

Towards Production FPGA-Accelerated Molecular Dynamics: Progress and Challenges

Matt Chiu Martin C. Herbordt

Computer Architecture and Automated Design Laboratory
Department of Electrical and Computer Engineering
Boston University; Boston, MA 02215

Abstract—Recent work in the FPGA acceleration of molecular dynamics simulation has shown that including on-the-fly neighbor list calculation (particle filtering) in the device has the potential for an $80\times$ per core speed-up over the CPU-based reference code and so to make the approach competitive with other computing technologies. In this paper we report on progress and challenges in advancing this work towards the creation of a production system, especially one capable of running on a large-scale system such as the Novo-G. The current version consists of an FPGA-accelerated NAMD-lite running on a PC with a Gidel PROCStar III. The most important implementation issues include software integration, handling exclusion, and modifying the force pipeline. In the last of these we have added support for Particle-Mesh-Ewald and augmented the Lennard-Jones calculation with a switching function. In experiments, we find that energy stability so far appears to be acceptable, but that longer simulations are needed. Due primarily to the added complexity of the force pipelines, performance is somewhat diminished from the previous study; we find, however, that porting to a newer (existing) device will more than compensate for this loss.

I. INTRODUCTION

Molecular dynamics simulation (MD) is a central method in high performance computing (HPC) with applications throughout engineering and natural science. *Acceleration* of MD is a critical problem — there is a many order-of-magnitude gap between the largest current simulations and the potential physical systems to be studied. To this end, GPUs are currently receiving much attention (e.g., [1], [2], [3]), along with dedicated hardware (see, e.g., the Anton computer from D.E. Shaw [4]). Although there have been many FPGA implementations (e.g., [5], [6], [7], [8], [9], [10], [11]), the heavy use of floating point in MD appeared to make these less competitive. In recent work [12], [13], however, we have shown that when FPGAs perform particle filtering (on-the-fly neighbor list generation), much of the floating point computation can be eliminated. As a result, FPGA-based MD acceleration is not only viable, but when power considerations are factored in, potentially an excellent fit.

In this paper we report on progress towards the goal of extending this work into a production FPGA-accelerated MD system. In particular, we are interested in eventually supporting large scale simulations, especially to be run on Reconfigurable Computing (RC) systems such as the Novo-G at the University

of Florida [14]. The Novo-G currently has 196 Altera Stratix-III E260 FPGAs with the capability of 1 Peta-Op (32 bit integer) while drawing less than 10K Watts of power. It is also projected to be continually upgraded, with 56 Stratix-IV E530s being installed in early 2011.

This paper describes progress in several areas.

1. We have integrated our multi-level MD force pipelines into NAMD-lite [15] (as a precursor to integration into NAMD [16], [17]). We describe methods, interfaces, data conversions, and exclusion.
2. Our MD force pipelines have been extended to now support the short-range part of the Particle Mesh Ewald method of computing the electrostatic potential (in addition to the Multigrid method previously implemented [13]). This has necessitated a substantial redesign: we now use table look-up with interpolation [7] rather than direct computation [18]. In addition, in order to improve energy fluctuation we have added a switching function to the van der Waals calculation.
3. These implementations have been completed and are currently running on one FPGA of a Gidel PROCStar III quad FPGA board [19], the per-node accelerator of the Novo-G. The FPGA is an Altera Stratix-III 260E [20]. We describe the current system as well as some of the issues in getting from working simulations to a working real system.
4. We have profiled this system. We describe areas where performance has diminished from the post Place-and-Route timing and the reasons why. We also project performance to more recent technologies, the Stratix-IV and Stratix-V, and other possible ways in which performance can be improved.
5. Substantial tuning is possible. We show preliminary results with respect to trade-offs between interpolation order and simulation quality. We present initial comparisons with respect to Energy Fluctuation among some of these alternatives and with respect to NAMD-lite and NAMD.
6. We have performed initial analysis with respect to methods of scaling to multiple nodes, e.g., how various numbers of nodes could be used to simulate a large benchmark such as STMV with over 1M atoms. We find that there are likely to be interesting differences in load balancing between FPGA-based systems (with multi-level force pipelines) and other technologies.

The rest of this article is organized as follows. We begin with background on MD and our current multi-level short-range force accelerator. We follow that with a description

This work was supported in part by the NIH through award #R01-RR023168-01A1. Web: www.bu.edu/caadlab. Email: {herbordt|matthiu}@bu.edu

of the issues in moving from ModelSim to full hardware and software integration. In the next section we present and analyze performance and simulation quality. We conclude with work in progress and a brief discussion.

II. MD PRELIMINARIES

This section provides background and is based on material appearing in [13].

