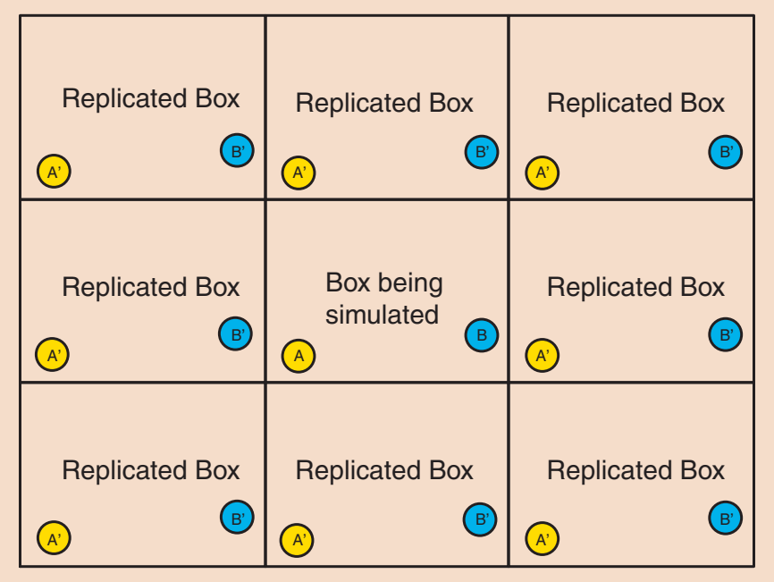
FPGA-Based Supercomputing: An Implementation for Molecular Dynamics

Ian Kuon, Navid Azizi, Ahmad Darabiha,Aaron Egier, and Paul Chow
Deparment of ECE, University of Toronto, Toronto, Ontario Canada
{ikuon,nazizi,ahmadd,aegier,pc}@eecg.utoronto.ca

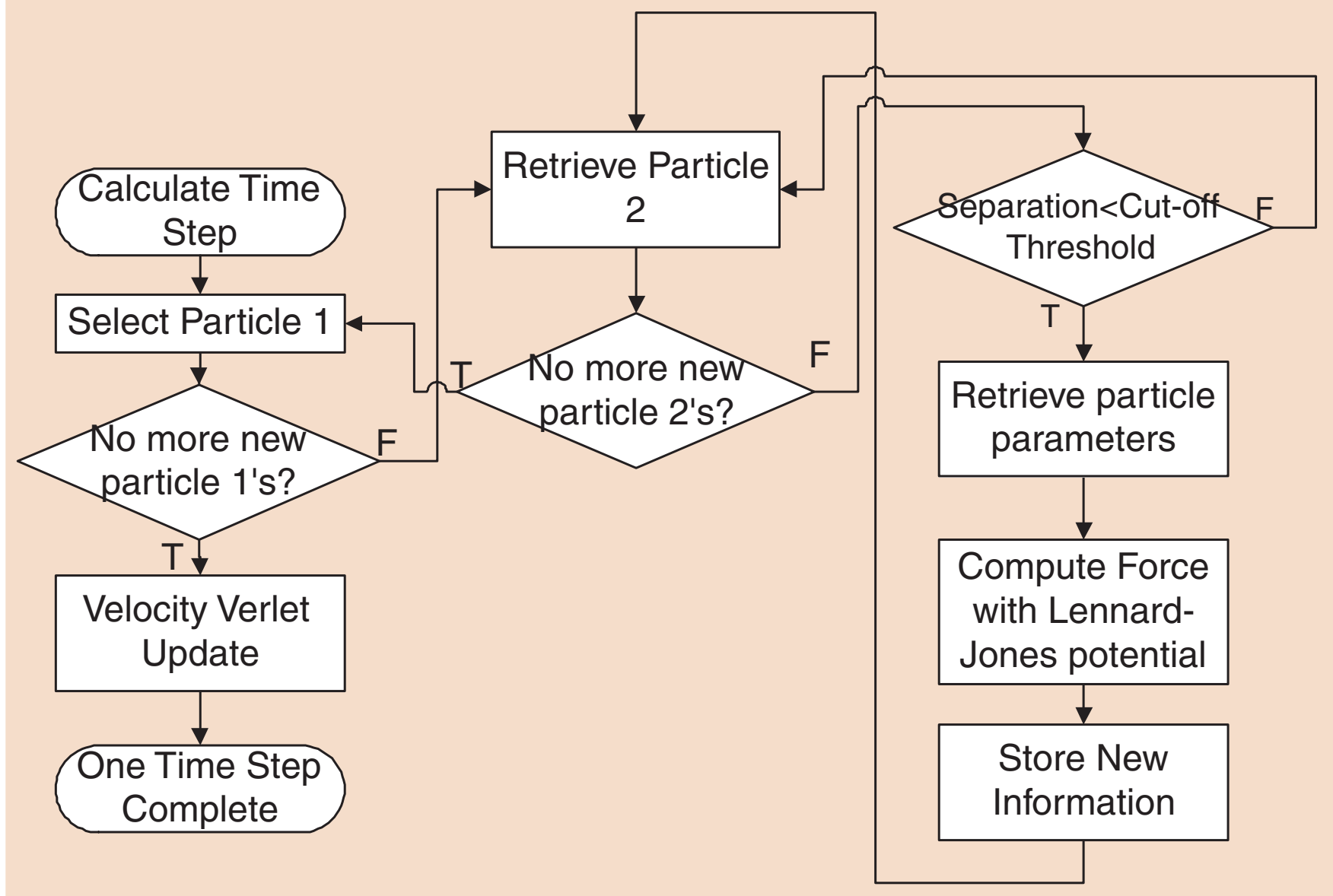
Molecular Dynamics

- Simulates atom and molecule interactions with Newtonian mechanics
- Applications - defect analysis, protein folding
- Any interesting volume has far too many particles to simulate
- Use periodic boundary conditions to reduce number of particles that must be tracked



- a cube of particles is simulated
- cube is replicated in every dimension
- a particle on the edge of the cube interacts with its *neighbour* on the other side of the cube

