FPGA-Accelerated Tersoff Multi-body Potential for Molecular Dynamics Simulations

Ming Yuan[1], Qiang Liu[1], Quan Deng[1], Shengye Xiang[2], Lin Gan[2(✉)], Jinzhe Yang[3], Xiaohui Duan[2], Haohuan Fu[2,4], and Guangwen Yang[2,4]

[1] Tianjin Key Laboratory of Imaging and Sensing Microelectronic Technology School of Microelectronics, Tianjin University, Tianjing, China
[2] Department of Computer Science and Technology, Tsinghua University, Beijing, China
lingan@tsinghua.edu.cn
[3] Imperial College London, London, UK
[4] Zhejiang Lab, Hangzhou, China

Abstract. Molecular Dynamics simulation (MD) models the interactions of thousands to millions of particles through the iterative application of fundamental physics, and MD is one of the core methods in High-Performance Computing (HPC). However, the inherent weak scalability problem of force interactions renders MD simulation quite computationally intensive and challenging to scale. To this end, specialized FPGA-based accelerators have been proposed to solve this problem. In this work, we focus on many-body potentials on a single FPGA. Firstly, we proposed an efficient data transfer strategy to eliminate the latency between on-chip and off-chip memory. Then, the fixed-point description of data type is developed for computation to increase the utilization of on-chip resources. At last, a custom pipelined strategy is presented for $Tersoff$ to get a better simulation performance. Compared with a floating-point implementation based on NVIDIA 28080ti GPUs, our design based on Xilinx U200 FPGA is 1.2 times better.

Keywords: FPGA · Molecular dynamics simulations · Pipeline · Accelerator

1 Introduction

Molecular Dynamics (MD) simulations have been widely used in various aspects of life [1] and material sciences [2,3]. They have tremendously succeeded in many application areas during the past several decades. In particular, with the rapid development of the semiconductor industry in recent years, MD simulation, which contains multi-body potential formulations, such as $Tersoff$ potential, plays an essential role in the design space of new semiconductor materials [4] such as GaN, CdS, and TIOZ. The results from MD simulations provide helpful information for developing novel composite materials, reducing the extra experimental cost.

L. Gan et al. (Eds.): ARC 2022, LNCS 13569, pp. 17–31, 2022.
https://doi.org/10.1007/978-3-031-19983-7_2

Recent enhancements of High-Performance Computing (HPC) power, especially the development of supercomputers, provide an opportunity for complex MD simulations with complex potentials. Several available HPC clusters use accelerators such as graphics processing units (GPUs) to improve the performance of MD simulations, and they can make simulations of millions of particles with sufficient FLOPs. The ever-increasing demands of MD simulations even push the development of special-purpose supercomputers like Anton supercomputers [5–7] and MDGRAPE [8,9], but they are very inaccessible and are not widely used. While Anton supercomputers are based on ASICs, some novel systems have radical architectural changes (e.g., Sunway TaihuLight Supercomputer, Fugaku, and CrayXT3), resulting in a better performance of simulations, and it makes plenty of outstanding works in simulating biological systems [10,11]. However, as the MD simulations grow more complicated, traditional general-purpose chips can no longer meet the complex demands, such as memory, bandwidth, and computing efficiency. Furthermore, the computing efficiencies, the memory wall, and the power issues are becoming more and more serious when mapping MD simulations onto leading-edge supercomputing systems. There is a significant gap between the widely-used MD simulations and the current physical systems.

Fortunately, reconfigurable computing systems, such as those based on Field Programmable Gate Array (FPGA) technology offer a brand-new computing pattern that enables researchers to use a unique data-flow computing model to achieve better performance. Implementing molecular dynamics (MD) on FPGAs has also drawn substantial attention, and serval works [12,13] are made to tap the potential capacity of FPGAs for MD simulation. However, existing studies [14,15] are focused on the simple force interaction such as $Lennard{-}Jones(L{-}J)potential$, which contains a few variables to be computed, and it is not suitable for simulating new semiconductor materials. Furthermore, most of the FPGA implementations in the literature are resided entirely on-chip for the whole computation, completely removing the dependency on off-chip devices, resulting in a limited simulation scale during MD simulations.

In this work, we extend the MD simulation with the typical multi-body potential ($Tersoff$), and it has been widely used to analyze the three-body MD interaction between partially rigid particles such as silicon (Si). As is often the case when looking for cost-effective ASIC replacement, Field Programmable Gate Arrays (FPGAs) provide a viable alternative. A customized FPGA-based solution can significantly improve energy efficiency and power consumption compared to CPU and GPU clusters. We want to explore the feasibility of making a deeply-pipelined system for MD simulations on state-of-the-art FPGAs and propose an efficient accelerator for complex $Tersoff$ multi-body potential with high power efficiency.

