

# AN FPGA IMPLEMENTATION OF RECIPROCAL SUMS FOR SPME

*Sam Lee<sup>1</sup> and Paul Chow*

Department of Electrical and Computer Engineering  
University of Toronto  
Toronto, Ontario, Canada, M5S 3G4  
sam.lee@amd.com; pc@eecg.toronto.edu

## ABSTRACT

Molecular Dynamics simulations have become an interesting target for acceleration using Field-Programmable Gate Arrays (FPGA). Still to be attempted completely in FPGA hardware is the computation of the Coulombic interactions using the Smooth Particle Mesh Ewald (SPME) algorithm.

In this work, we describe the design, the implementation, and the verification effort of an FPGA compute engine, named the Reciprocal Sum Compute Engine (RSCE), that calculates the reciprocal space contribution to the electrostatic energy and forces using the SPME algorithm. This is the first known attempt to fully capture the entire computation in dedicated FPGA hardware. We also investigate the fixed-point precision requirements, a strategy for parallelizing the RSCE and present an estimate of potential speedup.

## 1. INTRODUCTION

Molecular Dynamics (MD) is a numerical method for predicting the time-dependent microscopic behavior of a many-body system [1]. Due to its computational complexity, it is often solved using large supercomputers and clusters, which makes it a candidate for hardware acceleration.

Since the majority of MD simulation time is spent on non-bonded energy and force calculations, it is vital to speedup these non-bonded calculations. There are two types of non-bonded interactions; they are short-range Lennard-Jones (LJ) interactions and long-range Coulombic interactions.

For the Coulombic interactions, the Smooth Particle Mesh Ewald (SPME) algorithm [2], which scales as  $N \log(N)$ , is one approach used for computing the electrostatic energy and force. In the SPME algorithm, the calculation of the Coulombic interaction is divided into a short-range direct sum and a long-range reciprocal sum. The

direct-sum can be computed in a way similar to the LJ interactions. The greater challenge is to speedup the reciprocal sum calculation in hardware.

This paper describes the design and verification effort of a fixed-point FPGA compute engine, named the Reciprocal Sum Compute Engine (RSCE) that speeds up the SPME reciprocal sum calculation. In addition to the hardware implementation, results of a fixed-point precision analysis and the description of a parallelization strategy are also presented. The primary goal of this work is to undertake the task of building the engine as a way of learning the issues involved with implementing this computation in hardware.

This paper is divided into ten sections. Section 2 provides a brief background for the SPME reciprocal sum calculation. Section 3 discusses relevant work. Section 4 describes the development environment for this work. Section 5 describes the overall architecture of the RSCE and provides a brief overview of its design blocks. Section 6 describes the RSCE operations for calculating the SPME reciprocal sum. Section 7 presents a parallelization strategy for using multiple RSCEs. Section 8 describes the test environment and a precision analysis of the fixed-point computations. Section 9 describes a more efficient RSCE design and provides an estimation for its degree of speedup. Lastly, Section 10 concludes this paper and offers recommendations for future work.

## 2. SPME RECIPROCAL SUM BACKGROUND

The reciprocal-sum component of the energy and force computations are approximated by [2]:

$$\begin{aligned}\tilde{E}_{rec} &= \frac{1}{2\pi V} \sum_{m \neq 0} \frac{\exp(-\pi^2 m^2 / \beta^2)}{m^2} B(m_1, m_2, m_3) \\ &\quad \cdot F(Q)(m_1, m_2, m_3) F(Q)(-m_1, -m_2, -m_3) \\ &= \frac{1}{2} \sum_{m_1=0}^{K_1-1} \sum_{m_2=0}^{K_2-1} \sum_{m_3=0}^{K_3-1} Q(m_1, m_2, m_3) \\ &\quad \cdot (\theta_{rec} * Q)(m_1, m_2, m_3)\end{aligned}\tag{1}$$

<sup>1</sup>Now with Advanced Micro Devices Inc., Markham, Ontario, L3T 7X6

We would like to acknowledge Xilinx, CMC Microsystems and the SOCRN, and NSERC for their support.



