A High Performance Implementation for Molecular Dynamics Simulations on a FPGA Supercomputer

Server Kasap[1] and Khaled Benkrid[2]
[1]*Cyprus International University, Haspolat, Nicosia, Mersin-10, Turkey*
[2]*The University of Edinburgh, King's Buildings, Mayfield Road, Edinburgh, EH9 3JL, UK*
skasap@ciu.edu.tr, k.benkrid@ed.ac.uk

Abstract

The design and implementation of an FPGA core that parallelises all the necessary operations to compute the non-bonded interactions in a MD simulation with the purpose of accelarating the LAMMPS MD software is presented in this paper. Our MD processor core comprised of 4 identical pipelines working independently in parallel to evaluate the non-bonded potentials, forces and virials was implemented on the nodes of a FPGA-based supercomputer, namely Maxwell. Implementing our FPGA core on multiple nodes of Maxwell allowed us to produce a special-purpose parallel machine for the hardware acceleration of MD simulations. The timing performance figures of this machine for the pairwise LJ and short-range Coulombic (via PPPM) interaction computations in the MD simulations of the solvated Rhodopsin protein systems show performance gains over the pure software implementation by factors of up to 14 on two or more nodes of the Maxwell machine.

1. Introduction

Computer simulations are carried out to understand the properties of assemblies of molecules in terms of their structure and the microscopic interactions between them [1]. They act as a bridge between microscopic length and time scales and the macroscopic world of the laboratory, serving as a complement to conventional experiments.

There are two main families of simulation techniques: Molecular Dynamics (MD) and Monte Carlo (MC)-based simulations. MD is a deterministic simulation technique whereas results from MC simulations are stochastic. In MD, the time evolution of a set of interacting atoms modelled with classical mechanics is followed by integrating their Newtonian equations of motion. MD simulations of biomolecules provide a molecular picture of the structure and behaviour of biological systems such as enzymes, proteins, DNA strands and membranes. The MD method has applications in the fields of protein engineering [2], drug design [3] and refinements of structures [4].

However, biological systems of interest have sizes ranging from a few tens of thousands to millions of atoms. Performing MD simulation of a biological process, such as protein folding, for a reasonable physical time requires enormous amounts of computational effort and may take years to complete on conventional computers. Therefore, it is mandatory to utilize faster computing platforms.

Field Programmable Gate Arrays (FPGAs) in particular have recently been proposed as a viable alternative implementation platform for MD simulation due to their flexible computing and memory architecture which gives them ASIC-like performance with the added programmability feature [5][6][7]. Therefore, we chose FPGAs over ASICs as they offer reprogrammability, shorter development times and lower nonrecurring engineering (NRE) costs.

There are several MD simulation software tools. However, software for MD simulation can spend a very high percentage of the total computation time in calculating the non-bonded interactions among particles because the computational complexity of the evaluation of non-bonded potentials or forces is quadratic. Therefore, we can accelerate MD simulation by porting the calculation of the non-bonded interactions from software to FPGAs since non-bonded interactions lend themselves to be easily calculated in parallel. On the other hand, the remaining MD calculation, which is complex but only consumes a very limited percentage of the total computation time, can be left to software running on a host computer.

The design and implementation of an FPGA core that parallelises all the necessary operations to compute the non-bonded interactions in the Large-scale Atomic/Molecular Massively Parallel Simulation (LAMMPS) software tool [8] is explained in this paper. Our MD processor core is comprised of 4 identical pipelines working independently in parallel to evaluate the non-bonded potentials, forces and virials acting on a particle from all of the other particles in the simulated molecular system. A real hardware implementation of the designed core was achieved on the nodes of a FPGA-based supercomputer, called Maxwell [9], which consists of 64 Virtex-4 FPGA chips. Implementing our FPGA core on multiple nodes of Maxwell allowed us to produce a special-purpose parallel machine for the hardware acceleration of MD simulations. To the best of our knowledge, this is the first reported FPGA-based supercomputer acceleration of MD simulations.

The remainder of this paper will first present essential background information on MD simulation and then the

978-1-4577-0599-1/11/$26.00 ©2011 IEEE

general system architecture will be explained. Furthermore, the design and implementation of our FPGA core for computing the non-bonded interactions in a MD simulation will be elaborated. Following this, implementation results are presented and then evaluated comparatively with equivalent pure software implementations. Finally, conclusions are laid out with plans for future work.

