Special-Purpose Computer for Molecular Dynamics Simulations

分子動力学シミュレーション専用計算機の開発

Tetsu Narumi
成見　哲

THESIS

Special-Purpose Computer for Molecular Dynamics Simulations

Tetsu Narumi

Department of General Systems Studies,
College of Arts and Sciences,
University of Tokyo

December, 1997

Abstract

The author describes the system design of Molecular Dynamics Machine (MDM), which is a special-purpose computer for large-scale molecular dynamics simulations. Molecular Dynamics simulations have been widely used to study the physical properties of condensed matter in atomic level. MD simulations treat atoms as classical particles. An atom interacts with the other atoms through bonding force, Coulomb force, and van der Waals force.

The bottleneck of MD simulations is the calculation cost for non-bonding forces, such as Coulomb and van der Waals forces. In particular, almost all computation time is consumed in the calculation of Coulomb force. When periodic boundary condition is imposed, Ewald method is used in the calculation of Coulomb force. In Ewald method, the calculation of Coulomb force is divided into real-space and wavenumber-space parts.

MDM has a special-purpose computer to calculate non-bonding forces at high speed. MDM is composed of MDGRAPE-2, WINE-2, and a host computer. MDGRAPE-2 calculates van der Waals force and real-space part of Coulomb force. MDGRAPE-2 consists of 2560 MDGRAPE-2 chips. WINE-2 calculates wavenumber-space part of Coulomb force. WINE-2 consists of 3072 WINE-2 chips. The host computer calculates bonding-force and updates positions of atoms.

The author designed both of MDGRAPE-2 and WINE-2 chips. An MDGRAPE-2 chip has four pipelines to calculate arbitrary central forces, and is twelve times faster than an MD-GRAPE chip (Taiji et al. 1994). A WINE-2 chip has eight pipelines to perform Discrete Fourier Transformation (DFT) and Inverse DFT, which are used in the calculation of wavenumber-space part of Coulomb force, and is fifty times faster than an MD-GRAPE chip. Engineering samples of MDGRAPE-2 and WINE-2 chips will be shipped at the third and the first quarters of 1998, respectively.

The author also designed overall system of MDM and estimated its effective performance. He found that MDM calculates 10^6 time-steps of MD simulation with a million particles in about a day ($\sim 10^5$ seconds). Its peak speed is about 100 Tflops and will sustain one-third of a peak performance in the simulation with one million atoms. Total system of MDM will be completed in 1999.

The author compares MDM with other special-purpose computers and general-purpose supercomputers, estimates hardware specification to realize a time-span of micro-seconds for the systems with a hundred thousands atoms, and discusses a faster algorithms than Ewald method. He also discusses other applications of MDM than MD simulations.

Acknowledgments

It is my great pleasure to thank my adviser, Prof. T. Ebisuzaki, for suggesting me to work on this subject, and for his kind advice, continuous encouragement and stimulating discussions. His positive attitude toward problems especially made me carry forward my work. I am also grateful to Prof. Makino at University of Tokyo for teaching me most of what I know of GRAPE system.

I am deeply indebted to many teachers and collaborators: Prof. Sugimoto at University of Air has organized the GRAPE project. Dr. Fukushige at University of Tokyo has taught me many things not only the hardware design on MD-GRAPE but also strategies on research. Prof. Taiji at Institute of Mathematical Statistics has taught me many things of the hardware design including the details of the MD-GRAPE chip. Drs. Sunouchi and Susukita at Institute of Physical and Chemical Research (RIKEN) have helped me on the specification of MDGRAPE-2 chip. Drs Tateno and Yasuoka have taught me many things on molecular dynamics simulations. Thanks are also due to Drs. G. McNiven and B. Elmegreen at IBM Watson Research Center for stimulating discussions including the design of MDGRAPE-2 chip and warm hospitality during my visit to York Town, NY.

For their advice to me in my graduate school, I express my thanks to Prof. Eriguchi, Prof. Hachisu, Dr. Ueno, Dr. Funato, Dr. Kokubo, Dr. Wada, Dr. Nozawa at University of Tokyo, Dr. Sumiyoshi, Dr. Miura, Dr. Tanimura, Mr. Kawai, Dr. Xiao, Dr. Giovanni, Dr. Shimizu, Dr. Ohno, Dr. Shigetani, Mr. Turuoka and Mr. Yoshida at RIKEN. I list with special thanks the names of people who influenced and helped me for my thesis work: Mrs. Fujimori, Mrs. Hisaga, Mrs. Yamamoto, Mr. Masuda, Mr. Kawai, Miss Yoshinaga, Mr. Kawaguchi, Mr. Hamada at University of Tokyo, Mrs. Ohata, Mrs. Saitoh, Mr. Furusawa at RIKEN, and Mr. Yoshizawa at Tohoku University. I express my thanks to Mr. and Mrs. Saito, Kabu, and my family for providing me with comfortable environment and support.

I acknowledge the support by the Fellowships of the Japan Society for the Promotion of Science for Japanese Junior Scientists.

Finally I thank my dear wife, Chie, for her continuous encouragement and support.

Contents

Chapter 1

Introduction

Molecular Dynamics (MD) simulations have been widely used to study the physical properties of condensed matter at an atomic level. MD simulations treat atoms as classical particles. An atom interacts with the other atoms through bonding force, Coulomb force, and van der Waals force. Bonding forces, caused by quantum electro-dynamical effects of electrons, are treated as classical springs connecting two atoms in MD simulations. MD simulations give us many microscopic properties of structures and dynamics, which are usually difficult to be obtained by experimental methods. Their application spreads to many fields, in particular, life and material sciences.

MD simulations, which give us information on structures and the folding processes of proteins and nucleic acid, have been used to study in many fields of life sciences, such as structural biology, enzyme kinetics (Warshel 1991), protein engineering (Fasman 1989) and drug design (Burt et al. 1989) over the past decade. These biomolecules have strongly attracted attentions of researchers of life science, because they are the clues to the understanding of enzymic reactions in biological bodies.

Structures of proteins are believed to be strongly related to their functions as enzymes. The structures of proteins were determined by X-ray crystallography and Nuclear Magnetic Resonance (NMR) experiments, so far. X-ray crystallography can determine the structure of protein molecule with a molecular weight of one million. However, it requires crystals of proteins, which are generally difficult to form; in fact, crystallization process is the major limiting factor of X-ray crystallography of proteins. On the other hand, NMR, which dose not require a crystal, is difficult to analyze molecules larger than 30,000 in molecular weight, since a dense solution of such large proteins are difficult to form.

MD simulations can overcome these experimental difficulties in structural biology. They enhance the ability of these experimental method; Structures of many proteins belonging in a family are determined by MD simulations, if one of them are once determined by an experiment. For example, Sugita et al. (1998) studied the thermal stability of chymotrypsin inhibitor 2 in the case of valine to alanine mutation of the 57-th residue.

Folding process of proteins is one of the most important topics in life sciences. Statistical analysis reveals that folding time-scale of a protein molecule is much faster than that expected by a naive random search in the conformational phase space (Levinthal 1968):

It is evident that there is a pathway to lead the native conformation from the random coil state. For example, experiments, such as proton exchange NMR (Roder and Wuthrich 1986; Radford et al. 1992) and mutagenesis (Garvey et al. 1989; Sancho et al. 1992), showed that some protein sequences containing fragments of 10 to 20 amino acids quickly fold to their own secondary structure, independent of tertiary structures of the rest of the protein. MD simulations have greatly contributed on the mechanism of such initialization of helix formation and helix stability in an aqueous environment (DiCapua et al. 1990; VanBuuren and Berendsen 1993).

MD simulations also contribute to the advances in material sciences. For example, phase behaviors of supercooled water are studied by MD simulations taking into account hydrogen bond networks. Tanaka et al. (1996) confirmed the phase transition from normal water to water II with a lower density in super cooled water by molecular dynamics simulations at constant pressure and temperature.

Yasuoka et al (1998) studied homogeneous nucleation from super saturated vapor to liquid droplets by MD simulations. They found that nucleation rate is two orders of magnitude higher than that predicted by nucleation theory in the case of Lennard-Jones fluid and three orders of magnitude lower than the theoretical prediction in the case of water. The results of MD simulations are much closer to experimental results than theoretical prediction. They found that droplets are by no means spherical: all the theory of nucleation assumed the spherisity of droplets. This non-spherical nature could be a cause of the large differences in nucleation rate between their calculations and theoretical prediction. They also suggested that the size dependence of surface tension is also important to understand this contradiction.

There are strong motivations to perform MD simulations with a larger number of atoms to study larger molecular systems. In recent years, the structures of large biomolecules in the range of 10,000 and 100,000 atoms have been determined. Those are photosynthetic reaction center (Deisenhofer et al. 1985; Feher et al. 1988), ATPase (Abrahams et al. 1994), and cytochrome oxidase (Iwata et al. 1995). Furthermore, biomolecular systems consisting of many molecules have become targets of simulations, for example, lipid bilayer (Pastor and Venable 1993; Heller et al. 1993; Zhou and Schulten 1995), membrane-protein complexes (Zhou and Schulten 1996), F-actin (Lorenz et al. 1993; Tirion et al. 1995), and protein-DNA complexes (Beveridge and Ravishanker 1994; Eriksson et al. 1995; Harris et al. 1994). These biomolecule systems are often required to be surrounded by water molecules. In order to study the structures and dynamics of these large systems of molecules, simulations with million atoms are required. Moreover, investigating phenomena, such as the stability of biomolecules on the structure obtained by experiments, at least several hundreds of pico-seconds of MD simulations are often required.

However, the number of atoms included in MD simulations with a time span of nano-seconds are strictly limited to several tens of thousands (figure 1.1) due to the calculation cost for non-bonding forces. In MD simulations, almost all computational time is consumed in the calculation of Coulomb and van der Waals forces, since the number of the particles which interact through non-bonding forces tends to be much larger compared with that through bonding forces. In particular, the calculation cost for Coulomb force increases

in proportion to square of the number of particles in a system, when you use the naive direct summation method. When Coulomb force is not truncated, the number of atoms in the system is limited to about several tens of thousands even if you use the currently most powerful supercomputers. For example, when the number of atoms, N, is a hundred thousand (10^5), the number of pair-wise interactions per time-step is 10^{10} with direct summation method, because the calculation cost of direct summation is $O(N^2)$. Since it takes about 30 floating point operations to calculate a pair-wise Coulomb force, required calculation speed is about 350 Gflops to simulate 10^6 time-steps ($\sim$1 nsec) within ten days. Because of this large computational cost, Coulomb force is usually truncated at a length of $10\mathring{A} \sim 15\mathring{A}$. However, this truncation leads a very large computational error ($\sim$ several tens of kcal/mol) in free energy (Saito 1995). Even if we use a fast algorithm instead of direct summation, such as tree-code (Barnes et al. 1986) or P^3M (Particle-Particle Particle-Mesh; Eastwood et al. 1980) method, the cost for Coulomb force still dominates the computing time.

This difficulty can be overcome by a super-high-speed special-purpose computer for molecular dynamics simulations. In fact, several machines, such as Delft Molecular Dynamics Processor (DMDP; Bakker et al. 1988), ATOMS (Bakker et al. 1990), FASTRUN (Fine et al. 1991) were developed for the acceleration of MD simulations.

Sugimoto et al. (Sugimoto et al. 1990; Ebisuzaki et al. 1993) have developed a series of hardwares which is called as GRAPE (GRAvity PipE) for the calculation of gravitational many-body simulations. This approach has been extremely successful. They completed the first Tera-flops machine, GRAPE-4, in the summer of 1995 (Makino et al. 1995, 1997). The total cost for developing GRAPE-4 was less than 2 million dollars, and they took only three years to develop it. The price-performance of GRAPE-4 is a hundred times better than that of typical general-purpose computers. GRAPE hardwares are not only fast in calculation speed but also inexpensive and quick to develop.

A GRAPE hardware can be applied to molecular dynamics simulations (Sugimoto et al. 1990). In fact, Ito et al. (1993) have developed GRAPE-2A which can calculate Coulomb force and van der Waals force as well as gravitational force. It has a programmable RAM table for the evaluation of pair-wise force. It can calculate central forces with arbitrary functional forms by changing the content of the RAM table in the force evaluation unit.

Fukushige et al. (1993) developed another type of special-purpose computer, WINE-1, which is specialized for discrete Fourier transform and inverse discrete Fourier transform. These transformations are used to calculate Coulomb forces in wavenumber-space in Ewald method (Ewald 1921) under the periodic boundary condition. Taiji et al. (1994) developed a specialized LSI chip, MD-GRAPE chip, which implements the functions of GRAPE-2A and WINE-1 pipeline in a single chip. An MD-GRAPE board with four MD-GRAPE chips has a peak speed of 4 Gflops (Fukushige et al. 1996). Toyoda et al. (1995) also developed a similar machine, MD-engine.

In the present thesis, the author gives the hardware design of a highly parallelized GRAPE system for molecular dynamics simulations, which he named as Molecular Dynamics Machine (hereafter MDM). The target speed of MDM is about 100 Tflops. MDM will be able to perform one time-step of a million particles in 0.1 seconds. It takes only one

day ($\sim 10^5$ sec) to perform the simulations of a time span of a nano-seconds ($\sim 10^6$ time-steps). MDM will be about a hundred times faster than currently fastest supercomputers in sustained speed for MD simulations with a million atoms (figure 1.1).

MDM is composed of a host computer and two kinds of specialized processor chips, 2560 MDGRAPE-2 chips and 3072 WINE-2 chips. We can use a workstation cluster or supercomputer as a host computer. The author has designed both of MDGRAPE-2 and WINE-2 chips, and engineering samples of MDGRAPE-2 and WINE-2 chips will be shipped at the third and the first quarters of 1998, respectively. Total system of MDM will be completed in 1999. Total budget for MDM is about 5 million dollars including the host computer. MDM will be the fastest computer in the world in 1999, since the fastest supercomputer will reach to only 4 Tflops at that time. The aim of the present thesis is to describe the system design of MDM.

The present thesis is organized as follows. The author describes the basic equations and algorithms of molecular dynamics simulations in chapter 2. In chapter 3, he gives the structure of the MDM system and the policy of chip design of WINE-2 and MDGRAPE-2 chips. In chapter 4 and 5, the author presents the detailed description of of WINE-2 and MDGRAPE-2 chips, respectively. In chapter 6, he shows the predicted performance of MDM. In chapter 7, the author compares MDM with other special and general purpose computers and discusses some sophisticated algorithm for MDM. There are strong motivations to realize longer time span to understand longer phenomena, such as folding of tertial structure of proteins and vibrations of lipid bilayer. He also discusses the possibility to realize a micro-second MD simulation for a hundred thousands atoms using a supercomputer as a host computer.

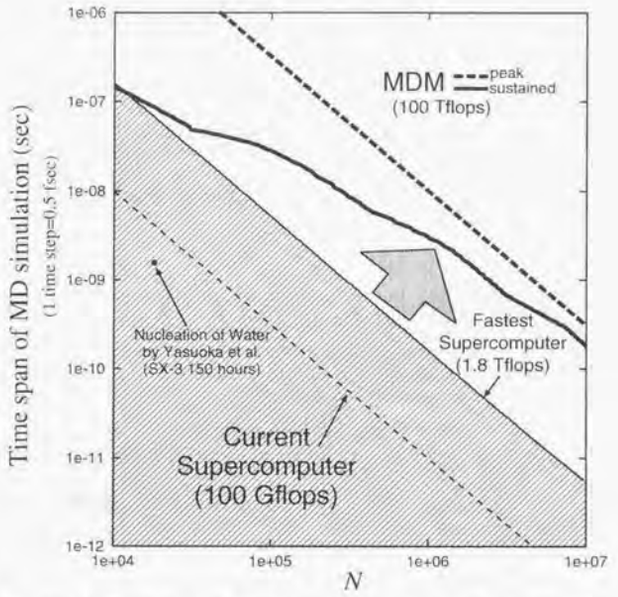


Figure 1.1: Time span of MD simulation is plotted against the number of atoms. We assume 7 days of CPU time and 0.5 femto-second of a time step. Ewald method is assumed to be used for calculating Coulomb forces. The number of atoms is limited to several tens of thousands, when we perform 1 nano-second of MD simulation by a moderate speed of supercomputer with a peek speed of 100 Gflops (thin dashed line). Thick solid curve shows the expected performance of MDM (see chapter 6): MDM can perform 3 nano-seconds of MD simulation with a million atoms. Even with the currently fastest supercomputer (1.8 Tflops; thin solid line), you can only perform up to 100 pico-seconds of MD simulation with a million atoms at its peak speed.

Chapter 2

Basic Equations and Algorithms of MD Simulation

In Molecular dynamics simulations, atoms in a system are treated as classical particles and receives Coulomb, Van der Waals, covalent bond, and hydrogen bond forces from other particles. Positions and velocities of atoms are updated by the Newton's equation of motion. In the present chapter, the author describes the basic equations and algorithms of MD simulations. In section 2.1, basic equations are summarized. In section 2.2, equations to calculate non-bonding forces, which dominates over the calculation cost, are described.

2.1 Basic Equations

2.1.1 Interaction among particles

In molecular dynamics simulations, all interactions between particles are governed by the following potential energy (Weiner et al. 1984):

$$\begin{aligned}\Phi &= \sum_{i<j}\frac{q_i q_j}{r_{ij}} + \sum_{i<j}\left(\frac{a_{ij}}{r_{ij}^{12}} - \frac{b_{ij}}{r_{ij}^{6}}\right) \\ &\quad + \sum_{\text{bonds}} K_b(r-r_0)^2 + \sum_{\text{bond angle}} K_\theta(\theta-\theta_0) \\ &\quad + \sum_{\text{torsion angle}} K_\phi[1+\cos(n\phi-\gamma)] \\ &\quad + \sum_{\text{H-bond}}\left(\frac{c_{kl}}{r_{kl}^{10}} - \frac{d_{kl}}{r_{kl}^{8}}\right),\end{aligned} \tag{2.1}$$

where r_{ij} is the distance between particles i and j, q_i is the charge of particle i, θ is the bond angle, ϕ is the torsion angle, K_b, K_θ, and K_ϕ are the constants of springs representing covalent bonds, and a_{ij}, b_{ij}, and c_{kl}, d_{kl} are the coefficients of van der Waals potential and hydrogen-bonding, respectively. The first and second terms represents Coulomb and Van

der Waals potentials, respectively, and the third to fifth terms represent covalent-bonds, and the last term represents hydrogen-bonds.

2.1.2 Force calculation

The forces exerted on a particle are obtained by taking gradient of equation 2.1 on the respect of the position of the particle. The force, $\vec{F}_i$, on particle i is expressed as:

$$\vec{F}_i = \vec{F}_i(\mathrm{Clb}) + \vec{F}_i(\mathrm{vdW}) + \vec{F}_i(\mathrm{bd}), \tag{2.2}$$

where $\vec{F}_i(\mathrm{Clb})$ is the Coulomb force, $\vec{F}_i(\mathrm{vdW})$ is the van der Waals force, and $\vec{F}_i(\mathrm{bd})$ is the bonding force by covalent and hydrogen bonds.

Coulomb force is calculated by Ewald method, when the periodic boundary condition is imposed. In Ewald method, $\vec{F}_i(\mathrm{Clb})$ is divided into two parts, *i.e.*, real-space part, $\vec{F}(\mathrm{re})$, and wavenumber-space part, $\vec{F}(\mathrm{wn})$:

$$\vec{F}_i(\mathrm{Clb}) = \vec{F}_i(\mathrm{re}) + \vec{F}_i(\mathrm{wn}), \tag{2.3}$$

where $\vec{F}_i(\mathrm{re})$ and $\vec{F}_i(\mathrm{wn})$ are expressed as:

$$\begin{aligned}\vec{F}_i(\mathrm{re}) &= \frac{q_i}{4\pi\varepsilon_0}\sum_j q_j \left[\frac{\mathrm{erfc}(\alpha r_{ij}/L)}{r_{ij}} \right. \\ &\quad \left. + \frac{2\alpha}{\sqrt{\pi}L}\exp(-\alpha^2 r_{ij}^2/L^2)\right]\frac{\vec{r}_{ij}}{r_{ij}^2}, \qquad (2.4)\\ \vec{F}_i(\mathrm{wn}) &= \frac{q_i}{2\pi\varepsilon_0 L^3}\sum_{\vec{k}}\frac{\vec{k}}{k^2}\exp(-\pi^2 L^2 k^2/\alpha^2) \\ &\quad \left[\sin(2\pi\vec{k}\cdot\vec{r}_i)\sum_j q_j\cos(2\pi\vec{k}\cdot\vec{r}_j)\right. \qquad (2.5)\\ &\quad \left. - \cos(2\pi\vec{k}\cdot\vec{r}_i)\sum_j q_j\sin(2\pi\vec{k}\cdot\vec{r}_j)\right],\end{aligned}$$

where $\vec{r}_i$ is the position of particle i, $\vec{r}_{ij}$ is the relative vector from particle j to particle i, r_{ij} is the distance between particle i and particle j, $\vec{k}$ is the wavenumber vector, q_i is the electrostatic charge of particle i, L is the length of a side of the computational box, ε_0 is the dielectric constant of vacuum, α is a parameter to adjust the computational cost for real and wavenumber part of the Coulomb force calculation, and $\mathrm{erfc}(x)$ is the complementary error function:

$$\mathrm{erfc}(x) = 1 - \frac{2}{\sqrt{\pi}}\int_0^x \exp(-t^2)dt. \tag{2.6}$$

The wavenumber vector, $\vec{k}$, can be expressed as $(k_x/L, k_y/L, k_z/L)$ by using three integers k_x, k_y, and k_z.

The author assumes that the computational box is a cube in the present thesis. He calls the particle on which forces are exerted as 'force-receiving particle', and the particle which exerts force as 'force-exerting particle'.

The van der Waals force, $\vec{F}_i(\mathrm{vdW})$, is calculated by

$$\begin{aligned}\vec{F}_i(\mathrm{vdW}) &= \sum_j \varepsilon(at_i, at_j)\left\{2\left[\frac{\sigma(at_i, at_j)}{r_{ij}}\right]^{14}\right. \\ &\quad \left. - \left[\frac{\sigma(at_i, at_j)}{r_{ij}}\right]^{8}\right\}\vec{r}_{ij}, \qquad (2.7)\end{aligned}$$

where $\varepsilon(at_i, at_j)$, $\sigma(at_i, at_j)$ are the coefficients which depend on both of atom types at_i and at_j, where at_i is the atom type of particle i. In the present thesis, van der Waals force of Lennard-Jones potential type is used.

2.1.3 Time integration

There are many kinds of integrator to integrate equation of motion, such as Verlet algorithm (Verlet 1967), Beeman algorithm (Beeman 1976), and multiple time-step algorithms (Tuckerman et al. 1992). Verlet algorithm, which is one of the simple and popular algorithms for time-integration of the positions of particles, is expressed as follows.

$$\begin{aligned}\vec{r}(t+\delta t) &= \vec{r}(t) + \delta t\,\vec{v}(t+\delta t/2), &(2.8)\\ \vec{v}(t+\delta t/2) &= \vec{v}(t-\delta t/2) + \delta t\,\vec{a}(t), &(2.9)\end{aligned}$$

where, $\vec{r}(t)$, $\vec{v}(t)$ and $\vec{a}(t)$ are the position, velocity, and acceleration of a particle at time t, respectively.

2.1.4 Pressure Control

There are several kinds of techniques to simulate various thermal conditions, such as constant-temperature or constant-pressure conditions. For example, Parrinello-Rahman equations (Parrinello and Rahman 1981) are often used for the simulation under the constant-pressure condition. When you use Nosé and Klein's extension of Parrinello-Rahman equations to rigid molecular systems (Nosé and Klein 1983), the pressure, P, is calculated as:

$$P = \frac{Nk_BT}{V}\left(1 - \frac{1}{3k_BT}\sum_{i\neq j} f_{ij}r_{ij}\right), \tag{2.10}$$

where k_B is the Boltzmann constant, N, T and V are the number of atoms, the temperature and the volume of the system, respectively, and f_{ij} is the pair-wise force between particles i and j. The second term, $\sum_{i\neq j} f_{ij}r_{ij}$, represents the effect of mutual interaction of particles on pressure in the system and is called as virial.

2.2 Calculation of Non-Bonding Forces with MDM

Calculation of non-bonding, such as Coulomb and van der Waals forces, dominates the total calculation of the MD simulation. In particular, the calculation for Coulomb force is most time-consuming. Although the number of interactions per atom of covalent and hydrogen bonds is less than 10, that for van der Waals is more than a hundred, and that for Coulomb is $\sim N$, where N is the number of particles in the computational box.

In the present thesis, the author used Ewald method for the calculation of Coulomb force. Other methods, such as tree-code, P^3M, PPPC, and fast multipole method, will be discussed in chapter 7.

2.2.1 Calculation algorithm for $\vec{F}_i$(re) and $\vec{F}_i$(vdW)

In the calculation of $\vec{F}_i$(re) and $\vec{F}_i$(vdW) by equations (2.4) and (2.7), the author uses cell-index method (Hockney et al., 1981). The computational box is divided into cells and force-exerting particles are treated in the unit of a cell. In the present thesis, he treats the case that a cell is a cube and the length of its side, L_{cell}, is the same as the cut-off radius, r_{cut}, of $\vec{F}_i$(re) and $\vec{F}_i$(vdW). Number of particles, N_{cell}, in a cell is expressed as:

$$N_{cell} \simeq \frac{N}{l_{box}^3} = \frac{r_{cut}^3 N}{L^3}, \tag{2.11}$$

where l_{box} is the number of cells of a side of the computational box, *i.e.*, $l_{box} = L/L_{cell}$. Total number of cells is l_{box}^3.

In the present thesis, the author assumes that particles in a cell interact with the particles in that cell and neighboring 26 cells. Figure 2.1 shows neighboring cells in the two-dimensional case. Suppose that we calculate the force of a particle (particle i) in the central cell, we sum up the forces from particles in the cells of the thin hatched region. One may reduce the calculation cost for real-space part of Ewald method and that of van der Walls forces by using smaller cells. This possibilities will be discussed in chapter 7.

2.2.2 Calculation algorithm for $\vec{F}_i$(wn)

The wavenumber space part of Coulomb force, $\vec{F}_i$(wn), is actually calculated with MDM by the following equations. Equation (2.5) can be rewritten as:

$$\vec{F}_i(\mathrm{wn}) = \sum_{n=1}^{N_{wv}} a_n \sin(2\pi \vec{k}_n \cdot \vec{r}_i + \theta_n) \cdot \vec{k}_n, \tag{2.12}$$

where, a_n and θ_n are calculated by:

$$a_n = \frac{q_i}{\pi \varepsilon_0 L^3} \frac{\exp(-\pi^2 L^2 k_n^2/\alpha^2)}{k_n^2} \sqrt{S_n^2 + C_n^2}, \tag{2.13}$$

$$\theta_n = \begin{cases} -\sin^{-1}\left(S_n/\sqrt{S_n^2 + C_n^2}\right) & \text{for } C_n \geq 0, \\ \pi + \sin^{-1}\left(S_n/\sqrt{S_n^2 + C_n^2}\right) & \text{for } C_n < 0, \end{cases} \tag{2.14}$$

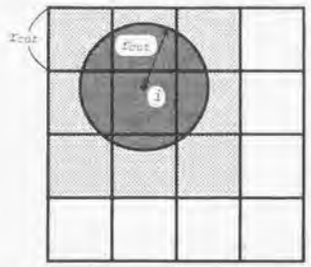


Figure 2.1: Cell-index method in two-dimensional case. The size of a cell is the same as r_{cut}. Force of a particle i is the sum of forces from the particles in thin hatched region.

where S_n and C_n are:

$$S_n = \sum_{j=1}^{N} q_j \sin(2\pi \vec{k}_n \cdot \vec{r}_j), \tag{2.15}$$

$$C_n = \sum_{j=1}^{N} q_j \cos(2\pi \vec{k}_n \cdot \vec{r}_j). \tag{2.16}$$

Here, N_{wv} is the number of wavenumber vectors to be taken into account, for the accumulation of $\vec{F}_i$(wn), *i.e.*,

$$N_{wv} \simeq \frac{2}{3}\pi L^3 k_{cut}^3, \tag{2.17}$$

where k_{cut} is the cut-off length of wavenumber vector, and N_{wv} is the half of the number of wavenumber vectors whose lengths are shorter than k_{cut}. Here, the author uses the fact that the term in the summation of $\vec{k}$ in equation (2.5) takes exactly the same value for the case of $\vec{k}$ and $-\vec{k}$.

Chapter 3

System Design of Molecular Dynamics Machine

Molecular Dynamics Machine is composed of a host computer and two special-purpose hardware, MDGRAPE-2 and WINE-2 (figure 3.1). A host computer is a general-purpose computer, such as a workstation or a supercomputer, which works as a front-end processor. Two special-purpose computers, which the author is newly developing, work as a back-end hardware.

In MDM, the host computer calculates bonding forces for covalent and hydrogen bonds and updates the positions and velocities of particles. The non-bonding forces, which dominates the total calculation cost, are calculated by MDGRAPE-2 and WINE-2. MDGRAPE-2 calculates $\vec{F}_i(\mathrm{re})$ (equation 2.4) and $\vec{F}_i(\mathrm{vdW})$ (equation 2.7), while WINE-2 calculates $\vec{F}_i(\mathrm{wn})$ (equation 2.5). Since the load on MDGRAPE-2 and WINE-2 is much heavier than those for host computer and communication interfaces among them, the total performance of the system is determined by the super-highspeed special-purpose computers rather than the host computer or communications, for a large number of particles.

The author designed two kinds of chips, *i.e.*, MDGRAPE-2 and WINE-2 chips. These chips are specialized for molecular dynamics simulations, and will be massively parallelized to form MDM. At the start of the design of MDM, the author studied the error of MD-GRAPE system (Fukushige et al. 1996), which has four MD-GRAPE chips (Taiji et al. 1994). He found that the accuracy in MD-GRAPE is too high by three orders of magnitude in the calculation of wavenumber space. If we use a shorter number of digits, we can save the transistor size, and use a higher clock frequency by a large factor even using the same technology. He, therefore, decided to develop two different chips, *i.e.*, WINE-2 chip for wavenumber-space calculation and MDGRAPE-2 chip for real-space calculation.

In the present chapter, the author describes the system design of MDM. The structure of the system is described in section 3.1. In section 3.2, he reviews the structure of MD-GRAPE, and presents the design policy of WINE-2 and MDGRAPE-2 chips.

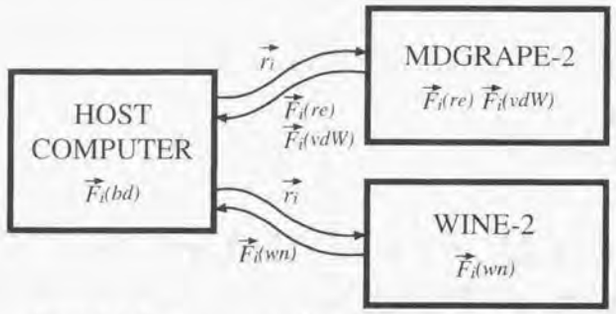


Figure 3.1: Basic structure of Molecular Dynamics Machine

3.1 Structure of Molecular Dynamics Machine

The Molecular Dynamics Machine (MDM) is divided into N_{nd} nodes connected with each other by a switch (figure 3.2). A node is composed of a node computer, MDGRAPE-2, and WINE-2 (figure 3.3). Each node computer is connected with MDGRAPE-2 by N_{gcl} links and with WINE-2 by N_{wcl} links. A node computer is a general-purpose computer, such as a PC, a workstation, or a supercomputer. It performs the calculation of bonding-force, $\vec{F}_i$(bd), and time integration of particles. MDGRAPE-2 calculates $\vec{F}_i$(re) and $\vec{F}_i$(vdW), while WINE-2 calculates $\vec{F}_i$(wn) (see figure 3.1). The structures of MDGRAPE-2 and WINE-2 are described in the following subsections.

3.1.1 MDGRAPE-2

MDGRAPE-2 has a hierarchical structure with three-levels, *i.e.*, G-cluster, G-board, and G-chip. MDGRAPE-2 is composed of N_{gcl} G-clusters (figure 3.3). Each G-cluster is connected to a node computer through a link. Each G-cluster has N_{gbd} G-boards and each G-board has N_{gcp} MDGRAPE-2 chips (hereafter, G-chips; figure 3.4). The author will get $N_{gcl} = 8$ and $N_{gbd} = 4$ as an optimal set based on the estimation in the section 6.2. He plans to put 10 ($= N_{gcp}$) G-chips on a G-board and adapt PCI (Peripheral Component Interconnect) bus (64-bit wide and 33 MHz clock frequency) as a link to the node.

G-cluster

G-boards and an interface board in a G-cluster are connected with each other by a bus. As for a bus, the author adopts PCI bus (64-bit wide and 33 MHz clock frequency) with an extension for broadcasting data: Data from a node computer to G-cluster can be broadcasted to all the G-boards through an interface board.

