

FPGA ACCELERATION OF MOLECULAR DYNAMICS COMPUTATIONS

Yongfeng Gu, Tom VanCourt, Martin C. Herbordt

Department of Electrical and Computer Engineering, Boston University, Boston, MA

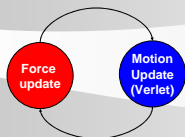


Abstract

Molecular Dynamics (MD) is of central importance to computational chemistry and its myriad applications. Here we show that MD can be implemented efficiently on a COTS FPGA board, and that speed-ups from 31x to 88x over a PC implementation can be obtained. The amount of speed-up depends on the stability required, with the upper end of that range being viable in many cases.

Introduction

MD is an iterative technique that runs in phases: the forces on each atom/molecule are computed, then applied using equations of motion, for example, Verlet.



Although modern force computations have become highly sophisticated (with 10 or more terms in some cases), the complexity generally resides in computing the Van der Waals (Lennard-Jones or LJ) and Coulombic terms. These are $O(N^2)$ in the number of particles N , while the motion updates are $O(N)$, and the other forces - which only look at bonds - are also $O(N)$.

Our work differs from previous approaches in that we combine the following: on the hardware side, that we use a COTS board; on the implementation side, that we model the Coulombic as well as the LJ term, and that we support the simultaneous modeling of multiple types of molecules. We have also investigated precision/accuracy tradeoffs.



Shown is the WildstarII-Pro board from Annapolis Micro Systems, Inc, with two Xilinx Virtex-II-Pro XC2VP70-5 FPGAs.

Methods

Forces: A typical force model has several components: $F^{total} = F^{bond} + F^{angle} + F^{torsion} + F^H + F^{non-bonded}$

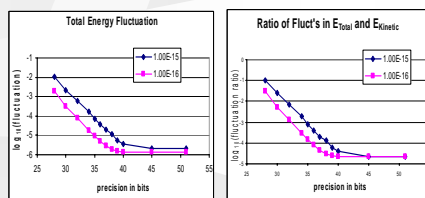
We implement the non-bonded forces, Lennard-Jones and Coulombic, with cut-off in a periodic model.

$$F_i^{LJ} = \sum_{j \neq i} \frac{\epsilon_{ab}}{\sigma_{ab}} \left[12 \left(\frac{\sigma_{ab}}{r_{ji}} \right)^{14} - 6 \left(\frac{\sigma_{ab}}{r_{ji}} \right)^8 \right] \vec{r}_{ji} \quad F_i^C = q_i \sum_{j \neq i} \left(\frac{q_j}{r_{ji}^3} \right) \vec{r}_{ji}$$

Reference Code: For serial reference code (as in [2]) we began with that described in [3]. We also created our own serial reference code that tracks the hardware implementation, e.g. in varying precision. The double precision code ran at about 9.5s per MD time-step on a PC with a 2.4GHz Xeon CPU. This is similar to the 10.8s for a 2.4GHz P4 described in [2].

Precision, Resources, Accuracy: MD applications are usually run with double precision floating point. One advantage of implementing MD on FPGAs is that we can trade off hardware resources for simulation accuracy by varying the precision. By measuring the relationship among precision, time-step resolution, and simulation accuracy, we find that for precision beyond 40 bits, accuracy improves only very slowly. (See also [1].) Another measure of that the simulation accuracy is acceptable is that $\Delta E_{total} / \Delta E_{kinetic} < .05$. For this, 30 bits are sufficient.

The left-hand graph shows fluctuation in total energy versus fixed point precision. On right is the ratio of fluctuations in total energy and kinetic energy.



Precision "sweet spots" for current FPGA technology: Another factor in choosing precision is its effect on the number of computation units. The most efficient designs using current technology have either 4 or 8 pipelines.

- 51 bits: precision similar to double precision FP, 4 pipelines on VP100
- 40 bits: beginning of highly stable region, 4 pipelines on VP70.
- 35 bits: More than the 30 bit minimum, 8 pipelines on the VP100.

