A Communication-Efficient Multi-Chip Design for Range-Limited Molecular Dynamics

Chunshu Wu
Boston University
Boston, USA
happycwu@bu.edu

Tong Geng
Boston University
Boston, USA
tgeng@bu.edu

Vipin Sachdeva
Silicon Therapeutics
Boston, USA
vipin@silicontx.com

Woody Sherman
Silicon Therapeutics
Boston, USA
woody@silicontx.com

Martin Herbordt
Boston University
Boston, USA
herbordt@bu.edu

***Abstract*—Molecular Dynamics simulation (MD) has been thought a promising FPGA application for many years, especially with clusters of tightly coupled FPGAs where the large-scale, general-purpose, low-latency interconnects provide a communication capability not available with any other COTS computing technology. Parallelization of one part of the MD computation, the 3D FFT, has been studied previously; for likely FPGA cluster sizes, however, the range-limited computation (RL) is more challenging. The motivation here is that the direct replication of the single-chip design suffers from inefficient inter-board bandwidth usage. In particular, although communication in RL is local, likely bandwidth limitations will constrain performance unless great care is taken in design and analysis.**

In the multi-chip scenario, inter-board bandwidth is the critical constraint and the main target of this work. We analyze it with respect to three application restructurings: workload distribution, data forwarding pattern, and data locality. We describe how bandwidth can be balanced by configuring workload distribution and data forwarding paths with respect to the number of on-board transceiver ports. We also show that, by manipulating data locality, the multi-chip design is efficiently migrated from the single-chip design, and the total bandwidth required can be configured to satisfy the bandwidth limit.

I. Introduction

Molecular Dynamics (MD) is a critical HPC application, e.g. playing a key role in the pharmaceutical industry for drug discovery. There are many production MD codes and it can be performed on a variety of platforms, with GPUs being popular [4], [8], [12], [18], [26]. But for the small molecule simulations ($<$, say, 100K particles) that dominate many critical bioapplications, GPU MD has so far failed to scale beyond even two devices. To address long timescales, therefore, ASIC-based MD simulators have been developed [7], [22] and have achieved orders-of-magnitude speed-ups. But while of great value, these systems none-the-less have limited availability. An alternative is the FPGA: for MD they have similar performance per-device to GPUs [27]. But the essential motivation of using FPGA-centric clusters (FCCs) for MD is that they couple this per-device performance with ASIC-quality connectivity.

It is well-known that part of MD's scalability problem is the global communication performed during (most instantiations of) the long-range force computation (LR). Since classical MD is iteration driven, this results in the strong scaling problem. To achieve longer timescales, more processors are needed to reduce the iteration time, but these additional processors lengthen the communication time until it dominates. ASIC- and FPGA-centric clusters effectively deal with this problem with dedicated chip-to-chip communication that can transfer data in $<$ 100ns, at very high bandwidth, and directly application-layer to application layer with minimal additional overhead [24], [29].

But while mapping LR onto FCCs has been studied [17], [23], [24], RL has not. RL consists of 90% of the FLOPs in MD and requires more data to be transferred [14]. But the fact that RL communication is local has perhaps made the problem seem straightforward, at least for small clusters [9]. We have found quite the opposite to be true.

The evolution from the existing a single-chip RL design to multi-chip brings multiple scalability problems. Even though the transceiver bandwidth on commercial FPGA boards is considerable, we find it still to be insufficient; but only due to poor data locality in typical MD layouts. If we inherit the data transmission design directly from the single-chip version, transceiver bandwidth on the order of Tbps is required or nearly all scaling is lost. To scale this up to a multi-chip design, we propose a neighbor data caching method to exploit data locality for improved data reuse, with negligible hardware overhead.

Another problem is the irregularity of the inter-board communication pattern, which leads to highly imbalanced loads on the different transceiver ports. To optimize bandwidth utilization, we take advantage of the topology of the underlying MD-RL particle interactions. First, particles only interacts with other particles inside a halo; then, the half-shell method is used to take advantage of Newton's 3rd law [5], [21]. For this specific topology, we investigate various workload distributions (slab, pillar, and block) and their data transfer patterns. In order to relieve the communication imbalance, we a propose routing configuration technique, which is able to improve the inter-chip bandwidth utilization for all three workload distributions.

The major contributions of this work are as follows. We propose

- a communication-efficient multi-chip MD-RL design;
- a neighbor data caching technique that improves neighbor particle data reuse for reduced data movements; and
- a routing configuration technique for improved inter-chip bandwidth utilization.

- Also, experimental results show the performance of the proposed design provides at least 10× speedup comparing with the baseline design.

II. Background

A. MD Review

MD is the iterative application of Newtonian dynamics to ensembles of charged particles. Each iteration consists of two phases: computing the forces among the particles and applying the forces to obtain new motion/displacement values for each particle. The second phase is $O(n)$ and requires no interaction so we ignore it here. The force computation has several parts, the most important of which are Coulombic, van der Waals (LJ), and bonded. As with the motion update, the bonded force is $O(n)$ and local so is not considered further here.

