Molecular Dynamics Range-Limited Force Evaluation Optimized for FPGA

Chen Yang[§], Tong Geng[§], Tianqi Wang[*§], Charles Lin[‡], Jiayi Sheng[†], Vipin Sachdeva[‡], Woody Sherman[‡], **Martin Herbordt**[§]

[§]Department of Electrical and Computer Engineering, Boston University, Boston, MA

[*]Department of Physics, University of Science and Technology of China, China

[‡]Silicon Therapeutics, Boston, MA

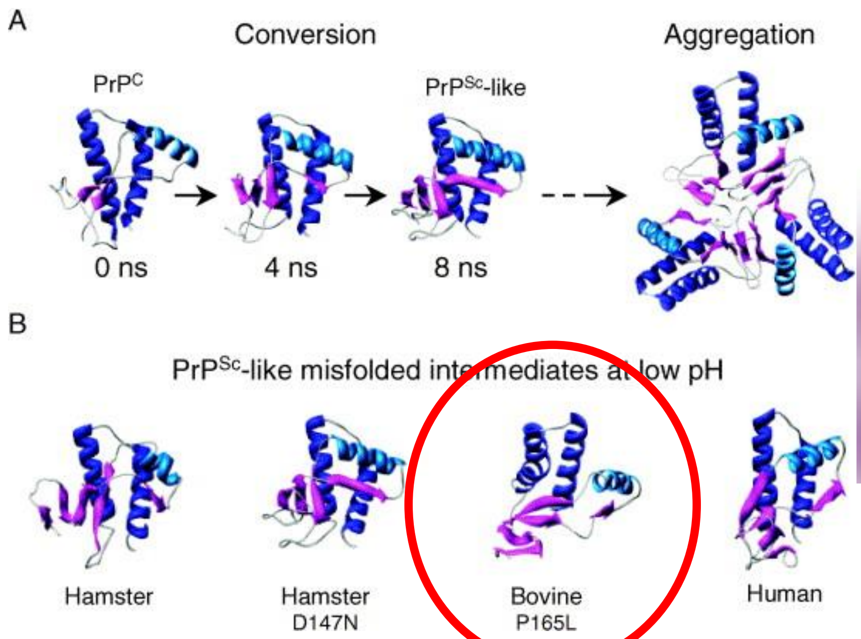
[†]Falcon Computing Solutions, Inc., Los Angeles, CA

07/17/2019

BOSTON UNIVERSITY

Why Molecular Dynamics Simulation is so important …

- Core of Computational Chemistry
- MD is a large fraction of supercomputing cycles (~25%)
- Central to Computational Biology, with applications to
  - → Drug design
  - → Understanding disease processes

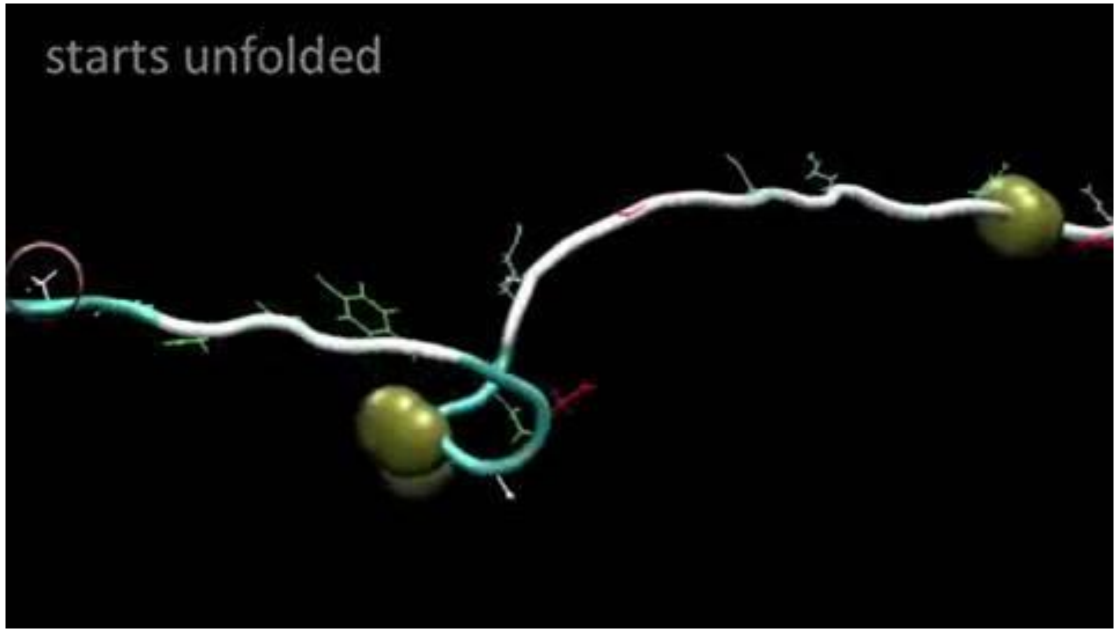


24/04

m
ation of
be the
"mad

covered
with simulation!

Why Acceleration of Molecular Dynamics is so important …



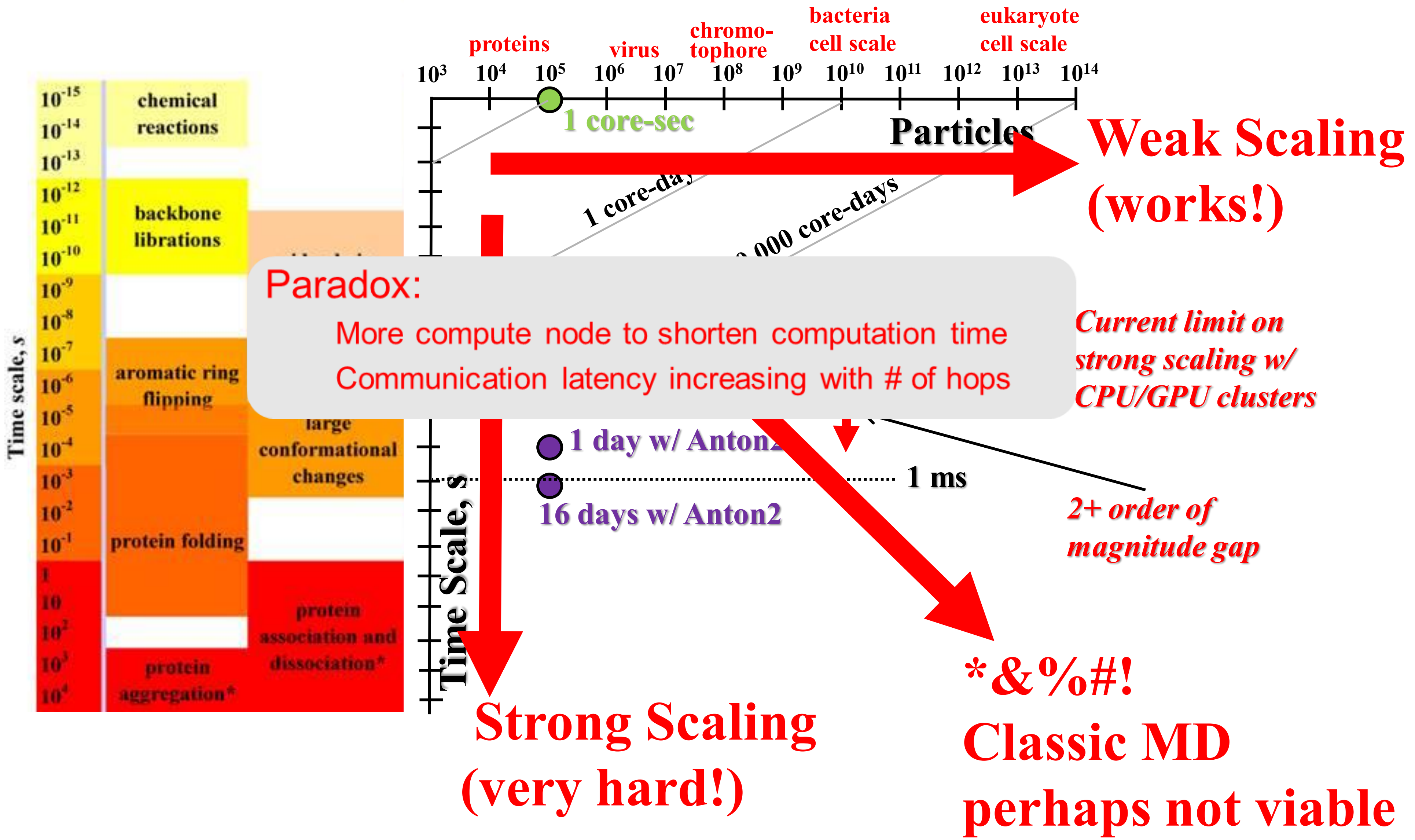
Source: folding @home

1ms of simulated reality for a 90K particle simulation takes

8 ***cores*** (PC)	4000 years	
1K ***cores*** (cluster)	30 years	
1.6M ***cores*** (Sequoia)	???	*>> scale 90K particles to 1.6M cores?*
0.5K ***ASICs*** (Anton)	.1 years	

Question – Can we get similar performance w/ off-the-shelf components?

