FPGA Based Accelerator for Molecular Dynamics

Nicolae Goga[(1)], Mihaela Malița[(2)], David Mihăiță[(3)], Gheorghe M. Ștefan[(3)]

[(1)] University of Groningen
[(3)] Saint Anselm College, Manchester, NH
[(3)]Politehnica University of Bucharest

N-Body method applied to model molecular dynamics is one of the most computationally intensive applications. Hybrid computation could be used to deal with this very demanding computational motif. Our proposal is based on a Xilinx platform to implement a 512-core vector machine, used as accelerator. The accelerator has the many-core Map-Reduce architecture developed in [6].

1. Introduction - *Molecular dynamics* (MD) is a type of N-body simulation, one of the 13 "dwarfs" emphasized in *The View from Berkeley* [2]. It is about atoms and molecules that are allowed to interact for a fixed period of time, giving a view of the dynamical evolution of the system. The trajectories of atoms and molecules are determined by numerically solving Newton's equations of motion for a system of interacting particles. The forces between the particles and their potential energies are calculated using interatomic potentials or molecular mechanics force fields. The method was initially developed in the field of theoretical physics but is applied today mostly in the modelling of biomolecules.
We selected **GROMACS** (**GRO**ningen **MA**chine for **C**hemical **S**imulations) as a molecular dynamics package used for simulations of proteins, lipids and nucleic acids [1]. This software package was developed in the Biophysical Chemistry department of the University of Groningen, but is now maintained by contributors in many universities and research centers. GROMACS is free, open source released.
The computation for one interaction for a *very short* fix period of time supposes, for *N* particles, a computation in $O(N^2)$. Because the fix period of time must be very short the computation must be repeated so many times that, for most of the real problems, the current computational resources are totally insufficient. Looking for ways to provide accelerating resources is mandatory. Current implementations use distributed computing and local ways to accelerate the computation. Locally, the acceleration is provided mainly on three ways: improving the sequential code at the assembly level, using programmable accelerators (such as GPUs) to run the critical section of the code, or by adding specialized hardware.
Our proposal is a FPGA based programmable accelerator supposed to work on the "leafs" of the Gromacs system. It is connected through a PCIe interface to the host which is an x86 system.
The next section describes the structure and the architecture of the accelerator and the parallel algorithm used for testing the accelerator. The third section evaluates the results of the simulation. Final comments conclude our paper.

2. The MapReduce Accelerator – We start from the following ascertainments: (1) the assembly code in x86 environment is unable to provide enough performance because of the small computational capabilities of the current processors (a four-core engine, each with its SIMD co-machine provides no more than 16 execution units) and the low degree of parallelism provided by an architecture which is a combination of multi-threading with SIMD capabilities, (2) using a GPU as a general purpose accelerator is limited in performance because it has an architecture inheriting limitations due to the graphic functions for which it was initially conceived, (3) the specialized hardware [3], [4] does not have the flexibility requested by the complexity of the computation involved in MD.
Our project has two stages: (1) developing the architecture and the structure of an accelerator implemented in FPGA technology, (2) based on the results provided in the first step, convert the FPGA solution into an ASIC or eASIC chip in order to reduce price, increase performance and reduce power. The current paper refers to the first stage.

The accelerator we propose is connected to the x86 host computer using a PCIe interface (see Figure 1) and consists of a MapReduce section controlled by Scalar section, both interfaced through two FIFOs. The computation is performed by the Scalar unit and the MapReduce unit (see Figure 2), where:

- **Scalar** unit is a RISC engine which fetches in each clock cycle from its program memory a pair of instructions, one for itself and another to be issued toward the Map array of cells
- **Map** section is an array of *p* cells each with its execution unit (*eng*) and local memory (*mem*); each cell executes, according to its internal state, the instruction received from the Scalar unit

- **Reduction** is a *log*-depth circuit which performs functions defined on vectors which return a scalar (for example: *add*, *min* …)

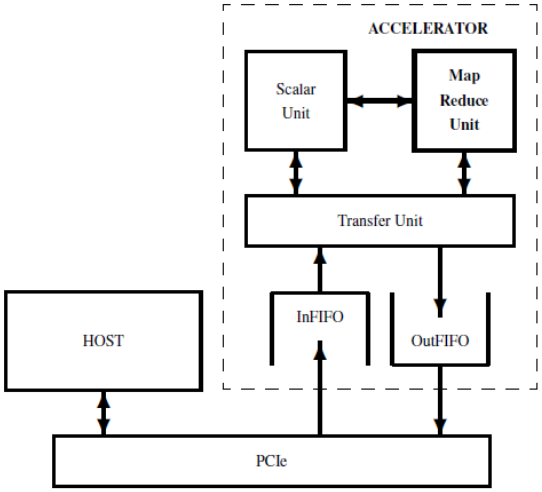

Figure 1. The system configuration. The accelerator consists of a scalar unit and a map-reduce unit connected through FIFOs to the x86 Host using a PCIe interface.

