PARALLEL PROCESSING

by Professor C. A. R. Hoare, Mr. I. Page, Dr. M. Giles, & Mr. R. McLatchie,
Computing Science Laboratory, University of Oxford

1. Theoretical Foundations – Professor C. A. R. Hoare FRS

My first and most painful encounter with the problems of parallel processing occurred in 1965, when I was in charge of an operating system project for a small computer manufacturer. Complete failure of that project has led me since that time to devote my main practical and theoretical research to the topic of parallelism. The results of widespread research are now available for practical application, at a time when economics, technology, commercial availability and even ultimate physical limits dictate that the next advance in high-speed computation must involve parallel processing.

My task is to explain the basic concepts of parallel processing and how they apply equally to hardware and to software, to microscopic as well as macroscopic parallelism. The necessary abstraction has been achieved by some rather deep mathematical research, but I hope to show that the results are as simple and memorable and usable as the elementary algebra which we learn at school.

Like all mathematicians, I have to use single letters to stand for possibly large and complicated physical objects in the real world. So let P and Q stand for computational devices, perhaps CRAYS, perhaps workstations, perhaps hardware components, or perhaps programs timeshared on a single computer.

I will use parallel bars ($P \| Q$) to indicate a processing assembly that consists solely of the two components P and Q operating in parallel or concurrently. Each component starts at the same time, proceeds independently, and the whole assembly terminates only when both components, have terminated. Already I can formulate a general mathematical law about parallelism. Like addition and scalar multiplication, the parallel operator is communicative:

$$P \| Q = Q \| P.$$

It does not matter which order we write them, their parallel behaviour is the same. Similarly, at any time we can add a third component R to the assembly, as shown by the familiar associative law

$$(P \| Q) \| R = P \| (Q \| R).$$

These are just two examples from several hundred algebraic laws which have been discovered and proved by research into parallelism. The particular achievement of research at Oxford is to combine both sequential and parallel programming in a single theory known as CSP—Communicating Sequential Processes.

The aspect of communication is essential. A computational device communicates with its environment through one or more directed *channels* to which we give names like c and d. These carry messages reliably from an outputting device to an inputting device. They may be implemented as simple wires, by local area networks, or even by subroutine calls between programs timeshared on a single computer; all of these communication media can be described by the same abstract theory of channels.

So if c is an output channel, then at any time, a process can attempt to perform an output on channel c (written $c!37$) of any value (say 37). If c is also an input channel of another process, then at any time that process can attempt to perform an input from c (written $c?x$), and the value is assigned to one of its own local variables (say x), so that afterwards the value of x is 37. Now, when processes are run in parallel, we arrange that the channels which have the same name are connected, so that anything that is output by one process on channel c is instantaneously available for input by the other, as soon as it is ready to perform the corresponding input on the same channel. This fact can also be conveniently expressed as an algebraic law. When the input is run in parallel with the output on the same channel c, the effect is the same as the execution of a simple assignment statement, which does nothing but store the communicated value 37 in the state variable x,

$$(c!37 \| c?x) = (x := 37)$$

so that afterwards the value of x is 37.

This is a remarkably succinct and yet exact statement of the essential property of communication of parallel devices; it relates the novel concepts of parallel processing to the more familiar concepts of sequential programming. It can be used to transform programs from sequential form to take advantage of parallel hardware, or in the other direction if parallelism is unavailable or uneconomic. Our theoretical research has already found beneficial application on an industrial scale in a project which recently won a Queen's Award jointly for INMOS and the Oxford University Computing Laboratory. Using algebraic laws like those I have shown you, we were able to ensure correctness of design of a hardware floating point unit before implementation and without testing.

Correctness by construction is the main contribution of a good theory to engineering practice. Other advantages are future-proofing of designs to take immediate advantage of continuous improvement in computer architecture; portability of designs across machines of differing manufacture, and transfer of technology from one application to another. At the Oxford University Computing Laboratory, we have a policy that our research should go more than half way towards its practical application in Industry. That is why we have established Oxford Parallel as a unit within the Laboratory, to share the advantages of our long term research with our network of scientific collaborations and customers.