Results

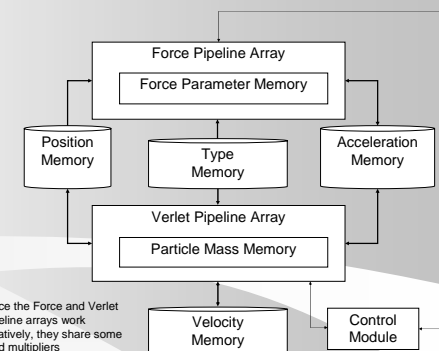
We implemented the configurations and validated them with the two serial reference codes. Timings are for a single VP70 on the WildstarII-Pro board, while timings for the VP100 are from simulation.

Precision (bit)	Pipe-lines	HW multipliers (% of usage)	Block Ram (% of usage)	Delay (ns)	Speed-up
35	4	176(53%)	214(65%)	11.1	50.8x
40	4	264(80%)	251(77%)	12.2	46.4x
45	4	288(88%)	285(87%)	13.2	42.7x
51	4	288(88%)	317(97%)	18.0	31.3x
35	8	256(78%)	326(99%)	22.2	51.0x
51*	4	288(65%)	317(77%)	13.6	41.5x
35*	8	256(58%)	334(75%)	12.8	88.5x

Configurations on XC2VP70 (* XC2VP100 in simulation only)

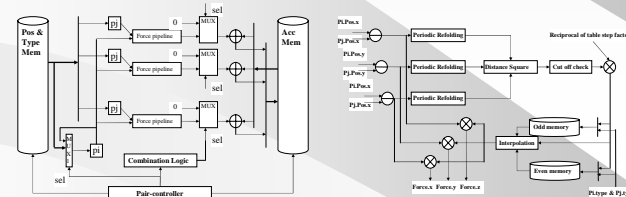
The configurations here are able to contain up to 8K particles and easy to expand. The delay of most configurations are limited by adders. But the 35 bits/8 pipelines configurations on VP70 has a long route delay, because it almost uses up slices.

Implementation



Since the Force and Verlet Pipeline arrays work iteratively, they share some hard multipliers

Fixed point is used. Precision varies as per configuration. Data are scaled to maintain near maximum resolution throughout the computation while bounding the data-path size. The force calculations use table look-up with interpolation.



Shown on left is the force pipeline array. On right is a single force pipeline.

Originally, the critical resource for both speed and area was the hard multipliers. In order to get the multipliers off the timing critical path, we created customized multipliers (up to 51 bits) with 9 hard multipliers rather than 3. The multiplier latency was reduced from 25ns to 9ns. Another optimization was to implement data delay FIFOs in pipelines with spare block RAMs rather than registers. The critical path is now through the adders - in the future, these will be optimized to improve the operating frequency.

Extensions

An interesting aspect of FPGA research is that relative computational complexity of methods often differs when implemented on an FPGA from when implemented on a PC or Super-computer. With MD, this is critical in the choice of boundary conditions. The implementation shown here works well with fixed and stochastic cut-offs. However, many implementations use periodic boundary conditions. These are susceptible to different kinds of error than the cut-offs, but perhaps more significantly, allow solutions based on Ewald sums or other $O(N \log N)$ methods. Investigating the relative trade-offs of precision, algorithm, and accuracy along this axis should be fruitful. Also, this work is part of a larger project involving the acceleration of applications in computational biochemistry (e.g. [4]) and will be integrated into that.

[1] Amisaki, T., Fujiwara, T., Kusumi, A., Miyagawa, H., and Kitamura, K. Error evaluation in the design of a special-purpose processor that calculates nonbonded forces in molecular dynamics simulations. J. Comp. Chem. 16, 9 (1995), 1120-1130.

[2] Azizi, N., Kuon, I., Egler, A., Darabiha, A., and Chow, P. Reconfigurable molecular dynamics simulator. In Proc. Field Programmable Custom Computing Machines (2004), pp. 197-206.

[3] Bargiel, M., Dzwiniel, W., Kitowski, J., and Mociński, J. C-language molecular dynamics program for the simulation of Lennard-Jones particles. Computer Physics Communication 64 (1991), 193-205.

[4] Van Court, T., Gu, Y., and Herbordt, M. C. FPGA acceleration of rigid molecule interactions. In Proc. Of Field Programmable Logic and Applications (2004).