2. Molecular dynamics simulation

Molecular Dynamics is commonly used for the simulation of the structural, thermodynamic and transport properties of large biological systems on a diverse range of timescales. In MD simulations, atoms in the system are treated as classical particles and are subject to covalent bond, Van der Waals and Coulomb forces from other particles. During a time-step of the MD simulation, forces are computed and accumulated on each atom due to its interaction with other atoms, and positions and velocities of atoms are updated by integrating the Newtonian equations of motion.

In MD simulations of biological systems, the potential for a particle i, Φ_i, is modelled as follows:

$$\Phi_i = \Phi_i^B + \sum_{j \neq i} \epsilon_{ab} \left\{ \left(\frac{\sigma_{ab}}{|\boldsymbol{r_{ji}}|} \right)^{12} - \left(\frac{\sigma_{ab}}{|\boldsymbol{r_{ji}}|} \right)^{6} \right\} + q_i \sum_{j \neq i} \frac{q_j}{|\boldsymbol{r_{ji}}|} \quad (1)$$

where $\mathbf{r_{ji}}$ is a vector from the particle j to i and q_i is the charge of the particle i. The first term Φ_i^B is the bonded potential due to interactions within the topology of the molecules and is expressed as:

$$\Phi_i^B = \sum_{bonds} K_b (r - r_0)^2 + \sum_{angles} K_\theta (\theta - \theta_0)^2 + \sum_{dihedrals} K_{\phi_p} \left[1 + d_p \cos(n_p \phi) \right] \quad (2)$$

Bonded potential is written here as sums over simple harmonic 2-body (bond), 3-body (angle) and 4-body (dihedral) interactions although other potential models could also be used. On the other hand, the last 2 terms in (1) are the non-bonded potential due to interactions between all pairs of atoms in the system. Note that the forces exerted on the particle i, $\mathbf{f_i}$, are obtained by taking the gradient of (1) with respect to the position of the particle.

The second term in (1), which describes van der Walls interaction, is the Lennard-Jones (LJ) potential characterized by a length parameter σ_{ab} and an energy parameter ε_{ab} where a and b denote the two atom types of particles. If we take the gradient of this potential, an LJ force $\mathbf{f_i^{LJ}}$ can be expressed as:

$$\boldsymbol{f_i^{LJ}} = \sum_{j \neq i} \frac{\epsilon_{ab}}{\sigma_{ab}^2} \left\{ 12 \left(\frac{\sigma_{ab}}{|\boldsymbol{r_{ji}}|} \right)^{14} - 6 \left(\frac{\sigma_{ab}}{|\boldsymbol{r_{ji}}|} \right)^{8} \right\} \boldsymbol{r_{ji}} \quad (3)$$

The third term on the right hand side of (1) is the Coulombic (C) potential, and the corresponding Coulombic force $\mathbf{f_i^C}$ is expressed as:

$$\boldsymbol{f_i^C} = q_i \sum_{j \neq i} \left(\frac{q_j}{|\boldsymbol{r_{ji}}|^3} \right) \boldsymbol{r_{ji}} \quad (4)$$

The computational complexity of evaluating Φ_i^B is $O(1)$ since only few particles are covalently bonded to the i^{th} particle. However, the computation time to evaluate nonbonded potential or force functions is $O(N)$ for each particle where N is the number of particles in the simulated system. Hence, the computational complexity to evaluate the functions for all particles in the system is $O(N^2)$. Accelerating these evaluations is therefore the prime target for the design of our MD core. Note that our MD processor core is able to deal with an arbitrary potential or force function although only the LJ and Coulombic interactions are mentioned in this section.

The simplest method for reducing the computation time is the cutoff convention. Contributions from particles outside a certain cutoff radius r_c are ignored in this method and hence, the time complexity is reduced to $O(N)$. For instance, since LJ force and potential decrease rapidly with increasing distance (refer to (3)), the sum over j can be truncated within the determined cutoff distance so that only a few neighbours of atom i contribute rather than all N. This does not affect the results in most cases provided that the particles are well separated with respect to an appropriate value of r_c.

In contrast, the Coulombic interaction is long-range which means it decreases slowly with an increase of distance (refer to (4)). Hence, evaluating Coulombic force as a truncated sum over neighbours rather than as a full sum introduces large inaccuracies [10]. On the other hand, applying the latter method is problematic in periodic systems. Consequently, other methods are often used for the evaluation of Coulombic force and potential. One of these methods is the Ewald method [11] and the one used by the LAMMPS software is the particle-particle particle-mesh (PPPM) method [12].

