Upgrade of FPGA Range-Limited Molecular Dynamics to Handle Hundreds of Processors

Chunshu Wu*, Tong Geng*†, Sahan Bandara*, Chen Yang*, Vipin Sachdeva‡, Woody Sherman‡, Martin Herbordt*
*Department of Electrical and Computer Engineering, Boston University, Boston, MA
†Pacific Northwest National Lab, Richland, WA ‡Silicon Therapeutics, Boston, MA
Email: *{happycwu,tgeng,sahanb,cyang90,herbordt}@bu.edu, †tong.geng@pnnl.gov, ‡{vipin,woody}@silicontx.com

Abstract—**With the current pandemic, the central role that Molecular Dynamics simulation (MD) plays in drug discovery makes advances in MD performance urgent. Recent work has demonstrated that among COTS devices only FPGA-centric clusters can scale beyond a few processors for relevant targets; other work has shown that single FPGA performance compares favorably to that of a GPU. In this study we demonstrate that an additional factor of 4× performance can be achieved which results in a factor of 5× speed up over a GPU. The problem addressed is that the designs of the last decade no longer scale when the number of processing pipelines grows from around ten to the hundreds. We begin by systematically evaluating existing work, exposing its flaws, and proposing a series of new design solutions. There are four major contributions. First, we address the massive routing problem by augmenting the design with three minimal networks in logic and latency. Second, we have developed a novel asynchronous out-of-order communication mechanism that removes nearly all bubbles from the routing networks. Third, we find that inverting the standard particle access algorithm results in improved locality and performance. Finally, we have created a custom numerical format that increases precision while saving space and logic.**

I. Introduction

Simulating Molecular Dynamics (MD) is one of the fundamental approaches to modeling molecular systems and is used across the fields of pharmacology, biology, and chemistry. Of particular importance are running simulations of long timescales [1], [2]. Such simulations of small sets of particles ($\sim$50K) are crucial in pharmaceutical industries for drug discovery. For example, the main protease (drug target protein) of COVID-19 consists of 2367 atoms, and it can take tens of nanoseconds for the protein and a drug candidate (typically small molecules, based on Lipinski's rule of five) to reach structural stability [3]. However, orders of magnitude longer are often required to achieve convergence, e.g., in computing relative binding free energies between congeneric molecules.

Numerous MD software packages [4]–[8] have been developed, with or without GPU acceleration. While GPUs are immensely beneficial for batch processing and for simulating large systems, they face scalability problems: the efficiency of GPUs is likely to drop greatly as the number of particles decreases [9]. One solution is ASIC-based supercomputer specifically for MD [10], [11]. FPGAs, however, have the advantage of both being COTS and supporting flexible, direct (and low-latency) communication.

In classical MD, the main computation (the non-bonded forces) is usually partitioned into range-limited (RL) and long-range (LR) components. RL consists of 95% of the FLOPs and was the subject of much study in the first generation of FPGA MD implementations [12]–[19]. More recently FPGA MD work has concentrated on demonstrating scalability of LR [20]–[25]. But without being competitive device to device, which means primarily with respect to RL, this scalability will only be useful for large clusters and therefore of limited potential value. Recent work, however, has shown just that: FPGAs can be competitive with GPUs [26], [27].

In the work here we note that despite these successes, much work remains. The problem is that, as FPGA technology has advanced from 2009 to 2021, the computational logic per FPGA has increased from 8-10 force processors to 200-250. Since each of these processors is fed by 6-10 filter pipelines, MD designs now need to be able to read and write around 2000 data records per cycle. Since the pipelines operate on heavily overlapped data structures, MD designers are faced with the problem common to complex FPGA designs: balancing efficiency (e.g. stall frequency), routing complexity, and operating frequency.

The current state of the art is a modified version of the design found in [28]–[31]. But without specifying a network (not needed in 2011), we are now faced with designs that take immense effort to simply P&R. And, when they do, either only use a fraction of chip resources or operate at a low frequency.

We begin by appending an explicit network that accounts for 3D to 2D mapping. The PEs and data caches are logically distributed in 3D space, but physically on a 2D chip. In the previous FPGA/MD system, they are directly connected. As a result, the interconnect among PEs and data caches becomes over-complex, and the number of PEs is difficult to scale up without significantly harming the frequency. In the 512-node (8x8x8) Anton 2 machine, the nodes are structured and connected as a 3D torus, and the longest connection is almost 2 meters long [32]. On the other hand, for a single FPGA, neither the space nor the wiring resources are practical for such topology. In fact, this mapping problem is a typical NP-hard graph embedding problem. Such embedding problem has been studied [33]–[35], but not completely solved. In this case, we propose and explore a ring routing method.

We find that solving one problem begets another: low utilization due to bubbles in the network. To solve this problem we develop an out-of-order particle broadcast mechanism which compensates for the high latency caused by the ring.

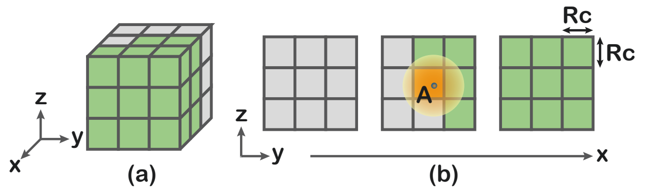

Fig. 1. The simulation space is divided into cells (only 3x3x3 cells are shown) whose edge length is the same as the cutoff radius R_c. (a) A home cell and its surrounding neighbor cells. (b) 2-D view of the 3x3x3 cells. Intuitively, the particles in green cells are sent to the home cell at the center and form pairs with home cell particles. The grey cells can be ignored because of Newton's 3rd law. The orange aura shows the interaction range of particle A.

Next, the lack of data locality leads to a heavily congested force transfer network. Figure 1 depicts the half-shell method [36] that exploits Newton's third law. In the previous FPGA/MD system, the neighbor particles are traversed and broadcast from green cells to the orange home cell (and other related home cells) once per cycle. Computed partial force fragments are returned to the source cells for accumulation. However, the distribution of the destinations of the partial forces lead to low concurrency and congestion at the destinations. In the worst case, 14 force fragments request writes to the same destination in a single cycle, while plenty of other destinations have empty inputs.

Finally, bulk synchronization is applied so that the input data can easily be shared among multiple PEs. The drawback is that all PEs must now wait for the slowest PE to finish evaluating a batch of particles. Algorithm 1 sketches the baseline procedure of a single PE. N filters in each PE are equipped to select the particle pairs that satisfy the cutoff radius. The inner loop workload tends to be imbalanced for different PEs, as the filter pass rate may differ drastically based on the particle positions. In order to eliminate the bubbles (that the synchronization causes) without giving up the particle broadcast mechanism, we propose a novel design that makes the outer loops of different PEs overlap, so that the filters and force evaluation units can be executed without stalls. For those reasons, we propose a fully pipelined range-limited molecular dynamics (FPMD-RL) system, an FPGA-based accelerator for MD-RL simulation with all data paths efficiently pipelined. The major contributions are listed below.

