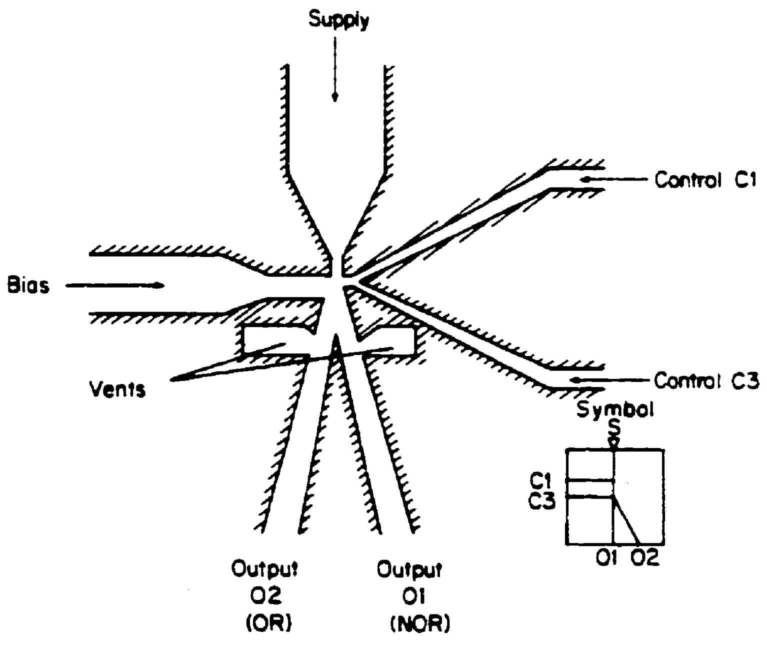

Figure 2: OR/NOR wall reattachment fluidic circuit *(source: Foster & Parker, 1970)*

The control channel on the left, shown as the bias, subjects the supply jet to incoming air pressure, thus forcing the supply jet to attach itself to the right wall and flow towards the output O_1 (the NOR exit) when the two input controls C_1 and C_3 are absent. If, on the other hand, either or both of the input control jets C1 and C3 are present, the liquid entering the circuit will deflect the supply to flow towards the output O_2 (the OR exit) (Foster & Parker, 1970). As a result, if either or both control jets are present, the OR output is activated; otherwise, the NOR output is activated. Thus, the desired behaviour has been achieved. The figures obtained from Foster & Parker represent implementations with devices operated using air where vents are put into place to allow excess or unwanted air to exit the circuit or to prevent the pressure from building up in the system. If the vents are designed to allow the air to exit as part of the control flow, then they can be replaced by reservoirs that accumulate liquid when implementing a fluidic system. In the case of Figure 2, since the vents are there to prevent the pressure from building up, they can probably be removed altogether when dealing with fluidic circuits (Angrist, 1964).

3.1.2 AND/NAND

A circuit implementing both AND and NAND logic operations using a single wall reattachment technique has not yet been successfully developed. However, it is possible to represent such a circuit as a combination of three OR/NOR circuits by letting each control jet enter a separate OR/NOR circuit and then by taking the NOR outputs of both of those circuits and inserting them into the third OR/NOR circuit. As a result, the NOR output of the third circuit is equivalent to the AND operation, whereas the OR output is equivalent to the NAND operation.

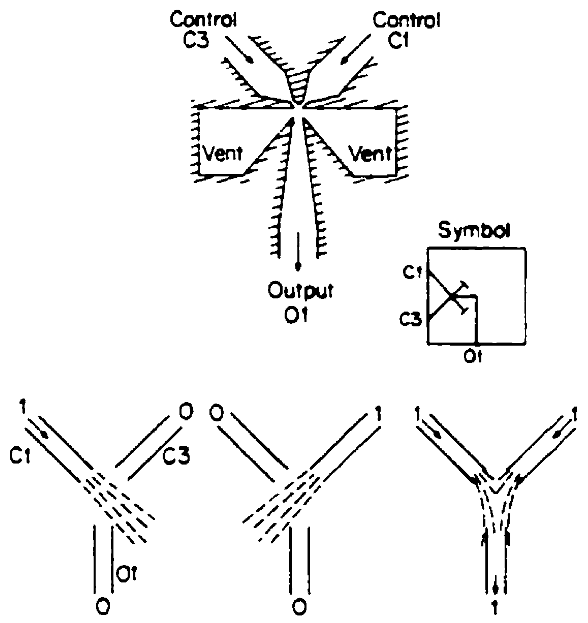

Figure 3: AND passive fluidic circuit *(source: Foster & Parker, 1970)*

If the dual functionality of having an AND and NAND operation supported by the same circuit is not required, then a passive AND circuit can be implemented very easily since the entire interaction is composed of solely the control jets, as shown in Figure 3. If either control jet is present (but not both) then the liquid would continue in a straight path to the corresponding vent or reservoir, thus preventing the liquid from interacting with the remainder of the system. However, if both control jets are present, then they collide with one another at an angle of 90 degrees, forcing both fluids to deflect exactly 45 degrees and flow through the centre channel (Conway, 1971).

According to Angrist, in order for a passive element such as this to function as intended, the width of the centre channel has to be three times the width of the control channels and the distance from the end of the control channels to the start of the output channel must be four times the width of the control channels. If the output channel is too close to the control channels, then fluid might flow into the centre channel even if only one of the control jets were present. The other extreme would present similar problems as well: a distance too great would result in fluid entering the reservoirs even if both control jets were present.

3.1.3 EXCLUSIVE-OR

The EXCLUSIVE-OR circuit is identical to the AND passive circuit with the exception of the interpretation of the outputs. As shown in Figure 4, the centre channel becomes the vent or reservoir, whereas the two side channels loop back and are ORed together to form the output channel.

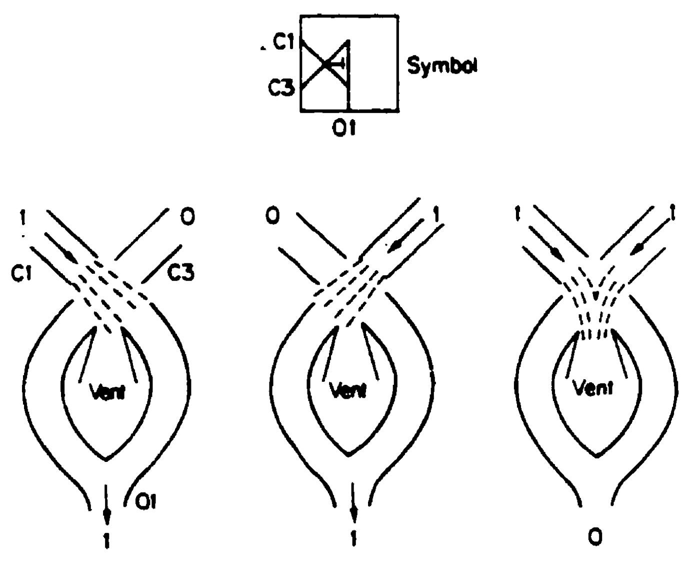

Figure 4: EXCLUSIVE-OR passive fluidic circuit *(source: Foster & Parker, 1970)*

3.1.4 Half Adder

In order to perform arithmetical operations such as addition and subtraction, a half adder logic element is required. The circuit takes the same form as an XOR structure, yet, instead of opening the centre channel to a reservoir, the centre channel is used to provide a carry signal (see Figure 5 (a)). The two remaining channels can be rejoined to represent the sum mod 2 of the control jets as shown in Figure 5 (b).

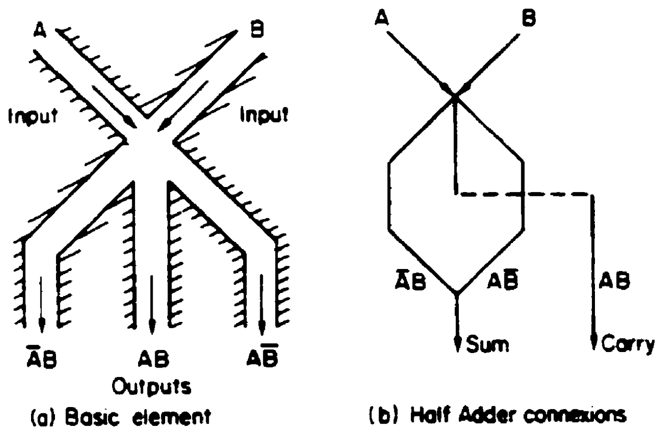

Figure 5: Half Adder passive fluidic circuit *(source: Foster & Parker, 1970)*

3.1.5 Equivalence

An equivalence circuit is a device that gives an output signal only when the two control jets are identical. As can be deduced from Figure 6 below, the supply will simply flow through the output channel untouched if both control jets are absent. If only a single control jet is present, the control jet will deflect the supply enough to direct it towards the vent diagonally opposite the control jet that aided in the deflection. When both control jets are present simultaneously, the supply receives an equal pressure from either side; as such, the control jets fail to deflect the supply liquid and the three jets simply continue their flow in parallel, and all flow through the output channel.

