GRAPE Project: special-purpose computers for many-body simulations

Junichiro Makino [a]

[a] *Department of Astronomy, University of Tokyo,*
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-0033, Japan
email:makino@astron.s.u-tokyo.ac.jp
phone: +81-3-5841-4276, fax: +81-3-5841-7644

Abstract

In this paper, we overview the GRAPE (GRAvity piPE) project to develop and use special-purpose computers for astrophysical many-body simulations.

First, we discuss the trends in semiconductor technology and computer architecture, and discuss merits and demerits of special-purpose computers. The main advantage of the special-purpose systems is the possibility to utilize a much higher fraction of available silicon to actually do calculations. Most of the silicon on general-purpose microprocessors are now used for control logics and cache, in other words, to assure the generality.

As an example, we describe our GRAPE project. In astrophysical many-body simulations, practically all computer time is spent to evaluate particle-particle interactions. This is because the gravitational interaction between astrophysical bodies is a long-range interaction and we cannot apply a cutoff. The basic idea behind the GRAPE project is let the special-purpose computer perform only the evaluation of the particle-particle interaction. All the rest of the calculation, such as the time integration of the orbits of particles, I/O, and diagnostics are handled by a general-purpose host computer. This partitioning has many important advantages, such as the ease of the development of the hardware, the flexibility of the overall system and long lifetime of the system.

The newest machine, GRAPE-6, which will be completed soon, will provide the peak speed exceeding 100 Tflops, for the development cost of less than 5 M dollars.

Key words: 02.70.Ns Molecular dynamics and particle methods; 45.50.Pk Celestial mechanics;

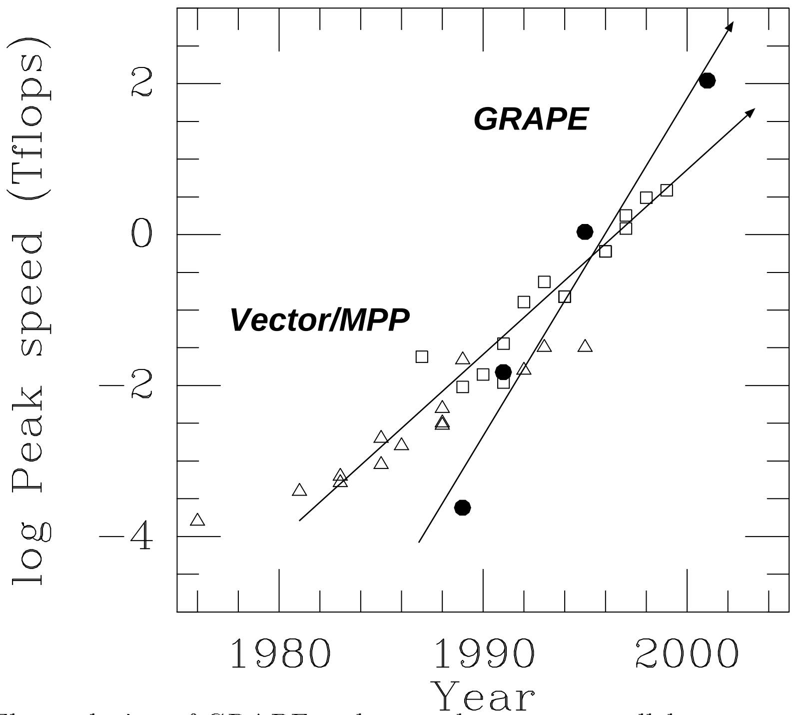

Fig. 1. The evolution of GRAPE and general-purpose parallel computers. The peak speed is plotted against the year of delivery. Filled circles, open triangles and open squares denote GRAPEs, vector processors, and parallel processors, respectively.

1 Introduction

In this paper, we discuss a rather old and not well-exploited approach to "Grand Challenge" problems in computational sciences, namely to design and build the computer hardwares specialized and optimized for relatively narrow range of problems.

The primary reason why we discuss this approach is simply that our GRAPE project [8,6] to develop special-purpose computers for astrophysical N-body problems has been, at least in our opinion, highly successful, compared to the conventional approach to adopt the algorithms to available computer hardwares. Figure 1 shows the peak speed of our GRAPE systems and that of representative high-performance general-purpose computers. In year 1995, we completed GRAPE-4 with the peak speed of 1.08 Tflops. It was the fastest computer for scientific simulations at that time. In year 2001, we will complete the GRAPE-6 system, with the peak speed around 120 Tflops. The development costs, or the total budgets, of GRAPE-4 and GRAPE-6 are 2.5 and 4.5 million USD, respectively, which should be compared to around 100 million USD of latest ASCI machines (12 Tflops by the end of year 2000).