Virials represent the effect of mutual interaction of particles on the pressure in the system. The virial $\mathbf{v_i}$ on the particle i can be calculated with the following equation where T denotes the transpose of the vector:

$$\boldsymbol{v_i} = \sum_{j \neq i} \boldsymbol{f_{ji}} \boldsymbol{r_{ji}}^T \quad (5)$$

Note that the time complexity of this operation for all particles is $O(N^2)$ since it is $O(N)$ for each particle. Our MD processor incorporates the computation of all components of each virial.

3. System architecture

Our special-purpose parallel machine for MD simulations is a reconfigurable hardware accelerator plugged into a number of host CPUs. Fig. 1 illustrates the basic process flow in our machine for each time-step. LAMMPS MD simulation software running on a general purpose computer first initialises the simulation environment, calculates the less time-consuming bonded interactions and builds a neighbour list for each particle i in the simulated system, which includes all nearby j particles within a certain radius of the particle i. Then, for each neighbour list, software on host CPUs first broadcast the coordinates and electric charge of the particle i to the reconfigurable hardware and subsequently the coordinates and electric charge of each j particle in the neighbour list as well as the interaction parameters and the cutoff distances for the specific pairs of i and j particles are sent to the reconfigurable hardware one by one as shown in fig. 1. Following this, our parallel machine computes all non-bonded forces, virials and potentials acting on each particle i due to the j particles in its neigbour list and then, send these pairwise values back to the host CPU. Software running on the host uses the force values to calculate the acceleration of each particle in the simulated system and then integrates Newtonian equations of motion by an integration technique to update the velocity and position values of all particles at the current time-step. Software also adds-up pairwise potential and virial values to compute the total per-atom potentials and virials, respectively. The total potential energy and pressure in the simulated system at the current time-step are also calculated by accumulating these potential and virial values, respectively. Note that a new C++ class was written for the LAMMPS software using the FHPCA Parallel Toolkit (PTK) to be able to co-operate with the reconfigurable hardware for MD simulations in the way explained above. Another important point is that all transfers between host CPU and reconfigurable hardware are done with Direct Memory Access (DMA) method.

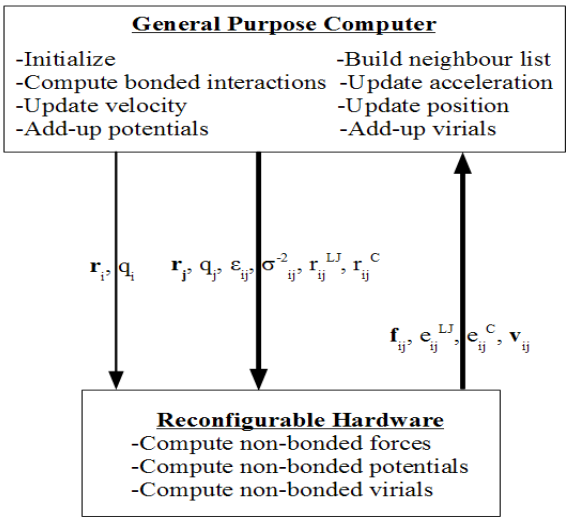

Fig. 1. Basic structure of our special-purpose parallel machine for Molecular Dynamics simutions

Fig. 2 shows the system connection diagram of our special-purpose parallel machine for MD simulations. Two processes of LAMMPS software run on each Intel Xeon CPU while an instance of our MD processor core resides in each user FPGA. Actually, a software process running on a host CPU and a hardware core in a user FPGA form a Maxwell node. The number of utilized Maxwell nodes where LAMMPS processes communicate with each other by Message Passing Interface (MPI) can be easily configured as desired.

Each Xeon CPU on PCI-X bus connects to two user FPGAs through bridge/control FPGAs mediating communication between the 32-bit wide PCI-X bus operating at 133 MHz and the 32-bit wide local buses of the user FPGAs operating at 80 MHz, as shown in fig. 2. Furthermore, user FPGAs in our MD machine are of Xilinx

