Towards Production FPGA-Accelerated Molecular Dynamics: Progress and Challenges**

Matt Chiu          Martin Herbordt

Computer Architecture and Automated Design Laboratory
Department of Electrical and Computer Engineering
Boston University
http://www.bu.edu/caadlab

* This work supported, in part, by the U.S. NIH/NCRR, by IBM, and by generous donations from Altera and GiDEL

* Thanks to Yongfeng Gu
Outline

1. Background → FPGA-Based Accelerators
2. Background → Molecular Dynamics
3. Our MD work → The Past
4. Our MD work → The Present
5. Our MD work → The Future
What is an FPGA-Based System?

**Accelerator** ↔ device optimized to enhance the performance or functionality of a computing system.

**FPGA** ↔ IC whose logic is configurable w.r.t. a user specification.

**FPGA-based System** ↔ FPGAs used as computational accelerators.

---

**Accelerators in the Platform**

- Accelerator: device optimized to enhance the performance or functionality of a computing system.
- Vertical solution: a performs a complete customer function without a additional server or client system. Example: game console.
- Appliance: a vertical solution with substantially reduced management & programming requirements.

*Accelerators = performance, programmability & precision at lower power and cost*

*From: S. Bhatt, FPL 2007*
Novo-G: RC Machine @ CHREC/ U. Florida

- 192 Stratix-III E260 FPGAs
  - 1 Peta OP (32-bit int)
  - 40 TFLOP (32-bit FP)
  - 8KW Max system power

- 48 quad-FPGA boards
  - GiDEL PCIe x8 PROCStar-III
    - Embedded-style boards, for both HPEC- & HPC-oriented research
  - 4¼ GB DDR2 attached to each FPGA
    - ~ 1 TB total RAM in Novo-G

- 24+1 Linux servers in cluster
  - 24 compute servers (2 boards/server)
  - 1 head-node server for management
  - 20 Gb/s non-blocking InfiniBand
  - 1 Gb/s Ethernet
  - 26 (24+2) quad-core Xeons

“Novo” is Latin, “to make anew, refresh, revive, change, alter,” essence of RC; “G” is for Genesis or Green. ☺
What are FPGAs?

A bag of configurable logic gates... + ... a few thousand ASICs cores
→ Block RAMs (1000x)
→ DSP Units (1000x) ...

... a bag of computer parts ...

Image courtesy of Dr. Andre DeHon
What are FPGAs?, cont.

A bag of computer parts ...

... configurable connections ...

Image courtesy of Dr. Andre DeHon

... a configuration ...

... an application-specific processor

Molecular Dynamics w/ FPGAs

HPRCTA – 11/2010

*FPL2009, TRET52010

Short-range force processor for Molecular Dynamics simulations.

- 80 filter banks, 20 stages each
- 8 force pipelines, 80 stages each
### Why FPGAs? - Basic Capabilities

<table>
<thead>
<tr>
<th>Performance Metric</th>
<th>Intel Xeon 7350 (Quad Core)</th>
<th>Xilinx Virtex-5 SX240T</th>
<th>Delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>Theoretical Issue Rate</td>
<td>47 Billion 64 bit Ops/Sec</td>
<td>2.59 Trillion 64 bit Ops/Sec</td>
<td>55.1X</td>
</tr>
<tr>
<td>Integer Operators</td>
<td>Classical 8/16/32/64 bit</td>
<td>Programmable to any bit size</td>
<td></td>
</tr>
<tr>
<td>FLOPs (Mul+Add)</td>
<td>94 Gflop/s SP</td>
<td>204 Gflop/s SP</td>
<td>2.2X</td>
</tr>
<tr>
<td>Pipeline Depth</td>
<td>14 Stages</td>
<td>Programmable to any depth</td>
<td></td>
</tr>
<tr>
<td>BW to Memory</td>
<td>CPU to MCH (FSB)</td>
<td>FPGA to MCH (FSB)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8.5 GB/s @ 1066MHz</td>
<td>8.5 GB/s @ 1066MHz</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FPGA to Local Memory BW (opt.)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>40 GB/Sec</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L1 Cache BW</td>
<td>188 GB/Sec</td>
<td>3.7 TB/Sec</td>
<td>20x</td>
</tr>
<tr>
<td>Block RAM BW</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Register File BW</td>
<td>750 GB/Sec</td>
<td>7.5 TB/Sec</td>
<td>10x</td>
</tr>
<tr>
<td>LUTRAM BW</td>
<td>30W</td>
<td>130W</td>
<td>0.2X</td>
</tr>
</tbody>
</table>