Of course, the peak speed does not tell much, since the actual speed one can achieve is, by definition, lower than the peak speed. In addition, what ultimately matters is what kind of problems can actually be solved. Here, it would be sufficient to say that more than 100 copies of various versions of GRAPE hardwares have been manufactured and are now being used in more than 30 institutes worldwide to explore wide variety of problems in astrophysics[5].

In the rest of this paper, we first discuss the trends in the semiconductor technology and that of general-purpose computers in section 2. Our main point is that general-purpose computers now utilize only a small fraction of available number of transistors to do useful arithmetic operations. This ratio has been decreasing for the last 30 years, and will continue to do so for the foreseeable future. In section 3, we discuss the potential advantages and drawbacks of designing and developing special-purpose systems. The potential gain is very large, since, at least in some cases, we might be able to use a much larger fraction of available transistors to perform useful arithmetic operation. On the other hand, there are many practical difficulties which would offset the potential gain. In section 4, we discuss our GRAPE project as an example of, well, reasonably successful projects to develop special-purpose computers. In section 5, we speculate on the future of the large-scale scientific computing.

2 The long-term trends in semiconductor technology and computer architecture

The architecture of the high-performance computer showed rather drastic changes in every 10-15 years. In 1960s, single-processor scalar computers such as CDC 6600 were "supercomputers." After the introduction of Cray-1 in 1976, vector processors took over. By around 1990, distributed-memory parallel computers, either based on the CMOS implementation of vector processors (Fujitsu VPP) or on COTS (commodity, off-the-shelf) microprocessors (Cray T3x, ASCI machines, etc.) started to replace vector-parallel computers in some application areas.

These changes were, to a large extent, the results of responses of the computer architects to the advance in the device technology. The advance in the device technology caused miniaturization of the transistors, which resulted in (a) the increase in the available number of transistors and (b) the increase in the switching speed of individual transistors.

Before Cray-1, or its predecessor CDC-7600, the number of transistors available for a computer architect was so small that he could not build a complete parallel multiplier which could deliver one result per clock period. Thus, compared to what the architects have to do now, the design of the high-performance computer at that time was rather simple. First one would design a multi-cycle multiplier which is as large (and therefore fast) as possible within the given budget of transistors, and then add control logics and a memory unit fast enough to keep the multiplier busy. Of course, there are a number of important issues, but the above picture gives the general idea.

As device technology advanced, the available number of transistors increased. At one point, thus, it exceeded the number of transistors needed to implement a fully parallel, single-cycle multiplier. This occurred by late 1960s. Once fully-parallel multiplier had become possible, we could further improve its performance by pipelining.

The natural architecture for fully parallel, pipelined multipliers is the vector processor, with the memory system organized into many independent banks to provide sufficient throughput required by the arithmetic unit. Here, however, the control logic and the memory system became much more complex than those of scalar processors, which meant that a large fraction of the transistors were used for them.

As the device technology advanced further, the available number of transistors increased even more, and by late 1970s it became possible to pack multiple vector arithmetic units into a single computer. Of course, this resulted in the sheer increase in the complexity of control logics and the memory system. This is simple to explain. Suppose one arithmetic unit requires n memory banks. If a computer has p arithmetic units, it needs np memory banks. Since each arithmetic unit need path to all memory banks, the number of paths is np^2. Thus, at relatively small p most of the transistors would be consumed by the memory units.

Additional problem with multiple arithmetic units and many memory banks is the wire length. Miniaturization of the transistors resulted in the faster switching speed. However, the speed of light does not change. Therefore, in order to improve the overall speed of a computer, one has to make it smaller and smaller. On the other hand, heat dissipation put the physical limit on the number of transistors per unit volume.

Nonetheless, the device technology advanced further. Here, the only viable solution was to organize large number of transistors to many processors with small number of vector units, and then to connect them with a relatively slow network. This approach is what was first adopted by NAL NWT (Numerical Wind Tunnel of National Aerospace Laboratory of Japan), which was then commercialized as Fujitsu VPP-500. This approach is still being followed by Fujitsu and NEC.

