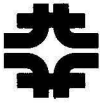
Fermilab Industrial Affiliates Roundtable on Supercomputer Developments in the Universities

Fermilab, May 1983

Fermilab Industrial Affiliates
Third Annual Meeting, May 19, 1983

Sponsored by Fermilab
And the Fermilab Industrial Affiliates



Fermi National Accelerator Laboratory
Batavia, Illinois

Operated by Universities Research Association Inc.
Under Contract with the United States Department of Energy

FOREWORD

The annual meeting of the Fermilab Industrial Affiliates provides an opportunity for research directors and senior technical personnel from the Affiliates and other companies to visit Fermilab and see first hand the on-going work of the Laboratory.

This year's meeting focused on supercomputers with a Round Table on Supercomputer Developments in the Universities. The round table grew out of the establishment of an advanced computer program at Fermilab headed by Tom Nash. That program has studied the possible approaches to supercomputers over the last year and a half. An active seminar series has brought in many university and industrial speakers in the computer area. At the same time, there has been growing national recognition that the U. S. must continue to play a role in the development of very powerful computers in the face of determined foreign competition.

The round table was spearheaded by Dr. Ken Wilson of Cornell, the 1982 Nobel Prize winner in physics. He emphasized that universities were good prospective buyers for the first model of a computer. They can do prototype software development and undergraduates do not mind the problems associated with getting the bugs out.

Other participants included the moderator and token industrialist Dr. Burton Smith of Denelcor, Dr. Arvind of MIT, Dr. Norman Christ of Columbia, Tom Nash, Dr. Jack Schwartz of NYU, and Dr. David Wallace of Edinburgh.

TABLE OF CONTENTS

Page

Dr. Kenneth Wilson.....1

Dr. David Wallace.....27

Dr. Jack Schwartz.....37

Dr. Thomas Nash.....47

Dr. Norman Christ.....57

Dr. Arvind.....69

Panel Discussion.....87

Fermilab Industrial Affiliates.....119

THE PARTICIPANTS

Dr. Arvind is on the faculty of Electrical Engineering and Computer Sciences of the Massachusetts Institute of Technology. He is an expert in data-flow concepts.

Dr. B. Norman Christ is a Theoretical Physicist on the faculty of Columbia University. Professor Christ is interested in using numerical methods to predict the fundamental quantum field theory. He is presently developing a special-purpose parallel vector processor to deal with such problems.

Dr. Thomas Nash is Deputy Head of the Fermilab Physics Department and Head of the Advanced Computer Program. He was one of the major designers of a super high-speed modular processor developed at Fermilab.

Dr. Jack Schwartz is a Professor in the Courant Institute at NYU. He is interested in machine architectures, programming languages and robotics. He is a member of the National Academy of Sciences. Dr. Schwartz is a principal mover behind the NYU Ultra-Computer Project which intends to produce a large scale fully parallel Super Computer in the early 1990's.

Dr. Burton Smith is Vice President of R&D of Denelcor, Inc. and concurrently Professor of Electrical Engineering at the University of Colorado, Denver. The Denelcor HEP is the first commercially available MIMD (parallel) Super Computer.

Dr. David Wallace is a member of the Physics Department at the University of Edinburgh. He is now involved with a group at Edinburgh doing theoretical physics calculations on the ICL Distributed Array Processor.

Dr. Kenneth Wilson is a Theoretical Physicist on the faculty of Cornell University. He is interested in applying new computer concepts to complex physical problems. He was the recipient of the 1982 Nobel Prize in Physics.

DR. KENNETH WILSON
CORNELL UNIVERSITY

Recently I have been in a learning process about technology transfer. The technology that takes place when you are dealing with computers is wholly different in kind and in scale when compared to any other form of transfer that I have been involved with. Let me give you an example that you will hear more about later. This hasn't resulted in technology transfer yet but one can see the possibilities. It's pretty hard to imagine how one would use a superconducting magnet of the kind they have here at Fermilab to make soap. On the other hand, Tom Nash of Fermilab has been analyzing the data analysis track finding programs that are used here. He has found after consultation with the computer science community that what they are doing is data base technology. Even the soap manufacturer has to keep track of how many kinds of soap he has and where they sell the best. This is just what data bases are good for.

This is what I mean when I imply that there really is something different in kind and scale for computing than virtually any other aspect of the technology transfer process.

First I will consider a definition of a supercomputer, then I will discuss the industrial supercomputer market, why it is necessary, the relation of industry and university in that market, and why I think that that market is presently too small for the health of US industry. I will then consider the barriers to progress in uses of supercomputers. Finally I will discuss what is happening in the university community to try to deal with these barriers.

office to shorten the time scale for all the paperwork that accompanies any new product. The other reason is the conversion from regional or national markets to world markets. This is driving the intense competition which is the business climate today. The businessman must take advantage of the shortened time scales to play a role in the market.

One consequence of the shortened time scales and accelerated rate of change is that the organization outruns internal experience with the product, with product design, with materials that go into products. There is not much use having 20 years of experience in how steel behaves if one is switching to a new composite material. When a business is in that situation it has to change the basis of manufacturing rapidly. It is a situation where basic science must often be substituted for experience.

Now one of the ways of using basic science in an industrial setting is through computer simulation, simulation of how a product will behave. This is totally standard when the science is just Newton's laws, the process of structural analysis for a bridge, let's say. No one goes out and builds a bridge to know whether it's going to hold up or not. The analysis is done in advance. The use of computers to do structual analysis is now totally standard. There are a variety of fluid and gas flow situations that are handled by simulation. A lot of the design of aircraft now involve simulation of the flow of air past the wing and other parts of the aircraft. Fluid flow simulations are heavily used in the oil industry especially when they are trying to understand what the oil does when they try to recover it. Not that those simulations are terribly successful all the time. The

What is a supercomputer? Well, a supercomputer in my definition (everybody has a different one) is a computer with high performance, targeted for scientific and engineering applications, and at least as powerful as the most powerful business data processing mainframe. The standard examples in magazine articles about the competition with Japan consist of the Cray machines which can be 10,20,30 times faster than a business mainframe or the CDC 205. Many of these stories fail to mention that our moderator, Burton Smith, is also the architect of a supercomputer which is in fact much more original than the Cray or the CDC machines. There are a variety of other computers of different kinds that fit the definition that I have given. This is one of the difficulties in dealing with the subject. It is not simply Cray and CDC anymore. For example, the company that I have worked most closely with, Floating Point Sytems, makes what are called array processors. I won't have anything more to say on what an array processor is except I've carefully crafted the definition so their products fit too.

Why are supercomputers important? Let me step back for a bit and let's look at what is happening today. Industry, all the non-defense industry and to some extent the defense industry too, faces shortened time scales for research and development. It's no longer true that one can develop a product and expect most of them to last 20, 50, or 100 years on the market before a replacement has to be designed. There are two reasons for these shortened time scales. One is that the impact of computers makes possible the shortened time scales with computer-aided design and computer-aided manufacturing. Now we even have the computerized

area where one would dearly love to use simulation, and it is useable in some circumstances but not all, is for microscopic problems, or problems at the molecular level. This is for the properties of materials, the equations of state of various kinds of substances and so forth. That is in an area where a lot of basic science is needed but there are some things that can be done.

One example where microscopic science was involved that created a lot of problems was the question of a natural gas pipeline through Northern Canada. This had to go through the region of the permafrost. What nobody was able to figure out was how strong the pipe would have to be to withstand 20 years of changes in the permafrost around the pipe and the pressures that would be put on the pipe. That's not something where one can go out and do an experiment, because the pipeline was needed immediately and not in 20 years. That gets into the subject of soil science. It happens that the world's expert on frost in soils is at Cornell. That's how I learned about the problem.

Another example with which I am more familiar in this whole problem of progress outrunning experience in industry is the situation in the high performance computing business itself where the industry presently faces the problem of switching from sequential processing to parallel processing. The need for parallel processing is simple and obvious. It is just that components of computers are becoming extremely powerful because of very large scale integrated circuits, and extremely cheap. However as computing becomes cheaper, while every other aspect of research and development gets more expensive, industry has to put

a larger fraction of its R and D investment into computers. Even though a computer comes down to a cost of several thousand dollars, the industry has to spend many millions of dollars on computers. The most efficient way to spend those million dollars would be to buy lots of 1000 dollar computers and have them run concurrently. But the computer industry has absolutely no experience in how to design parallel computers, has no experience in how to persuade their customers to buy parallel computers, and the users of computers have no experience in how they would use such parallel computers. I should say when I say there is no experience in design, about the sole exception is our moderator here, Burton Smith, which is one of the reasons he is our moderator today. When I talk to industrial computer designers they make no bones about the fact that that is the problem that is preventing them from moving towards parallel processing.

Now what is the role of the universities? I am not going to talk about the role of the universities in the traditional way - advance research which then leads to industrial advance research which leads in turn to industrial research and development and finally a product 20 years later. That is the typical technology transfer process. This was true, for example, for the laser which was invented around 1959 or 1960 and went through all those stages before it became an extremely important part of industrial communications.

But when we come to supercomputers the role of the universities is in the supercomputer market. We have a role in research and development but, for example, when we deal with Floating Point Systems we go through the Vice President for

Marketing. We don't go through the Research and Development Group at Floating Points Systems. That is because we have a role in the market, and, of course, when one has a role in the market the research and development role is subsidiary to that.

What is the role of the universities in the market? The one that is most obvious to everyone is the training of people and specifically the training of people who know what supercomputers are good for and who know what they are not good for. It is perfectly clear in the visits that I have made to industry that they suffer a very serious lack of people with that kind of training. That is, people lack the training to recognize opportunities for the use of large-scale scientific computing in an industrial setting. That is one of the reasons that when you talk about large-scale scientific computing in the business community they have it pigeon holed. There are a few areas which everybody recognizes where one has to use large-scale computing. These include structural analysis, aerodynamic simulation, seismic processing in the oil industry, and circuit optimization. If the application is not on the list, then the assumption is that large-scale computing is not involved. That is nonsense. The problem is there is a lack of people who can identify new areas where large-scale scientific computing should be involved.

It is not only a question of training people. This is graduate training, advanced training. (An undergraduate is probably not going to be very helpful in this kind of problem.) It is also a question of consultants because a consultant could often help industry to figure things out. Unfortunately there are very few people in the universities who are qualified to be

consultants when it comes to large-scale scientific computing, because there is so little experience in the universities with large-scale scientific computers. I must warn you specifically that it will not do to go out and hire somebody with a Nobel Prize and expect they will be able to help you on that problem. Most of the Nobel Prize winners cannot help. You can tell it if you start talking to them about large-scale scientific computing and their eyes glaze over and they sort of look sour. That's not a person to hire. There has to be a revolution in the universities so that we start training not only the students but the faculty in the realities of scientific computing.

There are also other roles that the universities play. Up until now they have received very little credit for these roles. Unfortunately when they don't play these roles the market suffers and it suffers very severely. One important role is the testing and demonstration of new large-scale computers as they come off the assembly line. Universities are better suited than industry to take delivery of model No. 1 or model No. 2 off the assembly line than industry. This is because universities can live with the prototype and lack of software and all the other minor but exceedingly annoying problems that these computers have in their early stages. The way it's handled is that professors decide what is going to be done and students have to do the work.

Now one of the secret but very powerful resources that universities have is their undergraduate population. There is a fraction of those undergraduates who would just love to play with computers. They will sit for hours waiting for something to come back. It doesn't bother them; they don't cost very much; and it

doesn't hurt their morale to sit and wait. They can doodle; it's all a game for them. It's programmers in industry paid $30-40 thousand dollars a year and always under the pressure of deadlines that get disgusted when the computer doesn't work. Of course their managers get disgusted too. What that means is we don't want to think in terms of taking all those Cray 1's which are now obsolete, dumping them into the universities, and considering that a favor to the universities. Rather the newest supercomputers just coming off the assembly line, such as the Cray XMP, or Burton Smith's Denelcor HEP, should be put in the universities. Then the universities will start making use of them early. They will also start the training process on their students so that they will learn what those computers are good for.

I learned about all this because five years ago we bought an array processor at Cornell. We bought it simply because it was a nice, very cost-effective device for research. The first hint that made it clear that this was a different kind of interaction than we had seen before with industry was when potential commercial customers started calling us up and asking us questions like, "We heard you just got the Floating Point Systems' array processor, what do you want it for? Does it work? Is the software any good? What applications do you have in mind for it?" All of these are very practical questions that someone wants the answers to before they go out and buy a computer. In other words, we just acted as an information exchange. And of course they also liked to know what is the competition, are the competitors products any better, or any worse, and why? Because

they are open, the universities are a good resource for this. They are open for someone to call up and ask us questions. Nothing we do is secret and also we talk to all the other universities. For example, if we had the Floating Point Systems device and somebody else has the CSPI device which is Floating Point's competitor, we're likely to know what's happening with the CSPI device. That's just because universities are open and God help us if anybody makes this secret.

The universities are the best place for conceptual software development. I am not talking about software of industrial quality, but demonstrating what kind of software is possible. The classic example is the UCSD PASCAL system which was built at the University of California, San Diego, largely with the help of undergraduates. Some of these people are now fantastic computer scientists and programmers. This system became one of the main operating systems for microcomputers. It was of course developed a lot further when it went to industry, but it started at UCSD. One of the reasons to turn to the universities for this conceptual software development is nobody else has any time to do it. Computing manufacturers are stretched out getting their FORTRAN compilers and operating systems not only out but fixing and fixing and fixing them as people find troubles with them. The industrial users of computing are stretched out just getting their application programs done, so there is nobody left to think about very advanced software development projects. I will come back to that point later.

Finally there is the subject the universities are all about: basic scientific research. Now we don't have to do research in the physics department on Newton's laws, we already know Newton's law. However there are areas of extreme importance to industry which are also very fundamental subjects for basic research. One example is turbulence, turbulent flow. It's not possible to do all the aerodynamic simulations that one would like to do. As soon as the flow becomes turbulent instead of having a nice reliable accurate simulation, one has to go to phenomenological shortcuts which may or may not work. These shortcuts certainly can't always be relied on. Another area is microscopic physics which is just littered with subjects of basic research; problems with phase changes, problems of properties at the molecular level, the subject of quantum chemistry, all kinds of problems in crack propagation in materials and areas like that.

These are the subjects of basic research. The importance of the computer for these subjects is that it makes possible basic research on problems of more realistic interest to industry. In a fluid flow problem one no longer has to consider only the perfect sphere, which is usually not the problem that is faced in industry. Unfortunately as soon as one starts using the computer in these areas, one finds that enormous computing power is needed to accomplish anything. Indeed the closer one gets to nearly basic research, the more the computing power is required to be able to do things that cannot be done in the traditional history of analytic science.

Let me just give an example of why computing power is important. NASA uses reasonably heavy computing power to track its spacecraft. They can do that and get pinpoint accuracy as to where that spacecraft is going to go and that is important. But if someone wants to do weather predictions, simply a problem in gas flow, one finds that weather is not a homogeneous problem. It is not a shell which rotates around the earth. Little regions of air have to be looked at. It is necessary to see what each little region is doing. First one takes a region that is ten thousand miles wide and marks it off into little regions of 100 miles across so that there are 100 segments of 100 miles each to make 10,000 miles East-West. There are 100 segments of 100 miles each to make 10,000 miles North-South. Perhaps there are 100 layers going up through the atmosphere to cover the different wind velocities at different altitudes. Altogether that's 1 million cubic segments of air that have to be tracked. It's like having to track 1 million spacecraft. That factor of 1 million increase basically represents the total gain in computing capabilities between 1950 and today. Unfortunately the problems in modern physics are much worse than that. We would like another factor 1 million in computing capabilities, and we are starting to figure out how to get it.

What are the barriers to progress in the large-scale scientific computing area? First of all, it's important to know that presently in the academic community large-scale scientific computing has the lowest priority. It is a lower priority than equipment for experimental research, it is a lower priority than funds for more graduate students, more post docs and more junior

faculty. Because of this low priority, it has received very little attention from the federal agencies and the universities themselves. In fact there are very few universities which are presently sufficiently organized to take advantage of large-scale computing even if they were given the money to get it. This is because one soon runs up against this barrier of priority.

The reason for that low priority is the experience that was obtained in the 1960s with the computers that were available in the universities. In the 1960s typically a computer mainframe served the entire university and an individual researcher could afford to use only a few seconds of the time on that mainframe. It was not possible to do large-scale scientific computing with one second of time on those mainframes. These were the mainframes of the 1960s, not the mainframes of today. What happened instead was the faculty and university saw their graduate students getting interested in computing but just spending enormous amount of times struggling with all the difficulties of using those computers and not accomplishing anything. That message has sunk deeply into the minds of the faculty. It is very difficult to get that experience out of their minds again. Those feelings are combined, of course, with the problem that the entire society has a reaction towards computers and the changes they bring which is as visible in the universities as it is anywhere else. Of course it is the universities' problem to get that priority raised. However, any help from outside would be useful.

There are some more specific barriers. First of all, there is a communications barrier. In order to make progress in scientific computing, which does not make science any easier but just makes it possible to do more difficult science, it is necessary to have communications. These are communications between different disciplines, because the scientific computing problems are basically the same across all disciplines. These are also communications between different universities, communications with the computer industry. These communications have to be electronic. It must be possible to be able to exchange software and to have access to computers which are elsewhere. In addition, the universities have to be able to take advantage of all the benefits to ordinary communication that computer networks provide.

One of the most important features of these computer networks to exploit is to start networking universities with the research groups in industry. The reason for that is that when industry has a problem that might involve computing, it needs to find exactly the right person to help with that problem. There may be only two or three in the whole country who can help with a specific problem. It is necessary to find those two or three people. If someone else is found they will say that is not their subject.

Now what can be done on a computer network is to have bulletin boards categorized by subject. A request for help is put on that bulletin board. In the network culture what happens is that the experts scan the relevant bulletin board for notes that they can reply to because if they are not replying to those

requests for help they will not be known as experts. Unfortunately, most people don't get the experience and training to be part of the networking culture. The main source of training has been the ARPA NET run by the Department of Defense. They have been very restrictive as to who could be on that network. The people who don't have that experience don't get into that culture and don't respond to network requests. In fact even in the computer science community the older computer scientists who are not trained to use the network culture, don't participate in it. So the networks have to be established and people have to be trained to use them.

One of the most important aids to technology transfer I know of is to get that network culture established between university and industry. That is, of course, not just computing itself, it is technology transfer.

The second barrier is the lack of training programs in universities involving computers. Overcoming this barrier means two things: first of all there needs to be massive use of computers in universities with everybody participating. This is not happening today. Second, in the case of large-scale scientific computing, there needs to be training of students in the management of large-scale software, that is training in computer science. It is not easy training to establish, especially when the programs for graduate students are already full and take too long. Nevertheless something has to be done about it.

Finally, of the barriers to progress, there is one that everybody recognizes as possibly the most critical. That is the software barrier. What I see in industry over and over again is that there is a mainframe bursting at the gills giving poor service and poor turnaround. It is running programs that are ten to twenty years old that have been developed and tinkered with and are now well-established. People have confidence in those programs and they are running them to death. Beside that mainframe is a supercomputer that is sitting idle for two reasons. First of all the programs running on the mainframe won't transfer to the supercomputer, because they are so targeted at the old mainframe that it is impossible to move them. Second they have got lots of new applications which should be running on the supercomputer and driving it to the wall, but their software isn't finished yet. That in a nutshell is the software problem.