The starting point in improving performance is to reduce the $O(n^2)$ complexity of the non-bonded force computation. It is convenient to define a cut-off radius R_c around each particle and partition the Coulombic and LJ terms into non-overlapping components: one for inside R_c (range limited or RL), the other for outside (long range or LR). This partition conveniently allows different methods to be applied: for LR $O(n \log n)$ transforms (studied elsewhere [23], [29]), for RL $O(n)$ cell lists plus filtering as described below.

B. Range Limited Force

RL force describes the interaction between electrically neutral atoms or molecules, and often refers to the force introduced by Lennard-Jones potential and the rapidly decaying component from Ewald summation for Coulombic force. The resultant force on particle i is

$$\mathbf{F}^i_{LJ} = \sum_{j \neq i} \frac{\epsilon(t_i,t_j)}{\sigma(t_i,t_j)^2}(48(\frac{\sigma(t_i,t_j)}{|r_{ji}|})^{14} - 24(\frac{\sigma(t_i,t_j)}{|r_{ji}|})^{8})\hat{\mathbf{r}_{ji}} \tag{1}$$

where t_i and t_j are particle types, σ and ϵ are parameters which depend on particle type, and $\hat{\mathbf{r}_{ji}}$ is a unit vector pointing from particle j to particle i.

In RL force computations, cutoff radius R_c is defined as the minimum distance between which two particles' interaction is beyond the current data precision. Therefore, a particle only pairs with particles within the cutoff radius, and the computational complexity is $O(N)$, with a multiplicative constant of 100-300, instead of all-to-all $O(N^2)$.

C. Modeling

Once the force is calculated, it is translated into the increment in velocity and displacement based on the formulae below. By convention, the force is a constant during a simulation time interval (approximately 10^{-15} seconds).

$$\mathbf{v}(t+\Delta t) = \mathbf{v}(t) + \mathbf{a}(t)\Delta t \tag{2}$$

$$\mathbf{r}(t+\Delta t) = \mathbf{r}(t) + \mathbf{v}(t+\frac{\Delta t}{2})\Delta t \tag{3}$$

We only consider particles that form a micro-canonical (NVE) ensemble. Therefore, it is convenient to divide the overall simulation space into small static cells. The simulation space is cubic and obeys the periodic boundary condition. Since the side length of a small cell can be arbitrarily determined, it is reasonable to set it as R_c to minimize the number of cells involved in interactions.

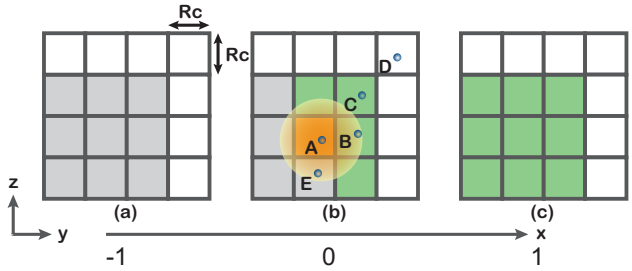

Fig. 1. 3-d particle interaction pattern in a 2-d view. (a), (b) and (c) are three layers stacked on x direction. The halo represent the interaction range of particle A. Orange: the home cell of particle A. Grey: Ignorable regions based on Newton's 3rd law. Green: Cells that may contain valid neighbor particles. White: Cells out of range.

Figure 1 shows the particle interaction pattern of three cell layers along the x axis. The reference particle A interacts with particles in at most 27 cells, including its home cell (orange). Thanks to Newton's 3rd law, only half of the 27 cells (13 green neighbor cells and 1 orange home cell) need to be accessed from a reference particle's perspective. Typical particle relations are illustrated in (b). Green cells are searched for particles that are within A's cutoff radius through particle filtering. B and A form a valid pair while C-A pair is invalid. Particle D and E are not taken into account when A behaves as the reference particle, even if E is inside the halo. In CPU implementations each particle has its own neighbor list of valid particles, which is updated periodically. In ASICs and FPGAs this is impractical. Instead particles in the home and valid neighboring cells are filtered on-the-fly by distance [5].

D. Parallel RL

To the first approximation parallel RL is a version of the halo problem. Each node N is responsible for some number of home cells. Neighbor cells may be on N or on a neighboring node. These *halo* cells must be exchanged. There are certain complications, however. First, the optimization for Newton's 3rd Law makes the halo asymmetric. Second, the number of particles within the cut-off in the neighboring cells varies drastically. Third, the halo is comparatively heavy or *thick*. In typical stencil problems it may be a single row or column of a large 2D array; here it is some number of complete cells.

We make one significant assumption: the number of devices is less than or equal to the number of cells. The remaining case is not likely to be practical for FCCs any time soon and would use other methods than those presented here.

III. Multi-chip baseline design

In this section, we first briefly introduce the architecture of a prior art single-chip design [28]. Second, we propose a multi-chip direct extension of this design which we refer to as the *baseline*; in Section IV we compare it with our optimized

multi-chip designs. Finally, we present the communication bottleneck of the baseline design and discuss the causes for it.