A. MD Review

MD is an iterative application of Newtonian mechanics to ensembles of atoms and molecules (see, e.g., [21] for details). MD simulations generally proceed in iterations each of which consists of two phases, force computation and motion integration. In general, the forces depend on the physical system being simulated and may include LJ, Coulomb, hydrogen bond, and various covalent bond terms:

$$\mathbf{F}^{total} = F^{bond} + F^{angle} + F^{torsion} + F^{HBond} + F^{non-bonded} \quad (1)$$

Because the hydrogen bond and covalent terms (bond, angle, and torsion) affect only neighboring atoms, computing their effect is $O(N)$ in the number of particles N being simulated. The motion integration computation is also $O(N)$. Although some of these $O(N)$ terms are easily computed on an FPGA, their low complexity makes them likely candidates for host processing, which is what we assume here. The LJ force for particle i can be expressed as:

$$\mathbf{F}_i^{LJ} = \sum_{j \neq i} \frac{\epsilon_{ab}}{\sigma_{ab}^2} \left\{ 12 \left(\frac{\sigma_{ab}}{|r_{ji}|} \right)^{14} - 6 \left(\frac{\sigma_{ab}}{|r_{ji}|} \right)^8 \right\} \mathbf{r}_{ji} \quad (2)$$

where the ϵ_{ab} and σ_{ab} are parameters related to the types of particles, i.e. particle i is type a and particle j is type b . The Coulombic force can be expressed as:

$$\mathbf{F}_i^C = q_i \sum_{j \neq i} \left(\frac{q_j}{|r_{ji}|^3} \right) \mathbf{r}_{ji} \quad (3)$$

A standard way of computing the non-bonded forces (Lennard-Jones or LJ and Coulombic) is by applying a cut-off. Then the force on each particle is the result of only particles within the cut-off radius r_c . Since this radius is typically less than a tenth of the size per dimension of the system under study, the savings are tremendous, even given the more complex bookkeeping required.

The problem with cut-off is that, while it may be sufficiently accurate for the rapidly decreasing LJ force, the error introduced in the slowly declining Coulombic force may be unacceptable. A number of methods have been developed to address this issue with some of the most popular being based on Ewald Sums (see, e.g., [22]) and multigrid (see, e.g., [23], [24]). Here we use the standard convention of calling *short-range* the LJ force and the Coulombic force generated within a cut-off radius. We refer to the Coulombic force generated outside the cut-off radius as *long-range*. Since the long-range force computation is generally a small fraction of the total (see, e.g., [25], [10]), we concentrate here on the short-range force.

B. Short-Range Force Computation

As just described, the short-range force computation has two parts, the LJ force and the rapidly converging part of the Coulomb force. The LJ force is often computed with the so-called 6-12 approximation given in Equation 2. This has two terms, the repulsive Pauli exclusion and the van der Waals attraction. Both require coefficients specific to the component particles of the particle pair whose interaction is being evaluated. These can be combined with the other constants (physical and scaling) and stored in coefficient look-up tables. Thus the LJ force can be expressed as

$$\frac{\mathbf{F}_{ji}^{LJ}(r_{ji}(a,b))}{\mathbf{r}_{ji}} = A_{ab}|r_{ji}|^{-14} + B_{ab}|r_{ji}|^{-8} \quad (4)$$

where A_{ab} and B_{ab} are distance-independent coefficient look-up tables indexed with atom types a and b .

Returning now to the Coulomb force computation, we begin by rewriting equation 3 as

$$\frac{\mathbf{F}_{ji}^{CL}(r_{ji}(a,b))}{\mathbf{r}_{ji}} = QQ_{ab}|r_{ji}|^{-3}, \quad (5)$$

where QQ_{ab} is a precomputed parameter (analogous to A_{ab} and B_{ab}). Because applying a cut-off here often causes unacceptable error, and also because the all-to-all direct computation is too expensive for large simulations, various numerical methods are applied to solve the Poisson equation that translates charge distribution to potential distribution. To improve approximation quality and efficiency, these methods split the original Coulomb force curve in two parts (with a smoothing function $g_a(r)$): a fast declining short range part and a flat long range part. For example:

$$\frac{1}{r} = \left(\frac{1}{r} - g_a(r) \right) + g_a(r). \quad (6)$$

The short range component can be computed together with Lennard-Jones force using a third look-up table (for QQ_{ab}). The entire short range force to be computed is:

$$\frac{\mathbf{F}_{ji}^{short}}{\mathbf{r}_{ji}} = A_{ab}r_{ji}^{-14} + B_{ab}r_{ji}^{-8} + QQ_{ab}(r_{ji}^{-3} + \frac{g'_a(r)}{r}). \quad (7)$$

C. Computing Short-Range Forces with Table Look-up

Since these calculations constitute the “inner loop,” considerable care is taken in their implementation. A major consideration is whether to compute them directly or to use table look-up with interpolation. We now briefly describe the latter method and in particular how we have implemented the force pipeline with three tables, one each for r^{-14} , r^{-8} and $r_{ji}^{-3} + \frac{g'_a(r)}{r}$ [26]. Equation 7 can be rewritten as a function of r_{ji}^2 :

$$\frac{\mathbf{F}_{ji}^{short}(|r_{ji}|^2(a,b))}{\mathbf{r}_{ji}} = A_{ab}R_{14}(|r_{ji}|^2) + B_{ab}R_8(|r_{ji}|^2) + QQ_{ab}R_3(|r_{ji}|^2) \quad (8)$$

where R_{14} , R_8 , and R_3 are lookup tables indexed with $|r_{ji}|^2$.

