Hardware Acceleration of Tensor-Structured Multilevel Ewald Summation Method on MDGRAPE-4A, a Special-Purpose Computer System for Molecular Dynamics Simulations

Gentaro Morimoto
RIKEN BDR, Osaka, Japan
gentaro.morimoto@riken.jp

Yohei M. Koyama
RIKEN BDR, Osaka, Japan
ym.koyama@riken.jp

Hao Zhang*
RIKEN BDR, Osaka, Japan
zhanghao2021@gusulab.ac.cn

Teruhisa S. Komatsu
RIKEN BDR, Osaka, Japan
teruhisa.komatsu@riken.jp

Yousuke Ohno
RIKEN BDR, Osaka, Japan
ohno@riken.jp

Keigo Nishida
RIKEN BDR, Osaka, Japan
keigo.nishida.jg@riken.jp

Itta Ohmura
RIKEN BDR, Osaka, Japan
ohmura@riken.jp

Hiroshi Koyama
RIKEN BDR, Yokohama, Japan
hkoyama@riken.jp

Makoto Taiji†
RIKEN BDR, Osaka, Japan
taiji@riken.jp

ABSTRACT

We developed MDGRAPE-4A, a special-purpose computer system for molecular dynamics simulations, consisting of 512 nodes of custom system-on-a-chip LSIs with dedicated processor cores and interconnects designed to achieve strong scalability for biomolecular simulations. To reduce the global communications required for the evaluation of Coulomb interactions, we conducted a co-design of the MDGRAPE-4A and the novel algorithm, tensor-structured multilevel Ewald summation method (TME), which produced hardware modules on the custom LSI circuit for particle–grid operations and for grid–grid separable convolutions on a 3D torus network. We implemented the convolution for the top-level grid potentials by using 3D FFTs on an FPGA, along with an FPGA-based octree network to gather grid charges. The elapsed time for the long-range part of Coulomb is 50 μs, which can mostly overlap with those for the short-range part, and the additional cost is approximately 10 μs/step, which is only a 5% performance loss.

*Current affiliation: Gusu Laboratory of Materials, Suzhou, China.
†Corresponding author.

SC '21, November 14–19, 2021, St. Louis, MO, USA

ACM 978-1-4503-8442-1/21/11…$15.00
https://doi.org/10.1145/3458817.3476190

CCS CONCEPTS

• **Computer systems organization~Architectures~Other architectures~Special purpose systems** • *Applied computing~Life and medical sciences~Computational biology~Molecular structural biology* •Mathematics of computing~Mathematical analysis~Quadrature

ACM Reference format:
Gentaro Morimoto, Yohei M. Koyama, Hao Zhang, Teruhisa S. Komatsu, Yousuke Ohno, Keigo Nishida, Itta Ohmura, Hiroshi Koyama and Makoto Taiji. 2021. Hardware Acceleration of Tensor-Structured Multilevel Ewald Summation Method on MDGRAPE-4A, a Special-Purpose Computer System for Molecular Dynamics Simulations. In *The International Conference for High Performance Computing, Networking, Storage and Analysis (SC '21), November 14—19, 2021, St. Louis, MO. USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3458817.3476190

I. Introduction

Molecular dynamics (MD) simulations—one of the most important applications of high-performance computing—are commonly used in various fields, such as material sciences, engineering, and biological sciences. Among them, the simulation of biological macromolecules such as proteins, RNAs, and DNAs has expanded its practical applications in the last 20 years because of advances in structural biology and computing performance. Classical MD simulations treat atoms as point masses that obey Newton's equation of motion. To simulate the dynamics of each atom, we calculate the forces between the atoms and integrations of the equation of motion by using a

discrete-time method. The fastest time scale of the system determines the time step, with steps of 0.5–2.5 fs typically used. As many biologically relevant phenomena occur beyond a microsecond, the required number of steps reaches at least 10^9; in some cases, it requires over 10^{12} steps. If a time step requires 10 ms to be simulated on a computer, 10^9 steps require 116 days. Therefore, faster computers are required for more realistic MD simulation applications.

To accelerate MD simulations, several special-purpose computer systems have been developed, starting with the Delft Molecular Dynamics Processor [3]. The development of modern computing accelerators started with GRAPE-2A [16], a modified version of the accelerator for astrophysical *N*-body simulations [20]. Development of the first large-scale integrated (LSI) circuit–based accelerator "MD-GRAPE" [9] and the accelerators for the Ewald method "WINE" [8,23] followed. These accelerators focused on the acceleration of nonbonded force calculations, which is the most computationally demanding part of the MD simulations. The other calculations, including those of bonded forces, long-range forces, and integration, were performed on a conventional general-purpose computer.

MDGRAPE-3, completed in 2006, was the first PFLOPS scale computer system in history [38]. This system consisted of a hundred-node PC cluster equipped with accelerator boards and Infiniband interconnects. It was one order of magnitude faster than the same simulation without the accelerator [22,26]; however, communication between cluster nodes limited the strong scalability of the MDGRAPE-3 system.

Because the accelerator gained rapid performance increase by following Moore's law, bottlenecks began to emerge in many other parts. With the recent increase in semiconductor performance, the elapsed time for a single time step of MD now reaches a millisecond order with an accelerator including GPGPU or even with a commodity cluster. However, reducing the elapsed time by simply increasing the number of compute nodes is impossible because not merely FLOPS counts matter. Communication latency limits the strong scalability of classical MD simulations, including synchronization. In particular, the evaluation of Coulomb interactions is the most difficult part of the MD computation to be accelerated. Because of its long-range nature, global communication over all computing nodes is essential for its evaluation.

The Anton family developed by D. E. Shaw research [34,35] achieved strong scalability by connecting the system-on-a-chip (SoC) LSIs as compute nodes with a direct communication interface. They successfully solved the bottlenecks in the MD computation by integrating dedicated pipelines, general-purpose processor cores, and network interfaces. They also designed hardware support to accelerate the 3D FFT calculation on nodes necessary to implement the Gaussian split Ewald method [33] for Coulomb interactions.

Another special purpose computer system, MDGRAPE-4, was designed using the integrated SoC approach, which is similar to the architecture of Anton, targeting a further improved performance compared with MDGRAPE-3 [25]. The development of MDGRAPE-4 was completed in 2014; however, it could not be fully operated owing to certain deficiencies. MDGRAPE-4A is the revised version of MDGRAPE-4; it was successfully completed in 2019. Our enhancements to the previous MDGRAPE-4 system included reconsidering the calculation of Coulomb interactions from an algorithm perspective, and developing a novel numerical algorithm named the tensor-structured multilevel Ewald summation method (TME) (Y. M. Koyama et al., in preparation). The TME method reduces the network latency at the expense of computation. In this paper, we describe the theory and performance of hardware accelerated implementation of TME on MDGRAPE-4A. Although its performance is still lower than that of Anton, its algorithm and architecture will be important for future accelerations of MD simulations not only on special-purpose computers but also on general-purpose machines.

II. MDGRAPE-4A: An Overview

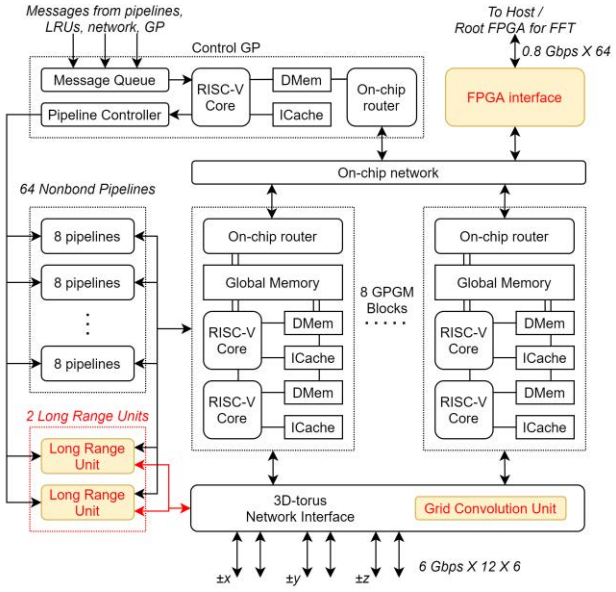

Fig. 1: Block diagram of MDGRAPE-4A SoC. DMem in GPGM and CGP blocks are scratchpad memories of 64 KB and 256 KB, respectively. ICache in GPGM and CGP blocks are instruction cache memories of 64 KB. Other parts are described in the main text.

This section provides a brief introduction to the MDGRAPE-4A system. The MDGRAPE-4A system consists of 512 SoCs to achieve strong scalability of MD simulations. We developed a proprietary network for the system interconnects in a 3D-torus topology of 8 × 8 × 8. The atoms in the simulation system, whose number is typically 100,000, were decomposed spatially into rectangular cells; each cell was managed by a node at a corresponding coordinate.

Fig. 1 depicts a block diagram of MDGRAPE-4A SoC. It contains eight GPGM blocks with a global memory (GM) of 256 KB, along with two general-purpose (GP) cores. The instruction set of the GP cores is based on RISC-V 32 bit with an extension

of a four-way 32-bit fixed-point SIMD unit to efficiently manipulate 3D vectors. The GP cores integrate the equation of motion, evaluations of bonded interactions, and a portion of a long-range part of the Coulomb interaction. The GM has a special write mode for accumulations of 32-bit fixed-point values on stored values, which is useful for summing all values calculated on distributed units and nodes without a lock, such as forces acting on atoms or charges interpolated on grid points. The GM also has a function to manage cells that can contain a maximum of 64 atoms. It enables direct memory access by using a cell index.

We implemented two types of computing accelerators: 64 pipeline processors (PP) for short-range interactions (direct Coulomb and van der Waals) and two long-range units (LRU) for particle-grid interactions (Section IV provides the details). We designed the control general-purpose (CGP) unit to orchestrate the activities of modules, which have the same RISC-V core as GP, the message interface with other modules, and the pipeline controller. We designed the network (NW) interface to transmit and route packets for the 3D-torus network and also to have a grid convolution unit (GCU, with details given in Section IV). The raw bandwidth of the NW connection in each direction is 7.2 GB/s, without consideration of a loss by the protocol. The latency of communication between neighboring nodes was measured to be 200 ns.

We used TSMC CMOS 40-nm technology to implement the MDGRAPE-4A SoC, with its die size at 16.2 × 16.2 mm^2. The clock frequency is 0.6 GHz, except for nonbond pipelines at 0.8 GHz. The nominal peak performance of the SoC is 2.5 TFLOPS when we counted a direct nonbonded force calculation for Coulomb and van der Waals forces as 50 floating-point operations. The power consumption was 85 W per chip for the estimated worst case, whereas the measured power consumption—including the regulation loss, FPGAs, and optical modules—was only 84 W.

We equipped the MDGRAPE-4A system board with eight MDGRAPE-4A SoCs logically connected to a 2 × 2 × 2 cube topology. The FPGA interface on the LSI manages transactions with the FPGA on the system board, which acts as a pathway for the host computer and the TME top-level network. The TME top-level network (TMENW) is an octree network of 64 system boards, as explained in Section IV. We connected the GPGM, CGP, and FPGA interface blocks by using a proprietary on-chip network. Modules particularly relevant to the TME are LRU, GCU, and TMENW through the FPGA interface; Section IV describes these functions and hardware details.