A. Prior-art Single-chip Design

The prior art single-chip MD architecture is illustrated in Figure 2. The particles in a simulation space are partitioned into c cells. The force, position, and velocity information of the particles in each cell are stored in 3 caches (force, position, and velocity), respectively. Force Evaluation Units (FEU) are in charge of force calculation. Each FEU evaluates the force applied on the particles in a unique cell. To evaluate forces, FEUs first read position data from position caches and then identify valid particle pairs. Subsequently, the partial forces between valid particle pairs are computed and sent to proper accumulation units through an arbitration network and so complete the force computation. Newly accumulated force results update the force caches until the complete forces are calculated.

There are two types of data movement which dominate inter-chip communication in multi-chip design. In *position data forwarding* neighbor particle position data are broadcast to FEUs that work on the related home cells for data reuse. Each piece of position data is broadcast to 14 force pipelines based on the half-shell topology. Upon evaluation, neighbor particle position data are traversed for a single reference particle; when this is finished, the evaluation to the next reference particle begins.

In *force write back*, since each FEU receives positions of particles from multiple cells, the force writing requests from different FEUs may target the same force cache. To resolve the conflict, each FEU is equipped with a multi-bank force writeback buffer. Each bank stores the partial force results of a unique destination cell. The forces are then selected by force writeback arbitrators and sent to the accumulation unit of the destination.

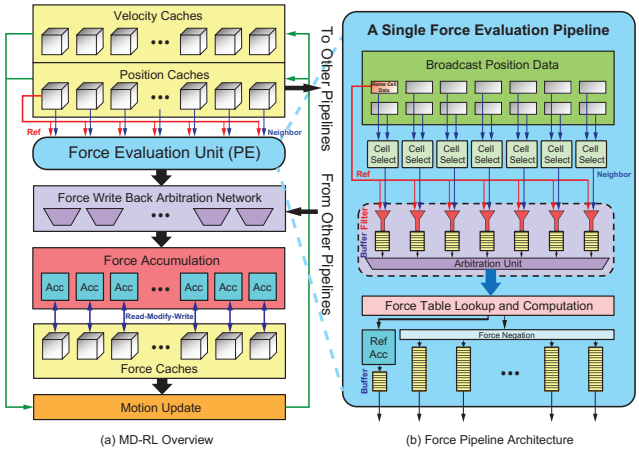

Fig. 2. Architecture of the single-chip design. (a): The general architecture of the iterative force evaluation and motion update. (b): Architecture of force evaluation pipelines

B. Baseline Multi-chip Design

We extend the prior art single-chip design introduced in Section III-A to multi-chip with obvious augmentations for the support of inter-chip data exchange. The position data forwarding and force write back both require inter-board communication. For position data forwarding, in the baseline design, the broadcast of neighbor particle position data can be across FPGA boards. Similarly, the destination cells of force writeback can be mapped to other FPGAs as well.

Inter-board force data transactions can be very frequent, resulting in high data production versus low data consumption. It is common for the baseline design that the data can be sent less frequently whenever the bandwidth is not sufficient, i.e. when the communication-computation ratio is > 1. To compensate for the long communication latency, a scheduling mechanism is adopted in the baseline design. The evaluated partial forces, which are directly computed from single particle pairs, are saved in an additional local buffer to schedule the inter-board transmission (see the scheduling buffer in Figure 3).

Also, the partial forces are only accumulated after being selected by the arbitration network, thus the select signal from remote arbitration network brings high latency. Some modifications are made from the single-chip design to target this problem, as Figure 3 shows. Instead of saving the returning neighbor forces in output buffers locally, the buffers are located in destination nodes. The straightforward benefit is that there is no need to send the force writeback arbitration result between nodes; thus the $\sim$100ns per transaction is spared.

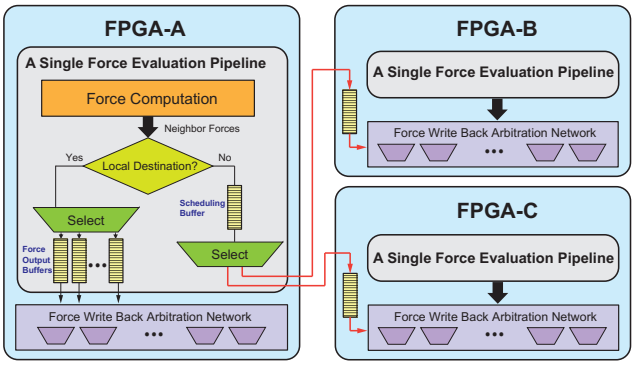

Fig. 3. Force output buffers are re-located to prevent high latency caused by data dependency. The scheduling buffer is added in case the bandwidth bottleneck is met.

C. Communication Bottleneck of Baseline

We note that the neighbor particle position data are discarded after being used on a single reference particle. They are therefore repeatedly sent from neighboring cells to home cells. This low data reuse leads to tremendous bandwidth demand. The mere data transmission between two cells costs $\sim$10Gbps bandwidth at 100MHz operating frequency, let alone there are $\sim$100 cell-to-cell communications between nodes, which sums up to 1Tbps requirement.