Around the time that Fujitsu VPP-500 arrived, however, a more fundamental transition was underway. High-performance computers illustrated above were implemented mostly in ECL logic. By mid 1990s, however, microprocessors based on CMOS VLSI started to offer the clock speed faster than that of ECL-based vector processors.

Miniaturization benefits both of ECL and CMOS technology, but not in the same way. Because of its comparatively low power consumption, CMOS LSI can house larger number of transistors than ECL LSI. The switching speed of CMOS transistor used to be much slower than that of ECL transistors. By mid 1990s, however, one-chip CMOS microprocessors started to achieve the clock speed comparable or faster than that of ECL-based processors, primarily because of shorter interconnection between transistors. As a result, both Fujitsu and NEC re-implemented their vector processors in CMOS by late 1990s. Both Cray Research and Cray Computer tried to pursue the ECL-like fast logic. The former was merged into SGI and then sold to Tera, and the latter closed the door.

Right now, however, the "mainstream" of the high-performance computing is not the parallel vector computers from Fujitsu or NEC, but the clusters of PCs or UNIX servers, sometimes called "Beowulfs"[2]. They are based on commodity one-chip microprocessors.

This dominance of Beowulfs is the result of the combination of technological and economical developments. At present, it is practically impossible to make a CPU which is faster than these commodity microprocessors, simply because the manufacturers of these "commodity" microprocessors actually use the most advanced technology both in the design and manufacturing. This use of the advanced technology was made possible by the sheer number of these processors sold.

However, if we look back the evolution of the microprocessors, we can see that they are now in the same trap as vector processors were in 1980s. Figure 2 shows the maximum number of floating point operations performed per clock cycle for representative microprocessors. The increase in the operation count per cycle was very fast, until it reaches two (one addition and one multiplication) by 1990. Then the increase stalled, and in 1990s there was essentially no increase. Even in the year 2000, there is no microprocessor which can perform more than four floating point operations per cycle.

In other words, practically *all* the increase in the available number of transistors in 1990s was spent for things other than the arithmetic units. This is exactly the same as what happened to vector processors in 1980s. The only difference is that microprocessors are much cheaper than vector processors. More importantly, their price has been dropping rather fast. Therefore, by hooking up tens of thousands of microprocessors, one can still deliver very impressive peak performance, and that peak performance grows rather nicely as the price drops.

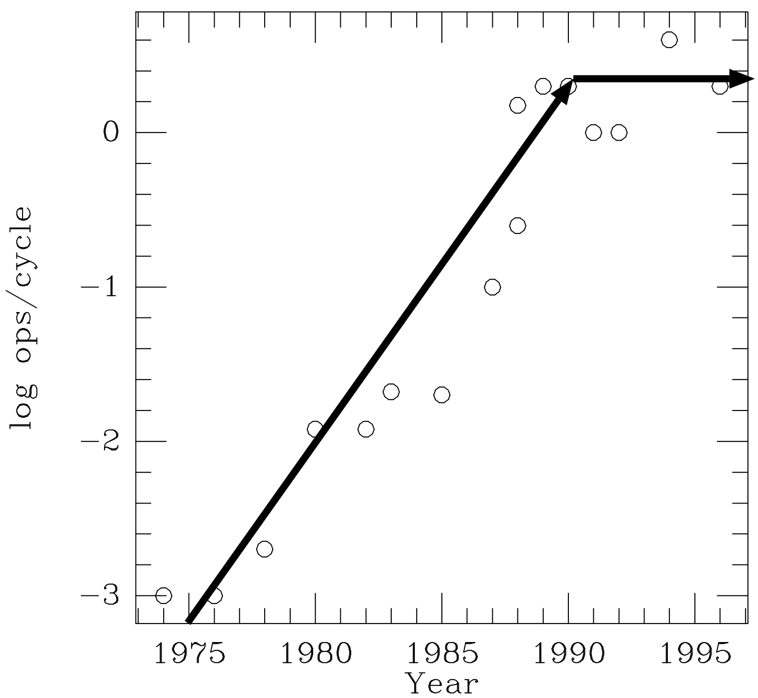

Fig. 2. The evolution of the maximum number of floating point operations performed on representative microprocessors, plotted versus time. open circles are the values of actual processors. Arrows indicates the trends.

