Integrating FPGA Acceleration into the Protomol Molecular Dynamics Code: Preliminary Report*

Yongfeng Gu Tom VanCourt Martin C. Herbordt
Department of Electrical and Computer Engineering
Boston University; Boston, MA 02215

Abstract: We describe a new pipeline for computing non-bonded forces and its integration into the ProtoMol molecular dynamics (MD) code. There are several innovations: a novel interpolation strategy, including use of higher order terms; coefficient generation with orthonormal functions; the introduction of "semi-floating point" numbering; and various issues related to system integration. As a result, we are able to model far more particle types, without relying on complex buffering, and obtain higher accuracy than previously. A two pipeline accelerator has been implemented on a 2004-era Xilinx VirtexII Pro VP70, integrated into ProtoMol, and tested with an enzyme inhibitor model having 8000 particles and 26 particle types. Despite performing all $O(n)$ work on the host PC, as well as the data conversion and communication overhead, this implementation yields 5.5× to 15.7× speed-ups over a 2.8GHz PC (depending on whether cell lists are used), and with accuracy comparable to the serial code.

1 Introduction

Several issues remain to be solved before FPGA implementations of MD simulation are likely to enter production use; we address the following here:

- **The number of particle types simulated.** The van der Waals (Lennard-Jones or LJ) force computation depends on atom type. For the table-lookup method used previously ([1, 2]), a separate table is required for each combination of atom types. This requires complex memory exchanges.
- **Simulation accuracy.** Until FPGAs have a number of hard-wired floating point units, non-standard arithmetic is likely to yield the most competitive implementations. This leads to concerns about general acceptance.
- **System integration.** MD codes are highly complex, and many MD functions are best performed on a microprocessor. FPGA implementations must therefore be integrated an existing system.

*This work was supported in part by the NIH through award #RR020209-01 and facilitated by donations from Xilinx Corporation. Email: {maplegu|tvancour|herbordt}@bu.edu Web: http://www.bu.edu/caadlab; http://protomol.sourceforge.net

This report describes an FPGA implementation of MD acceleration that addresses all of these issues. We demonstrate a 5.5× to 15.7× improvement, depending on whether cell lists are used, in total application performance for the widely used ProtoMol MD code [4]. Moreover, *there is no compromise in accuracy.*

Beyond the scope of this work is use of transform-based methods for long-range forces.

2 Methods

We have implemented a complete MD system based on ProtoMol: motion updates and bonded forces are computed on the host using the original system, while the LJ and Coulomb forces are computed on the FPGA. In this abstract we concentrate on describing the LJ force implementation; that of the Coulomb interaction is analogous and uses much of the same hardware. The LJ force for particle i can be expressed as:

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

where ϵ_{ab} and σ_{ab} are parameters related to the types of particles, i.e. particle i is type a and j is type b. These are short range forces, attractive or repulsive according to distances between the interacting atoms, which vary in strength according to the types of interacting atoms. Because of the complexity of the LJ expression, and because it is possible in principle for any two atoms to interact this way, this step has previously been particularly problematic.

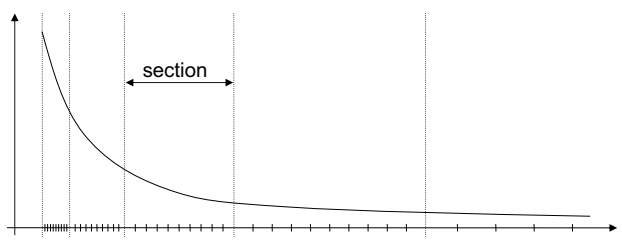

Figure 1: Table look-up varies in precision across r^{-k}.

The r^{-14} and r^{-8} expressions both display extreme changes in behavior over the range of possible interaction radii, as shown in Figure 1. It would be an

exorbitant and needless cost in table size to use the same number of coefficients and the same step sizes in the well-behaved regions of the curve. Also, the ranges of interpolation coefficients differ in different parts of the curve: they have relatively large magnitudes in the left-hand part of the curve, but lower magnitude in the right-hand regions. That implies the need for different numbers of bits to the left of the decimal point in each region of the curve, to maintain roughly constant accuracy. This implementation of interpolation divides the curve into multiple sections of different sizes, representing regions of the curve that have different numerical behavior. Sections of the curve have power-of-two sizes, and differ in the following ways:

- Step sizes differ between sections, although each section has the same number of steps,
- Numbers of non-zero coefficients in the Taylor expansion are different, with more terms stored in the more ill-behaved sections, and
- Positions of the radix points differ between sections ("semi floating point"), although they are constant within sections.