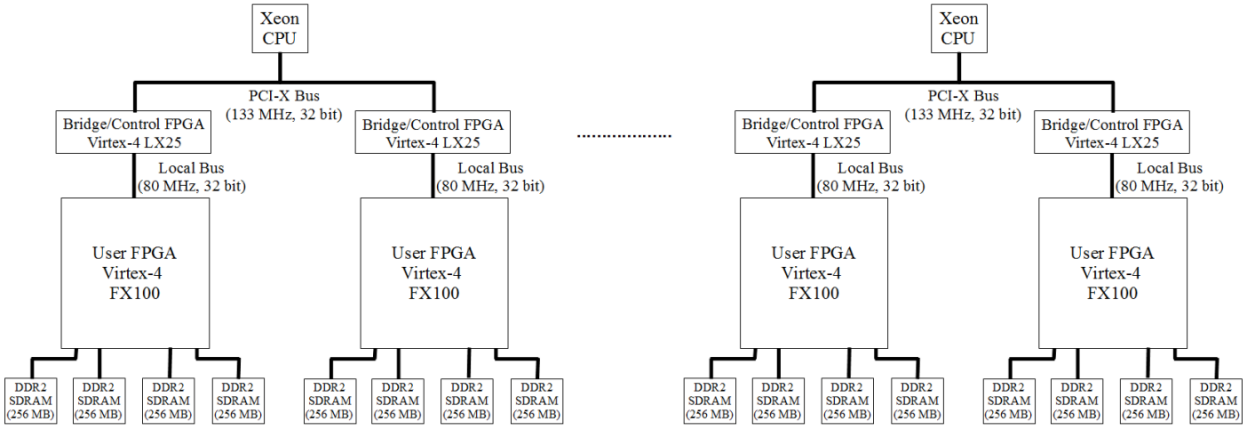

Fig. 2. System connection diagram of our special-purpose parallel machine for Molecular Dynamics simulations

Virtex-4 FX-100 type whereas smaller FPGAs bridging PCI-X and local buses are of Xilinx Virtex-4 LX25 type. On the other hand, four 256 MB DDR2 SDRAMs are connected to each user FPGA. The physical width and depth of the SDRAMs are 32 bits and 64M words, respectively while the logical width of the SDRAMs is 128 bits. Note that the MD processor core in a user FPGA runs at 150 MHz although the logic interfacing the user FPGA to the local bus it is connected to runs at 80 MHz.

Fig. 3 shows the block diagram of our MD processor core. As it can be seen, our MD core incorporates 4 identical MD pipelines which are working independently in parallel to evaluate the non-bonded potentials, forces and virials acting on a particle from each of the particles in the neighbour list of that particle. Each MD processor is associated with one of the SDRAM banks connected to the user FPGA. Furthermore, the LAMMPS process running on a host CPU transfers the simulation-related data mentioned above to the allocated first region of each SDRAM bank to be processed by the relevant MD processor. When these incoming transfers complete, the software process signals each MD processor to start its operation of reading data from its associated SDRAM bank through the use of an input buffer and then writing the evaluated potential, force and virial values back to the allocated second region of the associated SDRAM bank through the use of an output buffer, all under control of a Finite State Machine (FSM) as shown in fig. 3. When the MD processor is done with its operation, it signals the software process to transfer its computed values back from the relevant SDRAM bank for further processing as explained above. Moreover, two identical function coefficient memories shown in the fig. 3 store the coefficients of the interpolation used to evaluate a number of functions.

4. Design of Molecular Dynamics processor

In our design, internal format of the numbers used is the IEEE standard single precision (i.e. 32-bit wide) floating-point as shown in fig. 4 (a). All data are handled in this format. Hence, single precision floating-point arithmetic units are utilized throughout our MD processor. Several pipelined floating-point multipliers and adders/subtractors obtained from [13] are incorporated in our design whose operation latencies are 4 and 6 clock cycles, respectively, as explained in [14]. These arithmetic units do not support denormalized numbers and NaN ("not a number") to minimize the required hardware resources and realize high operation speed by simplifying the circuitry.

SDRAM banks in our MD machine were partitioned into two regions. The first region of a SDRAM bank was allocated to data transfers from host CPU while the second region was allocated to data transfers from the designated MD processor on a user FPGA, as mentioned in section 3. Fig. 4 (b) shows the layout of a memory portion in the first region of a SDRAM bank storing the coordinates $\mathbf{r}_i = (r_x, r_y, r_z)$ and electric charge q_i of a particle i whereas fig. 4 (c) shows the layout of another memory portion in the first region of a SDRAM bank storing the coordinates $\mathbf{r}_j = (r_x, r_y, r_z)$ and electric charge q_j of a j particle, as well as the interaction parameters ε_{ij}, σ^{-2}_{ij} and the cutoff distances for both Lennard-Jones r^{LJ}_{ij} and Coulombic r^{C}_{ij} interactions, all pertaining to the particular pair of i and j particles. Since the logical width of the memory banks is 128 bits, the layouts in fig. 4 (b) and (c) occupy the space of 1 and 2 logical words, respectively. Furthermore, the layout of a memory portion in the second region of a SDRAM bank storing the force $\mathbf{f}_{ij} = (f_x,$