Thus, even though the theoretical peak performance of Beowulfs has been, and will be, increasing, that fact does not imply that they make good use of the device technology. Quite the contrary, they are using ever diminishing fraction of total silicon to do useful floating point operations.

The other problem with the Beowulfs is that their architecture, or rather, non-existence of their architecture, limits their applicability to rather a narrow range of problems. Beowulfs are characterized as bunch of PCs connected via commodity network interface. Thus, the latency and bandwidth of the network interface is relatively low. In addition, usual adoption of TCP/IP based communication software tends to add huge overhead, resulting in a very high latency. One could achieve significantly lower latency and higher throughput by moving to special network hardwares such as Myrinet, but the price becomes much higher as well.

Even with fast networks, it is difficult to reduce the latency of the operations which requires bidirectional communication between the CPU and the network interface. In particular, it's not easy to achieve low latency for global operations, such as global reduction operation.

Finally, the programming model implemented on these clusters, the message-passing model, is the lowest-level libraries to achieve parallel computing, and by no means a high-level, easy-to-use programming environment. Thus, except for very simple problems such as to solve differential equations on a rectangular grid, it is very time-consuming to develop softwares in the message-passing model.

To summarize, the current direction of the evolution of general-purpose high-performance computing (that is, clusters of PCs) has the following problems:

(1) The fraction of the transistors on a processor LSI chip used for floating-point operations is small, and is decreasing further.
(2) The network architecture of PC clusters limits the range of applications
(3) The programming model currently used on clusters makes the application development expensive and time-consuming.

In the next section, we discuss an entirely different approach to large scale scientific computing, namely to build special-purpose computers.

3 Special-purpose computing

The idea of building special-purpose computers for specific numerical problems is not new. In fact, the very first digital electronic computers (ABC and ENIAC) were designed to solve specific problems. However, the designers of early computers found that their machines could be used for a much wider range of problems, and thus the evolution of the programmable general-purpose computer started.

At the time when even the largest supercomputer did not have a fully parallel multiplier, it did not make sense to design a special-purpose computer. Any computer, either special- or general-purpose, consists of arithmetic units (mainly multipliers), control logic and memory. A special-purpose computer might have simpler control logic or smaller memory than general-purpose computer. However, if the cost of the arithmetic unit itself is a fair fraction of the total cost of the machine, we cannot gain much by trying to reduce the cost of the rest of the system. Of course, if one could simplify the arithmetic unit itself, as in the case of machines for Ising models [3,9], a large gain was achieved.

This situation, however, has changed completely, as we've seen in the previous section. To give a concrete example, let us discuss the astrophysical N-body problem and our GRAPE (GRAvity PipE) hardware specialized for that problem. The master equation for the gravitational N-body problem is given by:

$$\frac{d^2 x_i}{dt^2} = -\sum_{j \neq i} G m_j \frac{x_j - x_i}{|x_j - x_i|^3}, \tag{1}$$

where x_i and m_i are the position and mass of the particle with index i and G is the gravitational constant. The summation is taken over all particles (typically stars) in the system under study.

The number of particles N varies widely depending on what kind of systems are studied. A small star cluster like Hyades might consist of several thousand stars. Globular clusters typically contain one million stars, and galaxies hundreds of billions of stars. In the case of galaxies, it is impossible to model them in a star-by-star basis, and we model the distribution of stars in the six-dimensional phase space with much smaller number of particles.

If N is larger than several tens, the calculation cost of the right hand side of equation (1) dominates the total calculation cost. If N is very large, we could use sophisticated algorithms such as the Barnes-Hut treecode[1], FMM[4], or the particle-mesh method. However, with treecode or FMM, the direct evaluation of gravitational interaction between near-neighbor particles still dominates the total cost. Even in the case of the particle mesh method, if we want high accuracy, near-neighbor interaction must be calculated directly.

Let us assume that we have at hand a modest number of processors, or more simply arithmetic units, to solve this problem. The general-purpose approach can be described as follows. First we build a complete programmable computer around each of the arithmetic unit and then connect them with some communication network. Then we develop the program for this parallel computer.

