Architecture of an FPGA Accelerator for Molecular Dynamics Simulation Using OpenCL

Hasitha Muthumala Waidyasooriya, Masanori Hariyama
Graduate School of Information Sciences, Tohoku University
Aoba 6-6-05, Aramaki, Aoba, Sendai, Miyagi,980-8579, Japan
Email: {hasitha, hariyama}@tohoku.ac.jp

Kota Kasahara
Institute for Protein Research, Osaka University,
3-2 Yamadaoka, Suita-shi, Osaka, 565–0871, Japan
Email: kota.kasahara@protein.osaka-u.ac.jp

***Abstract*—Molecular dynamics (MD) simulations are very important to study physical properties of the atoms and molecules. However, a huge amount of processing time is required to simulate a few nano-seconds of an actual experiment. Although the hardware acceleration using FPGAs provides promising results, huge design time and hardware design skills are required to implement an accelerator successfully. In this paper, we propose an FPGA accelerator designed using C-based OpenCL. We achieved over 4.6 times of speed-up compared to CPU-based processing, by using only 36% of the Stratix V FPGA resources. Maximum of 18.4 times speed-up is possible by using 80% of the FPGA resources.**

***Index Terms*—OpenCL for FPGA, molecular dynamics simulation, hardware acceleration, scientific computing.**

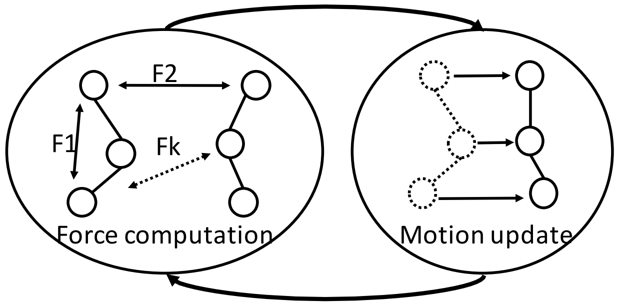


Fig. 1. Molecular dynamics simulation model.

I. Introduction

Molecular dynamics (MD) simulations [1] are very important in the fields of computational chemistry, materials science, bio-informatics, etc to study physical properties of the atoms and molecules. In MD simulations, an actual system is modeled at the atomic level and classical physics is used to simulate the movements of atoms. There are many publicly available and widely used software packages for MD simulations such as AMBER [2], Desmond [3], GROMACS [4], LAMMPS [5], etc. All of those are based on an iterative computation method, where the computation results of one iteration are used as the inputs in the next iteration. Each iteration consists of two major phases: force computation and motion update as shown in Fig.1. MD simulations require millions of iterations and a huge amount of processing time on general purpose CPUs to simulate few nanoseconds of the real time. Months to years of processing time is spend to find at least some useful results while simulating an actual laboratory experiment is not possible even today.

Hardware acceleration is already used to reduce the huge processing time. ASIC (application specific integrated circuit) implementations of MD simulation are already proposed in [6], [7] and [8]. However, designing such special purpose processors requires years of design, debugging and testing time and also involves a huge financial cost. Therefore, ASICs are out-of-reach for most researchers, although their performances are quite excellent. A cheap way of hardware acceleration is provided by FPGAs (feild-programmable-gate-arrays) [9]–[11]. Although the cost is extremely small compared to ASICs, the design time is still very large. FPGAs are designed using hardware description language (HDL) so that hardware design skills and experiences are required for a successful FPGA implementation. When there are algorithm changes and hardware updates, it is often required to redesign the whole FPGA architecture.

To overcome these problems, OpenCL for FPGA has been introduced [12]. It is a complete framework that includes firmware, software and device drivers to connect, control and transfer data to and from the FPGA. It provides a heterogeneous system consist of a host CPU and a device which is an OpenCL capable FPGA. Lightweight tasks can be processed on the host CPU while the heavyweight tasks can be offloaded to the FPGA. The host program is written in C code and the device program is written in OpenCL code [13] which is also similar to C code. FPGA implementation can be done entirely using software without requiring a single line of HDL code. Recently, some works such as [14] and [15] propose FPGA accelerators using OpenCL.