Table courtesy of Dr. Ivo Bolsens, Xilinx (with updates by MCH in RED)

---

Can we cast the problem in integer, especially low precision?
Why FPGAs? \(\rightarrow\) Low Power
Why FPGAs? → High Utilization

Configurable Microarchitectures

Configurable networks...

BRAM BRAM BRAM BRAM BRAM BRAM BRAM BRAM BRAM BRAM BRAM...
Outline

1. Background ➔ FPGA-Based Accelerators

2. Background ➔ Molecular Dynamics

3. Our MD work ➔ The Past

4. Our MD work ➔ The Present

5. Our MD work ➔ The Future
Why Molecular Dynamics Simulation is so important ...

- Core of Computational Chemistry
- Comprises a large fraction of all supercomputing cycles
- Central to Computational Biology, with applications to
  - Drug design
  - Understanding disease processes

From DeMarco & Dagett: PNAS 2/24/04

Shows conversion of PrP protein from healthy to harmful isoform. Aggregation of misfolded *intermediates* appears to be the pathogenic species in amyloid (e.g. “mad cow” & Alzheimer’s) diseases.

Note: this could *only* have been discovered with simulation!
Why Molecular Dynamics Simulation is so important ...

- Core of Computational Chemistry
- Comprises a large fraction of all supercomputing cycles
- Central to Computational Biology, with applications to
  - Drug design
  - Understanding disease processes

From DeMarco & Daggett: PNAS 2/24/04
Show conversion of PrP protein from healthy to harmful isoform. Aggregation of misfolded intermediates appears to be the pathogenic species in amyloid (e.g., "mad cow" & Alzheimer’s) diseases.

Note: this could only have been discovered with simulation!

Molecular Dynamics w/ FPGAs

2/24/04

I am MAD

HPRCTA – 11/2010
Why *Acceleration* of MD Simulations is so important ...

**MD simulations are**

*often “heroic” →*

100 days on 500 nodes ...

---

P.L. Freddolino, et al.  
*Structure, 2006*
What is Molecular Dynamics?

- **MD** ➔ an iterative application of Newtonian mechanics to ensembles of atoms and molecules

- Two-Phase iterative process, common time step ~1fs

---

**Force Calculation**

- Evaluate forces on atoms
- $O(N)$ part — **done on host**
- $O(N^2)$ part — **done on accelerator**

**Motion Update**

- Apply Newton’s Laws
- $O(N)$ — **done on host**

---

$$ F = m \frac{d^2r}{dt^2} $$
Force Calculation

Many forces typically computed,

but complexity lies in the non-bonded, spatially extended forces: van der Waals (LJ) and Coulombic (C)

\[ F_{\text{total}}^i = F_{\text{bond}} + F_{\text{angle}} + F_{\text{torsion}} + F^H + F_{\text{non-bonded}} \]

\[ F_{\text{C}}^i = q_i \sum_{j \neq i} \left( \frac{q_j}{|\vec{r}_{ji}|^3} \right) \vec{r}_{ji} \]

\[ F_{\text{LJ}}^i = \sum_{j \neq i} \frac{\epsilon_{ab}}{\sigma_{ab}^2} \left[ 12 \left( \frac{\sigma_{ab}}{|\vec{r}_{ji}|} \right)^{14} - 6 \left( \frac{\sigma_{ab}}{|\vec{r}_{ji}|} \right)^8 \right] \vec{r}_{ji} \]
Reduce the $O(N^2)$ Complexity to $O(N)$ ...