Since we know that almost all computing time is spent on the calculation of the interaction, we can take an alternative approach. In this approach, we use one arithmetic unit (or small fraction of the arithmetic units available) to build programmable computer(s), and use the rest only for the force calculation. In principle, those arithmetic units used for the force calculation could be programmable computers. However, since we decided to use them only for the force calculation, it's much easier and cheaper to organize them into pipeline processor to calculate the force between particles. If we have more processors than needed to build one pipeline, we can use them to build multiple pipelines. If the number of the pipelines is small, they can just be connected to the general-purpose processor (we call it "host computer" here and hereafter) independently. Since the host computer is a rather simple scalar computer, it's relatively easy to write softwares for it. This is the basic idea of our GRAPE systems, shown in figure 3. Figure 4 shows the force calculation pipeline.

The theoretical gain of this kind of architecture is huge, in particular when the special-purpose pipeline unit is implemented in a custom LSI (either gate array, standard cell or full-custom design). The reason is that we can actually use almost all of the available transistors on a chip to implement floating point arithmetic units. To give an idea, the GRAPE-6 processor chip which

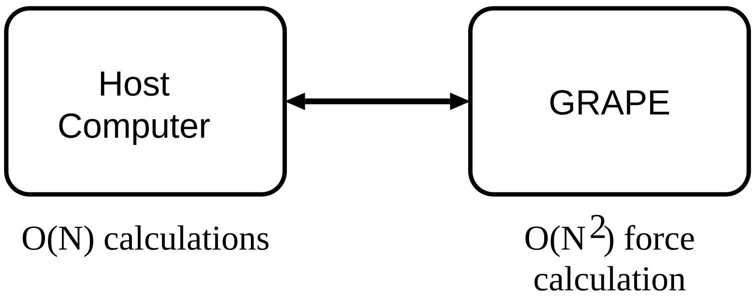

Fig. 3. The basic architecture of GRAPE

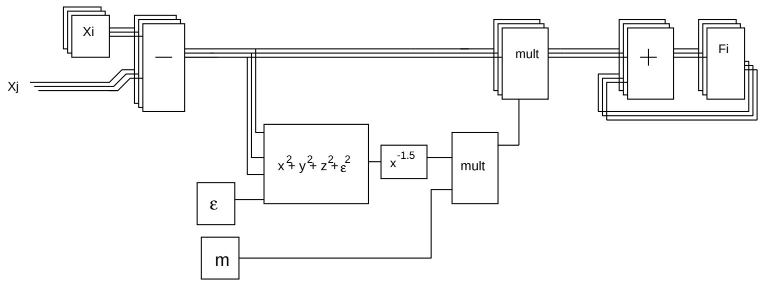

Fig. 4. The force calculation pipeline

we completed in 1999 contains more than 300 arithmetic units. The physical size of the chip is about 11 mm by 11 mm, with 0.25μm technology. Compare this number with just two arithmetic units of the latest Intel processors implemented using 0.18μm technology.

Another important advantage is that it is easy to utilize large number of pipelines. There are two ways to parallelize the force calculation. One is just let different pipelines calculate forces on different particles. In this case, these different pipelines can share the memory unit which provides the data of particles which exert the force. The other possibility is let different pipelines calculate the force on one same particle, but from different subset of particles in the system. By combining these two ways, we can construct a system with thousands or even tens of thousands of pipelines, and still enjoy pretty high actual performance.

Obviously, these advantages comes from the nature of the problem itself. In astrophysical N-body problem, each particle interacts with all other particles in the system (or, in the case of treecode or FMM, large number of neighbor particles). Thus, the calculation cost is $O(N^2)$, while data transfer between the memory and processors can be reduced to $O(N)$.

Our GRAPE project is not the only project to develop special-purpose computers for scientific simulations. As mentioned above, there were machines for Ising model simulations. In addition, Lattice QCD calculation has long and impressive history of highly successful specially-designed computers (see [10]). Also, there were various machines developed for molecular dynamics (MD) simulation, which is rather similar to astrophysical N-body simulation (for an overview see Makino and Taiji[6]).

These special-purpose computers can be classified into two categories. One is programmable parallel computers with various optimizations for specific problems. For example, if we do not need much memory, we can replace the DRAM main memory with a few fast SRAM chips, thus reducing the total cost by a fair factor and at the same time improving the performance. If we do not need fast communication, we can use rather simple network architecture. If we do not need double-precision arithmetic, we can reduce the number of bits. All machines for Lattice QCD and many for MD adopted this approach.