Simulation Process


Calculate Time Step
Select Particle 1
No more new particle 1's?
T
F
Velocity Verlet Update
One Time Step Complete
Retrieve Particle 2
No more new particle 2's?
T
F
Separation<Cut-off Threshold
T
F
Retrieve particle parameters
Compute Force with Lennard-Jones potential
Store New Information

- Model intermolecular forces using Lennard-Jones potential

$$F = -\nabla_r \phi_{LJ}(r) \qquad \phi_{LJ}(r) = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]$$

- Forces must be calculated for each pairwise interaction between particles
  - problem becomes $O(n^2)$
  - for simplicity insignificant interactions can be discarded
- Acceleration due to net force is calculated using Newton's second law F=ma
- Must perform time integration to get position and velocity updates.
- Use Velocity Verlet Update rule to perform this Integration

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

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

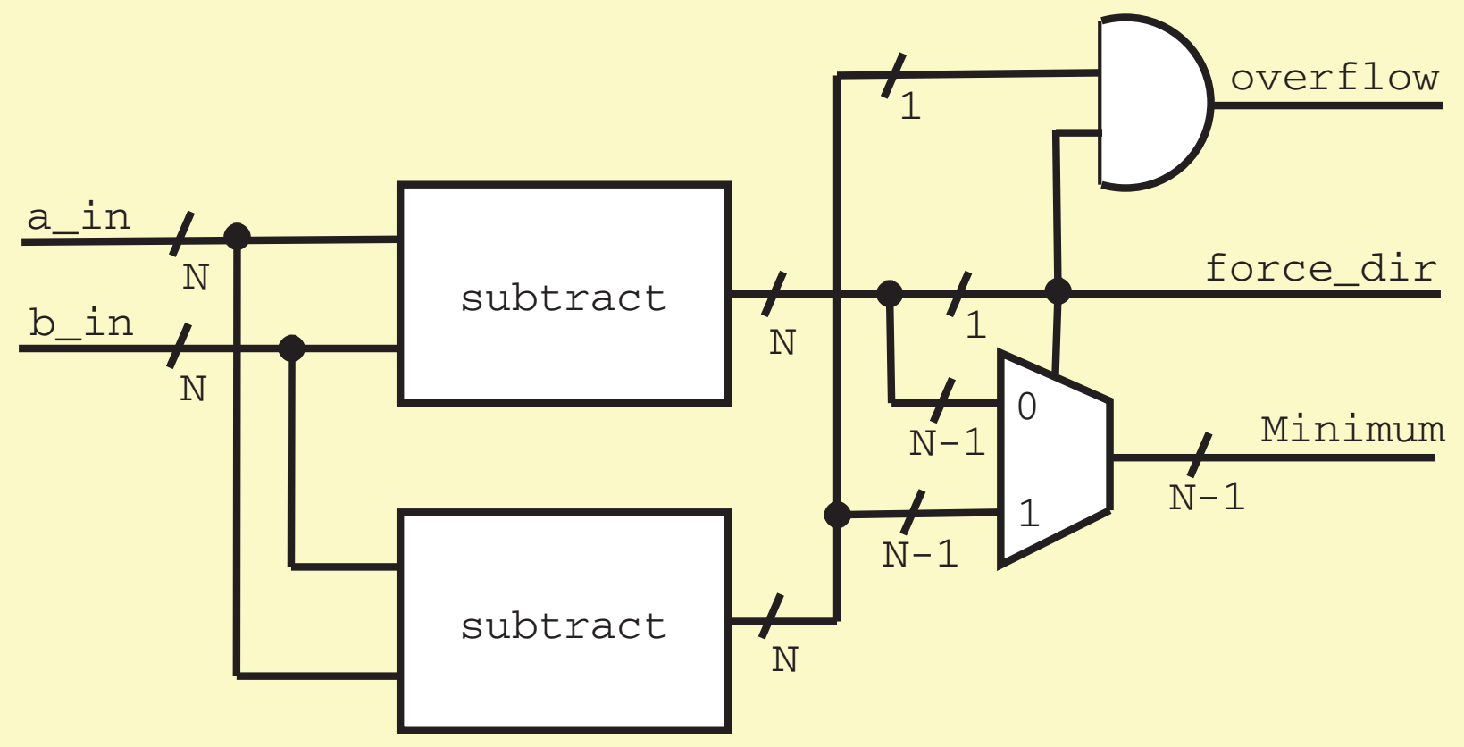
Previous Hardware

- no known previous implementations used programmable logic
- MODEL chip - floating point throughout
  - 76 MODEL chips are 50 times faster than a 200 MHz Sun Ultra 2
- MD-GRAPE - fixed and floating point design
  - Communication with host computer slows simulation speed