LJ force gets small quickly ...

\[ F^{LJ}_i = \sum_{j \neq i} \frac{e_{ab}}{\sigma_{ab}^2} \left[ 12 \left( \frac{\sigma_{ab}}{r_{ji}} \right)^{14} - 6 \left( \frac{\sigma_{ab}}{r_{ji}} \right)^8 \right] \vec{r}_{ji} \]

... while the Coulombic force does not ...

\[ F^C_i = q_i \sum_{j \neq i} \left( \frac{q_j}{r_{ji}^3} \right) \vec{r}_{ji} \]
Make Coulombic Short-Range with Cut-Off

- The Coulombic force can be approximated with a “cut-off” function with the remainder computed elsewhere.

\[ V_{i}^{CL} = \sum_{j \neq i} \frac{q_j}{|r_{ji}|} \quad \rightarrow \quad \frac{1}{r} = \left( \frac{1}{r} - g_a(r) \right) + g_a(r) \]

\[ \frac{F_{ji}^{CSh}}{r_{ji}} = QQ \times (r^{-3} + G_0 + G_1 \times r^2 + G_2 \times r^4) \]
Make S.R. Kernel $O(N)$ with Cell Lists*

Observation:
- Typical volume to be simulated $\geq 100\text{Å}^3$
- Typical cut-off radius = 10Å - 12Å

Therefore, with all-to-all $O(N^2)$ computation, *most work is wasted*

Solution:
- Partition space into “cells,” each roughly the size of the cut-off
- Compute forces on $P$ only w.r.t. partner particles in adjacent cells (the cell neighborhood)
- Creating cell lists is $O(N)$ so is done on the host
- $N$ particles x fixed neighborhood $\Rightarrow O(N)$
  - but with a large constant ...

*Implemented previously (FPL06, ParCo08)
Outline

1. Background ➔ FPGA-Based Accelerators

2. Background ➔ Molecular Dynamics

3. Our MD work ➔ The Past

4. Our MD work ➔ The Present

5. Our MD work ➔ The Future
Goal: Accelerate Short-Range Force*

Non-bonded short-range force consumes >95% of total simulation time.

\[ F_{ji}^{\text{short}} = A_{ab} \times r_{ji}^{-14} - B_{ab} \times r_{ji}^{-8} + QQ \times (r_{ji}^{-3} + G_0 + G_1 \times r_{ji}^2 + G_2 \times r_{ji}^4) \]

Partition:
- **FPGA**: short-range force
- **Host**: all the rest
  - motion update
  - bonded force
  - long-range non-bonded force

*FPL09, TRETS10*
1\textsuperscript{st} Force Pipeline: Table Lookup w/ Interpolation*

- Interpolation 1\textsuperscript{st}~4\textsuperscript{th} order
- Input is \( r^2 \)
  - Avoid expensive square root
- Three terms are interpolated individually:
  - \( r^{-3}, r^{-8}, r^{-14} \)

\[
f(x) = C_0 + C_1(x - a) + C_2(x - a)^2 + C_3(x - a)^3 + \ldots + C_M(x - a)^M + \alpha(x^M)
\]

- Accuracy
  - Interpolation order
  - Interval resolution

\[ x = r^2 \]

\[ \text{Format} \]
- Find most significant 1 to:
  - get format
  - extract a
  - extract \((x-a)\)

\[ (x-a) \]

\[ C_2(x-a) \]

\[ (C_3(x-a)+C_2)(x-a) \]

\[ (C_4(x-a)+C_3)(x-a)+C_2 \]

\[ ((C_5(x-a)+C_4)(x-a)+C_3)(x-a)+C_2 \]

\[ r^{-14}, r^{-8}, \text{or } r^{-3} \]

*Optimized previously (FPL06)

Molecular Dynamics w/ FPGAs

HPRCTA – 11/2010
2nd Force Pipeline: Direct Computation*

Single precision FP + 32-bit fixed point
Quality validated (see HPRCTA08)

Resource Utilization Stratix-III 340

<table>
<thead>
<tr>
<th>Mode</th>
<th>Precision</th>
<th>Logic Utilization (ALUT / Register)</th>
<th>Multipliers</th>
<th>Frequency (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hybrid FPC*</td>
<td>32-bit integer/single</td>
<td>5% (4% / 5%)</td>
<td>9%</td>
<td>251</td>
</tr>
</tbody>
</table>

Number of force pipelines = 11
Throughput = 2.8G forces per second

200x faster than a single core

*Optimized previously (HPRCTA08)
Table Lookup vs. Direct Computation*

• The cost of direct computation is about the same as that of third order interpolation.