On the other hand, not all ports on FPGA boards are completely utilized, and the imbalanced port utilization may eventually cause severe performance loss. For example, a node with 6 ports holds 8x6x4 cells, where the 6 ports are connected to 6 neighbor nodes on x, y, and z directions. Assuming

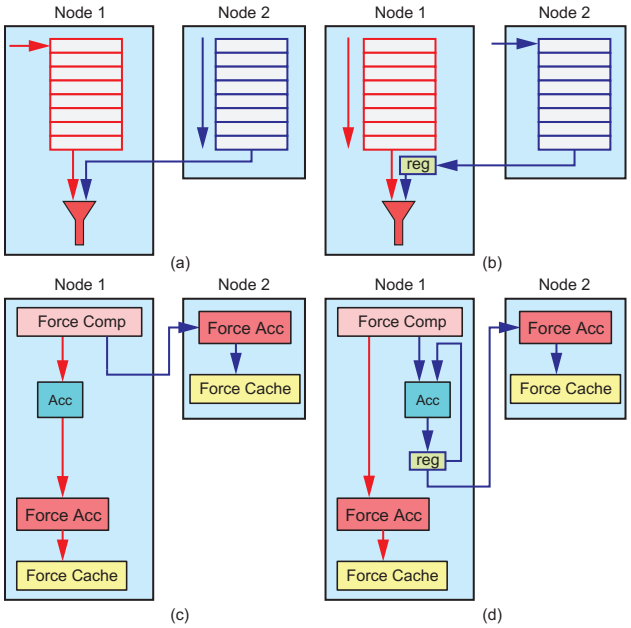

Fig. 4. Position reading and force writing architecture (only 2 position caches in 2 nodes are shown for demonstration). (a): Baseline position reading. (b): New position reading. (c): Baseline force writing. (d): New force writing. Red arrows: Reference data paths. Blue arrows: Neighbor data paths.

that the port with the highest bandwidth demand is fully utilized (U_z=100%), a common scenario is that U_x=78%, and U_y=59%, leading to an 80% overall port utilization, meaning 20% of data transmission opportunity is lost compared to perfect utilization. This situation may get worse for different number of ports on board. As a result, a way to balance the workload among different directions is important.

IV. Optimized Multi-chip Design

The optimized described in this section aims to relieve the communication bottleneck of the baseline. Two techniques, Neighbor Data Caching and routing configuration, are discussed to solve the two problems presented in section III-C respectively. Motion update is briefly discussed at the end of this section, as it also introduces inter-chip data exchange.

A. Neighbor Data Caching

A current high-end FPGA board may support ∼100s Gbs inter-chip data transmission rate. However, a rate of Tbit/s is required in our baseline design, i.e. neighbor particles are broadcast non-stop and the force is written back every cycle. Our solution is to trade the reference particles' temporal locality for the neighbor particles' temporal locality.

1) Neighbor Particle Position Caching: For the data reusability optimization, reference particles are traversed for incoming neighbor particles instead, as Figure 4(b) illustrates, where the neighbor particles are cached in registers. The position reading architectures are compared in Figures 4(a) and (b). In the new design, position data are only transmitted once a few cycles.

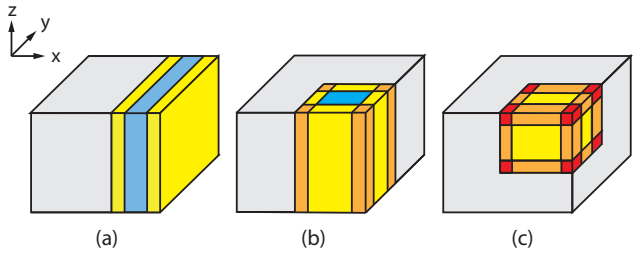

Fig. 5. Workload distribution patterns in a cuboid simulation space. Grey: The cells evaluated on other nodes. Blue: Cells in which particle pairs are formed without external memory involved. Yellow: Cells in which data from a neighboring FPGA are needed or delivered (1 hop). Orange: 2 hops. Red: 3 hops. (a): Slab distribution. (b): Pillar. (c): Block.

2) Neighbor Particle Force Caching: In the baseline design, reference forces (partial forces applied on a reference particle) are accumulated before read-modify-write. Now that the reference particle to be evaluated changes every cycle, the reference forces are no longer accumulated beforehand. That is, the accumulator on the red path in Figure 4(c) is removed in (d). Instead, the partial reference forces are directly accumulated in the destination force caches without conflict and with throughput guaranteed. Therefore the force arbitration network in Figure 2(a) is not needed.

On the other hand, although the temporal locality of the reference particles is lost, it is gained by the neighbor forces. In Figure 4(d), the partial forces for neighbor particles are registered and accumulated *in situ*. All neighbor forces share the same accumulator since only one neighbor force is produced in a FEU per cycle. The accumulated result is stored back in registers. Once all neighbor partial forces are accumulated, they are sent back to their original nodes and accumulated in corresponding force caches.

B. Routing Configuration

Various data/workload distributions show different data transfer patterns. We discuss 3 basic scenarios: slab, pillar, and block. The data transfer patterns of the three distributions can be modified to different degrees separately, which may cause workload imbalance. To address this problem, a software approach proposed to automatically configure routing paths based on the cell distribution and the number of ports.

1) Workload Distribution: The workload is spatially distributed so that device memory only contains data from a group of cells in Euclidean space. In this section, three typical distribution patterns are studied and the theoretical total bandwidth usage is given.