In this paper, we propose an FPGA Accelerator for MD simulations using OpenCL. The FPGA design time has been reduced to just few hours due to software based design. Any algorithmic change could be easily implemented by just changing the software code, and the same code can be re-used in any OpenCL capable FPGA board. Therefore, the proposed method supports any future hardware or software update. In this paper, we demonstrate that it is possible to achieve over 4.6 times of speed-up compared to CPU implementation for the most time consuming non-bonded force computations. This speed-up is achieved by using only 36 % of the FPGA

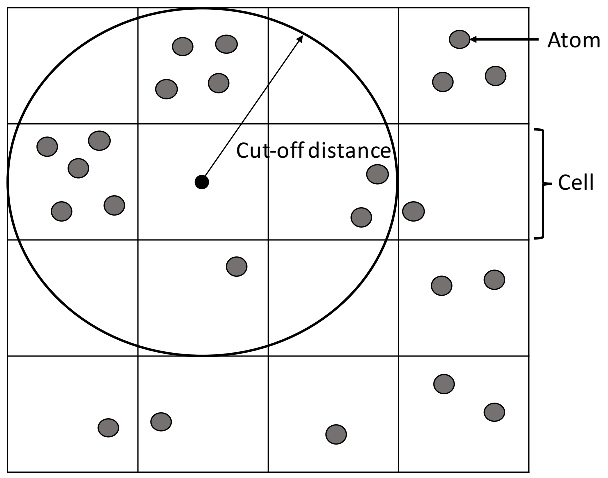


Fig. 2. Division of the simulation box in to cells.

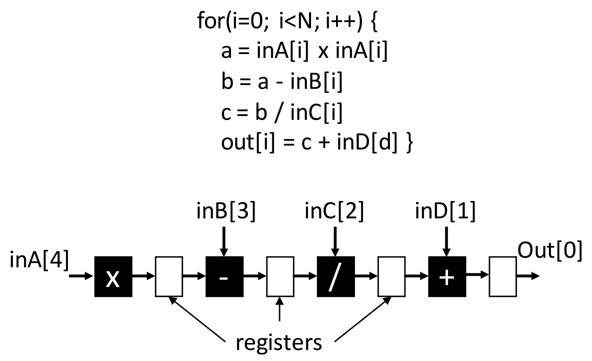
resources. If we assume a 80% resource usage, we can achieve a similar speed-up compared to custom FPGA accelerators designed using HDL. We also highlight the problems of the FPGA-based heterogeneous systems such as data transfers between the host and the device and shows some insights to tackle those problems in future OpenCL capable FPGA-based systems.

II. Molecular dynamics simulation

As explained in section I, MD simulations are based on two tasks; force computation and motion update. The force computation consists of bonded-force and non-bonded force computations. Bonded forces only affect few neighboring atoms, and can be computed in $O(N)$ time for N atoms. Non-bonded forces comprise of van der Waals and electrostatic forces. Those forces exist between two atoms for all atom-pairs so that the computation requires $O(N^2)$ processing time. There are several techniques available to reduce the computation cost and accelerate non-bonded force computation.

MD simulation is done for a box of atoms. To reduce the computation complexity, the box is divided in to multiple cells. Fig.2 shows a 2-D representation of the cell division. A cut-off distance is set between two atoms and the neighboring cell-pairs within the cut-off distance are extracted to a cell-pair list. Non-bonded force computation is done for the atoms of the cell-pairs in the list. As a result, we do not have to consider all atom-pair combinations for the force computation. Since the atoms move in the box, the cell-pair list is updated in each iteration. A periodic boundary condition is used when an atom leaves the box. We assume that the same box is replicated at the boundaries so that an atom leaves from the box reappears from the opposite direction. Using this method, we can simulate a large system by using only a small number of atoms.