Algorithm 1 High-level single PE baseline procedure

```
1: for particle ($p_1$) in home cell do
2:     for particles ($p_2[0 : N-1]$) in neighbor cells do
3:         $pairs = Filter(p_1, p_2[0 : N-1])$
4:         $forces = ForceEvaluation(pairs)$
5:         $Return(forces)$
6:     end for
7:     $SynchronizeAllPEs()$
8: end for
```

- To support hundreds of PEs, the ring router is studied. This efficiently maps the 3-D interconnect onto the 2-D FPGA fabric; the high latency is completely hidden by an out-of-order broadcast mechanism;
- To avoid the lack of concurrency during force return, and to reduce the congestion in the network, a home cell particle traversal method is proposed.
- A PE-overlapping force evaluation scheme is proposed to eliminate the load imbalance problem introduced by bulk synchronization.
- Some other optimizations are made to reduce hardware cost and increase efficiency, among which the most important is the use of a custom numerical format.

These contributions collectively lead to a factor of four increase in performance.

II. MD Range-Limited Background

MD Context. MD alternates between force calculation and motion update. The forces computed may include bonded and non-bonded terms [37]. Non-bonded is often partitioned into long-range (LR) and range-limited (RL) components. These computations are generally optimized separately; this work can therefore be viewed in the context of a comprehensive implementation [27], [38] with other components (LR [39], [40] and bonded [41]) remaining fixed.

RL Force Properties. The RL force originates from Lennard-Jones potential (Equation 1, the potential between particle i and j), which is widely used to describe the interactions between electronically neutral particles. The resultant force on particle i is shown in Equation 2, where ϵ and σ are constants determined by particle types, and r_{ij} is the distance between particle i and j.

$$V_{ij}^{LJ} = 4\epsilon_{ij}[(\frac{\sigma_{ij}}{r_{ij}})^{12} - (\frac{\sigma_{ij}}{r_{ij}})^{6}] \tag{1}$$

$$F_i^{LJ} = \sum_{j\neq i} \frac{\epsilon_{ij}}{\sigma_{ij}^2}[48(\frac{\sigma_{ij}}{|r_{ji}|})^{14} - 24(\frac{\sigma_{ij}}{|r_{ji}|})^{8}]\mathbf{r}_{ji} \tag{2}$$

It is obvious that the RL force decays rapidly with the increasing r, therefore a cutoff radius R_c is introduced to set the interaction range and to avoid trivial computations of distanced particles. As a result, the $o(N^2)$ pair-wise computation complexity is reduced to $o(N)$.

Periodic Boundary Conditions (PBCs). PBCs are frequently used in computational physics: a particle that moves out from the simulation space re-enters from the opposite side. Particle interactions also wrap around the boundaries.

Operation. An iteration (~2 fs of reality) is separated into two phases: *Force evaluation* and *motion update*. The resultant forces on all particles are obtained after force evaluation, and the particle position and velocity data are updated based on, e.g., Verlet's algorithm [42] during motion update.

Particle Data Structure. *Cell lists* [43], [44] are used for particle indexing, as Figure 1 shows. All the particles in neighbor cells are potential neighbor particles with respect to the particles in home cells. Filters are deployed to select the pairs within the cutoff range from home cells and neighbor cells [28] and replace the neighbor lists used in CPU implementations. FPGA implementations generally have a cell size = R_c.

Particle Migration. A particle may change cells after an iteration. The lists need to be updated every a few iterations to handle the migration, with the cutoff radius extended by the particle drift distance [45]. We use double buffering (e.g. [26]) to reduce computation at the expense of BRAMs.

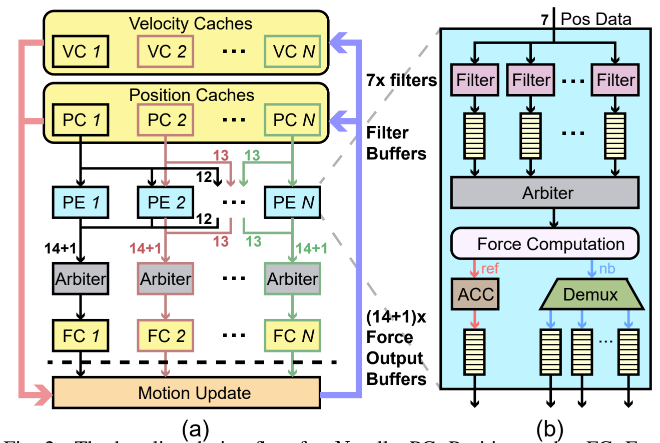

Fig. 2. The baseline design flow for N cells. PC: Position cache. FC: Force cache. VC: Velocity cache. The numbers next to the arrows stand for the number of data sources or destinations. For example, the data from PC 0 is broadcast to 14 PEs. The "+1" emphasizes the force data of reference particles. The dashed line above motion update (MU) means the motion update starts after the force evaluation.

Particles per Cell. A value that affects computational efficiency is the number of particles per cell [46]. This is related to the cut-off radius and the material being studied with typical values from ~40 to several hundred.

III. Baseline Design

In this Section, the data mapping scheme and data flow of the baseline design are introduced, followed by details about force evaluation and motion update.

A. Data-PE Mapping

As illustrated in Figure 1, particles in a simulation space are spatially grouped by cubic cells. The position, force, and velocity data of the particles in the same cell are stored in three separate caches. All data are represented with single precision floating point numbers.

The work for PEs is also spatially distributed. Each PE evaluates the pairwise forces between particles in a specific *home cell* and particles in 13 neighbor cells. Figure 2(a) shows that in the baseline scheme, a PE is connected to 14 position data caches based on cell locations.

As the force evaluation proceeds, the evaluated forces are returned to the force caches in their corresponding cells. Therefore, through the arbiters, the output of a PE is connected to 14 force caches from the same cell as the 14 position data caches as just mentioned (Figure 2(a)).

B. PE Functions

Figure 2(b) shows a single PE. The major functions are particle filtering and force evaluation.

Filters. To guarantee throughput and match the number of neighbor cells (14 cells), 7 filters are deployed. The ideal pass rate of a filter is

$$\frac{\frac{4}{3}\pi r_c^3}{27 \times r_c^3} = 15.5\% \tag{3}$$

where the numerator is the volume of a cutoff sphere, and the denominator is the volume of 3x3x3 cells. Although each PE receives data from 14 position caches (PCs), only 7 position data can be used simultaneously (number of filters).

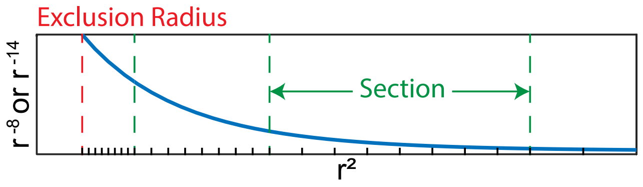

Fig. 3. Schematic of the table-lookup interpolation method. The lengths of sections vary in log-scale, and each section is uniformly divided into 256 intervals (only 8 are shown). Particles within another particle's exclusion radius are evaluated separately.

For each cycle during force evaluation, a filter receives a new neighbor particle's position data and the reference particle's position data stored locally in a register (not shown). The reference position data is shared among the 7 filters and is discarded after all neighbor position data have been evaluated with it. A new reference particle is then fetched.

