FCCM 2004

Reconfigurable Molecular Dynamics Simulator

Navid Azizi, Ian Kuon, Aaron Egier, Ahmad Darabiha and Paul Chow
University of Toronto

Why is Molecular Dynamics interesting?

- Simulates interaction of atoms over time
- Many possible applications
  - Biomolecules
- Computationally intensive to handle >1000 atoms
- Large computer clusters used in the past
- Can a reconfigurable simulation system do this better?

What is Molecular Dynamics?

- Simulate using classical Newtonian mechanics

$$F = m\,a$$

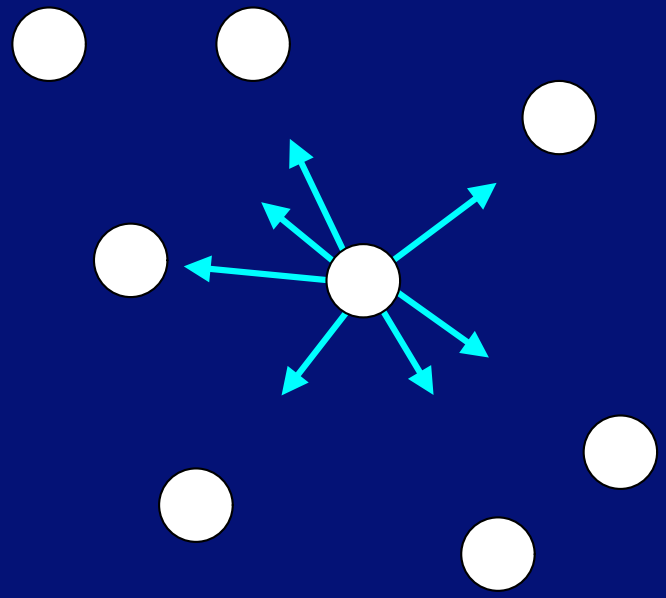
- Integrate acceleration to get position and velocity changes
- Use a very small timestep ~ 1 femtosecond

Molecular Dynamics Background

- Simulation Procedure per timestep
  - Sum force over all interacting atoms
  - Calculate acceleration
  - Integrate acceleration to update atom position and velocity
  - Repeat for all atoms

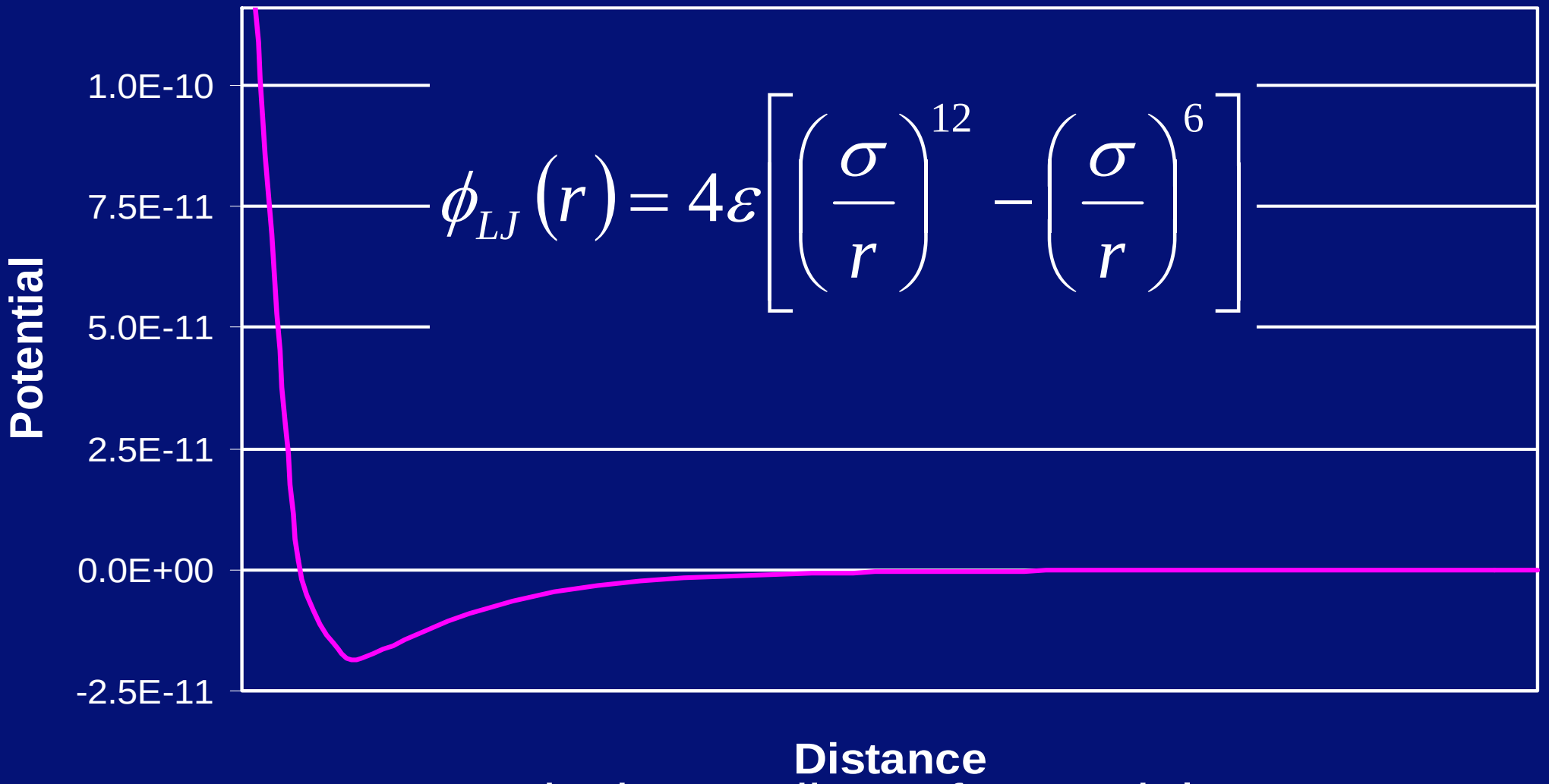
Background - Forces

- Two types of forces
  - Bonded – O(n)
    - No hardware acceleration required
  - Non Bonded – $O(n^2)$
    - Needs hardware acceleration