Time and Length Scales in Biology

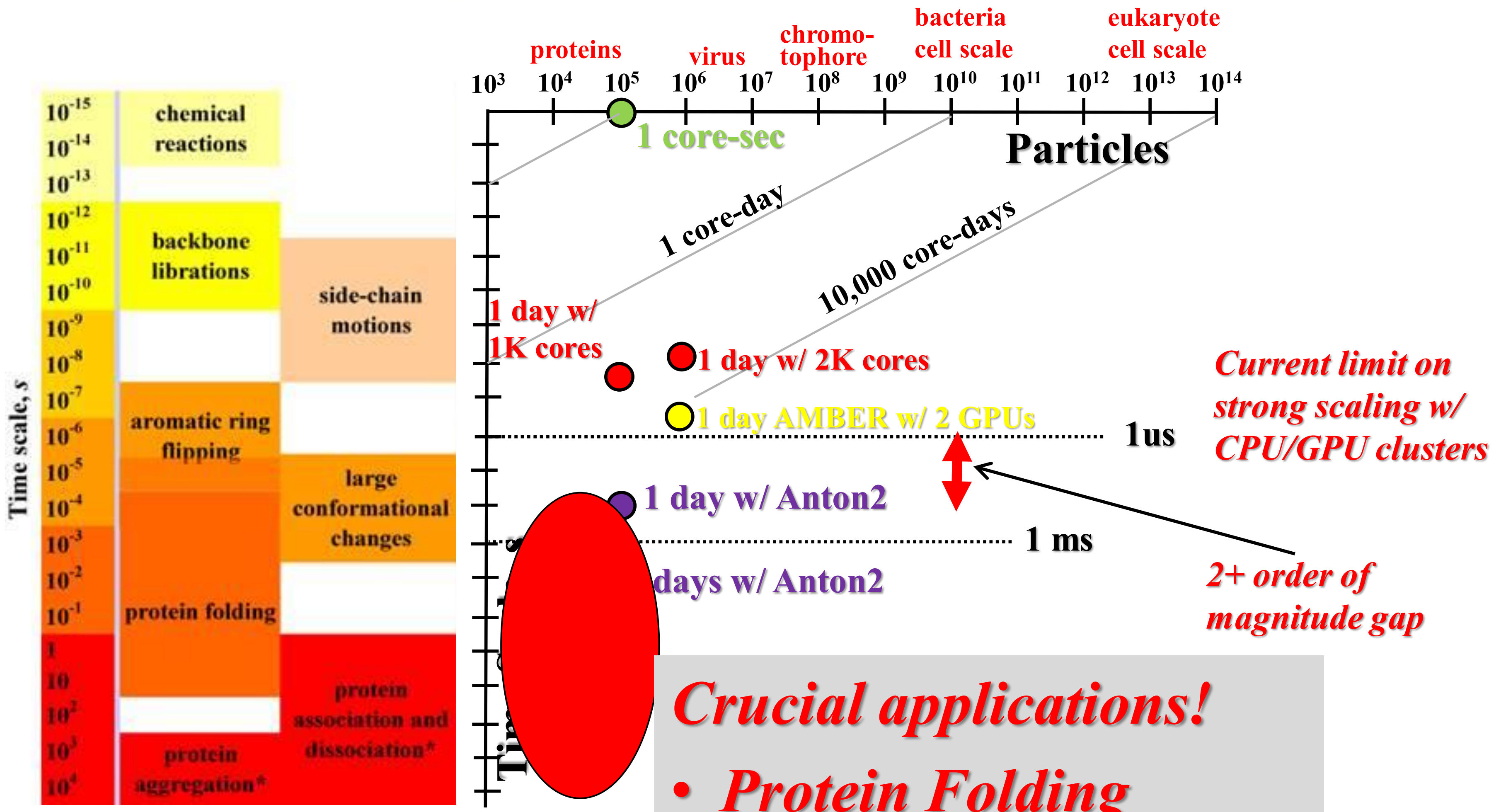


Paradox:

More compute node to shorten computation time

Communication latency increasing with # of hops

Time and Length Scales in Biology



Crucial applications!

- ***Protein Folding***
- ***Drug Discovery***

FPGA/MD Road Map

2008
Single FPGA 3x better than GPU ☺
But FPGAs are too expensive ☹
FPL2009, TRETS2010, FCCM 2011

2008
Anton 100x better than anything!
But costs $??? and is not available
→ Shows potential of clusters w/ ultra-low latency

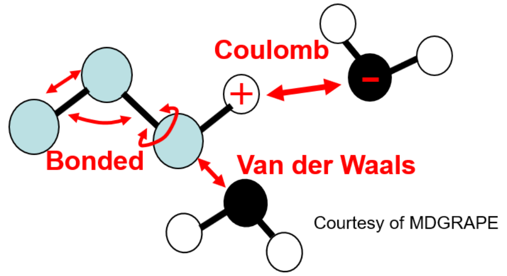
2014
FPGA Clusters demonstrate strong scaling ☺
But proof-of concept only. And FPGAs still not viable. ☹☹
HPEC2014, HEART2015

2016-Present - FPGAs become plausible HPC devices ☺☺

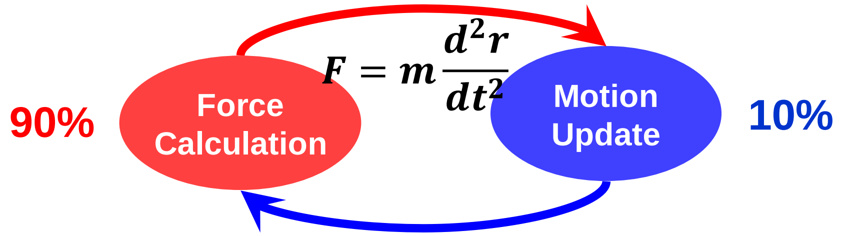
2016
Anton II still 100x better than anything!
But costs $??? and is not available
→ Shows potential of clusters w/ ultra-low latency

Today – Time to build FPGA cluster for MD?
We know strong scaling will work
BUT
Is single FPGA performance still competitive?
ASAP2019, SC2019

Molecular Dynamics Simulation



- Why MD important?
  - Core of Computational Chemistry
  - Central to Computational Biology, with applications to
    - Drug design
    - Understanding disease processes
- What need to be done?
  - Force Evaluation
    - Bonded force
    - Non-Bonded force
  - Motion Update



- Why MD is hard?

$$F_i^{total} = \underbrace{F^{bond} + F^{angle} + F^{torsion} + F^{H}}_{\text{Generally } O(n)} + F^{non-bonded}$$

Initially $O(n^2)$, performed on accelerators

All-to-all communication, hard to scale

Paradox:

More compute node to shorten computation time

Communication latency increasing with # of hops

- State-of-the-Art

1ms of simulated reality for a 90K particle

8 Cores PC	4000 years
1K Cores (HPC cluster)	50 years
0.5K ASICs (Anton)	0.1 year

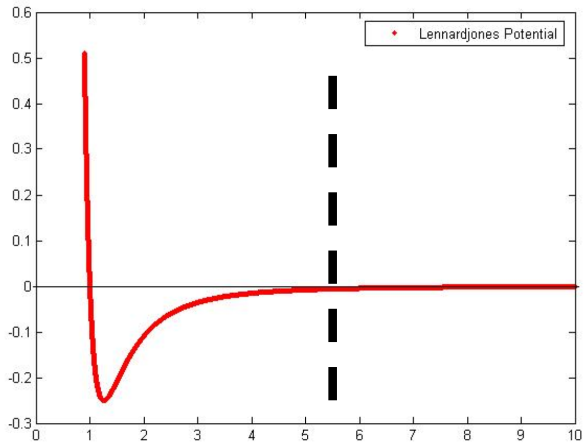
Question: Can we get similar performance with COTs like FPGA?

Outline

Background: Non-Bonded Force and $O(N^2)$ Complexity

- Lennard-Jones Force:
  - Decays quickly…. Cut-off !

$$\frac{F_i^{LJ}}{r_{ji}} = \sum_{j\neq i} \frac{\epsilon_{ab}}{\sigma_{ab}} \left\{ 12\left(\frac{\sigma_{ab}}{|r_{ji}|}\right)^{14} - 6\left(\frac{\sigma_{ab}}{|r_{ji}|}\right)^{8} \right\}$$

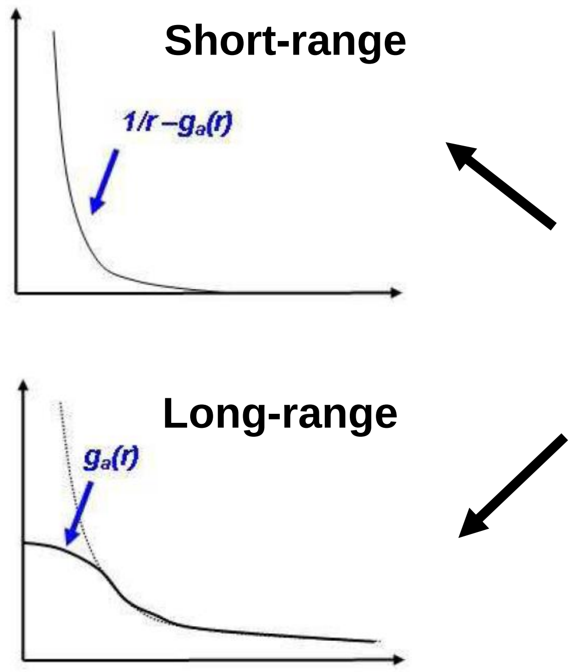


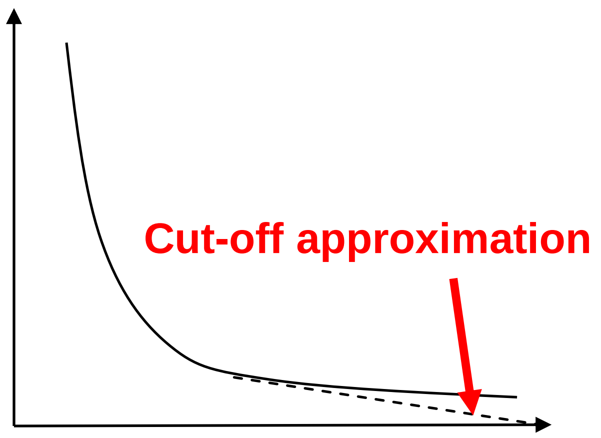
Lennard-Jones Force

- Coulombic Force:
  - Does not decay fast enough…..
  - Partitions!
    - Short-range part:
      - Directly computation
    - Long-range part:
      - Flat curve
      - Particle mesh Ewald algorithm
      - Update every few iterations

$$\frac{F_i^{C}}{r_{ji}} = q_i \sum_{j\neq i} \left(\frac{q_j}{|r_{ji}|^3}\right)$$

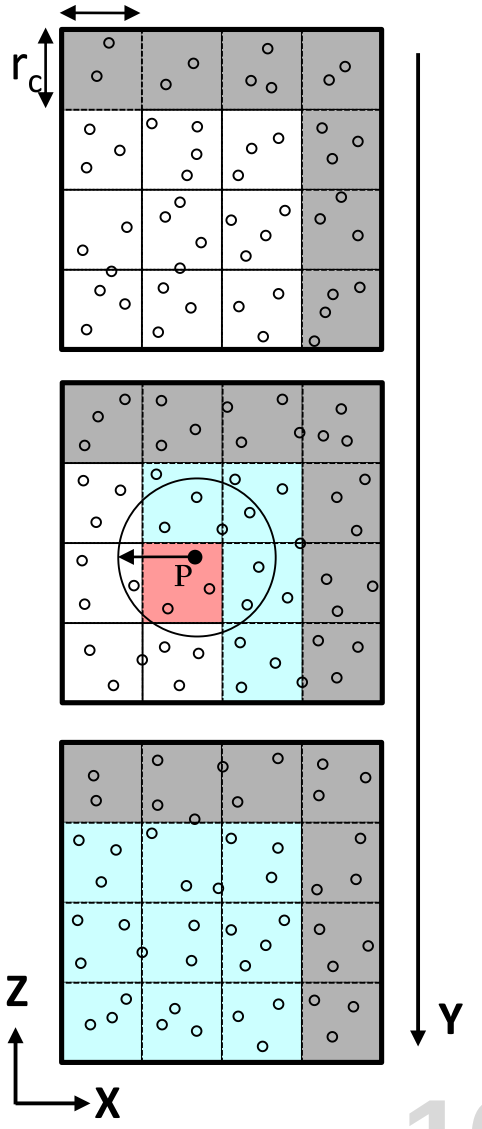
Not on critical path





Coulombic Force

Fundamental problem: computation is not on ***particles***, it's on particle ***PAIRS***