Figure 5 illustrates slab, pillar and block distributions. For small simulation spaces, slab and pillar are the first picks because of their advantage in local data access, whereas the block holds the highest generality. If a slab of cells is assigned to a node, the home FPGA only communicates with two FPGAs next to it. For the pillar distributions, cells on the sides require or receive data that arrives after up to two hops. For block distributions, corner cell communications are involved and requires data to be transmitted three hops away.

TABLE I
DATA CLASSIFICATION (BANDWIDTH UNIT: $w_d f$)

Pillar				Block					
1-hop		2-hop		1-hop		2-hop		3-hop	
Direction	Bandwidth	Direction	Bandwidth	Direction	Bandwidth	Direction	Bandwidth	Direction	Bandwidth
x_+	$l_y l_z$	$x_+ y_+$	l_z	x_+	$l_y l_z$	$x_+ y_\pm$	l_z	$x_+ y_+ z_+$	1
y_+	$l_x l_z$	$x_+ y_-$	l_z	$y_\pm$	$l_x l_z$	$x_+ z_\pm$	l_y	$x_+ y_+ z_-$	1
y_-	$l_x l_z$			$z_\pm$	$l_x l_y$	$y_+ z_\pm$	l_x	$x_+ y_- z_+$	1
						$y_- z_\pm$	l_x 1	$x_+ y_- z_-$	1

For the three distributions the total bandwidth is

$$B_{slab} = 2l_y^s l_z^s w_d f \tag{4}$$

$$B_{pillar} = 2(2l_x^p l_z^p + l_y^p l_z^p + 2l_z^p) w_d f \tag{5}$$

$$B_{block} = 2(2l_x^b l_y^b + 2l_x^b l_z^b + l_y^b l_z^b + 4l_x^b + 2l_y^b + 2l_z^b + 2) w_d f \tag{6}$$

where l_x, l_y, and l_z are the numbers of cells covered by a node on x, y and z dimensions, w_d is the width of a piece of data to be sent from a cell to another, and f is the data transmission frequency of a position cache or a force cache. The factor of 2 signifies that this includes both position send and force receive.

For example, let 256 cells be mapped to a node. Then a slab is configured as 1×16×16, a pillar as 4×16×4, and a block as 4×8×8; the pillar prevails in total bandwidth. It can be problematic, however, for a board with 6 transceiver ports to send data to 4 directions, but it becomes an advantage for boards with 4 transceiver ports. Users can decide which distribution best suits their hardware and problem.

2) Data Path Analysis: In this section, we explore how data are transmitted across nodes and how the data forwarding paths can be configured. The data path for the slab pattern is trivial, since an FPGA only sends data to the node at $x = 1$ and receives data from the node at $x = -1$ (see Figure 1). The data path can vary in the pillar and block cases. It is also shown in Figure 1 that for an FPGA node located at $x = 0$, data from nodes at $x = -1$ are not needed. This indicates that a node sends data to, or receives data from, 5 neighboring nodes for the pillar layout and 17 neighboring nodes for the block layout.

The I/O bandwidth requirement of an intermediate FPGA node on a data forwarding path is inevitably increased. Figure 6 shows examples of data transfer patterns corresponding to pillar and block distributions. A piece of 2-hop data can be obtained from 1-hop data, and 3-hop data can be obtained from 2-hop data for data reuse. The bandwidth along different directions can be re-configured, see the different data forwarding paths in (a) and (b) for pillar, and (c) and (d) for block.

3) Bandwidth Balancing: Since the number of cells on a single node is limited, the amount of data being forwarded two hops or three hops away is considerable compared with the data forwarded directly (one hop). As previously shown, the 2-hop and 3-hop data forwarding paths can be adjusted so that the data flow on different paths. This property can be exploited to achieve balanced communication workload in all directions and, ultimately, balanced data flow through all ports.

We now assume that all nodes use the same data forwarding pattern so that the data flow in opposite directions is symmetric, i.e., the bandwidths on direction α_+ and α_- are identical. The bandwidth of the block pattern in a certain direction is then

$$B_\alpha = \sum_{faces} l_\beta l_\gamma w_d f + \sum_{edges\ \beta} l_\gamma w_d f + \sum_{edges\ \gamma} l_\beta w_d f + \sum_{vertices} w_d f \tag{7}$$

For 2-hop data transmissions along the α direction, data on the sides with cell dimension l_β and l_γ can only be sent via neighbor nodes on the γ and β directions separately first, then forwarded along the α direction. This behavior is abstracted into the 2nd and the 3rd term. To achieve balance, the bandwidth bottleneck must be minimized as shown here

$$B_{bottleneck} = max(\{B_{port}^1, ..., B_{port}^N\}) \tag{8}$$

where B_{port}^i is the bandwidth demand for the ith port, based on routing configurations. The bandwidth is naturally balanced for a slab since data transmission workload can be uniformly distributed. However, slab and pillar distributions have scalability issues: they may be overwhelmed by an oversized simulation space. Based on the distributions in Figure 5, position/force data to be transmitted are classified spatially in Table I.

