Chapter 1

Introduction

Since the introduction of the first electronic computer ENIAC in 1946, the computing speed of a single processor has increased by more than an order of 8 in magnitude in five decades. However, the demand for faster computers has not decreased. Instead, the increased available computing capacity often stimulates the development of larger and more accurate computing models for the simulation and optimization of increasingly more complex problems. It is widely argued that the physical laws set fundamental limitations on the computational speed of a single processor computer:

- The finite speed of light constraints the propagation speed of signals,
- The Heisenberg's uncertainty principle constraints the minimal width on an integrated circuit and therefore the minimum distance a signal must propagate,

These two lead to the conclusion that the theoretical upper limit of a single processor is a few giga floating point operations per second (Gflops) (e.g. [1]). It is argued that the clock speed of the fastest processors is approximately O(10) GHz. The discussion on how fast a single processor can be is not yet concluded, some argue we might eventually arrive at a clock speed much higher than O(10) GHz through new technology. However, the demand for the ever greater computing power requires an alternative rather than "wait for the processor to get fast enough".

Parallel processing, by putting the computing power of a number of processors together, provides a way to by-pass the physical limits of the sequential computer. Another argument supporting the trend away from sequential towards (massively) parallel computation is that parallel processing is a more economic way to build computers with high computing speed. As opposed to many (vector) supercomputers which rely on the most advanced and therefore expensive technologies, parallel computers can be built by using off-the-shelf technology so that high-performance computers with a better price/performance ratio can be obtained. This trend is not only visible in today's large parallel computer market, it also happens in the area of servers and workstations (e.g. SGI's Power Challenge and SUN's Ultra Enterprise Servers), and PCs with multiple processors are emerging. Of course, there is yet another huge potential of parallel processing power. Workstations and PC's interconnected by high-speed networks form a large parallel system. An example is the Beowulf system built with a cluster of PCs.

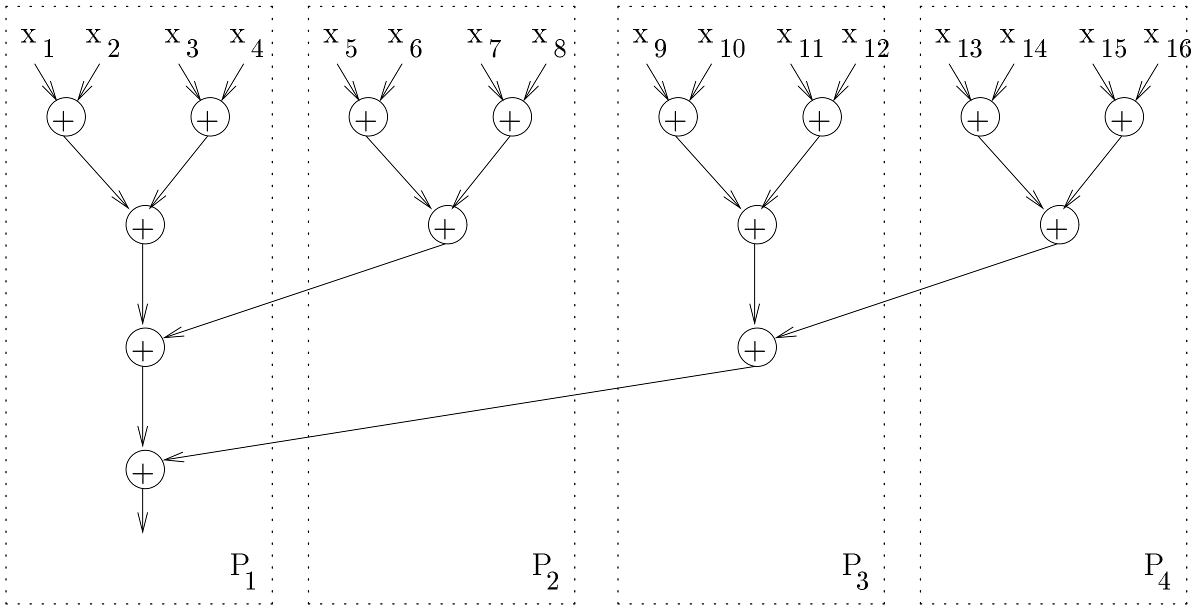

Figure 1.1: Illustration of the summation of n=16 numbers by p=4 processors.

The effective and efficient use of parallel systems requires a non-trivial transformation from sequential computing techniques to parallel computing techniques. This course aims at systematically introducing all the aspects of parallel computer architectures, designing parallel algrithms and parallel programming techniques in a fundamental and yet practical manner.