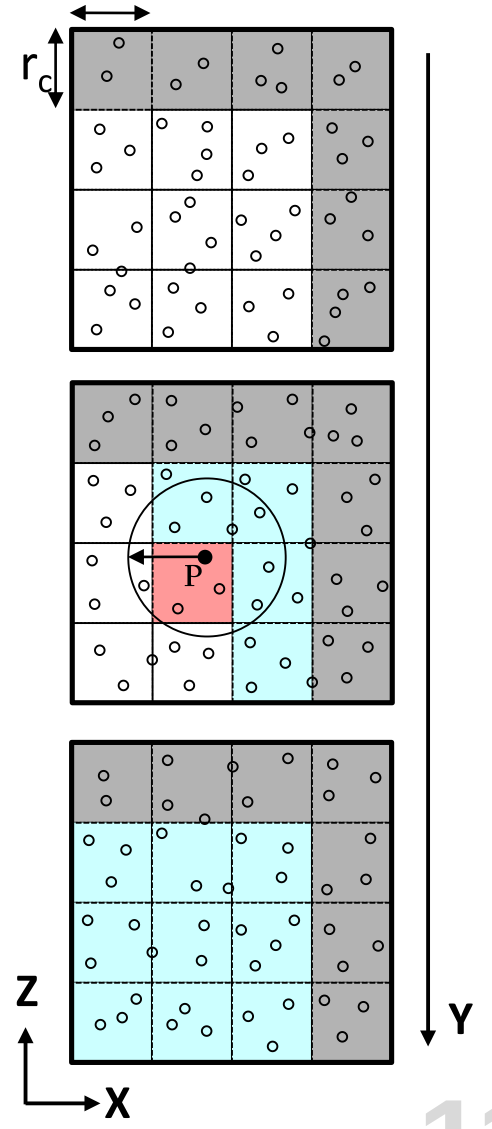


Analogous to MMM, ∃ solution based on spatial locality

- Each (reference) particle is part of many particle pairs (neighbor set)
- Each "other" particle interacts with its own (different) neighbor set of particles
- Method 1 – Coarse spatial sorting by cell
  - Problem: 85% of work is unnecessary
- Method 2 – Each particle maintains its own list
  - Problem: Large preprocessing overhead
  - Problem: Lots of storage

Background: Cutoff Radius and Cell-list

- Cutoff Radius r_c:
  - Evaluate the non-bonded force when distance within r_c
  - $F = 0$ when $r > r_c$
- Cell list:
  - Cell size = r_c^3
  - Given **reference particle** P, only evaluate **neighbor particles** within 26 nearest neighbor cells
- Newton's 3rd Law:
  - Particle interaction is mutual -> Only evaluate half particles within cutoff
- Filtering:
  - Remove unnecessary particles outside cutoff



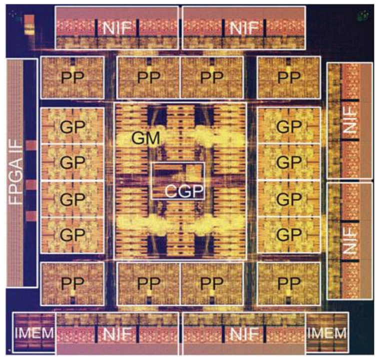
MD Related Work

- Software toolkit (Targeting CPU and GPU Cluster)
  - Amber, NAMD, OpenMM, Gromacs, LAMMPS, etc.
- GPU
  - GTX1080Ti (Courtesy of Silicon Therapeutics)
    - OpenMM with OpenCL (86.21 ns/day)
- ASICs
  - Anton & Anton 2 by DE Shaw Group
    - SoC covers force evaluation, motion integration
    - Off-chip SRAM for data storage & management
  - MDGRAPE-4 by Riken Quantitative Biology Center
    - SoC featuring 64 force evaluation pipelines
    - Focus on non-bonded force

Expensive

Hard to Reach

Anton 2



MDGRAPE-4

Where we are now…

- FPGA/GPU work rely on off-chip device to host/process the input/output
  - Not enough on-chip ram -> Particle data in DRAM
  - Host generate: input particle-pairs
  - Host perform: Motion Update & Particle Migration

➡ **Long Latency**

What if we can keep all the data chip?

- New Opportunity …
  - High-end FPGAs now featuring ~200 Mb on-chip SRAM ➡ **Enough for Small Dataset**
  - MGTs featuring high-bandwidth, low-latency communication ➡ **Distributed Storage**
  - Support for native floating-point ➡ **Better accuracy**
  - More resources on-chip ➡ **More function units**

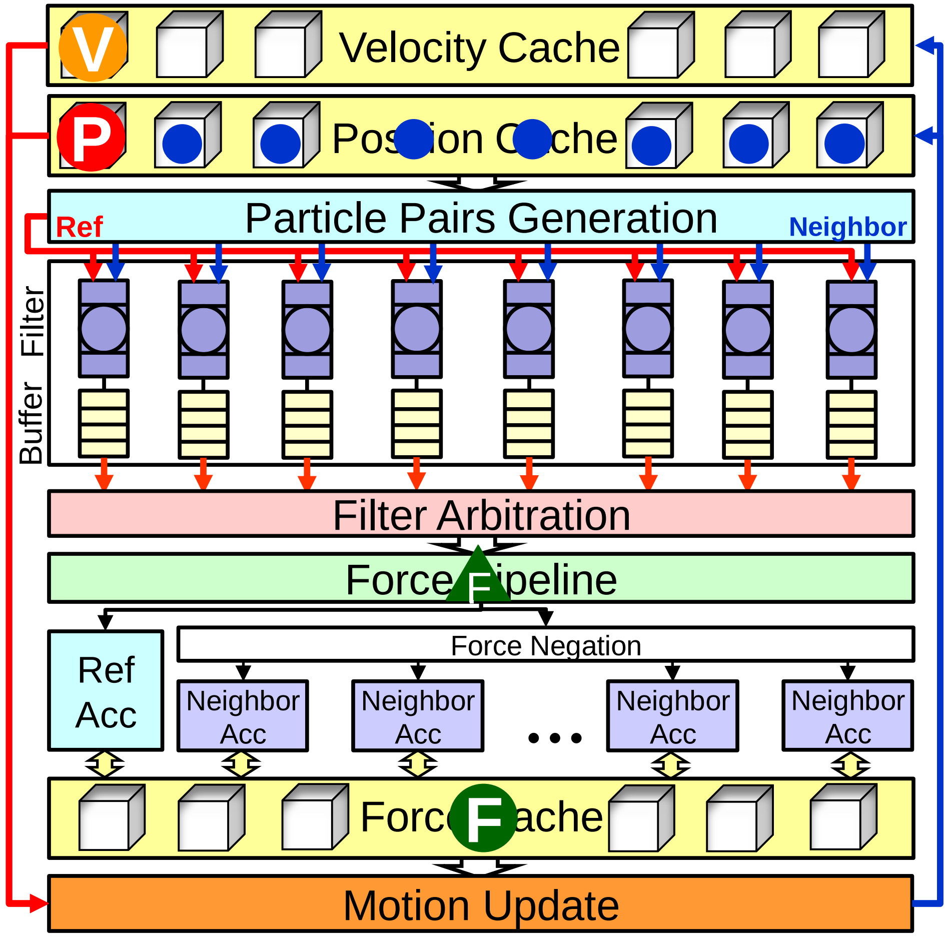
for an end-to-end MD system on FPGA(s)!

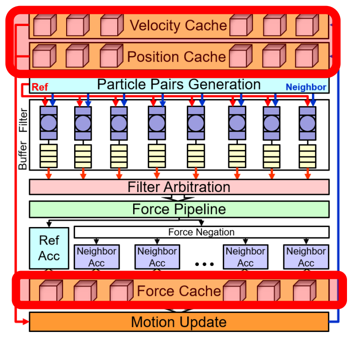
Outline

- MD Background
- **Single-chip End-to-end MD System Implementation**
- Evaluation
- Future Work:
  - Multi-FPGA Strong Scaling
  - Communication Pattern Analysis

Single-Chip End-to-End MD Architecture

- Dataset: Liquid Argon
  - Monoisotopic Element
  - LJ interaction only
- Particle Cache
  - Position, Velocity, Force
- Filter Banks
  - Multiple filters per pipeline with buffers
- Force Evaluation
  - Non-bonded force evaluation
- Accumulation
  - Accumulate particle force to both reference particle and neighbor particles
- Motion Update & Particle Migration
  - Update particle position
  - Update particle velocity

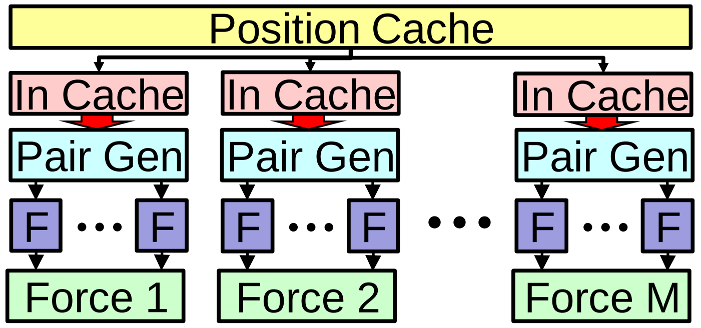




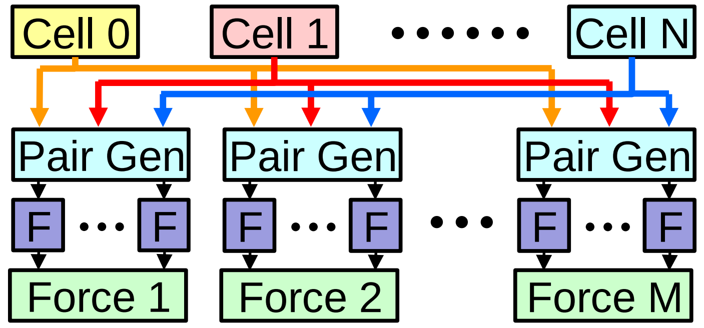
Mapping: Particles on Memory Modules

- Q1: Which Memory Resource?
  - MLAB: lut-based, suitable for small size
  - M20K: flexible, large capacity, high-performance
  - eSRAM: large capacity, low-latency, low flexibility, low availability
- Q2: Unified Mem or Sperate Mem for Different Datatypes?
  - Position, Velocity, Force
  - Different access time & access frequency
- Q3: Single Global Mem or Individual Cell Mem?
  - **Mem 1**: Single Global Memory
    - Simplified wiring, but limited rd&wr BW
  - **Mem 2**: Individual Cell Memory
    - Complex wiring, but sufficient BW