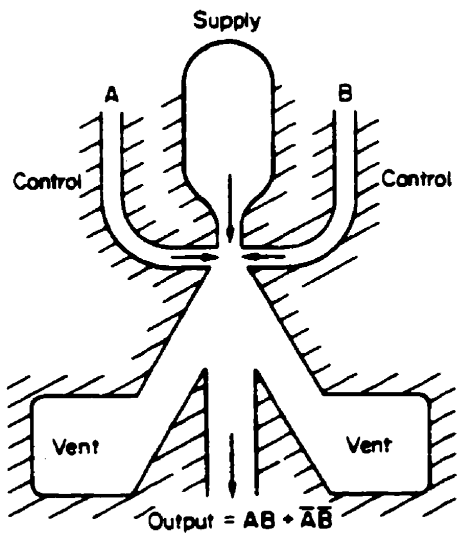

Figure 6: EQUIVALENCE active circuit *(source: Foster & Parker, 1970)*

3.2 Fluidic Arithmetic Devices

3.2.1 Addition

A logic circuit that takes two inputs and produces two outputs (notably a sum and a carry signal) is not adequate for adding larger numbers because no provision is made for including an input carry signal to the addition. Thus, in order to implement addition, a total of three inputs (two inputs plus a third carry input) are required. Such a circuit, known as a full adder, can be constructed from two half adders (shown as H.A. in Figure 7 where C is the carry and S is the sum) and an OR circuit (depicted by O in Figure 7) to regulate the carry signals produced by the two half adders. The full adders can then be linked in series to produce a device that can add an arbitrary number of bits. For instance, the fluidic adder shown in Figure 7 takes two three bit inputs ($A_2A_1A_0$ and $B_2B_1B_0$) and produces four outputs bits, the carry signal exiting the OR on the left side of the adder and the three sum signals $S_2S_1S_0$. To construct a practical adder, the full adders are linked together by summing the carry bit of the previous full adder to the second half adder of the current step. It is also possible to sum the carry bit to the first half adder of the current step; however, that would force each subsequent full adder to wait for the previous full adder to complete before adding its bits, thus making the latter implementation slower. Regardless of the connection scheme, the final sum output signals can only be read when the carry signals have crossed the entire device; thus, the sum signals do not appear simultaneously but depend upon the carry propagation time. Since the above device is a passive implementation, signal amplification components (represented as M in the Figure) may be required, although such a necessity would only be confirmed through experimentation.

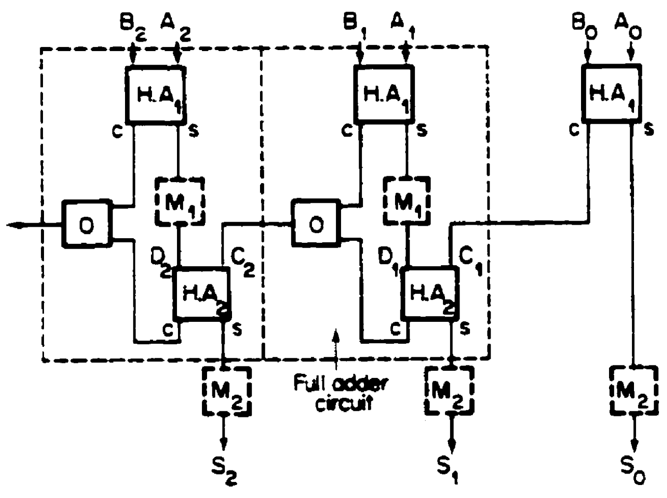

Figure 7: Three stage full adder passive fluidic device *(source: Foster & Parker, 1970)*

It is possible, however, to eliminate such dependencies and speed up the overall operating speed by handling all signals more or less simultaneously. By doing so, the complexity of the fluidic adder will be increased dramatically by including AND circuits at every stage, as shown in Figure 8. At each stage, the number of AND units will be one greater than the number for the previous stage. Also, the maximum number of inputs required by an AND unit will be one greater than for the previous stage. For example, the second stage requires one AND element with two inputs, the third requires two AND elements with a maximum of three inputs, the fourth requires three AND elements with a maximum of four inputs, and so on. In the past, such a design has not been successful when used in fluidic systems because of the timing issues involved in synchronizing the stages to produce a valid result. However, with the ability to miniaturize the fluidic components, the running time of the former implementations should no longer be a deterrent when integrating them in complex systems.

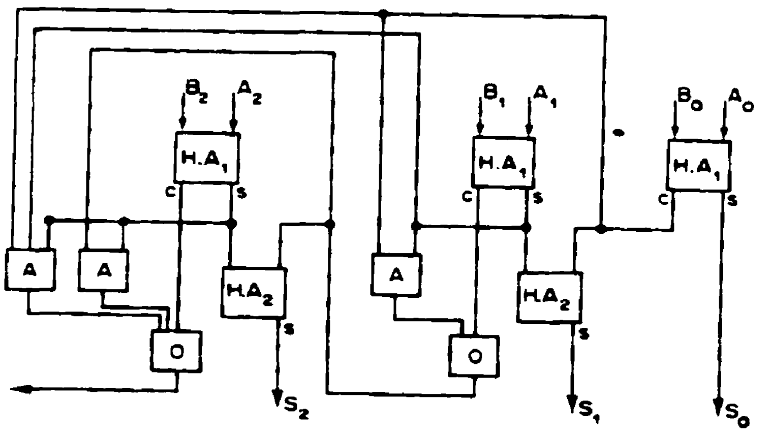

Figure 8: Three stage full adder using simultaneous carry *(source: Foster & Parker, 19˜0)*

3.2.2 Subtraction

Because of the necessity of determining whether a difference is positive or negative when subtracting two binary numbers, it is often more convenient to execute a subtraction by adding the complement representation of the numbers instead. Either the 1's complement or 2's complement may be used; however, the 1's complement is preferred due to the simplicity of converting the results back and forth. In order to implement such a fluidic device, the signals could be generated from the carry of the highest order digit of an adder device, as depicted in Figure 9.

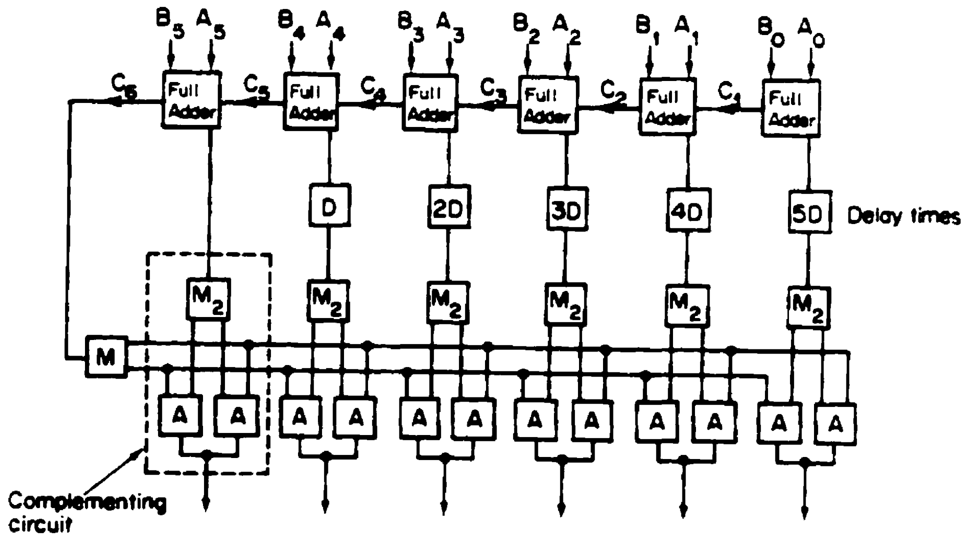

Figure 9: Six stage subtractor fluidic device *(source: Foster & Parker, 1970)*

The 1's complement is achieved through a complementing circuit that uses a wall attachment element (represented as M_2) that provides as its outputs the original signal and its negation. The outputs are then gated with passive ANDs (shown as A) and then joined together in a passive OR arrangement (represented as a solid dot). The output to the complementing circuit itself can then read the true output or its complement depending on which of the controlling signals are outputted by the component M, with M exhibiting the same behaviour as M_2 while receiving its input signal from the carry of the most significant digit (shown as C_6 in Figure 9).

When this technique is used in fluidic devices, the sum outputs of the adders must be synchronized for the complementing to function correctly. The easiest resolution to such a limitation is to delay (marked as D) the sum outputs of every stage, with the least significant stage having the longest delay. However, such a solution would not provide an exact matching since the carry propagation time can vary. A more precise method involves constructing an asynchronous device in which a signal is received from the previous stage to indicate that the operation has been completed. A clock generator is typically used so that it can control the logic elements and engage subsequent stages at known time intervals.

3.2.3 Comparator

Using a combination of equivalence circuits, it is possible to build a comparator that would generate "greater than", "less than", or "equal to" signals based on two numbers input as data. The highest significant jets are first compared using an equivalence circuit. If liquid flows towards either vent or reservoir, the comparison is over and the result is already known. If, on the other hand, the initial bits are found to be the same, then the output stream could be used as the supply to the equivalence circuit belonging to the next stage. That process would repeat until the least significant bit has been reached or until an equivalence circuit obtains different control jets as inputs.

3.2.4 Multiplication

Fluidic multiplication can be accomplished by building a device that mimics the binary multiplication process. The first step is to multiply one of the numbers with the least significant bit of the second number (which is either a 0 or a 1 when performing binary multiplication). Such a process would return either the first number itself if the least significant bit were one, or zero otherwise. In other words, a simple AND operation is required for each multiplication step. The results from each step are then added to one another. A three by two bit ($A_2A_1A_0 \times B_1B_0$) multiplication is shown in Figure 10 as a two-step process where each B bit is involved in a separate multiplication and where the addition is executed at the end of each stage to store a running total of the product. (The first adder is omitted because the first result is, in essence, being added to zero, thus making it pointless to include.)

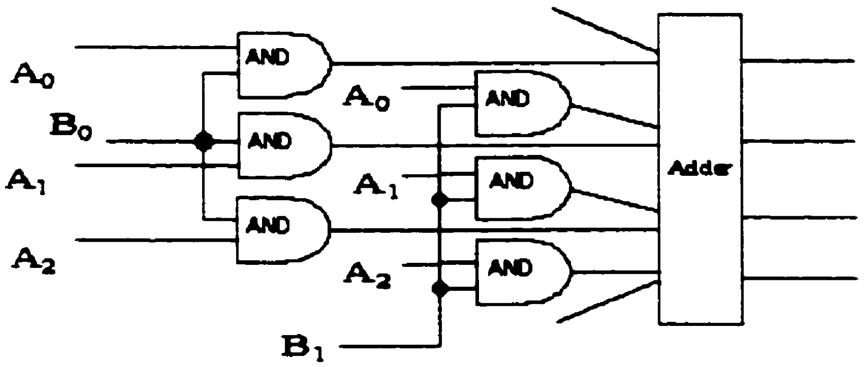

Figure 10: Passive three by two bit multiplier

Such a design can become overwhelming when trying to multiply two 64 bit integers due to the numerous redundant stages. If fluidic devices could mimic a digital computer by storing the result of each stage and shifting it by one prior to adding it to the result of subsequent steps, the device could be much more compact and easier to integrate into a fluidic computer should such a concept prove feasible.