III. Tensor-Structured Multilevel Ewald Summation Method (TME)

A. Theory

The most popular method for computing Coulomb interactions in classical MD simulations is the smooth particle mesh Ewald method (SPME) [6]. As shown in Fig. 2(a), SPME splits the Coulomb potential into short- and long-range parts based on the Ewald method [7] (Ewald splitting) with the Ewald splitting parameter $\alpha > 0$ as

$$\frac{1}{r} = \frac{2}{\sqrt{\pi}} \int_0^\infty \mathrm{e}^{-u^2 r^2} \mathrm{d}u = g_{\alpha,\mathrm{S}}(r) + g_{\alpha,\mathrm{L}}(r), \tag{1}$$

$$g_{\alpha,\mathrm{S}}(r) \coloneqq \frac{2}{\sqrt{\pi}} \int_\alpha^\infty \mathrm{e}^{-u^2 r^2} \mathrm{d}u = \frac{1}{r} \mathrm{erfc}(\alpha r), \tag{2}$$

$$g_{\alpha,\mathrm{L}}(r) \coloneqq \frac{2}{\sqrt{\pi}} \int_0^\alpha \mathrm{e}^{-u^2 r^2} \mathrm{d}u = \frac{1}{r} \mathrm{erf}(\alpha r). \tag{3}$$

The short-range interactions are computed by direct summation. The electrostatic potential from the long-range part is computed based on B-spline interpolation [31]. The interpolation coefficients (grid potentials) are computed in four steps (Fig. 2(b)): (i) Convert atomic coordinates and charges into 3D grid data (grid charges) called charge assignment, (ii) Apply 3D FFT, (iii) Multiply the lattice Green function, (iv) Apply 3D inverse FFT (IFFT). Then, the force acting on an atom is computed by B-spline interpolation with the grid potential, which is called back interpolation. For larger molecular systems—for example, 10^6 atoms—the node-to-node communication required for 3D FFT has a higher cost. To reduce the communication cost, hierarchical methods such as the fast multipole method (FMM) [10] and the multilevel summation method (MSM) [37] were implemented in MD simulations [2,11,12,14,17,27]. In Anton [34], a 3D FFT on 32-bit fixed-point with 512 nodes was conducted in 3.7 μs and 13.3 μs for 32 × 32 × 32 and 64 × 64 × 64 data, respectively [40].

For the *u*-series method implemented in Anton 2 [29,35], the approximation of the Coulomb potential with the summation of Gaussian functions reduced the communication cost, which enabled the evaluation of the 3D convolution by a series of 1D convolutions for each axis (separable convolutions). Conversely, if 3D data fits in a single node/GPU/FPGA and node-to-node communication is unnecessary, 3D FFT is still an attractive choice in terms of the availability of the optimized library for FFT. For example, a single-precision 3D FFT of 16 × 16 × 16 data was conducted in 4 μs in a single FPGA [15].

To incorporate the advantages of different methods to evaluate Coulomb interactions, we developed the TME method (as shown in Fig. 2). Before presenting the detailed expressions, we summarize the basic principles of the TME method, in which the Coulomb potential is split by Ewald splitting. The short-range part is identical to the Ewald method and SPME, which is evaluated by direct summation with the cutoff distance. The long-range part is further split into middle-range parts $g_{\alpha,l}(r), 1 \le l \le L$, and the top-level part $g_{\alpha/2^L,\mathrm{L}}(r)$, as shown in Fig. 2(a)–(d). With the scaling $\alpha/2^L$ of the parameter, the top-level part can be evaluated by SPME with coarser grids by a factor of 2^L for each axis, as shown in Fig. 2(b)–(d).

Because the charge assignment, also called anterpolation in the context of MSM, and the back interpolation are identical between SPME and B-spline MSM [13], these two methods are naturally incorporated in the TME. The conversion from the level l to $l+1$ grid charges is called restriction (Fig. 2(e)), and the conversion from the level $l+1$ to l grid potentials is called prolongation (Fig. 2(f)).

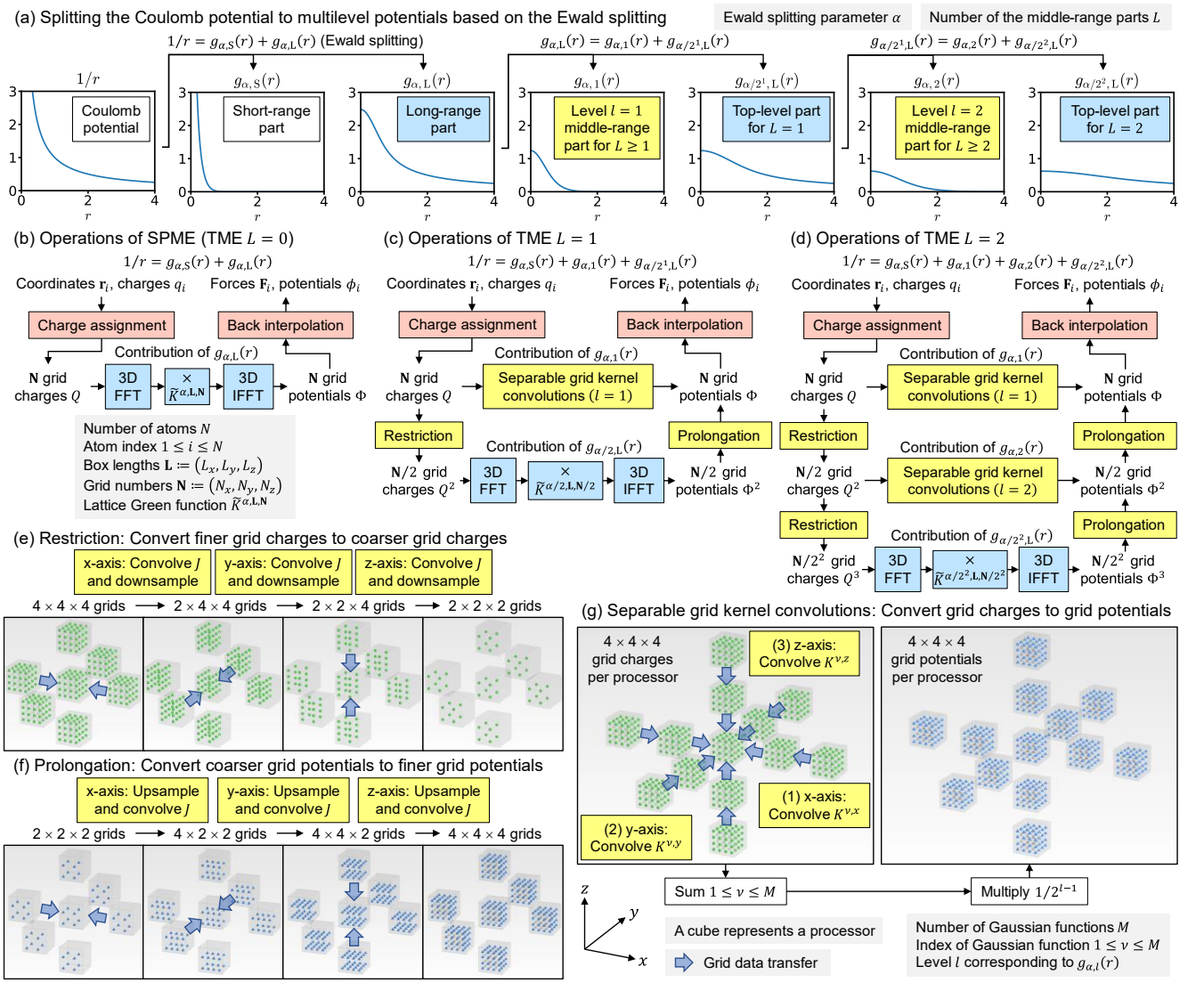

Fig. 2: Basic principles of TME method.

The special property of the B-spline MSM is that the restriction and the prolongation can be expressed by the exact expression because of the two-scale relation of the p-th order central B-spline function $M_p(x) = \sum_{m=-p/2}^{p/2} J_m M_p(2x - m)$, where $J_m \coloneqq 2^{1-p} \binom{p}{p/2+|m|}$ for $|m| \leq p/2$, $J_m \coloneqq 0$ for $|m| > p/2$, and p is an even number [13]. In this study, we assumed that p is an even number. The operations of the restriction are conducted by the axis-wise convolution of J followed by down-sampling, and those of the prolongation are conducted by axis-wise up-sampling followed by the convolution of J. Because J is nonzero in the limited range and the convolutions are performed in the axis-wise direction, the communication cost of the restriction and the prolongation is low. To convert the level l grid charges into level l grid potentials, MSM requires range-limited direct 3D convolution. Except for charge assignment and back interpolation, the most time-consuming part of the MSM is the level 1 grid kernel convolution. To reduce the computational cost of the convolution, Novalbos et al. implemented convolutions based on FFT with GPU [24].

Our Ewald-based level $1 \leq l \leq L$ middle-range part $g_{\alpha,l}(r)$ can be well approximated with a few Gaussian functions, which enables the evaluation of the 3D convolution efficiently by axis-wise convolutions. As shown in Fig. 2(a)–(d), the Coulomb potential in the TME is divided as

$$\frac{1}{r} = \frac{2}{\sqrt{\pi}} \int_0^\infty \mathrm{e}^{-u^2 r^2} \mathrm{d}u = g_{\alpha,\mathrm{S}}(r) + \sum_{l=1}^{L} g_{\alpha,l}(r) + g_{\alpha/2^L,\mathrm{L}}(r), \quad (4)$$

$$g_{\alpha,l}(r) \coloneqq g_{\alpha/2^{l-1},\mathrm{L}}(r) - g_{\alpha/2^l,\mathrm{L}}(r) = \frac{2}{\sqrt{\pi}} \int_{\alpha/2^l}^{\alpha/2^{l-1}} \mathrm{e}^{-u^2 r^2} \mathrm{d}u$$

$$= \frac{2\alpha}{2^{l-1}\sqrt{\pi}} \int_{1/2}^{1} \mathrm{e}^{-\left(\frac{\alpha u r}{2^{l-1}}\right)^2} \mathrm{d}u = \frac{g_{\alpha,1}(r/2^{l-1})}{2^{l-1}}, 1 \leq l \leq L. \quad (5)$$

Next, the level $1 \leq l \leq L$ middle-range part $g_{\alpha,l}(r)$ is approximated by the sum of Gaussian functions as

$$g_{\alpha,l}(r) = \frac{1}{2^{l-1}}\frac{\alpha}{2\sqrt{\pi}}\int_{-1}^{1} \mathrm{e}^{-\left(\frac{-u+3}{4}\frac{\alpha r}{2^{l-1}}\right)^2} \mathrm{d}u$$
$$\approx \frac{1}{2^{l-1}}\sum_{\nu=1}^{M} c_\nu \mathrm{e}^{-\left(\frac{\alpha_\nu r}{2^{l-1}}\right)^2}, 1 \le l \le L. \quad (6)$$

Although selecting the α_ν and c_ν values in Eq. (6) provides many possibilities, in the current implementation, we applied the M point Gauss–Legendre quadrature with nodes $u_{\mathrm{GL},\nu}$ and weights $w_{\mathrm{GL},\nu}, 1 \le \nu \le M$, to the integral in Eq. (6), which leads to

$$\alpha_\nu := \frac{-u_{\mathrm{GL},\nu}+3}{4}\alpha, c_\nu := \frac{\alpha}{2\sqrt{\pi}} w_{\mathrm{GL},\nu}, 1 \le \nu \le M. \quad (7)$$

In Fig. 3(a), $g_{\alpha,l}(r)/g_{\alpha,l}(0)$ and its approximation through Eqs. (6) and (7) are plotted against $\alpha r/2^{l-1}$. The deviation is small even in the single Gaussian approximation ($M = 1$), and we cannot discriminate the difference for $M = 2$ in Fig. 3(a). In Fig. 3(b), we can confirm that the error decreases rapidly with increasing M.