Even with these techniques, MD simulation takes a huge amount of processing time. Non-bonded force computation occupies most of the total processing time. Therefore, we accelerate the Non-bonded force computation using FPGA. The



```
for(i=0; i<N; i++) {
   a = inA[i] x inA[i]
   b = a - inB[i]
   c = b / inC[i]
   out[i] = c + inD[d] }
```

Fig. 3. Loop-pipelining in OpenCL for FPGA.

FPGA acceleration is based on the MD simulation software "myPresto/omegagene" [16], [17].

III. MD simulation accelerator architecture using OpenCL

A. Parallel processing using loop-pipelining

OpenCL for FPGA uses pipelines on FPGA to implement parallel computations. A pipeline contains many computation stages, and the results of one stage are stored in registers and used as the inputs of the next stage. The Altera offline compiler (AOC) reads an OpenCL code and implements pipelines for the computations inside loops. This is called "loop-pipelining". As shown in Fig.3, the computations in a loop are done sequentially similar to a typical "C program". However, for each computation, a pipeline stage is designed. Parallel processing is executed in all stages for different data. Since FPGAs have a large number of registers, pipelines with hundreds of stages can be designed easily.

When there is another loop inside a loop and data dependencies exist between loops, a separate pipeline is implemented for the inner-loop. Therefore, an iteration of the outer-loop proceeds only after all the iterations of the inner-loop are finished. Otherwise, the computation of the outer-loop stalls. To avoid this, we have to unroll the inner-loops by using more hardware resources.

B. FPGA architecture using OpenCL

The summery of non-bonded force computation method in an iteration is shown in Algorithm.1. We use a cell-pair list to reduce the amount of computations. Since the atoms moves in each iteration, the atoms in a cell also changes. Therefore, we have to refresh the cell-pair list in each iteration. The computation is done in three loops. The outer-loop proceeds for each cell-pair in the list. In the inner-loops, two atoms from each cell in the cell-pair are selected to compute non-bonded forces. As explained in section III-A, this algorithm is not suitable for OpenCL implementations due to the stalls in the outer-loops. Moreover, we cannot unroll the inner-loops since we do not know the loop boundaries. The loop boundaries are decided by the number of atoms in a cell and it is not a

```
refresh $cell\text{-}pair\ list$
Non-bonded force computation (cell-pair list, force)
    foreach cell-pair in the cell-pair list do
        $CELL1$ = cell-pair→cell1;
        $CELL2$ = cell-pair→cell2;
        foreach atom in CELL1 do
            foreach atom in CELL2 do
                calculate $force$ between the two atoms;
            end
        end
    end
end
```

Algorithm 1: Pseudo code of the non-bonded force computation method.

constant for all cells. Even for the same cell, the number of atoms changes in each iteration due their movements.

To solve this problem, we separate the force computation from the atom-pair selection. We first extract the complete list of atom-pairs based on the cell-pair list. Then we perform the force computation for each atom-pair in the list. The atom-pair-list extraction is just a searching procedure and no heavy computations are required. On the other hand, force computation contains many multiplications and divisions. Therefore, we use the host processor for atom-pair list extraction and transfer the list data to the FPGA for force calculation. Once the list is extracted, only a single loop is sufficient for the force computation of all the atom-pairs in the list. As a result, AOC can implement loop-pipeling on FPGA to accelerate this process.

Fig.4 shows the flow-chart of the proposed CPU-FPGA heterogeneous processing. Bonded-force computation is done on CPU while non-bonded force computation is done on FPGA. After computing all the forces, the atom coordinates are updated considering the motion due to forces. Then a new atom-pair-list is extracted based on the cell-pair-list. However, we get two overheads; atom-pair-list data transfer to FPGA and force data transfer from FPGA. We will further discuss this problem in the evaluation, and suggest some solutions.