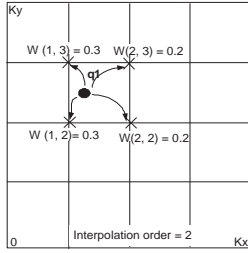
## 5.1. RSCE Functional Blocks

The five functional blocks of the RSCE are described in the following sections with the help of some simple diagrams representing a 2D simulation space.

### 5.1.1. B-Spline Coefficient Calculator (BCC)

The BCC block is responsible for calculating the B-Spline coefficients  $M_n(u_i - k)$  for all particles. As shown in (3), these coefficients are used in the composition of the charge grid  $Q$ . It also computes the derivatives of the coefficients, which are necessary in the force computation as shown in (2). As illustrated in Figure 3, for an interpolation order of two, each charge (e.g.  $q_1$  in Figure 3) is interpolated to four surrounding grid points. Each grid point gets a portion of the charge and the size of the portion depends on the value of the B-Spline coefficients.

$$Q(k_1, k_2, k_3) = \sum_{i=1}^N \sum_{n_1, n_2, n_3} [q_i M_n(u_{1i} - k_1 - n_1 K_1) \cdot M_n(u_{2i} - k_2 - n_2 K_2) \cdot M_n(u_{3i} - k_3 - n_3 K_3)] \quad (3)$$



**Fig. 3.** B-Spline Interpolation

### 5.1.2. Mesh Composer (MC)

The MC block goes through all particles and identifies the grid points each particle should be interpolated to. Then, it composes the charge grid  $Q$  by assigning the charge portions to the grid points using (3).

### 5.1.3. 3D-Fast Fourier Transform (3D-FFT)

The 3D-FFT block performs the three-dimensional forward and inverse Fast Fourier Transform on the charge grid  $Q$ . As shown in (1) and (2), the charge grid transformations (e.g.  $F(Q)(m_1, m_2, m_3)$ ) are necessary in both the reciprocal energy and force calculations.

### 5.1.4. Reciprocal Energy Calculator (EC)

The EC block goes through all grid points in the charge grid  $Q$ , calculates the energy contribution of each grid point, and

then computes the total reciprocal energy  $\tilde{E}_{rec}$  by summing up the energy contributions from all grid points. In addition to the energy computation, the EC block is also responsible for updating the charge grid  $Q$  for the force calculation [10].

### 5.1.5. Reciprocal Force Calculator (FC)

Similar to the MC block, the FC block examines each particle, identifying all the grid points that a particle has been interpolated to. Then, it computes the directional forces exerted on the particle by summing up the forces that the surrounding interpolated grid points exert on the particle.

## 5.2. RSCE Memory Banks

There are five memory banks that facilitate the RSCE calculation:

1. Particle Information Memory (PIM) - The upper half stores the shifted and scaled fractional  $(x, y, z)$  coordinates and the charge of all particles; while the lower half stores the computed directional forces for all particles. The size of this memory limits the total number of particles in the simulation system.
2. B-Spline Coefficients Lookup Memory (BLM) - The BLM stores the slope and function values of the B-Spline coefficients and their respective derivatives at the predefined lookup points. The size of this memory limits the maximum B-Spline interpolation order.
3. Charge Mesh Memory Real and Imaginary (QMMR and QMMI) - The QMMR(I) stores the real(imaginary) part of the charge grid  $Q$ . The size of these memories limits the simulation grid size.
4. Energy Term Memory (ETM) - The ETM stores the values of the “energy term” for all grid points. These values are used in the reciprocal energy and force calculations. The energy term is defined in (4). The size of this memory limits the simulation grid size.

$$eterm = \frac{\exp(-\pi^2 m^2 / \beta^2) B(m_1, m_2, m_3)}{\pi V m^2} \quad (4)$$

## 6. RSCE OPERATIONS

In this section, the steps to calculate the SPME reciprocal sum are described. These steps follow closely with the steps used in the software SPME program written by A. Touk-maji [11].

