Towards an FPGA-Based Dedicated Computer for Molecular Dynamics Simulations

Peter Hamm
Department of Chemistry, University of Zurich,
Winterthurerstrasse 190, CH-8057 Zurich, Switzerland,
and Department of Chemistry,
Graduate School of Science,
Kyoto University, Kyoto 606-8502, Japan
corresponding author: peter.hamm@chem.uzh.ch

(Dated: January 4, 2025)

First steps towards a molecular dynamics (MD) implementation in a cluster of field-programable gate arrays (FPGAs) are presented, reaching a simulation speed of a few microseconds/day. The nodes in this cluster are programmed into a mid-ranged FPGA (Artix 7 XC7A200T), interconnected as a 3D torus by fast optical links. The implemented MD algorithm is highly parallelized and highly pipelined internally. The FPGA cluster is freely scalable in tems of size, i.e., a larger MD system requires more nodes, however without compromising simulation speed. The performance in terms of energy stability and simulation speed is analyzed. At present, the focus lies on the fast networking, while only minimal MD functionality has been implemented so far, i.e., Lennard Jones interactions and a thermostat, that were needed to demonstrate the feasibility of the FPGA cluster to run multi-microsecond simulations. To that end, the nucleation of a super-cooled Lennard Jones liquid is investigated by unbiased MD simulations, which is a difficult MD problem since a high nucleation barrier has to be overcome. Finally the pathways towards a full MD implementation are outlined. The current implementation will be made available as an open-source development project.

I. INTRODUCTION

Molecular dynamics (MD) simulations are an important branch of computational chemistry. They simulate the motion of the atoms of molecules typically in the solutions phase, and hence are an important tool to understand the kinetics and thermodynamics of complex molecular systems, in particular proteins. Newton's equations of motion are propagated in a time-discretized manner, in most cases based on an empirical force field that comprises inter-molecular Lennard Jones and Coulomb forces as well as inter-molecular forces related to chemical bond stretches, bends and dihedral angles [1]. MD simulations are a very computer-intensive problem, as the intrinsic motion of chemical bonds is very fast (10 fs), while the overall dynamics can be very slow, i.e., often extending into the second range depending on the molecular system. Very many intrinisic time steps thus have to be computed, and that in a serial manner, as each time step depends on the outcome of the previous one. MD simulations cannot be parallelized beyond what has to be computed for individual time steps. The biggest limitation of MD simulations still is the sampling the huge conformational space of complex molecular system [2]. Arguably, all MD simulations performed so far on a protein, even the smallest ones, are under-sampled. It is thus important to speed up MD simulations. The common way to address that issue is by methods of enhanced sampling, such as umbrella sampling, replica exchange sampling or transition path sampling [3–6]. However, long unbiased MD simulations have their value as well, even though that approach will only improve on the sampling problem, not solve it, as it is not foreseeable at this point that one will ever reach a second time scale in that way.

With that in mind, D. E. Shaw Research LCC, a privately held research company run by David E. Shaw, has developed the extremely successful Anton computer [7–9]; by now the 3rd generation [10]. The Anton computer hard-codes the complete MD algorithm in an "application-specific integrated circuit" (ASIC). It breaks all records of simulation speeds by 2-3 orders of magnitudes, with GPU clusters being the reference, which currently are the state-of-the art of "conventional" computer hardware for MD simulations [11]. That is, simulation speeds up to 100 μs/day can be achieved for mid-sized MD box with around 100000 atoms [10], which is unheard of otherwise. The MDGRAPE series of computers developed at the Riken Institute, Japan, follows a similar philosophy [12–14], but has not made the same impact, presumably since it did not quite reach the performance of the Anton computer.

The MD problem is particularly well suited for hard-coding it, since the fundamental algorithm for each individual time step is relatively simple, and the challenge "merely" is to repeat it very many times. The advantage of ASICs, in comparison to a conventional CPUs, is that it can be massively parallelized inside the chip, without the need of fast data buses outside, which always are a bottleneck. In addition, the algorithm can be completely pipelined by specifically designing the data flow of the algorithm. In simple words, the complete calculation of the force on a given atom takes one clock cycle only (with a certain latency), which is the ultimate limit. In addition, many of those atoms can be treated in parallel.

Having said that, the development of an ASIC is extremely expensive. For example, the accumulated devel-

opment costs of MDGRAPE4 have been in the range of 30 Mio US$ [15] (D. E. Shaw Research LCC doesn't disclose their budget to the best of the authors knowledge). The ASIC needs to be designed and debugged completely with the help of a sophisticated simulation software, and is then burned into silicon by a semi-conductor company. Changes in the hard-coded part of the ASIC then are no longer possible. In essence, the development of a specialized hardware similar to the Anton computer is not affordable for a "normal" research institution. One can apply for computer time on the Anton computer, though.

The development of MDGRAPE started in 2003 [12], and that of the Anton computer in 2008 [7]. Since then, another type of electronics advanced tremendously: field programable gate arrays (FPGAs). FPGAs contain a large matrix of versatile logical units (logical slices), which can be interconnected by freely programable links. In that way, complex function such as an MD algorithm can be "hard-coded" into an FPGA in a similar way as into an ASIC. However, in contrast to an ASIC, it is a standardized chip that is freely programable and also reprogramable. This aspect makes the development and debugging much easier and less expensive. The conceptual advantages of FPGAs over conventional CPUs, massive parallelization and pipelining, is the same as for ASICs.

There has been a small number of attempts to use FPGAs for MD simulations [16–25], in particular by the Herbordt group [26–32] (to the best of the authors knowledge, that list of citations is close to exhaustive). Reviews on the feasibility of FPGAs for MD simulations, and how they compare to implementations based on GPUs and ASICs, can be found in Refs. [15, 33]. The early implementations suffered from the limited resources of FPGAs at that time. However, FPGAs have grown tremendously over the last two decades, in terms of the number of logical slices, memory and arithmetic units available on a single chip and to a lesser extent also in terms of the clock frequency. FPGAs have also been explored for other fields of computational chemistry, i.e., quantum chemistry as well as quantum dynamics [34–37], as well as neural networks [38].

Besides being an implementation by itself, FPGAs are also used as a comparably inexpensive debugging step towards an ASIC. The hardware description language to design either one of it, e.g., Verilog, is the same and the migration of a debugged FPGA implementation to an ASIC design is relatively straightforward.

Another aspect is the energy consumption of computers and their cooling, and given that MD simulations are very computer-intensive, it is a growing concern [11]. Due to the fact that the hardware is very specifically tailored to the particular problem and that in a very efficient manner since the resources on a chip are restricted, thereby avoiding any overhead that is not needed for the concrete calculation, the energy consumption of Anton 3 is at least ten times lower than that of a GPU cluster [10]. The same conceptual advantage is expected for an FPGA implementation.

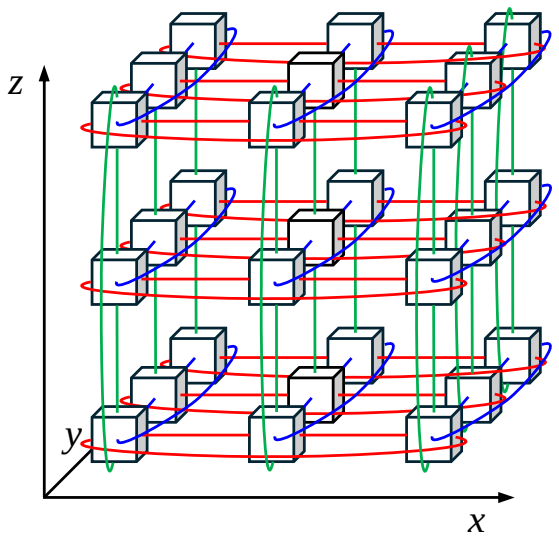

FIG. 1. A $3 \times 3 \times 3$ configuration with 27 nodes connected as a 3D torus. Links in the x-direction are shown in red, those in the y-direction in blue, and those in the z-direction in green. Each link is bidirectional. Not all cyclic links in the z-direction are shown in order not to overcrowd the picture.

Most MD implementations in FPGAs either were incomplete and thought to be a proof-of-concept only (the same applies for the implementation discussed here), or, did only part of the calculation, in particular the non-bonded forces, while the more complex, but less time-critical bonded forces were calculated in a conventional CPU/GPU. In the latter case, the communication between both parts will always be a bottleneck. In that regard, the probably most advanced MD implementation as of now is that of Ref. [31], which integrates all aspects of an MD simulation into a single high-end FPGA (Intel Stratix 10).

However, regardless how advanced the FPGA, its resources will always be hard limited. An alternative approach is therefore explored here, using very many (eventually), but comparably small FPGAs that are linked together in a 3D torus configuration, see Fig. 1, to account for the periodic boundary condition of the overall system. Obviously, the smallest conceivable box size of a node should be the same as one cut-off distance of the Lennard Jones and real-space Coulomb interactions. Typical values used for the cut-off distance used in MD simulations are around 1 nm, containing around 100 atoms given the density of water and/or proteins. This is what the present project is aiming for.

The ASIC implementations Anton and MDGRAPE use the same 3D torus structure shown in Fig. 1. Each node works largely independent and communicates only locally with its neighbors by exchanging atom data. In contrast, when using conventional computer hardware, including GPU clusters, there is the one overarching pro-

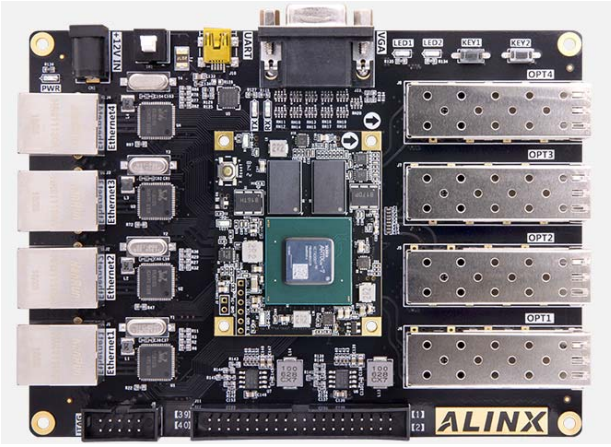

FIG. 2. FPGA development board AX7201 from ALINX, on which the MD computer is based.

gram that overlooks the complete MD box. It launches parallel threads that work on sub-problems, e.g. sub-boxes similar to those shown in Fig. 1, and orchestrated the data transfer between threads. Here, we will not have such an overarching layer. A conventional host computer is used only to load the system with initial data but does not interfere with data transfer between nodes since that eventually would become a performance limiting bottleneck. Much of the data flow discussed below is related to that local structure of each node.