To our best knowledge, most of the FPGA implementations in the literature are designed for two-body MD interactions with a limited simulation scale. This work is the first attempt to develop a large-scale MD simulation for three-body interactions ($Tersoff$) on a single-node FPGA system. Furthermore, our design for Tersoff potential is general enough, and we consider it has the potential to be

used for computing some similar problems in MD without completely redesigning the hardware. We also expect that the presented work can offer some ideas for designing and implementing similar applications on FPGAs .

Our significant contributions can be summarized as follows:

- We have presented an efficient data transfer strategy for large-scale MD simulations that overlapped computation and communication, improving the utilization of on-chip memories.
- We propose a fixed-point arithmetic for $Tersoff$ potential computation, which gives a tradeoff between resource and precision.
- We have proposed a custom pipelined computation engine for $Tersoff$ potential, which brought a significant performance improvement.

The remainder of this paper will first present the basic background information on $Tersoff$ potential and prior work for MD simulaions. Then, the data transfer design between the off-chip and on-chip memory, fixed-point quantization of $Tersoff$, and a custom dataflow computing model of $Tersoff$ will be elaborated. Following this, the results are presented and evaluated comparatively with different platforms. Finally, conclusions are detailed with plans for future work.

2 Background

2.1 Classical MD with $Tersoff$ Potential

The basic workflow of MD simulations consists of four essential parts: system initialization, neighbor list generation, force interactions, and motion update. Neighbor list generation and force interactions are much more time-consuming among the four parts. In the part of neighbor list generation, a cutoff distance is introduced, and both forces and energies between particles are assumed to be zero if the distance between two particles is beyond the cutoff distance. Cell linked list algorithm is used widely in modern MD simulations to build the neighbor list. In this algorithm, the simulation domain is partitioned into several cells, the edge of cells is equal to or larger than the cutoff distance, and there are 26 neighboring cells for each particle located in the home cell.

As mentioned before, a typical three-body potential ($Tersoff$) with N particles has a computational complexity of $O(N^3)$, which is far more complex than two-body algorithm. The total potential energy for the $Tersoff$ potential system can be written as $U = \frac{1}{2}\sum_i \sum_{j \neq i} U_{ij}$, where the energy U_{ij} can be written as

$$U_{ij} = f_C\left(r_{ij}\right)\left[f_R\left(r_{ij}\right) - b_{ij} f_A\left(r_{ij}\right)\right] \tag{1}$$

where f_C is a smooth cutoff function which contains trigonometric functions, $f_R(r) = Ae^{-\lambda r_{ij}}$ and $f_A(r) = Be^{-\mu r_{ij}}$ are the repulsive function and the attractive function, respectively.

Furthermore, the most crucial part is bond-order(ζ), and it takes the following forms:

$$b_{ij} = \left(1 + \beta^n \zeta_{ij}^n\right)^{-\frac{1}{2n}} \tag{2}$$

$$\zeta_{ij} = \sum_{k \neq i,j} f_C\left(r_{ik}\right) g_{ijk} \tag{3}$$

$$g_{ijk} = 1 + \frac{c^2}{d^2} - \frac{c^2}{d^2 + \left(h - \cos\theta_{ijk}\right)^2} \tag{4}$$

Here, A, B, β, n, c, d, λ, μ and h are parameters and θ_{ijk} is the angle formed by r_{ij} and r_{ik}.

2.2 Prior MD Work

In general, MD is one of the core methods in High-Performance Computing (HPC). Several well known high performance MD software packages (e.g. GROMACS [16], LAMMPS [17], AMBER [18], NAMD [19], CHARMM [20]) are making full use of modern HPC to achieve better performance. Furthermore, many supercomputers are used for MD simulations to get better performance. On Sunway TaihuLight supercomputer, Duan et al. [21] use the full supercomputer nodes for MD simulations, achieving a tremendous performance of over 2.43 PFlops. Meanwhile, in the year 2020, a machine learning-based simulation protocol [22] for MD can simulate over 100 million atoms more than $1ns$ per day on the Summit supercomputer, and this work can attain 91 PFLOPS (45.5% of the peak) in double precision and 162/275 PFLOPS in mixed-single/half-precision.

During the past several decades, Field Programmable Gate Arrays (FPGAs) have been explored as efficient accelerators for MD simulations. Most FPGA-based studies [23–25] only target the particle-particle (PP) computation to accelerate MD simulations, since they make up for over 92% of the runtime of simulations. However, they only accelerate non-bonded pair interactions on the FPGAs and do not use inter-FPGA communication. Although they can accelerate the interactions, the overall system is not competitive due to a limited bandwidth between the host processor and the BRAM on the FPGA card. The work proposed by Benjamin Humphries et al. [26,27] shows that the widely-used 3D FFTs in the order of 64^3 can be successfully presented on single FPGAs, which achieve a competitive speed within a few 100 µs. Kasap et al. [28] make the first attempt to propose a production-level MD accelerator using FPGA-based parallel computers. Another work [13] presents the first full-scale FPGA-based simulation engine implemented on a single FPGA and shows that its performance is competitive with a GPU.