A planar filtering technique is used to avoid overwhelming DSP usage for floating point operations. The floating point position data are converted to 28-bit fixed point data at this point such that the position difference can be obtained without DSP. For the position difference x, y, and z between two particles, the particle pair passes the filter if

$$\begin{aligned} &|x| < r_c,\ |y| < r_c,\ |z| < r_c, \\ &|x| + |y| < \sqrt{2}r_c,\ |y| + |z| < \sqrt{2}r_c,\ |z| + |x| < \sqrt{2}r_c, \\ &|x| + |y| + |z| < \sqrt{3}r_c \end{aligned} \tag{4}$$

are satisfied. The position data and IDs of neighbor particles that pass the filters are then stored in filter buffers and ready for (round-robin) arbitration.

Force evaluation. Instead of directly computing the forces based on Equation 2, we use an interpolation method with table-lookup [47]. As Figure 3 shows, for a given r^2, its corresponding section and interval can be located. The curve segment within the interval is linearly approximated, such that parameter a and b are obtained from the look-up table based on r^2, and r^{-k} ($k = 8$ or 14) is interpolated as

$$r^{-k} = ar^2 + b \tag{5}$$

Force return is initiated for an evaluated force fragment. Based on Newton's 3rd law, the force is accumulated with the force of the reference particle, and also sent to one of the force output buffers as a neighbor force (Figure 2(b)). The output buffers are designed to resolve conflicts upon force return.

C. Synchronization

Synchronization is applied at 3 points: New reference particle fetch, force evaluation completion, and motion update completion. A new reference particle is not fetched until all filter buffers of all PEs are empty. This is because, first, the position broadcast mechanism demands all PEs receive particles with the same ID – it's convenient for all PEs to start with the same particle ID, thus the global synchronization. And second, the reference position is not stored in filter buffers, but in a register (shared by the force computation unit and all the filters) to save BRAMs. As a result, the reference position cannot be discarded until all related operations are done. A read-after-write (RAW) hazard can occur if the motion update

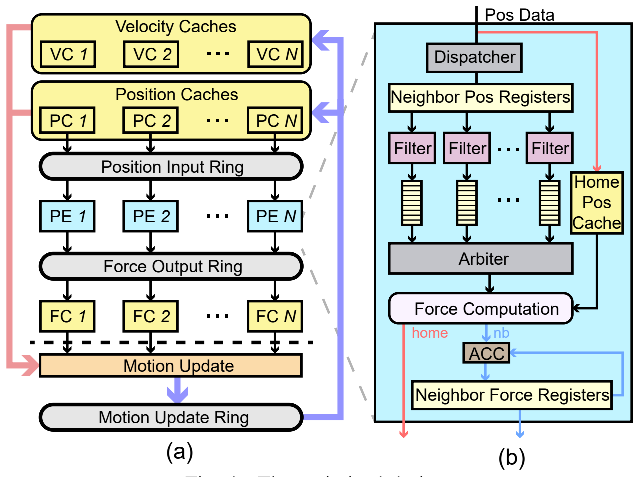

Fig. 4. The optimized design.

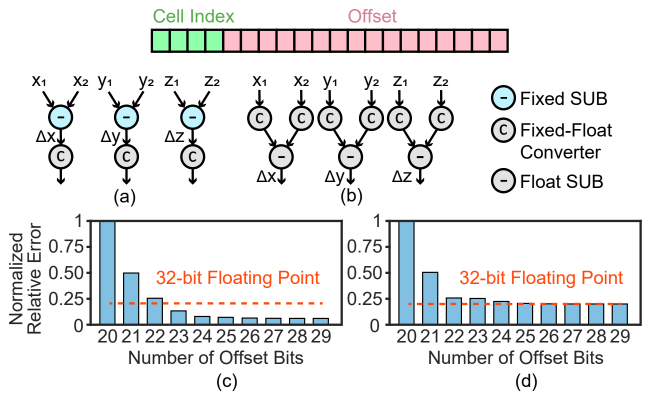

Fig. 5. The data format and precision analysis. Top: The fixed-point position format. (a) and (b): Two methods of computing the displacements of particles. (c) and (d): Error analysis of the two methods with different numbers of bits.

is run during force evaluation. A simple way to avoid these is for global synchronizations to occur separately after force evaluation and motion update.

D. Motion Update

A particle may end up in another cell after motion update. Because of the time interval and particle energy, a particle can move no further than to a cell next to its original cell. A motion update unit is therefore connected to (at least) 27 position caches and 27 velocity caches. Double buffering is used to resolve the discontinuity in data caches caused by particle migration. After a particle is updated, the updated position and velocity are written into a new set of caches, this way no packing is needed.

IV. FPMD-RL Architecture

The FPMD-RL design is described in this Section. By combining the five key points, the system is able to scale with the increasing number of PEs. The index-offset position format and the daisy chain based routing topology provide a simplified and cost-effective memory-PE data path. The data caching method solves the ring's throughput problem by drastically increasing the locality of data, with the out-of-order particle position broadcast mechanism appropriately modified. Finally, the PE overlapping method solves the the severe workload imbalance problem from the data caching.

A. Position Data Format

Compared with the 32-bit floating point position data in the baseline design, we find that a fixed-point data format has greater potential.

The Index-Offset Format. For scalability and data accuracy, the cutoff radius is normalized to 1, such that the position data can be conveniently divided into two parts: The cell index (integer part) and the offset (fraction part) (Figure 5). The cell index gives the cell id along x, y, or z axis, while the offset is the distance between a particle and the lower cell boundary of x, y, or z direction.

Since all data in a position cache share the same cell index, only the offset is stored. As a result, the position data path can be narrowed considerably (from 96 bits, 3x floating point numbers to even fewer than 70 bits). One significant benefit is that 33% of the BRAM resources used for position data storage can possibly be saved.

Displacement Computation Methods. Two displacement computing methods during force evaluation are sketched. In Figure 5(a), the fixed-point positions are subtracted before being converted to 32-bit floating point format. The computed forces are compared with 64-bit floating point forces, and the error results are displayed in Figure 5(c). The absolute relative error (typically $10^{-6}\sim10^{-5}$) varies with interpolation parameters and system parameters like σ; it is therefore normalized.

The second method is shown in Figure 5(b), where the six positions are converted to floating point before the subtraction. Although the number of converters is doubled, the hardware cost of a converter is negligible. This is because if the cell index starts from 1 instead of 0, then the leading-one of the position can be easily justified, so that the conversion is easily done with modest loss of precision (shown in Figure 5(d)).

Comparison. In method 2, a converter costs only $\sim$15 ALMs, compared with $\sim$170 ALMs of method 1. The Method 2 converter also improves latency (1 cycle vs $\sim$5 cycles). However, three more DSPs are required in method 2, and the precision does not change much for $>$22 offset bits.

B. Data Routing Mechanism

It is shown in Figure 4(a) that the PC-PE, PE-FC, and MU-PC/VC interconnects are replaced by daisy chain based rings to avoid frequency degradation. We choose the 1D topology over 2D based on the facts that the ring is more cost-effective and that the latency, in any case, can be hidden.