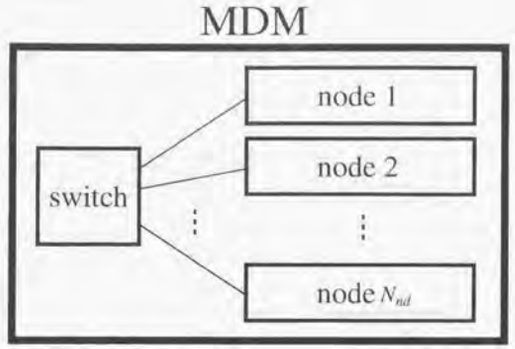


Figure 3.2: Block diagram of Molecular Dynamics Machine (MDM)

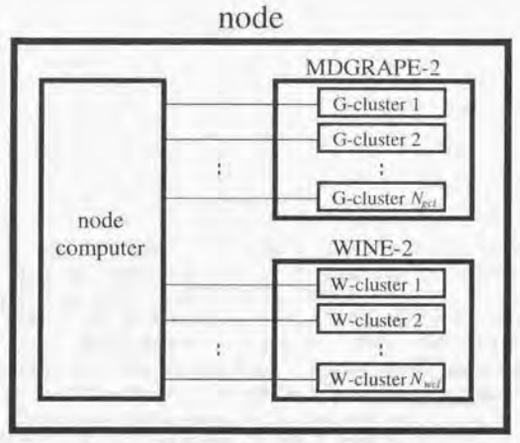


Figure 3.3: Block diagram of a node

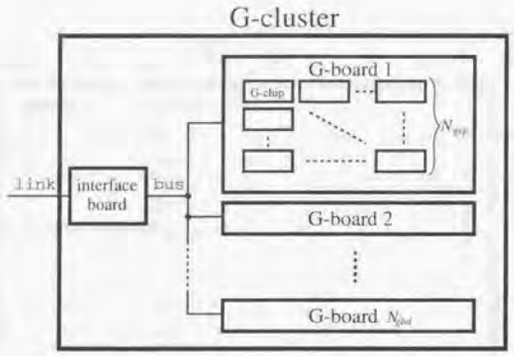


Figure 3.4: Structure of a G-cluster

G-board

A G-board calculates pairwise central forces such as real-space part of Coulomb and van der Waals forces with the cell-index method. It has $24N_{gcp}$ different virtual pipelines to calculate forces on $24N_{gcp}$ force-receiving particles:

$$\begin{aligned}
\vec{f}_1 &= \sum_{c=1}^{27} \sum_{j=jstart_c}^{jend_c} \vec{f}_{1,j}, \\
\vec{f}_2 &= \sum_{c=1}^{27} \sum_{j=jstart_c}^{jend_c} \vec{f}_{2,j}, \\
\vec{f}_3 &= \sum_{c=1}^{27} \sum_{j=jstart_c}^{jend_c} \vec{f}_{3,j}, \\
&\cdots \qquad\qquad \cdots \\
\vec{f}_{24N_{gcp}} &= \sum_{c=1}^{27} \sum_{j=jstart_c}^{jend_c} \vec{f}_{24N_{gcp},j},
\end{aligned} \tag{3.1}$$

where c is the index of the neighboring cells and $jstart_c$ is the index of the first particle in a cell c, $jend_c$ is the index of the last particle in a cell c, and $\vec{f}_{i,j}$ is the pairwise forces between particles i and j. Here, indices of particles in a cell are assumed to be contiguous.

In the case of real-space part of Coulomb force, $\vec{f}_{i,j}$ is given by:

$$\vec{f}_{i,j} = q_j\, g\left(\alpha^2 r_{ij}^2 / L^2\right) \vec{r}_{ij}, \tag{3.2}$$

where $g(x)$ is:

$$g(x) = \frac{2\exp(-x)}{\sqrt{\pi x}} + \frac{\mathrm{erfc}(\sqrt{x})}{x^{3/2}}. \tag{3.3}$$

A host computer calculates the real-space part of Coulomb force, $\vec{F}_i(\mathrm{re})$, as:

$$\vec{F}_i(\mathrm{re}) = \frac{q_i}{4\pi\varepsilon_0} \vec{f}_i. \tag{3.4}$$

In the case of van der Waals force, $\vec{f}_{i,j}$ is calculated by:

$$\vec{f}_{i,j} = \varepsilon(at_i, at_j)\, g\left(\frac{r_{ij}^2}{\sigma(at_i, at_j)^2}\right) \vec{r}_{ij}, \tag{3.5}$$

where $g(x)$ is:

$$g(x) = 2\,x^{-7} - x^{-4}, \tag{3.6}$$

for Lennard-Jones potential. The van der Waals force, $\vec{F}_i(\mathrm{vdw})$, is same as $\vec{f}_i$.

A G-board is composed of N_{gcp} G-chips, `Cell index counter`, `Cell memory`, `Particle index counter`, `Particle memory`, and `Interface logic` (figure 3.5). `Cell index counter` counts the index of the cell, c, from 1 to 27, and its output becomes the address of `Cell memory`. `Cell memory` stores pairs of indices of the first ($jstart_c$) and the last ($jend_c$) particles in cells, and outputs them to `Particle index counter`. `Particle index counter` counts the particle index, j, from $jstart_c$ to $jend_c$, and its output becomes the address of `Particle memory`. `Particle memory` stores positional coordinates ($\vec{r}_j$), charges (q_j), and atom types (at_j) of particles. These data are supplied to G-chips every six internal clock cycles of a G-chip. All the G-chips on a G-board share the output of `Particle memory`. `Interface logic` communicates with a G-board and the interface board in a G-cluster.

A G-board calculates forces on $24N_{gcp}$ force-receiving particles from a force-exerting particle every six internal clock cycles of a G-chip. It takes

$$6\,t_{gpipe} \sum_{c=1}^{27} (jend_c - jstart_c + 1) \simeq 6 \cdot 27 N_{cell} t_{gpipe} \tag{3.7}$$

to calculate equation (3.1) to get total forces, $\vec{f}_i$. Here, t_{gpipe} is the cycle time of the internal clock of a G-chip.

In the following sections, the author assumes that real-space part of Coulomb force on $12N_{gcp}$ particles and van der Waals force on $12N_{gcp}$ particles are calculated in parallel on a G-board, *i.e.*, $12N_{gcp}$ of pipelines are assigned to real-space part of Coulomb force and $12N_{gcp}$ of pipelines are assigned to van der Waals force.

G-chip

A G-chip has four G-pipelines for force calculation to perform the summations for $\vec{f}_i$ on 24 particles in every six internal clock cycles, because it uses six "virtual multiple pipelines" (Makino et al., 1993). The cycle time of the internal clock of G-chip, t_{gpipe}, is 10 nsec. The author presents the design policy of G-chip in section 3.2.2, and detailed description of a G-chip in chapter 5. Here, he describes the minimal description of MDGRAPE-2 chip (G-chip) to understand MDGRAPE-2.

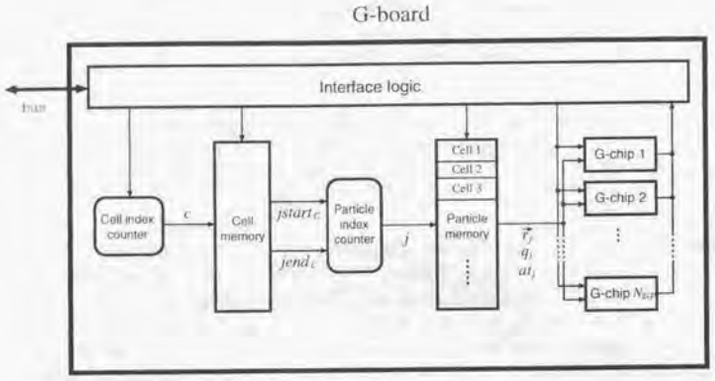


Figure 3.5: Block diagram of G-board

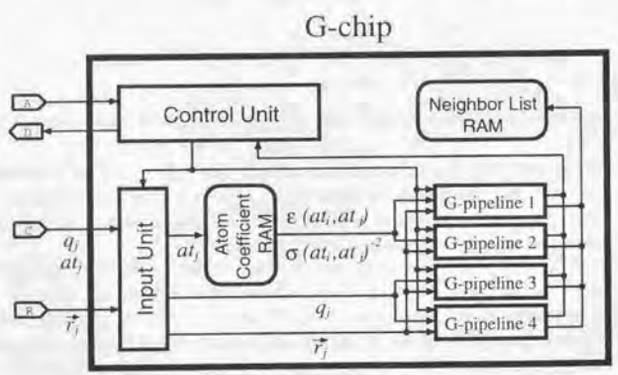


Figure 3.6: Block diagram of G-chip

A G-chip is composed of four G-pipelines, `Atom Coefficient RAM`, `Neighbor List RAM`, `Control Unit`, and `Input Unit` (figure 3.6). A G-pipeline calculates forces on 6 force-receiving particles from a force-exerting particle every six internal clock. Figure 3.7 shows the block diagram of a G-pipeline. Summation of $\vec{f}_{i,j}$ in equation (3.1) is performed by 64-bit double-precision floating-point accumulator. Most of other operations, such as multiply of q_j or $\varepsilon(p,q)$, are performed in 32-bit floating-point arithmetic units. An arbitrary function, $g(x)$, is approximated by 4-th polynomial interpolation in `Function Evaluator` in a G-pipeline. Coefficients of the polynomials are stored in the RAM in `Function Evaluator`. The RAM stores 1024 sets of coefficients. The relative error by this 4-th polynomial interpolation by table look-up is as low as about 10^{-7}, which is almost the same as that of 32-bit floating-point value (IEEE 754). The net accuracy in force calculation is also about that of 32-bit floating-point operation.

The data necessary to calculate are supplied as follows. `Particle memory` on a G-board sends q_j, at_j, and r_j to G-chips in every six internal clock cycles of them. A node computer writes $\vec{r}_i$, α^2/L^2, and at_i to G-chips before starting the calculation of equation (3.1). Coefficients $\varepsilon(p,q)$ and $\sigma(p,q)$ for all the pair of atom types p and q are stored in `Atom Coefficient RAM` in a G-chip before starting the calculation. One G-chip can calculate forces for 1024 pairs of atom types, *i.e.*, each G-chip can calculate all the pairs of atom types, when number of atom types is less than or equal to 32. You can handle a larger number of atom types than 32, if G-chips are divided into several groups for different sets of atom types of force-receiving particles.

A G-chip also can calculate potential of particles from particles in its potential mode as:

$$\Phi_i = \sum_j b_j G(a_j r^2). \tag{3.8}$$

A G-pipeline calculates and accumulates potentials of 6 force-receiving particles from a force-exerting particle in every six clock cycles.

The calculation speed of a G-chip corresponds to about $(55\times2+27\times2)\times100\times10^6 = 16.4$ Gflops, because two pipelines are used for real part of Coulomb force, two pipelines are used for van der Waals force, and the clock cycle of a G-chip is 100 MHz. Here, the author evaluates the number of floating point operations for calculating and accumulating $\vec{f}_{i,j}$ of real part of Coulomb force as 55, and that for van der Waals force as 27. Where he assumes that divide operation is equivalent to five floating-point operations and square root, exponential, and complementary error function are equivalent to ten floating-point operations.

Design of a G-chip is now in the final stage and engineering sample chips will be shipped in the third quarter of 1998. The number of gates in a G-chip is about one million.

3.1.2 WINE-2

WINE-2 has a hierarchical structure with three-levels, *i.e.*, W-cluster, W-board, and W-chip. WINE-2 is composed of N_{wcl} W-clusters (figure 3.3). Each W-cluster is connected to a node computer through a link and has N_{wbd} W-boards. Each W-board has N_{wcp} WINE-2

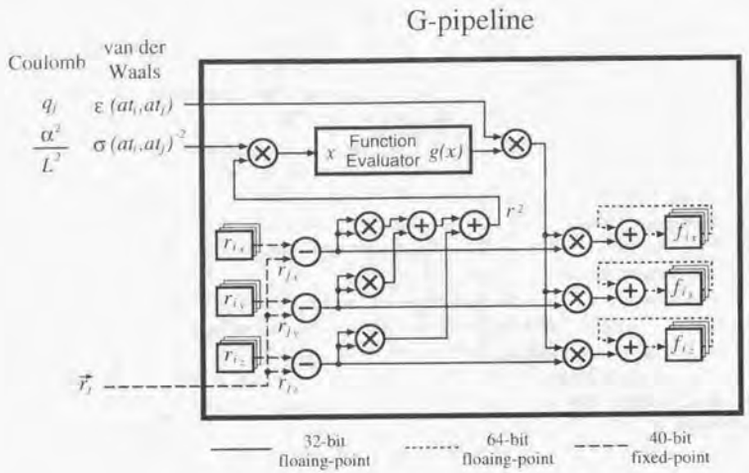


Figure 3.7: Block diagram of G-pipeline

chips (hereafter, W-chips; figure 3.8). The author will adapt $N_{wcl} = 3$ and $N_{wbd} = 8$ in discussions in section 6.2, and he plans to put 16 ($= N_{wcp}$) W-chips on a W-board and adapt PCI bus (64-bit wide and 33 MHz clock frequency) as a link.

W-cluster

W-boards and an interface board in the W-cluster are connected with each other by a bus, which is the same bus as that of G-cluster.

W-board

A W-board calculates S_n and C_n (equations 2.15 and 2.16) in DFT (Discrete Fourier Transform) mode and $\vec{F}_i(\mathrm{wn})$ in IDFT (Inverse Discrete Fourier Transform) mode. It has $64N_{wcp}$ virtual pipelines for DFT and IDFT.

In DFT mode, a W-board calculates S_n and C_n for $64N_{wcp}$ different wavenumber vectors (k_n):

$$S_1 = \sum_{j=1}^{jmax} q_j \sin(2\pi \vec{k}_1 \cdot \vec{r}_j),$$

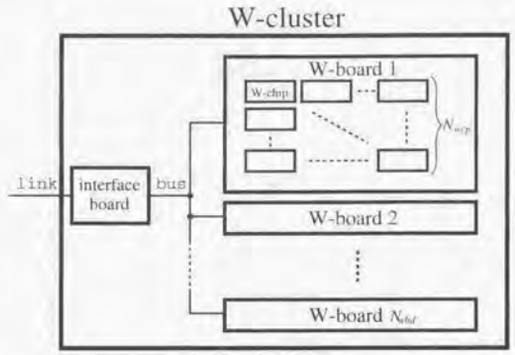


Figure 3.8: Structure of a W-cluster

$$\begin{aligned}
C_1 &= \sum_{j=1}^{jmax} q_j \cos(2\pi \vec{k}_1 \cdot \vec{r}_j), \\
S_2 &= \sum_{j=1}^{jmax} q_j \sin(2\pi \vec{k}_2 \cdot \vec{r}_j), \\
C_2 &= \sum_{j=1}^{jmax} q_j \cos(2\pi \vec{k}_2 \cdot \vec{r}_j), \\
&\cdots \qquad\qquad \cdots \\
S_{64N_{wcp}} &= \sum_{j=1}^{jmax} q_j \sin(2\pi \vec{k}_{64N_{wcp}} \cdot \vec{r}_j), \\
C_{64N_{wcp}} &= \sum_{j=1}^{jmax} q_j \cos(2\pi \vec{k}_{64N_{wcp}} \cdot \vec{r}_j),
\end{aligned} \tag{3.9}$$

where $jmax$ is the number of particles for summation. A W-board calculates one summation in S_n and C_n on $64N_{wcp}$ different wavenumber vectors in every eight internal clock cycles of a W-chip. It takes $8t_{wpipe}jmax$ to calculate equation (3.9), where t_{wpipe} is the cycle time of internal clock of a W-chip.

In IDFT mode, a W-board calculates $\vec{f}_i$ for $64N_{wcp}$ different force-receiving particles as:

$$\begin{aligned}
\vec{f}_1 = \sum_{n=1}^{N_{wv}/2} \Big[& c_{2n-1} \sin(2\pi \vec{k}_{2n-1} \cdot \vec{r}_1 + \theta_{2n-1}) \cdot \vec{k}_{2n-1} \\
& + c_{2n} \sin(2\pi \vec{k}_{2n} \cdot \vec{r}_1 + \theta_{2n}) \cdot \vec{k}_{2n} \Big],
\end{aligned}$$

$$\vec{f}_2 = \sum_{n=1}^{N_{wv}/2} \left[c_{2n-1} \sin(2\pi \vec{k}_{2n-1} \cdot \vec{r}_2 + \theta_{2n-1}) \cdot \vec{k}_{2n-1} + c_{2n} \sin(2\pi \vec{k}_{2n} \cdot \vec{r}_2 + \theta_{2n}) \cdot \vec{k}_{2n} \right], \quad (3.10)$$

$$\cdots \qquad \cdots$$

$$\vec{f}_{64N_{wcp}} = \sum_{n=1}^{N_{wv}/2} \left[c_{2n-1} \sin(2\pi \vec{k}_{2n-1} \cdot \vec{r}_{64N_{wcp}} + \theta_{2n-1}) \cdot \vec{k}_{2n-1} + c_{2n} \sin(2\pi \vec{k}_{2n} \cdot \vec{r}_{64N_{wcp}} + \theta_{2n}) \cdot \vec{k}_{2n} \right],$$

where c_n is written as:

$$c_n = \frac{\exp(-\pi^2 L^2 k_n^2/\alpha^2)}{k_n^2} \sqrt{S_n^2 + C_n^2}. \quad (3.11)$$

A W-board calculates one summation in $\vec{f}_i$ on different $64N_{wcp}$ particles every eight internal clock cycles of a W-chip. It takes $4\,t_{wpipe} N_{wv}$ to calculate equation (3.10). A host computer calculates the wavenumber-space part of Coulomb force, $\vec{F}_i$(wn), as:

$$\vec{F}_i(\mathrm{wn}) = \frac{q_i}{\pi \varepsilon_0 L^3} \vec{f}_i. \quad (3.12)$$

A W-board is composed of N_{wcp} W-chips, `Interface logic`, `Particle index counter`, and `Particle memory` (figure 3.9). `Interface logic` communicates with a W-board and the interface board in a W-cluster.

In DFT mode, `Particle index counter` counts the particle index, j, from 1 to $jmax$, and its output becomes the address of `Particle memory`. `Particle memory` stores positional coordinates ($\vec{r}_j$) and charges (q_j) of particles, and these data are supplied to all the W-chips on a W-board in every eight internal clock cycles of a W-chip.

In IDFT mode, `Particle index counter` counts the wavenumber index, n, from 1 to $N_{wv}/2$, and its output becomes the address of `Particle memory`. `Particle memory` stores wavenumber vectors ($\vec{k}_{2n-1}$) and four variables (a_{2n-1}, a_{2n}, θ_{2n-1} and θ_{2n}) of wavenumbers, and these data are supplied to all the W-chips on a W-board every eight internal clock cycles of a W-chip.

W-chip

A W-chip has eight W-pipelines for DFT or IDFT, `Controller`, and `Interface` (figure 3.10). A W-chip performs one summation in S_n and C_n of 64 different wavenumber vectors every eight internal clock cycle in DFT mode and one summation in $\vec{f}_i$ of 64 different particles every eight internal clock cycle in IDFT mode. It uses eight virtual multiple pipelines. The cycle time of the internal clock of a W-chip, t_{wpipe}, is 12.5 nsec. The author presents the design policy of W-chip is section 3.2.2, and the detailed description of a W-chip in chapter 4. Here, he describes the minimal description of WINE-2 chip (W-chip) to understand WINE-2.

A W-pipeline calculates $\sin(x)$ and $\cos(x)$ by the second polynomial interpolation. Coefficients of the polynomials are stored in a ROM in a W-pipeline. The ROM has 128 sets

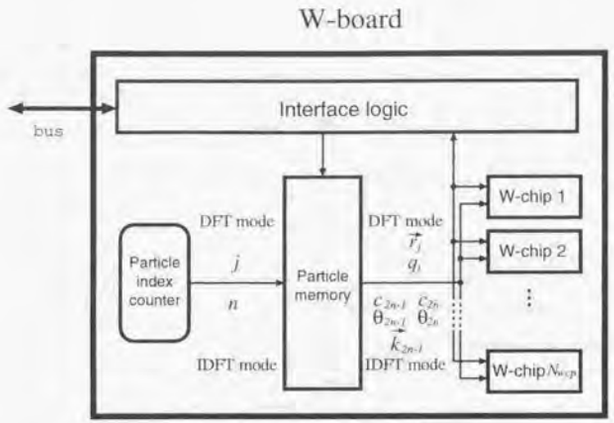


Figure 3.9: Block diagram of W-board

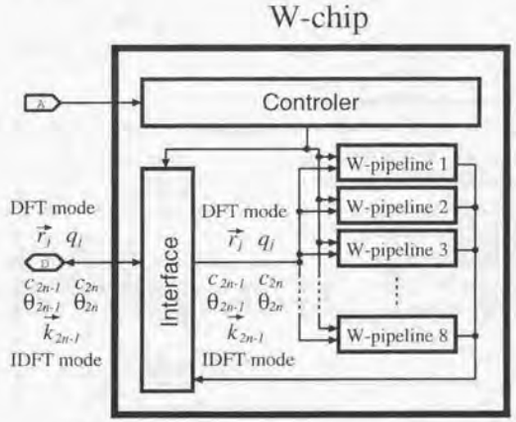


Figure 3.10: Block diagram of W-chip

of coefficients corresponding to the value of input x. The relative accuracy of this second polynomial interpolation is about $10^{-4.5}$.

In DFT mode, a W-pipeline performs discrete Fourier transformation on eight different wavenumber vectors from one j-particle and accumulates them in every eight clock cycles. Figure 3.11 shows the block diagram of a W-pipeline in DFT mode. The pipeline calculates inner product of vectors $\vec{r}_j$ and $\vec{k}_n$, and then evaluates *sine* and *cosine* of them. It multiplies the results to q_j, and accumulates the products to get S_n+C_n and S_n-C_n. A host computer calculates S_n and C_n from S_n+C_n and S_n-C_n. In DFT mode, 64 wavenumber vectors ($\vec{k}_n$) are loaded to a W-chip before starting the calculation. `Particle Memory` on a W-board supplies positions ($\vec{r}_j$) and charges (q_j) of particles to W-chips in every eight internal clock cycles of a W-chip.

In IDFT mode, a W-pipeline performs inverse discrete Fourier transformation on eight different particles from two wavenumber vectors and accumulate them in every eight clock cycles. Figure 3.12 shows the block diagram of a W-pipeline in IDFT mode. The pipeline calculates two inner products $\vec{k}_{2n-1} \cdot \vec{r}_i$, and $\vec{k}_{2n} \cdot \vec{r}_i$, then adds them to θ_{2n-1} and θ_{2n}, and finally evaluates the value of *sine* function, respectively. Here, the author arranges the order of wavenumber vectors so that $\vec{k}_{2n}$ are different from $\vec{k}_{2n-1}$ only in the sign of the z-components of them. The pipeline multiplies the results to a_{2n-1} and a_{2n}, and obtains their sum, s, and their difference, d, of the two products. The pipeline calculates $s \times k_{2n-1_x}$, $s \times k_{2n-1_y}$, and $d \times k_{2n-1_z}$ and accumulates them to get $f_{i,x}$, $f_{i,y}$, and $f_{i,z}$. In IDFT mode, positional coordinates of 64 particles ($\vec{r}_i$) are loaded to a W-chip before starting the calculation. `Particle Memory` on a W-board supplies wavenumber vectors ($\vec{k}_{2n-1}$), coefficients (a_{2n-1} and a_{2n}), and phase shifts (θ_{2n-1} and θ_{2n}) to W-chips in every eight internal clock cycles of a W-chip.

Fixed-point format is used in all the arithmetic calculations in a W-chip. The relative accuracy of $\vec{F}_i$(wn) is about $10^{-4.5}$, which is accurate enough for MD simulations. Researchers usually truncate the accumulation of Ewald method at the relative error of 10^{-3} to 10^{-5}. In actual case, $\vec{F}_i$(wn) is several times smaller than $\vec{F}_i$(re). The error in $\vec{F}_i$(wn) is smaller than either that of $\vec{F}_i$(re) and the truncation error of Ewald sum.

The calculation speed of a W-chip corresponds to about $30 \times 8 \times 80 \times 10^6 = 19.2$ Gflops in DFT mode, and $37 \times 8 \times 80 \times 10^6 = 23.7$ Gflops in IDFT mode. Here, the author assumes that *sine* and *cosine* operations are equivalent to ten floating-point operations, and the clock cycle of a W-chip is 80 MHz.

Design of a W-chip has already finished and several engineering samples will be shipped in the first quarter of 1998. The size of a W-chip is about 300 K gates. It will be developed by 0.5 μm technology by LSI Logic K. K. It will be operated by 3.3 V and consume about 7 W per chip.

3.2 Chip Design

The author designed MDGRAPE-2 and WINE-2 chips. Before the chip design, he reviewed MD-GRAPE (Fukushige et al. 1996), and found that the accuracy in MD-GRAPE is too

Pipeline of WINE-2 chip in DFT mode

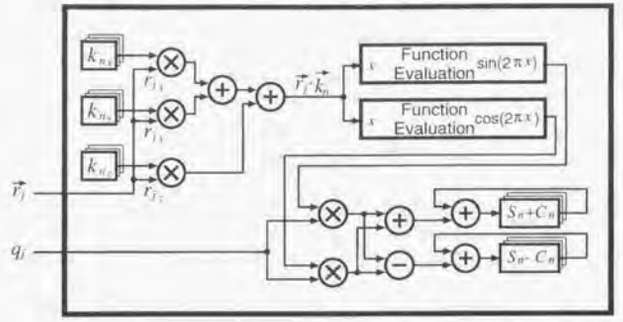


Figure 3.11: Block diagram of W-pipeline in DFT mode

Pipeline of WINE-2 chip in IDFT mode

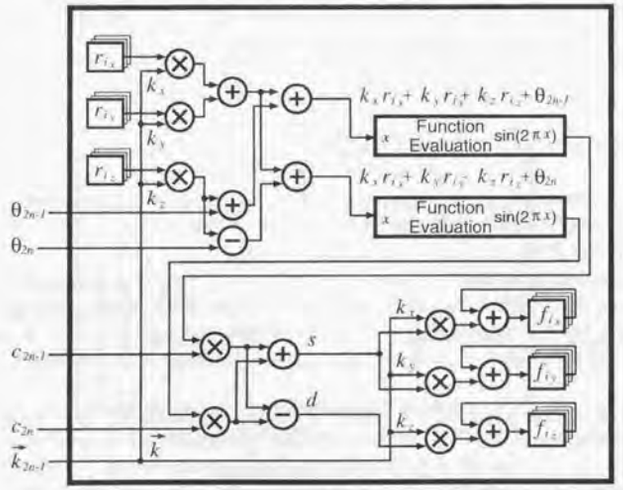


Figure 3.12: Block diagram of W-pipeline in IDFT mode

Table 3.1: Data input to MD-GRAPE chip

port	real-space part of Coulomb	van der Waals	DFT	IDFT
c	q_j $\alpha^2 L^{-2}$	$\varepsilon(at_i, at_j)$ $\sigma(at_i, at_j)^{-2}$	q_j	$\vec{a}_{c,n}$ or $\vec{a}_{s,n}$
r	$\vec{r}_j$	$\vec{r}_j$	$\vec{r}_j$	$\vec{k}_n$

high by three orders of magnitude in the calculation of wavenumber space. The author determined the least digits in every arithmetic units in the pipeline for achieving necessary accuracy based on the detailed error analysis in the calculation of wavenumber-space. As a result, a WINE-2 chip has eight times more pipelines (8 pipelines) and 2.3 times higher (80 MHz) clock frequency than an MD-GRAPE chip (1 pipeline and 35 MHz) with almost the same number of transistors and the same design rule.

As for MDGRAPE-2 chip, the author packed four pipelines, which are almost the same pipeline as MD-GRAPE chip, into a chip, and omitted the function for wavenumber-space calculation. Furthermore, he put `Atom Coefficient RAM` inside of a chip to improve the efficiency of pipelines in the massively parallelized case. The `Atom Coefficient RAM` stores the coefficients $[\sigma(at_i, at_j)$ and $\varepsilon(at_i, at_j)$ in equation 2.7] of van der Waals force.

In this section, the author first reviews MD-GRAPE (section 3.2.1), and then presents the design policy of WINE-2 and MDGRAPE-2 chips.

3.2.1 Structure of MD-GRAPE

MD-GRAPE (Fukushige et al. 1996) is a special-purpose computer for non-bonding calculation, such as van der Waals force, real-space part of Coulomb force, and wavenumber-space part of Coulomb force.

The structure of MD-GRAPE board is shown in figure 3.13. It is composed of four MD-GRAPE chips, `Cell index counter`, `Cell memory`, `Particle index counter`, `Particle memory`, `Atom Coefficient RAM`, `Neighbor List RAM`, and Interface logic. In the calculation of the real-space part of Coulomb force or van der Waals force, cell-index method are used.

An MD-GRAPE chip not only calculates arbitrary central forces but also performs discrete Fourier transform (DFT) or inverse DFT (IDFT). Figure 3.14 shows the block diagram of MD-GRAPE chip. An MD-GRAPE chip has three modes: FORCE mode, DFT mode, and IDFT mode, as described in the following three part. Furthermore, in the last part, the author describes the "virtual multiple pipeline" architecture (Makino et al. 1993), which reduces the bandwidth from memory chips during the force calculation.

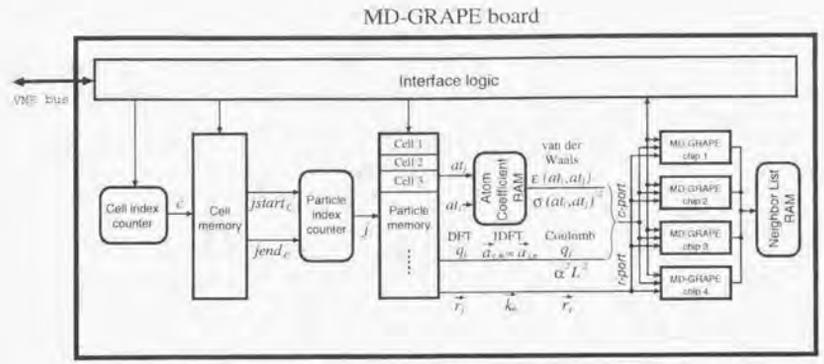


Figure 3.13: Block diagram of MD-GRAPE board. `Atom Coefficient RAM` is shared by four MD-GRAPE chips.

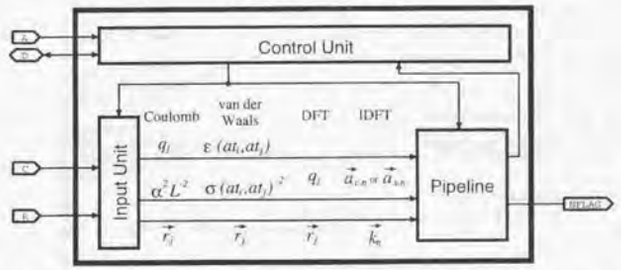


Figure 3.14: Block diagram of MD-GRAPE chip. C-port and r-port is used to load data to a pipeline as shown in table 3.1.

FORCE mode

In FORCE mode, an MD-GRAPE chip calculates forces:

$$\vec{f}_i = b_{ij} g(a_{ij} r_{ij}^2) \vec{r}_{ij}. \tag{3.13}$$

Figure 3.15 shows the block diagram of the pipeline of an MD-GRAPE chip in FORCE mode. Real-space part of Coulomb force, $\vec{F}_i(\mathrm{re})$, is calculated as:

$$\vec{F}_i(\mathrm{re}) = \frac{q_i}{4\pi\varepsilon_0} \vec{f}_i, \tag{3.14}$$

where

$$\begin{aligned} g(x) &= \frac{2\exp(-x)}{\sqrt{\pi x}} + \frac{\mathrm{erfc}(\sqrt{x})}{x^{3/2}}, && (3.15)\\ a_{ij} &= \alpha^2 L^{-2}, && (3.16)\\ b_{ij} &= q_j. && (3.17) \end{aligned}$$

On the other hand, van der Waals force, $\vec{F}_i(\mathrm{vdW})$, is calculated as:

$$\vec{F}_i(\mathrm{vdW}) = \vec{f}_i, \tag{3.18}$$

where

$$\begin{aligned} g(x) &= 2x^{-7} - x^{-4}, && (3.19)\\ a_{ij} &= \sigma(at_i, at_j)^{-2}, && (3.20)\\ b_{ij} &= \varepsilon(at_i, at_j). && (3.21) \end{aligned}$$

Coefficients, $\sigma(at_i, at_j)^{-2}$ and $\varepsilon(at_i, at_j)$, for van der Waals force are supplied by `Atom Coefficient RAM` on the MD-GRAPE board (figure 3.13). Note that all four pipelines on an MD-GRAPE board use the same coefficients.

DFT mode

In DFT mode, an MD-GRAPE chip performs DFT:

$$\begin{aligned} S_n &= \sum_{j=1}^{N} q_j \sin(2\pi \vec{k}_n \cdot \vec{r}_j), && (3.22)\\ C_n &= \sum_{j=1}^{N} q_j \cos(2\pi \vec{k}_n \cdot \vec{r}_j). && (3.23) \end{aligned}$$

Figure 3.16 shows the block diagram of the pipeline of an MD-GRAPE chip in DFT mode. Here, an MD-GRAPE chip takes $2NN_{wv}$ clock cycles in total for calculating N_{wv} pairs of S_n and C_n as shown in table 3.2.

Pipeline of MD-GRAPE chip in FORCE mode

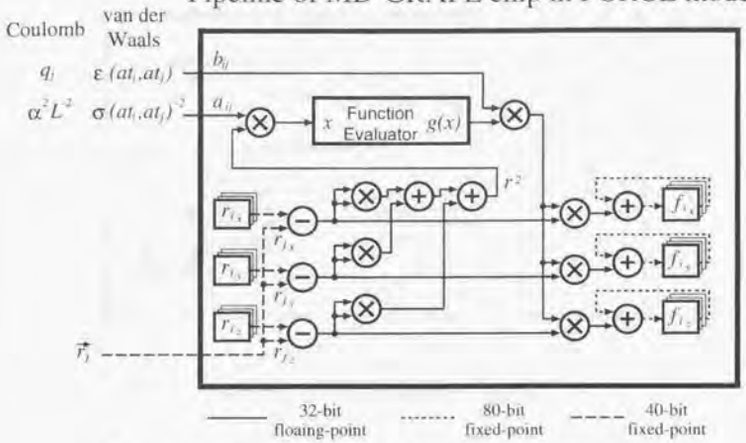


Figure 3.15: Block diagram of MD-GRAPE pipeline in FORCE mode. An MD-GRAPE pipeline has only one Function Evaluator.