This last optimization allows the same precision in each section, but higher coefficient values in the ill-behaved sections (with numbers of the form xxx.x) and lower values where the curve is flatter (x.xxx).

Varying precision of the look-up table was used by Azizi et al. for the entire LJ force [1]. The method here has several advantages over that approach. Although two look-up tables are required (for the 8 and 14 terms, respectively), in the other method a different look-up table is required for every combination of atom types, resulting in potentially hundreds of tables. The other advantages include the integration of higher order interpolation and the use of "semi floating point." These combined optimizations result in substantial savings in area over double precision floating point, but with little if any sacrifice in simulation accuracy [3].

3 Implementation

The LJ and Coulomb forces were implemented on an Annapolis Microsystems WildstarII-Pro board, which has two Xilinx Virtex-II Pro XC2VP70 -5 FPGAs. Only one of the FPGAs is currently used. Particle data are transferred between board and host PC using DMA routines from the Annapolis Micro Systems software library. Currently two pipelines fit on the chip; this is in contrast to the four pipelines previously obtained [2]. We have, however, achieved four pipelines with the VP100; more should be possible with newer chips and further optimization. The design supports up to 32 atom types and 11,200 atoms. Larger simulations are possible by using off-chip/on-board memory.

Using higher order interpolation involves a trade-off between the table size and the order of the interpolation. Interestingly, these two terms are limited by different resource types: block RAMs are critical for table size while multipliers and logic slices bound interpolation order. Currently the optimal is 3rd order interpolation with 128 intervals per section.

The accelerator was integrated into ProtoMol 2.03 by replacing the LJ and Coulomb force modules with the FPGA version. Particle position data are down- and up-loaded in double precision floating point; fixed point conversion on both ends is done in the FPGA.

4 Performance and Accuracy

Accuracy is a concern when a numbering system is modified. Because of the inherently chaotic nature of the calculation, "Obtaining a high degree of accuracy in the trajectories is neither a realistic nor a practical goal" [5]. Instead, the goal is to maintain accuracy, as measured by energy fluctuations, *no worse than the accuracy of the original calculations.* This has been achieved for a model of bovine pancreatic trypsin inhibitor in water, a model of 1101 particles run for 10,000 time steps. After 10 runs we have found comparable energy fluctuation between the original and accelerated systems, with both being close to 0.014.

For performance we compared full end-to-end runs of the same model with 8000 particles and 26 atom types. For all-to-all force computation with no cut-off, the PC-only implementation takes 15.1s per time-step, the FPGA-accelerated version 0.96s (15.7× speed-up). With cell lists and cut-offs enabled on both versions (5A cells, 10A cut-off), the time-steps are reduced drastically: they take 0.66s on the PC and 0.12s on the PC plus FPGA (5.5× speed-up). If instead of the VP70 a higher speed-grade VP100 were used, post place-and-route estimates indicate that these speed-ups would double.

References

[1] Azizi, N., Kuon, I., Egier, A., Darabiha, A., and Chow, P. Reconfigurable molecular dynamics simulator. In *FCCM* (2004), pp. 197–206.

[2] Gu, Y., VanCourt, T., and Herbordt, M. C. Accelerating molecular dynamics simulations with configurable circuits. *IEE Proc. CDT (in press)* (2006).

[3] Gu, Y., VanCourt, T., and Herbordt, M. C. Improved interpolation and system integration for FPGA-based molecular dynamics simulations. In *Field Programmable Logic and Applications* (2006).

[4] Matthey, T. ProtoMol, an object-oriented framework for prototyping novel algorithms for molecular dynamics. *ACM TMS 30*, 3 (2004), 237–265.

[5] Rapaport, D. *The Art of Molecular Dynamics Simulation.* Cambridge University Press, 2004.