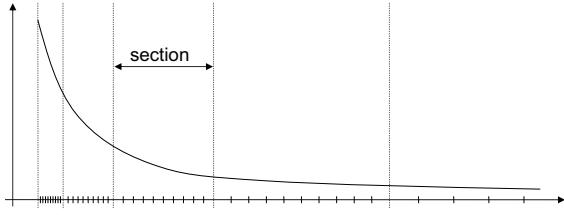


Fig. 1. Table look-up varies in precision across r^{-k} . Each section has a fixed number of *intervals*.

The intervals in the tables are represented in Figure 1. Each curve is divided into several sections along the X-axis such that the length of each section is twice that of the previous. Each section, however, is cut into the same number of intervals N . To improve the accuracy, higher order terms can be used. When the interpolation is order M , each interval needs $M + 1$ coefficients, and each section needs $N * (M + 1)$ coefficients:

$$F(x) = a_0 + a_1x + a_2x^2 + a_3x^3 \quad (9)$$

shows third order with coefficients a_i . Accuracy increases with both the number of intervals per section and the interpolation order. These issues are discussed in detail in [26].

D. Cell and Neighbor Lists

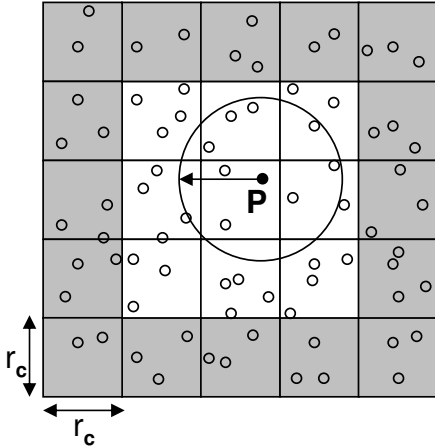


Fig. 2. Shown is part of the simulation space about particle P. Its two dimensional *cell neighborhood* is shown in white; cells have edge size equal to the cut-off radius. The cut-off circle is shown; particles within the circle are in P's neighbor list.

While MD in general involves all-to-all forces among particles, a cut-off is commonly applied to restrict the extent of the short-range force to a fraction of the simulation space. Two methods are used to take advantage of this cut-off: cell lists and neighbor lists (see Figure 2). With cell lists, the simulation space is typically partitioned into cubes with edge-length equal to r_c . Non-zero forces on the *reference particle P* can then only be applied by other particles in its *home cell* and in the 26 neighboring cells (the $3 \times 3 \times 3$ *cell neighborhood*). We refer the second particle of the pair as the *partner particle*. With neighbor lists, P has associated with it a list of exactly those partner particles within r_c . We now compare these methods.

- **Efficiency.** Neighbor lists are by construction 100% efficient: only those particle pairs with non-zero mutual force are evaluated. Cell lists as just defined are 15.5% efficient with that number being the ratio of the volumes of the cut-off sphere and the 27-cell neighborhood.
- **Storage.** With cell lists, each particle is stored in a single cell's list. With neighbor lists, each particle is typically stored in 400-1000 neighbor lists.
- **List creation complexity.** Computing the contents of each cell requires only one pass through the particle array. Computing the contents of each neighbor list requires, naively, that each particle be examined with respect to every other particle: the distance between them is then computed and thresholded. In practice, however, it makes sense to first compute cell lists anyway. Then the neighbor lists can be computed using only the particles in each reference particle's cell neighborhood.

From this last point, it appears that the creation of neighbor lists involves not only cell lists, but also a fraction of the force computation itself. At this point, the question arises whether or not to finish computing the forces of those particles that are within the cut-off and whether they neighbor lists should be saved.

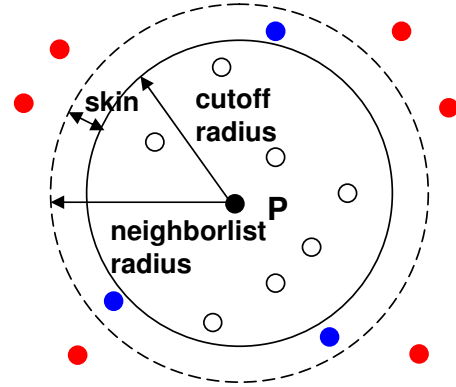


Fig. 3. Neighborlists are often computed for a larger radius than the cutoff.

Most MD codes reuse the neighbor lists for multiple iterations and so amortize the work in their creation. But because particles move during each iteration, particles can enter and exit the cut-off region leading to potential error. The solution is to make the neighborlist cut-off larger than the force cut-off, e.g., 13.5\AA versus 12\AA (see Figure 3). There is a trade-off between the increase in neighborhood size, and thus the number of particle pairs evaluated, and the number of iterations between neighbor list updates.

