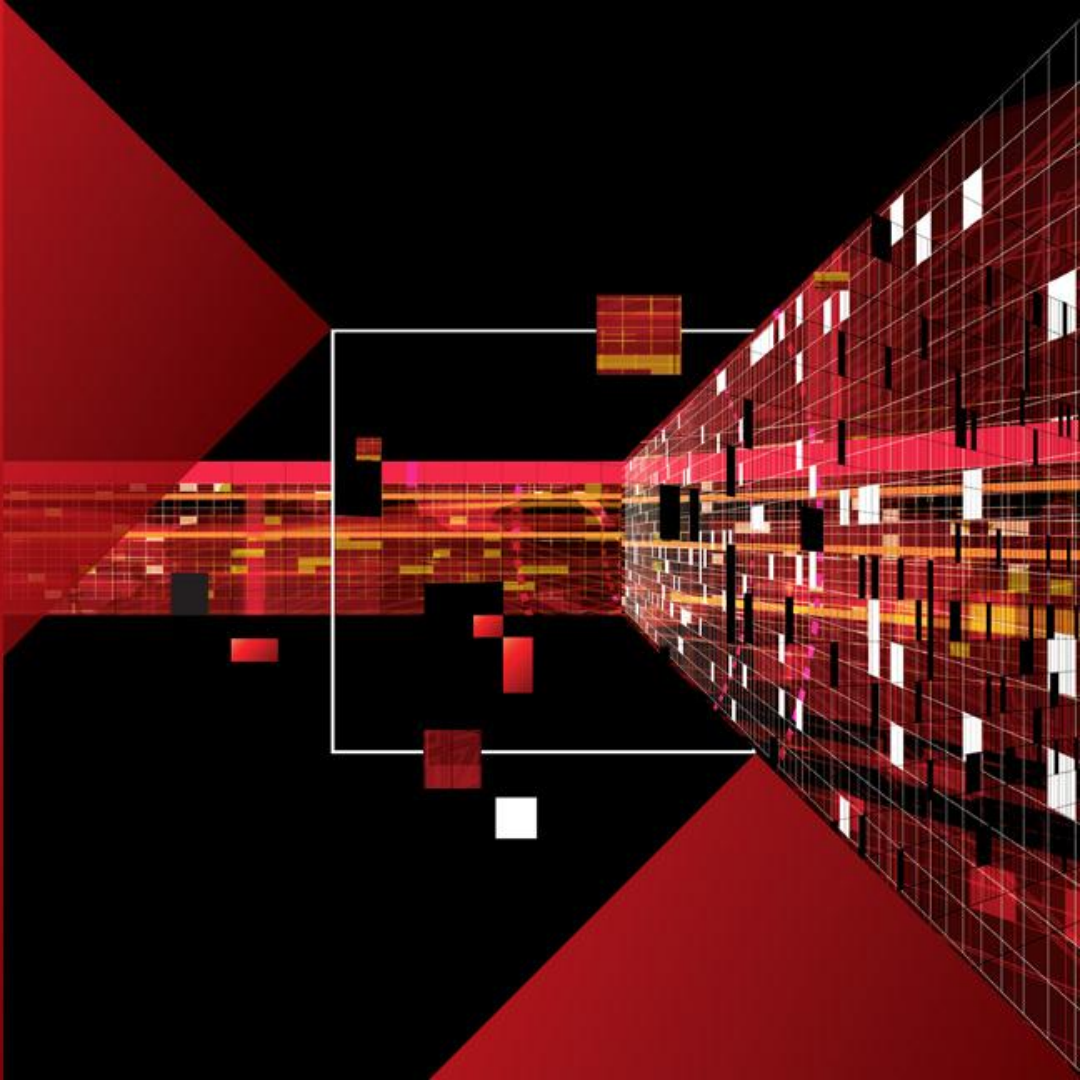




# Fusion<sup>™</sup>

DEVELOPER SUMMIT





**Fusion**<sup>11</sup>  
DEVELOPER SUMMIT

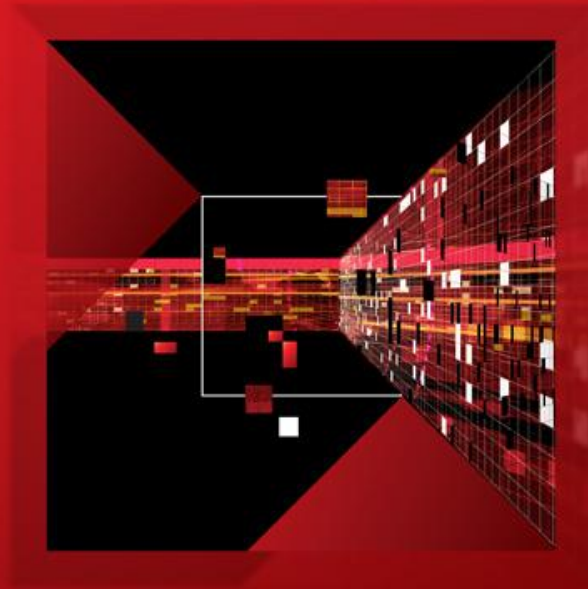
# *Using OpenCL for the Design of Molecular Dynamics Simulations Targeting Hybrid CPU/GPU Computing Architectures*