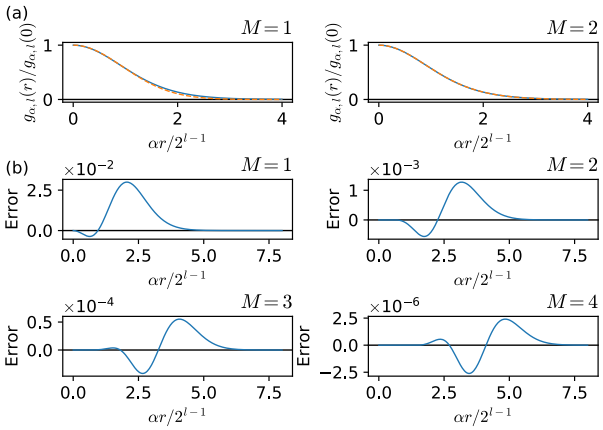

Fig. 3: (a) $g_{\alpha,l}(r)/g_{\alpha,l}(0)$ (solid line) and its approximation using Gaussian functions in Eqs. (6) and (7) (dashed line) are shown for $M = 1, 2$. (b) Approximation error of $g_{\alpha,l}(r)/g_{\alpha,l}(0)$ by Gaussian functions in Eqs. (6) and (7) is shown for $M = 1, 2, 3, 4$. As the second integral representation of Eq. (5) indicates that $g_{\alpha,l}(r)/g_{\alpha,l}(0)$ is function of $\alpha r/2^{l-1}$, each figure in (a) and (b) is invariant by changing $\alpha > 0$ and $1 \le l \le L$.

Then, a Gaussian kernel is approximated by the central B-spline function as

$$\mathrm{e}^{-\alpha^2(x-x')^2} \approx \sum_{m\in\mathbb{Z}}\sum_{m'\in\mathbb{Z}} \mathrm{G}_{m-m'}(\alpha) M_p(x-m) M_p(x'-m'). \quad (8)$$

The coefficients $\mathrm{G}_m(\alpha), m \in \mathbb{Z}$, are determined by $\mathrm{G}(\alpha) = \mathrm{g}(\alpha) * \omega * \omega$, where $\mathrm{g}_m(\alpha) := \mathrm{e}^{-\alpha^2 m^2}, m \in \mathbb{Z}$, ω is the interpolation coefficient of the fundamental spline function by the central B-spline function $M_p(x)$, and $*$ represents the convolution. The analytical expressions of ω are found in p. 39 of Schoenberg [31] for $p = 4$, Schoenberg and Sharma [32] for $p = 6$, and Richards [30] for even $p \ge 4$. The numerical values of $\omega' := \omega * \omega$ are listed in Table I by Hardy et al. [13]. By using Eqs. (6) and (8), periodic box lengths $\mathbf{L} = (L_x, L_y, L_z)$, grid numbers $\mathbf{N} = (N_x, N_y, N_z)$ for the finest grid, the finest grid width $h_j = L_j/N_j, j = x, y, z$, and the notation $M_p(\mathbf{r}/\mathbf{h} - \mathbf{m}) := M_p(x/h_x - m_x) M_p(y/h_y - m_y) M_p(z/h_z - m_z)$, we have

$$g_{\alpha,l}(|\mathbf{r}-\mathbf{r}'|)$$
$$\approx \frac{1}{2^{l-1}}\sum_{\mathbf{m}\in\mathbb{Z}^3}\sum_{\mathbf{m}'\in\mathbb{Z}^3} K_{\mathbf{m}-\mathbf{m}'} M_p\left(\frac{\mathbf{r}}{2^{l-1}\mathbf{h}} - \mathbf{m}\right) M_p\left(\frac{\mathbf{r}'}{2^{l-1}\mathbf{h}} - \mathbf{m}'\right), \quad (9)$$

$$K_{\mathbf{m}} := \sum_{\nu=1}^{M} K_{m_x}^{\nu,x} K_{m_y}^{\nu,y} K_{m_z}^{\nu,z}, \mathbf{m} \in \mathbb{Z}^3, \quad (10)$$

$$K_m^{\nu,j} := c_\nu^{1/3} \mathrm{G}_m(\alpha_\nu h_j), m \in \mathbb{Z}, j = x, y, z, 1 \le \nu \le M. \quad (11)$$

In Eq. (10), the grid kernel K is tensor-structured [18] and the 3D convolution between the grid kernel and grid charges can be evaluated by separable convolutions with 1D grid kernels $K^{\nu,j}, j = x, y, z, 1 \le \nu \le M$, as shown in Fig. 2(g).

In SPME, after applying 3D FFT to the grid charges, the lattice Green function (Eq. (28) of Deserno and Holm [4]) is multiplied. We denote the lattice Green function at grid $\mathbf{n} = (n_x, n_y, n_z), 0 \le n_j < N_j, j = x, y, z$, for SPME with long-range potential $g_{\alpha,\mathrm{L}}(r)$, periodic box lengths $\mathbf{L}$, and grid numbers $\mathbf{N}$ as $\tilde{K}_{\mathbf{n}}^{\alpha,\mathbf{L},\mathbf{N}}$. In the TME, the top-level potential $g_{\alpha/2^L,\mathrm{L}}(r)$ was used instead of $g_{\alpha,\mathrm{L}}(r)$. With this change, $\tilde{K}_{\mathbf{n}}^{\alpha/2^L,\mathbf{L},\mathbf{N}/2^L}$ was used instead of $\tilde{K}_{\mathbf{n}}^{\alpha,\mathbf{L},\mathbf{N}}$ (Fig. 2(b)–(d)).

B. Accuracy

In MSM, 3D convolution between the grid kernel and grid charge is performed by introducing a grid cutoff g_c for the grid kernel. With the grid cutoff, the 1D convolutions for separable convolutions in TME are approximated as $(K^{\nu,j} * a)_m \approx \sum_{|m'|\le g_c} K_{m'}^{\nu,j} a_{m-m'}$. In SPME, the accuracy depends on the Ewald-splitting parameter α, cutoff distance r_c for the short-range interactions, order p of the B-spline interpolation, periodic box lengths $\mathbf{L} = (L_x, L_y, L_z)$, and grid numbers $\mathbf{N} = (N_x, N_y, N_z)$. In addition to these parameters, the TME requires the number M of the Gaussian functions in Eq. (6). Through the construction of the TME, the accuracy is expected to be comparable to the SPME with identical values of $\alpha, r_c, p, \mathbf{L}$, and $\mathbf{N}$ by increasing g_c and M.

To confirm this expectation, we measured the error of the Coulomb forces by the TME and SPME. We computed the Coulomb forces for 32,773 TIP3P water molecules ($N = 98{,}319$) in a periodic cubic box, $L_x = L_y = L_z = 9.97270$ nm. For error evaluation, we implemented TME, SPME, and the Ewald method with C++. We used the Ewald method with $r_c = L_x/2$ to compute the reference double-precision Coulomb forces $\mathbf{F}_i^{\mathrm{ref}}, 1 \le i \le N$, and conducted the lattice summation in the reciprocal space ($\mathbf{k} = 2\pi\mathbf{n}/\mathbf{L}, \mathbf{n} \in \mathbb{Z}^3$) for $|\mathbf{n}| \le n_c$. We determined α and n_c to be the theoretical force error factors $\mathrm{e}^{-\alpha^2 r_c^2}$ in the real space and $\mathrm{e}^{-(\pi n_c/(\alpha L_x))^2}$ in a reciprocal space

[19] less than 10^{-15}, which leads to $\alpha = 1.178612\ \mathrm{nm}^{-1}$ and $n_c = 22$, respectively.

To measure the error of the single-precision Coulomb forces $\mathbf{F}_i, 1 \le i \le N$, of SPME or TME, we used the relative force error $\sqrt{\sum_{i=1}^{N}\left|\mathbf{F}_i - \mathbf{F}_i^{\mathrm{ref}}\right|^2 / \sum_{i=1}^{N}\left|\mathbf{F}_i^{\mathrm{ref}}\right|^2}$. For SPME and TME, we used $p = 6$, $\mathbf{N} = (32,32,32)$, and $r_c = 1,1.25,1.5$ nm, and determined the Ewald splitting parameter α by using $\mathrm{erfc}(\alpha r_c) = 10^{-4}$, which corresponds to the input parameter ewald-rtol = 10^{-4} in GROMACS [1].

Table 1 lists the relative force errors. The errors of the single Gaussian approximation $M = 1$ were larger than those of the others. By comparing $M = 3$ and $M = 4$, we saw that the errors almost converged for $M = 3$. For $r_c = 1.5$ nm, the error of $g_c = 4$ was larger than that of $g_c = 8$ and the error converges for $g_c = 8$ by comparing the error of $g_c = 12$. Thus, $M = 3$ and $g_c = 8$ were sufficient for the convergence of the TME in this example. With $M = 3$ and $g_c = 8$ of TME, the relative force errors 6.18×10^{-4} for $r_c = 1$ nm, 1.40×10^{-4} for $r_c = 1.25$ nm, and 5.99×10^{-5} for $r_c = 1.5$ nm are comparable to those of 5.86×10^{-4}, 1.33×10^{-4}, and 5.92×10^{-5} of SPME, respectively, as shown in Table 1.

Table 1: Relative force errors of SPME and TME ($L = 1$) with respect to the Ewald method. r_c and $\mathrm{erfc}(\alpha r_c) = 10^{-4}$ determined parameter α, which is satisfied by $\alpha r_c \approx 2.751064$.