III. MD SYSTEM DESIGN

We briefly state our assumptions about the target High Performance Reconfigurable Computing (HPRC) architecture. The overall system consists some number of nodes in a typical heterogeneous configuration with a high-end microprocessor and an accelerator board plugged into a high-speed socket (e.g., PCI Express). The processor runs the main application

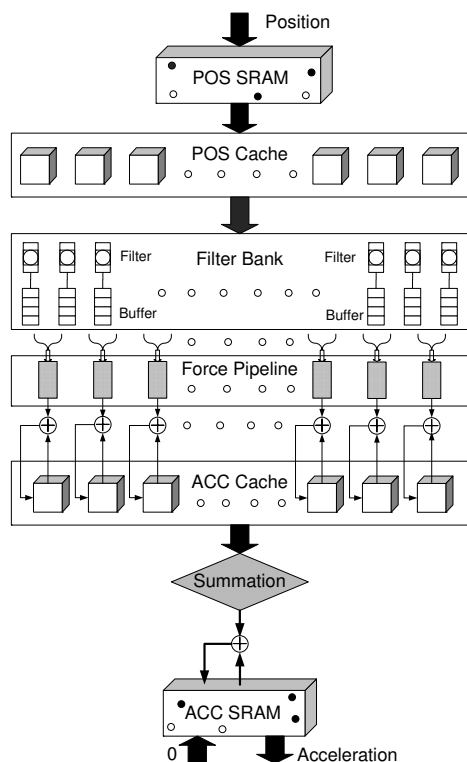


Fig. 4. Schematic of the HPRC MD system.

program and communicates with the accelerator through function calls. The accelerator board consists of a high-end FPGA, memory, and a bus interface. On-board memory is tightly coupled to the FPGA through several interfaces, either real or virtual (e.g., 6 x 32-bit).

The program is partitioned as follows. The accelerators process the short-range force, while the processors process the balance of the computation. Each iteration, new particle positions are downloaded to the accelerator and forces are uploaded to the processor. Some conversion of data may be necessary during the transfers. If the conversion is simple, such as between standard floating point formats, then it is performed on the processor; if it is between non-standard formats (as in [26]) then it is performed on the accelerator. In either case, conversion adds little latency. Cell lists are computed on the host and particle data is transferred to the accelerator by cell. Neighbor lists are computed on-the-fly by the accelerator in a process called particle filtering.

We now give an overview of the overall accelerator design (see Figure 4); for details please see [13].

Main computation pipeline

The main computation pipeline is partitioned into two levels. The first is the filter pipeline; it determines whether the particle pair has a non-zero force. The second level, the force pipeline, accepts the particle pairs that pass the filter and computes their mutual force. Four to eight force pipelines fit on the 260E, each

with 8-10 filter pipelines.

Host-accelerator data transfers

At the highest level, processing is built around the timestep iteration and its two phases: force calculation and motion update. During each iteration, the host transfers position data to, and acceleration data from, the coprocessor's on-board memory (POS SRAM and ACC SRAM, respectively).

Board-level data transfers

Force calculation is built around the processing of successive home cells. Position and acceleration data of the particles in the cell set are loaded from board memory into on-chip caches, POS and ACC, respectively. When the processing of a home cell has completed, ACC data is written back. Focus shifts and a neighboring cell becomes the new home cell. Its cell set is now loaded; in our current scheme this is usually nine cells per shift. The transfers are double buffered to hide latency.

Force pipelines to ACC cache.

To support an optimization due to Newton's Third Law, two copies are made of each computed force. One is accumulated with the current reference particle. The other is stored by index in one of the large BRAMs on the Stratix-III.

IV. IMPLEMENTATION: FROM SIMULATION TO PRODUCTION

A. NAMD-lite Integration

A number of MD software packages have been developed and widely used in the community, each with its own features and goals. NAMD is a MD simulation package which is highly regarded for its high scalability and parallel efficiency. It was written in C++ with Charm++ parallel objects and can be scaled up to hundreds of processors on a high-end parallel system. Despite of its excellent reputation and popularity, extra complications may be introduced to the development due to its complex parallel structure. NAMD-lite [15] is a prototyping framework whose purpose is to simplify and smooth the development process and to provide a way to examine and validate new features before integrating them into NAMD.

From a programming standpoint, NAMD-lite integration has been straightforward. The tasks are replacement of the short-range force computation with the appropriate accelerator calls, data conversion from double precision floating point to single precision and back again, packing and unpacking the data, and handling particle exclusion.

We now give some details of the data transfer and handling particle exclusion. On transfer to the accelerator, particles are grouped together by cell ID. The information that must be included in the transfer are the particles' position and properties (charge and type), and also the cell-list data itself: this last enables cell-level phases in the accelerator.

Particle exclusion refers to the necessity of not computing the non-bonded forces on bonded particles. To support this feature, our solution is to apply a short cut-off to the non-bonded force calculations based on the fact that two non-bonded particles generally cannot be too close to each other. Therefore, two particles within a certain short distance must

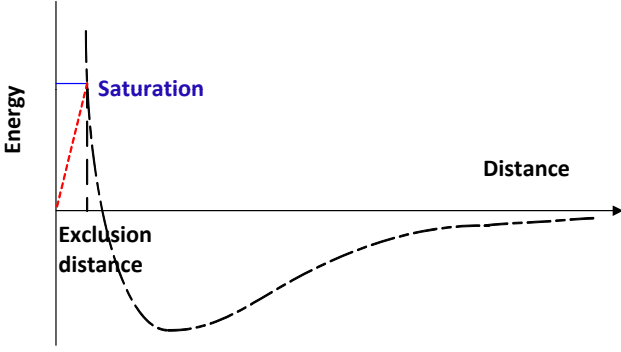


Fig. 5. Graph shows van der Waals interaction with cut-off check with saturation force.

be bonded. The short cut-off distance can be easily calculated by solving the inequality $F_{short} < range$, where $range$ is the dynamic range with a reasonable force value. The left term of the inequality is dominated by the 14 term. Multiple short cut-off values are required as this depends on the particle type. A simple graph is shown in Figure 5 to demonstrate this concept.