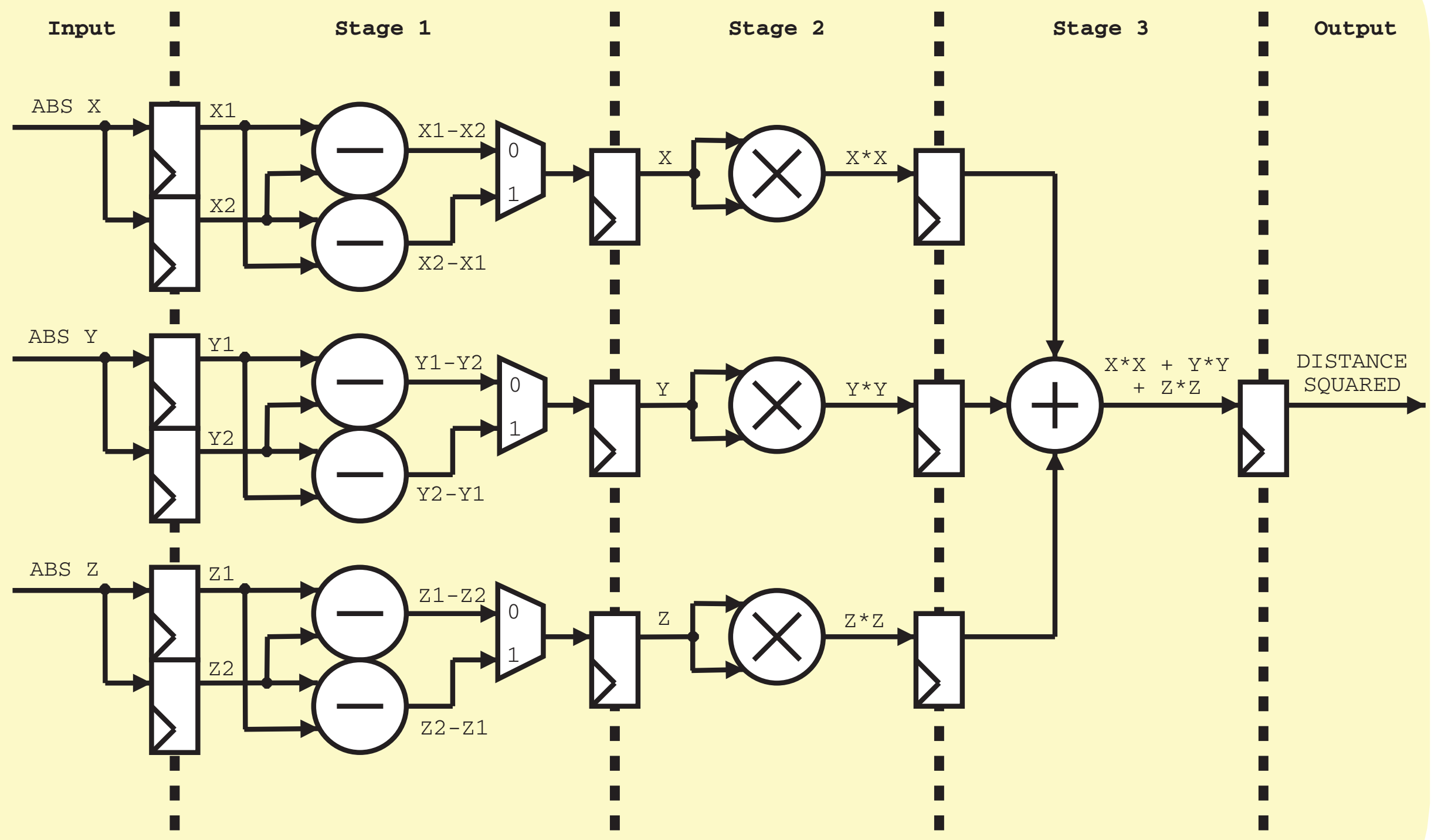
Pair Generator

- Retrieves 3-D co-ordinates for all pairs of particles from memory.
- Computes the distance squared between all pairs of particles and the force direction.
- Distance squared between two particles calculated using the formula:

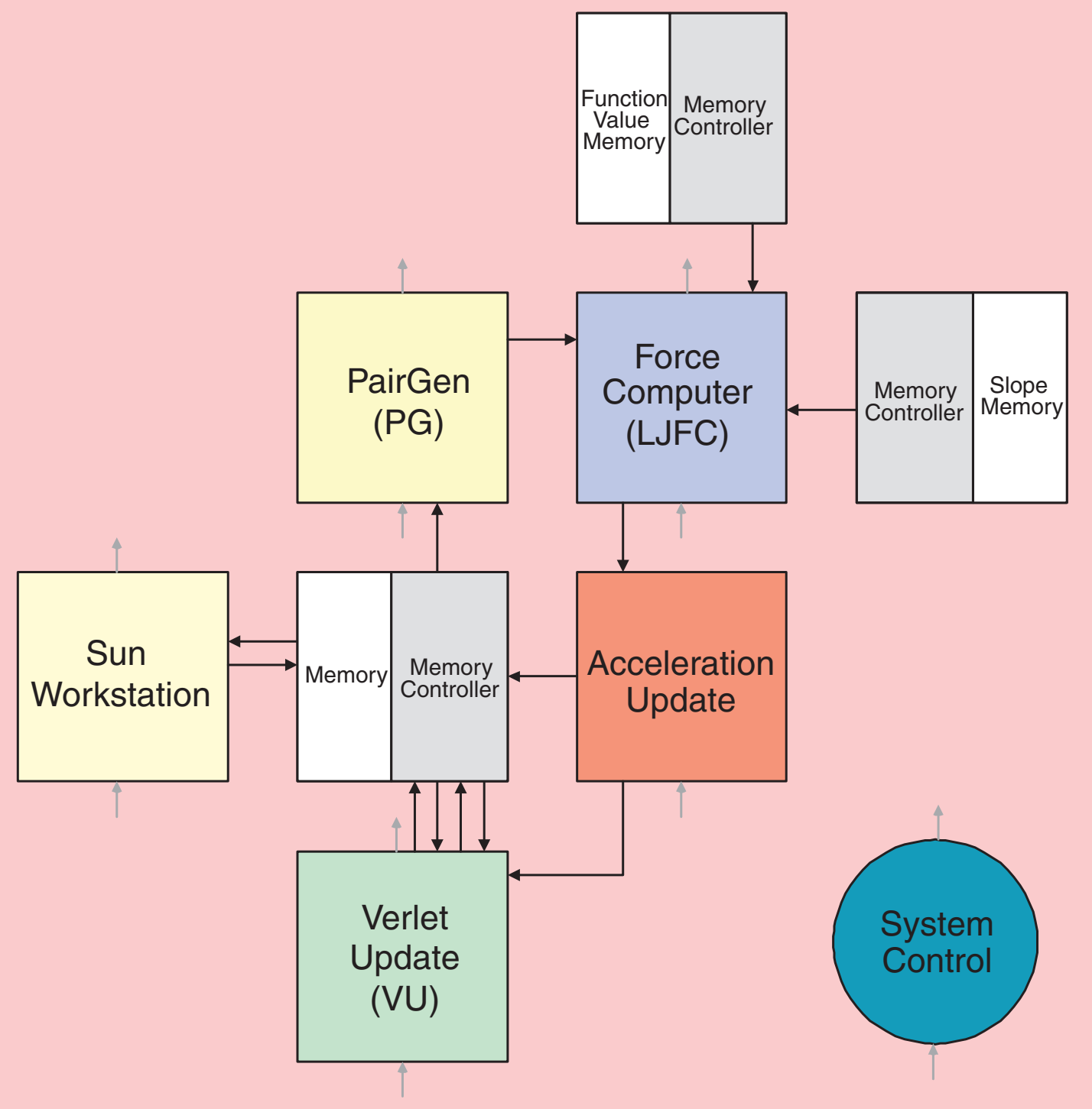
$$\min(x1-x2, x2-x1)^2 + \min(y1-y2, y2-y1)^2 + \min(z1-z2, z2-z1)^2$$