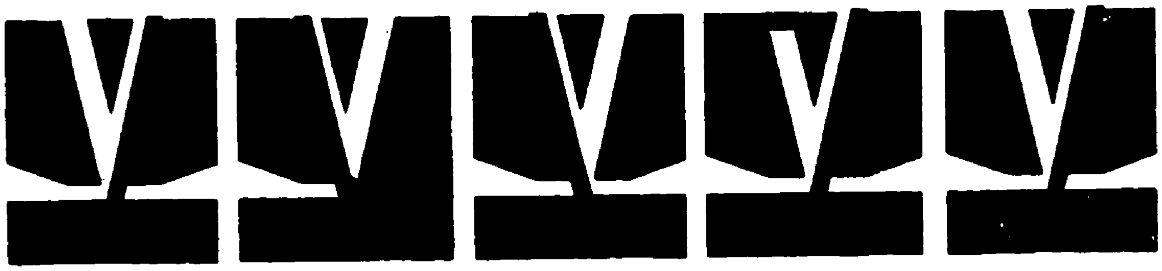

Figure 11: Wall reattachment fluidic memory component

As shown by Angrist (1964) in Figure 11, building a fluidic memory device is not very difficult when taking advantage of wall reattachment techniques. A constant flow enters through the centre channel (shown as the bottom channel in the figure), the two input channels on either side deflect the source fluid to the appropriate output channel and keep it flowing through the same output channel until another input flow deflects it elsewhere. Figure 11 (a) shows fluid from the source initially attached to the output channel on the right; (b) then subjects the source to an input flow from the right thus deflecting the source fluid to the left output channel. Due to the wall attachment property, when the input channel is discontinued (c), the source remains in the left output channel until that channel is closed (d), thus forcing the source to the right output channel once again. Removing the blocked channel on the left (e) will then result in the source keeping its current output channel. However, in order for the wall reattachment technique to function, the splitter must be at a minimum distance of six times the nozzle width from the nozzle itself. Such a memory component allows a previous multiplication result to be stored prior to being shifted for the next operation, as shown in Figure 12. A separate shifter is also required to rotate through all B bits; however, in order to synchronize the shifting of the B bits and the shifting of the memory, a delay would probably be required as in the description of a fluidic subtractor.

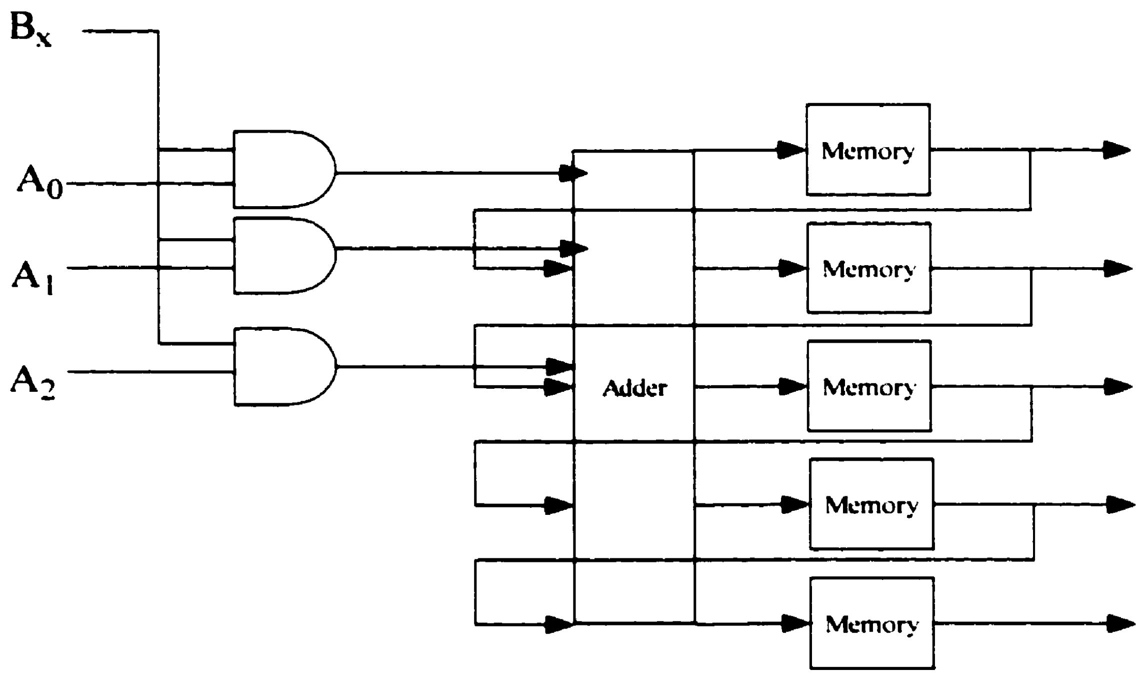

Figure 12: Wall reattachment three bit multiplier

3.2.5 Factoring

Factoring is one of the operations that digital computers apparently cannot carry out efficiently: since factoring produces two output numbers based on a single input number. a digital computer must in essence compute all possibilities if no special efficiency tricks are used, such as ignoring all even numbers as possible factors if the number is odd. However. due to the parallelism fluidics offers. it may be possible to solve such a problem in constant time using a fluidic device. Because factoring a number that has multiple prime factors is a complex issue and will have unknown and possibly disastrous consequences on the fluidic devices presented thus far. the following proposed approach assume the number to be factored is the product of two prime numbers.

Since fluidics is based on air or liquid pressure passing through a system. blocking certain outputs will create dead-ends and will prevent the fluid from entering paths that will eventually lead to these dead-ends. As a result. using the fluidic multiplication device constructed in Figure 10 and blocking all exits except for the output channels that represent ones in the binary equivalent of the product (ie. the number to be factored) should reveal the desired output when connecting all inputs to a single intake and spewing fluid into the intake. Therefore, all inputs have the ability to receive inflow yet the inputs that would have led to dead-ends (which are the zeros in the binary equivalent of the number to be factored) will not have fluid flow through them. Since the fluid will take the path of least resistance. the fluid will be compelled to flow through the input channels that will eventually lead to the output channels that remained open. As a result. the prime factors are simply represented by the binary representation of the input channels that have fluid flowing through them.

A problem that may be encountered stems from the symmetric nature of multiplication: that is. the operands can be reversed to produce the same output. Therefore. one of the input values (either A or B) must be forced to be greater than the other. Inserting a comparator prior to the multiplication device and enabling only the "greater than" output channel may provide a solution to that potential problem.

A second problem is the presence of reservoirs throughout the system. Even though certain output channels have been blocked, the liquid may attempt to go towards an undesired exit, find it blocked, and flow through the reservoir instead, thus displaying that a bit is part of the calculation when in fact the fluid should never have been allowed to flow towards the reservoir at all. To solve that potential problem, certain reservoirs must be linked directly to an output channel; thus, if an output channel is blocked, the corresponding reservoir(s) should also be.

Yet another possible problem is to prevent the number one from being returned as one of the factors. In order to do that, the input channels of the two factors must not be of equal length to the product itself. Doing so will eliminate the possibility for one of the factors to be the product itself, thus eliminating the possibility that the number one is returned as a factor also.

Much practical testing is required in order to determine whether the previously suggested resolutions would solve the anticipated pitfalls and whether other obstacles would be encountered.

4. Cryptography

The purpose of cryptography is to keep sent messages secret from potential eavesdroppers. Thus, the first goal of cryptography is to authenticate the message: in other words, the receiver of a message should be able to determine the origin of the message and be certain that the sender is not masquerading as someone else. The second goal is to provide integrity; in other words, the receiver should be able to determine whether the message has been modified in transit or whether a false message has been substituted for the real one. The final goal is to ensure that senders cannot falsely deny that they sent a message.

In 1918, Gilbert Vernam invented a code that cannot be decrypted by an eavesdropper that has since been in use for highly confidential messages between high military commanders and governments. Such codes are referred to as "one-time pads" because the cipher is used only once for a single message and then discarded (Bennett et al., 1992). The property that allows the one-time pad to be unbreakable to an eavesdropper is that the cipher is as long as the message sent and the first occurrence of a letter gets mapped to a different symbol than the second occurrence. If the cipher, or key, were to be used more than once, then there might be enough information that could be extracted from the previous messages in order to decipher the current one. As long as a new key is generated and sent prior to the message and as long as the key is at least as long as the anticipated message, then the message sent will never be able to be cracked by eavesdroppers (provided the key wasn't intercepted as well). However, even though using a one-time pad provides absolute secrecy, the strategy is not commonly used: it is too impractical to generate a new key for every message because the key would have to be exchanged in person, since electronic transmissions are inherently insecure (Gardner, 1977).

4.1 Conventional Algorithms

Due to the inconvenience of the one-time pad, cryptographic algorithms, which are mathematical functions used for encryption and decryption, that are viable for the electronic medium were proposed. There are generally two types cryptographic algorithms: one that uses symmetric keys and one that uses public keys. Symmetric key algorithms are algorithms in which the encryption key can be calculated from the decryption key and vice versa. In many cases, the encryption and decryption keys are one and the same. As a result, the security of such algorithms resides in the key: anyone obtaining the key would be able to decipher the messages. In contrast, public key algorithms are designed so that the keys for encryption and decryption are different. Also, the decryption key cannot be calculated from the encryption key. Consequently, the encryption key is made public, allowing anyone to encrypt a message: however, only a specific person with the corresponding decryption key will be able to decipher the message. In such cases, the encryption key is usually referred to as the public key and the decryption key as the private key.

Many cryptographic algorithms exist, yet the three most commonly used are the Data Encryption Standard (DES), the Digital Signature Algorithm (DSA), and the RSA Algorithm (named for its creators, Ronald Rivest, Adi Shamir, and Leonard Adleman).

The DES is the most popular encryption algorithm; this symmetric algorithm has become the international standard. The DSA, on the other hand, is a public key algorithm that can only be used for digital signatures (in order to prove that the message received truly originates from the stated sender) but not for encryption. Finally, the RSA algorithm is also a public key algorithm, but it can be used for both digital signatures and encryption.