TBD



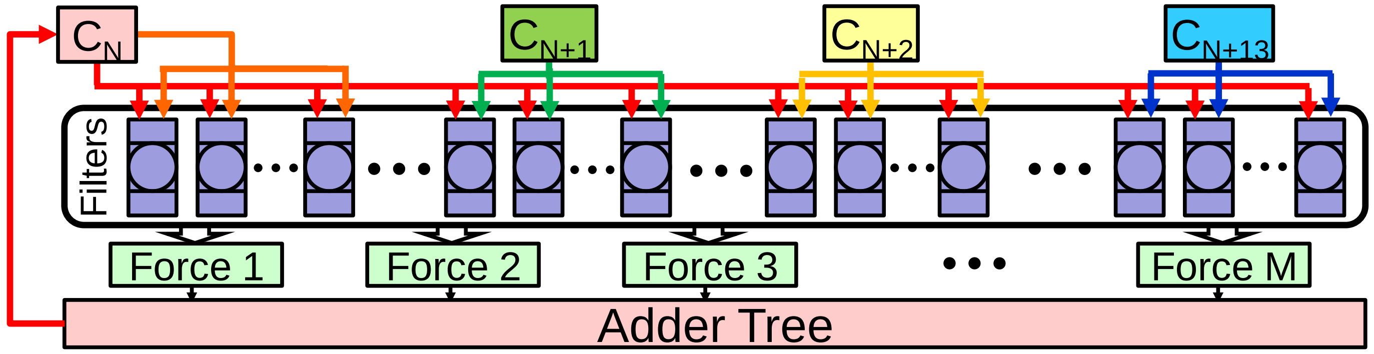
Mem 1: Single Global Memory



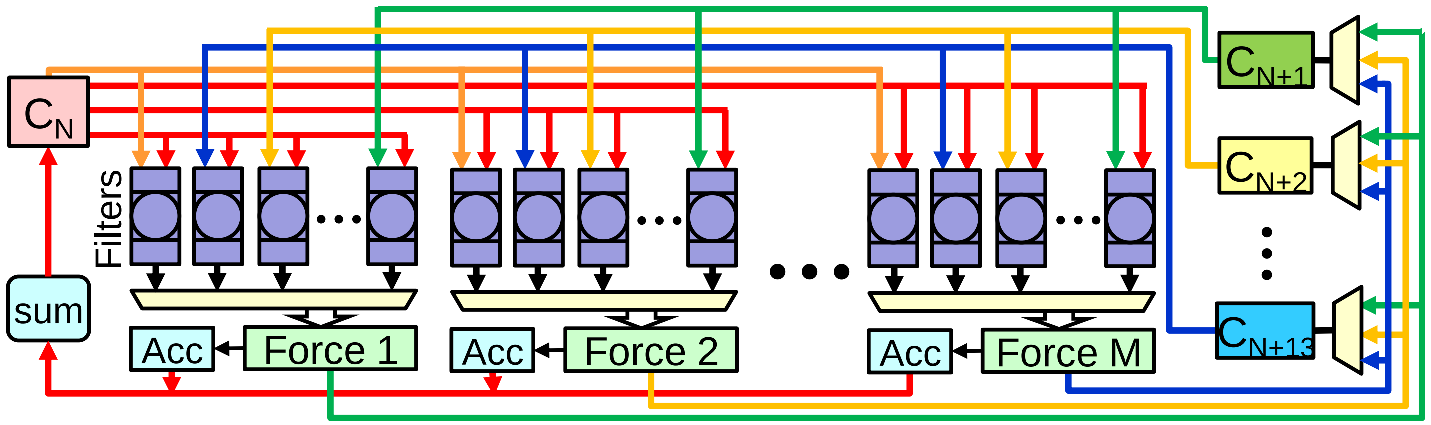
Mem 2: Individual Cell Memory

Mapping: Workload on Pipelines

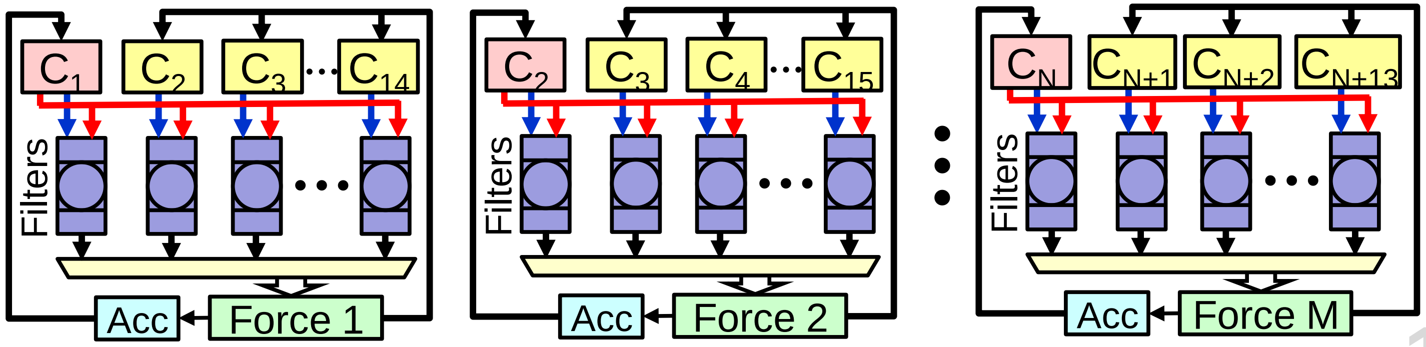
- **Distribution 1**: Pipelines working on same reference particle



- **Distribution 2**: Pipelines working on same cell, but different reference particles

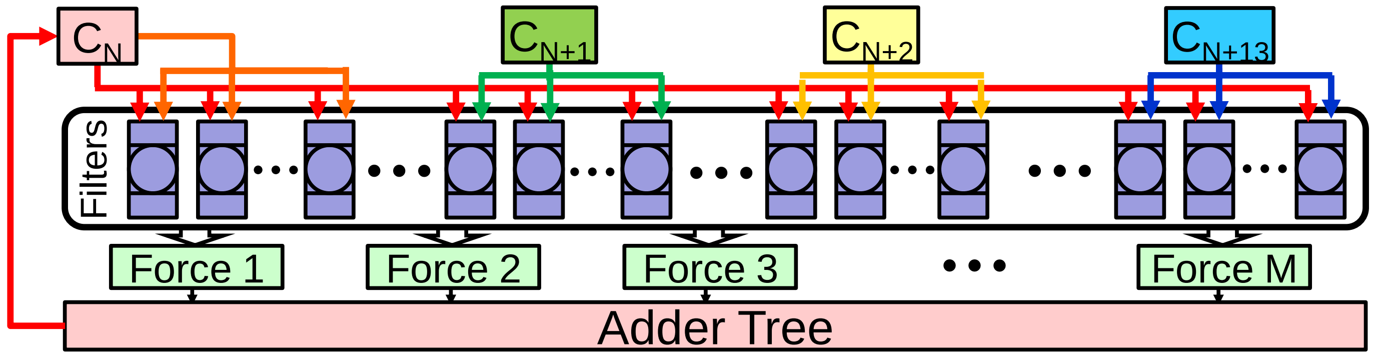


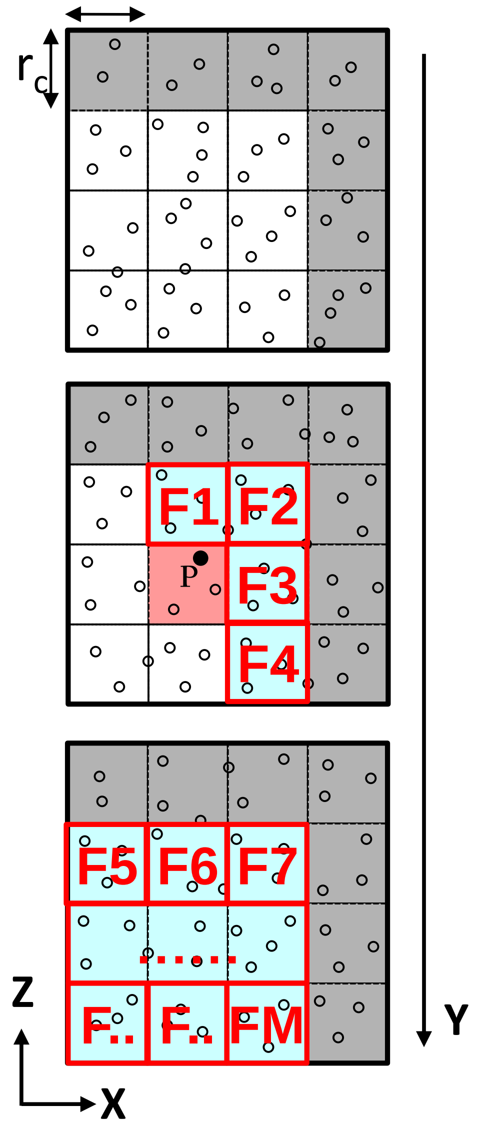
- **Distribution 3**: Pipelines working on different homecells



Mapping: Workload on Pipelines

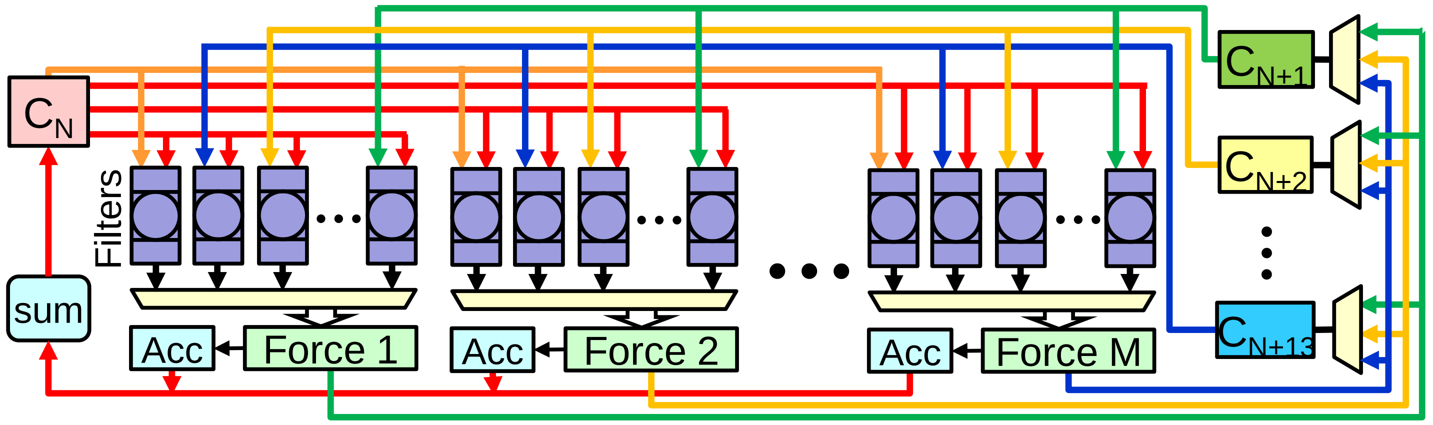
- **Distribution 1**: Pipelines working on same reference particle