- By using unsigned numbers in the Minimum operation the Periodic Boundary Conditions are accounted for



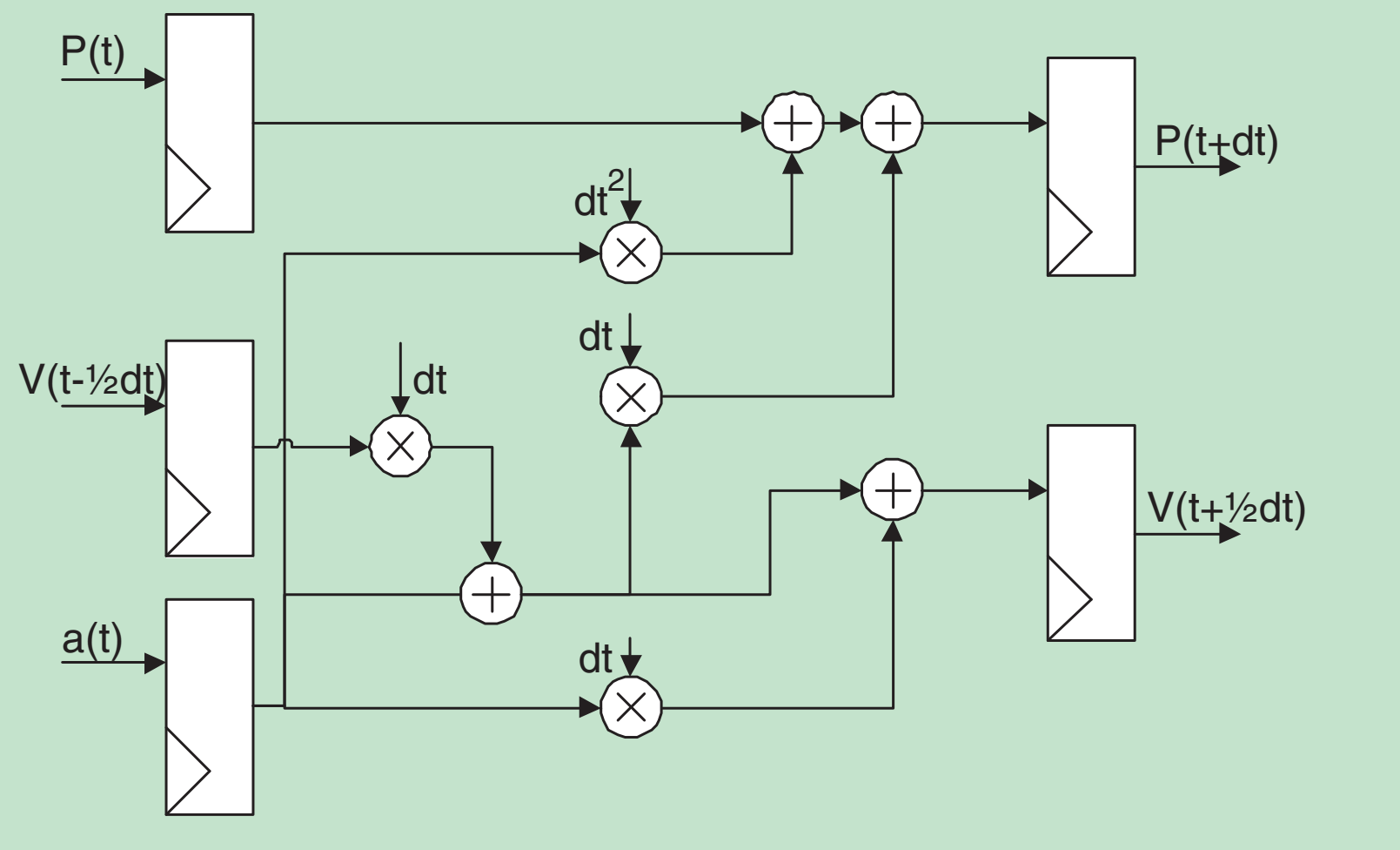
System Overview



For every timestep in the simulation

1. PG examines a pair of particles and determines the distance between the pair.
2. LJFC computes the force between the pair
3. AU computes the total acceleration on a particle from the series of incremental forces.
4. VU uses the total acceleration to update the position and velocity

Verlet Update



- Uses acceleration to perform time integration and update velocity and position
  - Performed once per particle
- Conventionally two separate steps are performed to obtain a half timestep velocity
- For hardware implementation this is inefficient Instead a single combined operation is performed
  - however, velocity stored in memory is a half timestep ahead of current time
- Multiplication is simplified by pre-multiplying acceleration in lookup table by the timestep

Software

- Constants Package
  - Generates shared constants for hardware and software
  - Includes precision, scaling factors, number of particles

