Artificial Intelligence Applications And Innovations Vol. 1 No. 3 (2024)

Article history:
Received 7 March 2024
Revised 25 April 2024
Accepted 02 June 2024
Published online 01 July 2024

Design of a Greedy Algorithm for Non-Uniform Space Partitioning across Homogeneous FPGAs in Molecular Simulation

Faezeh Sadat. Mozneb[1], Kambiz. Rahbar[1*], Parvaneh. Asghari[2], Parand. Akhlaghi[1]

[1] South Tehran Branch, Islamic Azad University, Tehran, Iran
[2] Central Tehran Branch, Islamic Azad University, Tehran, Iran

*** Corresponding author email address**: k.rahbar@azad.ac.ir

A r t i c l e I n f o

Article type:
Original Research

How to cite this article:
Mozneb, F. S., Rahbar, K., Asghari, P., & Akhlaghi, P. (2024). Design of a Greedy Algorithm for Non-Uniform Space Partitioning across Homogeneous FPGAs in Molecular Simulation. *Artificial Intelligence Applications and Innovations, 1*(3), 9-19.
https://doi.org/10.61838/jaiai.1.3.2

A B S T R A C T

Efficient partitioning of the atomic space among parallel FPGAs is crucial for accelerating molecular simulations. Existing research has primarily focused on uniform partitioning, assuming a homogeneous distribution of atoms. However, in scenarios with non-uniform atomic distributions, these approaches may lead to suboptimal performance. This study investigates the impact of non-uniform atom distributions on molecular simulation performance across parallel FPGAs. We propose a novel space partitioning scheme that optimizes the distribution of atomic space among FPGAs, taking into account the spatial heterogeneity of atoms. Our evaluation demonstrates that the proposed scheme consistently outperforms uniform partitioning in terms of simulation speed across various spatial dimensions and atom counts, particularly in scenarios with non-uniform atom distributions.

Keywords: *Molecular simulation, Acceleration, Parallelization, Greedy algorithm.*

1. Introduction

Molecular simulations on a single computer often demand extensive computational time, spanning months or even years. To expedite these simulations, auxiliary processors like GPUs and FPGAs are employed. Previous studies [1, 2] have shown that using a single GPU or FPGA (or a small number of them) does not yield significant speedup compared to execution on a CPU. Consequently, parallelization becomes essential. However, the number of GPUs that can be utilized on a single computer is limited (typically two, with a maximum of fewer than 15). In contrast, a virtually unlimited number of FPGAs can be connected to a computer via a network. As a result, the use of parallel FPGAs has emerged as a prevalent method for accelerating molecular simulations.

An FPGA consists of input blocks, output blocks, and logic blocks. Before using an FPGA, its logic blocks are programmed to perform specific logical computations. During operation, input data enters the FPGA through input blocks, undergoes the programmed computations, and exits through output blocks.

The capacity or size of an FPGA refers to the number of logic blocks it contains. Larger FPGAs can implement more complex logic circuits but are also more expensive.

When performing molecular simulations on parallel FPGAs, the partitioning of atomic space among them becomes crucial. Existing research has primarily focused on scenarios where atoms are uniformly (or near-uniformly) distributed within the simulation space. However, our research has revealed that when atoms are non-uniformly distributed, existing partitioning schemes may not achieve acceptable simulation speeds. To address this issue, we propose a novel partitioning scheme that optimizes the distribution of space among parallel FPGAs, aiming to achieve high simulation speeds. Our evaluations demonstrate that the proposed scheme consistently outperforms previous approaches across various numbers of parallel FPGAs, particularly in scenarios with non-uniform atom distributions.

The remainder of this paper is organized as follows: In Section 2, we delve into a novel problem in the realm of simulation space partitioning and formally define it. Section 3 reviews prior work on accelerating molecular simulations. In Section 4, we design a near-optimal algorithm to address this problem. Section 5 evaluates the proposed algorithm in various scenarios and discusses the evaluation results. Then, Section 6 discusses the challenges of this research. Finally, we conclude this study in Section 7.

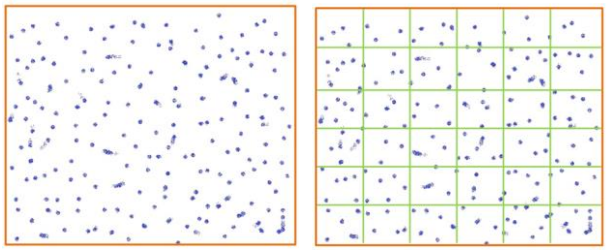
2. Problem Analysis and Formulation

To execute a molecular simulation on a processor or FPGA, existing research suggests partitioning the atomic space into small cubic cells [3]. The length of each side of these cubes is equal to the cutoff radius, which represents the distance within which an atom can exert an attractive force on its neighboring atom. In this study, we consider the size of each cubic cell to be fixed and equal to the cutoff radius. Previous research has shown that any other cell size would lead to reduced simulation speed and efficiency [3].

