GRAPE-4: A One-Tflops Special-Purpose Computer for Astrophysical N-body Problem

Junichiro Makino*, Makoto Taiji, Toshikazu Ebisuzaki
and Daiichiro Sugimoto
*Department of Information Science and Graphics,
Department of Earth Science and Astronomy,
College of Arts and Sciences, University of Tokyo,
Tokyo 153, Japan
Email: makino@chianti.c.u-tokyo.ac.jp

Abstract

We describe the GRAPE-4 (Gravity Pipe 4) system, a special-purpose computer for astrophysical N-body simulations. In N-body simulations, most of the computing time is spent to calculate the force between particles, since the number of interactions is proportional to the square of the number of particles. For many problems the accuracy of fast algorithms such as the particle-mesh scheme is not sufficient and we have to use the straightforward direct summation.

In order to accelerate the force calculation, we have developed a series of hardwares, the GRAPE (Gravity Pipe) systems. The basic idea of our GRAPE systems is to develop a hardware specialized for the force calculation. The rest of the calculation is performed on the general-purpose computer connected to GRAPE.

The GRAPE-4 system is our newest hardware, scheduled to be completed in early 1995. Planned peak speed is 1.15 Tflops. This speed is achieved by running 1920 pipeline LSIs, each provides 600 Mflops, in parallel. A prototype system has been completed July 1994, and the full system is now under manufacturing.

[1]ISSN 1063-9535 Copyright(c) 1994 IEEE All rights reserved.

Personal use of this material is permitted. However, permissopns to reprint/republish this material for advertising or promortional purposes or for creating new collective works for resale or redistribution must be obtained from the IEEE. For information on obtaining permission, send a blank email message to info.pub.permission@ieee.org.

By choosing to view this document, you agree to all provisions of the copyright laws protecting it."

1 Introduction

Many astronomical objects are well approximated as the system of self-gravitating point-mass particles. The solar system, open clusters, globular clusters, galaxies, clusters of galaxies are all expressed as the system of N particles interacting only through gravity. The whole universe can also be regarded as such a self-gravitating N-body system.

The N-body simulation technique, in which the equations of motion of N particles are integrated numerically, has been a very useful tool for the study of self-gravitating N-body systems for the last three decades. The calculation cost of N-body simulation increases rapidly as we increase the number of particles. There are two main reasons for this rapid increase. The first reason is that the gravity is a long-range attractive force. We cannot neglect the contributions of distant particles to the force on a particle. In many other particle simulations, the force is of short-range nature, as in the case of the van der Waals force, or the contributions of the distant particles can be neglected.

The second reason is that for some systems the timescale of the evolution of the system is proportional to the number of particles. For star clusters such as open clusters and globular clusters, the thermal relaxation timescale is considerably shorter than the age of the universe. Thus the simulation of such systems must cover the thermal timescale. The ratio between the thermal timescale of the cluster and the orbital timescale of typical star in the cluster is proportional to $N/\log N$ (e.g. [23]).

Because of the above two reasons, the calculation cost of N-body simulations of star clusters is proportional to N^3. Simulations with large N is very expen-

sive. For example, a simulation of 10^5-body system would take several hundred years, if performed on a supercomputer with the effective speed of 1 Gflops [8].

For some problems, it is possible to apply the particle-mesh (PM) algorithm to calculate the potential field and then calculate the force on each particles (see, *e.g.*, [7]). However, the PM scheme is not appropriate for the simulation of star clusters, since its spatial resolution is limited by the grid size. The number density and orbital timescale of stars range over many orders of magnitude in a single cluster. As a result, the fixed resolution of the PM scheme is not appropriate. The particle-particle particle-mesh (P^3M) scheme is not appropriate either, since the calculation cost of a grid cell with high density becomes very large. The tree algorithm [5] does not help much, since the required accuracy of the force is high [21].

There is another reason why the fast algorithms are difficult to use. As stated above, the orbital timescale of particles ranges over many orders of magnitudes. Thus, a small number of particles require very short timesteps. It is not practical to use the same timestep for all particles. To overcome this difficulty, an algorithm called the individual timestep scheme has been used ([1], [2]).

In the individual timestep scheme, each particle has it own timestep Δt_i and maintains its own time t_i. To integrate the system, one first selects the particle for which the next time $(t_i + \Delta t_i)$ is the minimum. Then, one predicts its position at this new time. Positions of all other particles at this time must be predicted also. Then the force on that particle from other particles is calculated following Newton's law of gravity. The position and velocity of the particle is then corrected, and new timestep is calculated. The integration scheme is a variation of Krogh's scheme [13] modified for second-order equations.

A modification of this individual timestep algorithm is now used to achieve higher efficiency on vector machines [20] and special-purpose computers [16]. This scheme is called the hierarchical timestep scheme. In this scheme, the timesteps of particles are forced to integer powers of two. In addition, the timstep of a particle is forced to be commensurate with its time. In other words, the new timestep of a particle is chosen so that the time of the particle is an integer multiple of the timestep. These two criteria make it possible to force many particles to share exactly the same time. As a result, the efficiency of vector/parallel hardware is improved significantly.