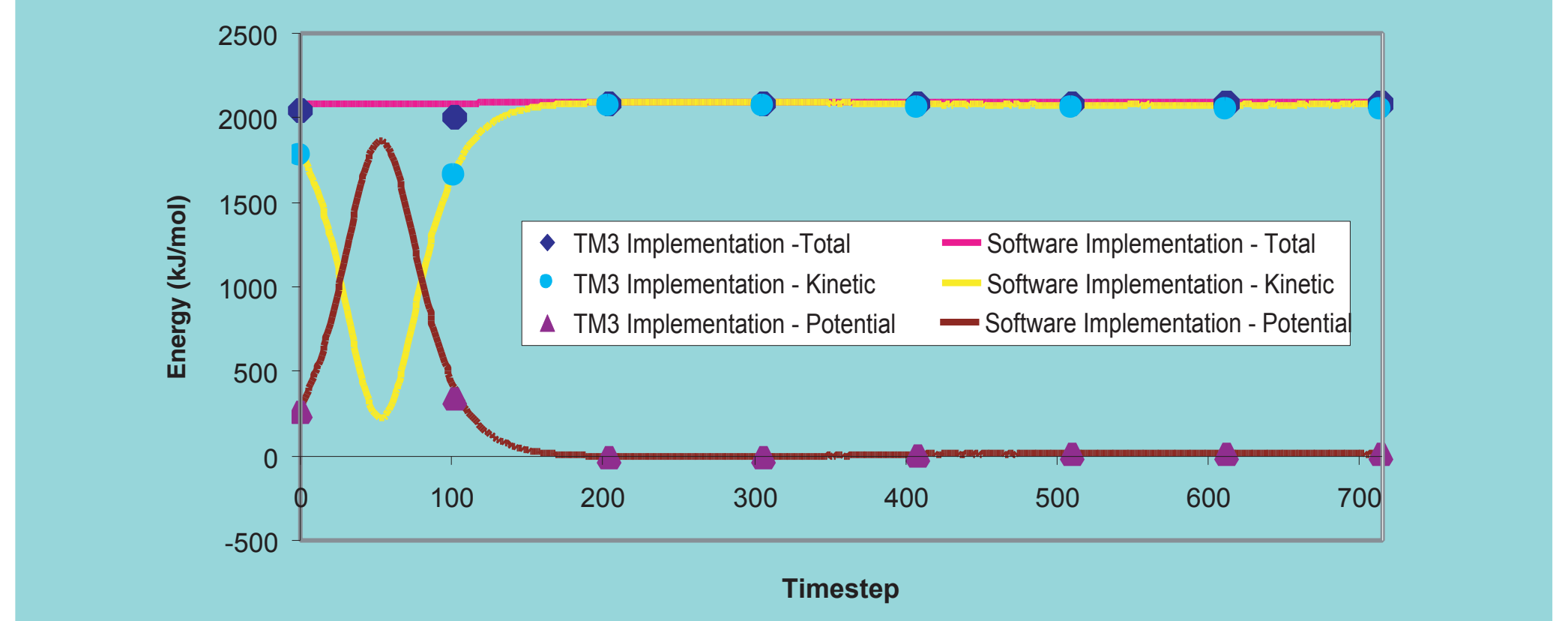
Scaling Factors and Precision of Various Fields

Field	Used in Block	Scaling Factor	Precision
r	PG	2^{-64}	38
r^2	PG, LJFC	2^{-128}	76
Slope	LJGC	2^{77}	36
Function	LJFC	2^{16}	47
Residual x slope	LJFC	2^{15}	48
Pseudo-acceleration	LJFC, AU	2^{-14}	50
Velocity	VU	2^{-15}	51
Velocity after step 2 of Verlet	VU	2^{-15}	51
Acceleration	VU	2^{-64}	37
Velocity x dt	VU	2^{-64}	37
Representation of t	VU	2^{-70}	22

- Top Level VHDL Generator
  - Uses the generated constants to create a wrapper

Validation

- Compared against academic C-based simulator:
  - MD3DLJ
  - Potential and Kinetic Energy track
  - Difference in energy for software and hardware was on the order of 1% RMS.