Background – Force Calculation

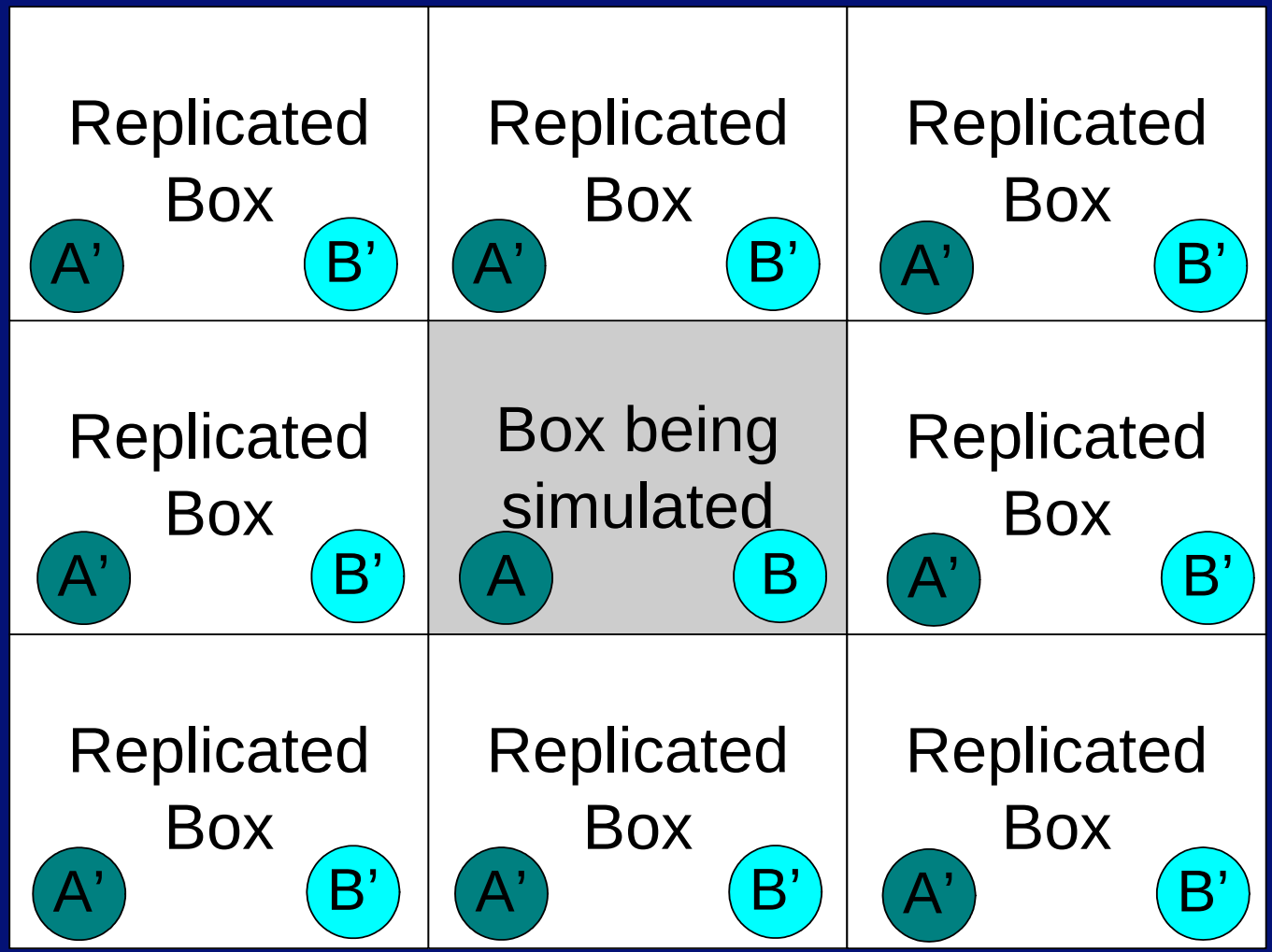
- Lennard-Jones (LJ) potential models interaction



- Force on a atom is the gradient of potential

Background – Simulating Large Volumes

- Any interesting volume has far too many atoms to simulate
- Solution – Periodic Boundary Conditions



Background – Time Integration

- Position and Velocity updated through time integration
  - Velocity Verlet Update rule used

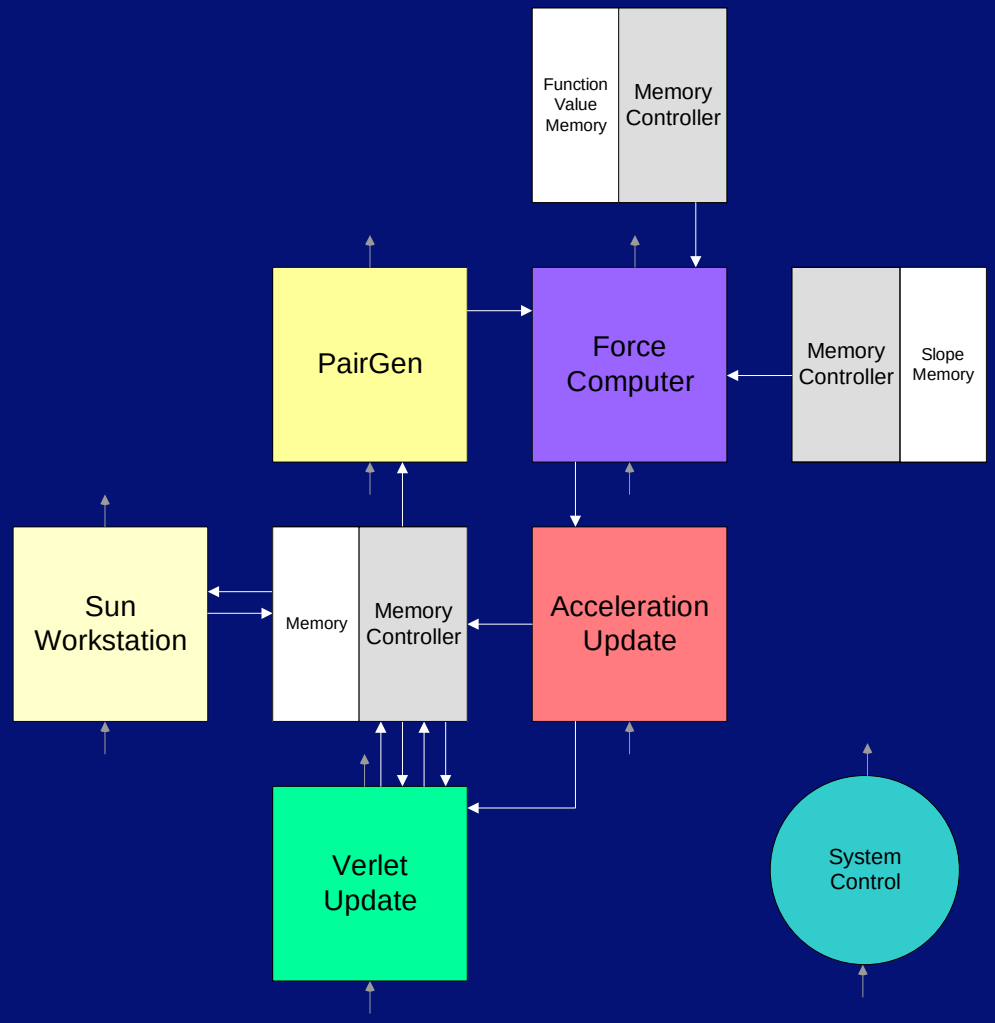
$$v(t) = v\left(t - \frac{\delta t}{2}\right) + \frac{\delta t}{2} a(t)$$

$$r(t + \delta t) = r(t) + \delta t \times v(t) + \frac{\delta t^2}{2} a(t)$$

$$v\left(t + \frac{\delta t}{2}\right) = v(t) + \frac{\delta t}{2} a(t)$$

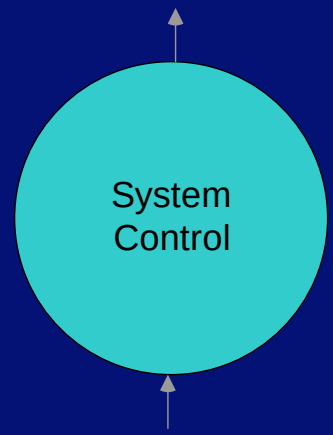
Architectural Design – System Overview

- Composed of 6 blocks
  - Pair Generator (PG)
  - Lennard-Jones Force Computer (LJFC)
  - Acceleration Update (AU)
  - Verlet Update (VU)
  - System Controller (SC)
  - Memory Controllers
- For each timestep
  - PG generates a pair of particles
  - LJFC computes force between particles
  - AU accumulates the forces
  - VU updates position and velocity