*HPRCTA08
Molecular Dynamics w/ FPGAs
HPRCTA – 11/2010
The Problem, even with Cell Lists

• Problem → Efficiency
  – ~15.5% efficiency when cell size is equal to cutoff
  – *Even with cell lists*,
    84.5% of computation is wasted!

• Solution → Filtering
  – Only compute forces for particle pairs where *force is non-zero*
  – Create neighbor lists on-the-fly on the FPGA
Filter Computation

Filter Computation \((i,j)\) = Distance Computation \((i,j)\) & Threshold

If \(r_{ij}^2 < (r_{threshold})^2\) then continue
Obvious Change to Force Pipeline

Filter = Distance computation & threshold

Idea: ~6x as many “filters” as force pipelines

• Ideally, throughput should improve by 6x
• Cost is only the replicated distance computation

6x distance computation per 1x rest of force pipeline
## 1st Candidate Design

<table>
<thead>
<tr>
<th></th>
<th>Original Design</th>
<th>Original Design</th>
<th>1st Candidate Filter Design</th>
</tr>
</thead>
<tbody>
<tr>
<td># of Force Pipelines</td>
<td>10</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Efficiency</td>
<td>15.5%</td>
<td>~100%</td>
<td></td>
</tr>
<tr>
<td># of Filters Per Force Pipeline</td>
<td>1</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>Normalized Throughput</td>
<td>1x</td>
<td>2.6x</td>
<td></td>
</tr>
</tbody>
</table>
Another Idea – *Reduce Filter Complexity*

- **Observation:** \( r \) computed for filter need not be as accurate as \( r \) computed for force computation
  - *Can round up \( r_{\text{threshold}} \) to compensate for inaccuracy*
  - *Efficiency is then \(<100\%\)*

- **Critical because** \( 9\times \) as many filters as force pipelines (Why \( 9\% \)? *See below*)

**Methods to simplify filters** ➔

1. **Reduce precision from 32 bits to 8 bits**
   - *still relies on multipliers, the critical resource*

2. **Approximate spherical geometry with Cartesian (planar)**
Planar filter – spherical approximation

- x, y, and z are the distances between a reference particle and its neighboring particles

- The following relations should be satisfied (constants)
  - |x| < r_c, |y| < r_c, |z| < r_c
  - |x| + |y| < sqrt(2) * r_c
  - |x| + |z| < sqrt(2) * r_c
  - |y| + |z| < sqrt(2) * r_c
  - |x| + |y| + |z| < sqrt(3) * r_c (3D)
  - x >= 0 (with Newton’s 3rd law)

- No multiplies!
## Filter Complexity

<table>
<thead>
<tr>
<th>Method</th>
<th>ALUTs / Registers</th>
<th>Multipliers</th>
<th>Filter Efficiency</th>
<th>Extra Work</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full precision</td>
<td>341 / 881 (0.43%)</td>
<td>12 (1.6%)</td>
<td>100%</td>
<td>0%</td>
</tr>
<tr>
<td>Full precision - Logic-only mults</td>
<td>2577 / 2696 (1.3%)</td>
<td>0</td>
<td>100%</td>
<td>0%</td>
</tr>
<tr>
<td>Reduced precision</td>
<td>131 / 266 (0.13%)</td>
<td>3 (0.4%)</td>
<td>99.5%</td>
<td>3%</td>
</tr>
<tr>
<td>Reduced precision - Logic-only mults</td>
<td>303 / 436 (0.21%)</td>
<td>0</td>
<td>99.5%</td>
<td>3%</td>
</tr>
<tr>
<td>Planar</td>
<td>164 / 279 (0.14%)</td>
<td>0</td>
<td>97.5%</td>
<td>13%</td>
</tr>
<tr>
<td>Force Pipeline</td>
<td>5695 / 7678 (5%)</td>
<td>70 (9.1%)</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