3 Efficient Data Transfer

3.1 Bandwidth-Friendly Particle Mapping

When it comes to large-scale particle mapping, off-chip memory (DRR4/HBM) can be used well to store particle information, especially for large-scale MD simulations. Due to the bandwidth-to-compute nature of MD simulations, it takes a large number of particles for few force interactions, and the performance is directly bound up with the available bandwidth offered by FPGAs. However, the random memory access nature of particle mapping presents an important issue: access flexibility. It is essential to offer a bandwidth-friendly particle mapping for the following computation.

Algorithm 1. A Design for Accelerating Particle Mapping

Require: $FETCH$: get the data of particles
N_{cell}: numbers of cell in in the x, y, z direction
$Cell_{size}(N_{cell})$: number of particles in each cell
$Cell_{offsets}(N_{cell})$: new index of particles in each cell
$Cell_{ptr}(N_{cell}+1)$: the whole number of particles in the first N_{cell} cells
Ensure: Resorted Particle Position: X(N),Y(N),Z(N)

for $i \in atoms$ **do** ▷ Assign and count cell index
  $FETCH(Data(i))$
  $k \leftarrow cell\ index\ of\ particle\ i$
  $Cell_{size}(k) \leftarrow Cell_{size}(k) + +$
  $Cell_{offsets}(k) \leftarrow Cell_{offsets} + +$
$Cell_{acc} \leftarrow 0$
for i in range $(0, N_{cell})$ **do**
  $Cell_{acc} \leftarrow Cell_{acc} + Cell_{size}(i)$
  $Cell_{ptr}(i) \leftarrow Cell_{acc}$
$Cell_{ptr}(N_{cell}) \leftarrow Cell_{acc}$
for $i \in atoms$ **do** ▷ Linear reorder
  $k \leftarrow cell\ index\ of\ atom\ i$
  $j \leftarrow Cell_{offsets}(k)$
  $base \leftarrow Cell_{ptr}(k)$
  $X(base + j) \leftarrow i$
  $Y(base + j) \leftarrow i$
  $Z(base + j) \leftarrow i$

Since particles are randomly initialized, the particles often cannot be stored continuously in off-chip memory. The subsequent batch data transfer will bring the problem of discrete memory access and reduce bandwidth utilization. To solve this problem, we propose a bandwidth-friendly data mapping design in this work. As we adopt the cell linked list algorithm mentioned before, 27 cells (1 home cell and 26 neighboring cells) are sent to FPGAs for each timestep computation. To increase the bandwidth utilization and decrease the data transfer

latency, we should map the memory locations of potential neighboring cells and the home cell as closely as possible.

The procedure of data mapping is shown in Algorithm 1. The first loop (lines 1–5) computes the cell index of each particle i. Then, the second loop (lines 6–9) accumulates the number of particles sent to on-chip memory for each batch. At last, the third loop (lines 11–17) makes a linear distribution of particles. By adopting this method, particles in the same home cell or neighboring cells will be located in the off-chip memory with a sequential access pattern, and no explicit performance degradation can be seen in simulations.

3.2 Zigzagging Buffer Design

As the scales of MD simulations increase rapidly, the need for more FPGA on-chip resources, especially BRAMs, becomes evident. However, the main issue becomes latency with more than enough storage offered by off-chip memory, such as HBMs or GDDR. Therefore, an efficient strategy is pursued to make data transfer between on-chip memory and off-chip devices, allowing for overlapping computation and communication.

In general, the on-chip buffer strategy is always used for the prefetch design based on FPGA, which efficiently narrows the gap between data transfer and on-chip computation. Similarly, the on-chip buffer design is an essential strategy used in MD simulations. To order to utilize the architectural compute resources fully, the on-chip buffer is optimized to the minimum size and only stores the data if reused in subsequent computations. Thus, we propose a zigzagging buffer design to meet the requirement.

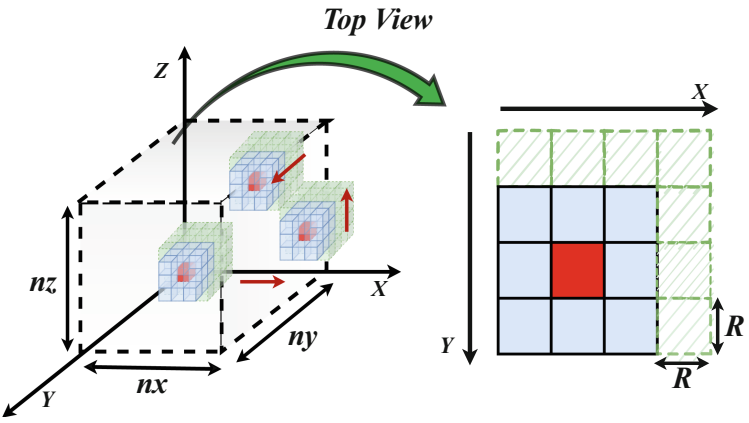

Fig. 1. Zigzagging buffer for MD simulaions.