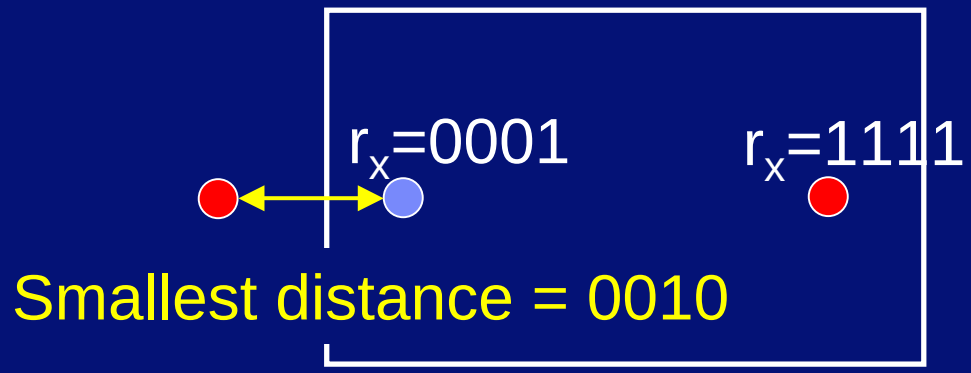
Architectural Design – System Controller

- Coordinates the activities of the different blocks during the simulation
- Determines the end of the timestep and signals start of new timestep
  - –Signals memory controllers to switch position banks to eliminate read after write hazards (more on this later)
- Serves as an Interface between the MD hardware system and the software modules on the Sun Workstation
  - –The Sun Workstation may signal the FPGA that it wants to read the memory contents
  - –After the completion of a complete timestep, the SC pauses the simulation



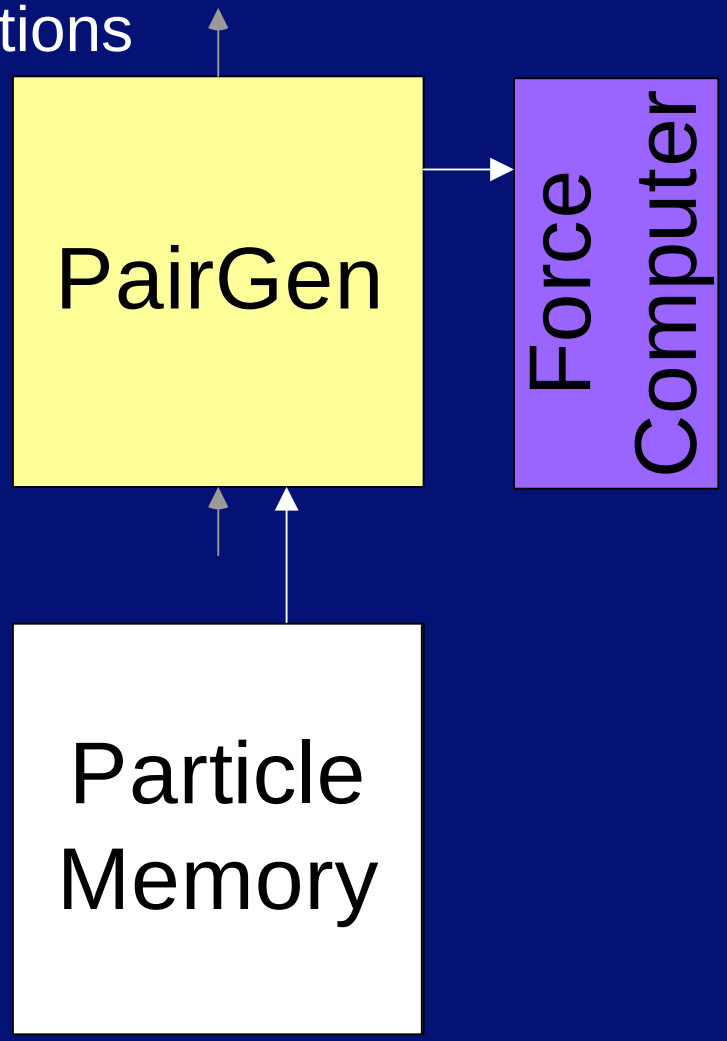
Architectural Design – Pair Generator

- Calculates distance between all atom pairs
- Must consider Periodic Boundary Conditions