4.1.1 Data Encryption Standard (DES)

The data encryption standard, initially created by IBM and later modified by the National Security Agency (NSA), is a symmetric block cipher algorithm. In other words, it encrypts data (or plaintext) in 64-bit blocks and it uses the same algorithm and key for both encryption and decryption. The key used for encryption and decryption is also represented as a 64-bit integer, yet its length is only 56 bits since every eighth bit is used for parity checking. There are a few numbers that are considered as weak keys, such as 0, but they are all easily avoided, an important trait seeing as all security rests within the key.

In its simplest form, DES is a single combination of a substitution followed by a permutation on the plaintext, based on the key. The algorithm begins with an initial permutation of the plaintext as shown in the appendix (bit 58 to bit 1, bit 50 to bit 2, bit 42 to bit 3, ...). The initial and final permutations do not increase the security of the algorithm; their main purpose is to load the text into a DES chip that, in the 1970's, used only eight-bit bus architecture. (Current personal computers use 32-bit bus architecture, and a 64-bit bus architecture is currently available only in high-end servers).

The next step is to reduce the 64-bit key to a 56-bit key by ignoring the parity bits. That is executed by performing a key transformation as shown in the appendix. The 56-bit key is then split into two 28-bit halves, where each half is shifted left by the number of bits shown in the appendix, based on the current round of the algorithm. After being shifted, 48 out of the 56 bits are chosen using the compression permutation shown in the appendix, thus generating a compressed key. Due to the shifting and the selection process, each bit of the key is used a different number of times; on average, each bit is used in 14 of the 16 rounds.

The next step in the encryption process is the expansion permutation, in which the right half of the plaintext, which is 32 bits long, is expanded to 48 bits (see Figure 13). As a result, the bits are rearranged and some duplicate bits are introduced. The purpose of this step is to make the halves of the input text the same size as the key so that an XOR operation can be applied. A second goal for the expansion permutation is to minimize the dependency of the output bits on the input bits at an earlier stage than otherwise possible.

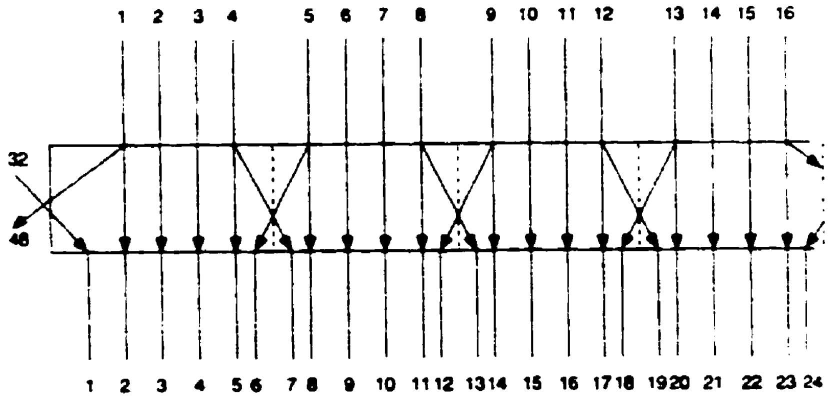

Figure 13: DES expansion permutation *(source: Schneier, 1996)*

Once the compressed key is XORed with the expanded plaintext, the resulting 48-bit value undergoes a substitution process as shown in Figure 14. Each S-box has a 6-bit input and a 4-bit output, thus reducing the block back to a size of 32 bits so that the two halves can be recombined to form a 64-bit output text (or ciphertext). Whereas all previous steps take linear time and are easy to analyze, the S-box substitution stage is not as simple: its implementation is a mystery to all but the NSA scientists. Although a message's security rests with knowing the key, the S-boxes are what give DES its security due to its unknown implementation.

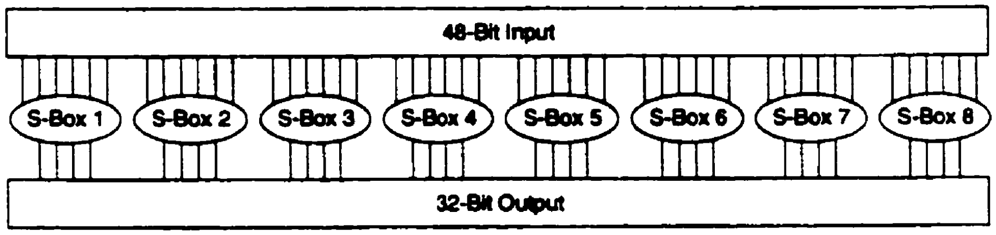

Figure 14: DES S-Box substitution *(source: Schneier, 1996)*

The output returned by the S-box is then permuted according to a P-box that places each bit in its correct output position. The value is then XORed with the left half of the initial 64-bit block. Each half then undergoes a final permutation separately (see the appendix) that has the same purpose as the initial permutation. The two halves are finally merged together to produce the 64-bit output block and then a final permutation is performed, thus completing the encryption process.

Since a different compressed key is used in the S-boxes, due to the variance in bits shifted prior to the key compression stage, the same algorithm can be applied to decrypt the ciphertext using the keys in the reverse order (the key used for the 16th round of the encryption algorithm will be used for the 1st round of the decryption algorithm. and so on) and right shifting the bits (found in the appendix) instead of left shifting them as previously described.

The security of DES has been challenged for many years. As Schneier (1996) states. there has been heated debate over why the NSA reduced IBM's initial key size from 112-bit to 56-bit. why it redesigned the S-boxes. and why it used 16 rounds. Although the answers to those questions are unknown. the NSA has insinuated that the 56-bit key length was chosen in order to satisfy the hardware limitations of the 1970's. and that. by 1990. DES would be totally insecure. However. because of the lack of another adequate cryptographic standard. the DES was re-certified in 1993 for another five years. even though specialized DES-cracking machines were a distinct possibility. The NSA also claims that the S-boxes were redesigned in case IBM had created a back door (a way to decrypt messages without knowing the key). though cryptographers believe that. by redesigning them. the NSA introduced a back door themselves. The use of 16 rounds was unexplained until 1990, when two Israeli mathematicians. Biham and Shamir. discovered "differential cryptoanalysis". This technique is much more efficient than a brute force attack. which was the only publicly known method to crack DES at the time. Differential cryptoanalysis does indeed work well when DES is used with less than 16 rounds. but once the 16th round is introduced. brute force becomes more efficient than differential analysis. which may signify that. in the mid 1970's. the NSA was fully aware of differential cryptoanalysis (Schneier, 1996).

Because of the possible security risks, numerous DES variants were introduced in order to make DES more secure. A few of these variants include RDES, which is a variant that replaces the swapping of the left and right halves at the end of each round with a key dependent swap; and key dependent S-boxes DES, which eliminates the differential analysis threat altogether since differential analysis is based on the notion of constant S-boxes.

In order to demonstrate the magnitude of the security risks, computers used to crack DES in the past should be compared to the capabilities of today's machines. In 1993, a brute-force DES-cracking machine costing $1 million could find a key in 3.5 hours by performing a million encryptions per second (Schneier, 1996). In contrast, an 80486 processor with a speed of 66MHz (which was the typical personal computer at the time) could perform 43 000 encryptions per second. Today, 1GHz (1000MHz) processors are readily available to the consumer, as are bus speeds between 133MHz to 200MHz. Another factor affecting a machine's speed is the amount of memory (or RAM) installed. It is not uncommon to have 128MB of RAM today, whereas 8MB was the standard in 1993. Also, hard drives can now transfer data at 160MB/s. As a result, today's processors, memory and hard drives in personal computers are 15 times faster than those found only five to ten years ago. Thus, theoretically, today's desktop computers could attain one million encryptions per second; therefore, they would be able to crack the DES keys in 3 to 5 hours. In other words, a brute-force method to crack the DES keys could reside on one's desktop at home at a cost of approximately $5000. Rumours state that the NSA can crack DES in 3 to 15 minutes using specialized hardware costing $50 000 (Schneier, 1996). Therefore, with improvements to digital computers attaining speeds up to three times faster than the previous year's models, the security DES offers is reaching its limits if it hasn't been eliminated already.

4.1.2 Digital Signature Algorithm (DSA)

The Digital Signature Algorithm (DSA), which was developed by the NSA, was proposed in August 1991 by the National Institute of Standards and Technology (NIST) for use in their Digital Signature Standard (DSS). Because there was no algorithm approved by the NIST to be used for digital signatures in the 1980's, companies such as IBM, Motorola, Nortel, Microsoft, Apple, and Lotus had invested large sums of money in implementing the RSA algorithm, thus making the RSA algorithm a de facto standard. Though these companies wanted RSA as the encryption standard, the NSA ignored their wishes and, on May 19, 1994, declared DSA to be the international standard (Schneier, 1996).

Unlike DES, which is used for encryption purposes, DSA is used for digital signatures. DSA uses a prime number p having a length between 512 and 1024 bits, with the length being constrained to a multiple of 64. The algorithm also requires a 160-bit prime factor q of $(p-1)$; a number $g = h^{(p-1)/q} \bmod p$, where h is any number less than $(p-1)$ such that g is greater than 1; a number x that must be less than q; and a number $y = g^x \bmod p$. The DSA also uses a one-way hash function H(m), where m is the message to digitally sign. The parameters p, q, and g are public and can be common across a group of users, while the private and public keys are x and y respectively.

To sign a message, a random number k less than q is chosen. The signature parameters, $r = (g^k \bmod p) \bmod q$, and $s = (k^{-1} (\mathrm{H(m)} + xr)) \bmod q$, are then evaluated and sent to the recipient along with the message. To verify the authenticity of the message, the recipient calculates $w = s^{-1} \bmod q$, $a = (\mathrm{H(m)} \times w) \bmod q$, $b = (rw) \bmod q$, and $v = ((g^a \times y^b) \bmod p) \bmod q$. If $v = r$, then the signature has been authenticated.

Certain pitfalls with the DSA include using a common modulus p and q, thus making DSA a viable target for cryptanalysis; and having a random generator that must produce a different k for every transaction, since it is possible to retrieve the private key x if a message has been signed using the same k twice. Another insecure property of the DSA is that it allows for a subliminal channel to be embedded into the signature, thus allowing an implementer of the DSA to use that channel to leak the private key with every signature; this technique may be used as a back door by the NSA to decode all messages transmitted using this algorithm.