David Richie  
Brown Deer Technology

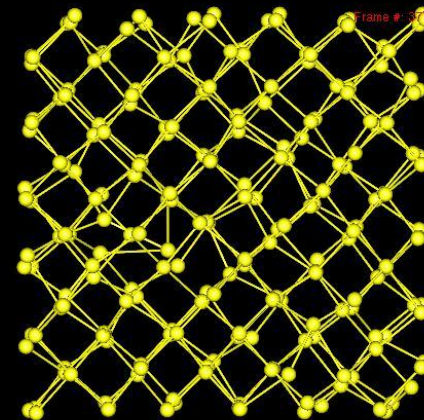
- The Physics : Molecular Dynamics Simulations
  - Complexity of the potentials
  - Example of a difficult problem: silicon defect diffusion
- Software Design
  - Targeting hybrid CPU/GPU architectures
  - The components of a Parallel Replica Molecular Dynamics Code
  - Stillinger-Weber three-body potential for silicon: cache-optimized cell-based approach
- Performance benchmarks ( CPU, GPU, CPU+GPU )

- Molecular Dynamics an early example of HPC applications “accelerated” with GPUs
- Approach was to speed-up localized computationally intensive parts
  - CPU and GPU work together, dividing up the work based on (perceptions) of what each is best at
  - Raised the question, what are GPUs *not* good for?
    - This question has been asked and answered repeatedly over the last several years – answer keeps changing
- Lead to a software design exercise in which multi-core CPUs and many-core GPUs are treated equally
  - Enabled by OpenCL as foundation with STDCL as a more convenient interface
  - Host platform acts as a scheduler (OS) for operations on a highly-parallel platform architecture
- In order to make it interesting, used a difficult molecular dynamics case as a target
  - Parallel-replica molecular dynamics with a silicon Stillinger-Weber three-body interaction
- The objective: put the entire parallel calculation on OpenCL devices
  - No aspect of the calculation is left on the host platform

*The Physics | Molecular  
Dynamics Simulations*

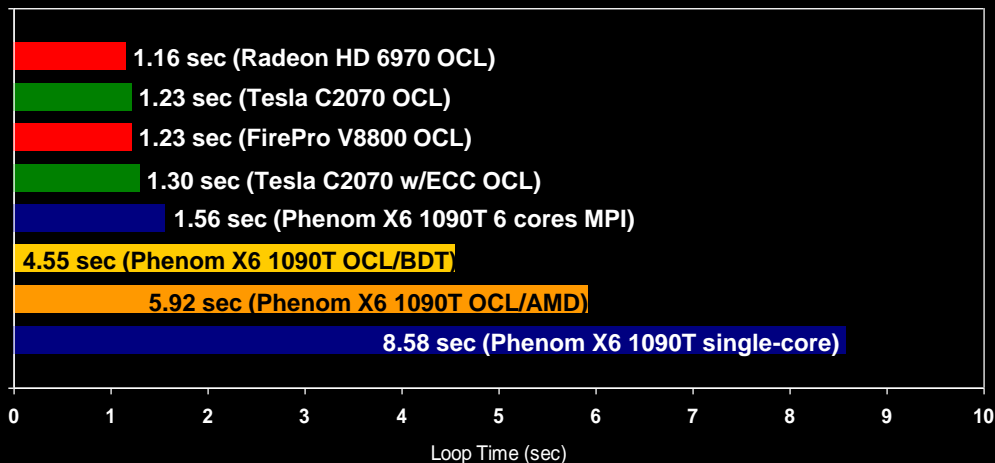


- Molecular dynamics (MD) is a method of simulating materials at the atomistic level
- Integrate the classical equations of motion iteratively using small time-steps
- At each step the force on each particle is calculated using the interaction with neighboring particles
- A critical characteristic is the choice of potential used to describe inter-particle forces
  - ALL such potentials are approximations designed to make the actual physics tractable
    - The real physics requires many-body quantum mechanical solutions
  - Classical potentials are “fit” in the hopes that they “work”
- Additional complexity in the objective of the simulation
  - Constant temperature, constant energy, etc.



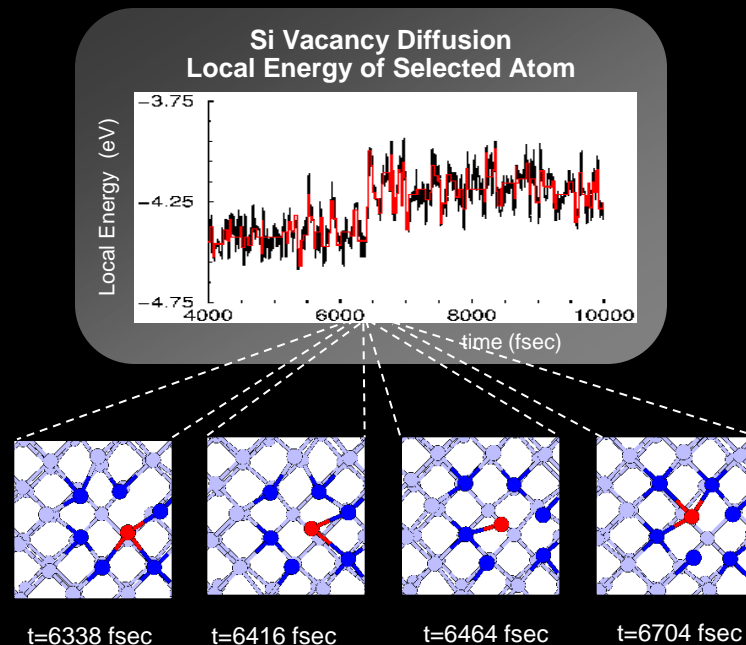
- 511 atom silicon lattice at a finite temperature