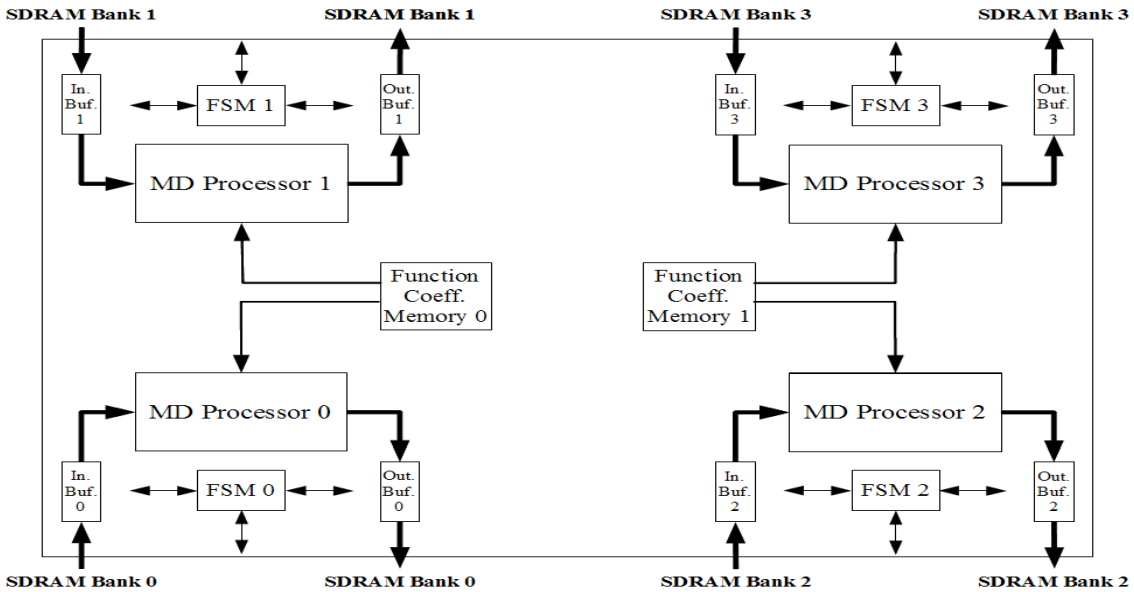

Fig. 3. Block diagram of our Molecular Dynamics processor core

f_y, f_z), Lennard-Jones potential e^{LJ}_{ij}, Coulombic potential e^{C}_{ij} and virial $\mathbf{v}_{ij} = (v_x^2, v_y^2, v_z^2, v_{xy}, v_{xz}, v_{yz})$ values computed for the specific pair of i and j particles is displayed in fig. 4 (d) which takes up the space of 3 logical words in a memory bank.

Fig. 5 shows the functional block diagram of our MD processor. It contains a pipeline comprised of three major functional units, namely *MD Squared Distance* unit, *MD Calculation* unit and *MD Force/Virial/Potential* unit in the order presented. MD processor whose operating frequency is 150 MHz calculates the non-bonded interactions in the simulated molecular system as stated above. Primary duty of *MD Squared Distance* unit is to calculate the squared distance between an i particle and the j particles in the neighbour list of that i particle. On the other hand, *MD Calculation* unit is to used to calculate several values which are then passed to the *MD Force/Virial/Potential* unit for the computations of the pairwise forces, virials and potentials acting on an i particle due to both Lennard-Jones and Coulombic interactions with the j particles in the neighbour list of that i particle. Note that only the short-range portion of the Coulombic interactions is computed in our MD processor.

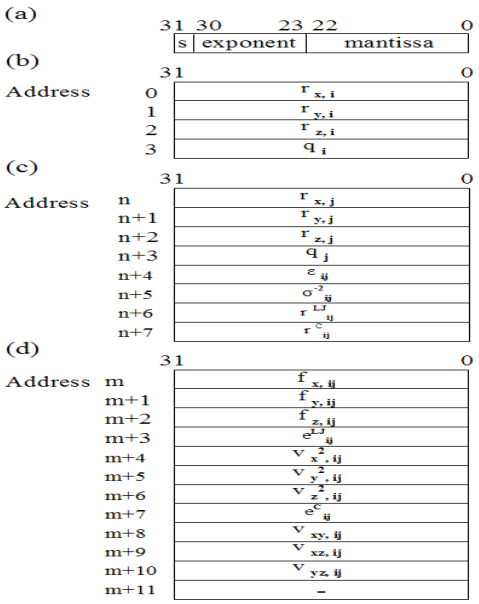

Fig. 4. (a) Internal format of the numbers used in our design (b) Layout of a memory portion in the first region of a SDRAM bank (c) Layout of another memory portion in the first region of a SDRAM bank (d) Layout of a memory portion in the second region of a SDRAM bank