The other approach is to develop "algorithm oriented" processors similar to that of GRAPE. All machines for Ising model and some machines for MD took this approach. The Delft Molecular Dynamics Processor (DMDP) is one of the earliest. It had a force calculation pipeline very similar to those of GRAPE hardwares (see figure 4). However, there is one rather important difference. In our GRAPE project, we developed the force calculation pipeline only, and then connect it to a workstation. On the other hand, with DMDP the designers built everything themselves. Things like the time integration and analysis of the simulation result were done on hardware pipelines designed for them.

The obvious advantage of the DMDP approach was that its performance is not limited by that of the general-purpose host computer. Thus, impressively high performance was achieved (more than 100 Mflops in early 1980s). However, there are several disadvantages. The most important one is the loss of the flexibility to use the system for wide range of problems. With GRAPE, all data is on the host at any time during simulation. Thus, we can do whatever analysis on the data we like. We also can change time integration method. We can even add particle-particle interactions different from the gravity, as far as they are not too expensive. This kind of flexibility allowed us to study wide variety of astronomical objects.

A second disadvantage of the DMDP approach is that hardware becomes more complex simply because hardwares for operations other than the force calculation must be developed. However, at the time of the design of the DMDP, their choice was probably inevitable, since there were no workstations or PCs yet. The cheapest computers with reasonable performance for floating-point operations were minicomputers, whose price was much higher than the budget of the entire DMDP project.

As we stated above, there were many projects to develop special-purpose computers. Those documented in the literature are mostly successful projects. At the least they have produced working machines. However, there have been many more not-well-documented projects. Large fraction of them faded out without producing a working machine. In the following, we list what we think are the typical cause of the failure of a project to develop special-purpose computers. This list is by no means comprehensive, but may be of some help to those who interested in building their own machine.

(1) **Long development time**. Even though there are enormous inefficiency, the performance of general-purpose computers is still improving at the rate of a factor of 10 in every five years. Therefore, if you spend five years to develop a machine, you lose a factor of ten in relative advantage. Of course, in many cases the designers originally underestimated the development time. So this problem would have been very difficult to avoid.

(2) **Too small gain**. Even if your machine achieved the advantage of a factor of 10 by the time of completion, in five years it will lose all advantage. So the lifetime of your machine is rather short. If the advantage is a factor of 10 *at the time of the design*, the project is almost guaranteed to fail.

(3) **Too wide applications**. This is not necessarily a failure, if high performance is achieved. However, this is the most common failure which leads to both small gain and long development time.

(4) **Too narrow applications or too difficult to use**. If there is not much scientific outcome from your machine, it would not be regarded as a great success even if it achieved very good performance.

(5) **Obsolete technology/design method**. Device technology is moving very fast, and so the design software/methodology. So, if the technology you used was three-year old, you have already lost the relative advantage of a factor of four.

(6) **Untested technology/design method**. Even though the advance of the device technology is pretty well predictable, in most cases what the manufactures claim that they can deliver would not be delivered on time. So be careful.

In the next section, we describe GRAPE-6, which is currently under development, as one example of what can be done with the special-purpose approach.

4 GRAPE-6

In 1997, we started the GRAPE-6 project. It's a five-year project funded by JSPS (Japan Society for the Promotion of Science), and the planned total budget is about 500 M JYE.

The GRAPE-6 is essentially a scaled-up version of GRAPE-4[7], with the peak

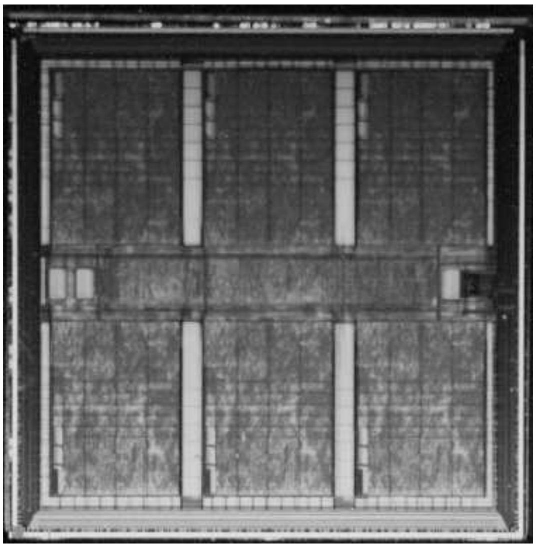

Fig. 5. The GRAPE-6 processor chip.

speed exceeding 100 Tflops. It will consist of around 3000 pipeline chips, each with the peak speed of 40 Gflops. In comparison, GRAPE-4 consists of 1700 pipeline chips, each with 600 Mflops. The increase of a factor of 100 in speed is achieved by integrating six pipelines into one chip (GRAPE-4 chip has one pipeline which needs three cycles to calculate the force from one particle) and using 3–4 times higher clock frequency. The advance of the device technology (from 1μm to 0.25μm) made these improvements possible. Figure 5 shows the first sample of the processor chip delivered in early 1999. The six pipeline units are visible.