As is shown in Fig. 1, $(0, 0, 0)$ is denoted for the cells with minimum coordinates, nx, ny, and nz indicate the number of cells in the X, Y and Z directions,

respectively. R represents the following phase's atoms to be computed, which are only loaded but not computed. We assume the X-axis as the most frequently varying dimension, followed by the Y- and Z-directions. The on-chip buffer is employed to store multiple cells of particles for the following computation. For each batch of data ($3 \times 3 \times 3$ cells) needed to be computed, an on-chip buffer with ($4 \times 4 \times 4$ cells) is loaded into on-chip memory, prefetch a cell data in the X, Y and Z directions, respectively. The prefetched buffer will move on the $X - Y$ plane, then prefetch the data along the Z axis in a zigzag motion. Computation is performed on zigzagging buffer data within the boundaries.

4 Fixed-Point Design

The practical algorithm using fixed-point arithmetic operation can significantly reduce the area and power consumption and obtain a cost-effective design. For brevity, we use the notation Fixed (IWL, FWL) to denote a fixed-point representation. IWL and FWL are integer word-length and fractional part word-length. The IWL optimizations determine the dynamic data range, while FWL optimizations consist of the numerical accuracy analysis.

4.1 Dynamic Range Analysis

Generally, overflow is quite dangerous for any practical applications in numerical simulation. Although a direct correlation between the application quality and the overflow probability is hardly determined, the dynamic range estimation usually determines the minimum and maximum values and computes the minimum number of bits for the integer part.

Most of the variables contained in $Tersoff$ potential are only influenced by the input distance (r_{ij}). Their range is easy to track, and the variables only change to a small degree. However, as $Tersoff$ potential contains plenty of transcendental functions computation, such as exp and pow, the final range is hard to decide on after going through the calculation of the transcendental functions. For example, when calculating the value of Bonded-Order ($\zeta = exp(lam3 * (r_{ij} - r_{ik})^3)$), where $lam3$ is a constant parameter, r_{ij} and r_{ik} are the distance between different particles. If taking the method of Extreme Values Theory [29], the maximum range of bonded-order (ζ) is up to 263428, resulting in the IWL being 20, which is quite expensive to set such a long word length for bonded-order (ζ). Hence, the first problem is whether it is necessary to cover the absolute theoretical bounds for IWL.

Considering that the distance r is the only input value for MD simulations, it is essential to analyze the input distribution carefully. Figure 2 shows a statistic of distribution of $|r_{ij}$ - $r_{ik}|$ and r for a system containing $5k$ particles, over 100 K iterations. It is clear that most values of $|r_{ij}$ - $r_{ik}|$ and r concentrate in a fixed interval, respectively. The maximum value and minimum value of $|r_{ij}$ - $r_{ik}|$ is 1.514 and 0.512, respectively. According to the formula of bonded-order (ζ) discussed before, IWL of ζ is limited to 10, much smaller than the absolute

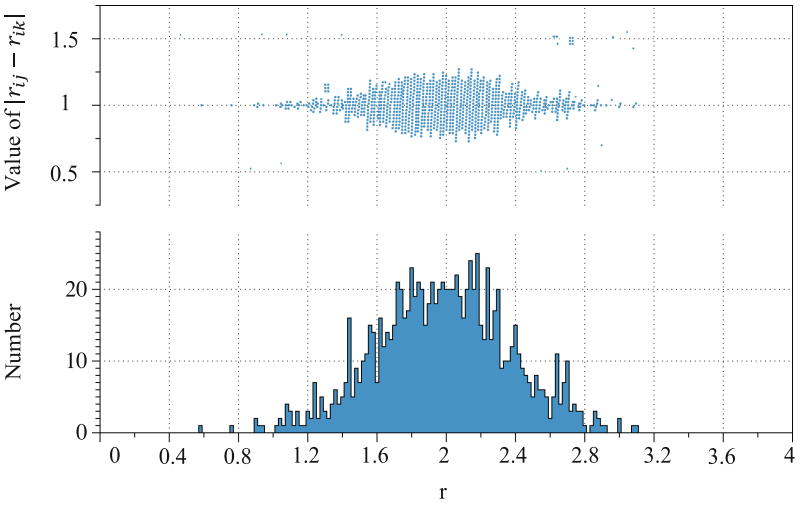

Fig. 2. The distribution of r_{ij}-r_{ik} and r

theoretical bounds. We also apply a similar approach to determine the integer bit width for the other fixed-point variables in the algorithm.

4.2 Precision Analysis

Computation in fixed-point arithmetic has limited accuracy and generates quantization error at the output. The quantization of fixed-point error is considered a noise added to the result and evaluated by the difference between the output with different precision. Therefore, verifying that the algorithm's fixed-point arithmetic behavior is modified within a reasonable limit is necessary. Thus, the second problem is whether to determine the suitable FWL between the needed accuracy and the circuit cost.