Now, if we intend to execute a molecular simulation on multiple FPGAs in parallel, we first need to divide the atomic space into small cubic cells. Following this, we need to partition the atomic space into sub-regions, each consisting of a number of cells. Then, each sub-region, along with its cells, is assigned to a separate FPGA. Each of the parallel FPGAs in the system processes a sub-region along with its internal cells.

2.1. Analysis in terms of the Number of Assigned Atoms

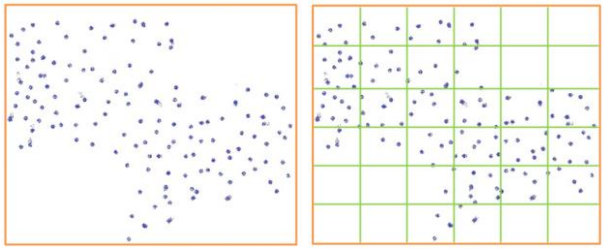
To investigate the problem, let's first consider the atoms depicted in Figure 1(a), which are uniformly distributed. This figure is drawn in two dimensions with a small number of atoms for better understanding and visualization, but the same concept can be extended to the actual three-dimensional case with a much larger number of atoms. To divide the space of these atoms among a number of FPGAs of equal size, we can simply divide the space into equal-sized sub-regions (as shown in Figure 1(b)) and assign each sub-region to an FPGA. This method of partitioning the simulation space, employed in existing schemes, is referred to as the uniform partitioning algorithm.



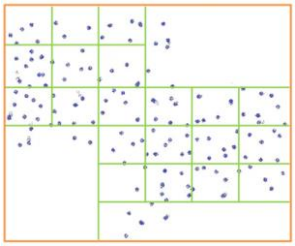
(a) Representation of atoms *(b) Division of space among FPGAs*

Figure 1. An example of atoms uniformly distributed in space

Now, consider the atoms depicted in Figure 2(a), which are non-uniformly distributed. To divide the space of these atoms among a number of FPGAs of equal size, one approach is to divide the space into equal-sized sub-regions (as shown in Figure 2(b)) and assign each sub-region to an FPGA. In this case, we observe that each FPGA is allocated the same area of simulation space, and the number of cells assigned to each FPGA is approximately equal. However, some FPGAs process only a few atoms, while others need to process a much larger number of atoms during the simulation. This disparity in the number of atoms assigned to FPGAs leads to significant differences in the processing time required at each time step across FPGAs. This difference, in turn, increases the overall processing time required by the system at each time step. The reason is that at any given time step, the total processing time for the entire system is determined by the processing time of the slowest FPGA.



(a) Representation of atoms *(b) Equal division of space among FPGAs*



(c) Unequal division of space among FPGAs

Figure 2. An example of atoms non-uniformly distributed in space

To divide the space of these atoms among a number of FPGAs of equal size, another approach is to partition the space into sub-regions of varying sizes (as shown in Figure 2(c)) such that each sub-region contains an equal number of atoms. If we assign these sub-regions to FPGAs, then each FPGA will not be allocated the same area of simulation space, and the number of cells assigned to each FPGA will not be equal. However, all FPGAs will process an equal number of atoms. This equality in the number of atoms assigned to FPGAs ensures that the processing time required at each time step across FPGAs is very similar. Consequently, compared to Figure 2(b), the processing time required at each time step in the slowest FPGA will be reduced. Therefore, the total processing time for the entire system will decrease.

2.2. *Analysis in terms of the Number of Neighboring Atoms Assigned to Different FPGAs*

When partitioning the simulation space among FPGAs, if two neighboring atoms are assigned to different FPGAs, then at each time step, those two FPGAs need to exchange the positions of these two atoms, and this process must be repeated at every time step. Now, if we have a large number of such neighboring atoms assigned to different FPGAs, the communication traffic and latency between FPGAs will increase excessively, significantly slowing down the simulation execution.

Therefore, another point to consider when partitioning the simulation space among FPGAs is to limit the number of neighboring atoms assigned to different FPGAs to an acceptable level.

2.3. *Problem Definition*

Based on the discussions in this section, we conclude that the following two quantities need to be minimized when assigning the simulation space to parallel FPGAs: 1) The maximum number of atoms assigned to a single FPGA, 2) The number of pairs of neighboring atoms assigned to different FPGAs.

Now, we define the following new problem in the context of partitioning the atomic space among FPGAs:

Problem 1: Consider a three-dimensional space of atoms called A. Assume we have N FPGAs with equal capacity M. Partition space A among the FPGAs such that the following two quantities are minimized, resulting in the minimum total processing time for atoms in those FPGAs at each time step: 1) The maximum number of atoms assigned to a single FPGA, 2) The number of pairs of neighboring atoms assigned to different FPGAs.

2.4. *Problem Requirements*

The space assigned to each FPGA should be a simple rectangular cuboid to simplify the geometric calculations that the FPGA needs to perform during molecular simulation execution, thereby reducing execution time. If the space assigned to each FPGA were an irregular shape with many sides, the number of comparisons required for geometric calculations within the FPGA would increase, leading to longer execution times.