If the exclusion cutoff is chosen conservatively, two particles would be bonded as long as their intra-distance is smaller than the exclusion distance. For bonded particle pairs whose intra-distance is larger than the exclusion cutoff, the non-bonded force is subtracted in the host. Since the exclusion distance check in FPGA is performed in integer arithmetic while it is done in double precision in the host, an inconsistency may occur when the distance between two particles is very close to the exclusion cutoff. In order to minimize the impact of this inconsistency, a saturation force is applied if the intra-distance between two particles is smaller than the exclusion cutoff, as shown by the horizontal line.

Another enhancement is to scale the saturation forces down with distance, as shown by the diagonal dashed line. This can help avoid overflow in force accumulation step and improve precision accuracy. This final feature has not yet been implemented.

B. Force Pipelines

In Section 2.2 we described the general methods involved in computing the short-range force. Here we describe issues in their actual implementation.

Van der Waals Energy/Force

While the van der Waals term shown in Equation 4 converges quickly, it must still be modified for effective MD simulations. In particular, a switching function is implemented to truncate van der Waals force smoothly at the cutoff distance (see Equations 10-12).

$$s = (cutoff^2 - r^2)^2 * \quad (10)$$

$$(curoff^2 + 2 * r^2 - 3 * switch_dist^2) * denom$$

$$ds_r = 12 * (cutoff^2) * (switch_dist^2 - r^2) * denom \quad (11)$$

$$denom = 1/(cutoff^2 - switch_dist^2)^3 \quad (12)$$

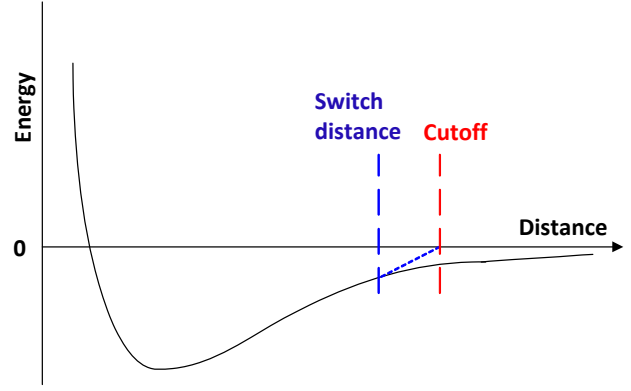


Fig. 6. Graph shows van der Waals potential with switching/smoothing function.

Without switching/smoothing function, the energy may not be conserved as the force would be truncated abruptly at the cutoff distance. The graph of van der Waals potential with the switching/smoothing function is illustrated in Figure 6.

The van der Waals force and energy are computed directly in single precision floating point format as shown here:

$$\begin{aligned} \text{IF } (r^2 \leq switch_dist^2) \quad U_{vdW} = U, F_{vdW} = F \\ \text{IF } (r^2 \leq switch_dist^2 \ \&\& \ r^2 < cutoff^2) \\ \quad U_{vdW} * s, F_{vdW} = F * s + U_{vdw} * ds_r \\ \text{IF } (r^2 \geq cutoff^2) \quad U_{vdW} = 0, F_{vdW} = 0 \end{aligned}$$

Electrostatic Energy/Force

The most flexible method in NAMD-lite of calculating the electrostatic force/energy is Particle Mesh Ewald (PME) and this we now support. PME is widely used to evaluate the standard Ewald Sums due to its computational efficiency. It approximates the long range part of the Ewald Sums by a discrete convolution on an interpolation grid; this can be performed using a discrete 3D Fast Fourier Transform (FFT).

As previously with multigrid, the short-range part of PME is accelerated in FPGA while long-range part is evaluated in the host. The short-range part of PME, E_s , is shown below.

$$E_s = \frac{1}{4\pi\epsilon_0} \frac{1}{2} \sum_n \sum_{i=1}^N \sum_{j=0}^n \frac{q_i q_j}{|r_i - r_j + nL|} \text{erfc}\left(\frac{|r_i - r_j + nL|}{\sqrt{2}\sigma}\right) \quad (13)$$

Since the E_s computation contains the evaluation of the complementary error function (erfc), which is expensive in FPGA logic, we use polynomial interpolation rather than direct computation. The polynomial coefficients were computed using Matlab by finding the coefficients of a polynomial $p(x)$ of degree n that fits the data, $p(x(i))$ to $y(i)$, in terms of least squares. Energy conservation is used to measure the quality of approximation for various polynomial interpolation orders (as shown below).