Pipeline of MD-GRAPE chip in DFT mode

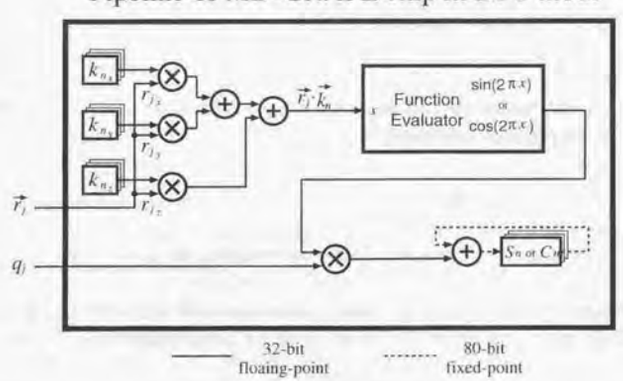


Figure 3.16: Block diagram of MD-GRAPE pipeline in DFT mode

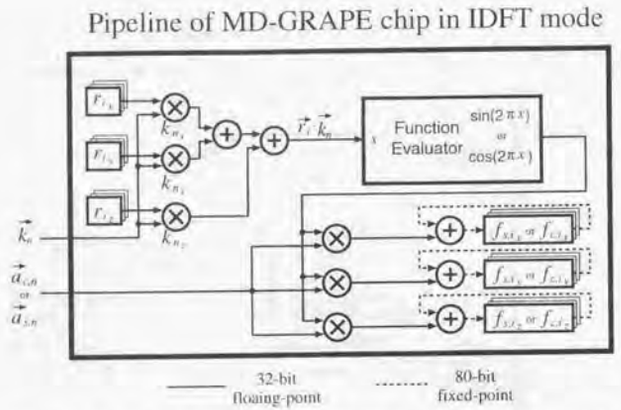


Figure 3.17: Block diagram of MD-GRAPE pipeline in IDFT mode

IDFT mode

In IDFT mode, an MD-GRAPE chip performs IDFT:

$$\vec{f}_{s,i} = \sum_{n=1}^{N_{wv}} \vec{a}_{c,n} \sin(2\pi \vec{k}_n \cdot \vec{r}_i), \tag{3.24}$$

$$\vec{f}_{c,i} = \sum_{n=1}^{N_{wv}} \vec{a}_{s,n} \cos(2\pi \vec{k}_n \cdot \vec{r}_i). \tag{3.25}$$

It takes $2NN_{wv}$ clock cycles in total for calculating N pairs of $\vec{f}_{s,i}$ and $\vec{f}_{c,i}$ as shown in table 3.2. Here, $\vec{a}_{s,n}$ and $\vec{a}_{c,n}$ are calculated by:

$$\vec{a}_{s,n} = \frac{\exp(-\pi^2 L^2 k_n^2/\alpha^2)}{k_n^2} S_n \vec{k}_n, \tag{3.26}$$

$$\vec{a}_{c,n} = \frac{\exp(-\pi^2 L^2 k_n^2/\alpha^2)}{k_n^2} C_n \vec{k}_n. \tag{3.27}$$

Figure 3.17 shows the block diagram of the pipeline of an MD-GRAPE chip in IDFT mode.

Wavenumber-space part of Coulomb force, $\vec{F}_i(\mathrm{wn})$, is calculated from $\vec{f}_{s,i}$ and $\vec{f}_{c,i}$ as:

$$\vec{F}_i(\mathrm{wn}) = \frac{q_i}{\pi \varepsilon_0 L^3} (\vec{f}_{c,i} - \vec{f}_{s,i}). \tag{3.28}$$

Virtual multiple pipeline

MD-GRAPE chip has "virtual multiple pipeline" (VMP) architecture (Makino et al. 1993), which reduces the bandwidth to supply necessary data to a chip during the force calculation. In this architecture, one pipeline acts as if there were many pipelines operating at a slower speed. An MD-GRAPE chip has six VMPs and calculates forces exerted on six particles, $\vec{f}_i, \ldots, \vec{f}_{i+5}$, simultaneously. The partial forces $\vec{f}_{ij}$ are evaluated in the following order:

$$\vec{f}_{11}, \vec{f}_{21}, \vec{f}_{31}, \vec{f}_{41}, \vec{f}_{51}, \vec{f}_{61}, \vec{f}_{12}, \vec{f}_{22}, \vec{f}_{32}, \vec{f}_{42}, \vec{f}_{52}, \vec{f}_{62}, \cdots$$

One interaction is evaluated in each clock cycle. An MD-GRAPE chip calculates the forces at six different positions using the same position vector, $\vec{r}_j$, and coefficients, a_{ij} and b_{ij}. Therefore, one position vector and two coefficients are supplied to an MD-GRAPE chip during six clock cycles.

3.2.2 Design policy of chips

The author designed a WINE-2 and an MDGRAPE-2 chips to achieve a higher, at least, by a factor of ten, compared with an MD-GRAPE chip. A WINE-2 chip calculates wavenumber-space part of Coulomb force, and is 50 times faster than an MD-GRAPE chip in DFT/IDFT mode (table 3.2). An MDGRAPE-2 chip calculates real-space part of Coulomb force and van der Waals force, and is 12 times faster than an MD-GRAPE chip in FORCE mode (table 3.3). In the following subsections, the author describes the design policy of a WINE-2 chip and an MDGRAPE-2 chip.

Design policy of WINE-2 chip

The design policy of WINE-2 chip is to pursue the highest performance by cutting off the accuracy. The author performed the detailed error analysis and determined the least digits in arithmetic units for achieving necessary accuracy. As a result, he achieved 50 times speed up from an MD-GRAPE chip by cutting off the unnecessary accuracy for the wavenumber-space part of calculation. Table 3.2 compares a WINE-2 chip with an MD-GRAPE chip in DFT/IDFT mode. The detailed descriptions of a WINE-2 chip is presented in the chapter 4. The speed up is achieved by the following four changes.

1. More pipelines in a chip
   A WINE-2 chip has eight pipelines, while an MD-GRAPE chip has one pipeline. This number of pipelines can be achieved by cutting off the accuracy in the force calculation as follows. Researchers of molecular dynamics simulations control the accuracy of force calculation so that the total energy conserve within a relative variance less than $10^{-2.5} \sim 10^{-3.5}$. An MD-GRAPE chip has a relative accuracy of 10^{-7}, which is, by far, higher than necessary in wavenumber-space part of calculation. The author reduced the accuracy of the arithmetic units in a WINE-2 chip to achieve a net relative accuracy of $10^{-4.5}$, which ensures us to achieve relative accuracy of $10^{-3.5}$ in total energy. This reduction in an accuracy of the arithmetic units leads the reduction of

Table 3.2: Comparison of WINE-2 chip and MD-GRAPE chip in DFT/IDFT mode

	WINE-2 chip	MD-GRAPE chip (DFT/IDFT mode)
number of transistors	1.2 M	1 M
design rule	0.5μm	0.6μm
number of pipelines	8	1
clock frequency	80 MHz	35 MHz
total number of clock cycles for DFT calculation with one pipeline	NN_{wv}	$2NN_{wv}$
total number of clock cycles for IDFT calculation with one pipeline	$NN_{wv}/2$	$2NN_{wv}$
speed up	50 $= 8 \times \frac{80}{35} \times \frac{2NN_{wv}}{NN_{wv}+NN_{wv}/2}$	1

the transistors for a pipeline compared with that of an MD-GRAPE chip by a factor of ten. The error analysis of an WINE-2 chip are described in section 4.4 in detail.

2. Higher frequency of pipeline
   A WINE-2 chip is operated by a 2.3 times higher clock frequency (80 MHz) than an MD-GRAPE chip (35 MHz) in DFT/IDFT mode. This can be realized by the reduction of the accuracy as described above.

3. Phase shifter
   A WINE-2 chip adds a phase shift, θ, unlike an MD-GRAPE chip, before a Function Evaluation Unit of the pipeline in IDFT mode (figure 4.3). As a result, the pipeline of a WINE-2 chip needs only one *sine* evaluation in one accumulation in the equation (2.12), while that of an MD-GRAPE chip needs to perform two function evaluations (one *sine* evaluation and one *cosine* evaluation) in one accumulation to get $\vec{f}_{s,i}$ and $\vec{f}_{c,i}$ [compare equation (2.12) with equations (3.24) and (3.25)].

4. Two function evaluators
   The number of function evaluators is increased to two in a WINE-2 chip from one in an MD-GRAPE chip per pipeline. In DFT mode, the pipeline of a WINE-2 chip calculates equations (3.22) and (3.23) concurrently, while that of an MD-GRAPE chip sequentially. As a result, the DFT calculation of a pipeline of a WINE-2 chip is twice faster than that of an MD-GRAPE chip per clock cycle.

   In IDFT mode, the pipeline of a WINE-2 chip performs two summation in equation (2.12) concurrently in one clock cycle [see equation (4.6)], while that of an MD-GRAPE chip performs one summation per clock cycle. As a result, combining the acceleration by the phase shifter described above, the IDFT calculation of a pipeline of a WINE-2 chip is four times faster than that of an MD-GRAPE chip per clock cycle.

Table 3.3: Comparison of MDGRAPE-2 chip and MD-GRAPE chip in FORCE mode

	MDGRAPE-2 chip	MD-GRAPE chip (FORCE mode)
number of transistors	4 M	1 M
design rule	0.25μm	0.6μm
number of pipelines	4	1
clock frequency	100 MHz	35 MHz
`Atom Coefficient RAM`	inside of a chip	outside of a chip
speed up	12 $=4 \times \frac{100}{35}$	1

Design policy of MDGRAPE-2 chip

The design policy of MDGRAPE-2 chip is to enhance performance without major changes in pipelines from MD-GRAPE chip using advanced technology. The author reviewed the micro electronics technology and determined to use four times larger number of transistors and 2.5 times narrower design rule for MDGRAPE-2 chip. As a result, an MDGRAPE-2 chip is 12 times faster than an MD-GRAPE chip for real-space part of calculation. Table 3.3 compares an MDGRAPE-2 chip with an MD-GRAPE chip in FORCE mode. The detailed descriptions of an MDGRAPE-2 chip is presented in the chapter 5. The speed up is achieved by the following three changes.

1. More pipelines in a chip
   An MDGRAPE-2 chip has four pipelines, while an MD-GRAPE chip has one pipeline in FORCE mode. This number of pipelines can be achieved by the advanced design rule (0.25 μm) compared with that of an MD-GRAPE chip (0.6 μm).

2. Higher frequency of pipeline
   An MDGRAPE-2 chip can be operated by three times higher clock frequency (100 MHz) than an MD-GRAPE chip (35 MHz) in FORCE mode. This also can be realized by the advanced design rule described above.

3. `Atom coefficient RAM` in a chip
   The author put `Atom Coefficient RAM` (hereafter ACRAM) into an MDGRAPE-2 chip, while MD-GRAPE has it outside of a chip (on the board). This ensures a high efficiency of calculation of pipelines in the calculation of van der Waals forces.

Table 3.4: Four cases of set up of ACRAM

case	A	B	C	D
efficiency	92%	88%	62%	11%
number of input pins for coefficients for van der Waals	$\sim 5^*$	256	64	64
number of ACRAMs on one board (outside of chips)	0	40	10	1
ACRAM is inside or outside of a chip	inside	outside	outside	outside
one ACRAM	in a chip	per pipeline	per chip	per board

* indices (5 bits) of atom type are loaded

Figure 3.19 shows the expected efficiency of pipelines as a function of the number, N_{aram}, of virtual multiple pipelines (VMPs) that share one ACRAM for the case of a protein atom with 13,620 atoms. Figure 3.18 shows the number of atoms for each atom type for this protein. Here, he assumed that the number of atoms in a cell is 580, which is the case of MDM with a million atoms (see N_{cell} in table 6.3).

The author compared four cases for different set up of ACRAM (see table 3.4). Here, he assumed that a board has ten chips, a chip has four pipelines, one pipeline has 6 VMPs, and the coefficients for van der Waals are loaded at 1/6 clock frequency of the internal clock cycle.

From figure 3.19, the efficiency of the pipeline becomes lower (62% and 11%) in the cases C and D, respectively. The number of ACRAMs and the input pins for a chip become inhibitably larger in the case B, which makes the cost for chips and boards higher. The author concluded from the above discussion that an MDGRAPE-2 chip should have ACRAM inside of a chip. In this case, the efficiency of the pipeline is 92%, which is high enough.

He determined the size of ACRAM so that any pair of coefficients for 32 atom-types of particles may be stored in the RAM. The protein studied in the figure 3.18 has the atoms with 26 atom types. He also studied other two proteins, which has 2,041 and 3,458 atoms, and found that they also have 26 atom types. Therefore, he concluded that 32 atom-types are sufficient for molecular dynamics simulations with proteins and water molecules. You can handle even a larger number of atom types than 32, if MDGRAPE-2 chips are divided into several groups for different sets of atom types of force-receiving particles. In such a case, however, the efficiency of pipeline could be lower.

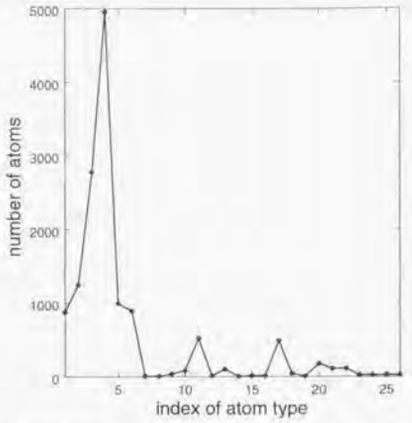


Figure 3.18: Number of atoms is plotted against the index of atom type for the case of a protein atom with 13,620 atoms.

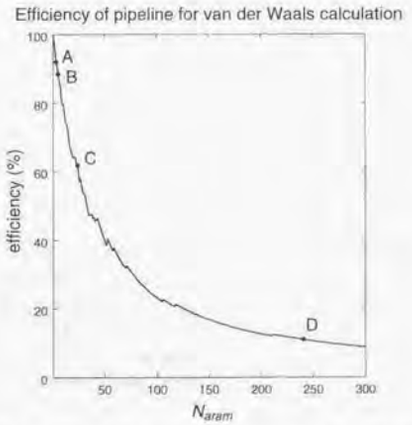


Figure 3.19: Efficiency of pipelines is plotted against the number, N_{aram}, of VMPs per `Atom Coefficient RAM`. A, B, C and D denotes four cases shown in table 3.4.

Chapter 4

Detailed Description of WINE-2 Chip

A WINE-2 chip is a data-flow-type numerical-processor LSI that is specially dedicated for wavenumber-space part of Coulomb interaction in Ewald method.

4.1 General Specification

The wavenumber-space part of Coulomb forces in Ewald method is calculated by the equation:

$$\begin{aligned} \vec{F}_i(\mathrm{wn}) &= \frac{q_i}{2\pi\varepsilon_0 L^3} \sum_{\vec{k}} \frac{\vec{k}}{k^2} \exp(-\pi^2 L^2 k^2/\alpha^2) \\ &\quad \left[\sin(2\pi\vec{k}\cdot\vec{r}_i) \sum_j q_j \cos(2\pi\vec{k}\cdot\vec{r}_j) \right. \\ &\quad \left. - \cos(2\pi\vec{k}\cdot\vec{r}_i) \sum_j q_j \sin(2\pi\vec{k}\cdot\vec{r}_j) \right], \end{aligned} \tag{4.1}$$

where $\vec{r}_i$ is the positional coordinate of particle i, $\vec{k}$ is the wavenumber vector, q_i is the electrostatic charge of particle i, and L is the length of a side of the computational box, ε_0 is the dielectric constant in vacuum, α is a parameter to adjust the computational cost for real and wavenumber part of the Coulomb force calculation.

A WINE-2 chip calculates this equation by the following two steps. In the first step, a WINE-2 chip calculates S_n and C_n as:

$$S_n = \sum_{j=1}^{N} q_j \sin(2\pi\vec{k}_n\cdot\vec{r}_j), \tag{4.2}$$

$$C_n = \sum_{j=1}^{N} q_j \cos(2\pi\vec{k}_n\cdot\vec{r}_j). \tag{4.3}$$

Then, a host computer calculates c_n and θ_n from S_n and C_n as:

$$c_n = \frac{\exp(-\pi^2 L^2 k_n^2/\alpha^2)}{k_n^2}\sqrt{S_n^2 + C_n^2}, \tag{4.4}$$

$$\theta_n = \begin{cases} -\sin^{-1}\left(S_n/\sqrt{S_n^2+C_n^2}\right) & \text{for } C_n \geq 0, \\ \pi + \sin^{-1}\left(S_n/\sqrt{S_n^2+C_n^2}\right) & \text{for } C_n < 0. \end{cases} \tag{4.5}$$

In the second step, a WINE-2 chip calculates $\vec{f}_i$ as:

$$\vec{f}_i = \sum_{n=1}^{N_{wv}/2} \left[c_{2n-1}\sin(2\pi\vec{k}_{2n-1}\cdot\vec{r}_i + \theta_{2n-1})\cdot\vec{k}_{2n-1} + c_{2n}\sin(2\pi\vec{k}_{2n}\cdot\vec{r}_i + \theta_{2n})\cdot\vec{k}_{2n}\right], \tag{4.6}$$

where N_{wv} is the the number of wavenumber vectors to be taken into account. Then, a host computer calculates the wavenumber-space part of Coulomb force, $\vec{F}_i(\mathrm{wn})$, from $\vec{f}_i$ as:

$$\vec{F}_i(\mathrm{wn}) = \frac{q_i}{2\pi\varepsilon_0 L^3}\vec{f}_i. \tag{4.7}$$

A WINE-2 chip performs Discrete Fourier Transform (DFT) in the first step (equations 4.2 and 4.3), and Inverse DFT (IDFT) in the second step (equation 4.6).

4.1.1 Clock cycle

A WINE-2 chip operates at a clock cycle higher than 80MHz.

4.1.2 Numerical expression

A WINE-2 chip uses two's complement format in mathematical operations.

4.2 Structure

A WINE-2 chip is composed of eight Pipelines, one Controller, and one Interface as shown in figure 4.1. It has a virtual multiple pipelines (VMP) architecture in which eight different calculations are performed in every eight internal clock cycles (one VMP cycle). A pipeline looks like eight pipelines with an eight times slower clock cycle.

A Pipeline has two modes, those are DFT and IDFT modes. In DFT mode, a Pipeline performs discrete Fourier transformation on eight different wavenumber vectors from one j-particle and accumulates them in every VMP cycle (eight internal clock cycles). Figure 4.2 shows the block diagram of a pipeline in DFT mode. The pipeline calculates inner product of vectors $\vec{r}_j$ and $\vec{k}_n$, and then evaluates *sine* and *cosine* of them. It multiplies the results to q_j, and accumulates the products to get $S_n + C_n$ and $S_n - C_n$. A host computer

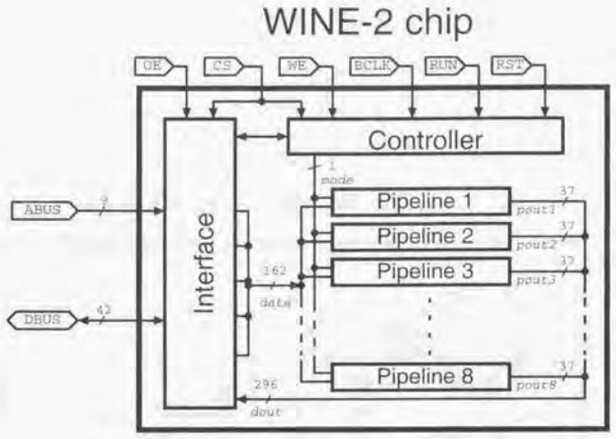


Figure 4.1: Block diagram of WINE-2 chip

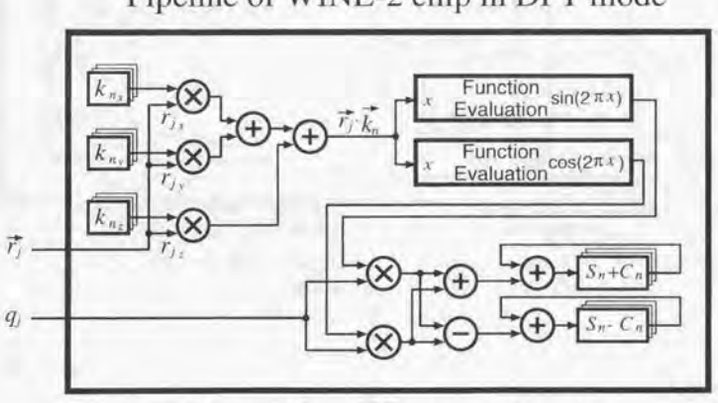


Figure 4.2: Block diagram of Pipeline in DFT mode

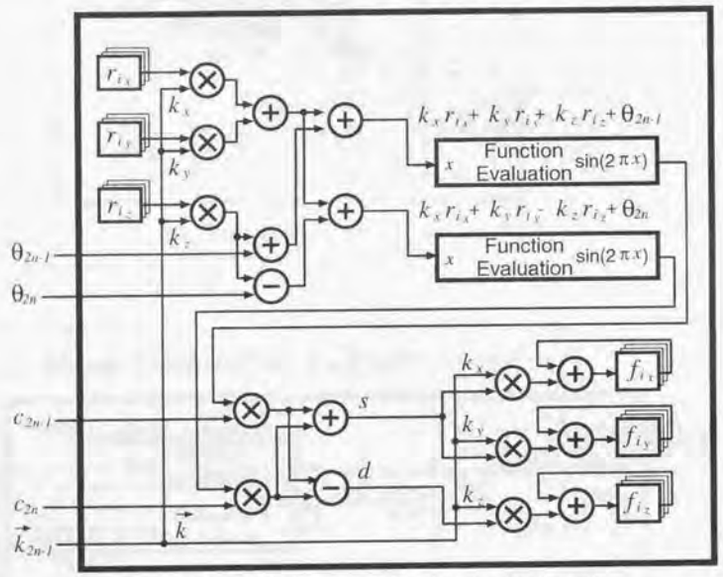


Figure 4.3: Block diagram of Pipeline in IDFT mode

calculates S_n and C_n from $S_n + C_n$ and $S_n - C_n$. In DFT mode, 64 wavenumber vectors ($\vec{k}_n$) are loaded to a WINE-2 chip before starting the calculation.

In IDFT mode, a pipeline performs inverse discrete Fourier transformation on eight different particles from two wavenumber vectors and accumulate them in every VMP cycle (eight internal clock cycles). Figure 4.3 shows the block diagram of a pipeline in IDFT mode. A pipeline calculates two inner products $\vec{k}_{2n-1} \cdot \vec{r}_{i_1}$ and $\vec{k}_{2n} \cdot \vec{r}_{i_1}$, then adds them to θ_{2n-1} and θ_{2n}, and finally evaluates the value of *sine* function, respectively. Here, the author arranges the order of wavenumber vectors so that $\vec{k}_{2n}$ are different from $\vec{k}_{2n-1}$ only in the sign of the z-components of them. The pipeline multiplies the results to c_{2n-1} and c_{2n}, and obtains their sum, s, and their difference, d, of the two products. The pipeline calculates $s \times k_{2n-1_x}$, $s \times k_{2n-1_y}$, and $d \times k_{2n-1_z}$ and accumulates them to get $f_{i,x}$, $f_{i,y}$, and $f_{i,z}$. In IDFT mode, positional coordinates of 64 particles ($\vec{r}_i$) are loaded to a WINE-2 chip before starting the calculation.

Controller controls the multiplexers and registers in the chip. Interface receives positions ($\vec{r}_j$) and charges (q_j) of j-particles from DBUS and ABUS in DFT mode, and wavenumber vectors (k_{2n-1}), coefficients (c_{2n-1} and c_{2n}), and phase-shifts (θ_{2n-1} and θ_{2n}) from DBUS and ABUS in IDFT mode. These data are stored in registers and kept in the next eight clock cycles. Interface also outputs the results of DFT or IDFT.

4.2.1 Pipeline

A pipeline is composed of one Data Selector Unit, one Inner Product Unit, two Function Evaluator Units, and one Accumulator Unit as shown in figure 4.4. A Data Selector Unit selects the data for an Inner Product Unit. An Inner Product Unit calculates inner product of $\vec{k}_n$ and $\vec{r}_j$, and adds the result to the phase shift from Interface. A Function Evaluator Unit calculates *sine* function of the result of the Inner Product Unit. An Accumulator Unit multiplies the coefficients c_1 and c_2 from the Interface to the output of the Function Evaluator Units, and accumulates the products.

Data Selector Unit

A Data Selector Unit selects the data for an Inner Product Unit according to the *mode* signal from Controller (see table 4.1). It is composed of three 8-word 21-bit RAMs (8W21RAXI, 8W21RAYI and 8W21RAZI), two multiplexers (21MXKN and 63MXRJ) as shown in figure 4.5. A multiplexer (21MXKN) selects $\vec{k}_n$ from Interface in DFT mode, while it selects $\vec{k}_i$ from 8W21RAXI in IDFT mode. A multiplexer (63MXRJ) selects $\vec{r}_j$ from Interface in DFT mode, while it selects $\vec{r}_i$ from 8W21RAXI, 8W21RAYI and 8W21RAZI. These two multiplexers are controlled by a (*mode*) signal from Controller.

Inner Product Unit

An Inner Product Unit calculates inner product of a position of a particle ($\vec{r}$) and a wavenumber vector ($\vec{k}$) and adds the result to the phase-shifts (θ_1 and θ_2) from Interface. An Inner Product Unit is composed of three 7-bit × 21-bit multipliers (721IMXI,

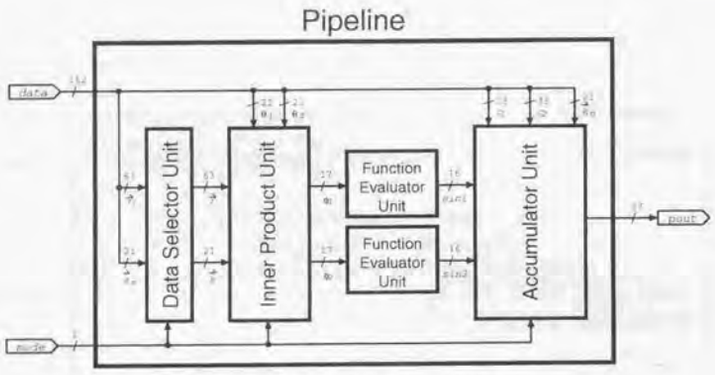


Figure 4.4: Block diagram of Pipeline

Table 4.1: Function of Data Selector Unit

mode	DFT	IDFT
mode signal	0	1
21MXKN	$\vec{k}_i$	$\vec{k}_n$
63MXRJ	$\vec{r}_j$	$\vec{r}_i$

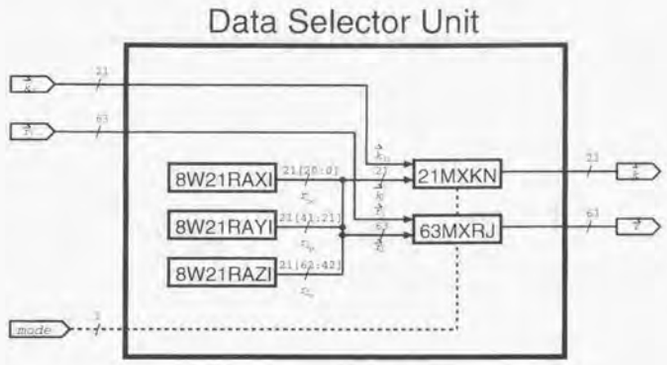


Figure 4.5: Block diagram of Data Selector Unit

721IMYI and 721IMZI), four 21-bit adders (21IAXY, 21IAZT, 21IAP1 and 21IAP2), one 17-bit adder (17IAP4), one 21-bit subtracter (21ISZT), and one multiplexer (17MXP2)

Three multipliers (721IMXI, 721IMYI and 721IMZI) and three adders (21IAXY, 21IAZT and 21IAP1) calculate ϕ_1 $(= k_x r_x + k_y r_y + k_z r_z + \theta_1)$ in both of DFT and IDFT modes. In DFT mode, θ_1 is set zero from outside of a chip to calculate ϕ_1 $(= k_x r_x + k_y r_y + k_z r_z)$. The function of the other part depends on the mode. In DFT mode, the other adder (17IAP4) calculates another outputs ϕ_2 $(= \phi_1 + 1/4)$. In IDFT mode, a subtracter (21ISZT) and an adder (21IAP2) calculate ϕ_2 $(= k_x r_x + k_y r_y - k_z r_z + \theta_2)$. An Inner Product Unit outputs ϕ_1 and ϕ_2 to two Function Evaluator Units, respectively. Note that overflows in this unit are ignored because periodic boundary condition is imposed.

Function Evaluator Unit

A Function Evaluator Unit evaluates the value of *sine* function corresponding to the value of the input, ϕ_1 or ϕ_2, from an Inner Product Unit using the second-order interpolation:

$$\sin\phi = (e_2\Delta x + e_1)\Delta x + e_0, \tag{4.8}$$

where Δx is the difference from the center of expansion, and e_0, e_1 and e_2 are the coefficients of the polynomial.

A Function Evaluator Unit is composed of a Segmentation Part, a Polynomial Part, and a 16-bit sign inverter (16IRSN) as shown in figure 4.7. A Segmentation Part generates Δx, the address (*adr*) of the ROMs, and a sign signal (*sign*) from the 17 bits data, ϕ, from an Interface (figure 4.8). A Polynomial Part has one 32-word 8-bit ROM (32W8ROC2), one 32-word 12-bit ROM (32W12ROC1), one 32-word 17-bit ROM (32W17ROC0), two

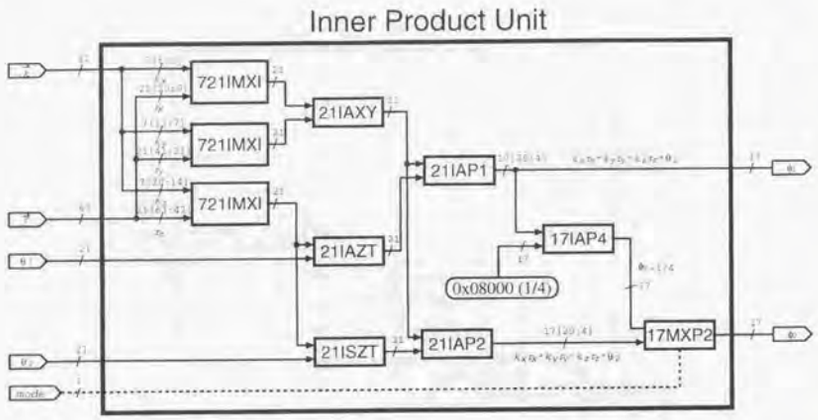


Figure 4.6: Block diagram of Inner Product Unit

multipliers (88IMC2 and 1011IMC1), and two adders (12IAC1 and 17IAC0) as shown in figure 4.9.

The coefficients e_0, e_1, and e_2 are stored in 32W17ROC0, 32W12ROC1, and 32W17ROC0, respectively. A sign inverter (16IRSN) inverts the sign of the output of Polynomial Part according to the signal (*sign*) from Segmentation Part. The result ($\sin\phi$) is outputted to Accumulator Unit.

Accumulator Unit

An Accumulator Unit is composed of two 16-bit × 18-bit multipliers (1618IMS1 and 1618IMS2), three 7-bit × 20-bit multipliers (720IMFX, 720IMFY and 720IMFZ), three 37-bit adders (37IAFX, 37IAFY and 37IAFZ), one 20-bit adder (20IAS1), one 20-bit subtracter (20ISS2), three 7-bit multiplexers (7MXFX, 7MXFY and 7MXFZ), one 37-bit multiplexer (37MXFP), and three 8-word 37-bit RAMs (8W37RAFX, 8W37RAFY and 8W37RAFZ).

Two 16-bit × 18-bit multipliers (1618IMS1 and 1618IMS2) multiply two outputs from two Function Evaluator Units and the coefficients, c_1 and c_2, respectively. The sum, s, and difference, d, of the two products are calculated by a 20-bit adder (20IAS1) and a 20 bit subtracter (20ISS2), respectively. In DFT mode, s and d are accumulated by two 37-bit adders (37IAFX and 37IAFZ). In IDFT mode, three 7-bit multiplexers (7MXFX, 7MXFY, and 7MXFZ) calculate $k_{n_x}s$, $k_{n_y}s$ and $k_{n_z}d$, respectively, where k_{n_x}, k_{n_y} and k_{n_z} are the three components of $\vec{k_n}$ supplied by Interface. Table 4.2 summarizes the function of Accumulator Unit.

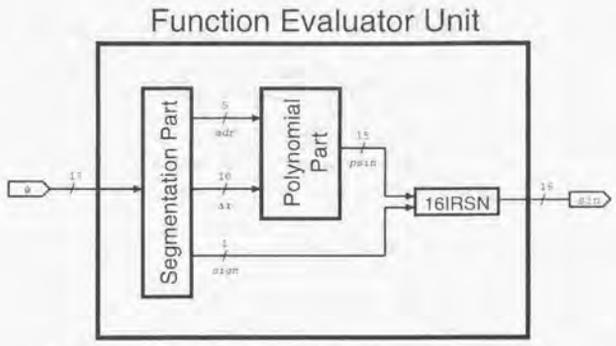
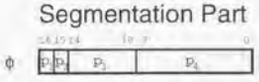


Figure 4.7: Block diagram of Function Evaluator Unit



p_1	p_2	*sign*	*adr*	Δx
0	0	0	p_3	p_4 XOR 512
	1	0	$16 - p_3$	$(512 - p_4)$ XOR 512
1	0	1	$p_3 - 16$	$(p_4 - 512)$ XOR 512
	1	1	$32 - p_3$	$(1024 - p_4)$ XOR 512

Figure 4.8: The outputs (*sign*, *adr* and Δx) of Segmentation Part are determined depending on the two most significant bits, p_1 and p_2, of ϕ. XOR denotes exclusive OR.

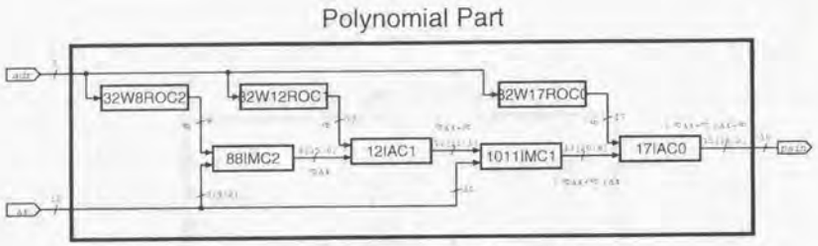


Figure 4.9: Block diagram of Polynomial Part

Table 4.2: Function of Accumulator Unit

mode	DFT	IDFT
input	$sin1$, $sin2$ c_1, c_2	$sin1$, $sin2$ c_1, c_2 $\vec{k}_n$
output of 20IAS1 (s) output of 20ISS2 (d)	$c_1\ sin1 + c_2\ sin2$ $c_1\ sin1 - c_2\ sin2$	
accumulation at 37IAFX accumulation at 37IAFY accumulation at 37IAFZ	s s d	$s\ k_{n_x}$ $s\ k_{n_y}$ $d\ k_{n_z}$

A 37-bit multiplexer (37MXFP) selects the output (*pout*) to Interface when one reads the results of accumulation.