- Complexity of molecular dynamics (MD) is in the inter-particle potential
  - Some potentials are easy: two-body potentials, e.g., variations of Lennard-Jones, EAM
  - Some potentials are difficult: higher-order potentials, e.g., three-body Stillinger-Weber
  - Classical MD can be dominated by the approach to nearest-neighbor bookkeeping

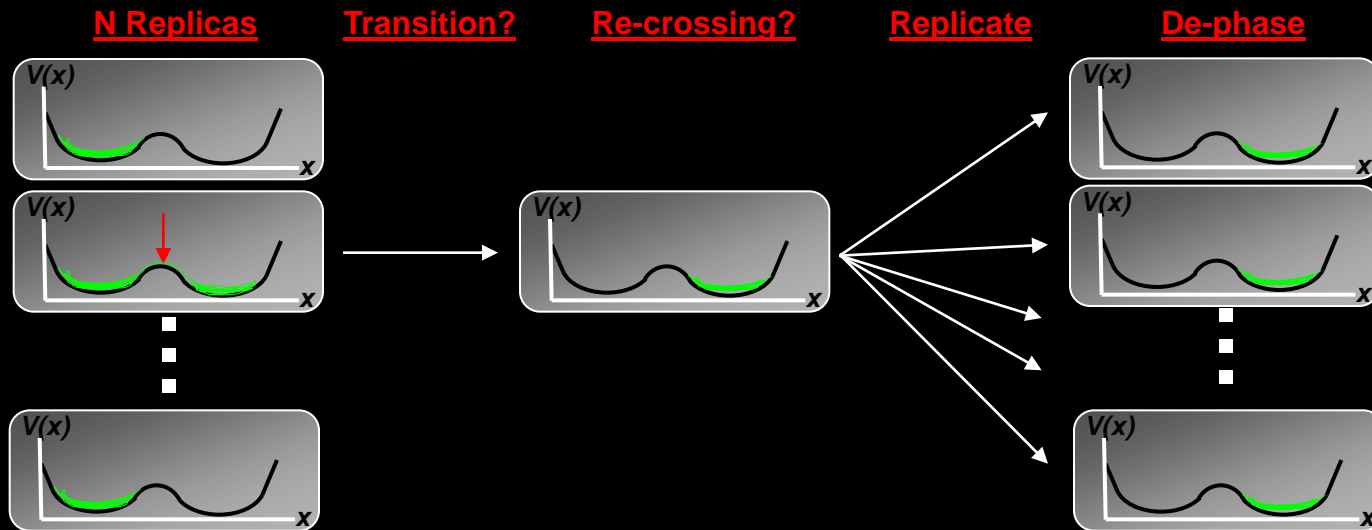


- Example of “easy” – the EAM benchmark
  - Port LAMMPS EAM benchmark to GPU
  - Require *limited* source code modifications
  - *No change* in CPU algorithm
  - *No GPU optimization*
  - Pair-potential and neighbor calculations
  - Less than 5 days effort
  - Enabled by STDCL interface to OpenCL

- Molecular Dynamics can be used to study thermally induced infrequent events
- Examples include defect diffusion in solids, can be used to study growth of large defects impacting material properties
- Simulation is challenging
  - Requires small time-step to correctly model thermal motion
  - Requires long simulations to observe events
- For most of the simulation nothing actually happens – events are infrequent
- Transition events occur over relatively small time intervals relative to the overall simulation time
- The real motion of interest is in the transition events
- The thermal motion is necessary, but uninteresting



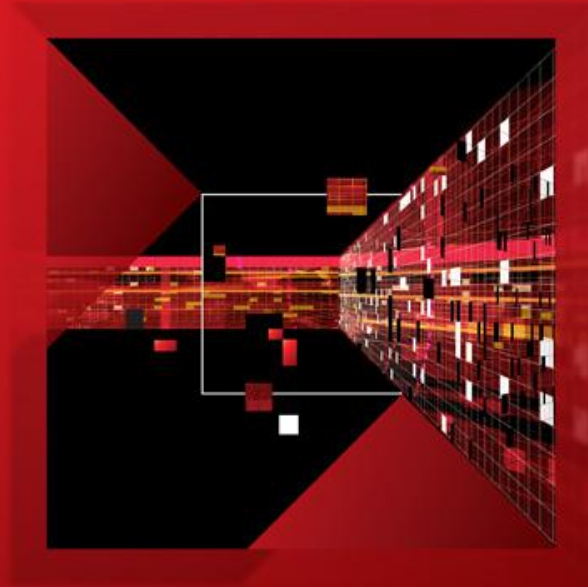
- Parallel Replica – (Voter, 1998)
  - Replicas of the system are run independently on multiple processors
  - First transition on any processor halts all runs
  - Accumulated time from all processors reflects correct dynamics