2. Cutting Costs in Engineering Development by Parallel Processing – Dr M. Giles

My aim is to explain why it is that Computational Fluid Dynamics (CFD) plays a major role in the Aerospace Group of Rolls-Royce, and what parallel computing can contribute to this. The modern aeronautical gas turbine is an extremely complex beast. Until relatively recently the design of these engines was based on overall system modelling together with extensive testing of individual components. The costs of this approach are high and as a consequence, it is almost prohibitively expensive to develop an entirely new engine from scratch. It is much more common to modify an existing successful engine to reduce the technical risks. Despite this, the test costs of a derivative engine can be of the order of £30M.

The primary aim of CFD is to reduce the amount of testing that is required, by performing extensive computer modelling and eliminating poor designs early on. In fact, in the last ten years, the heavy use of CFD has produced a 50% decrease in the number of tests required for a new design. Not only does that lead to cost savings, it also reduces the design time and gets the product to market faster.

A very important secondary aim of CFD is to allow the designer to be more innovative. International competition is fierce and it is getting increasingly difficult to make improvements in efficiency through incremental changes in existing engines. Often it is brand new ideas which produce significant improvements, but there are just as many ideas which are unsuccessful. CFD can evaluate all the ideas, good and bad, without being tied to the empirical experience of previous designs.

A consequence of the heavy dependence on CFD is that Rolls-Royce now owns a CRAY-YMP. The overall technical computing budget is of the order of several £m per year, with a large fraction of that due to CFD. Rolls Royce are now interested in getting the full potential out of parallel computing. There are three benefits here.

(i) The first is to achieve the same as today but at lower cost. Savings of as much as a factor of 10 should be easily possible, since, for example, 4 IBM RS/6000 workstations have the equivalent horsepower of a single processor CRAY-YMP, with the RS/6000's costing about £50k a piece and the CRAY-YMP processor costing about £2M.

(ii) In practice, lower overall costs are rarely the consequence of computing improvements. Much more likely is that the costs will stay fixed but that the speed of the calculations will be greatly increased (leading to shorter design times) or the number and quality of the calculations will be increased (leading to better designs).

(iii) Pushing further in this direction, it becomes possible to tackle a whole new range of problems which were previously considered impractical. One example is interactive design, in which a designer can examine 'what if?' scenarios with the speed and ease of using a spreadsheet tool. Another example which I will discuss in a bit more detail is the simultaneous modelling of a complete engine.

Before talking about the application, I would just like to comment on why it is that parallel processing is becoming so important now when it has been touted as up-and-coming for many years. The key here is the relative evolution of single processor workstations compared to single processor supercomputers.

Ten years ago, comparing the first CRAY computer to the first workstation from DEC, there was about a factor 100 difference in speed. Now there is about a factor 4 difference between the fastest CRAY processor and the fastest IBM workstation processor. In large part this is because improvements in processor speed require enormous investments in manufacturing technology, and that requires enormous sales to finance it.

The important quantity is the performance-to-price ratio, the relative performance per unit cost. Ten years ago, the CRAY and the DEC workstation delivered comparable performance per pound. If you wanted more speed at the minimum cost you simply bought a faster machine, and didn't have to worry about parallel computing. Today, with workstations and supercomputers having each stayed roughly constant in cost over time, there is now a big difference in the price/performance of the two. If you want the fastest overall performance for a given price the optimum solution is now to use lots of workstations, but this means that you have to think about parallel computing.

The supercomputer manufacturers are not stupid. They see this as clearly as anyone. All of the major players are now designing massively-parallel supercomputers connecting together lots of workstation processor chips. The key though is software development to make effective use of all these processors. Here, at Oxford, I am working on this issue for turbomachinery analysis by using a mildly parallel collection of a number of workstations connected by a fast network.

In the engine design problem, since the engine has a number of components, the simplest and most effective way to split the modelling problem is by component. The idea therefore is that each component is modelled on separate workstations. The simplest components such as a single blade row may need just one workstation; more complicated components may need more than one workstation. Information about the physical coupling between components then has to be exchanged between the workstations.

This approach is shown schematically in the sketch (Fig. 1). Here we are looking at a small engine with a three-stage compressor, the combustor, and a one-stage turbine. Each solid box represents a workstation, and the large dotted box is the collection of three workstations working together on the chemical modelling within the combustor. This is particularly computationally demanding because of today's concerns with the emissions of nitrous oxides.