Generally, accuracies of the final results should be guaranteed before we can apply the fixed-point strategy. However, it is difficult to model the link between the application quality and error occurrence probability in MD simulations. Hence, in order to determine the impact on final accuracy caused by quantization error, we propose a bit-width optimization through bit-accurate simulations for different bit-width configurations. In this work, we find an essential indicator (relative energy error) for the quick estimation of the accuracy from [30]. If the relative energy error is more extensive than 0.1%, the final result will no longer be more accurate than the baseline.

During the process of MD simulation, the force interaction F is a critical variable that needs to be quantized. We explore a set of different bit widths for F and observe the dynamic trend of the relative error and the on-chip resource cost. According to the formulations of F, the maximum IWF of F is 8 due to the IWF of ζ is set as 10 . Hence, to analyze the impact of different FWL

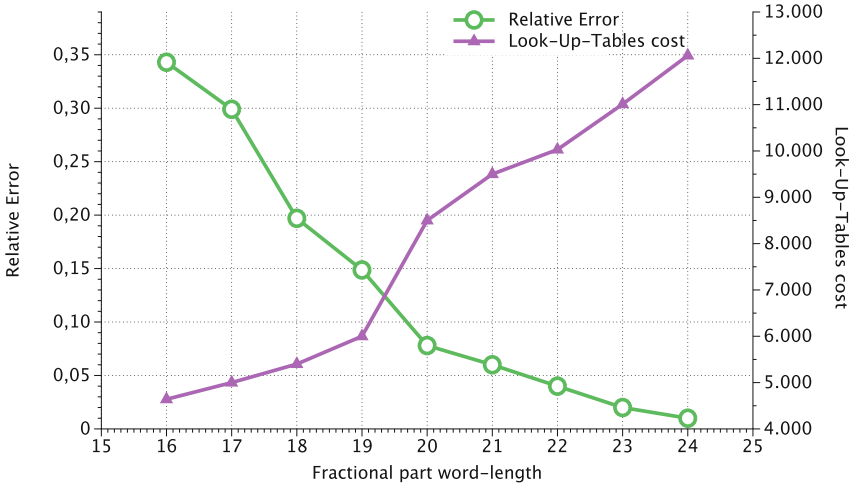

Fig. 3. The relative error and resource cost of LUTs according to different FWL of F .

of force (F) on the whole system, we explore the FWL of F from 24 to 16. From Fig. 3, we observe a similar relative error of different bit-width as with the baseline, ranging from 24-bits to 16-bits, and the relative error meets the requirement of 0.1% when the bit-width is larger than 20. However, when we further reduce the bit-width of F, we see a surge of the relative error to a level far above the required 0.1%. The sharp accuracy reduction at the bit-width of F to 20 indicates the precision threshold of the $Tersoff$. When the bit width of data decreases and cannot satisfy the precision threshold, the accuracy will break sharply. On the resource cost side, the bit-width of 20 is also a suitable choice that reduces the LUT usage from around 20000 to 10050 of the total capacity FPGA.

5 Custom Dataflow Design

Since the bandwidth-friendly data mapping strategy and fixed-point quantization design are proposed in Sect. 3 and Sect. 4, more attention should be paid to improving the performance of force interactions. Considering the dataflow architecture of FPGA, a custom pipelined strategy for $Tersoff$ interaction can be taken to improve the performance.

In this Algorithm 2, the overall force of particle i contains two parts: repulsive force and attractive force. After generating the neighbor list, the short-range repulsive and attractive forces will be computed quickly. However, adopting the original algorithm of computing $Tersoff$ potential is quite expensive. Firstly, the original method will require an almost triple computation workload ($0(N^3)$) for $Tersoff$ interaction, and all the computation parts are employed under the serial computing pattern. Secondly, the storage capacity of local value is far

beyond the on-chip memory size if we develop large-scale MD simulations. Since the on-chip memory size is limited, plenty of local values must frequently be swapped between on-chip and off-chip memory. It is a bad idea the design based on FPGAs.

Algorithm 2. Original method to calculate the $Tersoff$ potential

Require: r: The distance of the different particles
L: Neighbor lists of different particles
F_A, F_R:Attraction term and Repulsion term
Ensure: F: The force on the atom

```
1: procedure ALGO2
2:     for each i ∈ particles do                          ▷ Generate Neighbor list
3:         for each j ∈ particles do
4:             if r_(ij) < cutoff then
5:                 L ← L ∪ j
6:                 Store(L)
7:     for each i ∈ particles do
8:         for j ∈ L_i do
9:             if i ≠ j then
10:                F_i ← F_i + F_R(ij)                    ▷ Repulsion term
11:                UPDATE(F_i)
12:            for k ∈ L_i do
13:                if j ≠ k then
14:                    F_i ← F_i + FORCE(ζ_ijk)
15:                    UPDATE(F_i)
16:            for k ∈ L_i do
17:                if j ≠ k then
18:                    F_A ← ζ_ikj                        ▷ Attraction term
19:                    F_i ← F_i + F_A(j, i, k, ζ_ij)
20:                    UPDATE(F_i)
```

Thus, to solve this problem, a custom pipelined design is proposed for $Tersoff$ interaction. As the data prefetch strategy is adopted in this work, each batch contains 64 cells ($4 \times 4 \times 4$) for computation. As shown in Fig. 4, when the current home cell has completed the generation of the neighbor list, the process of force interactions for the current home cell and neighbor list generation for the next home cell can be operated simultaneously. Furthermore, considering the $\zeta(ijk)$ is shared in the repulsive term and attraction term at the level of k-loop, the local value $\zeta(ijk)$ can be pushed directly to the attraction term, which means when calculating the attraction term of the particle i, it is not necessary to wait for the repulsion term to be finished for all the particles. Hence, the custom computation pattern used in MD simulations can improve the performance well, allowing for overlapping different communication parts.