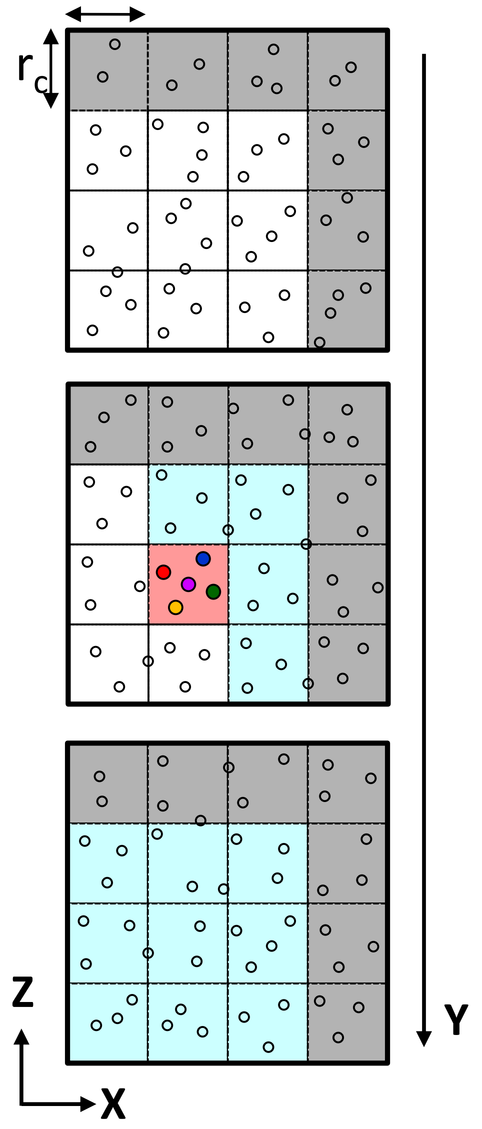




Mapping: Workload on Pipelines

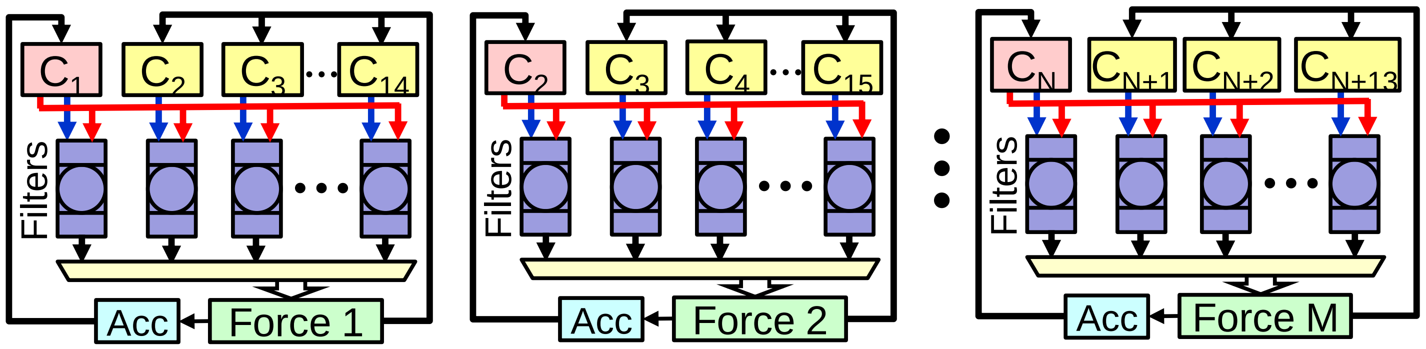
- **Distribution 2**: Pipelines working on same cell, but different reference particles

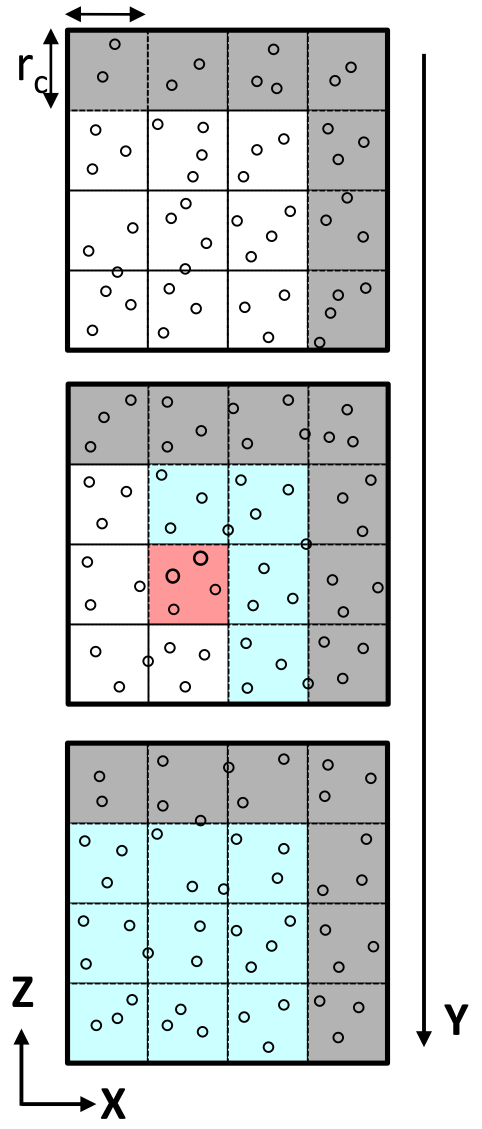




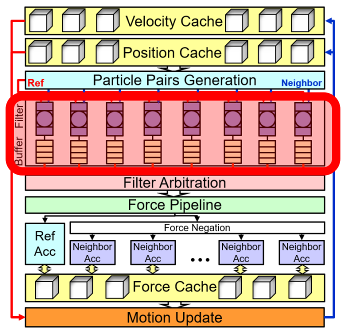
Mapping: Workload on Pipelines

- **Distribution 3**: Pipelines working on different homecells

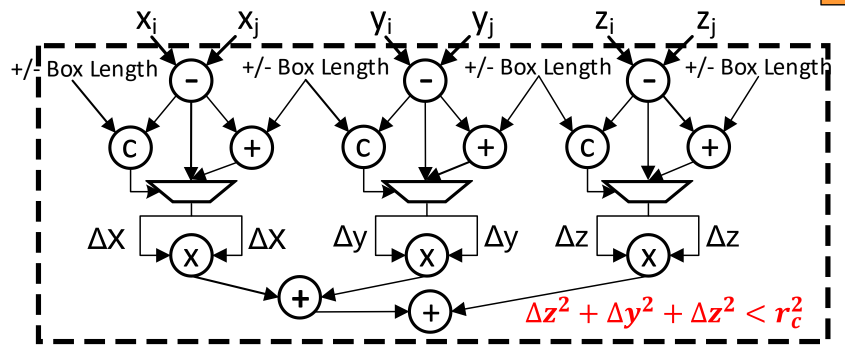




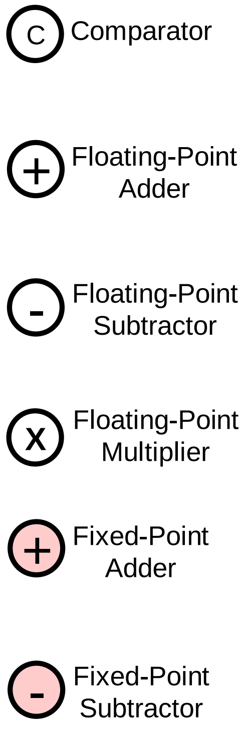
Implementation: Filter Logic

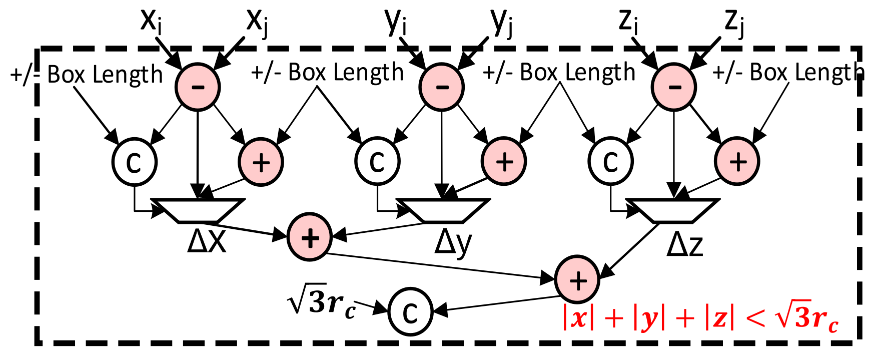


- 8 Filters per Pipeline
  - $AveragePassRate = \frac{\frac{4}{3}\pi r_c^3}{27 \times r_c^3} \approx 15.5\%$
  - $\geq$ 7 filters to guarantee a throughput of 1
- Direct Computation
  - Datatype: Single Precision
  - $r^2 = \Delta x^2 + \Delta y^2 + \Delta z^2 \leq r_c^2$
  - DSP usage: 10
- Planar Method[1]
  - Datatype: Fixed Point
  - $|x| + |y| + |z| < \sqrt{3} r_c$
  - DSP usage: 0
  - Overhead: 7% extra work



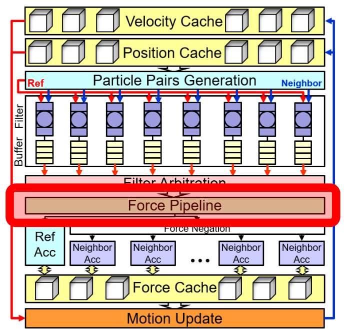
Direct Computation





Planar Method

* M. Chiu, *Molecular Dynamics Simulations on High Performance Reconfigurable Computing Systems*, ACM-TRETS (2010)