5. Implementation results

Molecular Dynamics simulations were implemented on the Alpha Data nodes of the Maxwell machine with the MD processor cores shown in fig. 3, each of which incorporating four MD processors working independently in parallel with a total pipeline latency of 74 clock cycles. Our MD core was written in Verilog language while the interfaces of the user FPGA with the local bus and the DDR2 SDRAM banks were provided by the Alpha Data in the VHDL language. The design was then synthesized, placed, and routed by the Xilinx ISE 11.5 tool. FPGA bitstreams were also generated by the same tool while the ModelSim tool was employed to test the MD core with a number of testbenches. Note that there is only one FPGA bitstream used to configure all FPGAs in the MD machine regardless of the number of atoms in the simulated system. Furthermore, MATLAB tool was used to compute the piecewise polynomial interpolation coefficients for the evaluation of the several functions needed.

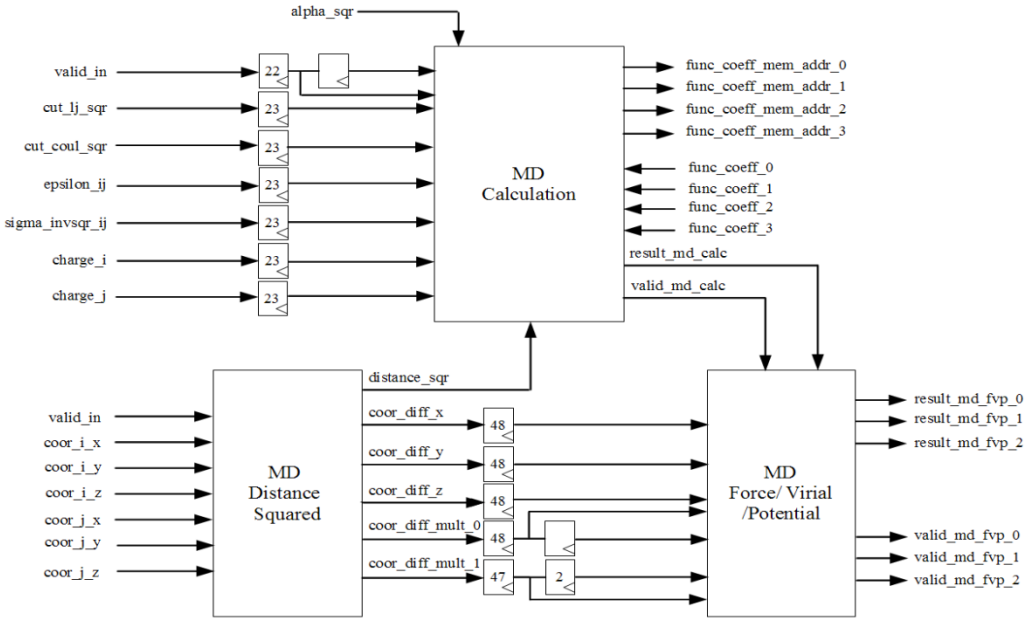

Fig. 5. Functional block diagram of a Molecular Dynamics processor

The clock frequency of the user FPGAs for the local bus interface was set to be 80 MHz whereas the clock frequency for the MD core was set to be 150 MHz. Due to this clock frequency of the MD core, the clock frequency for the DDR2 SDRAM banks was 300 MHz. For benchmarking purposes, an all-atom Rhodopsin protein in solvated lipid bilayer was simulated with the Lennard-Jones forces, and the Coulombic forces via PPPM (particle-particle particle mesh), incorporating SHAKE constraints. This model contains counter-ions and a reduced amount of water to make a 32K atom system.

Table 1 presents the timing performance figures of the LAMMPS software for the pairwise LJ and short-range Coulombic interaction computations of the above mentioned Rhodopsin protein system on two nodes of the Maxwell machine (i.e. two software processes running on one host Intel Xeon CPU). The protein system was replicated in X, Y or Z dimensions to achieve the simulation of systems with up to 256,000 atoms, as presented in table 1. For comparative purposes, table 1 also shows the timing performance figures of the MD machine configured to operate in the same way as the pure software implementation on two nodes of the Maxwell machine (i.e. two software processes running on one host Intel Xeon CPU and MD core instances on two Xilinx Virtex-4 XC4VFX100 FPGAs [15]). Note that the timing figures for the MD machine presented in table 1 do not include the I/O communication costs occurring during the data transfers between a host CPU and SDRAM banks.