A further roadblock exists to the full acceptance of DSA as the standard encryption algorithm: one that deals with patents on the algorithm itself. Though David Kravitz, a former NSA employee, holds a patent on DSA, three other individuals or groups of individuals claim that the DSA infringes on their own patents: Diffie-Hellman, Merkle-Hellman, and Schnorr (Schneier, 1996). The first two patents, having expired and having been developed using US government funding, have no bearing on the use of DSA. However, the latter patent holds worldwide, is valid until 2008, and has no ties to the US government. Therefore, any implementation of the DSA will infringe on that current worldwide patent. As a result, due to the security holes present in DSA and to the patent currently in effect, it is unlikely that influential companies such as Microsoft and IBM, after investing in RSA, will adopt this standard.

4.1.3 Rivest, Shamir, and Adleman (RSA)

RSA is the first public-key algorithm that works both for encryption and digital signatures. It is the easiest algorithm to understand and implement and has been under intense scrutiny by academic cryptographers, resulting in its widespread acceptance in the commercial marketplace (Rivest and Silverman,1999). Although extensive analysis has yet to prove or disprove its validity, there is enormous confidence and support for the algorithm because the public and private keys are factors of large prime numbers (100 to 200 digits long). The difficulty in factoring large numbers is what gives the RSA algorithm its security.

To begin with, prime numbers p and q are selected at random (for maximum security, the two primes should be of equal length), then their product n is computed. A random encryption key e is then chosen such that e and $(p-1)(q-1)$ are relatively prime. The decryption key d is then calculated using the following algorithm:

$ed \equiv 1 \bmod (p-1)(q-1)$

As a result, d and n are also relatively prime. Because the public key is composed of the numbers e and n and the private key is the number d, the primes are no longer needed; they should therefore be removed and never revealed since the security of the encryption rests on the inefficiency of factoring large numbers using any known method. For a message m to be encrypted, it must be divided into blocks smaller than n. The encrypted message c is then calculated by using $c_i = m_i^e \bmod n$ and then decrypted using $m_i = c_i^d \bmod n$.

Although it has not been proven that factoring is the only way to crack RSA, many attempts at doing so have been made. Currently, the number field sieve (NFS) is the fastest known factoring algorithm for numbers greater than 110 digits. Another oft used algorithm is the quadratic sieve (QS), which is the fastest known algorithm for numbers less than 110 digits. These sieves have proven to be very efficient. In 1993, a 120 digit number was factored using the quadratic sieve algorithm. The process used 825 mips-years and was completed in three months. A year later, computer networks across the Internet were being interconnected to solve a 116 digit number in two months using 400 mips-years. Rivest claimed in the mid 1970's that RSA would be unbreakable because the running time to crack the cipher would be approximately 40 quadrillion years due to the difficulty of finding the two prime factors of a 125 or 126 digit number obtained by multiplying two 63 digit primes (Smith, 1998) and (Gardner, 1977). However, in 1994, a 129-digit number was factored, again using the quadratic sieve, by a team of 600 mathematicians using 1600 machines over eight months. The calculation was equivalent to 4000 to 6000 mips-years, and it could have taken one-tenth that time if the number field sieve algorithm has been used instead (Schneier, 1996). As a result, it was concluded that anyone prepared to spend a few million dollars and wait a few months could factor the commonly used 512 bit number (roughly 150-160 digits) present in RSA encryption algorithms (Schneier, 1996). With the raw speed and computational power of today's computers, and with the incredible technological advances occurring every day, it is becoming difficult to develop reliable cryptographic implementations.

4.2 Quantum Cryptography

Since the most widely used cryptographic algorithms rely on assumptions, such as the belief that factoring a number efficiently is impossible, it is imperative that a truly secure method be discovered. In the early 1990's, the idea of using quantum mechanics to perform cryptographic algorithms was born. Because quantum cryptographic devices use individual photons of light and because measuring a quantum system disturbs it and yields incomplete information about its state before the measurement, eavesdropping on a private channel passively becomes impossible. Since eavesdropping on a quantum communications channel causes a disturbance, thereby alerting the legitimate users, quantum cryptography allows two individuals who have never met or shared secret information to communicate in absolute secrecy when using the theoretically unbreakable one-time pad to encode their messages, even if a malicious user is attempting to intercept those messages (Bennett et al., 1992).

Schneier (1996) explains that in order to create a secure communications channel, a one-time pad is first sent encoded as a set of photons. As the photons travel, they vibrate in some direction, whether it is up, down, left, right, or at some arbitrary angle. When numerous photons vibrate in the same direction, they become polarized. Secure communications channels could then be created by using polarization filters to allow through only those photons that are polarized in a certain direction, while blocking the rest. The channel is thus secure because a malicious listener intercepting a transmission will first have to guess the correct polarization filter to use. Then, even if the proper filter is used, because half of the photons will be blocked and because blocked photons change their polarization, an eavesdropper will always unwillingly introduce errors in the pulses as they reach their intended target. Therefore, since the eavesdropper unwittingly disrupts the intercepted message, the two parties sharing the communications channel know that their channel has been compromised and will simply retransmit a new one-time pad. Only once the pad has been transmitted successfully will an actual encoded message be sent.

Thus, quantum cryptography may be the only safe mechanism for public cryptography, since its safety rests on the validity of quantum mechanics, rather than on unproven cryptographic assumptions.

5. Travelling Salesman Problem

Fluidic devices that can determine the size of the largest clique in a graph (consult MAXIMUM CLIQUE SIZE in Garey and Johnson (1979) for a description of the problem) and that can reveal the exit to a maze have already been developed. Although only a maximum clique size of six has been handled in practice by the Harvard chemistry research laboratory, developing fluidic devices that can perform such computations in the first place shows immense promise in the technology. However, the real test in determining whether there is a future in fluidic systems and whether an interest in such an alternate computing method could be sparked, is in constructing a device that could potentially solve one of the best known and most difficult problems in computer science: the travelling salesman problem (TSP).

In the travelling salesman problem, a salesman must visit an arbitrary number of cities, must visit every city exactly once, and must then return to the city from which he began the trip. Finally, the salesman wishes to make the tour that will yield the minimum cost, such as distance travelled.

5.1 Problem Definition

Assuming the salesman must visit *n* cities, the problem can be represented as a complete graph with *n* vertices. A weight is associated with each edge of the graph signifying the distance between the two adjacent cities. For instance, if d_{ij} is the distance between city i and j, then the weight of the edge connecting vertices i and j is d. The formal language of the travelling salesman problem as described by Garey and Johnson (1979, p.18) is as follows:

> *INSTANCE: (TSP) A finite set $C=\{c_1,c_2,\ldots,c_n\}$ of cities, a distance or cost $d(c_i,c_j) \in Z^+$ for each pair of cities c_i, $c_j \in C$, and a bound $B \in Z^+$ (where Z^+ denotes the positive integers).*
> *QUESTION: Is there a tour of all cities in C having total length no more than B?*

When using graphs to represent the TSP, the problem is referred to as the Euclidean TSP where the distances $d(c_i,c_j)$ are taken as the Euclidean distances.

Conventional algorithms for the travelling salesman problem have very poor worst-case time. It seems unlikely that a fast algorithm to solve this famous computer science problem actually exists since it has been shown that this problem is NP-complete. NP-complete problems are a group of problems for which efficient algorithms have yet to be discovered and the running time of any algorithm to solve such a problem is intractable as far as we know. For a proof of the NP-completeness of the Euclidean travelling salesman problem, see Papadimitriou (1977).

5.2 Optimal Algorithms

Because efficient optimal algorithms for the travelling salesman problem have eluded scientists for roughly fifty years, hope in finding an algorithm that can guarantee the return of an optimal path has dwindled. Instead, scientists are focussing on approximation algorithms, which will be discussed in section 5.3. Among the best known optimal algorithms include the brute force method and dynamic programming.

5.2.1 Brute Force Method

The brute force method involves taking all $(n-1)!$ possible cyclic permutations of $C=\{c_1,c_2,\ldots,c_n\}$ and summing the distances to obtain the minimum. However, this algorithm would have a running time of $\Theta(n!)$, far worse than an exponential running time.

5.2.2 Dynamic Programming

Another method would be to use dynamic programming to reduce the worst-case time to $O(n^2 2^n)$, as shown by Bellman (1962). First, use c_1 as the starting vertex. For a subset $S \subseteq \{2,\ldots,n\}$, and for $c_n \notin S$, define $D(S,n)$ as the shortest distance for a path starting at c_n that visits all vertices in $\{c_i : i \in S\}$ and then ends at c_1. The dynamic programming algorithm then builds the solutions iteratively using the following algorithm:

```
For x = 2 to n do
        $D(\varnothing,x) = \| c_1 - c_x \|$
For k = 1 to n - 2 do
        For all sets S not containing 1 with |S| = k do
                For all $x \notin S, x \neq 1$ do
                        $D(S,x) = \min_{j \in S, 1 \neq x}\{ D(S - \{j\}, j) + \| c_j - c_x \| \}$
Return $\min_{x \geq 2, |S| = n-2, x \notin S}\{ D(S,x) + \| c_1 - c_x \| \}$
```

As can be deduced from the algorithm, when $|S| = k$, the time needed to calculate $D(S,x)$ is not greater than k. Thus all values for $n \notin S$ are computed in $O(n^2)$ time. Since there are fewer than 2^n different sets S, the running time of the algorithm is upper bounded by $kn^2 2^n$, for some positive constant k independent of n.

5.3 Approximation Algorithms

Since the optimal solutions presented in section 5.2 have factorial and exponential running times. solutions that can approximate the TSP and guarantee to provide a tour length not more than a constant k times the optimal solution are becoming more desirable due to the amount of processing time required using optimal algorithms when there are dozens of cities. The approximate solutions must. however. be computed in a fraction of the time required by algorithms that compute the optimal solution. as otherwise there would be little point in using approximation algorithms over their optimal counterparts since the solution returned is not necessarily the best one possible. The most common approximation algorithms use $k = 2$ or $k = 1.5$ as their approximation coefficients. where the approximation coefficient is the upper bound on the ratio of the approximation solution to the optimal one. As k approaches one (the optimal tour). the algorithms become increasingly complicated and less efficient. When k is equal to one. the running time for approximate solution is the same as the running time for the optimal solution obtained by using the dynamic programming algorithm.