In addition, the following criteria have been used as premise for the design of the FPGA cluster:

- The target for the simulation speed should be $\gtrapprox$ 1 μs/day, in which case it would outperform the current state-of-the-art, GPU clusters.

- FPGA cluster should be freely scalable in terms of size, i.e., a larger MD system will require more nodes, however, without compromising simulation speed.

II. HARDWARE

Each node of the MD cluster is realized with an inexpensive development board AX7201 from ALINX (https://www.en.alinx.com/), see Fig. 2. It features what is considered a mid-ranged FPGA, Artix 7 XC7A200T-2FGG484I from Xilinx/AMD, together with – very important for the present project – four bidirectional optical interfaces running at 4 Gb/s each and four 1 Gb/s ethernet interfaces. The optical interfaces as well as two of the ethernet interfaces are used to connect nodes in a 3D torus, as discussed further below, and a third ethernet interface for communication with the host computer. It also contains an expansion port with 34 freely programmable in/out (IO) lines used to set the node number, to synchronize all nodes and to control a cooling fan for the FPGA. To that end, a small additional printed-circuit board (PCB) has been designed. Finally, 1 GB of external DDR3 SDRAM is available on the development board, which currently is not used but will become relevant in future developments. The other interfaces of the development board (a USB and a VGA port) are not used.

The Artix 7 XC7A200T FPGA contains 33650 logical slices, each containing four look-up-tables (LUTs) with six inputs as well as eight flip-flops. It furthermore contains 730 dual port RAMs with 18Kb each, and 740 signed 25 bit$\times$18 bit-multipliers. For a comparison, the FPGA used in Ref. [31] (Intel Stratix 10) has about 20 times more resources in terms of logical slices, on-chip RAMs and arithmetic units. The RAMs can all be read and written in parallel, allowing for very wide data-buses inside the chip. The computational power of the multipliers add up to about 220G fixed-point operations per second for each node when running at a realistic frequency of 300 Mhz, which is an enormous computation power for a relatively inexpensive chip. Much of the design of the MD algorithm is directly related to the way how the RAMs and multipliers can be configured, for an efficient as possible usage of the resources of the FPGA. The FPGA is programmed in Verilog with the Vivado development tool, whose licence is free of charge for the Artix 7 XC7A200T chip.

One development board costs around 350 US$. For a $3 \times 3 \times 3$ configuration with 27 such development boards, which is the smallest meaningful configuration and which can house ca. 6000 atoms, this adds up to about 10000 US$, which in the same range as conventional hardware for MD simulations, or as the one FPGA board used in Ref. [31].

III. DATA FLOW

To understand the following, one needs to carefully distinguish a “node” from a “box”. Each node corresponds to one evaluation board with one FPGA, as shown Fig. 2. Each node can house only 256 atoms, a number that originates from the RAM size of the FPGA (see below). For larger MD simulations, a certain number of nodes need to be connected and run in parallel. Each node contains a “home box”, which houses the atoms of the node. However, for the calculation of the non-bonded forces, i.e., the Lennard Jones and Coulomb forces, also the positions of the atoms in the neighboring nodes need to be known. For computation time reasons, one cannot afford to request these positions from neighboring nodes at the moment when they are needed for the calculation; despite the reasonably high speed of the interfaces that would be way too slow. This is why there are 26 additional RAMs in a given node, which contain the data of the neighboring nodes; they will be called “neighboring boxes”. Whenever the MD algorithm finished to calculate the new positions and velocities of one atom, its data

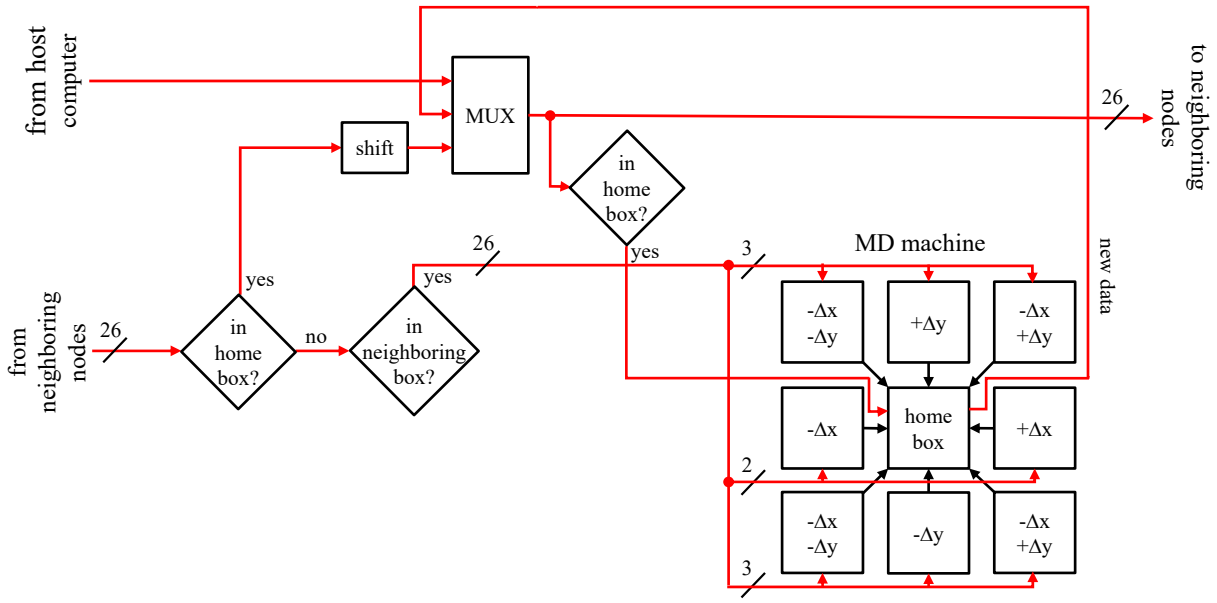

FIG. 3. Overall data flow within each node, see text for details. Of the in total 26 neighboring boxes around the home box, only 8 in the (xy)-plane are shown in order not to overcrowd the picture. Red arrows indicate data flow, black arrows force calculations (as discussed in Fig. 7). MUX stands for a "multiplexer".

are broadcasted to all neighboring nodes, and stored into on of its neighboring boxes, so that the information is available at high speed, i.e., one for each clock cycle, for the force calculation in the next MD time step.

The nodes need to be connected to reveal a 3D torus to account for the periodic boundary conditions, see Fig. 1. The FPGA is programmed in a way that the size of the overall MD system can be chosen freely with only minimal adjustments needed. While a $3 \times 3 \times 3$ configuration is the smallest meaningful configuration, smaller configurations may be useful for debugging purposes. Internally, each home box has 26 surrounding neighboring boxes, regardless what the configuration is. In a $3 \times 3 \times 3$ configuration, the 26 neighboring boxes around the home box of a given node contain the same information as the home boxes of the 26 neighboring nodes. For smaller configurations, some of the neighboring boxes around the home box of a given node will be duplicates of each other. For example, in a $1 \times 1 \times 1$ configuration, the home box and its 26 neighboring boxes will all contain identical information. In larger configurations with the number of nodes larger than 3, a given node will no longer "see" all other nodes. The number of nodes in the various directions does not have to be identical, allowing one to adjust the overall system to the shape of a particular MD problem.

A. Data Flow Within Each Node

Fig. 3 shows the overall data flow within each node. The central MD machine consists of the home box and the 26 neighboring boxes (of which only 8 in the (xy)-plane are shown in Fig. 3 in order not to overcrowd the picture). The atoms in the neighboring boxes may be within the cut-off distance to the home box, and hence need to be considered in the force calculation. Initially, the host computer writes atom data into the home box (it does so for all nodes). The data flow through a multiplexer (MUX) in front of the home box, since other data may enter into the home box as well (see below). The initial data, whose (xyz)-coordinates typically will be within the range of the home box, are also transmitted to the 26 neighboring nodes.

Subsequently, a certain number of MD time steps is executed. Each MD time step produces new atom data, which are written back into the home box, but only after checking whether their (xyz)-coordinates are within the range of the home box, as some atoms might have moved out of that range. In addition, the newly generated data are broadcasted to all neighboring nodes, regardless whether they are still within the range of the home box. All other nodes will do the same, hence, their results appear at the receiving side of Fig. 3. If the (xyz)-coordinates of the received data moved into the range of the home box of the node considered here (and hence moved out of the home box of the corresponding neighboring node), they will enter the home box via the MUX. In that way, an atom moved from the home box of a neighboring node into the home box of the node considered here. On the other hand, if the (xyz)-coordinates of the received data did not move into the range of the home box of the node considered here, but rather are in the range of the corresponding neighboring box, they are loaded into it.

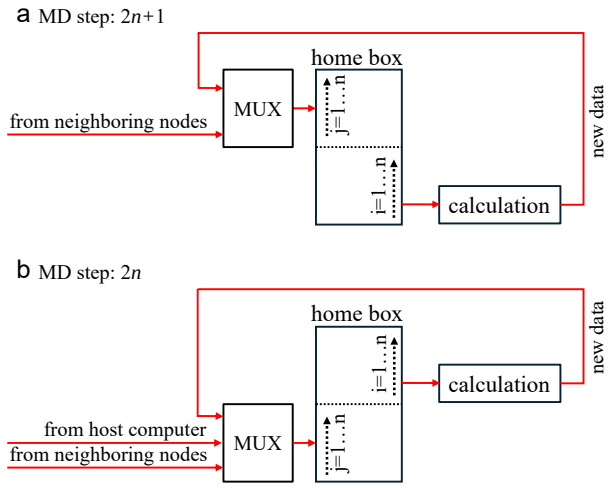

FIG. 4. Organisation of the RAM holding the home box data, and they way they are written and written for an odd MD step (panel a) and an even MD step (panel b). The MUX is the same as in Fig. 3, while the yes-no flowcharts of Fig. 3 are not shown here in order not to overcrowd the picture. The concept is exemplified here for the home box; the neighboring boxes are organized in a similar manner. MUX stands for a "multiplexer".