A general assumption in executing molecular simulations on parallel FPGAs is that the operational space of each FPGA is programmed and stored as a simple rectangular cuboid with 12 sides in the internal circuitry of that FPGA. If we were to increase the number of sides of the space, this general assumption would be violated, and we would be forced to modify and complicate the internal structure and calculations of the FPGA.

To balance the computational load among FPGAs, the number of atoms assigned to different FPGAs should be approximately equal. Moreover, the complexity of the space partitioning algorithm should be manageable.

To implement the defined problem, a central computer is needed to manage the FPGAs and perform

the space partitioning calculations centrally. Performing space partitioning by the FPGAs themselves in a distributed manner would increase the algorithm's complexity and reduce its accuracy. Additionally, it would consume internal FPGA blocks.

3. Related Work

Numerous software programs have been developed for molecular simulations on computers. Among the most widely used are NAMD [4], GROMACS [5], and OpenMM [6]. However, these programs are limited to execution on CPUs and GPUs, which may not offer the highest execution speeds. This limitation makes them suitable for simulations with a small number of atoms (less than 10,000) and short simulation durations (less than 10 seconds), where execution times are acceptable. But for longer simulations with a larger number of atoms, the execution time can extend to months or even years, which is impractical. Research such as [1, 7] has shown that this lengthy execution time persists even when running on a powerful GPU.

To address this issue, some researchers have proposed schemes to execute molecular simulations on FPGAs. However, their results indicated that the long execution times remained even with a powerful FPGA. Subsequently, other researchers proposed schemes to execute molecular simulations on multiple parallel FPGAs. Their results demonstrated that this approach could significantly reduce execution times. However, the degree of reduction depends on the number of FPGAs used and the optimality of the partitioning scheme.

Researchers in [8] proposed a scheme for implementing the LAMMPS molecular simulation software on FPGAs. Their proposed scheme, implemented on two FPGA accelerators, achieved a 13x speedup compared to the software implementation on a CPU.

The book [9] discusses the details of implementing molecular simulations on FPGAs and demonstrates how to design a processing pipeline in an FPGA for this purpose.

Researchers in [10] implemented molecular simulations using OpenCL in the C language on an FPGA called Stratix. Their evaluation showed that their FPGA implementation was about 4.6 times faster than execution on a CPU.

In [11], researchers implemented the molecular simulation algorithm of OpenMM on four FPGAs, but this study was only experimental and did not achieve satisfactory performance.

Researchers in [12] fully implemented the molecular simulation algorithm on a single FPGA. Additionally, researchers in [2] experimented with molecular simulations on FPGAs. In [13], a hardware design based on FPGAs was proposed to accelerate mRNA molecule simulations.

Researchers in [14] implemented parts of the molecular simulation algorithm on Intel FPGAs using OpenCL, which can program FPGAs. Their evaluation showed that using FPGAs offered shorter execution times compared to using GPUs in molecular simulations.

The Fourier transform is a computationally intensive part of molecular simulations. Researchers in [15] investigated the reduction in execution time achieved by implementing the three-dimensional Fourier transform on FPGAs.

Researchers in [16] examined the impact of bandwidth between FPGAs on the execution time of molecular simulations when using multiple parallel FPGAs. They proposed solutions to balance the bandwidth among FPGAs.

In [17], researchers explored scenarios where the number of FPGAs used reached tens and proposed solutions for implementing molecular simulation algorithms on dozens of FPGAs.

Researchers in [2] proposed a scheme for fully implementing molecular simulations on FPGAs with minimal use of the host computer. Their proposed scheme, implemented on a single FPGA, achieved a simulation rate of 1.4 microseconds of molecular simulation per day. This performance was about twice as good as that of a comparable GPU.

In [18], researchers implemented molecular simulations on a Xilinx-U200 FPGA. They proposed a data transfer strategy that eliminates latency between the FPGA's internal and external memory. Their evaluation showed that their FPGA implementation was about 1.2 times faster than the implementation on an NVIDIA-28080ti GPU.

All previous studies on molecular simulation execution on parallel FPGAs have only considered scenarios where atoms are uniformly (or near-uniformly) distributed in the simulation space. Therefore, they divide the simulation space equally among FPGAs. Our investigation in this research revealed that if atoms are non-uniformly distributed in the simulation space, equal division of the simulation space among FPGAs is not optimal, leading to unequal distribution of atoms among FPGAs. This non-uniformity results in an unequal distribution

of computational load among parallel FPGAs. Consequently, at each time step of the molecular simulation algorithm, the fastest FPGA has to wait for the slowest FPGA to finish its computations before the next time step can begin for all FPGAs. This reduces the simulation speed compared to the optimal case.

To date, none of the existing schemes have considered the non-uniform distribution of atoms in the simulation space or proposed a partitioning scheme for such scenarios. This research is the first work in this area.

4. Proposed Scheme

In this section, we present the proposed scheme for solving the problem and discuss its characteristics.

We assume we have n FPGAs of the same model and size. This assumption implies that the number of logic blocks within the FPGAs is equal, and they all have the same speed and capabilities. This assumption not only reduces the complexity of calculations related to atomic space partitioning but also simplifies the management of parallel FPGAs, ultimately increasing the final speed of the molecular simulation.