			Relative force error		
			r_c (nm)		
	g_c	M	1	1.25	1.5
SPME			5.86×10^{-4}	1.33×10^{-4}	5.92×10^{-5}
TME	4	1	1.15×10^{-3}	7.19×10^{-4}	5.28×10^{-4}
		2	6.31×10^{-4}	1.61×10^{-4}	1.25×10^{-4}
		3	6.26×10^{-4}	1.49×10^{-4}	1.18×10^{-4}
		4	6.26×10^{-4}	1.49×10^{-4}	1.18×10^{-4}
	8	1	1.15×10^{-3}	7.20×10^{-4}	5.28×10^{-4}
		2	6.23×10^{-4}	1.49×10^{-4}	7.29×10^{-5}
		3	6.18×10^{-4}	1.40×10^{-4}	5.99×10^{-5}
		4	6.18×10^{-4}	1.39×10^{-4}	5.98×10^{-5}
	12	1	1.15×10^{-3}	7.20×10^{-4}	5.28×10^{-4}
		2	6.23×10^{-4}	1.49×10^{-4}	7.29×10^{-5}
		3	6.18×10^{-4}	1.40×10^{-4}	5.99×10^{-5}
		4	6.18×10^{-4}	1.39×10^{-4}	5.98×10^{-5}

The advantage of mesh-based methods such as SPME and B-spline MSM against FMM for MD simulations is the stable dynamics with a lower computational cost (for example, Fig. 8 of Hardy et al. [13]). To evaluate the stable dynamics of the TME, we conducted NVE simulations of the water system (an identical system used in the force error evaluation). We implemented TME in GROMACS 2018.2 compiled it with double precision to exclude the energy drift caused by the restraint of water molecules. We conducted NVE simulations for 200 ps with a 1 fs time step by using a leap-frog algorithm with verlet-buffer-tolerance = 10^{-10} and restrained the water molecules by using the SETTLE algorithm [21]. For the common parameters between SPME and TME, we used eward-rtol = 10^{-4}, $p = 6$, $\mathbf{N} = (32,32,32)$, and $r_c = 1.25$ nm. For the TME-specific parameters, we used $L = 1$, $g_c = 8$, and $M = 1,2,3$. As shown in Fig. 4, we did not observe a systematic energy drift. Although the total energy was underestimated by approximately 80 kJ/mol for $M = 1$ compared with SPME, we observed that it improved for $M = 2$ and $M = 3$.

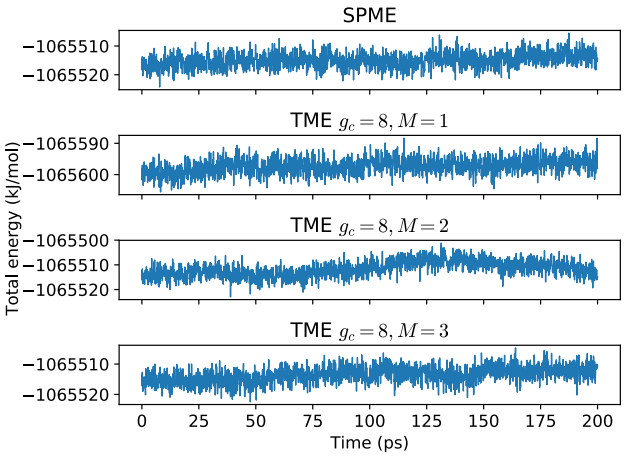

Fig. 4 NVE simulations of SPME and TME ($g_c = 8, M = 1, 2, 3$) conducted with GROMACS 2018.2.

C. Computational Benefit

Here, we discuss the level 1 grid kernel convolution, which is the most time-consuming part (except for the charge assignment and the back interpolation for B-spline MSM and TME). We assumed an identical grid number $N_x = N_y = N_z$ and an identical processor number $P_x = P_y = P_z$ for each axis. The computational costs of B-spline MSM and TME can be estimated as $(2g_c + 1)^3 (N_x/P_x)^3$ and $3(2g_c + 1)(N_x/P_x)^3 M$, respectively. By setting $N_x/P_x = \gamma g_c$, the communication costs of the level 1 grid convolution for the B-spline MSM and TME can be estimated as $(8 + 12\gamma + 6\gamma^2) g_c^3$ and $(2 + 4M)\gamma^2 g_c^3$, respectively. For MDGRAPE-4A, we used $N_x/P_x = 4$ or $N_x/P_x = 8$; also, $g_c = 8$ is sufficient for practical purposes, as discussed in Sec. III.B, which leads to $\gamma = 0.5$ or $\gamma = 1$. For the current implementation of MDGRAPE-4A, we used $M = 4$ for the TME. With these settings, we noticed that the computational and communication costs of the TME reduced with respect to the B-spline MSM. We also noticed that the axis-wise convolutions used in the TME were suitable for the 3D torus network used in MDGRAPE-4A.

Introducing the grid cutoff enables limited-range communications for MSM against axis-direction all-to-all communications for PME. With this difference, by increasing the

number of processors, we expected MSM to be more efficient than PME. In Fig. 10 of Hardy et al. [14], MSM became efficient compared with PME at 512 cores for the ApoA1 system (92 K atoms). Because the computational and communication costs of TME are smaller than those of MSM for the parameters used in MDGRAPE-4A, we can expect better performance against SPME and B-spline MSM for MDGRAPE-4A. With the theoretical background, the consistency with the SPME, and the expected computational benefit, we implemented the TME in MDGRAPE-4A as described in the following sections.

IV. Hardware Support for TME

A. Long-Range Unit

As discussed in the preceding section, the TME method used charge assignment (CA) and back interpolation (BI) based on cardinal B-spline interpolation. We used a long-range unit (LRU) as a dedicated hardware unit for the CA and BI calculations. In the CA mode, the LRU calculates the grid charges $Q_{\mathbf{m}}$ from the atomic coordinates $\mathbf{r}_i$ and charges $q_i, 1 \le i \le N$ as

$$Q_{\mathbf{m}} = \sum_{i=1}^{N} \sum_{\mathbf{n}\in\mathbb{Z}^3} q_i M_p(\mathbf{u}_i - \mathbf{m} - \mathbf{n}\mathbf{N}), \tag{12}$$

in which $\mathbf{N} = (N_x, N_y, N_z)$ are the grid numbers, $\mathbf{u}_i \coloneqq \mathbf{r}_i/\mathbf{h}$ are normalized coordinates with grid spacing $\mathbf{h} = (h_x, h_y, h_z)$, and the sum over $\mathbf{n}$ is taken for a periodic boundary condition. In the BI mode, the electrostatic potential $\phi(\mathbf{r})$ is interpolate from the grid potential $\Phi_{\mathbf{m}}$ as

$$\phi(\mathbf{r}) = \sum_{\mathbf{m}\in\mathbb{Z}^3} \Phi_{\mathbf{m}} M_p\left(\frac{\mathbf{r}}{\mathbf{h}} - \mathbf{m}\right). \tag{13}$$

The following equations show the potential energy E and force $\mathbf{F}_i$ acting on the i-th atom as expressed by $\phi(\mathbf{r})$:

$$E = \frac{1}{2}\sum_{i=1}^{N} q_i \phi(\mathbf{r}_i) = \frac{1}{2}\sum_{i=1}^{N} q_i \phi_i, \tag{14}$$

$$\phi_i \coloneqq \phi(\mathbf{r}_i) = \sum_{\mathbf{m}\in\mathbb{Z}^3} \Phi_{\mathbf{m}} M_p(\mathbf{u}_i - \mathbf{m}), \tag{15}$$

$$\mathbf{F}_i = -q_i \nabla\phi(\mathbf{r}_i) = -\frac{q_i}{\mathbf{h}} \sum_{\mathbf{m}\in\mathbb{Z}^3} \Phi_{\mathbf{m}} M_p'(\mathbf{u}_i - \mathbf{m}), \tag{16}$$

$$M_p'(\mathbf{u}) \coloneqq \begin{pmatrix} M_p'(u_x)M_p(u_y)M_p(u_z) \\ M_p(u_x)M_p'(u_y)M_p(u_z) \\ M_p(u_x)M_p(u_y)M_p'(u_z) \end{pmatrix}^{\mathrm{T}}. \tag{17}$$

These equations are the same as those used in the SPME [6] and B-spline MSM [13]. The following sections detail the hardware implementation. We implemented two LRUs per chip, and these units were responsible for the upper and lower halves along the z-axis of the grid points. We fixed the order of interpolation p at six in the current hardware. Each atom interacts with $6^3 = 216$ grid points for both CA and BI calculations.

Fig. 5 shows a block diagram of the LRU. We loaded the atom coordinates and charges into the "atom register file," which can store those for eight cells, with a maximum of 512 atoms. In both the CA and BI modes, we evaluated the piecewise polynomial M_p of the x, y, and z dimensions sequentially. We used the recursion formula for B-spline and evaluated M_p and M_p' on six grid points simultaneously by using a 12-stage pipeline. We performed calculations in a fixed-point format with a 24-bit fractional part (maximum of $1\text{-}2^{-24}$) and optimized their widths according to the maximum value for each point. To allow the next interpolation calculation in the background, we temporally stored the results in double-buffered registers.

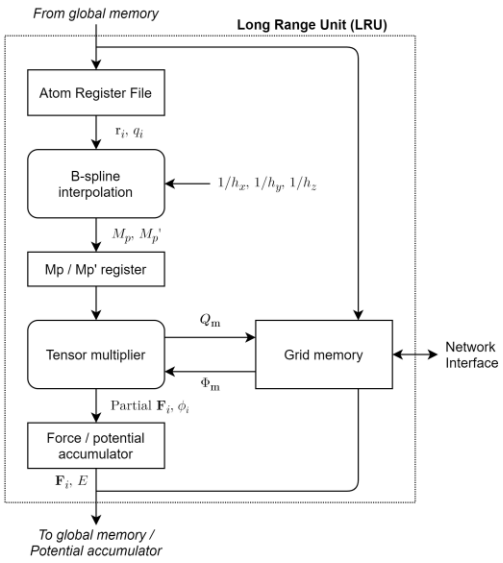

Fig. 5: Block diagram of LRU.

The tensor multiplier unit starts the calculations of the tensor products and their convolutions after all the x, y, and z polynomial evaluations have been completed. In the CA mode, we evaluated the tensor products according to Eq. (12) at a 32-bit fixed point for six grids in parallel; thus, a maximum of 36 cycles were required per atom. The number of cycles depended on the z coordinate of an atom, which determined the loads for the two LRUs. The results accumulated in the "grid memory" that can store grid charges or potentials for 16 × 16 × 16 local grids with four sleeve grids for each of the six directions. Because an atom can protrude from a cell within a certain range, the number of sleeve grids should be extended by 1. Thus, the total grid memory size was 24^3 = 13.5 K 32-bit words distributed in two LRUs. We connected the grid memory to the network interface to exchange the sleeve grid information and obtain the final grid charges. The local grid sizes supported by the LRU were 4^3, 8^3, 12^3, and 16^3. which corresponded to global grid sizes varying from 32^3 to 128^3.

In the BI mode, the calculation started after the grid potential was set to grid memory. It reevaluated M_p and M_p' again, and then performed the convolutions of grid potentials $\Phi_{\mathbf{m}}$ with M_p and M_p' in Eqs. (15) and (16), respectively. The tensor multiplier unit has two convolution units for M_p and M_p', and each unit calculated a one-dimensional convolution along the x-axis every cycle in parallel. Convolution along the y- and z-axes was performed sequentially. Thus, each atom required a maximum of

36 cycles. The accumulation of force was performed in a 32-bit fixed point with a tunable binary point. The potential for each atom was calculated with a 32-bit fixed point, and the total potential accumulated at a 64-bit fixed point. We evaluated the potential and force for each atom in parallel and accumulated in the potential and force accumulators. The size of the force accumulator was the same as that of the atom register—that is, eight cells or 512 atoms. The forces were sent to global memories that have an accumulation function to obtain the final forces.