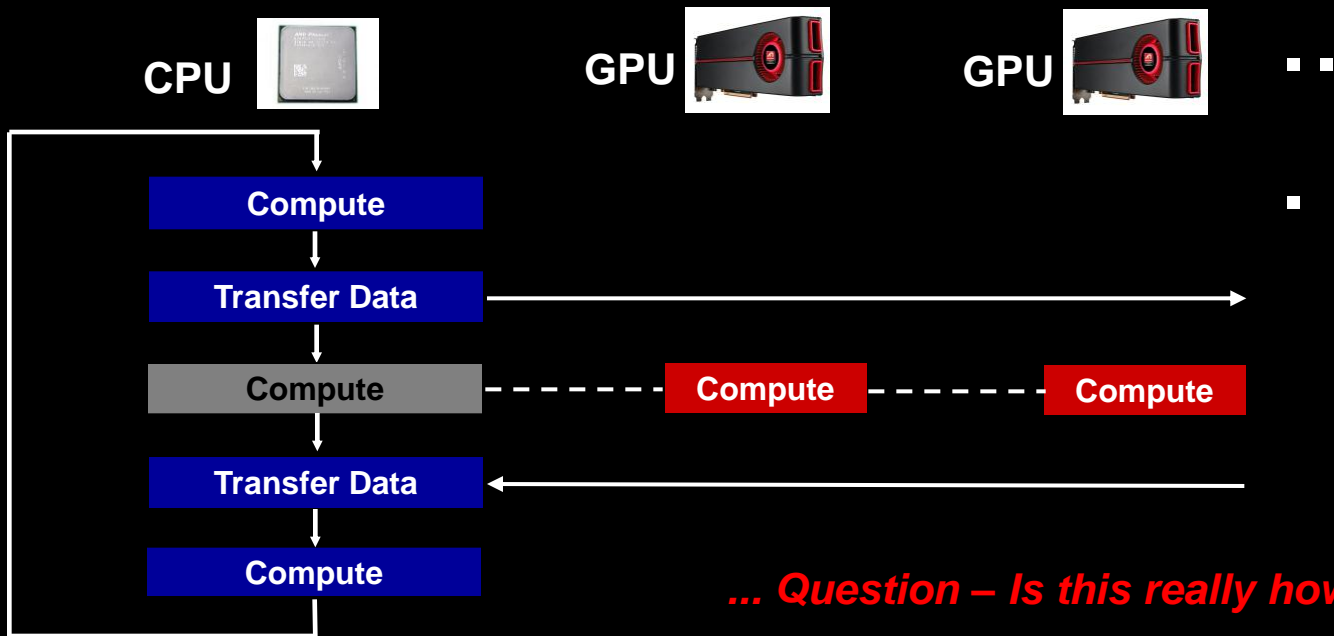


- Inter-particle force between atoms - for silicon use Stillinger-Weber three-body potential
  - The force calculation will dominate the computation time and be the focus of GPU optimization
- Propagators
  - Langevin – provides NVT ensemble simulating temperature with correct statistics
  - Parallel random number generator – LCG64 adapted from SPRNG
  - Velocity-Verlet – provides NVE for testing
- Transition detection
  - Steepest decent minimization (“stop and quench technique”)
  - Real-time multi-resolution analysis (use wavelets for detection)

*... objective is to move all computation to the OpenCL devices*

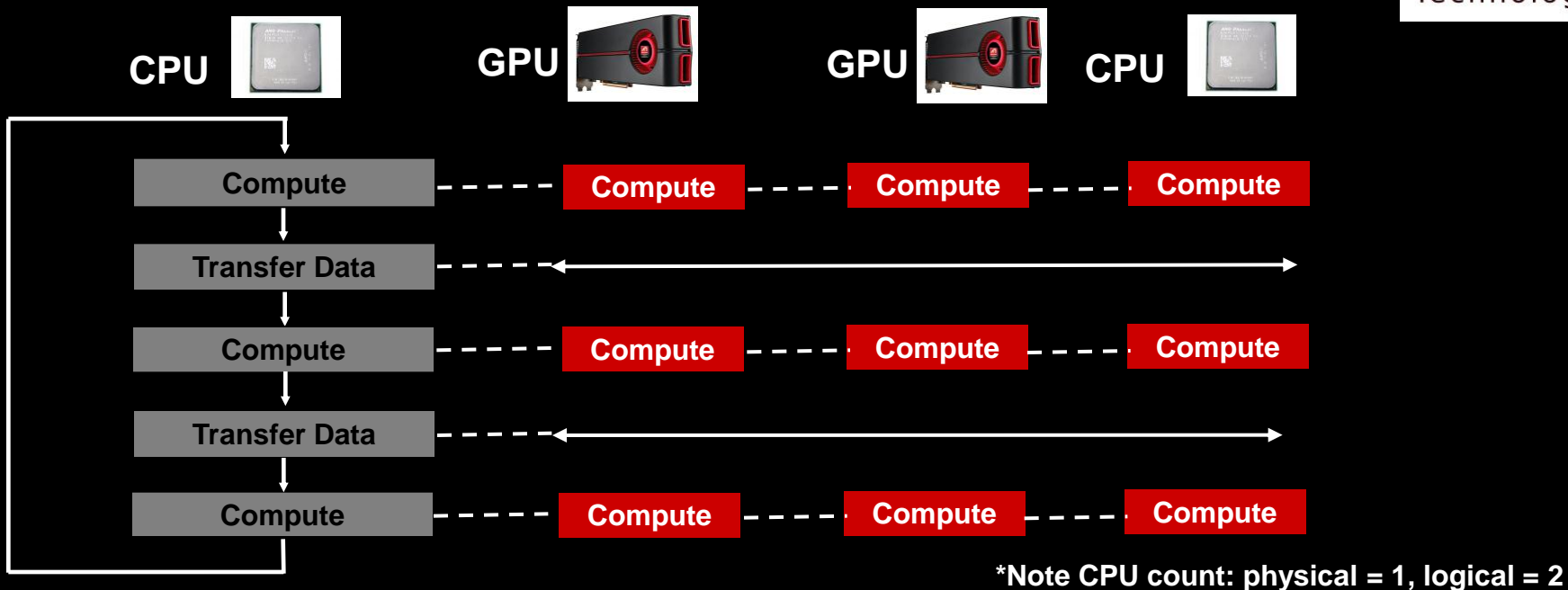
***Software Design for Hybrid  
CPU/GPU Architectures***