This individual timestep scheme, sometimes referred to as the Aarseth scheme, is an extremely powerful method when applied to the late stage of the evolution of star clusters. The gain in the computational speed is $O(N)$ [18] at the late stage of evolution. This $O(N)$ gain, however, implies that only a small number of particles have small timesteps. It is difficult to apply fast algorithms such as the P^3M scheme, since these schemes are fast only when the forces on all particles are calculated at one time.

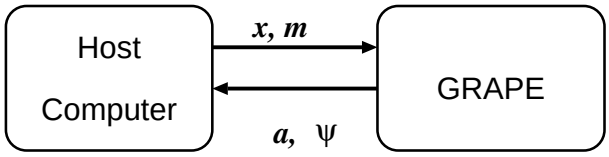

Figure 1: Basic structure of a GRAPE system

Hut et al. [8] estimated that the maximum possible speed-up factor for a sophisticated scheme such as the tree algorithm over the direct summation would be around a factor of 100. At first sight, this factor looks pretty large. However, the efficiency of parallel/vector machines is much better for the direct summation, because of its simplicity. Thus, as MPPs become more widely available, the algorithmic gain of sophisticated algorithms becomes less important.

Beside using parallel/vector machines, there is another approach to accelerate the force calculation, namely to develop the special-purpose hardware for the force calculation. The direct force calculation is well suited for acceleration by specialized hardware, because of its simplicity. We calculate the difference in the positions of two particles $\Delta \mathbf{r}_{ij}$, and calculate the Newtonian gravitational force, $-Gm_j \Delta \mathbf{r}_{ij}/|\Delta r_{ij}|^3$, where G is the gravitational constant and m_j is the mass of particle j. With the state-of-the-art VLSI technology, a fully hardwired pipeline to calculate the gravitational force can be implemented on a custom LSI chip.

This special-purpose hardware is used as a backend processor, on which only the force calculation is performed. The rest of the computation is performed on the host workstation. Figure 1 shows the basic structure of the system. We call the special-purpose hardware GRAPE (GRAvity PipE), since its main computing unit is the hardwired pipeline to calculate the gravitational force between particles. In the simplest case, the host computer sends the position and mass data of all particles to GRAPE. Then GRAPE calculates the force between particles, and sends the force on each particles back to the host computer.

From the software point of view, all program runs on the host. The special-purpose hardware is used through the calls to a few library functions, which calculate the force between particles. It is not very difficult to modify existing programs so that it uses the special-purpose hardware, since the task is the same as to utilize a software library. To develop the library functions themselves is not very difficult either, since the number of functions is very small (essentially just one).

We started developing special-purpose hardwares in 1989. Our first hardware, GRAPE-1 ([24], [9]) was completed in that year. Its peak speed was 120 Mflops. On GRAPE-1, we used a very short word length in order to simplify the hardware. In 1991, we completed GRAPE-3 [22]. In GRAPE-3, the pipeline for the force calculation is implemented on a custom LSI. Thus, we could package 48 pipelines, each providing 300 Mflops, on two boards of the size about 40 cm by 40 cm. The peak speed of GRAPE-3 was 14.4 Gflops. About 20 copies of minor-changed version of GRAPE-3, GRAPE-3A, have been manufactured and are being used in various institutes both within and outside Japan. The sustained speed we obtained on GRAPE-3A with the peak speed of 36 Gflops was 30.5 Gflops.

GRAPE-2 [10] was the first machine that used a high-accuracy floating-point number format. In GRAPE-2, we used the IEEE-754 double-precision (64 bit) format for the subtraction between the position vectors of two particles, $\Delta\mathbf{x} = \mathbf{x}_j - \mathbf{x}_i$. Then, $\Delta\mathbf{x}$ is converted to 32 bit format, and the gravitational force, $m_j\Delta\mathbf{x}/\Delta x^3$, is calculated in single-precision arithmetic. The final summation is performed in double precision, in order to avoid the underflow. The peak speed of GRAPE-2 was 40 Mflops. An enhanced version of GRAPE-2, GRAPE-2A [11] was made in 1992. Its peak speed was 180 Mflops. These machines are built using commercially available floating-point LSI chips.

In this paper, we describe GRAPE-4, a massively parallel machine with high accuracy arithmetic. In section 2, we briefly describe the Hermite integration scheme [17], for which GRAPE-4 is designed. The Hermite scheme is similar to the traditional Aarseth scheme. The main difference is that it uses the first time derivative of the acceleration explicitly calculated, in order to construct the predictor and the corrector. In section 3, we describe the hardware of GRAPE-4. In section 4, we give the estimate for the performance. In section 5, we report the present status of the project. Section 6 is for discussions.

2 The Hermite Integration Scheme

The Aarseth scheme, which has been widely used in the community of astrophysics, is a fourth-order variable-stepsize scheme, essentially same as the algorithm described in [13]. It is a linear-multistep scheme, in which the change in the position and velocity is calculated from the accelerations calculated at several previous timesteps.

The implementation of the Aarseth scheme is rather complicated, because it is a linear multistep scheme. When the time integration is started, the accelerations at previous timesteps are not available. Therefore a special procedure to initialize the integrator is required. For point-mass simulations, the initialization of the integrator occurs rather often, since we apply special procedures for close encounters and close binaries (see [2]). Thus, standard bootstrapping is not appropriate since it would significantly increase the number of timesteps.