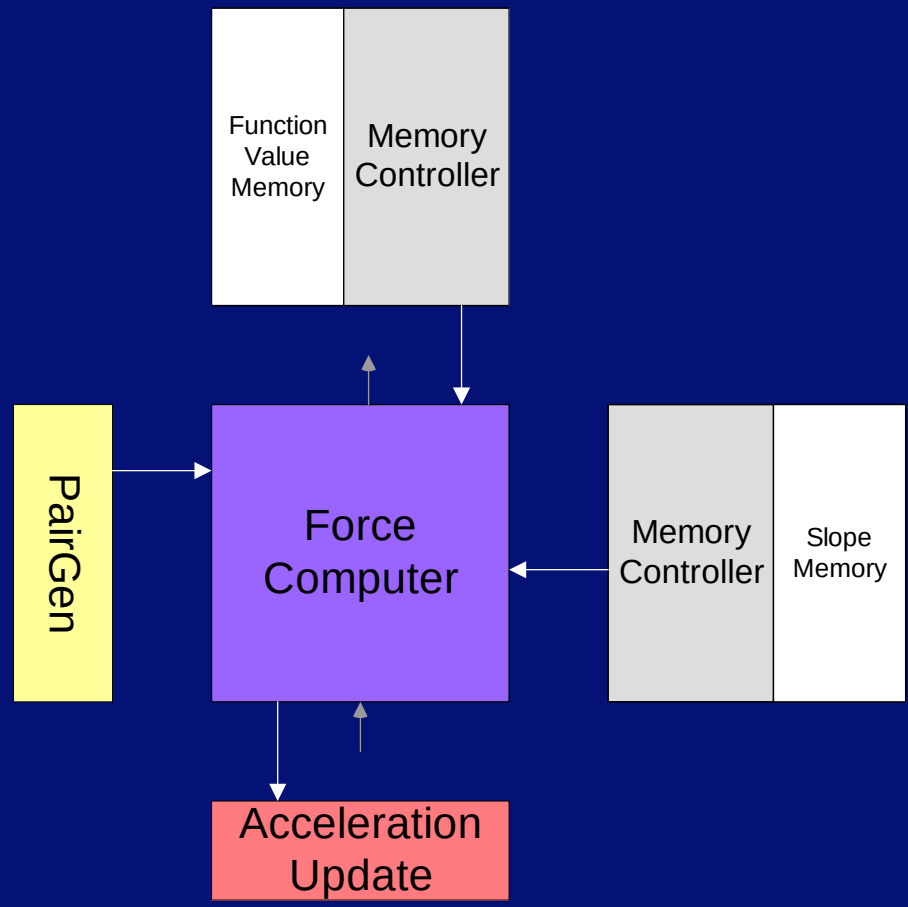
0001
-1111
0010

1111
-0001
1110



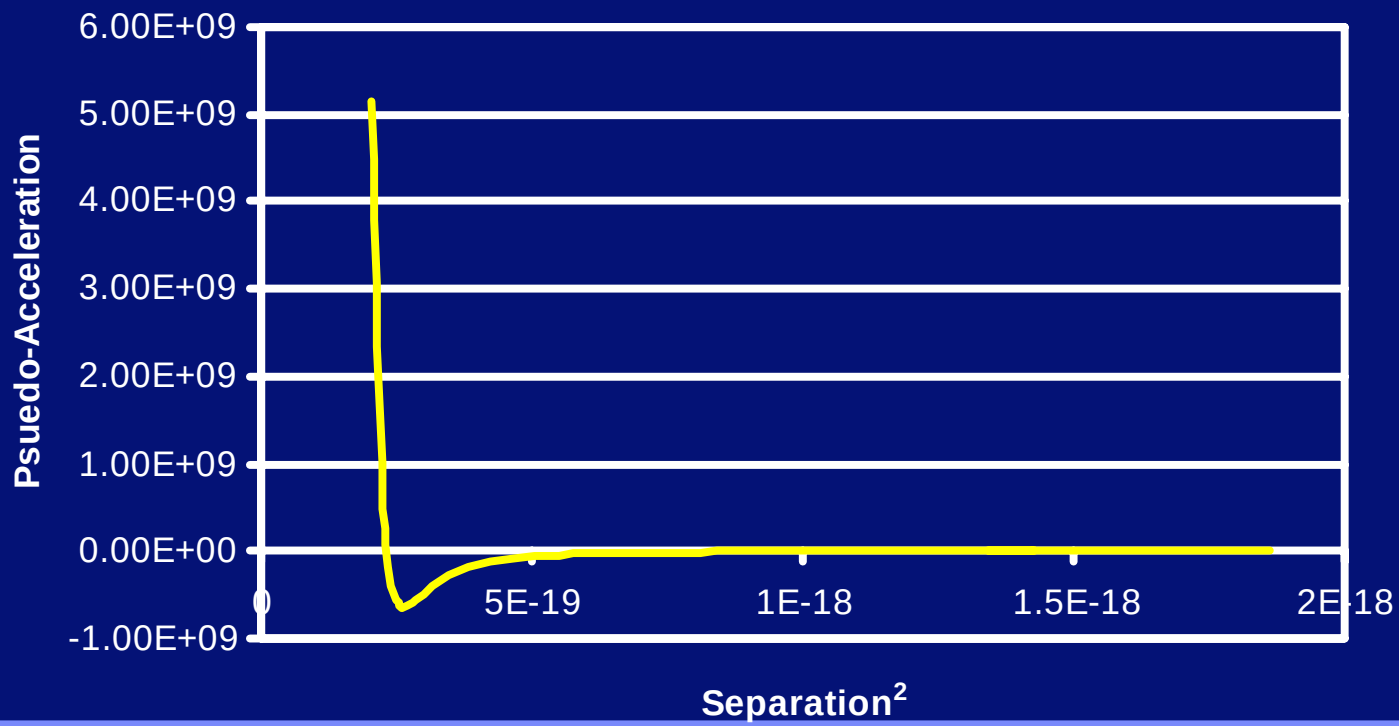
Architectural Design – LJ Force Calculator

- Uses r^2, obtained from PG, to look up value of the force and the slope of the force derivative
- Performs an interpolation to obtain a more accurate force magnitude
- Uses component distances, r_x, r_y, r_z, and force magnitude to calculate acceleration vector
- Sends the acceleration vector to the AU



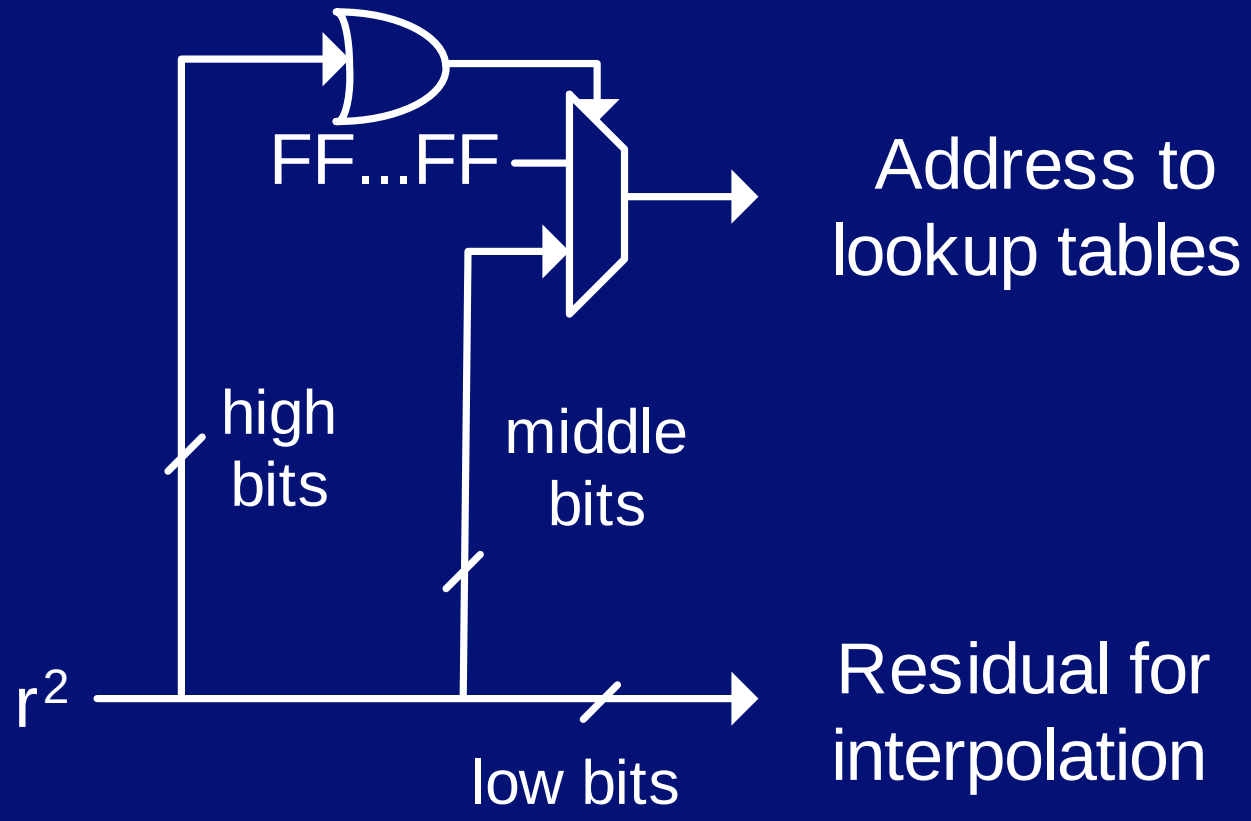
Architectural Design – LJ Lookup

- Use part of r^2 as index to into look up table
- Look up table has a 19-bit address, but r^2 is larger than 19-bits
  - –To choose which bits of to use, must look at characteristic of the LJ function
- Characteristic of LJ Function
  - –Changes rapidly near distances of 0
  - –At large distances the change in force is very little