The software problem in the scientific case comes down to the Fortran language, the programming language which is used to write that software. In Fortran it takes forever to get the programs up and running. The major reason for that is the nature of the language in which the software is written. The main problem with Fortran, for those of you who have some experience with it, is that when one tries to write up a scientific application and reduce it to Fortran to run in the computer, what's involved is a total scrambling of the lines of thought that go into that program. For example, a computer simulation is usually based on a set of scientific equations. However if one takes a Fortran program and tries to figure out what the equation is that it was based on one finds that the equation has been

ripped into hundreds of little pieces. It has been contorted incredibly to take into account the approximation methods that are used to put that equation on the computer. The equations are now scattered through a 60 page Fortran listing just like pieces of dust scattered around a room. Of course part of what takes so long is first doing that shredding process. But the most time-consuming part is understanding what that program is doing after that shredding process is complete. You can watch programmers leafing back and forth through those 60 pages of listings to try to find out what's going on. That puts a factor of 10-100 in the time scale to get those programs done and working.

As I said the only place where one can really hope to get this problem solved is in the universities, for nobody else has the time to work on the problems at that level. The question is how should scientific programming be done. At Cornell I have a project that is joint between myself and the Computer Science Department working precisely on the problem of a language for writing scientic programs. In this language, the equations would be in one place where they could be read. Once they had been read the programmer would go on to the other parts of the problem such as the numerical methods, the data structures, and the optimization procedures needed to make the program run really fast. This is especially important for these new architechtures on the supercomputer or the parallel processors. At Cornell we call this new approach the Gibbs project.

Let me conclude by pointing out what's happening today in the universities to deal with this whole question of large-scale scientific computing. The most important goal is to get people collaborating within the universities and the universities collaborating with industry, government and wherever else one has to deal with this problem. The collaboration must be across scientific disciplines; it must be between the scientist and the computer scientist. That is especially important. It must involve collaboration with the manufacturers of computers. This is beginning to develop. It also involves collaboration with the industrial users of computing.

In our dealings with Floating Point Systems we have tried to put ourselves in a situation where we act as a buffer between the commercial users of array processors and the manufacturers. For example, the manager of the project at Cornell is presently the President of the Users Group for Floating Point Systems. That's an ideal way of establishing ourselves as providing that buffer. We need to be in collaboration with the users of Floating Point Systems so that when we complain to Floating Point Systems about the way their products don't work, they listen to us because they know that we will just go to the commercial customers and get some support for everything we have to say. Of course their designers don't like to hear that things that they did are wrong so we have to have some pressure. You need collaboration with industrial users so that they get the benefits of what we learn about the products. Of course at the present time, since our primary need is money, we come out hat in hand for a little help for the services that we provide.

A number of us have been pressing for action at the government level. I was a member of the Lax panel last summer which reported the needs for large-scale computing to the NSF, DOD, and other government agencies across all sciences. The one sentence in that panel report which says that we were in danger of falling behind the Japanese in the large-scale computing area has received a lot of attention in the press. In fact there has been a major turn-around at high government levels between last July when the panel met and today. In the last two weeks there was an announcement from the White House that there would be a major effort within the Government to look at the problems of supercomputer needs and access of university scientists to supercomputers. I hope that that announcement really does get followed up. I would be grateful for help from anybody who can help maintain the pressure on the Congress and the White House and governmental agencies to move and to give higher priority to this whole large-scale scientific area than it has had in the past.

There are a number of hardware and software products in the universities. Some of them are discussed in more detail by the round table participants. These include hardware projects at places like MIT, Cal Tech, Columbia, and NYU. There are also software projects at many of the major universities. One other project I would mention from Cornell is that I am presently organizing all of the theoretical science at Cornell into one umbrella organization called the Theory Center to promote the collaboration across disciplines and to attack our computing support needs in common.

QUESTIONS

Question:

In some sense the world has moved in a direction of distributed processing rather than the supercomputers in part to tackle the multiproblem situation where a common data base is needed. Isn't that a more intelligent way of approaching the large volume of difficult problems rather than the alternative of concentrating on parallel processing?

Wilson:

What you hear at all computer conferences is about the micros that are being distributed in personal computers by the millions rather than centralized super main frames with lots of computing power in one place. Now, in fact, for large-scale scientific problems we need both. Just to give an example of the way we plan to proceed at Cornell is to start with a distributed network of super minicomputers. There will be one for each group of theorists. This might be a group of three or four faculty members and a number of students. All of these super minis will be networked together, with a typical local area network. We will hang array processors on that network or attach processors as though they were like the FPS 164 which is a general purpose number crunching engine at a reasonable cost. This costs about $500K for a nicely loaded system. We will simply add more and more FPS 164's as this demand develops. If we need 16 of them, we will have 16, if we need 32, we will have 32, so that nobody suffers from delays in turnaround. We do not want the productivity of our 500 theorists to go down just because the computer system is filled up.

We will then provide access to all kinds of very high performance computers, supercomputers of various kinds, but with the idea that those are for single jobs that have outrun the capability of our distributed network. These are for the jobs that must be like the tracking of a million space craft, which we cannot do on the 164. Maybe we can only track 100,000 on the 164. We will make our users compete for access for longish periods (weeks at a time) on whatever supercomputers we can get a hold of. We will not distribute that time democratically. Instead we will find the best of the projects that need extrodinary amounts of computing time and we will let them take over the supercomputer all to themselves like a personal computing system. And of course, when the supercomputers become cheap enough, we will just have enough of them for everybody.

Question:

As we go to larger computers, would it be possible for these computers to interact among each other in such a way that we don't have to go in there and make the necessary corrections? Is the time going to come when one of the systems is larger than one human being can possibly handle?

Wilson:

The important trend in artificial intelligence is the trend to provide assistance to users who are in charge of the programming process. There's a project which I like very much at MIT called the Programmer's Apprentice. There they are not trying to do artificial intelligence in the sense that you tell the computer, here's my problem and an hour later $e=mc^2$ appears on the screen as the answer to your question. That's all a pipe

dream. What can be done and I think the artificial intelligence community is doing a lot of good work in this area is to make it easier for the human programmer. This can be done by giving him assistance in places where one can't figure out how to give assistance so that the programmer is not working almost at the machine language level when he is writing programs. And, of course, at a higher level what is happening is that people are now packaging programs. For example, the structural anaylsis programs are very heavily packaged. One goes to Swanson Associates or to Nastran where the program is already packaged and learns how to run the package. It isn't necessary to read the 300,000 lines of Fortran that lies in Nastran; nobody could do that. As we learn how to do work at the micro structure level, there are again going to be packaged programs. There will be companies that will swarm just to do that packaging and make them available so that the industrial users will turn to these packages rather than having to do a totally do-it-yourself operation.

Question:

What is your view on the question of artificial intelligence?

Wilson:

Well, this is a country of 200 million people. First of all even if one decides that artificial intelligence has a higher priority, and one neglects the large-scale scientific computing especially the Japanese national project addressed specifically to large scale, super-speed scientific computing, it is still extremely dangerous to give the large-scale scientific computing

such a low priority that nothing gets done. As I say we have 200 million people and they don't all have to be doing artificial intelligence. The other thing that I would say is there is a lot of promise and not much performance in the artificial intelligence area. There are particular kinds of artificial intelligence which are extremely important. In our own Gibbs project (the software productivity project) we interact mostly with members of the artificial intelligence community because that is where a lot of the ideas we are going to use come from. On the other hand, it is clear there has been a big oversell in the last couple of years about artificial intelligence. You should be wary of this.

Question:

Would you care to comment on the impact of the MCC consortium?

Wilson

I believe the MCC is potentially extremely important. At the present time, I feel that it has far too little funding and far too little force behind it to have that much interaction. The critical questions are how it will grow, how fast it will grow, and how adventurous it will be in carrying out it's mandate. If it sticks to the statements as presently made, that it's just doing proprietary advanced research, then it won't really break out of that mold, especially in its interaction with the universities. Then it would be a rather minor force. If it does become aggressive in its relations with the universities and starts addressing issues that are not strictly proprietary

research issues but some of the issues that we have been dealing with in the universities on a global scale then it can be a powerful force on the scene. At the same time it must build support so it isn't a $50 million operation but a $500 million operation. I wish Bobby Inman the very best of luck in steering that corporation in the right direction and I hope he succeeds because it is important.

Question:

What do you see as the impact of IBM not being involved in MCC?

Wilson:

Let me describe my strategy with respect to IBM. I apologize to the IBM representative here. First of all, I myself can obviously have no impact on IBM. I talk to people at IBM and there are people who are on my side and people who are not. The people on my side are happy to talk to me and I do whatever I can with them. On the other hand what IBM responds to is larger forces than just one person. IBM responds to the market and the needs of the market, so the really important objective is to build up the large-scale scientific market so that IBM is motivated to respond to that market. The way we build up that market is by working with faster reacting smaller businesses. This includes organizations like Floating Point Systems, Cray Research, and Denelcor. We can try to build the culture of large-scale scientific computing around these projects and thereby build up the useage to the point where IBM reacts to that market. I expect that by 1990 that market will be there. I am

sure that in 1990 IBM will make more money from the sales of supercomputers than any other manufacturer, domestic or foreign. Perhaps they will blame me for some small fraction of that profit.

Comment:

Let me just assert that you don't have to convince top IBM management that this is of essential importance, they are convinced of that. The problem is that the IBM company is very large and its filled with people that don't know what they are doing, who are trying to push wrong actions in the name of scientific computing. IBM recognizes the need to keep people trained, the issue that you stated before. IBM needs to be salted more broadly with people who are skilled in understanding scientific computing so that they make the right decisions, so that they don't go rushing simultaneously in 90 wrong directions.

Wilson:

The problems I described they face too. But what I am saying is when it becomes an important business decision they find out how to do things right. I don't believe they really do things wrong when it has to do with business data processing. They don't do things wrong in personal computing because that's a big enough market so they have got to do it right and they do it right. I think they will do it right in the scientific market when the market is large enough. I don't know how they will do it; they may do amazing things in order to do it right and they may do all the things I said that need to be done, but I am convinced that when the market is big enough IBM will address

that market and will do it in a sensible fashion. So far it is not really large enough for them to do that yet.

DR. DAVID WALLACE
University of Edinburgh

This paper covers some of our interests and experience at Edinburgh using the Distributed Array Processor (DAP). This is a machine which was built by International Computers Limited (ICL), a U.K. company. It is a very interesting device because it is truly an intrinsic parallel processor, consisting of a 64 by 64 hardwired array of elementary processors. The design of the machine was published ten years ago and the device itself has been available for about three years. Each of the processing elements is extremely primitive. However, because there are 4,096 of them, it is a rather powerful machine which approaches the performance of CRAY-1 for many of the problems of physics that we are interested in. The DAP is also very inexpensive. I think that the reason for this is twofold. First, these modular-structure machines are cheaper to design and to build so they can be produced more cheaply. Second, the major mistake that ICL made when they built the machine was to tie it into ICL mainframes in the hope that this would sell more mainframes. What happened was they didn't sell many DAP's. As a result Edinburgh got a good bargain; ICL was selling the DAP's for about a quarter of a million pounds in the end. I understand that six DAP's have now been built. I regard it as a successful first-generation machine on which we have already been able to do a lot of interesting physics and the group at Edinburgh is enthusiastically committed to this type of machine for our future computing requirements.

Four topics are covered briefly here: 1) what the machine is; 2) how we got it (involving the national funding bodies and links with industry in the UK); 3) a little bit about how it is used and, 4) a few remarks about prospects for this kind of architecture.

Figure 1 is a schematic description of the machine. It is a 64 x 64 array of processing elements, PE's, each connected to its nearest neighbor on the square, with periodic boundary conditions if required. Each PE is very simple; arithmetic operations are done by sequential single-bit manipulations. Switches control the transfer of data between neighbors on the array. Each PE has 4K of RAM so that in total there are 2M bytes of central memory. There is a master control unit broadcasting through the machine which controls the whole system so this is a single instruction but multiple data (SIMD) device; all the processing elements are doing the same things at the same time but on different data. ICL were really rather secretive about what the master control unit looks like, but for the user everything is very explicit and straightforward, as I shall indicate later.

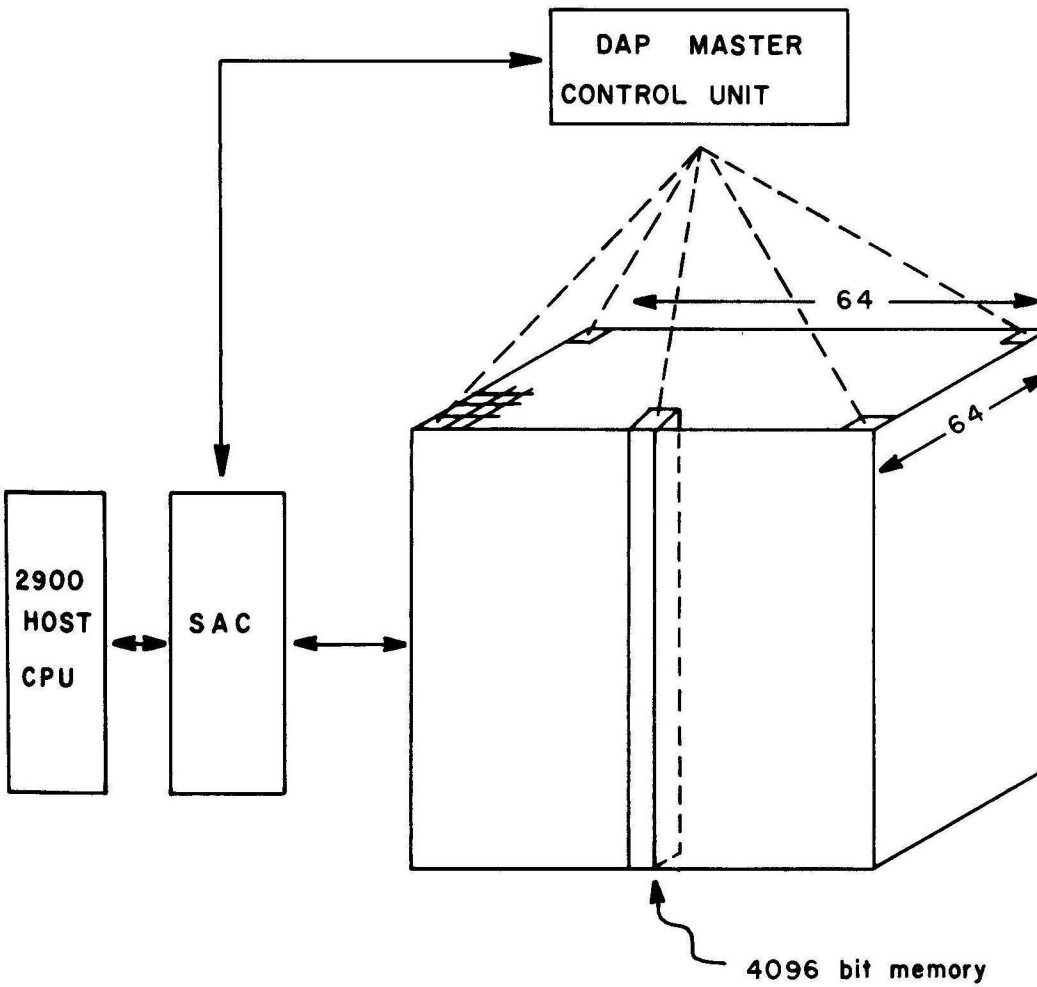


Fig. 1 DAP. Schematice architecture.

To understand how we got the machine it is necessary to understand the funding within the UK. By the standards of funding within the US, the UK has actually followed quite a sensible policy in recent years and the scale of the funding for a UK effort is not bad. The Science and Engineering Research Council (SERC) had a policy that it would fund central facilities which are what they describe as state-of-the-art computers, whatever that actually means. Three or four years ago this was easy to decide and they bought time on a CRAY which was installed near Manchester. This meant there was a general CRAY facility available to users in the scientific university community in Britain - not very much of it, but it was generally available. The SERC also set up a DAP unit at Queen Mary College in London. This arrangement was quite interesting in that the head of the DAP support unit also had an appointment with ICL so there was a clear link there on installing a machine at an early stage in a University in the expectation that fruitful developments would be made. Of course, this didn't quite meet Ken Wilson's criteria that prototypes should go free to universities because SERC had to pay for it. More recently the CRAY machine has been re-installed in the University of London Computing Centre for general southern region users and there will be a CYBER available in Manchester shortly. On top of that we have been able to acquire our DAP. We needed to find 270,000 pounds sterling or roughly $400,000 for it. After failing first in our efforts to set up a national Scottish facility we spent Christmas and New Years preparing an application to SERC in three blocks of 90,000

pounds sterling for work at Edinburgh in astronomy, solid-state physics, and elementary particle physics. This application went to different subcommittees of SERC who decided they would or wouldn't fund it. It's a very long story. The strength of our regional computing center links with ICL at this time should be gauged by the fact that they allowed us to ship the machine in and essentially set it in concrete in the machine room before SERC had decided to support it, in fact, when two of the three subcommittees had decided not to support it. That demonstrates ICL's commitments to get a machine to us. The relationship continues and we hope that they will build on the experience gained on the DAP.

Next, a little bit about the software for the machine. The way ICL set up the software for this new kind of machine on the first attempt is impressive and fun to use. It is a development of Fortran which they call DAP Fortran. Let me mention three features: (a) There is a lot of choice in specifying variables and constants. First, it is possible to have real and integer variables of various lengths (e.g., 1,2,...8 byte integer variables). Logical variables are particularly powerful and simple to handle. Both of these features one might of course expect in a machine with bit serial arithmetic. Second, in addition to the usual scalar etc. variables of standard Fortran one can also declare vectors of length 64 and 64 x 64 arrays whose elements are distributed over the 4,096 PE's. If A, B and C are declared as such arrays then in an equation such as A=B+C, the operation of adding B to C and putting it in A is done

simultaneously on the different data in all of the 4096 PE's. The same holds for operations with the standard mathematical functions, e.g., A=SIN(B). (b) One of the most powerful facilities that ICL has built in is the ability to switch off some of the processing elements and decide not to calculate at those elements. That is done by the use of logical masks which are simply defined as logical, as shown in Fig. 2. The DAP has built-in logical functions which may look somewhat unusual but they are precisely the kind of functions that are needed for scientific computation, for example, alternating rows by one ALTR(1), as shown in Figure 2. More complicated masks can be built up with simple lines of programming, for example the chequerboard defined as alternating rows logically equal to alternating columns (Fig.2). This is the kind of mask that is needed if an algorithm says that one must perform calculations only on every other processing element, for example, if all odd sites must remain passive. These are typical requirements in the kind of calculations that we do. The typical FORTRAN statement that one then uses is A(L MASK)=B and this just puts B into A everywhere that L mask is true. That is very simple software for the user. (c) A final point worth mentioning is the shift operation which transfers information between the various PE's. For example a=B+SHWC(C,3) simply takes C and puts it into a processing element three units to the left, adds it to B and puts the result into A. This is done in parallel throughout the machine. Similarly there are shifts east, north and south with cyclic or planar boundary conditions: SHWC, SHEC, SHNC, SHSC,

SHWP, SHEP, SHNP, SHSP. This is simple to implement and it is again precisely the kind of software that one has to have for the kind of calculations that we do.