1.1 What is parallel computing

Consider the problem of summing n numbers x_i, i.e. calculating $S = \sum_{i=1}^{n} x_i$. Suppose that we have p processors available for the summation. The n numbers are divided into p groups each of n/p numbers (assuming n is divisible by p), and each processor sums up a group of n/p numbers. This takes a time of $n/p - 1$ additions. Finally, the p partial sums are summed recursively in a pair-wise manner (Fig. 1.1). This will take a time of $log(p)$ additions. In total the summation takes $n/p + log(p) - 1$ additions as compared to $n - 1$ additions using a single processor. Therefore, in terms of the number of additions, the calculation time is reduced by a factor of nearly p for $n \gg p$. However, there are extra overhead introduced due to the requirement of communicating the partial sums between the processors.

An analogy can be drawn to a real life situation. For example, consider a team working on a large engineering design project. The design work is divided into a number of sub-designs, and then each team member makes a sub-design. Finally all these sub-designs are verified and put together into the final design. In this design process, we can identify a number of phases. First, the design project has to be divided into a number of sub-designs (*partitioning*). Then, the work of one or more sub-designs will be assigned to each

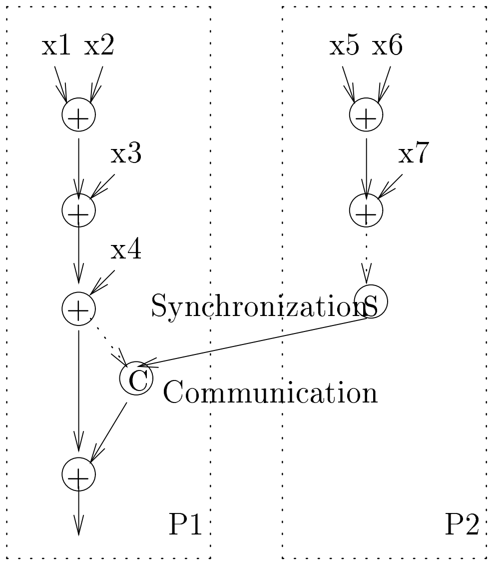

Figure 1.2: Illustration of synchronization and communication.

individual team member (*task assignment*, or *mapping*).

Generally, a good design team can make a complex design much faster than a single designer. Even a super designer (expensive) cannot beat an organized team of average (cheaper) designers. That is the power of parallel processing! However, we should recognize that besides all those advantages of concurrent engineering, it also bears additional overhead inherently. First, there must be *communication* between the team members in order to fit the sub-designs into the final design. Second, the sub-designs may depend on each others completion. Therefore it can happen that during the design process, some team member has to wait for the completion of the sub-design of another team member before he/she can continue (*synchronization*). These are the extra overhead introduced with 'parallel processing'.

Fig. 1.2 illustrates the synchronization and communication in a parallel summation of 7 numbers by 2 identical processors. Processor 2 finishes the addition of 3 numbers (x_5,x_6,x_7) before processor 1 finishes his. Hence, processor 2 must wait for processor 1 doing its fourth addition (synchronization). Then, processor 2 sends the sum of 3 numbers to processor 1 (communication). Processor 1 receives the result from processor 2 and adds it to its own result to produce the final result.

1.2 The need for parallel processing

The advent of computer technology and simulation methods has a great impact on the way of scientific research. It is becoming more and more evident that computer simulation is now a third approach[1] besides the theoretical and experimental approach. It is being

[1]Today, computational physics, computational chemistry, and computational mathematics are important studies in the department of physics, chemistry and mathematics, respectively.