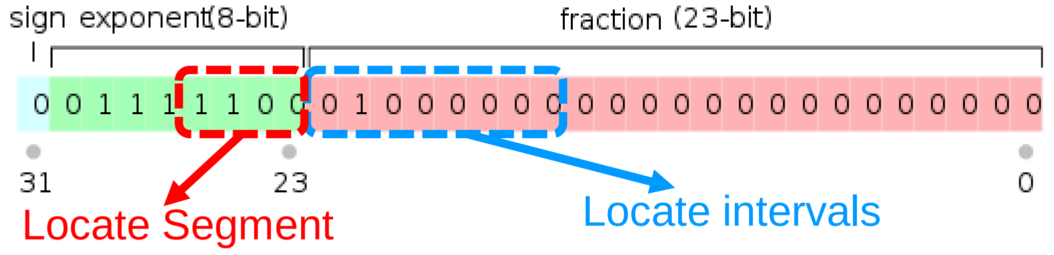


Imple: Force Evaluation using Interpolation

Single Float

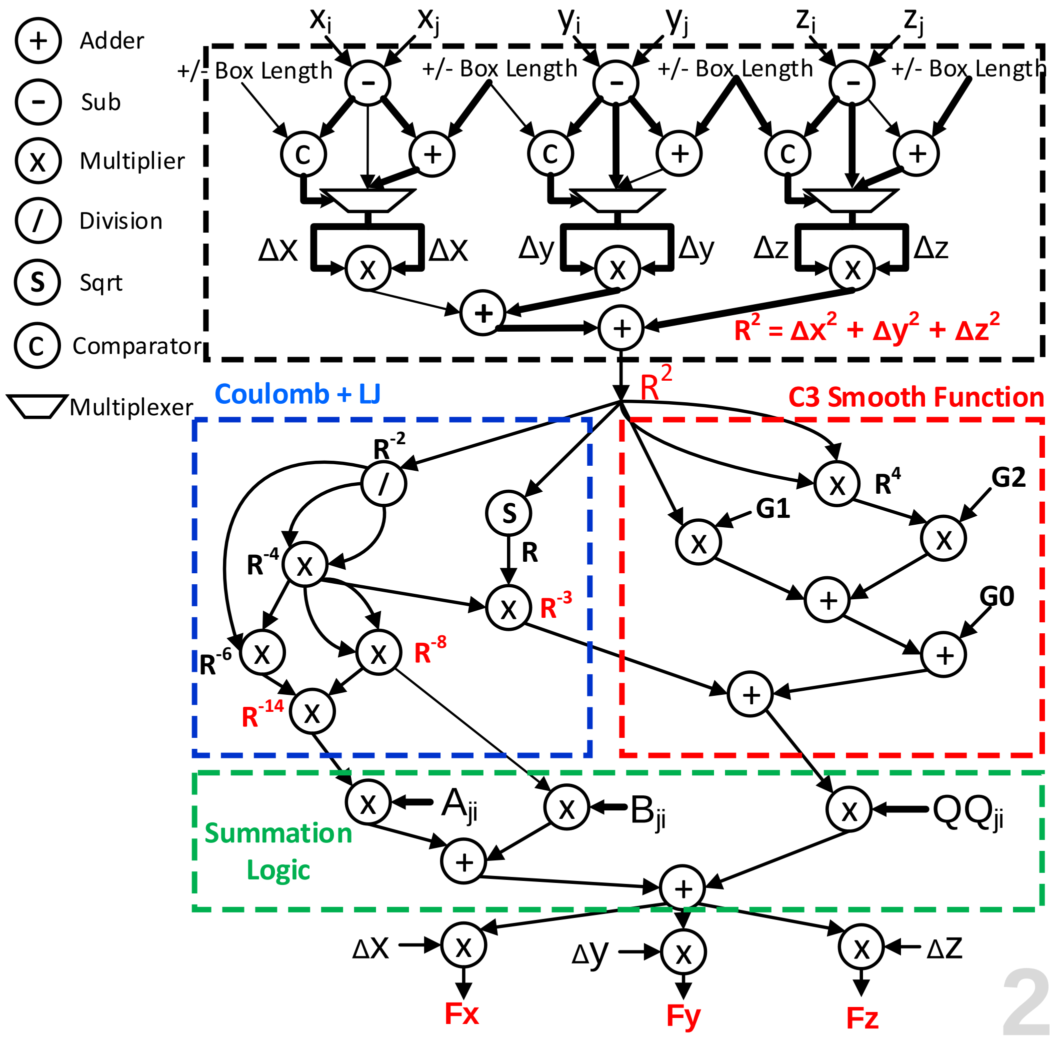
$$\frac{F_i^{LJ}}{r_{ji}} = \sum_{j \neq i} (A_{ab} r_{ji}^{-14} + B_{ab} r_{ji}^{-8})$$

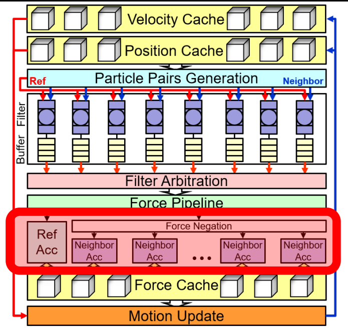
- Interpolation: Sections and Intervals
  - Divide the curve into multiple **sections**
  - Each section is doubled range compare with previous
  - Same # of **intervals** within each section



  - Each interval: Interpolation with polynomials
  - $r^k = (C_2(x-a) + C_1)(x-a) + C_0$
- Force Evaluation with Table Lookup
  - Pre-calculate coefficients and store in lookup tables
  - Using R^2 as table lookup entry

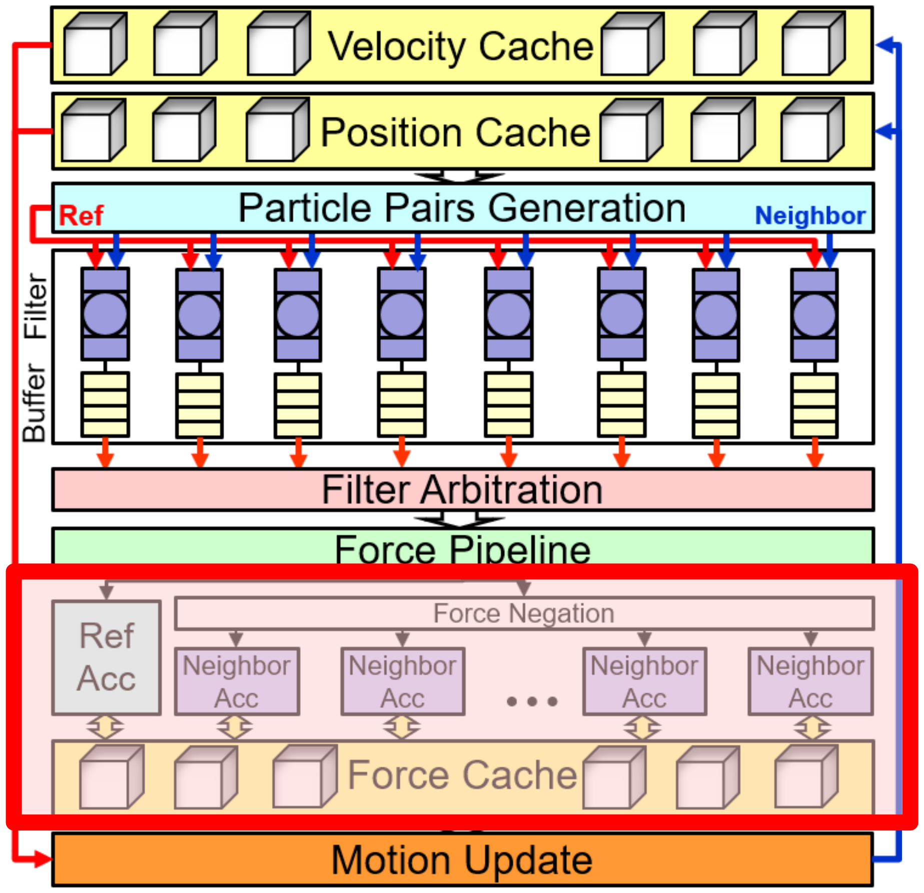
$$\frac{F_i^{RL}}{r_{ji}} = A_{ab} R_{14}(|r_{ji}|^2) + B_{ab} R_8(|r_{ji}|^2)$$





Implementation: Partial Force Accumulation

- Task:
  - Accumulate to Reference & Neighbor Particles
- Challenge:
  - DSP Latency: 3 cycles
- Reference Particle Acc
  - <u>Location:</u> output side of force pipeline
  - Accumulating to same value
  - Sol: Accumulate to 3 different temp values
- Neighbor Particle Acc
  - <u>Location:</u> input side of force cache
  - Accumulating to different values most of the time
  - Chances accumulating to same value
  - Sol: Data hazard detection by checking active ID



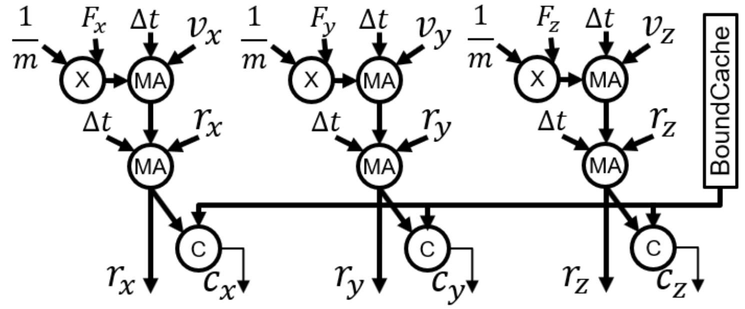
Imple: Motion Update & Particle Migration

$$\vec{a}(t) = \frac{\vec{F}(t)}{m}$$

$$\vec{v}(t+\Delta t) = \vec{v}(t) + \vec{a}(t) \times \Delta t$$

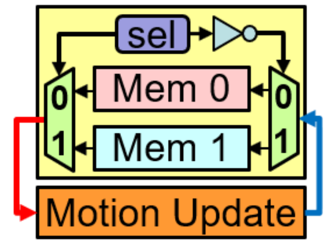
$$\vec{r}(t+\Delta t) = \vec{r}(t) + \vec{v}(t+\Delta t) \times \Delta t$$

- Motion Update
  - Classic Newtonian Law
  - Frequency: once every simulation timestep
  - Datatype: Floating Point
- Particle Migration
  - Small timestep -> Migrate within nearest neighbor
  - Cell Boundary: Table lookup
  - Cell Particle Update: double buffering in Pos&Vol Cache
    - Write new particle to alternative set of memory
    - Avoid maintaining list of vacant memory space