1-hop data paths are not configurable since the destinations are just next to the origin. In the 2-hop and 3-hop cases, however, the data widths can be re-configured to any corresponding direction. For example, for the first data width entry l_z in Pillar, 2-hop can be included in total x_+ bandwidth or y_+ bandwidth. Entries under Block have more freedom for configuration since more path combinations are available. In the following section, we demonstrate that up to 30% more data can be transmitted compared to average.

C. Motion Update Conflict

The number of cycles for motion update is exactly the number of particles N, while for force evaluation, the number is $\frac{\alpha N}{N_p}$, where α is the average number of neighbor particles that can form valid pairs with a reference particle. Based on our assumption that $N_p < N_c$ or $N_p \approx N_c$,

$$\frac{\alpha}{N_p} \gg 1 \tag{9}$$

is obtained. Amdahl's law suggests that it is reasonable to deploy only one motion update module in a single node.

Throughout the motion update process, a particles can migrate to another cell in another node. Rarely but possibly,

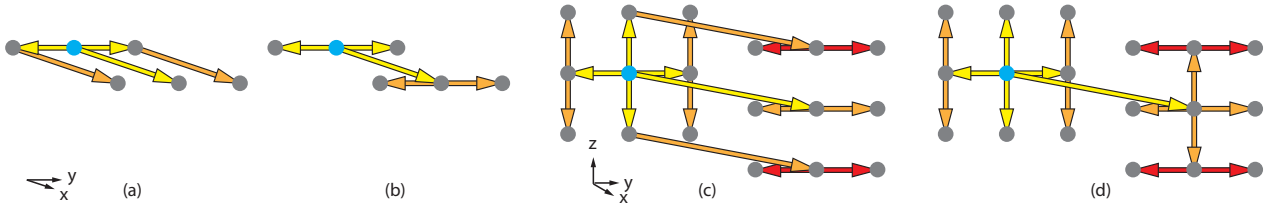

Fig. 6. Reconfigurable force data forwarding pattern examples for pillar and block distributions. (a) and (b): Pillar. (c) and (d): Block. Yellow, Orange and red represent the 1st hop, 2nd hop and 3rd hop separately. Blue node: Home FPGA node. Grey node: Neighbor FPGA node.

the remote update can cause a conflict with local motion update when they are being written to a same cache. The solution is to pause the local update and let the remote update finish first. Given the fact that all cells have a similar number of particles, the motion update units are almost synchronized and the conflict between two remote updates is nearly impossible. If more motion update units are added for more parallelism, the conflict becomes a problem and extra input buffers can be deployed to resolve the conflict.

V. Evaluation

In this section, we first evaluate independently the benefits of routing configuration and neighbor data caching. We then evaluate the overall speedup of the proposed communication-efficient MD design with respect to the baseline. The metric is execution time per iteration.

A. Evaluation of Routing Configuration

The discussions above show that bandwidth balancing is affected by various variables and can be optimized through different ways. In this section, we evaluate bandwidth balancing by taking into consideration the number of transceiver ports per board, the workload distribution pattern, and the cell distribution pattern.

Currently FPGA boards with high-bandwidth transceivers are common. For example, 2x, 4x and 6x QSFP28 transceiver ports are available on various of boards; some even have 12x ports. The bandwidth balancing mechanism can be adopted for any workload partitioning of simulation spaces, but we only demonstrate some representative cases here. To justify the effectiveness of the configurations discussed, transceiver port utilization is defined:

$$U_{ports} = \frac{B_{ideal}}{B_{bottleneck} \times N} \tag{10}$$

where B_{ideal} is the overall bandwidth budget (equation (4), (5), and (6)), and N is the number of ports. For convenience, we assume that at least one of the ports is fully utilized, otherwise the data transmission frequency can be increased to satisfy this condition. In other words, high port utilization leads to a high data transmission frequency bottleneck.

Figure 7 illustrates port utilization cases with conspicuous dependency on the number of ports and the distribution of cells on chip, and reveals the problem that ports may stay idle for a significant amount of time if the routing path is randomly configured. We use as a baseline the average of all routing paths. In Figure 7(a), the highest utilizations can be

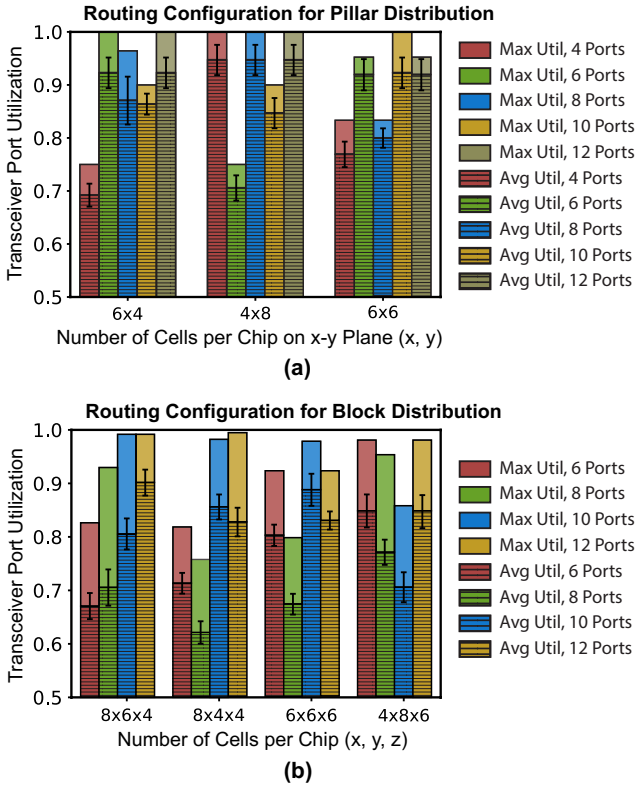

Fig. 7. The port utilization due to the best routing configuration (Max Util) and the average utilization (Avg Util) of all routing configurations for 3 typical pillar (a) and 4 typical block (b) distribution cases. Such cases indicate that the utilization caps can be heavily affected by distribution patterns for various numbers of ports. Error bar: Standard deviation