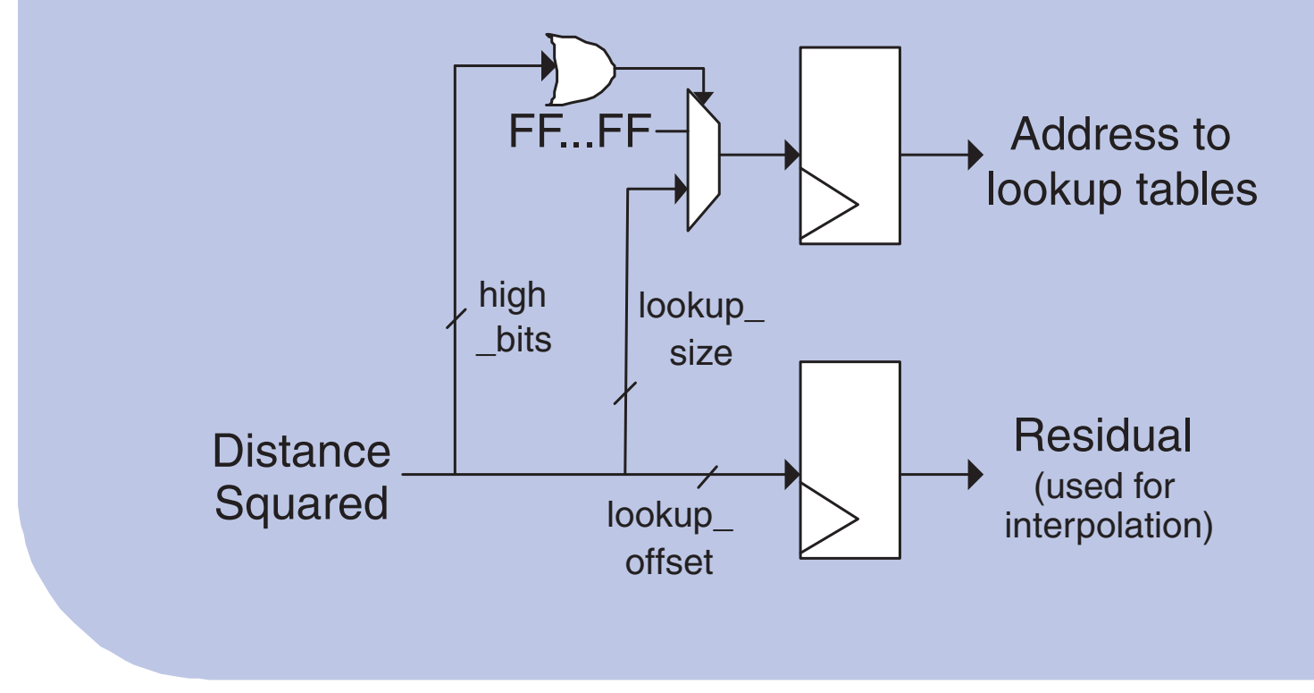


Comparison of Energy of TM3 vs Software Model

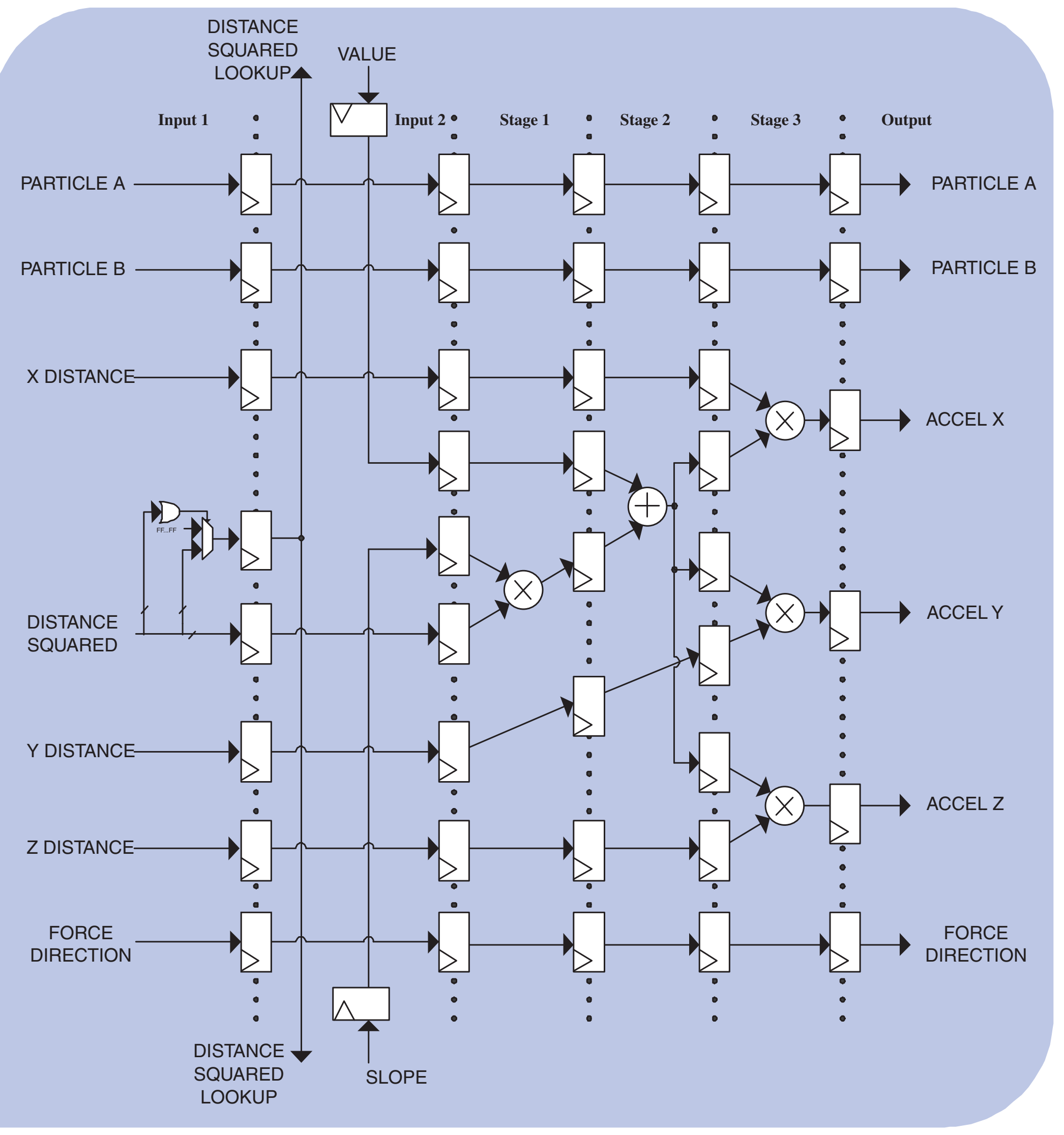
Time Step	Kinetic Energy (KJ/mol)	Potential Energy (KJ/mol)	Total Energy (KJ/mol)
Initial	1794	293.4	2087
Software – End	2078	14.4	2093
Hardware -End	2068	13.52	2081

Lennard-Jones Force Generator

- Receives distance from PG
- Uses middle-order bits of r^2 to lookup value and slope of LJ potential function
- Since LJ potential does not change much at large distances, the higher-order bits are only used to limit the maximum distance
  - Allows for finer granularity of lookup where function is rapidly changing