The key to good performance is that lots of work should

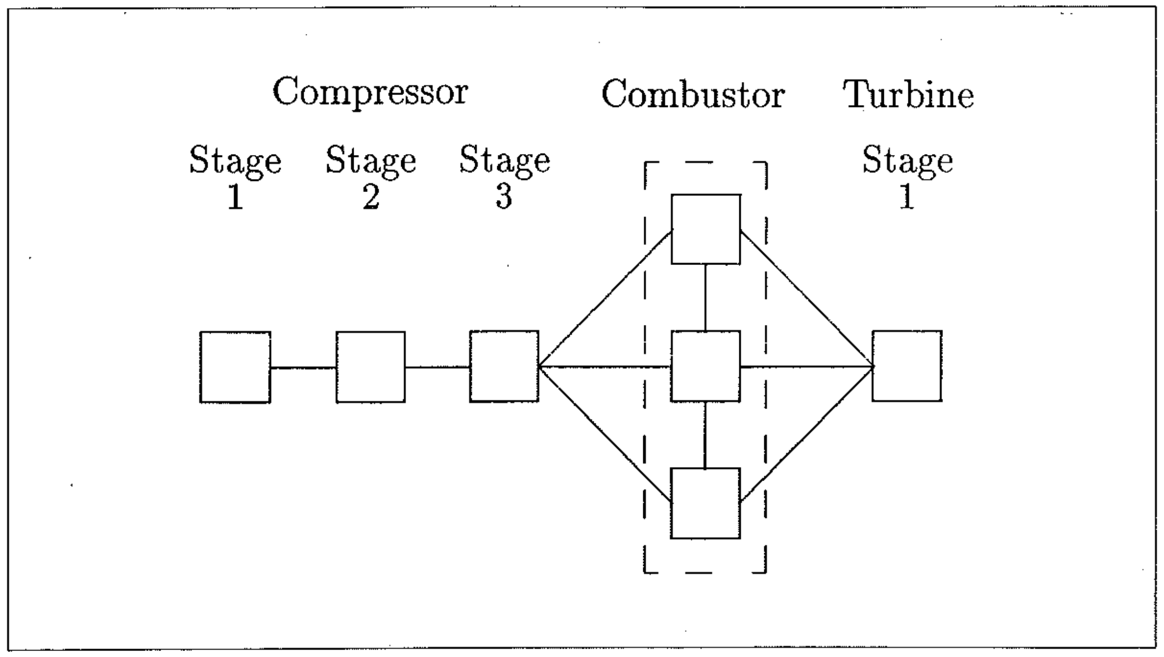

Figure 1
Schematic diagram for small engine with three-stage compressor, combuster, and a one-stage turbine.

be done within each box relative to the amount of information that must be communicated between boxes, because the communication is relatively slow.

Some typical numbers will give a feel for this aspect of the problem. The 3D grid in one of the stages may have 50,000 grid points, and each grid point will require on the order of 4,000 floating-point operations per iteration. This then takes about 5 seconds on a machine with a speed of 40Mflops.

The communication requirement is the exchange of about 1000 variables between each neighbouring pair of stages. Using message-passing software called PVM across a standard Ethernet network this takes about 20 milliseconds.

Therefore, for this type of application the communication costs are negligible.

A final comment is that an important attraction of the message-passing software PVM is its ability to operate on a network composed of computers from a wide variety of vendors, including workstations from IBM, HP, DEC, and SUN as well as vector and parallel supercomputers from CRAY, Thinking Machines, and Intel. This makes it possible for software developers to write portable code which protects their investment in developing these parallel applications, which is a critical feature for the long-term success of parallel computing.

3. Transforming Software into Parallel Hardware – Mr Page

I have been designing, building, and programming parallel computers for eighteen years. I have worked in industry and universities on demanding computing applications using both hardware and software. My engineering background has always disposed me towards practical solutions which are also based on sound theory. I'd like to illustrate this with a recent project which underlines the approach that my group is taking to designing and implementing new computing and IT systems involving parallelism.

Earlier this year, a colleague in the Engineering Department (Henry Lau) came to ask my advice on a problem. He is working with Harwell on robots to operate inside nuclear reactors. One of the (minor) problems of putting robots into such environments is knowing from the outside exactly where they are at each moment. Part of that knowledge involves measuring the angles of the various joints in the robot's arm. This is conveniently done with a device known as a shaft encoder, attached to the joint. The device has two output signals generated from photo-sensitive detectors whose light input is interrupted by a pattern of transparent and opaque regions on a glass disc. Two detectors are needed so that the direction of rotation can be determined as well as the amount of it. Rotating the encoder disc results in two digital pulse streams and it is necessary to build some special-purpose digital hardware to count these pulses so that we can monitor the shaft angle.