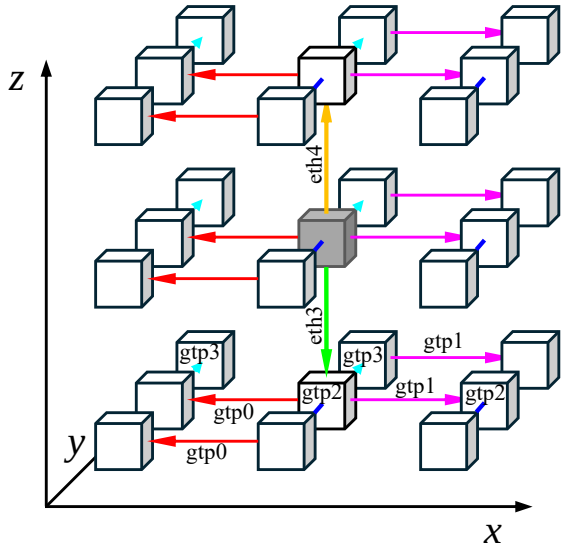

FIG. 5. Organisation of the link between a central node (in grey) and its 26 neighboring nodes via two ethernet interfaces (eth3 and eth4) and four optical interfaces (gtp0-gtp3). See text for details.

B. Organization of the Data RAMs

The RAMs containing the atom data of the home box and all neighboring boxes are organized as 512 lines of 27 bytes each, i.e., 3×3 bytes for the (xyz)-coordinates (see below), 3×3 bytes for the (xyz)-velocities, and 9 bytes of additional meta data such as atom number, mass, charge. The 512 lines are split into two blocks of 256 lines each, which defines the maximum number of 256 atoms each node can house. The RAMs are dual port with simultaneous writing and reading access. During calculation, data are read from one block and the results are written in parallel into the other block, see Fig. 4. From one MD time step to the next one, the role of these two blocks is swapped by simply flipping the highest address bit. Upon initialisation from the host computer, data are written into the lower block (Fig. 4b), and then evaluated in the first MD step (Fig. 4a). The sequence continues from there on. The number of MD steps always should be an even number. Furthermore, the MD algorithm is organized such that new data of a given atom are fully calculated before treating the next atom. In that way, while the calculation of further atoms is still ongoing, the data of the already finished atoms can be broadcasted to the neighboring nodes, thereby causing only minimal time overhead for the communication between nodes, provided that the transmission of the data for each atom is faster than its calculation.

A fatal error signal is generated when the number of atoms per node exceeds an threshold of 252, which aborts the calculation. For interacting particles that repulse each other (i.e. with the parameters used in the examples discussed in Results), it was found that this never happens when the average number of atoms per nodes is $\lessapprox 220$.

C. Linking the Nodes

Each node is linked to its 26 neighboring nodes via two ethernet interfaces (eth3 and eth4) and four optical interfaces (gtp0-gtp3), see Fig. 5. The organisation of these links is similar to Ref. [14], and accounts for the fact that the optical interfaces are faster (4 Gb/s) than the ethernet interfaces (1 Gb/s). That is, the two ethernet interfaces connect only up and down along the z-directions, and the receiving nodes funnel the data through into their (xy)-planes. Communication within the (xy)-planes is performed by the four optical interfaces, which connect up and down in the x- and y-directions, respectively. To also reach the corners of the (xy)-planes, each node in the $\pm x$-direction or the $\pm y$-direction also funnels through the atom data set by one additional step perpendicular to the previous one, e.g. the node in the $(+x)$-direction funnels the atom data through to the $(+x, -y)$-node, etc. For a given atom, each ethernet interface thus needs to transmit only one data set, while each optical interface needs to transmit six data sets (i.e., three (xy)-planes at $-z$, $z = 0$ and $+z$ times two data transfer steps in the (xy)-planes). Given that the overhead of an ethernet frame is larger than that of a frame of the optical interface, the transmission speeds of both types of interface are well balanced. Together with its frame, the transmission of one atom data set, i.e., 27 bytes (see above) to all 26

neighboring nodes takes 480 ns. If more than about 100 atoms are in a node, this is faster than the force calculation, and hence not time limiting for the overall speed performance of the MD computer (see Results).

Data integrity is guaranteed by a 3 byte CRC and a 1 byte sequence number. If a transmission error should occur (which did not happen in any of the calculations shown in Results), the MD step is repeated. To that end, an error signal is transmitted to all other nodes in the same way as the synchronisation signal (see below).

D. Synchronisation

The duplicated RAM structure Fig. 4 ensures that each node has all information it needs to perform the calculations of all atoms in that node during one MD time step. As such, synchronisation of nodes is only needed after the calculation of all atoms, i.e., once per MD time step. To that end, a separate link is used that is realized via the freely programmable IO lines the FPGA development board provides. The link connects a busy signal of all nodes via a logical OR. Only when all nodes have finished their calculation of a particular MD time step, and broadcasted all results, they proceed with the calculation of the next MD time step.

IV. MD ALGORITHM

A. Nearest Neighbor Search

Non-bonded forces are calculated only for atoms that are within a certain cut-off distance, typically around 1 nm. To determine which atoms are within that cut-off distance around a given atom, it is common in MD programs to first calculate a nearest neighbor list. While this requires only the calculation of a distance, and thus is less expensive than the full force calculation, it still requires on the order of n^2 such calculations (when no additional tricks are used such as those discussed below), where n is the number of atoms.

Calculating such a nearest neighbor list is against the very concept of the FPGA algorithm. That is, the concept of the FPGA algorithm is to fully pipeline the force calculation, hence, it takes a single clock cycle with a certain latency for the calculation of the force between pairwise two atoms. The calculation of the nearest neighbor list cannot be faster than that; just the latency would be shorter since the calculation is simpler. Precalculating a nearest neighbor list would not safe any computation time.

Conventional MD programs face the same problem (to a lesser extent), which is why it is common to calculate a nearest neighbor list only every, say, 10th MD time step. This however would require two parts of the FPGA, each of which using significant amount of resources, but which would not be active at the same time. In addition, storing a nearest neighbor list requires a relatively large amount of memory. While the internal RAMs of an FPGA can be used in a highly parallel manner, their total size is not huge; only about 1.6 MB for the Artix 7 XC7A200T. Very generally speaking, the resources within an FPGA are strictly limited and one needs to use it efficiently. It was concluded that the calculation of a traditional nearest neighbor list is too wasteful with regard to resources.

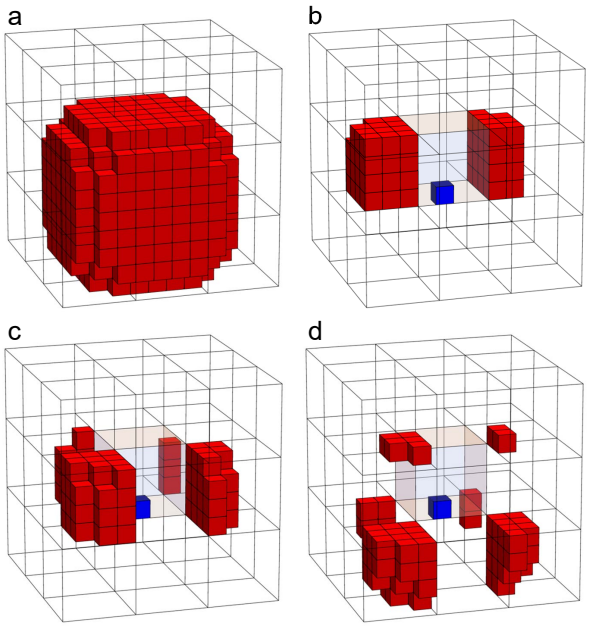

FIG. 6. (a) The sub-boxes within cut-off distance, when using $4 \times 4 \times 4 = 64$ sub-boxes. The embedding 27 boxes are shown as well. (b-d) Parallelizing the calculation of non-bonded forces between one atom in the home box and another atom in one of the neighboring boxes. Sub-boxes in the neighboring boxes considered by the different parallel pipelines are shown in red, and the sub-box in the home box, in which the one atom resides, is shown in blue. The home box (in faint white), as well as the neighboring boxes, are also indicated. Panel (b) exemplifies the sub-boxes in the $\pm x$-direction, and two additional ones exist for the $\pm y$ and $\pm z$-directions. Panel (c) exemplifies the sub-boxes in the (xy)-plane, and two additional ones exist for the (xz) and (yz)-planes. Finally, panel (d) shows the sub-boxes in the (xyz)-cube corners.

The size of the home box is set to one cut-off distance, hence the trivial solution to the nearest neighbor problem would be to just consider all atoms within the home box and its 26 surrounding neighboring boxes. While this would safely consider all atom pairs that are within the cut-off distance, very many are separated by larger distances. For these pairs, one would know without calculation that the force is zero, but it would still require one clock cycle to loop over it. To estimate the waste of clock cycles in that case, we consider the ratio of volume of a sphere with a radius equal to the cut-off distance

versus that of the 27 boxes [31]:

$$f_V = \frac{4\pi/3 r_{cut}^3}{27 r_{cut}^3} = 0.16 \quad (1)$$

84% of the clock cycles would be wasted in that case.

To improve on that, each box is divided in $4 \times 4 \times 4 = 64$ sub-boxes, also a common strategy in conventional MD programs. When atom data from other nodes are received, they are sorted into their corresponding sub-box, and the n^2-algorithm to generate a nearest neighbor list is avoided. When choosing sub-boxes in a way that all atom pairs within the cut-off distance are captured, as shown in Fig. 6a, the ratio f_V increases almost three fold to ≈ 0.44. The ratio would increase even further if more sub-boxes were used (it converges very slowly to 1), but then it would become likely that a certain number of sub-boxes contain zero atoms. This again would result in wasted clock cycles, since one has to check each sub-box whether or not it contains atoms. $4 \times 4 \times 4 = 64$ sub-boxes was deemed to be the best compromise.

It is common in MD programs that each atom pair is considered only once by making use of Newton's third law of motion, calculating the force between them and then having this force acting on both atoms with opposite signs. In terms of nested loops over atoms, the outer loop $i1$ runs from 1 to n (where n is the number of atoms), and the inner loop from 1 to $(i1 - 1)$, hence $n(n-1)/2$ force calculations. In the present case, it would be straight forward to do that for atom pairs that are within the home box. However, for atom pairs, in which one atom is in one of the neighboring nodes, and that situation corresponds to about 88% of the force calculations, that would result in unacceptably large data traffic over the links connecting the nodes. Therefore, in the present case, both inner and outer loops run from 1 to n, increasing the computational effort by a factor 2. Performing redundant calculations to avoid communication is a strategy that has been used before [39, 40]. Internal parallelization, as discussed next, can compensate for that.

B. Internal Parallelization

To speed up the overall performance, non-bonded forces are calculated by eight parallel pipelines, see Figs. 6 and 7. The data of the outer-loop atom ($i1$ in Fig. 7) need to be read only once and then are made available to all pipelines, but a data set for the inner-loop atoms ($i2$ in Fig. 7) need to be read for every clock cycle. Hence, it is important to organize the parallel pipelines in a way that they work on separate boxes, i.e., read separate RAMs. Here it is crucial that the various RAMs of the FPGA can be all read in parallel.