InFIFO receives programs to be loaded in the program memory of Scalar unit and data for both, Scalar unit and Map section. OutFIFO sends back to Host the result of the computation.

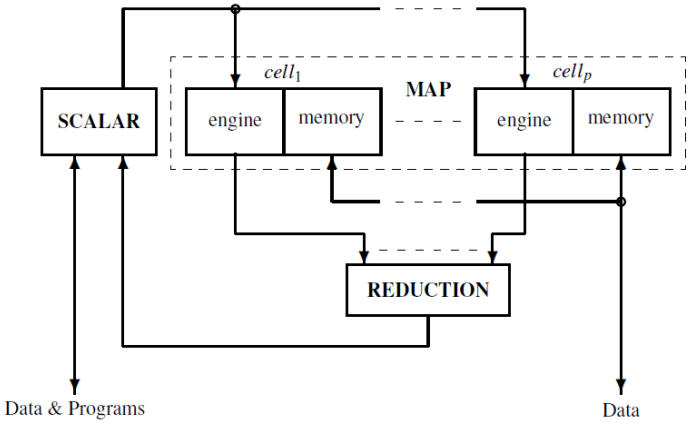

Figure 2. The computational part of the accelerator.

The computation accelerated for the Gromacs system is described in Figure 3. The accelerator deals with a system of *n* particles. Each particle is characterized by *m* scalars. Thus, the accelerator is loaded with a two-dimension array organized as *m* *n*-scalar vectors. If *n*>p, then the array is fragmented in *m*×*p* arrays stored in the accelerator's scalar memory. Two kinds of operations are mainly performed on this two-dimension array: (1) a search for neighbors (particles which interact with one another) for each of the *p* particles, (2) compute and apply the forces on each particle. In each processing cycle step (1) is followed by *q* steps (2), where the user can select the value of *q* from 1 to a higher value depending on the temperature, pressure etc. of the system, with the addition that increasing this value will result in higher errors, but a faster simulation. After a number of processing cycles, also defined by the user, the resulting array is sent back to the host.
In step (1), the position of each particle is compared, in parallel, with the positions of all other particles. The parameters of a particle are broadcast in the cells that are close to it, thus defining the neighborhood of each particle in each cell. In step (2), the new position for each particle is computed, also in parallel. The degree of parallelism is given by the mean value of the size of the neighborhoods.

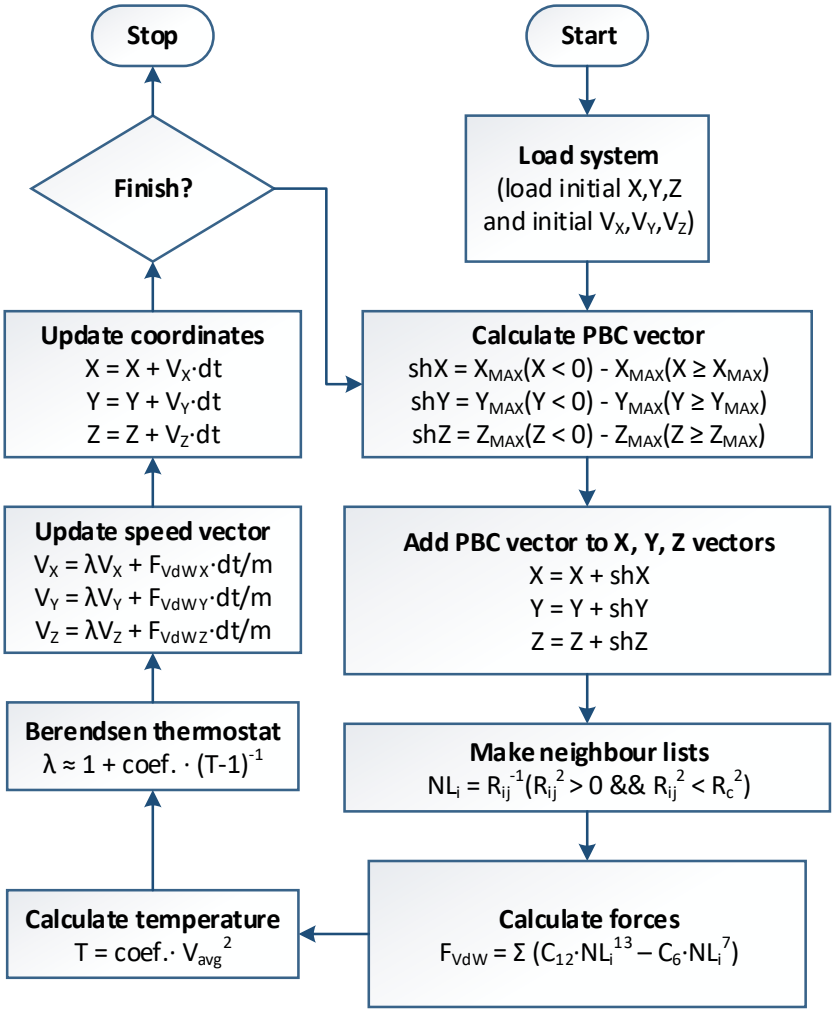

Figure 3. The flowchart of the computation implemented to validate the use of our MapReduce architecture as accelerator for the Gromacs system.