B. Grid Convolution Unit

As a newly designed module inside the network interface, the GCU performs a range-limited convolution on a grid, which is essential in the middle-range (level $1 \le l \le L$) calculation in the TME. Because we adopted a convolution kernel that is separable along the three-dimensional axis, we implemented sequential convolution along the x-, y-, and z-axes. Such an operation fits well with the 3D-torus topology of the network but requires frequent repetition of communication and computation. To reduce data movement and accelerate the computation, we designed the GCU as a dedicated unit embedded inside the network unit.

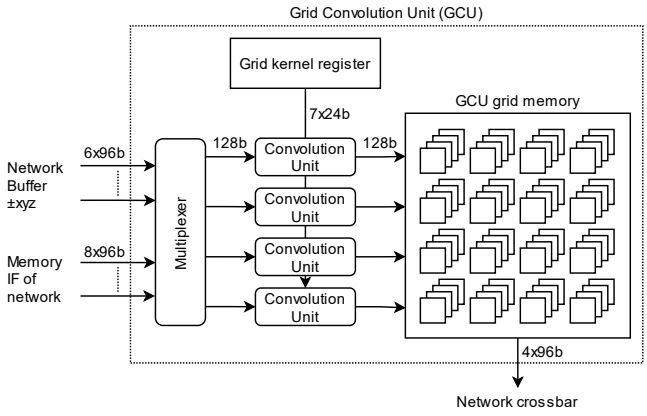

Fig. 6: Block diagram of GCU.

Fig. 6 shows a block diagram of the GCU. The GCU manipulates a grid block of 4 × 4 × 4 mesh points as a basic data unit; it receives and sends the blocks directly from/to the network buffer and global memories. It convolves the incoming grid block to the local blocks; the incoming block $h^{\nu}_{\mathbf{m}}$ with grid origin $\mathbf{m}$ is convolved with the 1D grid kernel $K^{\nu,j}_{m}$ ($K^{\nu,j}_{m} = 0$ for $|m| > g_c$), and the results are accumulated to the local grid points $g^{\nu}_{\mathbf{n}}$. The following equation provides the case for convolution along the x-axis:

$$g^{\nu}_{\mathbf{n}} \leftarrow g^{\nu}_{\mathbf{n}} + \sum_{i=0}^{3} h^{\nu}_{m_x+i,m_y+j,m_z+k} K^{\nu,x}_{n_x-m_x-i}\ , \qquad (18)$$

where $0 \le j \le 3, 0 \le k \le 3, n_y = m_y + j$, and $n_z = m_z + k$. The 1D grid kernels $K^{\nu,j}_{m}$ were precalculated and stored in the dedicated register. The GCU has four convolution units, each of which calculates one-dimensional convolution at four grids in parallel; thus, 16 grid points can be evaluated simultaneously. Because the data feed rate (96 bits, three words) from a single network buffer limits the calculation, four input blocks can be calculated simultaneously. In principle, it can handle five input blocks considering the data rate; however, we limited it to four for simplicity, and the maximum rate decreased to 12 grid points per cycle. We controlled the timing to avoid read and write collisions between the four inputs. The precision of the grid data was a 32-bit fixed point, and that of the convolution factors was a 24-bit fixed point. The arbitrary binary point for the grid data can be shifted by a specified amount in the convolution to avoid overflow.

The results accumulated in the GCU grid memory, which consisted of separate memory modules with a 4 × 4 × 4 configuration. Each of these 64 blocks consisted of a 32-bit width, 128-words one-read one-write memory. The depth stored the different blocks, and each grid point of a block was stored in a memory module corresponding to the grid coordinates. Enabling fast parallel convolution in multiple axes requires such a memory configuration. After completing all convolutions on the x-axis, the GCU should perform convolutions on the y- and z-axes. Simultaneous reads and writes along different axes are essential for efficient parallel operation. The GCU grid memory allowed reads and writes of 4 × 4 words on the x-y, y-z, and z-x planes according to the operation modes. It can send the results directly to the network and to the GM unit through the network crossbar.

A grid cutoff g_c of convolution can be 8 or 12, which affects the number of hops of the GCU packets on the NW. The GCU operation supported two grid sizes: a single 4 × 4 × 4 grid block and eight grid blocks per node. Therefore, the hardware accelerated TME on MDGRAPE-4A works with grid sizes of 32^3 and 64^3.

C. Top-level convolution using FPGA

As discussed in the theory section, we evaluated the top-level (level $L + 1$) grid potentials based on the SPME, which requires 3D-FFT. We implemented the hardware for the 16 × 16 × 16 3D-FFT-based convolution in the FPGA on the MDGRAPE-4A system. We used the dedicated octree network TME top-level network (TMENW) to gather data from all MDGRAPE-4A SoCs. Fig. 7 shows A block diagram of the TMENW. Inside the MDGRAPE-4A board, the data were transferred via the IO FPGA (Xilinx XC7K160), which was also used as the general I/O port for initialization and data retrieval, and the data are collected in the control FPGA (Intel 10AX048). The control FPGA sends them to a leaf FPGA (Intel 10AX057).

A leaf FPGA serves only as a router in the tree network, with each connected to eight boards and 64 SoCs. Better minimized latency involves connecting SoCs directly to the leaf FPGA; however, we selected the current implementation to avoid possible problems with the SoC–FPGA connection. We connected eight leaf FPGAs to the root FPGA (Intel 10AX115), which gathered all grid data and performed calculations. In all

connections between the control and leaf FPGAs, as well as the leaf and root FPGAs, we used optical connections with four lanes of 10.3125-Gbps full duplex (Samtec FireFly™). The data rate was 40 Gbps after 64B66B decoding. We also developed a proprietary network with an error-recovery function.

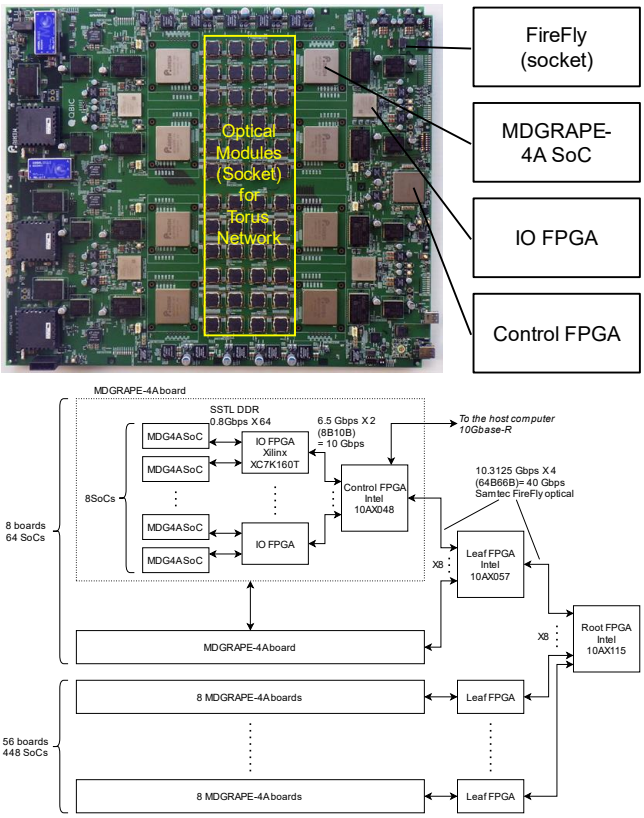

Fig. 7: Architecture of TME top-level network. Top: MDGRAPE-4A board. Bottom: Block diagram of TMENW.

As soon as all top-level grid charges $Q_{\mathbf{m}}^{L+1}$ of 16^3 = 4,096 points arrived at the root FPGA, the FPGA started the calculation of the top-level grid potentials $\Phi_{\mathbf{m}}^{L+1}$. The calculation based on the SPME consists of three steps:

1. Calculate the discrete Fourier transform $\widehat{Q^{L+1}} = \mathcal{F}Q^{L+1}$.
2. Multiply the lattice Green function to obtain the top-level grid potentials in k-space, $\widehat{\Phi^{L+1}}_{\mathbf{n}} = \tilde{K}_{\mathbf{n}}^{\alpha/2^L,\mathbf{L},\mathbf{N}/2^L} \widehat{Q^{L+1}}_{\mathbf{n}}$.
3. Calculate the inverse discrete Fourier transform $\Phi^{L+1} = \mathcal{F}^{-1}\widehat{\Phi^{L+1}}$ (refer to Fig. 2(c) and (d)).

In the calculation, we used the single-precision floating-point format. Further, we performed fixed-point to floating-point conversion and its reverse on receiving and sending the data with the leaf FPGA. We implemented all operations by using native floating-point digital signal processors (DSPs) equipped with FPGA, except for simple calculations such as negation.

Fig. 8 shows a block diagram of the top-level grid-potential solver. The basic calculation unit was CFFT16, which performed a flash calculation on complex radix-4 FFT of 16 points by using 160 DSPs (144 floating-point (FP) adders/subtractors and 16 FP multiply-adders). We implemented four CFFT16 units and had a 64-point complex FFT perform every cycle. The "post/preprocess" unit—combined with each CFFT16 unit—converted complex FFT results to real FFT ones and prepared inputs for inverse FFT to obtain real-space values. The post/preprocess units also performed multiplication of the lattice Green function, which was stored in the dedicated memory. Because wave numbers 0 and 8 (= 16/2) should be treated in a special manner in a real FFT, we designed a separate "post/preprocess 08" unit. The number of DSPs used for the post/preprocess unit and its special value for 08 were 32 and 60, respectively.

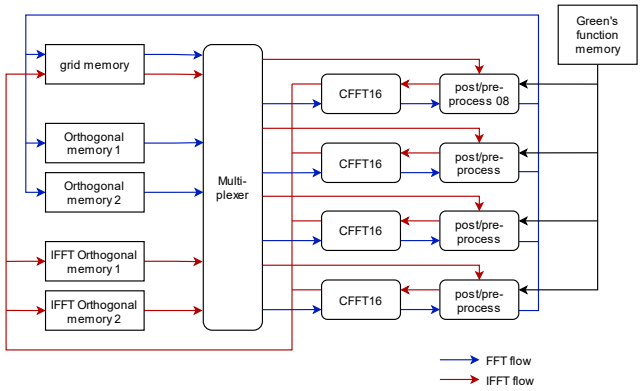

Fig. 8: Hardware for 3D-FFT based convolution.

The 3D FFT consisted of consecutive 1D FFTs in the x-, y-, and z-axes. Similar to the case of the GCU grid memory, the parallelization of FFTs required a transpose of memory. We designed the "orthogonal memory" with parallel read and write on the orthogonal axis for the purpose. Suppose that 1D FFTs are performed in the sequence of x, y, and z. In the first transformation along the x-axis, all data for x at specified y and z points should be read simultaneously. In addition, parallel 1D FFT units require more parallel reads; parallelization can be performed arbitrarily on y and z, but for simplicity, it was applied only on the y-axis. In this case, a parallel read/write on the x-y plane was required. Our implementation had four complex 1D FFT units, which consumed eight parallel reads on y, corresponding to the half x-y plane. Similarly, the next transform along the y-axis required simultaneous reads on the y-z or y-x planes. For this, we implemented an orthogonal memory with x-y write and y-z read. The same design can be used for the transformation on the z-axis (y-z write and z-x read). Inverse FFT required an opposite axis rotation. After performing all calculations, the level $L+1$ grid potentials were sent back to all SoCs via the same route in the opposite direction as receiving the level $L+1$ grid charges.