In more detail, one pipeline works on atom pairs with both atoms in the home box, i.e., for a given first atom it needs to consider 64 sub-boxes for the second atom. Three other pipelines work on the sub-boxes in the two boxes in the $\pm x$, $\pm y$ and $\pm z$-directions, respectively (Fig. 6b exemplifies that for the $\pm x$-direction). Depending on the sub-box in the home box (shown in blue in Fig. 6b for one example), these are either 80 or 78 sub-boxes, in average 79.5 sub-boxes. Three further pipelines work on the corners in the (xy), (xz) and (yz)-planes (exemplified for the (xy)-plane in Fig. 6c) with 81 sub-boxes in average. Finally, the 8th pipeline works on the (xyz)-cube corners (Fig. 6d) with 67.5 sub-boxes in average). The rather complex structure of the sub-boxes that need to be considered (see Fig. 6b-d) are programmed into a LUT. The computational effort of in particular the pipelines of type Fig. 6b and those of type Fig. 6c, which do the biggest share of the calculations since it is three of them each, is extremely well balanced.

The data of outer-loop atom are written into an first-in-first-out (FIFO) buffer for each one of the pipelines working on neighboring boxes, see Fig. 7. Since the first pipeline working on the home box is faster than all other pipelines due to the smaller number of sub-boxes to be considered, the other pipelines receive atom data from the FIFO buffer without interruption and thus can calculate the forces between the home-box atom and the various atoms in the neighboring boxes essentially without any waiting cycles. Once a particular pipeline finished the force calculation of one home-box atom, the result is stored into an output FIFO buffer. Once all pipelines finished the calculation of that home-box atom, the forces from all pipelines are added up, new velocities are calculated and eventually the new atom positions (see Eqs. 4 and 5 below). The FIFO buffers at the inputs and outputs of the various pipelines level out their computation times, which vary to a certain extent due to the fluctuating number of sub-boxes, as well as the fluctuating number of atoms per sub-box.

The parallel pipelines do the biggest share of the calculation, i.e., the part of the calculation of the non-bonded forces that scales as n^2. They run at 280 MHz, which turned out to be the highest frequency at which the design software (Vivado) is able to route the logic in the FPGA. All the rest, in particular the data flow shown in Fig. 3, which is less time-critical, runs at 100 MHz, in order to relax the demands on routing. The FIFO buffers mentioned above are also used for the clock-domain transfer.

C. Force Calculation, Time Propagation and Units

Newtons equation of motion are time-propagated with a Leapfrog algorithm:

$$\boldsymbol{v}_i \leftarrow \boldsymbol{v}_i + \frac{\boldsymbol{F}_i}{m_i}\Delta t \quad (2)$$

$$\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i + \boldsymbol{v}_i \Delta t \quad (3)$$

where $\boldsymbol{x}_i$ is the position of atom i, $\boldsymbol{v}_i$ its velocity, m_i its mass, $\boldsymbol{F}_i$ the force acting on it and Δt the time step.

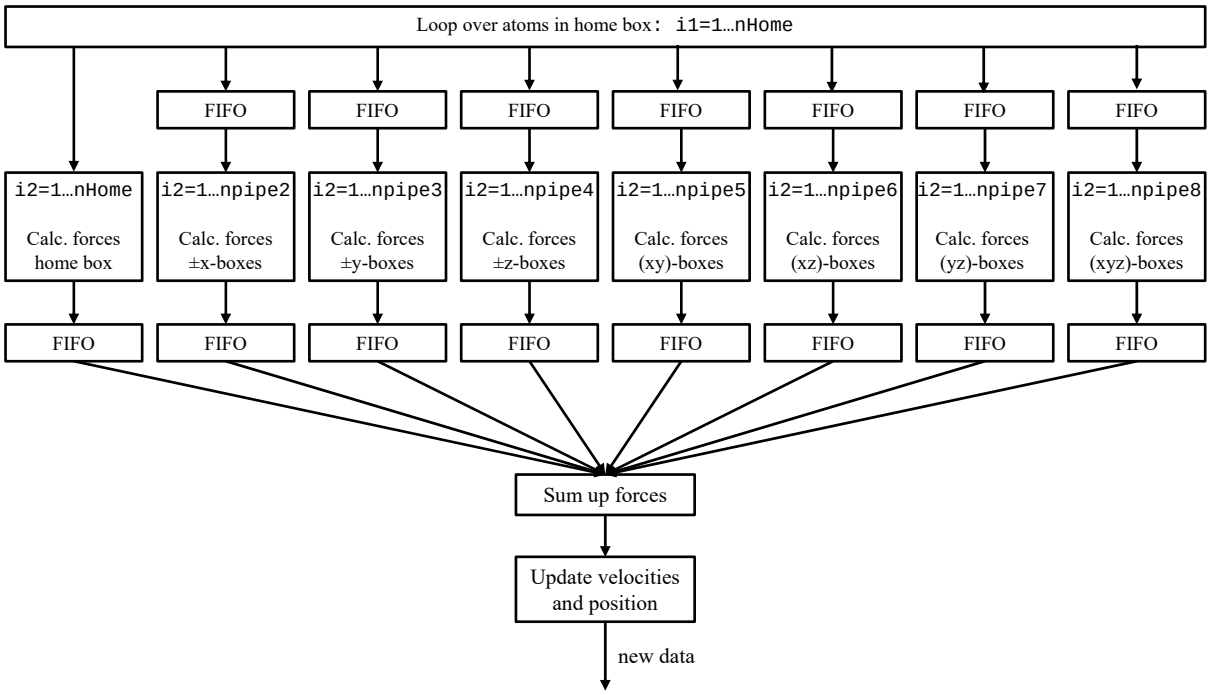

FIG. 7. Organisation of the MD machine with 8 parallel pipelines doing the calculations of the non-bonded forces. FIFO stands for a "first-in-first-out" buffer.

The left arrow indicates how $\boldsymbol{v}_i$ and subsequently $\boldsymbol{x}_i$ are updated in each MD time step. In order to minimize the computations that need to be done within the FPGA to the absolute minimum, we wish to write Eqs. 2 and 3 as:

$$\boldsymbol{v}_i \leftarrow \boldsymbol{v}_i + \boldsymbol{F}_i \tag{4}$$

$$\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i + \boldsymbol{v}_i \tag{5}$$

This hides both the atom mass and the time step in the forces, and forces as well as velocities have the same unit as positions. Furthermore, we define positions in units of the cut-off distance r_{cut}, as this will be the size of the home box. Hence, all properties in Eqs. 4 and 5 are unit-less. The forces are calculated with the help of a LUT (see below), hence the units can be pre-programmed into the LUT by the host computer.

The Lennard-Jones potential is:

$$V = 4\epsilon_{LJ}\left(\frac{\sigma^{12}}{r_{ij}^{12}} - \frac{\sigma^{6}}{r_{ij}^{6}}\right) \tag{6}$$

where ϵ_{LJ} is the Lennard Jones binding energy, σ the Lennard Jones distance, and r_{ij} the distance between atoms i and j. The force acting on atom i is:

$$\boldsymbol{F}_i = -\frac{\partial V}{\partial \boldsymbol{x}_i} = 24\epsilon_{LJ}\left(2\frac{\sigma^{12}}{r_{ij}^{14}} - \frac{\sigma^{6}}{r_{ij}^{8}}\right)(\boldsymbol{x}_i - \boldsymbol{x}_j) \tag{7}$$

The Lennard-Jones binding energy ϵ_{LJ} is typically tabularized in units of kJ/mol. Multiplying that with:

$$\epsilon_{LJ} \rightarrow \epsilon_{LJ} \cdot \frac{10^{-6} dt^2}{m_{\mathrm{O}} r_{cut}^2} \tag{8}$$

reveals the desired unit-less Lennard-Jones binding energy. Here, dt is in units of fs, r_{cut} in units of nm, m_{O} in atomic mass units, and the factor 10^{-6} is the accumulated power-of-ten that translated these units into SI units. Furthermore, when the Lennard-Jones distance r_{LJ} is written in units of r_{cut}, r_{ij} and $\boldsymbol{x}_i$ are as well.

For a first proof-of-principle demonstration of the MD computer, we will consider a Lennard-Jones liquid of identical particles, in which case the masses m_i in Eq. 2 are all the same. We will use the mass of an oxygen atom $m_{\mathrm{O}} = 16$. In the long run, we will use it as a reference mass, and multiply the forces in Eq. 4 with a unit-less number to account for atoms with different masses.

D. Fixed Point Representation of Coordinates, Velocities, and Forces

Adders are relatively easy to realize in an FPGA based on its universal logic slices, and the Artix 7 XC7A200T provides dedicated units for signed 25 bit×18 bit multiplications. However, in either case, they work with binary (integer) numbers, while realizing floating point calculation units would need significant amount of resources in the FPGA currently used (more advanced FPGAs have dedicated floating point units). All calculations are therefore performed using fixed-point numbers (which has been another reason to aim for a unit-less representation of of all properties). The choice of the bit size of the fixed-point numbers was guided by the bit

size of the building blocks the FPGA provides. In particular, the RAMs of the FPGA can be organized as 512 words, each 72 bits wide. The width of 72 bits suggest to store the (x, y, z)-component of coordinates velocities in one RAMs each with 24 bit precision for the three component. The available multipliers also work well with 24 bit numbers; a 24 bit$\times$24 bit multiplication can be realized either with two cascaded multipliers, or with one multiplier together with a relatively modest amount of additional slice logic. It will be shown in Results that a 24 bit representation is sufficient to reveal excellent energy stability. In that regard it should be noted that the significand of a single-precision floating-point number also has 24 bits, and that common MD programs typically do use single precision floating-point numbers.

The impact of fixed-point arithmetic for MD simulations has thoroughly been studied [41]. The MD problem is in fact particularly well suited for a fixed-point representation, since we know the range of positions and velocities quite well. That is, the position of all atoms is obviously bound to the MD box. Velocities are known to be Gaussian distributed:

$$\rho_k \propto e^{-\frac{m_i v_k^2}{2k_B T}} \tag{9}$$

with $k \in \{x, y, z\}$. The width of the Gaussian distribution scales with $\sqrt{1/m_i}$ and $\sqrt{T}$, respectively, i.e. varies relatively little, and the probability goes to zero very quickly beyond that width. Should an overflow really occur (extremely unlikely for the chosen implementation), the velocity is set to the largest positive or negative number that is possible in the fixed-point representation.