*... Question – Is this really how we should use GPUs?*

- Good - potential performance boost, easy way to get started (ok, maybe not so easy)
- Bad - Aamadah's Law will catch up with you
- Bad - Overhead can be substantial (data transfer, data structure conversion)

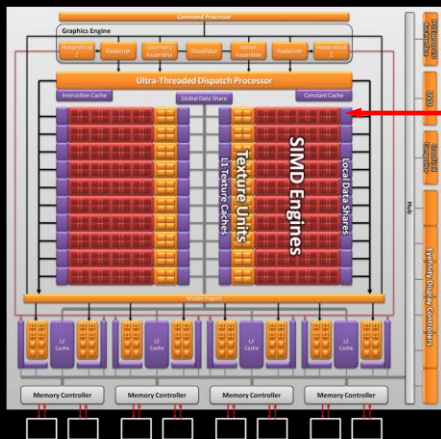


- Architect software for many-core GPU
- Multi-core host acts as operating system
- Same multi-core is “re-targeted” for compute

**... Use OpenCL for device-independent code**

# Using a Many-Core GPU Processor

... appears to be many, which one is the “core”?



20 SIMD Engines

## AMD Cypress Architecture



16 Stream Cores



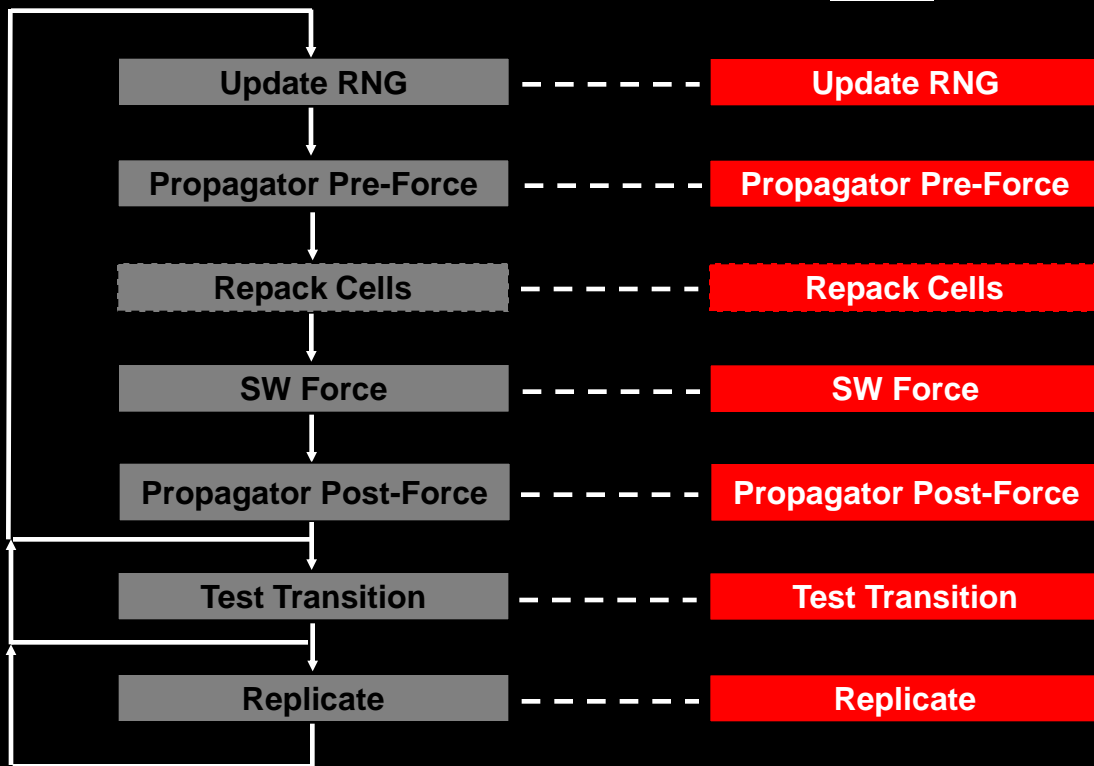
5 Processing  
Elements

- Are there 20, 320, or 1600 cores?
- View as 20 CPU “cores” with 16-way multi-threading and SSE for single-precision
- Use the 20 “cores” to run parallel replica ensembles