We assume a central computer connected to all FPGAs in the system, capable of managing them. Initially, this central computer reads the atomic space specifications and initial atom positions from an input file and then executes the proposed space partitioning algorithm. Subsequently, it assigns each resulting sub-region, along with its atoms, to a separate FPGA. After this, the FPGAs in the system can begin executing the molecular simulation.

We assume that the molecular simulation is fully executed on the parallel FPGAs, and the computer connected to the FPGAs is not involved in any part of the simulation. Our assumption is that the computer connected to the FPGAs is only used for setup, management, and control of the FPGAs.

The type of atoms is not relevant to the space partitioning algorithm. Atoms can be of the same type or of different types within the simulation space.

4.1. Proposed Algorithm

To find the optimal solution for the defined problem, all possible partitioning states of the simulation space with all possible sizes need to be calculated and evaluated. This process has exponential complexity and requires a very long execution time. Therefore, we design a polynomial-time complexity algorithm to solve the problem, which can achieve a near-optimal solution. The number of space partitioning states that the proposed algorithm needs to examine is much smaller than the optimal algorithm. Consequently, its complexity will be much lower.

Algorithm 1: (Molecular Simulation Space Partitioning)

1. Let A be the area of the atomic space in the simulation.
2. Let n be the number of parallel FPGAs available.
3. Let B = A/n
4. Divide the atomic space into small sub-regions with area B using the uniform partitioning algorithm.
5. Calculate the score of the current partitioning and store it in variable g.
6. Let S be the current partitioning scheme.
7. Let c = false
8. Repeat the following steps until c is true:
    a. Let c = true
    b. In partitioning scheme S, find all possible combinations of two neighboring sub-regions of S that, when combined, produce only one rectangular cuboid shape, i.e., no concave shape is produced.
    c. For each combination of sub-regions, do the following:
        i. Calculate the score of the space partitioning in that state.
        ii. If the obtained score is higher than g, then let c = false, store the partitioning score in g, and set parameter S to the partitioning scheme in that state.
9. Return the partitioning scheme S as the output of the algorithm.
10. End.

The use of condition c allows us to determine in each iteration of the inner loop of the algorithm whether continuing that loop is beneficial or not. If in one iteration of the inner loop, no new combination of sub-regions is found with a higher score, then we know that continuing that loop is no longer useful, and we have reached the best possible solution within the capabilities of this algorithm.

4.2. Definition of the Score Function

For a partitioning state of the simulation space to be optimal, the following objectives should be minimized in that state: 1) The maximum number of atoms assigned to a single FPGA, 2) The number of pairs of neighboring atoms in the entire simulation space that are assigned to different FPGAs.

To express these two objectives as numerical values for score calculation, we design the following function. In this function, we calculate two parameters named t and u. Since increasing the output score requires decreasing these two parameters, we define the output of the function as the inverse of these two parameters.

Function 1: Calculating the score for a partitioning state of the simulation space

1. Let g = 0
2. For all FPGAs in the system, find the FPGA with the maximum number of assigned atoms.
3. Set parameter t to the number of atoms assigned to that FPGA.
4. Set parameter u to the number of pairs of neighboring atoms in the entire simulation space that are assigned to different FPGAs.
5. Let g = 1 / (t + u)
6. Return the value of g as the output of the algorithm.
7. End.

4.3. Complexity Analysis

Now consider the proposed algorithm. In each iteration of the main loop, all possible combinations of two neighboring sub-regions need to be tested. Since the maximum number of sub-regions in this algorithm can be equal to the parameter n (i.e., the number of FPGAs), the number of possible combinations of two neighboring sub-regions is equal to (n(n-1))/2. Since the main loop of the program needs to be repeated n times in the worst case, the complexity of the proposed algorithm is n * (n(n-1))/2, which is of polynomial order.

Compared to the optimal solution algorithm, the proposed algorithm does not test all possible partitioning states of the space. Instead, it significantly reduces the number of states that need to be tested. In this research, we analyzed the problem and the characteristics of a good partitioning in the simulation space and gained a good understanding of them. Using this knowledge, we designed the proposed algorithm to only explore partitioning states that need to be tested and ignore unnecessary states. By testing a much smaller number of partitioning states compared to the optimal solution algorithm, the proposed algorithm achieves a near-optimal partitioning of the simulation space.

5. Evaluation of the Proposed Scheme

We have implemented a software program in Java to execute the proposed algorithm, which runs the algorithm and displays the space partitioning results graphically. The numerical evaluation outputs in this section are the same two parameters calculated in the score function. These outputs were obtained by running the algorithm in various molecular simulation scenarios. We compared the numerical results of the proposed algorithm with the uniform partitioning algorithm.