achieved when the $6{\times}4{\times}N_z$ cases are higher than 0.9, except for when there are only 4 ports (N_z is the number of cells along z direction, which is redundant for pillar evaluation). However, the utilization of 4 ports surpasses 6 ports by 25% for the $4{\times}8$ distribution, indicating that $4{\times}8$ is one of the optimal choices when only 4 ports are available. The last case shows that the $6{\times}6$ distribution can be fully exploited for 10-port boards, but not 4-port or 8-port boards. All those boards have significant weaknesses except for 12-port boards, for they lack sufficient configurability. On the other hand, the highest utilization possible exceeds the average by 3% to 9%, meaning that the data transmission efficiency is boosted by these amounts for free.

Similar phenomena occur in block distributions, as Figure 7(b) shows. 6-port, 6 and 8-port, 8-port, and 10-port boards behave badly in those cases. It is worth noting that the amount of data transmission can be increased by 10% to 30%, for

TABLE II
THROUGHPUT OF A 2X2X2 FPGA CLUSTER. THE DESIGN RUNS AT 350MHZ FREQUENCY, WITH 50 PARTICLES IN EACH CELL. PARTICLE: LIQUID ARGON. CUTOFF RADIUS: 8.5Å. NUMBER OF PORTS: 6. BANDWIDTH OF EACH PORT: 100GBPS. DATA SIZE PER PACKET: 120 BITS

Cell distribution	8x6x4		8x4x4		6x6x6		4x8x6	
	baseline	new	baseline	new	baseline	new	baseline	new
Force evaluation time (μs)	3784	76	2621	76	3051	76	2586	76
Overall time per iteration (μs)	3854	147	2573	123	3130	156	2656	147
Max frequency supported (MHz)	7.0	434	9.2	527	8.7	502	10.3	595
Communication/Computation Ratio	26.2	0.42	20.9	0.41	20.1	0.34	18.1	0.30

multiple possible routing paths can be taken, which gives more importance to the route configuration.

B. Evaluation of Neighbor Data Caching

The efficiency of neighbor data caching depends on the number of particles per cells. Since the reference particles are traversed as in Figure 4(b), neighbor position data are cached for the same number of cycles as the number of reference particles. Therefore, the bandwidth demand reciprocally depends on the number of particles per cell. Assuming that each particle participates in at least one valid pair,

$$B_{new} = \frac{B_{baseline}}{N_{max}} \tag{11}$$

where N_{max} is the largest number of particles of a cell among all cells, as all position broadcasts must be synchronized.

The neighbor data caching mechanism also brings the opportunity for routing configurations. The same amount of force data is sent in return for each piece of position data received. Therefore, the symmetry of position reading and force writing is reserved, which makes the routing configuration work for both position and force paths; i.e. the bandwidth requirement is the same for both reading and writing. Otherwise, uncertainty occurs in force packets' return, since destinations of the packets are random.

C. Overall Performance Evaluation

To avoid too much overhead in synchronization, a 2x2x2 FPGA cluster is chosen to demonstrate the overall speedup due to the optimizations. Each FPGA has six transceiver ports (QSFP28). Table II illustrates the amount of time used in force evaluation per iteration, and the time needed for an entire iteration for four different cell distributions (block). For further improvements, the maximum supported frequencies are also listed, i.e. the highest frequency at which the communication latency can be entirely hidden by computation. The communication-computation ratio is listed as well for more straightforward comparison. In the baseline design, the communication latency exceeds the computation latency by a factor of 10. Also, with a reasonable operating frequency, the communication latency can be hidden completely.

For simplicity, the number of FEUs per node is set to be the same as the number of cells, and only one motion update unit is used. Therefore, for our optimized design, the force evaluation time remains unchanged. For all four cell distributions, the performance roofline of the baseline design is data transmission, while the roofline of the optimized design is the operating frequency (350MHz), which is below the maximum frequency supported. It is evident that by solving the bandwidth problem alone, the overall performance can be increased by a factor of 10, compared with the baseline.

VI. RELATED WORK

MD on FPGAs has been studied for many years [1], [2], [6], [11], [15], [20]. The first complete MD systems on FPGAs accelerated the Range-limited (RL) force computations and used a CPU for the rest [10], [15], [19]. Khan accelerated NAMD with four FPGAs, but they were used as independent accelerators [13].