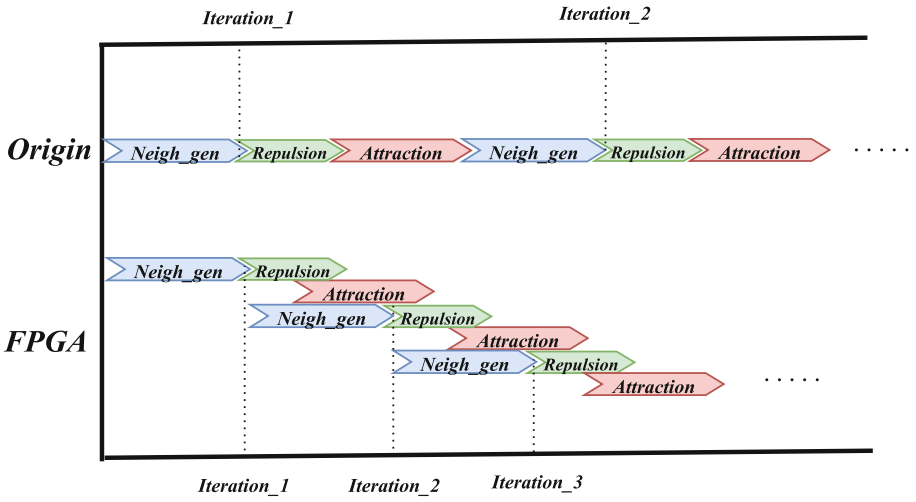

Fig. 4. Custom dataflow computation pattern for MD simulaions.

6 Evaluation

6.1 Environment Setup

We have implemented, tested, and verified our designs on Xilinx Alveo U200 FPGA, which has four DDR4 stacks. This high-end chip has 2586 CLBs, 6840 DSPs, 345.9 Mb block RAMs, and 2 QSFP28 (100GbE) interfaces, making it a good target for FPGA/MD. Then, our main evaluation metrics include overall performance, resource usage, power consumption, and energy evaluation.

Throughout the testing process, we select a typical crystalline structure of silicon (Si), equilibrated at temperature $T = 100\,\mathrm{K}$, to characterize the performance of our implementation. The atoms are highly mobile in the simulation system while only fluctuating around their equilibrium positions in the crystalline structure. The dataset has 512 K atoms, which is too large to fit in a single FPGA's BRAM. The dataset is constrained to a bounding box of $59.5 \times 51 \times 51\dot{A}$, with a cutoff radius of $4.2\dot{A}$. The simulation timestep is $2fs$.

6.2 Evaluation Performance

In this section, we will evaluate the performance of different platforms, including multi-core CPU, GPU, and the FPGA implementations of this work. Table 1 measures the performance of $Tersoff$ potential with 512 K atoms while using different devices. For a fair comparison, the benchmark is run on the device without any involvement of the host in the calculation. Firstly, Compared with CPU, our design implemented on FPGA has much better performance, 219.64× improvement for Intel Xeon 2690 v3 CPU with one core, and 14.69× improvement for Intel Xeon 2690 v3 CPU with eight cores.

As a cost-effective accelerator alternative in clouds and clusters, the low power consumption of FPGA is one of the advantages of HPCs. We evaluate the power consumption and power efficiency of different platforms. Due to the low power consumption for FPGA, the power efficiency of our design on U200 FPGA are 152.6× and 28.6× than that on Xeon CPU with one core and eight cores, respectively. Then, compared with GTX 2080ti GPU, our design implemented on FPGA has better performance, and the power efficiency is 4.1× than it. Much more evaluation needs to be done, but we believe these results to be promising.

Table 1. Evaluation performance.

Platform	Simulation rate	Speed up	Power	Power efficiency
Intel 2690 v3 1-core	1.04×10^{-2} (ns/day)	1	32 W	1
Intel 2690 v3 2-core	3.60×10^{-2} (ns/day)	3.7×	45W	2.2×
Intel 2690 v3 4-core	7.13×10^{-2} (ns/day)	7.1×	64W	3.55×
Intel 2690 v3 8-core	1.34×10^{-1} (ns/day)	14.9×	80W	5.96×
NVIDIA GTX 2080Ti GPU	2.17 (ns/day)	208.65×	184.6W	37.68×
Xilinx Alevo U200 FPGA	2.21 (ns/day)	219.80×	46.5W	152.2×

6.3 Resource Usage Evaluation

This section discusses the overall system resource utilization for $Tersoff$. As mentioned before, we propose an effective data transfer design to overlap computation and communication and improve the utilization of on-chip memories (BRAMs). Meanwhile, we find a rich design space for quantization of $Tersoff$ and propose a custom precision for force computation to reduce on-chip resource usage. Finally, we present a custom pipelined computation model for $Tersoff$, reducing the computational engine idling.