The system operated at 156.25 MHz with all calculations finishing in 330 cycles at 2.112 µs. Using larger devices and faster clock speeds can improve the speed. The current design

consumes 796 DSPs in the FPGA. Using a larger FPGA, such as Intel Stratix 10, can obtain a performance gain of at least four. In addition, because the current clock speed is determined to avoid asynchronous transfers with the network modules, asynchronization can improve the clock frequency.

V. Performance Measurements

A. Time chart of single-step MD calculation

We measured the precise timing of the transition of CGP status at 10-ns precision by using software to export the status to the output port of the SoC. Fig. 9 shows the time chart of the activity of each component in the SoC during a single time-step MD calculation. The target system of a protein molecule (480 residues, 7,775 atoms), ions, and solvent water molecules consisted of 80,540 atoms in total in a rectangular box with side lengths 9.7 nm, 8.3 nm, and 10.6 nm for each axis. For the TME, we used the grid numbers $\mathbf{N} = (32,32,32)$, the number of the middle-range part $L = 1$, the short-range cutoff $r_c = 1.2$ nm, the grid cutoff $g_c = 8$, and the number of Gaussian functions $M = 4$. The indicated finished time reflects the time when the CGP confirmed the arrival of the "end" message; therefore, the actual calculation could finish earlier than the time.

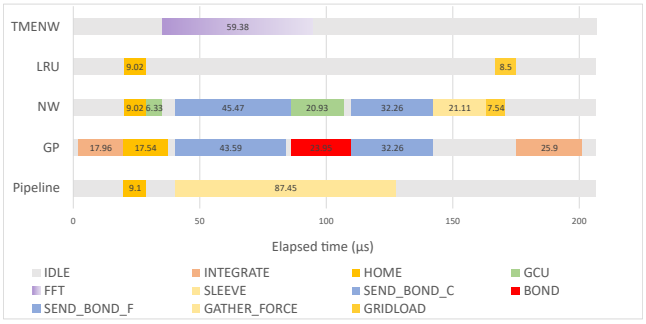

Fig. 9: Time chart of components in SoC during single-time-step MD calculation.

We implemented the Velocity-Verlet algorithm to integrate Newton's equation of motion in three phases. The first phase updated velocities to half time step by the force calculated in the previous MD step and updated the coordinates to the next time step according to the updated velocities. The second phase served as the force calculation phase, which used the updated coordinates to evaluate all particle interactions. The third phase used the updated forces to update the velocities to half the time step again. Fig. 9 shows the first and third phases performed by the GP processors as INTEGRATE. They also treated holonomic constraints on the molecules.

The force evaluation phase calculated three independent terms: direct nonbonded (a short-range part of Coulomb and van der Waals) forces, bonded forces, and a long-range part of Coulomb forces. The direct nonbonded forces and potentials were calculated by using the dedicated unit "nonbond pipeline." The coordinates and forces were exchanged with neighboring nodes by the NW. The GP processors evaluated the bonded terms, which also required NW activities to transfer the coordinates and forces among nodes. The calculations of the long-range part of the Coulomb forces involve four different activities: LRU, NW activities for neighboring grid transfer, NW activities for GCU calculations, and TMENW activities including top-level convolution. All calculations shared NW and GM resources and affected each other, whereas the calculations can overlap.

In the current software for production runs, some parts of the calculations used resources exclusively to avoid the system being stuck in long-run simulations. Particularly, GCU operations must be exclusive to other NW activities. Although scope is still available to optimize the performance by increasing the concurrency, it requires 206 μs to complete the single MD time step. The current performance of the system is approximately 1 μs/day when using a 2.5 fs time step. The performance of the same target using GROMACS 2020.5 on a machine equipped with a single AMD EPYC 7502P 32-core processor and two NVIDIA GeForce RTX 3090 boards was 73 ns/day. It requires 2.96 ms to complete the single MD time step on average.

B. Elapsed time for long-range part of Coulomb interaction

As explained in the theory section, the evaluation of the long-range part of Coulomb interactions by TME consisted of six calculation steps, which are mapped to the hardware as follows:

1. The first step involves the CA on the grid points, which involve the LRU and NW.
2. The second step involves the restriction to obtain the grid charge with a coarser grid, which involves the GCU operation.
3. The third step involves grid convolution to obtain the grid potential at the middle level.
4. The fourth step converts the grid charge to the grid potential at the top level, which involves the TMENW.
5. The fifth step involves the prolongation of obtaining the grid potential on the original finest grid from the coarse grid potential.
6. The final step involves BI to obtain the force and potential of an atom.

Because the GCU operation must be synchronized between nodes, the apparent duration of the GCU activities includes the waiting for data from the other nodes. This arises from the load imbalance because of fluctuations in the number and type of atoms (that is, protein or water) distributed on each node.

Fig. 10 shows a more detailed time chart of the GCU event phase. The actual GCU operation time was rather short, which was consistent with the results estimated by logic simulations. The duration of the prolongation also includes the elapsed time of the CGP code to prepare the input for the prolongation and to accumulate the results of the prolongation onto the grid kernel convolutions.

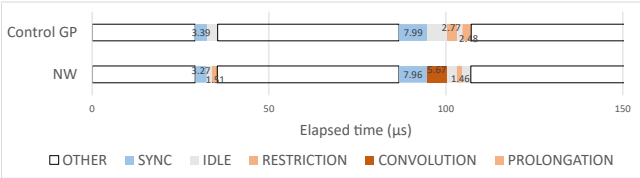

Fig. 10: Detailed time chart of GCU phases. First phase includes restriction followed by TMENW initiation. Second phase includes convolution and prolongation as well as time to run CGP software code to prepare for prolongation and accumulate results with those of convolution.

In summary, the LRU operations (CA and BI) required approximately 10 μs. For the GCU operations, the restriction, prolongation, and convolution of level 1 required 1.5 μs, 1.5 μs, and 6 μs, respectively. We measured the roundtrip time required to obtain the top-level grid potentials by the TMENW to be less than 20 μs. The total evaluation time for the long-range part of the Coulomb forces and potentials was approximately 50 μs.

C. Overlapping calculations

Using different units in the system can parallelize many parts of the calculations. Although the GCU operation is exclusive, the lower-level grid convolution and the higher-level TMENW calculation that involves data transfer on the octree network and the top-level convolution on the FPGA can be parallelized. In addition, the evaluation of non-bonded short-range interactions and bonded interactions can be performed in parallel with the calculation of the long-range term. Thus, the additional elapsed time by incorporating a long-range part of the Coulomb interaction was much less than the total elapsed time of approximately 50 μs. The elapsed time to complete a single MD step without a long-range part of the Coulomb interaction was 196 μs. Compared with the time chart of the simulations without it, the performance loss was approximately 10 μs, which is 5% of the single time step calculation.

D. Performance Comparison

Because MDGRAPE-4A has a fixed system size with the range of the simulation system sizes restricted, we estimated only the approximate performance for the target system size of 50k–100k atoms, which covered typical applications of MD. Table 2 compares several other computer systems. MDGRAPE-4A reaches at least three times faster than the best performance of any other commodity clusters, but still lower than that of Anton 1, which is much less than that of Anton 2.

The unexpected degradation of execution efficiency of the software codes running on GP cores was the root cause of the gap between MDGRAPE-4A and the Anton family, mainly because of memory and register-access latencies. However, when comparing the elapsed time to evaluate the long-range part of Coulomb interactions (the topic of this paper), the gap is relatively small. We implemented a prototype software for 3D FFT calculation on the previous MDGRAPE-4 system and realized the latency of the software implementation by repetition of 1D FFT and transposition on the torus network would be hundreds of microseconds; as a result, we had to reconsider the hardware acceleration of the long-range method. We used hardware to accelerate the TME. The actual TME performance is slightly lower than expected, mainly because of the latency of data transfer over TMENW for the top-level convolution and the software management of hierarchical TME calculations.

In summary, the achieved performance of the long-range part is one order of magnitude faster than that of commodity clusters and comparable to Anton 1 that implemented the 3D FFT based Ewald method. Anton 2 implemented the *u*-series method to eliminate 3D FFT by using separable real-space convolutions similar to the grid convolution of the TME and had achieved further performance. The computationally dominant phase of the TME is the lowest-level kernel convolution, which requires 6 μs for a 32 × 32 × 32 grid on MDGRAPE-4A. It can be shortened by migrating to advanced process technology and higher performance networks and by careful reconsideration of hardware balance.

Table 2: Performance comparison of MDGRAPE-4A and the other computer systems for target systems with 50k-100k atoms, with values obtained from literature [28,35]. The elapsed time to evaluate long-range part is estimated. For GROMACS, the strong scaling is limited by the long-range part that requires 3D-FFTs and saturate at 64 CPU nodes and 64 GPU boards (32 nodes). We estimated that most time is consumed for long-range part at the limit. For Anton family, they calculate long range part at every other step, and we estimated half of the elapsed time for two steps is close to that for long-range part [5].

Computer system	Long-range method	Performance (μs/day)	Average time/step (μs)	Long-range part (μs)
CPU cluster (64 nodes)	SPME	0.25	800	~500
GPU cluster (64 GPUs)	SPME	0.3	700	~500
MDGRAPE-4A (512 nodes)	TME	1.0	200	~50
Anton 1 (512 nodes)	*k*-GSE	10	20	~20
Anton 2 (512 nodes)	*u*-series	70	3	~3

VI. Discussion

A. Estimated performance for larger grid size (64 × 64 × 64)

A target system larger than a cube of 10 nm requires the number of grids increased to maintain the error levels. Although we

designed the hardware to allow a 64 × 64 × 64 grid-sized TME with $L = 2$, the current software does not support it. The tasks of the TMENW were the same. The additional time to process the TME on the larger grid can be estimated as the sum of the lowest-level GCU operations and the increase in the LRU operations. The time for GCU operations is eight times larger than 32 × 32 × 32 operations theoretically and is estimated to be 72 µs in total. RTL simulations also supported this estimation. The calculation time of the LRU is proportional to the number of atoms and does not depend on the grid point number. The number of atoms increases in proportion to the volume of the simulation box. Conversely, the time required to accumulate grid charges and potentials on remote nodes would also be eight times larger. The additional cost for grid data transfer was estimated to be approximately 10 µs for both CA and BI. The estimated total time is approximately 150 µs for the calculation of the long-range term that uses a larger grid TME, which is the most time-consuming part of computation but still comparable with other parts.

B. Further accelerations of MD simulations

Although MDGRAPE-4A outperforms commodity clusters or GPU accelerators, Anton 2—another MD special computer—has already achieved a performance one order of magnitude higher than MDGRAPE-4A. We plan to develop a next-generation system with further improvements and specializations. The current system requires 512 SoCs to satisfy the demands of performance and memory capacity. The next system will become more compact and can be scaled down to eight SoCs that uses an advanced semiconductor process.