CPU

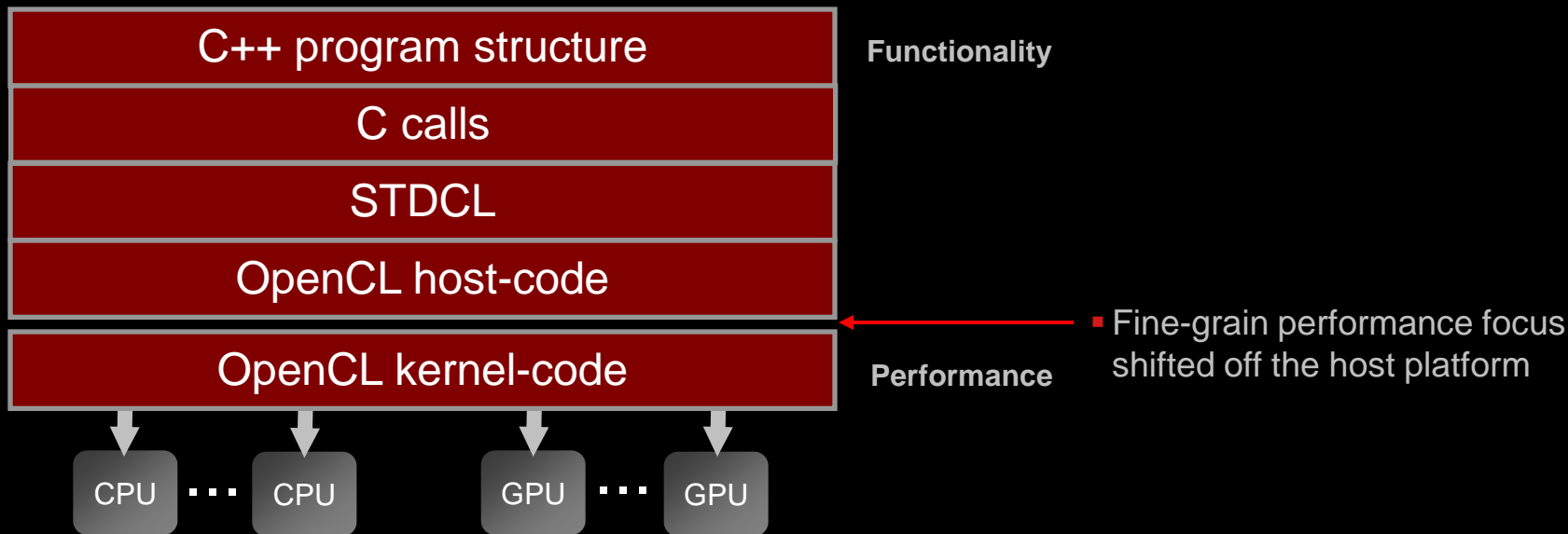


GPU



- Dynamics for all replicas are run for a fixed number of time steps
- Periodically test for transition
- Check for re-crossing
- Accept transition, replicate new state
- Run for a fixed number of steps to de-phase the replicas
- Repeat

- Software stack used for building the simulation code
  - C++ classes used for high-level structure, data containers, other code *not critical for performance*
  - *Performance-critical* host-side code uses STDCL C interface to OpenCL
  - Core algorithm performance/optimization addressed at the OpenCL kernel level

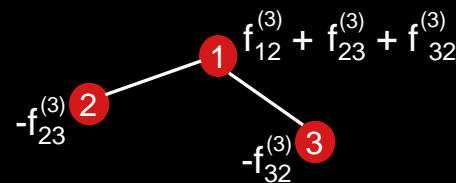
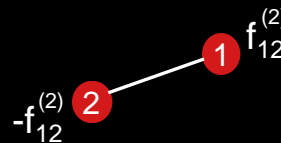


- OpenCL provides explicit, platform/device-independent control over execution and data movement
- In practice the syntax/semantics can be tedious
- Idea: provide simplified interface based on typical use cases with familiar UNIX/C syntax and semantics
- Lead to the development of the STDCL interface to OpenCL for high-performance computing
  - Implementation provided as part of the COPRTHR SDK (open-source GPL licensed)
  - Features include:
    - Default managed OpenCL contexts (stddev, stdcpu, stdgpu)
    - Dynamic OpenCL program loader support
    - OpenCL kernel and memory management
  - Introduces no restrictions on the use of “raw” OpenCL
  - Supports conventional memory allocation of device-sharable memory similar to malloc()
  - Supports non-blocking msync/fork/wait semantics – much simpler than OpenCL event management

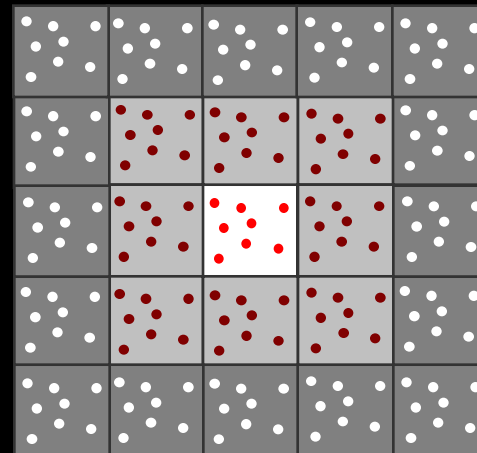
- Stillinger-Weber potential has the form:

$$V = \frac{1}{2} \sum_{ij} \phi(r_{ij}) + \sum_{ijk} g(r_{ij}) g(r_{ik}) \left( \cos \theta_{ijk} + \frac{1}{3} \right)^2$$