Ring Functions. The ring interconnect is shown in Figure 6, where two data paths are possible. If the data source and the data destination match in space. For example, the i'th PE can directly send force data to the i'th force cache. If the source and the destination do not match, the data enters the one-way ring path when the data slot in the network node is empty. The data in a network node is forwarded to the node next to it for every cycle, unless the data has reached its final destination.

The three rings have different functionality. A position datum has multiple destinations during position broadcast;

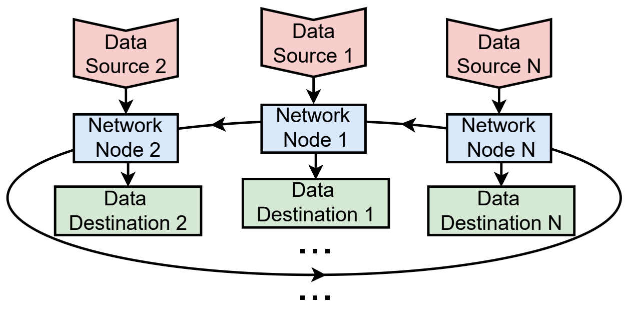

Fig. 6. The ring routing path. Position input ring: The data sources are PCs, and the data destinations are PEs. Force output ring: The sources are PEs, and the destinations are FCs. Motion update ring: The sources are motion update units, and the destinations are PCs and VCs.

therefore the data is kept in the position input ring until it arrives at the final PE. In contrast, the data in the force output and the motion update ring have only a single destination.

Cell-Ring Mapping. The main goal of the cell-ring mapping is to minimize the average data lifetime in a ring. We manage to obtain a reasonable mapping method which follows:

$$I_r = N^y_{cell} N^z_{cell}(x-1) + N^z_{cell}(y-1) + z \qquad (6)$$

I_r is the index shown in Figure 6 which starts from 1, N^y_{cell} is the number of cells along y axis, and x is the coordinate of cells on x direction, ranging from 1 to N^x_{cell}.

The equation represents an approximate solution: finding the best solution is an NP hard problem (N! possibilities). The spatial distribution of the 14 neighbor cells and the periodic boundary conditions also complicate the problem because the neighbor cells can be far away from the home cell on the ring. The average particle lifetime is evaluated for more than 100M randomly generated mapping approaches. The result shown leads to a solution better than any of these and twice the performance of the average.

Ring Concurrency Although the position input and the force output processes' latency can overlap with the force evaluation flow, it is possible that the latency may not be completely hidden when the number of cells is too large or the number of particles per cell too few, especially for the force output ring because the force data cannot be reused. In order to hide the latency introduced by the rings, the concurrency of the rings is adaptive to the dataset so that multiple data slots can be deployed on a single network node.

The motion update ring is only used when particle migration happens, and particle migration is relatively rare because of the particles' limited kinetic energy. Therefore congestion in the motion update ring is not a concern.

C. Data Caching

Neighbor Data Caching. The rings solve the path routing problem (P&R) and enable a high operating frequency. The increase in latency, however, means that, without modification, the desired throughput of neighbor position broadcast and force return cannot be met. To solve this problem, a neighbor particle data caching technique is used.

As Figure 4(b) shows, neighbor position data, including the home cell particle as a neighbor particle, are temporarily stored in registers through the dispatcher (the dispatcher is described in *PE Overlapping*). Instead of iterating through all neighbor positions for a single reference particle, the home cell positions are now traversed for those neighbor positions in registers.

The most straightforward benefit is that a PE requests data from neighbor cells much less frequently. Another is that the neighbor partial forces can be accumulated before being sent to the force caches. This way the congestion of the force output ring is greatly alleviated. Only one packet need to be sent through the ring interconnect after the evaluation of a particle from a neighbor cell. The expensive force output buffers in figure 2(b) are no longer needed, either.

Home Cell Data Caching. As the particle broadcast proceeds, the position data from the home cell are duplicated into the *Home Pos Cache* shown in Figure 4(b) (red arrow). The duplication can be done during the first round of position broadcast. The duplication aims to double the concurrency of the position cache and eventually saves a considerable amount of BRAM resources. Different position cache entries may be needed in both the filters and the force computation unit, thus the need for doubling the concurrency. In the baseline model, the position data are stored in filter buffers. Now that the neighbor particles are cached in registers, the filter buffers are only used to hold home cell particle IDs so that the desired position data can be fetched from the duplicated Home Pos Cache. With the home cell data caching method, 12 BRAMs per PE are saved.

D. Out-of-Order Particle Position Broadcast

The position broadcast mechanism also evolves with the data caching. Now that a neighbor particle position can be reused for an entire home particle traversing process, the broadcast doesn't need to be performed every cycle, except for under two conditions.

First, a slot is available in the position input ring (see Figure 7). In the 1st cycle, all input ring slots are empty; thus all position caches send the current value to the ring along the blue arrows. Afterwards, the position cache entries are marked as used with dirty bits.

Second, the pointed entry is not marked used. An address pointer is deployed to iterate through all position cache entries, and the position data are sent to the local PEs directly as shown by the red arrows. In the 5th cycle, the pointed entry of the Nth position cache is not used, and the corresponding slot in the ring is empty. Therefore data is broadcast along the blue arrow.

In the 8th cycle, the red arrow is from entries with different addresses, since the particle numbers are different for those caches. The asynchronous home particle traversal also makes the back pressure in filter buffers much cheaper: Only the related position cache traversal needs to be paused, rather than all of the caches, Therefore the filter buffers can be small and implemented with relatively cheap memory logic array blocks (MLABs) instead of M20Ks.

In the 9th cycle, all three slots in the ring are empty. Now that the entries pointed to in cache 1 and cache 2 are clean, the data within them are sent to the ring for broadcast, despite the difference in addresses.

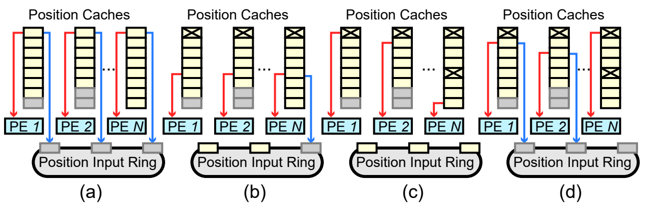

Fig. 7. Broadcast sketches. Each position cache has at most 8 particles for example. Gray boxes: Empty entries. Boxes with "x": Marked as used. (a) The 1st cycle. (b) The 5th cycle. (c) The 8th cycle. (d) the 9th cycle.

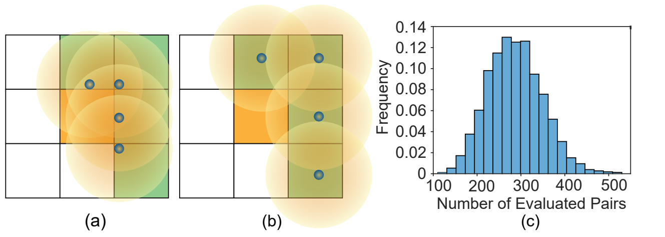

Fig. 8. Example cases of (a) high filter pass rate, (b) low filter pass rate in 2D illustration, (c) the percentage of PEs that have certain number of filter-passing pairs for a neighbor particle. The data are collected from an entire iteration. Dataset: 4x4x4 cells, 100 particles per cell.