The fourth-order Hermite integrator ([15], [17]) is much simpler than the Aarseth scheme, and yet offers a similar accuracy for the same calculation cost. The Hermite scheme (sometimes referred to as the Obrechhoff scheme), uses the Hermite interpolation formula to construct the predictor and the corrector. To construct a Hermite interpolation formula, both the values of functions and its derivatives are used. For example, if we know the values of the acceleration and its first time derivative at two points in time, we can construct a third-order interpolation polynomial. If the information of the time derivative is not available, we need values of accelerations at four different points in time to construct third-order polynomial.

The predictor of the fourth-order Hermite scheme is expressed as

$$\mathbf{x}_\mathrm{p} = \frac{\Delta t^3}{6}\dot{\mathbf{a}}_0 + \frac{\Delta t^2}{2}\mathbf{a}_0 + \Delta t\mathbf{v}_0 + \mathbf{x}_0 \quad (1)$$

$$\mathbf{v}_\mathrm{p} = \frac{\Delta t^2}{2}\dot{\mathbf{a}}_0 + \Delta t\mathbf{a}_0 + \mathbf{v}_0, \quad (2)$$

where $\mathbf{x}_\mathrm{p}$ and $\mathbf{v}_\mathrm{p}$ are the predicted position and velocity; $\mathbf{x}_0$, $\mathbf{v}_0$, $\mathbf{a}_0$ and $\dot{\mathbf{a}}_0$ are the position, velocity, acceleration and its time derivative at time t_0; and Δt is the timestep.

The acceleration $\mathbf{a}$ and its time derivative $\dot{\mathbf{a}}$ are calculated as

$$\mathbf{a}_i = \sum_j Gm_j \frac{\mathbf{r}_{ij}}{(r_{ij}^2 + \epsilon^2)^{3/2}} \quad (3)$$

$$\dot{\mathbf{a}}_i = \sum_j Gm_j \left[\frac{\mathbf{v}_{ij}}{(r_{ij}^2 + \epsilon^2)^{3/2}} - \frac{3(\mathbf{v}_{ij}\cdot\mathbf{r}_{ij})\mathbf{r}_{ij}}{(r_{ij}^2 + \epsilon^2)^{5/2}} \right], \quad (4)$$

where

$$\mathbf{r}_{ij} = \mathbf{x}_j - \mathbf{x}_i, \quad (5)$$

$$\mathbf{v}_{ij} = \mathbf{v}_j - \mathbf{v}_i. \quad (6)$$

Here, ϵ is the softening parameter.

The corrector is given by the following formula (see, *e.g.*, [14])

$$\mathbf{x}_c = \mathbf{x}_0 + \frac{\Delta t}{2}(\mathbf{v}_c + \mathbf{v}_0) - \frac{\Delta t^2}{12}(\mathbf{a}_1 - \mathbf{a}_0), \quad (7)$$

$$\mathbf{v}_c = \mathbf{v}_0 + \frac{\Delta t}{2}(\mathbf{a}_1 + \mathbf{a}_0) - \frac{\Delta t^2}{12}(\dot{\mathbf{a}}_1 - \dot{\mathbf{a}}_0). \quad (8)$$

The predictor given in equation (1) uses the position, velocity, acceleration and its derivative at time t. The acceleration and its time derivative are calculated from the position and velocity. Therefore, no information concerning the previous timestep is necessary. The corrector also requires information of the present and old timesteps only. Thus, the fourth-order Hermite scheme is a one-step, self-starting scheme. It is particularly suitable for use with the individual timestep scheme.

Makino [15] as well as Makino and Aarseth [17] made detailed comparisons between the Aarseth-type schemes and Hermite schemes of various orders. Their conclusion can be summarized as follows. First, the Hermite scheme allows a timestep that is roughly twice as long as that of the Aarseth scheme of the same order for the same accuracy. Second, the fourth-order scheme is close to optimal for a wide range of required accuracy. In short, the fourth-order Hermite scheme is better than the Aarseth scheme, regarding accuracy, efficiency, and simplicity of the algorithm.

3 The GRAPE-4 system

3.1 Architecture

Figure 2 shows the overview of the GRAPE-4 system. The GRAPE-4 system consists of a host computer and five clusters. One cluster has one host-interface board, one control board, and eight processor boards. The total number of processor boards is thus 40. Each processor board houses 48 HARP (Hermite AccceleRator Pipeline) chips, which are custom LSI chips to calculate the gravitational force and its first time derivative. It also has one PROMETHEUS chip, another custom LSI to calculate the predicted positions and velocities of particles at a given time.

The theoretical peak speed of GRAPE-4 is 1.15 Tflops, for the system clock of 15 MHz. We hope the total system will be able to operate at the clock period of 20 MHz, in which case the peak speed will be 1.54 Tflops. In the following, we describe each components in some details.

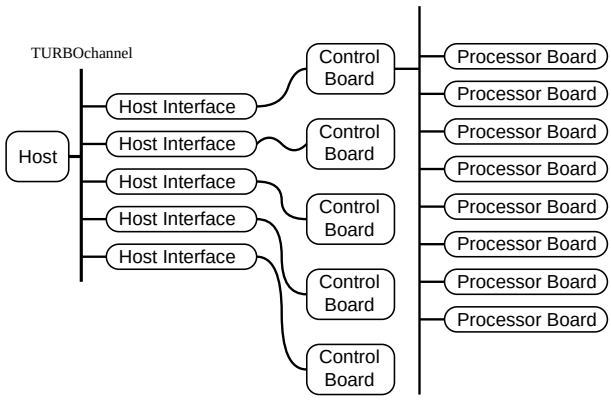

Figure 2: The GRAPE-4 system

3.2 The host computer

We use a DEC Alpha AXP workstation as the host computer. Currently, we are using a DEC AXP 3000/400 as the platform to develop the hardware and software. We plan to use the fastest desktop machine that will be available in this year, which will be $2 \sim 3$ times faster than our present host computer.