C. Implementation Details

Our MD acceleration solution has been successfully implemented and is currently running on one FPGA of a Gidel

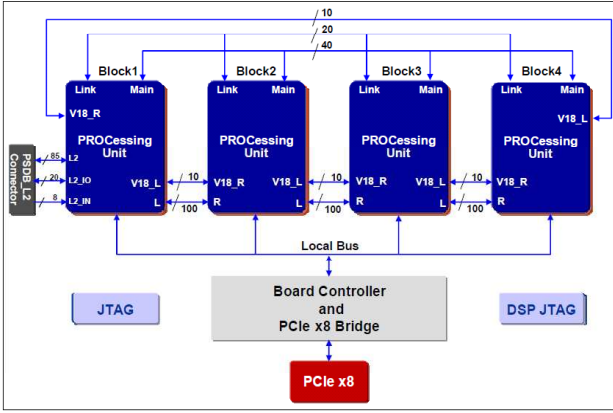


Fig. 7. PROCStar III System Overview (from Gidel PROCStar III User’s Guide)

TABLE I
PROCSTAR III MEMORY CONFIGURATION.

	Bank A on-board	Bank B SODIMM	Bank C SODIMM
Size 32-bit paths	256MB x 4	2GB x 4	2GB x 4
Perf. (DDR)	667 MHz	667 MHz	360 MHz
Throughput	16 GB/s	16 GB/s	8.5 GB/s

PROCStar III board, a single node of Novo-G. The PROCStar III is a PCI based system with an 8-lane PCI Express (PCIe x8) host interface (see Figure 7). Each processing unit contains an Altera Stratix III SE260 FPGA, three memory banks (see Table I), and connections to the other FPGAs.

The heterogeneous memory design leads to some design issues. For example, in order to prevent bank C from becoming a bottleneck, data are allocated to the different banks by type: Bank C stores the particle type, Bank A stores the particle positions and charge, and Bank B holds the forces.

V. RESULTS: PERFORMANCE AND QUALITY

A. Initial Performance Profile

For the results in this and the following subsection we refer to the NAMD benchmark NAMD2.6 on ApoA1. It has 92,224 particles, a bounding box of $108\text{\AA} \times 108\text{\AA} \times 78\text{\AA}$, and a cut-off radius of 12\AA .

In previous work [12], [13], we reported an expected running time of 22ms per iteration for the accelerator execution of the short-range force. This number is based on the following assumptions:

- ModelSim simulations and post place-and-route area and timing results, which indicate the following: 8 force pipelines, 9 filter pipelines per force pipeline, and an expected operating frequency of 196MHz.
- Software simulations which indicate a force pipeline efficiency of 95%.
- The FPGA logic required to interface with accelerator memory and I/O would be no more than 10% of the FPGA’s logic and memory resources.

- The electrostatic force would be computed using multi-grid.

Current measured times, on the configuration described in the previous subsection, yield a time of 70 ms per iteration. This reduction by a factor of more than $3\times$ is accounted for as follows.

- 1) **Interface/Peripheral Logic.** With highly complex devices having thousands of BRAMs and a dozen or more memory streams, hand-crafted interface logic is no longer viable. Interface packages provided by vendors (e.g., Gidel and BEEcube) fill the need extremely well, but also require substantially more FPGA resources than were used in previous generations.
- 2) **Pipeline Logic.** Two factors have made the pipeline logic substantially more complex. The first is implementing the L-J switching function, which was necessary to improve energy conservation. The second was moving from multigrid to PME. This has necessitate returning to our original design [26] which used polynomial interpolation.
- 3) **Pipeline Efficiency.** The assumptions we made for our software simulations did not accurately capture output conflicts and therefore the number of pipeline stalls.
- 4) **Phase Efficiency.** Particles in each home cell are processed in cohorts equal to the number of filter pipelines. For example, with 173 particles and 32 filters, there would be six phases. The final phase, however, runs at only 40% efficiency. We describe several methods for mitigating this issue [13], but they are not implemented in the current version.

The first two factors result in the number of force pipelines being reduced from 8 to 4 and the total number of filters, and thus the capacity, from 72 to 32. The operating frequency is currently about 190MHz. The third and fourth factors result in the efficiency dropping from 95% to about 80%. Substantial improvements are possible for the current device technology and, especially, by upgrading to the current generation FPGA; these are sketched in the next section.

B. Tuning and Simulation Quality

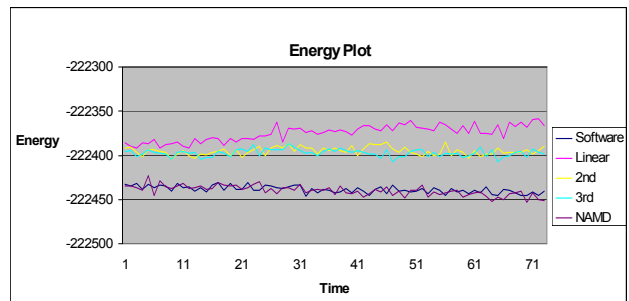


Fig. 8. Graph shows Energy Plot for various interpolation orders of the short-range part of PME van der Waals potential with switching/smoothing function. Comparisons are with NAMD and NAMD-lite. Time is in 100fs increments (up to 7.1ps). Results are summarized in Table 2.