E. PE Overlapping

The PEs are synchronized before the next reference particle positions are received. However, a workload imbalance problem escalates severely if the synchronization is inherited under the neighbor particle caching scheme, as Figures 8(a) and (b) show. The filter pass rate can vary drastically, depending on the neighbor particle positions.

The imbalance in filtering rate leads directly to imbalanced workload of the PEs. Figure 8(c) shows the distribution of workload among PEs of an example dataset. Within a single iteration, the number of evaluated pairs with respect to a single neighbor particle can vary as much as a factor of 5. Since it is no longer affordable to perform a bulk synchronization, a PE overlapping method is proposed.

Position Data Dispatch. Although not all neighbor particles can arrive at the same time with the position input ring, we manage to take advantage of it by designing the data loading hardware shown in Figure 9(a). The active register (AR) and backup register (BR) are used as neighbor position holders. ARs can be used in both a filter and the force computation unit, while a BR can only be used in a filter.

The input position data are stored in the position input buffer upon arrival. The arbiter first checks whether there is an AR slot empty, and if so fills it. If all AR slots are filled, then the arbiter sends the position data into an empty BR slot. When an AR's content has finished with all the filtering, but is still needed for force computation (in other words, particle pairs that related to this particle still exist in the filter buffer), its bundled BR can bypass the filter input and form pairs with home cell particles to keep the filter busy. A BR's content is moved to AR after the previous data in AR has been discarded.

Data Lifetime. The position data lifetime for filtering is determined by address stamps. Since the home particle position is traversed in loops and is independent of the neighbor particles, the address of the home particle is remembered as an address stamp when a new particle is dispatched. When the

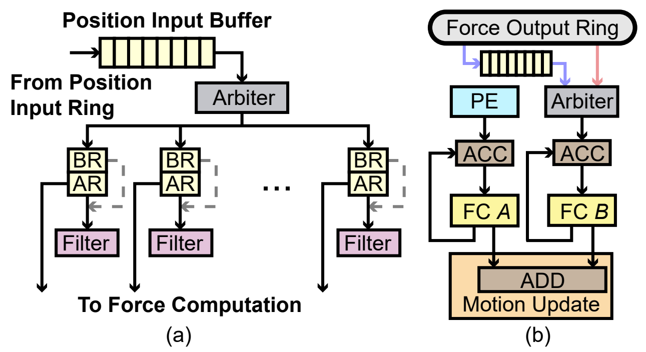

Fig. 9. (a) Position data dispatching. AR: Active register. BR: Backup register. The gray dashed arrows show that the neighbor data entering the filters can be bypassed. (b) Force accumulation. Red arrow: Data from slot 1 of a node. Blue arrow: Data from other slots of the same node.

address reaches the stamp again the dispatcher knows that the particle is finished with filtering. The position data lifetime for force computation is also well-defined. Neighbor position data stored in an AR can be discarded after all the related filtering is done and it does not appear in the corresponding filter buffer. Force data is stored in neighbor force registers and is sent to the force output ring as soon as it is completed.

F. Yin-Yang Force Caches

The force accumulation also suffers from a concurrency problem. The partial forces from the force output ring, together with the forces from the PE demand more than 1 write operation per cycle. The solution is shown in Figure 9(b). This force accumulation layout trades memory blocks for performance by using two force caches instead of one. The home cell forces are sent directly from the local PE to the local force cache A (Yang cache), and force cache B (Yin cache) is shared by all the data from the output ring. In case multiple slots exist in a ring node, we give priority to those slots to guarantee that the force from one of the slots can always participate in the accumulation whenever the data are ready, such that no data buffering is required for this slot (see the red arrow).

The force data are only used in motion update units. Therefore the half forces in Yin-Yang caches are then combined as a whole in motion update units.

V. Evaluation

A. Experimental Configuration

The design is implemented in Verilog HDL on an Intel D5005 board equipped with a Stratix 10 SX FPGA, which has 933120 ALMs, 5760 DSPs, and 11721 M20K BRAMs. The performance and hardware consumption are evaluated based on the board's specification.

For performance comparison, we run Amber [48] on an Intel Xeon 24-core CPU and an Nvidia GTX 1080Ti GPU. The time step is set to 2fs for all evaluations.

As for system configuration, six, rather than seven filters per PE are used because the average filter rate becomes $\sim$17% with the planar filtering method, and the number of filters no longer needs to match the 14 neighbor cells. Each motion update unit is assigned eight cells with the number of motion update units scaling with the number of cells.

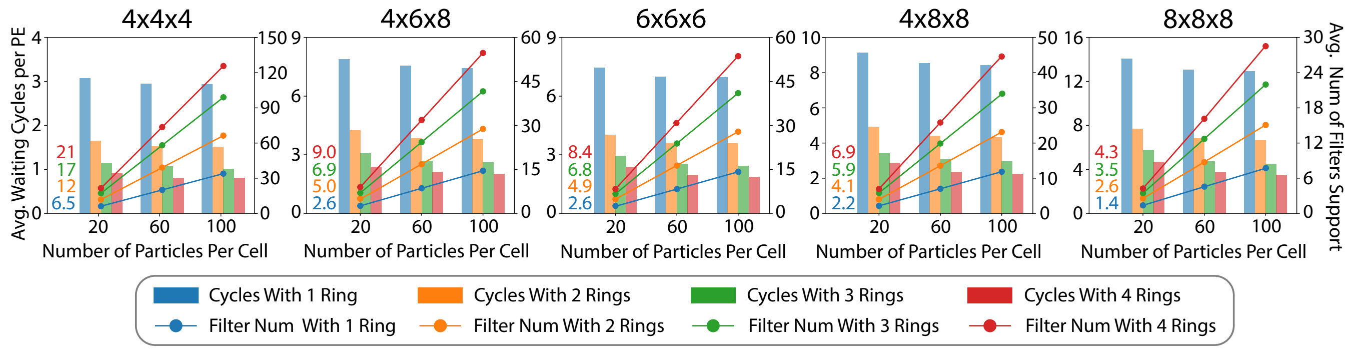

Fig. 10. The average PE waiting time and average number of filters that can run without pausing for input data. Colored numbers: Data points of the line plots for 20 particles per cell.