3. Results and Discussion – The results of the simulation done on a Verilog cycle accurate simulator of the accelerator are presented in Table 1. For each stage of the algorithm the number of clock cycles are listed.

Table 1

Simulation part	Cycles	Percent of algorithm
Full simulation	33040	100.00%
Box periodicity	80	0.24%
Neighbour search	26410	79.93%
Force computation	6409	19.40%
Thermostat	91	0.28%
Update	50	0.15%

The next table shows the overall degree of parallelism and the degree of parallelism for the main section of the algorithm. The overall degree of parallelism of 75.6% for an array of 400 cells could be considered very good.
In the last table the performance[1] of our solution is compared with x86 based architectures and the most performant solution provided by Anton accelerator circuit [3], [4]. For our solution, there are two cases: (1) the neighbor search is done at each force computation cycle (NS:F = 1:1), (2) the neighbor search is done at 10

[1] Performance is expressed in how many micro seconds of molecular behavior can be simulated per day (μs/day)

force computation cycles (NS:F = 1:10). Compared with a mono-core x86 engine, our system provides 32x acceleration, while compared with a four-thread, each with a 4-cell SIMD accelerator, our system provides 5.94x acceleration, due to the degree of parallelism achieved in the x86 based 16-cell engine the degree of parallelism results only 0.296, compared with ours which is 2.55x higher.

Table 2

Simulation part	Active cells	Controller
Full simulation	75.6%	13.8%
Box periodicity	66.8%	0.0%
Neighbour search	79.2%	15.30%
Force computation	60.4%	7.60%
Thermostat	51.0%	62.70%
Update	100.0%	17%

The power performance, expressed in energy used for 1 μs of simulation, is very good for a FPGA implementation and extremely good for an ASIC version of our architecture.

Table 3

Machine	Cores	NS:F	Freq. [GHz]	Price [USD]	Perf. [μs/day]	Power [W]	Energy cost [Wh/μs]
Intel i5	1	1:10	2.7 GHz	$200	5.84	65	267.1
Intel i5 (SSE)	1	1:10	2.7 GHz	$200	9.78	70	171.8
Intel i5	4	1:10	2.7 GHz	$200	18.94	90	114.0
Intel i5 (SSE)	4	1:10	2.7 GHz	$200	31.48	95	72.4
MRA (FPGA)	512	1:1	0.5 GHz	$1000	52.42	35	16.0
MRA (FPGA)	512	1:10	0.5 GHz	$1000	187.01	35	4.5
MRA (ASIC)	512	1:10	1.0 GHz	$10	374.02	3	0.2
Anton	1	–	0.4 GHz	-	572.32	75	3.1
Anton	512	–	0.4 GHz	$10Mil	293027.84	116500	9.5

4. Conclusion – Because the power efficiency of our FPGA solution exceeds 16x the x86 based solution and the ASIC implementation exceeds 15.5x the ASIC circuit, we consider that we provide an excellent solution for developing MD accelerators.

5. References

[1] M.J. Abraham, D. van der Spoel, E. Lindahl, B. Hess, and the GROMACS development team, GROMACS User Manual version 5.1.2, www.gromacs.org (2016) ftp://ftp.gromacs.org/pub/manual/manual-5.1.2.pdf

[2] Krste Asanovic, *et al.*, The landscape of parallel computing research: A view from Berkeley, 2006. http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.pdf

[3] Martin M. Deneroff, *et al.*, Anton: A Specialized ASIC for Molecular Dynamics, 2008. http://www.hotchips.org/wp-content/uploads/hc_archives/hc20/2_Mon/HC20.25.421.pdf

[4] D. E. Shaw, *et al.*, "Anton 2: raising the bar for performance and programmability in a special-purpose molecular dynamics supercomputer," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, Piscataway, NJ, USA, 2014. http://conferences.computer.org/sc/2014/papers/5500a041.pdf

[5] Gheorghe Ștefan, *et al.*, "The CA1024: A Fully Programmable System-On-Chip for Cost-Effective HDTV Media Processing", in *Hot Chips: A Symposium on High Performance Chips*, Memorial Auditorium, Stanford University, August 20 to 22, 2006. http://www.hotchips.org/archives/2000s/hc18/ see section S5

[6] Gheorghe M. Ștefan, Mihaela Malița, "Can One-Chip Parallel Computing Be Liberated from Ad Hoc Solutions? A Computation Model Based Approach and Its Implementation", *18th Inter. Conf. on Circuits, Systems, Communications and Computers*, Santorini, July 17-21, 2014, 582-597. http://www.inase.org/library/2014/santorini/bypaper/COMPUTERS/COMPUTERS2-42.pdf