LOGICAL LMASK (,) 64 x 64 1bit array

Examples :

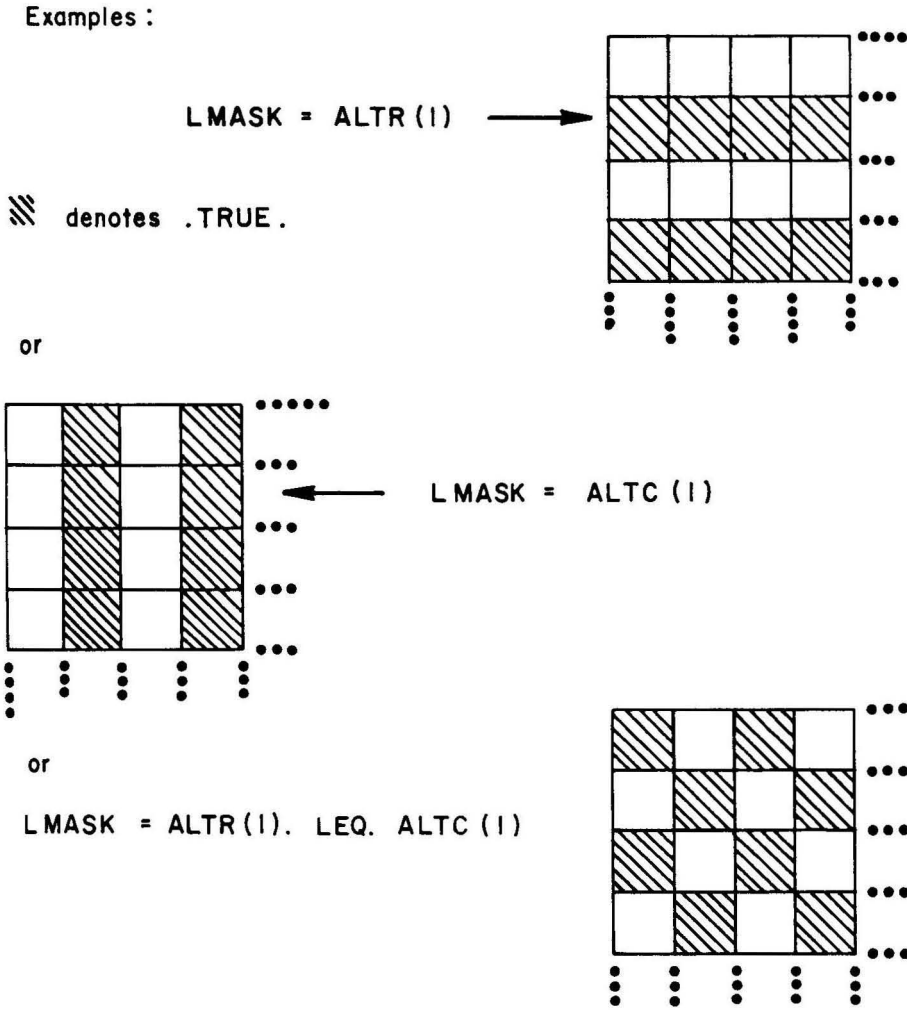


Fig. 2 Logical masks.

We use the machine for essentially the same kinds of calculations that Ken Wilson or Norman Christ would be interested in and which they will mention. These are problems where one wants to simulate a physical system. It is necessary to discretize that physical system (i.e., to approximate it by a lattice of points) in order to put it onto a computer and then one just associates so many of the lattice points of the physical system with a processing element. The updating algorithm is then begun and there is parallel updating of the simulation for all the processing elements in the computer.

Finally, what are the prospects for this kind of machine? The potential for future development is certainly very high. For example Goodyear is now building the "massively parallel processor" (MPP) for NASA and there are other developments along these lines in other companies. It seems certain that the next generation of machine will be twenty to thirty times faster and still be bit serial processing. Our general philosophy about the DAP is that it is a very good design for an engine, it is ideal for the kind of calculation that we do, and it has given us links with companies which will certainly increase.

The following is a selection of references covering general information and some specific applications developed at Edinburgh. The original DAP reference is: S.F. Reddaway, in Proc. 1st Annual Symposium on Computer Architecture (IEEE/ACM), Florida (Dec. 1973), pp. 61-65.

For further information on the DAP and its software see, for example, R.W. Hockney and C.R. Jesshope, Parallel Computers (Adam Hilger Ltd. Bristol, 1982); G.S. Pawley and G.W. Thomas, J. Comp. Phys. 47 , 165 (1982). 165.

Reviews of the Edinburgh group's work and further references are given in:

K.C. Bowler, in Proceedings of the Three Day In-depth Review on the Impact of Specialized Processors in Elementary Particle Physics, Padova, March 23-25, 1983 (University of Padua).

K.C. Bower and G.S. Pawley, to appear in Proceedings of IEEE, January 1984.

G.S. Pawley, in Proceedings of the Conference on Monte Carlo Methods and Future Computer Architecture, Brookhaven May 1983.

D.J. Wallace in Proceedings of Les Houches Workshop, March 1983, to appear in Phys. Reports.

DR. JACK SCHWARTZ
Courant Institute, New York University

The Supercomputer Situation

As the U. S. moves to meet the very ambitious supercomputer plans announced by Japan, the general level of architectural activity in the supercomputer area has been rising rapidly. Many universities have become involved; over fifty designs for parallel computers of various types have been proposed and more are coming. This creates a substantial problem of choice for the administrative agencies (principally DARPA, DOE, and NSF) that will have to set the main directions of research funding in this area.

Figure 1 gives a rough taxonomy of one major subclass of the supercomputers that have been proposed. It shows the parallel machines that are based on substantial individual processing elements where "substantial" means at least a high-performance microprocessor. These machines are to be contrasted with the other main class, shown in Fig. 2 -- machines that are composed of minimal processing elements, e.g., at an extreme, single bit processors. The first, "substantial processor" class of machines tend to use "universal" interconnections; machines of the second class tend to be more severely constrained in their choice of interconnection scheme by silicon layout considerations.

Figure 1 shows the substructure in the "substantial processor" machine subfamily.

A subfamily of these consist of packet-switching machines which use various types of optimal communication nets for coupling many microcomputers very efficiently and tightly. Among the machines of this subclass, there is a significant group of machines which are designed to be programmed in a fairly conventional "procedural" style -- one in Illinois, one at N.Y.U., one being developed commercially by Sullivan Associates, and lately one at Cal Tech having a slightly different, message-passing rather than shared memory design. Down the next branch of the taxonomic tree shown in Fig. 1, we find a class of data-flow machines distinguished by a different sort of programming paradigm; these will be discussed in more detail by Professor Arvind. Finally, the "tightly coupled" family of machines shown in Fig. 1 includes another branch on which appears the circuit switching, optimal communication net, TRAC machine developed by the University of Texas. Finally, getting further away from the ULTRA class of machine shown in the lower left hand of Fig. 1, one begins to find computing devices that from the point of view of the relatively tightly coupled "ULTRA" or "TRAC" machines are more esoteric; these use various types of supplemented nearest-network communication nets.

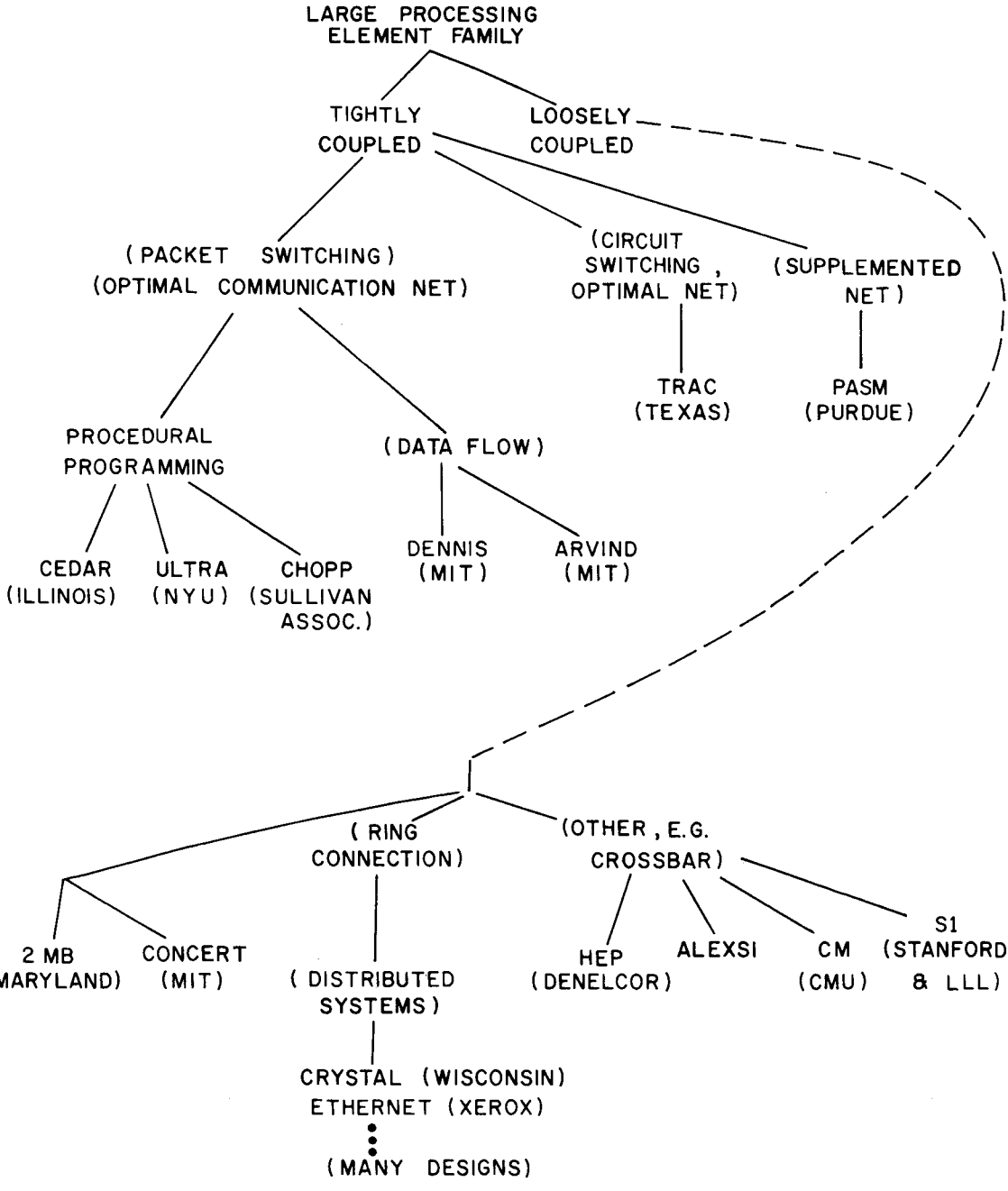


Fig. 1 Machines composed of substantial processors.

In a short talk like this I clearly cannot review too many machines in detail. By now there are at least 50, perhaps as many as 150, university supercomputer designs that have been proposed. The total number is continuing to expand rapidly as universities continue to get excited about this area.

Next I turn to the other part of our taxonomic diagram, Fig. 2, which shows machines composed of minimal processing elements.

These ultra-small-individual-processor designs tend to be constrained (though they are not invariably constrained) in their communication pattern; since designers of machines of this class are trying to optimize the use of silicon area, they ordinarily opt for simplified communication designs which lay out well in two dimensions. (However, there is a special subclass of these machines, including the so-called MIT "connection" machine, currently under active development, that use a more universal logarithmic communication network.) Typical of this class are the tree machines, which use a logarithmic but severely bandwidth-limited communication net; also the class of image processing machines exemplified by the ICL DAP. Finally, we have H. T. Kung's class of systolic array machines within which data flows through an "assembly line," with operations being done as the data moves, until finally results emerge.

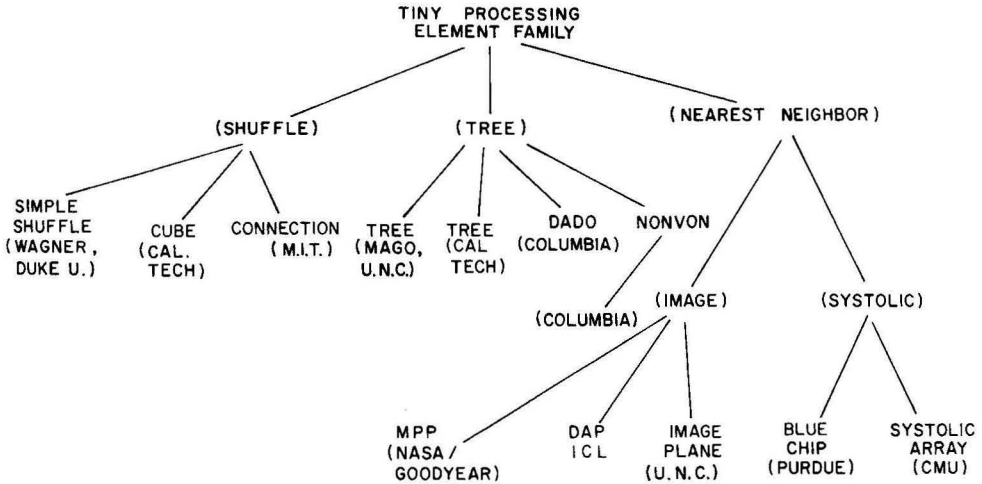


Fig. 2 Machines composed of "minimal" processing elements.

In connection with this general survey of architectural proposals, I cannot resist the temptation to say a few more words about NYU's own "ULTRACOMPUTER" proposal; this is shown in Fig. 3. The advantage claimed for this machine is a particularly "vanilla," general purpose design. A programmer would simply see it as a large collection of processors, each having a certain limited amount of private memory, but all connected to what the programmer would see (on the other side of the data communication switch shown in Fig. 3) as a giant, entirely homogenous, shared global memory. Relative to some of the more highly optimized, but also more special purpose machines that use powerful data communication schemes, the ultracomputer's reliance on shared global memory implies acceptance of a (hopefully slight) memory access inefficiency in order to increase the generality and easy use of this machine. However, this design decision does increase the weight of the hardware substantially, because of the necessity of accelerating memory communication as much as possible.

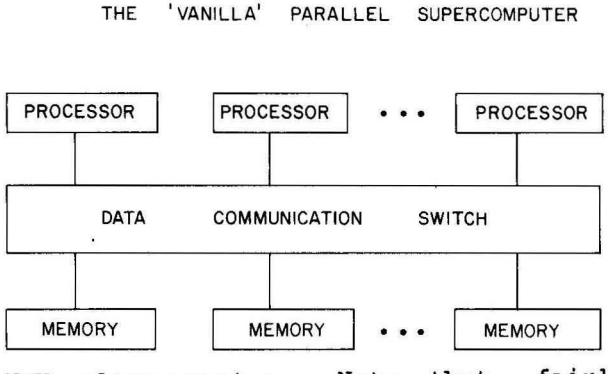


Fig. 3. NYU ultracomputer. Note that fairly substantial individual processors are used.

Additional details concerning the physical structure of the "omega network" that supports memory-to-processor communication in this machine are shown in Fig. 4.

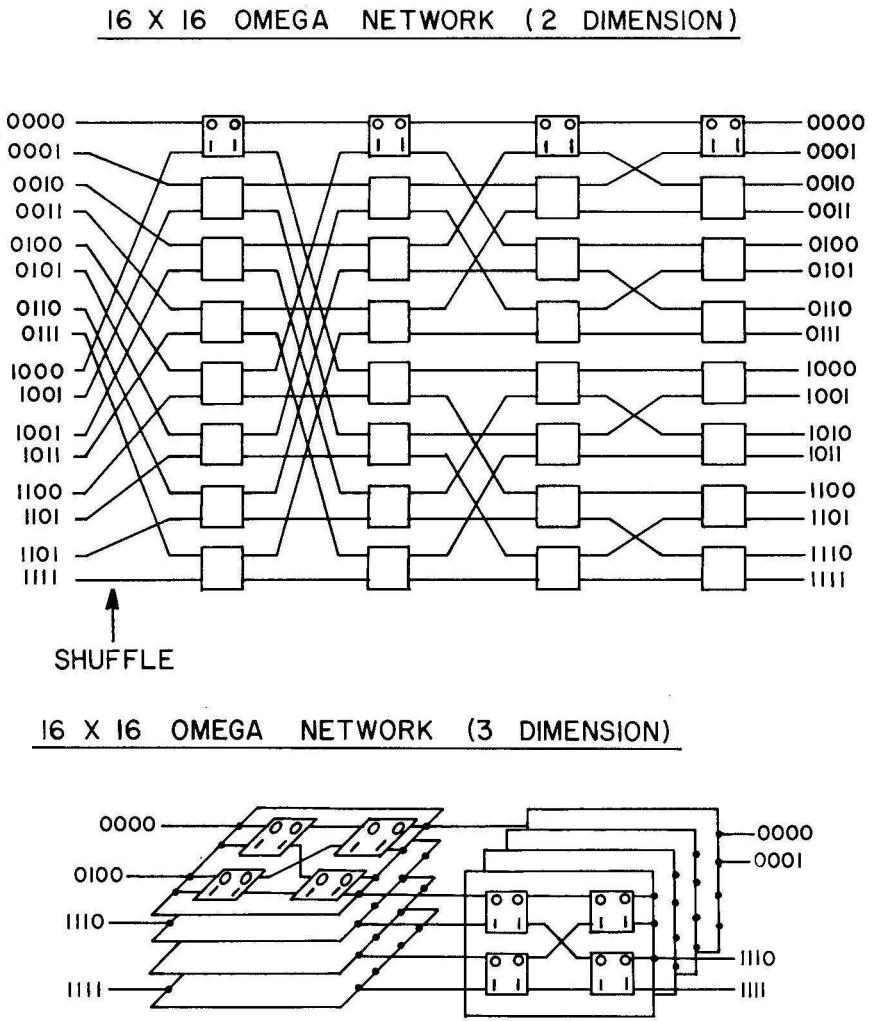


Fig. 4 Details of the physical structure of the "Omega network" that supports memory-to-memory processor communication in the NYU ultracomputer.

So much by the way of a quick survey of U. S. offerings in the supercomputer area. Next I would like to make some prognostications about the developing Japanese - U. S. competitive relationship in regard to supercomputers, which is one of the factors motivating activity within the United States now. It is easy to predict the Japanese supercomputer effort, like all their efforts, will be very well managed technically. Whatever they seek to do and are capable of defining precisely, they do very rapidly and well. On the other hand, I would say that the present conceptual basis for their fifth-generation machine is weak. Nevertheless, since Japan does represent competition that is very strong technically, the success of the developing U. S. response will depend on our ability to follow a better strategy. This will in turn depend on funding agency realism and will also require the effective involvement of industry: if only U. S. universities are involved, and a well-organized industrial participation able to move forward quickly from the university work is absent, it is easy to predict that U. S. universities will innovate very successfully, but only for the benefit of Hitachi and Fujitsu.

The administrators responsible for shaping the U. S. program-to-be in the supercomputer area therefore need to discern the strongest designs, the likeliest winners, from within a growing crowd which already includes many vocal contenders. Already something like 150 universities are each loudly proclaiming that their machine is best. How then should the funding agencies proceed? Concerning this difficult question, I

have time for only one comment. I believe it is important to avoid too heavy a concentration on artificial intelligence longshots. What one wants to do is fund a balanced set of architectural alternatives which can serve to explore the whole of the taxonomic spectrum set forth above; but one must also try to concentrate on those classes of machines most likely to be capable of serving a variety of purposes.

Next I would like to make several longer-term prognostications. I believe that the wave of design innovation represented by the best of the machines appearing in Figs. 1 and 2 will be successful, and that immense parallel machines, presently entirely hypothetical, will become everyday realities to which computing centers will become accustomed. No more than a few years hence, I expect these to be commercially available as the "Cray IV," the "IBM 5999," or what have you.

There is no secret in the construction of these parallel machines. Once one has perceived the new possibilities that large-scale parallelism opens up, the lines of design, especially of general purpose parallel machine design, are fairly obvious. I believe that the U. S. and Japanese large parallel machines will come on the market within a few years of each other. Thus the present race is for a quite temporary advantage.

Once this race has come to its natural end, i.e., once the first few of the new generation of superspeed parallel machines are around and computing centers start ordering them, the ensuing competition will take on a normal commercial character. Competition will then be a matter of quality of software

supplied, speed reached, features available, and price-performance. The crucial factor will simply be the level of corporate commitment, here in the U. S. and in Japan, to maintain a strong position in the large computer area.

A final technical comment. I believe that future large scientific applications systems will become partly hybrid. The pure "vanilla" machine appearing in Fig. 2 is a reasonable first supercomputer, but I expect that eventually one will have various types of special processors attached to this massive general purpose parallel computer base. Certainly in an environment like Fermilab, where there are many major computations that can be greatly accelerated by special- purpose devices, such an admixture of special and general purpose computing devices can have real advantage.