Table 1. Timing figures of the LAMMPS software and the MD machine for the pairwise interaction computations on two Maxwell nodes for one time-step

No. of Atoms	SW Comp. Time (s)	MD Machine Comp. Time (s)
32000	5.0513	0.3793
64000	9.9114	0.7859
128000	20.4078	1.6742
256000	40.7469	3.1309

Table 2 provides the speed-up values of the MD machine over the pure software implementation (LAMMPS) for the pairwise interaction computations of the systems with various numbers of atoms on two Maxwell nodes. Note that the MD machine outperforms the pure software implementation by 12x-13x.

Table 2. LAMMPS versus MD machine speed-up values for the interaction computations on two Maxwell nodes

No. of Atoms	MD Machine Speed-Up
32000	13.32
64000	12.61
128000	12.19
256000	13.01

Table 3 shows the timing performance figures of the MD machine on two Maxwell nodes, this time including the I/O communication costs of the data transfers between a host CPU and SDRAM banks, as opposed to table 1. As it can be seen, I/O communication times account for over 96 percent of the total time. Due to this very high cost of the communication, the overall timing performance of the MD machine is poor compared to the pure software implementation for all atom systems (refer to table 1). Although multithreading, where each of the four existing threads deals with its assigned MD processor in the MD core, was utilised in the software process to take advantage of the direct memory transfers (DMA), the total time could not be reduced to a desired level because of the very poor data bandwidth between the host CPU and SDRAM banks. Nonetheless this limitation is not conceptual but rather dependent on the hardware platform targeted in this implementation. This communication bottleneck can be significantly resolved by integrating FPGA boards tighter into the host systems. In this way, FPGAs will have high-speed access to host memory through, for instance, AMD's Hypertransport, Intel's Quick Path Interconnect or SGI's NumaLink which offer bandwidths ranging from 15 to 25.6 GB/s, thus reducing communication overheads by at least 50x in comparison to our currently used communication link. This would result in performance gains of the MD machine in overall over the pure software implementation by almost same factors listed in table 2.

Table 3. Timing figures of the MD machine including I/O communication costs on two Maxwell nodes for one time-step

No. of Atoms	Total Time (s)	I/O Comm. Time (s)	Pct. of I/O Comm.
32000	11.2021	10.8228	% 96.61
64000	22.2986	21.5127	% 96.48
128000	44.7494	43.0752	% 96.26
256000	89.5814	86.4506	% 96.50

Tables 4 and 5 show the comparative timing figures of the pure software implementation and the MD machine for the pairwise interaction computations of the Rhodopsin protein systems with up to over two million atoms on 8 and 16 nodes of the Maxwell machine, respectively. As it can be seen, the MD machine speed-up values for the pairwise interaction computations range from 10x to 14x. The total number of the floating-point adder/subtractors in the MD core is 36 while the total number of the floating-point multipliers is 44. In addition, 8 floating-point comparators are also utilised in our design. Furthermore, the floating-point adder/subtractors and comparators were entirely implemented in the slice logic. On the other hand, the floating-point multipliers were partially implemented in the DSP48 blocks on the FPGA. However, since each multiplier requires 4 DSP48 blocks and the total number

of the DSP48 blocks in the user FPGA is just 160, it was only possible to map 40 of the multipliers to the DSP48 blocks while the rest of them were entirely implemented in the user logic.

Table 4. Comparative timing figures of the LAMMPS software and the MD machine on eight Maxwell nodes for one time-step

No. of Atoms	SW Comp. Time (s)	MD Machine Comp. Time (s)	MD Machine Speed-Up
32000	1.20	0.12	10.16
64000	2.44	0.22	11.07
128000	4.86	0.39	12.38
256000	9.86	0.78	12.61
512000	19.56	1.52	12.90
1024000	40.57	2.85	14.21

Table 5. Comparative timing figures of the LAMMPS software and the MD machine on sixteen Maxwell nodes for one time-step

No. of Atoms	SW Comp. Time (s)	MD Machine Comp. Time (s)	MD Machine Speed-Up
32000	0.61	0.06	10.71
64000	1.21	0.11	10.87
128000	2.41	0.24	10.00
256000	4.97	0.44	11.24
512000	9.78	0.77	12.64
1024000	19.68	1.52	12.98
2048000	40.29	2.89	13.93

The accuracy in the computations was sufficient enough to carry out stable MD simulations but the accuracy could be improved if the single extended precision (i.e. width of 40-bit) was used for the floating-point numbers inside the design rather than the single precision (i.e. width of 32-bit) [16]. However, this precision increase would require higher amounts of slice logic and DSP48 blocks to implement the floating-point arithmetic units utilised in the MD core. Unfortunately, currently used Xilinx Virtex-4 XC4VFX100 FPGA chips cannot accommodate any higher resource demand.