Table 1 describes the steps and the corresponding complexity order for computing the reciprocal energy and forces using the RSCE with a host computer. The step number in Column 1 is also shown in the architectural diagram of the

RSCE (Figure 2) as dashed circles for easy reference. In Column 3,  $K_i$  is the grid size,  $N$  is the number of particles, and  $P$  is the interpolation order.

By analyzing the complexity order of each step, it can be concluded that the majority of the computation time is spent in steps 7 (mesh composition), 8 (3D-FFT), 11 (3D-IFFT), and 12 (force computation). The computation time of steps 8 and 11 depends on the grid size ( $K_1$ ,  $K_2$ , and  $K_3$ ), while that of steps 7 and 12 depend mainly on the number of particles  $N$  and the interpolation order  $P$ .

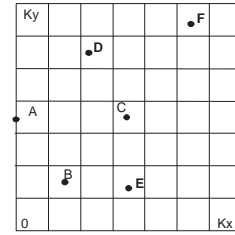
**Table 1.** Steps for SPME Reciprocal Sum Calculation

#	Operation	Complexity Order
1	Host computes the reciprocal lattice vectors for x, y, and z directions.	1
2	Host computes the B-Spline coefficients and derivatives for all lookup fractional coordinates and stores them into the BLM.	$2^{lookup\_precision}$
3	Host computes the energy terms for all grid points that are necessary in energy calculation and stores them into the ETM.	$K_1 \times K_2 \times K_3$
4	Host loads or updates the x, y, and z coordinates of all particles.	$3 \times N$
5	Host computes scaled and shifted fractional coordinates for all particles and loads them into the upper half of the PIM memory. It also zeros all entries in the QMMR and QMMI.	$3 \times N$
6	FPGA BCC performs lookup and computes the B-Spline coefficients for all particles for x, y, and z directions.	$3 \times N \times P$
7	FPGA MC composes the grid charge grid using the computed coefficients. The calculated values are stored in the QMMR.	$N \times P \times P \times P$
8	FPGA 3D-FFT computes $F^{-1}(Q)$ by performing inverse FFT on each row for each direction. The transformed values are stored in the QMMR and the QMMI.	$K_1 \times K_2 \times K_3 \times \log(K_1 \times K_2 \times K_3)$
9	FPGA EC goes through each grid point to compute the reciprocal energy and update the QMM. It uses the grid index to lookup the energy terms.	$K_1 \times K_2 \times K_3$
10	FPGA BCC performs lookup and computes the B-Spline coefficients and derivatives for all particles for all x, y, and z directions.	$2 \times 3 \times N \times P$
11	FPGA 3D-FFT computes the forward $F(Q)$ and loads the values into the QMMR.	$K_1 \times K_2 \times K_3 \times \log(K_1 \times K_2 \times K_3)$
12	FPGA FC goes through all particles, identifies their interpolated grid points, and computes the reciprocal forces for x, y, and z directions. The forces will be stored in the lower half of the PIM.	$3 \times N \times P \times P \times P$
13	Repeat 4 – 12 until simulation is done.	N/A

## 7. RSCE PARALLELIZATION STRATEGY

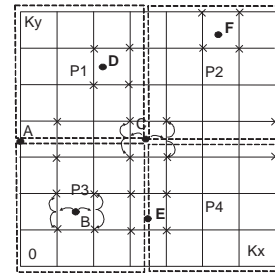
This section describes a strategy to parallelize the SPME reciprocal sum calculation using multiple RSCEs. It discusses the data communication requirement among the RSCEs and the synchronization ability of the RSCE, which is necessary for the parallelization. The parallelization strategy described in this section is similar to the way NAMD2 parallelizes its reciprocal sum computation [9, 12].