It is easy to surmise from what is already happening that future supercomputers will include attached image-processing machines like the Goodyear Aerospace, MPP, various signal processing devices, graphics chips, etc. Some of these attachments will have large enough markets to become regular market offerings of vendors concerned to furnish a rounded line of special-purpose devices supplementing their basic computer line.

DR. THOMAS NASH
Fermi National Accelerator Laboratory

The job of experimental high-energy physicists is twofold: for the cases where people like Norman Christ have successfully calculated predictions we have to check to see if they have done it correctly. For cases where they haven't calculated experimenters' results in advance, we provide them, in principle, with the intuition to understand how to get the right answers.

Experiments at Fermilab in the near future are somewhat typified by the apparatus shown in Fig. 1.

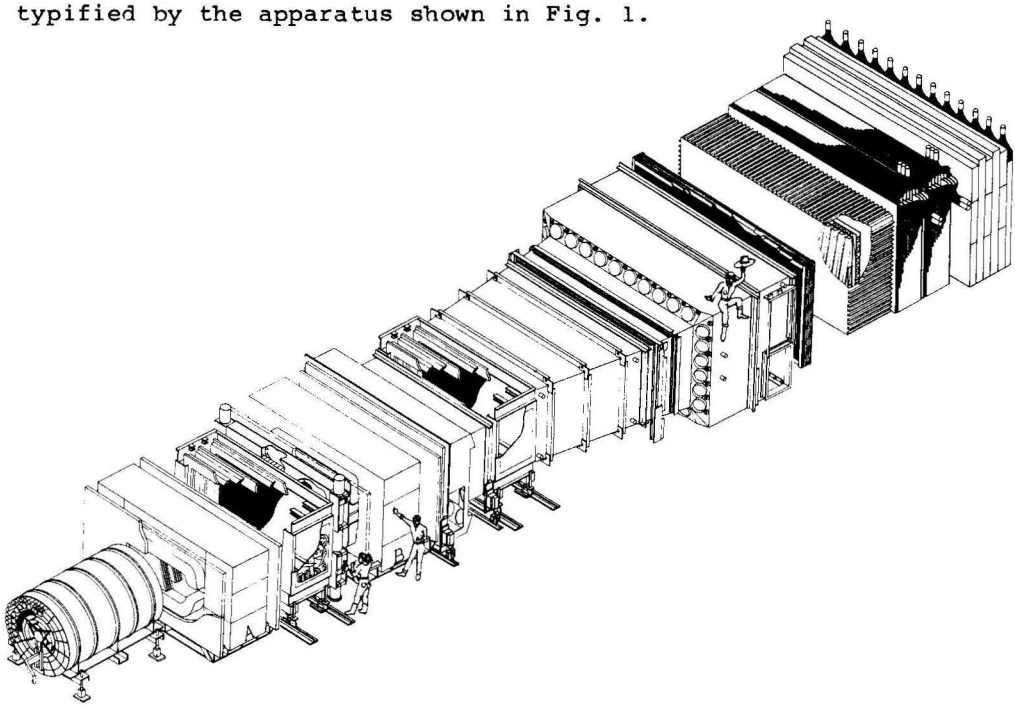


Fig. 1. Fermilab Tagged Photon Spectrometer. The apparatus is more than 25 meters long. The beam enters a target and recoil detector system on the left. The second figure is leaning on the second of two magnets. Drift chambers and an electromagnetic calorimeter are interlaced among the magnets. Large Cerenkov counters follow along with more drift chambers and calorimeters.

The scale of this experiment is indicated by the small figures. This was drawn by a Mexican artist and the sombrero is barely visible. The problem basically comes down to the following: there's a beam of particles of one kind or another that strike a target. A variety of secondary particles come off of the interaction of the beam with the target. This apparatus measures the angles and identities of all the secondary particles to study the physics of the interactions. This is done in a series of detectors that measure the point at which a particular projectile passed.

The analysis of this kind of experiment involves the reconstruction of all the data from these detectors. Just to indicate the scale of the problem for a recent experiment using this particular apparatus there were 1,000 6250 bpi tapes, containing about 25 million events with 1,500 words per event. Each event takes about a second on a Cyber 175 computer. This is pretty close to a Cyber year. This experiment is being analyzed on 20% of Fermilab's computer center, 30% of an IBM 3033, 3 VAXs, and 6 so-called 168E emulators, altogether equivalent to another 4 Cyber 175. Clearly there is a problem in getting this kind of data through.

We anticipate this problem will get worse with the Tevatron. We are trying to deal with this on two fronts, one is a 5 million dollar upgrade for the computer center. The other is the program that I'm involved with. This is the Advanced Computer R & D Program whose intention is to confront the computing-bound problems in high-energy physics by developing new approaches and

thereby generally stimulating the computing atmosphere here at Fermilab. The interaction with industry and university computer science departments is one of our important mandates. This interaction has been quite fruitful up to now, and we hope it will remain so in the future.

The first project that we are concentrating on is an event reconstruction processor that focuses on the problems that were just outlined. The idea that we're pursuing is combining the power of specialized devices with more general purpose machines. We have some experience with the special purpose processor shown in fig. 2 which was developed here. It is incredibly powerful but rather inflexible. In one example, this processor, costing about $100,000, was able to do in 7 microseconds what a million and a half dollar Cyber 175 could do in about 40 milliseconds. Thus it is possible to do a lot with such devices, but they are not easy to program. That is why it's desirable to combine that power with the programmability of microprocessors that have Fortran compilers. The intention is to stay extremely modular in order to allow optimizing architechure for different classes of problems, and thereby maximize hardware utilization. We hope this will include the possibility of array interconnections for lattice gauge problems.

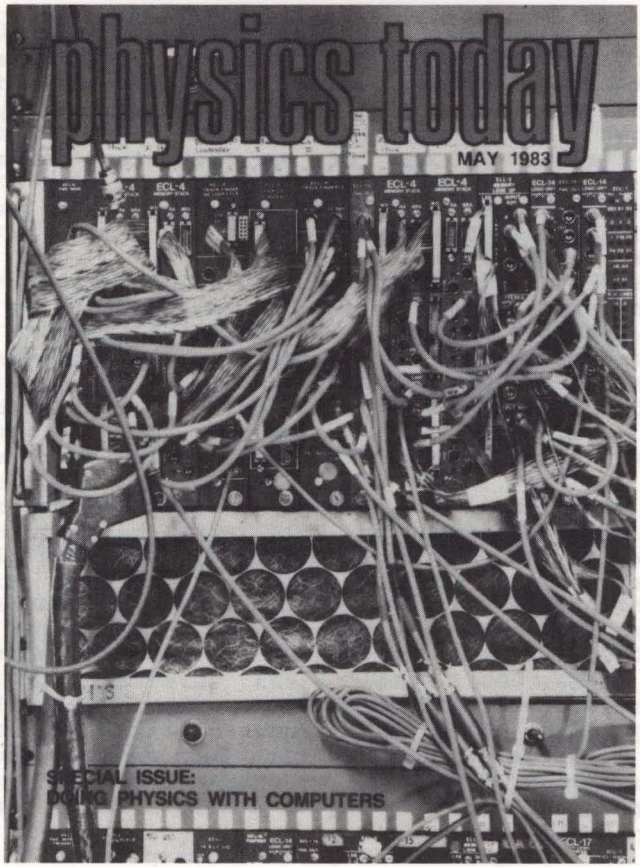


Fig. 2. Photograph of the powerful ECL Data Driven Processor developed at Fermilab. This system is configurable for different problems.

In the reconstruction problem multiprocessing is encouraged by several characteristics. The events are independent, the problem breaks easily into major vertical subroutines, within the events there is intrinsic parallelism, and most importantly there exists an instruction sequence that dominates the computing time.

What would a full-blown system naturally look like? Here one can divide the problem into a series of different subroutines each one of which takes a different amount of relative time, so that it is necessary to have the right number of processors to handle a particular level so that there aren't any traffic jams. Figure 3 illustrates the approach. The crucial idea that we're emphasizing, indicated by the circles, are the co-processors which are special purpose devices to do certain kernels of the algorithms extremely rapidly and effectively. However, as a first step in parallel with the co-processors, we are considering a simple system of microprocessors particularly appropriate to use in this kind of a system. The microprocessors must have good Fortran. We are now actively evaluating such processors and have a long list of candidates. We are discussing with various corporations the possibility of research agreements and arrangements that can help us solve our problem.

The co-processor concept is a generalization of the co-processors used as a commercially supplied adjunct to a microprocessor chip. The word is usually used in the context of the floating point co-processor. Here we mean it to be special purpose hardware to carry out at "blinding speed" the kind of algorithms that one needs, such as finding the line through 3

points in a set of wire chambers, using non Von Neumann techniques such as hit-arrays, memories, fast-cache, memory table look-ups and so forth.

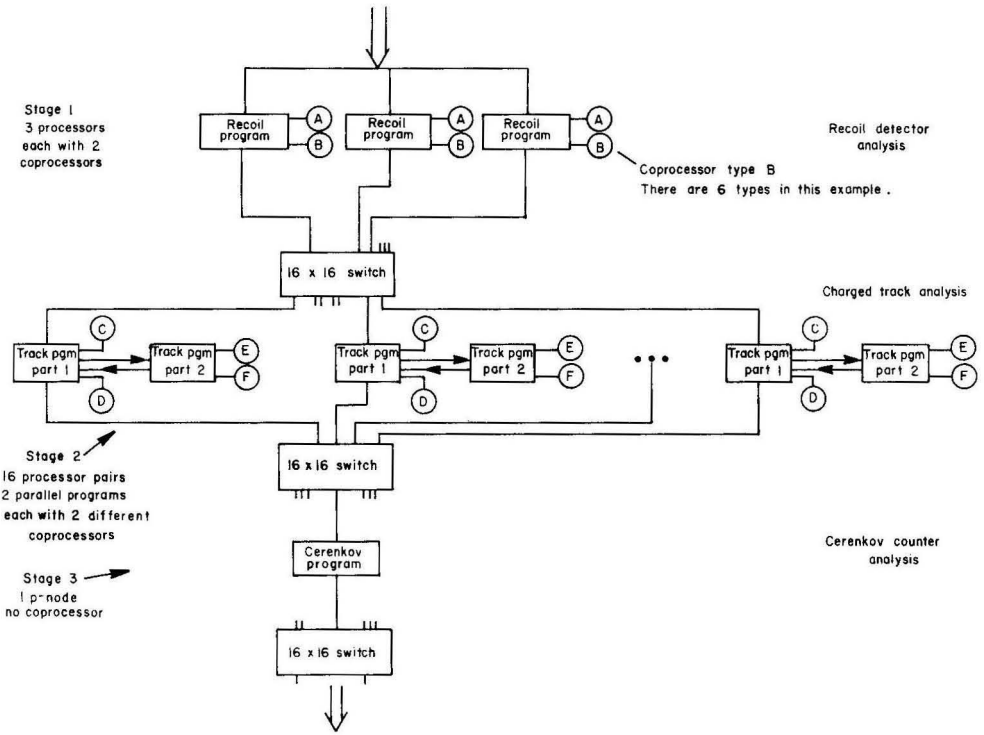
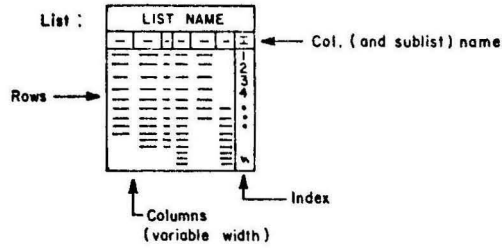


Fig. 3. Full blown co-processor system; the circles indicate co-processors.

The concept from a users standpoint is that the system looks very similar to a library. The user's program would be a series of subroutine calls to that library, well documented of course. Every time he calls a particular subroutine, the system would go into the hardware, execute the complex operation involving loops if necessary at high speed and return quickly. That kind of an approach has some broad applications. The speed up potential of co-processors (which is a critical issue), goes as $1/(1-f)$ where f is the fraction of the time spent in the co-processor algorithm when you are running without co-processors. For example, if 90% of the computing is inside the co-processor then there will be a maximum speed up of 10. On the other hand, if only one half is inside then there will be a speed up of no more than two. So the crucial issue is how much can be diverted into the co-processor in any particular problem. In order to answer that, a study has been made of the structure of our particular kind of problems. As Ken Wilson alluded to earlier, we find that they are dominated by lists and list manipulations and they turn out to be very similar operations to those used in relational data base problems which is clearly an application area far outside high energy physics. By a list we simply mean a series of columns of numbers that have identifiable attributes. In our problem what happens is that we have a series of disconnected lists that we start with as shown in Fig. 4.



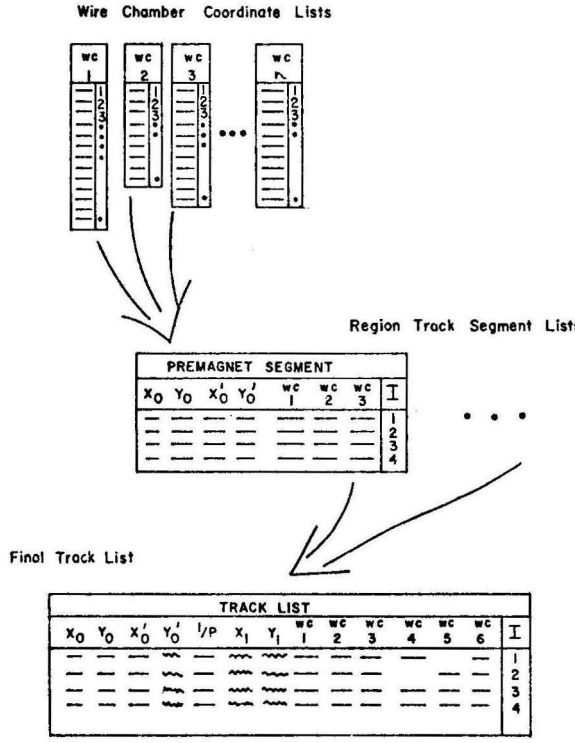


Fig. 4. List manipulation in a typical track recognition algorithm.

That is the raw data coming from the various wire chambers that have identified the particle track as it goes by. Through manipulation using what are equivalent to data base concepts these lists are related to make new lists which are the track segments in one section of the detector. Finally these segments are developed into a final track list. This is an over-simplified explanation but the operations involved are identifiable with those used elsewhere.

To summarize, at Fermilab the hardware subroutine-assisted multi microprocessor approach is natural for our three dominant computing problems which are track reconstruction, lattice gauge calculations, and beam orbit calculations. The latter are required to design the giant accelerators that are now being discussed.

In general the program is aimed at classes of computing problems which have some natural simple parallelism such as the event structure of high energy physics experimental data and that have a definable algorithm kernel that dominates the time. In addition the approach can take advantage of the structured vertical blocks in a program. This is not just for high-energy physics; it's really for problems that can run in a static configuration for days or weeks at a time, where the architechture can be reconfigured to optimize it for each problem. You can imagine that operators are not just plugging in tapes. They can also be plugging in modules for the programs working on that time scale.

There is a problem with semantics in the computing business. People think in terms of either fully general purpose computers or in terms of special purpose computers. But there is really a whole spectrum in between. I don't know what words to use and it's one that we are struggling with because sometimes semantics becomes important. The point here is that our kind of approach is not generally applicable. There are many problems for which it is totally inappropriate. But it is broadly applicable to many other problems. Perhaps the system should be called a flexible hardware assisted multiprocessor.

DR. NORMAN CHRIST
Columbia University

Many physicists view matter and in particular the strongly interacting particles as made up of quarks. The behavior of quarks is actually described by a theory that is specified in some detail called QCD (Quantum Chromo Dynamics). Figure 1 illustrates some of the strongly interacting particles such as the neutron, proton, and pi meson and contrasts them to the structureless point-like electron, muon, and photon. These strongly-interacting particles have a size and are believed to be made up of elementary, presumably point-like constituents, called quarks. The quarks interact with each other through a field, the gluon field, very much in analogy to the electromagnetic field that describes the interaction between electrons. Figure 2 illustrates the interaction between two electrons and the interaction between two quarks showing that there are many similarities. The gluon field between the two quarks is quite different from the electromagnetic field in that it obeys a non-linear equation and presumably is squeezed into a tube of flux between the two quarks so that the energy increases linearly with the separation of the two quarks. This suggests that a free quark is something that one will never see. This is quite a complicated problem. Classically one has a non-linear version of Maxwell's equations. Quantum mechanically one has a problem involving strong coupling.

Q C D

(QUANTUM CHROMO DYNAMICS)

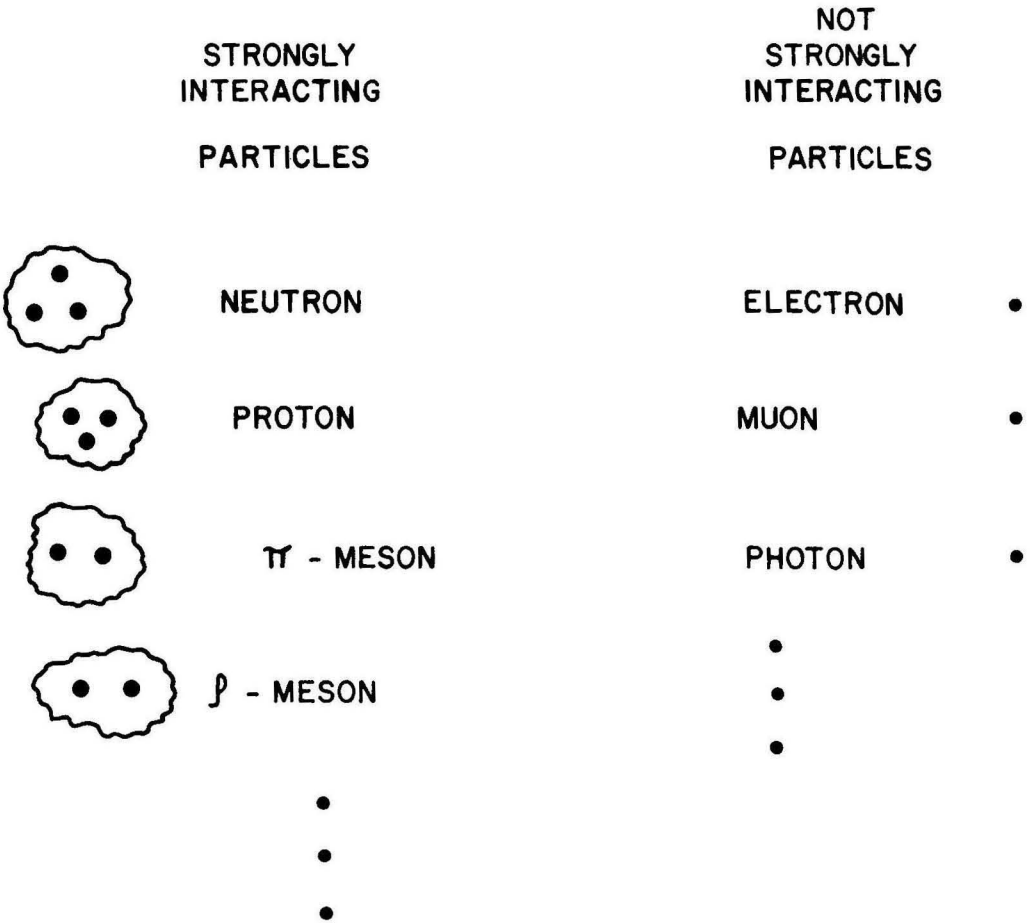


Fig. 1. Strongly interacting particle contrasted to point-like particles.