4.2.2 Interface

Interface has seven flip-flops (63FFRJ, 21FFKN, 21FFT1, 21FFT2, 21FFC1, 21FFC2 and 37FFDO) and one multiplexer (8W37MXDO). In DFT mode, Interface keeps $\vec{r}_j$, θ_1 and θ_2 from ABUS and DBUS in a VMP cycle, and outputs them to Pipelines (see table 4.3). In IDFT mode, Interface keeps $\vec{k}_n$, θ_1, θ_2, c_1 and c_2 from ABUS and DBUS in a VMP cycle, and outputs them to Pipelines (see table 4.3). Interface also outputs the results of accumulation in Pipelines to DBUS through 37-bit multiplexer (37MXDO) when both of OE pin and CS pin are active.

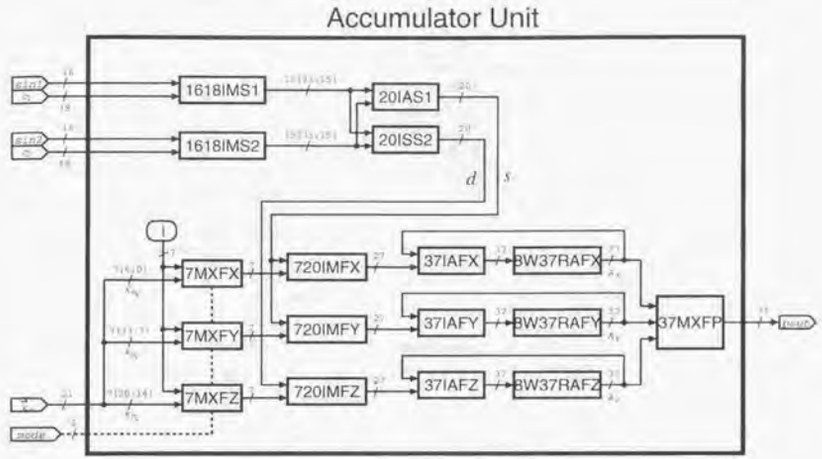


Figure 4.10: Block diagram of Accumulator Unit

Table 4.3: Input data for DFT and IDFT modes

mode	DFT		IDFT		
phase in a VMP cycle*	F	S	F		S
ABUS	bit 0-8 of c_1	bit 9-17 of c_1	bit 0-8 of c_1		bit 9-17 of c_1
bit 0-20 of DBUS	r_{j_x}	r_{j_z}	c_2		θ_2
bit 21-41 of DBUS	r_{j_y}	θ_1	bit 21-27 bit 28-34 bit 35-41	k_{n_x} k_{n_y} k_{n_z}	θ_1

*F and S denote the first and second half a VMP cycles. See figure 4.13.

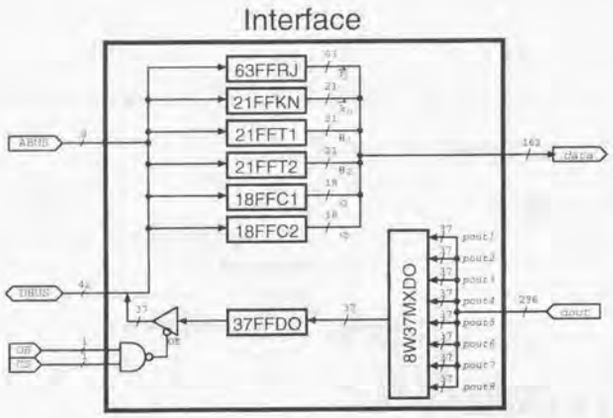


Figure 4.11: Block diagram of Interface

4.2.3 Controller

Controller has one 1-bit mode register (1RGMD), which determine the mode of a chip (DFT or IDFT mode). It also controls the read and write operation of the chip.

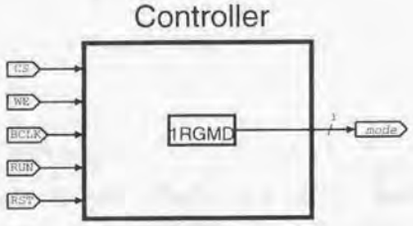


Figure 4.12: Block diagram of Controller

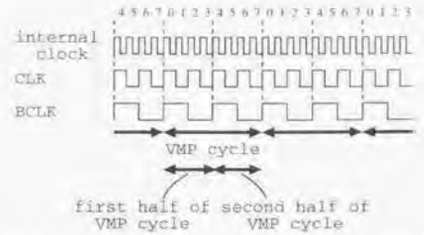


Figure 4.13: First and second half of VMP cycles

4.3 External Interface

4.3.1 External pins

Table 4.4 shows the pin assignments of a WINE-2 chip. ABUS has 9 pins and DBUS has 42 pins. When RUN pin is high, which indicates that the chip is in calculation, c_1 is supplied through ABUS in DFT and IDFT modes, and r_{j_x}, r_{j_y}, r_{j_z} and θ_1 are supplied through DBUS in DFT mode, and k_n, θ_1, θ_2 and c_2 in IDFT mode. These data are loaded in every eight internal clock cycles of a chip. When RUN pin is low, which indicates that the chip is not in calculation, one can access through DBUS RAMs and registers, whose addresses are specified by the ABUS, in a chip.

4.3.2 Chip registers and RAMs

An address map of the chip is shown in tables 4.5 and 4.6 when one writes data to a chip or reads data from a chip, respectively. Seven least significant bits of ABUS is used to specify the address of RAMs and registers in a chip. All the bits (42 bits) of DBUS are used to write data to a chip, while 37 least significant bits are used to read data from a chip.

Note that one can write to or read from different RAMs with the same address but with the different timing of clock cycles. For example, when the address is specified as 03, one can read the third word of 8W37RAFX in Pipeline 1 in the first half VMP cycle (0-3 internal clock cycles) and the seventh word of 8W37RAFX in Pipeline 1 in the second half of VMP cycle (4-7 internal clock cycles). Figure 4.13 shows a first and a second half VMP cycles.

Table 4.4: External Pins

name	I/O	description
ABUS[8:0]	In	Address bus to access to the RAMs and registers in the chip.
		Data bus to supply c_1 in DFT and IDFT modes.
DBUS[41:0]	In	Data bus to access to the RAMs and registers in the chip.
		Data bus to supply r_{j_x}, r_{j_y}, r_{j_z} and θ_1 in DFT mode, and k_n, θ_1, θ_2 and c_2 in IDFT mode.
	Out	Data bus to output results of accumulation (8W37RAFX, 8W37RAFY and 8W37RAFZ).
CS	In	Chip select signal. When it is high, read/write access to the chip is permitted.
WE	In	Write enable signal. High is write and low is read.
OE	In	Output enable signal. When it is low, DBUS turns to be high impedance.
RST	In	Reset signal. When it is high, 6 RAMs (8W21RAXI, 8W21RAYI, 8W21RAZI, 8W37RAFX, 8W37RAFY and 8W37RAFZ) and one register (1RGMD) are cleared.
CLK	In	Clock signal. Supply the clock with half of the internal clock cycle.
BCLK	In	Bus clock signal. Supply the clock signal for interfacing between the chip and other devices on the board, which has half frequency of CLK signal.
RUN	In	Calculation signal. While it is high, chip loads data from ABUS and DBUS, and performs accumulations. While it is low, writing to or reading from a chip is allowed.

Table 4.5: Address map of WINE-2 chip (write)

address (hex)	bit	phase in a VMP cycle*	registers, RAMs and control flags		
			name	Pipeline	word
00	0-20	F	8W21RAXI	1	0
		S			4
	21-41	F		2	0
		S			4
01	0-20	F		1	1
		S			5
	21-41	F		2	1
		S			5
02	0-20	F		1	2
		S			6
	21-41	F		2	2
		S			6
03	0-20	F		1	3
		S			7
	21-41	F		2	3
		S			7
04	0-20	F		3	0
		S			4
	21-41	F		4	0
		S			4
05	0-20	F		3	1
		S			5
	21-41	F		4	1
		S			5
...				...	
0f	0-20	F		7	3
		S			7
	21-41	F		8	3
		S			7
10-1f			8W21RAYI		
20-2f			8W21RAZI		
40	0-7	F or S	chip mode (0x24:DFT mode, 0x11:IDFT mode)		
	8	F and S	clear all words of 8W37RAFX, 8W37RAFY, and 8W37RAFZ in any Pipeline (0:do nothing, 1:clear)		

*F and S denote first and second half VMP cycles. See figure 4.13.

Table 4.6: Address map of WINE-2 chip (read)

address (hex)	phase in a VMP cycle	registers, RAMs and control flags		
		name	Pipeline	word
00	F	8W37RAFX	1	0
	S			4
01	F			1
	S			5
02	F			2
	S			6
03	F			3
	S			7
04-07			2	
...			...	
1c-1f			8	
20-3f		8W37RAFY		
40-5f		8W37RAFZ		

4.3.3 Timing chart of the chip

Write cycle

Timing chart of $2n$ words write is shown in figure 4.14. On every BCLK cycle, address and data are given to the chip. Note that the addresses, $A_0, A_1, \ldots, A_{2n}$, are not necessary to be contiguous.

Read cycle

Timing chart of $2n$ words read is shown in figure 4.15. The chip starts to reply the input address every BCLK cycle after a certain delay. Note that the addresses, A_0, A_1, ..., A_{2n}, are not necessary to be contiguous.

Calculation cycle

Timing chart of calculation of the chip is shown in figure 4.16. In this figure, DFT mode is assumed. One can freeze the calculation temporary, and restart the calculation. In figure 4.16, the chip freezes the calculation after loading one set of data from ABUS and DBUS, and restarts the calculation after two cycle of BCLK. This control is done by RUN pin.

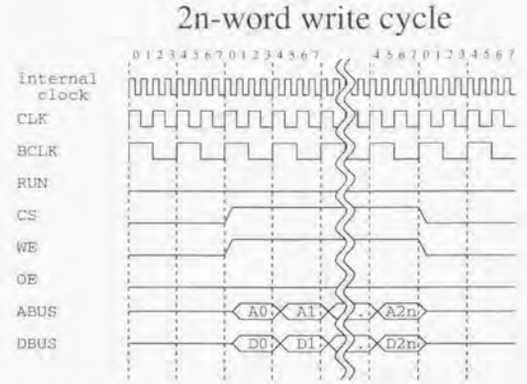


Figure 4.14: Timing chart of writing $2n$ words to the chip. The addresses A_0, A_1, ..., A_{2n}, are not necessary to be contiguous. Note that RUN and OE are low.

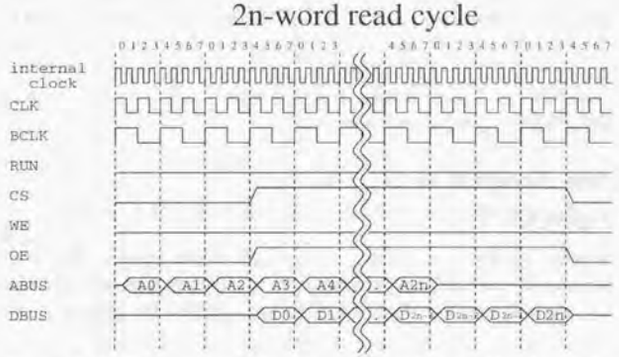


Figure 4.15: Timing chart of reading $2n$ words from the chip. The addresses, A_0, A_1, ..., A_{2n}, are not necessary to be contiguous. Note that RUN and WE are low.

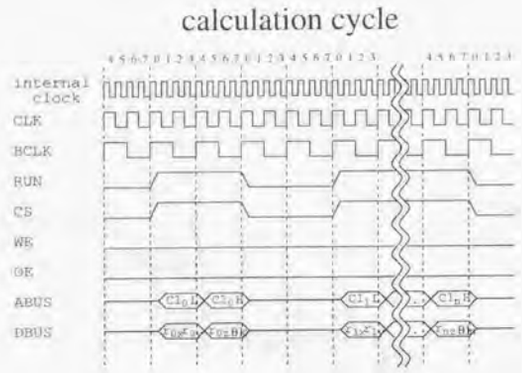


Figure 4.16: Timing chart while the chip is in calculation (RUN pin is high.

4.4 Error Analysis of WINE-2 Chip

The author designed the pipeline of a WINE-2 chip so that its relative accuracy is about $10^{-4.5}$. In subsection 4.4.1, he theoretically discusses the error in DFT mode, and studies it by numerical experiments with Monte Carlo simulation. He found that his theoretical estimates is in good agreement with numerical experiments. In subsection 4.4.2, the author presents further theoretical discussions on the error in IDFT mode. He found that the relative error of the pipeline of a WINE-2 chip is less than $10^{-4.5}$ for wavenumber-space part of force calculation. In this section, $V(x)$ denotes the mean square of a parameter x, δx denotes the error in x, and $L = 1$ is assumed for simplicity.

4.4.1 Error analysis in DFT mode

Theoretical analysis

The author estimates the error produced in each unit of the pipeline of a WINE-2 chip in DFT mode as follows. The summary of the error analysis in this part is presented in the table 4.7. Here, variables used in this part is summarized in the table 4.7.

1. Inner Product Unit
   The mean square of error, $V(\delta\phi)$, of the output, ϕ, of Inner Product Unit (figure 4.6) is evaluated as:

$$V(\delta\phi) = V(\delta r_x)k_x^2 + V(\delta r_y)k_y^2 + V(\delta r_z)k_z^2 + 4.849 \times 10^{-12}, \quad (4.9)$$
$$= 1.896 \times 10^{-14}k^2 + 4.849 \times 10^{-12}, \quad (4.10)$$

where k denotes the length of a wavenumber vector, i.e., $k^2 = k_x^2 + k_y^2 + k_z^2$

2. Function Evaluator Unit
   The mean square of error, $V(\delta sin)$, of the output of Function Evaluator Unit, sin, is evaluated as:

$$V(\delta sin) = V(\delta y_3) + V(\delta e_0) + 7.761 \times 10^{-11}, \quad (4.11)$$
$$= 1.974 \times 10V(\delta\Delta x) + 1.296 \times 10^{-10}, \quad (4.12)$$

where y_3, e_0 and Δx are defined in table 4.7.

3. Accumulator Unit
   The mean square of error, $V(\delta y_5)$, of the output, y_5, of the multiplier (1618IMS1 or 1618IMS2 in figure 4.10) in Accumulator Unit is evaluated as:

$$V(\delta y_5) = V(sin)V(\delta q) + V(\delta sin)V(q), \quad (4.13)$$
$$= 2.425 \times 10^{-12} + V(\delta sin)V(q), \quad (4.14)$$

where y_5 is $sin\, q$ and $V(q)$ is calculated by:

$$V(q) = \frac{1}{3}\frac{1 - (\frac{q_{min}}{q_{max}})^3}{1 - \frac{q_{min}}{q_{max}}}. \quad (4.15)$$

Here, q_{max} and q_{min} are the maximum and minimum absolute charge of particles, respectively, and the author assumed that charges are uniformly distributed in the range of $(-q_{max}, -q_{min}]$, $[q_{min}, q_{max})$.

The mean square of error, $V(\delta y_5)$, of the one summation of the DFT calculation is evaluated from equations 4.10, 4.12, and 4.14 as:

$$V(\delta y_5) = 1.637 \times 10^{-13}k^2 + 1.010 \times 10^{-10}. \quad (4.16)$$

Here, we assumed that the ratio of the minimum and maximum absolute charges, q_{min}/q_{max} is 1/4.

The relative root mean square of error, $\sqrt{V(\delta\sqrt{S_n^2 + C_n^2})/V(\sqrt{S_n^2 + C_n^2})}$, of the results of DFT calculation is evaluated as:

$$\sqrt{V(\delta\sqrt{S_n^2 + C_n^2})/V(\sqrt{S_n^2 + C_n^2})} = \sqrt{V(\delta y_5)/V(y_5)}, \quad (4.17)$$
$$= \sqrt{7.482 \times 10^{-13}k^2 + 4.616 \times 10^{-10}}. \quad (4.18)$$

The theoretical error [equation (4.18)] is plotted in figure 4.17 by dashed curve.

Numerical experiment

The numerical experiment is done as follows. First, the author randomly put 100 particles as they are uniformly distributed in the system. Second, he randomly assigned the charge of particles in the range of $(-1, -1/4]$, $[1/4, 1)$. Third, the author randomly selected 1,000 wavenumber vectors with a size of k. Finally, he performed DFT calculation and determined the relative accuracy of the result.

The result of the numerical experiment is plotted in figure 4.17 by solid lines. The numerical experiment is in good agreement with the theoretical estimates in DFT mode.

4.4.2 Error analysis in IDFT mode

The author estimates the error produced in each unit of the pipeline of a WINE-2 chip in IDFT mode as follows.

1. Inner Product Unit
   The mean square of error, $V(\phi)$, of the output, ϕ, of Inner Product Unit is evaluated as:

$$V(\delta\phi) = 1.896 \times 10^{-14}(k^2+1) + 4.849 \times 10^{-12}, \quad (4.19)$$
$$\sim 1.896 \times 10^{-14}k^2 + 4.849 \times 10^{-12}. \quad (4.20)$$

   Here, the author neglected the error from the phase shift, θ, since it is less than 1%.

2. Function Evaluator Unit
   The mean square of error, $V(\delta sin)$, of the output of Function Evaluator Unit is evaluated as:

$$V(\delta sin) = 3.743 \times 10^{-13}k^2 + 2.253 \times 10^{-10}. \quad (4.21)$$

3. Accumulator Unit
   The mean square of error, $V(\delta sin\sqrt{S_n^2+C_n^2})$, of the output of the multiplier (1618IMS1 or 1618IMS2 in figure 4.10) in Accumulator Unit is evaluated as:

$$V(\delta sin\sqrt{S_n^2+C_n^2}) = V(\sqrt{S_n^2+C_n^2})V(\delta sin) + V(\delta\sqrt{S_n^2+C_n^2})V(sin). \quad (4.22)$$

   The relative mean square of error, $V(\delta sin\sqrt{S_n^2+C_n^2})/V(sin\sqrt{S_n^2+C_n^2})$, of the one summation of the IDFT calculation is calculated from equations (4.18) and (4.22):

$$V(\delta sin\sqrt{S_n^2+C_n^2})/V(sin\sqrt{S_n^2+C_n^2}) = \frac{V(\delta sin\sqrt{S_n^2+C_n^2})}{V(sin)V(\sqrt{S_n^2+C_n^2})}, \quad (4.23)$$
$$= 1.497 \times 10^{-12}k^2 + 9.122 \times 10^{-10} \quad (4.24)$$

   Here, the author used $V(sin) = \frac{1}{2}$.

On the other hand, the mean square, $V(f)$, of the force by IDFT calculation is evaluated as:

$$V(f) = \sum^{k<k_{cut}} 2\pi k^2 \frac{\exp(-\pi^2k^2/\alpha^2)}{k} V(sin\sqrt{S_n^2+C_n^2}), \quad (4.25)$$
$$= \sum^{k<k_{cut}} \pi k \exp(-\pi^2k^2/\alpha^2) V(\sqrt{S_n^2+C_n^2}), \quad (4.26)$$

where f is determined as:

$$f = \frac{\pi^2\varepsilon_0}{q_i}|\vec{F}_i(\mathrm{wn})|. \quad (4.27)$$

The mean square of error, $V(\delta f)$, of the force by IDFT calculation is calculated by equations (4.22) and (4.24):

$$V(\delta f) = \sum^{k<k_{cut}} 2\pi k^2 \frac{\exp(-\pi^2k^2/\alpha^2)}{2k} V(\delta sin\sqrt{S_n^2+C_n^2}), \quad (4.28)$$
$$= \sum^{k<k_{cut}} \pi k \exp(-\pi^2k^2/\alpha^2) V(\sqrt{S_n^2+C_n^2}) \times (1.497 \times 10^{-12}k^2 + 9.122 \times 10^{-10}). \quad (4.29)$$

The relative root mean square of error, $\sqrt{V(\delta f)/V(f)}$, of the force by IDFT calculation is the theoretical estimate in figure 4.18. The relative error of the force calculated by the pipeline of a WINE-2 chip is estimated as less than $10^{-4.5}$ when $\alpha \leq 30$, which will be used when $N < 10^6$ in MDM system.

4.4.3 Drift of total energy in molecular dynamics simulations

Three MD simulations are performed to examine the drift in total energy (see table 4.9). All of the simulation is performed with TIPS2 water molecules under NVE ensemble (total energy should conserve) with a time step of 0.5 femto-second. The software emulator of a WINE-2 chip is used in wavenumber-space part of the calculation. In simulation A and B, calculations by double precision floating point format are also performed to compare with the software emulator. The drift in the total energy of these simulations are shown in figures 4.19, 4.20, and 4.21, respectively.

Although the total energy drifts as much as 0.3% in simulation A and B (figures 4.19 and 4.20), the difference between double precision and software emulator calculation is satisfactory small. The rather large drifts in total energy in these simulations is caused by the small number of atoms (64 and 256 water molecules, respectively). In fact, the drift in total energy is as small as 0.01% in simulation C with a larger number of atoms (2,560 water molecules) because of the cancelation of error. Since the target of MDM is very large system with more than a hundred thousand atoms, the author can safely conclude that the accuracy of an WINE-2 chip is sufficient for molecular dynamics simulations.

Table 4.7: Variables for error analysis

variable	sign bit	number of bits above the decimal point	number of bits under the decimal point	description
r_x, r_y, r_z	no	0	21	coordinate of position vector
k_x, k_y, k_z	yes	6	0	coordinate of wavenumber vector
ϕ_1, ϕ_2	no	0	17	input of function evaluator
Δx	yes	0	9	lower input of 1011IMC1
Δx_2	yes	0	7	lower input of 88IMC2
e_2	yes	5	2	output of 32W8ROC2
e_1	no	3	9	output of 32W12ROC1
e_0	no	0	17	output of 32W17ROC0
y_1	yes	0	7	output of 88IMC2 ($e_2 \Delta x_2$)
y_2	no	3	9	output of 12IAC1 ($y_1 + e_1$)
y_3	yes	0	13	output of at 1011IMC1 ($y_2 \Delta x$)
sin	no	0	15	output of 17IAC0 ($y_3 + e_0$)
q	yes	0	17	charge of a particle; lower input of 1618IMS1 or 1618IMS2
y_5	yes	0	18	output of 1618IMS1 or 1618IMS2 ($sin\ q$)

Table 4.8: Error analysis in DFT mode

Unit	mean square	value
Inner Product Unit	$V(\delta r_x), V(\delta r_y), V(\delta r_z)$	1.896×10^{-14}
	$V(\delta k_x), V(\delta k_y), V(\delta k_z)$	0.0
	$V(\delta\phi_1), V(\delta\phi_2)$	$V(\delta r_x)k_x^2 + V(\delta r_y)k_y^2 + V(\delta r_z)k_z^2 + 4.849 \times 10^{-12}$ $= 1.896 \times 10^{-14}k^2 + 4.849 \times 10^{-12}$
Function Evaluator Unit	$V(\Delta x)$	5.085×10^{-6}
	$V(\delta\Delta x)$	$V(\delta\phi)$
	$V(\Delta x_2)$	$V(\Delta x)$
	$V(\delta\Delta x_2)$	$V(\delta\Delta x) + 7.762 \times 10^{-11}$
	$V(e_2)$	1.949×10^2
	$V(\delta e_2)$	5.209×10^{-3}
	$V(e_1)$	1.974×10^1
	$V(\delta e_1)$	1.272×10^{-6}
	$V(e_0)$	5.000×10^{-1}
	$V(\delta e_0)$	1.940×10^{-11}
	$V(y_1)$	$V(e_2)V(\Delta x_2)$ $= 9.911 \times 10^{-4}$
	$V(\delta y_1)$	$V(e_2)V(\delta\Delta x_2) + V(\delta e_2)V(\Delta x_2) + 1.272 \times 10^{-6}$ $= 1.949 \times 10^2 V(\delta\Delta x) + 1.314 \times 10^{-6}$
	$V(y_2)$	$V(y_1) + V(e_1)$ $= 1.974 \times 10$
	$V(\delta y_2)$	$V(\delta y_1) + V(\delta e_1)$ $= 1.949 \times 10^2 V(\delta\Delta x) + 2.586 \times 10^{-6}$
	$V(y_3)$	$V(y_2)V(\Delta x)$ $= 1.004 \times 10^{-4}$
	$V(\delta y_3)$	$V(y_2)V(\delta\Delta x) + V(\Delta x)V(\delta y_2) + 1.940 \times 10^{-11}$ $= 1.974 \times 10 V(\delta\Delta x) + 3.255 \times 10^{-11}$
	$V(sin)$	$V(y_3) + V(e_0)$ $= 5.001 \times 10^{-1}$
	$V(\delta sin)$	$V(\delta y_3) + V(\delta e_0) + 7.761 \times 10^{-11}$ $= 1.974 \times 10 V(\delta\Delta x) + 1.296 \times 10^{-10}$
Accumulator Unit	$V(q)$	$\frac{1}{3}\frac{1-(\frac{q_{min}}{q_{max}})^3}{1-\frac{q_{min}}{q_{max}}}$,
	$V(\delta q)$	4.849×10^{-12}
	$V(y_5)$, $y_5 = sin\, q$	$V(y_4)V(q)$
	$V(\delta y_5)$	$V(sin)V(\delta q) + V(\delta sin)V(q)$ $= 2.425 \times 10^{-12} + V(\delta sin)V(q)$
Total	$V(y_5)$	2.188×10^{-1}
	$V(\delta y_5)$	$1.637 \times 10^{-13}k^2 + 1.010 \times 10^{-10}$

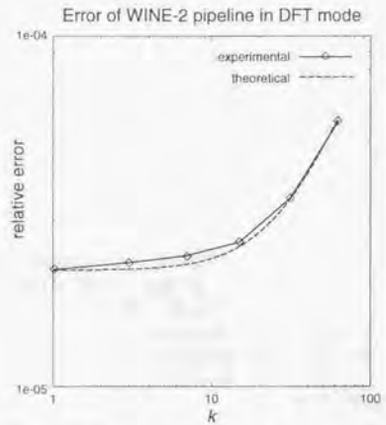


Figure 4.17: Error of pipeline in DFT calculation is plotted against length, k, of a wavenumber vector.

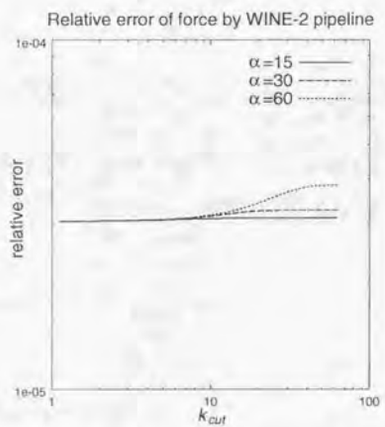


Figure 4.18: Error of pipeline in IDFT calculation is plotted against k_{cut}

Table 4.9: Three simulations of water molecules

simulation	number of water molecules	number of time steps
A	64	3,000
B	256	30,000
C	2,560	1,000

Figure 4.19: The drift in total energy in simulation A with 64 water molecules is plotted against time step. The difference between WINE-2 emulator (solid curve) and double precision (dashed curve) calculation is satisfactory small (less than 0.3%).

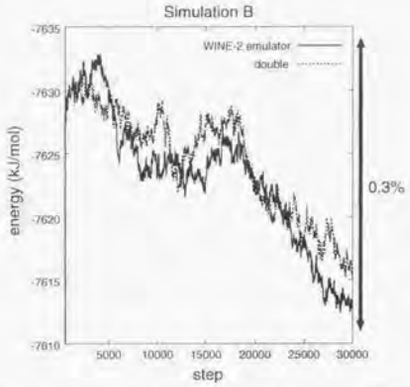


Figure 4.20: The drift in total energy in simulation B with 256 water molecules is plotted against time step. The difference between WINE-2 emulator (solid curve) and double precision (dashed curve) calculation is satisfactory small (less than 0.1%).

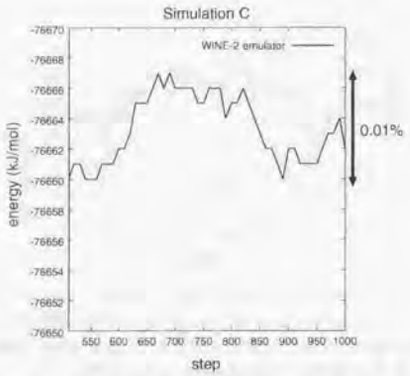


Figure 4.21: The drift in total energy in simulation C with 2,560 water molecules is plotted against time step. The drift is as small as 0.01% because of the cancelation of the error.

Chapter 5

Detailed Description of MDGRAPE-2 Chip

5.1 General Specification

An MDGRAPE-2 chip is a data-flow-type numerical-processor LSI that is specially dedicated for molecular dynamics simulations. It calculates the forces exerting on 24 different particles from other particles in every 6 clock cycles with four pipelines in force mode. The central forces with arbitrary smooth functions are calculated by the equation,

$$F_i = \sum_j b_j G(a_j r^2)(\vec{r}_j - \vec{r}_i), \tag{5.1}$$

where, $G(x)$ is an arbitrary smooth function, $\vec{r}_i$ is the position vector of the particle (i-particle) on which the force exerts, $\vec{r}_j$ is the position vector of the particle (j-particle) which causes the force, r^2 is the squared distance between i- and j-particles, *i.e.*, $|\vec{r}_j - \vec{r}_i|^2$, and a_j and b_j are the scale factors. An MDGRAPE-2 chip calculates arbitrary central forces which are necessary in molecular dynamics simulations: such as Coulomb force, Van der Waals force and the real space force in Ewald method. An MDGRAPE-2 chip evaluates $G(x)$ by 4-th polynomial interpolation. Coefficients of the polynomials are stored in RAMs in an MDGRAPE-2 chip. MDGRAPE-2 chip also calculates the potential contributions at 24 different particles from one particle in every 6 clock cycles in potential mode:

$$\Phi_i = \sum_j b_j G(a_j r^2). \tag{5.2}$$

5.1.1 Clock cycle

MDGRAPE-2 chip can operate a system clock higher than 100 MHz.

5.1.2 Numerical expression

1. Integer operation
   Use two's complement format in integer operations.

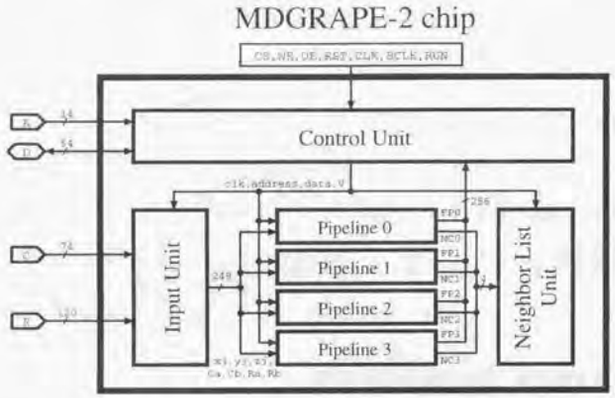


Figure 5.1: Block diagram of MDGRAPE-2 chip

2. Floating-point operation
   Use IEEE754 standard in floating-point operations.

3. Conversion from Integer to Floating point format
   Round to zero, in conversion from integer to floating-point format.

5.1.3 Pipeline

An MDGRAPE-2 chip has four pipelines. Each pipeline keeps data (positional coordinates and scale factors) of six i-particles. The data of one j-particle are supplied in every six clock cycles. A pipeline calculates six forces between these six i-particles and one j-particle in every six clock cycles. These forces are individually accumulated.

5.2 Structure

An MDGRAPE-2 chip is composed of four Pipelines, one Input Unit, one Neighbor List Unit and one Control Unit as shown in figure 5.1. Four Pipelines calculate the forces on 24 different i-particles from one j-particle and accumulate them in every six clock cycles. Using Neighbor flags, supplied by Pipelines, Neighbor List Unit makes lists of neighbor particles of 24 i-particles. Control Unit controls the multiplexers and registers in the chip. Input Unit receives coordinates of j-particles (x_j, y_j, z_j), and scale factors (a_j, b_j) and atom types in every six clock cycles from R-bus and C-bus, respectively. These data are stored in registers and kept in the next six clock cycles.

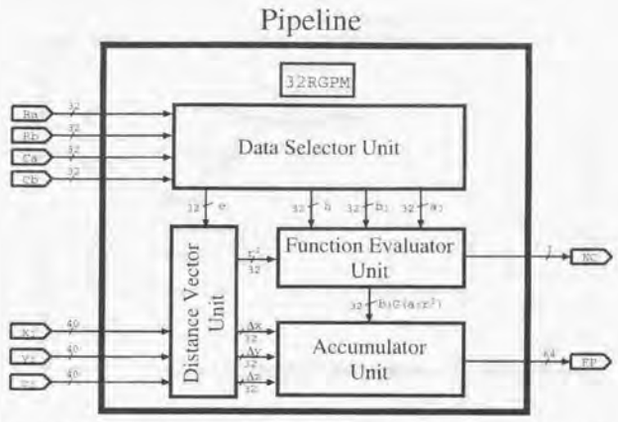


Figure 5.2: Block diagram of Pipeline. 32RGPM is a control register.

5.2.1 Pipeline

A pipeline is composed of a Distance Vector Unit, a Function Evaluator Unit, an Accumulator Unit, a Data Selector Unit, and a Control Register (12RGPM) as shown in figure 5.2.