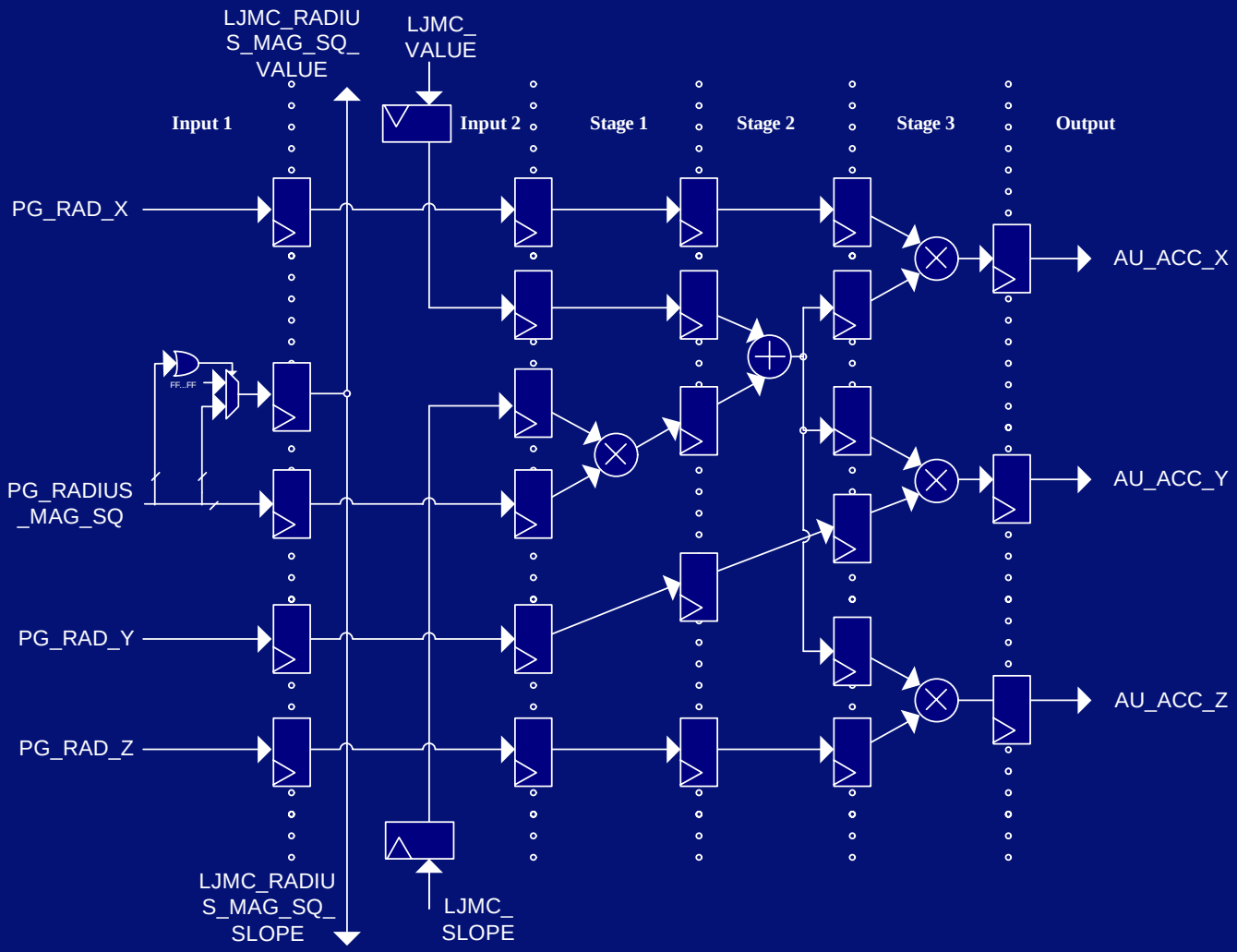
Architectural Design – LJ Lookup

- Use lower-order bits to interpolate
- Use the higher-order bits as a cutoff
  - –Since at very large distances, force is nearly 0
- Use middle-order bits to lookup force value and slope



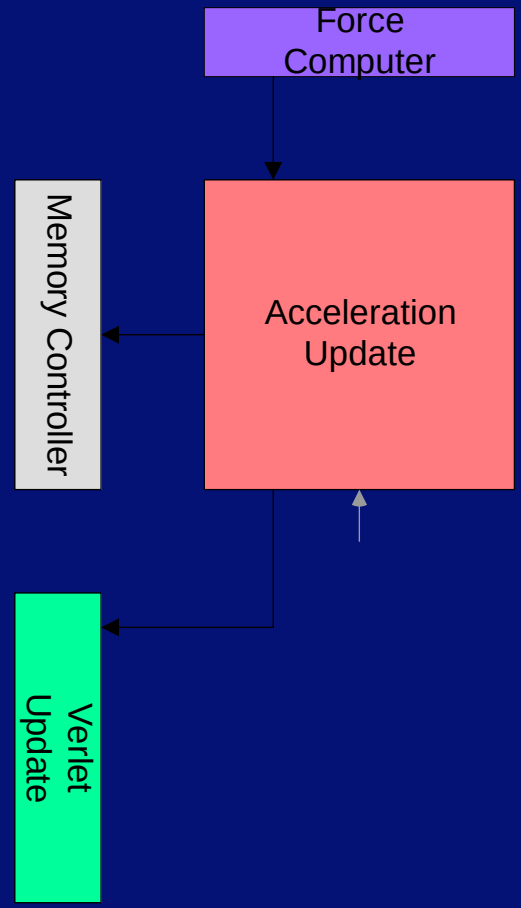
Architectural Design – LJ Force Calculator

- Multiply Slope by lower-order bits
- Add to function value
- Multiply by component distances to obtain pseudo-acceleration
  - –Look up table values contain multiplication by time to simplify downstream calculations



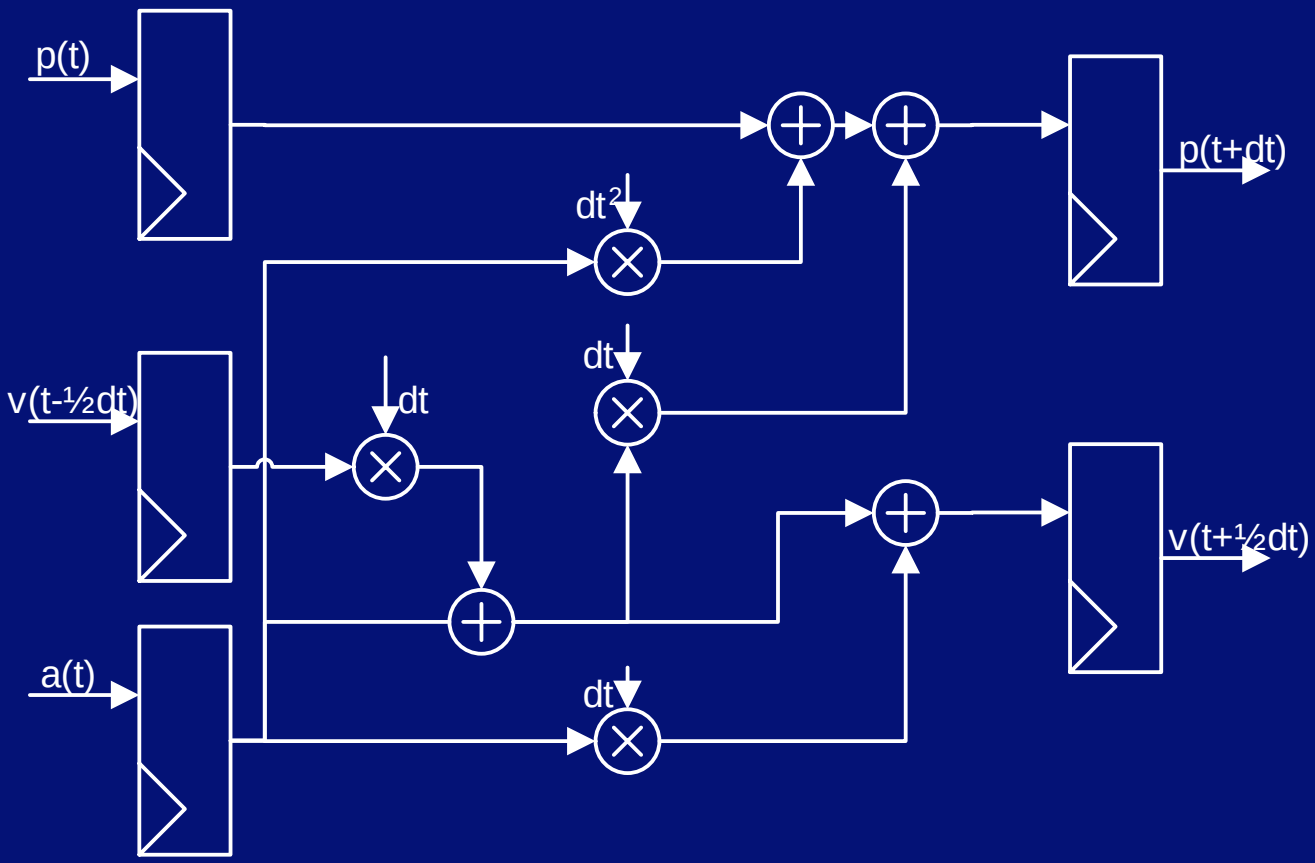
Architectural Design – Acceleration Update