Fig.5 shows the proposed CPU-FPGA heterogeneous processing system for molecular dynamic simulations. Fig.5(a) shows the block diagram of the system architecture. FPGA board is connected to the CPU through a PCI express bus. The data are transferred to the DRAM of the FPGA. The computed force data are written to the DRAM and read by the host CPU. Fig.5(b) shows a picture of the designed heterogeneous system that contains a CPU and an FPGA board.

IV. Evaluation

For the evaluation, we used DE5 board that contains Stratix V 5SGXEA7N2F45C2 FPGA. The operating system is CentOS 6.7. The FPGA is configured using Quartus 15.0 with OpenCL SDK. The molecular dynamics simulation contains 22,795 atoms.

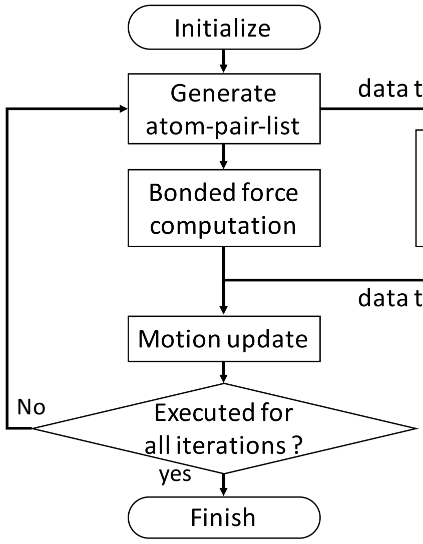


Fig. 4. Flow-chart of the CPU-FPGA heterogeneous processing.

TABLE I
FPGA resource usage.

Resource	Usage	Percentage used (%)
Logic (ALMs)	83,653	35.64
Registers	91,682	9.76
Memory (Mbits)	3.35	6.70
DSPs	49	19.14

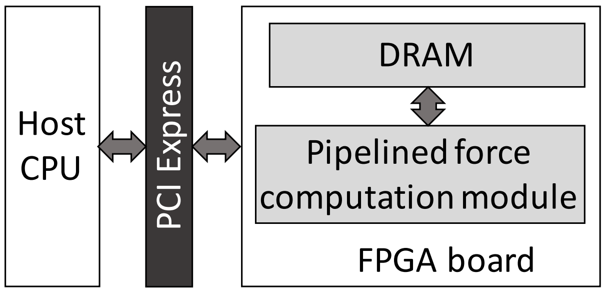
TABLE II
Comparison of the processing time using the straight forward method by using the same software code used in CPU implementation.

	CPU	FPGA
Non-bonded force computation	0.68 s	88.03 s
Total computation	0.86 s	88.24 s

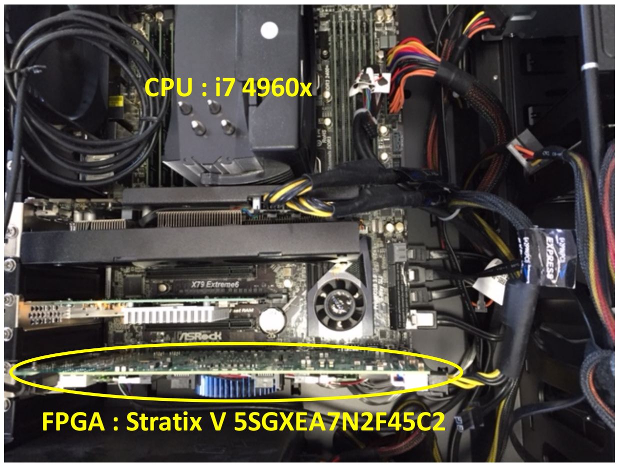
Table I shows the FPGA resource utilization details. The most used resource is ALMs (adaptive logic modules) and 36% of the total ALMs are used for the implementation. From those resources, around 19% of the resources are used for the I/O implementations such as PCIe and memory controllers, etc. As a result, we can increase the parallel processing by 4 times using 80% of the FPGA resources. The measured clock frequency of the accelerator is 202 MHz.