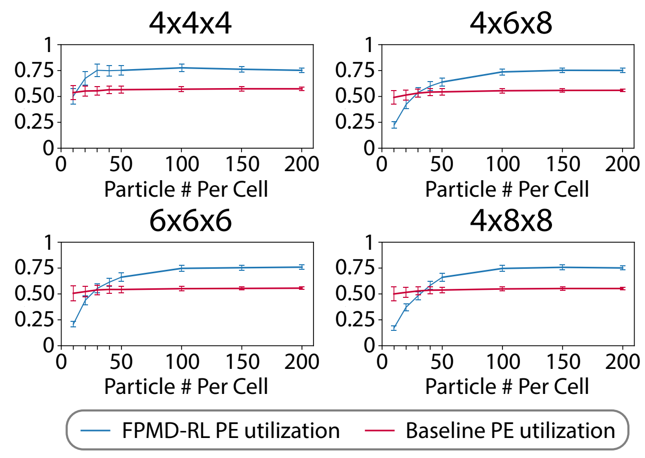

Fig. 11. PE utilization

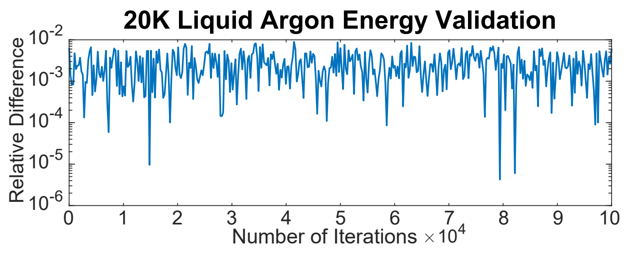

Fig. 12. Energy convergence validation

B. The Position Input Ring Evaluation

In this Section, we demonstrate that a small number of position input rings is sufficient for likely numbers of particles per cell. The force input ring is evaluated for five typical cell layouts as Figure 10 illustrates. The bar plots show the average number of cycles needed for each PE to wait before a consecutive position datum arrives through the position input ring. The line plots show the average number of filters that can operate without waiting for input data. The ring is evaluated with up to 100 particles per cell, and up to 4 input rings; this is sufficient because the trend is the same for more particles and more rings. The colored numbers represent the leftmost line plot data points and are given for reference.

It can be observed that the number of supported filters scales with the number of particles per cell. This implies that the ring latency can be completely hidden by filter operations.This feature also allows us to have two or more PEs working on one home cell (i.e. more filters for more PEs). Another observation is that the average PE waiting time only increases slowly with the number of cells, meaning that a certain number of rings can handle a wide range of cell counts. The benefit of extra rings is clear: >3.02x throughput for 4 rings, >2.45x for 3 rings, >1.83x for 2 rings. This also implies that only a small number of rings are needed in practice.

C. PE Utilization

Figure 11 compares the overall baseline and FPMD-RL PE utilization for different datasets. The error bars represent the standard deviation of individual PE utilization. In order to evaluate how many bubbles are caused by workload imbalance, the PE utilization is measured for all PEs during the time interval between the first and last force evaluations.

For all cases, the PE utilization of the new design reaches 77% for more than 100 particles per cell with small deviation among all PEs. The utilization also converges for more particles per cell. This shows the system is robust with respect to data set choices. The baseline utilization starts higher for a small number of particles per cell, but converges below 57%. Despite the baseline PE imbalance problem is not as bad as Figure 8 shows, the baseline utilization is lower. This means the PE efficiency is effectively improved with the out-of-order position broadcast mechanism and PE overlapping techniques.

TABLE I
PERFORMANCE COMPARISON.

Platform	Speedup	Platform	Speedup
CPU 1-core	959.5	CPU 24-core	158.6
CPU 2-core	546.4	GPU	5.40
CPU 4-core	298.5	baseline	4.32
CPU 8-core	270.0		

D. Energy Validation

Energy convergence validation is performed on an example 20K liquid argon dataset. Figure 12 compares the reference double-precision floating point OpenMM [49] result with the implementation described in this paper (23-bit offset precision scheme in Figure 5(c) and the linear interpolation described in III-B). Over 100k iterations (2 fs per iteration) the relative difference is typically on the order of magnitude of $10^{-4} \sim 10^{-3}$.

E. Ring-Related Performance

The performance with respect to four cell spaces is examined (see Figure 13). We use the same number of PEs as the

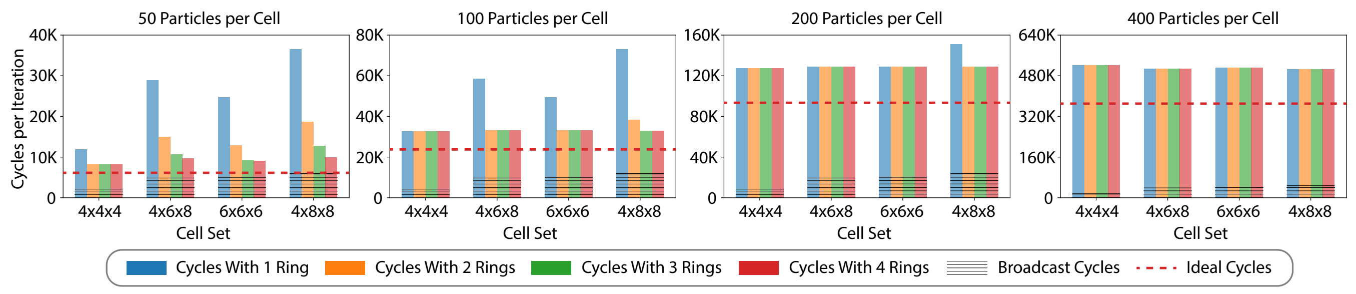

Fig. 13. Performance of representative datasets with up to four force output rings in cycles.

TABLE II

CASE STUDY: CONFIGURATION, HARDWARE COSTS, AND PERFORMANCE. MU = MOTION UPDATE UNIT

Substance	Particles	Cell Space	Particles/Cell	PE #	MU #	Force Rings	ALM	DSP	BRAM	ns/day
Ar	5000	4x5x5	50	200	25	4	59.7%	62.5%	54.6%	10803
Ar	20160	7x6x6	80	252	32	2	81.6%	80.0%	73.1%	2322
Ar	50000	5x5x5	400	250	16	2	68.8%	78.1%	70.4%	196.5
NaCl	5040	3x3x7	80	189	8	6	53.7%	65.6%	50.5%	6131
NaCl	20480	4x4x4	320	192	8	3	47.2%	66.7%	50.8%	466.7
NaCl	51200	4x8x8	200	128	32	2	51.2%	44.4%	45.9%	200.8

number of cells to simplify the comparison. Each cell space has various numbers of randomly generated particles per cell (50, 100, 200, or 400), and up to four force output rings. These configurations cover the cases that the system will operate on.

The red dashed lines show the ideal cycles computed using Equation 7, disregarding the latency and data transfer, and assuming perfect workload balance.

$$C_{ideal} = \frac{27}{2} \times N_{pc}^2 + 8 \times N_{pc} \tag{7}$$

C_{ideal} stands for the ideal number of cycles and N_{pc} is the number of particles per cell. The 1st term is the workload of force evaluation, the 2nd is for motion update. For the 4x4x4 cell set with 50 particles per cell, the time per iteration is only 1.33x ideal. For the scenarios of more particles per cell, 1.40x ideal time per iteration can be achieved.

The shadowed regions indicate the number of cycles required for particle broadcast with a single position input ring if no back-pressure is applied. Back-pressure indicates position data consumption is slower than data transfer; therefore the performance is not affected by particle broadcast. This further demonstrates that only one position input ring is sufficient for common cases.