A circuit board which was designed in the Engineering Department a few years ago to take the outputs from four shaft encoders and interface them to the control computer took six weeks of intensive effort to design, build and test. Henry's problem was to design and build another such interface board for a new robot; but with a higher speed of operation, higher accuracy, additional functionality, smaller physical size, lower development cost, reduced development time, and with increased flexibility.

My first step was to ensure that I completely understood what Henry wanted from his hardware interface, which I did by writing a program. The whole program is too large to be shown here. We present a key fragment which converts the two-bit values from the shaft encoder into a number which represents the angular position of the shaft:

```
SEQ
  ENCODER ? CURRENT
  IF CURRENT ≠ PREVIOUS
    THEN
      IF CURRENT [0] = PREVIOUS [1]
        THEN
          ANGLE := ANGLE –1
        ELSE
          ANGLE := ANGLE +1
  PREVIOUS := CURRENT
```

The program does three things in sequence. First, it takes the current output value from the shaft encoder device and stores the two bits in a variable called CURRENT. Secondly, it determines whether the encoder has moved since it was last looked at by comparing the value in CURRENT with the value in the variable PREVIOUS. If it has, then a comparison determines if the shaft has moved one unit clockwise or anticlockwise and this is recorded by incrementing or decrementing the value in the variable ANGLE.

The third action is simply to update the CURRENT po-

```
WHILE $\neg$ finished
 PAR
   previous := $\text{current} \lhd c2 \vee c3 \vee c4 \rhd$
                 previous
   current  := $\text{encoder} \lhd (\text{start} \vee c0) \wedge$
                 $\text{encoder\_rdy} \rhd \text{current}$
   angle    := $(\text{angle}+1 \lhd \text{incr} \rhd \text{angle}-1)$
                 $\lhd \text{decr} \vee \text{incr} \rhd \text{angle}$
   c0       := $(\text{start} \vee c0) \wedge \neg \text{encoder\_rdy}$
   c1       := $(\text{start} \vee c0) \wedge \text{encoder\_rdy}$
   c2       := decr
   c3       := incr
   c4       := $c1 \wedge \neg \text{changed}$
   finished := $c2 \vee c3 \vee c4$
   start    := 0
WHERE
   bitdiff  = $\text{current}[1] \neq \text{previous}[0]$
   changed  = $\text{current} \neq \text{previous}$
   incr     = $(c1 \wedge \text{changed}) \wedge \text{bitdiff}$
   decr     = $(c1 \wedge \text{changed}) \wedge \neg \text{bitdiff}$
```

Figure 2
Normal form program.

sition to be the same as the PREVIOUS position so that we can repeat this program again and again. In fact we have to repeatedly run this program much faster than the shaft encoder could ever move, which in practice means executing the program many millions of times each second. If we do this, we can be certain that the ANGLE variable is a true representation of the physical position of the robot arm. I have had to leave out a large amount of detail in this presentation, but it is worth noting that a simple, but yet complete, version of this program was constructed within two hours of the problem first being presented.

So why did I write this program and why have I inflicted it on you?

Firstly, the program is a formal specification of the interface. We can have much more confidence in such a specification than one that was written in technical English. Many of you will no doubt have suffered from specifications which were incomplete, ambiguous, contradictory, or were interpreted by an implementor in a way not envisaged by the author. Moving to an easily understood formal specification can help eliminate such problems.

Secondly, since this specification is in a formal (mathematical) language it is open to automatic checking for certain kinds of mistake—something that is not even remotely possible with specifications written in English. The sorts of things that can be easily checked for are that there are no missing parts of the specification, or that everything that is in the specification is actually relevant, or that the various parts of the specification agree precisely on their interfaces with each other.

Thirdly, since this specification is also a program, it can be executed on a computer and we can check that it actually does what we expect. So we have, almost for free, a simulation of the hardware that we want and it is available to us very early in the design cycle. It was through such a simulation that Henry was able to confirm that we had indeed captured precisely the behaviour that he wanted from the interface hardware.