Table II shows the processing times of one iteration on CPU and FPGA. According to the results, the non-bonded force calculation takes 79% of the total CPU processing time. We implemented the force computation method shown in Algorithm 1 without any changes on FPGA using OpenCL. However, the processing time on FPGA is more than 129 times larger compared to that in CPU. Therefore, just using the same code on hardware does not provide any acceleration and in fact increases the processing time.

Table III shows the processing time comparison when



(a) System architecture.



(b) Implemented system.

Fig. 5. CPU-FPGA heterogeneous processing system for molecular dynamic simulations.

TABLE III
COMPARISON OF THE PROCESSING TIME USING PROPOSED ATOM-PAIR LIST BASED IMPLEMENTATION.

	CPU	**FPGA**	**Speed-up**
Measured results	0.68 s	0.14 s	4.6
Estimation based on 80% resource usage	0.68 s	0.037 s	18.4 (maximum)

the proposed atom-pair list based method is implemented on FPGA using OpenCL. We achieved a speed-up of 4.6 times compared to CPU implementation. The work in [11] reports 11.1 times speed-up using a single FPGA, while the work in [18] reports over 13 times speed-up using multiple FPGAs. However, direct comparison is difficult since both CPUs and FPGAs of the previous works are quite different from what we have used. In our implementation, we used only 36% of the FPGA resources. If we used upto 80 % of the FPGA resources, we can theoretically increase the speed-up to 18.4 % if memory bandwidth permits. This evaluation shows that a considerable speed-up can be achieved using OpenCL implementation.

Although the processing time is reduced, frequent data transfers between CPU and FPGA is still a problem. This problem can be solved by using SoC (system-on-chip) based OpenCL capable FPGAs, which will be released in near future. Such a system contains a multicore CPU and an FPGA on the same chip. Since both host and device are on the same chip, PCI express based data transfers are no longer required. We can use on-board data transfers which are much faster. If we use the shared memory for both host and the device, we can completely eliminate the data transfers.

V. CONCLUSION

We propose an FPGA Accelerator for MD simulations using OpenCL. We used an atom-pair list to describe the force computation using a single loop in OpenCL. This allows AOC (Altera offline compiler) to automatically generate an efficient pipelined architecture. We achieved over 4.6 times speed-up compared to CPU implementation by using only 36% of the FPGA resources. Maximum of 18.4 times speed-up is possible by assuming an 80% resource utilization. Such a performance is similar to an HDL-designed custom accelerator.

Since the proposed architecture is completely designed by software, the same program code can be reused by recompiling it for any OpenCL capable FPGA board. We can also implement any future algorithm change by just updating the software and recompiling it by using just few hours of design time. However, the data transfers between CPU and FPGA is still a problem. This problem can be solved by future SoC based FPGA boards that contain a multicore CPU and an FPGA on he same chip. Therefore, PCI express based data transfers can be replaced by much faster on-board data transfers. We may also able to use shared memory to completely eliminate data transfers.

ACKNOWLEDGMENT

This work is supported by MEXT KAKENHI Grant Number 15K15958.