5.1. Evaluation Scenarios

To evaluate the proposed algorithm, we define several scenarios and specify the parameters for each of these scenarios. Table 1 describes these scenarios. The goal of scenario one is to evaluate the performance of the proposed algorithm in various space sizes while keeping the density of atoms in the space constant. The goal of scenario two is to evaluate the performance of the proposed algorithm in various atom densities within a space of fixed size. The goal of scenario three is to evaluate the performance of the proposed algorithm under various conditions of uniformity in atom distribution while keeping the space size and atom density constant. The goal of scenario four is to evaluate the performance of the proposed algorithm for different numbers of FPGAs in the system.

The outputs of our evaluations include the following: 1) Execution time of the algorithm on a personal computer, 2) Maximum number of atoms assigned to a single FPGA, 3) Variance of the number of atoms assigned to each FPGA, 4) Number of pairs of neighboring atoms in the entire simulation space assigned to different FPGAs, 5) Number of FPGAs required in the simulation, determined by the number of resulting sub-regions

The number of required FPGAs differs from the number of available FPGAs in the system. Initially, we have a certain number of FPGAs in the system among which we want to divide the atomic space. However, after executing the space partitioning algorithm, the same number of sub-regions may be identified, or the number of sub-regions may be slightly less than the number of

available FPGAs. If the latter occurs, it means the algorithm has determined that fewer FPGAs are needed. This happens in a space with sub-regions devoid of atoms, where using an FPGA for their simulation is unnecessary. Nevertheless, it is expected that the number of required FPGAs will always be equal to or very close to the number of available FPGAs.

Table 1. Defined Scenarios for Evaluating the Proposed Algorithm and Their Parameters

Input Parameter in Evaluation	Scenario 1	Scenario 2	Scenario 3	Scenario 4
Atom Space Size	50Å×50Å×50Å 100Å×100Å×100Å 200Å×200Å×200Å 400Å×400Å×400Å 800Å×800Å×800Å	200Å×200Å×200Å	200Å×200Å×200Å	200Å×200Å×200Å
Number of Atoms in the Space	1000000	1000000, 500000, 2000000, 4000000, 8000000	1000000	1000000
How Atoms are Spread in Space	Uniform	Uniform to Slightly Non-Uniform	Uniform to Completely Non-Uniform	Completely Non-Uniform to Uniform
Cutoff Radius	9Å	9Å	9Å	9Å
Cell Size	9Å×9Å×9Å	9Å×9Å×9Å	9Å×9Å×9Å	9Å×9Å×9Å
Number of FPGAs	5, 10, 20, 40, 80	20	20	20

5.2. *Numerical Results*

Figure 3 compares the execution time of the proposed algorithm with the uniform partitioning algorithm. The execution time of scenario one using the uniform partitioning algorithm is less than two seconds, while the execution time of the proposed algorithm reaches several seconds. The execution time of the proposed algorithm increases with the size of the space because the number of possible combinations of sub-regions increases with the size of the space. Although the execution time of the uniform partitioning algorithm also increases with the size of the space, the slope of this graph is less than that of the proposed algorithm. However, since the proposed algorithm is executed only once for space partitioning before the simulation begins, an execution time of a few seconds is acceptable. In contrast, using the proposed algorithm instead of the uniform partitioning algorithm leads to optimal space partitioning, resulting in a reduction of hours or even days in the execution time of the simulation by parallel FPGAs.

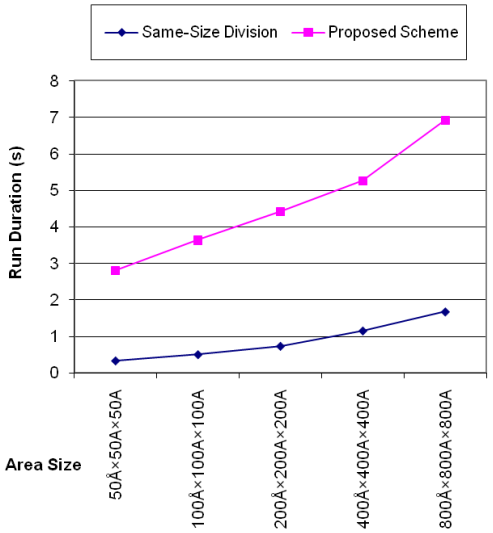


Figure 3. Execution Time of Space Partitioning Algorithms vs. Space Size (Scenario 1)

Figure 4 illustrates the maximum number of atoms assigned to a single FPGA as a function of space size. The number of atoms in the space is 1,000,000, and the number of FPGAs is 20. In the optimal scenario where atoms are equally divided among FPGAs, each FPGA would receive 50,000 atoms. Any number in the graph of Figure 4 that exceeds this value indicates a deviation from the optimal state. However, since the atoms are non-uniformly distributed in space and efforts are made to minimize the number of pairs of neighboring atoms assigned to different

FPGAs, it is natural to expect some deviation from the optimal state.

The uniform partitioning algorithm does not consider the location of atoms and divides the space solely based on area. As a result, the numbers obtained for this algorithm in Figure 4 show a significant deviation from the optimal value. In contrast, the proposed algorithm has been able to achieve near-optimal results with a smaller deviation from the optimal value. The graph in Figure 4 shows that the deviation from the optimal value increases with the size of the space. This characteristic is because atoms in a smaller space are closer to a uniform distribution and cannot deviate much from it. However, the same number of atoms in a larger space can be further apart and deviate more from the uniform distribution.