We selected the host taking into account the following considerations. First, its scalar computational speed should be as fast as possible. At present, this is achieved by selecting one of high-end RISC workstations. Second, the I/O bus must be reasonably fast, and the latency of I/O bus must be as small as possible. The DEC Alpha AXP systems offers the TURBOchannel, which is not the fastest I/O bus available but reasonably fast. The Alpha machines with TURBOchannel I/O buses do not have separate I/O processors. Therefore the latency of an I/O operation is short. Third, it is desirable that the development of the device driver is easy, since no one in our group is particularly good at hacking the operating system. The UNIX operating system from DEC is reasonably stable and development of the device driver is not very hard.

3.3 The host interface board

The most important function of the host interface board is to extend the I/O bus of the host. The phys-

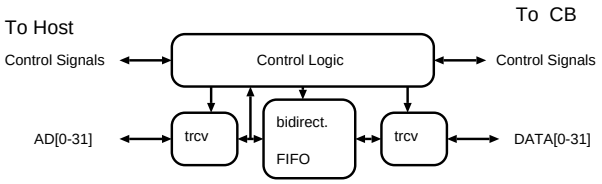

Figure 3: Host interface board

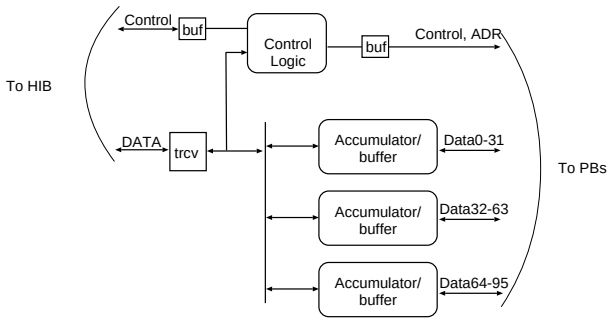

Figure 4: Control board

ical size of the I/O cards of modern desktop workstations is rather small. In order to connect a larger system, we need a small card to be inserted to the I/O bus. In addition, this interface board converts the data transfer protocol of the host I/O bus to the protocol used for the link between the host interface board and the control board. The protocol on the link is designed so that it does not depend on the protocol of the host I/O bus. Because of this structure, we can connect GRAPE-4 to different host computers by replacing the host interface board.

In the present design, the host-interface board is connected to the host by TURBOchannel, the bus used in desktop workstations from DEC. It's peak transfer rate is 100 MB/s, when DMA transfer is used. The host interface board and the control board are connected with a 32 bit parallel interface. We adopted a coaxial flat cable for the physical connection between them. The maximum length of the cable is 1 m. The transfer speed between the host interface board and the control board is 60 MB/s, when the system clock is 15 MHz.

Figure 3 shows the structure of the host interface board. It consists of the data transceivers (trcv) to exchange data with the host and the control board, the FIFO (first-in first-out) unit to buffer the data, and the control logic. The size of FIFO unit is 2048 32-bit words (8kbytes). The data transfer rate between the host interface unit and the control unit is slower than that between the host and the host interface board. Using the FIFO buffer, the host computer can send the data at the peak transfer speed. In addition, there are several restrictions that the host interface board must obey, concerning the DMA transfer on the TURBOchannel bus. The FIFO buffer allows the control board to transfer data without checking the status of the host bus. As a result, the design of the control board becomes independent of the host bus.

The control logic is made of three 22V10 PLDs and one AMD MACH230 complex PLD. The AMD MACH chip integrates most of the control logics such as the DMA sequence controller, DMA word address counter etc. Other PLD chips perform timing-critical handshake operations on both the TURBOchannel and the processor-board interface. We used Cypress CY7C453-14 clocked FIFO chips for the data FIFO. This clocked FIFO chip accepts separate clock signals for read and write data paths. The interface to TURBOchannel operates on the TURBOchannel clock, while the interface to the control board operates the clock provided by the processor board.

3.4 The control board

The control board has two main functions. The first one is to distribute the data received from the host computer to processor boards. The second one is to sum up the calculated force and potential from processor boards. The summed result is transferred to the host computer through the host interface board. This summation process is necessary to reduce the bandwidth of the communication with the host computer [19].

Consider the case that we calculate the force from N particles to M particles. First, the host computer sends the position (and other) data of all N particles to the control board. The control board distribute the data to processor boards, according to the indices of particles. Each processor board takes care of $N/8$ particles. Then, the host computer sends the position and velocities of M particles, in blocks of 96 particles. Each of the processor boards can calculates the forces on 96 different particles, from the same set of particles. The control board broadcasts the data of these 96 particles to all processor boards. Therefore, all processor boards calculate the forces on the same 96 particles, from different set of particles. After all processor boards finished the calculation, the control board sums up the partial forces calculated on proces-

sor boards, and sends the result to the host computer.