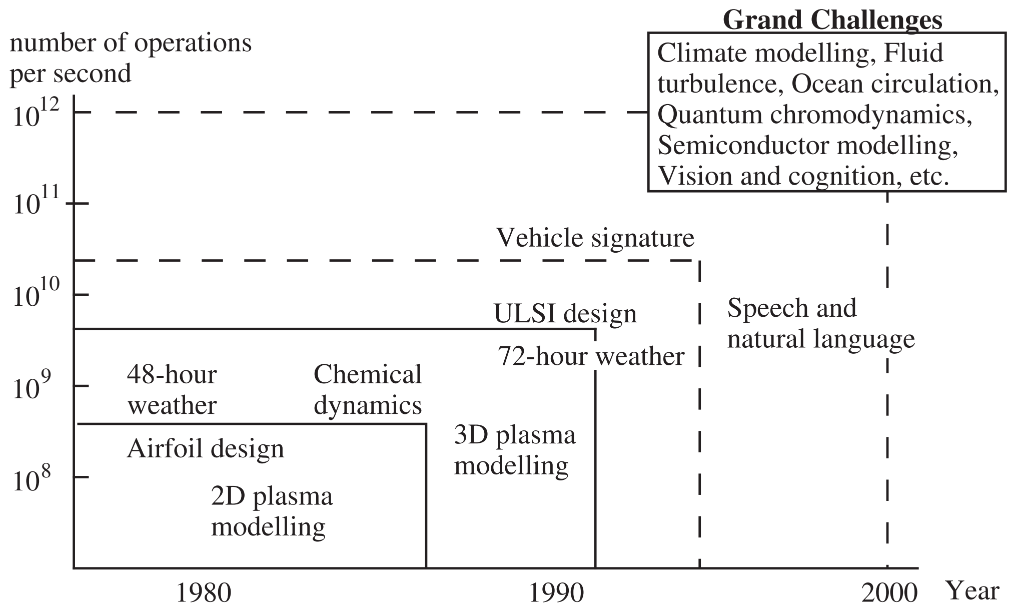

Figure 1.3: Illustration of the estimated demand for computing power by a number of "Grand Challenge" problems [2].

increasingly recognized by the industry computer simulation is an attractive alternative to conventional experimental tests. For example, in airplane design the computer can be viewed in context as an "electronic wind tunnel". In many areas of scientific research, a powerful computer is an instrument like a powerful telescope is for astrophysic discovery.

Although the computing speed of computers has increased significantly since the birth of the first electronic computer, the demand for more computing power has not diminished. For some scientific and industrial computer applications it is estimated that they require computers with gigabytes of total memory and with from 10 to 1000 gigaflops performance. For example, in a report [2] prepared by the Committee on Physical, Mathematical, and Engineering Sciences (U.S.A.), a large number of so-called "Grand Challenge" problems (Fig 1.3) have been identified. These problems require a computing power beyond the teraflops to solve them. Furthermore, many computer aided modeling, design, and optimization problems demand an increasing computing power in order to provide interactive facilities.

1.3 Parallelism in nature

What is parallelism? *Parallelism* is the measure of the number of tasks being or can be performed concurrently. So, we say the parallelism of a program (execution) is high if the number of tasks which are being executed is large, and a problem with a high degree of parallelism means that the problem has a large number of tasks which can be performed in parallel (thus the *potential* of being performed in parallel).

Parallelism exist every where in the real world. Natural systems, complex technical

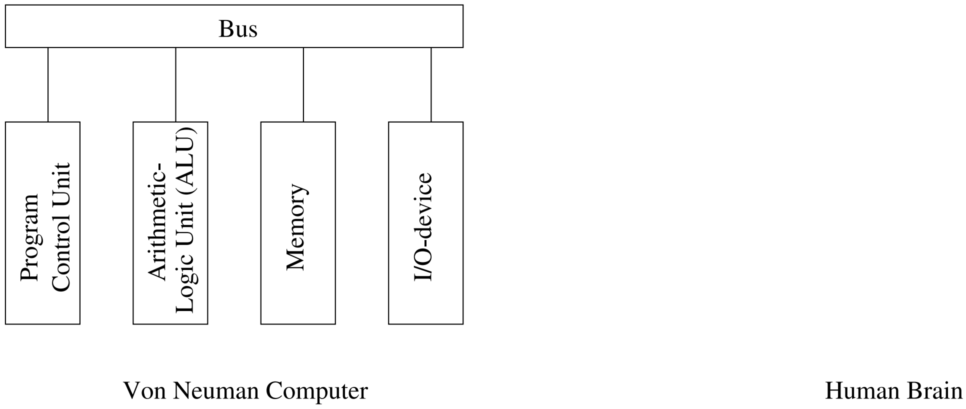

Figure 1.4: Von Neumann computer versus brain (neurons).

systems, and even social changes are all highly parallel processes. The growth of a plant is simultaneously influenced by a large number of factors. The starting of a motor requires the coordination of many components. On the stock market, share prices depend on the positions of thousands of purchasers and sellers at the same market as well as other stock markets over the world.

The human brain works in parallel as well. Figure 1.4 shows a comparison between a sequential von Neumann computer and a brain which consists of a (natural) network of *neurons*. A neuron is a device which can generate outputs to many other neurons using inputs from many neurons. Some neurons have up to 100,000 such inputs via an antenna-like structure referred to as a *dendrite*. The brain can be viewed as a parallel computer where many neurons simultaneously perform a variety of memory-functions, sensory, pattern recognition, control and reasoning. It is a flexible, powerful parallel computer which is in many cases the most sophisticated implementation of parallel processing yet discovered. The specific algorithms used by the brain are subject of intense theoretical and experimental study and are outside the scope of this text. One thing to be mentioned here is that the power of the brain lies more in its remarkable multiplicity of connections, rather than in the speed of individual neurons. Electronic circuits (IC chips) run at a cycle time of from 10^{-8} to 10^{-10} seconds. Neurons operate at a maximum frequency of about 10^3 hertz, or a cycle time of 10^{-3} seconds. However, neurons may have as many as 100,000 connections, in contrast to today's parallel computers which typically have less than 10 connections per processor.