When we broaden prior work to ASICs, a variety of spatial decomposition methods have been studied more than a decade ago by D.E. Shaw Research [21], where zonal methods [3] are discussed in detail. Among the zonal methods, the neutral-territory method and, specifically, tower-plate method, is used in their first Anton system [16] because of its low import volume. In this study, however, since a large amount of resources are available on modern FPGA chips and most data can be consumed locally, we chose half-shell decomposition so the data locality is naturally preserved.

Load balancing was further refined during the development of Anton 2 [25] where data forwarding routes can be adjusted at runtime. In this study we have chosen not to use in-network reduction since an FPGA only exchanges data with nearby neighbors. As a result, the force and position packets can be scheduled and the data forwarding route can be determined offline.

VII. CONCLUSION

This work demonstrates data forwarding and communication patterns of the slab, pillar, and block distributions for a range-limited MD system. Data in transmission are classified into 1-hop, 2-hop and 3-hop cases. We exploit the symmetry in data flow and illustrate that 2-hop and 3-hop data paths can be re-configured to balance the I/O bandwidth of FPGA nodes. We also conclude that by configuring the routing path and the cell distribution, the data transmission efficiency (or transmission frequency bottleneck) can be 30% higher than average. Apart from bandwidth balancing, the total bandwidth cost is much lowered with neighbor particle localization. By taking advantage of both methods, the overall performance can be potentially increased by a factor of 10 compared to the baseline design with the communication latency completely hidden.

REFERENCES

[1] S. Alam, P. Agarwal, M. Smith, J. Vetter, and D. Caliga, "Using FPGA devices to accelerate biomolecular simulations," *Computer*, vol. 40, no. 3, pp. 66–73, 2007.

[2] N. Azizi, I. Kuon, A. Egier, A. Darabiha, and P. Chow, "Reconfigurable molecular dynamics simulator," pp. 197–206, 2004.

[3] K. Bowers, R. Dror, and D. Shaw, "Zonal methods for the parallel execution of range-limited n-body simulations," *Journal of Computational Physics (JCP)*, vol. 221, no. 1, pp. 303–329, 2007.

[4] D. Case, I. Ben-Shalom, S. Brozell, D. Cerutti, T. Cheatham *et al.*, "Amber 18," University of California, Tech. Rep., 2018.

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

[6] 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 preprint arXiv:1611.04474*, 2016.

[7] D.E. Shaw et al., "Anton 2: Raising the Bar for Performance and Programmability in a Special-Purpose Molecular Dynamics Supercomputer," in *Proceedings of ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2014, pp. 41–53.

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

[9] J. Grossman, B. Towles, B. Greskamp, and D. Shaw, "Filtering, Reductions and Synchronization in the Anton 2 Network," in *Proceedings of International Parallel and Distributed Processing Symposium (IPDPS)*, 2015, pp. 860 – 870.

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

[11] ——, "Accelerating molecular dynamics simulations with configurable circuits," *IEE Proceedings – Computers and Digital Technology*, vol. 153, no. 3, pp. 189–195, 2006.

[12] D. Hardy, "NAMD-lite," http//www.ks.uiuc.edu/ Development/MDTools/namdlite/, University of Illinois at Urbana-Champaign, 2007.

[13] 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.

[14] M. Khan and M. Herbordt, "Communication requirements for FPGA-centric molecular dynamics," in *Symposium on Application Accelerators for High Performance Computing*, 2012.

[15] V. Kindratenko and D. Pointer, "A case study in porting a production scientific supercomputing application to a reconfigurable computer," pp. 13–22, 2006.

[16] R. Larson, J. Salmon, M. Deneroff, C. Young, J. Grossman, Y. Shan, J. Klepseis, and D. Shaw, "High-througput pairwise point interactions in Anton, a specialized machine for molecular dynamics simulation," in *Proc. High Performance Computer Architecture*, 2008, pp. 331–342.

[17] A. Lawande, A. George, and H. Lam, "Novo-G#: a multidimensional torus-based reconfigurable cluster for molecular dynamics," *Concurrency and Computation: Practice and Experience*, 2016.

[18] 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 of Computational Chemistry*, vol. 26, pp. 1781–1802, 2005.

[19] 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.

[20] R. Scrofano and V. Prasanna, "Preliminary investigation of advanced electrostatics in molecular dynamics on reconfigurable computers," 2006.

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

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

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

[24] J. Sheng, C. Yang, A. Caulfield, M. Papamichael, and M. Herbordt, "HPC on FPGA Clouds: 3D FFTs and Implications for Molecular Dynamics," in *Proceedings of IEEE Conference on Field Programmable Logic and Applications (FPL)*, 2017.

[25] 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 (ISCA)*, 2014, pp. 1–12.

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

[27] C. Yang, T. Geng, T. Wang, R. Patel, Q. Xiong, A. Sanaullah, C. Lin, V. Sachdeva, W. Sherman, and M. Herbordt, "FPGA Molecular Dynamics Simulations," in *Proceedings of ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2019.

[28] 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 *Proceedings of International Conference on Application Specific Systems, Architectures, and Processors (ASAP)*, 2019.

[29] C. Young, J. Bank, R. Dror, J. Grossman, J. Salmon, and D. Shaw, "A 32x32x32, spatially distributed 3D FFT in four microseconds on Anton," in *Proceedings of ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2009, pp. 1–11.