The internal structure of the cluster is not visible to the application program. The host computer sends the data of 96 particles, and receives the calculated results for these 96 particles. The number of processor boards in one cluster changes the speed of computation, but nothing else.

In order to distribute the computation to different clusters, we use the algorithm same as that used for different processor boards in one cluster. If we have five clusters, the host sends $N/5$ particles to each cluster, where N is the total number of particles that exert the force. In this case, each processor board takes care of the contribution of $N/40$ particles. Then, each cluster calculate the forces on the same 96 particles. The calculated result is summed up on the host computer.

The control board and processor boards are connected by a backplane bus (the HARP bus) with 96 bit data width. A synchronous, pipelined protocol with a fixed latency is used on this HARP bus.

Figure 4 shows the structure of the control board. It consists of the control logic unit, three accumulator/buffer units, and several transceivers and buffers. The control unit generates all necessary signals to control the HARP bus. The 96-bit data bus is divided into three 32-bit subbuses, each of which is connected to different accumulator/buffer unit. The accumulator/buffer unit contains a 64-bit floating-point ALU, which accumulates the result calculated on processor boards.

The control logic unit consists of three AMD MACH230 complex PLDs. The accumulator/buffer unit consists of a TI 74ACT8847 64bit floating-point LSI chip and transceivers, buffers and FIFO chips.

3.5 Processor board

Figure 5 shows the structure of a processor board. The particle data memory stores the data of particles which exert the force. The PROMETHEUS LSI is used to calculate the position (and velocity) of particles at a specified time. This function is necessary to use the hierarchical timestep algorithm described in the introduction. The HARP LSI chips calculate the gravitational accelerations and their first time derivatives for particles. One board has 48 HARP chips.

3.6 HARP chip

Figure 6 shows the architecture of the HARP chip. It is simply a fully-pipelined hardware implementation of the formulae (3) and (4). We adopted an architecture in which the x, y and z components of all vector

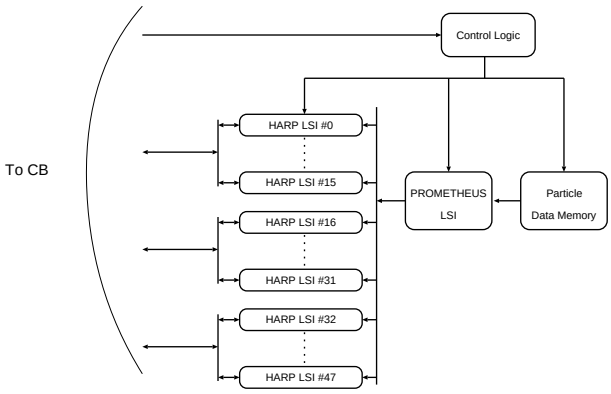

Figure 5: Processor board of GRAPE-4

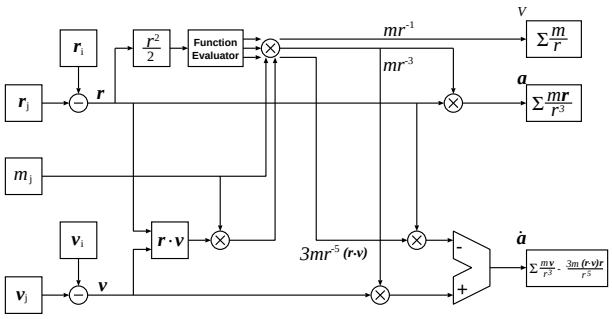

Figure 6: The block diagram of the HARP chip.

quantities are processed sequentially, in order to reduce the gate count. Thus, it takes three clock periods to calculate one interaction.

Each chip calculates the forces on two particles, using the "virtual multiple pipeline" (VMP). With VMP, the clock period of the pipeline LSI is two times as that of the system clock, and it calculates the forces on two different particles at alternate clock cycles. From the outside, one force calculation chip looks as if it has two pipelines. The advantage of this architecture is that we can increase the performance of the pipeline chip without increasing the system clock cycle.

This approach is conceptually similar to what is used in "superpipeline" chips such as MIPS R4k and "clock-doubled" chips such as Intel i486DX2. In our VMP architecture, however, the chip looks as if it really has two separate pipelines. In particular, the chip actually has two separate set of registers. On the other hand, in the case of the chips such as i486DX2, the CPU still looks as one CPU. Only the cycle time of

the external memory bus is reduced to a fraction of the internal clock cycle.

The clock-doubling in i486DX2 causes some performance penalty, since the relative speed of the main memory becomes slower. Thus, it is not practical to increase further the ratio between the internal and external clock, unless the width of the memory bus is increased.

In the case of VMP, we have no such performance penalty, since all virtual pipelines share the same input data. As a result, it is not very difficult to increase the number of virtual pipelines. In our architecture, it is also easy to have real multiple pipelines in one chip, when larger number of gates becomes available. In the present HARP chip, the number of VMPs is two because the external clock speed of 15 MHz is already sufficiently low to make the design of the processor board easy.

The number format used in the HARP chip is essentially same as that used in HARP-1. HARP-1 [12] is a machine made of off-the-shelf floating-point LSI chips. The subtraction of the position vectors and the accumulation of the calculated accelerations are performed in 64-bit floating-point format. Other calculations to obtain the acceleration is performed in 32-bit format. The subtraction of velocity vectors and the accumulation of the time derivatives are performed in 38 bit format. Other calculations to obtain the time derivative are performed in 29-bit format, in which the length of mantissa is 20 bits.