- Receives incremental pseudo-accelerations from LJFC and to obtain total pseudo-acceleration
- When pseudo-acceleration is complete, send data to
  - Memory Controller to store in Memory
  - Verlet Update to calculate new velocity, position



Architectural Design – Velocity Verlet Block

- Not performance critical
- No pipelining in current implementation



Architectural Design – Memory Controllers

- Memory controllers are interfaces between off-chip SRAMs and the rest of the MD system.
- Memory access is a bottleneck in the current system.
- Two types of memory controllers:
  - *Memory Control (MC)*

    to access position, velocity and acceleration arrays in the primary SRAM.

  - *Lennard-Jones Memory Control (LJMC)*

    to access lookup tables containing value and slope of Lennard-Jones function.

Memory Controllers - MC

- MC receives requests from PG, VU and AU blocks
  - SRAMs on TM3 are single-ported, 64-bit wide
  - A priority scheme arbitrates between simultaneous memory access requests

Block Name	Read/Write	Target Array
PG	R	Position
VU	R & W	Position & Velocity
AU	W	Acceleration

- Each system-level Read/Write through MC consists of three 64-bit Read/Writes for X, Y and Z directions in series.

- 7-clock cycle system-level read
- 6-clock cycle system-level write

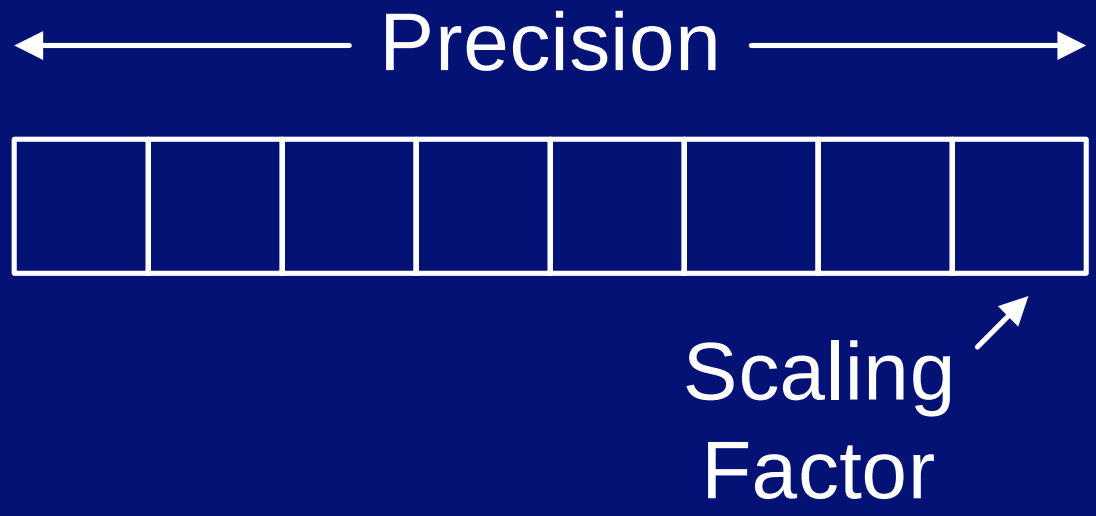
Memory Controllers - LJMC

- Two instantiations of LJMC used for Lennard-Jones Force Calculation:
  - To access Lennard-Jones function value lookup table
  - To access Lennard-Jones slope lookup table
- Only one source of request for memory access, no arbitration is required.
- Only read requests
- Each system-level read from LJMC is one single read and takes 4 clock cycles.

Precision and Scaling Factors

- Architecture uses integer operations to reduce complexity

- **Precision**: number of bits used to represent a value
- **Scaling Factor**: the weight of the least significant bit of the value



Precision and Scaling Factors – cont.

- To choose appropriate scaling values & precision
  - –Calculations made with particles at varying distances
  - –Found absolute minimum and absolute maximum for different values along our datapath
  - –Scaling Factor was found by taking $\log_2$ of the minimum value
  - –Precision was found by taking $\log_2$ of the difference between minimum and maximum
  - –Extra bits were placed to the upper and lower portions of most values

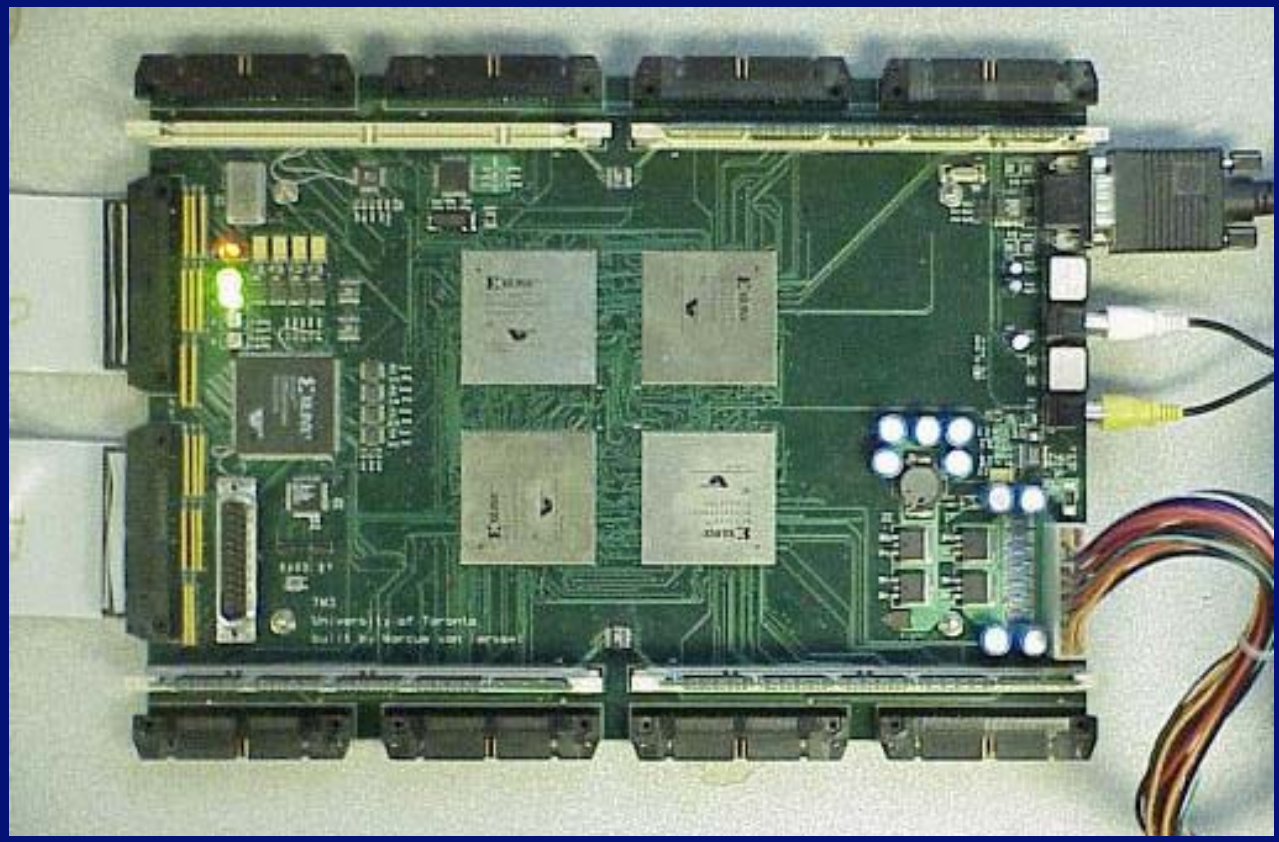
Quantity	Scaling Factor	Precision
Position	2^{-64}	38
Velocity	2^{-15}	51
Acceleration	2^{-64}	37