The idea of parallel computing dated back much earlier (even before the birth of the first electronic computer). In 1922, L.F. Richardson wrote in his book *Weather Prediction by Numerical Process* [3] the idea of using a number of individuals to perform the calculation in the weather prediction. In order to make the calculation fast enough (i.e. ahead the real weather development), he proposed to divide the globe into a number of subspaces and to assign the task of calculations corresponding to each subspace to an individual. He estimates that 64.000 individuals are required for simultaneously performing the calculations. This is a form of parallel computation. The technique of dividing a domain into a number of smaller parts and to solve them simultaneously is known today as domain decomposition which is often applied in parallelizing large scale computer applications.

From the above considerations, we have seen that there is a great potential of inherent parallelism in nature. This promises that many problems to be modeled and solved by computers can be efficiently solved by means of parallel computers. The sequential nature of today's programming practices is simply a historical outcome of the von Neumann computer model which was very successful. However, its sequential nature is an artificial restriction, a way to organize operations to fit the sequential computer.

The efficient use of parallel computers is often not a trivial transition from the sequential programs and experiences. First of all, many existing programs require a complicated transformation (e.g. loop-transformation or variable renaming or copying) in order to become parallelizable. Secondly, many existing algorithms, which are often optimized for sequential computing (e.g. requiring a minimal number of operations), have a low degree of parallelism or are even inherently sequential. In these cases, new algorithms with a higher degree of parallelism must be developed. Thirdly, many problems are modeled with the 'sequential' computing model in mind so that many inherent parallelism is lost during the modeling steps. Therefore, it is sometimes preferable to remodel the problem and keep as much parallelism in the original problem as possible (*parallel modeling*).

1.4 Issues in parallel computing

The use of parallel computers introduces a number of new problems which are not present or not important in the traditional use of serial computers. These problems can be distinguished into three categories:

- parallel computers — How to organize and construct computers with up to thousands or more processing elements. The increased number of degrees of freedom in a parallel architecture makes the design choices more complicated. The uncertainty of which architecture is the best one (which is dependent on the state of technology) has led to a large diversity of existing parallel computers.

- parallel algorithms — A parallel computer is of little use unless efficient parallel algorithms are available. The issues in designing parallel algorithms are very different from those in designing their sequential counterparts. As we will see in Section 2.2, the notion of *optimality* of a parallel algorithm is different from that of a sequential algorithm. In addition to the operation-count and memory-size requirement in the complexity analysis of sequential algorithms, the communication complexity must also be considered in analysis of parallel algorithms. Furthermore, the performance of a parallel algorithm depends on the architecture of the parallel computer being used.

- parallel programming — System software and application software are dramatically affected by the architectural choices. Parallel programming paradigms and languages which are flexible for adapting solution methods and algorithms and allow efficient implementation and are easy to program must be developed. It is also neccesary to develop compilers that can automatically extract implicit parallelism from (sequential) programs. To facilitate the programming of parallel computers, comprehensive programming environments and tools are required. In order to develop efficient and portable parallel programs, software development methods tailored to the construction of parallel software must be developed and applied.

1.5 Outline

The text is organized as follows. The three topics, parallel computers, parallel algorithms and parallel programming, are discussed in Chapter 2 to Chapter 5. The first part of Chapter 2 gives an overview of different parallel architectures and classification schemes of parallel computers are described. The second part of Chapter 2 discusses the properties of parallel algorithms. The notion of the complexity of a parallel algorithm in considered. Chapter 3 gives an introduction to the design and analysis of parallel algorithms. Because the nature of this course, the emphasis is on numerical algorithms. However some non-numerical algorithms are also included. A number of basic techniques for the design of parallel algorithms are discussed. They are generally applicable, also to non-numerical problems. We also discuss the relation between algorithms and architectures with respect to performance and optimality. Chapter 4 gives an introduction to parallel programming. Some basic issues like parallel languages, communication and synchronization are discussed. The well known message-passing libarary MPI (Message Passing Interface) and the BSP (Bulk Synchronous Parallel) programming library are described. Chapter 5 considers the issue of parallel software development. The chapter starts with quality aspects of software. Efficiency, portability and correctness are discussed. Subsequently the principles of abstraction is described and they are used to investigate a software development trajectory. Some parallel models of computation will be discussed, followed by the definition of three elementary development steps. Finally an approach for structure parallel software development is presented.

After the introduction to different aspects in parallel computing in Chapter 2 through 5, some example parallel applications are discussed in Chapters 6 and 7. Different aspects in parallelizing a problem are considered, varying from problem analysis to choices in parallel methods. The performance and communication overhead are analysed.