Figures 6 and 7 shows the processor board with 16 processor chips and the prototype four-board system. This four-board system has the theoretical peak speed of 2.1 Tflops, and has achieved the sustained speed of 1.1 Tflops for the simulation of 1 million-body system.

GRAPE-6 will be completed by the year 2001. We plan to make small version of GRAPE-6 (peak speed of around one teraflops) commercially available by that time. We've found that the commercial availability of small machines is essential to maximize the scientific outcome from GRAPE hardwares.

5 Discussion

We overviewed the current technological and economical trends in high-performance scientific computing. At least for certain range of problems, a special-purpose computer, or, a combination of special- and general-purpose computers such as our GRAPE systems, offer a real and proven advantage over traditional general-purpose computers. The relative advantage has been increasing, and

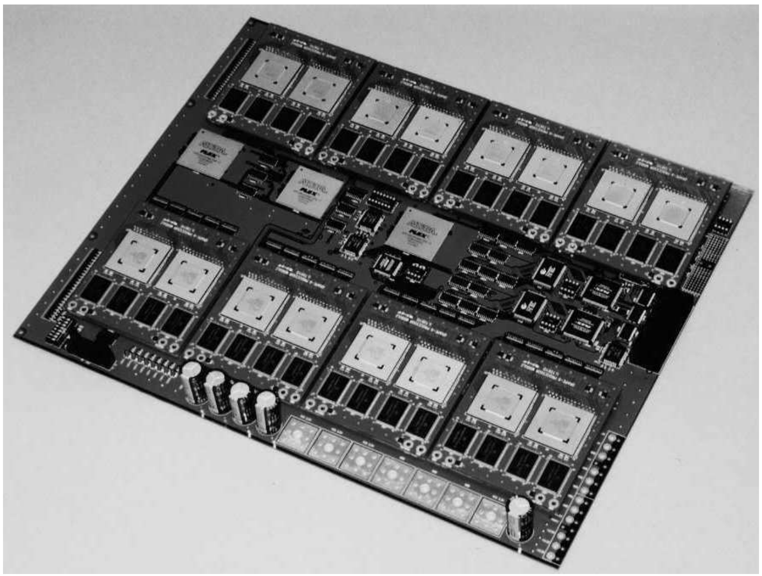

Fig. 6. The processor board of the GRAPE-6 with 16 processor chips. Two processor chips are mounted on modules, on which four memory chips are also mounted. One board houses eight modules.

Fig. 7. The prototype system with four processor board.

will keep to do so for the next one or two decades. Our GRAPE-6 system will be completed by next year, with the peak speed of 120 Tflops for the total budget of 4.5 M USD. These numbers could be compared with the price and performance of the newest ASCI machine, the Option White, which will deliver around 10 Tflops for the price of around 100 M USD by the end of the year 2000.

In the following, we speculate on the future of large-scale high-performance computing. As we have seen in section 2, the direction of the evolution of "general-purpose" parallel computers is such that the application range will be more and more limited, and yet the efficiency of the hardware will go down.

On the other hand, special-purpose systems, at least in principle, will not suffer these problems. In practice, however, the approaches we have taken so far is becoming more and more difficult, because the initial development cost of the custom LSI goes up as the technology advances. The development cost goes up because of two reasons. The first is that the amount of work to do the logic design, test design, physical layout and design validation increases as the number of transistors in a chip increases. In the case of special-purpose systems, the amount of work for the logic and test design, which we can do in-house, does not increase too rapidly, but physical layout and design validation, which is the work of the semiconductor company, take a long time and therefore lots of money. The second reason is that the investment needed to build the semiconductor plant increases rapidly as the technology advances.