### 2nd Candidate Design

- 8 force pipelines (rather than 10)
- each force pipeline has 9 filters
- 99.5% efficiency (rather than 15.5%)
- 73% of resources
- 15% of resources

---

Molecular Dynamics w/ FPGAs

HPRCTA – 11/2010
# 2nd Candidate Design

<table>
<thead>
<tr>
<th>Original Design</th>
<th>Original Design</th>
<th>1st Candidate Filter Design</th>
<th>2nd Candidate Filter Design</th>
</tr>
</thead>
<tbody>
<tr>
<td># of Force Pipelines</td>
<td>10</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>Utilization</td>
<td>15.5%</td>
<td>100%</td>
<td>99.5%</td>
</tr>
<tr>
<td># of Filters Per Force Pipeline</td>
<td>1</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>Normalized Throughput</td>
<td>1x</td>
<td>2.6x</td>
<td>5.2x</td>
</tr>
</tbody>
</table>
The REAL Problem(s)

Data Routing and Queueing
- Proc. Memory ➔ POS Memory (Board)
  - for each iteration
- Pos. Memory ➔ Position Cache
  - for each home cell
- ACC. Memory ➔ ACC. Cache
  - for each home cell
- Pos. Cache ➔ Filter Banks
  - for each reference particle and its partners
- Filter Banks ➔ Force Pipelines
  - for each particle pair with force > threshold
- Force Pipelines ➔ Acceleration Cache
  - for each home cell
- ACC. Cache ➔ ACC. Memory
  - for each iteration

Load Balancing
- Amount & variance of work for each 
  reference particle
- Amount & variance of work for each 
  force pipeline

Molecular Dynamics w/ FPGAs  
HPRCTA – 11/2010
Implementation Details

- 8 force pipelines fit into Altera Stratix III EP3SE260
  - Single precision, hybrid 32/36-bit direct force computation
  - Simulation quality checks (HPRCTA08)
  - Running @ ~200 MHz for PaR and on GiDEL PROCe III
  - Each pipeline contains 9 filters and one force pipelines
  - N3L partitioning via 18-cell “half-moon” scheme
  - Continuous queueing method with concentrator and throttling logic
- Validated on GiDEL Board

<table>
<thead>
<tr>
<th>Implementation</th>
<th>Logic (ALUTs /Registers)</th>
<th>Multipliers</th>
<th>Memory (M9K/M144K)</th>
<th>Max Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>SE260 - 8 force pipelines, reduced precision</td>
<td>78% (43% / 71%)</td>
<td>99%</td>
<td>87% (75% / 100%)</td>
<td>196 MHz</td>
</tr>
<tr>
<td>SE260 - 9 force pipelines, reduced precision</td>
<td>93% (53% / 85%)</td>
<td>98%</td>
<td>97% (95% / 100%)</td>
<td>190 MHz</td>
</tr>
<tr>
<td>SL340 - 6 force pipelines, planar filtering</td>
<td>53% (18% / 41%)</td>
<td>85%</td>
<td>97% (94% / 100%)</td>
<td>216 MHz</td>
</tr>
<tr>
<td>SL340 - 7 force pipelines, planar filtering</td>
<td>69% (24% / 52%)</td>
<td>99%</td>
<td>100% (100% / 100%)</td>
<td>198 MHz</td>
</tr>
<tr>
<td>SL340 - 8 force pipelines, Planar filtering</td>
<td>100% (40% / 75%)</td>
<td>100%</td>
<td>100% (100% / 100%)</td>
<td>122 MHz</td>
</tr>
</tbody>
</table>
Status 9/2009

• NAMD ApoA1 Benchmark
  – 92,224 particles, ~33.4M force computations per iteration
  – 108Å x 108Å x 78Å simulation box with 12Å cutoff radius

• Time = 22ms per iteration for short-range force
  – > 80x single core execution time
  – Independent of simulation size (cell set at a time)
  – Other latency can be hidden (details in longer version)
  – Assumes 195 MHz (from Post-P&R timing)
  – Assumes 90% utilization (from functional model)