Regarding the overall MD performance—especially for the bonded force term and integration—GP performance is a major bottleneck. We implemented dedicated special functions for bonded-force calculations for the current GP cores. However, our performance improvements were limited, and the GP achieved a lower execution efficiency than expected from the FLOPS counts. We plan to design a new programmable unit specialized for bonded-force calculations and integrations to increase the effective performance of the actual code.

We expect the acceleration of the long-range term of the Coulomb interaction to be more difficult because the network bandwidth and latency are not easy to scale up by migrating the manufacturing process. The performance bottlenecks of the current implementation remain in the calculation flow controls by the CGP software processes. The GCU automates axis transpositions in convolution; however, the management of hierarchical processes should be more integrated in hardware to reduce the latency by a more sophisticated event manager and specialized units. There are many ways to improve performance by parallelization in the TME. The performance and parallelization of the LRU were not optimized in the current system; for example, the load-balancing scheme in parallel LRUs should be redesigned in the next system. The performance and parallelization of the GCU also should increase. For the top-level convolution by TMENW, the latency should decrease by the direct communication between SoCs and FPGAs, and the speed of the convolution in an FPGA can also improve, as discussed in Section IV.

Recent FPGA technology has opened the way to implement all MD computations on large FPGAs. FPGA-MD [39] was implemented to single Intel Stratix 10 and achieved a simulation throughput of 0.63 µs/day on 23.5 K DHFR dataset that uses 16 × 16 × 16 3D FFT with the time consumption of PME at 500 µs. The 3D FFT can be parallelized on an FPGA cluster. However, the performance of 64 × 64 × 64 FFT by using a FPGA cluster would be saturated at 9.55 µs at 512 nodes according to the performance model that validated measurement up to an eight-nodes cluster [36]. Accelerating MD simulations of typical applications and the implementation of long-range methods is also key to achieving strong scalability of MD on an FPGA cluster. The TME algorithm remains an attractive choice for the next generation because of its simplicity of computation and communication schemes.

VII. Conclusion

By integrating the SPME [6] and B-spline MSM [13], we developed a TME for long-range electrostatic computations. It has preferred features to be implemented in a special-purpose computer, as compared with the SPME using 3D FFTs and the B-spline MSM using direct 3D convolutions. We implemented the combined networks for optimal communications: the 3D-torus network for tensorized convolutions, and the octree network for top-level convolution that uses 3D FFTs.

The TME uses major computational resources for the CA/BI and 1D grid-kernel separable convolutions. To accelerate them, we implemented two types of dedicated computing modules: LRU for CA/BI and GCU for range-limited separable convolutions. The GCU can also perform the restriction and the prolongation. The elapsed time of the long-range part of the Coulomb interaction by the TME on MDGRAPE-4A was approximately 50 µs in total, which mostly overlapped with short-range and bonded force evaluations. The additional cost of incorporating long-range electrostatics is approximately 10 µs, which corresponds to a 5% performance loss compared with the computation time without a long-range term. The performance of MDGRAPE-4A reached 1 µs per day for typical biomolecular systems with 100,000 atoms. The TME will scale better than the SPME on computer systems interconnected by a 3D-torus network as MDGRAPE-4A and has an advantage for future accelerator implementations.

ACKNOWLEDGMENTS

We thank Rio Yokota for valuable discussions. We also wish to thank Taisho Pharmaceutical Co., Ltd. for their collaboration in evaluating the design of the SoC. This work was supported by JSPS KAKENHI Grant Numbers JPD2010301 (G.M., Y.O., I.O., and M.T.), JP19H01107 (G.M., Y.M.K., T.S.K., Y.O., and M.T.) and AMED under Grant Number JP16nk0101102 (M.T.). We would like to thank Editage (www.editage.com) for English language editing.

REFERENCES

[1] Mark James Abraham, Teemu Murtola, Roland Schulz, Szilárd Páll, Jeremy C. Smith, Berk Hess, and Erik Lindah. 2015. GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. *SoftwareX* 1–2, (September 2015), 19–25. DOI:https://doi.org/10.1016/j.softx.2015.06.001

[2] Yoshimichi Andoh, Noriyuki Yoshii, Kazushi Fujimoto, Keisuke Mizutani, Hidekazu Kojima, Atsushi Yamada, Susumu Okazaki, Kazutomo Kawaguchi, Hidemi Nagao, Kensuke Iwahashi, Fumiyasu Mizutani, Kazuo Minami, Shin Ichi Ichikawa, Hidemi Komatsu, Shigeru Ishizuki, Yasuhiro Takeda, and Masao Fukushima. 2013. MODYLAS: A highly parallelized general-purpose molecular dynamics simulation program for large-scale systems with long-range forces calculated by fast multipole method (FMM) and highly scalable fine-grained new parallel processing algorithms. *Journal of Chemical Theory and Computation* 9, 7 (July 2013), 3201–3209. DOI:https://doi.org/10.1021/ct400203a

[3] A.F. Bakker and C. Bruin. 1988. Design and Implementation of the Delft Molecular-Dynamics Processor. In *Special Purpose Computers*. Elsevier, 183–232. DOI:https://doi.org/10.1016/b978-0-12-049260-2.50010-6

[4] Markus Deserno and Christian Holm. 1998. How to mesh up Ewald sums. I. A theoretical and numerical comparison of various particle mesh routines. *The Journal of Chemical Physics* 109, 18 (November 1998), 7678–7693. DOI:https://doi.org/10.1063/1.477414

[5] R O Dror, J P Grossman, K M Mackenzie, B Towles, E Chow, J K Salmon, C Young, J A Bank, B Batson, M M Deneroff, J S Kuskin, R H Larson, M A Moraes, and D E Shaw. 2010. Exploiting 162-Nanosecond End-to-End Communication Latency on Anton. In *SC '10: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, 1–12. DOI:https://doi.org/10.1109/SC.2010.23

[6] Ulrich Essmann, Lalith Perera, Max L. Berkowitz, Tom Darden, Hsing Lee, and Lee G. Pedersen. 1995. A smooth particle mesh Ewald method. *The Journal of Chemical Physics* 103, 19 (November 1995), 8577–8593. DOI:https://doi.org/10.1063/1.470117

[7] P. P. Ewald. 1921. Die Berechnung optischer und elektrostatischer Gitterpotentiale. *Annalen der Physik* 369, 3 (January 1921), 253–287. DOI:https://doi.org/10.1002/andp.19213690304

[8] T Fukushige, J Makino, T Ito, S Okumura, T Ebisuzaki, and D Sugimoto. 1993. WINE-1: special-purpose computer for N-body simulations with a periodic boundary condition. *Publications of the Astronomical Society of Japan* 45, 3 (1993), 361–375.

[9] Toshiyuki Fukushige, Makoto Taiji, Junichiro Makino, Toshikazu Ebisuzaki, and Daiichiro Sugimoto. 1996. A Highly Parallelized Special-Purpose Computer for Many-Body Simulations with an Arbitrary Central Force: MD-GRAPE. *The Astrophysical Journal* 468, (1996), 51. DOI:https://doi.org/10.1086/177668

[10] L. Greengard and V. Rokhlin. 1987. A fast algorithm for particle simulations. *Journal of Computational Physics* 73, 2 (December 1987), 325–348. DOI:https://doi.org/10.1016/0021-9991(87)90140-9

[11] Yongfeng Gu and Martin C. Herbordt. 2007. FPGA-based multigrid computation for molecular dynamics simulations. In *Proceedings 2007 IEEE Symposium on Field-Programme Custom Computing Machines, FCCM 2007*, IEEE Computer Society, 117–126. DOI:https://doi.org/10.1109/FCCM.2007.42

[12] David J. Hardy, John E. Stone, and Klaus Schulten. 2009. Multilevel summation of electrostatic potentials using graphics processing units. *Parallel Computing* 35, 3 (March 2009), 164–177. DOI:https://doi.org/10.1016/j.parco.2008.12.005

[13] David J. Hardy, Matthew A. Wolff, Jianlin Xia, Klaus Schulten, and Robert D. Skeel. 2016. Multilevel summation with B-spline interpolation for pairwise interactions in molecular dynamics simulations. *The Journal of Chemical Physics* 144, 11 (March 2016), 114112. DOI:https://doi.org/10.1063/1.4943868

[14] David J. Hardy, Zhe Wu, James C. Phillips, John E. Stone, Robert D. Skeel, and Klaus Schulten. 2015. Multilevel Summation Method for Electrostatic Force Evaluation. *Journal of Chemical Theory and Computation* 11, 2 (February 2015), 766–779. DOI:https://doi.org/10.1021/ct5009075

[15] Benjamin Humphries, Hansen Zhang, Jiayi Sheng, Raphael Landaverde, and Martin C. Herbordt. 2014. 3D FFTs on a single FPGA. In *Proceedings - 2014 IEEE 22nd International Symposium on Field-Programmable Custom Computing Machines, FCCM 2014*, Institute of Electrical and Electronics Engineers Inc., 68–71. DOI:https://doi.org/10.1109/FCCM.2014.28

[16] Tomoyoshi Ito, Toshiyuki Fukushige, Junichiro Makino, Toshikazu Ebisuzaki, Sachiko K Okumura, Daiichiro Sugimoto, Hiroo Miyagawa, and Kunihiro Kitamura. 1994. A special-purpose computer for molecular dynamics: GRAPE-2A. *Proteins: Structure, Function, and Bioinformatics* 20, 2 (1994), 139–148. DOI:https://doi.org/10.1002/prot.340200204

[17] Jesús A. Izaguirre, Scott S. Hampton, and Thierry Matthey. 2005. Parallel multigrid summation for the N-body problem. *Journal of Parallel and Distributed Computing* 65, 8 (August 2005), 949–962. DOI:https://doi.org/10.1016/j.jpdc.2005.03.006

[18] Boris N. Khoromskij. 2012. Tensors-structured numerical methods in scientific computing: Survey on recent advances. *Chemometrics and Intelligent Laboratory Systems 110*, 1–19. DOI:https://doi.org/10.1016/j.chemolab.2011.09.001

[19] Jiri Kolafa and John W. Perram. 1992. Cutoff Errors in the Ewald Summation Formulae for Point Charge Systems. *Molecular Simulation* 9, 5 (1992), 351–368. DOI:https://doi.org/10.1080/08927029208049126

[20] Junichiro Makino and Makoto Taiji. 1998. *Scientific Simulations with Special-Purpose Computers: The Grape Systems*. Wiley, Chichester.