Distance Vector Unit

A Distance Vector Unit outputs the distance vector $\Delta\vec{r} = \vec{r}_j - \vec{r}_i$ and the squared distance r^2. A Distance Vector Unit is composed of three 40-bit integer subtracters (40ISBX, 40ISBY, 40ISBZ), three 32-bit floating-point multipliers (32FMXS, 32FMYS, 32FMZS), three 32-bit floating-point adders (32FAXE, 32FAYZ, 32FARS), three 6-word 40-bit RAMs (6W40RAXI, 6W40RAYI, 6W40RAZI), three 40-bit integer to 32-bit floating-point converters (I2FDX, I2FDY, I2FDZ), and one 32-bit multiplexer (32MXR2) as shown in figure 5.3. The coordinates of six different i-particles are stored in three 6-word 40-bit RAMs (6W40RAXI, 6W40RAYI, 6W40RAZI). Three subtracters (40ISBX, 40ISBY, 40ISBZ) subtract the coordinates of i-particles from the coordinates of j-particles. In these subtracters, overflow must be neglected in order to represent the periodic boundary conditions. The differences are converted to 32-bit floating-point format by three format converters (I2FDX, I2FDY, I2FDZ). The results are squared by three multipliers (32FMXS, 32FMYS, 32FMZS). The squared value and softening parameter, ϵ, are added by three 32-bit floating-point adders (32FAXE, 32FAYZ, 32FARS) to get $r^2 = \Delta x^2 + \epsilon + \Delta y^2 + \Delta z^2$. The 32-bit multiplexer (32MXR2) is output selector of this unit. The output of 32FARS, the value of r^2, is always selected, when the control bit of 32MXR2 is 1. When the control bit is 0 and

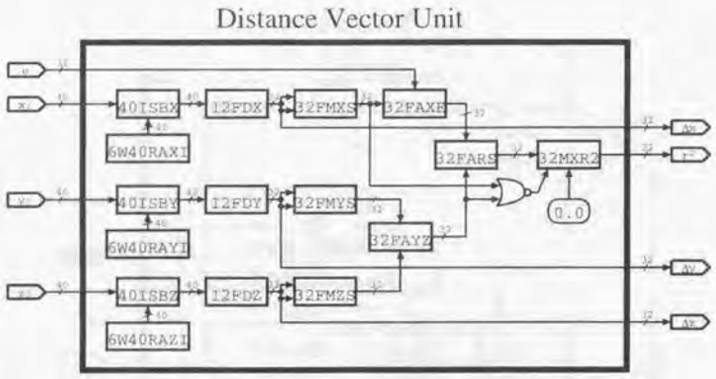


Figure 5.3: Block diagram of Distance Vector Unit

all three components of the distance vector ($\Delta x = x_j - x_i, \Delta y = y_j - y_i, \Delta z = z_j - z_i$) are 0, the fixed value 0.0 is selected. In this case ($i = j$), the output can be set to 0 even when ϵ is not zero. The values of r^2 are sent to Function Evaluator Unit and three components of distance vector ($\Delta x, \Delta y, \Delta z$) are sent to Accumulator Unit.

Function Evaluator Unit

A Function Evaluator Unit evaluates $b_j G(a_j r^2)$ and Neighbor Flag; NC. A Function Evaluator Unit is composed of two 32-bit floating-point multipliers (32FMAJ, 32FMBJ), one 32-bit function evaluator (32FE), and one 32-bit floating-point comparator (32FCNC) as shown in figure 5.4. The squared distance, r^2 from Distance Vector Unit, is multiplied with the scale factor a_j by a 32-bit floating-point multiplier (32FMAJ). A Function Evaluator (32FE) evaluates $G(a_j r^2)$ by 4-th order polynomial interpolation as,

$$G(w) = c_0 + w\,(c_1 + w\,(c_2 + w\,(c_3 + w\,c_4))). \tag{5.3}$$

In 32FE, $c_0, c_1, c_2, c_3, c_4, \Delta x$ and m are represented by 2's complement integer format. The others are represented by unsigned integer format. The four coefficients are stored in a 1024-word 104-bit RAM. Detailed description of 32FE will be given in section 5.3. The result of 32FE is multiplied with another scale factor b_j by a 32-bit floating-point multiplier (32FMBJ). The product $b_j G(a_j r^2)$ is sent to an Accumulator Unit. When r^2 is smaller than h, a comparator 32FCNC assert the Neighbor Flag (NC).

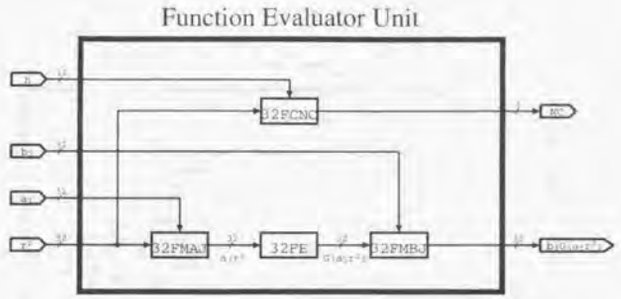


Figure 5.4: Block diagram of Function Evaluator Unit

Accumulator Unit

An Accumulator Unit accumulates forces in the force mode and potentials in the potential mode. An Accumulator Unit is composed of three 32-bit input 64-bit output floating-point multipliers (32FMFX64, 32FMFY64, 32FMFZ64), three 64-bit floating-point adders (64FAFX, 64FAFY, 64FAFZ), three 6-word 64-bit RAMs (6W64RAFX, 6W64RAFY, 6W64RAFZ), one 32-bit multiplexer (32MXFX), one 64-bit multiplexer (64MXFP), and three 32-bit delay lines (32DLDX, 32DLDY, 32DLDZ) as shown in figure 5.5. Three 32-bit floating-point multipliers (32FMFX64, 32FMFY64, 32FMFZ64), multiply $b_j G(a_j r^2)$ with the three components of distance vector, ($\Delta x, \Delta y, \Delta z$), which are delayed to match the corresponding $b_j G(a_j r^2)$ by three delay lines (32DLDX, 32DLDY, 32DLDZ). The x component of distance vector (Δx) can be replaced to fixed value (1.0) at the input of multipliers by the 32-bit multiplexers (32MXFX) inserted between 32DLDX and 32MXFX. Three components of pair-wise forces are accumulated in three 6-word 64-bit RAMs (6W64RAFX, 6W64RAFY, 6W64RAFZ), by three 64-bit floating-point adders (64FAFX, 64FAFY, 64FAFZ). A 64-bit multiplexer (64MXFP) is used for reading the forces during calculation (see section 5.4.1 in detail).

Data Selector Unit

A Data Selector Unit switches the sources of a_j, b_j, ϵ, and h. This unit is composed of two 6-word 32-bit RAMs (6W32RAH, 6W32RAE), two 32-bit registers (32RGA, 32RGB), and four multiplexers (32MXA, 32MXB, 32MXH, 32MXE) as shown in figure 5.6. Sources of a_j and b_j can be selected from preset value (32RGA, 32RGB), R-bus (R_a, R_b), or C-bus (C_a, C_b). For h and ϵ, preset value for each i-particles (6W32RAH, 6W32RAE), R-bus (R_a, R_b), or C-bus (C_a, C_b) can be connected. Two 6-word 32-bit RAMs are read corresponding to i-number count in Virtual Multiple Pipeline.

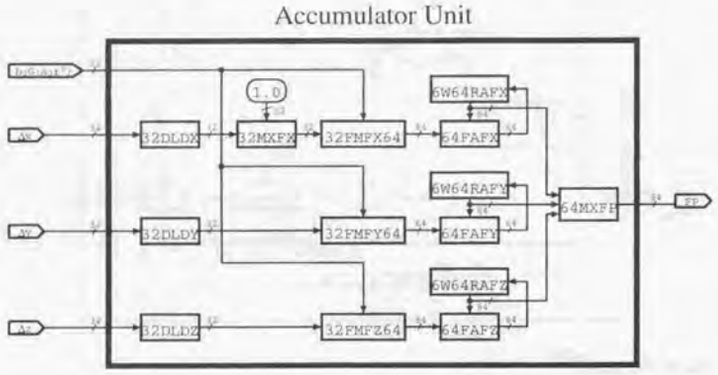


Figure 5.5: Block diagram of Accumulator Unit

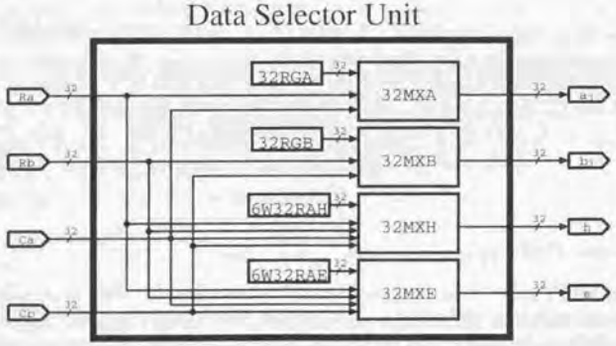


Figure 5.6: Block diagram of Data Selector Unit

Table 5.1: Bit assignments of Control Register (32RGPM)

bit	function
0–7	image of 8RGES
8–11	image of 4RGMS
12–15	*reserved*
16–17	control of 32MXA (00:R_a, 01:C_a, 1x:32RGA)
18–19	control of 32MXB (00:R_b, 01:C_b, 1x:32RGB)
20–22	control of 32MXH (000:R_a, 001:R_b, 010:C_a, 011:C_b, 1xx:6W32RAH)
23–25	control of 32MXE (000:R_a, 001:R_b, 010:C_a, 011:C_b, 1xx:6W32RAE)
26	control of 32MXFX (0:32DLDX, 1:fixed value 1.0)
27–28	*reserved*
29	control of 32MXR2 (0:fixed value 0.0 when $i = j$, 32FARS when $i \neq j$, 1:32FARS)
30–31	*reserved*

Control register

Each Pipeline has a 32-bit Control Register (32RGPM). The bit-images of two registers (8RGES, 4RGMS) in Function Evaluator Unit appear on bit 0–7 and bit 8–11 of Control Register, respectively. Bit 16–25 are assigned to control signals of multiplexers (32MXA, 32MXB, 32MXH, 32MXE) in Data Selector Unit. Bit 26–29 are assigned to control signals of multiplexers (32MXFX, 32MXFY, 32MXFZ, 32MXR2) in Accumulator Unit. Bit assignments of Control Register (32RGPM) is shown in table 5.1.

5.2.2 The other units

Input Unit

An Input Unit is composed of three 40-bit registers (40RGXJ, 40RGYJ, 40RGZJ), two 32-bit registers (32RGAJ, 32RGBJ), one 10-bit register (10RGV), one 1024-word 64-bit RAM (1024W64RAAB), 6-word 10-bit RAM (6W10RAT), and ten 1-bit multiplexers (1MXV0, 1MXV1, 1MXV2, ... 1MXV9), as shown in figure 5.7. An Input Unit receives three 40-bit position coordinates and two 32-bit scale factors of a j-particle from R-bus and C-bus, respectively in every six clock cycles. These data are stored in three 40-bit registers (40RGXJ, 40RGYJ, 40RGZJ) and two 32-bit registers (32RGAJ, 32RGBJ) and kept in next six clock cycles. The address lines of a 1024-word 64-bit RAM (1024W64RAAB), which stores the scale factors, are connected to ten 1-bit multiplexers (1MXV0, 1MXV1, 1MXV2, ... 1MXV9). Each of the 1-bit multiplexers is connected to C-bus and 6-word 10-bit RAM (6W10RAT), which stores atom types of six i-particles. Address lines of 6W10RAT are connected to a loop counter (6LCV) in Control Unit, which counts from 0 to 5.

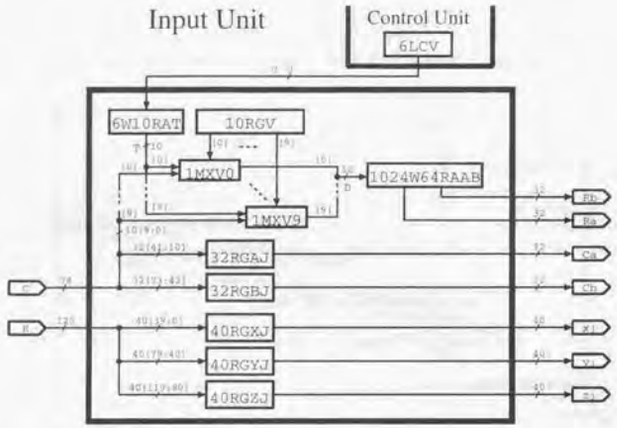


Figure 5.7: Block diagram of Input Unit

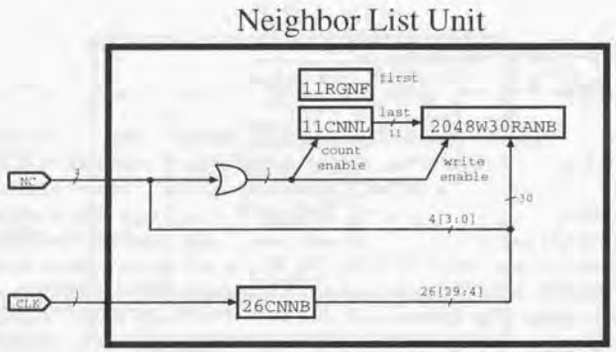
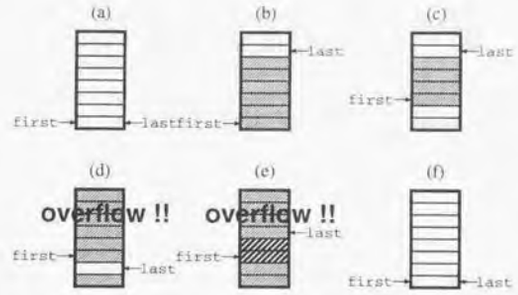


Figure 5.8: Block diagram of Neighbor List Unit



(a) First, neighbor RAM is empty. (b) Six words of neighbor flags are written. Thin hatched boxes show the valid neighbor flags stored in the RAM. (c) When host read two words, it increments *first* address by two. (d) An overflow flag is asserted when another three words are written (only one word left: $first - last = 1$). (e) The overflow flag keeps asserted, even after another three words are written. Thick hatched boxes denotes the lost neighbor flags. (f) When host set 11RGNF to zero with clear bit of 11CNNL and overflow flag of the FIFO.

Figure 5.9: Sample states of the neighbor RAM (2048W30RANB) as an FIFO

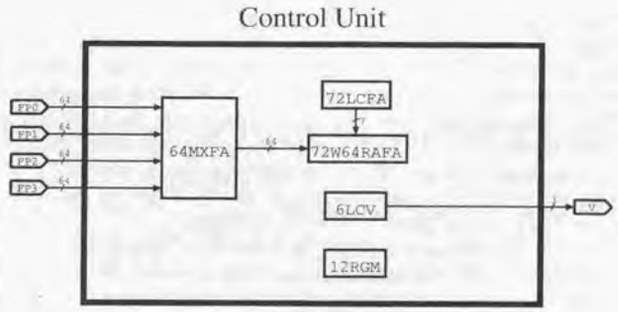


Figure 5.10: Block diagram of Control Unit

Neighbor List Unit

A Neighbor List Unit makes lists of neighbor particles of 24 i-particles using four neighbor flags from four pipelines. A Neighbor List Unit is composed of one 2048-word 30-bit RAM (2048W30RANB), one 11-bit counter (11CNNL), one 11-bit register (11RGNF), one 26-bit counter (26CNNB), and one 4 input OR gate, as shown in figure 5.8. The output of 11CNNL specifies the address of 2048W30RANB. A 26-bit counter (26CNNB) is incremented by a clock signal while the chip is in calculation. The outputs of 26CNNB specify the indices of neighbor particles. When one of four neighbor flags are asserted, 2048W30RANB stores the output of 26CNNB (26-bit) as well as four neighbor flags (NC0, NC1, NC2, NC3), and then 11CNNL is incremented.

2048W30RANB acts like an FIFO. A register 11RGNF holds the first address of the FIFO ($first$), which corresponds to the oldest word in the FIFO. The last address of the FIFO ($last$), which corresponds to the newest word in the FIFO, is specified by the output of 11CNNL. When no valid neighbor flags are stored in the FIFO, both addresses are the same, *i.e.*, $first = last$. The $last$ address is incremented when the neighbor flag is asserted. The overflow flag is asserted, when only one word remains vacant ($first - last = 1$). The overflow flag keeps asserted until it is cleared. When one reads a word on the address of 11CNNL, bit 31 of it indicates the overflow flag of the FIFO; it is 1 when overflow occurred at the FIFO.

After one reads some part of 2048W30RANB, one changes 11RGNF for decreasing the contents of the FIFO. When one writes 1 to bit 31 on the same address as 11RGNF, 11CNNL and the overflow flag of the FIFO are cleared. Figure 5.9 shows how the FIFO does act. Thin hatched boxes indicate the valid neighbor flags in the FIFO. Thick hatched boxes indicate the broken neighbor flags by the overflow.

Control Unit

Control Unit controls flow of input- and calculated- data, and behavior of Pipelines. Control Unit writes signals from D-bus onto RAMs and registers in Input Unit and Pipelines (RAMs and registers are specified by the address from A-bus). In addition, Control Unit has two loop counters (6LCV, 72LCFA), one 72-word 64-bit RAM (72W64RAFA), one 64-bit multiplexer (64MXFA), and one mode register (12RGM) as shown in figure 5.10. A counter, 6LCV, is incremented in every clock cycle from 0 to 5. And 72LCFA counts from 0 to 71. A 64-bit multiplexer (64MXFA) is used for reading the forces during calculation (see section 5.4.1 in detail). Bit assignments of 12RGM is shown in table 5.2. Bit image of 10RGV appears on bit 0–9. When one writes 1 to bit 10 of 12RGM, 26CNNB is cleared. When one writes 1 to bit 11 of 12RGM, 72LCFA is cleared.

Control Unit gives the addresses of RAMs in Input Unit (6W10RAT) and in Pipelines [(6W40RAXI, 6W40RAYI, 6W40RAZI) in Distance Vector Units, (6W64RAFX, 6W64RAFY, 6W64RAFZ) in Accumulator Units], and the addresses of multiplexers in Pipelines (64MXFP) and in Control Unit (64MXFA). Control Unit gives several signals to Input Unit and Pipelines, such as resetting, starting and ending of the calculation for all j-particles for six i-particles per pipeline. Control Unit also asserts write enables of three accumulation

Table 5.2: Bit assignments of Mode Register (12RGM)

bit	function
0–9	image of 10RGV
10	clear 26CNNB (0:do nothing, 1:clear)
11	clear 72LCFA (0:do nothing, 1:clear)

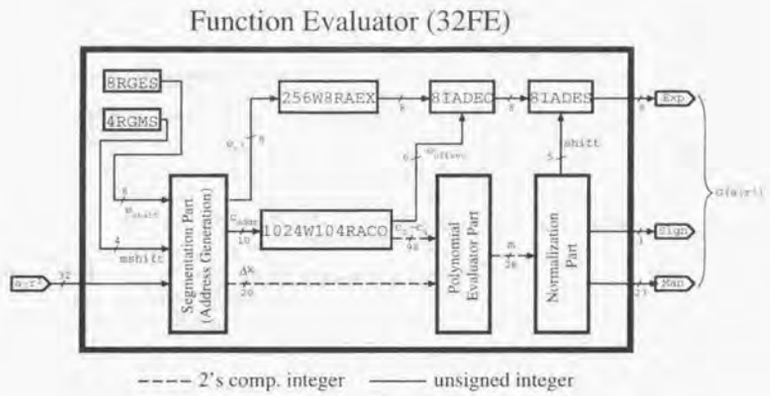


Figure 5.11: Block diagram of Function Evaluator (32FE)

RAMs (6W64RAFX, 6W64RAFY, 6W64RAFZ), Neighbor RAM (2048W30RANB) and an enable signal of 26CNNB taking into account of the state of RUN pin and pipeline delay.

5.3 Architecture of Function Evaluator Unit

The basic concept on calculation of an arbitrary function by the Function Evaluator Unit is as follows. A whole range $[x_{min}, x_{max})$ is divided into 2^{10} segments. Assuming that an input x belongs to the $(k+1)$-th segment, *i.e.*, $x \in [x_k, x_{k+1})$, the function $f(x)$ is approximated by a polynomial of degree 4;

$$f(x) \sim \sum_{i=0}^{4} c_i^{(k)} (\Delta x)^i, \tag{5.4}$$

where Δx is defined as the difference of x from the center of the segment x_c;

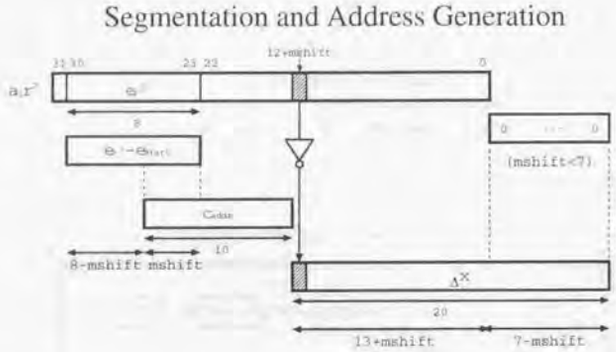


Figure 5.12: Algorithm of segmentation and address generation

$$\Delta x = x - x_c, \tag{5.5}$$

$$x_c = \frac{x_k + x_{k+1}}{2}. \tag{5.6}$$

The coefficients $c_i^{(k)}$ are constant in each segment.

5.3.1 General design

The general design of the Function Evaluator Unit is shown in figure 5.11. The input $a_j r^2$ and the output of the unit $G(a_j r^2)$ are 32-bit floating point numbers (Exp, Sign and Man in the figure are corresponding to the exponent, sign and mantissa part, respectively, of the output of the unit).

First, the Function Evaluator Unit calculates the Δx, the segment number k (c_{addr}), and the exponent part of $a_j r^2 (e_{r^2})$ by the Segmentation Part (see below in detail). The coefficient table (1024W104RACO) outputs the coefficients (c_0, c_1, c_2, c_3, c_4, and e_{offset}) of the polynomial according to the segment number k. Here, c_0, c_1, c_2, c_3, and c_4 are the mantissa parts of $c_i^{(k)}$, and e_{offset} is the common exponent part of $c_i^{(k)}$ $(i = 0, 1, \ldots, 4)$;

$$c_i^{(k)} = c_i \times 2^{e_{offset}}, \quad \text{for } i = 0, 1, \ldots, 4. \tag{5.7}$$

Second, the Function Evaluator Unit evaluates the polynomial by the Polynomial Evaluator Part. Last, the result (m) is normalized to a floating point number, which is composed of the three parts; Exp, Sign, and Man, as shown in figure 5.11.

5.3.2 Segmentation Part

As shown in figure 5.12, the Segmentation Part generates Δx (see the previous section) and the addresses of the coefficient and exponent tables (c_{addr} and e_{r^2}). The input value $a_j r^2$ is expressed in IEEE-754 32-bit floating-point format, which contains 23-bit mantissa, 8-bit exponent (e_{r^2}) and a sign.

As for generation of the addresses, first, the Segmentation Part calculates the difference between e_{r^2} and e_{start}, where e_{start} is the exponent part of x_{min}, and is stored in the register (8RGES). Second, the lower $mshift$ bits of the difference are used as the higher $mshift$ bits of c_{addr}, where $mshift$ is stored in the register (4RGMS), and is defined as the following equation;

$$x_{max} = 2^{e_{start} + 2^{mshift}}. \tag{5.8}$$

Consequently, the input $a_j r^2$ is valid from $2^{e_{start}}$ to $2^{e_{start} + 2^{mshift}}$. For the rest, *i.e.*, the lower $(10 - mshift)$ bits, of c_{addr}, the higher $(10 - mshift)$ bits of mantissa part of $a_j r^2$ are used. In this way, $mshift$ determines the dynamic range of input to the Function Evaluator Unit. For example, when $mshift$ is a small number, dynamic range is also small. Thereby, the width of a segment comes to be small, resulting in the high accuracy of interpolation.

As for generation of Δx, the bit $(12 + mshift)$ of $a_j r^2$ is inverted, and then, the lower $(13 + mshift)$ bits of $a_j r^2$ are used as the higher bits of Δx. By the inversion, Δx is converted to the deviation from the center of a segment in the fixed-point 2's complement expression. When $mshift$ is smaller than seven, the rest of bits of Δx, which is $(7 - mshift)$ bits, are all set to 0's.

5.3.3 Polynomial Evaluator Part

The Polynomial Evaluator Part is implemented as shown in figure 5.13. The unit is composed of four adders (ADC0, ADC1, ADC2, and ADC3) and four multipliers (MUC1, MUC2, MUC3, and MUC4). The coefficients c_0, c_1, c_2, c_3, and c_4, and the exponent part e_{offset} are defined as follows (see section 5.3.1 as well);

$$f(x) = 2^{e_{offset}} \times m, \tag{5.9}$$

where

$$m = c_0 + \Delta x(c_1 + \Delta x(c_2 + \Delta x(c_3 + c_4 \Delta x))). \tag{5.10}$$

The Polynomial Evaluator Part evaluates the value of m. Though the coefficients; c_0, c_1, c_2, c_3, and c_4 are expressed in the fixed-point 2's complement, they are considered as the 2's complement integers in the following description. In addition, all inputs and outputs of the adders and the multipliers in figure 5.13 are evaluated as 2's complement integers. For instance, when the output of the multiplier MUC1 is added to the coefficient c_0 by the adder ADC0, their sign bits should be expanded.

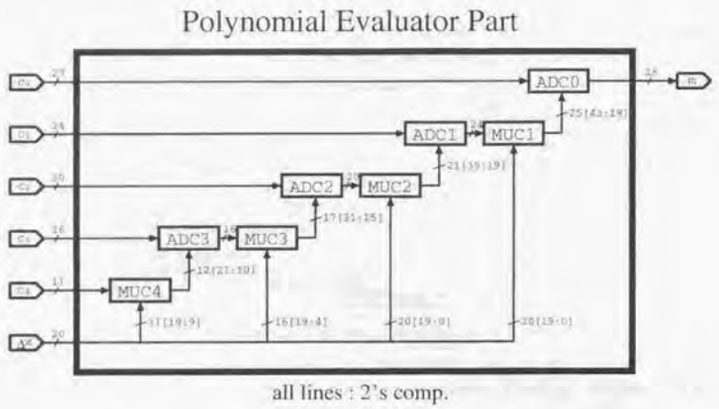


Figure 5.13: Block diagram of Polynomial Evaluator Part

5.3.4 Normalization Part

The Normalization Part converts the format of m, which is a 2's complement integer, into the floating point expression (Sign, Man and shift) as shown in figure 5.11. Exp in the figure is sum of $shift$, e_{offset} and the output of the exponent table (256W8RAEX). In the special cases, such as this summation overflows, m equals to zero, or overflows, the information is expressed through Exp, Sign and Man.

5.3.5 Remark

When e_{rel} is smaller than e_{start}, output of the Function Evaluator Unit must be set to zero, *i.e.*, Sign = Man = Exp = 0. This is carried out by the Segmentation Part and Normalization Part.

5.3.6 Error in Function Evaluator of MDGRAPE-2 chip

The author set a net relative accuracy of 10^{-7} in an MDGRAPE-2 chip, since this accuracy is usually satisfactory in molecular dynamics simulations. All the arithmetic units in an MDGRAPE-2 chip except Function Evaluator use 64-bit and 32-bit floating point format, which has an error smaller than 10^{-7}. Therefore, the author only evaluates the error generated in the Function Evaluator as follows.

In the calculation of the real-space part of Coulomb force, the Function Evaluator in a pipeline of an MDGRAPE-2 chip calculates equation (3.15). Figure 5.14 shows the relative

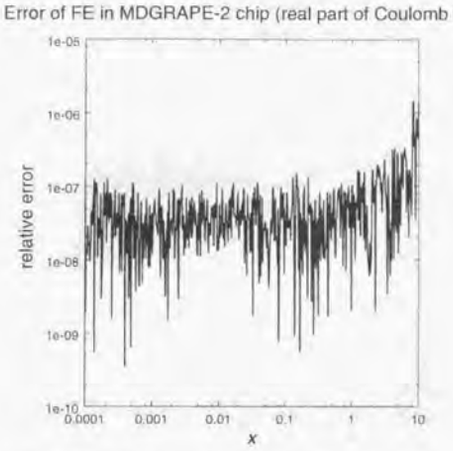


Figure 5.14: Relative error in $g(x)$ for real-space part of Coulomb force $[g(x) = \frac{2\exp(-x)}{\sqrt{\pi x}} + \frac{\mathrm{erfc}(\sqrt{x})}{x^{3/2}}]$ is plotted against x.

difference in $g(x) = \frac{2\exp(-x)}{\sqrt{\pi x}} + \frac{\mathrm{erfc}(\sqrt{x})}{x^{3/2}}$ between that calculated by double precision and that calculated by the software emulator of the function evaluator. Relative accuracy of 10^{-7} is almost achieved.

In the calculation of van der Waals force, the function evaluator calculates equation (3.19). Figure 5.15 shows the relative difference in $g(x) = 2x^{-7} - x^{-4}$ between that calculated by double precision and that calculated by the software emulator of the function evaluator. Relative accuracy of 10^{-7} is almost achieved, too.

5.4 External Interface

5.4.1 Chip registers and RAMs

A sample address map of the chip is shown in table 5.4. The lower 2-bit of the addresses of most of the registers and RAMs is assigned to pipeline indices, so that one may access to these registers and RAMs contiguously.

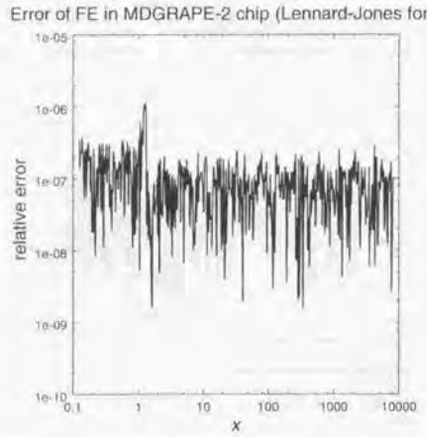


Figure 5.15: Relative error in $g(x)$ for Lennard-Jones force $[g(x) = 2x^{-7} - x^{-4}]$ is plotted against x.

Special meanings of writing certain addresses

There are special meanings in writing to 6W40RAXI, 6W40RAYI, 6W40RAZI, 12RGM, and 11RGNF.

- 12RGM
  see section 5.2.2.

- 6W40RAXI, 6W40RAYI and 6W40RAZI
  When one writes a word to 6W40RAXI, 6W40RAYI or 6W40RAZI, the corresponding word of 6W64RAFX, 6W64RAFY, or 6W64RAFZ is copied to 72W64RAFA in Control Unit through 64MXFP and 64MXFA, and then it is cleared. For example, when one writes a value to word 4 of 6W40RAXI in Pipeline 0, word 4 of 6W64RAFX in Pipeline 0 is copied to 72W64RAFA and then cleared to zero. Address of 72W64RAFA is specified by the loop counter 72LCFA. This counter is incremented by one, when one word is written.

- 11RGNF
  see section 5.2.2

Reading or writing during calculation

Most of registers and RAMs are not allowed to be read or written, while the chip calculates forces. However, 72W64RAFA, 2048W30RANB, and 11CNNL and overflow flag of the FIFO (2048W30RANB) are allowed to be read even when the chip is in calculation, and also writing a word to the address of 11RGNF is allowed during calculation.

When one reads the neighbor RAM (2048W30RANB) during calculation, one may get wrong value of it because of address collision. If one reads 2048W30RANB just when the chip wants to write the neighbor flags to it, writing to it is performed but reading from it is not performed. When such an address collision happens, bit 0–3 of the word are set to be all zero. One may have to read the same word several times until one of the bit 0–3 of it becomes 1.

5.4.2 External pins

C-bus has 74 pins (32 pins for a_j and b_j each and 10 pins for $atom\_type_j$) and R-bus has 120 pins (40 pins for each). a_j, b_j, x_j, y_j, and z_j are loaded in every six internal clock cycles through C-bus and R-bus. D-bus has 64 pins and A-bus has 14 pins. D-bus is used for accesses to RAMs and registers. A-bus is used to give internal address of accesses through D-bus. Pin assignments and address map of external interface are shown in table 5.3.

5.4.3 Timing chart of the chip

Write cycle

Timing chart of n words write is shown in figure 5.16. On every BCLK cycle, address and data are given to the chip. Note that the addresses, $A_0, A_1, \ldots, A_n$, are not necessary to be contiguous.

One word read cycle

Timing chart of one word read is shown in figure 5.17. The chip replies the input address after a certain delay.

n−word read cycle

Timing chart of n words read is shown in figure 5.18. The chip starts to reply the input address every BCLK cycle after a certain delay. One may call this cycle as *pipelined reading cycle*. This cycle ensures the bandwidth of data transfer between the board and the chip. Note that the addresses, $A_0, A_1, \ldots, A_n$, are not necessary to be contiguous.

Calculation cycle

Timing chart of calculation of the chip is shown in figure 5.19. One can freeze the calculation temporary, and restart the calculation. In figure 5.19, the chip freezes the calculation after

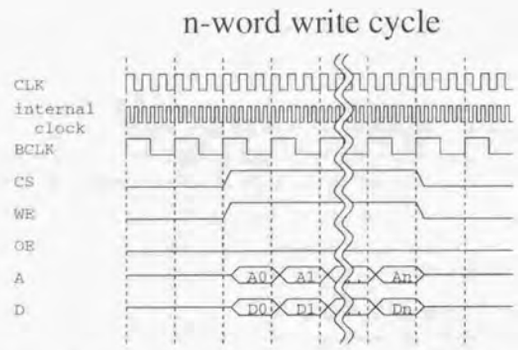


Figure 5.16: Timing chart of writing n words to the chip

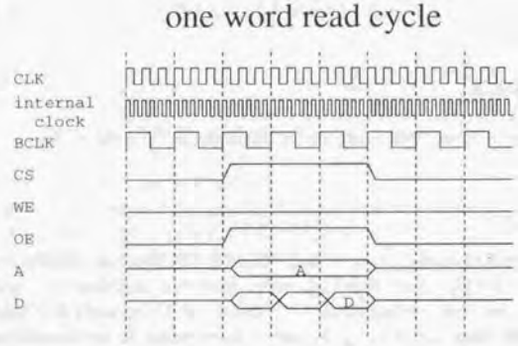


Figure 5.17: Timing chart of reading one word from the chip

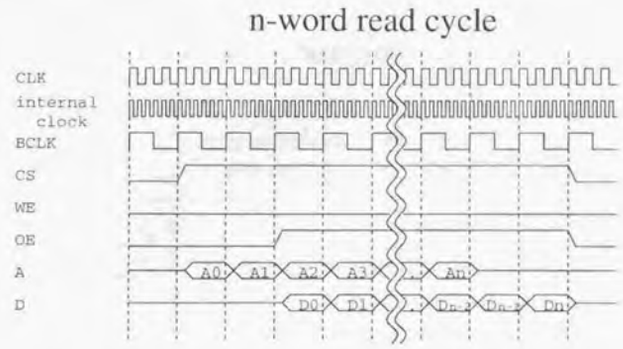


Figure 5.18: Timing chart of reading n words from the chip

loading 2 sets of data from C-bus and R-bus, and restarts the calculation after one cycle of BCLK. This control is done by RUN pin.