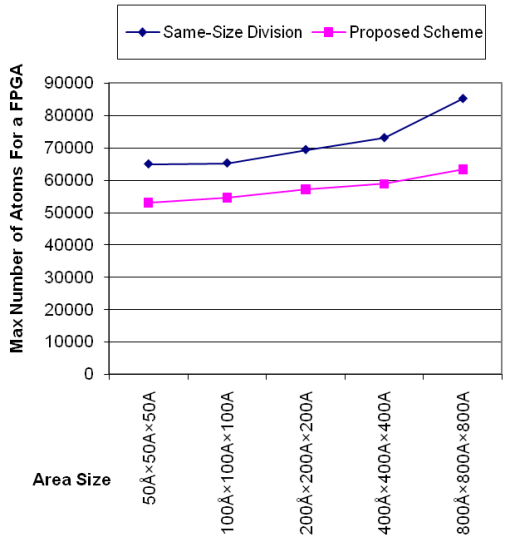


Figure 4. Maximum Number of Atoms Assigned to a Single FPGA vs. Space Size (Scenario 1)

Figure 5 shows the number of neighboring atom pairs in the simulation space that are assigned to two different FPGAs. The equal-size partitioning algorithm makes no attempt to reduce this output, while the proposed algorithm attempts to reduce both the outputs of Figure 4 and Figure 5.

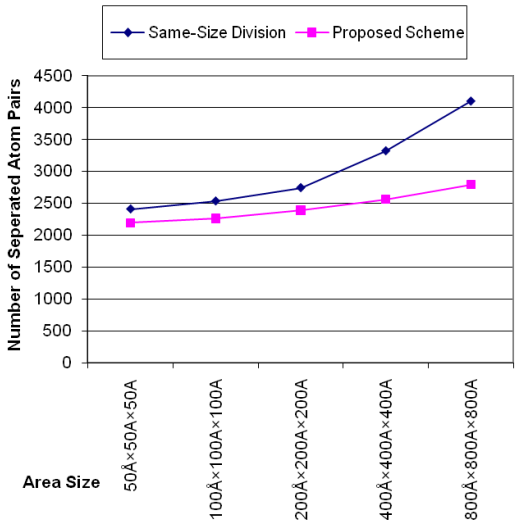


Figure 5. Number of neighboring atom pairs assigned to two different FPGAs as a function of space size (Scenario 1)

Figure 6 shows the number of FPGAs required for the simulation in Scenario 2. This graph shows that the proposed algorithm has been able to reduce the number of required FPGAs by combining some sub-sections when partitioning the space between FPGAs. However, this reduction has not occurred when the number of atoms is high. The reason is that if the number of atoms is high relative to the area of space, then no empty space can be found in that space and all sub-sections are full of atoms. As a result, the sub-sections cannot be combined and all available FPGAs in the system will be used.

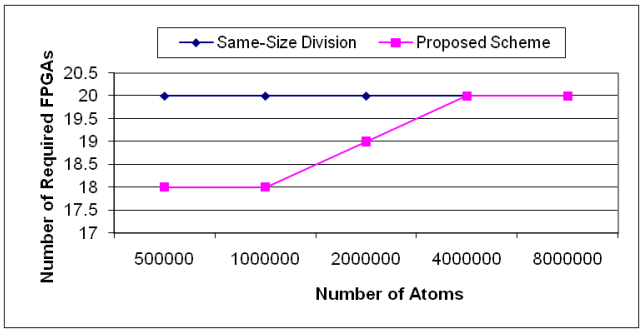


Figure 6. Number of required FPGAs as a function of space size (Scenario 1)

Figure 7 and Figure 8 show two parameters in Scenario 2 that the proposed algorithm tries to reduce both of them. In both of these figures, the graph of the proposed algorithm is closer to the optimal value compared to the equal-size division algorithm. As the number of atoms increases, it is natural that these two graphs increase. As the number of atoms in the same space

increases, the uneven distribution of atoms must be closer to the uniform distribution state. Therefore, the difference of the diagram from the optimal solution decreases.

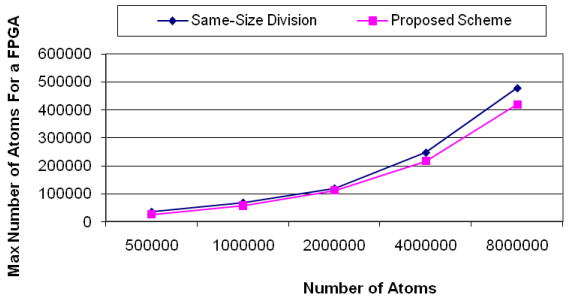


Figure 7. Maximum number of atoms assigned to an FPGA as a function of the number of atoms in the space (Scenario 2)