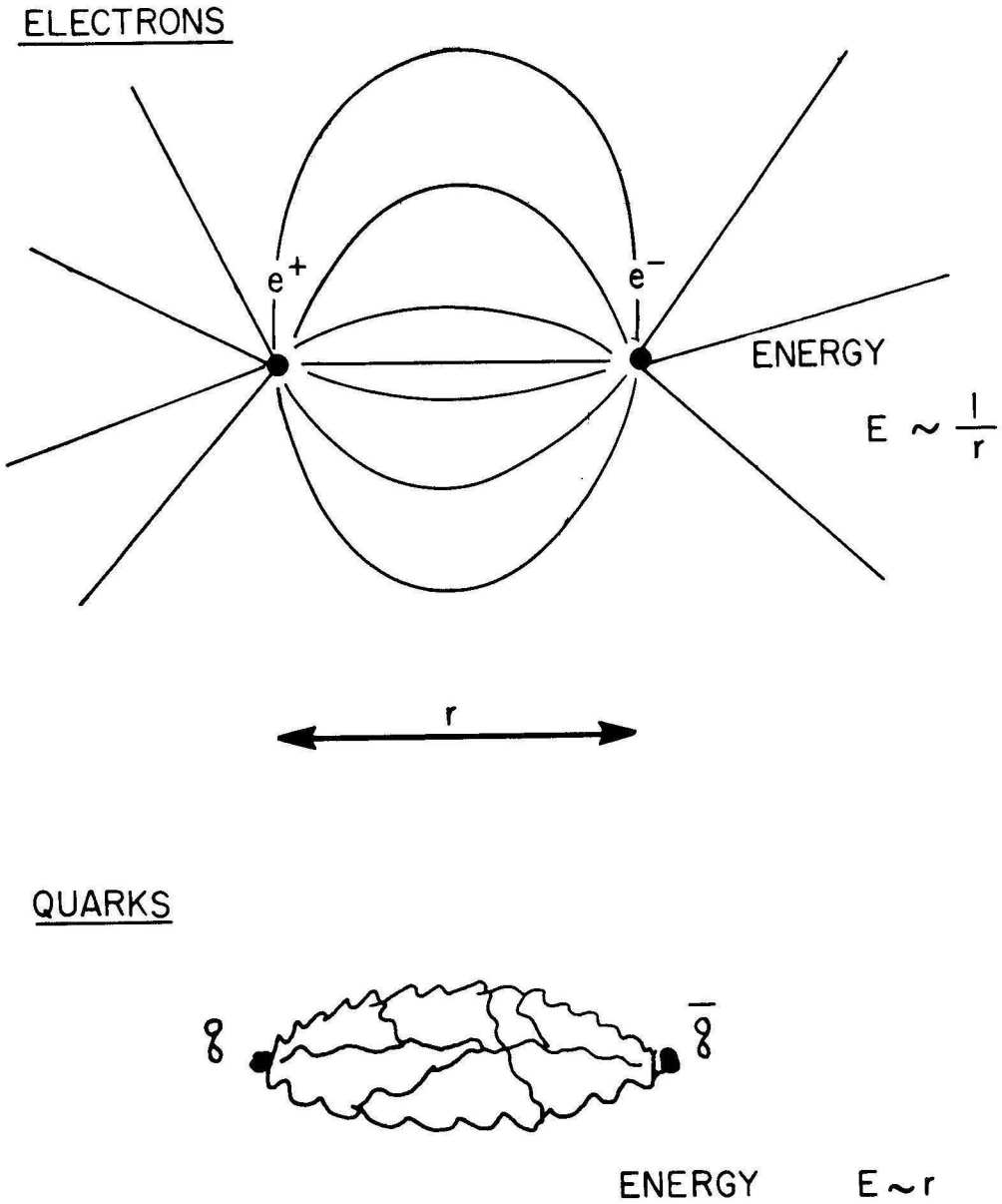


Fig. 2. Interactions between electrons compared to interactions between quarks.

Now consider a quantum mechanical state that starts off with some number of quarks and then add to it an array of anti-quarks to give a sea of various kinds of particles as shown in Fig. 3 (a). Then if the gluons are included, the system is very complicated indeed. These particles interact strongly. For example, the usual weak coupling approximation that's made to analyze the interaction of electrons and photons doesn't work. This is a problem where conventional theoretical techniques have made very little progress but in the last four years there has been significant progress using numerical methods. These methods begin by replacing the space-time continuum by a rectangular grid so one imposes on the problem a lattice structure and requires that all of these particles lie on the lattice. Finally the picture looks something like Fig. 3 (b) with the quarks on the vertices of the lattice and the gluons going on the links between them.

The quantum mechanical problem involving all of these degrees of freedom is best approached by the Feynman path integral, that is the Feynman sum over histories. The actual quantity that must be computed is the rather simply specified, but in fact quite complicated, integral shown in Fig. 4. The idea is that to each link in the lattice one associates a three by three matrix. Imagine that you want to measure a physical observable (O).

GLUONS, QUARKS AND ANTIQUARKS

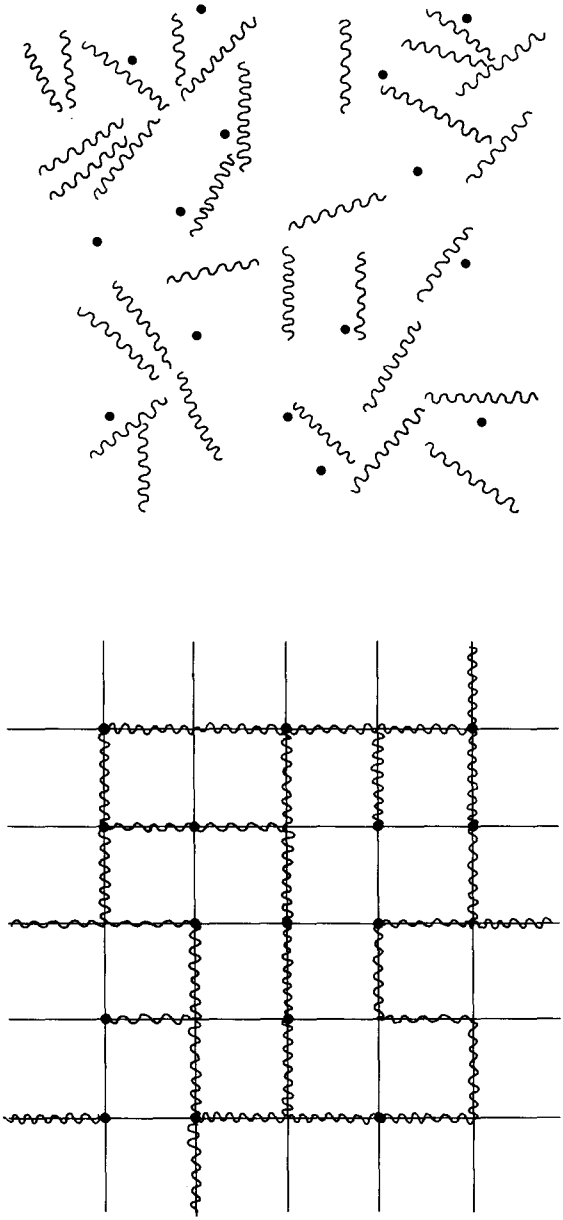
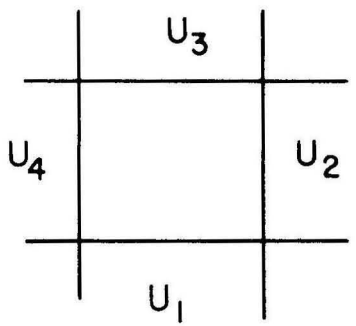


Fig. 3. Solving QCD. (a) a sea of quarks and antiquartks interacting through gluons; (b) the system is modeled by a lattice structure.

$$\langle O \rangle = \frac{\prod_{\ell} \int dU_{\ell} e^{-\beta \sum \mathrm{tr} U_{\square}} \det [\not{\partial}] O(U)}{\prod_{\ell} \int dU_{\ell} e^{-\beta \sum \mathrm{tr} U_{\square}} \det [\not{\partial}]}$$



$$U_{\square} = U_4 \; U_3 \; U_2 \; U_1$$

Fig. 4. The equation for a QCD observable.

This may be an energy or mass or correlation function. It is necessary to take that observable (O) as it depends on the degrees of freedom and integrate over all the degrees of freedom, that is all of the 3 by 3 matrices corresponding to all of the links in the lattice. This integral is weighted with the exponential of a sum of traces. Each term is a trace of a 3 by 3 matrix, one matrix for each elementary square in the lattice, where that matrix is constructed by multiplying together the four matrices corresponding to the four links that bound the square. For a big lattice there are a lot of squares and a lot of traces. The worst thing is the determinant det. Here is an operator defined on the lattice in the discrete approximation. It is also a matrix but a matrix whose number of rows and numbers of columns equals the number of vertices in the lattice. Interesting preliminary results that are not at all satisfactory have been obtained by using lattices as large as 10 by 10 by 10 by 10. This is in four dimensional space time, so that such a lattice has ten thousand sites and 40 thousand links. There are eight variables in each of these matrices, 320 thousand degrees of freedom in this integral, and finally the determinant of a matrix which is 120 thousand by 120 thousand.

The problem has now gone into a regime where the number of degrees of freedom are so large that very good use can be made of statistical techniques. The integrals here are really quite successfully treated, it appears, by using a Metropolis technique, that is a Monte Carlo algorithm of the Metropolis type. One generates samples of configurations, assignments of

matrices to links, distributed according to the product of exponential and determinant in the integrand of Fig. 4. The expectation value, that is the value of a measurable quantity O is gotten by averaging O over the ensemble. The problem is such that recent calculations have used 10 hours of Cray time and in one case 100 hours. This is just touching the surface of the problem. Because the technique is statistical, it is necessary to run the program 100 times longer to get 10 times the accuracy. In addition, it is desirable to deal with much bigger lattices.

The problem, then, requires two or three orders of magnitude increase over the amount of power that is being devoted to it today. Also this problem, the physics of strongly interacting particles, may not be the most interesting one. This is a class of phenomena that experimental physicists have studied for the past 20 or 30 years. Both the theory and the experimental results are known and here one is just making the connection. However this type of theory, these strongly coupled gauge theories, are believed to explain perhaps all of reality and there are very large areas where the theory is not yet known, where the experimental results aren't known and the calculations are much harder.

Two of us at Columbia, Tony Toronto and myself, have designed and are building a special purpose computer intended to give this needed increase in computer power. The computer takes advantage of special properties of this particular problem. The interactions, the physics of the particles on the lattice, are local so that we could easily do with the the kind of

architecture that David Wallace just described. This is a grid of processors, arranged in two dimensions with only nearest neighbors in communication. The array is homogeneous. In fact the same physics is going on at every node. At least in one mode the processors could conceivably operate in lock step with the same calculation being done at every site. The matrix multiplication, which is the big difficulty, is heavily arithmetic but it is very organized so that it can be easily pipe-lined. Finally, because the whole problem is statistical and the answers are not very precise, the method is one which doesn't require high numerical precision. So what we propose is an array of processors, perhaps in the end a 16 by 16 array, capable of doing this kind of arithmetic very fast. The structure is shown in Fig. 5. The square boxes are memories, each containing the data for those sites and links with a group of x and y coordinates but all values of z and t. The circles are processors. Each neighboring pair of memories is connected by a single processor. The design of the processors is quite straightforward. One begins with a microprocessor, the Intel 80286, that is really a quite fast and sophisticated, and also general purpose. A specially designed arithmetic unit is added to that. The memories are divided into two independent halves from which two arguments can be simultaneously fetched to perform the multiplication. The result of the previous multiplication can be accumulated with that of the previous additions and finally the result written back into one of the memories. This is all done in a pipeline fashion at 8 megahertz so 16 million

floating point operations can be performed per second. The whole process is controlled by a microprogram which can contain the instruction for doing one of these matrix multiplications.

This year we have been talking about the physics of SU3. Next year it may be E6, SU5 or whatever looks interesting. The point is that the device must be some what general. Finally, these devices have to be coupled to their neighbors. We do this in the crudest possible way. All of the operations are supposed to be synchronous. When the communication between neighbors is occurring all of the processors have to be executing the same instructions in lock-step; there is no hand-shaking between units. The multiplier can get its arguments from its local memory or from its neighbor's memories at exactly the same rate, that is at 16 megabytes per second per node per operand. The final result is a fairly inexpensive node. We have one built and two-thirds working, at a cost of $2,500 for the single node. All the nodes there are identical so it's possible to make one board and then reproduce them. We hope to hook together 256 of them to achieve 4 billion floating point operations a second.

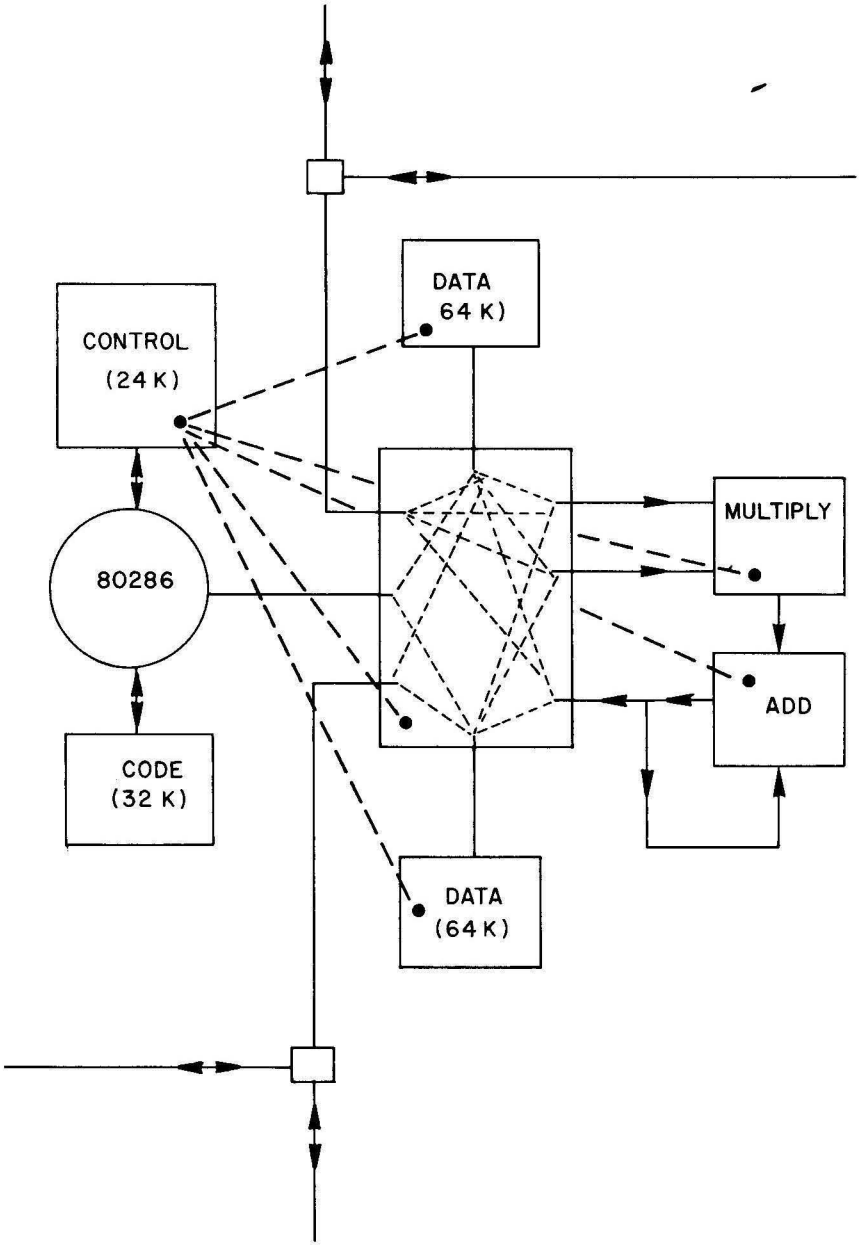


Fig. 5. The Columbia special purpose QCD computer.

DR: ARVIND
Massachusetts Institute of Technology

Among many machine projects at MIT, including the "Connection machine" Jack Schwartz alluded to, are two major data-flow projects. I am going to review one of them, The Tagged-Token Data-Flow Machine. The goal of my project, simply stated, is to design a general-purpose parallel computer in which all processors will cooperate to solve one problem. Clearly we are interested in big problems, and the question of "What is general purpose computing?" has to be understood in that context. If an application does not have any parallelism, we are no magicians and therefore we can't invent it. However, many applications have plenty of parallelism and one can build a useful machine to exploit parallelism in a large class of applications. Table I lists some characteristics of typical applications which have massive amounts of parallelism.

Table I. Parallel Applications and their Characteristics.

Number Crunching, e.g.,
- -Scientific Computing
- -High performance
- -Simple data structure - arrays

Symbol Manipulation, e.g.,
- -AI type applications-algebraic simplifier
- -Complex data structure
- -Higher order functions
- -Structure of the program itself is important

Concurrent Real time Computing, e.g.,
- -Process control - Missile defense system
- -Number of asynchronous inputs
- -Adhoc hardware structures
- -No coherent functional view

In the area of symbol manipulation also, there are lots of programs with parallelism except that these programs are not as well understood as scientific computing. One reason is that algorithms in AI programs are not so stable. AI Programs tend to be far more complex then scientific programs. I understand Professor Wilson's concern that in scientific computing the equations are spread all over the program. In AI programs, often there are no equations: the program is the sole document of the algorithms used and the programmer's intentions.

While it may be hard to substantiate, I believe that if there is a large program which runs for long periods of time then it must have parallelism. I think it is impossible to write a 100,000 line Fortran program which runs for 2 days and which is devoid of parallelism. So I am proceeding from the assumption that if you have a large program you must have parallelism, even though you may not know about it. The third class of applications (see Table I) that I am interested in, is concurrent real-time computing, that is, complex process control. In a chemical refinery, one may find 1,000 one-board computers doing calculations in various parts of the system. Generally people don't view process control systems as application programs because they don't have a good model of parallel computing.

I have to do a little bit of preaching here. First of all, Fortran as a computer language won't do for parallel computing. This is not because the scientific programs cannot be written well in Fortran. Actually Fortran is expressive enough for these applications because most of the scientific computing involves no more than simple do-loops, and arrays and matrices as data

structures. The problem is that by the time an algorithm has been coded in Fortran, lots of parallelism has been obscured. Compiler designers have to work very hard to uncover parallelism that the Fortran programmer has obscured inadvertently. The theory of compilers for parallel machines may be well understood, but such compilers face many practical problems in optimizing a large (say, 50,000 line) code because of interprocedural and global data flow analysis. We should allow the scientific programmer to express the problem in such a way that the code retains whatever parallelism there is in the first place. The issue is not whether people "think parallel" but rather if they have tools -- languages and compilers which do not make the code unnecessarily sequential.

On the hardware side, I don't believe that multi-processing based on commercial processors can work. To employ many processors on one problem requires a fundamental change in the architecture of the processor itself regardless of what is done with the switching networks and memory structures. This change is already taking place in very high performance units. For example, in the Cray-1 one finds that the concept of Program Counter (PC) is rather fuzzy. It's not as "focused" as the PC in a Motorola 68000 microprocessor where one knows precisely which instruction is being executed. Instructions can often be executed out of order to increase performance in a high-performance system. By suitable use of interlocks a machine designer can make this shuffling of instructions transparent to the user. The negative effect of large memory latency on performance can be avoided only by changing the sequential nature

of the processors. My point is that we must accept and confront the fundamental limitations of single PC based machines so that processor designs would not appear to be a collection of "hardware hacks" implemented to achieve high performance.

I am going to propose a radical solution: change languages to functional languages and the basis of the architecture underlying the hardware to data-flow. I believe change in both language and architecture is required because that's the only way to get the best performance out of machines. Fortran is ideally suited for conventional Von Neuman computers. Nobody has been able to displace Fortran because the match is so perfect. Anytime something fancy is done to Fortran it's compilation becomes inefficient. Anytime changes are made in the architecture, changes which can't be exploited by a Fortran compiler, we either pay in terms of increased programming effort or underutilized hardware. The symbiosis of language and architecture has to be maintained, and I think this will happen with functional languages and data-flow architectures.