The HARP chip is cell-based. It is fabricated by LSI Logic Corp. The design rule is 1 μm and the gate count is about 100k. The die size is 14.2 mm by 14.2 mm. The worst-case clock cycle of the chip is 30 MHz. We received the engineering sample of the HARP chip in August 1993. All the sample chips operated successfully with the clock speed up to 50 MHz.

Eight HARP chips are packaged in an MCM (multi-chip module) package manufactured by Kyocera. As shown in figure 5, 16 HARP chips share the same data bus. Therefore, all eight HARP chips in a MCM shares the same input data bus and I/O bus. This design is well-suited for a MCM package because several difficult problems with MCM packages do not arise.

In general, MCMs have two difficult problems, testability and yield. The test of an MCM is difficult to design, because some of the signals cannot be set/sensed from outside the module. In the case of HARP MCM, there is no such difficulty, since all the I/O pins of HARP chips can be directly accessed from outside. Therefore the test procedure for an MCM is not much different from that for a single chip.

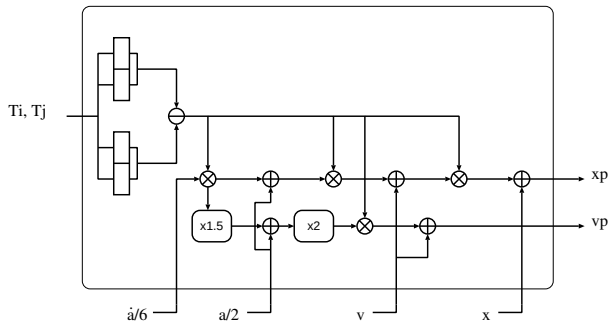

Figure 7: The block diagram of the PROMETHEUS chip.

The yield of an MCM tends to be low, since the probability to get a working MCM is the product of probabilities to get working chips for all chips. For example, if the yield of a single chip is 90%, the yield of an MCM with 8 chips would be $(0.9)^8 = 43\%$. We expect the yield of the single chip after DC test to be around 98%. In this case, the yield of the MCM is still higher that 80%. In other words, the yield is not a severe problem. For the case if the yield turns out to be much lower than our expectation, we plan to use one of 8 chips as spare, and use 7 chips on each MCM. The probability that more than one chip is faulty is about 20%, if the yield of the single chip is 90%. Thus, we do not have to require a very high yield for single chips.

3.7 PROMETHEUS chip

The architecture of the PROMETHEUS chip is shown in figure 7. It is a straightforward hardwired implementation of the predictor formulae (1) and (2). The subtraction of time and the addition of $\mathbf{x}_j$ and the higher order term are performed in 64-bit format. All other calculations are performed in 32-bit format. The PROMETHEUS chip handles x, y, and z components sequentially, in the same way as the HARP chip.

The PROMETHEUS chip is a 1 μm CMOS gate array. It is fabricated by LSI Logic Corp. The raw gate count is about 182k. The actual used gates is about 60k. It is packaged into a 391-pin CPGA package.

4 Performance estimate

In this section, we give an estimate for the speed of the GRAPE-4 system, for simulations of the thermal evolution of globular clusters. This is a worst-case estimate of the performance, because simulations of this type require very long integration. The number of particles must be small, because the number of timesteps is very large. The performance of a GRAPE is higher for larger number of particles, because of Amdahl's law.

Conceptually, the computing time to integrate one particle on GRAPE-4 system is given as follows

$$T_{\rm step} = T_{\rm host} + T_{\rm grape} + T_{\rm comm}, \tag{9}$$

where $T_{\rm host}$, $T_{\rm grape}$, and $T_{\rm comm}$ are the time to integrate the orbit of one particle on host, the time to calculate the force on one particle on GRAPE-4, and the time to transfer data between the host and GRAPE-4, respectively. In reality, the actual time is shorter than this estimate, because calculation on GRAPE, calculation on the host, and communication are partially overlapped. We first give the estimate for non-overlapping execution, and then discuss the effect of the overlapping.

The three terms in the right-hand side of equation (9) are expressed as

$$T_{\rm host} = 150 t_{\rm host}, \tag{10}$$

$$T_{\rm grape} = \frac{3N t_{\rm pipe}}{48 n_{\rm cl} n_{\rm pr}}, \tag{11}$$

$$T_{\rm comm} = (34 + 20 n_{\rm cl}) t_{\rm comm1}. \tag{12}$$

Here, $t_{\rm host}$ is the average time for the host computer to perform one floating-point operation. For the Hermite scheme, the total number of floating-point operations per particle per timestep is $100 \sim 200$. The speed of the host computer would be around 50 Mflops. We thus estimate that $t_{\rm host} = 2 \times 10^{-8}$sec. In equation (11), N is the number of particles in the system, $t_{\rm pipe}$ is the cycle time of the HARP chip, and $n_{\rm cl}$ and $n_{\rm pr}$ denote the number of clusters and the number of processor boards per cluster. We assume that $t_{\rm pipe} = 3.33 \times 10^{-8}$sec, $n_{\rm cl} = 5$, and $n_{\rm pr} = 8$. In other words, the clock cycle of HARP chips is 30 MHz, the number of cluster is 5, and the number of processor board per cluster is 8. The total number of HARP chips is 1920 and the nominal peak performance is 1.15 Tflops. In equation (12), $t_{\rm comm1}$ is the time to transfer one word (4 bytes) between the host and a control board. The number of words to be transferred between the host and a control board is $34 + 20 n_{\rm cl}$, in the present implementation of the control board. We assume that $t_{\rm comm1} = 4 \times 10^{-8}$sec. This is the burst transfer rate of TURBOchannel DMA. We confirmed that this speed is actually obtained on the prototype system.