6. Concluding remarks

The design and implementation of a FPGA core, namely MD core, carrying out all the necessary operations to compute the non-bonded interactions in a MD simulation with the purpose of accelarating the LAMMPS MD software was presented in this paper. Our MD processor core comprised of 4 identical pipelines working independently in parallel to evaluate the non-bonded potentials, forces and virials was implemented on the nodes of a FPGA-based supercomputer, named Maxwell, which consists of 64 Virtex-4 FPGA chips. This implementation allowed us to produce a special-purpose parallel machine for the hardware acceleration of the MD simulations.

The timing performance figures of the MD machine for the pairwise LJ and short-range Coulombic (via PPPM) interaction computations in the MD simulations of the solvated Rhodopsin protein systems with various numbers of atom show performance gains over the pure software implementation by factors of up to 13 on two nodes of the Maxwell machine. These MD machine speed-up values for the pairwise interaction computations were also maintained on different numbers of Maxwell nodes. However, the overall timing performance of the MD machine is worse than the pure software implementation due to the very high I/O communication costs of the data transfers between a host CPU and SDRAM banks. Nonetheless, if FPGA boards are integrated tighter into the host systems through, for instance, AMD's Hypertransport, Intel's Quick Path Interconnect or SGI's NumaLink, the bandwidth of the I/O communications would be greatly enhanced up to 25.6 GB/s, thus yielding much lower communication costs (i.e. at least 50x reduction in comparison with our currently used communication link). This would result in performance gains of the MD machine in overall over the pure software implementation. On the other hand, the accuracy of the computations could be improved if the number of slices and DSP48 blocks available in the user FPGA was higher.

7. References

[1] M. P. Allen and D. J. Tildesley, *Computer Simulation of Liquids*, Oxford University Press, 1987.

[2] G. D. Fasman, *Prediction of Protein Structure and the Principles of Protein Conformations*, Plenum Press, New York, 1989.

[3] Z. R. Wasserman and C. N. Hodge, "Fitting an inhibitor into the active site of thermolysin: A molecular dynamics case study", *J. Proteins: Structure, Function, and Bioinformatics*, 24, pp. 227-237, 1996.

[4] N. L. Greenbaum, I. Radhakrishnan, D. J. Patel and D. Hirsh, "Solution structure of the donor site of a trans-splicing RNA", *J. Structure*, 4, pp. 725-733, 1996.

[5] R. Scrofano, M. B. Gokhale, F. Trouw and V. K. Prasanna, "Accelerating Molecular Dynamics Simulations with Reconfigurable Computers", *IEEE Trans. on Parallel and Distributed Systems*, 19, pp. 764-778, 2008.

[6] Y. Gu, T. VanCourt and M. C. Herbordt, "Improved interpolation and system integration for FPGA-based molecular dynamics simulations", *International Conference on Field Programmable Logic and Applications*, pp. 21-28, 2006.

[7] M. Chiu and M. C. Herbordt, "Efficient particle-pair filtering for acceleration of molecular dynamics simulation", *In-*

ternational Conference on Field Programmable Logic and Applications, pp. 345-352, 2009.

[8] LAMMPS manual, http://lammps.sandia.gov/doc/Manual.html.

[9] R. Baxter Et al., "Maxwell—a 64 FPGA supercomputer", *NASA/ESA Conference on Adaptive Hardware Systems*, pp. 287-294, 2007.

[10] D. C. Rapaport, *The Art of Molecular Dynamics Simulation*, Cambridge University Press, New York, 2004.

[11] P. P. Ewald, "Evaluation of optical and electrostatic lattice potentials", *Ann. Phys Leipzig*, 64, pp. 253-287, 1921.

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

[13] OpenCores Floating Point Adder and Multiplier, http://opencores.org/project,fpuvhdl.

[14] G Marcus, P. Hinojosa, A. Avila and J. N. Flores, "A Fully Synthesizable Single-Precision, Floating-Point Adder/Subtractor and Multiplier in VHDL for General and Educational Use", *IEEE International Conference on Devices, Circuits and Systems*, pp. 319-323, 2004.

[15] Xilinx Virtex-4 datasheets, http://www.xilinx.com/products/silicon_solutions/fpgas/virtex/ virtex4/index.htm.

[16] T. Amisaki, T. Fujiwara, A. Kusumi, H. Miyagawa and K. Kitamura, "Error evaluation in the design of a special-purpose processor that calculates nonbonded forces in molecular dynamics simulations", *J. Computational Chemistry*, 16, pp. 1120-1130, 1995.