To illustrate the basic steps involved in the parallelization strategy, a 2D simulation system example is assumed and is shown in Figure 4. In this 2D simulation system, the grid size is  $K_x \times K_y = 8 \times 8$ , the number of particles is  $N = 6$ , the interpolation order is  $P = 2$ , and the number of RSCEs is  $NumP = 4$ .



**Fig. 4.** 2D System with Six Particles

To parallelize the calculation, each RSCE is assigned a portion of the grid at the beginning of a simulation timestep. After the grid-point assignment, each RSCE is responsible for the B-Spline coefficients calculation and mesh composition for all charges that are either inside its mini-grid area or are  $P/2$  grid points away from its mini-grid boundary. The mini-grid assignment and its boundary are indicated by the dashed lines in Figure 5.



**Fig. 5.** Parallel Mesh Composition

To realize the grid point distribution step, the host only programs the PIM memory with the coordinates and charges of the particles that the RSCE is responsible for. Furthermore, it also sets the number of particles  $N$ . The BLM memory is programmed with the full content as in the single RSCE case. Upon finishing mesh composition, each RSCE holds its mini-grid in the QMMR memory and it is then

halted until all other RSCEs finish their mini-grid composition; this is the first synchronization point.

The next step is the inverse 2D-FFT operation. In this operation each RSCE is responsible for  $(K_x \times K_y / \text{NumP})$  grid points. Before performing the IFFT, data communication is needed among all RSCEs such that each RSCE holds the grid-point data for the IFFT rows it is responsible for. In our 2D example, the 2D-IFFT is broken down into x- and y-direction 1D-IFFTs. During the x-direction 1D-IFFT operation, each RSCE holds  $K_y / \text{NumP}$  rows of grid points and each RSCE holds  $K_x / \text{NumP}$  columns during the y-direction 1D-IFFT. As in the single RSCE case, the energy calculation can be overlapped with the 1D-IFFT for the last dimension. Thus, at the end of the y direction 1D-IFFT step, each RSCE has calculated the energy contribution of its grid-point rows and has also updated the QMMR and QMMI memories. The total reciprocal energy can then be calculated by adding the calculated energy from all RSCEs.

Following the inverse transformation and the QMM update, the forward 2D-FFT can be performed on the updated grid. The data communication requirement is the same as that of the 2D-IFFT step. To save one transposition, the forward 2D-FFT can be performed in reverse order. That is, the 1D-FFT for the  $y$  direction is done before that of  $x$  direction. Again, synchronization is needed in between the 1D-FFT passes.

Before the force calculation can proceed, each RSCE should hold the grid-point data of its previously assigned mini-grid. The force calculation involves interpolating the force from the grid points to the particles. After the calculation, each RSCE writes the calculated forces into the PIM memory. For a particle (e.g. particle c in the centre of Figure 5) that interpolated its charge to several RSCEs, its force is calculated by a summation of the calculated forces from all RSCEs. This summation is carried out by the host after all RSCEs report the partial forces of the particle.

As observed from the above steps, the sizes of the memories for each RSCE can be reduced when multiple RSCEs are used. The PIM memory needs only hold the data for approximately  $N / \text{NumP}$  particles. The QMMR and QMMI memories need only hold the grid-point values for the mini-grid or the FFT rows. The ETM memory need only hold the energy term that corresponds to the grid points of the mini-grid. However, the BLM memory has to hold the same amount of data as in the single RSCE case.

With the described parallelization strategy, there is a restriction on the number of RSCEs that the computations can be parallelized to. The reason is that when performing the FFT, it is more efficient for an RSCE to hold all grid-point data for the FFT rows. The simplest configuration is to set  $\text{NumP} = K$  such that each RSCE can perform the FFT for a 2-D plane. On the other hand, if  $\text{NumP}$  is set to  $K \times K$ , then each RSCE can perform the FFT for one row.

For a 3D simulation system, the steps taken for parallelizing the SPME calculations into multiple RSCEs are similar; a detailed description can be found in Lee's thesis [10].