REFERENCES

[1] D. C. Rapaport, *The art of molecular dynamics simulation*. Cambridge university press, 2004.
[2] D. A. Case, T. E. Cheatham, T. Darden, H. Gohlke, R. Luo, K. M. Merz, A. Onufriev, C. Simmerling, B. Wang, and R. J. Woods, "The amber biomolecular simulation programs," *Journal of computational chemistry*, vol. 26, no. 16, pp. 1668–1688, 2005.
[3] K. J. Bowers, E. Chow, H. Xu, R. O. Dror, M. P. Eastwood, B. A. Gregersen, J. L. Klepeis, I. Kolossvary, M. A. Moraes, F. D. Sacerdoti, *et al.*, "Scalable algorithms for molecular dynamics simulations on commodity clusters," in *SC 2006 Conference, Proceedings of the ACM/IEEE*, pp. 43–43, IEEE, 2006.
[4] S. Pronk, S. Páll, R. Schulz, P. Larsson, P. Bjelkmar, R. Apostolov, M. R. Shirts, J. C. Smith, P. M. Kasson, D. van der Spoel, *et al.*, "Gromacs 4.5: a high-throughput and highly parallel open source molecular simulation toolkit," *Bioinformatics*, vol. 29, no. 7, pp. 845–854, 2013.
[5] S. Plimpton, P. Crozier, and A. Thompson, "LAMMPS-large-scale atomic/molecular massively parallel simulator," *Sandia National Laboratories*, vol. 18, 2007.
[6] T. Narumi, Y. Ohno, N. Okimoto, A. Suenaga, R. Yanai, and M. Taiji, "A high-speed special-purpose computer for molecular dynamics simulations: MDGRAPE-3," in *NIC Workshop*, vol. 34, pp. 29–36, 2006.
[7] D. E. Shaw, M. M. Deneroff, R. O. Dror, J. S. Kuskin, R. H. Larson, J. K. Salmon, C. Young, B. Batson, K. J. Bowers, J. C. Chao, *et al.*, "Anton, a special-purpose machine for molecular dynamics simulation," *Communications of the ACM*, vol. 51, no. 7, pp. 91–97, 2008.

[8] D. E. Shaw, J. Grossman, J. A. Bank, B. Batson, J. A. Butts, J. C. Chao, M. M. Deneroff, R. O. Dror, A. Even, C. H. Fenton, *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*, pp. 41–53, IEEE Press, 2014.

[9] E. Cho, A. G. Bourgeois, and F. Tan, "An FPGA design to achieve fast and accurate results for molecular dynamics simulations," in *Parallel and Distributed Processing and Applications*, pp. 256–267, Springer, 2007.

[10] M. Chiu and M. C. Herbordt, "Molecular dynamics simulations on high-performance reconfigurable computing systems," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 3, no. 4, p. 23, 2010.

[11] M. A. Khan, *Scalable molecular dynamics simulation using FPGAs and multicore processors*. PhD thesis, Boston University College of Engineering, 2013.

[12] "Altera SDK for OpenCL." https://www.altera.com/products/design-software/embedded-software-developers/opencl/overview.html, 2016.

[13] "The open standard for parallel programming of heterogeneous systems." https://www.khronos.org/opencl/, 2015.

[14] S. Tatsumi, M. Hariyama, M. Miura, K. Ito, and T. Aoki, "OpenCL-based design of an FPGA accelerator for phase-based correspondence matching," in *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*, p. 90, 2015.

[15] N. Suda, V. Chandra, G. Dasika, A. Mohanty, Y. Ma, S. Vrudhula, J.-s. Seo, and Y. Cao, "Throughput-optimized OpenCL-based FPGA accelerator for large-scale convolutional neural networks," in *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '16, pp. 16–25, 2016.

[16] T. Mashimo, Y. Fukunishi, N. Kamiya, Y. Takano, I. Fukuda, and H. Nakamura, "Molecular dynamics simulations accelerated by GPU for biological macromolecules with a non-Ewald scheme for electrostatic interactions," *Journal of chemical theory and computation*, vol. 9, no. 12, pp. 5599–5609, 2013.

[17] "mypresto." http://presto.protein.osaka-u.ac.jp/myPresto4/index.php?lang=en, 2015.

[18] S. Kasap and K. Benkrid, "Parallel processor design and implementation for molecular dynamics simulations on a FPGA-based supercomputer," *Journal of Computers*, vol. 7, no. 6, pp. 1312–1328, 2012.