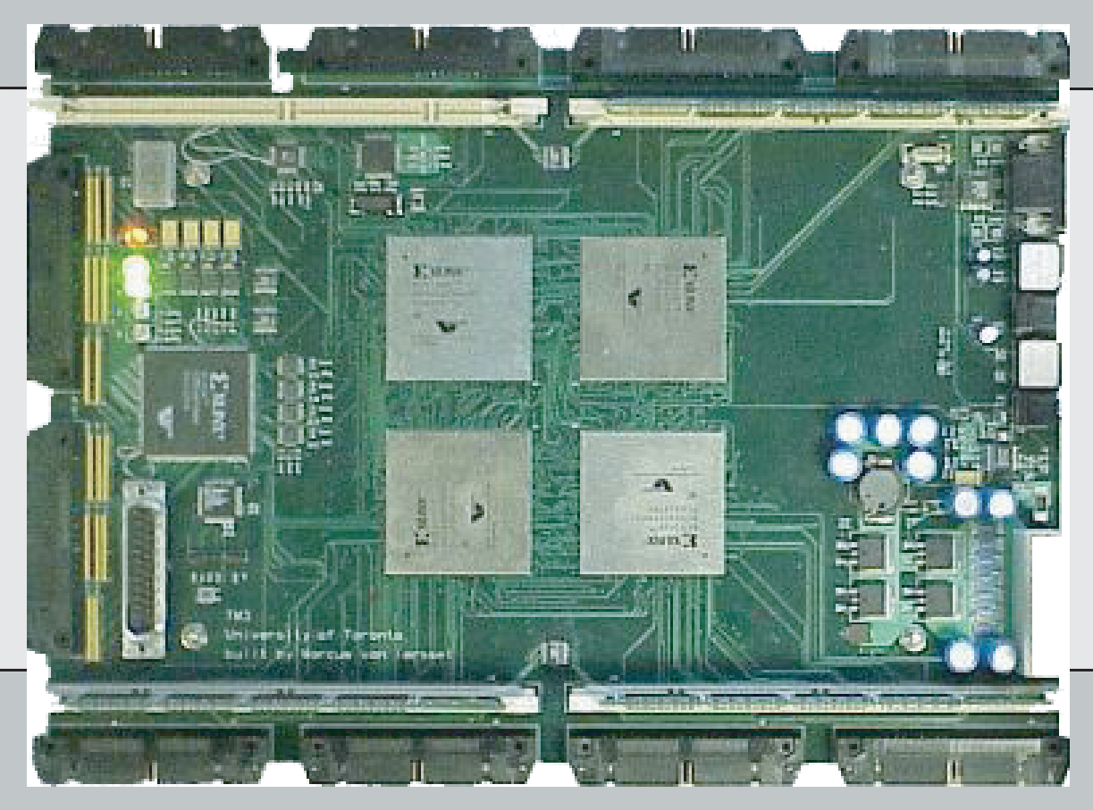


- Uses lower bits of r^2 to multiply with slope to interpolate Adds function value to interpolated amount to obtain pseudo-acceleration
- Not true acceleration because it includes division by r and multiplication by dt, which were included in the look up table to simplify hardware
- Final multiplication by distance components to obtain acceleration components