TABLE II
TABLE SHOWS ENERGY DRIFT AND VARIANCE OF FPGA ACCELERATED
AND BASELINE CODES.

	NAMD	NAMD-lite	Linear	2nd Ord	3rd Ord
Slope	-0.1898	-0.1167	0.338	0.007	-0.0235
Std Dev	5.755	5.219	9.131	5.136	4.533

In order to validate and measure quality of our FPGA design, energy was plotted as a function of time (see Figure 8). In particular, we measure how energy is conserved for various implementations of the short-range part of PME. The Energy drift slope and standard deviation are presented in Table II. The results labeled NAMD and NAMD-lite are from those codes running on the processor only. The other results have the short-range forces computed on the accelerator using table look-up with polynomial interpolation with the order as shown. The time scale is in increments of 100fs.

These results are highly preliminary and the time scale is still too short to draw conclusions. The difficulty in generating longer time-scale simulations is that NAMD-lite is an unoptimized serial code and so each of these graphs (except NAMD) takes several hours to generate. Still we find these results promising: an implementation with 2nd or 3rd order polynomial interpolation could have good energy stability. We are in the process of generating Energy Plots on a much longer time scale as well as measuring the highly robust invariant, the Shadow Hamiltonian [27].

VI. WORK IN PROGRESS, DISCUSSION

A. Performance

TABLE III
ALTERA STRATIX FPGA OVERVIEW.

	Stratix III EP3SE260	Stratix IV EP4SE820	Stratix V 5SEBA	Stratix V 5SGSMB8
Equiv. LEs	203	650	1087	530
18x18 Mults	768	1024	1100	3510
Mem. (Mb)	1.5	2.3	4.3	3.6

One way to improve performance immediately is to convert the van der Waals computation from direct to polynomial interpolation. This should enable a 20% improvement in performance due to either an added pipeline or increased operating frequency. There is also substantial performance improvement possible with various low-level optimizations, but the engineering time might be better spent in creating a parallel design (below) and in porting the design onto the current generation FPGAs (see Table III).

We have ported the current design to the Stratix IV and, without device-specific changes, fit six pipelines (through post Place-and-Route). Discounting the anticipated increase in operating frequency, this reduces latency to under 40ms. Moving to the Stratix V allows for both immediate increase in performance due to scaling, but also potential for redesign. Depending on which device is selected there will be either twice as much logic or three times as many multipliers. The

first favors interpolation, the second direct computation. In either case, a conservative estimate gives 12 pipelines with 16 being viable. This should reduce the latency to under 20ms, again not counting the likely improvement in operating frequency.

B. Simulation Quality

FPGA modifications to incorporate the Shadow Hamiltonian have been completed and the initial data are being collected. Since energy drift can become a significant factor even after remaining stable for long periods, we are conducting long time-scale measurements. If there are significant differences with the original codes, there are various solutions: these include increasing the precision of all or part of the computation and increasing the interpolation order.

C. Scalability

We are investigating parallel versions of accelerated NAMD-lite. A version using all four FPGAs on the PROCStar III board has been created and tested. Its success depends on hiding the latencies of the data transfers.

Because of the transfer latencies, scalability to larger numbers of FPGAs, at least in the short term, depends on increasing the problem size. One example is another NAMD benchmark: the STMV (virus) with 1.07M atoms and a 216A x 216A x 216A simulation domain. With 12A cells, this yields an 18^3 cell configuration.

To reduce the frequency of the neighbor list calculations and data redistribution among nodes, NAMD uses two larger dimensions: the pair-list distance (typically 13.5A) and the patch dimension (typically 16A). Pair lists are commonly generated every 10 steps. Movement of atoms among nodes is highly optimized, but a common step is to move hydrogen groups (a heavy atom and all of its bonded hydrogen) among patches at the beginning of every cycle, which itself is a tunable parameter often set at 20.

One major benefit of the multi-level force pipeline in the FPGA-based acceleration is the on-the-fly neighborlist generation (particle filtering). Each filter pipeline requires less than 1/20th the logic of a force pipeline and no multipliers. Since there is currently a surplus of logic (with respect to multipliers), a doubling of the cell volume, say, from $12A^3$ to $15A^3$, would likely be possible with little affect on performance.

D. Discussion

We have described progress towards creating a production multi-FPGA MD simulator. So far we have successfully integrated our short-range force/energy pipelines into NAMD-lite and this is now running on a workstation containing a single node of the Novo-G. The initial measurements of simulation quality indicate that this approach is viable, although more testing is needed (and in progress). Substantial performance was lost due, especially, to the need to support a more complex force model. Some of this loss can be recovered through design improvements. Much more performance is possible by porting

to current generation devices. We also find that the multi-level force pipeline has certain features that may facilitate creation large-scale systems.