Force output rings with more than two slots may be needed for peak performance, especially when a cell has a limited number of particles or the number of cells is large. The ring is not congested with 200 particles per cell except for the 4x8x8 case, and with 400 particles for all cases.

We also observe that the 6x6x6 cell set is evaluated with fewer cycles than the 4x6x8 cell set. This phenomenon is related to cell-ring mapping discussed in Section IV-*B*. The average force traveling distance of 6x6x6 on a ring is shorter than 4x6x8.

F. Overall Performance

Performance of FPMD-RL is compared with CPU, GPU, and the baseline design. The reference model is typical with 7x6x6 cells and 80 particles per cell. The index-offset position data format is also applied to the baseline. Together with minor optimizations, the baseline design is able to fit ∼200 PEs. The baseline design only runs at 100 MHz because of the routing problem discussed in Section IV-*B*. The new design operates at 297 MHz and has 252 PEs for this dataset. The results are given in Table I.

Hardware costs and performance for six datasets are listed in Table II, including the dataset (the 2nd) used for comparison in Table I. The substances Ar and NaCl are similar in RL force computation, but NaCl requires 2 additional DSP units per PE for coefficient multiplication. For all the test cases, only one position input ring is used.

We observe that our performance achieves 5.40x speedup compared to GPU, 158.6x speedup compared to 24-core server-level CPU, and 4.32x compared to the prior-art FPGA implementation.

VI. CONCLUSION

In this paper, we present a series of technologies to solve the problems that originate from the severe frequency drop as PE populates with hardware resource optimizations. Although the ring method efficiently raises the frequency, the concurrency problem of the ring is brought in. Therefore, the data caching technique is utilized for higher data locality. However, neighbor particle caching is followed by the PE workload imbalance problem. As a counter measure, the previously used bulk synchronization is replaced by out-of-order position broadcast mechanism and position data dispatching. At last, we implement the yin-yang force caches to solve the concurrency problem in force accumulation. All the efforts above collectively contribute to the performance thus 5x speedup and 4x speedup are achieved over GPU and prior-art FPGA/MD, respectively.

ACKNOWLEDGMENTS

This work was supported, in part, by the NSF through award CCF-1919130; the NIH through awards 1R41GM128533 and R44GM128533; by a grant from Red Hat; and by Xilinx and by Intel both through donated FPGAs, tools, and IP.

References

[1] M. Aminpour, C. Montemagno, and J. A. Tuszynski, "An overview of molecular modeling for drug discovery with specific illustrative examples of applications," *Molecules*, vol. 24, no. 9, p. 1693, 2019.

[2] J. Mortier, C. Rakers, M. Bermudez, M. S. Murgueitio, S. Riniker, and G. Wolber, "The impact of molecular dynamics on drug design: applications for the characterization of ligand–macromolecule complexes," *Drug Discovery Today*, vol. 20, no. 6, pp. 686 – 702, 2015.

[3] M. M. Rahman, T. Saha, K. J. Islam, R. H. Suman, S. Biswas, E. U. Rahat, M. R. Hossen, R. Islam, M. N. Hossain, A. A. Mamun, M. Khan, M. A. Ali, and M. A. Halim, "Virtual screening, molecular dynamics and structure–activity relationship studies to identify potent approved drugs for covid-19 treatment," *Journal of Biomolecular Structure and Dynamics*, vol. 0, no. 0, pp. 1–11, 2020.

[4] D. Case, T. Cheatham III, T. Darden, H. Gohlke, R. Luo, K. Merz, Jr., A. Onufriev, C. Simmerling, B. Wang, and R. Woods, "The Amber biomolecular simulation programs," *Journal Computational Chemistry*, vol. 26, pp. 1668–1688, 2005.

[5] J. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. Skeel, L. Kale, and K. Schulten, "Scalable molecular dynamics with NAMD," *Journal Computational Chemistry*, vol. 26, pp. 1781–1802, 2005.

[6] P. Eastman and V. Pande, "OpenMM: A Hardware-Independent Framework for Molecular Simulations," *Computing in Science and Engineering*, vol. 4, pp. 34–39, 2010.

[7] A. Goetz, M. Williamson, D. Xu, D. Poole, S. L. Grand, and R. Walker, "Routine microsecond molecular dynamics simulations with AMBER on GPUs. 1. Generalized Born," *J. Chem. Theory Comput.*, vol. 8, pp. 1542–1555, 2012.

[8] D. van der Spoel, E. Lindahl, B. Hess, G. Groenhof, A. Mark, and H. Berendsen, "GROMACS: fast, flexible, and free," *Journal Computational Chemistry*, vol. 26, pp. 1701–1718, 2005.

[9] J. Glaser, T. D. Nguyen, J. A. Anderson, P. Lui, F. Spiga, J. A. Millan, D. C. Morse, and S. C. Glotzer, "Strong scaling of general-purpose molecular dynamics simulations on GPUs," *Computer Physics Communications*, vol. 192, pp. 97 – 107, 2015.

[10] Shaw, D.E., et al., "Anton, a special-purpose machine for molecular dynamics simulation," in *International Symposium on Computer Architecture*, 2007, pp. 1–12.

[11] D.E. Shaw et al., "Anton 2: raising the bar for performance and programmability in a special-purpose molecular dynamics supercomputer," in *SC '14: Conference on High Performance Computing Networking, Storage and Analysis*, 2014, pp. 41–53.

[12] N. Azizi, I. Kuon, A. Egier, A. Darabiha, and P. Chow, "Reconfigurable molecular dynamics simulator," in *IEEE Symposium on Field Programmable Custom Computing Machines*, 2004, pp. 197–206.

[13] T. Hamada and N. Nakasato, "Massively parallel processors generator for reconfigurable system," *IEEE Symposium on Field Programmable Custom Computing Machines*, 2005.

[14] Y. Gu, T. VanCourt, and M. Herbordt, "Accelerating molecular dynamics simulations with configurable circuits," in *IEEE Conference on Field Programmable Logic and Applications*, 2005.

[15] V. Kindratenko and D. Pointer, "A case study in porting a production scientific supercomputing application to a reconfigurable computer," in *IEEE Symposium on Field Programmable Custom Computing Machines*, 2006, pp. 13–22.

[16] Y. Gu, T. VanCourt, and M. Herbordt, "Accelerating molecular dynamics simulations with configurable circuits," *IEE Proceedings on Computers and Digital Technology*, vol. 153, no. 3, pp. 189–195, 2006.

[17] Y. Gu, T. VanCourt, and M. Herbordt, "Improved interpolation and system integration for FPGA-based molecular dynamics simulations," in *2006 International Conference on Field Programmable Logic and Applications*, 2006, pp. 21–28.

[18] R. Scrofano, M. Gokhale, F. Trouw, and V. Prasanna, "Accelerating Molecular Dynamics Simulations with Reconfigurable Computers," *IEEE Trans. Parallel and Distributed Systems*, vol. 19, no. 6, pp. 764–778, 2008.

[19] Y. Gu, T. VanCourt, and M. Herbordt, "Explicit design of FPGA-based coprocessors for short-range force computation in molecular dynamics simulations," *Parallel Computing*, vol. 34, no. 4-5, pp. 261–271, 2008.