Here is a thirty-second explanation of functional languages and data-flow (see Fig. 1). Functional languages are really much closer to the way scientists and engineers think about problems. I have a harder time with computer scientists because they already know programming. If someone dosn't know programming they are much better off starting with functional languages, because basically one has to know only primitive or base functions like plus, minus, test-for-zero, and rules for combining functions. Rules for combining functions are simple function composition, conditional composition and recursion.

Composition of functions is something that engineers and scientists understand very well. To take the trivial example shown in Fig. 1, the program, f(g(a,b),h(a)) may be written as

```
Let x = g(a,b);
    y = h (a);
in f(x,y).
```

It almost looks like an imperative program where first x is computed, then y is computed, and then x and y are substituted in f. However, note that if one thinks in terms of functions, one doesn't ask absurdly simple questions like can g and h be done in parallel. Of course they can be done in parallel since they are functions, and functions don't effect each other. The value of sin(x) does not get affected by the evaluation of cos (x)! Those are the kind of beautiful properties functional languages have. They are also easier to program in and eventually they will be more efficient to execute than imperative languages. Today, functional languages are compiled on sequential machines and the compiled code is inefficient because the underlying architecture is not well suited to the task. It should be noted that this problem is analogous to the problem of Fortran compilers which generate very inefficient code for data-flow machines.

Figure 1 shows the connection between functional languages and data-flow graphs. It is easy to view the composition of functions in terms of data-flow graphs. Each box in the graph represents a function which can be a plus, minus, fast Fourier transform or even a linear equation solver. Boxes are connected by lines which represent data-dependencies among functions. The excution of these programs can be thought of in terms of arrival of data along these lines at a box, the box being enabled and

then "firing" or executing. Finally data is produced as results and is forwarded to other boxes. The natural consequence of viewing things in this manner is that any operator that is enabled can be fired. So the default is parallelism here, the execution is constrained only by the data dependencies. Note in Fig. 1, f cannot fire until h has finished execution; however, after g has output something, it can accept the next round of data and start computing with it. So given a stream of data, g, h and f may all fire simultaneously.

Next, lets consider the possibility of queuing tokens on the arcs of a data-flow growth. Let's label each token with its destination instruction address and its position in the queue. As shown in Fig. 2, the ith token as well as the i + 1st token may be in the queue at the same time. Why am I doing all this? Because I would also like to exploit, what I call, temporal parallelism in programs. If there are enough processors and several sets of tokens on input arcs, I should be able to perform several firings of the same function simultaneously. This is the kind of parallelism my machine would exploit. The basic rule in the abstract machine is that whenever two tokens have the same label they get together, the instruction specified in the label is fetched, and the operation specified in the instruction is performed. Thus, as stated earlier, you should think of a token as carrying a name (a tag) and some data.

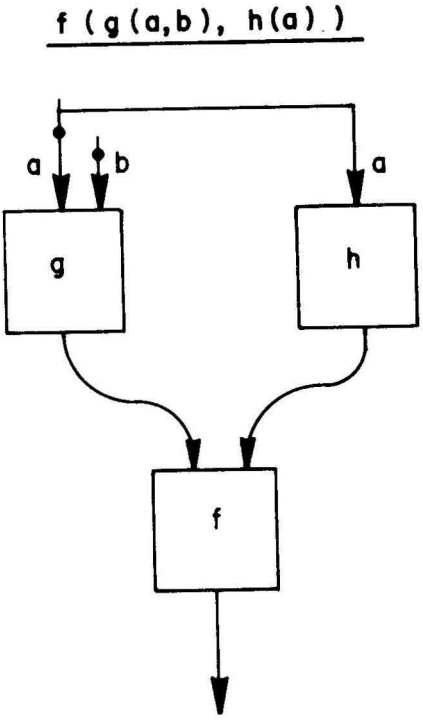


Fig. 1. Functional languages and data-flow. Here g and h can be executed in parallel; execution of f and g may also overlap.

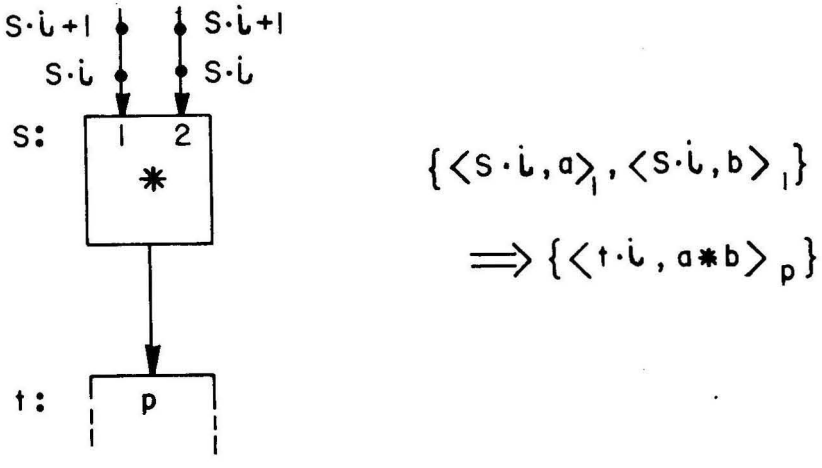


Fig. 2. The U-interpreter. A scheme for tagging tokens. Each distinct execution of an operator is given a unique (activity) name, each token carries a destination activity name.

What kind of machine will execute in this manner? Figure 3 shows an architecture consisting of N identical Processing Elements (PE's). It doesn't matter, as far as the functionality of the machine is concerned, how the processors are connected. The interconnection network may affect the performance but is not reflected in the programming model. We assume that every processing element is capable of sending tokens to any other processing element. Figure 4 shows the internal structure of a processing element and is important to understand because it's very different from a conventional Von Neumann computer. A token carrying a tag and data arrives at the processing element. The first thing the token encounters is the Waiting-Matching section which is initially empty. Remember our abstract machine has the very simple rule that when two tokens have the same label they must get together. If the token finds its partner in the Waiting-Matching Section it goes to the Instruction Fetch section, otherwise it "waits" in the Waiting-Matching Section. The Instruction-Fetch Section has a program memory associated with it.

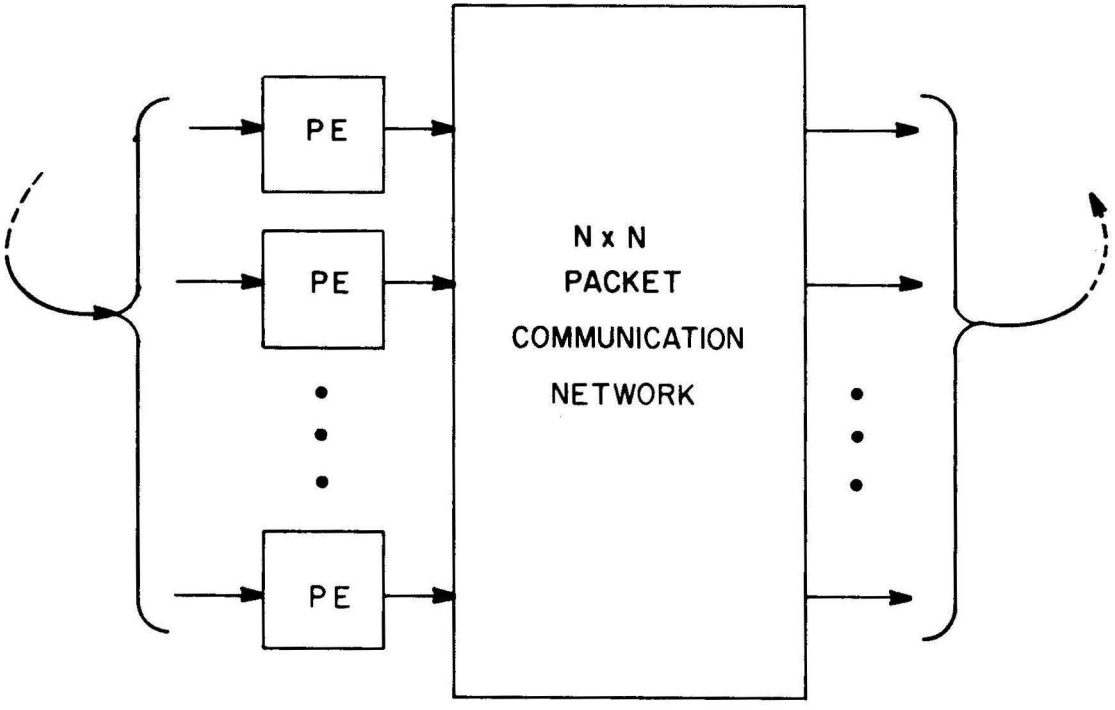


Fig. 3. An overview of the proposed architecture.

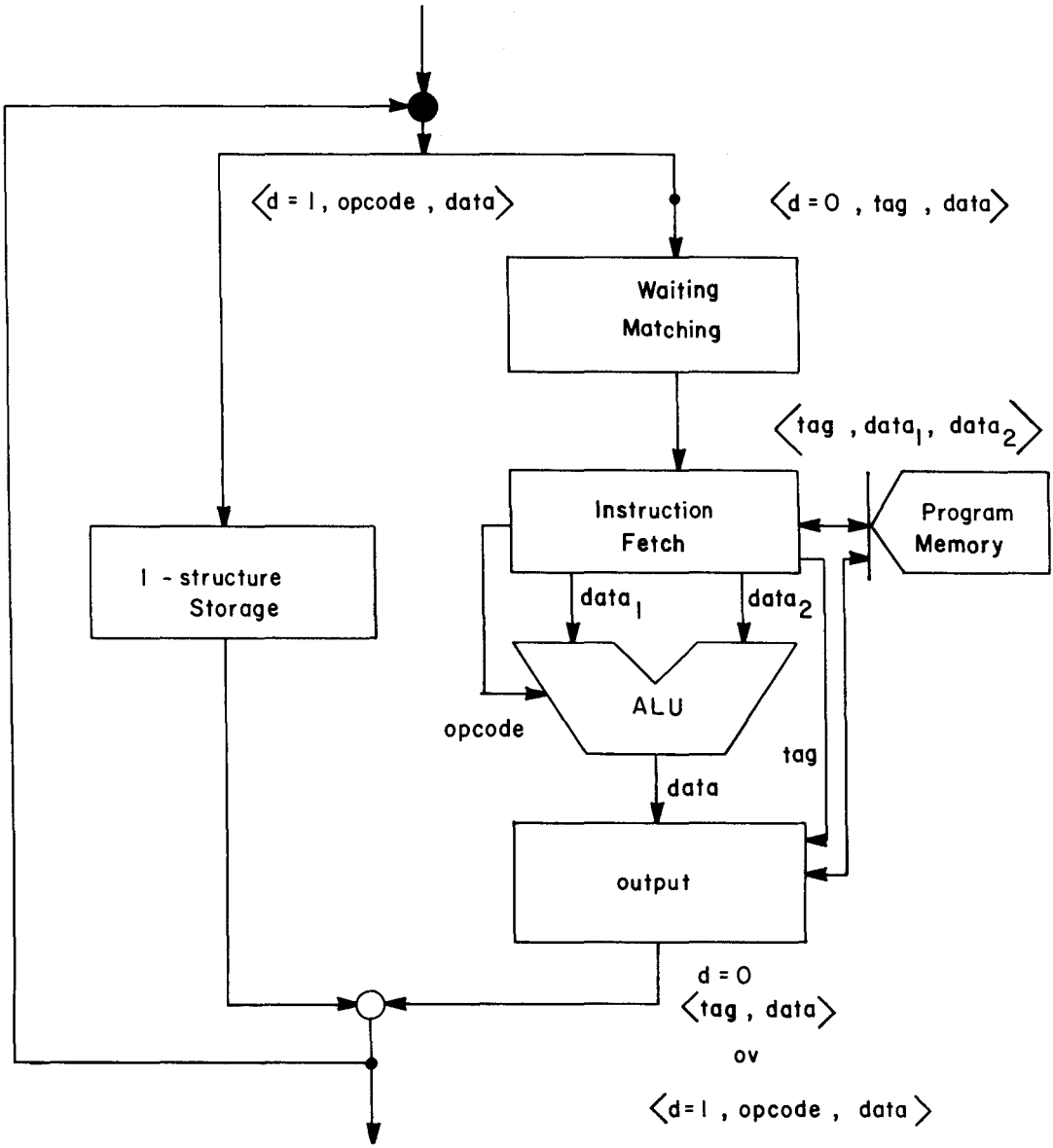


Fig. 4. One processing element.

The instruction at the address indicated by the tag in the packet is fetched. The fetched instruction says for instance, a-ha I am an addition operation. After this the operator and the operands are passed to the ALU. Notice the difference from a conventional computer where after the instruction has been fetched, the operands indicated by the instructions are fetched from the memory. Only then something is done with the operands in the ALU. In our processor, instruction fetch is done after the operands have arrived to find out what is to be done with the operands. You can have any type of Arithmetic Logic Unit here. The ALU produces data as well as tags for the data. Finally the processor outputs the results packet. This is how one Processing Element which is a complete computer in itself, works.

Now, if two such devices are available how will we make use of them? Well, a very simple strategy can be followed. One can say all the tokens with even tags remain on the left-hand processor and all the tokens which have odd tags should go to the right-hand processor. (Of course, more sophisticated schemes than this can be imagined.) This will automatically divide the work, roughly equally, among two PE's. Many different strategies for distributing work are supported by our machine. The important point is that no central authority is involved in distributing work. The Output Section only deals with the input tag and data, and a copy of the program to generate a new tag, and hence, the number of the number of the destination processor.

The data structure storage in this machine has something similar to the HEP computer; there are extra bits assiciated with each word of the memory. As shown in Fig. 5, these bits indicate

whether a word is empty or full. If a "read" is attempted on an empty word, the I-structure storage controller remembers the destination (i.e., the tag) where the data should be forwarded whenever it is stored in the word. The "store" operation causes the status of the word to be changed to "full," and in case there are deferred reads, the data to be sent to the destination of the deferred read operations. This type of storage, I think, is essential for high performance multiprocessor machines to avoid the so-called "read-before-write" problem.

Now, I will describe the communication system. Every PE is provided a 4x4 or 8x8 switching element and switching elements are connected to each other in any reasonable topology (see Fig. 6). A switching element receives a token (a packet) with a destination address on any of its input parts. Packets arrive asynchronously at the input parts and, hence, several packets may arrive simultaneously at a switching element. The switch looks up the destination address in a table which is kept inside the switch. The table essentially tells which output ports will take the packet closer to its final destination. If any of these output ports is free the packet is forwarded, otherwise it is held in a buffer in the switch. Basically the communication system is a store and forward packet communication network of very flexible topology.

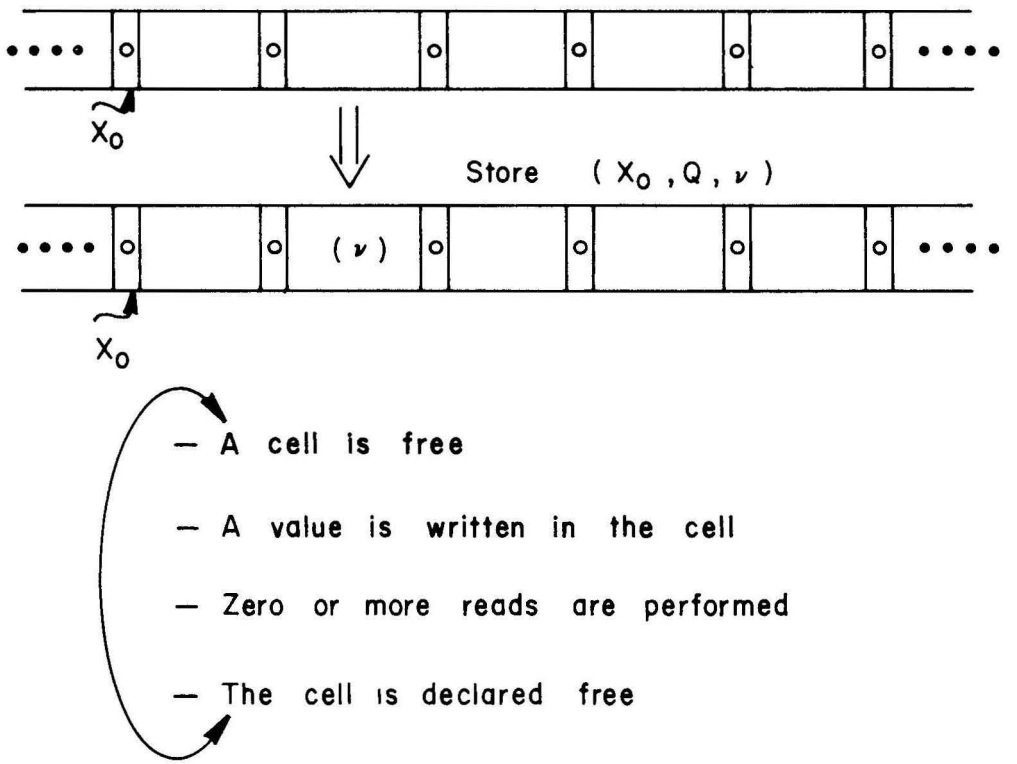


Fig. 5. I-structure storage.

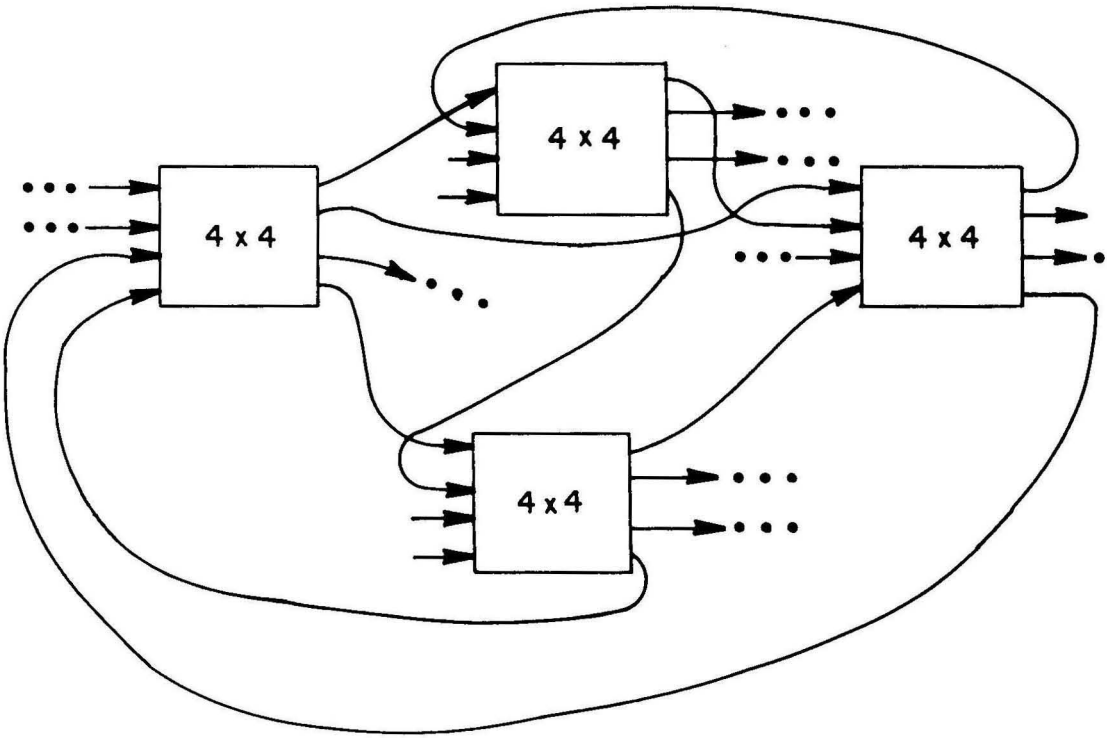


Fig. 6. Communication system. An interconnection of switching elements.