• Initial quality validation performed using
  – ProtoMol
  – Multigrid
  shows little difference from unaccelerated code using double precision
Outline

1. Background ➔ FPGA-Based Accelerators
2. Background ➔ Molecular Dynamics
3. Our MD work ➔ The Past
4. Our MD work ➔ The Present
5. Our MD work ➔ The Future
Recent Work – Towards a Production System

Goals:
• Integrate accelerator into NAMD-lite
• Get accelerated NAMD-lite working on a PC with GiDEL PROCStar III (4 x Stratix III SE260), initially on a single FPGA

Issues:
• PME must now be supported (previously used Multigrid)
• Data transfer interface: conversions and organization
• Integrate particle exclusion
• Maintain accuracy with original code (and with NAMD)
  – Necessary to augment LJ force with a switching function
Non-bonded Force Exclusion

- **Optimization:**
  - Close-range cut-off prevents huge fake forces
  - Calculate the cut-off with \( F^{\text{short}}(r^2) < \text{dynamic\_range} \)
  - Multiple cut-offs according to particle types are required
  - Compensation in host -- recompute as necessary
LJ Addition to Force Pipelines

Van der Waals force/energy is computed directly in single precision floating point format and implemented with the aid of Altera floating point compiler (FPC).

\[
\begin{align*}
    s &= (cutoff^6 - r^6) + (cutoff^6 - r^6) + (cutoff^6 + 2 \times r^6 - 3 \times switch\_dist^6) \times \text{denom} \\
    d_{sp} &= 12 \times (cutoff^6 - r^6) + (switch\_dist^6 - r^6) \times \text{denom} \\
    \text{denom} &= 1/(cutoff^6 - switch\_dist^6)^3
\end{align*}
\]

If \( r^2 \leq \text{switch\_dist}^2 \)
\[
U_{vdw} = U, \quad F_{vdw} = F_{vdw}
\]

If \( r^2 > \text{switch\_dist}^2 \) \& \( r^2 < \text{cutoff}^2 \)
\[
U_{vdw} = U_{vdw} \times s, \quad F_{vdw} = F_{vdw} \times s + U_{vdw} \times ds_r
\]

If \( r^2 \geq \text{cutoff}^2 \)
\[
U_{vdw} = 0, \quad F_{vdw} = 0
\]

\( U_{vdw} \): Van der Waals potential  \\
\( F_{vdw} \): Van der Waals force  \\
\( \text{Cutoff} \): Van der Waals cutoff distance  \\
\( \text{Switch\_dist} \): switching distance  \\
\( R^2 \): distance square between two atoms
PME Addition to Force Pipelines

- The complementary error function $E_g = \frac{1}{4\pi\varepsilon_0} \frac{1}{2} \sum_{n} \sum_{i=1}^{N} \sum_{i=0}^{n} \frac{q_i q_j}{|n - \eta + nL|^2} \text{erfc}\left(\frac{|n - \eta + nL|}{\sqrt{2}\sigma}\right)$

- Replaces $\frac{F_{ji}^s}{\mathbf{r}_{ji}} = QQ \times (r^{-3} + G_0 + G_1 \times r^2 + G_2 \times r^4)$

- Since the complementary error function (erfc) is expensive in hardware with direct computation, polynomial interpolation is used to approximate short-range part of PME electrostatic force.

- Polynomial coefficients are computed using Matlab, which finds the coefficients of a polynomial $p(x)$ of degree $n$ that fits the data, $p(x(i))$ to $y(i)$, w.r.t. least squares.

- Binning is weighted by section

- Energy conservation is used to measure the quality of the approximation for various polynomial interpolation orders and interval sizes.

\[ f(x) = C_0 + C_1(x-a) + C_2(x-a)^2 + C_3(x-a)^3 + \ldots + C_M(x-a)^M + o(x^M) \]
Now the force pipelines ...