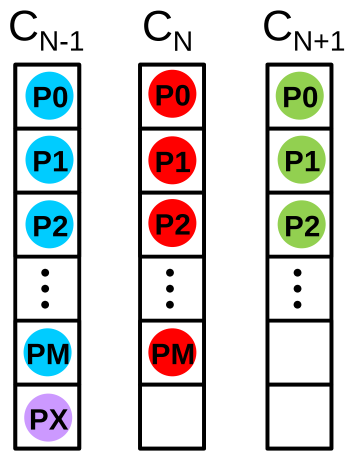


$\frac{1}{m}$ F_x Δt v_x X MA Δt r_x MA C r_x c_x
$\frac{1}{m}$ F_y Δt v_y X MA Δt r_y MA C r_y c_y
$\frac{1}{m}$ F_z Δt v_z X MA Δt r_z MA C r_z c_z
BoundCache

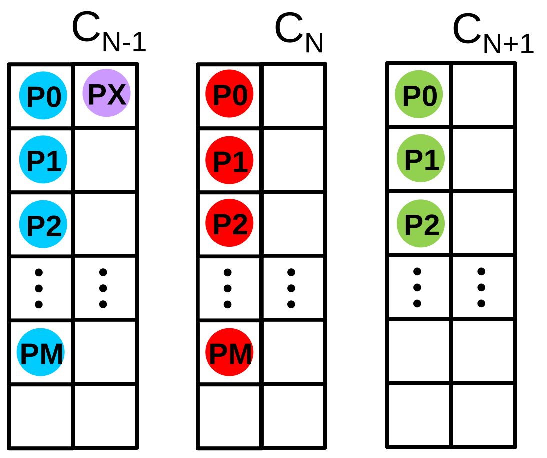
Motion Update Datapath



Position & Velocity Cache



On-Site Update



Update with Double Buffer

Outline

- MD Background
- Single-chip End-to-end MD System Implementation
- **Evaluation**
- Future Work:
  - Multi-FPGA Strong Scaling
  - Communication Pattern Analysis

Evaluation: Experimental Setup

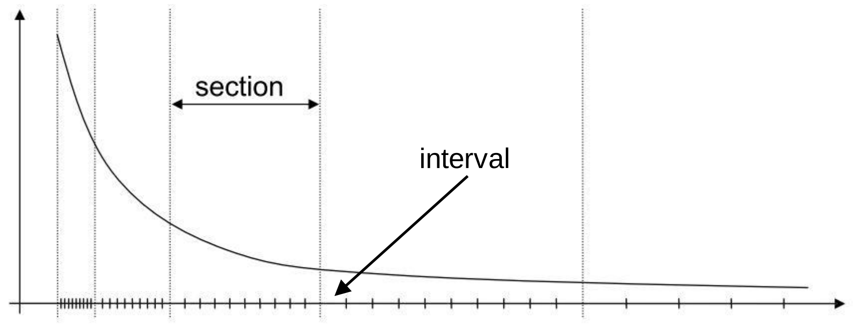
- FPGA: Reflex XpressGX S10-FH200G Board
  - Intel Stratix 10 1SG280LU2F50E2VG
  - Abundant DSP and RAM Resources:

ALMs	RAM Blocks	DSP Units
933,120	11,721	5,760

- MD Dataset:
  - Liquid Argon Dataset: 20K atoms -> Generated by Packmol[1], only has LJ interaction
- MD Benchmark:
  - Amber: software-based benchmark, support CUDA

[1] L. Martinez, R. Andrade, E. Birgin, and J. Martinez. PACKMOL: a package for building initial configurations for molecular dynamics simulations. Journal of Computational Chemistry, 30(13):2157–2164, 2009.

Evaluation: Interpolation Accuracy



- Which interpolation order?
- How many intervals per section?

More Memory Usage for Indexes

More DSP

Interval	16	32	64	128	256
1st Order	99.7700	99.9336	99.9842	99.9960	99.9990
2nd Order	99.9899	99.9988	99.9998	99.9999	99.9999
3rd Order	99.9996	99.9999	99.9999	99.9999	99.9999

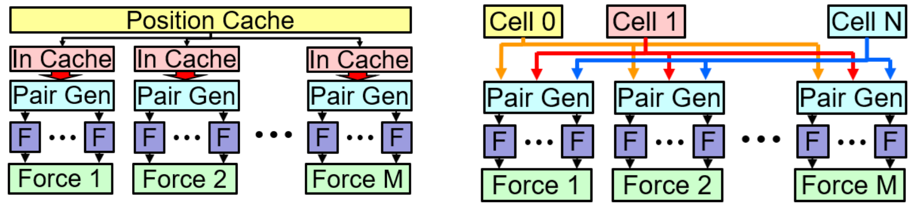
Higher order ➡ More DSP usage & More index to store

More Intervals ➡ More memory usage

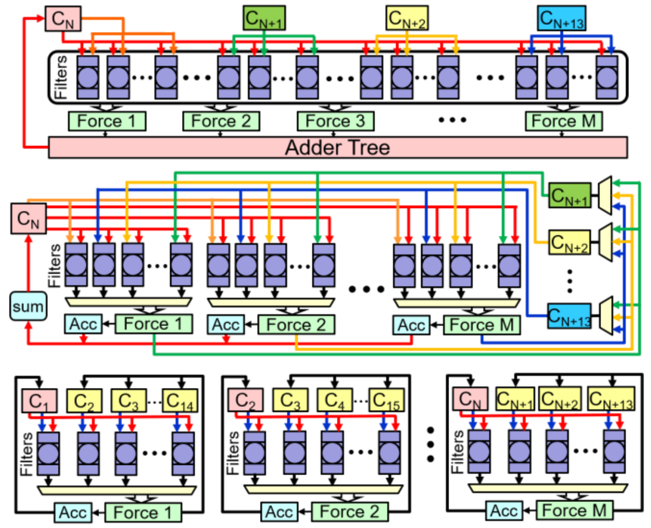
DSP ⟺ Block RAM

Evaluation: System Resource Utilization

- Quick Recap:
  - 2 particle-to-memory mapping:
    - **Mem 1:** All particle in a single large memory unit
    - **Mem 2:** Individual cell memory
  - 3 workload distributions:
    - **Distribution 1:** All pipelines working on same reference particle
    - **Distribution 2:** All pipelines working on same homecell, but different ref particle
    - **Distribution 3:** Pipeline working on different homecells
- Total of 6 combinations



Particle-to-Memory Mappings



Workload Distributions

Evaluation: System Resource Utilization

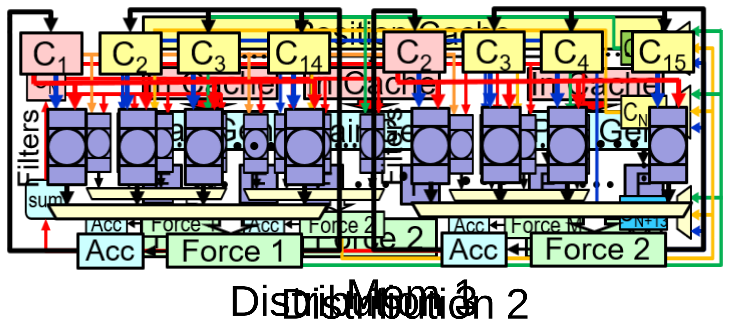
Design	ALM	BRAM	DSP	Freq (MHz)	# Pipe	# MU
Design1: Mem1+Dist1	728,678(78.1%)	8,530(72.8%)	2,772(48.1%)	352	105	10
Design2: Mem2+Dist1	740,954(79.4%)	5,078(43.3%)	2,037(35.4%)	338	55	10
Design3: Mem1+Dist2	728,378(78.1%)	8,320(70.1%)	2,391(41.5%)	343	105	10
Design4: Mem2+Dist2	740,654(79.4%)	5,078(43.3%)	1,656(28.8%)	340	55	10
Design5: Mem1+Dist3	746,561(80.0%)	8,734(74.5%)	2,436(42.3%)	346	110	10
Design6: Mem2+Dist3	753,060(80.7%)	8,420(71.8%)	2,466(42.9%)	346	110	10

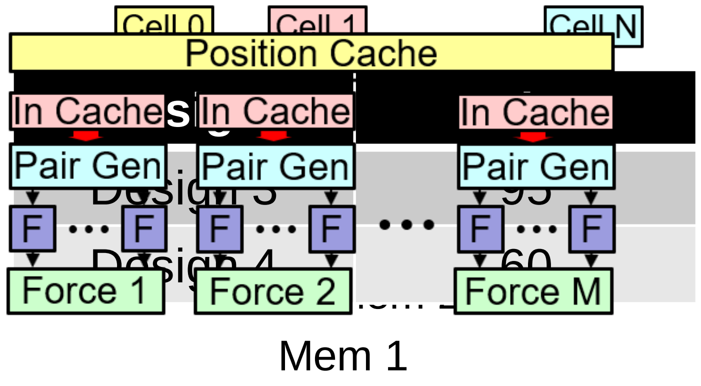
- Design 2 & 4: require all-to-all connection between distributed cell mem and pipelines
- Design 1 & 3: Design 1 requires a large adder tree to sum up partial results from all pipelines, thus slight less pipelines
- Design 5 & 6: no adder tree, no all-to-all connection
- <u>In general, Mem1 seems beneficial due to simplified interconnection</u>

Global Memory is the way to go??

Evaluation: Simulation Time Performance

Platform	Iteration Time (µs)	Simulation Rate (ns/day)
CPU (i7-8700K) 1-core	71,300	2.42
CPU (i7-8700K) 24-core	11,800	14.64
GPU (GTX 1080Ti)	402	430.05
Design1: Mem1+Dist1	63,645	2.72
Design2: Mem2+Dist1	184	941.45
Design3: Mem1+Dist2	832	207.72
Design4: Mem2+Dist2	255	677.53
Design5: Mem1+Dist3	180	959.43
Design6: Mem2+Dist3	122	1412.98



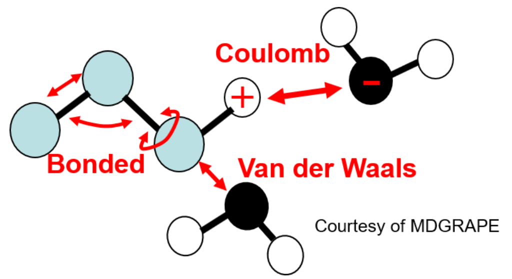


- Design 1: Memory bottleneck: loading time to fill the input cache cannot be hide by computation
- Design 2: Distributed cell memory alleviate the bandwidth pressure
- Design 3 & 4: each cell has less particles (80) than the # of pipelines, some pipelines is idle
- Design 3: also suffered from memory bandwidth limit
- Design 5 & 6: similar performance; no memory bottleneck, same amount of pipelines
- Design 5: overhead on reading out the first set of input data, later on is hided by computation time

Outline

- MD Background
- Single-chip End-to-end MD System Implementation
- Evaluation
- **Future Work:**
  - **Multi-FPGA Strong Scaling**
  - **Communication Pattern Analysis**