Table 2 lists the available resource of FPGA and several pipeline units that can fit onto a single FPGA chip. We note that our force pipeline for $Tersoff$ can include 64 pipelines. Due to the design of cell linked list, this design needs hundreds of memory modules, while the workload mapping requires each pipeline to accumulate the local variable on-chip. Because of this, a substantial on-chip memory is required. Compared with the standard floating-point implementation of $Tersoff$ based on FPGAs, our fixed-point design reduces LUT, BRAM, and DSP usage by 60.5%, 79.2%, and 74.1%, while the fixed-point design increase LUT, BRAM, and DSP use by 55.1%, 74.1%, and 48.1%, respectively.

6.4 Energy Evaluation

Energy evaluation adds complexity but is needed only every few iterations. We thus stream off the energy values computed by FPGA via PCIe every few iterations. A software simulator based on CPU is made to perform simulation on

Table 2. Overall system resource utilization

	Available	Force pipeline (float)	Force pipeline (fixed)
Pipelines	–	64	64
Kernel frequency (MHZ)	–	300	300
LUT	892K	542.88K (60.5%)	491.18K (55.0%)
FF	1.74M	0.98M (56.4%)	0.95M(53.2%)
DSP	6.84K	4.40K (74.1%)	3.20K (48.1%)
BRAM	35 MB	28.04 MB (79.2%)	26.04 MB (74.2%)
Latency	–	87	71
Power (W)	–	49.8	46.5

the same input dataset in single precision for validation. The energy waveform is shown in Fig. 5. Our software simulator's energy waveform matches our FPGA implementation with a slight variance.

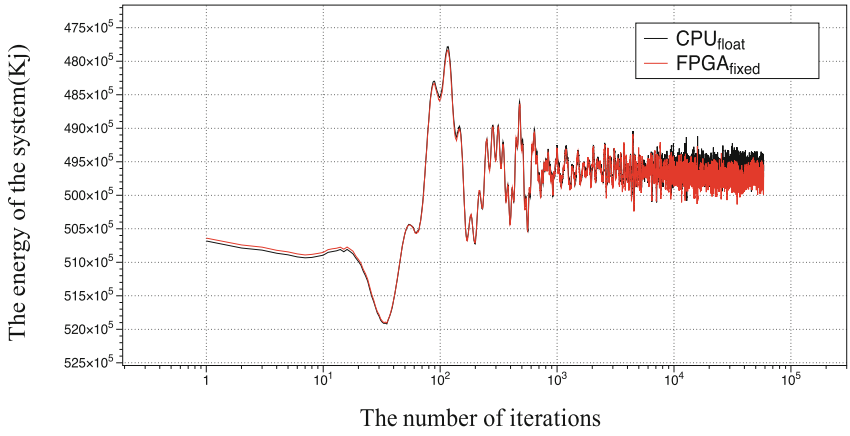

Fig. 5. Energy Waveform

7 Conclusion

In this work, we focus on many-body potentials ($Tersoff$) on a single FPGA with 512 K particles. Compared with conventional pair-wise potentials, the many-body potential ($Tersoff$) requires much more arithmetic operations and data dependency. Due to the limited on-chip resource of FPGA, several approaches are pursued to increase simulation performance, such as input/output throughputs, pipeline design, custom precision, and parallelism.

Compared with a floating-point implementation based on NVIDIA 28080ti GPUs, our design based on Xilinx U200 FPGA is 1.2× better, and the power efficiency is 4.1× than it. In the future, we would like to find ways to improve the performance further and verify FPGAs as promising candidates for both current and next-generation supercomputing architectures.

Acknowledgments. This work was supported in part by the National Natural Science Foundation of China (No. 62102114, U21B2031), and the Key Research Project of Zhejiang Lab (No. 2021PB0AC01).