5.3.1 Guaranteeing a Length Twice the Optimal

There are many approximation algorithms that guarantee a solution within twice the optimal. One of these methods uses a minimal spanning tree through the *n* vertices. The resulting tour is obtained by connecting the vertices in a minimal spanning tree using depth-first search order as shown in the Figure 15 (thus all vertices are visited twice). and then closing the tour by connecting all end nodes in the reverse order in which they were visited; to obtain a tour that visits each vertex only once, "short-cuts" are taken. Every time the tour would otherwise have revisited a previously visited vertex, the tour instead skips to the next vertex, which may in turn be skipped, and so on until an unvisited vertex is encountered.

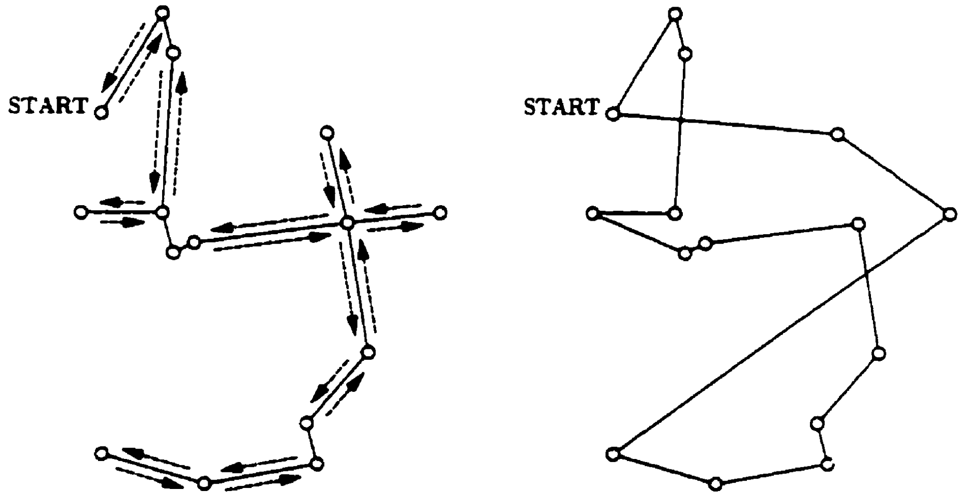

Figure 15: On the left is the minimal spanning tree and the depth-first search path, while on the right is the resulting tour obtained by the heuristic

Because the minimal spanning tree does not exceed the minimal length of any tour, the depth-first search traversal cannot exceed twice the minimal spanning tree length (by using the triangle inequality); thus the returned path is guaranteed to come within a factor of two of the optimal path. Consequently, the running time of this algorithm is determined solely by the running time of the construction of the graph's minimal spanning tree, which is $O(n^2)$, using Prim's algorithm (Papadimitriou & Steiglitz, 1982).

5.3.2 Guaranteeing a Length 1.5 Times the Optimal

Another approximation algorithm, this one with a running time of $O(n^3)$, can return a tour with a length of at most one and a half times the optimal. To begin with, a minimal spanning tree is constructed as in the previous heuristic. Since it is a well-known fact of graph theory that any graph has an even number of vertices of odd degree (a vertex of odd degree has an odd number of edges stemming from it), the vertices of odd degree can be paired. Such a pairing can be completed in $O(n^3)$ time. The edges of the pairings are then added to the minimal spanning tree, thus converting all vertices to an even degree.

Now that all vertices are of even degree, an Euler tour exists and can be discovered in $O(n^2)$ time (West, 1996). Since the optimal tour can be deconstructed into two sets of pairings by choosing in two ways every other edge on the tour, the total length of the pairings is less than half of the total length of the optimal tour itself. Also, since the length of the minimal spanning tree is not more than that of the optimal tour, the heuristic guarantees a tour length not exceeding one and a half times the optimal tour (Christofides, 1976).

5.3.3 Karp's Fixed Dissection Algorithm

A third approximation algorithm was introduced by Karp (1997) and Karp and Steele (1985) to solve the Euclidean travelling salesman problem. First, partition the unit square into a grid of n/t_n equal squares (thus, each square has a size of $(t_n/n)^{1/2} \times (t_n/n)^{1/2}$), where t_n is a given parameter (that is slowly increasing with n) to be set later and is such that n/t_n is an integer and a perfect square. Then, for each grid cell, find the optimal tour of all points in that cell using dynamic programming as described in section 3.2.2. Finally, disconnect each mini tour at an arbitrary point and link the mini tours together as shown in Figure 16.

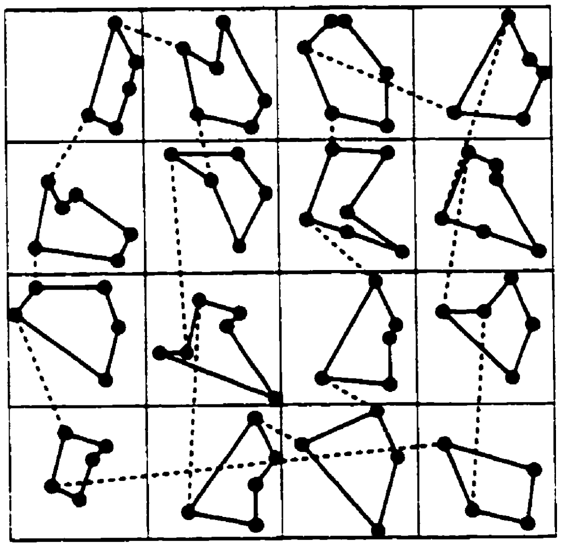

Figure 16: A $(t_n/n)^{1/2}$ ' $(t_n/n)^{1/2}$ grid of squares. Each grid contains a small optimal tour that is later detached and connected to the next cell (shown as dotted lines)

The time needed to initialize the grid and to determine to which cell each point belongs is O(n). If the ith grid cell contains N_i points, then the time taken by the dynamic programming algorithm to calculate all the optimal mini tours is upper bounded by a constant k multiplied by $\sum_i N_i^2 2^{N_i}$. Subsequently, connecting all mini tours takes time $O(n/t_n) = O(n)$. If one assumes that the data points are independently and uniformly distributed in the plane, then all N_i's are independent and identically distributed binomial random variables with parameters n and t_n/n. In other words, the expected value of N_i is

t_n. Thus, using Jensen's inequality, the expected running time of this algorithm can be calculated as follows:

$$E(\Sigma_i\, k\, N_i^2\, 2^{N_i}) = k\, \Sigma_i\, E(N_i^2\, 2^{N_i}) \geq k\, \Sigma_i\, E(N_i)^2\, 2^{E(N_i)} = k\, \Sigma_i\, t_n^2\, 2^{t_n} = knt_n(2^{t_n})$$

As a result, if $t_n = n$ then the grid has only one cell and the running time is equal to that of the dynamic programming algorithm. On the other hand, if $t_n \sim \log_2 n$ then the time needed becomes $\Theta(n^2 \log n)$, and if $t_n \sim \log_2 \log n$ then the time would be $\Theta(n \log n \log \log n)$.

However, the analysis to determine the relationship between the resulting tour and the optimal solution is beyond the scope of this paper. A lengthy proof that can be found in Karp and Steele (1985) and Karp (1977) shows that the resulting tour returned has length $\leq L_n + 2 + 10(n/t_n)^{1/2}$, where L_n is the optimal tour.

5.4 Initial Fluidic Models

Initial designs and simulations of a fluidic system for solving an instance of the TSP revealed many problematic issues and intricate difficulties that will arise when implementing such models. In order to demonstrate the difficulties associated with constructing a fluidic system that could potentially solve the TSP, an initial model will be presented and expanded upon as the problematic issues are pinpointed.

To begin with, constructing a fluidic system that was identical to the Euclidean representation of a TSP instance would allow the fluid to treat the fluidic system as a maze, and hence it would simply find the shortest route from the entrance to the exit of the system without necessarily visiting each of the nodes. Consequently, the next logical step is to alter the graph in such a way as to force the fluid to visit the exact number of nodes the graph possesses. That can be accomplished by duplicating the graph n times. stacking the copies of the graph on top of one another. and connecting together all the n copies of a given node. much like an elevator would in an n floor building. The fluid would then be injected into the top floor, and the only output channel would reside on the bottom floor. Since the fluid searches for the most efficient route from the entrance to the exit. it would always be inclined to drop a level whenever possible because the only exit is on the lowest level. Furthermore, since the only way to drop a level is to enter a node. a node would have to be visited on every level the fluid encounters (thus n nodes are visited). However. such a model does not prevent the liquid from visiting copies of the same node twice.

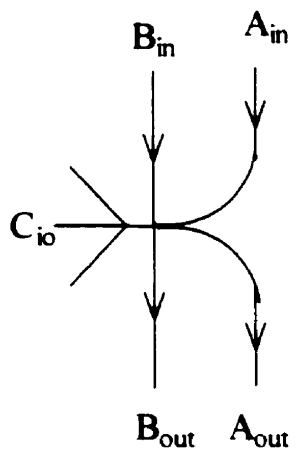

Figure 17: Fluidic structure ensuring that nodes are not visited twice

Therefore. a structure to ensure that copies of the same nodes are not visited twice would need to be incorporated into every node. Figure 17 depicts a structure to be inserted in every node i (where $1 \leq i \leq n$), at every level j (where $1 \leq j \leq n$). that could possibly do just that. If a node has not yet been visited, there would be no flow through

$A_{in}(i,j)$ and $B_{in}(i,j)$; the only flow would occur when fluid enters the node through $C_{io}(i,j)$ (*io* for input/output channel), which is the structure consisting of the set of channels within the same level, *j*, that connects the node to all of its adjacent nodes as specified by the TSP graph. The inflow through $C_{io}(i,j)$ would then close a gate at the intersection, preventing all other adjacent nodes belonging to the same level from accessing the node being entered and also preventing the liquid from exiting through $B_{out}(i,j)$. Instead, the fluid is forced to leave through $A_{out}(i,j)$, which is equivalent to fluid entering through $A_{in}(i,j+1)$ in the subsequent level. As a result, the next level, $j+1$, would have fluid entering through $A_{in}(i,j+1)$, causing fluid to escape through B_{out} and choosing a single C_{io} exit, thus heading to a different node. Because B_{out} has been activated, all subsequent levels will have an inflow through B_{in} for that particular node, causing a second gate to shut on the C_{io} and A_{out} channels, thereby preventing further access to this node and guaranteeing that the nodes visited in subsequent levels are newly encountered nodes.