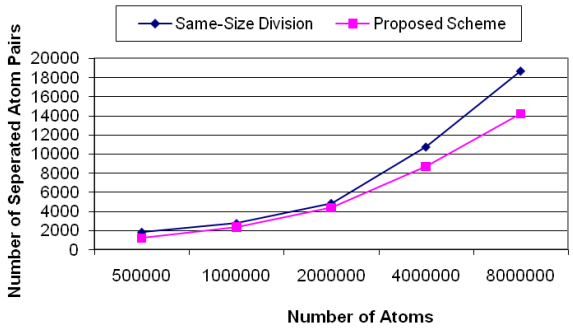


Figure 8. Number of neighboring atom pairs assigned to two different FPGAs as a function of the number of atoms in the space (Scenario 2)

Figure 9 shows the maximum number of atoms assigned to an FPGA in Scenario 3, while the size of the space and the number of atoms are constant. In this figure, as the degree of uniformity in the distribution of atoms decreases, the results of the equal division algorithm deviate further from the optimal state, and the superiority of the proposed algorithm compared to the equal division algorithm becomes more apparent.

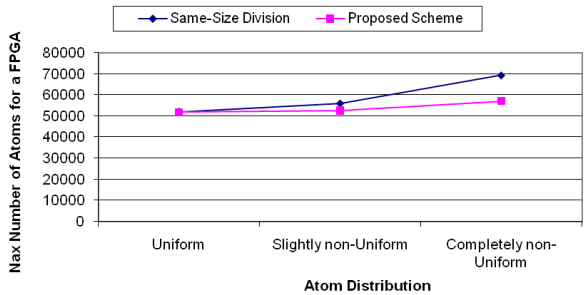


Figure 9. Maximum number of atoms assigned to an FPGA as a function of the distribution of atoms in space (Scenario 3)

Figure 10 shows the number of neighboring atom pairs assigned to two different FPGAs in scenario three, where the degree of uniformity in the distribution of atoms is different in each run of the space partitioning algorithm. In this figure, the graph of the proposed algorithm is closer to the optimal value compared to the equal division algorithm. Also, as the degree of uniformity in the distribution of atoms decreases, the results of the equal division algorithm and the proposed algorithm get closer to the optimal state. The reason is that if the atoms are not evenly distributed and are clustered in a few small sub-sections in the space, then assigning those few sub-sections to a small number of FPGAs can largely prevent neighboring atoms from being placed in different FPGAs.

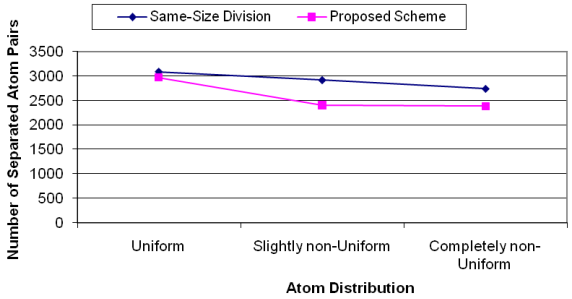


Figure 10. Number of neighboring atom pairs assigned to two different FPGAs as a function of the distribution of atoms in space (Scenario 3)

6. Discussion

Field-Programmable Gate Arrays (FPGAs) offer tremendous advantages in data processing due to their hardware-based operational paradigm, often showing improved speed compared to traditional processors. Their small size and economic viability encourage widespread use, thus making them an appealing platform for parallel molecular simulations.

However, two main hurdles hinder efficient parallelization on clusters of FPGAs: uneven load balancing and the communication overhead between FPGAs. The work herein addresses these problems through a novel algorithmic approach cantered around minimization.

While using high-speed interconnection networks may appear to be a straightforward way to reduce inter-FPGA communication delays, its feasibility decreases as the number of FPGAs increases. To be useful, large-scale molecular simulations require hundreds of FPGAs, making high-speed interconnects too costly. Thus, algorithmic inter-FPGA communication reduction is a more practical and less expensive approach.

The efficiency of the proposed partitioning algorithm depends on the distribution of particles in the simulation space. While it obviously decreases the communication between FPGAs for certain spatial distributions, its impact can be smaller for other distributions. Notably, such less-optimal distributions often represent those cases where existing algorithms struggle to achieve comparable performance improvements. This suggests that the presented approach is a solid solution, particularly for intricate particle distributions.

7. Conclusion

In this research, we have defined a new problem in the partitioning of molecular simulation space, which is tasked with dividing the space of atoms in a way that the atoms may be placed uniformly or non-uniformly in that space. The goal of this problem is to partition the space in such a way that the execution time of the molecular simulation is minimized. This problem tries to minimize two partitioning parameters to achieve this goal. Then, we proposed an algorithm with polynomial execution time that can reach a near-optimal solution by searching for various states of partitioning.

Also, we have implemented a software tool to run this algorithm, which executes the proposed algorithm and displays the partitioning result graphically. The evaluations performed with this software tool show the better performance of the proposed algorithm compared to the equal size division algorithm.

Authors' Contributions

All authors equally contributed to this study.

Declaration

In order to correct and improve the academic writing of our paper, we have used the language model ChatGPT.

Transparency Statement

Data are available for research purposes upon reasonable request to the corresponding author.