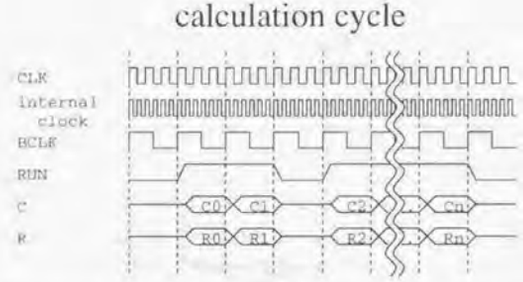


Figure 5.19: Timing chart while the chip is in calculation

Table 5.3: External Pins

name	I/O	description
A[13:0]	In	address bus to access to the RAMs and registers in the chip
D[63:0]	In/Out	data bus to access to the RAMs and registers in the chip
C[73:0]	In	C-bus to supply a_j, b_j and $atom\_type_j$.
R[119:0]	In	R-bus to supply r_x, r_y, r_z.
CS	In	Chip select signal. When it is high, read/write access to the chip is permitted.
WE	In	Write enable signal. High is write and low is read.
OE	In	Output enable signal. When it is low, D-bus turns to be high impedance.
RST	In	Reset signal.
CLK	In	Clock signal. Supply the clock with half of the internal clock cycle.
BCLK	In	Bus clock signal. Supply the clock signal for interfacing between the chip and other devices on the board.
RUN	In	Calculation signal. While it is high, chip loads data from C-bus and R-bus, accumulates forces, makes a neighbor list, and counts up 26CNNB.

Table 5.4: A sample address map of the chip

address	read/write	registers, RAMs and control flags
0000–07ff	write	1024W104RACO in Pipeline 0
0800–0fff	write	1024W104RACO in Pipeline 1
1000–17ff	write	1024W104RACO in Pipeline 2
1800–1fff	write	1024W104RACO in Pipeline 3
2000–20ff	write	256W8RAEX in Pipeline 0
2100–21ff	write	256W8RAEX in Pipeline 1
2200–22ff	write	256W8RAEX in Pipeline 2
2300–23ff	write	256W8RAEX in Pipeline 3
2800	write	32RGPM in Pipeline 0
2801	write	32RGPM in Pipeline 1
2802	write	32RGPM in Pipeline 2
2803	write	32RGPM in Pipeline 3
2804	write	32RGA in Pipeline 0
2805	write	32RGA in Pipeline 1
2806	write	32RGA in Pipeline 2
2807	write	32RGA in Pipeline 3
2808	write	32RGB in Pipeline 0
2809	write	32RGB in Pipeline 1
280a	write	32RGB in Pipeline 2
280b	write	32RGB in Pipeline 3
2900	write	word 0 of 6W40RAXI in Pipeline 0
2901	write	word 0 of 6W40RAXI in Pipeline 1
2902	write	word 0 of 6W40RAXI in Pipeline 2
2903	write	word 0 of 6W40RAXI in Pipeline 3
2904	write	word 1 of 6W40RAXI in Pipeline 0
2905	write	word 1 of 6W40RAXI in Pipeline 1
2906	write	word 1 of 6W40RAXI in Pipeline 2
2907	write	word 1 of 6W40RAXI in Pipeline 3
...	...	...
2914	write	word 5 of 6W40RAXI in Pipeline 0
2915	write	word 5 of 6W40RAXI in Pipeline 1
2916	write	word 5 of 6W40RAXI in Pipeline 2
2917	write	word 5 of 6W40RAXI in Pipeline 3
2920–2937	write	6W40RAYI in Pipeline 0–3
2940–2957	write	6W40RAZI in Pipeline 0–3
2960–2977	write	6W32RAH in Pipeline 0–3
2980–2997	write	6W32RAE in Pipeline 0–3
2400–27ff	write	1024W64RAAB
2a00–2a05	write	6W10RAT
2b00	write	12RGM
bit 0–10		11RGNF
2b01 bit 31	write	clear 11CNNL and overflow flag of the FIFO (2048W30RANB) (0:do nothing, 1:clear)
2b02 bit 0–10	read	11CNNL
bit 31		overflow flag of the FIFO (2048W30RANB)
2c00–2c47	read	72W64RAFA
3000–37ff	read	2048W30RANB
3800–3fff		

Chapter 6

Performance of Molecular Dynamics Machine

In the present chapter, the author estimates the performance of MDM. The calculation flow is described in section 6.1. In section 6.2, the specification for hardware, which is derived from the result of the preceding section, is given. In section 6.3, author shows the predicted performance of MDM. In the last section of the present chapter, he summaries this chapter.

6.1 Calculation Flow

In Molecular Dynamics Machine, calculation of one time-step is divided into three parts, *i.e.*, GRAPE-part, WINE-part, and Host-part. In GRAPE-part, MDGRAPE-2 calculates $\vec{F}_i$(re) and $\vec{F}_i$(vdW), while in WINE-part, WINE-2 calculates $\vec{F}_i$(wn). In Host-part, node computers calculate bonding-force and perform time integration of particles using $\vec{F}_i$(vdW), $\vec{F}_i$(re) and $\vec{F}_i$(wn) as well as bonding-force.

Table 6.1: Hierarchy structure of simulation space

class	number of cells in one dimension
Simulation box	l_{box}
domain	l_{dm}
G-block	l_{gbl}
W-block*	l_{wbl} , l_{dm}
cell	1

*W-block is not a cube (see figure 6.1).

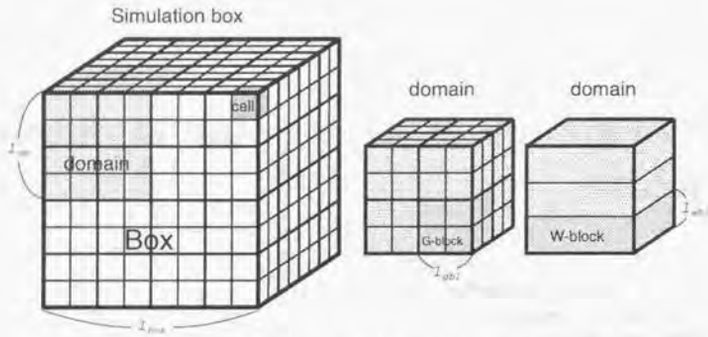


Figure 6.1: Structure of simulation space

Table 6.2: Hierarchy structure of hardware

class	number per upper level	total number	assigned space
MDM	1	1	Simulation box
node	$N_{nd} = (l_{box}/l_{dm})^3$	N_{nd}	domain
G-cluster	$N_{gcl} = (l_{dm}/l_{gbl})^3$	$N_{gcl}N_{nd}$	G-block
W-cluster	$N_{wcl} = l_{dm}/l_{wbl}$	$N_{wcl}N_{nd}$	W-block
G-board	N_{gbd}	$N_{gbd}N_{gcl}N_{nd}$	
W-board	N_{wbd}	$N_{wbd}N_{wcl}N_{nd}$	
G-chip	N_{gcp}	$N_{gcp}N_{gbd}N_{gcl}N_{nd}$	
W-chip	N_{wcp}	$N_{wcp}N_{wbd}N_{wcl}N_{nd}$	

6.1.1 Hierarchical structure in simulation space

In order to avoid unnecessary calculations and communications, the author organizes particles in a hierarchical structure with three-levels (table 6.1 and figure 6.1). The simulation box, the largest unit of the simulation, is composed of l_{box}^3 cells. The simulation box is divided into $(l_{box}/l_{dm})^3$ domains which is a cube with a side length of $l_{dm}L_{cell}$. Each domain, composed of l_{dm}^3 cells, is assigned to a node: $N_{nd} = (l_{box}/l_{dm})^3$. A node computer calculates bonding-force and performs the time integration of the particles in the corresponding domain.

When one calculates $\vec{F}_i$(re) and $\vec{F}_i$(vdW) with an MDGRAPE-2, one divides a domain into $(l_{dm}/l_{gbl})^3$ G-blocks, which is a cube with a side length of $l_{gbl}L_{cell}$. A G-block is assigned to a G-cluster: $N_{gcl} = (l_{dm}/l_{gbl})^3$. A G-cluster calculates $\vec{F}_i$(re) and $\vec{F}_i$(vdW) exerting on particles in the corresponding G-block.

When WINE-2 calculates $\vec{F}_i$(wn), a domain is divided into l_{dm}/l_{wbl} W-blocks. A W-block is a rectangular solid which has a side with a length of $l_{wbl}L_{cell}$ and two sides with a length of $l_{dm}L_{cell}$. A W-block is assigned to a W-cluster: $N_{wcl} = l_{dm}/l_{wbl}$. A W-cluster calculates $\vec{F}_i$(wn) exerting on particles in the corresponding W-block. Here, the author assumes in the present thesis that l_{dm} and l_{gbl} is a divisor of l_{box} and l_{dm}, respectively, and that l_{dm}/l_{wbl} is a natural number. Table 6.2 summarizes the hierarchy structure of the hardware and the simulation space. In the present thesis, the author assumes that simulation box, domains, and G-blocks are all cubes for simplicity.

6.1.2 GRAPE-Part

In GRAPE-part, a node computer and MDGRAPE-2 calculate $\vec{F}_i$(re) and $\vec{F}_i$(vdW). It is divided into G1- and G2-stages.

G1-Stage

A node computer sends positions ($\vec{r}_j$), charges (q_j) and atom types (at_j) of force-exerting particles of its domain to its N_{gcl} G-clusters through N_{gcl} links. Each G-cluster receives positions, charges and atom types of $N_{cell}(l_{gbl}+2)^3$ force-exerting particles. Here, $(l_{gbl}+2)^3$ is the number of cells which interact with particles in a G-block. The interface board in the G-cluster broadcasts these data to all the N_{gbd} G-boards, connected to it. The time, T_{G1}, for G1-stage can be evaluated as:

$$T_{G1} = 4wN_{cell}(l_{gbl}+2)^3 t_{bus}, \tag{6.1}$$

where t_{bus} is the cycle time of a link and w is the average number of clock cycles required to write one word through the link. Here, the data of one force-exerting particle are composed of 4 words, those are three positional coordinates (one word each), electrostatic charge, and atom type (packed into one word), where one word is 64-bit.

G2-Stage

A node computer sends the positions (r_i) of $N_{cell}l_{dm}^3$ force-receiving particles to its N_{gcl} G-clusters through N_{gcl} links in G2-stage. Each G-cluster calculates F_i(re) and F_i(vdW) exerting on $N_{cell}l_{dm}^3/N_{gcl}$ (=$N_{cell}l_{gbl}^3$) particles, and the calculated forces are sent back to their parent node computer. G2-stage is divided into $N_{cell}l_{dm}^3/(12N_{gcp}N_{gbd}N_{gcl})$ [=$N_{cell}l_{gbl}^3/(12N_{gcp}N_{gbd}$ steps. Here, $12N_{gcp}N_{gbd}N_{gcl}$ is the total number of virtual multiple pipelines in MDGRAPE-2. In each step, MDGRAPE-2 calculates the force on $12N_{gcp}N_{gbd}N_{gcl}$ particles as follows.

A node computer sends positions of $12N_{gcp}N_{gbd}$ force-receiving particles to each G-cluster. The interface board in a G-cluster sends positions of $12N_{gcp}$ particles to each G-board. A G-board calculates forces [$\vec{F}_i$(re) and $\vec{F}_i$(vdW)] exerted on $12N_{gcp}$ particles in $6 \cdot 27N_{cell}$ clock cycles of a G-chip. Each of G-boards calculates forces on particles in different cell in a G-block. The results, F_i(re) and F_i(vdW), are sent back to the parent node computer through the interface board of the G-cluster. This step is repeated $N_{cell}l_{gbl}^3/(12N_{gcp}N_{gbd})$ times to get the forces on all the particles in the simulation box. For efficient use of G-pipelines, N_{cell} is necessary to be larger than $12N_{gcp}$, *i.e.*,

$$N_{cell} > 12N_{gcp}, \tag{6.2}$$

and N must be larger than $12N_{gcp}N_{gbd}N_{gcl}N_{nd}$, *i.e.*,

$$N > 12N_{gcp}N_{gbd}N_{gcl}N_{nd}. \tag{6.3}$$

The time, T_{G2}, for G2-stage can be expressed as:

$$\begin{aligned} T_{G2} &= 2 \cdot 4w N_{cell}l_{gbl}^3 t_{bus}, \\ &+ \frac{N_{cell}l_{gbl}^3}{12N_{gcp}N_{gbd}} \cdot 6 \cdot 27N_{cell}t_{gpipe}, \\ &+ 2 \cdot 3r N_{cell}l_{gbl}^3 t_{bus}, \end{aligned} \tag{6.4}$$

where r is the average number of clock cycles for reading a word through a link, and t_{gpipe} is cycle time of the clock of a G-chip, *i.e.*, t_{gpipe}=10 nsec. The first term represents the time to send the data of $N_{cell}l_{gbl}^3$ force-receiving particles through a link [three components of positional coordinate and information of neighboring cells: 4 words per particle for each of $\vec{F}_i$(re) and $\vec{F}_i$(vdW)]. The second term represents the time to calculate the forces. The third term is the time of nodes to receive the forces (three components of each of $\vec{F}_i$(re) and $\vec{F}_i$(vdW): 6 words per particle). Here, the author assumes that N_{cell} and $N_{cell}l_{gbl}^3$ can be divided by $12N_{gcp}$ and $12N_{gcp}N_{gbd}$, respectively.

6.1.3 WINE-part

In WINE-part, a node computer and WINE-2 obtain $\vec{F}_i$(wn). It is divided into five stages, those are W1-, W2-, W3-, W4-, and W5-stages.

In W1-stage through W3-stage, S_n, C_n, a_n and θ_n are calculated. You obtain S_n and C_n by summing up $N_{wcl}N_{nd}$ sub-summations ($s_n^{(m)}$, and $c_n^{(m)}$), which are calculated by the m-th W-cluster:

$$S_n = \sum_{m=1}^{N_{wcl}N_{nd}} s_n^{(m)}, \tag{6.5}$$

$$C_n = \sum_{m=1}^{N_{wcl}N_{nd}} c_n^{(m)}. \tag{6.6}$$

In W4- and W5-stages, $\vec{F}_i$(wn) is calculated from a_n and θ_n by equation 3.10.

W1-Stage

In W1-stage, a node computer sends positions of different $l_{dm}^2 l_{wbl}N_{cell}$ particles in each W-block to each of N_{wcl} (= l_{dm}/l_{wbl}) W-clusters through N_{wcl} links. The interface board in a W-cluster broadcasts these data to all the boards which are connected to it. The time, T_{W1}, for this stage is expressed as

$$T_{W1} = 2wl_{dm}^2 l_{wbl}N_{cell}t_{bus}, \tag{6.7}$$

where the author assumes that positional coordinates and charges of one particle are packed into two words.

W2-Stage

A node computer sends N_{wv} wavenumber vectors to all of its N_{wcl} W-clusters through N_{wcl} links, each cluster calculates $s_n^{(m)}$ and $c_n^{(m)}$ for N_{wv} wavenumber vectors, and these results are sent back to the node computer. This stage is divided into $N_{wv}/(64N_{wcp}N_{wbd})$ steps. Here, $64N_{wcp}N_{wbd}$ is the total number of virtual pipelines per W-cluster, and the author assumes that N_{wv} can be divided by $64N_{wcp}N_{wbd}$. Each step calculates $s_n^{(m)}$ and $c_n^{(m)}$ for $64N_{wcp}N_{wbd}$ wavenumber vectors as follows.

A node computer sends $64N_{wcp}N_{wbd}$ wavenumber vectors to all of N_{wcl} W-clusters under it. The interface board in a W-cluster sends $64N_{wcp}$ wavenumber vectors to each W-board. A W-board calculates $s_n^{(m)}$ and $c_n^{(m)}$ of $64N_{wcp}$ wavenumber vectors in $8l_{dm}^2 l_{wbl}N_{cell}$ clock cycles of a W-chip. The results are sent back to the parent node computer through the interface board of the W-cluster. For efficient use of W-pipelines, the number, N_{wv}, of wavenumber vectors is necessary to be larger than $64N_{wcp}N_{wbd}$, *i.e.*,

$$N_{wv} > 64N_{wcp}N_{wbd}. \tag{6.8}$$

This procedure is repeated $N_{wv}/64N_{wcp}N_{wbd}$ times.

The time, T_{W2}, for this stage is expressed as:

$$\begin{aligned} T_{W2} &= \frac{w}{2}N_{wv}t_{bus} \\ &+ \frac{N_{wv}}{64N_{wcp}N_{wbd}}8l_{dm}^2 l_{wbl}N_{cell}t_{wpipe} \\ &+ 2rN_{wv}t_{bus}, \end{aligned} \tag{6.9}$$

where t_{wpipe} is the cycle time of internal clock of a W-chip, *i.e.*, t_{wpipe}=12.5 nsec. The first term represents the time to send the data of the three components of N_{wv} wavenumber vectors, $\vec{k}_n$ (two wavenumber vectors packed into one word). The second term is the time to calculate $c_n^{(m)}$, $s_n^{(m)}$. The third term is the time to send $c_n^{(m)}$ and $s_n^{(m)}$ back to the parent node computer (two words per wavenumber vector).

W3-stage

In W3-stage, node computers calculate S_n and C_n [equations (6.5) and (6.6)], and a_n and θ_n [equations (2.13) and (2.14)] in the following five steps. First, each node computer sums up $s_n^{(m)}$ and $c_n^{(m)}$ from N_{wcl} W-clusters under it for all the wavenumber vectors. Second, each node computer broadcasts these values to other node computers. Third, each node computer sums up these values received from other nodes computers, *i.e.*, calculates S_n and C_n for N_{wv}/N_{nd} wavenumber vectors [equations (6.5) and (6.6)]. Forth, each node computer calculates a_n and θ_n for N_{wv}/N_{nd} wavenumber vectors from equations 2.13 and 2.14. Finally, each node computer broadcasts these values to other node computers.

The time for W3-stage, T_{W3}, which is dominated by transferring $s_n^{(m)}$, $c_n^{(m)}$, a_n and θ_n (four words per wavenumber vectors):

$$T_{W3} = \frac{4N_{wv}(N_{nd}-1)t_{cn}}{N_{nd}}, \tag{6.10}$$

where, t_{cn} is the time for transferring a word through a channel between nodes. Here, calculation time in the node computer is neglected.

W4-stage

A node computer sends a_n and θ_n for N_{wv} wavenumber vectors to its N_{wcl} W-clusters through N_{wcl} links. The interface board in a W-cluster broadcasts these a_n and θ_n (packed into one word per wavenumber vector) to all the N_{wbd} W-boards under it. The time, T_{W4}, for W4-stage is expressed as

$$T_{W4} = wN_{wv}t_{bus}. \tag{6.11}$$

Here, the author assumes that all the wavenumber vectors, $\vec{k}_n$, are sent to all the W-boards before starting the simulation.

W5-stage

A node computer sends the positions of $N_{cell}l_{dm}^3$ force-receiving particles to its N_{wcl} W-clusters through N_{wcl} links. A W-cluster calculates $\vec{F}_i$(wn) exerting on $N_{cell}l_{dm}^3/N_{wcl}$ [$=N_{cell}l_{dm}^2 l_{wbl}$] particles, and these forces are sent back to the node computer. This stage is divided into $N_{cell}l_{dm}^3/(64N_{wcp}N_{wbd}N_{wcl}) = N_{cell}l_{dm}^2 l_{wbl}/(64N_{wcp}N_{wbd})$ steps. Here, the author assumes that $N_{cell}l_{dm}^2 l_{wbl}$ can be divided by $64N_{wcp}N_{wbd}$. Each step calculates $\vec{F}_i$(wn) on $64N_{wcp}N_{wbd}N_{wcl}$ particles, where $64N_{wcp}N_{wbd}N_{wcl}$ is the total number of virtual pipelines in WINE-2. The calculations of this one step are done as follows.

First, a node computer sends positions of $64N_{wcp}N_{wbd}$ force-receiving particles to each W-cluster. The interface board in a W-cluster sends positions of $64N_{wcp}$ particles to each W-board. A W-board calculates forces exerted on $64N_{wcp}$ particles in $4N_{wv}$ clock cycles of a W-chip. The results, $\vec{F}_i$(wn), are sent back to its parent node computer through the interface board. This procedure is repeated $N_{cell}l_{dm}^2 l_{wbl}/64N_{wcp}N_{wbd}$ times to get the forces on all the force-receiving particles. For efficient use of W-pipelines, $N_{cell}l_{dm}^3$ must be larger than $64N_{wcp}N_{wbd}N_{wcl}$ or, equivalently,

$$N > 64N_{wcp}N_{wbd}N_{wcl}N_{nd}. \tag{6.12}$$

The time, T_{W5}, for W5-stage is expressed as:

$$\begin{aligned} T_{W5} &= 2wN_{cell}l_{dm}^2 l_{wbl}t_{bus} \\ &+ \frac{N_{cell}l_{dm}^2 l_{wbl}}{64N_{wcp}N_{wbd}} \cdot 4N_{wv}t_{wpipe} \\ &+ 3rN_{cell}l_{dm}^2 l_{wbl}t_{bus}. \end{aligned} \tag{6.13}$$

The first term is the time for sending three components of positional coordinates of force-receiving particles (packed into two words per wavenumber). The second term is the time for calculating $\vec{F}_i$(wn). The third term represents the time for sending back the result, $\vec{F}_i$(wn), (three words per particle) to the parent node computer.

6.1.4 Host-part

Each node computer calculates bonding forces, $\vec{F}_i$(bd) exerted on the particles in its domain, then it performs the time integration of the particles in its domain. They are sent to other node computers. The time, T_H, for the Host-part is expressed as

$$T_H = N_{cell}l_{dm}^3 N_{fp}t_{fn} + 3N_{cell}\left\{(l_{dm}+2)^3 - (l_{dm}+2\delta_{l_{box},l_{dm}})^3\right\}t_{cn}, \tag{6.14}$$

where N_{fp} is the number of floating-point operations required to the calculation of bonding-force, time integration of a particle, and coordinate transformation, t_{fn} is the time required to perform one floating-point operation in a node computer, and $\delta_{i,j}$ is the Kronecker's delta. Second term is the time for transferring new positions of particles, and it becomes zero when $l_{dm} = l_{box}$.

6.1.5 Total time for one time-step

The total time, T_{total}, for one time-step is not a simple sum of the times for GRAPE-, WINE- and Host-parts, since some parts can be overlapped. For example, you can overlap most of the GRAPE-part and WINE-part. Taking into account of such overlapping operation, the author evaluates T_{total} as:

$$T_{total} = T_{host} + T_{pipe} + T_{com}, \tag{6.15}$$

where T_{host} is the calculation time in the host computer, T_{pipe} is the calculation time for $\vec{F}_i$(re), $\vec{F}_i$(vdW), and $\vec{F}_i$(wn), and T_{com} is the time for communications. They are expressed as:

$$T_{host} = N_{cell} l_{dm}^3 N_{fp} t_{fp}, \tag{6.16}$$

$$T_{pipe} = \max(T_{gpipe}, T_{wpipe}), \tag{6.17}$$

$$\begin{aligned} T_{com} = \max \Big(& \left[4(l_{gbl}+2)^3 N_{cell} + 8 N_{cell} l_{gbl}^3 \right] w t_{bus} \\ & + 6r N_{cell} l_{gbl}^3 t_{bus}, \\ & \left(4 l_{dm}^2 l_{wbl} N_{cell} + 1.5 N_{wv} \right) w t_{bus} \\ & + \left(2 N_{wv} + 3 l_{dm}^2 l_{wbl} N_{cell} \right) r t_{bus}, \\ & \{ 4 N_{wv} (N_{nd}-1)/N_{nd} \\ & + 3 \left[(l_{dm}+2)^3 - (l_{dm} + 2\delta_{l_{box}, l_{dm}})^3 \right] N_{cell} \} t_{cn} \Big), \end{aligned} \tag{6.18}$$

where T_{gpipe} and T_{wpipe} are expressed as:

$$T_{gpipe} = \frac{27 N_{cell}^2 l_{gbl}^3}{2 N_{gcp} N_{gbd}} t_{gpipe}, \tag{6.19}$$

$$T_{wpipe} = \frac{3 N_{cell} l_{dm}^2 l_{wbl} N_{wv}}{16 N_{wcp} N_{wbd}} t_{wpipe}. \tag{6.20}$$

6.2 Required Specification

In this section, the author determines the required specification for the MDM. This includes the calculation speed of the specialized hardware, MDGRAPE-2 and WINE-2, the communication speed between node computers and MDGRAPE-2 or WINE-2, communication speed between nodes, and calculation speed of node computers.

The target performance is to realize T_{total} less than 0.1 sec for the case that $N = 10^6$ and the relative accuracy of Ewald sum, ξ, is 10^{-3}. The author tentatively sets the upper limits of T_{host}, T_{pipe} and T_{com} as

$$T_{host} < T_{host}^* \tag{6.21}$$

$$T_{pipe} < T_{pipe}^* \tag{6.22}$$

$$T_{com} < T_{com}^*. \tag{6.23}$$

Note that those upper limits are expected to be close to 3.3×10^{-2} sec, one third of the 0.1 sec in an initial guess.

The author obtains the number of chips for MDGRAPE-2 and WINE-2 (subsection 6.2.1), number of channels between nodes (subsection 6.2.2), number of links between node computers and MDGRAPE-2 (subsection 6.2.3), number of links between node computers and WINE-2 (subsection 6.2.4), and required calculation speed for the node computers (subsection 6.2.5).

6.2.1 Number of chips

The number of chips for MDGRAPE-2 and WINE-2 are determined by two constraints: Those are minimizing the cost for chips and inequality equation (6.22).

The author defines a cost function e as:

$$e = \left(\frac{p}{p+1} N_g + \frac{1}{p+1} N_w \right) T_{pipe}^*, \tag{6.24}$$

where N_g and N_w are the total numbers of G-chips and W-chips, respectively, and p is the ratio of the cost of a G-chip to that of a W-chip, *i.e.*, a G-chip is p times expensive than a W-chip. Here, N_g and N_w are given by:

$$N_g = N_{gcp} N_{gbd} N_{gcl} N_{nd}, \tag{6.25}$$

$$N_w = N_{wcp} N_{wbd} N_{wcl} N_{nd}. \tag{6.26}$$

If the author assumes the ideal case those two terms of equation (6.17) are the same, equations (6.25) and (6.26) are reduced to:

$$N_g = \frac{27 N N_{cell} t_{gpipe}}{2 T_{pipe}^*}, \tag{6.27}$$

$$N_w = \frac{3 N N_{wv} t_{wpipe}}{16 T_{pipe}^*}, \tag{6.28}$$

where $l_{gbl}^3 N_{cell} = N/(N_{gbl} N_{nd})$ and $l_{dm}^2 l_{wbl} N_{cell} = N/(N_{wcl} N_{nd})$ are used.

According to Fincham et al. (1993), r_{cut} and k_{cut} are determined for a given relative accuracy, ξ:

$$r_{cut} = \frac{L}{\alpha} \sqrt{-\ln \xi}, \tag{6.29}$$

$$k_{cut} = \frac{\alpha}{\pi L} \sqrt{-\ln \xi}. \tag{6.30}$$

Using equations (2.17), (2.11), (6.24), (6.27), (6.28), (6.29), and (6.30), the author derives N_{cell}, N_{wv}, N_g, N_w and e using l_{box} as follows.

$$N_{cell} = N l_{box}^{-3}, \tag{6.31}$$

$$N_{wv} = \frac{2(-\ln \xi)^3}{3\pi^2} l_{box}^3, \tag{6.32}$$

$$N_g = \frac{27 N^2 t_{gpipe}}{2 T_{pipe}^*} l_{box}^{-3}, \tag{6.33}$$

$$N_w = \frac{(-\ln \xi)^3 N t_{wpipe}}{8\pi^2 T_{pipe}^*} l_{box}^3, \tag{6.34}$$

$$\begin{aligned} e = & \left(\frac{27 p N t_{gpipe}}{2(p+1)} l_{box}^{-3} \right. \\ & \left. + \frac{(-ln\xi)^3 t_{wpipe}}{8\pi^2 (p+1)} l_{box}^3 \right) N. \end{aligned} \tag{6.35}$$

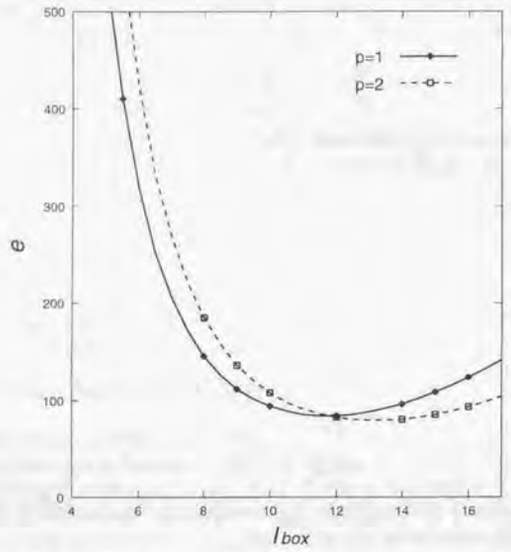


Figure 6.2: Cost function e is plotted against l_{box} for the case that $N = 10^6$ and $\xi = 10^{-3}$. In both cases that cost of a G-chip and W-chip is same (solid curve; $p = 1$) and that cost of a G-chip is twice expensive than W-chip (dashed curve; $p = 2$), e becomes almost minimum.

Table 6.3: Optimal sets of parameters

	$N = 10^6$, $\xi = 10^{-3}$	$N = 10^5$, $\xi = 10^{-3}$
l_{box}	12	8
r_{cut}/L	8.3×10^{-2}	0.13
Lk_{cut}	26	18
N_{cell}	580	200
N_{wv}	3.8×10^4	1.1×10^4
$N_g T^*_{pipe}$	78	2.6
$N_w T^*_{pipe}$	90	2.7

In figure 6.2, e is plotted against l_{box} based on equation (6.35) in the case that $N = 10^6$ and $\xi = 10^{-3}$. Solid curve ($p = 1$) shows that e takes the minimum value at l_{box}=12. Dashed curve ($p = 2$) shows that e takes the minimum value at $l_{box} = 14$. However, figure 6.2 suggests that the minimum value of e is not sensitive to p. The author adopts $l_{box} = 12$. Note that $l_{box} = 12$ with more number of divisor is more preferable to 14, since the former leads the more flexibility to construct the system in hierarchical structure. In the case of $l_{box} = 12$, he obtains the lower limit of N_g and N_w, as follows for $T^*_{pipe} = 3.3 \times 10^{-2}$ sec,

$$N_g \geq 2367, \tag{6.36}$$

$$N_w \geq 2733, \tag{6.37}$$

where the author uses equations (6.32) and (6.33), $N = 10^6$, and $\xi = 10^{-3}$.

6.2.2 Communication speed between nodes

Total number of channels required between nodes are obtained from the condition that the third part of right hand side of the equation 6.18 must be less than T^*_{com} (equation 6.23). To get the required communication speed between nodes, l_{dm} must be equal to or smaller than l^*_{dm}, where l^*_{dm} is the solution of the equation:

$$\frac{4N_{wv}(N - N_{cell}x^3)}{N} + 3[(x+2)^3 - (x + 2\delta_{l_{box},x})^3]N_{cell} = \frac{T^*_{com}}{t_{cn}}, \tag{6.38}$$

where $N_{nd} = \frac{N}{l^3_{dm}} N_{cell}$.

Thin solid curve of figure 6.3 shows the l^*_{dm} as a function of $\frac{T^*_{com}}{t_{cn}}$ when $l_{box} = 12$. Taking into account that l_{dm} must be a divisor of l_{box}, the author gets $l_{dm} = 6$, *i.e.*, $N_{nd} = 8$, in the case of $\frac{T^*_{com}}{t_{cn}} = 7.3 \times 10^5$ (= 33msec/45nsec). Here the effective transfer speed of a channel is assumed to be 178 Mbyte/sec, *i.e.*, $t_{cn} = 45$ nsec.

There is another trivial solution at $l_{dm} = 12$ ($N_{nd} = 1$) of equation (6.38) independent of T^*_{com}/t_{cn}, since no communication is necessary between nodes. Since no single CPU does not satisfy the calculation speed required for host computer (> 12 Gflops, see section

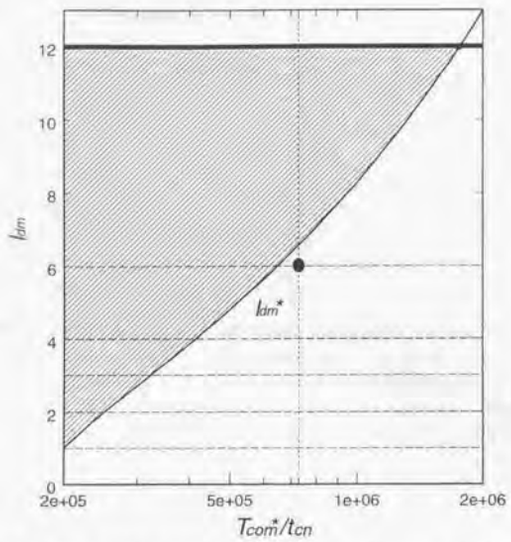


Figure 6.3: Thin solid curve shows the l_{dm}^* as a function of T_{com}^*/t_{cn}. The hatched region is prohibited for $l_{box} = 12$. The filled circle shows that l_{dm}=6 satisfies $l_{dm} < l_{dm}^*$ at $T_{com}^*/t_{cn} = 7.3 \times 10^5$ (vertical dotted line). Thick line shows that l_{dm}=12 satisfies the condition at any T_{com}^*/t_{cn} for the case that $l_{box} = 12$.

6.2.5 for details), a host computer for $N_{nd} = 1$ must be a parallel computer with a shared memory architecture. In that case, communication among nodes is done by system bus, which is much faster than usual communication speed among nodes of distributed memory system. Therefore, the author can safely consider a single node shared memory parallel machine as a solution of MDM.