A multi level diagram is presented in Figure 18 to illustrate the fluid flow. Fluid will always flow down levels (never up) and the C_{io} channels allow fluid to flow in both directions; ways to guarantee those constraints is to use one way valves to force fluid down the A and B channels while using gates to switch the behaviour of the C_{io} channels. To begin with, fluid enters through the input on the first level and reaches the first node. Because the fluid is always attempting to find the output on the last level, it will always be inclined to drop by a level whenever possible. Just in case that isn't the case, whenever fluid enters through a C_{io} channel, all other C_{io} channels are immediately blocked preventing the fluid from exploring the current level further. This process continues in selecting a single node in each level until the last level is encountered. Only the nodes that are adjacent to the starting node will have output channels as one of their C_{io} channels with a length equivalent to the Euclidean distance between it and the starting

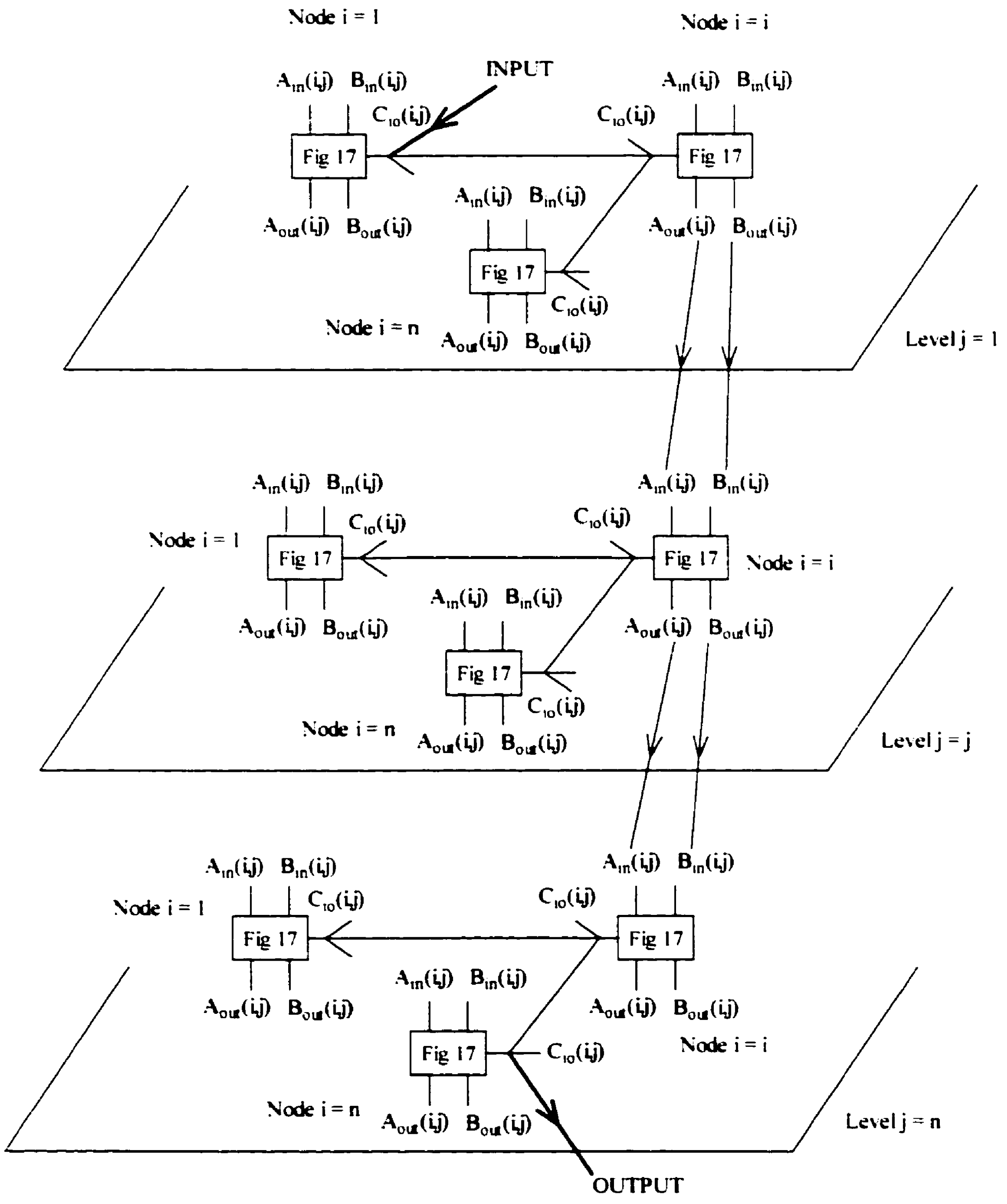

Figure 18: Fluidic Euclidean travelling salesman problem (TSP) system

node (since the TSP states that the salesman must start and end in the same city or node). Because nodes that are not adjacent to the start node do not have output channels, they are seen as dead-ends and would have already been visited in previous levels.

Whether the fluidic system described above helps in solving the TSP depends on whether the system would in fact force the fluid to follow the entire system's optimal path (the TSP solution) or whether it would force the fluid to follow the current level's optimal path (the equivalent to solving the TSP using a dynamic programming method). A quick way to test whether the suggested model would function as desired would be to construct a four-way intersection with the east, west, north, and south channels having lengths of 1, 3, 2, and 1, respectively. The only paths that should be allowed are east to west and north to south, with gates being used to prevent the other path from being used once a path has been committed. As can be deduced, the east to west path has a total length of 4 (1+3) and the north to south path has a total length of 3 (2+1). The north to south path is shorter and should be returned by the system for the TSP model mentioned above to function correctly. However, the east path (length of 1) is shorter than the north path (length of 2), thus the fluid from the east may reach the intersection first and prevent the fluid from the north path from completing its shorter route. If the desired result of north to south is obtained in such a setup, then the fluidic TSP model described should return a valid result.

5.5 Future Directions

Regardless of the model developed, in order to implement a fluidic device that would solve the travelling salesman problem, a few required components first need to be constructed. The first such component is one that guarantees that liquid can only enter and exit a node once. while another required component is one that guarantees that the entire system would be completed only when every node is visited exactly once.

Even if the model proposed here could. in principle. solve the travelling salesman problem. many improvements need to be adopted before it can become usable. Because the suggested model requires that the graph be duplicated as many times as there are nodes in the graph. a fluidic system could easily become cumbersome and difficult to manufacture when the quantity of nodes reaches a dozen or more. The ideal model would require only a single representation of the graph with all necessary structures hidden within the nodes: however. a single level structure would probably also require a component similar to an AND circuit to be inserted in the exit node in order to determine whether all nodes have been visited prior to allowing the liquid to escape through the output channel.

6. Conclusion

Because of the problems facing traditional silicon chip computers in the near future, such as latency issues and thermodynamic limits, the possibility of fabricating even smaller computer circuits is arriving at a ceiling. No longer will miniaturization be an option to increase speed because of impending heat dissipation limits. Also, with the popularity of the Internet and the sheer volume of information it disperses, security is of paramount importance; something that digital computers and current cryptographic techniques cannot provide nor ensure reliably. As a result, fluidic, quantum, and molecular computing techniques were introduced to allay all those fears and impending bottlenecks.

Possible uses and implementations of fluidic devices, such as solving mathematical or NP-complete problems, were also proposed since practical molecular and quantum systems are still years away from becoming reality. Although the fluidic model for the travelling salesman problem leaves many questions unanswered, it shows promise for future fluidic implementations that can solve complex graphical problems. The only regrettable property of such devices is that a new fluidic device would have to be built for every different graph that would require testing, a procedure that can prove to be tedious. On the other hand, the proposed fluidic structures to perform mathematical operations such as factoring appear to be very efficient. Since the mathematical structures described could be reused to perform the same mathematical operation on different numerical values, it seems viable that a completely generic calculator using fluidic devices could be constructed, although no such device was described in this paper.

Until the roadblocks for molecular and quantum computing are resolved. fluidic devices seem to be an adequate means of testing algorithms and structures that can take advantage of the immense parallelism such alternate computing techniques exhibit. However. the future of computing will eventually lie in integrating all technologies: the mechanical capabilities of silicon chips. the speed of quantum mechanics coupled with fluidic devices. and the sheer capacity and miniaturization of molecular computing will all play a vital role in building the supercomputer of the future.

7. References

Adleman. L.. Rothemund. P.. Roweis. S.. & Winfree. E. (1997). On applying molecular computation to the data encryption standard. In Proceedings of 2nd DIMACS Workshop on DNA Based Computers. The American Mathematical Society. Retrieved (2000/03/20) from the World Wide Web: hope.caltech.edu/pub/pwkr/DIMACS/des.ps.

Anderson. J.R.. Chiu. D.T.. Jackman. R.J.. Cherniavskaya. O.. McDonald. J.C.. Wu. H.. Whitesides. S.H.. Whitesides. G.M. (1999). Fabrication of topological complex three-dimensional microfluidic systems in PDMS by rapid prototyping. Harvard University. Internal report.

Angrist. Stanley W. (1964. December). Fluid control devices. Scientific American. 211 (6). 81-88.

Bellman. R. (1962). Dynamic programming treatment of the travelling salesman problem. Journal of the ACM. 9. 61-63.

Bennett. C.H.. Brassard. G.. & Ekert. A. (1992. October). Quantum cryptography. Scientific American. 267 (4). 50-57.

Christofides. N. (1976). Worst case analysis of a new heuristic for the travelling salesman problem. Technical report 388. Graduate School of Industrial Administration. Carnegie Mellon University. Pittsburgh. PA.

Conway. Arthur (1971). A guide to fluidics. New York: Elsevier.