## 8. TESTING AND PRECISION ANALYSIS

The RSCE design is verified using a self-checking design verification testbench written in SystemC [13]. In addition to being a verification model for the design, the simulation model for the RSCE can be also configured to use either fixed-point or double-precision floating-point arithmetic allowing for some precision analysis to be done. The SPME software implementation [11] is used as a reference to ensure the correctness of this simulation model.

### 8.1. Design Verification Environment

The RSCE verification environment is shown in Figure 6; it consists of the following components:

1. A testcase that takes a Protein Data Bank (PDB) file as input and provides the simulation configuration settings to the RSCE testbench (RSCE\_TB) via an On-chip Peripheral Bus (OPB) packet (OpbPkt).
2. A host model that loads the PIM with particle information and loads the BLM and the ETM with the lookup values.
3. An RSCE testbench that instantiates the RSCE RTL design, the RSCE SystemC behavioral model, an OPB bus functional model that sends the register programming instructions to the RSCE RTL and behavioral model, the five memory models, and a checker that compares the calculated energy and forces from the RSCE RTL and its behavioral counterpart.

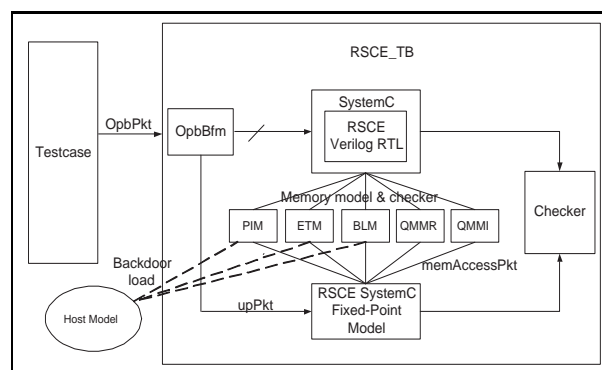


Fig. 6. RSCE SystemC Verification Environment

A verification testcase is done by running a single timestep MD simulation of a molecular system described by

the input PDB file with predefined interpolation order and grid size.

## 8.2. Precision Analysis with the RSCE Model

The RSCE uses fixed-point arithmetic to reduce resource utilization [10]. However, the accuracy of the fixed-point arithmetic is limited. To obtain an accurate MD simulation result, Narumi [5] shows that it should be enough to set the relative error bound for the reciprocal energy and force computations to  $10^{-4.5}$ . To further ensure the accuracy of the RSCE computations, the relative error bound goal for the RSCE is set to  $10^{-5}$ . Because of the resource limitation, the current RSCE implementation is not able to achieve this precision goal. The precision settings to achieve the error bound goal of  $10^{-5}$  are studied with the RSCE SystemC model operating in fixed-point mode.

This section describes how the fixed-point precision of the B-Spline calculation and the 3D-FFT calculation affects the errors of the reciprocal energy and force calculations. Furthermore, this section also states the minimum required B-Spline and 3D-FFT calculation precision to achieve the relative energy and force error goal of less than  $10^{-5}$ .

To derive the relationship between the calculation precision and the magnitude of the relative force and energy errors, a script to cycle through different precision settings and different MD simulation configurations is written to automate the precision analysis simulation process. Using the script, 200 single-timestep MD simulations are carried out with each precision setting for different simulation configurations (i.e., different grid size  $K$  and interpolation order  $P$ ). The system being simulated is charge-neutral and it contains 100 randomly located particles with randomized charge within the range of [5.0, -5.0]. The simulation box is orthogonal with dimensions of  $16.0\text{\AA} \times 16.0\text{\AA} \times 16.0\text{\AA}$ . The errors are computed and recorded for each simulation run. The absolute error, the relative error, and the RMS relative error in the energy and force computations are calculated using (5), (6), and (7) respectively. The ‘‘Golden’’ subscript in the equations indicates that the reciprocal energy or forces are calculated with the SPME software package [11] while the ‘‘FxptSc’’ subscript indicates those are calculated with the fixed-pointed SystemC RSCE model.