For the positions, we choose a fixed-point representation from -2...+2 in 24 bit, to be able to represent differences of positions between two atoms, one of which is in the home box and the other in one of the neighboring boxes. For velocities, we choose a fixed-point representation from $-1/2^s$ to $+1/2^s$ in 24 bit, where s=4 is chosen so that the probability of an overflow is negligible, considering Eq. 9. When evaluating the sum in Eq. 5, the lower $s-1$ bits are disregarded, and the subsequent bit is used for round off. Since the velocities accumulate forces over many MD time steps (Eq. 4), it is important to represent them at higher bit resolution than actually needed for Eq. 5, to minimize round-off errors during accumulation. Finally, forces are also represented from $-1/2^s$ to $+1/2^s$, albeit in 32 bit. Forces are not stored in any RAM, hence one is not bound to the RAM architecture. Forces also accumulate the contributions of many atoms, hence it is also important represent them at higher precision than what is needed for the sum in Eq. 4.

E. Force LUT

When calculating the force between two atoms, the first value to be computed is r_{ij}^2. However, the distance r_{ij} is not calculated, since the square-root would cost a lot of FPGA resources. Rather, the first terms of Eq. 7 are expressed as functions of r_{ij}^2 instead of r_{ij}. These functions are approximated by quadratic interpolation with coefficients tabularized in a LUT, as commonly done in FPGA implementations [16, 18, 20, 26, 27, 31, 32], while the multiplication with $(\boldsymbol{x}_i - \boldsymbol{x}_j)$ is performed explicitly within the FPGA. Using LUTs for the force provide flexibility with regard to the interaction force, e.g. replace the Lennard Jones by a Buckingham potential, as the LUT can be programmed at run time without have to re-synthesize the FPGA design. The LUT are organized considering to the architecture of the RAMs, in which they are stored. That is, 512 equidistance grid points are defined for $0 <= r_{ij}^2 < 1$, each storing three 24 bit numbers. The first number defines the force at the grid point, and the second and third number a linear and a quadratic term, respectively, defined such that it interpolates to the force at next grid point, as well as to the force midway between both grid points.

The force calculation Eq. 7 in each of the pipelines shown in Fig. 7 is fully pipelined and takes one clock-cycle for each pair of atoms with a latency of 19 clock cycles.

F. Thermostat

A simple thermostat is implemented using velocity rescaling, taking into account the constraints of an FPGA for performing certain arithmetic operations and the architecture of the interconnected nodes. First, the temperature $T = m/(3Nk_B)\sum_i v_i^2$ is calculated in the unit-less representation discussed above:

$$T = \sum_i \boldsymbol{v}_i^2 \tag{10}$$

where the sum runs over all atoms of the home box and its 26 neighboring boxes. The contribution of the home box is calculated after each MD step, and then broadcasted to all neighboring nodes, hence, the temperature T is available with one time step delay. Since the thermostat has a time constant that is multiple times the time step, that delay is not a problem. The temperature is referenced to a target temperature:

$$r = \frac{T}{T_{target}} \tag{11}$$

Since divisions are slow and resource-intensive in an FPGA, the inverse of T_{target} is calculated in the host computer in the same unit-less representation and then multiplied with T using one of the dedicated multipliers of the FPGA. When defining a velocity rescaling factor

$$s = r^{-\frac{1}{2n_\tau}}, \tag{12}$$

a thermostat with time constant $\tau = n_\tau \cdot dt$ is obtained, where dt ist the MD time step. That expression is calculated with the following trick: The temperature ratio r

will be a number $r = 1 + r'$ with $r' << 1$, since the temperature never will deviate much from the target temperature. With that, we can power-expand the scaling factor as:

$$s = 1 - \frac{r'}{2n_\tau} \tag{13}$$

Only powers of 2 are considered for n_τ, in which case the division in Eq. 13 can be performed by an inexpensive shift operation. Finally, Eq. 4 is modified by introducing the velocity rescaling factor s for each MD time step:

$$\boldsymbol{v}_i \leftarrow s \cdot \boldsymbol{v}_i - v_{cm}/n'_{tau} + \boldsymbol{F}_i \tag{14}$$

In addition, the center of mass motion v_{cm}, which accumulates due to round-off errors (see Fig. 8b), is subtracted from the velocities for each time step with a time-constant $n'_{tau} \approx 2dt$ [42]. The center of mass motion is averaged over all atoms of the home box and the 26 neighboring boxes in the same way as the temperature.

For large enough n_τ (in the examples discussed in Results, $n_\tau = 512$ will be used) and small r', the power expansion Eq. 13 reveals the same scaling factor as the Berendsen thermostat in leading order [43], and the known deviations from a correct canonical ensemble are considered to be acceptable for the time being [44]. A stochastic term for the velocity rescaling thermostat to ensure a correct kinetic energy distribution [45] will be possible without significant impact on the simulation speed and will be added in a future version.

V. RESULTS

A. Performance

As a first proof-of-concept, a Lennard-Jones liquid of identical atoms was considered. Historically, the first MD simulation ever has been performed on that model system by A. Rahman in 1964 [46]. The Lennard Jones parameters of the oxygen atom in SPC/E water were chosen, i.e., σ=0.3166 nm and ϵ_{LJ}=0.65 kJ/mol [47]. For a first test, the time step was set to $dt = 2$ fs, and the size of each node was set to $2 \times 2 \times 2$ nm^3, which at the same time defines the (pretty long) cut-off for the Lennard Jones interaction. A system of $3 \times 3 \times 3$ nodes was used. Simulations were running in the NVE ensemble, i.e., without thermostat, and the initial velocities were set such that the temperature reached 295 K after an initial equilibration. At this temperature and with 5832 atoms in the 27 nodes, the system is a supercritical fluid [48–50], guaranteeing fast equilibration.

Fig. 8a shows the drift of the temperature, $T = m/(3Nk_B)\sum_i v_i^2$, as a function of simulation time. Note that the total energy of the simulation boxes is not calculated in the FPGA, since it is not needed for the time propagation; only forces are calculated according to Eq. 7. Given the equipartition theorem and the fact

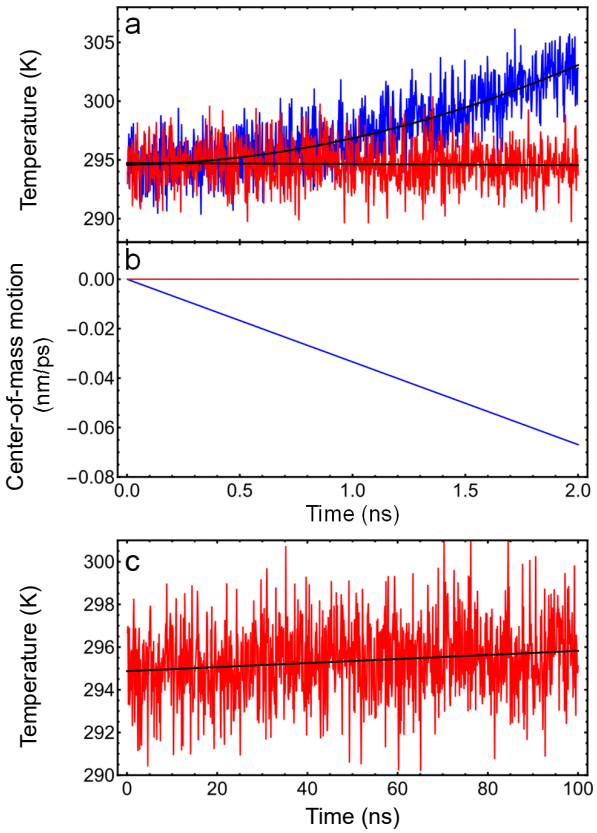

FIG. 8. (a) Drift of temperature of a Lennard Jones liquid with 5823 particles in the NVE ensemble, and (b) center-of-mass motion, exemplified here for the x-component (the y- and z-components show the same trend). In blue are shown the results when center-of-mass motion accumulates due to round-off errors, in red when the center-of-mass motion is removed, but still without thermostat, i.e., with the scaling factor in Eq. 14 is set to s=1. Panel (c) shows the same as the red data in panel (a), however, extended the time-range to 100 ns. The black lines in panels (a) and (c) show linear (red data) or quadratic (blue data) fits of the temperature data.

that the super-critical liquid equilibrates very quickly, one may however use the temperature drift as an indicator for the energy drift. Fig. 8a shows two results: The blue line shows the temperature when not removing the center-of-mass motion, the latter of which accumulates in a linear fashion, see Fig. 8b, and consequently contributes a quadratic term to the temperature. The red line, on the other hand, shows the temperature when removing the center-of-mass motion, but still without thermostat, i.e., when setting the scaling factor in Eq. 14 to s=1. Over a 2 ns simulation time, no temperature drift is observable beyond noise. The time range therefore has been extended to 100 ns in Fig. 8c. A linear fit of these data reveals a very small temperature drift of only about 0.01 K/ns, indicating an excellent energy stability. The bit-accuracy defined by the fixed-point representation of positions, velocities and forces is thus sufficient and in-

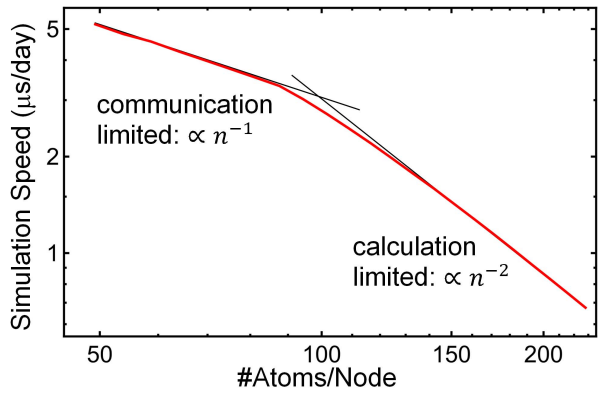

FIG. 9. Calculation speed of $3 \times 3 \times 3$ interconnected nodes in dependence of the number of atoms per node, using a time step of 2 fs.

evitable round-off errors compensate each other to a very large extent.

Fig. 9 shows the performance in terms of simulation speed of a system with $3 \times 3 \times 3$ interconnected nodes in dependence of the number of atoms per node. Two phases are observed: below ≈100 atoms/node, the simulation speed scales as $1/N$ and above as $1/N^2$. In the first phase, the overall simulation speed is limited by the communication between nodes, whereas above it limited by the force calculation. About 100 atoms/node is a sweet spot, where the time required for communication and calculation are balanced and were a simulation speeds of about 2.5 μs/day can be achieved with 2 fs time step. Liquid water or proteins have a density of about 100 atoms/nm^3 Consequently, with a cut-off distance of 1 nm, which is a very typical value used in MD simulations, one would work at that sweet spot.