Duffy. David C.. McDonald. Cooper. Schueller. Oliver J.A.. & Whitesides. George M. (1998. December). Rapid prototyping of microfluidic systems in poly(dimethylsiloxane). Analytical Chemistry. 70 (23). 4974-4984.

Foster. K.. & Parker. G. A. (1970). Fluidics: components and circuits. London: John-Wiley & Sons.

Gardner. Martin (1977. August). Mathematical games: a new kind of cipher that would take millions of years to break. Scientific American. 237 (2). 120-124.

Garey. Michael R.. & Johnson. David S. (1979). Computers and intractability: a guide to the theory of NP-completeness. New York: W.H. Freeman.

Gershenfeld. Neil. & Chuang. Isaac L. (1998. June). Quantum computing with molecules. Scientific American. 278 (6). 66-71.

Haykin. Simon (1999). Neural networks: a comprehensive foundation. New Jersey: Prentice Hall.

Karp. R.M. (1977). Probabilistic analysis of partitioning algorithms for the travelling salesman problem in the plane. Mathematics of Operations Research. 2. 209-224.

Karp. R.M.. & Steele. J.M. (1985). Probabilistic analysis of heuristics. New York: John Wiley & Sons.

Kenis. Paul J.A.. Ismagilov. Rustem F.. & Whitesides. George M. (1999. July). Microfabrication inside capillaries using multiphase laminar flow patterning. Science. 285 (5424). 83-85.

Kiernan. Vincent (1997. November 28). DNA-based computers could race past supercomputers. researchers predict. The Chronicle of Higher Education. Retrieved (2000/04/07) from the World Wide Web: www.chronicle.com/data/articles.dir/art-44.dir/issue-14.dir/14a02301.htm.

Li. M.. & Vit'Anyi. P.M.B. (1996). Reversibility and adiabatic computation: trading time and space for energy. Proceedings Royal Society of London. 452 (A). 769-789.

Liu. Q.. Wang. L.. Frutos. A.. Condon. A.. Corn. R.. & Smith. L. (2000. January 13). DNA computing on surfaces. Nature. 403. 175-179.

Papadimitriou. C.H. (1977). The Euclidean travelling salesman problem is NP-complete. Theoretical Computer Science. 4. 237-244.

Papadimitriou. C.H.. & Steiglitz. K. (1982). Combinatorial optimization: algorithms and complexity. New Jersey: Prentice Hall.

Reed. Mark A.. & Tour. James M. (2000. June). Computing with Molecules. Scientific American. 282 (6). 86-93.

Rivest. Ronald L.. & Silverman. Robert D. (1999. November). Are strong primes needed for RSA. Massachusetts Institute of Technology. Cambridge. MA. Retrieved (2000/03/20) from the World Wide Web: www.toc.lcs.mit.edu/~rivest/publications.html.

Russell. Stuart. & Norvig, Peter (1995). Artificial intelligence: a modern approach. New Jersey: Prentice Hall.

Schneier. Bruce (1996). Applied Cryptography. (2nd ed.). New York: John Wiley & Sons.

Shor. Peter W. (1994). Algorithms for quantum computation: discrete log and factoring. Proceedings 35th IEEE Symposium on Foundations of Computer Science. 124-134.

Smith. Richard E. (1998). Are Web Transactions Safe? Retrieved (2000/05/26) from the World Wide Web: www.pbs.org/wgbh/nova/decoding/web.html.

Vit'Anyi. P.M.B. (1988). Locality. communication and interconnect length in multicomputers. SIAM Journal of Computing. (17). 659-672.

West. Douglas B. (1996). Introduction to Graph Theory. New Jersey: Prentice Hall.

Zeilinger. Anton. (2000. April). Quantum Teleportation. Scientific American. 282 (4). 50-59.

Appendix: DES Permutation Values

The following tables should be read from left to right, top to bottom. Each number corresponds to the bit value's new position. For example, in the case of the initial permutation, the 58th bit of the plaintext moves to bit position 1, the 50th bit moves to position 2, the 42nd bit moves to bit position 3, and so on.

Initial Permutation of the Plaintext

58, 50, 42, 34, 26, 18, 10, 2, 60, 52, 44, 36, 28, 20, 12, 4,
62, 54, 46, 38, 30, 22, 14, 6, 64, 56, 48, 40, 32, 24, 16, 8,
57, 49, 41, 33, 25, 17, 9, 1, 59, 51, 43, 35, 27, 19, 11, 3,
61, 53, 45, 37, 29, 21, 13, 5, 63, 55, 47, 39, 31, 23, 15, 7

Key Transformation

57, 49, 41, 33, 25, 17, 9, 1, 58, 50, 42, 34, 26, 18,
10, 2, 59, 51, 43, 35, 27, 19, 11, 3, 60, 52, 44, 36,
63, 55, 47, 39, 31, 23, 15, 7, 62, 54, 46, 38, 30, 22,
14, 6, 61, 53, 45, 37, 29, 21, 13, 5, 28, 20, 12, 4

Number of Key Bits Shifted per Round

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Left Shift (Encrypt)	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1
Right Shift (Decrypt)	0	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Key Compression Permutation

14, 17, 11, 24, 1, 5, 3, 28, 15, 6, 21, 10,
23, 19, 12, 4, 26, 8, 16, 7, 27, 20, 13, 2,
41, 52, 31, 37, 47, 55, 30, 40, 51, 45, 33, 48,
44, 49, 39, 56, 34, 53, 46, 42, 50, 36, 29, 32

Plaintext Expansion Permutation

32, 1, 2, 3, 4, 5, 4, 5, 6, 7, 8, 9,
8, 9, 10, 11, 12, 13, 12, 13, 14, 15, 16, 17,
16, 17, 18, 19, 20, 21, 20, 21, 22, 23, 24, 25,
24, 25, 26, 27, 28, 29, 28, 29, 30, 31, 32, 1

S-Box Substitutions

S-Box 1

14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0, 7,
0, 15, 7, 4, 14, 2, 13, 1, 10, 6, 12, 11, 9, 5, 3, 8,
4, 1, 14, 8, 13, 6, 2, 11, 15, 12, 9, 7, 3, 10, 5, 0,

15, 12, 8, 2, 4, 9, 1, 7, 5, 11, 3, 14, 10, 0, 6, 13
S-Box 2
15, 1, 8, 14, 6, 11, 3, 4, 9, 7, 2, 13, 12, 0, 5, 10,
3, 13, 4, 7, 15, 2, 8, 14, 12, 0, 1, 10, 6, 9, 11, 5,
0, 14, 7, 11, 10, 4, 13, 1, 5, 8, 12, 6, 9, 3, 2, 15,
13, 8, 10, 1, 3, 15, 4, 2, 11, 6, 7, 12, 0, 5, 14, 9
S-Box 3
10, 0, 9, 14, 6, 3, 15, 5, 1, 13, 12, 7, 11, 4, 2, 8,
13, 7, 0, 9, 3, 4, 6, 10, 2, 8, 5, 14, 12, 11, 15, 1,
13, 6, 4, 9, 8, 15, 3, 0, 11, 1, 2, 12, 5, 10, 14, 7,
1, 10, 13, 0, 6, 9, 8, 7, 4, 15, 14, 3, 11, 5, 2, 12
S-Box 4
7, 13, 14, 3, 0, 6, 9, 10, 1, 2, 8, 5, 11, 12, 4, 15,
13, 8, 11, 5, 6, 15, 0, 3, 4, 7, 2, 12, 1, 10, 14, 9,
10, 6, 9, 0, 12, 11, 7, 13, 15, 1, 3, 14, 5, 2, 8, 4,
3, 15, 0, 6, 10, 1, 13, 8, 9, 4, 5, 11, 12, 7, 2, 14
S-Box 5
2, 12, 4, 1, 7, 10, 11, 6, 8, 5, 3, 15, 13, 0, 14, 9,
14, 11, 2, 12, 4, 7, 13, 1, 5, 0, 15, 10, 3, 9, 8, 6,
4, 2, 1, 11, 10, 13, 7, 8, 15, 9, 12, 5, 6, 3, 0, 14,
11, 8, 12, 7, 1, 14, 2, 13, 6, 15, 0, 9, 10, 4, 5, 3
S-Box 6
12, 1, 10, 15, 9, 2, 6, 8, 0, 13, 3, 4, 14, 7, 5, 11,
10, 15, 4, 2, 7, 12, 9, 5, 6, 1, 13, 14, 0, 11, 3, 8,
9, 14, 15, 5, 2, 8, 12, 3, 7, 0, 4, 10, 1, 13, 11, 6,
4, 3, 2, 12, 9, 5, 15, 10, 11, 14, 1, 7, 6, 0, 8, 13
S-Box 7
4, 11, 2, 14, 15, 0, 8, 13, 3, 12, 9, 7, 5, 10, 6, 1,
13, 0, 11, 7, 4, 9, 1, 10, 14, 3, 5, 12, 2, 15, 8, 6,
1, 4, 11, 13, 12, 3, 7, 14, 10, 15, 6, 8, 0, 5, 9, 2,
6, 11, 13, 8, 1, 4, 10, 7, 9, 5, 0, 15, 14, 2, 3, 12
S-Box 8
13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5, 0, 12, 7,
1, 15, 13, 8, 10, 3, 7, 4, 12, 5, 6, 11, 0, 14, 9, 2,
7, 11, 4, 1, 9, 12, 14, 2, 0, 6, 10, 13, 15, 3, 5, 8,
2, 1, 14, 7, 4, 10, 8, 13, 15, 12, 9, 0, 3, 5, 6, 11

P-Box Permutation
16, 7, 20, 21, 29, 12, 28, 17, 1, 15, 23, 26, 5, 18, 31, 10,
2, 8, 24, 14, 32, 27, 3, 9, 19, 13, 30, 6, 22, 11, 4, 25

Final Permutation of the Plaintext
40, 8, 48, 16, 56, 24, 64, 32, 39, 7, 47, 15, 55, 23, 63, 31,
38, 6, 46, 14, 54, 22, 62, 30, 37, 5, 45, 13, 53, 21, 61, 29,
36, 4, 44, 12, 52, 20, 60, 28, 35, 3, 43, 11, 51, 19, 59, 27,
34, 2, 42, 10, 50, 18, 58, 26, 33, 1, 41, 9, 49, 17, 57, 25