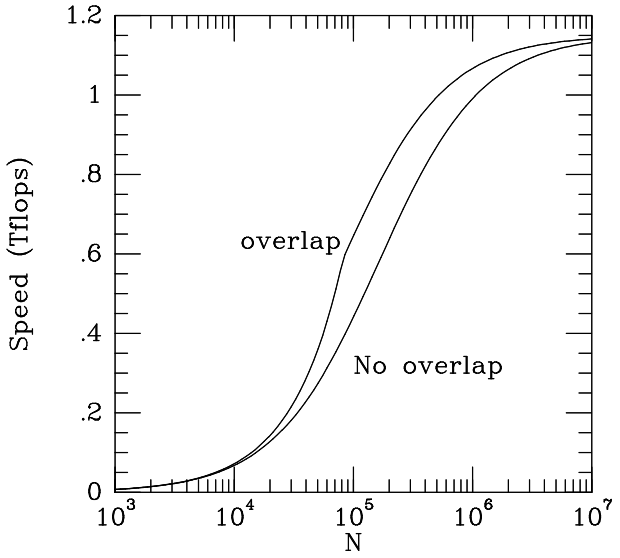

Figure 8: The computing speed of the GRAPE-4 system as a function of the number of particles N. Unit is Tflops (10^{12} floating-point operations per second). The upper curve is obtained assuming partial overlapping of the communication and the calculation on GRAPE.

Figure 8 gives the estimated computing speed as a function of the number of particles in the system. Here, the computing speed is calculated as

$$R_{\rm eff} = \eta R_{\rm peak}, \tag{13}$$

where $R_{\rm peak}$ is the peak speed of GRAPE-4, which is attained when all HARP chips are working at peak speed, η is the efficiency defined as

$$\eta = \frac{T_{\rm grape}}{T_{\rm step}}. \tag{14}$$

In the case of the non-overlapping execution, half of the peak performance is achieved for $N \sim 2 \times 10^5$. In the following, we investigate how we can improve the performance by overlapping the computation and communication.

As discussed by Okumura et al. [22], it is relatively easy to overlap the communication and the computation on GRAPE, if we have more than two clusters. The host computer first send the necessary data to

the first cluster and start the calculation on the first cluster, then the host sends the necessary data to the second cluster. While the host is sending data to the second cluster, the first cluster is performing computation. In the case of the two-cluster system, if the calculation time is less than 50% of the communication time, the calculation time is completely hidden in the background of the communication time.

Here we consider the effect of overlapping the communication and the calculation on GRAPE. The calculation on the host computer can also be overlapped to some extent. We neglected this effect here since it is difficult to predict accurately. The total time per one particle per timestep is now expressed as

$$T_{\rm step} = T_{\rm host} + T_{\rm comm} + \max\left[0, T_{\rm grape} - \frac{n_{\rm cl}-1}{n_{\rm cl}} T_{\rm comm}\right]. \quad (15)$$

Figure 8 gives the estimated computing speed for this overlapped case also. In this case, half of the peak speed is achieved for $N \sim 8 \times 10^4$.

5 Project status

We started the GRAPE-4 project in summer 1992. The engineering samples of the HARP and PROMETHEUS chips was completed in July 1993. The test board which houses one PROMETHEUS and two HARP chips was made August 1993, and both chips proved to operate as designed. Both chips operated normally with the system clock speed of up to 25 MHz (HARP chip clock of 50 MHz). The prototype system which consists of one host interface board, one control board and one processor board has been completed and is working as designed.

6 Discussion

6.1 Applications to other N-body problems

The N-body simulation is being used in various fields of computational sciences. One of the most well-known example is the molecular dynamics, in which the trajectories of atoms are simulated in order to study the behavior of materials. From the point of view of the simulation algorithms, there are two important differences between astrophysical N-body simulations and molecular dynamics simulations. First, the force laws are different. In astrophysics, we deal with the Newtonian gravity. In MD simulations, we have to deal with several different force laws. If we want to calculate the Coulomb force using the Ewald method, we must calculate $1/r$ potential with Gaussian cutoff. For van der Waals force, we use Lennard-Jones potential or other empirical formulae.

The GRAPE-2A system ([11]) was designed so that it could evaluate the forces with an arbitrary functional form, using the table lookup and linear interpolation. We used a RAM table so that the content of the table can be downloaded from the host computer. On GRAPE-4, we did not use this approach since the number of transistors was severely limited. We are currently developing the GRAPE-6 system, in which the force calculation pipeline of GRAPE-2A is implemented in a single custom LSI chip. This change in the design is made possible by using 0.6μm cell-based technology.