... use direct computation for the LJ force and table lookup with interpolation for the Coulomb force ☹
Energy Variance - Initial Results (9/10)

NAMD benchmark NAMD2.6 on ApoA1

- 92,224 particles, bounding box = 108Å x 108Å x 78Å, Cut-off radius = 12Å
- Energy measured every 100 fs (80ps after initial transient removed)

Energy Plot

<table>
<thead>
<tr>
<th></th>
<th>NAMD-Lite</th>
<th>Linear</th>
<th>2nd order</th>
<th>3rd order</th>
<th>NAMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drift slope</td>
<td>-0.1167</td>
<td>0.338</td>
<td>0.007</td>
<td>-0.0235</td>
<td>-0.1898</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>5.219</td>
<td>9.131</td>
<td>5.136</td>
<td>4.533</td>
<td>5.755</td>
</tr>
</tbody>
</table>

Longer simulations and simulations using the Shadow Hamiltonian now in progress
Performance - Initial Results (9/10)

NAMD benchmark NAMD2.6 on ApoA1
- 92,224 particles, bounding box = 108Å x 108Å x 78Å, Cut-off radius = 12Å

4 force pipelines, 32 filters (8 per pipeline)
  - reduced from 8 & 72

Operating frequency = 205MHz
  - reduced from 195MHz

Some unanticipated inefficiencies -- efficiency = 60%
  - reduced from 90%

Current performance of short-range force = 70ms
  - from 22ms
Outline

1. Background ➔ FPGA-Based Accelerators
2. Background ➔ Molecular Dynamics
3. Our MD work ➔ The Past
4. Our MD work ➔ The Present
5. Our MD work ➔ The Future
Simple Optimizations in Progress

• Convert LJ from direct computation to table look-up w/ polynomial interpolation
  – will reuse much of the same logic as the Coulomb calculation
  – design space exploration: simulation quality vs. HW cost

• More aggressive synthesis

• Larger queues (between filter & force pipelines)
  – to improve efficiency

Anticipated benefit: at least 30%
  – gets us back under 50ms
Another Simple Optimization

Technology upgrade: Stratix-III is now two generations behind:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Equivalent LEs (K)</td>
<td>203</td>
<td>650</td>
<td>1,087</td>
<td>530</td>
</tr>
<tr>
<td>18 x 18 multipliers</td>
<td>768</td>
<td>1024</td>
<td>1100</td>
<td>3510</td>
</tr>
<tr>
<td>On Chip Memory</td>
<td>150</td>
<td>230</td>
<td>430</td>
<td>360</td>
</tr>
</tbody>
</table>

Stratix V Selection: Different mixes of resources lead to different optimizations
- *e.g., table size versus arithmetic operations*

Numbers of pipelines with same design (initial -- Synthesized P-PaR):
- Stratix-IV → 6 pipelines
- Stratix-V → 12 pipelines

Operating frequency should also be substantially higher
Lots of ways to improve efficiency

A concern is the granularity of the processing phases. The number of phases necessary to process the particles in a single home cell is

\[
\text{ceiling}(\frac{\text{particles-in-home-cell}}{\text{filters}})
\]

For small cells and lots of filters, the loss of efficiency can become significant. There are several possible solutions.

1. Increase the number of filters and further decouple neighbor list generation from consumption. The reasoning is that as long as the force pipelines are busy, some inefficiency in filtering is tolerable.

2. Overlap processing of two home cells. This increases the working set from 18 to 27 cells for a modest increase in number of BRAMs required. One way to implement this is to add a second distribution bus.

3. Another way to overlap processing of two home cells is to split the filters among them. This halves the phase granularity and so the expected inefficiency without significantly changing the amount of logic required for the distribution bus.
In Progress -- Large Scale RC System

Novo-G integration
- Port to 4 FPGAs (single node) has been tested
- Larger scale being implemented (with Alan George’s group at U. Florida)

Key Issues
- Overlapping communication w/ data transfer
- Balance between short-range force computation and rest of the application (long-range force now also needs acceleration?)
Questions?

• Thank you for your attention!

BoF: Large Scale Reconfigurable Computing  Tuesday, 11/16, 5:30-7:00

This work is supported in part by the NIH through award R01-RR023168-01, by IBM through a faculty award, and by generous donations from GiDEL and Altera.