In fact, the only thing wrong with our program as a final implementation of the interface is that it is not fast enough. Executed on a conventional computer this program would be perhaps a thousand times slower than necessary. There is simply no alternative but to implement our interface as special-purpose hardware; hardware whose behaviour has been captured by our program.

To take stock, we now have a formal description of the desired behaviour and what we have to produce is a description of some hardware that implements that behaviour. That description will be a circuit design consisting of many logic gates wired together. Normally at this point, we'd give our specification to a trained engineer who would design and implement the hardware. And if that were all, I think I would have already shown you a cost-effective way of designing new digital systems, more quickly, and with a higher degree of confidence in the results.

The fourth and most compelling reason for writing specifications as programs is that we can apply the algebraic laws mentioned by Prof. Hoare to our program and transform it into variety of different programs. In particular, we can transform it into what we call a Normal Form program. Here is the Normal Form program derived from the encoder handling fragment (Fig 2).

All Normal Form programs consist of a single assignment embedded in a repeat loop. This program captures all the parallelism available in the original program, and these Normal Form programs can be generated automatically by a computer which applies the algebraic transformations.

This program is much harder to understand than the original program, but then it is not intended for human consumption. So what good is it?

The really nice thing about this program is that it can be interpreted directly as a hardware circuit. Everything on the left hand side of the parallel assignment can be interpreted as hardware storage devices (or latches), and everything on the right hand side can be interpreted as a set of logic gates. In everything but superficial appearance, this program is the circuit diagram of the hardware that we seek (Fig. 3). This circuit diagram contains exactly the same information as the Normal Form program. What we have in effect done is to move smoothly from the world of computer programs to the world of electronic circuits, taking with us a guarantee that we cannot possibly have introduced any errors by crossing this boundary. This transformation of software into hardware is called hardware compilation.

The Normal Form program, when interpreted as hardware, proceeds with the parallel assignment being executed once each clock cycle. The number of clock cycles for the program to complete represents the sequentiality of the program and the complexity of the assignment statement represents its parallelism. Using the algebraic laws, we can also trade off the sequential and parallel aspects against each other to achieve different cost/performance constraints for the hardware.

In fact, we can now compile systems with both hardware and software components, since we have encompassed software and hardware in the same theoretical framework. We can use this to produce different implementations by trading off the hardware and software components against each other to achieve desired cost and performance measures. As a further advantage, development of the normally problematical hardware/software interface becomes risk-free, since it is derived automatically.

The new shaft encoder interface board that was constructed with our hardware compilation technology uses Field-Programmable Gate Arrays (FPGAs) which enable us to implement hardware almost instantly. The board consists of little more than the FPGA chip itself, a chip to communicate directly with the control computer, and a few (non-digital) components necessary for each shaft encoder. It took only a few hours to design this board.

One result of using FPGAs is that the physical construction of the hardware can be started (and maybe even completed) before the full specification of the system is available. They allow us to defer a great many design decisions and achieve a high-degree of product flexibility. In our case, the same encoder interface board has been re-used a number of times with different interface circuits without resoldering a single wire!