[20] J. Sheng, B. Humphries, H. Zhang, and M. Herbordt, "Design of 3D FFTs with FPGA Clusters," in *IEEE High Performance Extreme Computing Conference*, 2014.

[21] J. Sheng, C. Yang, and M. Herbordt, "Towards Low-Latency Communication on FPGA Clusters with 3D FFT Case Study," in *International Symposium on Highly Efficient Accelerators and Reconfigurable Technologies*, 2015.

[22] A. Lawande, A. George, and H. Lam, "Novo-G#: a multidimensional torus-based reconfigurable cluster for molecular dynamics," *Concurrency and Computation: Practice and Experience*, vol. 28, no. 8, 2016.

[23] J. Sheng, C. Yang, A. Caulfield, M. Papamichael, and M. Herbordt, "HPC on FPGA Clouds: 3D FFTs and Implications for Molecular Dynamics," in *27th International Conference on Field Programmable Logic and Applications*, 2017.

[24] C. Wu, T. Geng, V. Sachdeva, W. Sherman, and M. Herbordt, "A Communication-Efficient Multi-Chip Design for Range-Limited Molecular Dynamics," in *IEEE High Performance Extreme Computing Conference*, 2020.

[25] L. Stewart, C. Pascoe, B. Sherman, M. Herbordt, and V. Sachdeva, "An OpenCL 3D FFT for Molecular Dynamics distributed across multiple FPGAs," in *ArXiv Preprint arXiv:2009.12617*, 2020.

[26] C. Yang, T. Geng, T. Wang, J. Sheng, C. Lin, V. Sachdeva, W. Sherman, and M. Herbordt, "Molecular Dynamics Range-Limited Force Evaluation Optimized for FPGA," in *2019 IEEE 30th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, 2019, pp. 263–271.

[27] C. Yang, T. Geng, T. Wang, R. Patel, Q. Xiong, A. Sanaullah, C. Lin, V. Sachdeva, W. Sherman, and M. Herbordt, "Fully Integrated FPGA Molecular Dynamics Simulations," in *International Conference for High Performance Computing, Networking, Storage and Analysis*, 2019, pp. 1–31.

[28] M. Chiu and M. Herbordt, "Efficient filtering for molecular dynamics simulations," in *2009 International Conference on Field Programmable Logic and Applications*, 2009.

[29] M. Chiu and M. Herbordt, "Molecular dynamics simulations on high performance reconfigurable computing systems," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 3, no. 4, pp. 1–37, 2010.

[30] M. Chiu and M. Herbordt, "Towards production FPGA-accelerated molecular dynamics: Progress and challenges," in *2010 4th High Performance Reconfigurable Technology and Applications*, 2010.

[31] M. Chiu, M. Khan, and M. Herbordt, "Efficient calculation of pairwise nonbonded forces," in *2011 IEEE 19th Annual International Symposium on Field-Programmable Custom Computing Machines*, 2011.

[32] B. Towles, J. Grossman, B. Greskamp, and D. Shaw, "Unifying on-chip and inter-node switching within the Anton 2 network," in *International Symposium on Computer Architecture*, 2014, pp. 1–12.

[33] E. Ma and L. Tao, "Embeddings among toruses and meshes," Department of Computer and Information Science, University of Pennsylvania, Tech. Rep. Technical Report No. MS-CIS-88-63, 1988.

[34] R. G. Melhem and G. Y. Hwang, "Embedding rectangular grids into square grids with dilation two," *IEEE Transactions on Computers*, vol. 39, no. 12, pp. 1446–1455, 1990.

[35] T.-H. Lai and W. White, "Mapping pyramid algorithms into hypercubes," *Journal of Parallel and Distributed Computing*, vol. 9, no. 1, pp. 42 – 54, 1990.

[36] D. Shaw, "A Fast, Scalable Method for the Parallel Evaluation of Distance-Limited Pairwise Particle Interactions," *Journal Computational Chemistry*, vol. 26, no. 13, pp. 1318–1328, 2005.

[37] J. Haile, *Molecular Dynamics Simulation*. New York, NY: Wiley, 1997.

[38] M. Khan, M. Chiu, and M. Herbordt, "FPGA-Accelerated Molecular Dynamics," in *High Performance Computing Using FPGAs*, K. Benkrid and W. Vanderbauwhede, Eds. Springer Verlag, 2013, pp. 105–135.

[39] Y. Gu and M. Herbordt, "FPGA-based multigrid computations for molecular dynamics simulations," in *15th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, 2007, pp. 117–126.

[40] A. Sanaullah, A. Khoshparvar, and M. Herbordt, "FPGA-Accelerated Particle-Grid Mapping," in *IEEE 24th Annual International Symposium on Field-Programmable Custom Computing Machines*, 2016, pp. 192–195.

[41] Q. Xiong and M. Herbordt, "Bonded Force Computations on FPGAs," in *2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2017, pp. 72–75.

[42] H. Grubmüller, H. Heller, A. Windemuth, and K. Schulten, "Generalized Verlet algorithm for efficient molecular dynamics simulations with long-range interactions," *Molecular Simulation*, vol. 6, no. 1-3, pp. 121–142, 1991.

[43] Z. Yao, J.-S. Wang, G.-R. Liu, and M. Cheng, "Improved neighbor list algorithm in molecular simulations using cell decomposition and data

sorting method," *Computer Physics Communications*, vol. 161, no. 1, pp. 27 – 35, 2004.

[44] W. Brown, P. Wang, S. Plimpton, and A. Tharrington, "Implementing molecular dynamics on hybrid high performance computers–short range forces," *Computer Physics Communications (CPC)*, vol. 182, no. 4, pp. 898–911, 2011.

[45] J. Grossman, B. Towles, B. Greskamp, and D. Shaw, "Filtering, reductions and synchronization in the Anton 2 network," in *Proc. International Parallel and Distributed Processing Symposium*, 2015, pp. 860 – 870.

[46] A. Obeidat, A. Jaradat, B. Hamdan, and H. Abu-Ghazleh, "Effect of cutoff radius,long range interaction and temperature controller on thermodynamic properties of fluids: Methanol as an example," *Physica A: Statistical Mechanics and its Applications*, vol. 496, 2018.

[47] L. Nilsson, "Efficient table lookup without inverse square roots for calculation of pair wise atomic interactions in classical simulations," *Journal of Computational Chemistry*, vol. 30, no. 9, pp. 1490–1498, 2009.

[48] D. Case, I. Ben-Shalom, S. Brozell, D. Cerutti, T. Cheatham *et al.*, "Amber 18," 2018.

[49] P. Eastman, J. Swails, J. D. Chodera, R. T. McGibbon, Y. Zhao, K. A. Beauchamp, L.-P. Wang, A. C. Simmonett, M. P. Harrigan, C. D. Stern, R. P. Wiewiora, B. R. Brooks, and V. S. Pande, "Openmm 7: Rapid development of high performance algorithms for molecular dynamics," *PLOS Computational Biology*, vol. 13, 07 2017.