B. Test Case: Nucleation of a Super-Cooled Lennard Jones Liquid

Nucleation of super-cooled liquids is a rare event, since a high nucleation barrier has to be overcome. Due to its model character, nucleation of a super-cooled Lennard Jones liquid has been extensively studied [51–57] – till now. The time it takes to pass the nucleation barrier is a steep function of temperature below the melting temperature. Most studies use some method of enhanced sampling [3, 4, 6], while unbiased MD simulations are rare. Since one is tempted to speed up barrier crossing by lowering the temperature in unbiased MD simulations, one runs the risk that the system has no time to equilibrate as a metastable super-cooled liquid before crossing the barrier. Since the motivation of the FPGA based MD computer discussed here is to allow for long simulation times, nucleation of a Lennard-Jones was investigated by unbiased MD simulations at temperatures high enough that equilibration should occur.

Nucleation of super-cooled liquids is typically described by "classical nucleation theory" [58, 59], which defines a free energy profile as a function of the number n of solid-like atoms in a nucleus:

$$G(n) = s \cdot n^{2/3} - \Delta\mu \cdot n \tag{15}$$

The coefficient of the first term, s, describes the surface free energy and that of the second term, $\Delta\mu$, the difference in chemical potential between the bulk liquid and bulk solid. The first term is always positive, the second always negative. The free energy profile therefore initially rises with n, but at some point the second term supersedes the first one, since n grows faster than $n^{2/3}$, revealing the nucleation barrier. The temperature dependence of the barrier is introduced by the temperature dependence of $\Delta\mu$:

$$\Delta\mu(T) = \Delta H \frac{T_m - T}{T_m} \tag{16}$$

where ΔH is the heat of formation of the crystal and T_m the melting temperature.

The same Lennard Jones parameters as above have been used for the 5832 atoms, but since the temperature was significantly lower, the time step could be increased to dt =5 fs. The reduced density was set to $\rho^* = \rho \cdot \sigma^3 = 0.95$ by reducing the box-size of the nodes to 1.933 nm, as this density has been used previously for related studies [53, 55]. The system was equilibrated at a reduced temperature $T^* = k_B T/\epsilon_{LJ} = 1.2$ (i.e., 94 K), just above the coexistence region at this density with a melting temperature of the liquid state $T_m^* = 1.15$. The system was then quenched to a lower temperature within 5 ps. The lowest temperature considered here, $T^* = 0.58$, was deemed high enough for classical nucleation theory to be valid [53–55]. The simulations were run in the NVT ensemble with a time-constant of the thermostat of 2.5 ps.

In order to detect solid-like clusters, a Q_6-related order parameter has been used, as introduced in Refs. [52, 53]. In brief, a 2*6+1=13-component complex vector was calculated for each atom (it will be called Q_6-vector) by summing over the spherical harmonics of order $l = 6$ as a function of the unit vectors connecting that atom with its nearest neighbors within a cut-off distance of $1.4 \cdot \sigma$. The Q_6-vector is a high-resolution representation of the local structure around the atom. Next, the scalar product of that Q_6-vector (after normalization) with the corresponding Q_6-vectors of all its neighboring atoms was calculated. If its real part was > 0.5 for at least 11 of the nearest neighbors, i.e., they were approximately "parallel" in that 13-dimensional conformation space indicating local translational symmetry, the atom was considered to be solid-like. Finally, two solid-like atoms were considered to be in the same solid-like cluster, when they were connected by a sequence of solid-like atoms, each within one cut-off distance. As a side remark, the local density in solid-like clusters is different from the liquid. This will

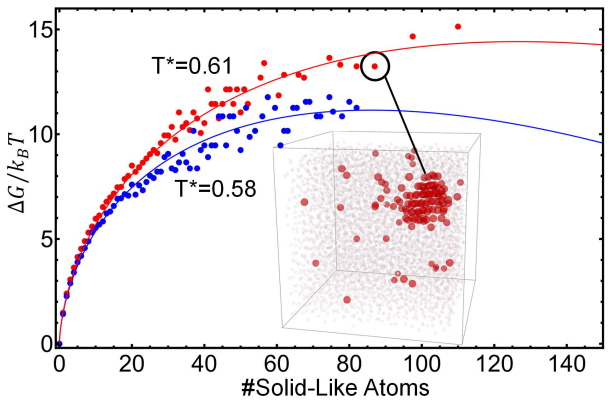

FIG. 10. Free energy curves as a function of the number of solid-like atoms n in a cluster, for two temperatures $T^* = 0.58$ (blue) and $T^* = 0.61$ (red). The solid lines are fits according to classical nucleation theory Eqs. 15 and 16. The inset shows a critical nucleus close to the top of the barrier for $T^* = 0.61$ (see encircled data point), plotting all solid-like atoms in red and all liquid-like atoms in faint white. There are two data points with even larger cluster size, but those where not "productive" and didn't lead to nucleation.

cause a slight timing misbalance of the various nodes, with the overall simulation speed given by that of the slowest node, i.e., the node with the largest number of atoms.

It is important to stress that the FPGA cluster performed only the MD simulation, while the analysis described above was performed in the host computer (that will be a general strategy also in the future). To that end, the FPGA cluster sent data to the host computer with a saving time of 100 ps, while the host computer performed the cluster analysis during those 100 ps MD runs.

Fig. 10 shows free energy curves as a function of the number of solid-like atoms n in a cluster for two temperatures $T^* = 0.58$ (blue) and $T^* = 0.61$ (red). To compute those free energy curves, a histogram $p(n)$ of how often a cluster with size n appeared during the MD simulation was accumulated, and the free energy was calculated as:

$$\Delta G(n) = -k_B T \ln\left(\frac{p(n)}{p(1)}\right) \tag{17}$$

In the first case ($T^* = 0.58$), 10 trajectories have been accumulated with an averaged nucleation time of 11 ns, in the second case ($T^* = 0.61$), 5 trajectories with an averaged nucleation time of 360 ns. When a simulation passes the nucleation barrier, it is in fact no longer in quasi-equilibrium and proceeds very quickly towards a fully crystalized MD box. The simulation was therefore stopped at this point, and the free energy curves are valid only up to the barrier top.

The solid lines in Fig. 10 represent global fits according to classical nucleation theory Eqs. 15 and 16, with the same values for s and ΔH for both temperatures. The best fit was obtained for a meting temperature $T_m^* = 0.81$, which is within the coexistence region, but closer to the melting point $T_m^* \approx 0.7$ of the solid [50]. The latter would be the relevant melting temperature in Eq. 16. A possible reason for this discrepancy could be the fact that the shape of the critical nucleus deviates from spherical, as classical nucleation theory assumes, in which case the surface term becomes larger and hence the barrier higher. Furthermore, it has been shown in Refs. [53, 54] that the deviation from a spherical shape is more pronounced at lower temperature, moving the barrier height of the $T^* = 0.58$ simulation up by a larger amount than that of the $T^* = 0.61$ simulation. An apparent melting temperature larger than the real one can mimic that result.

Nonetheless, the very good agreement of the fit with the sampled data verifies that classical nucleation theory is essentially correct for super-cooled Lennard-Jones liquids, as previously concluded [52]. The fits reveal a critical nucleus of size of $n = 86$ for $T^* = 0.58$ and of $n = 135$ for $T^* = 0.61$, in good agreement with previous studies [53, 55]. The free energy difference of the barrier top at the two temperatures is 3.3 kBT, explaining the factor ≈30 difference for the average nucleation time. Finally, the (close to) critical nucleus shown in the inset of Fig. 10 indeed has a roughly spherical shape at $T^* = 0.61$. A spherical shape is considered a criterion for the temperature being high enough that a quasi-equilibrium can be established before nucleation [53, 55].

The total simulation time that went into Fig. 10 is 1.9 μs. While this is of course doable with conventional computer hardware, it starts to become tedious, while it took only about one day of simulation time on the FPGA-based MD computer. To the best of the authors knowledge, it is the unbiased MD simulation at the highest temperature as of now calculating nucleation of a super-cooled Lennard Jones liquid.

VI. CONCLUSION

First steps towards an MD implementation in an FPGA clusters have been presented. Up to this point, the focus lies on the networking, while only minimal MD functionality has been implemented , i.e., Lennard Jones interactions and a thermostat, which were needed to demonstrate the feasibility of the FPGA cluster for a non-trivial MD problem, the nucleation of a super-cooled Lennard Jones liquid. The good agreement with literature results on this topic indicates that the MD implementation is correct. The results also show that multi-microsecond simulation times become easily accessible with the FPGA cluster, hence one important design goal listed in the Introduction has been achieved.

Clearly, there is still a long way to go. For a full MD implementation, at the minimum the following functionalities will have to be added (it is planned to add them in the order of the list below):

- The implementation of Coulomb interactions with

cut-off, such as the reaction field model [60], will be a straight forward addition to the already existent Lennard Jones calculation and can run in parallel. In addition, the information on mass, charge and Lennard Jones parameters will be added to each atom, so that mixtures of non-identical atoms can be considered, such as molten salts.

- The calculation for the virial needed to calculate the pressure of the simulation box will also be straight forward. An additional barostat will require to change the box size. Since all calculations are done in reduced units, with the box size of a single node being the reference, this will result in rescaling the distances in the force calculation.

- Calculation of bonded interactions will be tedious, since it requires more complex mathematical expression. It is anticipated to do this in pipelines parallel to the non-bonded forces, and since it scales only linear with the number of atoms, it is expected that it will not slow down the overall performance. Atoms are currently sorted into the internal RAMs in the order of appearance in a box, and not by atom number, hence some additional book-keeping will be needed. Most likely that will require the DDR3 SDRAM of the evaluation board, which is not used as of now.

- Shake [61] or an equivalent algorithm to constrain X-H bonds, in particular for water, will be part of the non-bonded interactions.

- A particle-mesh Ewald calculation for the long range Coulomb interactions will be the hardest part, as it contradicts the local character of the current design, i.e., the fact each node sees only its 26 neighboring nodes. It will require faster optical links to communicate information further in the network. The problem of the 3D FFT with FPGAs has been addressed before [25, 30]. Alternatively, one might think of a multipole expansion of long range Coulomb interactions [62], a concept that is not widely used in MD simulations, but might be advantageous for the present architecture.