6.2.3 Communication speed between node computers and MDGRAPE-2

Total number of links between node computers and MDGRAPE-2, N_{glnk} [$= N_{nd}N_{gcl} = (l_{box}/l_{gbl})^3$], is determined from the condition that first part of right hand side of equation 6.18 must be less than T_{com}^* (equation 6.23). To achieve required communication speed between them, l_{gbl} must be equal to or smaller than l_{gbl}^*, where l_{gbl}^* is the solution of the

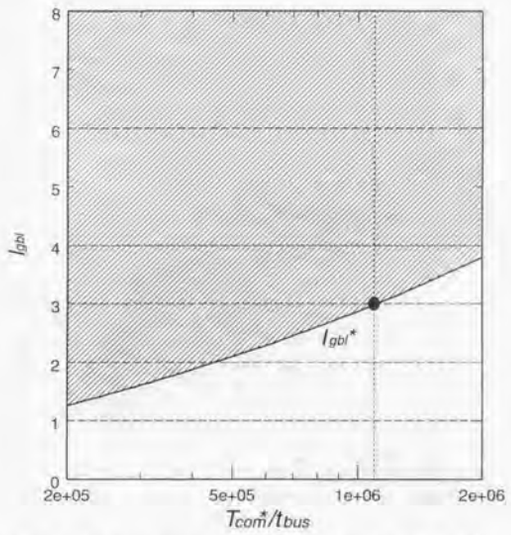


Figure 6.4: Solid curve shows the l_{gbl}^* as a function of T_{com}^*/t_{bus}. The hatched region is prohibited. The filled circle (l_{gbl}=3) marginally satisfies the constraints $l_{gbl} < l_{gbl}^*$ at $T_{com}^*/t_{bus} = 1.1 \times 10^6$ (vertical dotted line).

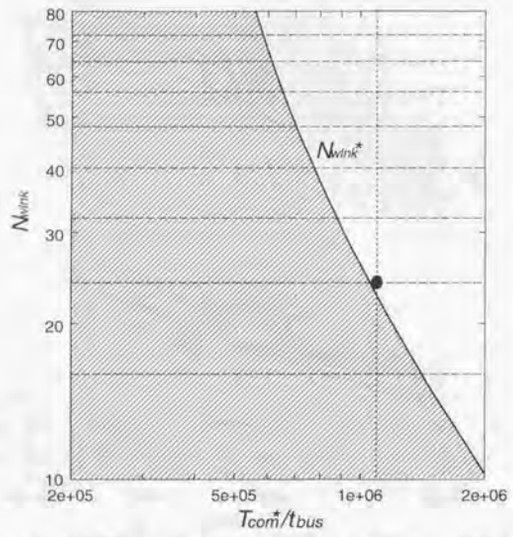


Figure 6.5: Solid curve shows the N^*_{wlnk} as a function of T^*_{com}/t_{bus}. The hatched region is prohibited. The filled circle shows that N_{wlnk}=24 satisfies $N_{wlnk} \geq N^*_{wlnk}$ at $T^*_{com}/t_{bus} = 1.1 \times 10^6$ (vertical dotted line).

Table 6.4: Two solution for the set up of MDM

N_{nd}	8	1
N_{wcl}	3	24
N_{gcl}	8	64
N_{gbd}	4	4
N_{wbd}	8	8
N_{gcp}	10	10
N_{wcp}	16	16
N_g	2560	2560
N_w	3072	3072

Table 6.5: Constants

N	10^6
ξ	10^{-3}
t_{gpipe}	10 nsec
t_{wpipe}	12.5 nsec
t_{bus}	30 nsec
t_{cn}	45 nsec
w	2
r	3
N_{fp}	400

equation:

$$(12w + 6r)x^3 + 24wx^2 + 48wx + 32w = \frac{T^*_{com}}{N_{cell}t_{bus}}. \tag{6.39}$$

Figure 6.4 shows the constraints on l_{ghl}, where $w = 2$, $r = 3$, and $l_{box} = 12$ are assumed. The hatched region is prohibited. The dotted vertical line indicates $T^*_{com}/t_{bus} = 1.1 \times 10^6$ (= 33 msec / 30 nsec), where the author adopts 30 nsec for t_{bus} which is the cycle time of system clock of PCI *i.e.*, N_{glnk}, total number of links between node computers and MDGRAPE-2, is 64 [$= (12/3)^3$].

6.2.4 Communication speed between node computers and WINE-2

Total number of links between node computers and WINE-2, N_{wlnk} ($= l^3_{box}/l^2_{wbl}$), is determined from the condition that the second part of right-hand side of equation 6.18 must be less than T^*_{com} (equation 6.23). To achieve required communication speed between them,

Table 6.6: Number of cells and calculation time

l_{box}	12
l_{dm}	6
l_{gbl}	3
l_{wbl}	2
T_{pipe}	0.031 sec
T_{com}	0.035 sec
T_{host}	0.033 sec
T_{total}	0.099 sec

l_{wlnk} must be equal to or larger than l^*_{wlnk}, where l^*_{wlnk} is given by:

$$l^*_{wlnk} = \frac{(4w + 3r)l^3_{box}N_{cell}}{T^*_{com}/t_{bus} - (1.5w + 2r)N_{wv}}. \tag{6.40}$$

Solid curve of figure 6.5 shows the l^*_{wlnk} as a function of T^*_{com}/t_{bus} for the case that w=2, r=3, and $l_{box} = 12$. The lower region of the solid curve (hatched) is prohibited. Dashed horizontal lines indicate possible values for N_{wlnk} when $l_{dm} = 6$. In the case of $l_{dm} = 12$, N_{wlnk} is necessary to be only a natural number. The dotted vertical line indicates $T^*_{com}/t_{bus} = 1.1 \times 10^6$ (= 33msec/30nsec). The author adopts $l_{wlnk} = 24$ for the number of links between node computers and WINE-2.

6.2.5 Calculation speed of node computers

The calculation speed of node computers, t_{fn}/N_{nd}, can be determined from inequality 6.21 using equation 6.16 and $N_{cell}l^3_{dm} = N/N_{nd}$:

$$\begin{aligned} \frac{t_{fn}}{N_{nd}} &\leq \frac{T^*_{host}}{NN_{fp}} \\ &= 8.3 \times 10^{-11} \times (\frac{T^*_{host}}{3.3 \times 10^{-2}\text{sec}})(\frac{10^6}{N})(\frac{400}{N_{fp}}). \end{aligned} \tag{6.41}$$

This inequality suggests that t_{fn}/N_{nd} must be equal to or smaller than 8.3×10^{-11}, in the case that $N_{fp} = 400$ and $T^*_{host} = 3.3\times10^{-2}$sec. The total sustained speed of node computers needs to be higher than 12 Gflops.

6.2.6 Hardware specification

The author gets $N_{nd} = 1$ or 8 (subsection 6.2.2), $N_{gcl}N_{nd} = 64$ (subsection 6.2.3), $N_{wcl}N_{nd} = 24$ (subsection 6.2.4), and $t_{fn}/N_{nd} = 8.3 \times 10^{-11}$ (subsection 6.2.5). Tables 6.4 summarizes the two possible sets of parameters of MDM. Total number of G-chips is 2560, and 3072 for W-chips.

In the case of $N_{nd} = 1$, the author gets $N_{gcl} = 64$, $N_{wcl} = 24$, and $t_{fn} = 8.3 \times 10^{-11}$. A host computer must be a shared memory computer that has 96 links and have an effective calculation speed higher than 12 Gflops. Such a type of host computer, for example, a supercomputer, may be too expensive compared with the cost of MDM hardware.

In the case of $N_{nd} = 8$, the author gets $N_{gcl} = 8$, $N_{wcl} = 3$, and $t_{fn} = 6.6 \times 10^{-10}$. A host computer must be a cluster of eight nodes. A node computer must have 11 links (PCI-bus) and have an the effective calculation speed higher than 1.5 Gflops. The cost for such a type of host computer, for example, workstation cluster, is comparable to that of MDM hardware. On the other hand, programming on the host computer will be more complicated, because one must develop parallel program on the workstation cluster.

Table 6.5 summarizes the constraint used to determine the set up parameters. Table 6.6 summarizes the number of cells for each region and calculation time for one time-step when $N = 10^6$ and $\xi = 10^{-3}$. It achieves $T_{total} \leq 0.1$ sec. Here, note that all the conditions (equations 6.2, 6.3, 6.8, and 6.12) are satisfied in these parameter sets.

6.3 Performance of MDM

In figure 6.6, the calculation time, T_{total}, of one time-step are plotted against the number of atoms (Solid curve: $\xi = 10^{-3}$, Dashed curve: $\xi = 10^{-5}$). The time, T_{total}, is proportional to N for $N < 10^6$, since it is almost dominated by T_{com} and T_{host}. On the other hand, the slope gradually increases as N increases for $N > 10^6$, since T_{pipe} become the dominant term of T_{total}. In the limit of a large N, this slope asymptotically goes to 3/2.

Figure 6.7 shows the ratios of T_{host}, T_{com}, and T_{pipe} to T_{total} against N for the case of $\xi = 10^{-3}$. At $N = 10^6$, T_{host}, T_{com}, and T_{pipe} are almost equal each other. For $N > 10^6$, T_{pipe} is the dominant term, on the other hand, T_{com} ratio is the dominant term for $N < 10^6$. The ratio of T_{host} to T_{total} almost remains constant for $N = 10^5 \sim 10^7$.

6.4 Summary

The author described the system design of Molecular Dynamics Machine, and determined the set up parameters to perform one time-step of a million-atom simulation within 0.1 second. Two types of MDM are possible.

One type of MDM is composed of eight nodes, each node is composed of a node computer, MDGRAPE-2, and WINE-2. The calculation speed of a node computer is 1.5 Gflops. MDGRAPE-2 is composed of eight G-clusters. WINE-2 is composed of three W-clusters. Each G-cluster or W-cluster is connected to a node computer by a PCI-bus. G-cluster is composed of four G-boards, each G-board has ten MDGRAPE-2 chips. W-cluster is composed of eight G-boards, each G-board has sixteen WINE-2 chips. Total number of MDGRAPE-2 and WINE-2 chips are 2560 and 3072, respectively. Communication speed between nodes must be faster than 178 Mbyte/sec. Total sustained transfer speed between node computers and MDGRAPE-2 or WINE-2 is about 14 Gbyte/sec.

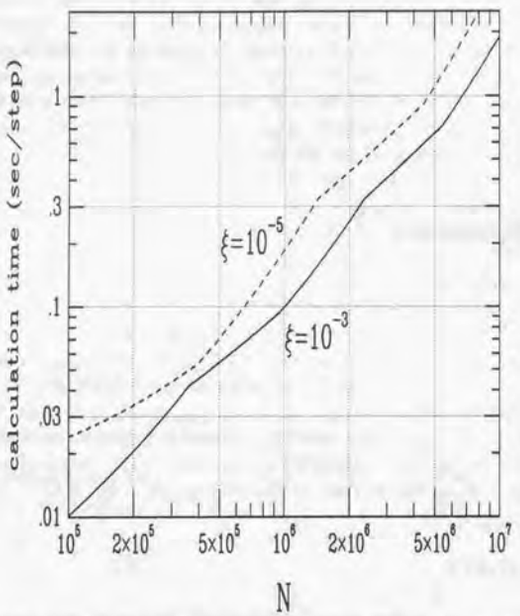


Figure 6.6: Total calculation time, T_{total}, of one time-step is plotted against N for $\xi = 10^{-3}$ (solid curve) and for $\xi = 10^{-5}$ (dashed curve).

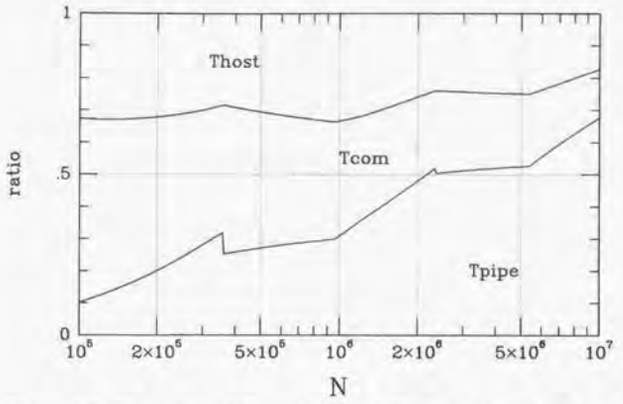


Figure 6.7: The ratios of T_{host}, T_{com}, and T_{pipe} are plotted against N when $\xi = 10^{-3}$. At larger N than 10^6, T_{pipe} is dominant, while T_{com} is dominant at smaller N. The ratio of T_{host} to T_{total} keeps almost constant.

Another type of MDM is composed of a node, MDGRAPE-2, and WINE-2. The calculation speed of a node computer is 12 Gflops. MDGRAPE-2 is composed of 64 G-clusters. WINE-2 is composed of 24 W-clusters. The structures of G-cluster and W-cluster, communication speed between a node computer and MDGRAPE-2 or WINE-2, and total number of MDGRAPE-2 and WINE-2 chips are almost the same as that of the first type of MDM.

Theoretical peak speeds of either types of MDM is about 100 Tflops. MDM is expected to achieve a sustained speed of 30 Tflops in the MD simulation with a million atoms. MDM calculates 10^6 time-steps of MD simulations with a million atoms in about a day ($\sim 10^5$ seconds). MDM will be several tens times faster than the fastest supercomputer in 1999, whose performance will be 4 Tflops.

Chapter 7

Discussion

The author described the system design of Molecular Dynamics Machine, which can perform one time-step of a million-atom simulation within 0.1 second. MDM has a theoretical peak speed of about 100 Tflops. The author found that it can sustain one third of the peak speed for the simulations with one million atoms. MDM will be several tens times faster than the fastest supercomputer scheduled in 1999 with a peak speed of 4 Tflops. MDM will enable us to simulate large biomolecules, which contain a hundred thousands atoms, surrounded by several hundreds of thousands of water molecules, without truncating Coulomb force. Such a simulation, which takes several years with a current (1998) moderate supercomputer, will accelerated by three order of magnitudes.

In the present chapter, the author reviews the current status of MDM (section 7.1), compares with other computers, (section 7.2) and then discusses the hardware to achieve a time-span of microseconds to simulate longer phenomena (section 7.3). Sophisticated algorithms to reduce calculation cost and other applications of MDM are also discussed in sections 7.4 and 7.5.

7.1 Current Status of Development

The development of the MDM hardware is now proceeding. The detailed design of WINE-2 chip is completed, and its engineering sample chips will be shipped on February in 1998. The behavior model of the design of MDGRAPE-2 chip is under construction in IBM, and the engineering sample chips of it will be shipped in the summer of 1998. The conceptual level of design of MDGRAPE-2 and WINE-2 board completed and they will complete in 1998. Total system of MDM will be completed in 1999.

7.2 Comparison with Other Computers

As described in chapter 1, several special-purpose computers for molecular dynamics have already been developed. The author compares MDM with these special-purpose computers as well as general purpose supercomputers and GRAPE-4.

7.2.1 History of special-purpose computers for molecular dynamics simulations

The requirement of molecular dynamics simulation for computer resources are so intensive that several specialized hardware were developed for them. They are divided into three generations, those are the first, second, and third generations.

First generation: Random Logic

In 1980's, several groups started to develop special-purpose computer for molecular dynamics simulations. These machines in the first generation are made by random logics, because of the limitation of technology at that time and the available budget. The machines in the first generation are DMDP, ATOMS, FASTRUN, GRAPE-2A, and WINE-1.

Bakker et al. (1988) developed Delft Molecular-Dynamics Processor (DMDP) at Delft University of Technology in 1982. DMDP, connected to HP 1000 minicomputer, performs not only force calculation but also time integration by leapfrog method. It supports linked-list (cell-index method) and PM part of P^3M method, whose microprogram is written in PROM on the board. Its performance is comparable to that of CRAY-1. DMDP is composed of many TTL-compatible components, and all the calculations were done in fixed-point arithmetics, which causes a low accuracy of calculation.

Bakker et al. (1990) developed ATOMS at AT & Bell Laboratories. ATOMS calculates 3-body force calculation as well as 2-body force calculation. It is composed of three kinds of boards, *i.e.*, an interface board, memory boards, and processor boards. A machine with eight processor boards has a theoretical peak speed of 182 Mflops, and its performance in actual simulations is 30%~100% faster than Cray XMP. Floating-point arithmetic LSIs in the processor boards calculate 3-body and 2-body forces by a microprogram downloaded from a host computer, Micro VAX II.

Fine et al. (1991) developed FASTRUN at Columbia University in 1990. FASTRUN is specialized for non-bonding force calculation, and has no microprogram in it. An increase in a computational speed is roughly a factor of six over its host computer, a Star ST100 array processor, running alone. It has a peak speed of about 270 Mflops and can calculate roughly 3 times faster than a Cray YMP. FASTRUN has four kinds of pipelines dedicated for Coulomb force, Coulomb potential, van der Waals force, and van der Waals potential calculations. The quadratic interpolation is used for pair-wise force calculation.

Ito et al. (1993) developed GRAPE-2A at University of Tokyo in 1992. GRAPE-2A is specialized for non-bonding force calculation, and has no microprogram in it. Its peak speed is about 180 Mflops and 6.5 times faster than TITAN 3000 for simulations with 3000 atoms. GRAPE-2A has one pipeline, which calculates pair-wise forces by linear interpolation.

Fukushige et al. (1993) developed WINE-1 at University of Tokyo in 1992. WINE-1 is specialized for the wave-space part of Ewald method, and performs DFT and IDFT without microprogram. Its peak speed is 480 Mflops. It is composed of fixed-point arithmetic units and values of *sine* and *cosine* are stored in ROMs.

Here, one can find two evident trends: specialization to non-bonding force and hardwired program (no-microprogram). These trends made hardwares much simpler and reduced the time and manpower for development. These were the key to win the severe race with general-purpose computer, which also quickly advancing.

The development of specialized hardware in the first generation become to be very difficult, since the speed up of the Digital Signal Processor for floating point calculation has stopped in 1990's. These DSP chips are so slow compared with advanced CPUs that the merit of specialized hardware made by random logic was almost gone. The further speed-up was achieved by making application specific LSIs, specially designed for molecular dynamics simulations.

Second generation: Specialized LSI

The machines in the second generation have a specialized LSI which includes the pipeline to calculate non-bonding forces. Those are MD-engine and MD-GRAPE.

Toyoda et al. (1995) developed MD-engine at Fuji Xerox Co. Ltd. The structure of MD-engine is similar to that of GRAPE-2A. One board of MD-engine has four specialized processor chips dedicated to non-bonding force calculation. Its peak speed is 6.5 Gflops with 19 boards. MD-engine can calculate van der Waals forces or potentials using neighbor lists generated in the boards. Pair-wise forces are evaluated by segmented interpolation, and the coefficients for interpolation are stored in the RAMs outside of the processor chips.

Fukushige et al. (1996) developed MD-GRAPE at University of Tokyo. MD-GRAPE can perform not only non-bonding force calculation but also DFT and IDFT calculation, which are used in the wavenumber-space part of Ewald method. One board of MD-GRAPE has four specialized processor chips (MD-GRAPE chips; Taiji et al. 1994), and its peak speed is 4.2 Gflops. Pair-wise force calculations are performed by 4-th order segmented interpolation, and the coefficients for interpolation are stored in RAMs inside of MD-GRAPE chips.

One board of MD-GRAPE, which is only two times larger in size than that of MD-engine, is more than ten times faster than that of MD-engine. MD-GRAPE has a higher performance because a coefficient table RAM is stored in chips unlike MD-engine. The communication between the RAM and the processor chips limits the performance of MD-engine.

The performances of the machines in the second generation are limited to several Gflops due to the limit in the clock speed, since they did not use many (less than a hundred) LSI chips. One can accelerate using many LSIs in parallel to achieve even more than Tflops, since the calculation of non-bonding force has a lot of parallelism. The extension of this direction is realized by the machines in the third new generation. It is worth noting that the machines are impressively cost-effective compared with general-purpose supercomputers, though their peak performances are moderate. The discussions in this respect will be given in the subsection 7.2.2.

Third generation: Highly Parallelized Architecture

MDM will be the first machine of the third generation for molecular dynamics simulations. As the author described in chapter 1, GRAPE-4, the first Tflops machine, has already

Table 7.1: Differences between GRAPE-4 and MDM

	GRAPE-4	MDM
peak speed	1 Tflops	100 Tflops
potential shape	$1/r$	arbitrary
individual time-step	yes (with Hermite scheme)	no
pipeline calculates	force and its derivative	force
input data	position, velocity and mass	position and charge

used many (1692) HARP chips in parallel. The author discuss MDM in comparison with GRAPE-4 in this subsection.

Major difference between them is originated from their purposes: MDM is a machine specially dedicated for molecular dynamics simulations, while GRAPE-4 is that for gravitational many body simulations. This difference in purpose produces three differences in hardware design as follows (see table 7.1).

First, MDM can treat arbitrary shape of potential including that used in real-part of Ewald method: Coefficients of interpolation polynomial are stored in a RAM in an MDGRAPE-2 chip. A host computer can rewrite the coefficients in the RAM. On the other hand, GRAPE-4 can treat only $1/r$ shape of potential (gravitational potential). Coefficients of interpolation polynomial are stored in ROM in a HARP chip. A host computer cannot rewrite the coefficients in the ROM.

Second, while GRAPE-4 can calculate time derivative of force, MDM cannot. HARP chip in GRAPE-4 calculate not only force but also its time derivative, which are used in Hermite scheme (Makino 1991, Makino and Aarseth 1992). It is worth noting that Hermite type scheme may have some merits in molecular dynamics simulations as well as in gravitational many-body simulations, although it has not been used in molecular dynamics simulations, yet.

Finally, MDM does not have specialized hardware to predict positions of particles. On the other hand, GRAPE-4 has 36 PROMETHEUS chips to predict positions (and velocities) of particles. These predicted values are used in individual time step method, which frequently used in gravitational many body simulations of globular clusters.

7.2.2 Comparison with general-purpose computers

The most impressive point of special-purpose computer like MDM or GRAPE-4, is their very good cost-effectiveness. They are typically achieve a 100 $\sim$ 1000 times good cost effectiveness compared with commercially available computers. Molecular dynamics simulations intrinsically have many parallelisms as can be seen in the previous chapter. These parallelism give MDM the following three characteristics which greatly contributes to their good cost-effectiveness.

First, MDM uses very deep pipelines with a delay of ~ 30 to calculate non-bonding forces. The current microelectronics technology allows us to squeeze more than hundred floating-point units in a chip. A specialized LSI with very deep pipelines has a high ($\sim$ several hundred) ratio of the operations to data communication with a main memory compared with conventional CPUs ($\sim$ several). MDM with deeply pipelined LSIs, memory bandwidth dose not limit the performance of the MDM system, unlike conventional computers. Also note that you can use almost all of the gates to calculations for the case of MDM with deep pipelines, while a conventional CPU wastes their gates to a huge cash memory and controllers.

Second, many specialized LSIs in MDM share the output of data from the memory on the board, since different LSIs calculate forces exerted on different particles but from the same particle in parallel. The bandwidth from the memory can be much lower than conventional computers without losing performance.

Third, MDM can use a simple and relatively slow communication network as described in the previous chapter. The communication between specialized hardware and host computer is relatively lighter than the force calculation, since in molecular dynamics simulations for large number (N) of particles, the cost of force calculations rapidly increases in proportion to $O(N^{3/2})$, while communication cost only proportional to $O(N)$. Furthermore, boards is not necessary to communicate with each other. The usage of a simpler and slower communication hardware makes system cost lower significantly compared with general-purpose parallel computer.

7.3 How to Achieve Micro-second MD Simulation: Super-MDM

There are strong motivations to extend a computational time-span to increase from nano-seconds to micro-seconds. For example, experimental data suggested that short peptides folded on a time scale of 100 nano-seconds to 10 micro-seconds (Hammes and Robers, 1968; Gruenewald, 1979). Ballew et al. (1996) showed that myogrobin molecule made a conformational change with a time scale of several hundred nanoseconds. When you could perform the molecular dynamics simulations for a relatively large system with a hundred thousand atoms, you would bring a breakthrough in the investigation of folding process of protein molecules.

However, even with MDM, it is almost impossible to perform microsecond simulations. In the present section, the author discusses the possibility to perform the micro-seconds molecular dynamics simulations by some extension of MDM.

In order to perform simulations with a time span of microseconds ($\sim 10^9$ time-steps) within reasonable time (for example, two weeks $\sim 10^6$ seconds), T_{total} must be shorter than 10^{-3} second. Figure 6.6 shows that MDM takes as long as about 10 msec at $N = 10^5$ for $\xi = 10^{-3}$. As can be seen in figure 6.7, T_{com} and T_{host} rather than T_{pipe} dominate T_{total} at $N = 10^5$.

You can reduce T_{total} to about 1 msec, if you use a larger number of chips, and faster node computers and links. The optimal set of l_{box}, k_{cut}, $N_g T^*_{pipe}$, and $N_w T^*_{pipe}$ for $N = 10^5$ are determined by a similar discussion in subsection 6.2.1. They are summarized in table

6.3. In order to achieve $T_{pipe} < 0.33$ msec, N_g and N_w must be larger than 7911 and 8016, respectively.

If taking into account all of l_{gbl}, l_{dm} and l_{dm}/l_{wbl} should be larger than or equal to 1, following constraints are obtained using similar discussions as the subsection 6.2.2, 6.2.3 and 6.2.4 for $T^*_{com} = 0.33$ msec,

$$t_{bus} < 3.2 \times 10^{-9} \text{ sec}, \tag{7.1}$$
$$t_{cn} < 5.6 \times 10^{-9} \text{ sec}. \tag{7.2}$$

These inequalities show that the sustained speed of a link between a node and MDGRAPE-2 or WINE-2, must be higher than 1.2 Gbyte/sec and that of inter-node channel must be higher than 1.4 Gbyte/sec. This restriction is much relaxed if the data can be broadcasted to all the boards of the MDM. In this case, the speed of a link between a node and MDGRAPE-2 or WINE-2 needs to be only 100 Mbyte/sec.

From equation (6.41), the constraint on the speed of node computer is obtained:

$$t_{fn} < 4.2 \times 10^{-9}, \tag{7.3}$$

where the author sets to $l_{box} = 8$, $l_{dm} = 1$, and $T^*_{host} = 3.3 \times 10^{-4}$ sec. The total sustained speed of host computers must be higher than 120 Gflops: The sustained speed of a node computer must be faster than 240 Mflops, since the total number of node computers, N_{nd}, is 512.

The requirements of the hardware for micro-second simulations are summarized as follows.

1. Number of MDGRAPE-2 and WINE-2 chips is both about 8,000, respectively.
2. Effective calculation speed of a host computer must be faster than 120 Gflops in total.
3. Effective transfer speed of links must be faster than 600 Gbyte/sec in total. Note that this specification for links would be much lower if one could develop additional hardware to broadcast data to many boards.

One may think that those are too high to achieve in near future. However, they may be relaxed by using faster algorithms, such as tree-code, P^3M method (Particle-Particle Particle-Mesh), fast multipole method (Greengard et al. 1987), or multiple time-step method (Streett et al. 1978). These faster algorithms will be discussed in the next section.

7.4 Sophisticated Algorithms

7.4.1 Smaller cells for real part of the calculation

In the present thesis, the author assumes that the side-length of a cell is set to be equal to cut-off radius, *i.e.*, $r_{cut} = L_{cell}$. However, one may reduce calculation cost for real space part (T_{gpipe}) significantly by using smaller cells in cell-index method. If one use smaller

cells, *i.e.*, large n_{rcut}, the number of interaction per particle can be reduced by a factor of 6.4, since the volume of the cube with a side of $3r_{cut}$ is $81/4\pi$ ($\simeq 6.4$) times larger than that of a sphere with a radius of r_{cut} (see figure 7.1). Here, n_{rcut} is the number of cells in one dimension within r_{cut}, and written by:

$$n_{rcut} = \frac{r_{cut}}{L_{cell}}. \tag{7.4}$$

Figure 7.2 shows the ratio of calculation cost of the cases with actual finite sized cells to the ideal case with infinitesimal sized cells. The calculation cost $\vec{F}_i$(re) decreases with increasing n_{rcut}.

However, the reduction in T_{pipe} is not so large because the efficiency of pipelines of MDGRAPE-2 is lower for smaller N_{cell} (large n_{rcut}), in actual case (see inequality 6.2 and figure 7.3). If the efficiency in the pipelines is taken into account, T_{gpipe} and T_{wpipe} are rewritten as follows:

$$T_{gpipe} = g_g \frac{27 N_{cell}^2 l_{gbl}^3}{2 N_{gcp} N_{gbd}} t_{gpipe}, \tag{7.5}$$
$$= \frac{27 g_g N N_{cell} t_{gpipe}}{2 N_g}, \tag{7.6}$$
$$= \frac{27 g_g n_{rcut}^3 N^2 t_{gpipe}}{2 N_g l_{box}^3}, \tag{7.7}$$
$$T_{wpipe} = \frac{(2g_{wd} + g_{wi}) N_{cell} l_{dm}^2 l_{wbl} N_{wv}}{16 N_{wcp} N_{wbd}} t_{wpipe}, \tag{7.8}$$
$$= \frac{(2g_{wd} + g_{wi}) N N_{wv} t_{wpipe}}{16 N_w}, \tag{7.9}$$
$$= \frac{(2g_{wd} + g_{wi})(-\ln \xi)^3 l_{box}^3 N t_{wpipe}}{24 \pi^2 n_{rcut}^3 N_w}, \tag{7.10}$$

where, g_g, g_{wd}, and w_{wi} are the loss of parallel efficiency of multiple pipelines for G2-stage, W2-stage, and W5-stage, respectively:

$$g_g = \frac{12 N_{gcp}}{N_{cell}} \left[\frac{N_{cell} + 12 N_{gcp} - 1}{12 N_{gcp}} \right], \tag{7.11}$$
$$g_{wd} = \frac{64 N_{wcp} N_{wbd}}{N_{wv}} \left[\frac{N_{wv} + 64 N_{wcp} N_{wbd} - 1}{64 N_{wcp} N_{wbd}} \right], \tag{7.12}$$
$$g_{wi} = \frac{64 N_w}{N} \left[\frac{N + 64 N_w - 1}{64 N_w} \right], \tag{7.13}$$

where $[x]$ denotes the maximum integer that does not exceed x.

In figures 7.4, g_g, g_{wd}, and g_{wi} are plotted against N_{cell}, N_{wv}, and N, respectively. The loss of efficiencies in pipelines decrease with increasing N_{cell}, N_{wv}, and N. For example, N_{cell} must be large enough compared with the number of G-pipelines on the MDGRAPE-2 board for an efficient usage of G-pipelines. As discuss in section 6.2, N_{cell} is 580 for the

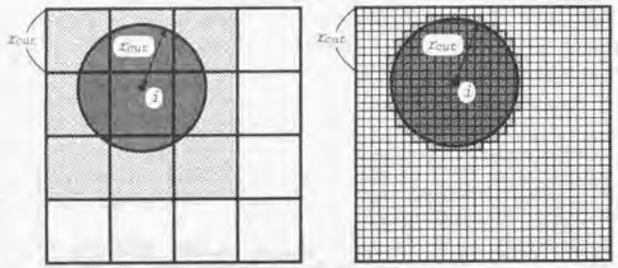


Figure 7.1: Cell-index method of two-dimensional case with large (left) and small (right) cells. The number of cells within r_{cut}, n_{rcut} is 1 (left) and 8 (right). Force of the particle i is the sum of forces from the particles in thin hatched region. In the ideal case, only the particles in thick hatched region interact with the particle i. Thin hatched region is about 2.9 or 1.2 times larger than thick hatched region in two-dimensional case, and 6.4 or 1.6 times larger in three-dimensional case in the left or right figure, respectively.

optimal case of $N = 10^6$ and $\xi = 10^{-3}$, and is larger than 120, which is the number of G-pipelines on the MDGRAPE-2 board. If you use the cell with a half side ($n_{rcut} = 2$), N_{cell} becomes smaller than the number of G-pipelines on the MDGRAPE-2 board, which causes the loss of a parallel efficiency of multiple pipelines.

Figures 7.5 shows T_{gpipe} and T_{wpipe} as functions of L/r_{cut}. Diamonds denotes T_{wpipe} and circles, squares, and crosses denote T_{gpipe} for the cases of $n_{rcut} = 1$, 2, and 3, respectively. The minimum of T_{pipe} is about 0.3 seconds, and does not change so much even for a larger n_{rcut}. In summary, there are no reduction of calculation time, even for smaller cells (larger n_{rcut}) in the calculation of the real-space part of the Coulomb force.

One may gain another factor of two, if shorter cut-off radius for $\vec{F}_i$(vdW) is used: $\vec{F}_i$(vdW) more rapidly decreases than $\vec{F}_i$(re) in typical cases. However, from similar discussions above, the author can safely conclude that the time reduction is relatively small in the calculation of van der Waals force even if one uses smaller cells.

7.4.2 Calculation of virial

The virial represents the internal stress of the system by the volume-derivative of the internal energy. The virial is used to perform molecular dynamics simulations under the constant pressure condition (equation 2.10).

The virial, π_{lm}, is calculated by,

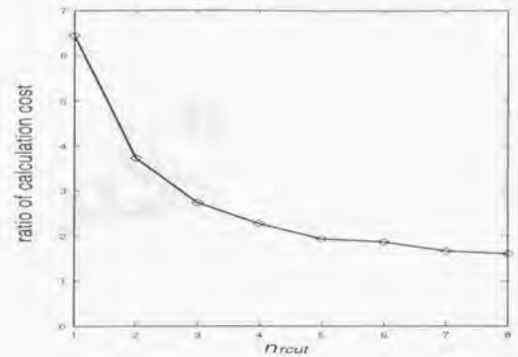


Figure 7.2: Ratio of calculation cost to the ideal case (infinitesimally small cells; $n_{rcut} \to \infty$) is plotted against n_{rcut}. As n_{rcut} increases, the ratio of increase of calculation cost for $\vec{F}_i$(re) and $\vec{F}_i$(vdW) decreases and approaches to unity at the limit of $n_{rcut} \to \infty$.