[21] Shuichi Miyamoto and Peter A Kollman. 1992. Settle: An analytical version of the SHAKE and RATTLE algorithm for rigid water models. *Journal of Computational Chemistry* 13, 8 (October 1992), 952–962. DOI:https://doi.org/https://doi.org/10.1002/jcc.540130805

[22] Tetsu Narumi, Yousuke Ohno, Noriaki Okimoto, Takahiro Koishi, Atsushi Suenaga, Noriyuki Futatsugi, Ryoko Yanai, Ryutaro Himeno, Shigenori Fujikawa, Makoto Taiji, and Mitsuru Ikei. 2006. A 55 TFLOPS simulation of amyloid-forming peptides from yeast prion Sup35 with the special-purpose computer system MDGRAPE-3. In *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing, SC'06*, ACM Press, New York, New York, USA, 49. DOI:https://doi.org/10.1145/1188455.1188506

[23] Tetsu Narumi, Ryutaro Susukita, Toshikazu Ebisuzaki, Geoffrey McNiven, and Bruce Elmegreen. 1999. Molecular Dynamics Machine: Special-Purpose Computer for Molecular Dynamics Simulations. *Molecular Simulation* 21, 5–6 (1999), 401–415. DOI:https://doi.org/10.1080/08927029908022078

[24] Marcos Novalbos, Jaime González, Miguel A. Otaduy, Roberto Martinez-Benito, and Alberto Sanchez. 2014. Scalable on-board multi-GPU simulation of long-range molecular dynamics. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Springer Verlag, 752–763. DOI:https://doi.org/10.1007/978-3-319-09873-9_63

[25] Itta Ohmura, Gentaro Morimoto, Yousuke Ohno, Aki Hasegawa, and Makoto Taiji. 2014. MDGRAPE-4: A special-purpose computer system for molecular dynamics simulations. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 372, 2021 (August 2014). DOI:https://doi.org/10.1098/rsta.2013.0387

[26] Yousuke Ohno, Eiji Nishibori, Tetsu Narumi, Takahiro Koishi, Tahir H Tahirov, Hideo Ago, Masashi Miyano, Ryutaro Himeno, Toshikazu Ebisuzaki, Makoto Sakata, and Makoto Taiji. 2007. A 281 Tflops calculation for X-ray protein structure analysis with special-purpose computers MDGRAPE-3. In *Proceedings of the 2007 ACM/IEEE Conference on Supercomputing, SC'07*, ACM Press, New York, New York, USA, 1. DOI:https://doi.org/10.1145/1362622.1362698

[27] Yousuke Ohno, Rio Yokota, Hiroshi Koyama, Gentaro Morimoto, Aki Hasegawa, Gen Masumoto, Noriaki Okimoto, Yoshinori Hirano, Huda Ibeid, Tetsu Narumi, and Makoto Taiji. 2014. Petascale molecular dynamics simulation using the fast multipole method on K computer. *Computer Physics Communications* 185, 10 (October 2014), 2575–2585. DOI:https://doi.org/10.1016/j.cpc.2014.06.004

[28] Szilárd Páll, Mark James Abraham, Carsten Kutzner, Berk Hess, and Erik Lindahl. 2015. Tackling Exascale Software Challenges in Molecular Dynamics Simulations with GROMACS. In *Solving Software Challenges for Exascale*, Springer International Publishing, Cham, 3–27.

[29] Cristian Predescu, Adam K. Lerer, Ross A. Lippert, Brian Towles, J. P. Grossman, Robert M. Dirks, and David E. Shaw. 2020. The u -series: A separable decomposition for electrostatics computation with improved accuracy. *Journal of Chemical Physics* 152, 8 (February 2020), 084113. DOI:https://doi.org/10.1063/1.5129393

[30] F.B. Richards. 1991. A Gibbs phenomenon for spline functions. *Journal of Approximation Theory* 66, 3 (September 1991), 334–351. DOI:https://doi.org/10.1016/0021-9045(91)90034-8

[31] I. J. Schoenberg. 1973. *Cardinal Spline Interpolation*. Society for Industrial and Applied Mathematics. DOI:https://doi.org/10.1137/1.9781611970555

[32] I. J. Schoenberg and A. Sharma. 1971. The interpolatory background of the Euler-MacLaurin quadrature formula. *Bulletin of the American Mathematical Society* 77, 6 (November 1971), 1034–1039. DOI:https://doi.org/10.1090/S0002-9904-1971-12851-3

[33] Yibing Shan, John L. Klepeis, Michael P. Eastwood, Ron O. Dror, and David E. Shaw. 2005. Gaussian split Ewald: A fast Ewald mesh method for molecular simulation. *Journal of Chemical Physics* 122, 5 (February 2005), 054101. DOI:https://doi.org/10.1063/1.1839571

[34] David E. Shaw, Martin M. Deneroff, Ron O. Dror, Jeffrey S. Kuskin, Richard H. Larson, John K. Salmon, Cliff Young, Brannon Batson, Kevin

J. Bowers, Jack C. Chao, Michael P. Eastwood, Joseph Gagliardo, J. P. Grossman, C. Richard Ho, Douglas J. Lerardi, István Kolossváry, John L. Klepeis, Timothy Layman, Christine McLeavey, Mark A. Moraes, Rolf Mueller, Edward C. Priest, Yibing Shan, Jochen Spengler, Michael Theobald, Brian Towles, and Stanley C. Wang. 2008. Anton, a special-purpose machine for molecular dynamics simulation. *Communications of the ACM* 51, 7 (July 2008), 91–97. DOI:https://doi.org/10.1145/1364782.1364802

[35] David E. Shaw, J.P. Grossman, Joseph A. Bank, Brannon Batson, J. Adam Butts, Jack C. Chao, Martin M. Deneroff, Ron O. Dror, Amos Even, Christopher H. Fenton, Anthony Forte, Joseph Gagliardo, Gennette Gill, Brian Greskamp, C. Richard Ho, Douglas J. Ierardi, Lev Iserovich, Jeffrey S. Kuskin, Richard H. Larson, Timothy Layman, Li-Siang Lee, Adam K. Lerer, Chester Li, Daniel Killebrew, Kenneth M. Mackenzie, Shark Yeuk-Hai Mok, Mark A. Moraes, Rolf Mueller, Lawrence J. Nociolo, Jon L. Peticolas, Terry Quan, Daniel Ramot, John K. Salmon, Daniele P. Scarpazza, U. Ben Schafer, Naseer Siddique, Christopher W. Snyder, Jochen Spengler, Ping Tak Peter Tang, Michael Theobald, Horia Toma, Brian Towles, Benjamin Vitale, Stanley C. Wang, and Cliff Young. 2014. Anton 2: Raising the Bar for Performance and Programmability in a Special-Purpose Molecular Dynamics Supercomputer. In *SC14: International Conference for High Performance Computing, Networking, Storage and Analysis*, IEEE, 41–53. DOI:https://doi.org/10.1109/SC.2014.9

[36] J Sheng, C Yang, A Sanaullah, M Papamichael, A Caulfield, and M C Herbordt. 2017. HPC on FPGA clouds: 3D FFTs and implications for molecular dynamics. In *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, 1–4. DOI:https://doi.org/10.23919/FPL.2017.8056853

[37] Robert D. Skeel, Ismail Tezcan, and David J. Hardy. 2002. Multiple grid methods for classical molecular dynamics. *Journal of Computational Chemistry* 23, 6 (April 2002), 673–684. DOI:https://doi.org/10.1002/jcc.10072

[38] Makoto Taiji, Tetsu Narumi, Yousuke Ohno, Noriyuki Futatsugi, Atsushi Suenaga, Naoki Takada, and Akihiko Konagaya. 2003. Protein explorer: A petaflops special-purpose computer system for molecular dynamics simulations. In *Proceedings of the 2003 ACM/IEEE Conference on Supercomputing, SC 2003*, ACM Press, New York, New York, USA, 15. DOI:https://doi.org/10.1145/1048935.1050166

[39] Chen Yang, Tong Geng, Tianqi Wang, Rushi Patel, Qingqing Xiong, Ahmed Sanaullah, Chunshu Wu, Jiayi Sheng, Charles Lin, Vipin Sachdeva, Woody Sherman, and Martin Herbordt. 2019. Fully Integrated FPGA Molecular Dynamics Simulations. In *SC '19: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (SC '19), Association for Computing Machinery, New York, NY, USA, 1–31. DOI:https://doi.org/10.1145/3295500.3356179

[40] Cliff Young, Joseph A. Bank, Ron O. Dror, J. P. Grossman, John K. Salmon, and David E. Shaw. 2009. A 32x32x32, spatially distributed 3D FFT in four microseconds on Anton. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, SC '09*. DOI:https://doi.org/10.1145/1654059.1654083

Appendix: Artifact Description/Artifact Evaluation

SUMMARY OF THE EXPERIMENTS REPORTED

Fig. 3 and Fig. 4 were plotted by Python 3.7.3 with matplotlib 3.0.3 and numpy 1.16.2 on Intel Xeon E5540. For Table 1, we prepared a coordinate file of TIP3P water molecules by single precision GROMACS 2018.2 on Intel Xeon X5570 and ran a C++ program implementing the theory section on Intel Xeon E5540. The total energies of the first row in Fig. 4 were generated by double precision GROMACS 2018.2 on Intel Xeon E5540. The total energies of 2 to 4 rows in Fig. 4 were generated by double precision GROMACS 2018.2 implementing the theory section on Intel Xeon E5540.

Components of the MDGRAPE-4A SoC were coded using SystemVerilog and synthesized by Synopsys DesignCompiler N-2017.09. General purpose cores of the SoC were designed using Synopsys ASIP Designer M-2017.03-SP3. As the 3D-torus network interface, Rambus 6G SerDes PHY was embedded in the SoC, 6 directions and 12 lanes each. The MDGRAPE-4A SoC was implemented by TSMC CMOS 40-nm technology, and its die size was 16.2 mm x 16.2 mm. The clock frequency is 0.6 GHz except for nonbonded pipelines at 0.8 GHz. The MDGRAPE-4A board consists of 8 SoCs and two types of FPGAs, Xilinx XC7K160T for SoC interface and Intel Arria 10 10AX048 for host and TMENW interface. The TMENW boards contain Intel 10AX057 or 10AX115 (root board). Xilinx Vivado v2019.1 and Intel Quartus Prime Version 19.1 Standard Edition were used to implement designs of these FPGAs. 3D-FFT operation was implemented on 10AX115 using native floating-point DSPs.

Software codes for GP cores in the SoC was written in C++ and compiled by Synopsys ASIP Programmer <MDGRAPE-4A> P-2019.09-SP2 that supports custom instructions. MDGRAPE-4A boards are connected to Linux host computers by 10 Gbps ethernet. The access library and utilities for MDGRAPE-4A system on the host computer was written in C++. Some of utilities are written in Python-3.6 that wraps the access library by pybind11. We converted a topology file created by GROMACS 2018.2 into input files for MDGRAPE-4A by the in-house utilities. Fig. 9 and Fig. 10 depict a detailed timing log of a MD simulation run that contains 80,540 atoms.

Author-Created or Modified Artifacts:

```
Persistent ID: https://www.bdr.riken.jp/en/research/⌋
↪  labs/taiji-m-computational/paper.html
Artifact name: MDGRAPE-4A
```

BASELINE EXPERIMENTAL SETUP, AND MODIFICATIONS MADE FOR THE PAPER

Relevant hardware details: Intel Arria 10 10AX115

Operating systems and versions: CentOS 7.9 running Linux kernel 3.10.0

Compilers and versions: GCC 6.2

Applications and versions: GROMACS 2018.2, 2020.5

Key algorithms: multilevel summation method, FFT