Next, I want to describe what we are building. We started out four years ago thinking, rather naively, in terms of custom VLSI chips. We hoped that our PE would fit in a single chip. This dream did not last long but we still hoped that the PE would fit at least on one board. It took another year to realize that the amount of custom hardware we would have to build to fit the PE on one board will involve seven custom chips of M68000 complexity. A hardware project of that magnitude is just too risky. That is to say, we would never have been able to find out if our architecture was defective or if the hardware was flaky. It's clear to us that we are in the business of testing an architectural idea and therefore it is necessary to take a fairly conservative approach to constructing hardware. Even the ultimate speed of the machine is not of real importance to us except to the extent that it should be fast enough to run some real user code. It doesn't have to be as fast as a Cray-1, but it has to be fast enough so that an application programmer who spends time programming the machine does not feel that his time has been wasted. In this way a programmer will get a taste of the future, at least as far as programming is concerned, and can take comfort in the fact that the next version of the machine may be faster than any sequential computer.

Thus, at some point we gave up the idea of building a real machine and decided to simulate as well as emulate the architecture. Since we had simulated an earlier version of the machine and were aware of the effort required, we were not thrilled about simulation initially. The push towards simulation came from IBM people. They said look if you guys really want us

to believe the potential of your architecture, you have to simulate the machine in a fair amount of detail. A cooperative effort with IBM Yorktown is underway now. We have received the gift of an IBM 4341 with 16 megabytes of physical storage for simulation experiments. We are running the same simulation program at Yorktown and MIT, and we hope to start running experiments on the simulator in the fall. This system is already about 250 pages of Pascal Code and it may grow by another 100 or 150 pages when the code to monitor the performance of the data-flow machine is included. My guess is that it will take about 24 CPU hours on the IBM 4341 to execute about 20 million data-flow instruction. Twenty million instructions do not represent a large time on a supercomputer, but properly designed simulation experiments should increase our understanding of the dynamic behavior of data-flow programs.

In order to execute even more instructions per experiment we are building a Multiprocessor Emulation Facility (MEF). The facility, funded by DARPA, will consist of 64 Lisp Machines connected together by a high bandwith packet communication network. The Lisp machines are of the Symbolic 3600 variety. Most of you are probably not familiar with these machines. Well, a Symbolic 3600 is a single user machine costing about $90,000! The minimum configuration consists of 2 megabytes of storage per processor. The interconnection network which is being designed by us will provide a bandwith of 4 megabytes per second per port. We think a 3600 will not be able to generate more than this much traffic if it is doing any useful computation.

We will make the MEF behave like the data-flow machine by making each 3600 emulate a Processing Element. Thus, 3600's won't look like Lisp machines, and the Lisp run time environment won't play any role in the emulated data-flow machine. However, Lisp machines provide a sophisticated programming environment and we are doing all our program development in Lisp. It should be noted that the internal parallelism of a PE would be emulated on a 3600 (which is a sequential machine) by multitasking or virtual concurrent processes.

Figure 7 shows the complete Multiprocessor Emulation Facility. Because this facility is going to be very expensive, external users will also have access to it. Only 8 of the 64 machines in MEF will be full machines while the rest will be without disks and displays. The terminals on full machines may be thought of as operator consoles on a main frame. Of course the system will be connected to local networks, so that remote program development can be done. It will be possible to do the development of an interpreter for a novel architecture on a local Lisp machine, and then ship the interpreter to the facility. In a sense this emulation facility is an analog of a big accelerator laboratory, where people would come to do experiments after having designed their experiments at home.

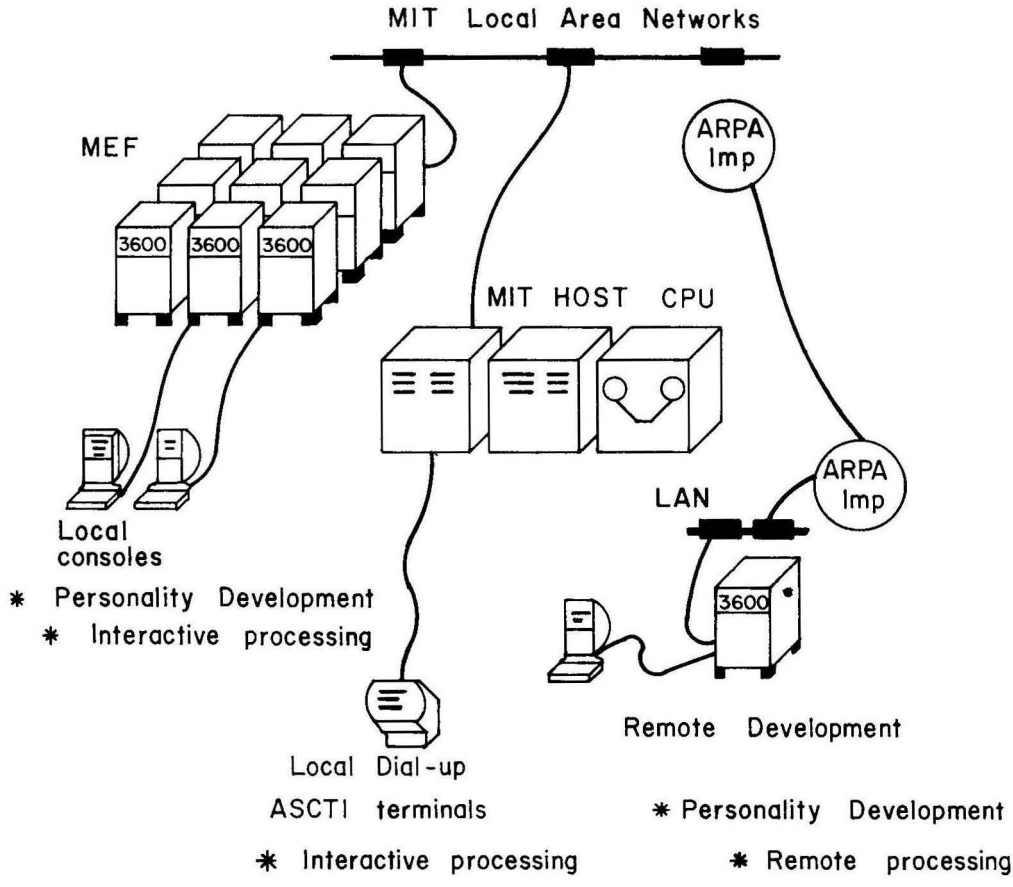


Fig. 7. Multiprocessor Emulation Facility (MEF).

PANEL DISCUSSION

THE NEW INTEREST IN PARALLEL COMPUTING.

Smith:

Many university researchers are suddenly interested in the architecture of parallel computers. What is the motivation?

Schwartz:

The basic motivations are two fold. First, there is a genuine technological opportunity, coming from the falling cost of the elementary processor. Second, everybody smells money and sees this as an opportunity to build up their own activities.

Smith:

Is the primary technological opportunity afforded by the reduced cost of hardware, the ease of designing the LSI, or more understanding of parallel computing?

Schwartz:

The falling cost.

Arvind:

My point of view is slightly different. I have been interested in parallel computing at least since 1975. I did not think in terms of building one then; the investigation was theoretical. I really believed that something was wrong in the way parallel computing was being attempted even though there were not very many parallel machines at that time. I always had doubts as to whether one could connect 16 commercially available processors together to get high performance out of the system.

Smith:

There are a few notably quiet members of the panel who are not saying what I expected them to say, namely, that we need to get our problems solved. Where are you guys when I need you?

Nash:

The point is that there is no more than another order of magnitude and a half, or so, in the conventional approach to enhancing computing and that's far away. There is just no other way of gaining large factors of computing power except by going parallel.

Wilson:

What excited me was when it became clear that by going to parallelism the jump in performance would be far more than the normal jump expected from commercial development. The factor of 100 per decade is going to be totally eclipsed by the gains that come from parallelism.

THE IMPORTANCE OF IMPLEMENTING ARCHITECTURAL IDEAS IN HARDWARE

Smith:

Under what circumstances is it important that university researchers have their architectural ideas implemented in real hardware? It's fairly clear that if you have a problem to solve and you are desperate you build real hardware. At what stage do people like Arvind at MIT or Schwartz at NYU need to build hardware?

Arvind:

I have had lots of discussions about this very question within MIT and with other colleagues outside. I believe it is important to go as far as possible with analysis, because building machines is very tedious. To build a machine which somebody else can use, is really a big task. It's not worth undertaking unless there is a good reason to believe that there is a bright architectural idea; the machine based on the idea cannot be analyzed any further but is worthy of further exploration. In other words, premature construction of new machines can be a colossal waste of time and money.

I strongly recommend that people should go as far as they can with the analysis and simulation of their architecture. They should talk to real users and program some applications before attempting to put together a machine which can be used. For example, there are two machines on Schwartz's chart, CM* and C. mmp, both at Carnegie Mellon University. The reliability of CM* was so poor and the total computing power so limited that few application programmers were interested in writing programs for the machine. The main claim the designers can make is that they put together 60 PDP - 11's and set some sort of a world record. I don't think it is worth building such machines because the learning is not commensurate with the money and the effort involved. Either the technology itself is to be tested by putting these machines together or the architectural concept is to be proven. The goals should be clear at the start.

Christ:

This is one aspect of the problem that has been changing. There was a time when building a computer was much more difficult than programming it. But as microprocessors and large scale integrated chips have become available that are really able to be interconnected by amateurs. At some point the possibility of building hardware for particular processes involves less work than programming. We may not have yet reached that stage, but in fact we might.

Schwartz:

This may be true for existing hardware. However building some new device like a Fast Fourier Transform chip is definitely several orders of magnitude harder than doing it in software.

Panel:

Perhaps our moderator could describe the Denelcor HEP approach to parallelism?

Smith:

The Denelcor HEP computer system is a dance hall machine in the sense of having all the boys on one side of the room and all the girls on the other with a switch in the middle. It has processors and memories and a connection network that ties them together, very much like the picture that Schwartz showed. It's not a data-flow machine according to some people but it is according to others. In fact, it's a hybrid lying somewhere between a data-flow machine and a multiprocessor.

It overcomes two of Arvind's objections to parallel architectures: it is possible to not wait for memory references and it is possible to avoid rewrite races in this architecture. It is still necessary to schedule programs using program counters as well as by the availability of data. In that sense it is a hybrid between the Von Neumann approach which schedules everything with program counters and the data-flow approach which schedules everything with the availability or the need for data. The HEP machine is pretty fast. It executes from 10 to 160 million instructions a second on 64 bit words. It's also pretty new. There are six HEP processors in the field in three systems. Each processor is a parallel processor though.

Wilson:

Bert Smith's HEP machine is the first machine for parallel processing which was designed the right way. The inter-communication network was designed first. The processors and the memory are designed to have all the features that the network requires including various signaling primitives, so that different processors can communicate with each other. The processors are designed so that they are not slowed down at all by the time-delay in the network. There are a number of really nice features on the HEP machine. The reason that it is not selling like hotcakes is that it is also expensive. It does not give the cost performance one would like to have these days.

However, we are leaning very hard on the agencies like the Department of Energy to point out that we have got to have HEPs available for experiments on parallel processing for which it is unique.

Smith:

The word unique is maybe a little strong in the light of what is going on in places like MIT and NYU. However, it is an available machine. Machines like the Ultra computer, the Tagged Token Data-flow machine that Arvind was describing, machines with static data-flow of which Jack Denis is an advocate, the Cedar machine at the University of Illinois, are being looked at, evaluated and in some cases designed at various universities. However, these machines are not yet available. Our machine is very much akin to those machines, that is, it has the same sort of applications and the same generic architecture.

Arvind:

I would like to make a simple point. Besides the fact that it is probably the most innovative machine which is out in the field today, I believe if people actually started using it we, the architects, would learn something from it. If there were eight or ten users of the machine, we would learn answers to questions such as "Does the machine stay up? Is it easy to program?"

STRATEGY OF DEVELOPMENT - HOW MANY GROUPS CAN BE INVOLVED?

Audience:

If you restrict attention to the people who have been working seriously on computing rather than on special purpose applications, that is, the people for whom the computing is more of an end in itself other than a means to a scientific end, then there is easily a score of very worthy projects. All of these are short of money and even more short of talent.

There is a difficult decision that has to be made because there is not enough money to go around. So Jack Schwartz raised the question about how do we narrow this down. Do we let one hundred flowers bloom? We certainly don't go as far as the Japanese in mandating from above. Perhaps there is a happy mid-ground.

Smith:

Underlying our concerns is the fact that in some cases it is necessary not just to build machines to avoid the problems of simulation but to build serious machines. In fact these must be quite serious machines. In Arvind's words these have to be machines that get some of the industrial people and some of the national laboratory people interested in actually writing programs and developing applications for the hardware. We need to develop computers with performance in the multimillion operation per second range and with concomitant tolerance invested in them. Are there comments?

Schwartz:

The right way to organize this would be to spot in a judiciously chosen set of projects across the spectrum of possibilities which tried all the major alternatives at a significant but neverless a relatively modest level of capability. Right now the universities are by and large trying to advance to the 64 processor level with relatively small processors. To go beyond that, one is starting to talk of major facilities and larger dollars. The right way to do that is to have some judiciously organized technical runoff with a smaller number of machines allowed to advance to the thousand and four thousand processor level. I am not sure that that degree of judicious judgment is being brought to bear.

COOPERATION WITH INDUSTRY

Smith:

What cooperation with industry have you had so far, and what cooperation would you like to have?

Wallace:

Within the UK scene we have had rather good cooperation with ICL. We would almost certainly not have had the machine but for a meeting between the regional computing center director and the director of ICL. It does appear to us that we have been able to show some of the things that this kind of machine can do. There is now a wider interest in other companies both within the United

Kingdom and the United States. We are optimistic that our future needs for computing engines can be met by early access to the manufacturers' prototypes that we really want.

Arvind:

It is not very difficult to convince a manufacturer to give you a computer if you say you will write programs for it. They will love you if you do that. We are made offers like this all the time. We have to turn them down because we cannot absorb too many different types of machines. The really difficult cooperation with industry is where we want industry to build a machine based on our ideas. This is the type of cooperation that is required to build new types of supercomputers.

Nash:

Over the last year we have had a seminar series on new computer concepts. We had a very difficult time getting people from industry to speak though we did have a couple of very good talks. What we are learning is that industry likes to talk on a one-on-one basis. That is, if I call up someone with whom I made a contact and start talking they will pretty quickly get me information on a non-disclosure basis. This does create a difficulty. With any large audience an industry person is apparently unlikely to say anything of great substance. On the other hand, if you try to talk to them individually there appears to be a lot of cooperation. As to what cooperation we are looking for, first, we are trying to get our hands on certain proprietary chips. Second, we need help with simulation software. Third, is an area where we are having difficulty which

I call crystal ball gazing. This is getting realistic estimates on what memories or processors will cost 18 or 24 months down the road. Industry is very good at that. They have in house crystal ball gazers who specialize in that. That sort of information could be very helpful to us.

Brenner:

I have a question for Wallace. After Illiac 4 the DAP is the first commercially available processor of its kind. There has been good cooperation between ICL and the universities and the National Research Council. Nevertheless it has been a commercial flop. What went wrong? What should one do differently? What can one do better to make it work next time around?

Wallace:

The one specific point that I already mentioned was that the manufacturer tied it very tightly into ICL main frames. These don't sell in any numbers abroad, but there are a number of centers in the United Kingdom with ICL main frames, so there is some kind of market for the DAP within Britain. Tying them to the mainframes was (at least with hindsight) a major mistake. If they had mounted it on a smaller machine like a VAX (as, one suspects, the designers proposed but were over ruled by the previous management) they would have had a viable machine three years ago costing $500,000 that would have approached Cray power for a wide range of problems. I think that would have been an interesting proposition.

Wilson:

Note that there was an ICL DAP that was bought commercially and then turned back to the company because the company couldn't make use of it. One of the problems with these machines is that someone has to learn how to use them. It's easier to learn in a university setting, partly because the problems people are trying to solve in a university setting are simpler than the problems that industry has to solve. I believe the logical progression is to learn how to use these machines starting in a university. Some expertise is established in what the machine is really good for and how to program it for that use. Then it is sent out to industry, but at same time with industry consulting with the universities to make sure they are buying it for something that is feasible to use it for.

This is not a Cray substitute. There are plenty of things that can be done very well on the Cray which could be done in principle on the DAP, but it would be a devil to program. This is especially true if the programs are already written for some other machine. Then it would be a disaster.

Wallace:

I don't think ICL thought very deeply about what the machine would be used for. I suspect if they thought about the possibility of having general image processing devices using the basic arrays out of which the DAP is built, then they could have built the DAP as a special machine based on these arrays. Possibly use of the basic arrays for image processing would have been commmercially feasible.

This is one of the points that encourages us to continue to work in this area because we can see that we are riding on the back of what could be a commercially successful architecture.

Christ:

At a somewhat lower level I could describe our interactions with the Intel Corporation. We are using their microprocessor in our device. We have gotten a fair amount of assistance from them. They have a program for interacting with universities that in our case essentially saved us 50% in the cost of items that we bought. We hope that perhaps our direction in using microprocessors might at some point produce a market for them.

Wilson:

Our interaction with Floating Point Systems was thrust on them. At first it was not something they particularly welcomed. It started when we bought one of their processors and asked people in the traditional FPS markets about Fortran. The general reation was "what?", because they were used to programming these machines in assembly language. That was the FPS market niche. I talked with the FPS people about Fortran and they said well maybe sometime. Then we started our Fortran project and a month later they hired a director of their Fortran project. We are not quite sure what the connection was. They actually bought half of the compiler we wrote. There were enormous difficulties that developed around that because of course they didn't understand very much about Fortran. They had various difficulties that they hadn't anticipated with our compiler. Actually it wasn't a very good compiler.

It took us a couple of years to get back their confidence after that.

We got a lot more respect from FPS after the manager of the project at Cornell started sending one paragraph summaries of every phone call he received with respect to the array processor. Somebody would call him up and a description of that phone call went off to FPS. One thing that became very clear is that it is necessary to identify a "friend" inside industry who can help. I think that everybody that deals with industry finds this. This is a particular person who is willing to fight on your behalf in company politics. The friend we had at FPS was the Vice President for Marketing. First, he had come from CDC and he was used to the role that universities have, because CDC had had some experience with that where as FPS did not. I remember one session when they were discussing the software for the FPS 164 and I was explaining what the universities would do. The FPS people were looking skeptical so they asked the Vice President for Marketing about the earlier experience of CDC. He described how important it was to have the universities running their equipment. Then someone pointed out the universities were running their own software, and didn't that hurt. The Vice President just laughed, because of course there was no CDC software worthy of the name at that time.

THE RELATION BETWEEN INDUSTRY, UNIVERSITIES AND NATIONAL LABORATORIES

Nash:

It's interesting to compare the roles and capabilities of industry, the weapons laboratories like Livermore and Los Alamos, the high-energy physics laboratories, and the universities to see where they are strong and where they are weak. There are a number of relevant factors like secrecy, hardware capability, and architectural creativity that one can see. There may be some hidden variables in industry that we don't know about such as the available dollars, the ability to work fast which is most important, the near versus far-sightedness, and the software capability. Recently for fun I put down all these factors; I scored them and added it all up. Surprisingly, their scores came out exactly the same to the last decimal point.

The point is, and it's an important point, in each area there are some great strengths and some weaknesses. For example, my personal view is that we at Fermilab are poor in software. Another example, is that the universities are creative and have a lot of foresightedness, but the ability to push something out the door is admittedly weak. (Even though this is one persons evaluation, Burt Smith looked at the chart and felt it in fact was pretty close to his own point of view.) The question is how can we get these four different, rather entrenched perspectives together. It's not easy, because they each have different motivations, different interests and fundamentally

different perspectives. I don't know the answer but in some sense that is why I put this up here.