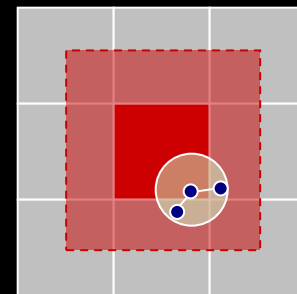
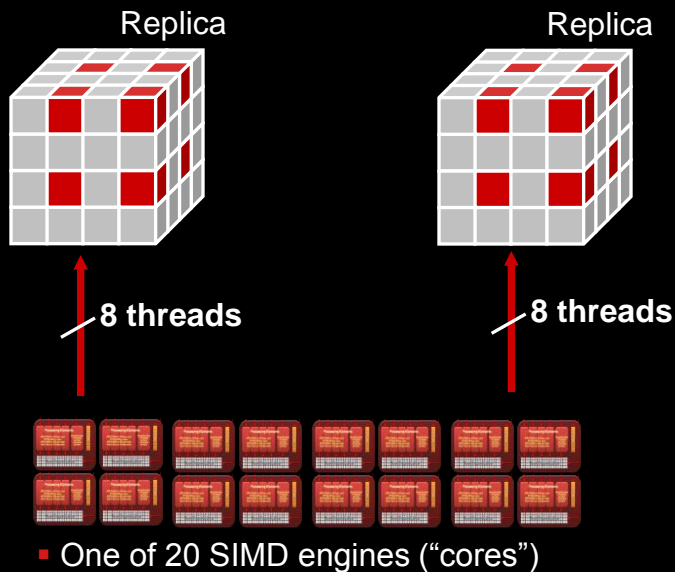
- Two-body term leads to a simple optimization:  $f_{12} = -f_{21}$ 
  - Can chose to exploit this, or not – factor of 2x in FLOPS
- Three-body term is more difficult:  $F_1 += f_{12} + f_{13}$ ,  $F_2 -= f_{12}$ ,  $F_3 -= f_{13}$ 
  - Difficult to exploit this on GPU – leads to bad scatter memory write



- Most MD codes, e.g., LAMMPS, are driven by NN list
  - $O(N^2)$  problem becomes  $O(N)$  problem
  - Leads to arbitrary global memory access, not good for GPU
  - This is not the only approach
- Group atoms into small cells large enough to ensure that each atom can only interact with other atoms in adjacent cells
  - Still provides the pre-screening needed for an  $O(n)$  algorithm
- For silicon the physics provides some order to this approach
  - Approximately 8 atoms per cell with NN cut-off less than cell size



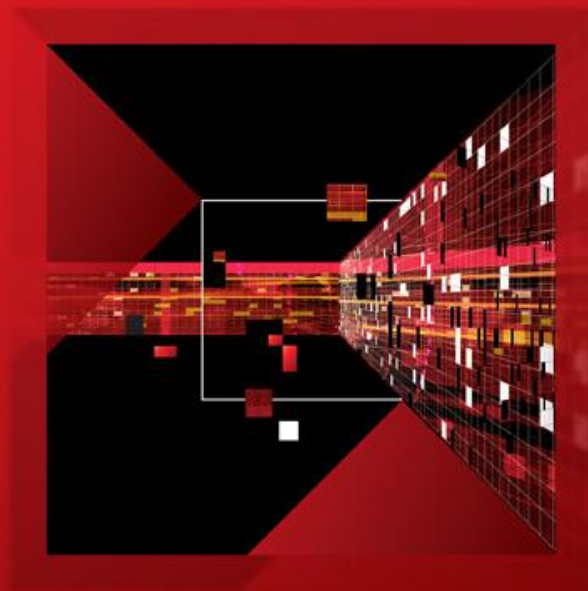
- Basic data structure is a grid of cells per replica
- Algorithm based on cell-cell interaction



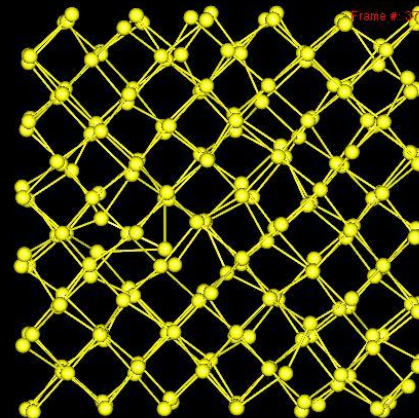
- Short-range radial cutoff imposes data collision limit

- For silicon, approximately 8 atoms per cell – dynamics changes number and cell association
- Small system, large replica count configuration - ~ 4096 atoms, 160 replicas (using 20 “cores”)
- For large systems, fewer replicas, completely scalable solution