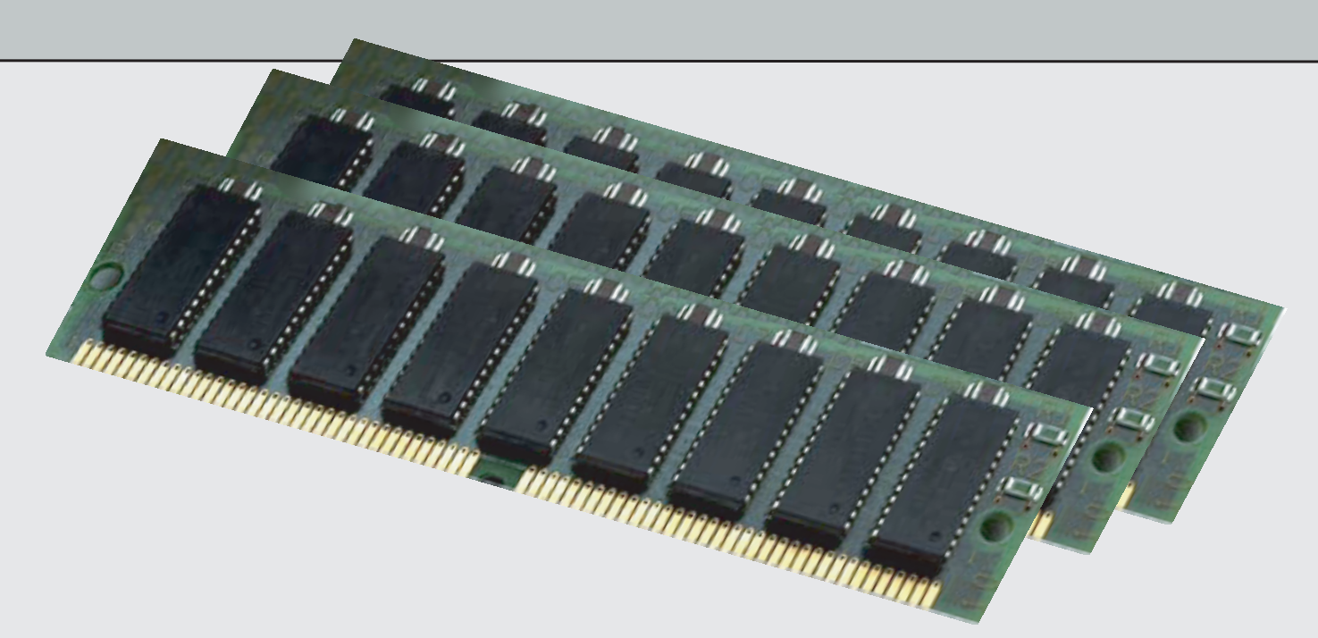


Results

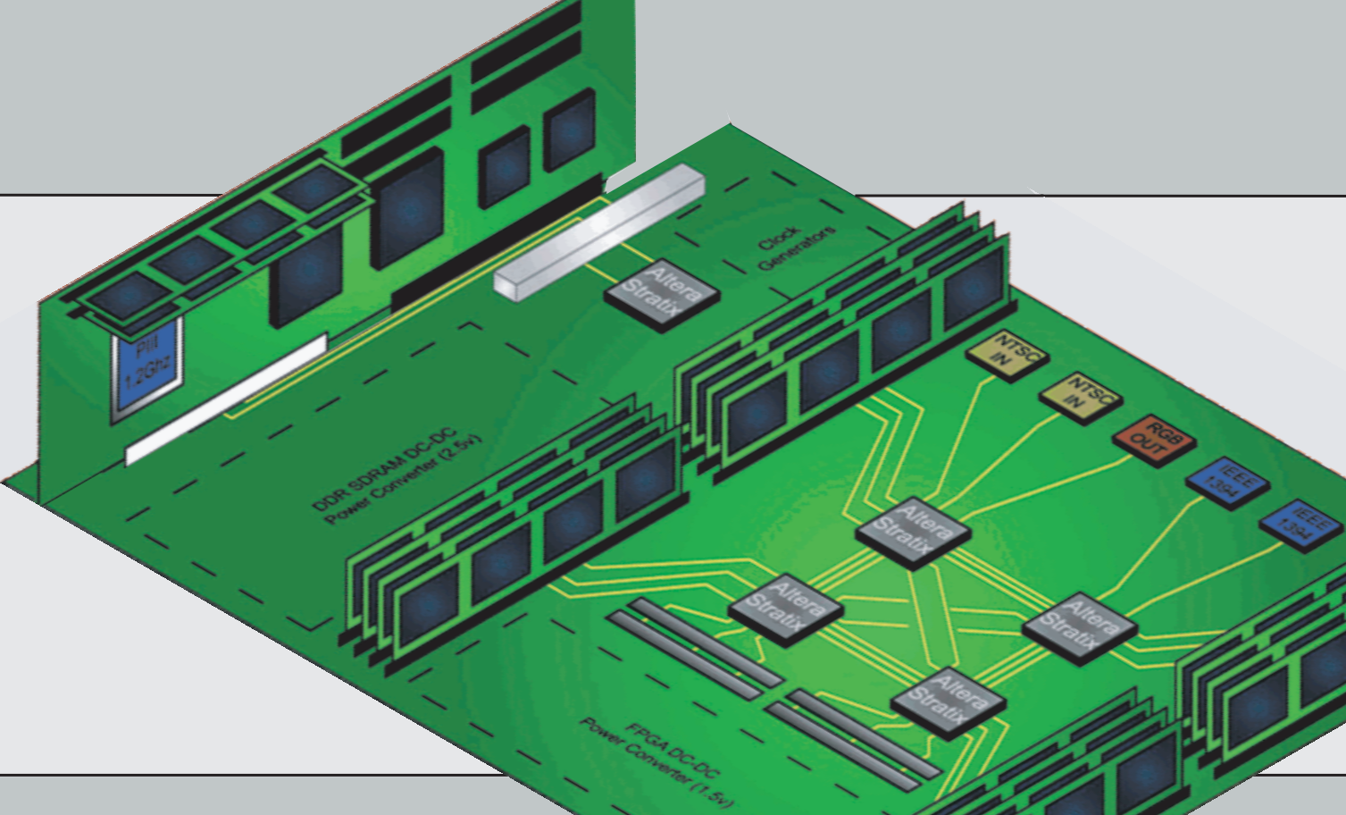
TM3



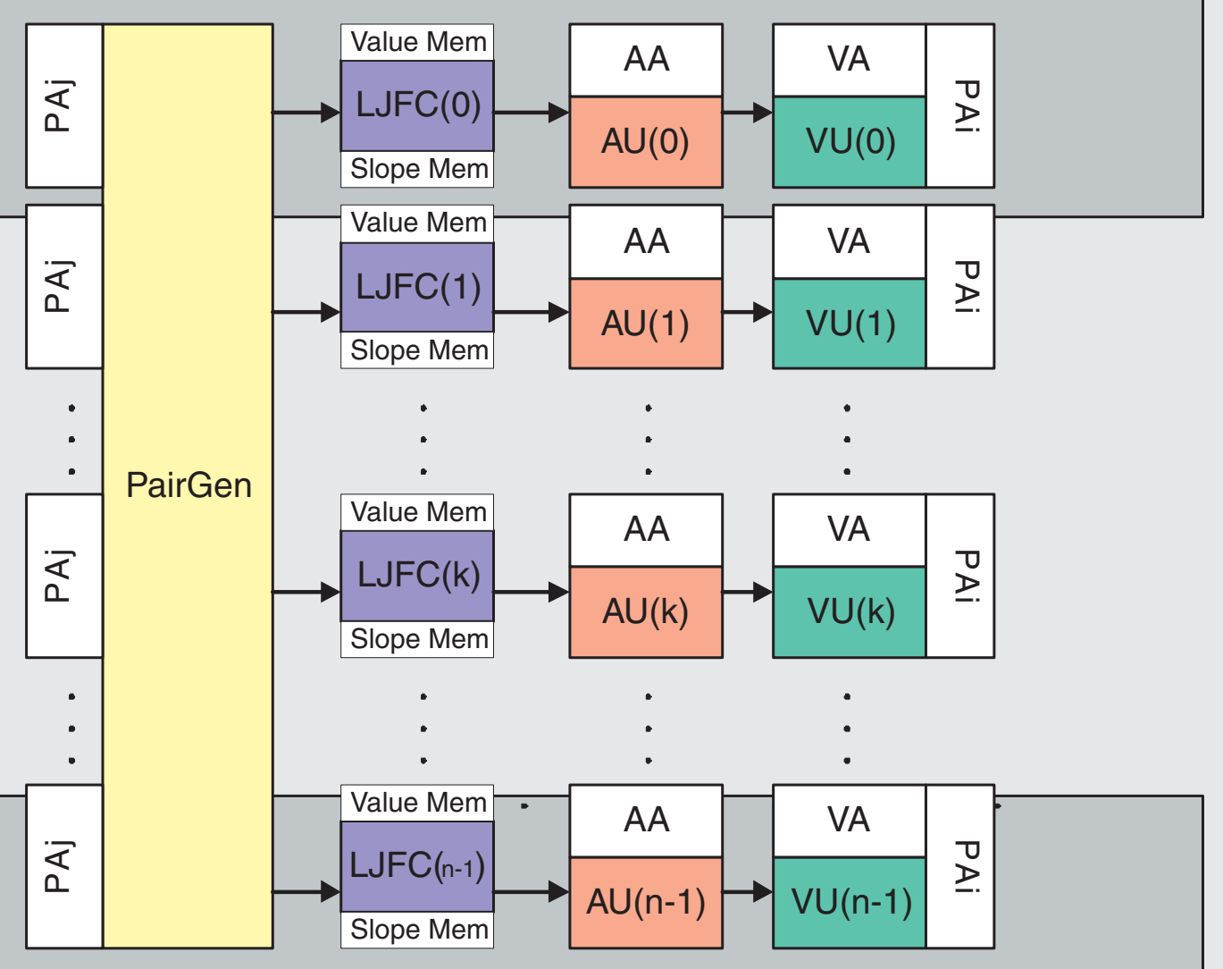
Better Memory Organization



Modern FPGA



Multiple FPGAs



	TM3	Better Memory Organization	Modern FPGA	Multiple FPGAs
Timestep (s)	37 sec	2.1 sec	0.51 sec	0.51/n sec
Relative Speedup to Software (10.7s)	0.29X	5.1X	21X	21*nX

Comparison of MD Simulator and State-of-the-Art Processor

Per Board	2.4GHz P4 + 1GB DRAM	4 FPGAs + 24 MB SRAM
Performance	1X	Min 40X
Cost	~$1500	~$1500
Power	106W	40W

Metric	Improvement of MD System
Performance/Power	106X Improvement
Performance/Cost	40X Improvement
Performance/Space	40X Improvement

Supercomputers = many P4-like processors
Why not FPGA processors instead?

The TM3 project has been made possible by support from Micronet and Xilinx.