Smith:

George Michael of Livermore and Bill Buzbee of Los Alamos, with the cooperation of Don Austin of the Department of Energy, have been sponsoring a series of meetings pertaining to architecture, applications, algorithms, programming environments, new programming languages, and the like. These meetings get people from the national laboratories active in defense together with colleagues from industry and universities. The problem you have pointed out is well recognized within the Department of Energy.

Arvind:

I think this is one area where big bucks can make a big difference. If a national initiative is taken you can really bring together users as well as industries and universities. Somebody has to figure out all the details of how this cooperation is to be brought about.

Audience:

These are big institutional units, but isn't this really a matter of making something happen between two people?

Smith:

When I first started taking part in these meetings I found that what I was learning was language. What we have to do is teach each other our languages so that we can develop those one-to-one communications. I find now that I can go to a Monte-Carlo conference and understand about 50% of what's said.

I can talk to people about aspects of computational physics or other application areas much better by virtue of the fact that people have been talking to me about these subjects for some time. In part that's what this meeting is about.

Audience:

Some of these projects are very large. A lot of money is needed to develop these kinds of systems. The question is how can you work with industry? From my perspective I am not sure that there are that many products in industry that can support that kind of funding. I think that this is part of the problem. Is there some way where one can work with a university in a useful way without a product?

Nash:

That is what I was getting at earlier in terms of our needs at Fermilab. We can get funding at some reasonable level through our basic sources but from our perspective what we need are certain of the things that industry is very good at. The crystal ball gazers, the simulation, the proprietary chips, board level computers that they can produce in large quantities, and computer-aided design capabilities. In our case it is not funding that we are after.

Schwartz:

In this area the universities have started to function as fast-moving scouts wandering over the terrain and discovering many interesting possibilities. The ideal role for industry would be to be the large battalions that come marching behind them and do a good job of putting substantial equipment in place.

A crucial element that is missing now is that they are not marching, with the exception of a couple of small companies like DENELCOR.

The basic problem is that in this area industries are trying to decide whether they want to be involved at all. The smaller manufacturers have their product concerns. They have to have short-term product development goals to make money. In Japan the situation is different and industry is on the march. They may not have scouted the terrain very well but as a matter of fact we are doing that for them.

Wilson:

It is often difficult to find product support for university operations. Take lasers as an example. The time delay from when the university research was done to the present $100 billion revolution in laser communications was two decades. Twenty years ago a university researcher couldn't get access to any of that $100 billion. On the other hand, the computing situation is different. Floating Point System's next 32 bit product has to find a one billion dollar market otherwise Floating Point Systems goes down the tube. This product is something it has to produce in the next year. This is not long term. FPS's basic problem is having the right ideas as to what to build and to know that product will find the necessary market. Remember that's one billion dollars and Floating Point Systems is a company many people have never heard of, it's not IBM.

The bucks we are talking about in the computing business are incredible and they are incredible compared to other areas where one talks about technology transfer.

One serious problem is getting the universities into the computing game. The universities don't recognize the opportunity that is there. The second problem is working out how to pay the universities for their function once they got into the game.

Audience:

Would some kind of a clearing house for funding be useful where smaller industries not big enough to back entire projects could make contact with and assist university researchers?

Smith:

As I understand it, the panelists want other support than just financial support. There was some discussion of MCC earlier. That isn't such a clearing house. There are research foundations and other methods of channeling industrial funds into research activities. I believe the emphasis here was on something rather different, namely how do we get technical information from industry and how do we communicate what we find. How do we act as scouts, perhaps speaking a different language. I wonder how many of the battalions speak Arapaho or Cheyenne which seems to be what some of us are speaking at times. However this may be difficult to pay because instead of costing money it costs talent. Talent is as expensive in industry as it is in academia.

Wilson:

Of course there is a problem with money, but there is another problem. The number of computer scientists is absurdly small for the needs of the United States. The requirements for people are not calculated properly either, because people usually don't estimate how many computer scientists are needed to go in and start up companies. Manpower estimates are always in terms of the established market. The proper way to count is not just how many people Hewlett-Packard hires.

The money for computers and computer science is absurdly small in the university scene today even compared with other subjects like physics. This is because computer science started late and only really gathered steam in the seventies after the big funding crunch. Universities that have to think of terms of a twenty year time scale for a professor aren't eager to run up their funds rapidly. On the other hand the computer science students are doubling every year and the funding is not growing to match.

Part of the trouble is the big funding from ARPA only goes to a very few universities so NSF as usual is left holding the bag. Nobody is giving support to raising the funding for computer science at NSF including most of the people inside NSF. As a result an absurdly small sum of money is determining how many computer scientists we're going to have ten to twenty years from now.

I hope some of you will start complaining, at least to the government, that the ratio of funds for computer science has got to be changed to be more sensible.

IS THERE SUCH A THING AS A GENERAL PURPOSE PARALLEL COMPUTER?

Schwartz:

We believe the design we are proposing and the data flow machines are relatively general purpose. I see them as the parallel analogs of the IBM 3081. There will also be special devices, and the special devices will perform well.

Of course it's hard to define "general purpose" as precisely for parallel machines as it is for the 3081. The IBM 3081 is a general-purpose machine because it can be programmed for hundreds of applications. I believe the same will be true of parallel machines except that those will be exclusively for large applications.

Nash:

Maybe one should say that "general purpose" for a parallel machine means that the machine can be efficiently programmed for just about every problem, but not necessarily be the optimal processor for every problem.

Wilson:

It isn't reasonable to call a parallel processing machine general purpose because of the way typical users would understand that term. It will always be possible to find a problem which cannot be solved any faster on a parallel processor than it can be solved on one element of that processor. There are even mathematical theorems to that effect. What is fair to say is that there are going to be enormously cost effective ways to use parallel processors for problems for which they are suited. There will be a distortion of the computer market towards the problems which are suited to parallel processing because they will be enormously effective. Huge markets will develop around those applications in data bases and scientific computing. There will be other areas which will not develop because they are not suitable to parallel processing.

Smith:

Schwartz is saying regardless of whether there is one problem or perhaps half-a-dozen it doesn't make any difference because for economic reasons, people will want to buy parallel processors for what we now call general purpose computing.

Wilson:

I think we should talk about single purpose and multi-purpose parallel processing but I think it is dangerous to call it general purpose.

Smith:

But you are saying that in some future time we will be doing data base management and payroll checks in parallel?

Nash:

Why not?

Arvind:

I think it is a non-issue, you don't buy a truck to drive to your office, yet trucks are general purpose.

Schwartz:

Looking at those present I would come down a little differently. If one looks ahead to the availability of general-purpose parallel machines, Al Brenner in charge of a large computer center, will buy a general-purpose machine where Tom Nash is interested in a very special purpose machine for a particular experiment.

Nash:

Semantics are important here because planners are often not that cognizant of the details. One has to be careful that by using buzz words like "general purpose" or "artificial intelligence" the problem is stereotyped too much and one might end up funding and planning things in the wrong way.

ARTIFICIAL INTELLIGENCE AND SCIENTIFIC COMPUTING

Smith:

At a recent workshop, Duane Adams of DARPA walked up to Ken Wilson and me and asked if there were any possibility that the same sort of computer could be used to do scientific computing and artificial intelligence. Do we need one score or two of

DARPA machines and one score or two of scientific machines? Who knows what's next? Is some sort of synthesis possible?

Wilson:

It's a little hard to know what artificial intelligence is these days. It does seem to me that there is going to be an enormous need for high bandwidth data movement. That is as close as one comes to a general need. There is a limit to what can be done with one bus to move data around. What we are talking about in these parallel systems ultimately comes down to the form of the network on which the data movement takes place. The question is what is its total aggregate bandwidth? Artificial intelligence is going to need that just as much as the scientific processor.

Brenner:

The semantics question is serious. These days DARPA is also using the term supercomputer. This confuses the issue of overlap. What we in the scientific community mean by supercomputer and what they want in an artificial intelligence super machine are not necessarily the same. What we want is a number crunching super machine. One should be very careful about that. DARPA has done a disservice to society in making that confusion. We should all straighten that out whenever possible.

Arvind:

Even though today these machines are very different I don't agree with that. Both sides require the other. It is clear that there are AI applications which really require very fast floating point arithmetic.

One moves into robotics the more that is the case. Robotics applications cannot be done very effectively on many of these AI type of machines. All the proposed artificial intelligence machines will have fast floating point units on them, precisely for that reason. Similarily, in scientific computing it is very hard to graduate beyond machines designed to execute "inner loops" efficiently unless applications are examined in totality. Until now, designers of supercomputers have ignored the problems of managing large address spaces, and the I/O bandwith between primary and secondary storage. These problems have to be solved if data-base management and graphics are to be integrated in scientific computing. Designs of AI machines have a head start over designs of supercomputers in these areas.

Schwartz:

If I were asked what is the ideal computing machine for dealing with those equations that are going to replace quantum chromodynamics, I would have to reply that I don't know. I don't know what those equations are, hence I can't say what computing machine is ideal for dealing with them. I have the same sense of bewilderment about the ideal artificial intelligence machine. If you look at the field technically there is such a shifting mass of paradigms in use that it is hard to identify what artificial intelligence is.

Smith:

I agree. In particular, the differences between a machine designed for a language like LISP and a machine designed for a language like PROLOG can be quite large.

Nash:

The phrase "artificial intelligence" is an example of the problem with semantics. I've been trying to find out for at least six months what artificial intelligence is. Every time I encounter a computer scientist I ask him. The best answer I got was from David Kuck. He said "artificial intelligence is anything that you can't write a program for." That's a rather large category of problems. Perhaps the buzz words "artificial intelligence" mean what the defense people want them to mean, and what they want is their own computers. It gets to be a territorial question at a certain point. We have to avoid these territorial problems as much as possible.

Smith:

That's very interesting. It seems that the AI machines are machines that we can't write programs for.

Arvind:

Actually I find that comment very strange because some of the largest programs that have been written to date are all in AI.

Smith:

Kuck's statement is "to solve problems you can't write programs for," not that you can't write AI programs. I was jesting of course.

HOW FAST CAN WE EXPECT SUPERCOMPUTERS TO BE BY THE YEAR 2000?

Smith:

We would like to ask the panel to prognosticate a bit and answer the question, "How fast can we expect supercomputers to be by the year 2000?" I don't care what measure you choose, if you all choose different ones then you'll have the advantage of not being compared to your neighbor. Nevertheless, let's have something that we can get our hands on. Considering what's happened in the last seventeen years, what are the next seventeen years going to bring in machines that you can buy or issue a purchase order for in the year 2000.

Arvind:

I would like to know where we are today?

Smith:

So would we all. We don't know where we are today and that's why your metric can be in any scale that you like. You can just say how many Crays or how many IBM 3081s.

Wallace:

For us the simplest point of view is to consider the next generation of the kind of machine that we are working with, which is rather special purpose but still more general than might be thought at first sight. The next generation will be 30 times faster. That is still bit serial, so take 32-bit machines. Another factor has to be included for unanticipated hardware developments that may take place by the year 2000.

So I would suggest that a thousand-fold increase can be anticipated, with further factors from hardware and special algorithms which cannot be foreseen.

Schwartz:

I think I would like to ratify that estimate. That would suggest something around 100,000 million instructions per second (MIPS) for the general-purpose machines and maybe a factor of as much as 100 beyond that for special purpose machines.

Wilson:

Every time I have tried to predict what will happen in the next three years, let alone in the year 2000, anything I have said has been an underestimate. It has been impossible to make an overestimate. For instance the last case that wiped me out was when I said we needed a floating point chip, something with a one microsecond cycle time. I speculated that we might have it in three years. The next day the announcement of the Hewlett-Packard chip was in Electronics Magazine

Smith:

So you are underestimating industry production?

Wilson:

I have underestimated a lot of factors. I underestimated the importance of the ICL DAP. For parallel processors, I underestimated the importance of the Monte Carlo processor at Santa Barbara. These underestimates have forced me to become more radical in my thinking of how computing is developing. A second factor to consider is that for parallel processing "how fast" depends on how may processors are available.

The number of processors depends partly on just how much money someone is willing to spend.

Now I doubt that anybody here today with the possible exception of myself has a concept of how much money people will be spending on computers by the year 2000. Scientists have consistently underestimated the willingness of society to spend money on their science after it has been developed into something.

With these factors in mind I predict that by 1990, not 2000, there will be one billion dollars worth of scientific computing equipment at Cornell. These factors have to be folded into estimates of how fast computers will be because obviously with a billion dollars we can have a lot faster system than my present budget of ten thousand dollars permits. This is true because obviously industry is going to have to spend more money on research and development than it does today. More of the price of the goods will be the research and development cost. Second, industry will have to put a larger fraction of the total R&D budget in computing because computing costs are going down while the cost of everything else goes up. Unfortunately a typical university's management is ridiculous, so that the fact that the cost of computers goes down and the fact that everything else goes up means that they put their money in everything else instead of computers. But, I believe that by 1990 the universities will be important enough to the economy and to industry in particular, so that they will be forced into doing some sensible budgeting.

This will mean that the budget for computers in universities will be a larger fraction of their total budget than it is today. In addition, the importance of universities in the computer market will mean that by 1990 the amount of computing equipment that we have in the universities will be limited not by money but by the total production capacity of the computer industry. In those terms it is perfectly obvious that a billion dollars of computing equipment in one of the best universities of the United States is probably an underestimate rather than an over estimate.

Christ:

I don't see the difference between general purpose and special purpose as clearly as my colleagues here. I wouldn't be surprised if that difference was quite blurred by the year 2000. Scaling up the sort of numbers that one can easily talk about now, a gigaflop or a thousand million floating point operations per second, I wouldn't be at all surprised to see that increased by four orders of magnitude. The hardware may improve by a factor of 100 and the scale of the system that is considered feasible by a similar factor.

Nash.

I like the units megaflops per megadollars. By the year 2000 one will be able to get an enormous boost by using broadly flexible "catered" processors. It is simply a question of how many transistors will be packed on the head of a chip by industry and how effective computer-aided design is going to be for devising circuits to do special purpose co-processors in our concept.

With these one can probably get that factor of a million in cost effectiveness which is the real issue now. If you can get a factor of another million in dollars you have available then you're really doing well.

Arvind:

Today we can probably do twenty million floating point operations per second on a sustained basis. Based on that, I would say in the year 2000 we probably would be able to do 1-10 billion useful operations per second.

Smith:

In summary, we are getting numbers between a thousand and ten thousand with the exception of Wilson who suggests perhaps 10^{14}. My numbers are more-or-less 10^4 for both special purpose and general purpose machines. (These comments were followed by several moments of intense discussion among the panelists as they tried to arrange betting odds with each other on the possibilities.)

WHAT IS THE SINGLE MOST IMPORTANT UNSOLVED PROBLEM IN PARALLEL COMPUTING?

Schwartz:

Getting industry moving on the possibilities.

Wilson:

Learning how to program not for parallel processors but for ordinary sequential processors.

Arvind:

I think technically programming, otherwise cooperation with industry.

Christ:

Knowing how to use effectively parallel architecture, that is thinking of the problem in the right way and programming efficiently.

Nash:

Software!

Wallace:

High level software.

Smith:

Having posed the question, I had an advance look at it. I had a difficult time deciding whether industrial cooperation or software was more important. I came down in favor of industry cooperation primarily because the importance of high speed computing really depends on making it available for people. If we don't have manufacturers and industrial users of high speed computing cooperating in order to bring it into currency then we won't have to worry about how easy the art of programming is or anything else. We just won't be in the ball game. We will only get leverage in high speed computing to the extent that new devices are supplied by industry and used by people like you.

FERMILAB INDUSTRIAL AFFILIATES

The Fermilab Industrial Affiliates organization was established in 1980 to improve university-industry research communications and to foster technology transfer from Fermilab. By now the Affiliates number more than 30 institutions including many research-oriented companies in the Fortune 500 list as well as several companies formed by Fermilab staff members and users.

Direct activities of the Affiliates include visits of company representatives to Fermilab and Fermilab personnel to Affiliates. The annual meeting is one of the principal opportunities for such visits. Affiliates also receive selected Fermilab reports and other information about on-going work at the Laboratory.

Affiliate members have direct access to Fermilab staff for information on the work at the Laboratory. They receive copies of significant Fermilab technical publications and are kept abreast of important seminars on technical matters at the Laboratory. Efforts are also underway to stimulate more visits of staff members from Affiliates' institutions to Fermilab and vice-versa. An annual meeting is held in the spring. This Round Table was presented at the third annual meeting. At the meeting the visitors are given a comprehensive presentation of the activities underway at the Laboratory. Tours and individual conferences present an opportunity to see the Fermilab work in detail. A membership fee of $1000 per year is charged to offset part of the cost of the program.

Specific technology innovations are only one facet of the work of the Laboratory that is emphasized. The "scientific culture" related to particle physics is given heavy weight as well as the long range potential of activities such as the development of superconductivity technology. The participation of more than a hundred universities in all phases of the Laboratory is also important to Affiliate members. Often, an Affiliate's interests in the Laboratory is hard to gauge. A major farm equipment manufacturer turned out to be one of the heaviest users of large computers in the United States.

Fermi National Accelerator Laboratory, by its nature, amalgamates a wide array of engineering and physics disciplines of interest to Affiliate members and industry in general.

The acceleration of particles requires a working together of systems of high voltage electrostatics, high power radiofrequency signals, and rapidly pulsed magnets all under rigid and precise computer control. Beam optics, high vacuum techniques, ion sourcery are also involved. Particle detection adds new areas in terms of spatial and temporal resolution, fast logic circuitry and decision making, techniques of multi-dimensional pattern recognition, signal processing, and efficient number crunching.

A seven year R&D effort in superconductivity has culminated in the construction of a four-mile ring of superconducting magnets with associated cryogenic systems. A substantial fraction of the world helium refrigeration capacity is at Fermilab. Advanced R&D looks to new materials, refrigeration,

and understanding which will lead to pulsed magnets operating with magnetic fields of greater than 100 kilogauss.

Since the scale of Fermilab is so large, four miles of tunnel filled with sophisticated magnets and a 6800 acre site, an important ingredient in much of the R&D has been a search for innovative, cost-conscious designs. Special fabrication techniques such as laminar tooling have been invented in pursuit of precision coupled with economy. Remote and automomous control is important for the same reason. This has led to important developments in large-scale distributed control and data-collection systems.

Technology-related programs at Fermilab include holography, solar energy, and neutron cancer therapy. A brochure is available (Technology Development at Fermilab) with capsule descriptions of all these activities.

For information on the Affiliates contact:

Dr. Richard A. Carrigan, Jr.
Assistant Head of the Research Division

or

Dr. Leon M. Lederman
Director

Fermi National Accelerator Laboratory
P. O. Box 500
Batavia, IL 60510
(312) 840-3333

FERMILAB INDUSTRIAL AFFILIATES

Arco Petroleum Products
Bell Laboratories
Borg-Warner Corporation
Brunswick Corporation
Cherry Electrical Products Corporation
CMD Development
Commonwealth Edison
Deere & Company
Digital Equipment Corporation
Digital Pathways, Inc.
Eaton Corporation
FMC Corporation
General Electric Corporation
The Goodyear Tire & Rubber Company
W. W. Grainger, Inc.
The Harshaw Chemical Company
Hewlett Packard
State of Illinois
International Business Machines Corporation
Johnson & Johnson
Litton Industries
McGraw-Edison Company
Nalco Chemical Company
Nuclear Data, Inc.
NYCB Real-Time Computing, Inc.
Omnibyte Corporation
Raychem Corporation
Richardson Electronics
Shell Development Company
Standard Oil Company (Indiana)
Sunbeam Appliance Company
UOP, Inc.
Varian Associates
Westinghouse Electric Corporation