The current implementation uses about 25% of the resources of the particular FPGA (Artix 7 XC7A200T), i.e., 27% of its LUTs, 19% of its FFs, 24% of its RAMs, and 18% of its multipliers. While this still leaves significant room, it is hard to estimate how many resources the features listed above will need in addition to that. Therefore, the plan is to keep on going with the present evaluation boards as long as possible, before deciding on the hardware that eventually will be needed in the long run. Most likely, a somewhat larger FPGA will be needed, which also should have at least 8 high-speed interfaces: 6 instead of 4 for the links between nodes, and 2 for the communication with the host computer (the current 1Gb/s ethernet is limiting). Furthermore, in order to reduce the required amount of fiber cables, the design of a dedicated PCB with probably $2 \times 2 \times 2 = 8$ FPGAs is anticipated, linked among each other directly.

The electrical power consumption is low, only about 9 W/node, or 250 W in total for 5832 atoms. Only about 3 W/node go into the FPGA, while the remainder is consumed by the other components on the evaluation board. It is too early to judge how this compares to GPU clusters, but the numbers are encouraging.

The obviously fiercest competitor to an FPGA implementation for MD simulations are GPUs. In terms of resources, for example the number of mathematical operations per time unit a single chip can perform, GPUs clearly outperform FPGAs by orders-of-magnitudes, and floating point units are the absolute standard. In addition, they are general-purpose computers with sophisticated compiles available, making the software development much easier. Finally, they are produced for the gaming industry with a huge market, and hence relatively inexpensive. Yet, it has been demonstrated that FPGAs have the potential to keep up with GPUs in terms of simulation speed, and potentially even outperform them by a modest factor. That is, simulation speeds up to about 500 ns/day have been achieved with GPU's in certain benchmark tests (for 2 fs time steps) [15, 63, 64], while simulation speeds $\gtrsim 1$ μs seem realistic with FPGAs, see Refs. [31, 32] and this work. GPUs do not offer the hardware flexibility to custom-define dataflow pathways in a very direct manner, equivalent to the one shown in Fig. 3. Furthermore, GPUs face execution delays due to control flow overheads, such as synchronization and instruction fetching. Conversely, FPGAs can be programmed in a way as to adopt an extremely optimized architecture that enables user-defined, long pipelines, thereby significantly reducing control flow latency, and also allow for a very direct access to high-bandwidth I/O interfaces.

The comparably low clock frequency of FPGAs will always be limiting. In the present implementation, a clock frequency of 280 MHz was the highest, for which the design software could achieve time closure, in comparison to 2.8 GHz of Anton 3 [10]. While slightly higher clock frequencies might be possible with more advanced FPGAs, the clock frequency of an ASIC will probably not become available in the foreseeable future. In essence, that is the prize one has to pay for the reconfigurable connections between logical slices. One will not be able to compensate for a factor 10 higher clock frequency by any other measure. Hence, the goal of an FPGA-based MD computer cannot be the fastest ever simulation speed, rather it can be a still very fast simulation speed for a fraction of the prize of an ASIC solution. In essence, the goal should be a simulation speed equal or faster than a GPU cluster but with lower power consumption, and for the same prize as a GPU cluster. Finally, in contrast to an ASIC implementation, the MD algorithm in an FPGA can be changed and adapted to specific needs.

None of the FPGA implementations presented so far

in literature went into a production phase [16–32], indicating that the concepts have not yet converged. Despite the fact that the implementation described in this paper cannot (yet) compete with that of Ref. [31], it is justified to explore alternative strategies, to eventually find a configuration that does work. Also as a new approach, the current implementation will be made available as an open-source development project.

Acknowledgement: I wish to thank Markus Meuwly for insightful discussions on the project and Yoshitaka Tanimuara for his hospitlity during my sabbatical stay at Kyoto University. The research has been supported financially by University of Zurich

[1] D. Frenkel and B. Smit, *Understanding Molecular Dynamics* (Academic Press, San Diego, 2002).

[2] D. M. Zuckerman, Equilibrium sampling in biomolecular simulations, Annu. Rev. Biophys. **40**, 41 (2011), arXiv:1009.2958.

[3] G. M. Torrie and J. P. Valleau, Nonphysical sampling distributions in Monte Carlo free-energy estimation: Umbrella sampling, J. Comput. Phys. **23**, 187 (1977).

[4] P. G. Bolhuis, D. Chandler, C. Dellago, and P. L. Geissler, Transition path sampling: Throwing ropes over rough mountain passes, in the dark, Annu. Rev. Phys. Chem. **53**, 291 (2002).

[5] P. Liu, B. Kim, R. A. Friesner, and B. J. Berne, Replica exchange with solute tempering: A method for sampling biological systems in explicit water, Proc. Natl. Acad. Sci. U. S. A. **102**, 13749 (2005).

[6] A. Laio and M. Parrinello, Escaping free-energy minima, Proc. Natl. Acad. Sci. U. S. A. **99**, 12562 (2002).

[7] J. S. Kuskin, C. Young, J. P. Grossman, B. Batson, M. M. Deneroff, R. O. Dror, and D. E. Shaw, Incorporating flexibility in Anton, a specialized machine for molecular dynamics simulation, 2008 IEEE 14th Int. Symp. High Perform. Comput. Archit. Salt Lake City, UT, USA, , 343 (2008).

[8] D. E. Shaw, R. O. Dror, J. K. Salmon, J. P. Grossman, K. M. MacKenzie, J. A. Bank, C. Young, M. M. Deneroff, B. Batson, K. J. Bowers, E. Chow, M. P. Eastwood, D. J. Ierardi, J. L. Klepeis, J. S. Kuskin, R. H. Larson, K. Lindorff-Larsen, P. Maragakis, M. A. Moraes, S. Piana, Y. Shan, and B. Towles, Millisecond-scale molecular dynamics simulations on Anton, Proc. Conf. High Perform. Comput. Networking, Storage Anal. SC '09 , 1 (2009).

[9] D. E. Shaw, J. P. Grossman, J. A. Bank, B. Batson, J. A. Butts, J. C. Chao, M. M. Deneroff, R. O. Dror, A. Even, C. H. Fenton, A. Forte, J. Gagliardo, G. Gill, B. Greskamp, C. R. Ho, D. J. Ierardi, L. Iserovich, J. S. Kuskin, R. H. Larson, T. Layman, L. S. Lee, A. K. Lerer, C. Li, D. Killebrew, K. M. Mackenzie, S. Y. H. Mok, M. A. Moraes, R. Mueller, L. J. Nociolo, J. L. Peticolas, T. Quan, D. Ramot, J. K. Salmon, D. P. Scarpazza, U. Ben Schafer, N. Siddique, C. W. Snyder, J. Spengler, P. T. P. Tang, M. Theobald, H. Toma, B. Towles, B. Vitale, S. C. Wang, and C. Young, Anton 2: Raising the Bar for Performance and Programmability in a Special-Purpose Molecular Dynamics Supercomputer, SC1 '14 Int. Conf. High Perform. Comput. Networking, Storage Anal. New Orleans, LA, USA , 41 (2014).

[10] D. E. Shaw, P. J. Adams, A. Azaria, J. A. Bank, B. Batson, A. Bell, M. Bergdorf, J. Bhatt, J. Adam Butts, T. Correi, R. M. Dirks, R. O. Dror, M. P. Eastwoo, B. Edwards, A. Even, P. Feldmann, M. Fenn, C. H. Fenton, A. Forte, J. Gagliardo, G. Gill, M. Gorlatova, B. Greskamp, J. P. Grossman, J. Gullingsrud, A. Harper, W. Hasenplaugh, M. Heily, B. C. Heshmat, J. Hunt, D. J. Ierardi, L. Iserovich, B. L. Jackson, N. P. Johnson, M. M. Kirk, J. L. Klepeis, J. S. Kuskin, K. M. Mackenzie, R. J. Mader, R. McGowen, A. McLaughlin, M. A. Moraes, M. H. Nasr, L. J. Nociolo, L. O'Donnell, A. Parker, J. L. Peticolas, G. Pocina, C. Predescu, T. Quan, J. K. Salmon, C. Schwink, K. S. Shim, N. Siddique, J. Spengler, T. Szalay, R. Tabladillo, R. Tartler, A. G. Taube, M. Theobald, B. Towles, W. Vick, S. C. Wang, M. Wazlowski, M. J. Weingarten, J. M. Williams, and K. A. Yuh, Anton 3: Twenty Microseconds of Molecular Dynamics Simulation before Lunch, in *SC21 Int. Conf. High Perform. Comput. Networking, Storage Anal. St. Louis, MO, USA,* (2021) pp. 1–11.

[11] C. Kutzner, S. Páll, M. Fechner, A. Esztermann, B. L. de Groot, and H. Grubmüller, More bang for your buck: Improved use of GPU nodes for GROMACS 2018, J. Comput. Chem. **40**, 2418 (2019), arXiv:1903.05918.

[12] R. Susukita, T. Ebisuzaki, B. G. Elmegreen, H. Furusawa, K. Kato, A. Kawai, Y. Kobayashi, T. Koishi, G. D. McNiven, T. Narumi, and K. Yasuoka, Hardware accelerator for molecular dynamics: MDGRAPE-2, Comput. Phys. Commun. **155**, 115 (2003).

[13] M. Taiji, T. Narumi, Y. Ohno, N. Futatsugi, A. Suenaga, N. Takada, and A. Konagaya, Protein explorer: A petaflops special-purpose computer system for molecular dynamics simulations, SC '03 Proc. 2003 ACM/IEEE Conf. Supercomput. Phoenix, AZ, USA, 2003 , 15 (2003).

[14] I. Ohmura, G. Morimot, Y. Ohno, A. Hasegawa, and M. Taiji, MDGRAPE-4: a special-purpose computer system for molecular dynamics simulations, Phil. Trans. R. Soc. A **372**, 20130387 (2014).

[15] D. Jones, J. E. Allen, Y. Yang, W. F. Drew Bennett, M. Gokhale, N. Moshiri, and T. S. Rosing, Accelerators for Classical Molecular Dynamics Simulations of Biomolecules, J. Chem. Theory Comput. **18**, 4047 (2022).

[16] N. Azizi, I. Kuon, A. Egier, A. Darabiha, and P. Chow, Reconfigurable molecular dynamics simulator, 12th Annu. IEEE Symp. Field-Programmable Cust. Comput. Mach. Napa, CA, USA, , 197 (2004).

[17] T. Hamada and N. Nakasato, Massively parallel processors generator for reconfigurable system, 13th Annu. IEEE Symp. Field-Programmable Cust. Comput. Mach. (FCCM'05), Napa, CA, USA, **2005**, 329 (2005).