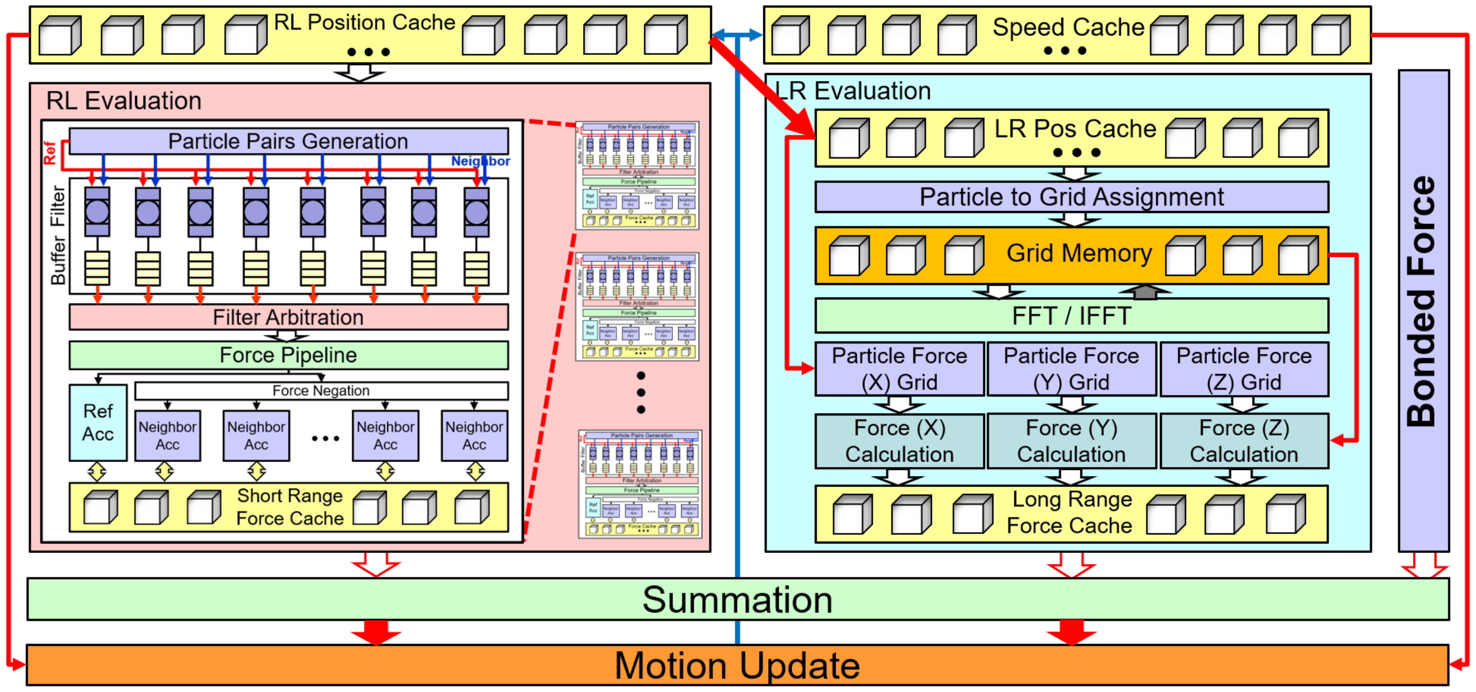
Future Work on MD: Full Evaluation



- What we achieved
  - Particle Pair Generation
  - Short Range Pipeline
  - Motion Update
- To be done
  - Long-Range Part
  - The O(N) part:
    - Bonded Force
    - Angle, torsion, etc.

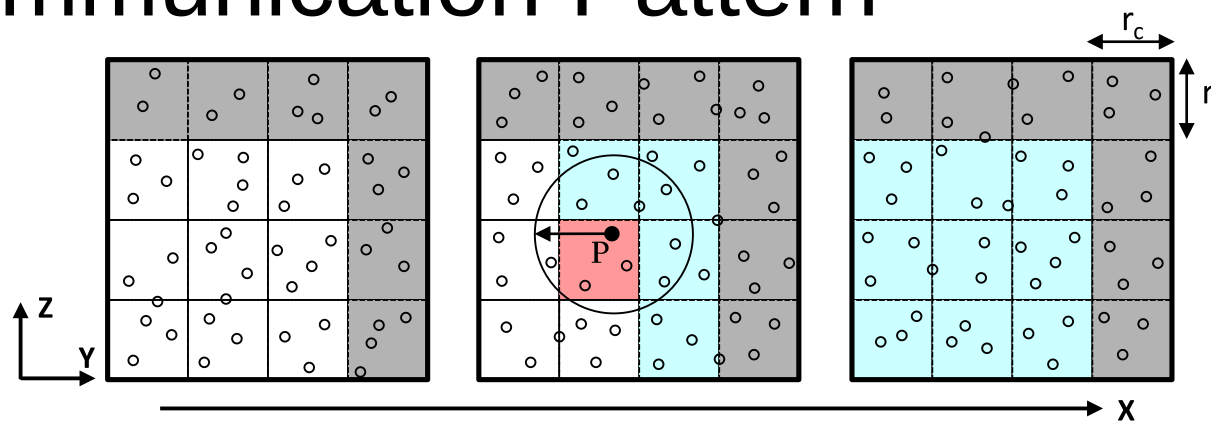
$$F_i^{total} = \underbrace{F^{bond} + F^{angle} + F^{torsion} + F^{H}}_{\text{Generally O(N), performed on host}} + F^{non-bonded}$$

Initially O(n²), performed on coprocessor



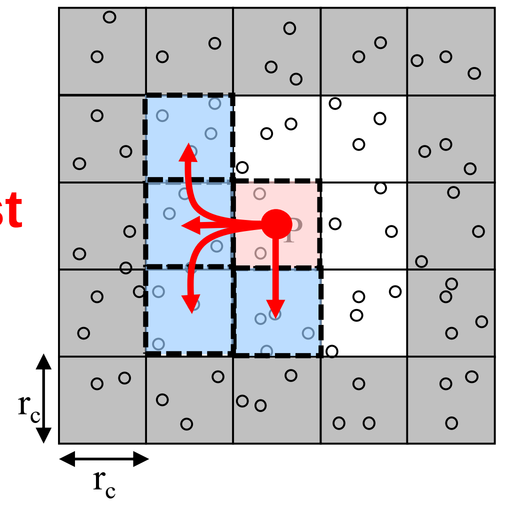
Future Work: Multi-FPGA Communication Pattern

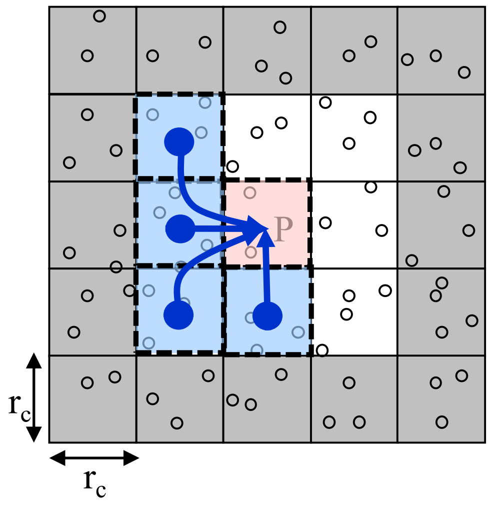
- Each FPGA Handles a Single Cell:
  - Exchange with 26 neighbor nodes
- Newton's 3rd Law -> Half Shell Method*
  - Exchange with 13



- Data Movement:
  - **Export position data**
  - **Import and sum partial forces**

Multicast



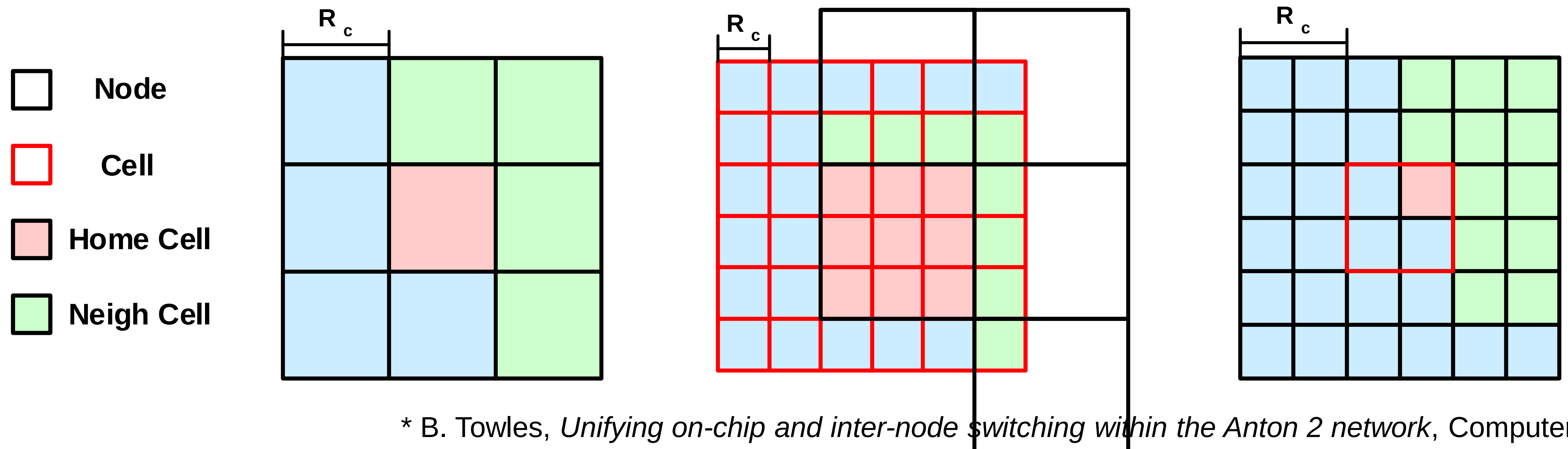


Reduction

* D. Shaw, *A Fast, Scalable Method for the Parallel Evaluation of Distance-Limited Pairwise Particle Interactions*, Computational Chemistry, 2005

Future Work: Mapping Cell onto Multi-FPGA

- Mapping cells to FPGA*
  - Case 1: Single Cell per FPGA
    - Multicasting to 13 neighbor nodes in 3D (single cell)
  - Case 2: Multiple cells per FPGA
    - Multicasting to 13 neighbor nodes in 3D (multiple cells)
  - Case 3: Single cell on multiple FPGAs
    - Multicasting to 62 neighboring nodes (partial cell)



* B. Towles, *Unifying on-chip and inter-node switching within the Anton 2 network*, Computer Architecture News, 2014

Thank you!

Q & A