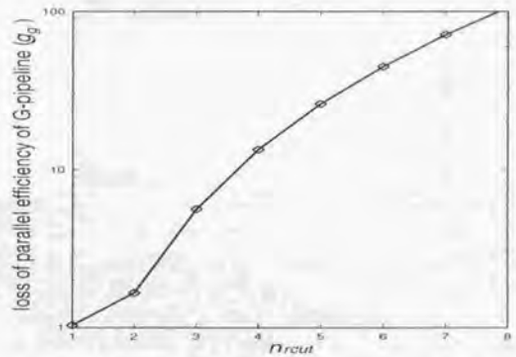


Figure 7.3: Loss, g_g, of parallel efficiency of G-pipelines is plotted against n_{rcut} when $N = 10^6$, $r_{cut} = L/12$ and $\xi = 10^{-3}$. As n_{rcut} increases, g_g increases.

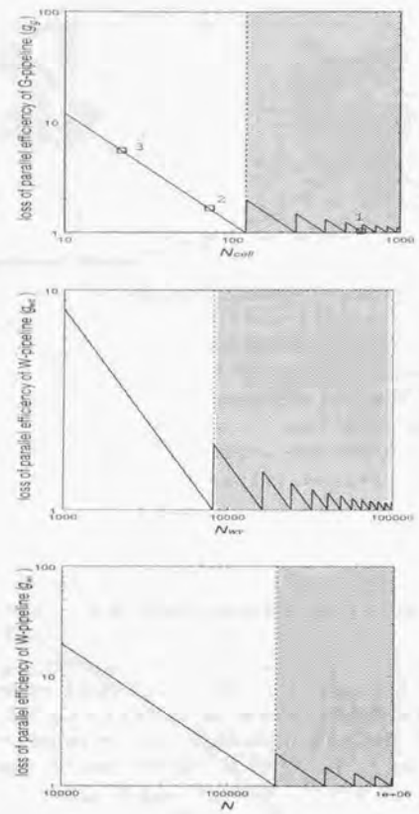


Figure 7.4: Loss, g_g, g_{wd}, and g_{wi}, of parallel efficiency of pipelines are plotted against N_{cell}, N_{wv}, and N, in the top (a), middle (b), and bottom (c) figure, respectively. Loss of efficiencies decrease with increasing N_{cell}, N_{wv}, and N, respectively. Hatched region indicate $N_{cell} > 12N_{gcp}$ (=120; inequality 6.2), $N_{wv} > 64N_{wcp}N_{wbd}$ (=8192; inequality 6.8), and $N > 64N_w$ (=196,608; inequality 6.12). Open squares in the top figure indicate the cases that $n_{rcut} = 1$, 2, and 3, respectively.

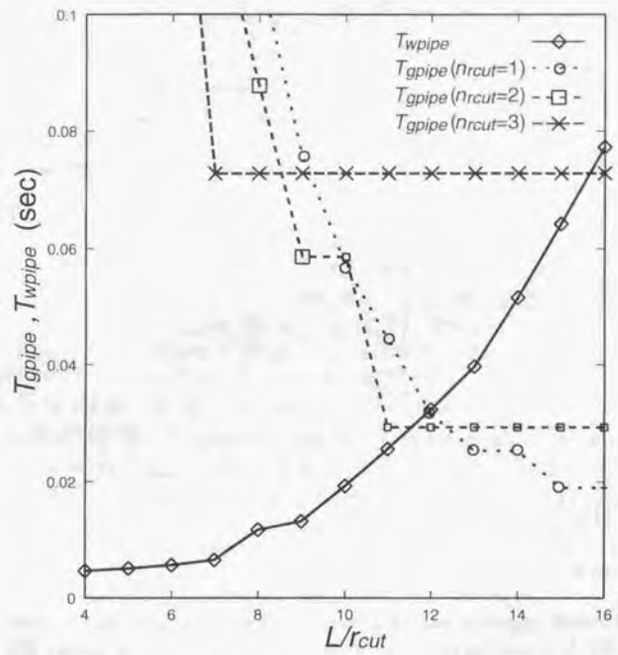


Figure 7.5: Times for hardwares, T_{gpipe} and T_{wpipe} are plotted against L/r_{cut} taking into account the efficiency of pipelines for $N = 10^6$ and $\xi = 10^{-3}$. Diamonds denote T_{wpipe} and circles, squares, and crosses denotes T_{gpipe} for the case of $n_{rcut} = 1$, 2, and 3, respectively. The minimum of T_{pipe} is about 0.03 seconds, and does not change so much even for a larger n_{rcut}.

$$\pi_{lm} = \frac{1}{2}\sum_i \sum_j f_{ij_l} r_{ij_m}, \tag{7.14}$$

where f_{ij_l} is the l-th component of $\vec{f}_{ij}$, the pair-wise force between particles i and j, and r_{ij_m} is the m-th component of $\vec{r}_{ij}$.

In the case of non-periodic boundary condition, the virial, π_{lm}, is rewritten as:

$$\begin{aligned}
\pi_{lm} &= \frac{1}{2}\sum_i \sum_j f_{ij_l} r_{ij_m}, \\
&= \frac{1}{2}\sum_i \sum_j f_{ij_l}(r_{j_m} - r_{i_m}), \\
&= \frac{1}{2}\left(\sum_i \sum_j f_{ij_l} r_{j_m} - \sum_i \sum_j f_{ij_l} r_{i_m}\right), \\
&= \frac{1}{2}\left(\sum_j r_{j_q} \sum_i f_{ij_m} - \sum_i r_{i_q} \sum_j f_{ij_m}\right), \\
&= \frac{1}{2}\left[\sum_j r_{j_m}(-f_{j_l}) - \sum_i r_{i_m} f_{i_l}\right], \\
&= -\sum_i f_{i_l} r_{i_m}, \qquad (7.15)
\end{aligned}$$

where f_{i_l} is the l-th component of $\vec{f}_i$, the force on particle i. As can be seen in equation 7.15, π_{lm} is calculated by an cost of $O(N)$ from the forces, $\vec{f}_i$, calculated with MDGRAPE-2. This virial can easily be calculated by a host computer, because $\vec{f}_i$ is already calculated by MDGRAPE-2 chips.

In the case of periodic boundary condition, the situation is a bit changed due to minimum image conversion. Suppose that the interaction region of a particle beyond the boundary of the simulation box (figure 7.6). Here, $\boldsymbol{R1}$ is the region inside the original simulation box, and $\boldsymbol{R2}$ is the region outside of it. MDGRAPE-2 calculates force from the particles in the region $\boldsymbol{R2}$, in which the positions of particles are shifted to the region $\boldsymbol{R'2}$. Therefore, π_{lm} is calculated as:

$$\begin{aligned}
\pi_{lm} &= \frac{1}{2}\sum_i \sum_j f_{ij_l} r_{ij_m}, \\
&= \frac{1}{2}\sum_i \left(\sum_j^{\boldsymbol{R1}_m} f_{ij_l} r_{ij_m} + \sum_j^{\boldsymbol{R2}_m} f_{ij_l} r_{ij_m}\right), \\
&= \frac{1}{2}\sum_i \left[\sum_j^{\boldsymbol{R1}_m} f_{ij_l}(r_{j_m} - r_{i_m}) + \sum_j^{\boldsymbol{R2}_m} f_{ij_l}(r_{j_m} + n_{i_m} L - r_{i_m})\right], \\
&= \frac{1}{2}\sum_i \left[\sum_j^{\boldsymbol{R1}_m} f_{ij_l} r_{j_m} + \sum_j^{\boldsymbol{R2}_m} f_{ij_l} r_{j_m}\right.
\end{aligned}$$

$$\begin{aligned}
&\left. - r_{i_m}\left(\sum_j^{\boldsymbol{R1}_m} f_{ij_l} + \sum_j^{\boldsymbol{R2}_m} f_{ij_l}\right) + n_{i_m} L \sum_j^{\boldsymbol{R2}_m} f_{ij_l}\right], \\
&= \frac{1}{2}\sum_i \left(\sum_j f_{ij_l} r_{j_m} - r_{i_m}\sum_j f_{ij_l} + n_{i_m} L \sum_j^{\boldsymbol{R2}_m} f_{ij_l}\right), \\
&= -\sum_i f_{i_l} r_{i_m} + \frac{L}{2}\sum_i n_{i_m} f_{ij}^m. \qquad (7.16)
\end{aligned}$$

where $\boldsymbol{R1}_m$ is the region of interaction within the simulation box in the respect of the coordinate m, and $\boldsymbol{R2}_m$ is the region of interaction outside of the simulation box in the respect of the coordinate m. Here, f_{ij}^m is the m-th component of the force, $\vec{f}_i^m$, on particle i from force-exerting particles in the region $\boldsymbol{R2}_m$:

$$f_{i_l}^m = \sum_j^{\boldsymbol{R2}_m} f_{ij_l}. \tag{7.17}$$

Minimum image conversion shifts particles from $\boldsymbol{R'2}_m$ to the region $\boldsymbol{R2}_m$, when calculating r_{ij_m}: the coordinate m of the position of particle j in $\boldsymbol{R'2}_m$ is shifted by $n_{i_m} L$. The conversion coefficient, n_{i_m}, becomes either 1 or -1, because the cut-off radius is shorter than the half length of a side of a cell, *i.e.*, $r_{cut} < L/2$.

The force from outside of the simulation box, $\vec{f}_i^m$, can be calculated by MDGRAPE-2 by setting the particles in the region $\boldsymbol{R2}_m$ as force-exerting particles. The additional calculation of force for virial by equation 7.16 is not so large. For example, this increase of calculation cost is about 17% when cutoff radius, r_{cut}, is $L/12$ ($l_{box} = 12$; see figure 7.7), because total number of cells is $1728 = 12^3$, while the minimum image conversion occurs only in the 728 ($=12^3 - 10^3$) cells on the surface of simulation box and the region $\boldsymbol{R2}$ is 1/3 of the total region $\boldsymbol{R1}$+$\boldsymbol{R2}$.

7.4.3 Fast algorithms for calculation of Coulomb force

There are faster algorithms, tree-code, P^3M method (Particle-Particle Particle-Mesh), fast multipole method (Greengard et al., 1987), or multiple time-step method (Streett et al., 1978) than Ewald Method.

MDM also can accelerate these fast algorithms significantly. In fact, tree-code and P^3M method has been already implemented on GRAPE-1A and MD-GRAPE, respectively, for astrophysical simulations (Makino et al. 1991, Miyazaki 1996).

In P^3M method, the contributions from distant particles are calculated with First Fourier Transformation by a regular mesh (PM part), while those from nearby particles are calculated by direct method (PP part):

$$\vec{F}_{pp} = f(r_{ij})\vec{F}_i, \tag{7.18}$$

$$\vec{F}_{pm} = (1 - f(r_{ij}))\vec{F}_i \tag{7.19}$$

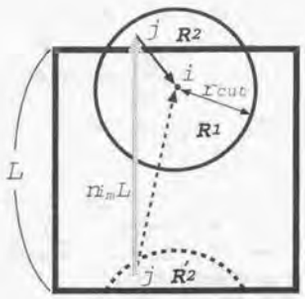


Figure 7.6: Minimum image conversion occurs at force calculation. The region $\boldsymbol{R_1}$ is inside of the simulation box, while the region $\boldsymbol{R_2}$ is outside of it. The positions of particles in the region $\boldsymbol{R_2}$ are shifted from the original positions of particles in the region $\boldsymbol{R'_2}$.

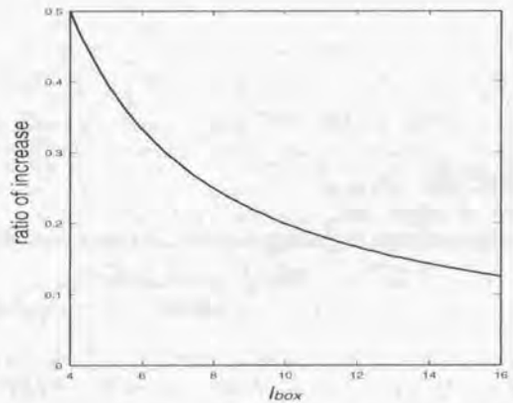


Figure 7.7: Ratio of increase of calculation cost for virial is plotted against the number of cells in one dimension, l_{box}. The ratio decreases with increasing l_{box}. The increase is only about 17% in the case that $l_{box} = 12$, which is the optimal value when $N = 10^6$ and $\xi = 10^{-3}$.

where $f(r_{ij})$ is the truncation function, which vanishes in the limit of a large r_{ij}. An theoretical discussion suggested that the calculation time for P^{3}M method is scaled as $O(N\log N)$.

MDM with MDGRAPE-2 chip and WINE-2 chip can accelerate both PP and PM part. MDGRAPE-2 chip calculate PP part, since the Particle-Particle force, $\vec{F}_{pp}$, is a central force. Miyazaki et al. (1996) implemented P^{3}M on a system composed of MD-GRAPE with a peak speed of about 4 Gflops and a workstation with an SuperSPARC (36MHz) as a host computer. He found that computational time is reduced by a factor of 7 in the case with MD-GRAPE, compared with the case that whole of the calculation done in a host computer.

MDM with MDGRAPE-2 chip and workstation cluster is, therefore, expected to show a high performance for P^{3}M method. It is worth noting that WINE-2 chip, perform discrete Fourier transformation (not FFT) may be also usable in PM part.

In the tree-code, particles are organized in the form of a tree and the each node of tree represents of the group of particles. The calculation cost of tree code is found to be scaled as $O(N\log N)$. The force of the distant node is replaced by the force from the center of mass of the node. This method first developed for the calculations of gravitational many body system (Barnes and Hut 1986). Saito (1992) applied this method to molecular dynamics calculation of proteins. Since molecular dynamics simulations usually require a higher accuracy than the gravitational many-body systems, he calculated the force from a dipole at the center of the node instead of a monopole.

Makino (1991) implemented this algorithms on GRAPE-1A for the simulations of galaxy collisions. He found that tree code is actually compatible with GRAPE system and significant (by a factor of $30 \sim 50$) acceleration can be gained by GRAPE-1A with a peak speed of 240 Mflops compared with the case that whole of the calculations are done by the host computer with a MIPS R-3000. Therefore, MDM with MDGRAPE-2 chip is expected to show a high performance also for tree-code.

Fast multipole method calculates force by multipole expansion. The force-exerting particles are organized in the form of a tree, like tree code. In fast multipole method, unlike tree-code, force-receiving particles are also organized in a tree structure. The forces exerted on particles in a node are obtained by Taylor expansion at the center of the node. The calculation cost is proven to be scaled as only $O(N)$ Fast multipole method is also compatible with GRAPE systems and is expected to be accelerated by them like tree-code, though its implementation on GRAPE systems have not yet done.

The detailed discussions of the performance of MDM for these fast algorithms are certainly important issues, but is beyond of the scope of the present thesis. They will be discussed elsewhere.

7.5 Other Application of MDM than MD Simulation

7.5.1 Smoothed particle hydrodynamics

Smoothed Particle Hydrodynamics (hereafter SPH) method represents fluid by the overlapping of many smoothed particles, which convey the mass, momentum, internal energy and other characteristics of fluid (Evrard 1988; Hernquist and Kats 1989; Navarro and Benz 1991; Cen 1992; Umemura 1993). In SPH method, hydrodynamical interactions are represented by short range interactions between particles. Most cosmological hydrodynamical simulations have been performed by SPH method (for example, Lucy 1977).

In SPH method, the most of computational time are consumed in the calculations of gravitational force and hydrodynamical interactions between particles. Umemura (1993) suggested that GRAPE can accelerate both of them, if it can make the list of neighbor particles for the hydrodynamical interactions during calculations of gravitational force. A host computer calculate hydrodynamical interactions using this neighbor list in a high speed. Steinmets (1996) implemented an SPH code on GRAPE-3, which he calls GRAPESPH.

MDM can use the astronomical calculations with SPH code, since MDGRAPE-2 chip in MDM can make a list of neighbors during the calculation of the forces. The neighbor RAM in MDGRAPE-2 can store 2048 neighbors for 24 force-receiving particles. According to Steinmets (1996), the number of neighbors is necessary to be larger than a hundred. The number of neighbors per particle is limited to 85 in the worst case that the force-receiving particles do not share neighbors at all. This limitation is, however, much relaxed in actual cases that 24 force-receiving particles which are calculated simultaneously in a MDGRAPE-2 chip share their neighbors.

7.5.2 Vortex dynamics simulation

The behavior of incompressible fluid can be represented by vortices The vortex elements move with fluid. The velocity field, $\vec{u}_i$, of fluid is calculated by Biot-Savart's law:

$$\vec{u}(\vec{r}) = \frac{1}{4}\int \frac{\vec{w}(\vec{r}\,') \times (\vec{r} - \vec{r}\,')}{|\vec{r} - \vec{r}\,'|^3} d\vec{r}\,', \tag{7.20}$$

where $\vec{w}$ is the vorticity and $\times$ denotes cross product.

Hachisu et al. (1995) found that GRAPE-3AF can accelerate this vortex dynamics simulations. They modified Biot-Savart's law a little as:

$$\vec{u}(\vec{r}) = \frac{1}{4}\int \frac{\vec{w}(\vec{r}\,') \times (\vec{r} - \vec{r}\,')}{(|\vec{r} - \vec{r}\,'|^2 + \varepsilon^2)^{3/2}} d\vec{r}\,', \tag{7.21}$$

to avoid the numerical singularity (Rosenhead 1930). Vortecies are divided into many vortex elements and each element is described as Lagrangian particles which obey Biot-Savart's law. If one define $\vec{q}_j$ as:

$$\vec{q}_j = (q_{x,j}, q_{y,j}, q_{z,j}) = -\frac{1}{4}\vec{w}_j V_j, \tag{7.22}$$

where $\vec{w}_j$ is the vortecies and V_j is the volume of each vortex element, then one obtains,

$$\vec{u}_i = -\sum_{j=1}^{N} \frac{\vec{q}_j \times (\vec{r}_i - \vec{r}_j)}{(|\vec{r}_i - \vec{r}_j|^2 + \varepsilon^2)^{3/2}}. \tag{7.23}$$

Then, the induced velocity is calculated by using GRAPE-3AF three times to get the following three quantities $\vec{K}_{x,i}$, $\vec{K}_{y,i}$ and $\vec{K}_{z,i}$:

$$\vec{K}_{x,i} = -\sum_{j=1}^{N} \frac{q_{x,j}(\vec{r}_i - \vec{r}_j)}{(|\vec{r}_i - \vec{r}_j|^2 + \varepsilon^2)^{3/2}}, \tag{7.24}$$

$$\vec{K}_{y,i} = -\sum_{j=1}^{N} \frac{q_{y,j}(\vec{r}_i - \vec{r}_j)}{(|\vec{r}_i - \vec{r}_j|^2 + \varepsilon^2)^{3/2}}, \tag{7.25}$$

$$\vec{K}_{z,i} = -\sum_{j=1}^{N} \frac{q_{z,j}(\vec{r}_i - \vec{r}_j)}{(|\vec{r}_i - \vec{r}_j|^2 + \varepsilon^2)^{3/2}}. \tag{7.26}$$

The three components of the velocity of the fluid $(u_{i_x}, u_{i_y}, u_{i_z})$ at the element i is calculated as:

$$u_{i_x} = K_{y,i_z} - K_{z,i_y}, \tag{7.27}$$

$$u_{i_y} = K_{z,i_x} - K_{x,i_z}, \tag{7.28}$$

$$u_{i_z} = K_{x,i_y} - K_{y,i_x}, \tag{7.29}$$

in a host computer. The position of vortex elements are updated based on the velocity $\vec{u}_i$ on the host computer. The vortex, $\vec{w}_j$, is calculated so that the circulation around vortex elements may conserve.

MDM will accelerate vortex dynamics simulations like GRAPE-3AF. MDGRAPE-2 chip can calculate velocity induced by vortex elements with an arbitrary core shape. Researchers of vortex dynamics want to use core functions other than Plummer function (equation 7.21) to improve convergence of simulation. MDGRAPE-2 chip can handle scale factors, a_{ij} and b_{ij}, and softening parameter, ε_{ij}, even if they depend on not only force-receiving particles but also force-exerting particles. This function of MDGRAPE-2 chip is usable to introduce viscosity in the simulations of incompressible fluid.

Bibliography

[1] Abrahams, J.P., Leslie, A.G.W., Lutter, R and Walker, J.E. 1994. Structure at 2.8 angstroms resolution of F_1-ATPase from bovine heart mitochondria. *Nature*, **370**, pp. 621-628.

[2] Alder, B. J. and Wainwright, T. E. 1959. *J. Chem. Phys.*, **31**, 459.

[3] Allen, M. P. and Tildesley, D. J. 1987. Computer Simulation of Liquids. Clarendon, Oxford.

[4] Bakker, A. F., and Bruin., C. 1988. in *Special Purpose Computers*, edited by B. J. Alder (Academic Press, San Diego), pp. 183-232.

[5] Bakker, A. F., Gilmer, G. H., Grabow, M. H., and Thompson, K. 1990. A Special Purpose Computer for Molecular Dynamics Calculations. *J. Comp. Phys.*, **90**, pp. 313-335.

[6] Ballew, R. M., Sabelko, J. and Gruebele, M. 1996. Observation of distinct nanosecond and microsecond protein folding events. *Nature Struct. Biol.*, **3**, pp. 923-926.

[7] Barnes, J., and Hut, P. 1986. A hierarchical θ $(N \log N)$ force-calculation algorithm. *Nature*, **324**, pp. 446-449.

[8] Beeman, D. 1976. Some multistep methods for use in molecular dynamics calculations. *J. Comp. Phys.*, **20**, pp. 130-139.

[9] Beveridge, D.L. and Ravishanker, G. 1994. Molecular dynamics studies of DNA. *Curr. Opinion Struct. Biol.*, **4** pp. 246-255.

[10] Burt, S. K., Machey, D., and Hagler, A. T., 1989. *Computer-Aided Drug Design*, Perun, T. J., and Propst, C. L., Eds., *Theoretical Aspects of Drug Design*, Marcel-Dekker Inc., New York. pp. 55-91.

[11] Cen, R. Y. 1992. *ApJS*, **78**, p. 341.

[12] Deisenhofer, J., Epp, O., Mikki, K., Huber, R. and Michel, H. 1985. Structure of protein subunits in the photosynthetic reaction center of *Rhodopseudomonas viridis* at 3 Åresolution. *Nature*, **318**, pp. 618-624.

[13] DiCapua, F. M., Swaminathan, S., and Beveridge, D. L. 1990. Theoretical evidence for destabilization of an α-helix by water insertion: molecular dynamics of hydrated decaalanine. *J. Am. Chem. Soc.*, **112**, pp. 6768-6771.

[14] Eastwood, J. W., Hockney, R. W., and Lawrence, D. 1980. P3M3DP-the three dimensional periodic particle-particle / particle-mesh program. *Comput. Phys. Commun.*, **19**, pp. 215-261.

[15] Ebisuzaki, T., Makino, J., Fukushige, T., Taiji, M., Sugimoto, D., Ito, T., and Okumura, S.K. 1993. GRAPE Project: An Overview, *Publs. Astron. Soc. Japan*, **45**, pp. 269-278.

[16] Eriksson, M., Härd, T. and Nilsson, L. 1995. Molecular dynamics simulations of the glucocorticoid receptor DNA-binding domain in complex with DNA and free in solution. *Biophys. J.*, **68**, pp. 402-426.

[17] Evrard, A. E., 1988, *MNRAS*, **235**, p. 911.

[18] Ewald, P.P. 1921. *Ann. Phy.*, **64**, 253.

[19] Fasman, C. D., Ed., 1989. *Prediction of Protein Structure and the Principals of Protein Conformations*, Plenum Press, New York.

[20] Feher, G., Arno, T.R. and Okamura, M.Y. 1988. The effect of an electric field on the charge recombination rate of $D^+Q_A^- \rightarrow DQ_A$ in reaction centers from *rhodobacter sphaeroides* R-26. *The Photosynthetic Bacterial Reaction Center: Structure and Dynamics*, eds., J. Breton and A. Vermeglio. pp. 271-287.

[21] Fine, R., Dimmler, G., and Levinthal, C. 1991. FASTRUN: A Special Purpose, Hardwired Computer for Molecular Simulation. *Proteins*, **11**, pp. 242-253.

[22] Fukushige, T., Makino, J., Ito, T., Okumura, S., Ebisuzaki, T., and Sugimoto, D. 1993. WINE-1: Special-Purpose Computer for N-body Simulations with a Periodic Boundary Condition, *Publs. Astron. Soc. Japan*, **45**, pp. 361-375.

[23] Fukushige, T., Taiji, M., Makino, J., Ebisuzaki, T., and Sugimoto, D. 1996. A Highly-Parallelized Special-Purpose Computer for Many-body Simulations with An Arbitrary Central Force: MD-GRAPE. *Astrophysical Journal*, **468**, pp. 51-61.

[24] Fincham, D. 1993. Optimization of the ewald sum. *Information Quarterly - CCP5*, **38**, pp. 17-24.

[25] Garvey, E. P., Swank, J. and Matthews, C. R. 1989. A hydrophobic cluster forms early in the folding of dihydrofolate reductase. *Proteins: Struct. Funct. Genet.*, **6**, pp. 259-266.

[26] Greengard, L., and Rokhlin, V. 1987. A fast algorithm for particle simulation. *J. Comp. Phys.*, **73**, pp. 325-348.

[27] Gruenewald, B., Nicola, C. U., Lesig, A. and Schwarz, G. 1979. Kinetics of the helix-coil transition of a polypeptide with non-ionic side groups, derived from ultrasonic relaxation measurements. *Biophys. Chem.*, **9**, pp. 137-147.

[28] Hachisu, I., Makino, J., Ebisuzaki, T., and Sugimoto, D. 1995. Three-dimensional vortex method on GRAPE-3AF - a special purpose computer for vortex method. *Parallel Computational Fluid Dynamics: New Algorithms and Applications*, eds. N. Satofuka, J. Periaux, and A. Ecer (Elsevier Sciences), pp. 155-160.

[29] Hammes, G. G. and Roberts, P. B. 1968. Dynamics of the helix-coil transition in poly-L-ornithine. *J. Am. Chem. Soc.* **91**, pp. 1812-1816.

[30] Harris, L. F., Sullivan, M. R., Popken-Harris, P. D. and Hickok, D. F. 1994. Molecular dynamics simulations in solvent of the glucocorticoid receptor protein in complex with a glucocorticoid response element DNA sequence. *J. Biom. Struct. Dyn.*, **12**, pp. 249-270.

[31] Heller, H., Schaefer, M. and Schulten, K. 1993. Molecular dynamics simulation of a bilayer of 200 lipids in the gel and in the liquid crystal-phases. *J. Phys. Chem*, **97**, pp. 8343-8360.

[32] Hernquist, L. and Katz, N. 1989. *ApJS*, **70**, p. 419.

[33] Hockney, R. W., and Eastwood, J. W. 1981. Computer Simulation Using Particles. New York, McGraw-Hill.

[34] Hoover, W. G. 1985. Canonical dynamics: equilibrium phase-space distributions. *Physical Review A*, **31**, pp. 1695-1697.

[35] Ito, T., Makino, J., Fukushige, T., Ebisuzaki, T., Okumura, S.K., and Sugimoto, D. 1993. A Special-Purpose Computer for N-Body Simulations: GRAPE-2A, *Publs. Astron. Soc. Japan*, **45**, pp. 339-348.

[36] Iwata, S., Ostermeier, C., Ludwig, B. and Michel, H. 1995. Structure at 2.8 *A* resolution of cytochrome *C* oxidase from *Paracoccus denitrificans*. *IWAT95*, **376**, pp. 660-669.

[37] Levinthal, C. 1968. Are there pathways for protein folding? *J. Chem. Phys.*, **65**, pp. 44-45.

[38] Lorenz, M., Popp, D. and Holmes, K.C. 1993. Refinement of the F-actin model against X-ray fiber diffraction data by the use of a directed mutation algorithm. *J. Mol. Biol.*, **234**, pp. 826-836.

[39] Lucy, L. 1977. *AJ*, **82**, p. 1013.

[40] Makino, J. 1991. Treecode with a Special-Purpose Processor. *Astrophysical Journal*, **369**, p. 200.

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

[42] Makino, J., Kokubo, E., and Taiji, M. 1993. HARP: A Special-Purpose Computer for N-Body Problem, *Publ. Astron. Soc. Japan*, **45**, pp. 377-392.

[43] Makino, J. and Taiji, M. 1995. Astrophysical N-Bodysimulations on GRAPE-4 Special-Purpose Computer, *Supercomputing '95* San Diega, USA, in Proceedings Supercomputing '95 1995 (IEEE Computer Press, Los Alamitos) (CD-ROM).

[44] Makino, J., Taiji, M., Ebisuzaki, T., and Sugimoto, D. 1997. GRAPE-4: A massively-parallel special-purpose computer for Collisional N-body Simulations. *Astrophysical Journal*, **480**.

[45] Miyazaki, N. 1996. Acceleration of Particle-Particle/Particle-Mesh Method by GRAPE. *in Master's thesis*, College of Arts and Sciences, University of Tokyo.

[46] Navarro J. F. and Benz, W. 1991, *ApJ*, **380**, p. 320.

[47] Nosé, S. 1984. A molecular dynamics method for simulations in the canonical ensemble. *Molecular Physics*, **52**, pp. 255-268.

[48] Nosé, S. and Klein, M. L. 1983. Constant pressure molecular dynamics for molecular systems. *Molecular Physics*, **50**, pp. 1055-1076.

[49] Parrinello, M. and Rahman, A. 1981. Polymorphic transitions in single crystals: A new molecular dynamics method. *J. Applied Phys*, **52**, pp. 7182-7190.

[50] Pastor, R.W. and Venable, R.M. 1993. Molecular and Stochastic dynamics simulation of lipid molecules. *Computer simulation of biomolecular systems: Theoretical and Experimental applications*, eds., W.F. van Gunsteren, P.K. Weiner and A.K. Wilkinson, pp. 443-463.

[51] Radford, S. E., Dobson, C. M., and Evans, P. A. 1992. The folding of hen lysozyme involves partially structured intermediates and multiple pathways. *Nature*, **358**, pp. 302-307.

[52] Roder, H. and Wuthrich, K. 1986. Protein folding kinetics by combined use of rapid mixing techniques and NMR observation of individual amide protons. *Proteins: Struct. Funct. Genet.*, **1**, pp. 34-42.

[53] Rosenhead, L. 1930. *Proc.R. Soc. London*, **A127**, p. 590.

[54] Saito, M. 1992. Molecular Dynamics Simulations of Proteins in Water Without the Truncation of Long-range Coulomb Interactions. *Molecular simulation*, **8**, pp. 321-334.

[55] Saito, Minoru. 1995. Molecular Dynamics/Free Energy Study of a Protein in Solution with All Degrees of Freedom and Long-Range Coulomb Interactions. *The Journal of physical chemistry*, **99**, pp. 17043.

[56] Sancho, J., Neira, J. L. and Fersht, A. R. 1992. An N-terminal fragment of barnase has residual helical structure similar to that in a refolding intermediate. *J. Mol. Biol.*, **224**, pp. 749-758.

[57] Steinmetz, M. 1996. *MNRAS*, **278**, p. 1005.

[58] Sugimoto, D., Chikada, Y., Makino, J., Ito, T., Ebisuzaki, T., and Umemura, M. 1990. A Special Purpose Computer for Gravitational Many-Body Problems, *Nature*, **345**, pp.33-35.

[59] Sugita, Y. and Kitao, A. 1998. Improved Protein Free Energy Calculation by More Accurate Treatment of Nonbonded Energy: Application to Chymotrypsin Inhibitor 2, V57A. *Proteins*, in press.

[60] Streett, W., B., Tildesley, D. J., and Saville, G. 1978. Multiple time step methods and an improved potential function for molecular dynamics simulations of molecular liquids. *Computer modeling of matter.(ed. P. Lykos)*, pp. 144-158, American Chemical Society, Washington.

[61] Taiji, M., Fukushige, T., Makino, J., Ebisuzaki, T., and Sugimoto, D. 1994. MD-GRAPE: A Parallel Special-Purpose Computer System for Classical Molecular Dynamics Simulations, *Physics Computing '94* Lugano, Switzerland, in *Proceedings of the 6th Joint EPS-APS international conference on Physics Computing*, European Physical Society, Geneva, pp. 200-203.

[62] Tanaka, H. 1996. *Nature*, **380**, p. 328.

[63] Tirion, M.M., ben-Avraham, D., Lorenz, M. and Holmes, K.C. 1995. Normal modes as refinement parameters for the F-actin model. *Biophys. J.*, **68**, pp. 5-12.

[64] Toyoda, S., Hashimoto, E., Ikeda, H., and Miyakawa, N. 1995. Development of a High Speed Special-Purpose Parallel Processor System for Molecular Dynamics Simulations. *Fuji Xerox Technical Report*, **10**.

[65] Tuckerman, M., Berne, B. J., and Martyna, G. J. 1992. *J. Chem. Phys.*, **97**, p. 1990.

[66] Umemura, M. 1993. *ApJ*, **406**, p. 361.

[67] Van Buuren, A. R. and Berendsen, H. J. 1993. Molecular dynamics simulation of the stability of a 22-residue α-helix in water and 30% trifluoroethanol. *Biopolymers*, **33**, pp. 1159-1166.

[68] Verlet, L. 1967. Computer 'experiments' on classical fluids. I. thermodynamical properties of lennard-jones molecules. *Physical Review*, **165**, pp. 201-214.

[69] Warshel, A., 1991. *Computer Modeling of Chemical Reactions in Enzymes and Solution*, John Wiley & Sons, New York.

[70] Weiner, S. J., Kollman, P. A., Case, D. A., Singh, U. C., Ghio, C., Alagona, G., Profeta, S., Jr., Weiner, P. 1984. *J. Am. Chem. Soc.*, **106**, pp. 765-784.

[71] Yasuoka, K. and Matsumoto, M. 1998. Molecular Dynamics of Homogeneous Nucleation in the Vapor Phase. *J. Chem. Phys.*, in press.

[72] Zhou, F and Schlten, K. 1995. Molecular dynamics study of a membrane-water interface. *J. Phys.*, **99**, pp. 2194-2208.

[73] Zhou, F. and Schulten, K. 1996. Molecular Dynamics Study of Phospholipase A_2 on a Membrane Surface. *Proteins*, **25 (1)**, pp. 12-27.