[18] R. Scrofano, M. Gokhale, F. Trouw, and V. K. Prasannat, A hardware/software approach to molecular dynamics on

reconfigurable computers, 2006 14th Annu. IEEE Symp. Field-Programmable Cust. Comput. Mach. Napa, CA, USA, , 23 (2006).

[19] M. Gokhale, C. Rickett, J. L. Tripp, C. Hsu, and R. Scrofano, Promises and Pitfalls of Reconfigurable Supercomputing, Proc. Int. Conf. Eng. Reconfigurable Syst. Algorithms , 11 (2006).

[20] V. Kindratenko and D. Pointer, A case study in porting a production scientific supercomputing application to a reconfigurable computer, 4th Annu. IEEE Symp. Field-Programmable Cust. Comput. Mach. Napa, CA, USA , 13 (2006).

[21] S. R. Alam, P. K. Agarwal, M. C. Smith, J. S. Vetter, and D. Caliga, Using FPGA devices to accelerate biomolecular simulations, Computer (Long. Beach. Calif). **40**, 66 (2007).

[22] R. Scrofano, M. B. Gokhale, F. Trouw, and V. K. Prasanna, Accelerating molecular dynamics simulations with reconfigurable computers, IEEE Trans. Parallel Distrib. Syst. **19**, 764 (2008).

[23] S. Kasap and K. Benkrid, Parallel processor design and implementation for molecular dynamics simulations on a FPGA-Based supercomputer, J. Comput. **7**, 1312 (2012).

[24] J. Cong, Z. Fang, H. Kianinejad, and P. Wei, Revisiting FPGA Acceleration of Molecular Dynamics Simulation with Dynamic Data Flow Behavior in High-Level Synthesis, arXiv:1611.04474 (2016).

[25] A. G. Lawande, A. D. George, and H. Lam, Novo-G#: a multidimensional torus-based reconfigurable cluster for molecular dynamics, Concurr. Comput. Pract. Exp. **28**, 2374 (2016).

[26] Y. Gu, T. VanCourt, and M. C. Herbordt, Accelerating molecular dynamics simulations with configurable circuits, Int. Conf. F. Program. Log. Appl. 2005, Tampere, Finl. , 475 (2005).

[27] Y. Gu, T. Vancourt, and M. C. Herbordt, Improved interpolation and system integration for FPGA-based molecular dynamics simulations, 2006 Int. Conf. F. Program. Log. Appl. Madrid, Spain , 1 (2006).

[28] Y. Gu, T. VanCourt, and M. C. Herbordt, Explicit design of FPGA-based coprocessors for short-range force computations in molecular dynamics simulations, Parallel Comput. **34**, 261 (2008).

[29] M. Chiu, M. A. Khan, and M. C. Herbordt, Efficient calculation of pairwise nonbonded forces, 2011 IEEE 19th Annu. Int. Symp. Field-Programmable Cust. Comput. Mach. FCCM , 73 (2011).

[30] J. Sheng, B. Humphries, H. Zhang, and M. C. Herbordt, Design of 3D FFTs with FPGA clusters, 2014 IEEE High Perform. Extrem. Comput. Conf. (HPEC), Waltham, MA, USA , 1 (2014).

[31] C. Yang, T. Geng, T. Wang, R. Patel, Q. Xiong, A. Sanaullah, C. Wu, J. Sheng, C. Lin, V. Sachdeva, W. Sherman, and M. Herbordt, Fully integrated FPGA molecular dynamics simulations, SC '19 Proc. Int. Conf. High Perform. Comput. Networking, Storage Anal. 2019; , 1 (2019).

[32] C. Yang, T. Geng, T. Wang, C. Lin, J. Sheng, V. Sachdeva, W. Sherman, and M. Herbordt, Molecular dynamics range-limited force evaluation optimized for FPGAs, IEEE 30th Int. Conf. Appl. Syst. Archit. Process. , 263 (2019).

[33] M. Schaffner and L. Benini, On the Feasibility of FPGA Acceleration of Molecular Dynamics Simulations, arXiv:1808.04201 (2018).

[34] V. Kindratenko, I. S. Ufimtsev, and T. J. Martinez, Evaluation of two-electron repulsion integrals over gaussian basis functions on src-6 reconfigurable computer, in *Proceedings of the 4th Annual Reconfigurable Systems Summer Institute (RSSI'08)* (2008).

[35] M. S. Gordon, G. Barca, S. S. Leang, D. Poole, A. P. Rendell, J. L. Galvez Vallejo, and B. Westheimer, Novel Computer Architectures and Quantum Chemistry, J. Phys. Chem. A **124**, 4557 (2020).

[36] J. M. RodrÍguez-Borbón, A. Kalantar, S. S. Yamijala, M. B. Oviedo, W. Najjar, and B. M. Wong, Field Programmable Gate Arrays for Enhancing the Speed and Energy Efficiency of Quantum Dynamics Simulations, J. Chem. Theory Comput. **16**, 2085 (2020).

[37] X. Wu, T. Kenter, R. Schade, T. D. Kuhne, and C. Plessl, Computing and Compressing Electron Repulsion Integrals on FPGAs , in *2023 IEEE 31st Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)* (IEEE Computer Society, Los Alamitos, CA, USA, 2023) pp. 162–173.

[38] X. Zhang, Y. Xia, Z. Kang, X. Du, and J. Gao, A review of fpga-based graph neural network accelerator architectures, in *Proceedings of the 2024 7th International Conference on Signal Processing and Machine Learning*, SPML '24 (Association for Computing Machinery, New York, NY, USA, 2024) p. 335–343.

[39] E. Luttmann, D. L. Ensign, V. Vaidyanathan, M. Houston, N. Rimon, J. Øland, G. Jayachandran, M. Friedrichs, and V. S. Pande, Accelerating molecular dynamic simulation on the cell processor and playstation 3, Journal of Computational Chemistry **30**, 268 (2009), https://onlinelibrary.wiley.com/doi/pdf/10.1002/jcc.21054.

[40] I. S. Ufimtsev and T. J. Martinez, Quantum chemistry on graphical processing units. 2. direct self-consistent-field implementation, Journal of Chemical Theory and Computation **5**, 1004 (2009).

[41] S. Le Grand, A. W. Götz, and R. C. Walker, Spfp: Speed without compromise—a mixed precision model for gpu accelerated molecular dynamics simulations, Computer Physics Communications **184**, 374 (2013).

[42] S. C. Harvey, R. K.-Z. Tan, and T. E. Cheatham, The flying ice cube: Velocity rescaling in molecular dynamics leads to violation of energy equipartition, J. Comput. Chem. **19**, 726 (1998).

[43] H. J. C. Berendsen, J. P. M. Postma, W. F. van Gunsteren, A. DiNola, and J. R. Haak, Molecular dynamics with coupling to an external bath, J Chem Phys **81**, 3684 (1984).

[44] T. Morishita, Fluctuation formulas in molecular-dynamics simulations with the weak coupling heat bath, J. Chem. Phys. **113**, 2976 (2000).

[45] G. Bussi, D. Donadio, and M. Parrinello, Canonical sampling through velocity rescaling, J. Chem. Phys. **126**, 014101 (2007).

[46] A. Rahman, Correlations in the Motion of Atoms in Liquid Argon, Phys. Rev. **136**, A405 (1964).

[47] H. J. C. Berendsen, J. R. Grigera, and T. P. Straatsma, The missing term in effective pair potentials, J. Phys. Chem. **91**, 6269 (1987).

[48] J. K. Johnson, J. A. Zollweg, and K. E. Gubbins, The lennard-jones equation of state revisited, Mol. Phys. **78**, 591 (1993).

[49] S. T. Lin, M. Blanco, and W. A. Goddard, The two-phase model for calculating thermodynamic properties of liquids from molecular dynamics: Validation for the phase diagram of Lennard-Jones fluids, J. Chem. Phys. **119**, 11792 (2003).
[50] E. A. Mastny and J. J. De Pablo, Melting line of the Lennard-Jones system, infinite size, and full potential, J. Chem. Phys. **127**, 104504 (2007).
[51] L. A. Báez and P. Clancy, The kinetics of crystal growth and dissolution from the melt in Lennard-Jones systems, J. Chem. Phys. **102**, 8138 (1995).
[52] P. R. Ten Wolde, M. J. Ruiz-Montero, and D. Frenkel, Numerical calculation of the rate of crystal nucleation in a Lennard-Jones system at moderate undercooling, J. Chem. Phys. **104**, 9932 (1996).
[53] H. Wang, H. Gould, and W. Klein, Homogeneous and heterogeneous nucleation of Lennard-Jones liquids, Phys. Rev. E - Stat. Nonlinear, Soft Matter Phys. **76**, 031604 (2007).
[54] F. Trudu, D. Donadio, and M. Parrinello, Freezing of a Lennard-Jones fluid: From nucleation to spinodal regime, Phys. Rev. Lett. **97**, 105701 (2006).
[55] S. E. Lundrigan and I. Saika-Voivod, Test of classical nucleation theory and mean first-passage time formalism on crystallization in the Lennard-Jones liquid, J. Chem. Phys. **131**, 104503 (2009).
[56] J. R. Espinosa, C. Vega, C. Valeriani, and E. Sanz, Seeding approach to crystal nucleation, J. Chem. Phys. **144**, 034501 (2016).
[57] V. G. Baidakov and K. R. Protsenko, Spontaneous Crystallization of a Supercooled Lennard-Jones Liquid: Molecular Dynamics Simulation, J. Phys. Chem. B **123**, 8103 (2019).
[58] D. Turnbull and J. C. Fisher, Rate of nucleation in condensed systems, J. Chem. Phys. **17**, 71 (1949).
[59] D. W. Oxtoby, Homogeneous nucleation: Theory and experiment, J. Phys. Condens. Matter **4**, 7627 (1992).
[60] O. Steinhauser, Reaction Field Simulation of Water, Mol. Phys. **45**, 335 (1982).
[61] J. P. Ryckaert, G. Cicotti, and H. J. C. Berendsen, Numerical integration of the cartesian equations of motion of a system with constraints: molecular dynamics of n-alkanes, J. Comput. Phys. **23**, 327 (1977).
[62] L. Greengard and V. Rokhlin, A fast algorithm for particle simulations, J. Comput. Phys. **73**, 325 (1987).
[63] pmemd.cuda gpu implementation, https://ambermd.org/gpuperformance.php (2024).
[64] Openmm. https://openmm.org/benchmarks (2024).