Simulation Environment is Configurable

- Simulation reconfigurability
  - Change precision, scaling factors, number of atoms. forces
  - No wasted hardware
  - No time overhead when precision is reduced
- Entire process is automated
  - One input file controls entire process
    - Hardware
      - C program creates appropriate VHDL
    - Software interface, Software initialization
      - Always match the hardware

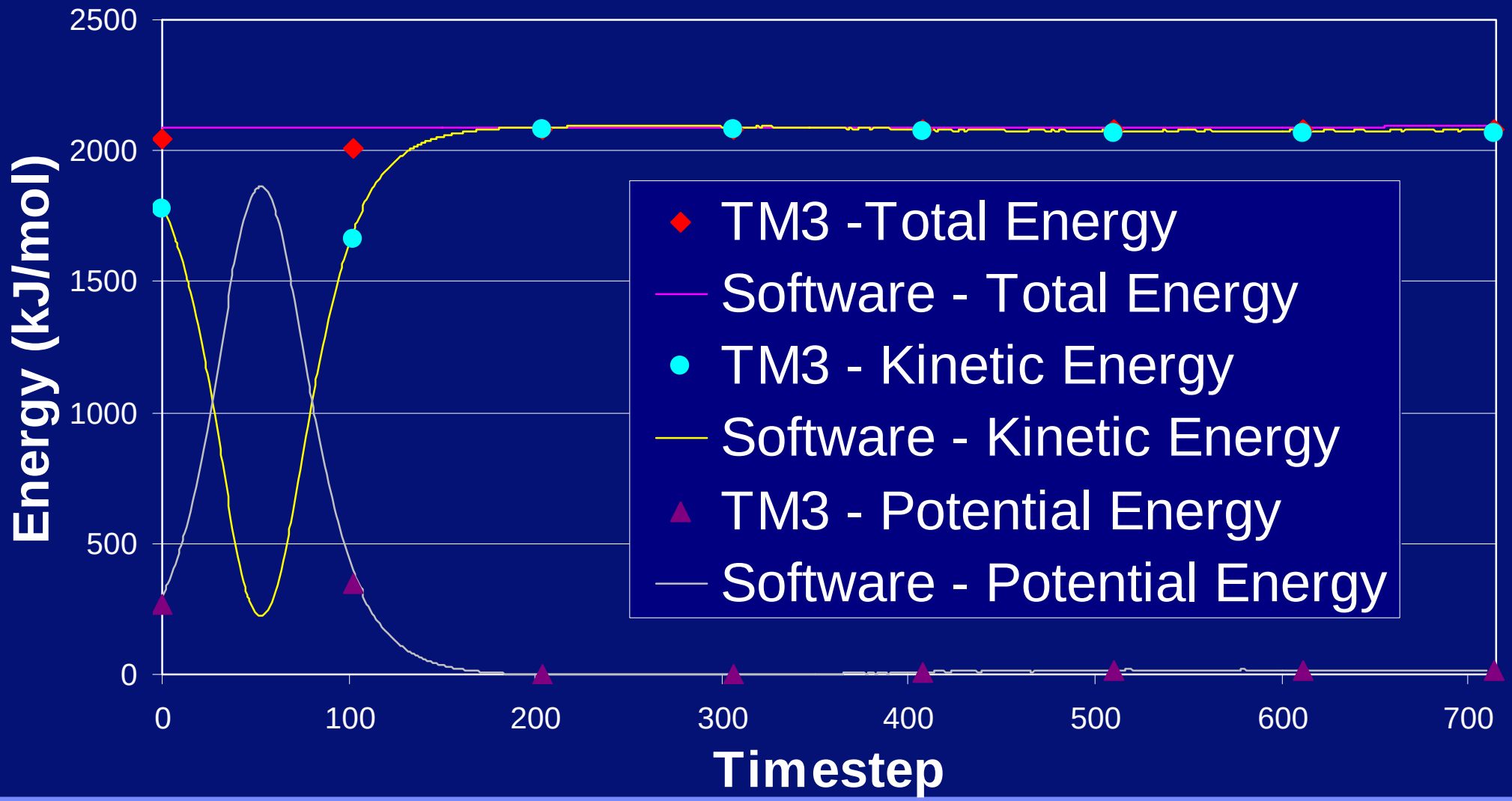
Implementation

- Used the Transmogrifier 3
  - 4 interconnected Virtex-E 2000's
  - 2MB memories connected to each Virtex-E
  - Slow by today's standards



Verification

- Tested accuracy of implementation
  - Compared TM3 results with software



System Performance

- For a 8192 atom MD system running on the TM3
  - Frequency: 26 MHz
  - Timestep Duration: 37 sec
- For a 8192 atom software system running on a 2.4GHz Pentium 4
  - Timestep Duration: 10.8 sec
- MD system is 3.4X slower than software

How to improve this?

- Memory
- New FPGA
- Parallelism

Memory Requirements of MD System

- Acceleration Array (8192 atom system uses 0.17 MB)
- Velocity Array (8192 atom system uses 0.17 MB)
- Position Array (8192 atom system uses 0.34 MB)
- Lookup Tables: 2 MB

Improving Performance – Memory Organization

- On TM3 there is only one external SRAM per FPGA
- Single SRAM for all atom information causes large slowdowns
  - Handshaking
  - Serial reads for x, y and z
  - Hardware issues

Better memory system

2.1 seconds/timestep (5X faster than software)

Improving Performance – Memory Organization

- On TM3 there is only one external SRAM per FPGA
  - 1 SRAM used to store positions, velocities, accelerations
  - 1 SRAM used to store function lookup table
  - 1 SRAM used to store slope lookup table

- The SRAM used to store positions, velocities, accelerations caused large slowdowns
  - Factor of 3 slowdown due to handshaking that is needed when arbitrating accesses to a single memory array, which semantically are different arrays
  - Factor of 3 slowdown because the width of the data bus was too small, so the x, y, and z components of position/velocity/acceleration were read serially
  - Factor of 2 slowdown due to wait states required by the memory system (a TM3 issue) between each access

Having a better memory system
Allows for a speedup of 18
2.1 second timestep (5X faster than software)