In summary, I have shown you how we use parallel pro-

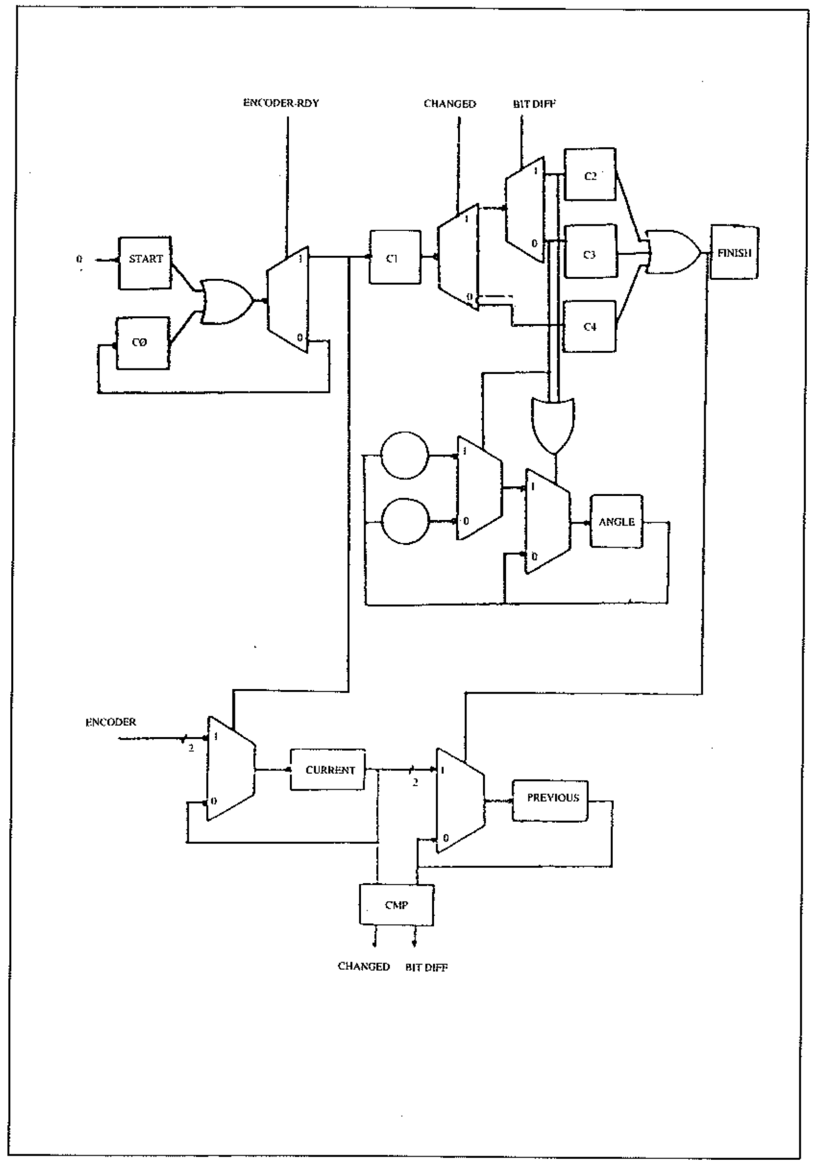

Figure 3
Equivalent circuit diagram.

grams as specifications of systems and automatically generate the hardware and software components. In straightforward cases, we can go from problem specification to a working system in days, or even hours. I have shown how the technology allows us easily to re-engineer systems to meet a variety of cost/performance constraints, and how it significantly reduces risk in a notoriously difficult area of system design.

We're investigating a number of application areas and we have so far found this methodology has useful benefits in all of them. I've told you about part of a motor control system and we hope soon to extend this to a complete, high-precision motor control system using exquisitely sensitive laser-based encoders. A feature of our approach is that we have a high degree of control over real-time behaviour which therefore makes it at-

tractive for many real-time applications. We have some quite exciting ideas for new methods of providing fast access to large databases using hardware search engines which will also support transaction processing. We have demonstrated some basic graphics and image processing algorithms implemented this way, and we are now looking at real-time video processing. We have also implemented some data compression, computational chemistry, and neural network demonstrations. And finally, I want to mention the whole area of telecommunications, which is awash with good applications for this technology. We believe that there are many other applications which could benefit from this approach.

In short, from science and engineering to information technology and data processing, this technology has, I believe, enormous possibilities for massive improvements in productivity and cost-effectiveness. I think the UK may have a world lead in understanding this new technology and its implications. Whether, as a country, we can turn that into a long-lasting economic success story may partly depend on you.

4. Oxford Parallel - Mr R. McLatchie

The previous talks describe how the leading edge in Programming Theory, Numerical Computation and Hardware Production is being advanced in Oxford. I will describe a unit, known as Oxford Parallel, which has been set up here to help industry and commerce exploit the exciting cost performance breakthroughs following in the wake of this leading edge.

Oxford Parallel is a self-funded unit within the Oxford University Computing Laboratory drawing on the combined talents of all departments of the University and the SERC Rutherford Appleton Laboratory who are partners in this enterprise. Oxford was successful in becoming one of only four Parallel Application Centres established under the DTI/SERC Parallel Applications Programme. It has an initial funding of £3M over a four year period, to stimulate the exploitation of parallel processing in industry and commerce.

The thinking behind the Parallel Application Programme is that stimulation is best done by means of active collaborations between Oxford Parallel and client organisations within industry and commerce. The focus is on working together to tackle the real problems of current and vital interest to the client. The kind of problems that will make a real difference to the economic performance of the client's business.
Industrial and commercial partners retain full rights to exploit the joint results while Oxford Parallel retains rights to its own results which will be applied in future projects and to assist the organisation to become self-financing in due course.