Acknowledgments

We would like to express our gratitude to all individuals helped us to do the project.

Declaration of Interest

The authors declare that they have no conflict of interest. The authors also declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Funding

According to the authors, this article has no financial support.

Ethical Considerations

The study placed a high emphasis on ethical considerations. Informed consent obtained from all participants, ensuring they are fully aware of the nature of the study and their role in it. Confidentiality strictly maintained, with data anonymized to protect individual privacy. The study adhered to the ethical guidelines for research with human subjects as outlined in the Declaration of Helsinki.

References

[1] S. Páll *et al.*, "Heterogeneous parallelization and acceleration of molecular dynamics simulations in GROMACS," *The Journal of Chemical Physics*, vol. 153, no. 13, 2020, doi: 10.1063/5.0018516.

[2] C. Yang *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)*, 2019, pp. 263-271, doi: 10.1109/ASAP.2019.00016.

[3] C. Wu *et al.*, "Optimized mappings for symmetric range-limited molecular force calculations on FPGAs," in *2022 32nd International Conference on Field-Programmable Logic and Applications (FPL)*, 2022, pp. 101-108, doi: 10.1109/FPL57034.2022.00026.

[4] J. C. Phillips *et al.*, "Scalable molecular dynamics with NAMD," *Journal of Computational Chemistry*, vol. 26, no. 16, pp. 1781-1802, 2005, doi: 10.1002/jcc.20289.

[5] M. J. Abraham *et al.*, "GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers," *SoftwareX*, vol. 1, pp. 19-25, 2015, doi: 10.1016/j.softx.2015.06.001.

[6] P. Eastman *et al.*, "OpenMM 7: Rapid development of high performance algorithms for molecular dynamics," *PLoS Computational Biology*, vol. 13, no. 7, p. e1005659, 2017, doi: 10.1371/journal.pcbi.1005659.

[7] D. Lu *et al.*, "86 PFLOPS deep potential molecular dynamics simulation of 100 million atoms with ab initio accuracy," *Computer Physics Communications*, vol. 259, p. 107624, 2021, doi: 10.1016/j.cpc.2020.107624.

[8] 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, doi: 10.4304/jcp.7.6.1312-1328.

[9] M. A. Khan, M. Chiu, and M. C. Herbordt, "FPGA-accelerated molecular dynamics," in *High-Performance Computing Using FPGAs*, 2013, pp. 105-135.

[10] H. M. Waidyasooriya, M. Hariyama, and K. Kasahara, "An FPGA accelerator for molecular dynamics simulation using OpenCL," *International Journal of Networked and Distributed Computing*, vol. 5, no. 1, pp. 52-61, 2017, doi: 10.2991/ijndc.2017.5.1.6.

[11] C. Pascoe, L. Stewart, B. W. Sherman, V. Sachdeva, and M. W. Herbordt, "Execution of complete molecular dynamics simulations on multiple FPGAs," in *2020 IEEE High Performance Extreme Computing Conference (HPEC)*, 2020, pp. 1-2, doi: 10.1109/HPEC43674.2020.9286155.

[12] C. Yang *et al.*, "Fully integrated FPGA molecular dynamics simulations," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2019, pp. 1-31, doi: 10.1145/3295500.3356179.

[13] D. Shallom, D. Naiger, S. Weiss, and T. Tuller, "Accelerating whole-cell simulations of mRNA translation using a dedicated hardware," *ACS Synthetic Biology*, vol. 10, no. 12, pp. 3489-3506, 2021, doi: 10.1021/acssynbio.1c00415.

[14] L. C. Stewart, C. Pascoe, B. W. Sherman, M. Herbordt, and V. Sachdeva, "Particle mesh Ewald for molecular dynamics in OpenCL on an FPGA cluster," in *arXiv preprint arXiv:2009.12617*, 2020, doi: 10.1109/FCCM51124.2021.00055.

[15] A. Ramaswami, T. Kenter, T. D. Kühne, and C. Plessl, "Efficient ab-initio molecular dynamic simulations by offloading fast Fourier transformations to FPGAs," in *2020 30th International Conference on Field-Programmable Logic and Applications (FPL)*, 2020, pp. 353-354, doi: 10.1109/FPL50879.2020.00065.

[16] C. Wu, T. Geng, C. Yang, V. Sachdeva, W. Sherman, and M. Herbordt, "A Communication-Efficient Multi-Chip Design for Range-Limited Molecular Dynamics," in *2020 IEEE High Performance Extreme Computing Conference (HPEC)*, 2020, pp. 9-19, doi: 10.1109/HPEC43674.2020.9286146.

[17] C. Wu *et al.*, "Upgrade of FPGA range-limited molecular dynamics to handle hundreds of processors," in *2021 IEEE 29th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2021, pp. 142-151, doi: 10.1109/FCCM51124.2021.00024.

[18] M. Yuan *et al.*, "FPGA-accelerated Tersoff multi-body potential for molecular dynamics simulations," in *International Symposium on Applied Reconfigurable Computing*, 2022, pp. 17-31, doi: 10.1007/978-3-031-19983-7_2.