References

1. Daggett, V.: Protein folding- simulation. Chem. Rev. **106**(5), 1898–1916 (2006)
2. Koyanagi, J., et al.: Evaluation of the mechanical properties of carbon fiber/polymer resin interfaces by molecular simulation. Adv. Compos. Mater. **28**(6), 639–652 (2019)
3. Jensen, F.: Introduction to computational chemistry. Wiley (2017)
4. Zhou, X.: Impact of molecular dynamics simulations on research and development of semiconductor materials. MRS Adv. **4**(61–62), 3381–3398 (2019)
5. Shaw, D.E., et al.: Anton, a special-purpose machine for molecular dynamics simulation. Commun. ACM **51**(7), 91–97 (2008)
6. Shaw, D.E., et al.: Anton 2: raising the bar for performance and programmability in a special-purpose molecular dynamics supercomputer. In: SC 2014: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE, pp. 41–53 (2014)
7. Shaw, D.E., et al.: Anton 3: twenty microseconds of molecular dynamics simulation before lunch. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–11 (2021)
8. Ohmura, I., et al.: MDGRAPE-4: a special-purpose computer system for molecular dynamics simulations. Philos. Trans. R. Soc. A Math. Phys. Eng. Sci. **372**(2021), 20130387 (2021)
9. Morimoto, G., et al.: Hardware acceleration of tensor-structured multilevel ewald summation method on MDGRAPE-4A, a special-purpose computer system for molecular dynamics simulations. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–15 (2021)
10. Zhang, T., et al.: SW_GROMACS: accelerate GROMACS on sunway TaihuLight. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–14 (2019)
11. Gao, P., et al.: Millimeter-scale and billion-atom reactive force field simulation on Sunway Taihulight. IEEE Trans. Parallel Distrib. Syst. **31**(12), 2954–2967 (2020)
12. Yang, C., et al.: Molecular dynamics range-limited force evaluation optimized for FPGAs. In: 2019 IEEE 30th International Conference on Application-specific Systems, Architectures and Processors (ASAP), vol. 2160, pp. 263–271. IEEE (2019)
13. Yang, C., et al.: Fully integrated FPGA molecular dynamics simulations. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–31 (2019)
14. Chen, Y.: OpenCL for HPC with FPGAs: case study in molecular electrostatics. In: IEEE High Performance Extreme Computing Conference (HPEC). IEEE, vol. 2017, pp. 1–8 (2017)

15. Cong, J., et al.: Revisiting FPGA acceleration of molecular dynamics simulation with dynamic data flow behavior in high-level synthesis. In: arXiv preprint arXiv:1611.04474 (2016)
16. Hess, B., et al.: GROMACS 4: algorithms for highly efficient, loadbalanced, and scalable molecular simulation. J Chem. Theory Comput. **4**(3), 435–447 (2008)
17. Plimpton, S.: Fast parallel algorithms for short-range molecular dynamics. J. Comput. Phys. **117**(1), 1–19 (1995)
18. Salomon-Ferrer, R., Case, D.A., Walker, R.C.: An overview of the Amber biomolecular simulation package. Wiley Interdisc. Rev. Comput. Mol. Sci. **3**(2), 198–210 (2013)
19. Phillips, J.C., et al.: Scalable molecular dynamics with NAMD. J. Comput. Chem. **26**(16), 1781–1802 (2005)
20. Brooks, B.R., et al.: CHARMM: the biomolecular simulation program. J. Comput. Chem. **30**(10), 1545–1614 (2009)
21. Duan, X., et al.: Redesigning LAMMPS for peta-scale and hundredbillion- atom simulation on Sunway TaihuLight. In: SC18: International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 148–159. IEEE (2018)
22. Jia, W., et al.: Pushing the limit of molecular dynamics with ab initio accuracy to 100 million atoms with machine learning. In: arXiv preprint arXiv:2005.00223 (2020)
23. Maliţa, M., Mihăiţă, M., M ştefan, G.: Molecular dynamics on fpga based accelerated processing units. In: MATEC Web of Conferences, vol. 125, p. 04012. EDP Sciences (2017)
24. Escobar, F.A., Chang, X., Valderrama, C.: Suitability analysis of FPGAs for heterogeneous platforms in HPC. IEEE Trans. Parallel Distrib. Syst. **27**(2), 600–612 (2015)
25. Khan, M.A., Chiu, M., Herbordt, M.C.: FPGA-Accelerated Molecular Dynamics. In: Vanderbauwhede, W., Benkrid, K. (eds.) High-Performance Computing Using FPGAs, pp. 105–135. Springer, New York (2013). https://doi.org/10.1007/978-1-4614-1791-0_4
26. Benjamin, H.: 3D FFTs on a Single FPGA. In: IEEE 22nd Annual International Symposium on Field-Programmable Custom Computing Machines. IEEE, vol. 2014, pp. 68–71 (2014)
27. Sheng, J., et al.: Design of 3D FFTs with FPGA clusters. In: 2014 IEEE High Performance Extreme Computing Conference (HPEC). IEEE, pp. 1–6 (2014)
28. Kasap, S., Benkrid, K.: Parallel processor design and implementation for molecular dynamics simulations on a FPGA-based supercomputer. JCP **7**(6), 1312–1328 (2012)
29. Chapoutot, A., Didier, L.S., Villers, F.: Range estimation of floating-point variables in simulink models. In: Proceedings of the 2012 Conference on Design and Architectures for Signal and Image Processing. IEEE, pp. 1–8 (2012)
30. Piana, S., Klepeis, J.L., Shaw, D.E.: Assessing the accuracy of physical models used in protein-folding simulations: quantitative evidence from long molecular dynamics simulations. Curr. Opin. Struct. Biol. **24**, 98–105 (2014)