DTI/SERC support provides matched funding for UK enterprises. The client company meets all of its own costs. Her Majesty's Government meet some of Oxford Parallel's costs but the client will be expected to contribute to an increasing degree as the Parallel Application Programme proceeds. In effect Oxford Parallel's contribution to the collaborative project is available to client companies at a substantial discount.

Oxford Parallel operates in three main areas of computing application.

Michael Giles has already described an application in science and engineering showing that there is growing justification for using computation in place of building engineering prototypes. In many fields of scientific enquiry, a large amount of computation is being applied to model the systems under study or, indeed, to assist in the search for new knowledge. So in pharmaceuticals, for example, parallel computation can be applied both to modelling molecular dynamics and to improve the selection of likely compounds for future study. Here is a way to reduce the average time of 7 years to bring a new drug to market, reducing risk and cost.

As Ian Page has shown, the real time processing of ever increasing volumes of data force one to seek the highest possible performance from computing hardware. One contribution here is the development of the tools for the rapid and reliable development of reconfigurable hardware enabling the best possible trade-off between cost and performance.

Undoubtedly the greatest potential area for application of parallel processing is to problems in business management and in commerce. Until recently, organisations in this sector have shown a reluctance to become involved in parallel processing. Yet a number of the problems facing these organisations are ideal candidates for parallel processing. Organisations handling very large numbers of essentially independent data records—for example retail organisations with large amounts of data from point-of-sale, or life assurance companies with large collections of policy data—can analyse this information in a more timely manner by spreading the computation over a number of processors. The benefits from more rapid analysis include better control, faster response to market needs and the ability to prevent fraud by the early detection of unusual patterns. Oxford Parallel has tools in each of these areas to help clients begin to make more effective use of parallel processing at lower cost.

Oxford Parallel is currently working on eight collaborative projects. For reasons of commercial confidentiality I am unable to name all the companies involved. Where I can, I will indicate the company involved.

Michael Giles' talk has already shown that substantial sums can be saved by using computer simulation in Computational Fluid Dynamics. The same is true of the related field of electromagnetics. In both cases the computer modelling is saving substantial sums of money that would otherwise have to be expended in building and measuring prototypes. Our work in this area includes the development of parallel versions of AEA Technology's FLOW3D Code.

For British Gas plc, we are developing optimisation techniques to enable their operations managers to plan activities so as to make more efficient use of resources. In particular, the cost of operating the national transmission grid for gas delivery is expected to be reduced as the result of being able to calculate optimum operational tactics more rapidly than ever before. Since the introduction of gas fired power stations will inflict larger transients on the network than ever before, the improved control this program provides is vital. A further benefit, from a visual point of view as well as an economic one, is the possibility of reducing the number of bulk gas holders in the network because delivery can be better attuned to demand.

In two cases we are assisting hardware manufacturers to develop more effective methods for designing parallel hardware and putting it into production. Sharp Laboratories of Europe Ltd, for example, are working with us on techniques for the more rapid development of applications specific circuits which will enable them to get product to market faster than before. A very useful edge in a highly competitive market.

With Barclays Bank plc, we have been carrying out a study of more rapid methods for evaluating a large Life Assurance Portfolio. This application is typical of many in this market segment in that it consists of relatively straightforward calculations on very large numbers of independent data records. As I have already said, such calculations can be reorganised quite simply to use several processors in parallel, cutting the time to complete the exercise substantially.

We are working with one of Britain's largest oil companies on improved methods for modelling oil reservoirs. Because of the very high cost of finding new fields, it is vital to extract all of the economically available oil from existing fields. The more accurate modelling of the complex geology of fields in the North Sea, made possible by the increasing amount of computation that parallel processing provides, will allow more profitable decisions to be made about oil recovery.

For HR Wallingford, we are studying a range of existing software applications in order to provide guidance on how these applications can be developed to exploit future parallel systems. Building in provision now for the parallel future is expected to produce large gains in due course.

Let us look more broadly at what is happening on a worldwide basis. The sales of massively parallel computers in the last two completed years show a strong increase in demand that has already produced sales of $271M in 1991. The main players are Intel, Meiko, Thinking Machines and Kendal Square Research.