Not increasing the amount of memory

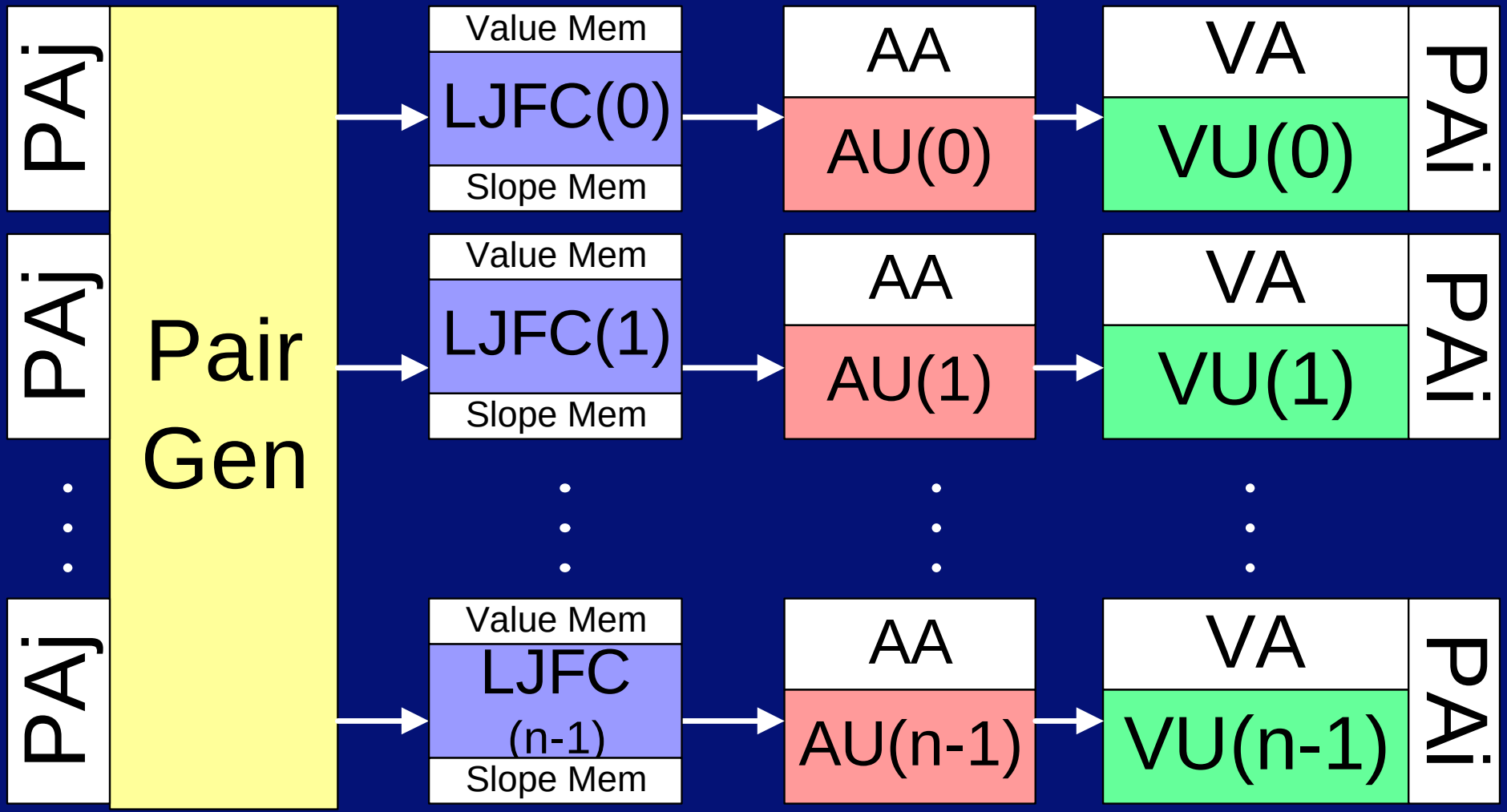
Improving Performance – Clock Speed

- Run on modern FPGA
- All possible improvements for clock speed not explored
- Expect a factor of 4 increase to a 100MHz

Better memory system + Faster Clock Speed

0.51 seconds/timestep (21X faster than software)

Improving Performance – Parallel Architecture

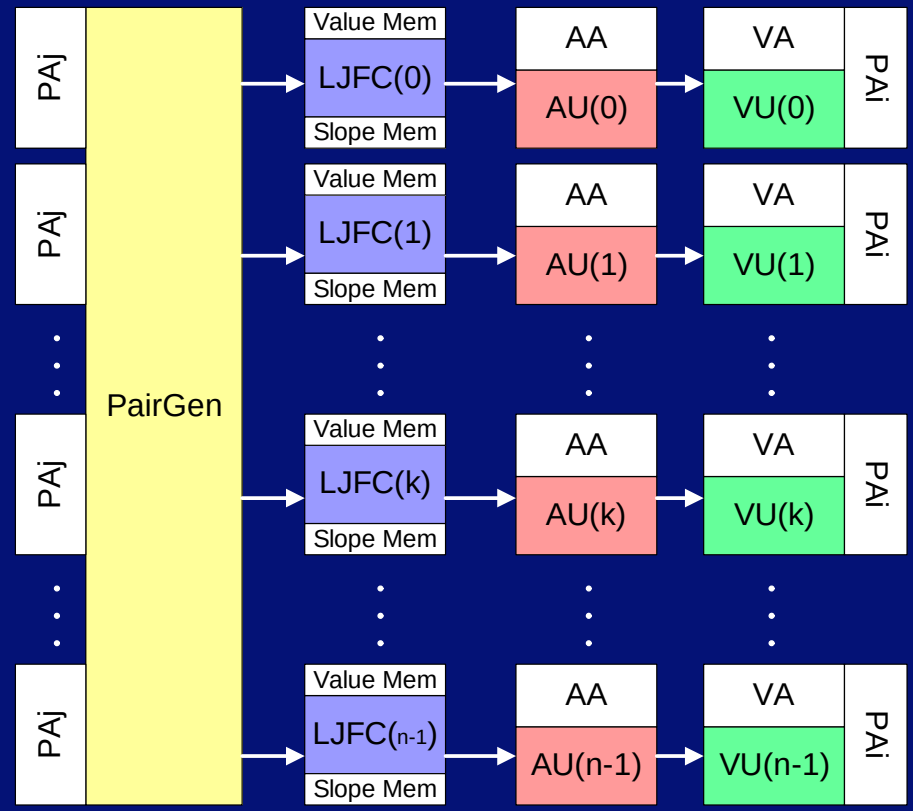


Better memory system + Faster Clock Speed + Parallelize

0.51/n seconds/timestep (21n X faster than software)

Improving Performance – Parallel Architecture

- Multiple MD engines are placed in parallel
- Logic Requirements
  - –LJFC, AU, VU do not need to be changed
    - –For each pipeline, one LJFC, AU, VU fit on one FPGA
  - –PG
    - –More sophisticated to send different pairs of particles to different pipelines
- Memory Requirements
  - –3 additional 512K x 32 memory chips are needed for each pipeline
  - –No increase in memory used to store position, velocity or acceleration as they are distributed throughout the system



Better memory system + Faster Clock Speed + Parallelize

0.51/n second timestep (21n X faster than software)

Cost, Power Benefits

- Performance
  - MD system can deliver a 21X performance benefit over software
  - Assume a conservative 10X performance advantage
- Cost
  - Microprocessor motherboard-sized board can fit 4 FPGAs
  - 4 FPGAs (each $200) + board + SRAM + misc. ~ $1500
  - Microprocessor Motherboard + CPU + DRAM ~ $1500

Comparison of MD Simulator and Supercomputers

Per Board	1 Pentium + 1 GB DRAM	4 FPGAs + 24 MB SRAM
Performance	1X	40X
Cost	~$1500	~$1500
Power	106W	40W

Metric	Improvement of MD System
Performance/Power	100X Improvement
Performance/Cost	40X Improvement
Performance/Space	40X Improvement

Conclusions

- Easily reconfigurable MD System designed
- Molecular dynamics simulation can be done on FPGAs
- Simple enhancements will improve speed
  - Power, Cost and Space savings over software

Future Work

- Improve accuracy
- Target newer FPGA platform
- Support new forces

Acknowledgements

- Funding for the TM3 Project was provided by Micronet and Xilinx