$$Err_{Abs} = |Result_{Golden} - Result_{FxptSc}| \quad (5)$$

$$Err_{Rel} = \left| \frac{Result_{Golden} - Result_{FxptSc}}{Result_{Golden}} \right| \quad (6)$$

$$Err_{RMS\_Rel} = \sqrt{\frac{\sum_{i=1}^N (Result_{Golden} - Result_{FxptSc})^2}{N}} \quad (7)$$

The precision analysis result for the simulation setting of  $P=8$ ,  $K=64$ , and  $N=100$  is partially presented in Table 2. Complete results are found in Lee’s thesis [10].

**Table 2.** Error Results of 200 Single-Timestep Simulations

FFT SFXP	BSP SFXP	Avg. Energy Relative Error (E-06)	Max. Energy Relative Error (E-06)	Avg. Force RMS Relative Error (E-06)	Max. Force RMS Relative Error (E-06)
14.26	1.21	0.755	2.12	36.7	476
14.26	1.24	0.687	1.45	4.01	39.2
14.26	1.27	0.611	1.17	3.35	20.8
14.26	1.30	0.601	1.03	2.06	11.2
14.30	1.21	0.716	1.78	5.29	827
14.30	1.24	0.581	1.00	3.98	38.9
14.30	1.27	0.599	1.28	1.98	9.80
14.30	1.30	0.570	1.01	1.70	7.80

Based on the simulation results, it is found that an FFT calculation precision of {14.30} (14 integer bits, 30 fractional bits) and a B-Spline calculation precision of {1.27} can achieve the relative error goal of  $10^{-5}$  in all 200 simulation runs. A much more thorough analysis of fixed-point precision throughout the system is required to completely validate these results.

## 9. BETTER DESIGN AND SPEEDUP ESTIMATION

The RSCE implementation uses approximately 88% of the logic resources available in the XC2V2000 FPGA. This means that it is not possible to add more logic and limits the maximum operating frequency to 40MHz.

Based on the drawbacks [10] of the current implementation, a ‘‘better’’ RSCE design should have the following improvements:

1. Overlapping of the BCC and the MC/FC calculations,
2. Parallelization of the  $x$ -,  $y$ -, and  $z$ - direction force calculations,
3. Utilization of the fastest FFT LogiCore,
4. Utilization of multiple sets of QMM memories, and
5. Utilization of an FPGA with higher speed grade or with better technology.

With the above improvements and the assumption that the RSCE can operate at 100 MHz with newer FPGA technology, it is estimated that the RSCE could provide a 3-fold to 14-fold speedup compared to the software implementation running on an Intel P4 machine [10]. An RSCE with more QMM memories can further accelerate the SPME calculation by increasing the QMM memory access bandwidth. More information on the RSCE speedup estimation can be found in Lee’s thesis [10].

## 10. CONCLUSIONS AND FUTURE WORK

This paper discusses the design and implementation of the reciprocal space part of the SPME algorithm in an FPGA. It is believed this work is the first such effort. In this paper, several key findings are discussed.

It is found that to limit the energy and force relative error to be less than  $10^{-5}$ ,  $\{1.27\}$  fixed-point precision in the B-Spline coefficients and derivatives calculation and  $\{14.30\}$  fixed-point precision in the 3D-FFT calculation are necessary. It is also shown that the relative error for energy and force calculations increases with increasing grid size  $K$  and interpolation order  $P$ .

The precision analysis performed in this work emphasizes the intermediate calculation precisions. More detailed precision analysis on the precision of the input variables should be performed to make the analysis more complete. To further enhance the precision analysis, the effect of precision used in each arithmetic stage should also be investigated. The RSCE SystemC model developed in this work can help with the analysis.