With the possible exception of Intel, none of the companies currently in this market are exactly household names to the general computer user. However the major computer manufacturers have spotted this trend and within the last year IBM, Cray, and Digital Equipment Corporation have all announced their intention of building and marketing massively parallel computers in the near future. ICL have also been saying 'the future of computing is parallel'.

The predictions from the Smaby Group in 1992, for the sales of supercomputers suggest that the traditional vector-based supercomputer will continue to generate substantial sales revenues rising to $1.5B in 1996. But the inexorable rise of parallel machines will produce a worldwide market of over $1B by 1996. I am inclined to believe that parallel computers will in fact form a higher proportion of the supercomputer market in 1996 than predicted. In any case, there will also be a much larger market in parallel systems by 1996, which, while not classified as supercomputers because they are not expensive enough, are nevertheless more powerful than today's supercomputers.

In partial response to these predictions, President Bush's Chief Science Adviser Allan Bromley said in

November 1991: 'I believe that [the High Performance Computing and Communications Initiative] is the most important single initiative that we in the federal government can take, measured in terms of its potential impact on our society and citizens.' What is even more impressive is that he was successful in persuading Congress to spend of the order of $638M in 1992 on this initiative and to agree to increase this figure to over $1B per annum thereafter. In Europe the Rubbia Committee on High Performance Computing has made similar statements and will probably recommend that similar sums be spent on European programmes of work in supercomputing i.e. parallel processing.

Oxford Parallel is looking forward to helping clients from industry and commerce to obtain a fair share of these funds to assist them to obtain the benefits of parallel processing. Indeed the most important aim of all for Oxford Parallel is to help clients from industry and commerce to exploit parallel processing for profit.

DOLLARS, DIVIDENDS, AND DEVELOPMENT; AN INNOVATOR'S VIEW

by Peter Williams, Chairman and Chief Executive, Oxford Instruments plc

An address to The Oxford Innovation Society on September 24th 1992

First, let me echo James Hiddleston's apologies concerning tonight's change of speaker*. I realise it is a little like arriving to hear Pavarotti and getting instead the tenor from the local amateur Gilbert and Sullivan Society. However, the devaluation of the speaker was at least in sympathy with the mood of the times, so here I am.

Which brings me neatly to the title of my talk—Dollars, Dividends and Development—and to those very events which deprived us of the President of the Board of Trade's presence here tonight. If I may, during the next few minutes, I would like to draw this triplet of topics together and offer a few gratuitous observations. A very famous 'kotowaza' or proverb in Japanese is 'deru kugi wa utareru', which literally translated means 'the nail which protrudes is hit'. The flat shape of the top of my head is testimony to the wisdom in that proverb, but if a little more levelling is carried out as a result of tonight, so be it. I hope I speak amongst friends.

I also speak as one who spent a good part of last week glued to our Telerate screen watching our fate as an exporter unfolding in the currency markets. Whatever one's views of the sequence of events during those extraordinary days, can any company with a dependence like Oxford Instruments on worldwide markets be other than relieved at the final outcome? Those of you who have recently visited the USA, where something called 'gasolene' is now dispensed free every time you pay for a full windscreen service, must surely agree that the level at which the pound was previously valued relative to the dollar beggared belief. Yes, like most businessmen, I hope we will see a rapid return to normal relations with our European colleagues, but I must admit, I heaved a huge sigh of relief that sanity had at last prevailed at least as far as the value of the pound relative to the dollar is concerned.

It is reassuring to see recovery from recession now assuming priority over defence of the pound, although it is possible that few of you outside industry realise the true depth of the current recession. Yesterday's announcement by British Aerospace should, however, be enough to remind all of us of its severity. Fewer still perhaps realise the true international extent of the problem. The worlds markets are in dissarray everywhere, as we have seen in our export order books at Oxford Instruments. So how do we pull ourselves out of it? How do we plan constructively for the future?

Isn't this where innovation comes in? Despite the recession, we must all plan now for better days, and be aware that when these arrive, it will be the best prepared who will prosper. Which brings me to Dividends and Development. Those profits and cash flows we hope to see as the benefits of the lower pound begin to come through the system have conflicting claims placed upon them. Perhaps we will see inflation fueled, as some fear, and end up financing unwarranted pay reviews, as we saw in the 80's. I hope, and believe not. But how do we resolve

* The Rt. Hon. Michael Heseltine was unable to attend because of an emergency debate in the House of Commons following sterling's withdrawal from the ERM.