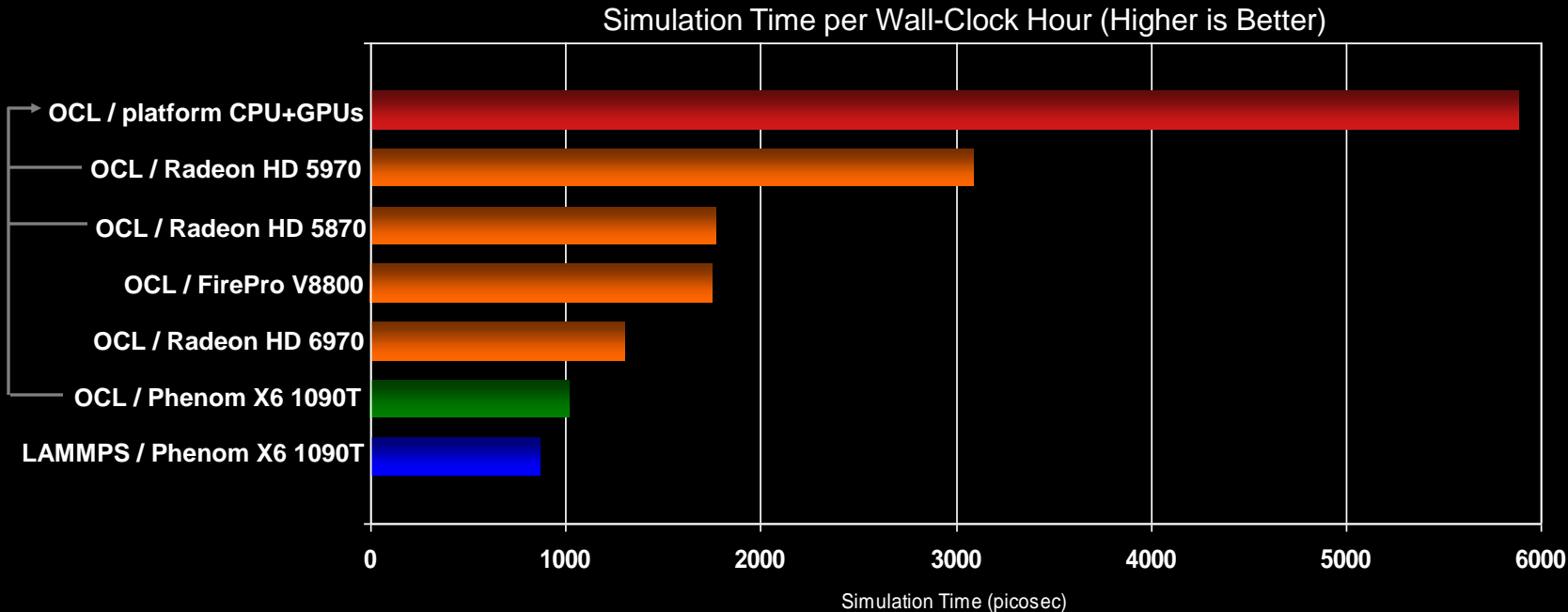
# *Performance Benchmarks*



- Platform used for development and testing was high-end desktop platform
  - ASUS M4A79T / 16 GB DDR3-1600
  - AMD Phenom II X6 1090T 3.2 GHz
  - AMD Radeon HD 6970
  - AMD FirePro HD V8800
  - AMD Radeon HD 5870
  - AMD Radeon HD 5970
- Software
  - CentOS 5.4 / GCC 4.1
  - AMD APP SDK 2.4 (OpenCL implementation)
  - BDT COPRTHR SDK 1.2 (STDCL implementation)
  - LAMMPS (March 11, 2011 snapshot)



- Silicon vacancy diffusion simulation
- *Entire simulation* run on an AMD Radeon 5870



- Plot shows approximate NVT simulation time accumulated in 1 hour of wall-clock time
- A few “gamer boards” increases the capability of the machine by 5x
- The same parallel code is used across the platform (CPU + GPUs)

- OpenCL enables the design of device-independent CPU/GPU simulation codes
  - Complete, scalable platform solution as opposed to “GPU as accelerator”
  - Write code once, run on any device, cross-device, cross-vendor
- Non-trivial example - parallel replica molecular dynamics with a three-body potential
  - ALL of the computation moved onto OpenCL devices
  - CPU and GPUs treated as peer processors, decomposition of work load, not functional tasks
- STDCL interface to OpenCL greatly simplifies the use of OpenCL in the design of HPC applications
- Benchmarks using a high-end desktop system with a 6-core Phenom and AMD GPUs:
  - OpenCL code on CPU outperforms reference code (LAMMPS) on CPU
  - OpenCL code on GPU outperforms CPU
  - Combined platform solution (CPU + GPUs) faster than CPU alone by more than 5x

## Disclaimer & Attribution

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions and typographical errors.

The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. There is no obligation to update or otherwise correct or revise this information. However, we reserve the right to revise this information and to make changes from time to time to the content hereof without obligation to notify any person of such revisions or changes.

NO REPRESENTATIONS OR WARRANTIES ARE MADE WITH RESPECT TO THE CONTENTS HEREOF AND NO RESPONSIBILITY IS ASSUMED FOR ANY INACCURACIES, ERRORS OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION.

ALL IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE ARE EXPRESSLY DISCLAIMED. IN NO EVENT WILL ANY LIABILITY TO ANY PERSON BE INCURRED FOR ANY DIRECT, INDIRECT, SPECIAL OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

AMD, the AMD arrow logo, and combinations thereof are trademarks of Advanced Micro Devices, Inc. All other names used in this presentation are for informational purposes only and may be trademarks of their respective owners.

The contents of this presentation were provided by individual(s) and/or company listed on the title page. The information and opinions presented in this presentation may not represent AMD's positions, strategies or opinions. Unless explicitly stated, AMD is not responsible for the content herein and no endorsements are implied.