6.2 Comparison with other special-purpose machines

Our GRAPE project is not the first project to develop special-purpose computers for N-body simulations. Bakker et al. ([3], [4]) developed DMDP, and Fine et al. [6] developed FASTRUN. These machines are specialized to MD simulations.

DMDP and FASTRUN have similar architectures. Both have the force calculation pipeline similar to that in our GRAPE-2A. The peak speed is also similar (100-200 Mflops). However, both of DMDP and FASTRUN have additional hardware for time integration of particles. Moreover, DMDP has additional hardware to implement the linked-list algorithm, while FASTRUN has hardware for the neighbor-list algorithm. As a result, both machines are very large, comprising 10-20 large printed-circuit boards. In addition, the time needed to develop the hardware was rather long ($\sim$ 5 years). GRAPE-2A was developed in roughly 8 months. Even the GRAPE-4 system will take less than three years.

The long development time has a strong negative impact on the performance of any computer, since the speed of general-purpose computer doubles roughly in every one and half years. Thus, five years corresponds to about a factor of 10 in performance.

The principal reason for the difference in the development time is the complexity of the hardware. In our GRAPE system, the special-purpose hardware performs only the force calculation. On the other hand, in both DMDP and FASTRUN, virtually everything is performed on the special-purpose hardware.

At the time these machines were designed, the choice to do everything in hardware might be justified,

since the price of the minicomputers then was orders of magnitude higher than the price of fast workstations now. We, however, strongly feel that the lifetime of these machines would be longer if they are designed to work in cooperation with the host computer. The design of the hardware would be much simpler and therefore faster, the cost of the hardware would be cheaper, and the total performance of the system would be improved as the faster host computer become available.

References

[1] S. J. Aarseth, *Monthly Notices Roy. Astron. Soc.*, Vol. 126, p. 223, 1963.

[2] S. J. Aarseth, in *Multiple Time Scales*, ed. J. U. Brackhill and B. I. Cohen, Academic, New York, p. 377, 1985.

[3] A. F. Bakker, C. Bruin, F. van Dieren, and H. J. Hilhorst, *Phys. Lett.*, Vol. 93A, p. 67, 1982.

[4] A. F. Bakker and C. Bruin, in *Special Purpose Computers*, ed. Berni J. Alder, Academic Press, p. 183, 1988.

[5] J. E. Barnes and P. Hut, *Nature* Vol. 324, p. 446, 1986.

[6] R. Fine, G. Dimmer and C. Levinthal, *Proteins*, Vol. 11, p. 242, 1991.

[7] R. W. Hockney and J. W. Eastwood, *Computer Simulation Using Particles*, Adam Hilger, New York, 1988.

[8] P. Hut, J. Makino and S. L. W. McMillan, *Nature*, Vol. 336, p. 31, 1988.

[9] T. Ito, J. Makino, T. Ebisuzaki and D. Sugimoto, *Computer Physics Communications*, Vol. 60, p. 187, 1990.

[10] T. Ito, T. Ebisuzaki, J. Makino and D. Sugimoto, *Publ. Astron. Soc. Japan*, Vol. 43, p. 547, 1991.

[11] T. Ito, J. Makino, T. Fukushige, T. Ebisuzaki, S. K. Okumura, and D. Sugimoto, *Publ. Astron. Soc. Japan*, Vol. 45, p. 339, 1993.

[12] E. Kokubo, J. Makino, and M. Taiji, in *Proceedings of the 27th Annual Hawaii International Conference on System Sciences*, ed. T. N. Mudge and B.D. Shriver, IEEE Computer Society Press, vol. 1, p. 292, 1994.

[13] F. T. Krogh, in *Proceedings of the Conference on the Numerical Solution of Ordinary Differential Equations*, ed. D.G. Bettis, Springer, p. 22, 1974.

[14] J. D. Lambert, *Computational Methods in Ordinary Differential Equations*, John Wiley and Sons, 1973.

[15] J. Makino, *Astrophys. J.*, Vol. 369, p. 200, 1991.

[16] J. Makino, *Publ. Astron. Soc. Japan*, Vol. 43, p. 859, 1991.

[17] J. Makino and S. J. Aarseth, *Publ. Astron. Soc. Japan*, Vol. 44, p. 141, 1992.

[18] J. Makino and P. Hut, *Astrophys. J. Suppl.*, Vol. 68, p. 833, 1988.

[19] J. Makino, E. Kokubo and M. Taiji, *Publ. Astron. Soc. J.*, Vol. 45, p. 349, 1993.

[20] S. L. W. McMillan, in *The Use of Supercomputers in Stellar Dynamics*, eds. S. McMillan and P. Hut, Springer, p. 156, 1986.

[21] S. L. W. McMillan and S. J. Aarseth, *Astrophys. J.*, Vol. 414, p. 200, 1993.

[22] S. K. Okumura, J. Makino, T. Ebisuzaki, T. Ito, T. Fukushige, D. Sugimoto, E. Hashimoto, K. Tomida and N. Miyakawa, *Publ. Astron. Soc. J.*, Vol. p. 45, 329, 1993.

[23] L. Spitzer,, *Dynamical Evolution of Globular Clusters*, Princeton University Press, 1987.

[24] D. Sugimoto, Y. Chikada, J. Makino, T. Ito, T. Ebisuzaki and M. Umemura, *Nature*, Vol. 345, p. 33, 1990.