The current fixed-point FFT core should be replaced with a block floating-point core to avoid overflow due to the FFT dynamic range expansion and to increase calculation precision of the 3D-FFT operation.

With larger FPGAs becoming available, in addition to implementing the recommendations described in Section 9, it may be simpler to just use floating-point arithmetic everywhere and not worry about the precision analysis.

This work has shown that it will be difficult to achieve significant speedups (much greater than 10) of the reciprocal sum calculation for SPME in hardware because the overall computation is not amenable to fine-grain techniques such as pipelining. Portions of the computation can be pipelined, but eventually, the parallelism becomes limited by access to the charge mesh memory, which must be accessed several times throughout the computation. This also makes it difficult to achieve overlapping of the various phases of the computation. Partitioning of the charge mesh memory will help, but it will also bring in the overhead of communication between the partitions.

It is also important to remember that the SPME calculation is only part of the overall MD simulation. How a hardware SPME accelerator fits with the rest of the system will determine the speedup of the overall MD simulation.

## 11. REFERENCES

- [1] M.P. Allen and D.J. Tildesley. *Computer Simulation of Liquids, 2nd Ed.*. Oxford Science Publications, 1989.
- [2] U. Essmann, L. Perera, and M.L. Berkowitz. A Smooth Particle Mesh Ewald Method. *J. Chem. Phys.*, 103(19):8577–8593, 1995.
- [3] S. Toyoda, H. Miyagawa, K. Kitamura, T. Amisaki, E. Hashimoto, and A. Kusumi H. Ikeda, and N. Miyakawa. Development of MD Engine: High-Speed Accelerator with Parallel Processor Design for Molecular Dynamics Simulations. *Journal of Computational Chemistry*, 20(2):185–199, 1999.
- [4] Tetsu Narumi, Ryutaro Susukit, Toshikazu Ebisuzaki, Geoffrey McNiven, and Bruce Elmegreen. Molecular Dynamics Machine: Special-purpose Computer for Molecular Dynamics Simulations. *Molecular Simulation*, 21:410–415, 1999.
- [5] Tetsu Narumi. *Special-purpose Computer for Molecular Dynamics Simulations*. PhD thesis, Department of General Systems Studies, College of Arts and Sciences, University of Tokyo, 1998.
- [6] Makoto Taiji, Tetsu Narumi, Yousuke Ohno, Noriyuki Futatsugi, Atsushi Suenaga, Naoki Takada, and Akihiko Konagaya. Protein Explorer: A Petaops Special-Purpose Computer for Molecular Dynamics Simulations. *Genome Informatics*, 13:461–462, 2002.
- [7] Ronald Scrofano and Viktor K. Prasanna. Preliminary Investigation of Advanced Electrostatics in Molecular Dynamics on Reconfigurable Computers. In *SC '06: Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*, New York, NY, USA, 2006.
- [8] Xilinx Multimedia Board Product Website. <http://www.xilinx.com/products/boards/multimedia/>, [Accessed Feb, 2007].
- [9] J. C. Phillips, G. Zheng, S. Kumar, and L. V. Kale. NAMD: Biomolecular Simulation on Thousands of Processors. In *Proceedings of the IEEE/ACM SC2002 Conference*. IEEE Press, 2002.
- [10] Sam Lee. An FPGA Implementation of the Smooth Particle Mesh Ewald Reciprocal Sum Compute Engine. Master's thesis, Department of Electrical and Computer Engineering, University of Toronto, 2005.
- [11] A. Toukmaji. PME and DPME software package. The DPME package is also included in the NAMD2.1 public distribution.
- [12] L. Kale, R. Skeel, M. Bhandarkar, R. Brunner, A. Gursoy, N. Krawetz, J. Phillips, A. Shinozaki, K. Varadarajan, and K. Schulten. NAMD2: Greater Scalability for Parallel Molecular Dynamics. *J. Comp. Phys.*, 151:283–312, 1999.
- [13] SystemC Community Website. <http://www.systemc.org>. [Accessed Feb, 2007].