REFERENCES

- [1] J. Anderson, C. Lorenz, and A. Travesset, "General purpose molecular dynamics simulations fully implemented on graphics processing units," *J. Computational Physics*, vol. 227, pp. 5342–5359, 2008.
- [2] J. Phillips, J. Stone, and K. Schulten, "Adapting a message-driven parallel application to GPU-accelerated clusters," in *Supercomputing*, 2008.
- [3] C. Rodrigues, D. Hardy, J. Stone, K. Schulten, and W.-M. Hwu, "GPU acceleration of cutoff pair potentials for molecular modeling applications," in *Proc. ACM Int. Conf. on Computing Frontiers*, 2008.
- [4] Shaw, D.E., et al., "Anton, a special-purpose machine for molecular dynamics simulation," in *Proc. International Symp. on Computer Architecture*, 2007, pp. 1–12.
- [5] S. Alam, P. Agarwal, M. Smith, J. Vetter, and D. Caliga, "Using FPGA devices to accelerate biomolecular simulations," *Computer*, vol. 40, no. 3, pp. 66–73, 2007.
- [6] N. Azizi, I. Kuon, A. Egier, A. Darabiha, and P. Chow, "Reconfigurable molecular dynamics simulator," in *Proc. IEEE Symp. on Field Programmable Custom Computing Machines*, 2004, pp. 197–206.
- [7] Y. Gu, T. VanCourt, and M. Herbordt, "Improved interpolation and system integration for FPGA-based molecular dynamics simulations," in *Proc. IEEE Conference on Field Programmable Logic and Applications*, 2006, pp. 21–28.
- [8] T. Hamada and N. Nakasato, "Massively parallel processors generator for reconfigurable system," *Proc. IEEE Symp. on Field Programmable Custom Computing Machines*, 2005.
- [9] V. Kindratenko and D. Pointer, "A case study in porting a production scientific supercomputing application to a reconfigurable computer," in *Proc. IEEE Symp. on Field Programmable Custom Computing Machines*, 2006, pp. 13–22.
- [10] R. Scrofano and V. Prasanna, "Preliminary investigation of advanced electrostatics in molecular dynamics on reconfigurable computers," in *Supercomputing*, 2006.
- [11] J. Villareal, J. Cortes, and W. Najjar, "Compiled code acceleration of NAMD on FPGAs," in *Proc. Reconfigurable Systems Summer Institute*, 2007.
- [12] M. Chiu and M. Herbordt, "Efficient filtering for molecular dynamics simulations," in *Proc. IEEE Conference on Field Programmable Logic and Applications*, 2009.
- [13] —, "Molecular dynamics simulations on high performance reconfigurable computing systems," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 3, no. 4, 2010.
- [14] A. George, "Novo-G Overview," Presentation at CHREC: NSF Center for High-Performance Reconfigurable Computing, 16 June 2010, <http://www.chrec.org/george/Novo-G.pdf>, 2010.
- [15] D. Hardy, "NAMD-lite," <http://www.ks.uiuc.edu/Development/MDTools/namd-lite/>, University of Illinois at Urbana-Champaign, 2007.
- [16] J. Phillips, G. Zheng, and L. Kale, "NAMD: biomolecular simulation on thousands of processors," in *Supercomputing*, 2002.
- [17] Phillips, J.C., et al., "Scalable molecular dynamics with NAMD," *J. Computational Chemistry*, vol. 26, pp. 1781–1802, 2005.
- [18] M. Chiu, M. Herbordt, and M. Langhammer, "Performance potential of molecular dynamics simulations on high performance reconfigurable computing systems," in *Proceedings High Performance Reconfigurable Technology and Applications*, 2008.
- [19] *PROCStar III*, Gidel Reconfigurable Computing, <http://www.gidel.com/PROCStar>
- [20] *Stratix III Device Handbook*, Altera Corporation, http://www.altera.com/literature/hb/stx3/stratix3_handbook.pdf accessed 9/2010, 2010.
- [21] D. Rapaport, *The Art of Molecular Dynamics Simulation*. Cambridge University Press, 2004.
- [22] T. Darden, D. York, and L. Pedersen, "Particle Mesh Ewald: an $N \log(N)$ method for Ewald sums in large systems," *Journal of Chemical Physics*, vol. 98, pp. 10 089–10 092, 1993.
- [23] J. Izaguirre, S. Hampton, and T. Matthey, "Parallel multigrid summation for the n-body problem," *Journal of Parallel and Distributed Computing*, vol. 65, pp. 949–962, 2005.
- [24] R. Skeel, I. Tezcan, and D. Hardy, "Multiple grid methods for classical molecular dynamics," *Journal of Computational Chemistry*, vol. 23, pp. 673–684, 2002.
- [25] Y. Gu and M. Herbordt, "High performance molecular dynamics simulations with FPGA coprocessors," in *Proc. Reconfigurable Systems Summer Institute*, 2007.
- [26] Y. Gu, T. VanCourt, and M. Herbordt, "Explicit design of FPGA-based coprocessors for short-range force computation in molecular dynamics simulations," *Parallel Computing*, vol. 34, no. 4–5, pp. 261–271, 2008.
- [27] R. Engle, R. Skeel, and M. Drees, "Monitoring Energy Drift with Shadow Hamiltonians," *J. Computational Physics*, vol. 206, pp. 432–452, 2005.