Very roughly speaking, for same physical size of the chip, the amount of money we paid was inversely proportional to the design rule: around a quarter million USD for 1μm (GRAPE-4) and around 1 million USD for 0.25μm (GRAPE-6). In both case, the size of the chip is around 110mm^2. If this trend continues, the design cost of 0.13μm would be two million USD. The total budget must be significantly larger than the development cost of the chip, since otherwise the price per performance will be too high. It is not easy to get such a large fund for a project of theoretical study in pure science.

One possible compromise is the use of FPGA (Field programmable gate array) chips as the building block of the pipelined processors. This choice will reduce the initial cost from one million dollars to less than ten thousand dollars (the price of the design software), since the FPGA chip itself is mass-produced and the design is loaded to a FPGA chip by configuring the switches and lookup tables in the chip.

Roughly speaking, because of this programmability, the calculation speed achieved by one FPGA chip is 100 times lower than that can be achieved with a custom LSI of the same size and same technology. However, this difference can be offset by the possibility of using the most advanced technology, the possibility to fine-tune the design to individual problems, and most importantly, much shorter design cycle time.

To give an example, large FPGA chips at the time of writing (Summer 2000) have the nominal gate count of around half million, which is sufficient to implement the logic for a GRAPE-4 chip (100K gates), and the clock speed would be around 50 MHz or more. Thus, one FPGA chip can deliver 1 Gflops. This might not sound very impressive compared to 40 Gflops of GRAPE-6 chip or around 1 Gflops of present microprocessors. However, compared to microprocessors, to build a massively parallel machine out of FPGA is much easier, and we can expect higher execution efficiency, for the same reason as we achieved higher efficiency on GRAPE hardwares.

Here again, in the long run we might see the same problem as the general purpose computer. The on-chip wiring would ultimately limit the speed and the circuit density. Because of the requirement of the programmability, as the number of transistors on the FPGA chips increases, more and more fraction of these chips will be used for wiring. However, for the next several years, this limitation would not be too severe, and it is possible that some new design philosophy will allow us to make better use of FPGA chips.

It is at least possible to use FPGAs for "proof of the concept" studies, where we demonstrate that one particular custom design is actually usable and that it achieves better cost-performance than general-purpose solutions. If that demonstration was successful, the grant large enough to make custom LSI might be offered.

To summarize, the initial cost of the large custom LSI might become too high for the level of the amount of grants we can reasonably expect. However, machines based on FPGAs can be used for small projects. The cost advantage of FPGAs will not be as large as that of custom LSI chips, but compared to general-purpose microprocessors, they still offer large advantage. So we expect to see many successful projects to apply FPGAs for large-scale computing in the near future. The largest ones will be done on custom LSIs, but the rest will be done on FPGAs.

Acknowledgments

We would like to thank Daiichiro Sugimoto, Toshikazu Ebisuzaki, Makoto Taiji, Tomoyoshi Ito, Toshiyuki Fukushige and many others who were involved in the development of the six generations of GRAPE hardwares, and Yoko Funato, Simon Portegies Zwart, Steve McMillan, Piet Hut and again many others for discussions and collaborations in software development and scientific works on GRAPE hardwares. This work is supported by the Research for the Future Program of Japan Society for the Promotion of Science (JSPS-RFTP97P01102).

References

[1] J. Barnes and P. Hut, Nature 324 (1986) 446.

[2] D. J. Becker, T. Sterling, D. Savarese, J. E. Dorband, U.A. Ranawake, and C. V. Packer, in Proceedings of the 1995 International Conference on Parallel Processing (ICPP) (IEEE Comp. Soc., Philadelphia, 1995), p. 11

[3] J. H. Condon and T. Ogielski, Reviews of Scientific Instruments 56 (1985) 1691.

[4] L. Greengard and V. Rokhlin, Journal of Computational Physics 73 (1987) 325.

[5] P. Hut and J. Makino, Science 283 (1999) 501.

[6] J. Makino and M. Taiji, Scientific Simulations with Special-Purpose Computers — The GRAPE Systems, (John Wiley and Sons, Chichester, 1998).

[7] J. Makino, M. Taiji, T. Ebisuzaki and D. Sugimoto, The Astrophysical Journal 480 (1997) 432.

[8] D. Sugimoto, Y. Chikada, J. Makino, T. Ito, T. Ebisuzaki and M. Umemura, Nature 345 (1990) 33.

[9] M. Taiji, N. Ito and M. Suzuki, Reviews of Scientific Instruments 59 (1988) 2379.

[10] R. Tripiccione, in this volume (2000).