## CHAPTER 2

# LITERATURE SURVEY

### 2.1 A BRIEF HISTORY OF QUANTUM COMPUTING:

The idea of computational device based on quantum mechanics was first explored in the 1970's and early 1980's by physicists and computer scientists such as Charles H. Bennet of the IBM Thomas J. Watson Research Centre, Paul A. Beniof of Arogonne National Laboratory in Illinois, David Deustch of the University of Oxford and Richard P. Feynman of Caltech. The idea emerged when scientists were pondering on the fundamental limits of computation. In 1982 Feynman was among the fewer to attempt to provide conceptually a new kind of computers which could be devised based on the principles of quantum physics. He constructed an abstract model to show how a quantum system could be used to do computations and also explain how such a machine would be able to act as a simulator for physical problems pertaining to quantum physics. In other words, a physicist would have the ability to carry out experiments in quantum physics inside a quantum mechanical computer. Feynman further analysed that quantum computers can solve quantum mechanical many body problems that are impractical to solve on a classical computer. This is due to the fact that solutions on a classical computer would require exponentially growing time where as the whole calculations on quantum computer can be done in polynomial time.

Later, in 1985, Deutsch realized that Feynman assertion could eventually lead to a general-purpose quantum computer. He showed that any physical process, in principle could be modelled perfectly by a quantum computer. Thus, a quantum computer would have capabilities far beyond those of any traditional classical computer. Consequently efforts were made to find interesting applications for such a machine. This did not lead to much success except continuing few mathematical problems. Peter Shor in 1994 set out a method for using quantum computers to crack an important problem in number theory which was namely factorisation. He showed how an ensemble of mathematical operations, designed specifically for a quantum computer could be organized to make such a machine to factor huge numbers extremely rapidly, much faster than is possible on conventional computers. With this breakthrough, quantum computing transformed from a mere academic curiosity directly to an interest world over.

Perhaps the most astonishing fact about quantum computing is that it took exceedingly large time to take off. Physicists have known since 1920's that the world of subatomic particles is a realm apart, but it took computer scientists another half century to begin wondering whether quantum effects might be harnessed for computation. The answer was far from obvious.

## 2.2 LIMITATIONS OF CLASSICAL COMPUTER AND BIRTH OF ART OF QUANTUM COMPUTING

### 2.2.1: Public Key Cryptography and Classical factoring of big integers:

In 1970 a clever mathematical discovery in the shape of "public key" systems provided a solution to key distribution problem. In these systems users do not need to agree on a secret key before they send the message. The principle of a safe with two keys, one public key to lock it, and another private one to open it, is employed. Everyone has a key to lock the safe but one person has a key that will open it again, so anyone can put a message in the safe but only one person can take it out. In practice the two keys are two large integer numbers. One can easily derive a public key from a private key but not vice versa. The system exploits the fact that certain mathematical operations are easier to perform in one direction that the other e.g. multiplication of numbers can be performed much faster than factorising a large number. What really counts for a "fast" algorithm is not the actual time taken to multiply a particular pairs of numbers but the fact that the time does not increase too sharply when we apply the same method to ever-large numbers. We know that multiplication requires little extra time when we switch from two three digit numbers to two thirty digit numbers using the simpler trial division method about 10^13 times more time or memory consuming than factoring a three digit number. In case of factorisation the use of computational resources is enormous when we keep on increasing the number of digits. As a matter of fact public key cryptosystems could avoid key distribution problem. However their security depends upon unproven mathematical assumptions such as the difficulty of factoring large integers. Nevertheless one such protocol is RSA, which maps electronic banking possible by assuming banks and their customers that a bogus transfer of funds or a successful forgery would take the world's fastest computer millions of years to carry out. Another is the under spread Data Encryption Standard (DES) which remains secure far most ordinary business transactions.

The procedure of factorising a large integer can be quantified as follows. Consider a number N with L decimal digits (N ~ 10 to power L). The number is factored using trial division method. On conventional computers one of well known factoring algorithm runs for number of operations of the order of

$$s \sim O\left(\exp\left( (64/9)^{1/3} (\ln N)^{1/3} (\ln \ln N)^{2/3} \right)\right)$$

or explicitly,

$$s \sim A \exp\left( 1.9\, L^{1/3} ( \ln L)^{2/3} \right)$$

This algorithm therefore, scales exponentially with the input size log N (log N determines the length of the input. The base of the logarithm is determined by our numbering system. Thus base 2 gives the length in binary, a base 2 gives the length in binary, a base of 10 in decimal and so on) e.g. in 1994 a 129 digit number (known as RSA 129) was successfully factored using this algorithm on approximately 1600 workstations scattered around the world, the entire factorisation took eight months. Using this to estimate the per factor of the above exponential scaling, it is found that it would take roughly 800,000 years to factor a 250 digit number with the same computer power, similarly a 1000 digit number would require 10 to the power 25 years (much longer than the age of universe). The difficulty of factoring large numbers is crucial for public key cryptography such as used